Université Paris Sud

Département de formation doctorale en informatique **École doctorale Université Paris Sud**

Contribution aux infrastructures de calcul global : délégation inter plates-formes, intégration de services standards et application à la physique des hautes énergies.

THÈSE

présentée et soutenue publiquement le 21 septembre 2006

pour l'obtention du

Doctorat de l'université Paris Sud

(spécialité informatique)

par

Oleg Lodygensky
LAL, Laboratoire de l'Accélérateur Linéaire, France
LRI, Laboratoire de Recherche en Informatique, France

Composition du jury

Présidents :	Franck Cappello, Alain Cordier
Rapporteurs :	Christophe Cérin Pierre Manneback
Examinateurs :	Guy Wormser Miron Lyvni
Invité :	Charles Loomis

Laboratoire de Recherche en Informatique — CNRS Université Paris XI –UMR 8623

Mis en page avec la classe thloria.

Remerciements

Je remercie tout particulièrement Franck Cappello pour son soutien indéfectible et auprès de qui j'ai beaucoup appris.

Mais aussi François Couchot, pour son piano, sa raquette de tennis et, surtout, sa capacité incommensurable à donner confiance à tout un chacun.

Je remercie d'autre part Alain Cordier, Christian Helft et Michel Jouvin, sans qui cette thèse n'aura pas pu se faire.

Enfin, je ne saurais oublier Vincent Néri et Gilles Fédak, mes gurus.

En mémoire de Marc Frenkel, qui nous a quitté trop tôt.
En hommage à mon frère, Alexis, et ma soeur, Sabine.
A Karine Ramage Lodygensky, ma muse.

Métaphore.

```
Il y a des Italiens, des Grecs, des Espagnols
Des Russes, des Bulgares, des Persans, des Mongols
Ce sont des bêtes de cirques qui sautent les méridiens
On leur jette des morceaux de viande noire, comme à des chiens
C'est leur bonheur à eux que cette sale pitance
Seigneur, ayez pitié des peuples en souffrance.
```

Blaise Cendrars.

Résumé

La généralisation et les puissances aujourd'hui disponibles des ressources informatiques, ordinateurs, espaces de stockages, réseaux, permettent d'imaginer de nouvelles méthodes de travail ou de loisir, inconcevables, il y a encore peu. Les ordinateurs monolithiques centralisés, ont peu à peu laissé place à des architectures distribuées «client/serveur» qui se trouvent elles mêmes concurencées par de nouvelles organisations de systèmes distribués, les systèmes «pair à pair». Cette migration n'est pas le fait de spécialistes ; les utilisateurs les moins avertis utilisent tous les jours ces nouvelles technologies, que ce soit pour échanger des courriers électroniques, à des fins commerciales à travers le «e-commerce» sur le Web, ou encore pour échanger des fichiers musicaux, hors de toute infrastructure, «d'égal à égal».

Les mondes du commerce, de l'industrie et de la recherche, ont bien compris les avantages et les enjeux de cette révolution et investissent massivement dans la recherche et le développement autour de ces nouvelles technologies, que l'on appelle les «grilles», qui désignent des ressources informatiques globales et qui ouvrent une nouvelle approche. Une des disciplines autour des grilles concerne le calcul. Elle est l'objet des travaux présentés ici.

Sur le campus de l'Université Paris-Sud, à Orsay, une synergie est née entre le Laboratoire de Recherche en Informatique (*LRI*) d'une part, et le Laboratoire de l'Accélérateur Linéaire (*LAL*), d'autre part, afin de mener à bien, ensemble, des travaux sur les infrastructures de grille qui ouvrent de nouvelles voies d'investigation pour le premier et de nouvelles méthodes de travail pour le second.

Les travaux présentés dans ce manuscrit sont le résultat de cette collaboration pluridisciplinaire. Ils se sont basés sur XtremWeb, la plate-forme de recherche et de production de calcul global développée au LRI. Nous commençons par présenter un état de l'art des systèmes distribués à grande échelle, ses principes fondamentaux, son architecture basée sur les services. Puis nous introduisons XtremWeb et détaillons les modifications que nous avons dû apporter, tant au niveau de son architecture que de son implémentation, afin de mieux répondre aux exigences et aux besoins de ce type de plate-forme. Nous présentons ensuite deux études autour de cette plate-forme permettant de généraliser l'utilisation de ressources inter grilles, d'une part, et d'utiliser sur une grille des services qui n'ont pas été prévus à cette fin, d'autre part. Enfin, nous présentons l'utilisation, les problèmes à résoudre et les avantages à tirer de notre plate-forme par la communauté de recherche en physique des hautes énergies, grande consommatrice de ressources informatiques.

Mots-clés: Grille de PC ; Calcul Distribué ; Plate-forme à Grande Echelle ; Tolérance aux Pannes ; Physique des Hautes Energies.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

La banalisation et la puissance des outils informatiques, les réseaux à haut débit ont ouvert des débouchés inimaginables il y a encore peu. Ces nouvelles caractéristiques, cette nouvelle puissance disponible, tant quantitative que qualitative, a bouleversé nos modes de vies personnel et professionnel. Il est aujourd'hui banal d'avoir plusieurs outils électroniques, ordinateurs, assistants personnels, téléphones, appareils photo numériques et consoles de jeu. Il est aujourd'hui naturel d'envoyer un mail, d'acheter des biens et des services sur le Web, d'échanger toutes sortes de données électroniques entre personnes physiques ou morales et ce, justement, grâce à nos merveilleux outils suscités qui ont tous deux caractéristiques communes : ils marchent tous à l'électricité et ils sont tous connectés à un réseau informatique, quel qu'il soit.

Toutefois, dans le domaine de l'informatique, banalisation ne rime pas forcément avec simplification, bien au contraire, car la multiplicité des outils disponibles n'a pu faire apparaître de standard de fait qui se serait imposé à tous, bien que le public en connaît quelques uns, comme le système d'exploitation *Windows* de *Microsoft* qui équipe la majeur partie des ordinateurs individuels, ou le protocole *HTTP* pour l'échange d'informations sur le Web ; mais ce ne sont là que la partie émergente de l'iceberg et ces deux produits ne couvrent pas, loin s'en faut, toutes les possibilités liées à l'informatique.

De cette constatation de la puissance mais de la complexité des outils informatiques, et des problèmes coopératifs qui en découlent, est née l'idée de *grille* informatique [47] qui souhaite rendre l'utilisation des outils informatiques aussi simple et fiable que l'utilisation de l'électricité. En appuyant sur un interrupteur électrique, nous faisons un geste tellement banal que nous ne réalisons même plus qu'il se passe trois choses essentielles :

- A chaque fois que nous demandons la mise en route d'un appareil électrique, le courant électrique nécessaire nous est effectivement fourni ;
- Quelque soit la puissance que nous demandons, de quelques milliwatts pour une lampe de chevet jusqu'à plusieurs watts pour une machine à laver ou un téléviseur cathodique, nous obtenons la puissance demandée (dans les limites de notre abonnement) ;

– A tout instant, nous sommes incapables de déterminer l'origine de la puissance fournie : nous ne pouvons, en tant qu'utilisateur, savoir quelle centrale électrique nous fournit la puissance à un instant donné. Peut être pensez vous connaître la centrale qui vous fourni votre électricité ; mais avez vous déjà réalisé que vous êtes toujours fourni, même lorsque cette centrale est arrêtée pour cause de réparation, par exemple ? Peut être n'y aviez vous jamais pensé, et c'est une bonne chose, car vous n'avez pas à vous en soucier : votre fournisseur électrique vous a automatiquement redirigé vers une autre source d'électricité le temps de la réparation et vous ne vous êtres aperçu de rien.

Le fournisseur électrique est donc pour nous, utilisateurs, une boîte noire à laquelle nous accédons grâce à nos prises électriques. Ce service est fiable : les pannes électriques sont rares. Il est générique : ça marche pour tout appareil électrique. Il est standardisé : il existe une et une seul norme par pays et même au delà. Il est transparent : les utilisateurs n'ont en aucune manière besoin de se soucier de son mode de fonctionnement interne.

Les grilles informatiques souhaitent obtenir les même caractéristiques pour les services informatiques : facilité d'utilisation, fiabilité, standardisation et transparence, afin que les utilisateurs obtiennent les services électroniques demandés, aussi simplement qu'en appuyant sur un bouton. Dès le début des années 1960, à la suite d'un projet américain de défense militaire, le docteur Licklider avait écrit que "les ordinateurs devraient être conçus afin de pouvoir coopérer dans la prise de décision et le contrôle de situations complexes" [64] ; ses travaux ont été à l'origine de *ARPANET*, l'ancêtre de *Internet*. Aujourd'hui Internet est adopté par toutes les communautés, des chercheurs aux industriels, en passant par le grand public, et le commerce (le *e-commerce*) y est florissant. L'intégration des technologies de l'information à tous les niveaux de la société n'a pu se faire que grâce aux travaux de recherche incessants vers une connectivité toujours plus universelle dont un des résultats les plus connus a été le *Web*. Les grilles sont la continuité actuelle de cette recherche d'universalité des échanges électroniques qu'elles souhaitent étendre à toute ressource afin de répondre aux besoins toujours plus importants pour la science et l'industrie en termes de capacités de calcul, de stockage, ainsi que d'outils collaboratifs. Une ressource est traditionnellement une entité physique, telle que les ordinateurs, le réseau ou les systèmes de stockages. Les grilles étendent cette notion à un niveau plus élevée pour désigner toute entité pouvant être partagée et utilisée à distance. Les organisations, les gouvernements eux mêmes, ont saisi l'ampleur des enjeux et investissent massivement dans les infrastructures nécessaires aux développements de ces nouveaux outils.

1.1 Contribution

Les travaux de cette thèse ont entièrement été basés sur *XtremWeb* [23], une plate-forme de calcul global développée au Laboratoire de Recherche en Informatique de l'université Paris Sud.

Les contributions principales de cette thèse sont la ré-architecture de la plate-forme, l'étude de coopération inter grilles, le passage et l'utilisation de protocoles standards sur une plate-forme de calcul global, ainsi que la mise en production d'XtremWeb. Ainsi, nous verrons que :

- l'architecture d'XtremWeb a du être repensée afin de répondre à de strictes exigences de qualité de service. Cette ré-architecture a aussi permis de mettre en œuvre les points qui suivent ;
- notre plate-forme de calcul global offre une solution innovante de coopération inter-grilles. Cette solution légère, standardisée et globale d'agrégation de ressources de calcul distribuées sur des domaines administratifs séparés permet d'ordonnancer des applications multi-paramétriques grâce à un gestionnaire de ressource centralisé ;
- notre plate-forme, bien qu'initialement architecturée afin de distribuer des applications multi-paramétriques, est capable de prendre en charge des protocoles standards afin de faire circuler des messages entre ressources distribuées parmi différents domaines administratifs. Nous verrons que notre proposition sécurise de tels transferts ;
- la physique des hautes énergies utilise notre plate-forme malgré les contraintes sévères sur ses besoins applicatifs.

XtremWeb repose sur un ensemble de caractéristiques originales : c'est une plate-forme pluridisciplinaire, générique, ouverte et sécurisée. La plate-forme XtremWeb est générique en ce sens qu'elle n'est pas dédiée à une seule classe d'applications parallèles. Le premier type d'applications visées fut les études paramétriques où une application séquentielle est exécutée sur grand nombre de paramètres différents. Plus tard nous avons proposé une bibliothèque permettant la programmation d'application maître/esclave et une extension pour le passage de message proposé par des protocoles "standards". Bien que les grilles puisse exploiter tout type de ressources (CPU, stockage, logicielle, réseau), XtremWeb elle-même se destine principalement à la mutualisation de d'ordinateurs (ou *ressources de calcul*).

La méthodologie privilégiée pour l'étude du calcul global pair-à-pair est celle de l'expérimentation. L'analyse théorique des protocoles et des propriétés du système n'est pas abordée dans cette étude, car bien qu'essentielles nous pensons qu'elles constituent un sujet d'étude à part entière. Nous proposons une évaluation de performances de la plate-forme elle-même sur une application destinée à la physique des hautes énergies. En tant qu'instrument d'étude du calcul global et pair-à-pair, XtremWeb est également évaluée à travers le regard des utilisateurs. A cet égard nous nous intéressons principalement au déploiement, dans un but de production et dans le cadre d'une collaboration académique internationale : le projet Auger.

1.2 Plan du manuscrit

Le manuscrit est divisé en chapitres comme suit :

Chapitre 2 expose l'état de l'art des grilles de calcul, des systèmes pair à pair et du calcul global. Ce chapitre caractérise ces différents systèmes et en expose les avantages et les limitations qui nous ont amené à proposer une nouvelle plate-forme.

Chapitre 3 introduit XtremWeb dans sa version disponible au début de cette thèse et détaille la version obtenue grâce aux travaux de cette thèse. Nous expliquons en quoi et pourquoi il a été nécessaire de ré-architecturer cette plate-forme et de redéfinir les services. Nous argumentons notre propos en nous appuyant sur différents principes théoriques que notre plate-forme doit suivre. Nous exposons les choix techniques qui ont du être faits afin d'appliquer ces principes théoriques ; nous exposons son architecture en couches et les services proposés par cette plate-forme. Nous détaillons les aspects en termes de sécurité, de communications et de capacités d'entrées sorties. Ce chapitre conclut par les apports d'XtremWeb dont a pu bénéficier la communauté de recherche en informatique, en présentant sommairement les projets qui ont pu être menés autour de notre plate-forme.

Chapitre 4 détaille le concept de « gliding », un concept qui permet d'échanger des ressources entre différents types de plate-forme de grille. Grâce à ce nouveau concept, on peut tirer parti du meilleur de différents types de plate-forme. Nous verrons qu'XtremWeb peut par exemple profiter de la plate-forme *Condor*, de ses capacités de *checkpoint* ou d'entrée/sortie distantes tout en proposant, de son côté, une solution d'ordonnancement global incluant la migration automatique de tâches. On peut ainsi générer dynamiquement différentes plates-formes permettant de répondre à des besoins variés. Il est même alors facile de répondre à des besoins ponctuels. Les grilles ainsi obtenues sont configurables à la demande. Nous montrons que les performances de plates-formes ainsi obtenues sont satisfaisantes, ainsi que l'équilibre des charges des différentes ressources utilisées sur des sites géographiquement distribués. Ce chapitre conclut que les différents types de grille ne sauraient être la solution universelle dans l'état actuel des choses et que ce nouveau concept, le gliding, peut permettre de mixer des environnements, à priori orthogonaux, et d'obtenir dynamiquement différentes plates-formes en fonction des besoins.

Chapitre 5 détaille la problématique de la translation des services de type "cluster" sur une plate-forme de calcul global. Nous expliquons ce qu'est un service "cluster" et son mode de fonctionnement. Nous résumons l'historique de ce type service et expliquons en quoi il est nécessaire de leur faire passer la transition des grilles. Nous prenons comme exemple de type de service, ceux basés sur le protocole *SunRPC* et nous proposons une solution pour les faire élégamment migrer des clusters aux grilles. Notre proposition ne demande aucune régénération des services existants, ni au niveau des fichiers sources (il n'est pas nécessaire de recompiler), ni au niveau des librairies (il n'est pas nécessaire de réinstaller les services existants). Nous montrons qu'un déploiement et une configuration adéquats de notre plate-forme permet de *gridifier* les services clusters. Nous conclurons que l'optimisation des performances pour une telle utilisation reste encore un travail à faire.

Chapitre 6 détaille la mise en production d'XtremWeb pour la physique des hautes énergies. Nous expliquons l'environnement informatique de cette communauté scientifique et ses besoins auxquels nous présentons une proposition. Nous comparons notre plate-forme avec les ressources de calcul standards, comme les centre de calcul, et les grilles institutionnelles et montrons le profit que peut tirer la communauté scientifique d'une telle plate-forme. Nous concluons qu'XtremWeb peut être très utile à résoudre les besoins de cette communauté et mettons en perspective l'avenir de notre plate-forme dans ce contexte.

Chapitre 7 Ce chapitre est la conclusion du manuscrit.

Chapitre 2

État de l'art

Résumé. Dans ce chapitre nous présentons les différents enjeux relatifs aux grilles, les différents types de plate-forme, ainsi que les différentes implémentations qui existent à l'heure actuelle.

Nous distinguons deux grandes familles de grille, les grilles institutionnelles et les plate-forme de calcul global, que nous détaillons en termes d'architecture et de services, mais aussi de qualité et de sécurité. Nous comparons ces deux familles afin de bien comprendre les tenants et aboutissants de chacune.

Nous voyons que si les caractéristiques semblent orthogonales, les architectures et les moyens pour les mettre en œuvre sont similaires. Nous comprenons clairement que ces deux familles convergent grâce, notamment, à une volonté de standardisation des différentes technologies.

Enfin, après avoir présenté et comparé les différentes plates-formes de calcul global, nous justifions le choix de la plate-forme XtremWeb afin de venir à bout des travaux de cette thèse.

2.1 Introduction

Les avancées technologiques récentes permettent de simuler des systèmes d'une complexité jamais atteinte, ainsi que de stocker et de gérer des quantités d'informations inimaginables il y a encore quelques années. Ces avancées permettent l'apparition de nouveaux champs d'études et projettent des objectifs jusqu'alors hors de portée. Grâce à nouvelles technologies, de nouveaux scénarii organisationnels peuvent être mis au point, où le travail et ses nécessaires outils sont distribués et communautarisés.

Les grilles informatiques sont apparues à la fin des années 1990 sur le constat que la généralisation des moyens informatiques, leurs augmentations en nombre et en puissance, tant en termes de capacités de calcul que de stockages, ainsi que l'avancée exponentielle des infrastructures des réseaux, l'augmentation et la complexité de leurs interconnexions et de leurs débits, entraînaient une administration et une utilisation toujours plus complexes. Foster et Kesselman, dans leur ouvrage « The Grid, Blueprint for a New Computing Infrastructure » [47], ont proposé une approche standardisant l'utilisation de ces moyens et définissaient la notion « d'organisation virtuelle » (*VO*). Cette idée émergente et prometteuse a rapidement été suivie par les plus grandes équipes de recherche à travers le monde.

La communauté internationale de la physique des hautes énergies, par exemple, va bientôt utiliser un nouvel outil, le *LHC (Large Hydron Collider)*, un accélérateur de particule de nouvelle génération construit par le CERN à Genève afin de tenter de détecter le *boson de Higgs*, ce qui permettrait de comprendre les problèmes posés par les masses des particules élémentaires (pourquoi en ont elles une et pourquoi diffèrent elles ?) et de tester des modèles d'interactions fortes telles que la supersymétrie (ou l'unification des quatre forces : la gravité, la force électromagnétique, l'interaction faible et l'interaction forte). Le LHC générera huit cent millions de collisions par seconde pour une masse de données de l'ordre de plusieurs péta-octets par an, distribuée dans différents laboratoires dans le monde. Plusieurs projets, EGEE [29], OSG [39], NorduGrid [37], proposent de déployer des systèmes de gestion et d'utilisation de ressources réparties entre différentes communautés virtuelles, regroupant tous les intervenants autorisés.

La notion de *grille* qui recouvre l'ensemble des dispositifs logiciels et matériels nécessaires pour l'exécution d'applications de grande taille sur plusieurs sites informatique et/ou sur une communauté de ressources (ordinateurs personnels ou *PC*) est devenue indissociable du calcul numérique et du stockage à grande échelle. L'affluence rencontrée lors des récentes réunions du Global Grid Forum et du projet EGEE montre la mesure de l'intérêt de la communauté internationale pour les GRID. Les expériences menées dans le cadre d'EuroGRID indiquent que les grands centres de calcul sont très actifs dans ce domaine et poussent les GRID vers la production. Le programme d'ACI GRID en France a permis de fédérer des équipes de recherche multi-sites autour de projets pluridisciplinaires avec des applications en physiques, biologie, géologie, des recherches fondamentales sur les mécanismes associés aux systèmes de GRID et des projets middleware d'envergure internationale.

Dans le même temps, l'apparition de projets tels que BOINC[13] (initialement connu sous le nom de *Seti@Home* [12]), leurs architectures de calcul inédites ainsi que leurs champs d'application, ouvraient la voie au calcul global (GC). Depuis, de nombreux travaux ont été menés autour de ce concept et il existe aujourd'hui plusieurs plate-forme de grille de PC[65, 6, 26, 14, 23], de calcul global[13] et de calcul pair à pair (P2P)[50].

2.2 Les grilles

Les grilles ont initialement été des interconnexions de gros centres de calcul. Les premières expériences ont été réalisées aux Etats Unis, en reliant plusieurs sites de calcul grâce au réseau NFSNET [47] dont l'objectif était de construire et mettre en œuvre un réseau haut débit (1,5 Mbis/s) parmi une douzaine de centres. Les réseaux Gigabit/s qui succédèrent à NFSNET et plus particulièrement vBNS furent l'occasion de lancer en 1995 un appel à projets pour démontrer la faisabilité d'applications réunissant une douzaine de laboratoires, l'utilisation d'un réseau haut débit et la visualisation par des environnements de réalité virtuelle. Les projets retenus furent regroupés dans le projet Iway. Ce projet avait la responsabilité de mettre en œuvre l'infrastructure système permettant l'exécution de ces applications. Les infrastructures de grille de calcul actuelles descendent directement de cette progression incrémentale.

Cette progression nous montre que les mécanismes mis en œuvre sont des extensions des mécanismes existants afin de mettre à disposition d'un ensemble d'utilisateur des ressources appartenant à sites géographiquement et administrativement distribués. Par exemple, la sécurité repose sur des systèmes d'identification et de quota traditionnels et l'accès aux ressources est réalisé par l'intermédiaire de systèmes de batch avec des mécanismes de priorité pour répartir l'accès aux ressources suivant les droits des utilisateurs. L'ensemble des outils du système Globus [46] qui fournit ces mécanismes illustrent parfaitement cette extension des principes utilisés dans les centres de calcul.

Le succès des grilles doit beaucoup à l'émergence relativement récente de principes architecturaux clairs ainsi que de solutions logicielles standardisées et ouvertes qui permettent de définir et mettre en œuvre des “organisations virtuelles” où les ressources sont partagées (dématérialisées, en quelque sorte), les utilisateurs répartis entre administrations séparées non seulement géographiquement, mais aussi du point de vue légal.

La mise en place d'une telle organisation nécessite une architecture, des définitions et des protocoles clairement établis. Ce chapitre présente l'architecture générale des grilles, introduit la notion de *services* nécessaires à l'implémentation de cette architecture et aborde les problèmes de sécurités à résoudre pour le déploiement et l'utilisation d'une telle plate-forme.

2.2.1 Architecture

Ce chapitre présente une architecture simplifiée de grilles, avec ses principaux composants et leur liens. Cette architecture, synthétisée par la figure 2.1, est présentée sous forme de couches, dont chacune s'appuie sur les fonctionnalités de la couche inférieure.

La couche (0) (la couche *matériel*) représente les ressources elles mêmes, un ensemble technologique hétéroclite comprenant les capacités de calcul, de stockage et tout autre entité physique ou logique. Les opérations spécifiques à chaque ressource permettent un partage plus ou moins complexe de cette dernière au niveau de la grille (le co-ordonnancement, la réservation etc.). Le partage sur une grille exige que les ressources soient introspectives (qu'elles puissent fournir leurs états) et configurables (qu'elles puissent être contrôlables). Par exemple une ressource de calcul doit permettre de démarrer, stopper et contrôler des tâches, elle doit être capable de fournir ses caractéristiques logicielles et matérielles ainsi que la liste des tâches en

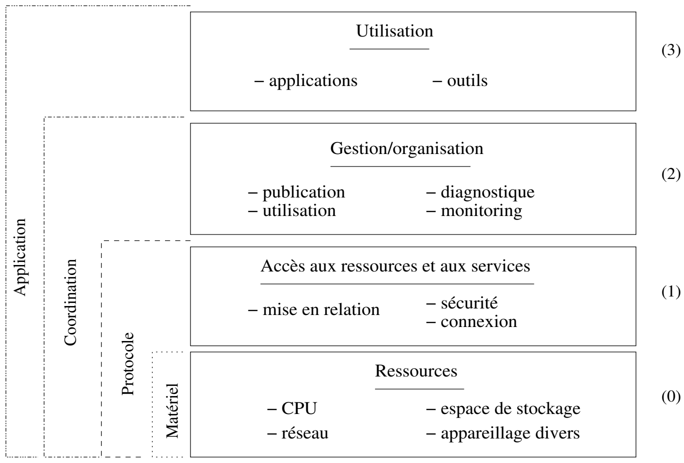

FIG. 2.1 – L'architecture des grilles.

attente ou encore le taux d'occupation du CPU. Les implémentations proposées par les plates-formes de grilles considèrent généralement que les services d'introspection et de gestion des ressources sont fournis par les possesseurs et les administrateurs des ressources eux mêmes. Elles offrent cependant les outils permettant la standardisation des accès aux ressources afin de faciliter leur intégration aux niveaux supérieurs.

La couche (1) (la couche *protocole*) définit un ensemble d'abstractions et de protocoles (*TCP, HTTP, SSL...*) permettant l'accès aux ressources de la couche inférieure. Elle permet la construction de services et d'outils applicatifs dans les couches supérieures, tels que la découverte et la publication des ressources du niveau *matériel*, leur gestion et leur surveillance. Cette couche contient les protocoles nécessaires aux transactions réseaux telles que l'authentification et la communication. Les implémentations de cette couche offrent l'authentification unique (*single sign-on*) et la délégation ainsi que la sécurisation des communications. Elles s'appuient sur des standards tel que les certificats *X.509* ou la couche de transport sécurisée *TLS*. La sécurité est discutée dans le chapitre 2.2.6. Cette couche utilise un protocole multiphase [66], tel que celui de la figure 3.13, pour assurer une prise en compte fiabilisée des messages.

La couche (2) (la couche *coordination*) contient les outils nécessaires à la gestion de la grille. Elle s'appuie sur les couches inférieures pour construire deux protocoles distincts : le protocole *information* qui permet d'accéder aux informations d'introspections des ressources, et le protocole *gestion* qui permet de configurer et d'utiliser les ressources. Ces deux protocoles permettent de construire, au niveau de la grille (à un *macro* niveau, donc), des services de coordination des ressources parmi lesquels on trouve la *publication*, l'*allocation*, la *réservation*, le *monitoring*, la *tolérance aux pannes*, la *réplication* etc.

La couche (3) (la couche *application*) est la couche où se situent les applications finales. Elle peut être plus ou moins complexe en fonction des besoins des utilisateurs et peut elle même se diviser en plusieurs plusieurs couches ; on peut par exemple y trouver une première couche contenant certains protocoles et librairies de paradigmes de programmation parallèle comme *MPI*, sur laquelle on trouverait la couche des applications finales.

L'architecture présentée par la figure 2.1 est implémentée par différents projets parmi lesquels on peut citer la *boîte à outils* Globus [46] proposée par le laboratoire d'Argonne. Les technologies d'implémentation ont évolué depuis l'émergence de l'idée des grilles ; on est ainsi passé de l'utilisation de standards de fait dans le *Globus Toolkit 2 (GT2)* à une formalisation en terme de services dans *Globus Toolkit 3 (GT3)* qui propose OGSA (*Open Grid Service Architecture*), une architecture tournée vers les services.

Le chapitre suivant présente les services permettant de mettre en œuvre une grille.

2.2.2 Les services

Les services sont une avancée qu'on peut qualifier de "naturelle" dans l'histoire de l'informatique ; ils sont directement issus du passage de l'informatique monolithique telle qu'on la pratiquait encore dans les années 1990 à une informatique de plus en plus distribuée avec la montée en puissance des services applicatifs, du nombre de ressources en constante augmentation dans les institutions publiques ou privées, grandes ou petites, de l'éclatement géographique des sites institutionnels, des besoins de collaborations inter-institutions et, finalement, de *l'e-business.*

Un service est une entité communicante et auto descriptive grâce à une interface -un ensemble de méthodes publiques- et des comportements standardisés. Ensemble, ces deux notions définissent des entités identifiables et éphémères, c'est à dire susceptibles d'être créées et détruites par requête, dynamiquement publiables et utilisables à la demande, disponibles et gérables à distance (ceci inclut la durée de vie). L'effort de standardisation comprend l'utilisation des standards des services web [3] (WSDL, SOAP, UDDI...)

La spécification *Open Grid Service Infrastructure (OGSI)* définit les interfaces, les comportements et les conventions permettant de créer, nommer, contrôler etc. les services grilles. Les services grilles permettent de créer une plate-forme fonctionnelle. Il existe des services de découverte et de virtualisation, des services de gestion de données, d'autres pour la sécurité. Le GGF [1] est un groupe de travail réfléchissant à la définition des services et de leurs différentes interactions et proposant d'éventuelles nouvelles interfaces pour OGSI.

La distribution des infrastructures des systèmes d'information a complexifié leurs utilisations et leur gestion, jusqu'à faire apparaître la nécessité d'une *virtualisation* en termes de services afin d'améliorer la cohésion de ces systèmes de plus en plus compliqués. La figure 2.2 (extraite de [38])compare les différentes infrastructures des systèmes d'informations et montre le parallèle entre l'évolution des besoins applica-

[1] GGF (Global Grid Forum) `http ://www.gridforum.org`

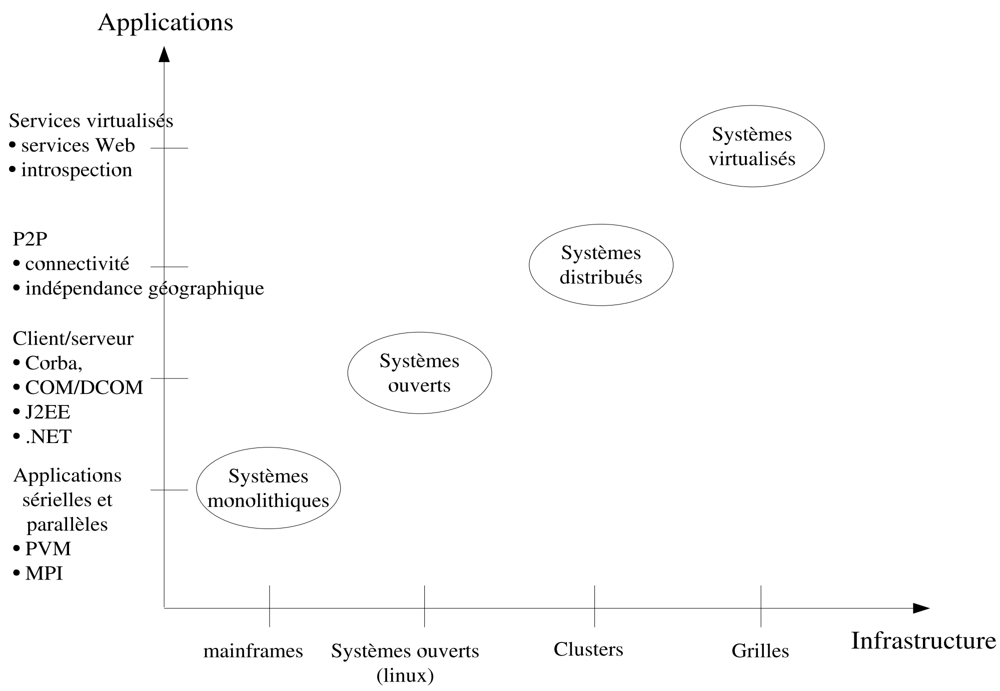

FIG. 2.2 – De la centralisation à la virtualisation.

tifs et les infrastructures mises en place ; nous voyons que les technologies applicatives se détachent de plus en plus des infrastructures.

Dans les systèmes monolithiques, les applications sont fortement liées aux infrastructures. L'évolution des applications est compliquée et coûteuse. Les applications de type client/serveur reposent sur des systèmes ouverts tels que Linux ou Windows, ce qui permet de lever la dépendance aux matériels. Malheureusement ces systèmes ouverts ne sont pas intér-opérants et les applications type client/serveur dépendent d'une infrastructure logicielle telle que *.NET, J2EE, Corba* etc., ce qui rend toute relative la flexibilité de ce type d'application. L'approche distribuée étend la notion de client/serveur à une plus large échelle grâce à des implémentations fiables de passage de messages à grande échelle et à une plus grande modularité quant à l'utilisation des ressources gérées par ensembles ou *clusters*, ce qui permet de configurer les nœuds de manière dynamique afin d'allouer les ressources aux applications en fonction de leurs besoins. Enfin, les systèmes virtualisés, tels qu'architecturés par *OGSA*, introduisent la notion de services permettant la standardisation, l'introspection, la découverte, l'instanciation etc. d'applications distantes.

La virtualisation des services est un concept permettant de simplifier les systèmes d'information devenus aujourd'hui extrêmement complexes tout en diminuant les coûts de maintenance, de partager des ressources tout en assurant un bon niveau de sécurité. La virtualisation est architecturée en trois étages comme le montre la figure 2.3. Comparativement à la figure2.1, nous remarquons que les couches matériel et protocole disparaissent pour être englobées dans des notions plus abstraites, ne tenant aucun compte des caractéristiques physiques, car seules importent les capacités fonctionnelles. La couche *application* contient un ensemble d'interfaces et

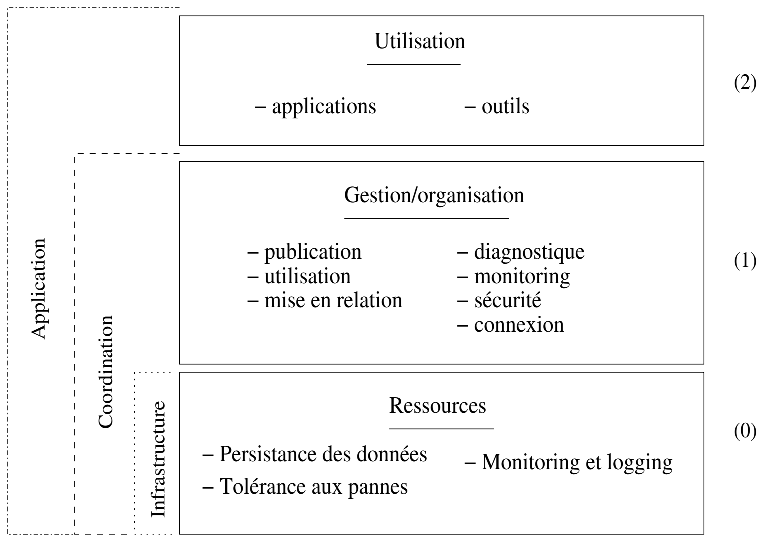

Fig. 2.3 – Architecture de la virtualisation

d'environnements permettant aux applications finales d'accéder à la grille. La couche *coordination* gère les ressources et ordonnance les requêtes ; elle propose les services de base de la grille (introspection, sécurité, gestion...) permettant de gérer les requêtes de la couche applicative en fonction des priorités organisationnelles (sécurité, occupation des ressources, temps de réponse etc.). Ces services sont construits grâce aux services de la couche inférieure. Enfin, la couche *infrastructure* gère les ressources hétérogènes et les pannes qu'elles peuvent occasionner. Elle propose ses services à la couche supérieure qui sont, pour la plupart, inaccessibles à la couche applicative.

2.2.3 La gestion des services

La gestion des ressources est l'art de les contrôler, les gérer et les utiliser. Avant les grilles, les ressources étaient distinguées par leur type et leurs capacités, et ne concernaient que des entités physiques. On trouvait des ressources réseaux, clusters et stockages de données, par exemple, et chacune avait leurs propres protocoles d'accès et de gestion. Les grilles généralisent cette notion de ressource à toute entité partageable et gérable à distance afin de standardiser leur accès et leur gestion, d'une part, et de aussi pour briser les limites des sites administratifs, d'autre part. Les ressources dans les grilles étant des entités abstraites, elles ne se limitent pas aux entités physiques mais désignent toute entité accessible par des moyens de communication. Les grilles permettent une approche à la fois plus unifiée et plus générale que ce que l'on a pu connaître dans l'utilisation traditionnelle des clusters. Par exemple, sur une grille, une soumission de tâches peut dépendre de la disponibilité d'autres ressources distantes en fonction de leurs politiques de disponibilité (base de données, nombre et types de CPU disponibles etc.). Sur un cluster, la résolution d'une telle problématique ne pourrait être dynamique et nécessiterait l'intervention d'un administrateur pour ajouter des machines, ouvrir les accès aux différents serveurs etc.

La gestion des ressources dans les grilles a pour ultime but de résoudre des demandes (les *consommations*) grâce à un ensemble de disponibilités (les *productions*) selon des politiques d'échanges établies entre les producteurs et les consommateurs (les *Service Level Agreement (SLA)* [70]).

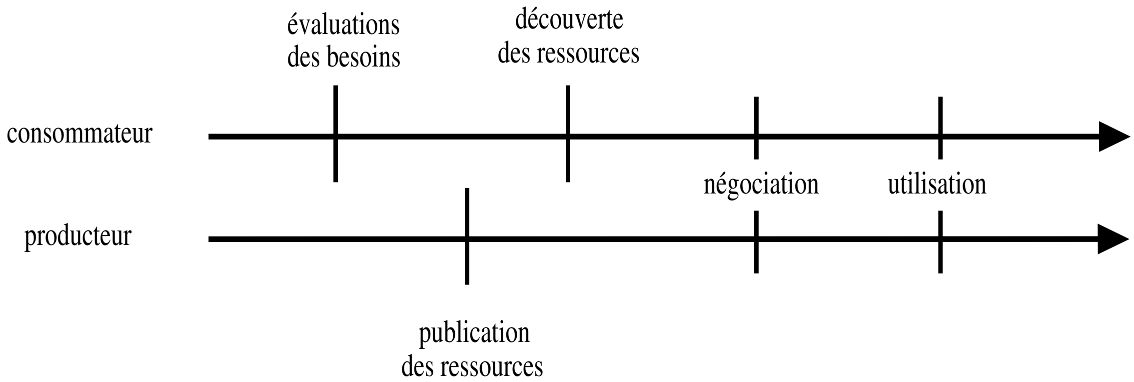

FIG. 2.4 – Publication et utilisation des services

La figure 2.4 schématise un cycle d'utilisation de service d'une grille. Les deux flèches représentent le temps, celle du haut pour le consommateur et celle du bas pour le producteur. Le producteur propose des services et les publie sur la grille. Le consommateur évalue ses besoins et interroge la grille quant aux capacités (aux *services*) qu'elle propose afin de répondre à ces besoins, grâce aux services de découvertes de la grille : la grille est *introspective*. Une négociation, transparente pour les deux parties, permet de vérifier la conformité aux politiques d'utilisation ainsi que les droits nécessaires : la grille est *sécurisée*. L'utilisation des services par le consommateur est alors éventuellement acceptée. Dans ce cas, le consommateur pourra suivre l'évolution des ses requêtes : la grille est *gérable* ; un consommateur, relativement à ses droits, peut intervenir sur les ressources (vérifier leur statut, les configurer, les arrêter, les activer...) Toutes ces caractéristiques, l'introspection, la sécurité, la configuration et la gestion, sont elles mêmes représentées par des services sur la grille : ce sont les services de base de la grille.

La négociation (SLA) est un point important qui permet d'assurer la mise en relation des services nécessaires au consommateur relativement à ses besoins tout en tenant compte des politiques de sécurité et d'utilisation souhaitées par le producteur. La résolution d'une telle négociation peut être relativement complexe et la grille assure la bonne utilisation des contrats ainsi négociés. Par exemple si un consommateur souhaite des services de calcul pour exécuter des tâches ayant besoin de services d'accès à certaines données, la plate-forme de la grille assure que le contrat d'accès aux données sera avalisé afin que le contrat de calcul puisse être rempli dans les conditions nécessaires au bon déroulement des tâches soumises. Tout ceci, bien sûr, sous couvert que les conditions de sécurité soient remplies pour les deux contrats.

Il existe trois types de SLA. Les *Task Service Level Agreement (TSLA)* sont créés au moment de la soumission d'une tâche ; ils contiennent la définition de la tâche et de ses besoins en termes de ressources (par exemple, type de CPU et d'OS, quantité de mémoire etc.). Les *Resource Service Level Agreement (RSLA)* permettent de négocier le droit à l'utilisation d'une ressource (par exemple, l'accès à une base de données). Enfin, les *Binding Service Level Agreement (BSLA)* qui permettent de relier entre eux d'autres SLA afin de répondre globalement à des requêtes clientes. Dans notre exemple précédent, un TSLA est crée pour la soumission de la tâche, un RSLA est crée pour l'accès aux données et un BSLA met le tout en relation afin d'exécuter correctement les tâches.

2.2.4 Qualité de service

Les capacités toujours croissantes en hautes technologies permettent d'obtenir des résultats inimaginables il y a encore peu dans l'industrie comme dans les laboratoires de recherche. En contre partie, elles demandent toujours plus d'attention et de soins que ne cessent de lui apporter les ingénieurs afin qu'elles rendent, jour après jour, les services demandés avec une qualité optimum. On parle donc de *qualité de service (QoS)*. Ce QoS dans les systèmes centralisés, multi-utilisateurs, multi-applications était *relativement* simple. L'apparition des clusters au sein de domaines administratifs, la multiplication des services et leur distribution inter-serveurs ont grandement compliqué la gestion et la maintenance des services proposés aux utilisateurs. Les connexions inter-domaines en ont fait un cauchemar.

Le QoS peut se mesurer selon différents facteurs dépendants du type d'application et du spectre de performances recherché. On peut la mesurer selon une précision souhaitée (la précision des réponses à une requête, par exemple), le nombre de réponses souhaitées ou encore le temps de réponse. Le QoS, dans un site administratif, est assuré par le site lui même et dépend des ressources mises en œuvre : systèmes de surveillance, main d'œuvre affectées etc.

Le QoS sur les grilles dépend de celui des ressources partagées, les grilles n'intervenant guère qu'au niveau de la tolérance aux pannes. D'un autre côté, les grilles permettent un très haut niveau de sophistication de gestion de ces services distribués, grâce à la standardisation de leurs accès. Les grilles, avec la standardisation qu'elles proposent, offrent une vue simplifiée de l'ensemble des services partagés et fortement hétérogènes, et offre une vision d'un seul système. C'est, paradoxalement, comme un retour vers les systèmes monolithiques et leur gestion *simple* : on voit un seul type de plate-forme, on a besoin d'un seul type d'outil. Les grilles proposent un *système d'exploitation (OS)* virtuel offrant une vision unifiée des services hautement distribués et hétérogènes.

2.2.5 Services et fonctionnalités

Après avoir présenté l'architecture et les services des grilles, ce chapitre décrit les relations entre les fonctionnalités visibles par l'utilisateur et les mécanismes internes des ces plates-formes. Ces fonctionnalités et ces mécanismes sont communs aux différents types de plates-formes de grilles. Les grilles institutionnelles, les plates-formes de calcul global ainsi que les plates-formes pair à pair ont toutes ces caractéristiques ou, du moins, tous ces types de plates-formes visent à les avoir. Par exemple les grilles institutionnelles ne mettent pas en œuvre la tolérance aux pannes, car elles s'appuient sur des infrastructures considérées stables et sécurisées, et ne fournissent pas de mécanisme quant à cette capacité.

Les fonctionnalités principales sont au nombre de cinq.

1. La sécurité inclut deux composantes : la confidentialité qui concerne l'identité des ressources et des utilisateurs ; et l'intégrité qui concerne la bonne exécution des services disponibles dans le respect des ressources mises en œuvre.
2. La disponibilité et la persistance assurent la pérennité des services afin qu'ils restent accessibles. Cette caractéristique est particulièrement dépendante de la tolérance aux pannes de la plate-forme et de ces capacités à dupliquer et migrer les services entre les ressources.
3. L'équité assure la bonne répartition des ressources entre tous les utilisateurs. Cette caractéristique s'appuie sur des politiques de partage et éventuellement de crédits.
4. Les performances concernent les accès à tous les services de la plate-forme ; la découverte des services et leur le monitoring, aussi bien que l'accès aux données et l'ordonnancement de tâches. En particulier les performances ne doivent souffrir ni de la gestion ni des pannes des ressources, ni même du nombre de ressource mis en œuvre.
5. La taille de la plate-forme ne doit affecter ni la gestion des ressources, ni leurs accès car la disponibilité et les performances en seraient alors affectées, ce qui n'est pas admissible.

On voit clairement que si certaines caractéristiques sont indépendantes entre elles, elles partagent les responsabilités quant aux fonctionnalités. Par exemple, les performances dépendent de la découverte et de la mise en relation, de la gestion des ressources et de leurs pannes, ainsi que des capacités de migration et de réplication des services. Tout comme la taille de la plate-forme qui est d'ailleurs intimement liée aux performances, même si l'utilisateur a l'impression de voir deux caractéristiques séparées.

2.2.6 La sécurité

Tous les jours, nous achetons et nous vendons des biens et des services. Un tel geste n'est pas anodin ; il est le résultat de toute une démarche, consciente ou non, afin d'établir une relation de confiance et une sensation de sécurité. En échangeant des coordonnées banquaires, numéro de carte de paiement ou simple chèque, nous échangeons des informations extrêmement sensibles, nous accordons notre confiance, que nous soyons vendeur ou acheteur.

Aujourd'hui, la plupart d'entre nous connaissent et utilisent les services commerciaux disponibles sur le *Web* pour acheter des services ou des biens. On le fait généralement en donnant ses coordonnées de carte banquaire. De même que pour des échanges physiques du monde réel, des processus de sécurisation sont nécessaires dans le monde virtuel présenté par l'Internet. Les différences entre la sécurisation du monde physique et celle du monde virtuel résident dans les moyens à mettre en œuvre pour établir cette sécurité. Sur l'Internet, sur le Web et sur les grilles, la sécurité concerne l'intégrité des ressources et des informations, les droits nécessaires à leur accès, la confidentialité des transferts ainsi que l'authentification à la fois des ressources et des utilisateurs. Comme dans le monde réel, la sécurité et la confiance sont des notions difficiles à définir et compliquées à mettre en œuvre.

La sécurité n'est pas statique et sa mise en place dépend d'un nombre important de paramètres. Il convient de définir les buts à atteindre, les moyens pour les mettre en œuvre, les comportements et habitudes des utilisateurs, les caractéristiques des ressources elles mêmes, celles qui sont intrinsèquement sécurisées et sûres, et les autres. Dès lors, il faut comprendre les implications d'une telle sécurité et établir un compromis acceptable entre la sécurisation et ses conséquences, d'une part, et les conditions d'utilisation, d'autre part, car les mécanismes et les technologies à mettre en œuvre afin d'obtenir le niveau de sécurité souhaité ont un coût qui se paie, en général, par des conditions spécifiques d'utilisation, ainsi qu'en termes de performances.

Les grilles, en partant d'environnements dynamiques et à large échelle, construisent des organisations virtuelles (*VO*) permettant de partager des ressources de manière standardisée et sécurisée. La sécurité de ces VO est à deux niveaux, celui de la VO elle même, et celui des différents domaines administratifs la composant. La sécurité de la VO est une couche au dessus des sites compris dans cette VO ; elle intègre la sécurité définie dans les différents sites et ajoute un niveau d'abstraction supplémentaire afin que les membres de la VO aient accès à toutes les ressources de celle ci dans la mesure de leurs autorisations. La sécurité au niveau de la grille ne remplace pas la sécurité des sites pris en compte, mais rend ces différentes sécurités communicantes et inter-opérantes. La figure 2.5 schématise la sécurité d'une VO composée de trois sites. Chaque site a sa propre politique de sécurité avec ses utilisateurs et ses ressources référencées, dont certains font partie de la grille construite. Cette dernière fournit une couche de sécurité supplémentaire permettant l'interconnexion des utilisateurs et des ressources des trois sites appartenant à la grille. Les utilisateurs des ressources de la grille peuvent alors utiliser les ressources partagées entre les trois sites, dans le cadre de la grille.

Un des problèmes posés par la sécurisation de la grille est son évolution au cours du temps. La liste des ressources, des services et même des utilisateurs est susceptible d'être modifiée au cours du temps ; la sécurité au niveau de la VO doit prendre

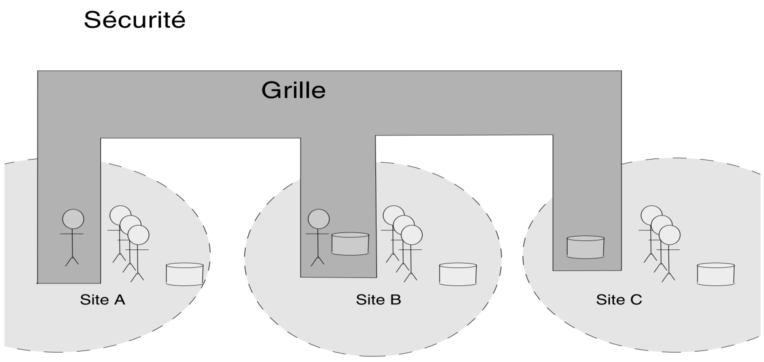

FIG. 2.5 – Les domaines de sécurité

en compte cette dynamicité.

Pour atteindre ce niveau de sécurité sur les grilles, la sécurité doit implémenter plusieurs points précis [53].

L'intéropérabilité. La sécurité des grilles ne fige spécifiquement aucune technologie ; n'importe quelle infrastructure de sécurité peut être utilisée dans le cadre de la sécurité globale d'une VO. De fait, les grilles rendent inter-opérables les différents systèmes d'authentification existants [62].

La protection des messages. Généralement la protection des messages se fait au niveau de la couche transport, grâce au protocole de sécurité *SSL*. Ce protocole nécessite une connexion point à point, ce qui n'est pas satisfaisant dans le cadre des grilles où l'émetteur ne peut souvent pas se connecter directement au destinataire, mais doit passer par un ou plusieurs intermédiaires. La protection des messages dans une grille doit donc être assurée dans la couche message et non pas dans la couche transport.

La gestion des droits. L'authentification est unique au sein de la VO ; un utilisateur ne doit s'authentifier qu'une seule fois afin d'accéder aux ressources dont les accès lui sont autorisés, quel que soit le nombre de ressources. Les droits sont délégables entre infrastructures afin de permettre la gestion des ressources inter-sites ; lors d'une délégation, les intermédiaires sont tous identifiables afin de pouvoir reconstituer la chaîne des responsabilités, en cas de besoin. Les droits sont toutefois renouvelables après un certain temps afin de s'assurer que les accès ne restent pas constamment ouverts. La figure 2.6 schématise la délégation de pouvoir : l'utilisateur s'identifie et crée un mandataire de pouvoir qui se connecte au site distant (typiquement, le *GateKeeper* est expliqué au chapitre 2.2.7) qui crée un mandataire local représentant le mandataire distant afin d'accéder aux ressources locales.

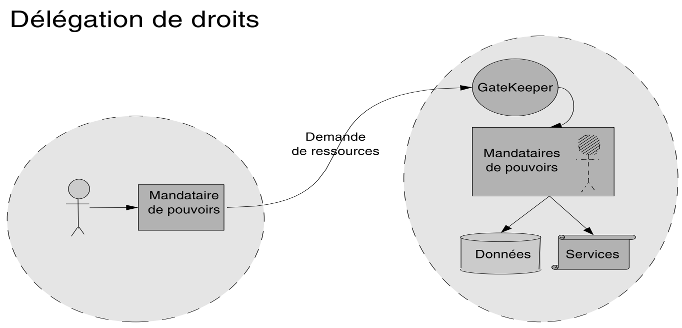

FIG. 2.6 – La délégation des droits

La confidentialité. La confidentialité, voire l'anonymat, peut être requis dans certaines situations. Les différents sites partagés dans le cadre d'une VO peuvent avoir une politique à ce propos qui doit être respectée par la VO.

La journalisation des événements. La sécurité inclut tous les moyens permettant d'auditer la plate-forme afin de pouvoir déterminer les responsabilités en cas de problème. On utilise à cette fin des systèmes de journalisation d'événements qui enregistrent les messages dans des journaux. Ces journaux doivent être eux mêmes sécurisés car ce sont des informations sensibles.

La sécurité concerne trois points supplémentaires : l'intégrité des données, l'intégrité des ressources, et le passage des pare-feux. Ces trois points sont discutés dans le chapitre 2.3.5.

2.2.7 La boîte à outils Globus

La boîte à outils *Globus* propose une infrastructure pour construire une grille afin de partager des ressources distribuées entre différents sites administratifs. Les outils Globus permettent de créer, superviser et contrôler un environnement hétérogène et dynamique, de sécuriser cet environnement, et enfin de gérer la soumission de tâches dans cet environnement. La boîte à outils contient un ensemble de composants offrant une infrastructure de base et permettant de construire une grille fonctionnelle ; cette construction est facilitée par l'adoption de protocoles standardisés et ouverts. Cette boîte à outils est devenue un standard de fait adopté dans de nombreux projets. Elle ne permet pas de construire à elle seule une grille, mais propose les composants de base afin de développer et d'intégrer une telle plate-forme. Par exemple, la boîte à outil ne fournit aucun système d'ordonnancement permettant de gérer les tâches soumises à la plate-forme. Il est de la responsabilité des différents sites de fournir ce composant et de l'intégrer à l'infrastructure de la grille.

La figure 2.7 schématise l'infrastructure proposée par Globus et représente un

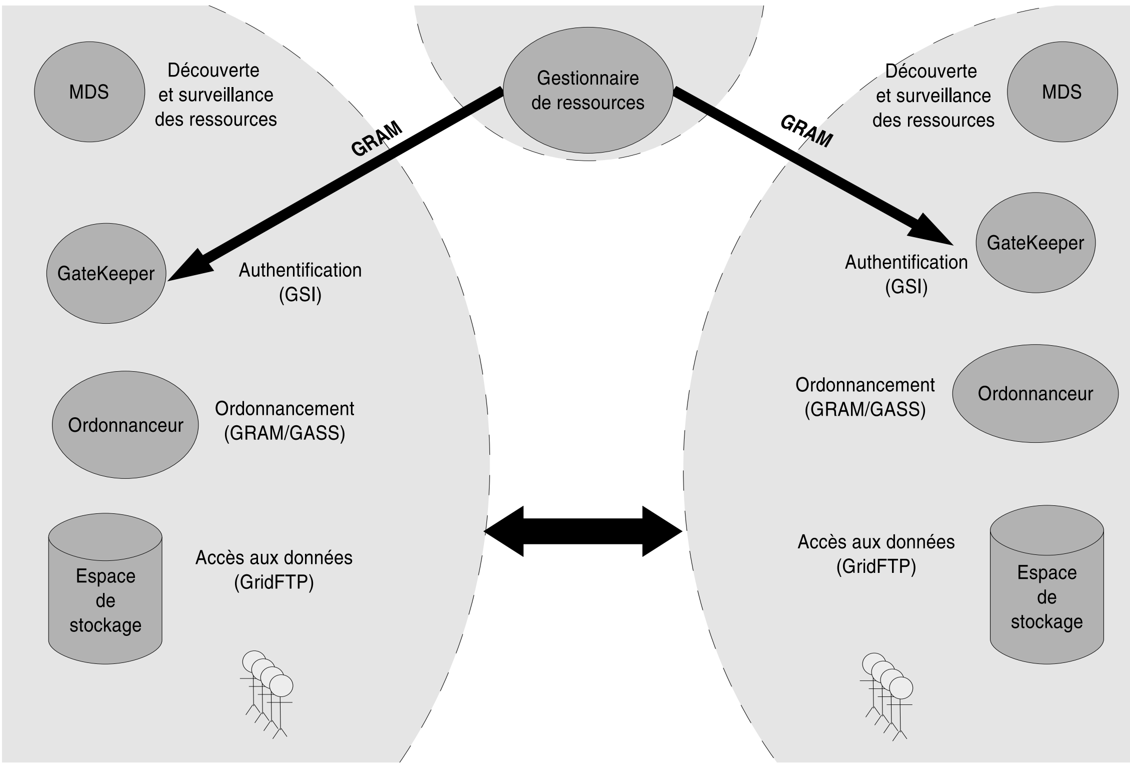

FIG. 2.7 – L'infrastructure « Globus »

sous-ensemble des protocoles et des services de cette boîte à outils. On y voit les services et protocoles principaux. Le service *MDS (Monitoring and Discovery Service)* offre un accès uniforme aux ressources, gère leur publication dynamique, leur surveillance et leur gestion. Le protocole *GSI (Grid Security Infrastructure)* permet l'authentification et l'identification des utilisateurs et des ressources ainsi que la délégation des droits. Ainsi un utilisateur de la grille ne doit se connecter qu'une seule fois sur la grille ; l'utilisation de ressources et de services distribués est alors possible en délégant les droits de l'utilisateur aux différents sites gérant les ressources et services. Le protocole *GRAM (Grid Ressource Allocation and Management)* permet l'allocation et la gestion de calculs distants. C'est un protocole multi-phases évitant les tâches orphelines et perdues ; le protocole multi-phases est décrit au chapitre 3.3.5. GRAM s'appuie sur GSI pour la gestion des droits et du service *GateKeeper* pour celle des ressources de calcul. Ce service, le GateKeeper est le point d'entrée utilisateur pour la soumission de tâche sur la grille ; un utilisateur souhaitant soumettre des tâches doit s'adresser à un GateKeeper déterminé. Celui-ci les ordonnancera sur le site dont il a la charge. Grâce à la délégation des droits utilisateurs, le GateKeeper peut récupérer sur la grille, au nom de l'utilisateur, les données dépendantes de l'ordonnancement des tâches. Le protocole *GASS (Global Access to Secondary Storage)* conjointement au protocole GRAM, est utilisé pour assurer la bonne exécution des tâches ; il permet de prendre en compte les besoins en entrées/sorties de tâches et de leurs résultats. Toutefois, ce protocole n'est généralement pas utilisé pour le transfert de données. On utilise pour cela le protocole *GridFTP* qui est une extension du protocole *FTP (File Transfert Protocole)* pour accéder aux données sur une grille. Ce service s'apprit, lui aussi, sur le protocole GSI pour la sécurité.

Une couche supplémentaire propose le service *Gestionnaire de Ressources (Ressource Manager)*. Ce service permet une abstraction supplémentaire pour la soumission et l'ordonnancement de tâches sur la grille. Nous venons de voir qu'un utilisateur doit s'adresser à un GateKeeper spécifique pour ordonnancer ses tâches sur les ressources du site géré par le GateKeeper choisi. Le service Gestionnaire de Ressource est un service d'ordonnancement global sur la grille, virtualisant les différents sites et leur GateKeeper. Grâce à la délégation de ses droits, un utilisateur peut soumettre globalement ses tâches au service Gestionnaire de Ressources. Celui ci choisira le site (et donc le GateKeeper) où soumettre les tâches de l'utilisateur. Il est toutefois notable que ce service ne propose pas de migration automatique des tâches soumises. Si le site choisi pour l'ordonnancement des tâches de l'utilisateur est soudainement indisponible, les tâches soumises sont bloquées tant que le site reste indisponible. Il reste de la responsabilité de l'utilisateur de résoudre de tels cas de blocage.

Nous verrons au chapitre 4 qu'XtremWeb propose une solution d'ordonnancement global incluant la migration automatique des tâches. Cette solution permet d'améliorer la qualité de service grâce aux mécanismes de tolérance aux pannes de la plate-forme XtremWeb.

2.3 Les systèmes pair à pair *(P2P)*

Les systèmes P2P permettent de construire une plate-forme à partir de ressources individuelles, que ce soient des données, des espaces de stockage ou encore de la puissance de calcul. Bien que la littérature considère les systèmes de Calcul Global comme des systèmes P2P [32], la définition la plus communément admise pour les plates-formes P2P les définit comme étant le partage de ressources éphémères, non stabilisées, par des échanges directs, sans passer par aucun intermédiaire, aucun serveur. Ce type d'architecture se caractérise par ses capacités à agréger des ressources instables, incontrôlées et incontrôlables, pour construire une plate-forme stable grâce aux capacités de tolérance aux pannes intrinsèques à l'infrastructures P2P. Des projets comme BOINC [13], Gnutella [100], OceanStore [61] ou d'autres, intégrant complètement ces caractéristiques, ont prouvé non seulement la faisabilité, mais aussi la fonctionnalité de ce type de plate-forme.

L'utilisation de ressources éphémères hors de toute d'administration centralisée offre des caractéristiques radicalement différentes des systèmes distribués plus classiques, contenus au sein de domaines administratifs définis :

1. elle permet l'agrégation de ressources à une échelle bien plus importante ;
2. elle distribue les responsabilités et même la notion de possession ;
3. et donc elle résiste à la censure et peut même jusqu'à poser des problèmes de propriété et des droits afférents.

Finalement cette architecture change radicalement les méthodes de travail [95] en accélérant les communications et en réduisant les coûts de partage.

Les grilles et les systèmes P2P ont le même but d'organiser le partage d'un grand nombre de ressources [49]. Pour ce faire les grilles s'appuient sur des infrastructures stables et sécurisées ainsi que sur des protocoles standardisés. Avec l'augmentation de la taille des plates-formes souhaitées, les grilles doivent aujourd'hui résoudre des problèmes d'organisation et de tolérance aux pannes. Ces deux caractéristiques sont intrinsèques aux systèmes P2P qui ont été imaginés et conçus afin d'atteindre des plates-formes de très grande taille et ont dès le départ intégré les mécanismes de gestion de l'instabilité intrinsèque aux ressources partagées et donc de tolérance aux pannes, ainsi que les mécanismes permettant l'organisation et la réorganisation automatiques. Cependant les plates-formes P2P existantes sont généralement incapable d'interagir entre elles afin de partager des ressources inter plates-formes, bien qu'il existe des travaux en ce sens (Cf chapitre 3.4.4), car, étant partie de rien si ce n'est de l'existence de l'Internet, elles ont tout crée et n'ont pas utilisé de technologies standardisées de haut niveau. On peut résumer les différences entre grilles et P2P ainsi : « les grilles gèrent les infrastructure mais pas encore les pannes, tandis que les P2P gèrent les pannes, mais pas encore les infrastructures » [51]. De nombreux travaux sont en cours afin d'unifier les avantages de ces deux types de plates-formes.

Pour construire un système Pair à Pair, il existe plusieurs points de départ suivant le type de réalisations visées. Pour simplifier, nous pouvons considérer trois types de réalisations possibles : créer un protocole de système Pair à Pair, créer un système distribué Pair à Pair, créer une application reposant sur une infrastructure Pair à Pair.

Pour créer un protocole de système Pair à Pair, les technologies logicielles sont à la base celles associées à l'Internet : protocoles TCP, HTTP. Certains systèmes ont été construits au dessus le protocole de mail classique SMTP. Rapidement, le mode d'interaction par appel de procédures distantes s'est avéré comme particulièrement adapté et des projets comme XtremWeb ont été construits par dessus Java RMI ou JINI. Depuis les premières expériences et les premiers systèmes, d'autres protocoles ont été proposés comme SOAP et XML RPC. SOAP (Simple Object Access Protocol) qui encapsule des RPC encodés au format XML dans des messages HTTP (notons que SOAP n'est pas limité au protocole HTTP) est retenu par beaucoup de projets comme protocole de base pour le développement. Indépendamment du protocole et du mode d'interaction, le système de mise en relation avec les mécanismes évoqués comme la durée de vie d'une requête, l'évitement de boucle dans le protocole de diffusion, le cache, etc. doivent être construits de toute pièce par le développeur. Il est à noter que le Peer-to-Peer Working Group, notamment soutenu par Intel, entend développer des technologies de base pour faciliter le développement de systèmes Pair à Pair en abordant par exemple les problèmes de la traduction des adresses IP (NAT) et des pare-feux, de la sécurité. Actuellement une bibliothèque de sécurisation des communications fondée sur OpenSSL est mise à disposition.

Pour simplifier la tâche du développeur, des environnements sont proposés permettant de construire des systèmes Pair à Pair plus facilement. Ainsi Jxta de SUN propose de structurer un système autour de 4 éléments :

1. les « Peer pipes », mécanisme de connexion d'un pair à un autre et de partage d'informations sur le réseau et de manière distribuée ;
2. « les Peer groups » qui permettent la création de groupes de façon dynamique et le regroupement logique et cohérent de contenu ;
3. la possibilité de surveiller et de mesurer les interactions et de définir des politiques de contrôle entre pairs (Peer monitoring) ;
4. des mécanismes de sécurité permettant de garantir la confidentialité, l'identité et l'accès contrôlé aux services.

Le niveau supérieur dans la hiérarchie des environnements pour le développement de systèmes Pair à Pair consiste en des plates-formes complètes et opérationnelles dans lesquelles il « suffit » d'intégrer l'application désirée. FastTrack et XtremWeb proposent des plates-formes adaptées respectivement pour les applications d'échanges de documents et de calcul distribué. FastTrack est une plate-forme industrielle payante alors qu'XtremWeb est une plate-forme de recherche à code source ouvert. Tous les deux proposent des outils de configuration et d'administration permettant à l'administrateur d'installer simplement, de mettre en œuvre et de contrôler le système. Ils incluent aussi les mécanismes de diffusion des programmes clients et offrent des interfaces utilisateurs.

2.3.1 Définition

Il est difficile de trouver un consensus dans la définition du P2P. La littérature s'accorde toutefois sur la capacité qu'ont ces systèmes à atteindre des tailles énormes en agrégeant un très grand nombre de ressources. Les puristes estiment que ces systèmes sont complètement distribués et *plats*, c'est à dire qu'aucun nœud ne se distingue des autres car ils ont tous les mêmes fonctionnalités. Une telle définition rejette toutefois les systèmes comprenant des *super nœuds* pouvant faire office de serveurs locaux. Cette notion est implémentée, par exemple, dans Kazaa [2] qui est pourtant largement accepté par tout un chacun comme un système P2P. Une autre définition [99] estime que « une application P2P se distingue par ses capacités à tirer profit de ressources dans chaque coin de l'Internet », ce qui inclut alors les systèmes basés sur des serveurs centraux comme BOINC [13] et XtremWeb.

Quoi qu'il en soit, ces différentes définitions, divergentes sur l'architecture d'un système P2P, semblent toutes s'accorder sur la vision qu'ont les utilisateurs de ces systèmes : un systèmes P2P permet d'agréger des ressources hors de toute contrainte administrative (des ressources « libres ») pour atteindre un but commun (partage de données ou de CPU). Nous adopterons cette définition pour notre propos.

On peut classifier les systèmes P2P selon leur destination.

1. Les systèmes de *collaboration* permettent d'échanger des messages en temps réel dont les systèmes de messagerie instantanée en sont les plus connus. On y trouve Msn, Yahoo Messenger, mais aussi Jabber [1].
2. Les systèmes *d'échange de données* sont peut être les systèmes P2P les plus populaires. Ils permettent d'échanger des fichiers électroniques. La gamme de ces systèmes varie des échanges relativement simples et directs entre nœuds, comme Napster [77], FreeNet [28] et Gnutella [100], jusqu'à la construction de systèmes de fichiers hautement élaborés, comme c'est le cas dans CAN [83], CHORD [103], PASTRY [87] ou TAPESTRY [108] qui construisent des tables de hashage distribuées.
3. Les systèmes de *calcul distribué* permettent d'utiliser les puissances de calcul sur le mode du vol de cycle. Elles sont centralisées pour la coordination mais aussi pour la collecte des résultats. On y trouve des plates-formes comme BOINC [13], Condor[66] et XtremWeb.

2.3.2 Architecture

La figure 2.8 présente l'architecture des systèmes P2P. On note la très grande similitude avec la figure 2.1, avec toutefois une différence de taille : la couche (1) contient une fonctionnalité inédite pour les systèmes de grilles, la *tolérance aux pannes* qui permet de construire une plate-forme stable sur un ensemble de ressources éphémères. Si cette différence est la plus notable, ce n'est pas la seule. Les couches ne sont pas interprétables de la même manière entre les systèmes P2P et les autres.

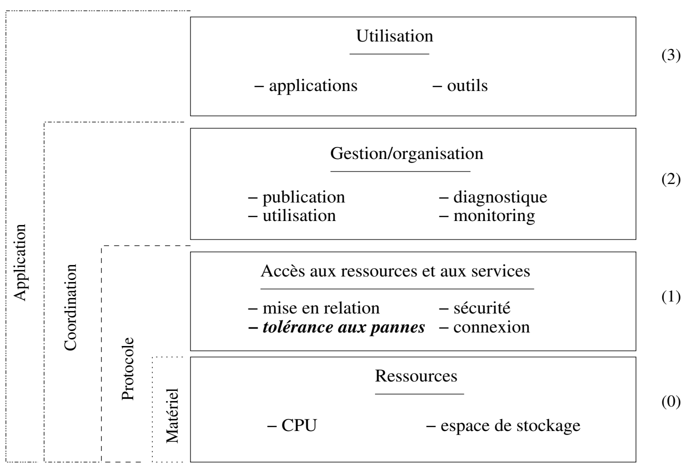

FIG. 2.8 – L'architecture des systèmes P2P.

La couche (0) contient les ressource elle mêmes, quel que soit le type de grille. Toutefois les ressources sont moins hétéroclites dans les systèmes P2P qui, contrairement aux grilles, ne proposent en général qu'un seul but, que ce soit le partage de données ou de ressources de calcul. Il n'existe pas pour le moment de systèmes P2P visant à résoudre plus d'une problématique à la fois. Dans les systèmes P2P, les ressources partagées sont introspectives mais non configurables.

La couche (1) diffère en ce que les systèmes P2P intègrent les protocoles nécessaires à la gestion de la tolérance aux pannes, ce que ne font pas les autres systèmes de grille.

La couche (2) contient les protocoles sur lesquels sont construits les services de base de la grille comme la *publications*, le *monitoring*, l'*allocation* etc.

Enfin la couche (3) contient les applications utilisateurs.

2.3.3 Technologies P2P

Les protocoles mis en œuvre dans un systèmes P2P dépendent de la topologie et du degrés de centralisation de celui-ci. En utilisant le degrés de centralisation, on peut regrouper les systèmes P2P en trois grandes familles.

Les systèmes entièrement décentralisés mettent toutes les ressources au même niveau en supprimant la notion de serveur ; les ressources sont alors dites « servents » (pour SERVeur et cliENT). Dans cette catégorie, on trouve par exemple FreeNet [28] où la transmission de l'information est réalisée par l'infrastructure de mise en relation, toute comme Gnutella et Fastrack qui utilisent une infrastructure distribuée pour remplir la fonction de mise en relation. FreeNet utilise une propagation en chaîne pour transférer les messages de nœud en nœud suivant une politique de transition définie localement par les nœuds eux mêmes afin d'optimiser la propagation. On peut résumer les caractéristiques des systèmes entièrement distribués ainsi : les ressources sont très éphémères ; le système inclut des mécanismes de réplication suffisants pour assurer la pérénité des services ; les utilisateurs acceptent des performances dites du « meilleur effort ». Les premières méthodes connues de mise en relation étaient assez brutales et inondaient l'infrastructure réseau afin d'essayer de trouver des ressources incluses dans la plate-forme. Ces méthodes avaient l'inconvénient d'utiliser inutilement le réseau et de ne pas passer les pare-feux. De nouvelles méthodes ont été depuis imaginées parmi lesquelles on trouve la marche aléatoire [68] qui permet de réduire significativement l'utilisation du réseau et d'améliorer la découverte de ressources voisines. L'étude de Gnutella [84] a montré que la plupart des ressources ont une *connexion faible*, c'est à dire que leur présence n'influe pas sur la connectique globale de la plate-forme, alors que nombre très restreint de ressources ont une *connexion forte* dans la mesure où elles jouent un rôle central quant aux possibilités de mise en relation des ressources de la plate-forme. Le système de recherche de ressources de Gnutella fonctionne d'une façon totalement distribuée. Un pair n'est connecté qu'à un ensemble limité de pairs voisins. L'ensemble des voisins forme une topologie logique qui n'a pas de correspondance avec la topologie physique d'Internet. Le système de mise en relation de Gnutella est construit pardessus cette topologie logique et fonctionne par diffusion en utilisant le protocole TCP. Un pair qui recherche un document (ou un service) diffuse une requête à ses voisins dans la topologie logique. La requête est propagée par les voisins à leurs propres voisins. Le protocole agit par inondation des nœuds (pairs) présents dans le système. Si un pair qui reçoit la requête possède le document recherché, il ne propage plus la requête et envoi une réponse à destination de l'émetteur de la requête, en indiquant son adresse. La réponse suit le parcours inverse de la requête en se propageant de pair en pair jusqu'au destinataire. Cette méthode fonctionne car tous les pairs propageant une requête retiennent l'origine et la destination (voisins immédiats) par lesquels la requête est passée. Gnutella possède un mécanisme pour limiter l'inondation : chaque requête possède une durée de vie (un compteur de sauts) qui est décrémentée à chaque fois qu'un pair propage la requête. Un mécanisme permet aussi d'éviter les boucles dans le protocole d'inondation.

Le mécanisme de recherche de ressources de FreeNet fonctionne aussi de manière totalement distribuée, selon une organisation différente de celle de Gnutella. Comme dans Gnutella, un nœud FreeNet ne connaît que des voisins logiques. Cependant, dans FreeNet toute ressource est cryptée et identifiable à partir d'une clé unique. Un nœud FreeNet stocke des relations clés-adresses et le système de propagation de requêtes ne transmet la requête qu'au voisin possédant une clé proche de la clé de la ressource recherchée. Une clé peut représenter le cryptage du contenu de la ressource entière ou d'un ensemble de mots clés identifiant la ressource.

Dans Napster comme dans Gnutella, lorsque l'émetteur de la requête reçoit une réponse, il contacte directement le pair possédant le document recherché en utilisant l'adresse IP contenue dans la réponse. Dans Gnutella, le protocole de transfert consiste en une requête HTTP (get) émise par le pair recherchant le document. Le pair qui possède le document reçoit la requête HTTP sur un numéro de port différent de 80 et y répond en transmettant le document. FreeNet a été conçu pour minimiser la possibilité d'identifier les machines clientes et serveurs de documents. Il n'utilise pas de système de connexion directe pour éviter que le pair émetteur de la requête obtienne l'adresse IP du serveur. Le document est transmit au destinataire par le système recherche de ressource lui-même qui devient aussi le système de transport. En fait, chaque nœud FreeNet fonctionne comme un cache pouvant stocker dans chaque relation clé-adresse, la ressource correspondante. Le système de cache ne retient que les ressources les plus recherchées si bien que le temps de recherche d'une ressource souvent référencée est très rapide. En revanche les ressources les moins recherchées ont peu de chance de se trouver dans le système de caches et demandent plus de temps de recherche. Des mécanismes de transports plus sophistiqués peuvent être mis en œuvre pour faciliter l'échange de fichiers très volumineux. Par exemple, dans Edonkey2000, l'émetteur de la requête peut solliciter plusieurs machines possédant un document (ou un segment du document) pour paralléliser le téléchargement. Il demande à chaque machine un segment différent du document. Dès qu'il a lui même reçu un premier segment, il peut servir de serveur pour d'autres machines recherchant ce segment du document.

Les systèmes partiellement décentralisés définissent certaines ressources comme des serveurs locaux afin de limiter le nombre de messages nécessaires à la découverte et de faciliter la mise en relation. Dans le protocole compatible Gnutella proposé par la société Clip2, chaque serveur local agit comme un mini serveur Napster ; c'est à dire qu'il stocke un index des ressources et des pairs possédants ces ressources. Ainsi le serveur local ne propage pas les requêtes pour les références qu'il possède dans son index, il répond directement à l'émetteur de la requête. Les serveurs locaux modifient uniquement le système de mise en relation. Le mécanisme de transport reste le même. Le système FasTrack fonctionne de manière analogue. L'un des problèmes liés à l'introduction de serveurs locaux dans un système comme Gnutella est la cohérence des informations stockées dans l'index avec la réalité. En principe un serveur local doit mettre à jour son index lorsqu'une machine disposant d'une ressource se retire du système et lorsqu'une machine ne dispose plus d'une ressource. Ce type de plate-forme est mis en œuvre par des systèmes comme CAN [83], CHORD [103], PASTRY [87] ou TAPESTRY [108] qui construisent des tables de hashage distribuées. Toutefois ce type de système pose des problèmes dans un environnement où les nœuds ne restent pas suffisamment longtemps connectés, empêchant la plate-forme de se stabiliser.

Les plates-formes de calcul global définissent un serveur central coordonnant la plate-forme. Les nœuds sont autonomes et peuvent dialoguer entre eux sans passer par le serveur, mais le serveur reste la pièce maîtresse de la plate-forme, assurant la cohérence et la coordination. L'une des questions ouvertes sur les systèmes de Calcul Global Pair à Pair concerne l'intérêt d'établir une connexion directe entre les nœuds. On trouve dans cette catégorie des systèmes comme BOINC, Napster et XtremWeb. Dans Naptser, le répertoire qui permet à un client d'identifier les serveurs potentiels est centralisé. Dans BOINC et XtremWeb, l'ordonnanceur de calcul et le collecteur de résultats sont centralisés.

2.3.4 Qualité de service

La qualité de service (QoS) dans des environnements hautement volatiles reste une question ouverte sujette à de nombreux travaux. Contrairement aux grilles institutionnelles (Cf chapitre 2.2.4), les systèmes P2P ne peuvent s'appuyer sur l'infrastructure pour assurer une qualité de service ; les plates-formes P2P doivent intégrer les mécanismes nécessaires à la résolution de ce problème.

Par exemple, on mesure le QoS des systèmes de fouille de données, comme Gnutella [84], par le nombre de réponses obtenues dans un temps raisonnable ; passé ce délai, le système est considéré comme inopérant. Il convient donc de garder le temps de réponse le plus bas possible. La technique *BFS* [107] essaie de minimiser ces coûts en envoyant prioritairement les messages aux nœuds les plus actifs. D'autres techniques consistent à regrouper les nœuds par familles afin de leurs envoyer les messages correspondants aux caractéristiques familiales, afin d'augmenter la probabilité d'obtenir plus rapidement une réponse pertinente. Ceci peut se traduire par une spécification des requêtes par genres musicaux, si on considère les systèmes d'échange de ce type de fichiers. Ces deux propositions sont toutefois incomplètes ou trop compliquées : les BFS dépendent d'une topologie ad-hoc ce qui rend les résultats non prédictibles dans un environnement hautement éphémère ; les caractérisations familiales n'ont d'intérêt que pour les requêtes utilisant des paramètres compatibles et clairement définis.

Mais le QoS concerne aussi la disponibilité des services qui sont assurés par les mécanismes de tolérances aux pannes des système P2P. Ce problème est généralement abordé par des mécanismes de détection de faute et de réplication. La détection de fautes s'appuie souvent sur la notion de consensus qui permet de prendre une décision quant une faute est détectée.

Il a été démontré qu'aucun consensus ne peut être établi dans un environnement asynchrone, sans ressource stable, dès l'apparition de la première panne [44]. Les plates-formes à grande échelle (*LSDS*), telles le P2P, communiquent en mode *non connecté*, c'est à dire qu'il n'existe pas de lien permanent entre les différentes entités souhaitant communiquer. On peut comparer ce mode de communication avec les communications téléphoniques ; pour communiquer, on compose le numéro, on dialogue avec l'interlocuteur et on raccroche à la fin de la communications. A contrario, les communications *connectées* offrent un lien permanent entre interlocuteurs, comme avec les communications télévisuelles où il n'existe pas de protocoles de mise en relation, ni de fin de communication : le lien n'est jamais interrompu.

Le mode non connecté à une grande importance quant à la compréhension des LSDS et aux mécanismes utilisables pour y résoudre les pannes. Le mode non connecté introduit des délais théoriquement infinis dans la transmission des messages, en ce sens qu'il est impossible de détecter de manière sûre qu'un message qui ne semble pas arriver à destination n'a jamais été envoyé ; c'est à dire qu'on ne peut être sûr de détecter une erreur, seules des suppositions peuvent être faites. Nous sommes donc amenés à considérer les LSDS comme des systèmes asynchrones [15]. D'autre part, les ressources étant volatiles, elles peuvent disparaître de la plate-forme et ne jamais s'y reconnecter. L'utilisation de détecteurs de faute afin d'obtenir un consensus [25] n'est pas une approche satisfaisante pour les LSDS, car elle suppose la définition de deux hypothèses non valables pour de tels systèmes, la définition d'un écart de temps permettant de déterminer de façon sûre la non arrivée d'un message et la définition d'une *majorité* de ressources non fautives. Ces deux définitions sont inapplicables sur un LSDS à cause de la dynamique d'une telle plate-forme où les ressources se connectent et se déconnectent constamment.

La tolérance aux pannes a largement été étudié dans les systèmes d'objets distribués ; par exemple FT-Corba [15] propose des mécanismes de détection de fautes et de reprise sur erreur grâce à la réplication. Le système détecte les objets suspectés de n'être plus fonctionnels et choisit une nouvelle instance pour continuer le travail en cours. La sélection de la nouvelle instance suppose l'établissement d'un consensus pour son élection, ce qui est incompatible avec les LSDS, comme nous venons de le voir. Legion [73] propose un modèle permettant l'implémentation de la tolérance aux pannes suivant différents modèles : la réplication passive, les checkpoints, la journalisation pessimiste de messages. Legion nécessite toutefois que le réseau soit fiable ou, à défaut, que certaines ressource le soient, ou encore les deux ; ces conditions ne peuvent malheureusement pas être remplies sur un LSDS.

2.3.5 Sécurité

La sécurité est un point crucial dans les systèmes P2P qui sont des plates-formes où la notion de confiance n'existe pas, contrairement aux plates-formes institutionnelles.

La sécurité des ressources. Les ressources ne peuvent être partagées entre différents individus ou institutions que si leur intégrité sont assurées. Quand les codes à partager sont certifiés comme dans BOINC [13], les ressources n'ont pas à craindre de dysfonctionnement quant à l'exécution de ces codes, pour peu que l'intégrité du serveur soit assurée et que les ressources ne soient pas dirigées vers un serveur malicieux. La certification des connections permet de résoudre l'identité des intervenants, à la fois des ressources partagées et des serveurs. Toutefois cette certification est lourde à mettre en place en fonction du niveau de sécurité souhaité. Par exemple la génération de clés électroniques ne devrait pas se faire à travers l'Internet car elles pourraient être volées ou copiées lors du transfert et il conviendrait d'utiliser une infrastructure infaillible sur ce point précis (ce que n'est pas Internet, bien sûr). Mais une telle contrainte va à l'encontre d'une distribution à grande échelle et des projets comme BOINC [13] n'aurait jamais eu le succès qu'on lui connaît si cette contrainte avait été suivie à la lettre.

La sécurité du serveur relève de la sécurité de toute ressource informatique connectée sur l'Internet et n'est donc pas spécifiquement discutée ici. Toutefois quelques points notables entrent dans la distribution du calcul sur des ressources non sécurisées. Des ressources malicieuses pourraient en effet essayer de détourner les calculs en retournant des calculs volontairement erronés (Cf paragraphe suivant). D'autres pourraient essayer de retourner des résultats avec une taille énorme, ce qui revient à tenter une attaque de type *déni de service* [36] en bloquant l'accès réseau et en accaparant les capacités de stockage du serveur. Ceci est résolu en limitant la taille des résultats ; au delà d'une certaine limite, les résultats ne sont ni acceptés, ni pris en compte. BOINC utilise ce mécanisme de sécurisation. Enfin certains utilisateurs pourraient être tentés de tricher sur les crédits d'utilisation tels que mis en place dans BOINC. Ceci n'a pas d'importance tant les ressources utilisées ne sont pas rétribuées, mais pourraient avoir un impact énorme dans un système basé sur des rétributions économiques.

Dans les systèmes où les codes ne sont pas certifiés, comme XtremWeb, la sécurité des ressources est un problème beaucoup plus épineux et il convient alors de trouver des solutions confinant l'exécution des codes. Ce problème spécifique est étudié au chapitre 3.2.2.

La sécurité des données. Dans un environnement distribué, particulièrement lorsqu'ils partagent des ressources éphémères, une même ressource peut apparaître sous différentes identités. A moins d'utiliser une autorité de certification, ce type d'environnement reste sensible aux attaques de type *déni de service*. Une telle attaque tente de rendre indisponible une ressource ou un service du système en le surchargeant de tel manière qu'il ne puisse plus tenir son rôle. Il existe plusieurs algorithmes et protocoles de cryptographie permettant de sécuriser les données stockées ou en cours de traitement sur une ressource. L'auto-certification des données permet de s'assurer que les données ne sont modifiées entre deux accès. On peut, par exemple, calculer une clé de hashage à chaque accès aux données. Cette méthode, utilisée dans TAPESTRY [108], suppose que les nœuds connaissent la fonction de calcul de la clé. PASTRY [87], pour sa part, utilise un schéma de *secret partagé* où la clé d'encryption K permettant de calculer la clé d'un fichier est partagée en blocs stockés sur différents nœuds de telle manière que k morceaux suffisent à régénérer la clé K, mais $k-1$ morceaux soient insuffisants. De cette manière la responsabilité quant à la sécurité des données est partagée. Enfin, on peut trouver un autre exemple, mis en œuvre dans OceanStore [61], qui consiste à associer le codage par effacement avec le protocole de négociation *Byzantin* [63]. Cette méthode divise les données en blocs distincts qui sont stockés sur n nœuds de telle manière qu'une fraction de ces blocs suffisent à reconstruire les données. Il revient aux nœuds possédant les blocs de se mettre d'accord sur la pérénité des données et, éventuellement, de reconstruire les données en cas de corruption. Cette méthode de reconstruction accepte jusqu'à $n/3$ erreurs.

La sécurité des données concerne aussi les résultats générés par des calculs distribués. Il n'y a aujourd'hui pour ainsi dire pas de solution à ce problème. Par exemple, l'infrastructure BOINC [13] propose de faire exécuter plusieurs fois le même calcul par des ressources différentes afin de tenter de détecter les ressources défaillantes ou même malicieuses. Notre plate-forme, pour sa part, ne propose aucune piste quant à cette problématique et laisse cette responsabilité aux utilisateurs eux-mêmes.

Les pare-feux. Dans tous les systèmes Pair à Pair, se pose le problème des pare-feux qui protègent les pairs connectés au système. Ce problème concerne d'abord le système de mise en relation. En effet, pour mettre en œuvre le système de mise en relation, il est nécessaire que des pairs acceptent des connexions externes. Si aucun pair n'accepte de connexion externe ou si leur nombre est insuffisant, le système ne pourra pas être mis en œuvre ou sera très limité en performance. Le problème des pare feux se pose aussi pour le transport du document. Il arrive très souvent que le possesseur du document soit protégé par un pare feu. Dans ce cas, la requête HTTP de l'émetteur de la requête sera bloquée (port fermé). Pour contourner le problème Gnutella prévoit un mécanisme qui permet à l'émetteur de la requête de recontacter le possesseur du document par le système de mise en relation en lui demandant d'exécuter lui-même une requête HTTP (put) en direction du destinataire. Ce mécanisme appelé « push » inverse les rôles pour la transmission du document. Bien évidemment si l'émetteur de la requête de document est lui-même protégé par un pare feu, la transmission ne sera pas possible. C'est là une des limites intrinsèques des systèmes de transport de documents par connexion directe. La technique de transport utilisée dans FreeNet permet d'éviter, en partie, le problème des pare-feux puisque les pairs ne communiquent par directement mais par l'intermédiaire du système de mise en relation.

Dans notre plate-forme, bien que les nœuds soient capables de communiquer directement entre eux, la solution retenue afin de résoudre les problèmes liés aux pare-feux consiste, comme dans FreeNet, à utiliser un serveur centralisé.

Les problèmes que nous avons évoqués jusqu'à ce stade concernent globalement les applications d'échange de documents et de calcul Pair à Pair. Le calcul Pair à Pair pose cependant des problèmes spécifiques que nous évoquons dans les parties suivantes.

2.3.6 Systèmes P2P et grilles

Le Metacomputing et les systèmes de GRID (Globalisation des Ressources Informatiques et des Données) [47], activement développés pour les applications scientifiques de calcul ou de stockage massif ont pour ambition d'établir une infrastructure permettant, d'une part, de faire inter-opérer des ressources de calcul, des grands instruments de mesure, des grandes bases de données et des centres de visualisation et de réalité virtuelle, et, d'autre part, de faciliter l'accès et l'organisation de ces ressources pour l'exécution d'applications scientifiques.

Les systèmes de Calcul Global (Global Computing) concernent les applications scientifiques mais impliquent la participation de volontaires particuliers trouvant un intérêt social dans une expérience spécifique (lutte contre le cancer, recherche pharmaceutique, recherche d'intelligence extraterrestre, généthon). Le principe du Calcul Global est différent de celui des GRID. Il s'agit d'utiliser un très grand nombre d'ordinateurs volontaires distribués géographiquement à l'échelle mondiale et communiquant uniquement par Internet pour exécuter des applications informatiques de très grande taille. Le modèle de calcul est un parallélisme massif nécessitant peu de communications entre les sites. Le modèle d'exploitation est l'utilisation des ordinateurs d'institutions ou d'individus volontaires essentiellement pendant leurs périodes d'inactivité. Les systèmes de partage de ressources Pair à Pair ont, jusqu'à maintenant, concerné principalement la communauté des utilisateurs d'Internet cherchant à échanger des documents multimédias (musiques, films, articles). Dans ces systèmes, toutes les machines peuvent remplir le rôle de client et de serveur ; d'où le terme de ServEnt dans Gnutella [100]. Les ressources partagées peuvent être de différentes natures : données, espaces de stockage, puissance CPU. Le rôle de l'infrastructure système est de mettre en relation directe les machines recherchant des ressources (les clients) et celles disposant des ressources recherchées (les serveurs), à un instant donné.

Dans un article récent, Ian Foster et al. suggèrent une fusion des systèmes de GRID, de Calcul Global et de Pair à pair. Il est très probable que les systèmes de Calcul Global et de Pair à pair vont fusionner simplement parce qu'ils reposent sur des contraintes identiques et ont recourt à des principes de fonctionnement similaires. La fusion avec les systèmes de Metacomputing (interconnexion de grands sites de calcul) est plus délicate à court terme essentiellement parce que les mécanismes à mettre en œuvre dans les deux types de systèmes sont fondés sur des contraintes très différentes (une solution est toutefois présentée au chapitre 2.4.2). Les deux sections suivantes présentent l'origine des différences entre ces systèmes dans une perspective historique.

2.3.7 Les systèmes de calcul global pair à pair

Les systèmes de calcul global sont apparus alors que les clusters et les plates-formes à architectures hiérarchiques commençaient à être considérés comme des solutions trop limitantes, trop lourdes et trop chères. Ces systèmes apportent principalement l'utilisation de puissance de calcul sur le mode «vol de cycle»[98] au delà des frontières administratives. L'apparition des plates-formes de calcul global a introduit des caractéristiques fondamentalement différentes par rapport aux systèmes distribués connus jusqu'alors : le nombre d'ordinateurs connectés participant à la plate-forme d'une part, et le bannissement des frontières administratives, d'autre part. Ces deux paramètres ont à eux seuls des conséquences qui rendent ces plates-formes bien spécifiques et en tout premier lieu parce qu'elles ne contrôlent ni les participants (ni qui, ni quand, ni comment ils peuvent contribuer), ni la topologie (pares-feux, translation d'adresses etc.). Ce type de plate-forme ne contrôle donc pas sa propre infrastructure. On doit donc les considérer comme complètement asynchrones, dans la mesure où aucun consensus ne peut être établi[44] : il est impossible de détecter les erreurs avec certitude car on ne peut contrôler le transfert des messages.

Le vol de cycle a été le sujet de nombreux travaux tels que Condor [66], Glunix [55] and Mosix [16]. D'autres travaux ont étudié le partage inter-administrations : Jet [79], Charlotte [17], Javeline[27], Bayanihan [90], SuperWeb [11], ParaWeb [21] et PopCorn [75]. L'industrie s'intéresse aussi au potentiel de ces systèmes et propose des solutions de «grille de PC» (ou *«Desktop Grid»* (DG)), parmi lesquelles on peut citer *Entropia, United Devices, Platform, Grid Systems* ou *DataSynapse.*

Comme détaillé dans [50], les systèmes de calcul global et les grilles ont un même but d'agréger des ressources par delà les limites des frontières administratives avec, toutefois, deux différences majeures : la taille du systèmes (le nombre de nœuds agrégés) et leur contrôle (inexistant, donc, dans les GC). Ces deux aspects ont des conséquences directes quant aux propriétés théoriques de ces systèmes (l'architecture, les modèles de programmation, les méthodes de déploiement, la sécurité).

Un système de calcul global multi-utilisateurs, multi-application, est finalement très proche des systèmes pair à pair tel que Napster[40] ou Gnutella[91] avec des solutions similaires dans les mécanismes fondamentaux tels que les protocoles de communication, la publication et la recherche de ressources, ainsi que la coordination. Les systèmes de calcul global sont à la puissance de calcul ce que Napster ou Gnutella sont aux fichiers.

Les premières expériences de Calcul Global ont eu lieu dans les années 90. Les domaines d'applications identifiés étaient essentiellement la cryptographie [33] [18], et certains problèmes mathématiques (nombres premiers de Mersenne [71], calcul des décimales de Pi [80]). Les infrastructures mises en œuvre reposaient sur des protocoles comme le SMTP. Dans certains systèmes, les contributeurs devaient manuellement introduire de nouveaux paramètres de calcul dans l'application exécutée sur leur station de travail. Aucun mécanisme de sécurité n'était réellement mise en œuvre et les contributeurs comme l'utilisateur des résultats se faisaient réciproquement confiance. Le projet BOINC [13], sans résoudre le problèmes de sécurité, a permis d'organiser une infrastructure extensible jusqu'à plusieurs millions de contributeurs. Il a aussi démontré la faisabilité du calcul haute performance à partir d'une fédération de PCs connectés par Internet.

Les fédérations de PCs ont fait émerger la notion de ressources distribuées et mutualisées, connue actuellement sous le terme de système Pair à Pair. En parallèle au Calcul Global, s'est donc développé une autre catégorie de systèmes distribués fondés sur le partage des ressources de stockage et la messagerie instantanée et non plus de calcul. Les projets initiateurs qui sont devenus vite populaires ont été Napster [77], FreeNet [28] et Gnutella [100]. Dans ces systèmes, les ressources participantes fonctionnent à la fois comme des clients et des serveurs. Le système Napster a lui aussi démontré la faisabilité d'un système de mutualisation des ressources de stockage jusqu'à plusieurs millions de participants.

Les systèmes de Calcul Global et Pair à Pair se différencient des grilles de calcul principalement par quatre caractéristiques :

1. le nombre de ressources connectés est plusieurs ordres de grandeur plus grand (typiquement 100 000 ressources) et les ressources sont rarement parallèles (biprocesseur au maximum) ;
2. les ressources sont extrêmement volatiles ;
3. la topologie de l'infrastructure est à priori indéterminée ;
4. les utilisateurs sont aussi en nombre très important (typiquement un utilisateur par ressource).

Comparativement aux grilles de calcul, il faut composer avec une infrastructure matérielle déjà présente et évoluant de façon non coordonnée et en fonction d'impératifs sans relation évidente avec le calcul à grande échelle (les PCs sont plutôt conçus pour la bureautique et les jeux et les réseaux véhiculent les applications classiques d'Internet : mail, web, flux vidéo, audio).

L'ordre de grandeur et l'impossibilité d'action ou d'influence sur l'infrastructure engendre des problèmes spécifiques qui ne se présentent pas dans le cas des grilles de calcul. La sécurité, par exemple, doit être fondée sur des mécanismes extensibles à plusieurs centaines de milliers d'utilisateurs et de ressources. D'autre part, le placement et l'ordonnancement des tâches doivent prendre en compte l'évolution à court terme (la connexion et déconnexion des ressources) et l'évolution à long terme (modification d'infrastructure) du système. Globalement, les techniques adaptées dans le cadre des grilles de calcul sont inapplicables dans le cadre du calcul à très grande échelle. Les fonctions traditionnellement associées à la procédure de login (identification, sélection des utilisateurs, droits d'accès, priorité d'exécution, protection du site et traçage des opérations) ne passent pas à l'échelle. Inversement, la confiance de l'utilisateur envers les résultats renvoyés et la facturation de l'utilisation des ressources ne peut pas reposer, comme dans le cas des grilles de calcul, sur le caractère institutionnel des sites accédés. Les sites de calcul sont des machines quelconques appartenant à des utilisateurs auxquels on ne peut pas accorder à priori une confiance absolue. Le calcul à très grande échelle repose donc sur la capacité de certifier les résultats ou d'être capable de discerner les bons et les mauvais résultats. De même, contrairement au cas des grilles de calcul, l'ordonnancement d'un très grand nombre de tâches sur une centaine de milliers de ressources ne peut être géré à long terme et coordonné à court terme par des mécanismes centralisés.

Ainsi, les GRID et les systèmes Pair à Pair ont des racines historiques très différentes qui expliquent leur différence d'organisation et de problématique propre. L'étude de la fusion de ces systèmes est une question ouverte. La spécification OGSA (Open Grid Services Architecture) proposé par le laboratoire d'Argonne comme servant de base à la version 3 de Globus est une première tentative dans cette direction.

2.3.8 Les plates-formes de calcul global

A la fin du vingtième siècle, il commençait à être clair que les avancées technologiques ne pourraient suivre les besoins des utilisateurs toujours plus importants en termes de puissance de calcul et d'espaces de stockage[85, 78]. Il fallait trouver une solution aux limitations technologiques des systèmes d'alors qui étaient monolithiques, chers à l'achat comme à la maintenance, spécialisés et peu flexibles. Les systèmes complètement centralisés, où les terminaux ne communiquent qu'à travers une machine centrale, ont peu à peu été abandonnés pour le modèle client/serveur dont les premières implémentations étaient encore basées sur des logiciels et même des réseaux propriétaires, jusqu'à ce qu'apparaissent Unix et Arpanet (l'ancêtre d'Internet). Grâce à ces derniers, les applications ont commencé à s'émanciper du matériel ainsi que de l'architecture client/serveur stricto-sensus (TCP-IP est un protocole symétrique). Les infrastructures se sont alors ouvertes sur un modèle distribué, permettant de construire des clusters, qui rivalise avec les plus gros supercalculateurs [35] en termes de performances, tout en étant beaucoup moins chers.

La montée en puissance des capacités de calcul mais aussi et surtout de celles des réseaux ont permis à la communauté scientifique de résoudre des problèmes de plus en plus complexes. De même que la genèse du Web était issue de la volonté des chercheurs de se faciliter les partages d'informations, les grilles ont initialement été imaginées par les scientifiques, afin de résoudre leurs besoins toujours plus pressants en ressources coûteuses comme les espaces de stockage et les puissances de calcul. La montée en puissance des réseaux informatiques a permis d'imaginer des solutions distribuées d'autant plus nécessaires que nombres de travaux sont aujourd'hui régionaux, nationaux et même souvent internationaux.

A la fin des années 90, plusieurs projets regroupés sous le terme de « Calcul basé sur Internet » (Web-based Computing) comme JET [79], Bayanihan [90], SuperWeb [11], Javelin [74], Popcorn [75] ou Charlotte [17] ont vu le jour. Ces projets proposaient des environnements de développement pour les applications sur Internet. Le terme *Global Computing* était alors déjà utilisé pour représenter ces systèmes. La plupart sont nés avec le langage Java et exploitent certaines de ses caractéristiques comme un langage intermédiaire (bytecode) qui permet aux applications d'être portables sans recompilation et d'exécuter les applications dans environnement sécurisé pour la machine participante. Pour participer à une application distribuée, un utilisateur visite simplement une page Web dans laquelle est embarquée l'applet de calcul. Dans le principe, plusieurs de ces projets proposaient une vision du calcul sur Internet de type Pair à Pair dans laquelle toute machine offrant ses moyens de calcul pouvait aussi soumettre des requêtes de calcul. Le projet SuperWeb [11] avait identifié de nombreux problèmes scientifiques et techniques qui redeviennent d'actualité avec l'intérêt actuel pour les systèmes de Calcul Global. Toutefois, plusieurs questions n'avaient pas été abordées comme l'exécution de code natif, la prise en compte de l'extrême volatilité des ressources notamment dans le contexte des communications entre les PCs participants.

Le projet qui a réellement popularisé le concept de Calcul Global est SETI@home. Le lancement officiel du projet est très récent (Avril 1999) mais le projet et le développement du code remonte à 1997. Avant son lancement officiel, le projet SETI@home avait reçu environ 400 000 pré-inscriptions. Le dernier recensement indique 3 millions de machines participantes ou ayant participé à ce projet.

SETI@home a démontré plusieurs points : 1) il est possible de fédérer une communauté de plusieurs centaines de milliers d'utilisateurs pour faire du calcul numérique, 2) le système développé soutient une production de calcul considérable (20 Teraflops) ; rappelons que cette valeur n'est pas directement comparable aux performances des machines parallèles classiques, 3) l'organisation de serveur de calcul et du collecteur de résultats permettent de satisfaire plusieurs millions de tâches par jour. Le projet a déjà produit l'équivalent 500000 années de calcul de PC. Notons toutefois que la limite à l'approche centralisée de SETI@home est le réseau. Deux éléments montrent la vulnérabilité d'un tel système : l'expérience SETI@home utilise à elle seule la moitié de la bande passante réseau du campus de Berkeley (information obtenue lors d'un entretien avec David Anderson) ; la coupure mécanique (vandalisme) de la fibre optique reliant les serveurs de l'expérience au campus a provoqué l'interruption d'activité pendant plusieurs dizaines d'heures.

Ces résultats ont ouvert la voie à des projets académiques et industriels. Il faut noter que la plupart des projets visent la mise en place de plates-formes de Calcul Global. Il existe actuellement dans le monde plusieurs dizaines de projets de ce type. Les plus connus sur le plan industriel sont Entropia, Parabon, United Devices, Popular Power. Ces projets sont multi-applications. Dans le monde académique, les projets sont souvent dédiés à une application (Xpulsar, Evolution, Distributed.net) ou visent la généralisation du concept de Calcul Global au Pair à Pair (COSM). Les projets académiques mono-application ont comme objectif principal la production de calcul. Ces plates-formes peuvent être considérées comme des spécialisations de plates-formes de Calcul Global généralistes. Leur spécialisation permet de résoudre beaucoup des problèmes de sécurité que nous détaillerons par la suite. Les projets industriels ne dévoilent ni leur architecture ni les mécanismes utilisés pour la sécurité des ressources. Certains projets commerciaux et la plupart des projets académiques généralistes proposent la description du protocole de communication entre les ressources et offrent le code source des programmes fonctionnant sur les ressources. En revanche, l'architecture générale et notamment les logiciels qui ne sont pas liés aux ressources ne sont pas dévoilés. Ces plates-formes ne peuvent donc pas être utilisées comme base pour conduire des expériences.

Comme nous l'avons évoqué précédemment, les systèmes de Calcul Global peuvent être considérés comme des extensions à l'échelle d'Internet du principe de vol de cycles. Sur les réseaux locaux, le vol de cycles a déjà été étudié dans les projets tels que Condor [65], Glunix [55] et Mosix [16]. Mais les contextes des réseaux locaux et Internet sont radicalement différents. Les techniques d'ordonnancement [9, 86] doivent aussi être adaptées à un environnement de Calcul Global en raison : 1) du nombre de ressources mis en jeu, 2) de l'extrême volatilité des ressources. Le projet Condor propose une évolution de son système pour les ressources connectées sur Internet. Ainsi Condor/G est une adaptation de Condor par dessus les mécanismes offerts par Globus.

2.4 Les grilles pour la physique des hautes énergies

Après avoir présenté les différents travaux sur les grilles, nous allons plus particulièrement nous intéresser aux grilles mises en place pour la physique des hautes énergies *(HEP)*.

2.4.1 La grille pour le LHC

Contrairement aux grilles de recherche, les grilles de production ont pour premier objectif de rendre les mêmes services que les centres de calcul traditionnels. Le Centre Européen de la Recherche Nucléaire *(CERN)* est activement engagé dans des grands projets de grille afin de répondre aux besoins informatiques des expériences HEP. Le projet DATAGRID a permis de développer de nouvelles technologies et de mettre en évidence les capacités des grilles. La mise en place de la grille LCG [2], destinée à répondre aux besoins informatiques de l'expérience LHC (*Large Hadron Collider*), est consitutée de 450 participants répartis dans 47 laboratoires internationaux. Les besoins des expériences du LHC (*ALice, Atlas, LHCb et CMS*) sont estimés à deux péta-octets de stockage et sept milles unités de calcul ; à l'horizon 2010, ces besoins sont estimés à onze péta-octets et dix huit milles unités de calcul.

Comme toute expérience de physique, les besoins informatiques de LHC sont

[2]http ://lwg.web.cern.ch/LCG

énormes et concernent la simulation et le traitement des données [3]. La simulation d'un évènement entre typiquement dans la famille des applications multi-paramètriques, séquentielles et non communicantes ; elle demande plusieurs heures de calcul et génère entre cinq cents méga-octets et un giga-octets de résultats. Le traitement des données entre dans la même famille d'application ; il traite les données acquises ou simulées afin d'obtenir un histogramme de un méga-octets.

Les données acquises sont distribuées sur plusieurs centres participants au LHC et membre de LCG. Les centres sont structurés hiérarchiquement sur quatre niveaux (ou *« Tiers »*) en fonction de leurs capacités de stockage et de calcul. La figure 2.9 [4] schématise cette hiérarchie.

Seul le CERN est Tiers-0. C'est la ressource mondial qui héberge les données acquises et les distribue aux Tiers-1 à travers des connections à 10Gbps. Les Tiers-1 sont des ressources nationales (bien qu'ils puissent servir plus d'une nation). Les Tiers-2 sont des ressources régionales. Ils sont plus directement destinés aux calculs et accédent aux données stockées au niveau des Tiers-1. Les Tiers-3 et Tiers-4 sont des ressources institutionnelles, locales aux laboratoires participants.

2.4.2 DIRAC : les concepts de calcul global appliqués à la grille

Il existe plusieurs solutions de grilles de calcul adaptées spécifiquement aux expériences du LHC, comme pour l'expérience *Atlas* [8]. Ces solutions sont qualifiées de grille « légères » car spécifiques à une seule expérience et en surcouche d'infrastructure plus lourdes comme LCG, GRID3 ou Nordugrid.

Parmi ces solutions, le système DIRAC a pour but de fournir une interface pérenne et compatible masquant cette dynamicité et hétérogénéité. Le projet DIRAC [104] est coordonné par le Centre de Physique de Marseille (CPPM-IN2P3) ; il est utilisé en production depuis mai 2004 par l'expérience LHCb.

DIRAC est une solution visant à résoudre les besoins applicatifs d'une communauté (par exemple *LHCb*), en agrégeant toutes les ressources disponibles pour cette même communauté. Ces ressources sont des systèmes de traitements par lot, de simples PCs ou même des systèmes de grille. DIRAC agrège ces ressources par une surcouche qui les uniformise et permet un niveau supplémentaire d'abstraction. Ces ressources s'insèrent ensuite dans l'organisation DIRAC, basée sur la notion de Services/Agent. Les éléments de contrôle, appelés « agent », correspondent à une ressource. Ces agents sont indépendants, multifonctionnels et actifs. Ils ont par exemple la charge de la gestion des ressources afin de pouvoir les utiliser selon leurs disponibilités. En cas de disponibilité, les agents demandent du travail à des services de files d'attente globales.

Cet exemple illustre l'approche innovante de DIRAC qui applique le paradigme de calcul global dans un contexte de grilles institutionnelles.

[3] Nous ne parlerons pas ici de l'informatique *online*.

[4] Cette figure est tirée de la thèse de Vincent Garonne

Toutefois, nous notons que cette solution reste dans un domaine institutionnel et n'intègre aucune ressource personnelle. Ce climat de confiance permet de sous-traiter les questions de sécurité avec les infrastructures sous-jacentes en respectant les contraintes de compatibilité comme l'authentification et l'autorisation. En outre, DIRAC n'offre aucun mécanisme de confinement de l'exécution et s'appuit directement sur les systèmes de traitements par lot disponibles pour la consommation de temps processeurs et mémoire. De plus, les ressources institutionnelles sont plus stables et moins volatiles que des ressources personnelles.

2.4.3 L'implication du Laboratoire de l'Accélérateur Linéaire

Le Laboratoire de l'Accélérateur Linéaire (*LAL*) est actif dans trois projets de grille :

- XtremWeb, à travers les travaux décrits dans ce présent manuscrit qui sont le résultats d'une collaboration LAL/LRI ;
- *Enabling Grid for E-sciencE (EGEE)* est un projet financé par la communauté européenne qui regroupe quatre vingt dix institutions parmi trente deux pays. Il propose de fournir une grille pluridisciplinaire aux scientifiques. Le LAL est Tiers-2 pour cette grille en hébergeant un nœud de quarante processeurs et trois téra-octets de stockage. Cette puissance est appelée à monter jusqu'à mille processeurs et cent téra-octets de stockage. Le LAL intervient aussi au niveaux des développements logiciels et de la réflexion sur l'architecture de cette plate-forme ;
- le LAL, en collaboration avec le DAPNIA et le LPNHE propose la ressource EGEE à la grille LCG (LCG est une des composante EGEE) dans le cadre du projet « Grille en Région Ile de France » (*GRIF*) dont l'objectif est de fournir une ressource globale de plus de mille cinq cents processeurs et trois cent cinquante téra-octets de stockage, répartis entre Paris, Saclay et Orsay.

Du fait de la forte implication du LAL et du DAPNIA dans le projet EGEE, dont ils constituent déjà des centres de ressources opérationnels, les ressources mises en place dans le cadre du Tier-2 resteront ouvertes aux autres communautés d'utilisateurs EGEE (VO) dans des proportions qui seront définies par le comité de pilotage. Il est prévu que vingt pour cent des ressources seront accessibles aux besoins locaux et aux autres utilisateurs EGEE. Les ressources du LAL et du DAPNIA seront unifiées vis-à-vis de la grille EGEE, une fois la partie LCG mise en place.

2.5 Conclusion

Dans ce chapitre nous avons vu que les grilles sont aujourd'hui le sujet d'importants travaux de recherche de la communauté internationale. Nous nous sommes intéressés aux différentes propositions et à l'évolution des solutions mises en œuvre. Notre tour d'horizon a montré que deux types de modèles prévalaient : les grilles institutionnelles et les plates-formes pair à pair. Nous avons pu constater que le but à atteindre est bien le même, agréger des ressources distribuées. Si leurs choix technologiques peuvent paraître orthogonaux, c'est que les problèmes à résoudre sont bien différents, à cause des caractéristiques mêmes de ces deux types d'environnement partagé.

Il existe toutefois des solutions hybrides et génériques, comme DIRAC, qui savent tirer profit des avantages des deux types de plates-formes.

Nous avons été amenés a développer notre propre plate-forme de calcul global car il n'existe pas de solution ouverte proposant d'utiliser des ressources indépendantes et volontaires, implémentant ses propres mécanismes de sécurité et de tolérance aux pannes, qui nous aurait permis de mener à bien les travaux de cette thèse.

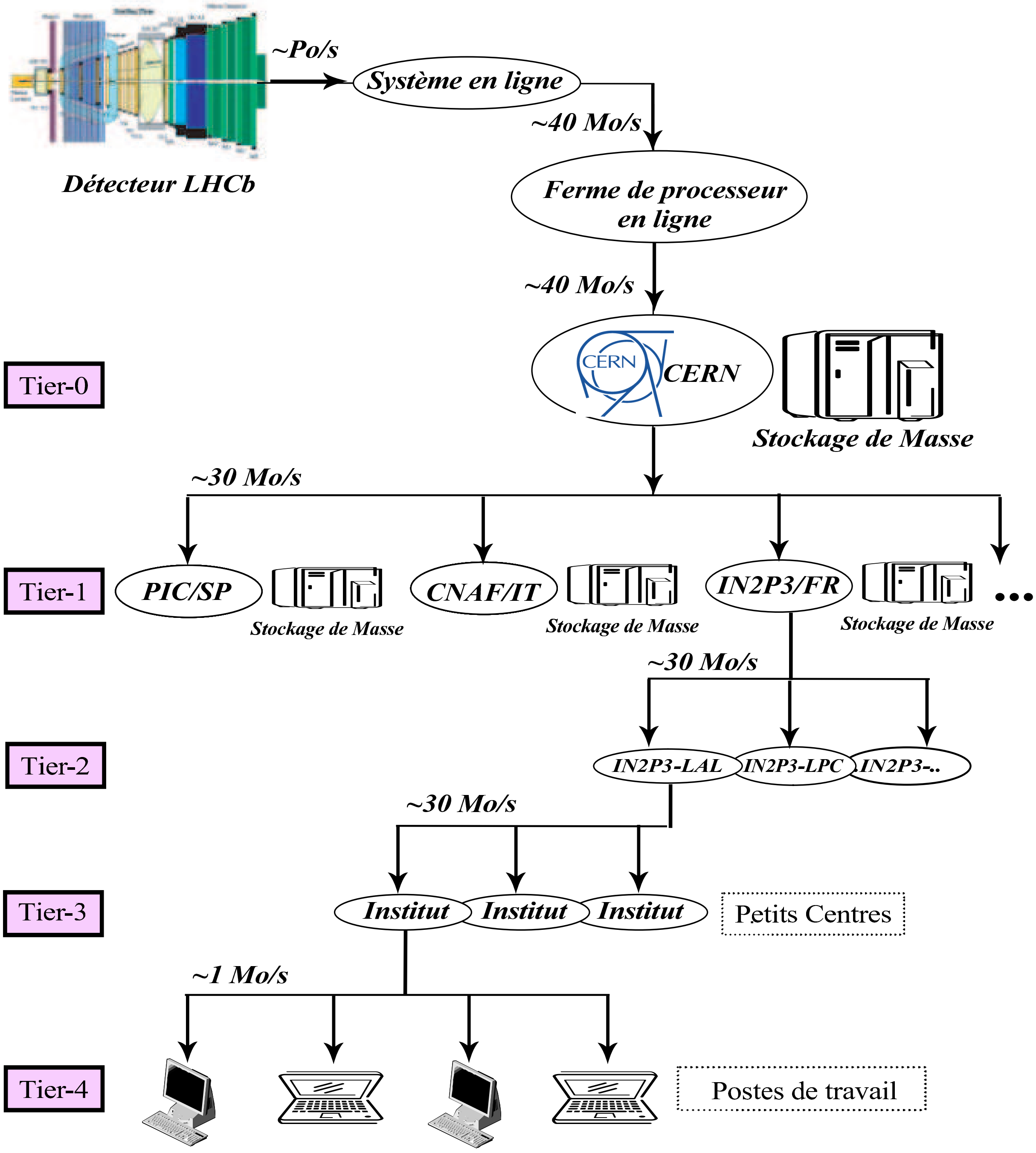

FIG. 2.9 – L'architecture du LCG.

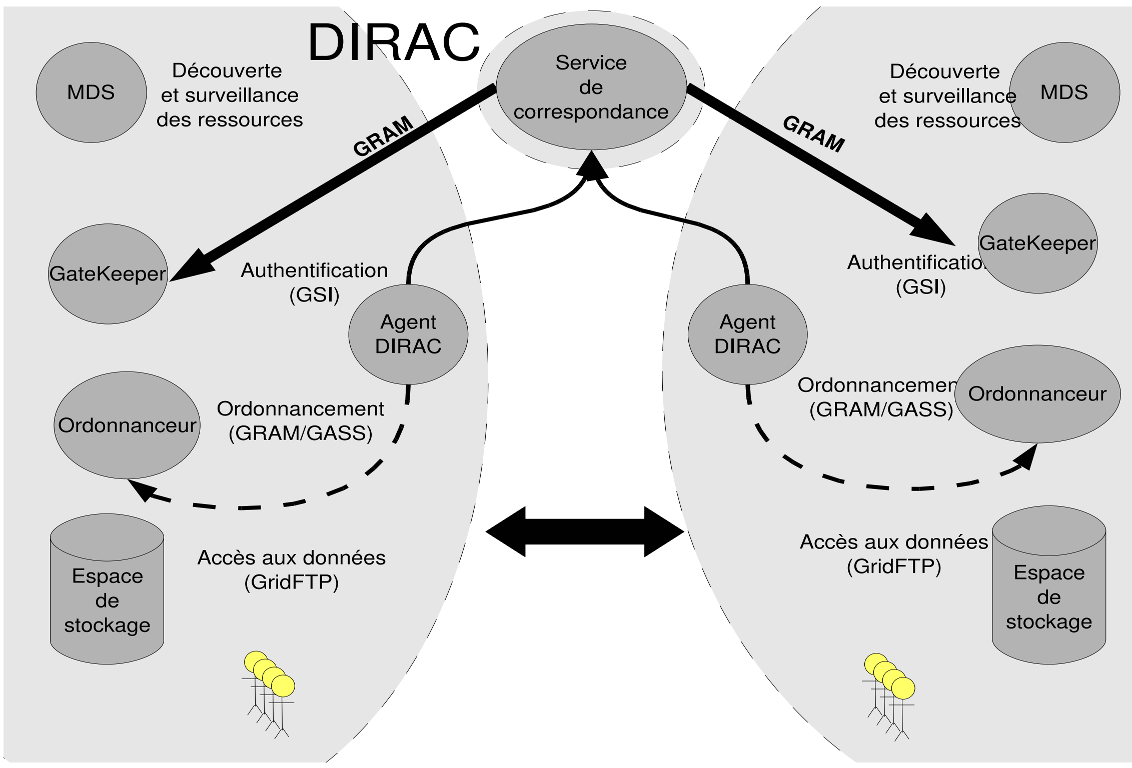

FIG. 2.10 – DIRAC : un modèle de partage de ressources multi-grilles.

Chapitre 3

XtremWeb

Résumé. Dans ce chapitre, nous présentons XtremWeb, une plate-forme de calcul global multi-utilisateurs, multi-applications permettant d'exécuter des applications multi-paramétriques sur des ressources participantes. Nous verrons qu'XtremWeb propose des solutions innovantes afin de répondre aux contraintes des systèmes distribués à large échelle (*LSDS*) telles que la volatilité, l'hétérogénéité et la sécurité et que cette plate-forme permet l'exécution de services basés sur des protocoles standards. Nous décrirons son architecture, basée sur les services afin de faciliter son implémentation et son déploiement, tout en restant raisonnable en terme de complexité.

Nous introduisons tout d'abord la plate-forme telle qu'elle existait avant cette thèse et explicitons les changements nécessaires en terme d'architecture et de services afin d'aboutir à la plate-forme nécessaire à la réalisation de nos travaux :interconnecter différentes plates-formes de calcul, distribuer des protocoles et services standards, ainsi que la mise en production pour la physique des hautes énergies. Nous justifions ces nécessaires changements par des bases théoriques.

Ce chapitre conclut par l'intérêt porté à XtremWeb par différentes équipes de recherche, en présentant sommairement quelques travaux effectués autour de cette plate-forme dans le cadre de différents projets de recherche, tels que la communications inter-nœuds ou le stockage distribué.

3.1 Introduction

XtremWeb permet de construire un environnement d'exécution pour services et applications sur un ensemble non spécifique de ressources distribuées et volatiles grâce à ses mécanismes de tolérances aux pannes. XtremWeb est un projet né de la volonté d'étudier les plates-formes à grande échelle (*LSDS*) et les opportunités d'utiliser de telles plates-formes de la même manière qu'un cluster « standard » en proposant des interfaces d'utilisation, de maintenance et de programmation. XtremWeb est une plate-forme de recherche et de production de la famille des *grilles* et tend à suivre l'effort de standardisation des services grilles [48].

XtremWeb est une plate-forme de calcul global permettant d'utiliser des ressources de calcul distribuées ; cette plate-forme repose sur une architecture trois-tiers, basée sur trois services : le service client qui est l'interface avec l'utilisateur, le service de coordination qui assure l'intégrité de la plate-forme, et le service de calcul qui est utilisé pour effectuer les calculs proprement dits. La figure 3.1 schématise la plate-forme.

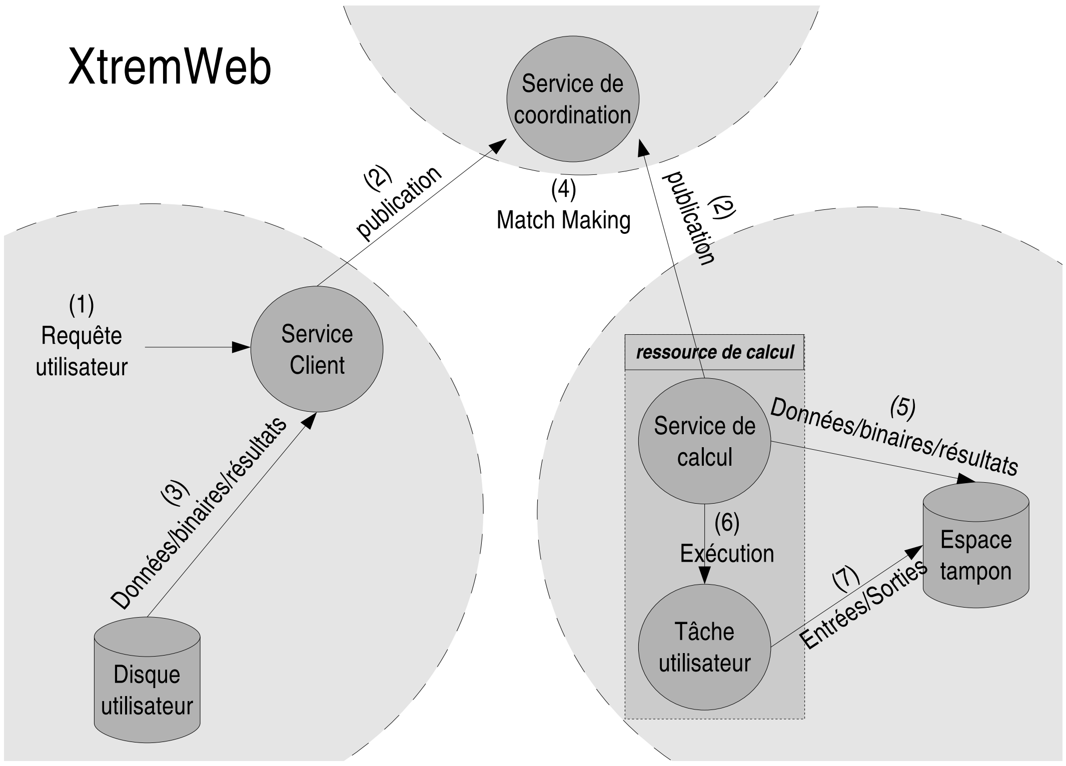

FIG. 3.1 – L'architecture trois tiers d'XtremWeb.

3.1.1 L'environnement de développement.

Le langage *Java* est le langage de développement choisi pour mettre en œuvre le projet XtremWeb. Plusieurs considérations ont été prises en compte dans le choix de ce langage de programmation : sa portabilité, ses bibliothèques implémentant les différents protocoles réseaux, sa sécurité et ses caractéristiques relatives à l'ingénierie logiciel. Nous explicitons rapidement ici ces différents points.

Portabilité. Au contraire des langages *natifs*, Java est un langage indépendant de toute infrastructure matérielle car il est *semi-interprété* et peut être copié et exécuté sur n'importe quel matériel sans aucune modification, grâce à sa *machine virtuelle (la JVM)* qui exécute le code Java. La JVM a été portée sur toutes les plates-formes par les concepteurs de langage eux mêmes (Java est un produit de

Sun Inc.). Les développeurs écrivant des programmes avec le langage Java peuvent donc très justement considérer que leur programme est compatible avec toutes les plates-formes. Le langage assure, entre autre, la validité des types de base quel que soit l'environnement d'exécution (les entiers font toujours 32 bits etc.)

Cette portabilité est bien sûr essentielle pour un projet de grille comme XtremWeb, puisqu'il est impossible de connaître *à priori* les environnements d'exécution qui seront fournis à la plate-forme.

Capacités réseaux. Pour des environnements distribués comme les grilles, les mécanismes de communications sont vitaux. Ils doivent offrir un haut niveau d'abstraction avec des fonctionnalités indépendantes des protocoles et des matériels. Java fournit les librairies permettant cette abstraction. Ce langage propose aussi un protocole d'accès de méthode à distance (*RPC*) appelé *RMI* qui permet d'invoquer des méthodes d'objets distants. Nous verrons toutefois, au chapitre **??**, que ce protocole ne convient pas à une plate-forme de calcul global, comme XtremWeb.

Sécurité. La sécurité est un point majeur des plates-formes distribuées. Les ressources exécutant du code distribué doivent pouvoir contrôler ce dernier, en surveillant (et même limitant) son exécution, et en vérifiant son origine. La JVM possède les mécanismes nécessaires à la surveillance des applications Java elles mêmes ; grâce à la gestion de politiques de sécurité, elle peut *confiner* (restreindre) l'exécution des applications afin de protéger la machine hôte sur laquelle l'application s'exécute. Mais les applications Java peuvent aussi être *signées* afin de certifier leur origine ; selon sa politique de sécurité, la JVM peut autoriser l'exécution sans restriction d'applications signées.

Notons toutefois que cette sécurité assurée par la JVM ne concerne que les applications Java. La plate-forme XtremWeb, qui est capable de distribuer des codes natifs, doit donc aussi assurer la sécurité des ressources face à la menace potentielle représentée par ces codes natifs. Ce point est étudié au chapitre 3.2.2.

Ingénierie logiciel. Java est un langage *orienté objet*. La programmation orientée objet facilite la structuration des développements logiciels et la réutilisation de code, réduisant d'autant les temps de développement et de mise au point. Le langage a de nombreuses caractéristiques permettant d'éviter de nombreux problèmes de développement. Un typage fort de données ; une arithmétique des pointeurs inexistante ; une vérification en temps réel des accès aux tableaux de données ; une gestion des erreurs par exception ; une gestion dynamique de la mémoire (un ramasse miettes).Le ramasse miettes, tout particulièrement, évite un grand nombre d'écueil de gestion de mémoire que l'on trouve avec des langages tels que le *C* ou le *C++*.

Intégration de service. Le langage Java intègre les mécanismes d'auto-description et d'introspection qui sont particulièrement adéquats à l'implémentation des services et à leur gestion dynamique. On peut ainsi charger dynamiquement et à tout moment de nouveaux services, décrire et découvrir dynamiquement leurs interfaces, gérer leurs cycles de vie. On peut aussi les décharger dès qu'ils ne sont plus utiles.

Ces mécanismes sont particulièrement importants et nécessaires au développement d'une plate-forme basée sur les services.

Déploiement. Java inclut une gestion de code par paquets facilitant leurs déploiements, quelle que soit leur complexité. Grâce à cette gestion par paquets, les codes sont réutilisables et surtout *versionnés*, alors qu'avec d'autres langages, les déploiements et les gestions de versions sont souvent très complexes à résoudre. Enfin, les paquets sont auto descriptifs (Cf paragraphe précédent) et peuvent être signés et certifiés par des autorités de certification, augmentant ainsi le niveau de sécurité quant au déploiement et à l'exécution de ces paquets.

Coût. En conclusion quant au choix du langage de programmation, nous pouvons dire que le langage Java réduit les coûts de développement : portage, ingénierie, mise au point, déploiement et sécurité sont facilités par le langage et l'environnement d'exécution Java, ce qui réduit considérablement les efforts à fournir pour ces points précis.

3.2 Analyse d'XtremWeb 1.1.0

Les travaux menés dans le cadre de cette thèse, commencée en 2001, ont été dirigés par la volonté de proposer des solutions innovantes pour la plate-forme XtremWeb afin de la rendre plus fonctionnelle, plus sécurisée et plus générique. Ces travaux ont été dirigés dans le but de mettre en application les notions théoriques décrivant des environnements distribués tel que l'Internet mais aussi par la volonté de proposer un environnement répondant aux besoins concrets des utilisateurs industriels et institutionnels.

La base des travaux de cette thèse était XtremWeb 1.1.0 [41]. Le résultat de cette thèse est XtremWeb 1.5.0 que nous détaillons au chapitre 3.3.

Nous présentons les bases théoriques justifiant une réorganisation du projet XtremWeb. La figure 3.2 synthétise les événements marquants de ces quatre années de travail.

L'année 2002 a été mise à contribution afin de finaliser la première version d'XtremWeb. Un service de monitoring permet de surveiller les différentes parties de la plate-forme, les éléments distribués aussi bien que le service de coordination. Un service de collection de traces d'activité, embarqué avec le service de calcul, envoie des statistiques d'activité de la ressource de calcul au service de coordination afin que celui-ci puisse améliorer la coordination des tâches selon une étude probabiliste de ces statistiques. La configuration du service de calcul peut toutefois empêcher la collection de ces statistiques si le propriétaire de la ressource ne souhaite pas que de telles informations soient collectées. Un système de gestion des versions a été mis au point pour assurer une meilleure cohérence de la plate-forme. Les parties distribuées de la plate-forme envoient leur numéro de version à leur connexion au service de coordination ; ce dernier peut alors leur demander de se mettre à jour si une version plus récente est disponible.

L'année 2003 a été la période d'amélioration de la qualité de service de la plate-forme avec la mise en œuvre de la réplication des coordinateurs assurant le service de coordination permettant une abstraction plus élevé de ce service. La qualité de service a aussi été améliorée grâce à l'enregistrement des messages envoyés par chaque composant de la plate-forme afin de pouvoir se synchroniser en cas de faute de l'un d'eux. Ces mécanismes ont été détaillés au chapitre 3.3.3.

L'année 2004 a principalement été une année de déploiements et de tests. Un test de un million de tâches a pu être mené (figure 3.3) avec la version 1.2.7.

Ce test de grande ampleur a permis de stabiliser la plate-forme et d'affiner de nombreux points. Il a été mis en évidence, par exemple, que le serveur de résultat devait mettre en œuvre un système de gestion de fichiers permettant de dépasser

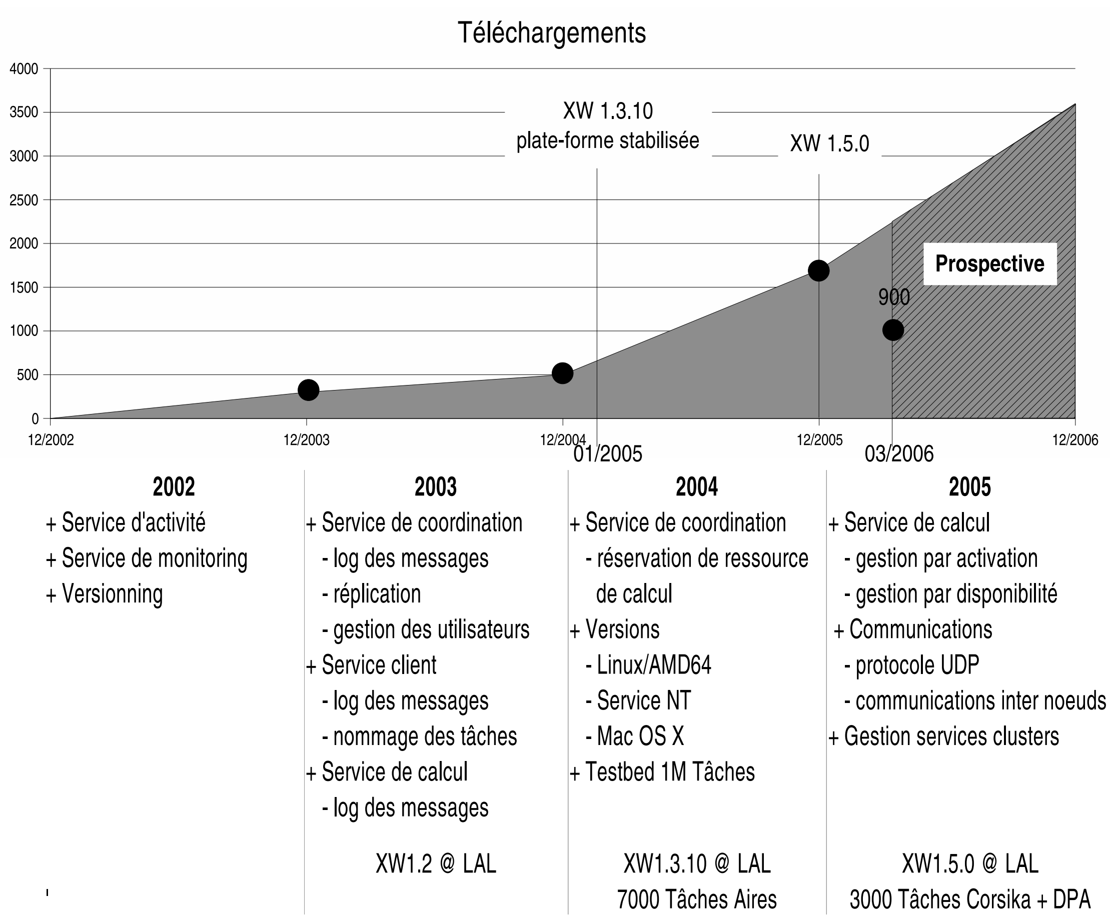

FIG. 3.2 – Evolution d'XtremWeb

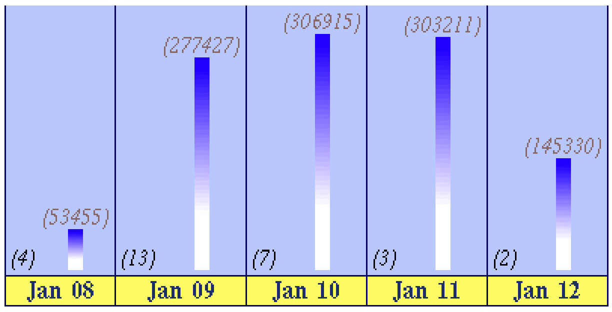

FIG. 3.3 – Test du million de tâches

certaines limites des systèmes d'exploitation comme *Linux*, qui n'autorise que trente deux milles fichiers par répertoire et il donc convient d'organiser la plate-forme afin qu'elle ne soit pas bloquée par de telles limitations.
Dans le même temps cette version était installée au Laboratoire de l'Accélérateur Linéaire où sept mille gerbes à très haute énergie ont pu être simulées en distribuant des tâches pour le programme de simulation de gerbes *Aires* [96] de septembre à décembre sur une douzaine de machines installées au LAL et à l'Institut de Physique Nucléaire d'Orsay (*IPNO*). Cette mise en application à permis de mieux comprendre les attentes des utilisateurs et de résoudre des problèmes qui existaient encore.

Ces travaux de tests et de déploiement nous ont menés à de nombreux changements et amélioration qui ont finalement été intégrés et publiés dans la version 1.3.10 qui a été considérée comme la première version stabilisée d'XtremWeb. Ce travail a été agréablement récompensé par une montée en puissance des téléchargements de XtremWeb (figure 3.2), ce qui montre l'intérêt portée à ces travaux, d'une part, mais aussi l'attente générée par ce type de solution de calcul global.
Les téléchargements n'ont plus démenti ces espoirs depuis cette version 1.3.10, première version stable de la plate-forme. A partir de cette version, la plate-forme a rempli son cahier des charges : effectuer des calculs sur des ressources distribuées et hautement volatiles. Les versions suivantes ont depuis proposé bien des avancées, que ce soit en termes d'utilisation ou de nouvelles fonctionnalités ; mais il n'a dès lors plus été nécessaire de corriger d'erreurs relatives à ce cahier des charges.

L'année 2005. Les travaux de cette année ont permis d'étudier la mise en œuvre de services clusters, tels que *NFS*, sur une plate-forme de calcul global. Les résultats de ces travaux sont présentés au chapitre 5.
Pour arriver à de tels résultats, il a fallu augmenter la visibilité des services de calcul par le service de coordination. C'est à cette fin qu'ont été introduites les notions d'activation et de disponibilité. Ces deux notions permettent d'améliorer les études probabilistes relatives à l'ordonnancement. Les services de calcul peuvent être *actifs* ou non ; cette caractéristique est gérée par le service de coordination qui déterminent les services qui sont actifs et ceux qui ne le sont pas. Aucune tâche n'est ordonnancée sur un service de calcul non actif. Un service de calcul est désactivé s'il génère une faute de calcul ; par exemple, s'il n'a pas assez d'espace disque pour

exécuter une tâche, ou tout autre erreur système. Les ressources de calcul peuvent aussi être *disponibles* ou non ; cette caractéristique est déterminée par la politique locale d'activation du service. C'est une information qui est rapportée au service de coordination afin d'améliorer l'ordonnancement des tâches.

3.2.1 Architecture

L'architecture d'XtremWeb en couches tend à augmenter la stabilité de la gestion des tâches dans un environnement toujours plus complexe, distribué et volatile. La figure 3.4 schématise l'architecture de la version 1.1.0.

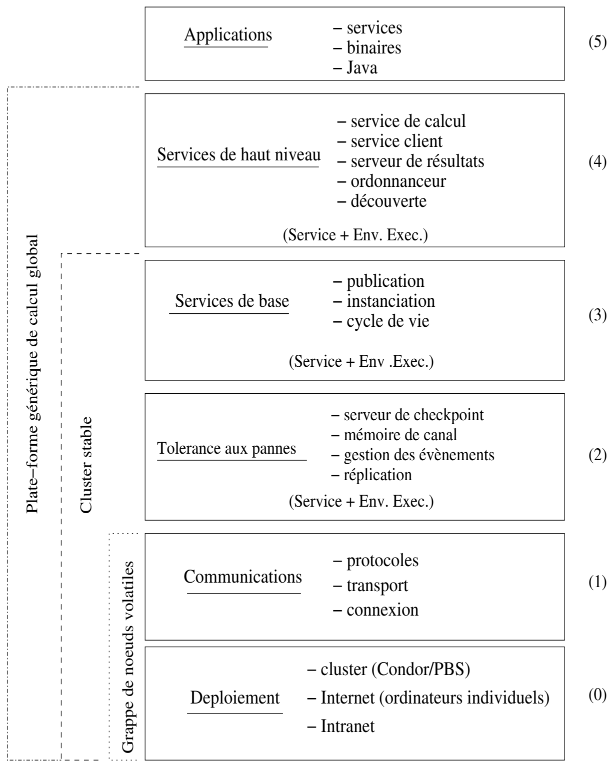

Fig. 3.4 – Architecture d'XtremWeb 1.1.0.

L'architecture est composée d'un total de trois couches, elles mêmes divisées en un total de six sous couches. Le rôle de la première (notée 0) est de permettre le déploiement des services d'XtremWeb. On retrouve ce type de mécanisme dans d'autres environnements comme Jxta. Le chapitre 3.3.7 détaille cette première couche. La sous couche (1) contient les mécanismes nécessaires à la communication inter nœuds. La sous couche (2) comprend les mécanismes et les environnements d'exécution dédiés à la tolérance aux pannes. Ceux-ci dépendent du type de plate-forme de calcul global souhaité ainsi que du type de déploiement utilisé. Les services de réplication, de journalisation des messages ainsi que de détection de faute sont dans cette sous couche afin d'implémenter les protocoles de tolérance aux pannes. La sous couche (3) est composée des dispositifs de publication, de découverte et de construction des services de la plate-forme. La sous couche (4) contient les services de haut niveau tels que le serveur de tâches, l'ordonnanceur, le serveur de résultats, le service de calcul servant de contexte d'exécution aux tâches (le *worker*), le service client permettant aux utilisateurs d'accéder à la plate-forme. La sous couche (5) est la couche applicative ; c'est à ce niveau que les utilisateurs peuvent intervenir.

3.2.2 La sécurité.

Il est particulièrement difficile d'assurer la sécurité dans un contexte LSDS dans la mesure où il est impossible de faire confiance aux milliers de ressources partagées. Il convient en tout premier lieu d'assurer la sécurité des ressources elles mêmes. Aucune application agressive ne doit être en mesure de corrompre ni les données ni les ressources. Ce point est rendu délicat par la distribution de code binaire. Un autre problème est le manque d'outils permettant la certification des résultats. Il est impossible de vérifier la bonne exécution des binaires, puisqu'on ne contrôle pas les participants qui peuvent non seulement générer des erreurs de calculs, mais aussi détourner sciemment les calculs eux mêmes.

Les problèmes de sécurité sur une plate-forme GC peuvent être divisés en quatre parties : l'intégrité des données ; la certification des résultats ; l'intégrité de l'infrastructure elle même ; et enfin, l'intégrité des ressources participantes. Plusieurs travaux ont déjà traité des trois premiers points : [30], [89]. Nous discutons ici le quatrième.

La plate-forme doit assurer la sécurité des ressources participantes contre toute intrusion et doit s'assurer de leur intégrité afin de leur éviter toute altération matérielle aussi bien que logicielle, ainsi que toute tentative de détournement ou d'espionnage. La première condition à vérifier pour sécuriser l'exécution de code partagé est de ne faire exécuter ces codes qu'avec des droits non privilégiés afin d'éviter qu'ils ne puissent accéder à des informations sensibles. Ceci suppose que la plate-forme elle même n'ait aucun droit privilégié. Dans XtremWeb, il convient donc de porter une attention toute particulière aux droits fournis au service de calcul car c'est ce service qui exécute les codes distribués.

Une approche pour sécuriser les ressources consiste à confiner l'exécution des codes binaires dans une enveloppe inviolable : le *bac à sable*. C'est une technique bien connue basée sur le filtrage des appels systèmes qui est le premier angle d'attaque de tout code mal intentionné. En fonction d'une politique de sécurité, le bac à sable neutralise les comportements à risque, limite l'utilisation des ressources (disque, réseau etc.) et tente de protéger le système de ses propres « trous de sécurité ». En proposant ces contrôles au cours de l'exécution des codes binaires, ce mécanisme augmente singulièrement la sécurité du système hôte. En cela, il est un point essentiel des LSDS.

Le bac à sable peut être le plus connu est la machine virtuelle Java (JVM) qui interprète le code Java en suivant une politique de sécurité qui filtre les interactions entre le code et le système lui même. Mais la JVM est dédiée au monde Java et n'est d'aucune utilité pour les codes binaires.

Trois types de bac à sable «natif» peuvent être utilisés pour la protection des ressources lors d'exécution de codes binaires. Plusieurs implémentation de ces bacs à sable sont basé sur *Ptrace*, un mécanisme inclut dans tous les systèmes compatible *Posix*. La tâche à surveiller est sous le contrôle d'un moniteur. A chaque fois que la tâche effectue un appel système, le moniteur la bloque afin de vérifier la validité des arguments. Si les arguments utilisés par la tâche pour son appel système semblent corrects (selon la politique de sécurité), l'appel demandé est exécuté et le résultat retourné à la tâche appelante. Dans le cas contraire, la tâche est tout simplement détruite. Le bac à sable le plus connu utilisant Ptrace est «Subterfugue» [72]. Malheureusement, Ptrace souffre lui même d'un trou de sécurité le rendant vulnérable aux attaques de type *race condition*. Une unité d'exécution peut très bien modifier les paramètres validés par le moniteur avant que le système n'ait le temps de lancer l'appel système qui se fera donc avec des paramètres non validés et donc potentiellement dangereux.

On peut combler ces problèmes en émulant les appels systèmes dans une machine virtuelle. De la même manière que la JVM exécute des codes Java en fonction d'une politique de sécurité, il existe des machines virtuelles (*VM*) qui simulent des environnements d'exécution complet et on peut donc exécuter des codes binaires dans ses environnements confinés de manière sécurisée pour la ressource de calcul, grâce à des politiques de sécurité. Une VM simule et intercepte éventuellement les appels systèmes de la tâche confinée, vérifie les conditions d'appel (paramètres...) et exécute réellement l'appel système si ces conditions satisfont la politique de sécurité. La VM retourne ensuite les résultats à la tâche confinée, ou un code d'erreur au cas où les conditions de sécurité ne sont pas satisfaites. Cette technique offre un contrôle souple de l'environnement d'exécution. Quelques projets implémentent cette technique. *User Mode Linux (UML)* est un applicatif émulant tous les appels systèmes. *VMWare*, *Bochs* ou encore *coLinux* proposent des VMs dans lesquelles on fait tourner un système d'exploitation (*OS*) complet pour y exécuter les tâches confinées. Ces dernières n'ont donc qu'accès à cet OS simulé et lui seul ; elles n'ont aucun accès à la machine hôte qui tourne la machine virtuelle. Notons que celles-ci peuvent très bien faire tourner des OS différents de celui de la machine hôte (on peut faire tourner un *Linux* virtuel sur une machine *Windows*, par exemple).

Toutefois cette approche ralentie l'exécution des codes à un facteur qui peut même aller jusqu'à 1000. Cette limitation peut être prohibitive pour des plates-formes GC à haute performance.

Une méthode de bac à sable plus récente basée sur le module de sécurité Linux (LSM) propose d'intégrer les mécanismes d'interposition directement dans le noyau du systèmes. LSM est un squelette logiciel (*framework*) qui permet d'insérer du code de sécurité dans les appels systèmes. Cette solution est plus efficace en termes de sécurité parce qu'elle est incluse au noyau lui même. Elle est d'autre part plus efficace en terme de temps d'exécution car elle ne nécessite par de changement de contexte qui est une opération longue. Toutefois cette solution est difficilement déployable, puisqu'elle suppose un noyau Linux spécifique.

La sécurité des ressources dans XtremWeb repose donc sur de tels systèmes de confinement. Il est toutefois notable qu'aucune solution gratuite ne soit disponible pour confiner du code *Windows* ou *Macintosh* ; il conviendra donc aux utilisateurs de ces OS de fournir à la plate-forme les systèmes de confinement choisis. La plate-forme XtremWeb est par contre capable de confiner du code binaire pour *Linux* grâce à des systèmes ouverts, disponibles et gratuits, comme UML. Le service de calcul contient le système UML et peut l'utiliser sur simple configuration de la part du propriétaire de la ressource.

3.2.3 Services

Les grilles sont souvent décrites autour de la notion de service [48], un paradigme largement partagé standardisant les composants des systèmes distribués. Les services ont été présentés au chapitre 2.2.2.

Une plate forme de calcul global agrégeant des ressources qui soumettent des tâches qui sont elles mêmes exécutées sur des ressources distribuées de calcul peut être décrite et implémentée en termes de services : le service de coordination (le *coordinateur*) publie les services applicatifs disponibles et en ordonne des instances sur des ressources de calcul. Le service *client* demande des exécutions (associant des tâches et leurs paramètres) de services applicatifs et récupère les résultats grâce au service de coordination. Le service de calcul (le *worker*) récupère des tâches à exécuter auprès du service de coordination et en lui retourne les résultats. On notera que le service de coordination peut lui même être décrit par un ensemble de sous services, l'ordonnanceur, le serveur de données, l'instanciateur, et peut être implémenté sur une architecture centralisée ou distribuée. Le chapitre 3.3.2 discute plus précisément des services XtremWeb.

Les services entre les versions 1.1.0 et 1.5.0 n'ont pas changé en nombre, mais en qualité. Leur interfaces, leurs spécifications, restent presqu'inchangés ; par contre leurs fonctionnements internes ont évolués.

Leurs interfaces ont changé afin de mettre en œuvre la qualité de service décrite au chapitre 3.3.3. Le service de coordination est maintenant composé d'un ensemble de coordinateurs synchronisés grâce à un protocole suivant le modèle de la réplication passive. Les services de calcul suivent strictement leur politique d'activation avant toute action ; en particulier, ils ne téléchargent aucune tâche avant d'être activés afin d'améliorer l'ordonnancement des tâches et de ne pas utiliser de bande passante inutilement. L'ordonnancement bénéficie d'autre part des notions *d'activité* et de *disponibilité* expliquées plus bas. Ces informations ont été intégrées dans le protocole de communication entre les services de calcul et le service de coordination.

Un autre point important relatif à l'ordonnancement qui n'existait pas dans la version 1.1.0 : le service client peut choisir le service de calcul sur lequel sera exécutée la tâche. Ce point est particulièrement utile à la distribution de services cluster sur XtremWeb ; il est détaillé au chapitre 5. Enfin, le service client implémente désormais toutes les options qui lui sont dévolues ; certaines, dans la version 1.1.0, étaient implémentées du côté du service de coordination, ce qui n'était pas conforme à une mise à grande échelle.

Qualité de service. La qualité de service était insuffisante dans la version 1.1.0 et il a été nécessaire de l'améliorer afin d'obtenir celle de la version 1.5.0 décrite au chapitre 3.3.3. Elle comprend un ensemble de mécanismes permettant d'assurer que des tâches soumises à la plate-forme seront effectivement ordonnancées quels que soient les événements et erreurs qui peuvent intervenir, principalement dues à la grande volatilité des ressources mises en œuvre.

La journalisation de tous les messages envoyés par les émetteurs eux mêmes ; la réplication des ressources utilisées pour le service de coordination ; la migration des services, aussi bien du service client que des services de calcul ; le protocole de communication multi-phases ; l'ordonnancement sur les ressources de calcul disponibles et elles seules. Tous ces mécanismes ont pu être implémentés et mis au point grâce à de bonnes bases théoriques, mais aussi grâce à des tests et des déploiements de mise en production qui ont permis d'être en relation avec les utilisateurs et de rester à leur écoute.

3.2.4 Communications

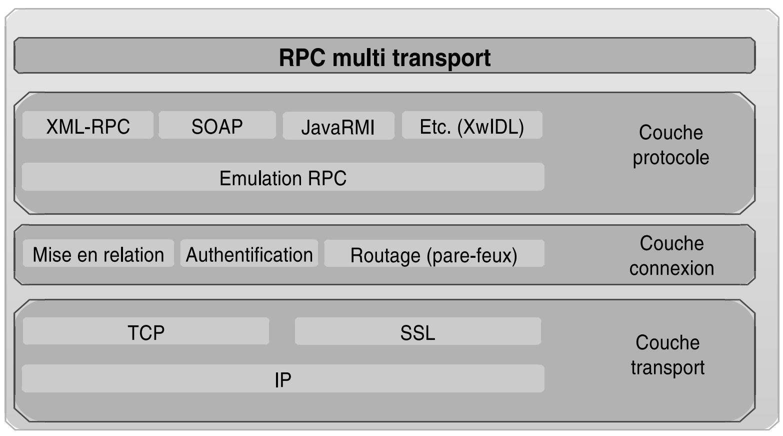

Fig. 3.5 – La couche communication dans XtremWeb 1.1.0.

Les communications ont été ré-architecturées afin d'augmenter leurs niveaux d'abstraction et de standardisation, d'une part, et de répondre aux besoins des services de type cluster, d'autre part. Les figures 3.5 et 3.11 permettent de comparer les couches communications des versions 1.1.0 et 1.5.0 respectivement. Plusieurs points ont notablement changés. Les protocoles de sécurisation ne sont plus dans la couche transport, mais dans la couche message pour une meilleure conceptualisation, comme décrit au chapitre 3.3.4. En effet, l'utilisation de la librairie *SSL* implique des communications point à point, ce qui n'est pas applicable dans le cadre des plates-formes de calcul global où un ou plusieurs nœuds peuvent intervenir dans le transport de messages et des données ; il convient donc de conceptualiser la sécurité des communications au niveau message, ce qui a été fait dans la version 1.5.0.

La couche « transport » de la version actuelle implémente d'autre part les communications basées sur le protocole *UDP*. Cette modification a été nécessaire afin de pouvoir transporter des services de type cluster sur cette plate-forme de calcul global ; ceci est détaillé au chapitre 5.

La couche « protocole » de la dernière version n'implémente plus le protocole *JavaRMI*. Les tests menés dans le courant de l'année 2004 (figure 3.3) ont en effet montré les limitations de ce protocole empêchant une mise à l'échelle de la plate-forme par le nombre de processus nécessaires au niveau des ressources du service de coordinations. Ces problèmes de mise à l'échelle ne sont toutefois pas limités au seul protocole RMI, les protocoles dits *bloquants* posent tous le même problème de surcharge des ressources. Les différents protocoles implémentés dans la plate-forme sont donc maintenant *non bloquants*. Cette avancée est possible grâce à la librairie *Java NIO (New Input/Output)*. Il n'existe toutefois pas de mode non bloquant pour le protocole RMI.

Les communications de la version actuelle d'XtremWeb ont été étendues à tous les nœuds de la plate-forme (figure 3.12), ce qui n'existait pas dans la version ini-

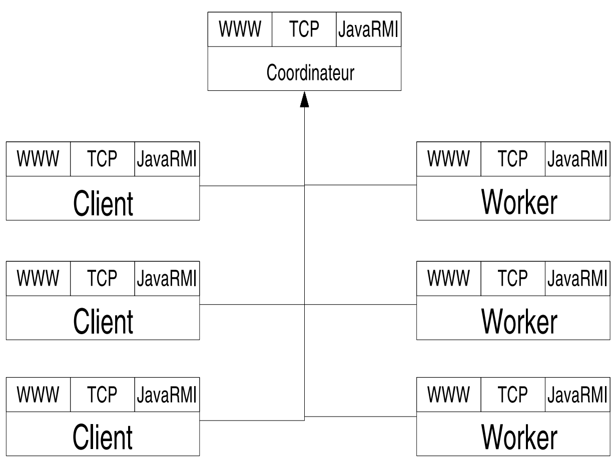

FIG. 3.6 – Les communications dans XtremWeb 1.1.0.

tiale (figure 3.6). La plate-forme contient désormais les protocoles de mise en relation entre les nœuds de la plate-forme, quels qu'ils soient, client, ressource de calcul et service de coordination. La couche communication ne contient toutefois pas de couche message pour les communications entre services client, entre services de calcul, ni entre service client et service de calcul. Mais grâce à un framework basé sur la notion de *handler* [54] qu'il convient d'implémenter, la plate-forme peut facilement être spécialisée afin de mettre en œuvre une couche message adéquate.

3.2.5 Protocole

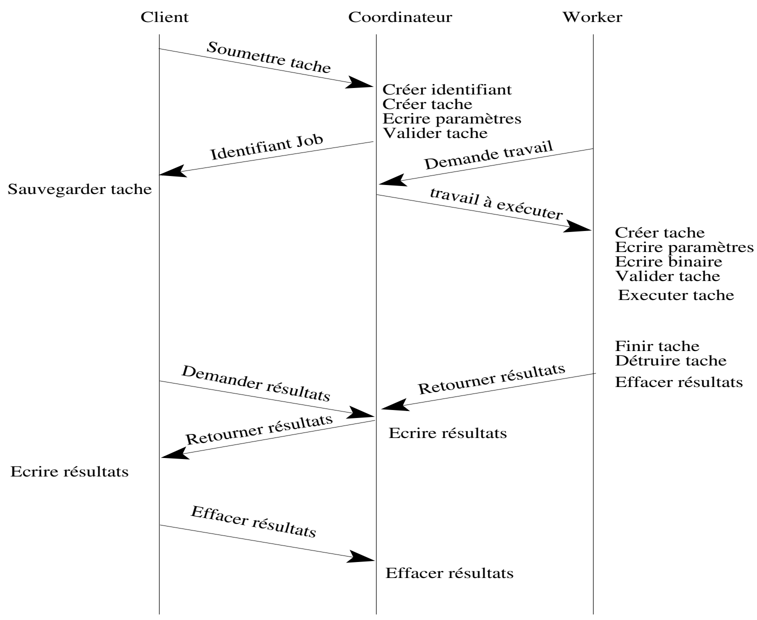

FIG. 3.7 – Le protocole mono-phase de la version 1.1.0.

Dans le cadre de cette thèse, le protocole de communication d'XtremWeb a du être repensé afin d'aboutir à celui présenté au chapitre 3.3.5. Cette ré-architecture a été nécessaire car le protocole de la version 1.1.0 était un protocole *mono-phase*, comme schématisé sur la figure 3.7. La soumission des tâches consistait alors en un message unique contenant les informations de la tâches. Le coordinateur créait un identifiant unique pour cette tâche, stockait la tâche et ses paramètres et retournait au client l'identifiant unique ainsi créé. Ce protocole a posé beaucoup de problèmes de stabilité au niveau de la plate-forme et pouvait générer des tâches *orphelines* consommant des ressources inutilement dans la mesure où il n'y avait plus aucun moyen de contrôler l'exécution de la tâche. Des tâches pouvaient s'avérer orphelines à cause de problèmes réseaux entre le client et le coordinateur empêchant le client d'obtenir l'identifiant de la tâche après que celle ci eût été correctement créée sur le coordinateur. Dans ce cas la tâche était ordonnancée et exécutée inutilement puisque le client ne pouvait plus ni la gérer ni en récupérer les résultats.

D'autres cas d'erreur générait des tâches *perdues* en ce sens qu'il pouvait s'avérer impossible de retourner les résultats au client, car le chemin de retour des résultats n'étaient pas garanti. En effet, à la fin de l'exécution d'une tâche, le service de calcul envoyait les résultats au coordinateur et en effaçait immédiatement sa copie locale, sans être sûr que le coordinateur les ait bien sauvegardés de son côté. En cas de problème du côté du coordinateur, les résultats étaient tout simplement perdus.

Pour finir, les tâches étaient relativement lourdes à envoyer aux services de calcul car elles venaient avec leur paramètres et le binaire de l'application, si le service de calcul ne l'avait pas localement. En cas d'erreur réseau, il fallait alors tout renvoyer, ce qui pouvait gâcher de la bande passante.

Le protocole a été repensé pour la version 1.5.0 est a été ré-implémenté suivant un schéma multi-phases afin d'éviter les tâches orphelines et perdues. Ceci est détaillé au chapitre 3.3.5.

3.2.6 Résultats d'analyse

La meilleure compréhension des bases théoriques décrivant les environnements distribués va nous permettre de réaliser une plate-forme stable et intéressant non seulement différentes équipes de recherche, mais aussi les utilisateurs en quête de toujours plus de ressources de calcul.

En résumé :

- les services de calcul se conforment à leur politique locale d'activation, permettant un meilleur ordonnancement ;
- le service de coordination est basé sur des mécanismes de réplication passive, optimisant la qualité de service ;
- le suivi et l'enregistrement des messages transitant sur la plate-forme permet à celle ci de rester cohérente tout en autorisant la migration des services ;
- l'ordonnancement mis en œuvre afin de choisir le service de calcul, conjointement aux communications *UDP* permettent l'utilisation de services clusters sur une plate-forme de calcul global (Cf chapitre 5) ;
- l'optimisation du service client ainsi que de la gestion des fichiers transitant sur la plate-forme permettent aujourd'hui une mise à grande échelle.

Le reste de ce chapitre détaille les travaux qui ont été nécessaires afin d'obtenir une plate-forme mettant en œuvre les contraintes théoriques.

3.3 Nouvelle proposition : XtremWeb 1.5.0

Nous présentons ici la plate-forme 1.5.0 issue de l'analyse de la version 1.1.0.

3.3.1 Architecture

La figure 3.8 présente les couches de l'architecture générale d'XtremWeb, qui sont au nombre de quatre, elles mêmes divisées en un total de sept sous couches.

En comparaison à la figure 3.4, on voit qu'une couche supplémentaire apparaît, permettant de mieux prendre en charge les environnements de calcul (*RPC*, *MPI*).

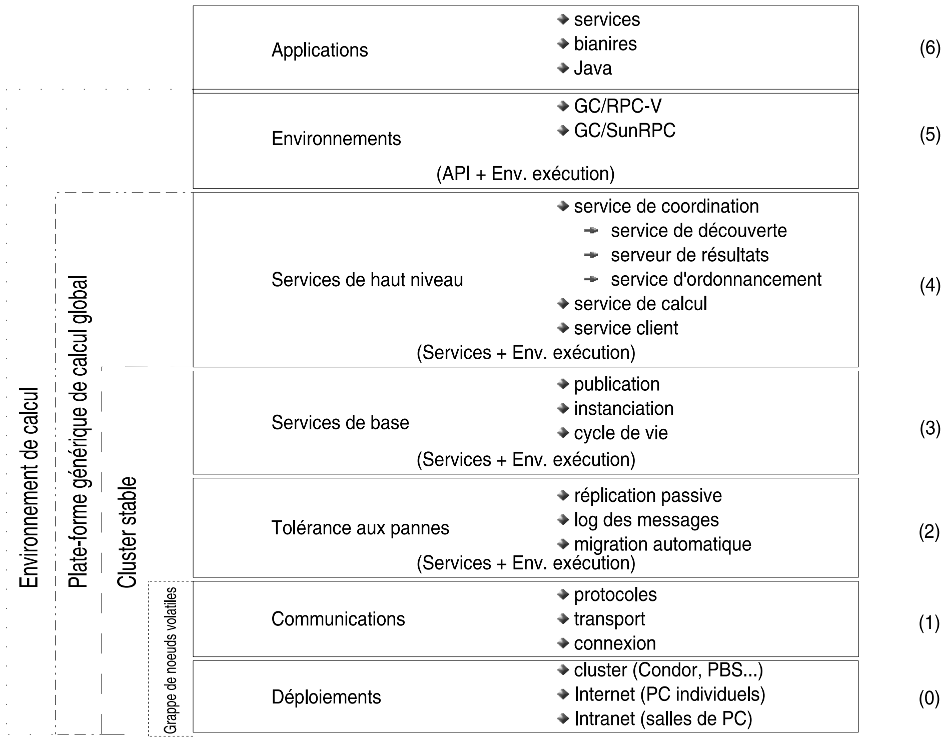

FIG. 3.8 – Architecture d'XtremWeb.

Le rôle de la première couche est de construire un cluster, éventuellement instable, en agrégeant un ensemble de ressources non spécifiques (clusters, PC individuels ou institutionnels etc.). La seconde couche permet d'obtenir un cluster stable à partir des ressources fournies par la couche précédente. Du fait de la volatilité des ressources agrégées, le cluster stable ainsi obtenu peut contenir moins de ressources que potentiellement disponibles (certaines ressources peuvent être gardées comme ressource de «secours»). La troisième couche permet de construire une plate-forme de calcul global générique. La quatrième couche déploie les environnements d'exécution pour les environnement de calcul parallèle tels que les environnements client/serveur. Les applications s'exécutent alors au dessus de cette quatrième couche.

L'architecture est composée des six sous couches vues sur la figure 3.4, avec une septième qui vient s'intercaler entre la couche (4) des services de haut niveau et la couche applicative, pour un total de sept sous couches. Les interfaces de programmation (API) et les environnements d'exécution sont dans la nouvelle sous couche (5).

L'intégration de tout nouvel environnement de programmation (RPC...) devra intégrer ses services de gestion de fautes au niveau de la sous couche (3) et les interfaces de programmation ainsi que les environnements d'exécution dans la sous couche (5).

La version 1.5.0 d'XtremWeb implémente un sous ensemble de cette architecture. En tant que plate-forme de calcul global, XtremWeb permet à des clients d'insérer des binaires afin d'en soumettre des tâches que le service de coordination fera exécuter sur des services de calcul.

Le design d'XtremWeb suit trois grands principes :

1. une architecture trois tiers où le service de coordination connecte des clients à des service de calcul (les *workers*) à travers le service de coordination (les clients et les workers ne sont jamais directement connectés) ;
2. un ensemble de mécanismes de sécurité basé sur des décisions autonomes ;
3. des mécanismes de tolérances aux pannes permettant la mobilité des clients, la volatilité des workers ainsi que des défaillances au niveau du service de coordination.

L'architecture trois tiers insert un médiateur entre le client et le worker qui ne sont donc jamais en liaison directe. Le rôle de ce médiateur, qu'on appelle le service de coordination dans XtremWeb, est de découpler le client du worker, ainsi que de coordonner l'exécution des tâches sur les workers. Le service de coordination reçoit les requêtes des clients, distribue les tâches aux workers en fonction de leur politique d'activation, tient à disposition de ces derniers le code binaire des applications qu'ils peuvent télécharger si nécessaire, coordonne la bonne exécution des tâches, détecte les éventuelles fautes des workers et en tire des conclusions quant à la redistribution des tâches qu'il suspecte d'avoir été interrompue par de telles fautes, collecte les résultats fournis par les workers et les tient à la disposition des clients.

Le service de coordination n'est pas nécessairement implémenté de manière centralisée (même répliqué), il peut être le résultat d'une agrégation sur un ensemble de nœuds implémentant les services nécessaire à la coordination.

L'implémentation d'XtremWeb est le résultat de plusieurs choix découlant d'une volonté de répondre aux contraintes liées aux environnements des utilisateurs ainsi que de la nécessité de progresser vers l'architecture telle que décrite dans le chapitre précédent. En conséquence de quoi, certaines parties sont encore implémentées de manière centralisée. Nous allons ici décrire quelques couches du système présenté par la figure 3.8 : le déploiement (sous couche 0), les communications (sous couche 1), les services de bases (sous couche 2) et les services de calcul global (sous couche 4). Nous concluons cette présentation de l'implémentation d'XtremWeb par la sécurité mise en œuvre dans cette plate-forme.

3.3.2 Les services.

Comme toute plate-forme de calcul global, XtremWeb propose un ensemble de services proposant une fonctionnalité et une fiabilité de la plate-forme afin d'assurer une bonne prise en charge des requêtes des clients. La publication dynamique des ressources ; l'utilisation dynamique de ces ressources à travers une « fabrique » qui permet de les instancier sur les workers ; la gestion du « temps de vie » qui permet de contrôler la terminaison des instances de services (sous couche 3 de la figure 3.8).

L'accès aux services se fait schématiquement en accédant tout d'abord au service de publication qui retourne la liste des services disponibles avec les adresses des «fabriques» permettant de les instancier. Les fabriques tiennent à disposition la liste des ressources où les services sont accessibles. L'instanciation de service dans une plate-forme de calcul global diffère de ce que l'on connaît dans les grilles institutionnelles, dans la mesure où ils sont susceptibles d'être déplacés, de manière transparente, d'une ressource à une autre au cours du temps, afin de résoudre le problème de la tolérance aux pannes. Enfin, un service reste actif jusqu'à ce qu'il détecte une condition de terminaison (s'il n'est plus utilisé ou s'il reçoit un signal de terminaison).

Toutefois, ces services ne suffisent pas à mettre en place une plate-forme de calcul global. D'autres services, de plus haut niveau, sont nécessaires. L'implémentation actuelle comprend trois services principaux : le *service client* permet de soumettre et de gérer des tâches qui sont exécutées par le *le service de calcul (les workers)* grâce à l'interposition du *service de coordination*. Ce dernier implémentant lui même plusieurs services tels que l'ordonnanceur, le répertoire de binaires et le serveur de résultats.

Le service de coordination Le service de coordination dans XtremWeb inclut un ensemble services regroupés sous l'appellation de *coordinateur* que l'on trouve dans les sous couches 2, 3 et 4 de la figure 3.8. Bien qu'il existe des plates-formes implémentant des services complètement distribués, comme c'est le cas, par exemple, dans CAN [83], CHORD [103], PASTRY [87] ou TAPESTRY [108] qui implémente le service de découverte appuyant sur une table de hashage distribuée, nous avons choisi d'implémenter les services de XtremWeb de manière centralisée pour trois raisons :

1. la facilité de développement et de mise au point ;
2. il y a encore peu de résultats concernant des comparaisons en termes de performances entre une architecture centralisée et hiérarchique, d'une part, et une architecture complètement distribuée, d'autre part ;
3. des incertitudes théoriques quant aux architectures distribuées.

Il n'y a pour le moment, en effet, aucun résultat théorique sur une éventuelle classification fondamentale des systèmes P2P en tant que systèmes asynchrones. On ne peut toujours pas dire de manière certaine que leurs caractéristiques essentielles -à savoir a) Internet comme réseau de communication et b) la volatilité des nœuds participants pouvant se déconnecter sans prévenir- les catégorisent comme des systèmes distribués asynchrones. En l'absence de tels résultats, il est tout simplement impossible de savoir si l'on peut s'appuyer sur la notion de consensus en ce qui concerne les services distribués et nous avons donc implémenté XtremWeb en partant du postulat que le consensus est impossible dans de telles conditions.

Le service de coordination est composé de différents services. Les services de réplication et de gestion des événements (sous couche 2) ; ces points sont spécifiquement traités au paragraphe 3.3.3. Les services de publication, d'instanciation et de cycle de vie (sous couche 3) ; et les services de découverte, d'ordonnancement et de résultats (sous couche 4). Ces services travaillent conjointement sur une liste de tâches maintenue par un graphe d'états.

Le service de publication permet de publier les services et les applications afin que les clients puissent y accéder dynamiquement. Le service d'ordonnancement gère la bonne exécution des tâches par les services de calcul, en s'appuyant sur le service d'instanciation. Il s'assure que les tâches sont correctement prises en compte et exécutées par les services de calcul. Il peut demander au service d'instanciation de les arrêter en fonction de certains événements (demande d'arrêt de la part du client, fin de vie) et il corrige les fautes dues à la plate-forme elle même (volatilité des nœuds) en s'appuyant sur le service d'instanciation pour les relancer sur d'autres service de calcul si besoin est. L'instanciation des tâches aux services de calcul est gérée sur le mode *pull* : ce sont les services de calcul qui viennent demander des tâches à l'ordonnanceur afin de résoudre les problèmes de connexion (Cf paragraphe "les pare-feux", chapitre 2.3.5). Le service de résultat permet aux services de calcul de déposer les résultats obtenus après calcul des tâches, d'une part, et au service client de récupérer ces résultats, d'autre part.

Le service de calcul Le service de calcul est un des deux services distribués de XtremWeb ; il prend en charge les tâches à exécuter, les exécute et en retourne les résultats. Ce service est lui même constitué de trois services : le service de communication, le service d'exécution et le service *moniteur* qui gère localement l'utilisation de la ressource de calcul. Ce dernier contrôle quand et comment un calcul peut être effectué sur la machine hôte, en tenant compte de certains paramètres déterminés par la politique d'activation de la configuration du service de calcul lui même. La politique d'activation détermine les conditions d'utilisation de la ressource par le service de calcul (pourcentage d'utilisation du CPU, activité clavier/souris...) et inhibe les services de communication et d'exécution tant que ces conditions ne sont pas réunies. La seule activité autorisée étant alors la communication du *«poul»* (*heart beat*) (Cf chapitre 3.3.3).

La liste des tâches est le point central du service de calcul ; elle est gérée par le protocole de production/consommation entre les services de communication et d'exécution. Chaque tâche peut être dans l'un des états suivants (Cf figure 3.13) : *créée*, signifie que le service de communication n'a pas fini de télécharger les paramètres et/ou le binaire de la tâche ; *prête*, qui signifie que le téléchargement s'est correctement déroulé et que la tâche peut maintenant être exécutée ; *exécution*, qui signifie que la tâche est en cours d'exécution ; *erreur*, qui signifie que la tâche n'a pas pu s'exécuter correctement ; *annulée*, qui signifie que l'exécution de la tâche a été interrompu par le service *moniteur* ; *sauvegarde*, qui signifie que les résultats sont en cours de transfert vers le serveur de résultats ; et enfin *finie*, qui signifie que les résultats de la tâche ont été correctement envoyés au serveur de résultats. Le service de calcul garde les tâches finies jusqu'à ce que le serveur de résultats l'informe qu'il peut maintenant les effacer. Ceci assure que le serveur reçoive et stocke correctement les résultats. En cas d'erreur de transmission, le serveur de résultat a donc la possibilité de redemander les résultats qu'il n'a pas pu sauvegarder correctement.

Le gestionnaire de communication a en charge les communications avec les autres entités, qui sont de trois types. Il y a tout d'abord le *«poul»*, qui est un signal constamment envoyé au service de coordination afin d'avertir celui-ci que le service de calcul est toujours là ; ce «poul» contient les informations relatives aux éventuels tâches et résultats gérés par ce service de calcul. Le service de "moniteur" ne peut interrompre l'envoie périodique de ce signal. Le gestionnaire de communication gère aussi le téléchargement des tâches (et leurs paramètres et fichiers) et des binaires. Enfin, ce service prend en charge le retour des résultats à la fin des tâches.

Dès qu'une nouvelle tâche est téléchargée par le service de communication, elle est insérée dans la liste des tâches. Le service d'exécution la prend alors en compte. Il prépare l'environnement de la tâche, fourni par le client (fichier d'entrée, arborescence de répertoires), lance la tâche et la marque en *"exécution"*; il surveille cette exécution la lance et attend qu'elle ait fini son calcul. Dès que la tâche est finie, le gestionnaire d'exécution prépare le résultat (fichier output et tout fichier crée ou modifié par la tâche) et change son état à *sauvegarde*. Le gestionnaire de communication peut alors prendre en charge le retour des résultats vers le serveur de résultats.

A n'importe quel moment (pendant l'exécution ou les communications), le moniteur peut demander l'interruption du travail en cours, selon la politique d'activation. La tâche est alors marquée *interrompue* et le gestionnaire de communication en informe le service de coordination afin qu'il ait la possibilité d'essayer d'ordonnancer la tâche sur un autre service de calcul.

Le résultat est gardé par le service de calcul jusqu'à ce que le serveur de résultats l'autorise à l'effacer. Ainsi le service de coordination se garde une chance de pouvoir redemander le résultat au worker en cas de problème de transfert.

Quand un service de calcul démarre (démarrage de la machine hôte, reprise sur crash...), il efface les tâches interrompues parce qu'il suppose qu'elles ont été relancées sur un autre worker. Par contre, il reprend le transfert des résultats qu'il possède encore.

Le service client Le service client est le deuxième des services distribués de XtremWeb ; il est l'intermédiaire entre les applicatifs utilisateurs et la plate-forme GC. Son implémentation comporte une librairie et un «daemon». La librairie propose une interface permettant le dialogue entre les applicatifs et le service de coordination. Les actions de base permettent l'authentification, la soumission de tâches et le rapatriement des résultats. Pour répondre la tolérance au panne, le «daemon» implémente deux mécanismes : 1) sauvegarde locale des messages échangés entre les différentes parties et 2) synchronisation avec le service de coordination à chaque (re)connexion. Le premier permet à un participant de retrouver son état au dernier arrêt et permet d'arrêter des applicatifs utilisateurs sur une machine et de les relancer dans le même état sur une autre machine. Les deux conjugués permettent une grande mobilité des applicatifs utilisateurs, dont le premier est le client.

Le service de spécialisation de la plate-forme La plate-forme est architecturée afin d'être aisément spécialisée en fonction des besoins des utilisateurs. Elle implémente une interface de base qu'il convient d'étendre selon la spécialisation souhaitée (figure 3.9). Grâce à cette modélisation, tout nouveau service spécialisant la plate-forme est auto descriptif et dynamique. La plate-forme propose à cet effet une interface de base, `xtremweb.services.Interface`, que tout nouveau service doit implémenter. Cette interface spécifie trois méthodes : `exec()`, `getResult()` et `setLevel()`. La première contient le cœur de métier du service, elle est appelée pour exécuter le service ; la seconde permet de récupérer les résultats, après exécution. La troisième permet de définir le niveau des messages d'exécution permettant de suivre l'exécution du service. Tout nouveau service doit au moins contenir une interface dérivant de l'interface des services de la plate-forme, afin de permettre une gestion dynamique du service (découverte, cycle de vie), et une classe implémentant cette interface.

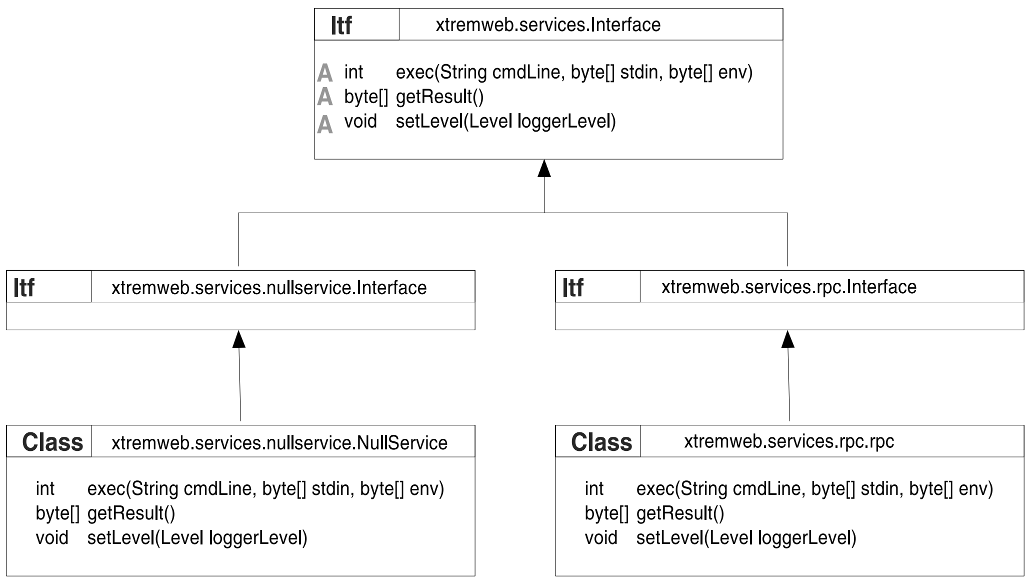

FIG. 3.9 – Structure des services de spécialisation de XtremWeb.

La plate-forme propose un service de démonstration afin d'aider les développeurs à bien comprendre l'implémentation et la mise en œuvre de nouveaux services : `xtremweb.services.nullService`. Ce package contient une interface dérivant de `xtremweb.services.Interface` et une classe d'implémentation. L'exécution de ce service de démonstration ne fait absolument rien, bien entendu.

Sur la base de ce service de spécialisation, XtremWeb propose un service de virtualisation des environnements d'exécution parallèle tel que *RPC (Remote Procedure Call)*. De tels environnements sont architecturés selon le paradigme client/serveur ; XtremWeb propose donc les services nécessaires à la virtualisation des serveurs de ces environnements. Actuellement la plate-forme implémente le service de virtualisation pour le protocole client/serveur *SunRPC*. Les mécanismes du transport des communications SunRPC sont détaillés au chapitre 5.

Cette architecture de service de spécialisation est suffisamment générique pour imaginer le transport de tout autre protocole. De telles implémentations nécessiteront toutefois une forte spécialisation de chaque nouveau service de virtualisation.

3.3.3 Qualité de service

La qualité de service (*QoS*), introduite au chapitre 2.2.4, concerne deux caractéristiques dans un système de calcul global tel qu'XtremWeb : la bonne exécution des tâches soumises et le temps nécessaire à ces exécutions.

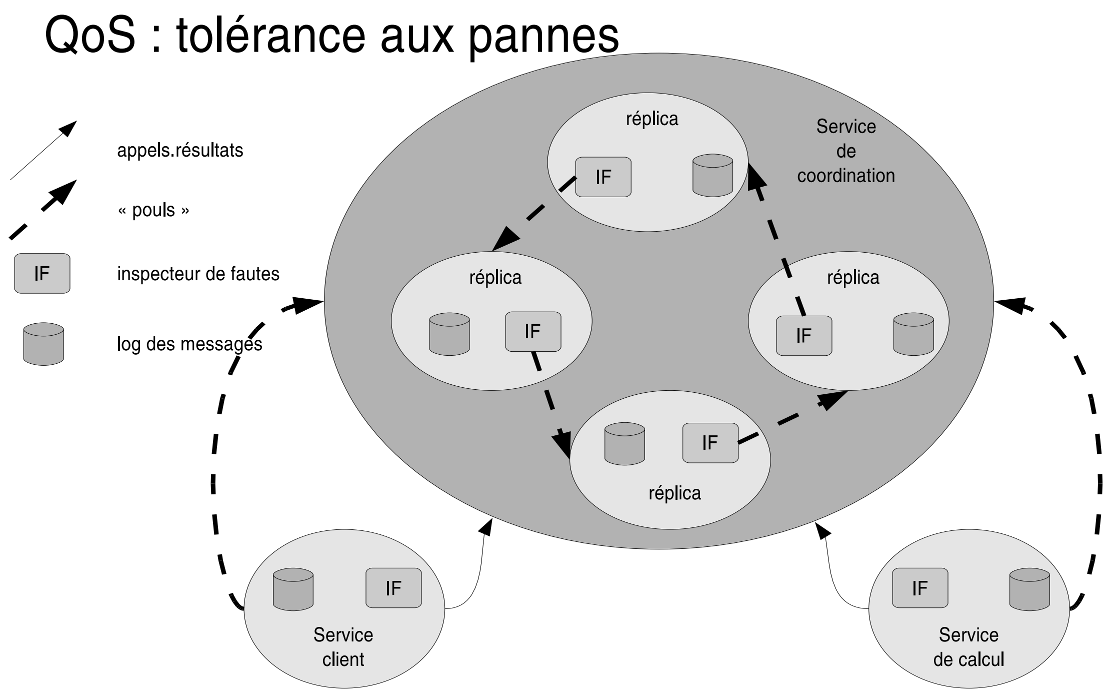

FIG. 3.10 – La tolérance aux pannes dans XtremWeb.

La bonne exécution des tâches dépend des capacités de tolérance aux pannes de la plate-forme. XtremWeb inclut un système de tolérance aux pannes transparent et automatique en mettant en œuvre deux mécanismes : la réplication et la journalisation de messages (sous couche 2 de la figure 3.8). La réplication sur la base de l'élection d'une nouvelle ressource [43] est impossible sur une plate-forme de calcul global car elle repose sur la notion de consensus qui est impossible sur de telles plates-formes (Cf chapitre 2.3.4). Les protocoles XtremWeb de réplication sont donc basés sur le mode pessimiste où toute ressource est soupçonnée de ne pas fonctionner correctement et de s'être déconnectée de manière intempestive. La plate-forme est capable de resynchroniser les ressources non soupçonnées afin d'assurer la cohérence du système grâce à ses capacités de réplication passive du service de coordination ; le service de coordination est donc un ensemble de réplicats du coordinateur. La journalisation de message est assuré sur toutes les ressources de la plate-forme : les coordinateurs du service de coordination, le client et le worker. La tolérance aux pannes mise en œuvre dans XtremWeb ne fonctionne que pour les services applicatifs *stateless*, c'est à dire que l'exécution d'une tâche applicative ne doit pas dépendre de résultats d'exécutions antérieures.

Des pannes peuvent intervenir sur tout ou partie de la plate-forme, de manière intermitante ou définitive, bien qu'un minimum d'un service client, d'un service de calcul et d'un service de coordination composé d'au moins un coordinateur, tous capables de communiquer, soit nécessaire afin d'assurer un service minimum. Le cœur de la tolérance aux pannes dans XtremWeb est *l'inspecteur de faute* qui permet, dans un environnement asynchrone tel que l'Internet, de *suspecter* les fautes des ressources, à défaut de les détecter. Cette suspicion se fait grâce à un signal périodique, le *poul*, qui est envoyé vers l'entité sous surveillance. Si on ne parvient pas à déterminer le poul de cette ressource (si cette ressource ne répond pas au signal), elle est suspectée de faute. Une mauvaise suspicion, où la ressource aurait répondu un *peu* tard au signal, est résolue par une resynchronisation des entités, comme nous allons le voir. La figure 3.10 montre que chaque composant de la plate-forme possède son propre inspecteur de faute ; le client a le sien afin de suspecter les fautes du coordinateur auquel il est connecté ; les replicats du service de coordination ont chacun le leur afin de suspecter les fautes entre eux. Le worker a le sien, mais avec un double impact ; il lui permet de suspecter une faute du réplica auquel il est connecté, mais il permet aussi au service de coordination de suspecter les fautes des services de calcul. Ceux ci sont en effet censés envoyer un signal périodique au service de coordination. Le service de coordination suspecte de faute tout service de calcul qui n'envoie plus ce signal. Les inspecteurs des fautes vont toujours *vers* un coordinateur afin de résoudre les problèmes liés aux pare-feux (Cf chapitre 3.3.4). Les clients mis à part, on voit donc que toutes les ressources du système sont sous la surveillance d'un inspecteur. Les clients ne sont pas inspectés, car ils restent sous la responsabilité des utilisateurs à qui il revient d'assurer le bon fonctionnement de leurs clients.

En plus d'être inspectée, chaque ressource (client compris) garde la liste des messages envoyés afin de pouvoir se synchroniser s'il survient une reprise sur erreur (si une ressource a disparue). Cette synchronisation se fait sur un mode passif ; elle évite les tâches orphelines ou perdues. Les replicats du service de coordination se synchronisent entre eux afin d'avoir tous la même liste de tâches et de résultats et d'assurer un état cohérent au service. Ainsi si un réplicat s'arrête inopinément, les clients et workers peuvent continuer à travailler avec un autre réplicat. Des cas d'incohérence de la plate-forme peuvent arriver si les réplicats du service de coordination n'ont pas pu se synchroniser à cause d'une faute de l'un d'eux ; les clients et les workers doivent donc utiliser leur journaux de messages afin d'aider le service de coordination à retrouver un état cohérent.

Si un client vient à être interrompu, il se synchronise avec le service de coordination lors de son redémarrage. La bonne synchronisation des réplicats du service de coordination assure une indépendance des réplicats par rapport au client qui peut donc redémarrer en se connectant à n'importe lequel d'entre eux. Si un worker vient à être interrompu, le coordinateur choisi un autre worker pour réordonnancer la tâche en cours, ainsi que toutes les tâches pour lesquelles on aurait perdu les résultats consécutivement à la perte du worker. Si un réplicat du coordinateur vient à être interrompu, les réplicats voisins décident, *seuls*, de se réorganiser afin d'assurer une topologie cohérente de la plate-forme. Ils le font *seuls*, car il est impossible d'établir un consensus sur ce type de plate-forme, comme nous l'avons déjà dit (Cf chapitre2.3.4). Les clients et les workers qui étaient connectés à ce réplicat, se connectent alors automatiquement à un autre réplicats ; ils restent ainsi en contact avec le service de coordination.

Un client pourra toujours continuer à travailler, à partir du moment où il existe un lien entre celui-ci et un réplicat du coordinateur, d'une part, et entre un réplicat et un worker, d'autre part. Cette condition, assurée par la plate-forme, est suffisante pour qu'un client puisse progresser dans son travail.

La plate-forme propose donc bien une qualité de service quant à la prise en charge des calculs soumis. La QoS se mesure aussi en terme de performances et il convient que la plate-forme exécute les tâches qui lui sont soumises sans dégrader les temps d'exécution, ou du moins en restant dans des perspectives raisonnables. Ce point précis est discuté au chapitre 6.

3.3.4 Communications

Les communications entre les différentes parties d'XtremWeb s'appuient exclusivement sur le protocole Internet (*IP*). Elles incluent les appels de procédure à distance (*RPC*) et les transferts de données. On trouve le modèle de communication RPC dans de nombreux environnements de calcul distribué : Corba, COM de Microsoft, JavaRMI, SOAP du consortium W3C. Ce dernier représente un effort de standardisation du modèle RPC afin d'augmenter l'inter-opérabilité des appels RPC (alors que COM est destiné au monde Windows, JavaRMI à Java).

L'architecture de communication de XtremWeb repose sur un protocole en quatre couches (figure 3.11) qui font elles mêmes partie de la sous couche (1) de la figure 3.8.

La couche «transport» assure le transfert des messages. Elle inclut deux protocoles connus et utilisés sur IP, *TCP* et *UDP*. TCP est un protocole assurant un transport fiable et connecté de paquet ; il assure une transmission des données sans perte ni erreur, suivant l'ordre dans lequel elles ont été envoyées. Pour assurer cette fiabilité, ce protocole induit un coût non négligeable dû au nombre important de mes-

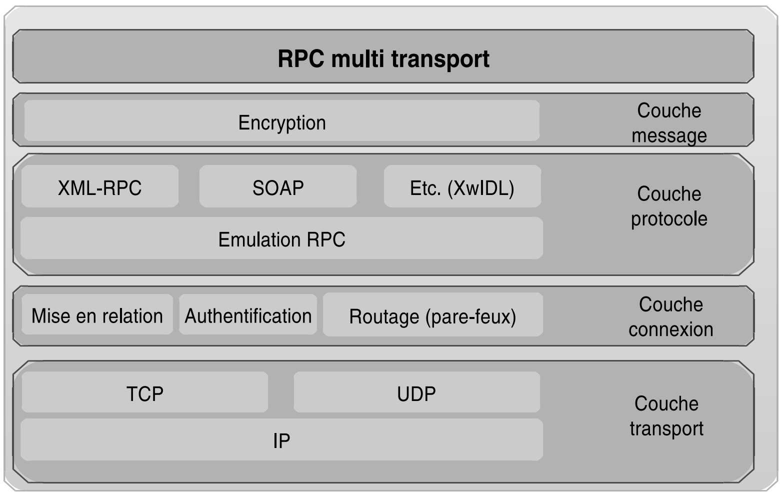

FIG. 3.11 – La couche communication dans XtremWeb.

sages transitant pour la vérification du transport des données. Dans de nombreux cas, toutefois, ce coût induit par TCP peut se révéler prohibitif et certains services, dont la qualité dépend fortement du temps, peuvent préférer utiliser une couche de transport moins fiable mais plus performante, quitte à résoudre eux mêmes les problèmes de communication induits par un transport non fiabilisé. De tels services, ne requérant que le seul multiplexage applicatif, peuvent alors préférer le protocole UDP, beaucoup plus performant que TCP, mais non viabilisé. UDP ne garantie en aucune manière que les données envoyées seront effectivement reçues. Il est alors de la responsabilité des services et applications utilisant ce protocole de prendre en compte ce manque de fiabilité et de fournir les moyens de le corriger.

La couche «connexion» permet de mettre en relation les différentes entités qui peuvent être derrière un pare-feu, en NAT ou derrière un proxi (Cf chapitre 2.3.5). Elle assure aussi l'authentification des ressources ainsi que la stabilité des communications en s'appuyant sur le protocole TCP de la couche inférieure. XtremWeb ne propose aucun système de fiabilisation du protocole UDP et laisse ce travail à la charge des services et applications souhaitant utiliser ce protocole. Cette politique a été délibérément choisie afin de ne pas dégrader les performances gagnées par l'utilisation de UDP. La problématique des pare-feux, qui sont en général configurés afin de bloquer les communications entrantes -sauf les plus connues (web, ssh etc.)- et laisser passer les communications sortantes, est résolue dans XtremWeb en laissant libre d'accès le service de coordination (décrit au chapitre 3.3.2). Il convient donc de configurer les éventuels pare-feux devant ce service afin de laisser passer les communications entrantes vers les coordinateurs formant le service de coordination, en ouvrant les *ports de communication* nécessaires aux protocoles mis en œuvre par le service de coordination. Chaque protocole a besoin de son port et du service correspondant ; le port est géré par le pare-feu, le service est implementé par XtremWeb. Par exemple, *XML-RPC* a besoin d'un serveur *HTTP* dont le port par défaut est le numéro 80. Le service de coordination n'étant jamais à l'origine des communications, il revient aux autres parties (client, worker) de le contacter. Ceci permet de passer les éventuels pare-feux derrière lesquels il se trouverait. Les clients et workers le contactent, envoient leurs messages et reçoivent les réponses par un seul et même canal de communication. Ceci est possible car les communications sont basées sur *IP* qui est un protocole *full-duplex*, c'est à dire qu'il permet spécifiquement d'utiliser un même canal de communication pour envoyer *et* recevoir des données. Dès qu'une communication est établie, les deux intervenants peuvent envoyer et recevoir des données. Ceci est vrai, indépendamment de toutes les couches au dessus de IP.

La couche « protocole » propose différentes implémentations des interfaces de programmation des RPC. Elle émule une couche RPC sur laquelle tout autre protocole peut venir s'ajouter. *XML-RPC* est fourni avec la plate-forme, mais le package contient un outil, *XwIDL*, afin de générer automatiquement cette couche dans divers protocoles. Il est facile d'utiliser cet outil si l'on souhaite ajouter d'autres implémentations comme *SOAP*, ou d'autres. Le choix du protocole dépend de plusieurs facteurs. Les protocoles XML-RPC et SOAP décrivent les appels de méthode et les échanges de messages en langage *XML*, un langage de description largement adopté pour la description des services. Ce langage est un peu lourd et verbeux, mais à le grand avantage d'être parfaitement indépendant de tout environnement de programmation et d'exécution. Ces différents protocoles limitent toutefois la taille des messages échangés (quelques centaines de kilo-octets) et ne conviennent pas aux échanges de données binaires. XtremWeb implémente un transfert transparent de données binaires, en plus des protocoles d'échanges de messages, afin de pouvoir transférer les binaires des applicatifs, les paramètres des tâches ainsi que leurs résultats.

La couche « message », avec l'aide de la couche « connexion », assure la protection et l'intégrité des messages. Elles s'appuient sur la librairie *SSL* qui permet de chiffrer les messages ainsi que d'authentifier les partenaires afin que les communications ne s'établissent qu'entre ressources autorisées. Dans une telle plate-forme, la protection et l'intégrité des communications ne peut se conceptualiser qu'au niveau de cette dernière couche, dans la mesure où le transfert d'information peut nécessiter plusieurs communications, car, en général, les interlocuteurs ne peuvent se connecter directement entre eux (Cf chapitre 2.2.6).

La figure 3.12 représente les communications possibles dans XtremWeb. Cette plate-forme étant principalement centralisée, les communications convergent majoritairement vers le service de coordination : les communications entre coordinateurs du service de coordination dans le cadre de la réplication (Cf chapitre 3.3.3), ainsi que les communications entre service client et service de coordination, et entre service de calcul et service de coordination.

Il existe toutefois des communications entre clients, entre workers, et même entre clients et workers. La couche connexion d'XtremWeb permet de mettre en relation des clients, des workers et même directement des client et des worker, grâce à un

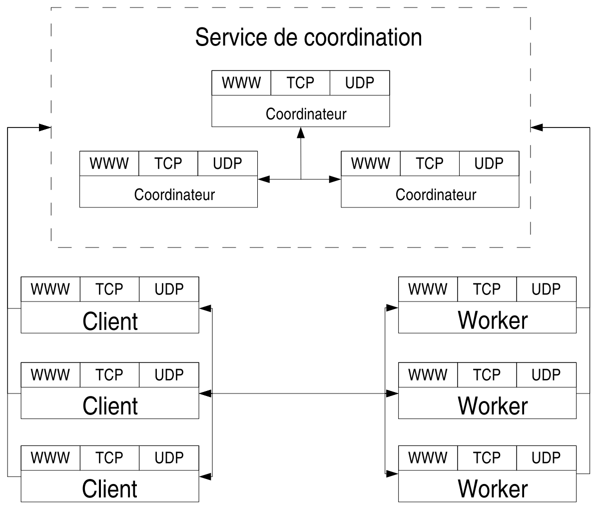

FIG. 3.12 – Les communications dans XtremWeb.

framework basé sur la notion de *handler* [54] qu'il convient d'implémenter afin de spécialiser la plate-forme. Cette implémentation des différents handlers est laissée libre et à la responsabilité de quiconque souhaite étendre la couche communication de la plate-forme afin de pouvoir échanger des messages vers les clients ou les workers. Un tel déploiement, où les clients et les workers seraient susceptibles de recevoir des messages dans le cadre de communications qu'ils n'auraient pas eux mêmes initiées, nécessite une bonne gestion des différents pare-feux.

3.3.5 Le protocole multi-phases

Le protocole de communication suit le schéma *multi-phases* tel que proposé par Condor générant un certain nombre de petits messages afin d'optimiser l'utilisation de la bande passante grâce à une gestion optimum des erreurs de communication. Les messages sont journalisés sur les différents intervenants afin d'assurer un suivi des communications, ce qui permet de synchroniser les différentes parties de la plate-forme en cas d'erreur. Ce protocole, associé aux journaux des messages, assure la cohérence de la plate-forme.

Ce schéma évite les tâches *orphelines* qui consommeraient des ressources inutilement, sans aucun moyen d'en contrôler les exécutions. Une tâche orpheline est une tâche que le client ne peut plus contrôler. Ce schéma évite aussi les tâches *perdues* en garantissant que les résultats seront toujours retournés au client les ayant soumises. Une tâche perdue est une tâche dont le client ne peut plus récupérer les résultats.

La figure 3.13 schématise le protocole de communication implémenté dans XtremWeb.

Chaque tâche soumise doit avoir un identifiant unique dans le temps et dans l'espace qu'on appelle un *UID*. Il existe aujourd'hui plusieurs algorithmes permettant de résoudre ce problème ; les librairies Java sont fournies avec une implémentation

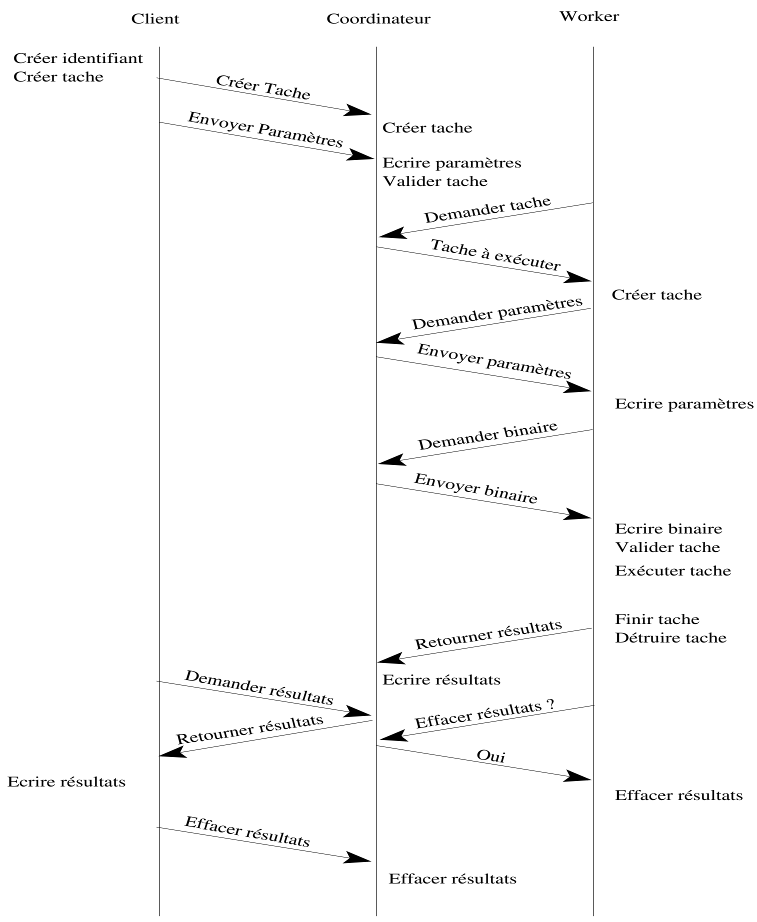

FIG. 3.13 – Le protocole multi-phases.

d'un tel algorithme. Il est de la responsabilité du client souhaitant soumettre une tâche, d'associer un tel identifiant à chaque tâche soumise afin d'éviter d'insérer des tâches orphelines dans le système. Ainsi le client connaît forcément toute tâche qu'il a lui même crée et il est de la responsabilité du client de se souvenir des tâches qu'il crée. La création d'une tâche se fait en deux messages : la création sur le service de coordination avec l'envoie de la tâche et de son identifiant unique, et l'envoie des paramètres. En cas de problème sur la transmission d'un des deux messages, on ne revoie que le message en erreur et on gagne en bande passante. De plus, l'identifiant unique étant généré par le client, une erreur lors de la création de la tâche n'a pas d'incidence sur les tâches gérées ni sur l'utilisation de la bande passante. Le fait d'envoyer les paramètres et les binaires séparément permet d'assurer ces transferts pour les tâches correctement commitées sur le service de coordination. Notons qu'on ne peut soumettre de tâche *que* pour une application insérée dans la plate-forme, c'est à dire connue du service de coordination. Insérer une application dans la plate-forme se fait suivant le même protocole multi-phases, avec la création de l'application puis l'envoie de son (ses) binaire. Ainsi la création d'une tâche ne nécessite pas l'envoie du binaire de l'application. Cette manière de faire sécurise aussi la plate-forme contre les codes malicieux puisque les clients n'ont généralement pas le droit d'insérer de binaire dans la plate-forme ; il faut pour cela des droits spécifiques.

Pour éviter que deux instances d'un même client n'entrent en conflit (suite à un mauvais arrêt du client, par exemple) XtremWeb introduit la notion de *session cliente*. Dès que le service de coordination perd le contact avec un client, il détruit toute tâche incluse dans la session de celui ci. Ainsi, lors qu'une instance de ce même client se reconnecte, cette instance est bien la seule responsable de ses tâches.

Le client doit vérifier que la soumission s'est correctement passée et que les paramètres aient été correctement reçus par le service de coordination. Une fois la tâche validée par le client, le service de coordination la prend finalement en compte afin qu'elle soit susceptible d'être ordonnancée.

Lorsqu'un worker veut exécuter une tâche, son téléchargement se fait en trois messages afin de pouvoir surveiller le bon déroulement de ceux-ci et de perdre le moins de bande passante possible en cas d'échec de l'un d'eux. Le service de calcul commence par créer une tâche localement en la téléchargeant depuis le service de coordination ; il en télécharge ensuite les éventuels paramètres et les valide. Enfin il télécharge le binaire, si nécessaire et s'il ne l'a pas déjà dans son cache local. Le binaire est validé de la même manière. Les validations sont faites par des *MD5Sum* des objets téléchargés. Le téléchargement effectué et vérifié, la tâche est validée par le worker afin d'être exécutée. Une fois la tâche exécutée, le worker récupère et sauvegarde les résultats, détruit la tâche et envoie les résultat au serveur de résultat du service de coordination. Il interroge ce dernier afin, qu'ensemble, ils puissent vérifier que les résultats ont été correctement reçus et sauvegardés par le service de coordination. Dans ce cas, le worker peut effacer sa copie locale ; sinon il réessaye d'envoyer les résultats au service de coordination et ce, jusqu'à ce qu'ils soient transférer correctement. Finalement, la récupération des résultats par le service client peut se faire dès que les résultats sont validés par le service de coordination. Cette action ne se fait qu'à la demande du client. Le client sauvegarde ses résultats localement et peut demander au service de coordination d'en effacer sa copie.

Ainsi, grâce à ce protocole multi-phases, on évite que les tâches ne soient ni orphelines, ni perdues. La cohérence de la plate-forme est garantie.

3.3.6 Les entrées/sorties

Les opérations de lecture et d'écriture de données (les *I/O*) sont un paramètre essentiel du calcul car il serait inutile d'exécuter du calcul sans être capable de récupérer le résultat généré (et vice-versa). Les I/O ont un prix non négligeable, et l'exécution d'une tâche peut être plus moins efficace en fonction des capacités d'entrées/sorties de la ressource qui exécute la tâche et de la disponibilité des données elles mêmes ; une tâche peut même être complètement bloquée à cause d'opérations d'entrées/sorties non satisfaites.

Les opérations de lecture sont aujourd'hui un problème traité de manière satisfaisante ; les systèmes de fichier distants (NFS, AFS), les bases de données et les accès *Web* [106] permettent de résoudre le problème des accès en lecture seule à des ressources distantes, par réplication si nécessaire. La plupart de ces solutions proposent des systèmes de cache afin d'améliorer les temps d'accès.

Il est en revanche plus difficile d'assurer la consistance des données générées par des calculs (des données *écrites*, donc). Dans un modèle producteur/consommateur où une tâche (le *producteur*) écrit des données attendus en lecture par d'autres tâches (*les consommateurs*), la bonne exécution de l'ensemble des tâches nécessite la mise en place d'un système de message. Il est d'autre part nécessaire, dans un système distribué, d'assurer la consistance des données afin que les consommateurs puissent lire ce qu'écrit réellement le producteur, dans la version attendue.

Les I/O nécessitent elles aussi un schéma multi-phases (Cf 3.3.5) afin d'assurer un service tolérant aux pannes non seulement des différentes parties de la plate-forme, du réseau, mais aussi des ressources de stockage elles mêmes dans la mesure où ces dernières n'implémentent pas, en général, de processus transactionnel tel qu'on peut en trouver dans les bases de données et ne sont donc pas capable, à elles seules, d'assurer la consistance des données dont elles ont la charge. En effet, les ressources de stockage exigent soit qu'on ferme le fichier, soit qu'on vide les buffers, ce qui suppose qu'on arrive à le faire : que les transferts réseaux soient corrects, que les processus engagés dans le transfert aillent jusqu'au bout de leur travail.

Comme toujours, les méthodes à mettre en œuvre pour rendre les écritures tolérantes aux pannes dépendent des caractéristiques que l'on veut mettre en place. Sans aller jusqu'à un vrai système de fichier distribué, il est aisément imaginable de proposer quelques actions simples dont les opérations sont indépendantes de leur nombre d'exécution, c'est à dire des actions dont le résultat est constant, qu'on exécute leurs opérations une ou plusieurs fois. Dans le cadre de cette limitation, une action est complète dès que toutes ses opérations ont été exécutées correctement, quelque soit le nombre d'exécutions ou de réexéctions nécessaires à leur réussite. En cas d'erreur, seule l'opération en cause doit être réexécutée. Par exemple, l'écriture de données de longueur fixe à un emplacement spécifique dans un fichier répond à ces critères. On peut exécuter plusieurs fois cette même action, le résultat reste constant et il suffit que l'ensemble de ses opérations soit correctement effectuées (indépendamment du nombre d'essais nécessaires) pour que le résultat soit valide. A contrario, ajouter des données à la fin d'un fichier ne répond pas à ces critères, car au moment où se fait l'écriture, on ne connaît pas la taille du fichier et on ne pourra donc pas obtenir deux fois le même résultat s'il s'avère nécessaire de répéter l'opération, car la taille du fichier peut varier entre deux essais.

On peut encore alléger la gestion des fichiers en limitant les actions possibles à la simple possibilité de création de nouveaux fichiers, en excluant toute capacité de mise à jour de fichier existant. Un protocole tolérant aux pannes et simple à mettre en œuvre consiste à utiliser un fichier temporaire du côté de la ressource de stockage. Les données sont téléchargées et écrites sur le fichier temporaire. La ressource de stockage crée le fichier final en faisant une copie du fichier temporaire à la fin du transfert des données. En cas d'erreur de transfert des données ou d'arrêt inopiné d'un des intervenants, le fichier temporaire est effacé et on recommence l'écriture depuis le début.

Ce protocole est mis en œuvre dans beaucoup de système de grille. XtremWeb implémente cette seule fonctionnalité dans la mesure où la notion de fichier n'existe pas dans cette plate-forme. Les binaires, les paramètres et les résultats sont bien sûr gérés et transférés entre les différents tiers de la plate-forme, mais ce ne sont pas des fichiers manipulables par les utilisateurs, ce ne sont *que* des paramètres des tâches ordonnancées et ce protocole assure leurs bons transferts.

Une version d'XtremWeb incluant les fichiers en tant que ressources est actuellement à l'étude dans le cadre d'un doctorat de recherche.

3.3.7 Le déploiement

Le déploiement est une des premières difficultés des plates-formes de calcul global. Le déploiement inclut l'installation et la mise à jour des différentes parties du système. En fonction des utilisateurs et des applications, la plate-forme GC peut être déployée sur des ordinateurs individuels connectés à l'Internet sur ADSL par exemple, sur des pools d'ordinateurs connectés en réseau privé comme ceux que l'on peut trouver dans des salles de TP dans les écoles ou encore sur des clusters de centre de calcul. Ensemble, ces différents déploiements couvrent un large spectre de procédures d'installation, de configurations de la sécurité, de l'administration du système ainsi que des limites de l'intrusion au niveau des ressources. Avec XtremWeb, le déploiement concerne principalement les workers et les clients qui forment la partie distribuée de la plate-forme puisque, dans cette version, le service de coordination est installé sur une ou plusieurs machines qu'il faudra installer spécifiquement.

Dans ce chapitre, nous ne détaillerons l'installation ni sur des ressources individuelles, ni sur des ressources confinées dans un réseau privé dans la mesure ou il existe plusieurs techniques largement répandues pour ce faire (installation et mise à jour automatisées par accus Web, serveur le logiciels, CD-ROM etc.)

Nous allons par contre nous intéresser au déploiement sur un cluster. Tandis que les plates-formes GC ont initialement été pensées comme un moyen d'utiliser des puissances de calcul inutilisées sur le mode du vol de cycle, elles sont vite apparues comme une alternative légère aux infrastructure de grille permettant d'agréger les ressources de type cluster entre différents domaines administratifs.

Utiliser ces ressources de clusters nécessite la prise en compte des politiques de gestion et de sécurité mises en place par leurs administrateurs respectifs. En général, les accès à ces clusters ne sont autorisés que par noms d'utilisateur et mots de passe et la soumission de tâches se fait grâce à un ordonnanceur de tâches. Avec XtremWeb, l'utilisateur n'a pas accès au cluster ; il délègue cet accès au service de coordination qui gère la communauté des utilisateurs grâce à une autorisation unique (de type grille) que les administrateurs des clusters lui ont conférée. Les utilisateurs délèguent la soumission de tâches à travers une interface de soumission de type batch que le service de coordination met à leur disposition. Le service de coordination se charge de représenter les utilisateurs, en utilisant son propre compte « grille », afin de soumettre les tâches sur les cluster. Le service de coordination gère les informations sous la forme de t-uple [tâche, utilisateur] afin de gérer correctement les informations relatives à tout utilisateur et toute tâche et de pouvoir prendre toute décision qu'il jugera nécessaire. Par exemple, un utilisateur pourra être révoqué si une de ses tâches génère une faute de type sécurité.

On peut interfacer la soumission des tâches entre le service de coordination et le cluster de deux manières différentes. On peut soit faire tourner un service sur le front-end du cluster qui a en charge 1) de prendre en compte le tâches du service de coordination (téléchargement des codes binaires et des paramètres des tâches), 2) de soumettre ces tâches sur l'ordonnanceur du cluster, 3) de transmettre toute commande du service de coordination à l'ordonnanceur (status ou arrêt d'une tâche etc.), 4) transmettre les résultats de toute commande au service de coordination, 5) transférer toute information utile au service de coordination relative aux tâches en cours (terminaison etc.), 6) transmettre les résultats des tâches au service de coordination, et 7) exécuter tout action nécessaire à la fin d'une tâche (nettoyer les espaces disque etc.) Les interfaces entre la plate-forme GC et les différents ordonnanceurs existants pose les problèmes d'hétérogénéité comme pour toute grille de calcul : il existe plusieurs système d'ordonnancement déployés sur les clusters et il n'existe pas de standardisation à ce niveau.

Une autre façon de soumettre des tâches est de faire exécuter le worker sur le cluster comme étant lui même une tâche du cluster et en le soumettant à travers l'ordonnanceur du cluster. Une fois lancé, le worker se connecte au service de coordination et prend en charge des tâches de la plate-forme GC. Cette approche à l'avantage de ne pas avoir à s'interfacer avec les différents système de batch. En faisant tourner un « daemon » sur le front-end du cluster, on peut faire exécuter des services de calcul selon les besoin du service de coordination. Toute la complexité de la procédure précédente relative au transfert des binaires, à la gestion des tâches et de leurs résultats, à la bonne terminaison des tâches est alors de la responsabilité du worker. Cette procédure réduit la complexité d'adaptation à l'interface de chaque cluster et de leurs ordonnanceurs à un petit ensemble de commandes génériques.

3.4 Travaux autour d'XtremWeb

Les travaux menés au LRI dans le cadre du projet XtremWeb ne prétendent pas traiter tous les aspects du calcul global et pair à pair. Nous allons ici présenter quelques travaux qui n'ont pas été directement traités dans le cadre de cette thèse ; ils ont été effectués par d'autres chercheurs.

L'intérêt d'en parler ici réside en ce que ces travaux ont étudié des points précis du calcul global et pair à pair en partant d'XtremWeb et de l'implémentation précédemment décrite. Ces travaux sont alors des preuves suffisantes que la plate-forme XtremWeb est une plate-forme de recherche suffisamment générique pouvant servir de base de travail aux recherches sur les grilles.

3.4.1 Problématique de l'automatisation des traitements des résultats

Dans [31], les auteurs présentent l'étude d'une base logicielle (*framework*) autour d'XtremWeb pour la gestion des résultat en bio-informatique. Cette discipline a aujourd'hui un grand besoin en termes de ressources de calcul qui génère une importante masse de données : *GenBank*, par exemple, contient 34 milliards de bases nucleotidiques pour 27 millions de séquences [19].

Des projets tels que le projet de grille *Smallpox* ou encore *FightAID@Home* mettent en place des plates-formes de calcul global pour la biologie, sans proposer de solution quant au traitement des résultats. Il est d'autant plus difficile de faire de telles propositions que ces plates-formes sont implémentées sur des produits commerciaux dont les codes sources sont protégés par des droits d'auteur et donc non modifiables. Par exemple, le projet *Smallpox* s'appuie sur une infrastructure proposée par *IBM*, une plate-forme GC proposée par *United Devices* et une solution logicielle proposée par *Accelrys*. *FightAID@Home* de son côté s'appuie sur une plate-forme proposée par *Entropia* pour distribuer le logiciel de simulation *AutoDock*. Aucun de ces projets ne proposent de solution quant aux traitement des données.

Les auteurs souhaitent donc déployer une plate-forme GC pour la biologie afin de bénéficier de ces capacités de calcul global. Ayant remarqué qu'aucune plate-forme disponible ne propose de solution quant aux traitements des résultats, il proposent de développer un framework afin de résoudre ce manque. Entre *Models@Home*, *Condor*, *Globus* et *XtremWeb*, ils ont finalement choisi ce dernier pour son *« architecture modulaire et ses facilités de déploiement d'applications »*.

Les auteurs proposent un framework générique pour XtremWeb, permettant de prendre en charge n'importe quel type de résultat. Ce framework fonctionne sur la base de *handler* [54] qu'il convient d'implémenter afin de spécialiser la plate-forme. Les handlers sont implémentés du côté du du service de coordination et sont exécutés pour tout résultat retourné par un service de calcul.

Les auteurs propose un premier *handler* afin qu'il puisse prendre en charge automatiquement les résultats de l'application *Autodock*, l'application choisie pour la simulation d'interaction des liens protéiniques. Grâce à ce framework, la plate-forme est spécialisée et capable d'interpréter automatiquement des résultats de biologie. La figure 3.14 présente le cœur du framework (le petit « A » représente les méthodes abstraites et le petit « S » les méthodes statiques).

Toute extension de ce framework doit implémenter des classes dérivant les classes *PkgHandler* ainsi que *FileHandler* afin d'en implémenter les méthodes abstraites. La classe *CompositeHandler* ne devant pas être changer ; c'est une classe dite *finale*, une classe dont aucune autre classe ne doit ni ne peut dériver. Cette classe est en effet une brique fondamentale du framework : elle est responsable de la gestion de tout handler déclaré.

Conclusion : cette solution, bien qu'intéressante, ne répond pas à la probléma-

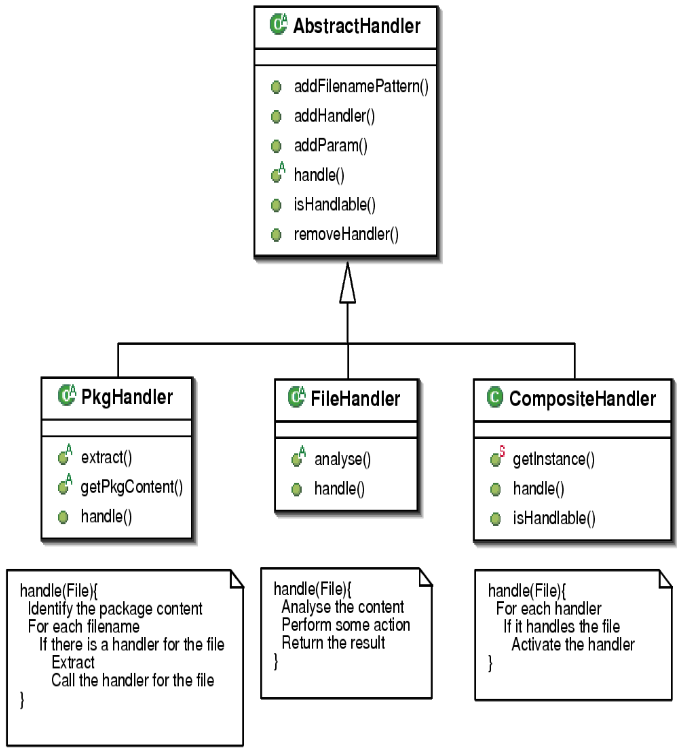

FIG. 3.14 – Le coeur du framework de traitement des résultats.

tique des plates-formes LSDS en ce sens que la scalabilité souffrira rapidement de la surcharge du travail au niveau du service de coordination. Il est dommage que les auteurs n'aient pas proposé de mesure de performance de leur solution.

3.4.2 Problématique de la communication inter-nœuds

L'équipe du Dr. Abdennadher de l'université des sciences appliquées de Genève a proposé *XWCH*, une couche placée au dessus de XtremWeb prenant en compte le déploiement d'applications distribuées et communicantes sur XtremWeb.

Deux versions sont proposées ; *XWCH-sMs* est centralisée : toutes les communications inter-nœuds passent par le service de coordination ; *XWCH-p2p* est entièrement décentralisée : les nœuds peuvent directement communiquer. Dans la première version, les communications transitent par le service de coordination sous forme de résultat de la part du worker qui exécute la tâche initiant la communication ; XWCH-sMs inclue un module, appelé module *espion*, qui sait redistribuer ce pseudo résultat au worker exécutant la tâche a laquelle le message est destiné. XWCH-sMs agit donc comme un tunnel de messages.

Dans la version XWCH-p2p, les messages ne transitent plus par le service de coordination, ils sont directement envoyés d'un worker à un autre. Pour ce faire l'espion est toujours utilisé pour l'ordonnancement des tâches en fonction des messages disponibles ; une tâche voulant envoyer un message à un worker donné, prépare son message et signale à l'espion de la disponibilité de ce message. L'espion se charge alors de prévenir la tâche destinataire qu'elle a un message et lui indique où le retrouver (adresse IP du worker et chemin de répertoire). Le worker destinataire va chercher lui même le message qui lui est destiné sur le worker ayant initié la communication. Si ce dernier n'est pas accessible parce qu'il est derrière un pare-feu, on utilise un serveur de données qui peut être le service de coordination lui même (on revient alors à une situation identique à XWCH-sMs).

Des mesures expérimentales de performances ont été effectuées dans le cadre d'une application de génération *d'arbres phylogénétiques*. Un arbre phylogénétique se construit à partir de séquences d'ADN et permet de montrer les liens de parenté entre différentes espèces vivantes. Les tests consistent à faire exécuter un algorithme parallèle (dit de *tree puzzle*) utilisant MPI [94], optimisé pour s'exécuter sur un cluster. La figure 3.15 présente le graphe de l'application. Le code a été modifié afin de tourner sur XWCH : les appels MPI ont été remplacé par des transferts de fichiers afin de valider les choix retenus pour la plate-forme.

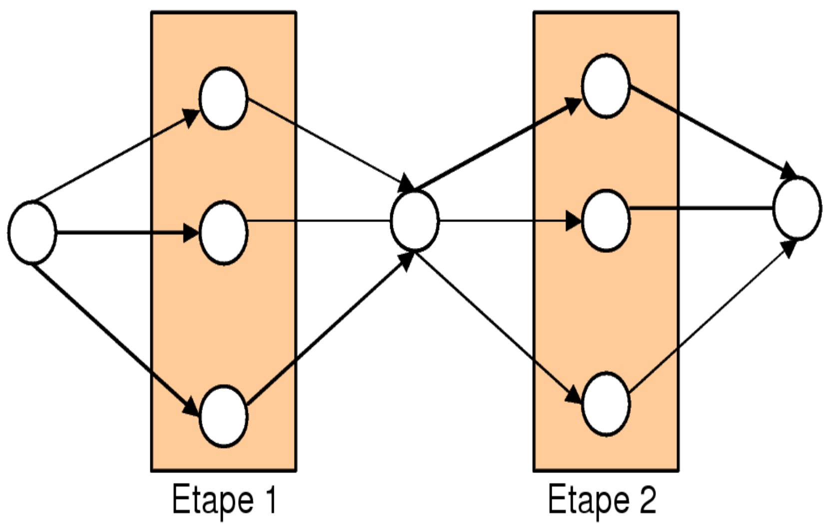

FIG. 3.15 – Graphe de l'application *Tree Puzzle*.

Les tâches de chaque étape partagent le même code. Le nombre de tâches de l'étape 1 est égale aux nombres de séquences d'ADN en entrées moins trois ; le nombre de tâches de l'étape 2 est variable. L'application Tree Puzzle a été exécutée avec deux ensembles de données : 64 et 128 séquences d'ADN, sur XWCH-p2p avec cent machines hétérogènes.

L'étape 2 consomme 70% du temps de calcul ; les tests sont fonction du nombre de tâches affectées a cette étape : 8, 16, 32, 64 et 128. Pour un nombre de séquences à 128, il est impossible d'exécuter l'application avec moins de 32 tâches à l'étape 2. Les figures 3.16 et 3.17 montrent les résultats obtenus en temps de calcul.

Le figure 3.18 montre l'utilisation de la bande passante en sortie du serveur. La phase I correspond au lancement de cinq workers ; la phase II au lancement de l'ap-

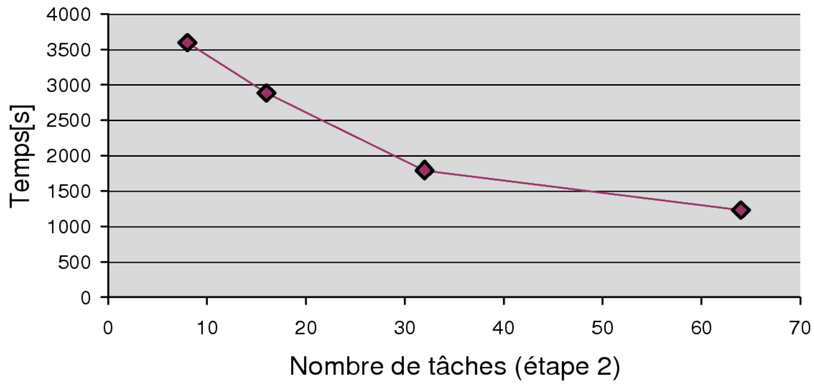

FIG. 3.16 – Temps d'exécution du *Tree Puzzle*. Nombre de séquences = 64

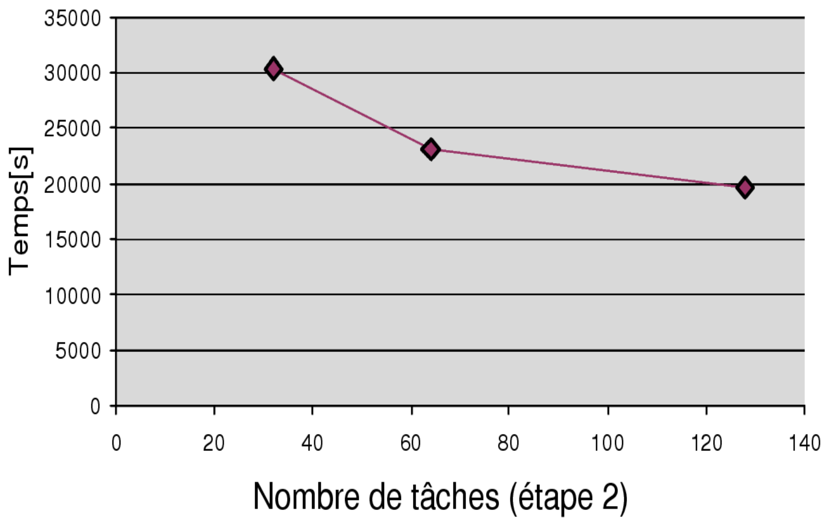

FIG. 3.17 – Temps d'exécution du *Tree Puzzle*. Nombre de séquences = 128

plication (code et paramètres) ; la phase III correspond à la communication services de calcul/service de coordination ; la phase IV à la montée en puissance jusqu'à vingt-quatre workers. La phase V au déploiement de l'étape 2 avec des données plus importantes en taille ; on voit qu'après le pic de communications sortante, la seconde moitié de la phase V génère un trafic presque nul en sortie du service de coordination : c'est que les communications se font entre les nœuds sans passer par le service de coordination. Enfin, la phase VI est similaire à la phase IV.

Conclusion : il est dommage que les auteurs n'aient pas proposé de comparaison entre les deux versions de XWCH, XWCH-sMs et XWCH-p2p. Cela aurait permis de mieux comprendre les avantages des deux méthodes.

Comme nous l'avons vu dans le chapitre 3.3.4, XtremWeb propose de rendre les nœuds communicants qui n'existait pas au moment de ces travaux de l'équipe du

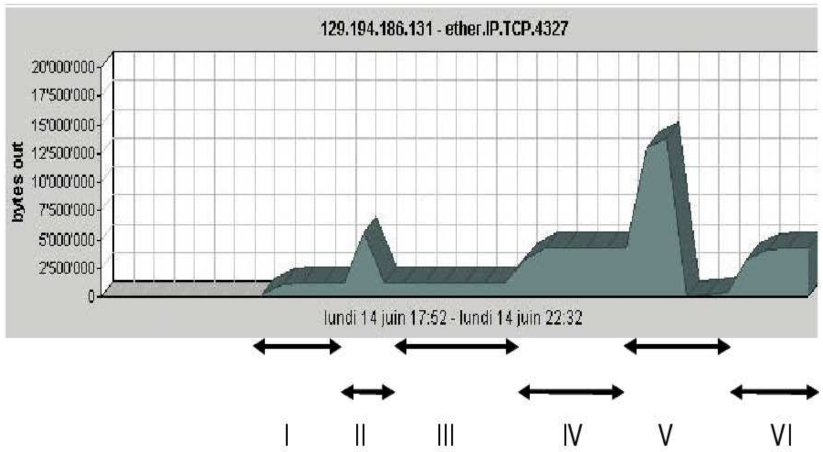

FIG. 3.18 – Trafic en sortie du serveur XWCH

Dr. Abdennadher ; il serait intéressant de comparer les résultats de ces travaux avec la couche de communication d'XtremWeb tel qu'elle existe aujourd'hui.

3.4.3 Problématique du stockage de données pair à pair

L'équipe du Laria, à Amiens, a propose *Ubiquitous Storage (US)*, un système de stockage de données sur une plate-forme pair à pair [101]. La problématique ici est de concilier la notion de pérénnité des données avec celle de volatilité intrinsèque aux plates-formes GC. US gère des blocs de données bas niveau ; il n'est pas ici question de gérer des méta-données ni de fournir une quelconque gestion de droits d'utilisation ; US peut être vu comme un disque dur qui doit être géré par un système de stockage de plus haut niveau.

US s'appuie sur une architecture trois tiers ; il y a les *clients* qui sont demandeurs d'espace de stockage, les *fournisseurs* qui gèrent les espaces demandés en découpant les blocs de stockage en fragments répartis de manière redondante [81] sur les *stockeurs* à qui ces fragments sont finalement alloués. La figure 3.19 synthétise l'architecture de US. Les blocs sont découpés en S fragments qui sont stockés de manière redondante sur $S+R$ stockeurs.

US dispose de plusieurs modes de déploiement, dont un sur XtremWeb que nous allons rapidement présenter ici.

XtremWeb est utilisé par US comme fournisseur de ressources. Un client XtremWeb spécialisé est dédié à l'utilisation de la plate-forme ; c'est le *fournisseur* de US auquel doivent se connecter les client US. Les tâches déployées par le service de coordination XtremWeb sur ordre du broker sont des instances du binaire d'allocation et de gestion d'espaces de stockage US ; ces tâches sont déployées sur les workers XtremWeb qui jouent donc le rôle de *stockeur* pour US.

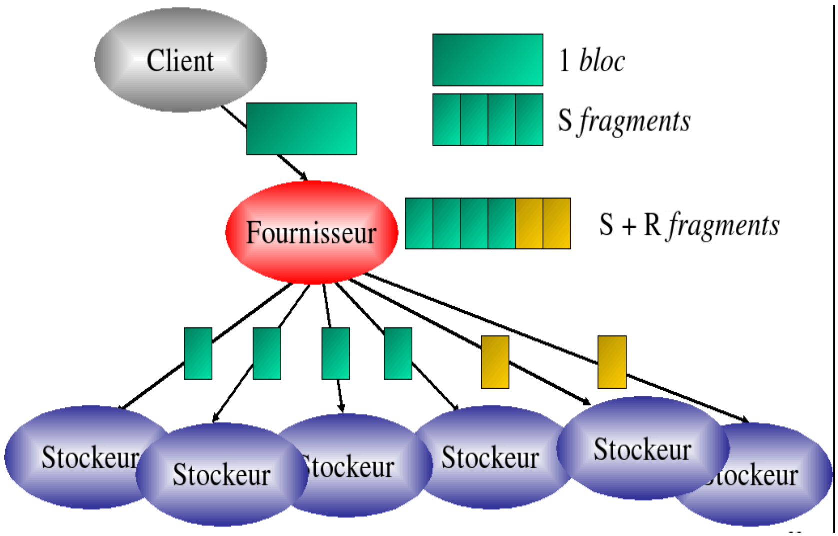

FIG. 3.19 – Architecture d'Ubiquitous Storage

3.4.4 Problématique de la standardisation des plates-formes de grille

L'équipe du laboratoire *PRISM* [5], de l'université de Versailles Saint Quentin en Yvelines, travaille sur la standardisation de l'utilisation des plates-formes globales et pair à pair afin de les rendre intéropérantes et transparentes aux applications nécessitant des ressources distribuées. Les travaux de recherche s'intéressent aux applications parallèles et aux problèmes relatifs à leur bonne exécution sur une plate-forme de calcul global. Cette équipe propose *YML* [76], un environnement d'exécution applicatif permettant de rendre transparente l'utilisation de ressources distribuées, indépendamment des plates-formes mises en place. YML inclut un nouveau langage de programmation parallèle (*YvetteML*) proposant une gestion par graphes et par composants.

Il existe plusieurs environnements permettant de distribuer des applications parallèles sur des plates-formes de calcul global qui se distinguent selon les modèles de programmation parallèle utilisés, d'une part, et les plates-formes sur lesquelles ils s'appliquent, d'autre part. Par exemple, certains environnements proposent des solutions pour les applications basées sur les passages de messages qui est un paradigme difficile à appliquer aux plates-formes à grande échelle : *OpenMP* utilise les mémoires partagées et n'est donc pas opérationnel sur ces plates-formes, et *MPI* a besoin d'un ensemble de ressources statiques pour démarrer, ce qui n'est pas compatible avec des environnements de calcul global où les ressources sont fluctuantes en quantité comme en qualité. On peut citer, parmi ces environnements pour plate-forme de calcul global, *MPICH-V* [20] ou encore *MPICH-G* [88], incluant, en outre, une tolérance aux pannes transparente. D'autres environnements proposent des so-

[5] Parallèlisme Réseaux Informatique Systèmes et Modélisations `http ://www.prism.ucsq.fr`

lutions aux applications parallèles utilisant l'appel de procédures à distance (*Remote Procedure Call, ou RPC*) : OmniRPC [92], GridRPC [97] ou encore P2P-RPC [34]. Tous ces environnements ne sont ni compatibles, ni intéropérants entre eux.

YML propose une couche transparente d'abstraction entre les applications et les plates-formes de calcul global, telle que représentée par la figure 3.20.

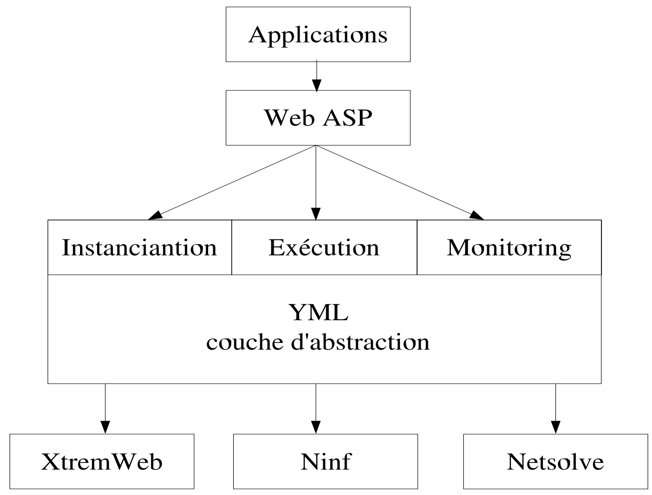

Fig. 3.20 – Architecture de YML

Les utilisateurs utilisent les services Web de YML (*Web Application Service Provider*) permettant d'exécuter des applications complexes sur les ressource proposées. Les services proposés par YML représentent le plus petit dénominateur commun aux services de tout middleware et permettent de soumettre des tâches, de les contrôler et d'en récupérer les résultats quelle que soit la plate-forme utilisée.

L'application de YML sur XtremWeb met en œuvre une mémoire partagée sous la forme d'un espace de stockage persistant permettant aux différentes parties de l'application parallèle de communiquer. YML assure la cohérence des messages échangés et interagit avec XtremWeb afin que l'ordonnancement suive un graphe de tâches fournit par l'utilisateur.

3.5 Conclusion

Dans ce chapitre, nous avons présenté XtremWeb, notre plate-forme de calcul global. Nous avons montré que la plate-forme, dans sa version initiale, ne présentait pas les caractéristiques nécessaires aux travaux de cette thèse, avait quelques manquements en terme d'architecture, de protocoles et de qualité de service, et ne répondait pas à certaines caractéristiques théoriques.

Nous avons détaillé les changements qu'il a été nécessaire d'apporter et présenté le planning des tâches accomplies durant cette thèse. Nous avons explicité les nouvelles caractéristiques mises en œuvre en introduisant la nécessité de chacune afin de venir à bien de nos travaux présentés dans les chapitres suivants.

Nous avons expliqué les notions de tâches perdues et orpheline, détaillé la nécessité de résoudre cette problématique et montré qu'un protocole multi-phases les résoud correctement.

Nous avons détaillé les capacités de tolérance aux pannes et de qualité de service de notre plate-forme et montré que dans un environnement asynchrone comme l'Internet où il n'est pas possible d'obtenir de consensus, ces capacités ne peuvent s'appliquer qu'à des applications *stateless* dont les exécutions sont indépendantes les unes des autres.

Enfin, nous avons montré tout l'intérêt porté à notre plate-forme par la communauté de recherche en informatique, en présentant quelques travaux réalisés autour d'XtremWeb par différentes équipes.

Chapitre 4

Interconnexion de plates-formes de calcul

Résumé. Dans ce chapitre nous introduisons le concept de *gliding*, ou l'art d'interconnecter différentes plates-formes de calcul.

Nous introduisons les différences majeures entre différents types de plates-formes et explicitons les avantages et les inconvénients de chacune d'elles. Nous expliquons les technologies mises en œuvre afin de dépasser les limites administratives, et nous prenons comme exemple la plate-forme Condor [66] qui utilise la boîte à outils Globus [4] à cette fin.

Nous détaillons comment et pourquoi notre plate-forme XtremWeb peut arriver aux mêmes capacités, plus facilement et en apportant un nouvel atout majeur par rapport aux autres technologies : la gestion globale des ressources multi-domaines, grâce à ses grandes capacités de tolérance aux pannes, d'adaptation et d'auto-organisation dans un environnement hautement volatile. Notre proposition permet de gérer les tâches des utilisateurs sur plusieurs domaines et de les faire automatiquement migrer entre ces domaines en fonction des pannes et de la disponibilité des ressources.

Ces travaux, faits avec l'équipe de Miron Lyvni de l'Université du Wisconsin (USA) ont été publiés sous forme d'un chapitre, dans le livre «Grid2» [69].

4.1 Introduction

Les plates-formes de calcul global et les grilles institutionnelles ont chacune leurs avantages et leurs inconvénients. Les premières permettent de construire un cluster virtuel, sécurisé, tolérant aux pannes, en partant d'un ensemble de ressources volatiles, non sécurisées et pouvant être dans des domaines administratifs hors de tout contrôle. La plate-forme ainsi obtenue peut être vue comme un système d'ordonnancement de tâches à part entière. Une plate-forme de calcul global est entièrement responsable de l'intégrité et de la sécurité des ressources partagées, mais aussi des informations échangées (codes, résultats). Elle doit intégrer tous les mécanismes permettant d'atteindre ce but car elle ne peut s'appuyer sur les infrastructures ni les ressources utilisées.

Les secondes s'appuient sur des infrastructures existantes, contrôlées et sécurisée, afin de construire un grand cluster disponible selon certaines politiques d'utilisations et d'échanges clairement définies. Elles assurent la sécurité et la standardisation des échanges entre les différents sites ainsi que la virtualisation des systèmes mis en place. L'intégrité et la sécurité des ressources ainsi que la pérénité des informations partagées ne sont pas directement du ressort de la plate-forme car elle s'appuie totalement sur les infrastructures existantes. Par contre, la grille standardise et sécurise les accès aux différents sites.

L'outil Globus [46] est la référence en matière de mise en place de plate-forme institutionnelle. Il est utilisé par plusieurs projets académiques et industriels. Il permet d'accéder à des ressources administrativement distribuées grâce à un ensemble de protocoles et de services standardisés. Une des contributions importantes de Globus est son mécanisme de sécurité, le *Grid Security Infrastructure (GSI)*, qui gère les droits d'accès à la grille. Mais cet outil prend d'autres aspects en charge. Il s'occupe de l'ordonnancement des tâches sur la grille grâce à son *Grid Ressource Allocation Manager (GRAM)* et gère la distribution des fichiers avec son *Global Access to Secondary Storage (GASS)*.

L'efficacité de cette technologie se paie par un coût de mise en œuvre et de gestion qui peut s'avérer prohibitif en fonction des buts à atteindre. D'autre part, Globus ne propose aucune solution quant à la tolérance aux pannes. Le *Heartbeat Monitor* [102] propose une solution concernant ce point précis, mais il n'est pas transparent : les applications doivent explicitement s'enregistrer afin de bénéficier de la tolérance aux pannes.

Certains environnements de calcul, limités à l'intérieur de domaines administratifs, utilisent Globus afin de dépasser les frontières de ces domaines et de les connecter entre eux. Condor [66] propose Condor-G, une passerelle Globus au dessus de Condor permettant de relier différents sites Condor ; Nimrod [6] propose Nimrod/G [5] afin de relier des sites gérés par Nimrod. On voit que ces solutions proposées s'appliquent chacune à un type de plate-forme (Condor-G pour Condor ; Nimrod-G pour Nimrod). Il en résulte une perte de deux points spécifiquement importants pour les grilles, la standardisation et la généricité. Un autre point dommageable est la perte de contrôle des objets gérés. Condor-G, par exemple, ne sait pas gérer globalement les tâches soumises sur les sites inter-connectés ; chaque site gère les tâches qui lui sont soumises et il est compliqué de faire migrer une tâche entre les différents sites.

Ce chapitre présente une proposition permettant de tirer partie de différents systèmes d'ordonnancement installés sur des sites séparés en les connectant entre eux afin d'obtenir une plate-forme globale, indépendante des systèmes installés sur les différents sites. Cette nouvelle proposition est globale, distribuée entre différents domaines administratifs, sécurisée et tolérantes aux pannes. Elle est, d'autre part, plus légère à mettre en place et à gérer que l'outil Globus. Nous illustrons notre propos autour de la technologie Condor, dans la mesure où l'université du Wisconsin, qui propose cette technologie, a été étroitement liée à ces travaux.

D'un point de vue de recherche en informatique, ce mode de fonctionnement permet d'étudier les rapprochements entre deux types de LSDS distincts, les systèmes pair à pair (P2P) et les grilles institutionnelles [52].

4.2 Condor

Condor, un projet de l'université du Wisconsin [67], est un système de gestion de calcul distribué gérant la soumission est la distribution de tâche sur le mode du vol de cycle. Les ressources participantes sont organisées par groupes administratifs, appelés des *pools Condor*. Un pool ne dépasse pas les limites d'un domaine administratif et les ressources le composant ne sont pas gérées par Condor lui même. Pour cela, Condor s'appuie sur l'infrastructure et l'administration existantes quant à la gestion les utilisateurs et de leurs droits, ainsi que tout aspect concernant la sécurité. Par exemple, les tâches sont exécutées en fonction des droits de l'utilisateur tels que définis dans le domaine.

La gestion des ressources utilisées par Condor se fait suivant des règles d'utilisation appelées les *ClassAds* qui permettent de définir les conditions d'utilisation demandées par l'utilisateur. Grâce à ces ClassAds, un utilisateur peut demander un système d'exploitation spécifique pour l'exécution de sa tâche, ou encore une machine avec un certain montant de mémoire vive etc. Condor essaie alors de résoudre les besoins des demandes grâce aux ressource à sa disposition. Les ClassAds permettent aussi aux possesseurs de ressources de définir les conditions d'utilisation ; par exemple le propriétaire d'une ressource de calcul peut souhaiter de n'allouer à la plate-forme Condor que 50% de son CPU disponible ou encore des plages horaires durant lesquelles Condor peut utiliser le CPU (par exemple de 19 heures a 7 heures du matin).

Condor est modélisé selon le paradigme *master-worker*, un concept répandu en informatique parallèle où un service (le *master*) contrôle les *workers*. Toute ressource incluse dans le pool est gérée par un master local qui doit s'assurer de la bonne marche des différents composants de la plate-forme. Ce modèle, où chaque ressource est contrôlée par le master, est connu sous le nom de *modèle push*.

Condor est architecturé autour de cinq composants comme schématisé sur la figure 4.1. Le *schedd* sert d'interface utilisateur à la plate-forme et a la responsabilité de la bonne exécution des tâches qui lui sont soumises. Le *startd* gère les ressources de calcul en fonction de leur ClassAds définissant leur politique d'utilisation. Il autorise et arrête les exécutions sur les ressources en fonction de ces politiques. Le *matchmaker* a la responsabilité de résoudre les besoins des tâches des schedd grâce aux ressources proposées par les startd. Si ce composant est momentanément indisponible, les tâches en cours peuvent continuer leur travaux, mais aucune nouvelle

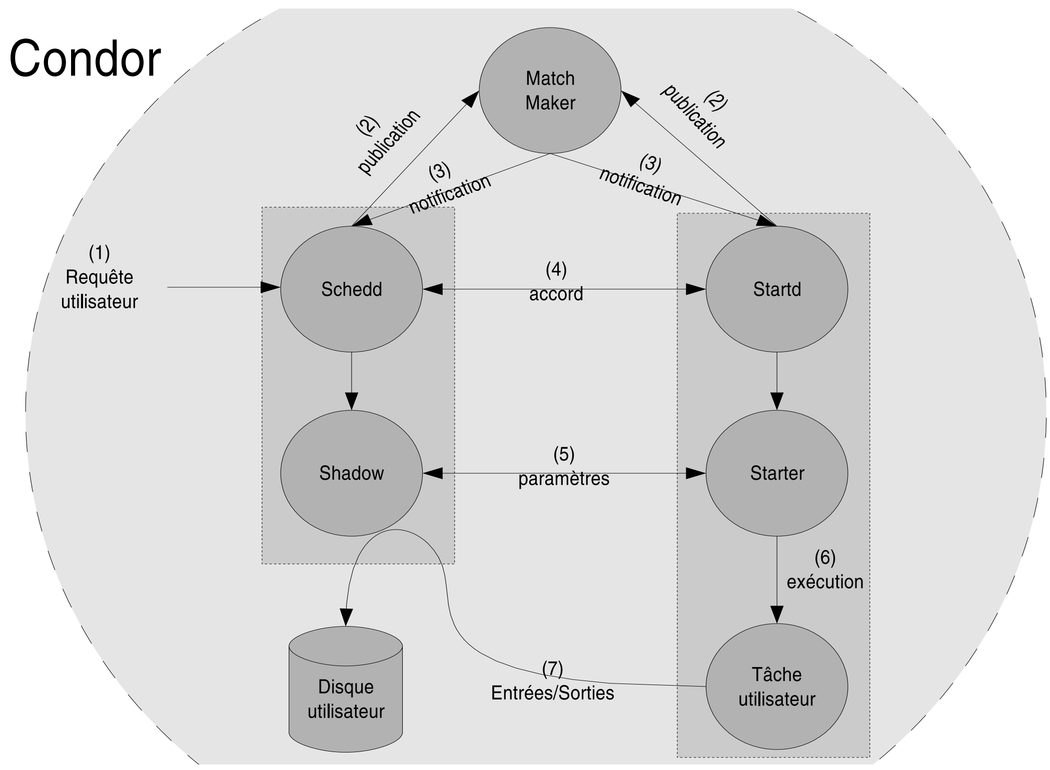

FIG. 4.1 – Condor

tâche ne pourra être ordonnancée par le système. Dès qu'une solution résolvant les besoins des premières tout en respectant les politiques de ces dernières, le matchmaker met en relation le schedd et le startd qui vérifient, ensemble, leur compatibilité. La tâche est finalement exécutée, si les deux parties ont réussi à se mettre d'accord. Deux composants travaillent conjointement à l'exécution des tâches, le *shadow* et le *starter*. Le starter prend la responsabilité de l'exécution de la tâche ; il crée l'environnement nécessaire, lance l'exécution de la tâche et surveille son déroulement. Il a aussi la responsabilité de tenir informés les autres composants de la plate-forme quant à cette exécution. Le starter s'appuie sur le shadow pour savoir comment exécuter les tâches. Ce dernier permet de mettre en place l'environnement d'exécution en fournissant au starter toutes les information nécessaires ; en particulier le shadow est utilisé comme serveur de données afin que la tâche accède à celles qui lui sont utiles.

Le shadow et le starter sont fortement connectés par une liaison TCP qui permet de transférer les binaires, les données et les résultats. Ces composants s'arrêtent immédiatement en cas d'erreur inopportune, ce qui rompt la liaison. Toute rupture détectée sur la liaison entraîne l'arrêt du composant encore en vie. Il est de la responsabilité du schedd et du startd de surveiller l'évolution de leur shadow et starter respectif et de prendre les décisions adéquates à l'état de ces derniers ; il leur revient en particulier de tuer les tâches orphelines et de nettoyer les espaces disques indûment utilisés. En cas d'erreur le schedd a la charge de prendre une décision quant à la tâche qui a généré l'erreur. Il peut essayer de trouver une nouvelle ressource grâce au matchmaker, vouloir réutiliser la même ressource en se reconnectant au même startd, ou même ne plus ordonnancer la tâche.

Afin de mener à terme la bonne exécution d'une tâche, on voit que le modèle de Condor repose sur une interaction forte entre le shadow et le starter qui s'appuie, en particulier, sur un réseau stable puisque la perte de connexion entre ces deux composants implique l'arrêt du calcul de la tâche. Ce modèle s'applique particulièrement bien sur un réseau local, au sein d'un domaine administratif, mais n'est pas compatible avec une exécution à large échelle, sur un réseau comme l'Internet.

4.3 Condor-G

Les pools Condor, dont les ressources sont toutes incluses dans un seul domaine administratif, peuvent dépasser leurs limites administratives grâce à Globus. Cette solution d'ajouter la couche Globus au dessus de Condor s'appele *Condor-G*.

L'environnement *Globus* avec ses outils, tels que le GSI et GridFTP, et ses protocoles tels que le GASS et le GRAM, offre une solution élégante, mais relativement lourde à mettre en place et à gérer, pour connecter des pools Condor.

Grâce à ces outils, un utilisateur peut accéder à un ensemble de pools Condor en s'adressant toujours au même schedd décrit dans la section précédente. La figure 4.2 schématise des accès Condor à travers la grille ; elle montre l'utilisation de Condor comme ordonnanceur local aux différents sites (Cf figure 2.7). Il incombe à l'utilisateur de spécifier le pool choisi. Le schedd instancie un *grid manager* par tâche soumise afin de gérer la tâche sur le site distant en s'identifiant auprès du *Gatekeeper* du site choisi. Les tâches sont transférées grâce aux protocoles *GRAM* (pour déléguer l'ordonnancement des tâches) et *GASS* (pour transférer les paramètres et les résultats) de la couche Globus jusqu'au site choisi par l'utilisateur et finalement prises en charge par le pool Condor visé. Le grid manager est au site distant ce que le shadow est au pool Condor local. Le GateKeeper gère les autorisation d'accès aux différents sites en y déléguant les droits des utilisateurs. Le *job manager* gère les tâches soumises au site distant avec les droits utilisateurs délégués par le GateKeeper. Il est au site distant ce que le *starter* est au pool Condor local. A la différence du couple shadow/starter, le grid manager et le job manager n'ont pas d'interaction forte et travaillent en mode "déconnecté" ; ils se contactent de temps en temps pour échanger des informations et suivre l'évolution de la tâche. Un protocole multi-phase permet d'éviter les tâches orphelines ou perdues (Cf chapitre 3.3.5).

Une grille construite avec Condor-G n'est toutefois ni *globale*, ni *standardisée*. Elle n'est pas globale car elle différencie les ressources des différents sites. Un utilisateur souhaitant soumettre des tâches sur cette plate-forme multi-sites doit spécifier explicitement le site sur lequel il désire faire exécuter ses tâches et n'a pas de solution automatisée de migration de tâche entre sites. Elle n'est pas standardisée car les sites utilisables dans une telle organisation sont obligatoirement des sites Condor.

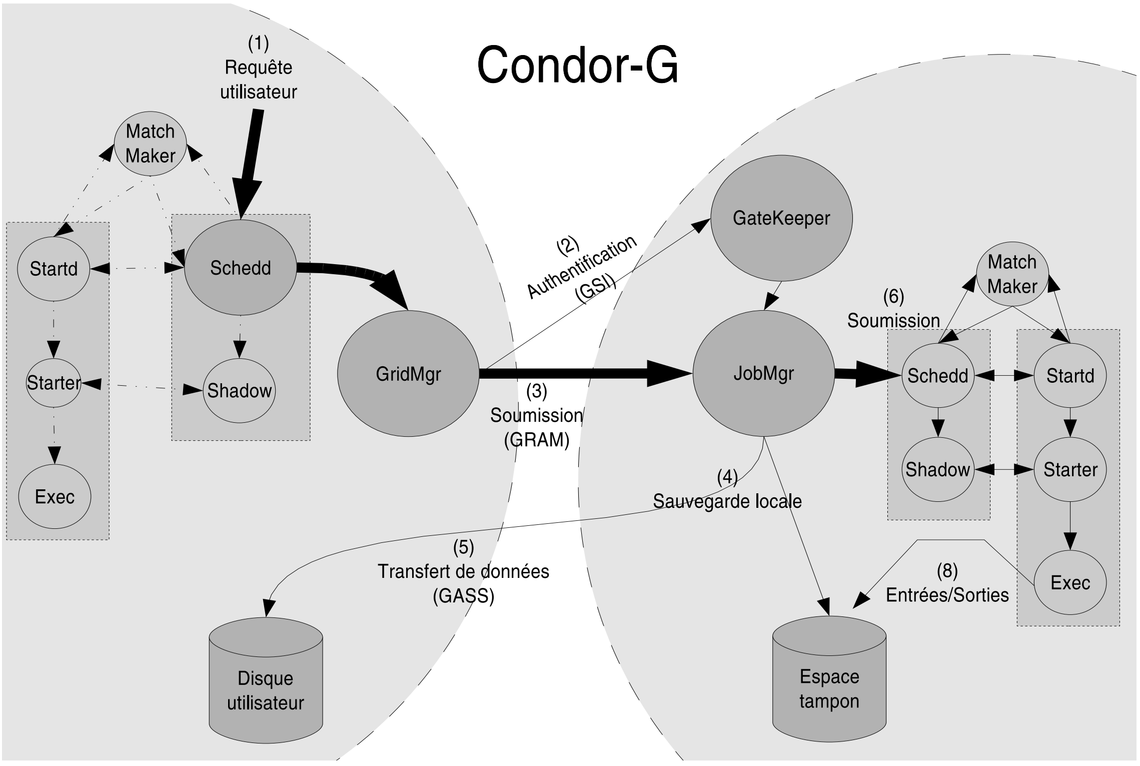

FIG. 4.2 – Condor-G : Condor et Globus

Nous proposons, dans les sections suivantes, d'utiliser XtremWeb comme gestionnaire de ressource (Cf figure 2.7) pour connecter plusieurs pools Condor et montrons qu'une telle installation est à la fois générique et standardisée. Nous allons ici l'appliquer sur Condor, mais elle s'applique exactement de la même manière quelle que soit le type de plate forme à relier. Nous n'avons ici pris comme exemple deux sites Condor, mais on aurait pu mélanger avec des sites utilisant n'importe quel autre ordonnanceur, comme nous le verrons plus bas.

4.4 Condor-XW : une grille globale standardisée

Le modèle de gestion des ressources avec XtremWeb est à l'opposé de celui mis en œuvre par Condor. Avec XtremWeb, les ressources ne sont par sous le contrôle du service de coordination et sont seules à décider de la prise en charge de tâches et de leurs paramètres. Ce modèle, où chaque ressource est indépendante, est connu sous le nom de *modèle pull* (Cf chapitre 3.3.2).

Les tâches sont prises en charge par les services de calcul sur demande expresse de ces derniers. Le service de coordination garde trace de toute connexion et enregistre les caractéristiques matérielles et logicielles des ressources de calcul (CPU,

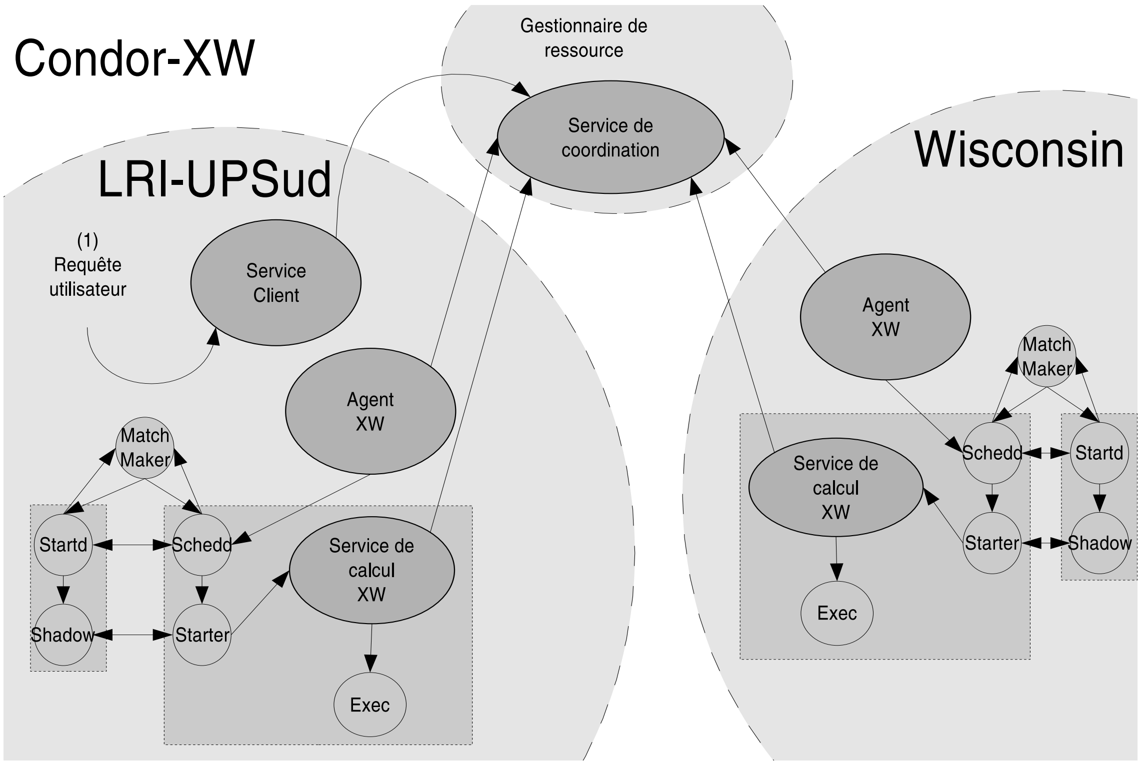

FIG. 4.3 – Condor-XW : Condor et XtremWeb

mémoire, système d'exploitation). Les communications dans XtremWeb sont dites *non connectées*. Contrairement à Condor, il n'y a donc pas de lien fort entre les différents composants de la plate-forme. Celle-ci reste cohérente grâce à un ensemble de signaux que le service de coordination reçoit périodiquement des autres parties (Cf chapitre 3.3.3). Il résulte qu'XtremWeb ne peut que suspecter des fautes là où Condor les détecte. Ceci est dû aux environnements dans lesquels évoluent ces deux plates-formes ; l'Internet pour XtremWeb, les réseaux locaux pour Condor.

Nous souhaitons tirer profit de deux technologies différentes, Condor et XtremWeb, afin de partager des ressources entre différents pools Condor, comme montré dans la figure 4.3.

Pour cela, il faut d'une part distribuer le service de calcul d'XtremWeb sur les pools Condor possédant les ressources à partager (détaillé dans la section 4.4.1) et d'autre part coordonner ces ressources agrégées sur la plate-forme ainsi construite (section 4.4.2). Le troisième point, détaillé dans la section 4.4.3, est d'assurer la sécurité d'une telle plate-forme. Enfin, la section 4.4.4 est relative à la tolérance aux pannes.

En résumé, nous allons voir que nous pouvons utiliser les pools Condor pour déployer les services de calcul XtremWeb qui exécuteront les tâches utilisateurs. Le service de coordination XtremWeb est alors utilisé afin de coordonner la plate-forme et d'ordonnancer les tâches utilisateurs. Cette plate-forme ainsi obtenue est plus légère qu'une mise en oeuvre de Globus pour un résultat similaire, tire partie des avantages des deux types de plate-forme (Condor et Xtremweb) et apporte une migration automatique de tâches entre les différents sites. Cette dernière caractéristique est jusqu'à présent inédite.

4.4.1 Des services embarqués pour la standardisation des ressources

Déployer les services de calcul XtremWeb est le premier point à régler afin de pouvoir réunir les ressources de différents pools Condor. Pour ce faire, il est nécessaire qu'ils puissent être gérés par Condor tout en suivant la politique de sécurité de ce dernier afin de ne pas inquiéter son intégrité.

Les services de calcul utilisent les librairies «standards» Java telles que les connexions sécurisées grâce à *SSL* ou encore les systèmes de journalisation. Le service de calcul XtremWeb, en tant que tâche Condor, ne peut faire aucun présupposé quant à la plate forme sur laquelle il va s'exécuter. Il est nécessaire que le service de calcul XtremWeb vienne comme un package auto suffisant, incluant tout ce qui lui est nécessaire pour une bonne exécution, telles les librairies Java dont il a besoin. Toutefois, le service de calcul ne peut faire l'impasse sur deux pré-requis. Le premier est qu'il a besoin d'un espace disque temporaire afin de pouvoir télécharger les tâches qu'il va lui même exécuter ; le second est qu'il a besoin d'une connexion Internet en sortie afin de pouvoir contacter le coordinateur XtremWeb.

Comme nous l'avons dit, l'utilisation des ressources d'un pool Condor nécessite des droits ; nous devons avoir les droits nécessaires pour distribuer le service de calcul XtremWeb sur les pools choisis. Au LRI, nous avons bien sûr les droits d'utiliser les quarante machines de notre pool Condor. Il nous faut obtenir les mêmes droits à l'université du Wisconsin, où un pool Condor de neuf cents machines est installé. Nous pouvons soit déléguer les droits à un administrateur local pour installer notre service de calcul, soit lui demander d'installer un agent XtremWeb de soumission de tâche Condor afin de faire démarrer les services de calcul XtremWeb sur les ressources Condor. Ces deux pools ne sont pas dédiés à nos expérimentations et sont largement utilisés par les utilisateurs locaux. Notre service est donc distribué sur ces pools selon les politiques locales à chaque ressource de ces pools.

XtremWeb, dans cette configuration, ne définit pas de politique d'activation puisque ce travail est déjà fait au niveau de Condor grâce à ces *ClassAds*. Le *master* Condor ne voit aucun résultat dans la mesure où le service de calcul XtremWeb n'en génère aucun. De fait, Condor ne peut voir ni vérifier ce que sont les tâches puisqu'elles sont gérées par XtremWeb. Condor ne voit que le service de calcul d'XtremWeb. Les tâches et leurs résultats sont entièrement gérés par le service de coordination d'XtremWeb.

Afin de ne pas gêner le pool Condor en monopolisant inopportunément des ressources de calcul, les services de calcul XtremWeb sont configurés pour s'arrêter automatiquement dès qu'ils ont calculé une tâche. Avec cette limitation, il convient de s'assurer que toutes les tâches soumises seront effectivement exécutées. En effet, des services de calcul peuvent être interrompus en fonction de la politique d'activation des ressources. Il convient donc de demander plus de ressources aux pools Condor que de tâches soumises afin de s'assurer de la bonne exécution des tâches prises en charge ; il faut aussi libérer le reliquat de ressources à la fin de l'exécution de la dernière tâche. Par précaution les services de calcul XtremWeb sont donc configurés afin de libérer la ressource Condor s'il ne reçoivent pas de tâches après un délai jugé suffisant. Si ce délai a été mal estimé et qu'il reste des tâches à exécuter mais plus de ressources disponibles, l'agent XtremWeb en redemanderait.

4.4.2 Le service de coordination XtremWeb comme gestionnaire global de ressources

Résoudre les problèmes de connexion de différents domaines administratifs protégés par des pare-feux peut être fait grâce au service de coordination d'XtremWeb. Il existe bien sûr, d'autres solutions. Par exemple, Nimrod propose Nimrod/G [5], Condor propose Condor-G ; ces deux solutions s'appuient sur Globus qui est contraignant à installer et à administrer. Il faut installer le *GSI* pour l'authentification, le *GRAM* pour la gestion des tâches, le *GASS* pour la gestion des espaces de stockage. Ce dernier est utilisé par exemple par Condor-G pour transférer les binaires et les paramètres de tâches à exécuter. Nous allons montrer ici qu'XtremWeb est une solution tout aussi efficace mais bien plus légère pour arriver au même résultat.

Les protocoles d'XtremWeb permettent de résoudre les problèmes d'accès à travers les pare-feux en utilisant TCP, qui est un protocole symétrique (Cf chapitre 3.3.4). Le service de coordination d'XtremWeb n'implique aucun changement au niveau des pools Condor ; il peut être mis en place n'importe où sur Internet pour peu qu'il soit accessible, dans ou hors des domaines administratifs des pools Condor. Une fois que ce service mis en route, la plate-forme de calcul globale est prête.

La figure 4.3 schématise l'activation et l'utilisation de la plate-forme globale obtenue. L'utilisateur soumet ses tâches (en (1), (2) et (3)), comme d'habitude avec XtremWeb. Le service de coordination, conjointement aux agents présents sur les différents sites proposant des pools Condor, alloue des ressources de calcul proposés par les pools (5). Comme dit précédemment, on réserve plus de ressources que de tâches pour être sûr d'avoir assez de ressources malgré leur politique d'activation. Les ordonnanceurs locaux aux sites allouent les ressources et déploient le services de calcul d'XtremWeb qui se mettent en relation (6) avec le service de coordination qui peut ordonnancer les tâches (7) qui sont finalement exécuter (8), (9) et (10).

Cette manière de faire est indépendante de Condor. On aurait tout aussi bien pu utiliser d'autres systèmes d'ordonnancement tels que PBS ou LSF. Seul l'agent

XtremWeb est dépendant du système d'ordonnancement. Cet agent peut d'ailleurs prendre plusieurs formes ; ce peut être un GateKeeper de Globus ou tout autre service local au site proposant les ressources de calcul. On peut même se contenter, dans le plus trivial des cas, d'un simple accès utilisateur au site qui permet d'accéder à l'ordonnanceur avec des outils de type cluster (client de soumission en ligne de commande ou autre).

4.4.3 La sécurité

Il convient d'assurer la sécurité des pools Condor ainsi connectés. XtremWeb doit prendre en charge l'authentification des utilisateurs, l'intégrité des ressources participantes ainsi que la protection des tâches, de leurs paramètres et de leurs résultats. La sécurité dans XtremWeb a été présentée au chapitre 3.2.2 ; nous en rappelons ici les grandes lignes et leur utilisation pour partager des ressources distribuées dans le cadre d'échanges entre pools Condor.

L'authentification des utilisateurs et la journalisation des exécutions. Le service de coordination gère une liste des utilisateurs autorisés à utiliser la plate-forme grâce à des *ACLs*. Il est de la responsabilité des administrateurs d'autoriser ou de révoquer les utilisateurs auprès du service de coordination d'XtremWeb. Toute connexion est initiée par un échange de clés permettant de vérifier les droits des utilisateurs. Une fois les vérifications faites, le service de coordination prend le rôle de mandataire pour l'utilisateur qui lui délègue ses droits afin d'interagir avec les différents sites. Les différentes actions sont toutes journalisées sur chaque partie de la plate-forme. Ainsi, ce suivi précis permet de prendre les bonnes décisions en cas de problèmes de sécurité ; un utilisateur essayant de corrompre la plate-forme, à quelque niveau que ce soit, serait immédiatement et automatiquement révoqué.

L'implémentation actuelle n'utilise pas les *certificats* et présente un certain degrés de risque relativement à des attaques du type *Man in the Middle*. Ces risques sont toutefois limités : 1) ces attaques devraient provenir de l'intérieur des clusters Condor eux mêmes et restent donc un problème plus général de sécurisation des clusters ; 2) les différents services distants d'XtremWeb possèdent la clé publique du service de coordination qui peut ne jamais être échangée (et donc être vulnérable à ce type d'attaque) si l'installation des binaires des tâches à exécuter peut être faite de manière entièrement sécurisée.

Un système de certificat, tel que GSI dans Globus, devrait être inclus, à terme, dans XtremWeb afin d'augmenter la sécurité des ressources et des utilisateurs.

La protection des applications, des paramètres et des résultats. La couche message d'XtremWeb, grâce à des échanges de clés et des communications encryptées dans des tunnels *SSL*, assure la sécurité des applications, des tâches et de leurs paramètres. Ces mécanismes empêchent d'éventuels participants malicieux d'intercepter les communications et de s'introduire dans la plate-forme déployée. Elle permet aussi d'éviter les erreurs de manipulation en interdisant les connections de services distants non autorisés.

L'intégrité des ressources. Si toutefois un participants malicieux parvenait à s'introduire dans la plate-forme, il ne pourrait exécuter de tâches mal intentionnées car les tâches sont exécutées au sein d'un bac à sable (*sand-boxing*) [10, 7, 56] sur les services de calcul, que ce soient des tâches natives ou Java, ces dernières s'exécutant toujours dans une machine virtuelle selon une politique de sécurité [57]. XtremWeb utilise ces politiques de sécurité à deux niveaux : au niveau du service de calcul lui même et aux niveaux des tâches exécutées. Le coût des bacs à sable n'étant pas nul [22], la sécurité peut être allégée en fonction du niveau de sécurité des ressources elles mêmes.

Le bac à sable utilisé pour le service de calcul afin d'exécuter les applications binaires permet de configurer l'accès à la machine par le binaire : cela va de l'usage de la mémoire aux accès disques.

4.4.4 La qualité de service

Plusieurs mécanismes transparents de tolérance aux pannes sont mis en œuvre dans XtremWeb afin de prendre en charge les pannes des services distribués ainsi que du service de coordination, dans le but de maintenir la cohérence de la plate-forme. Ces mécanismes ont été présentés en détail au chapitre 3.3.3. Nous en rappelons ici les grandes lignes et leurs utilisations.

La soumission de tâches et la récupération des résultats se font par transaction. Avant de soumettre une tâche, le service client vérifie l'état de ses tâches déjà soumises ; ceci lui permet de s'assurer de ne pas soumettre deux fois la même tâche (ce cas pourrait arriver suite à une panne par exemple). Le service de coordination centralise la gestion des résultats et les fournit aux clients à leurs demandes ; il garde une copie des résultats jusqu'à ce que le client lui demande de les effacer définitivement. Ainsi la gestion des résultats est déléguée au niveau du service client, assurant leur consistance jusqu'à l'utilisateur.

L'organisation de l'exécution des tâches est assurée par le coordinateur grâce à un signal de *poul* périodiquement envoyé par les service de calcul au service de coordination. Un service de calcul n'envoyant pas ce signal pendant un certain temps est considéré comme fautif ; s'il avait la charge de l'exécution d'une tâche, celle ci est susceptible d'être réordonnancée sur une autre ressource. Le service de coordination prend les décisions nécessaires afin qu'une même tâche ne s'exécute pas simultanément sur deux ressources différentes. Par exemple lorsqu'un service de calcul « disparaît » pendant un certain temps puis revient, il est probable que la tâche dont il avait la charge ait été attribuée à un autre service de calcul ; le service coordination lui demande alors de ne plus s'occuper de cette tâche et lui en alloue éventuellement une autre.

Les services distribués et le service coordination d'XtremWeb sauvegardent toute information, tout message échangé. Au démarrage ils se remettent dans l'état dans lequel ils se trouvaient à la dernière sauvegarde. Ils se synchronisent alors entre eux. Ces mécanismes assurent un état cohérent à la plate-forme, même après une faute de n'importe quel de ses services.

Tous ces mécanismes sont totalement transparents pour les tâches et les utilisateurs. Ces derniers ne voient que les états de leurs tâches et de leurs résultats. Grâce à eux la plate-forme intègre une migration transparente des tâches entre sites administratifs. Un utilisateur souhaitant soumettre des tâches sur cette plate-forme multi-sites ne spécifient pas explicitement de site sur lequel faire exécuter ses tâches ; il les soumet à la plate-forme qui décide du site choisi et peut, si besoin est, faire migrer les tâches d'un site à un autre afin que l'utilisateur puisse finalement obtenir ses résultats.

4.5 Expérimentation et mise en oeuvre : structure, dynamique et fonctions des protéines

Les applications multi-paramètres représentent un cas parfaitement adapté aux déploiements sur les grilles. Ce type d'application est typique des gros calculs de simulation dans la recherche et l'industrie ; leur exécution se fait en général sur des clusters de machines contrôlées par un système de batch centralisé. Pour des questions de sécurité ou d'accès aux données, l'utilisation de machines volontaires anonymes, comme dans Seti@Home, peut se révéler non souhaitable. On peut, par contre, souhaiter agréger les machines volontaires mais connues (ordinateurs personnels administratifs, salle de TP d'une université etc.) et les clusters d'une organisation donnée afin de construire un cluster virtuel à travers des domaines administratifs sécurisés.

Nous avons soumis une application de biochimie moléculaire sur la plate-forme ainsi obtenue afin de valider les mécanismes mis en œuvre. Cette application est utilisée à l'IBBMC pour comprendre la stabilité et l'activité des protéines en fonction de leur structure. La modélisation moléculaire est utilisée pour explorer les possibilités de conformations des macro-molécules.

L'application consiste à lancer un grand nombre de tâches indépendantes avec des jeux de paramètres différents. Elle est composée de quatre étapes. La première est de générer n conformations de départ ; la seconde exécute m simulations moléculaires pour chaque conformations, ce qui génère $(n * m)$ tâches. La troisième étape récupère les résultats et la dernière calcule les profils d'énergie.

L'expérimentation consiste en 320 tâches. L'exécution sur une machine typique (AMD 1.8GHz) nécessite 43 heures et 44 minutes de calcul. Cette valeur est gardée comme référence pour les comparaison de performances. Le déploiement est fait sur

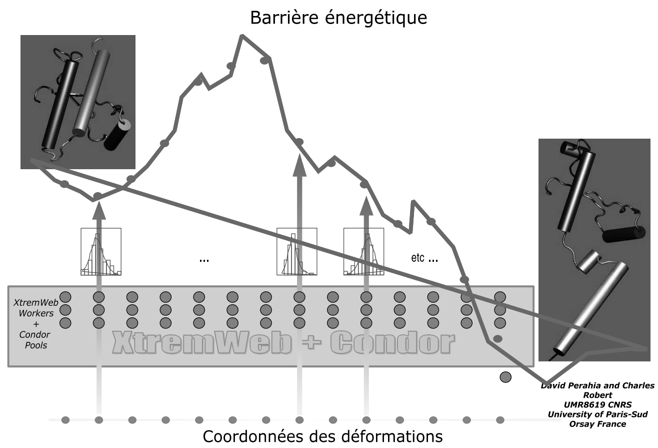

FIG. 4.4 – Application moléculaire.

trois configurations différentes, toutes basées sur des pools Condor connectés entre eux par un coordinateur XtremWeb.

Le tableau 4.1 présente les types de machines pour chaque expérience.

	Wisc		LRI	
Exp.	600Mhz	900Mhz	1800Mhz	Total
`LRI`			30	30
`Wisc`	61	104		165
`W+LRI`	50	73	9	132

TAB. 4.1 – La participation en CPU des différents pools Condor

La figure 4.5 présente le nombre de CPUs utilisés pour chaque expérience.

Chaque expérience présente le même type de courbe à trois phases. La première est la montée en puissance de la prise en charge des tâches ; c'est le moment où les tâches sont en cours de distribution sur les services de calcul. La seconde phase est relativement stable (la courbe est relativement plate) ; elle correspond à l'utilisation optimale de la plate-forme, celle où les ressources sont toutes utilisées. La dernière phase est descendante ; c'est celle de l'arrivée des derniers résultats où il y a moins de tâches à exécuter que de services de calcul disponibles. L'algorithme d'ordonnancement est « naïf » et organise la prise en charges tâches au fur et à mesure qu'elles arrivent sans tenir d'autres facteurs que leur ordre d'arrivée. Par exemple, il ne tient pas compte de la vitesse du CPU des ressources de calcul disponibles ni de leur taux de disponibilité. Les dernières exécutions sont donc celles qui se sont exécutées sur les machines les plus lentes. La figure 4.6 montre, pour chaque expérience, les temps nécessaires à l'obtention de tous les résultats.

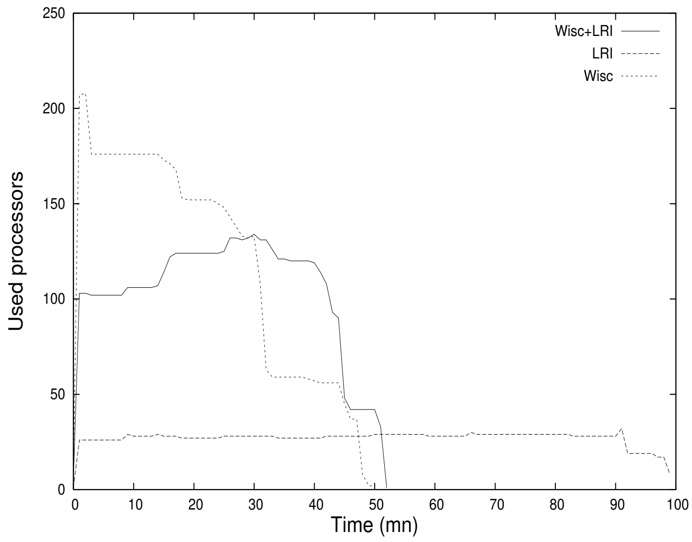

FIG. 4.5 – Utilisation des CPU par expériences.

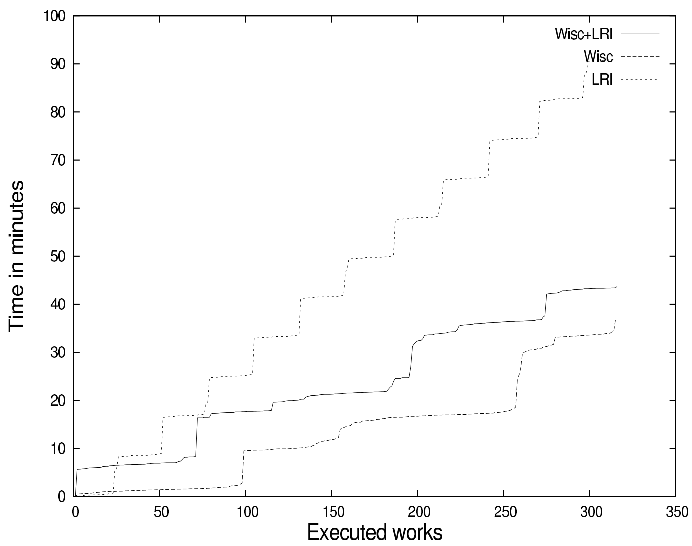

FIG. 4.6 – Dates d'obtention des résultats par plate-forme.

On voit très clairement les différences de puissance disponible entre les expériences.

La figure 4.7 montre les temps d'exécution des tâches par ordre décroissant pour toutes les expériences. Plusieurs types d'exécution apparaissent clairement en fonction du type de CPU utilisé. Le pool Condor du LRI est composé d'un seul type de machine, des AMD 1.8GHz, alors que le pool du Wisonsin est composé de machines à 800Mhz et 900Mhz.

La figure 4.8 montre la bande passante utilisée en utilisant les deux pools Condor agrégés grâce à XtremWeb.

Enfin, la figure 4.9 montre la distribution des tâches en fonction des machines.

La première configuration utilise trente machines au LRI. Tous sont à base d'AMD 1.8GHz ; dix sont des bi-processeurs, les autres des mono-processeur. Cette plate-forme est une plate-forme de production pour les utilisateurs du laboratoire ; nos expériences partagent donc les machines avec ceux ci. La figure 4.6 montre un temps de calcul de une heure, trente minutes et cinquante quatre secondes. Le gain de puissance n'est donc que de l'ordre de vingt neuf ; ceci est dû au partage des machines avec les utilisateurs. Les temps de calcul par tâche sont uniformes puisque les machines sont toutes configurées de la même manière. La courbe montre qu'un plateau est atteint au plus fort de l'utilisation de la plate-forme ; cela s'explique par le fait que les services de calcul font pratiquement tous simultanément leurs opérations (récupération des tâches, retour des résultats) ; c'est ce que l'on peut voir sur la figure 4.7. Ce point est confirmé par la figure 4.9 qui montre une bonne distribution des tâches par machine.

La seconde configuration est composée de cent soixante cinq services de calcul fournis par l'université du Wisconsin ; 61 machines sont des *Pentium III* à 600Mhz, le reste à 900Mhz. Toutes sont des bi-processeurs. La figure 4.6 montre un temps d'exécution pour cette expérience de trente six minutes et quarante et une secondes ; le gain en temps de calcul est donc de l'ordre de quatre-vingt huit. On voit sur cette courbe aussi des plateaux qui sont dûs aux deux types de machines présentes dans cette plate-forme et qui ont donc des comportements différents (figure 4.7). La distribution des tâches est, là aussi, régulière (figure 4.9).

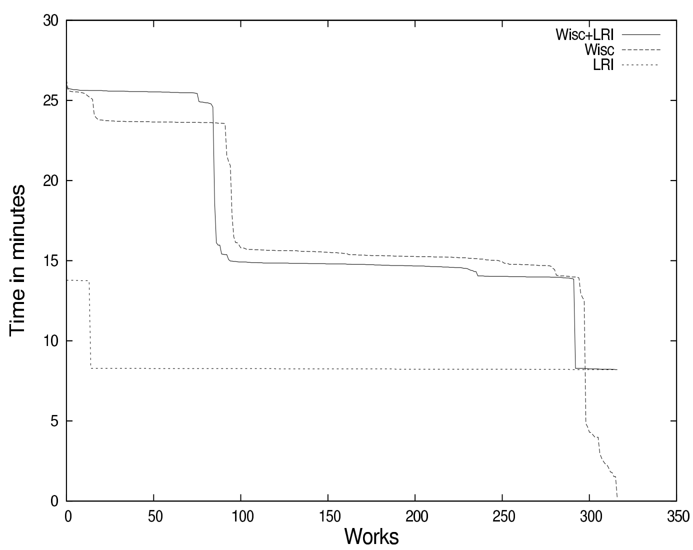

FIG. 4.7 – Temps d'exécution par tâche.

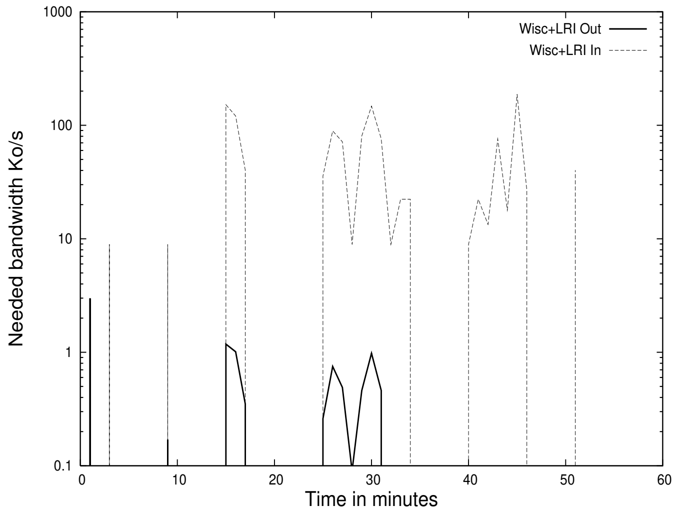

FIG. 4.8 – Utilisation de la bande passante.

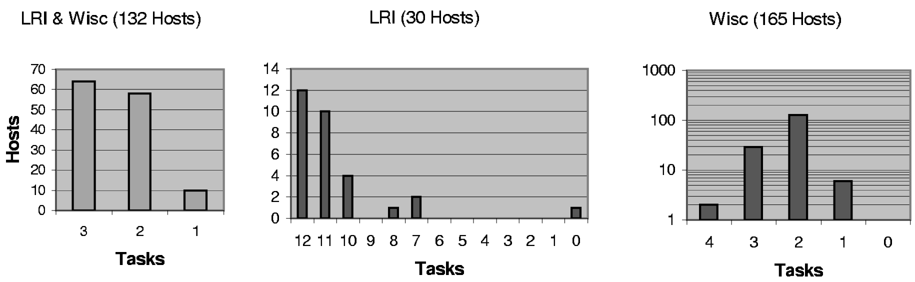

FIG. 4.9 – Distribution de tâche de bio-chimie.

La troisième configuration utilise les machines du LRI et du Wisconsin. Cette fois seules neuf machines étaient disponibles au LRI (5 mono-processeur et 4 bi-processeur) et cent vingt trois au Wisconsin (toutes sont des bi-processeurs ; 50 à 500Mhz et 72 à 900Mhz). Le gain baisse de vingt pour cent par rapport à la seconde expérience, conformément à la baisse de puissance de calcul disponible (figure 4.6 et 4.7). Les courbes montrent une plus grande irrégularité par rapport aux configurations précédentes. Ceci est dû à une plus grande hétérogénéité de cette configuration.

La figure 4.8 représente la bande passante utilisée par cette expérience. La ligne en **gras**, sous titrée *Wisc+LRI Out* représente le trafic réseau du coordinateur vers les services de calcul (les binaires et les paramètres de tâches). La première ligne verticale correspond aux téléchargements des binaires (qui ne le seront plus ensuite puisqu'ils sont dès lors en cache sur les services de calcul et qu'on ne lance des tâches que pour une seule application) et des paramètres des premières tâches. La ligne plus fine, sous titrée *Wisc+LRI In* montre le trafic réseau des services de calcul vers le coordinateur (les résultats). On voit que la partie droite de la figure ne comporte que des lignes fines, à partir de la trente deuxième minute. Il n'y a en effet plus de tâches à exécuter à partir de ce moment là, il n'y a donc plus de trafic réseau du coordinateur vers les services de calcul. Par contre, de la trente deuxième minute à la fin de l'expérience, les services de calcul finissent d'exécuter les dernières tâches et continuent donc d'en retourner les résultats. Les lignes fines représentent ce trafic des services de calcul vers le coordinateur.

La figure 4.9 montre une bonne répartition des tâches pour cette expérience aussi. Les tâches sont régulièrement distribuées sur les deux sites utilisés, le LRI et le Wisconsin, modulo l'hétérogénéité des machines en présence.

4.6 Conclusions

Condor et XtremWeb savent gérer des plates-formes hétérogènes et prendre en charge des requêtes utilisateurs en terme de services. Le premier gère des ressources de type cluster en ce sens qu'elles doivent appartenir à un même domaine administratif. Le second applique les mêmes principes sur l'Internet, hors de tout domaine. On peut vouloir généraliser l'utilisation de ressources de type cluster comme celles gérées par Condor en dépassant les limitations des domaines administratifs. Un première solution, Condor-G permet de former des grilles inter-domaines en inter-connectant de manière sécurisée des pools Condor autonomes. Toutefois cette solution ne résout pas les problèmes globalement ; elle ne sait ni prendre en charge l'hétérogénéité inter-sites (elle ne permet pas de gérer les ressources individuellement), ni considérer un ordonnancement global.

Ce chapitre a présenté une proposition légère, standardisée et globale d'agrégation de ressources de calcul distribuées sur des domaines administratifs séparés afin d'ordonnancer des applications multi-paramétriques grâce à un gestionnaire de ressource centralisé. Cette proposition répond à des besoins d'utilisations académiques et industrielles en permettant d'aggréger un grand nombre de ressource grâce à une délégation de droits adéquats.

Cette architecture hiérarchique basée, d'une part, sur des pools Condor afin d'agréger et de déployer des ressources de calcul, et sur l'utilisation du service de coordination d'XtremWeb, d'autre part, pour coordonner la distribution des tâches a permis d'uniformiser la plate-forme finale en faisant abstraction des spécificités de chaque ordonnanceur. Elle a aussi réussi à gérer les services de calcul individuellement et à ordonnancer les tâches globalement, sur toutes les ressources disponibles, indépendamment de leur situation géographique et administrative.

Cette architecture répond aux besoins des grilles en termes de sécurité et de tolérance aux pannes et de qualité de service. La sécurité des ressources de calcul est assurée par le système hôte (ici, Condor) et la sécurité des communications est assurée par les protocoles mis en œuvre par XtremWeb (communications cryptées, gestion des autorisations). La tolérance aux pannes de cette plate-forme étant assurée par XtremWeb (transactions, reprise sur erreur).

Chapitre 5

Intégration de services standards sur une plate-forme GC

Résumé. Dans ce chapitre nous montrons les capacités de transport de protocole standard de la plate-forme XtremWeb. Nous prenons comme exemple le protocole *SunRPC* pour illustrer notre propos.

Nous commençons par introduire le protocole *SunRPC* lui même en expliquant particulièrement les mécanismes de mise en relation et de sécurité. Nous présentons ensuite *NFS*, un système de partage de données largement répandu. Basé sur SunRPC, nous avons choisi NFS afin de montrer la mise en œuvre de notre proposition pour partager des données entre différents domaines administratifs. Nous détaillons les mécanismes de NFS et nous mettons tout particulièrement en évidence les problèmes de sécurité liés aux partages de volumes inter-domaines.

Nous montrons que l'utilisation de notre plate-forme pour partager des volumes NFS entre différents domaines permet de virtualiser les services NFS, d'une part, ainsi que d'augmenter la sécurité des données partagées, d'autre part, grâce aux capacités de sécurité et de qualité de service de notre plate-forme : définition d'utilisateur inter-domaines et journalisation des messages transportés.

5.1 Introduction

Les systèmes distribués à large échelle (LSDS) tels que les grilles [49], les grilles de PC [65, 26, 14, 23], les plates-formes de calcul global [13] et les plates-formes P2P [50] agrègent un grand nombre de ressources afin d'exécuter des calculs de manière distribuée. Les ressources sont souvent agrégées parmi différents domaines administratifs ayant chacun leurs propres politiques de sécurité, leurs propres configurations réseau (pare-feux, translation d'adresses, proxies...) ainsi que leurs gestions d'authentification et d'accès aux différents services. La construction de LSDS nécessite donc un niveau au dessus des ressources afin d'en standardiser les déploiements, les accès et la gestion, et de sécuriser les services et les échanges au sein de la plate-forme obtenue.

L'architecture des applications scientifiques distribuables et distribuées sur de telles plates-formes est basée sur la notion de services [45], des entités connectées au réseau et définies par un ensemble d'interfaces et de protocoles. De nouveaux standards sont apparus en ce sens, comme OGSA [48], GridRPC [97] ou OmniRPC [92].

D'un autre côté, il existe un nombre considérable de services -les bases de données distribuées, les applications de monitoring, les systèmes de fichiers distribués etc.- qui n'ont pas été initialement prévus pour être accessibles à travers une grille et qui ne proposent ni les interfaces, ni les protocoles nécessaires à leur virtualisation. Se pose alors la question de savoir comment virtualiser ces services et comment exécuter sur une plate-forme de calcul global des applications utilisant de tels services non virtualités. Dans le reste de ce chapitre nous dénommerons ce type de service comme les services *standards*.

Une solution pour dépasser les limites des domaines administratifs consiste à mettre en œuvre des infrastructures dédiées telles que les réseaux privés (*VPN*) sécurisés avec *IPSec* ou *Vtune*, ou le déploiement de relais entre ces domaines afin de permettre leur interconnexion. Cependant, ces solutions sont lourdes à mettre en place, nécessitent des interventions administratives ou même des matériels spécifiques. La spécificité des ressources partagées et utilisées dans le cadre de LSDS, leur volatilité et leur indépendance rend ces solutions inapplicables.

Une nouvelle approche présentée dans ce chapitre permet de démontrer que l'utilisation d'une plate-forme de calcul global comme XtremWeb [23] facilite le déploiement d'applications standards de type client/serveur au delà des limites des sites administratifs. Nous démontrons que cette plate-forme peut être utilisée comme couche de transport transparente et sécurisée aux applications utilisant le protocole *SunRPC* [105] entre machines distribuées sur des domaines administrativement séparés. Certaines applications clés, standardisées et largement diffusées, utilisent ce protocole : *YP (Yellow Pages)* pour le partage d'informations entre ressources informatiques, *NFS (Network File System)* pour le partage de fichiers distants. Nous illustrons la faisabilité de cette approche avec le système de fichiers distribués NFS afin d'accéder à des fichiers depuis des espaces de stockage distribués (des *volumes*) hors de leurs organisations, même virtuelles, et donc, à priori, inaccessibles par quelque moyen que ce soit. Virtuellement, cette approche permet aux applicatifs nécessitant de tels services d'être déployables et utilisables sur une plate-forme de calcul global proposant des ressources n'importe où sur l'Internet, y compris hors de tout domaine administratif. Ainsi, des applications nécessitant des structures de données précises, issues de montages de volumes NFS spécifiques, peuvent être distribuées sur une telle plate-forme et être exécutées de la même manière qu'elles le seraient à l'intérieur d'un site administratif clos. Grâce à notre proposition, ces applications peuvent bénéficier des services du site administratif, hors de celui ci.

Une plate-forme de calcul global est conçue et dévolue aux applications multi-paramétriques dont les différentes tâches soumises sont indépendantes les unes des autres et qui peuvent être exécutées par n'importe quel service de calcul. Nous sommes donc amener à considérer les services standards comme une application multi-paramétrique répondant à ces critères. D'autre part, la sécurité est indépendante des ressources utilisées ; elle est mise en œuvre par la plate-forme elle même grâce à un ensemble de protocoles de sécurisation au niveau de chaque service, ainsi qu'au niveau des messages qu'ils échangent.

Nous allons montrer ici qu'il est possible de distribuer des services standards sur XtremWeb sans toutefois détourner cette plate-forme de son but premier : ordonnancer des tâches d'applications multi-paramétriques sur des ressources distribuées hautement volatiles. Nous détaillons les améliorations qui ont dues être apportées à la plate-forme afin d'ajouter la prise en charge de services client/serveur. Notre proposition propose le transport des messages de ces services, hors de tout contexte administratif, ce qui est normalement impossible, même à travers les grilles. Nous démontrons que la distribution de services standards peut être utilisée dans un environnement sécurisé dans la mesure où les services standards sont eux mêmes sécurisés. Nous démontrons que la plate-forme en elle même ne détériore pas la sécurité de ces services.

5.2 Appel de procédures à distance

Le concept d'appel de procédure à distance (*RPC* [105]) est utilisé depuis longtemps en informatique distribuée dans la mesure où il offre une solution simple et facile à mettre en œuvre pour permettre des communications entre composants distants. Nombre d'applications scientifiques et industrielles sont implémentées suivant le paradigme d'appel de procédures distantes concurrentes pour implémenter des logiciels client/serveur. La partie cliente de ce type d'application lance des appels RPC non bloquants à différents serveurs. Ceci revient à exécuter des codes en parallèles. Le client suit et contrôle l'exécution de ses appels et détecte les eventuelles erreurs dûes à des serveurs défaillants. Le paradigme RPC a acquis une grande popularité dans le cadre des grilles et plusieurs environnement ont été proposés. CondorMW [58] a été parmi les premiers environnements de programmation pour ce type d'application sur une grille de PC, mais il en existe d'autres tout autant fameux, comme GridRPC [97] et OmniRPC [92].

GridRPC est une proposition pour standardiser les mécanismes RPC sur les grilles. Deux plates-formes, NetSolve [24] et Ninf [93], proposent des implémentations de ce standard. Ces systèmes ne peuvent prendre en charge des environnements hautement volatiles car ils n'intègrent pas de mécanismes de tolérance aux pannes. Notons que des travaux ont été menés en ce sens avec NetSolve [82]. OmniRPC, pour sa part, est une autre solution proposant un environnement de programmation pour les grilles.

L'implémentation RPC de XtremWeb (*XWRPC* [34]) propose un client qui transpose automatiquement les appels RPC en messages pour le service de coordination. Il transpose de la même manière les résultats obtenus sur la plate-forme XtremWeb afin de les retourner dans le format attendu au client RPC. XWRPC implémente les appels RPC bloquants et non bloquants ; il implémente en conséquence les méthodes *Wait* qui sont les méthodes standards RPC pour la synchronisation.

La plupart de ces travaux se sont concentrés sur le développement de mécanismes RPC haute performance et sur les RPC *gridifiés*. Peu d'attention a été portée sur la tolérance aux pannes dans le contexte des grilles et des systèmes de calcul global, bien que ce champ de recherche ait été largement étudié dans le contexte d'objets distribués, notamment *Corba*. FT-Corba, par exemple, est une solution élégante architecturée sur le modèle trois tiers, proche de celle présentée et implémentée dans XtremWeb, où le tiers d'interposition (le service de coordination) est la clé de voûte du système de tolérance aux pannes [15]. Cette proposition, FT-Corba, s'appuie sur l'hypothèse que le tiers d'interposition est déployé dans un environnement pseudo synchrone. Cette supposition n'est pas malheureusement pas valable dans des systèmes distribués à grande échelle (*LSDS*), comme nous l'avons expliqué au chapitre 2.3.4.

5.3 Le protocole Sun RPC

Le protocole *SunRPC* [105], aussi connu sous le nom de *ONC* (*Open Network Computing*), est un protocole de communications client/serveur proposé par la compagnie *Sun*. Ce protocole s'appuie sur la couche *IP* et sur les protocoles *TCP* et *UDP* grâce auxquels il résout les mises en relation distantes. SunRPC est une couche d'abstraction au dessus de la couche de transport : il cache la couche de transport ainsi que le port de communication. Il doit être considéré comme un service de mise en relation. Il n'offre aucun autre service ; en particulier, il n'offre aucun mécanisme de tolérance aux pannes ni de qualité de service. Les applications et services se basant sur SunRPC doivent donc implémenter leurs propres mécanismes relatifs à ces problématiques. Par exemple, la non réception de réponse à un appel distant ne signifie en aucune manière que cet appel n'ait pas été effectué ; il se peut simplement que la réponse n'ait pas réussi à revenir (ce qui est particulièrement probable avec UDP). Les services doivent donc en tenir compte dans leur architecture et implémenter les mécanismes nécessaires à la résolution de tels problèmes. Une solution communément adoptée est la génération d'identifiant unique (*UID*) par message au niveau des clients. Ceci permet d'éviter les messages perdus et orphelins (Cf chapitre 3.3.5) d'une part, mais aussi d'assurer le comptage de la réception des messages. Le client, de son côté, peut relancer un message considéré perdu (un message pour lequel aucune réponse attendue n'a été reçue) avec le même UID ; le serveur peut garder les UID des messages reçus et correctement exécutés afin de ne pas les exécuter plusieurs fois en cas de réception multiple.

Sur une ressource locale, les services SunRPC sont accessibles grâce à un service, le *port mapper* qui agit comme une table de hashage mettant en relation les couples *[protocole, numéro de port]* avec les services eux mêmes. Un service s'appuyant sur SunRPC est déterminé par un couple *[numéro de service, numéro de version]*. Les numéros de service sont des ressources critiques et doivent être unique sur chaque ressource. Un même service peut avoir le même numéro sur des ressources

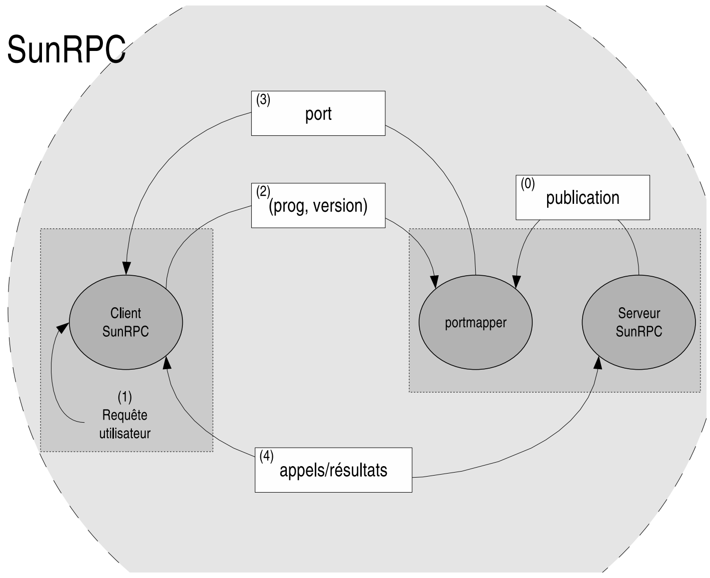

FIG. 5.1 – Le protocole SunRPC.

distribuées ; certains services «standards», tel que NFS, ont des numéros de services spécifiques qui ne doivent pas être utilisés par d'autres services ; ceci permet d'unifier l'utilisation des services les plus connus sur des ressources distribuées.

Un client souhaitant accéder à un service SunRPC sur une ressource donnée, commence par s'adresser au port mapper de cette ressource en fournissant le couple [numéro de service, numéro de version] du service souhaité. Si le port mapper est disponible sur la ressource visée et si celui-ci connaît le service demandé, il retourne le numéro de port sur lequel ce dernier peut être contacté. La détermination du protocole (TPC ou UDP) est celle du protocole utilisé par le client pour contacter le port mapper.

La figure 5.1 schématise les accès aux services basés sur SunRPC :
- (0) le service est publié au niveau du port mapper de la ressource ;
- (1) l'utilisateur souhaite accéder au service grâce à son client local ;
- (2) le client contacte le port mapper du serveur distant, en lui fournissant le numéro du service demandé et éventuellement son numéro de version. Le client fait son propre choix quant au protocole souhaité (UDP ou TCP) ;
- (3) le port mapper retourne le port sur lequel le service est joignable sur le protocole utilisé ;
- (4) le client et le serveur communique directement à travers de ce port de communication, en utilisant le même protocole (UDP ou TCP).

Si le service n'est pas disponible le port mapper retourne en (3) un code d'erreur et (4) ne peut avoir lieu ; le client, s'il le souhaite peut alors essayer de vérifier si le service ne serait pas disponible dans un autre protocole ou sur une autre ressource.

Cette procédure ne peut fonctionner qu'à la condition que les éventuels pare-feux devant les serveurs (le port mapper et le serveur applicatif) soient configurés afin de laisser passer les messages entrants.

Les paquets SunRPC ont une structure bien définie :

- [0] 4 octets : un entier de 32 bits : XID (identifiant unique de la commande) ;
- [1] 4 octets : un entier de 32 bits : MSG type (appel, réponse) ;
- [2] 4 octets : un entier de 32 bits : RPC version ;
- [3] 4 octets : un entier de 32 bits : numéro de programme (nécessaire pour détecter le port de communication) ;
- [4] 4 octets : un entier de 32 bits : version de programme (nécessaire pour détecter le port de communication) ;
- [5] 4 octets : un entier de 32 bits : numéro de méthode ;
- [6] 4 octets : un entier de 32 bits : type de certificat ;
- [7] 4 octets : un entier de 32 bits : nombre d'octets du certificat ;
- [8] 4 octets : un entier de 32 bits : date et heure de création du certificat ;
- [9] 4 octets : un entier de 32 bits : longueur du nom de la machine émettrice ;
- quelques octets : nom de la machine émettrice ;
- quelques octets ($0 <= n <= 3$) : octets d'alignement (Longueur(nom machine émettrice + octets d'alignement) % 4 = 0) ;
- 4 octets : un entier de 32 bits : identifiant utilisateur ;
- 4 octets : un entier de 32 bits : identifiant groupe utilisateur ;
- 4 octets : un entier de 32 bits : nombre de groupes utilisateur auxiliaires ;
- 4 octets : un entier de 32 bits : GID auxiliaire [0] ;
- 4 octets : un entier de 32 bits : etc. ;
- 4 octets : un entier de 32 bits : type de vérification ;
- 4 octets : un entier de 32 bits : longueur vérification.

Nous y voyons les deux informations utiles à la mise en relation d'un client à un serveur ; les quatrième et cinquième entiers contiennent respectivement les numéro de programme et numéro de version du service demandé, permettant de connaître le numéro de port de communication du serveur implémentant le service. Les autres informations contenues dans les paquets SunRPC concernent principalement la sécurité et ne sont pas détaillées ici dans la mesure où notre plate-forme ne s'y intéressera pas : notre proposition ne regardera ni ne modifiera en rien les informations de sécurité mis en place par SunRPC.

Nous verrons au chapitre 5.4, que le service de spécialisation XtremWeb pour le transport de paquet SunRPC nécessite une bonne compréhension de ces paquets afin de rester dynamique et générique.

Déploiement. RPC est un service qui est fourni avec les principaux systèmes d'exploitation (*OS*), en particulier les différents Unix disponibles sur le marché, dont *Linux*. C'est donc, de fait, un service largement déployé. Il est très facile d'installer ce service sur les ressources qui ne le posséderaient pas, car les packages d'installation sont disponibles sur le Web. Nous ne occuperons donc pas du déploiement des services RPC dans ce chapitre.

Sécurité. Le protocole inclut des éléments de sécurité permettant de vérifier l'authentification des utilisateurs et la confidentialité des communications. L'un des premiers types de sécurité pris en charge par RPC fut *AUTH_SYS* (également appelé *AUTH_UNIX*). Cette sécurité fournit un justificatif d'identité de type UNIX, avec les identifiants des utilisateurs et des groupes, afin d'identifier l'expéditeur et le destinataire d'un message. AUTH_SYS est facile à mettre en œuvre ; cependant, il est aussi facile à contourner, puisqu'il n'assure pas une véritable authentification : un serveur n'a aucun moyen de vérifier qu'un client est réellement ce qu'il prétend. Il est donc relativement simple de contrefaire une requête réseau avec AUTH_SYS.

Un autre type de sécurité, *AUTH_DES*, est basé sur une authentification à clé publique : la clé privée d'un client et et la clé publique d'un serveur permettent de créer une clé de session *DES*, laquelle est déchiffrée par un serveur pour l'établissement d'une session. Cette solution constitue un progrès significatif. Elle a toutefois certaines limitations empêchant son utilisation répandue. La principale objection soulevée est que la taille des clés est faible par rapport aux normes de chiffrement actuelles.

Pour améliorer la sécurité, une nouvelle couche réseau, l'API *« Generic Security Standard » (GSS)*, a été ajoutée. La structure de l'API GSS offre deux services de sécurité supplémentaires en plus de l'authentification :

1. L'intégrité. Avec le service d'intégrité, l'API GSS utilise le mécanisme sous-jacent pour authentifier les messages échangés entre programmes. Les sommes de contrôle cryptographiques établissent :
   (a) L'identité de l'émetteur des données pour le destinataire.
   (b) L'identité du destinataire pour l'émetteur (si une authentification mutuelle est demandée)
   (c) L'authenticité des données transmises elles-mêmes
2. La confidentialité. Le service de confidentialité englobe le service d'intégrité. En outre, les données transmises sont aussi chiffrées afin de les protéger contre les oreilles indiscrètes.

Pour notre propos, nous nous intéresserons à la sécurité mise en œuvre avec *AUTH_UNIX* seulement, car l'utilisation de certificats nécessite la mise en œuvre d'une autorité de certification, ce qui est hors de propos du travail présenté ici.

La sécurité de type AUTH_UNIX est basée sur les identités des utilisateurs et des groupes d'utilisateurs telles que définis au niveau de chaque ressource, pour sécuriser et autoriser les appels distants. Dans le reste de ce chapitre nous nommons « utilisateur », l'ensemble des utilisateurs et des groupes d'utilisateurs. La sécurité repose donc sur la capacité de chaque ressource à identifier les utilisateurs, ce qui peut s'avérer compliqué dans un environnement distribué. Dans un cluster, au sein d'un seul domaine administratif, les informations sont généralement distribuées grâce au protocole *LDAP (Lightweight Directory Access Protocol)* qui permet la distribution standardisée à toutes les ressources des informations relatives au domaine (identités des utilisateurs, définitions des groupes etc). Cela assure l'uniformité des identités des utilisateurs du domaine. Ainsi un utilisateur souhaitant utiliser des volumes partagés au sein du domaine est identifié sans équivoque par toute ressource du domaine à laquelle il essaie d'accéder.

Il convient toutefois d'assurer la sécurité dans le cas où les ressources ne partagent pas les identités des utilisateurs. Avec le mode AUTH_UNIX, il faut alors définir une politique relative à la distribution des droits. On peut, par exemple, définir des identités d'utilisateurs communes ; mais cette solution peut s'avérer fastidieuse au delà d'un certain nombre d'utilisateurs.

5.3.1 NFS : un service basé sur le protocole SunRPC

Le service NFS est un service de stockage distribué permettant d'accéder à des espaces de stockage distants, qu'on appelle des volumes de stockage, de la même manière que les volumes locaux. NFS est basé sur le protocole SunRPC. Comme SunRPC, il suit le modèle applicatif client/serveur. NFS s'appuie sur plusieurs services, dont nous ne citerons que les deux qui nous intéressent pour la présente démonstration : *nfsd* qui est le service principal, et *mountd* qui gère les exports et les montages des volumes.

La figure 5.2 schématise les accès aux services NFS :
- (0) les services nfsd et mountd sont publiés au niveau du port mapper de la ressource ;
- (1) l'utilisateur souhaite accéder au service grâce à son client local (mountd) ;
- (2) le client contact le port mapper du serveur distant ;
- (3) le port mapper retourne les ports sur lesquels les services sont joignables ;
- (4) le client et le serveur communique directement ;
- (5) les accès au volume sont gérés (exportation, montage...) et les données peuvent transiter grâce au service nfsd.

Déploiement. Tout comme les services RPC, le service NFS est fourni avec les principaux systèmes d'exploitation (*OS*), en particulier les différents Unix disponibles sur le marché, dont *Linux*. C'est donc, de fait, un service largement déployé. Il est très facile d'installer ce service sur les ressources qui ne le posséderaient pas, car les packages d'installation sont disponibles sur le Web. Nous ne occuperons donc pas du déploiement de NFS dans ce chapitre.

Sécurité. L'intégrité des volumes et des données ne sont pas pris en charge par NFS lui même qui laisse cette responsabilité à la ressource hôte. L'authentification et la confidentialité sont, d'autre part, une composante importante de la sécurité dans un environnement distribué. Sur ce point, NFS s'appuie sur les mécanismes

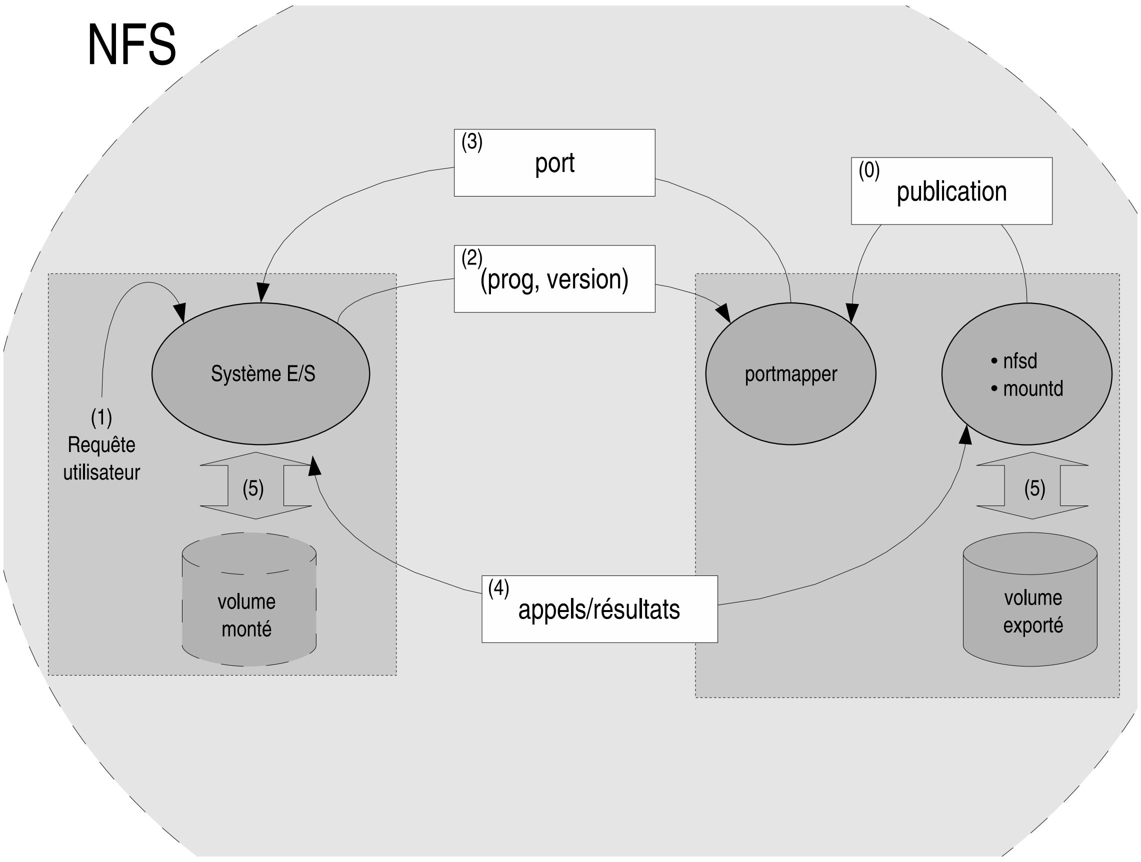

FIG. 5.2 – Distribution de fichiers par NFS.

mis en œuvre par SunRPC.

Comme tout service, la configuration de NFS lui même nécessite des privilèges élevés ; la configuration définit les volumes exportés, les volumes montables et les autorisations afférentes : qui a le droit de monter quoi, où et comment. Cette configuration peut définir des utilisateurs autorisés à monter et démonter des volumes, à écrire et/ou lire des données dans certains volumes.

Nous avons toutefois vu qu'il est difficile, voire impossible, d'unifier les identités entre différentes ressources. Afin de résoudre cette problématique, NFS propose un système de translation de droits, où l'accès à un volume peut être fait sous la responsabilité d'un utilisateur local ; cette solution a toutefois l'inconvénient de perdre des informations quant aux identités réelles des utilisateurs accédant aux volumes. Il sera alors difficile d'éviter ou de corriger les problèmes (corruptions ou pertes de données, utilisateurs malintentionnés), la journalisation ne contenant pas les identités réelles des utilisateurs.

Partage de ressources. Partager des volumes nécessite des configurations adéquates, tant du côté client que du côté serveur ; chaque ressource possède sa propre configuration. Ces configurations décrivent explicitement les droits pour chaque volume exportable et montable . elles ne peuvent être créées ou modifiées qu'avec les droits adéquats, en général les droits administratifs. Du côté du serveur, le fichier de configuration décrit les volumes exportés et les droits associés ; les utilisateurs distants pouvant monter les volumes, ainsi que leurs droits : droit en écriture, droit en lecture etc. Du côté du client, le fichier de configuration décrit les volumes distants montables ; l'adresse des serveurs exportant des volumes et les points de montage locaux où ces volumes distants sont montés. Les points de montage permettent d'accéder localement aux volumes.

Nous ne détaillerons pas de manière exhaustive les différents paramètres de configuration de NFS, mais seulement ceux qui nous seront utiles pour le partage de volumes NFS sur une plate-forme de calcul global.

- Configuration du serveur.
  La configuration de l'export d'un volume en lecture/écriture pour un utilisateur avec translation de droits à travers un processus non privilégié sur la ressource locale, est la suivante :

  ```
  /tmp localhost(rw,insecure,anonuid=1001)
  ```

  Ainsi, le répertoire local `/tmp` est exporté. Le serveur NFS accepte des requêtes de clients NFS locaux uniquement (adresse du client : `localhost`) ; ceci assure que seules les requêtes locales à la ressource elle même seront acceptées. Le volume est exporté en lecture/écriture (option `rw`). Ces requêtes peuvent être émises par des process non privilégiés, c'est à dire des process utilisant des ports de communications supérieurs à 1024. La configuration le précise car NFS n'accepte pas, par défaut, de tels ports de communication pour des raisons de sécurité (option `insecure`). Enfin, le volume exporté est accessible par un mécanisme de translation des droits à utilisateur référencé (option `anonuid=1001`) ; ici, pour notre exemple, nous supposons que `1001` est un identifiant utilisateur référencé localement sur la ressource. Tous les accès relatifs à ce volume seront donc effectués sous la responsabilité, et avec les droits, de cet utilisateur. On voit bien ici la perte d'information relative à l'identité réelle de l'utilisateur effectuant des requêtes d'entrées/sorties. Ceci est un point faible de NFS, car la sécurité au niveau du serveur ne permet donc pas de journaliser les actions des différents utilisateurs autorisés à utiliser le volume partagé.
- Configuration du client.
  La configuration d'un volume montable depuis la ressource locale est la suivante :

  ```
  localhost :/tmp /mnt/tmp nfs port=4325, mountport=4325, timeo=30000,rw,noauto
  ```

  Ainsi, le volume montable `/tmp`, tel que défini par le serveur, sera monté sur le point de montage `/mnt/tmp` ; les données distantes seront donc accessibles localement sur `/mnt/tmp`. Le volume ne peut être utilisé que depuis la ressource locale uniquement (adresse du serveur : `localhost`). Le volume est montable en lecture/écriture (option `rw`). Les requêtes doivent être émises sur les ports de communications 4325 ; le même numéro de port est utilisé pour le montage du volume (option `mountport=4325`) et pour l'accès aux données (option `port=4325`). L'option `timeo=30000` autorise un délai de 30 s de temps d'accès. L'option `noauto` spécifie que ce volume n'est pas monté automatique par le système.

De telles configurations NFS interdisent donc tout requête distante. Dans la section suivante, nous expliquons l'usage de ces configurations pour utiliser NFS sur notre grille de calcul global. Nous expliciterons en particulier l'usage de configuration en ressource locale avec des adresses qui seront toujours `localhost` sur les serveurs comme sur les clients NFS.

5.4 NFS sur une plate-forme de calcul global

Pour pouvoir utiliser NFS sur notre plate-forme de calcul global, nous allons utiliser celle ci comme couche de transport pour les paquets SunRPC grâce au service de spécialisation de XtremWeb (Cf chapitre 3.3.2). Comme tout service de spécialisation, celui pour le transport de paquets SunRPC est dynamique et auto-descriptif.

Comme nous allons le voir ici, l'utilisation de notre plate-forme GC pour transporter des paquets NFS a plusieurs intérêts :
- la virtualisation des serveurs NFS, et donc des volumes partagés ;
- la définition d'identités utilisateurs inter domaines, ce qui permet d'éviter les écueils d'identification des utilisateurs dans le partage de volume, comme nous l'avons décrit à la section précédente ;
- une meilleure sécurisation des volumes partagés grâce à la journalisation des messages sur notre plate-forme GC qui permet de suivre les actions engagées et de corriger d'éventuelles erreurs de corruptions ou de pertes de données. Cela permet même, si besoin est, de révoquer des utilisateurs malintentionnés grâce à la définition d'identités inter domaines.

Architecture. L'architecture d'XtremWeb est mise en œuvre telle que nous la connaissons (Cf chapitre 3.3.1) : c'est une architecture trois tiers, avec son service client, son service de coordination et son service de calcul.

Services. Il convient de spécialiser la plate-forme, grâce au service de spécialisation d'XtremWeb (Cf chapitre 3.3.2). Nous avons donc implémenté un nouveau service, `xtremweb.services.rpc`, qui permet de transporter les paquets SunRPC afin de virtualiser les serveurs basés sur ce protocole (Cf figure 3.9). Notons bien que ce service n'est pas spécialisé pour NFS, mais bien pour SunRPC. NFS a été pris comme exemple, mais n'importe quel applicatif basé sur le protocole SunRPC, peut être transporté sur la plate-forme. NFS est toutefois un exemple de choix pour le transport de paquet SunRPC, car il est largement distribué et facilement configurable. De plus, comme tous service basé sur SunRPC, NFS est utilisable de manière dynamique.

Le service client d'XtremWeb enveloppe les requêtes SunRPC dans des tâches XtremWeb qu'il soumet au service de coordination. La soumission peut être définie pour un service de calcul précis qui exécutera cette tâche ; nous utilisons ici cette capacité afin de fournir les paquets SunRPC au serveur NFS qui exporte le volume souhaité. Quand cette tâches est correctement exécutée, le service client extrait les résultats XtremWeb de leur enveloppe et les fournit au client SunRPC qui avait initialement soumit le paquet.

Le service de coordination est utilisé sans modification spécifique autre que la spécialisation pour le transport de paquet SunRPC. Il gère dynamiquement tout service de spécialisation, en particulier leur publication et leurs cycles de vie. Il publie dynamiquement les volumes exportés et visibles par les services de calcul spécialisés.

Le service de calcul contient le service de spécialisation de transport de paquet SunRPC. Ainsi, il peut décoder les numéro de programme et de version de l'applicatif destinataire, lui transmettre le paquet et attendre la réponse. A la réception d'une tâche XtremWeb, ce service extrait le paquet SunRPC, lit les informations relatives à l'applicatif *[num prog, num version]* afin de se mettre en relation avec celui-ci, grâce aux services SunRPC (le port mapper). S'il reçoit une réponse, il l'enveloppe dans un résultat XtremWeb qu'il retourne au service de coordination.

Qualité de service. Nous avons vu au chapitre 3.3.3 que notre plate-forme propose un ensemble de mécanismes permettant d'offrir une bonne qualité de ses différents services (service client, service de coordination et service de calcul). Toutefois, nous avons également dit qu'elle ne sait prendre en charge que des services applicatifs sans état (*stateless*), c'est à dire des services applicatifs dont les exécutions sont indépendantes les unes des autres. Le partage de volumes à travers le service NFS n'entre pas dans une telle définition ; les actions prises en charge par ce service dépendent du nombre d'actions et de leur ordre : on en peut pas effacer un fichier qui n'a pas été crée, ni même écrire le deuxième octet avant le premier etc. XtremWeb en l'état ne peut donc pas proposer de qualité de service relatif au partage et à l'intégrité de volumes NFS, malgré ces capacités de migration automatique des services applicatifs entre les différentes ressources de calcul. Il est en effet inutile de faire migrer un service NFS d'une ressource à une autre dans la mesure où notre plate-forme ne saura faire redémarrer le service dans son dernier état connu. De tels mécanismes permettant de redémarrer des services applicatifs dans leur dernier état connu existent bien sûr et sont l'objet de travaux intensifs [20], mais sont hors de propos des travaux présentés ici.

La qualité de service des volumes partagés devrait donc être assurée par des moyens hors de notre plate-forme. On peut par exemple facilement imaginer des mécanismes de synchronisation entre différentes ressources d'un même domaine administratif.

Communications. Nous avons vu, au chapitre 3.3.4, que les communications dans XtremWeb sont implémentées afin que les ressources puissent communiquer entre elles, quelles qu'elles soient. Ceci est possible grâce à une conception basée sur les *handlers* [54]. Nous utilisons ici cette capacité de la plate-forme afin que le service client puisse recevoir des messages entrants, en l'occurrence les paquets SunRPC, et les transmettre sur la plate-forme jusqu'au service SunRPC visé. Notre service client implémente donc le serveur UDP de la couche de communication d'X-tremWeb et utilise un handler spécialisé et fournit par le service de spécialisation `xtremweb.services.rpc`.

Notre plate-forme de calcul global ainsi spécialisée permet de connecter des serveurs et des clients au delà des limites administratives des domaines et de leurs pare-feux. Elle permet aussi de virtualiser les services basés sur SunRPC grâce à leur publication dynamique.

Bien sûr le transport de paquet SunRPC sur XtremWeb se paye au prix de la sur-couche de communication.

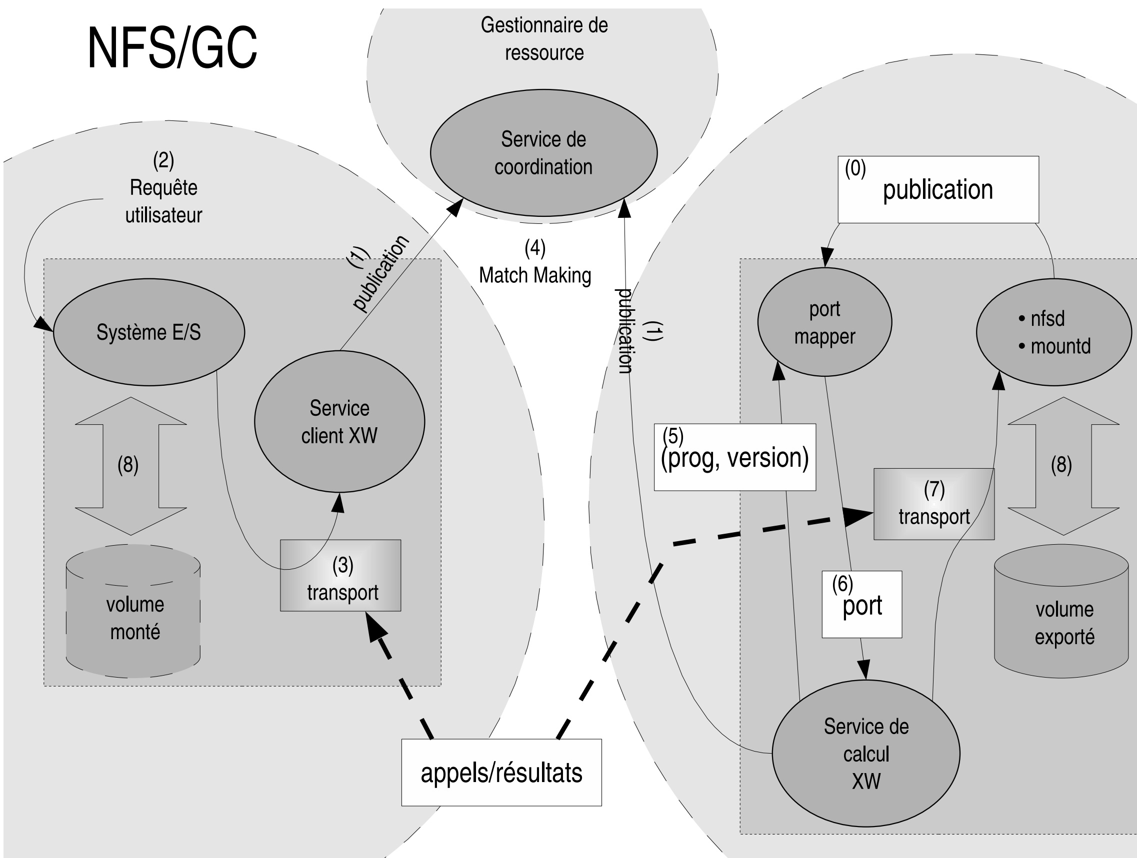

FIG. 5.3 – NFS sur une plate-forme de calcul global : schématisation.

La figure 5.3 schématise les accès au service NFS sur XtremWeb.

- (0) les services nfsd et mountd sont publiés au niveau du port mapper de la ressource ;
- (1) les services XtremWeb sont publiés au niveau du service de coordination ;
- (2) l'utilisateur souhaite accéder au service grâce à son client local (mountd) ;
- (3) celui-ci contacte le service client d'XtremWeb ;
- (4) le service de coordination prend en charge la tâche du service client et la soumet au service de calcul ;
- (5) le service de calcul contacte le port mapper local ;
- (6) qui lui fournit les informations permettant de se mettre en relation avec l'applicatif demandé ;
- (7) le service de calcul soumet le paquet SunRPC reçu à l'applicatif visé ;
- (8) la mise en relation faite, le travail continue.

Cette figure montre les actions spécifiques à SunRPC dans des cadres sur fond blanc. Les cadres marqués (3) et (7) correspondent au service de transport et montrent les messages échangés avec l'applicatif SunRPC.

Déploiement. Le déploiement concerne les services client et calcul de XtremWeb. Nous ne considérons pas le déploiement des services NFS et n'utilisons pas les ressources ne disposant pas de ces services. Les services de calcul et de coordination publient automatiquement les informations relatives à la disponibilité des services NFS et des volumes exportés.

Sécurité. La sécurité des ressources et des communications est mise en œuvre par la plate-forme elle même, comme décrit au chapitre 3.2.2. La sécurité des volumes partagés est assurée par NFS et ses mécanismes afférents. La sécurisation de partages de volumes à travers notre plate-forme de calcul global s'appuie donc à la fois sur les mécanismes de sécurité de notre plate-forme de calcul global, et sur ceux mis en œuvre par NFS. Ce dernier à la responsabilité de la gestion des autorisations de l'export et/ou du montage des volumes, au niveau de chaque ressource ; il doit, entre autre, définir l'identité de l'utilisateur autorisé à utiliser les volumes partagés (option `anonuid`). Les configurations et autorisations de NFS sont du domaine de la responsabilité des administrateurs des ressources. On reste donc, au niveau des ressources, dans une problématique de sécurisation de clusters et de leurs services associés, au sein de chaque domaine administratif. La sécurisation du partage de volumes NFS sur notre plate-forme est relative aux accès aux ressources grâce à la possibilité de spécifier les clients et les serveurs connectables. Nous configurons le serveur NFS afin de n'autoriser que les clients locaux à la ressource elle même (adresse du client : `localhost`). Les clients sont configurés de la même manière ; ils ne sont autorisés à se connecter qu'à un serveur local (adresse du serveur : `localhost`). Les ressources ne sont donc pas autorisées à se connecter au monde extérieur mais seulement à des services locaux. La sécurité au niveau de la plate-forme a déjà été discuté au chapitre 3.2.2. Elle concerne la sécurité des ressources et des communications, mais aussi la définition d'utilisateurs ainsi que la journalisation des actions transitant sur la plate-forme. Elle permet ainsi de détecter d'éventuels problèmes de consistance et/ou de sécurité ; on peut, par exemple, révoquer des utilisateurs ayant un comportement inapproprié.

Ainsi, le partage des volumes sur notre plate-forme est globalement sécurisé ; au niveau des ressources, au niveau des communications, comme au niveau de la plate-forme.

5.5 Expérimentations

Nous détaillons, dans cette section, la mise en œuvre de partages de volumes sur notre plate-forme. Les expériences ont été faites dans la configuration décrite dans le tableau 5.1.

	CPU	Vitesse CPU	Mémoire	*Swap*
service de coordination	AMD	1.9 Ghz	1GB	2GB
service client	AMD x 2	2.0 Ghz	512MB	1GB
service de calcul	Pentium IV	1.8 Ghz	512MB	1GB
réseau	cartes 100Mbit sur switch 1Gbit			

TAB. 5.1 – Configuration de la plate-forme de test.

Une première expérience a été faite avec XtremWeb dans sa configuration de base, consistant en quatre montages/démontages successifs. Le tableau 5.2 montre les résultats obtenus au niveau de chaque service. Cette expérience nous montre que la plate-forme sait transporter les paquets SunRPC.

Les temps de réponses obtenus sont bien sûr incomparablement plus longs que NFS sur cluster. Atteindre de telles performances à travers notre plate-forme est hors de propos, pour plusieurs raisons :

- notre proposition ajoute une couche de communication supplémentaire qui multiplie par quatre le nombre de communications nécessaires ;
- notre plate-forme est écrite en langage Java qui est un langage interprété ;
- notre plate-forme se place au niveau de la couche applicative sur chaque ressource, alors que NFS se place au niveau des pilotes du noyau ;
- notre plate-forme journalise les messages en transit.

	Temps de réponse
Service client	80157
Service de coordination	79992
Service de calcul	79042

TAB. 5.2 – Temps d'exécution de quatre montages/démontages successifs (en ms).

Nous pouvons toutefois améliorer nos performances. Le tableau 5.3 liste ces différents goulots identifiés. Les accès à la base de données ne concernent que le service de coordination. Les accès disque au niveau des services client et de calcul ne sont pas sujets à optimisation dans la mesure où le service de spécialisation n'utilise pas le disque. Les accès réseau ainsi que les temps de synchronisation concernent l'ensemble de la plate-forme. Ces derniers sont les temps nécessaires à la synchronisation des différents services entre eux.

	service client	service de coordination	service de calcul
accès base de données		X	
accès disque		X	
accès réseau	X	X	X
temps de synchronisation	X	X	X

TAB. 5.3 – Goulots d'étranglement.

Optimisation des services client et de calcul. Le service de calcul boucle indéfiniment sur des demandes de tâches auprès du service de coordination. Entre deux requêtes, il attend un peu ; ce temps d'attente est multiplié par deux entre deux requêtes non satisfaites et retombe à sa valeur d'origine dès qu'il a reçu une nouvelle tâche. Ce délai reste à son minimum tant que le service de calcul reçoit des tâches. La courbe d'attente entre deux requêtes est représentée sur la figure 5.4.

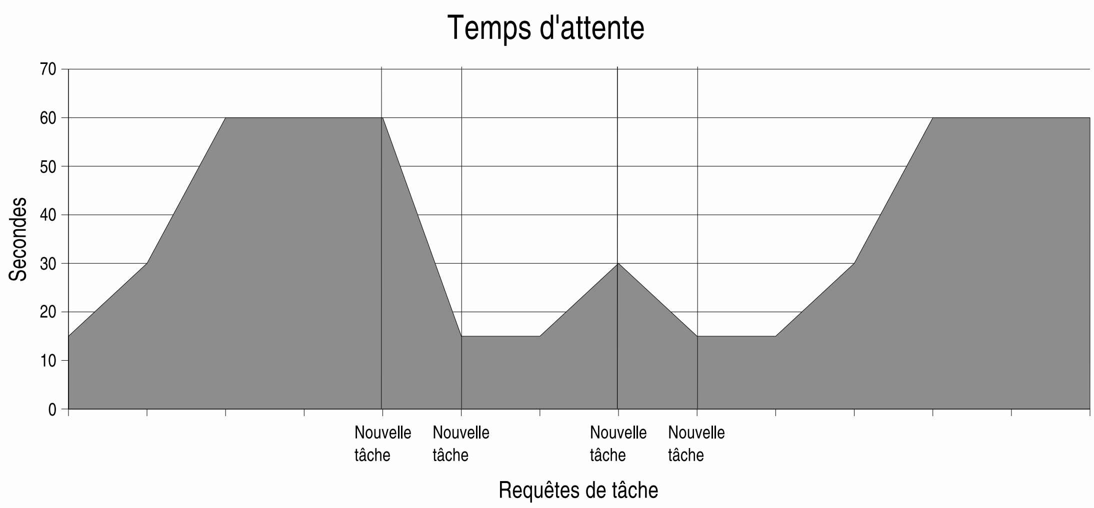

FIG. 5.4 – Temps d'attente au niveau du service de calcul.

L'optimisation du délai d'attente du service de calcul est représenté sur la figure 5.5. Cette figure montre qu'un délai de 10ms est optimum. En deçà de cette valeur, les performances diminuent et peuvent même être pires. Cela est du à l'im-

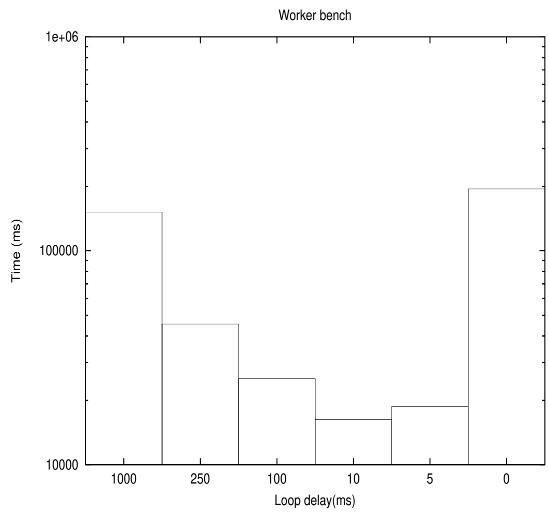

FIG. 5.5 – Tests de délai de synchronisation du service de calcul

plémentation *multi-thread* [6] du service de calcul ; un thread pour la gestion de la ressource, un pour celle des communications et un thread par tâche à exécuter. Ces threads partagent les mêmes données et doivent donc se synchroniser. En deçà de 10ms, les threads ont du mal à se synchroniser et ils finissent même par passer plus de temps à se synchroniser qu'à travailler.

Les mêmes conclusions s'imposent quant au service client. Celui boucle indéfiniment sur des demandes de résultats auprès du service de coordination, afin de récupérer les résultats de ses jobs. Le service client attend un peu entre deux demandes non satisfaites. En deçà de 10ms, on observe le même phénomène que sur le service de calcul : le service client fini par perdre du temps pour des raisons de synchronisation inter-threads.

Optimisation du service de coordination. Le temps d'attente au niveau du service de coordination est au niveau de la gestion des tâches. Le service de coordination tient une liste des tâches en cours d'exécution et les vérifie à intervalle régulier afin de détecter celles qui sont à ré-ordonnancer. On peut observer les mêmes phénomènes que pour le service de calcul ; en deçà de 10ms, les performances du service de coordination se dégradent.

Le service de coordination utilise grandement une base de données afin de gérer la plate-forme et les tâches qu'elle prend en charge de manière sécurisée et pérenne. Le service de coordination est indépendant du système de gestion de base de données (*SGBD*) utilisé. Nous avons testé deux SGBD, *MySQL 4.0.24* et *Hs-*

[6] Un "thread" est une unité d'exécution ; une implémentation multi-threads permet de paralléliser certaines actions qui doivent alors être synchronisées.

qlDb 1.8.0, avec les moteurs *MEMORY*, qui stocke les informations en mémoire sans entrée/sortie sur aucune unité de stockage, et *MyISAM* qui enregistre les informations sur une unité de stockage.

Optimisation des accès réseau. Notre plate-forme fait bien sûr grand usage du réseau.

Nous proposons deux types d'optimisation. Le premier nous est fourni par les librairies réseau de Java qui nous permettent de paramétrer les connexions :

- socket.setSoLinger(false, 0) invalide la persistance des messages en cas de rupture de liaison ;
- socket.setTcpNoDelay(true) permet de forcer l'envoie immédiat ;
- socket.setTrafficClass(0x08) maximise le débit ;
- socket.setKeepAlive(false) invalide le maintien de connexion non utilisée.

Le second type d'optimisation est interne à notre plate-forme. Il concerne la compression. Si un paramètre ou un résultat est "gros", ou s'il est composé de plusieurs fichiers, notre couche de message crée automatiquement un fichier compressé ; si par contre il n'est composé que d'un fichier, pas trop gros (inférieur à 250Ko), il n'est pas compressé et est transmit tel quel.

Les différentes expériences Afin de mettre en évidence ces différentes optimisations, nous avons fait des expériences consistant toutes à soumettre 100 jobs. Nous avons décomposé ces expérimentations afin de se rendre compte de leurs conséquences sur les différents services.

Le tableau 5.4 montre les temps (en ms) obtenus sans aucune optimisation.

	Temps de réponse
Service client	10642
Service de coordination	3552508
Service de calcul	3036936

TAB. 5.4 – Temps d'exécution de 100 jobs (en ms), sans optimisation.

Le tableau 5.5 indique les configurations des différentes expériences menées. Les délais de synchronisations des différents services ont tous été positionnés à 10ms, dans la mesure où nous avons vu que c'est la valeur optimum.

Les résultats montrent clairement qu'on obtient les meilleures performances avec la base de données HsqlDb et son moteur MEMORY. Les autres optimisations sont plus difficiles à interpréter dans la mesure où les différences des résultats obtenus ne sont pas énormes et pour lesquelles il faut tenir compte des erreurs de prise de mesure. La soumission semble relativement indépendante des conditions d'expérimentation. Ceci s'explique par le fait que l'on soumet des tâches sans paramètres et qu'une soumission n'attend pas de réponse de la part du service de coordination. Il en est de même pour la suppression des jobs. Les temps sont mêmes plus faibles que

	Zip	Réseau	Disque
MySQL ENGINE=MEMORY			
exp03	F	T	F
MySQL ENGINE=MyISAM *(Disk)*			
exp13	F	T	F
HsqlDb ENGINE=DISK			
exp23	F	T	F
HsqlDb ENGINE=MEMORY			
exp50	F	T	T
exp52	F	F	T
exp53	F	T	F
exp54	F	F	F
exp51	T	T	T
exp55	T	F	T
exp56	T	T	F
exp57	T	F	F

TAB. 5.5 – Configuration des différentes expériences.

	soumission		exécution		status		résultat		suppression	
	client	coord	coord	worker	client	coord	client	coord	client	coord
MySQL ENGINE=MEMORY										
exp03	541	1966	6897	6904	1026	1323	4202	4488	269	798
MySQL ENGINE=MyISAM *(Disk)*										
exp13	536	2100	7072	7133	952	1252	4245	4517	268	750
HsqlDb ENGINE=DISK										
exp23	533	1816	7247	7257	975	1210	4246	4461	297	673
HsqlDb ENGINE=MEMORY										
exp50	540	1883	8826	8788	974	1232	4016	4299	267	621
exp52	521	1972	9088	9088	1011	1288	4188	4380	260	523
exp53	559	1912	14162	14144	1047	1283	4394	4611	358	645
exp54	516	2031	9045	9077	964	1237	4032	4238	281	516
exp51	536	1846	8999	9024	1033	1289	2878	3082	275	712
exp55	526	1833	8784	8810	972	1226	2654	2851	252	441
exp56	575	1962	8165	8155	996	1221	2734	2920	278	656
exp57	533	1952	8889	8884	964	1205	2706	2887	256	518

TAB. 5.6 – Résultats des différentes expériences.

pour les soumissions car une opération de suppression nécessite l'envoie d'un UID et non d'une définition de tâche.

Le point remarquable est que l'on voit très bien qu'il existe un même goulot d'étranglement pour toutes les expériences : l'exécution au niveau du service de calcul qu'il conviendrait d'optimiser. La figure 5.6 montre les mille deux cents cin-

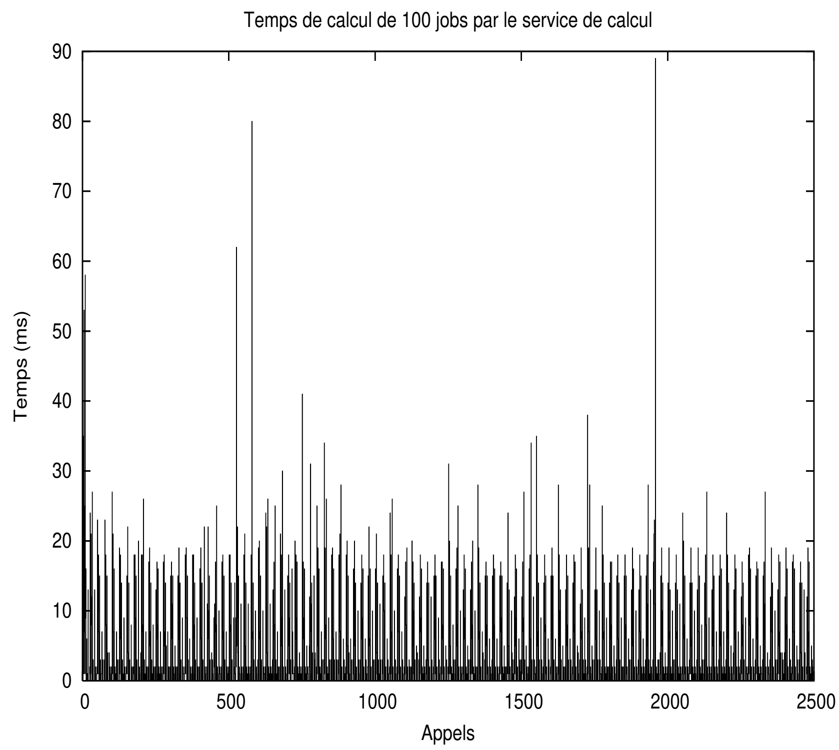

Fig. 5.6 – Temps de calcul de 100 jobs par le service de calcul

quante actions nécessaires à exécuter 100 jobs [7]. On voit que la répartition du temps est relativement homogène. Une optimisation du service de calcul nécessiterait une refonte complète de celui-ci.

Enfin, le tableau 5.7 montre les temps obtenus (en ms) pour quatre montages/démontages successifs avec les optimisations.

	Temps de réponse
Service client	5004
Service de coordination	4959
Service de calcul	4496

Tab. 5.7 – Temps d'exécution de quatre montages/démontages successifs (en ms), avec les optimisations.

Finalement, nous avons mené quelques expériences représentatives des actions les plus courantes. Le tableau 5.8 compare les résultats de NFS et NFS/XW. Les mesures sont au niveau de la seconde ; les opérations de moins de une seconde sont

[7]Le graphique contient 2500 points en abscisse qui correspondent à 1250 points de mesure

marquées *n/m* (pour *non mesurable*).

	NFS	NFS/XW
Montage de volume	*n/m*	1
Création d'un fichier 80Ko	*n/m*	3
Liste du répertoire	*n/m*	1
Suppression du fichier	*n/m*	*n/m*
Création d'un fichier 400Ko	1	14
Liste du répertoire	*n/m*	1
Suppression du fichier	*n/m*	*n/m*
Création d'un fichier 800Ko	1	25
Liste du répertoire	*n/m*	*n/m*
Suppression du fichier	*n/m*	1
Création d'un fichier 2Mo	2	56
Liste du répertoire	*n/m*	*n/m*
Suppression du fichier	*n/m*	1
Création de 10 fichiers vides	1	1
Suppression des fichiers	*n/m*	2
Création de 50 fichiers vides	1	5
Suppression des fichiers	*n/m*	1
Création de 100 fichiers vides	1	13
Suppression des fichiers	*n/m*	1
Création de 250 fichiers vides	1	28
Suppression des fichiers	*n/m*	2
Démontage de volume	*n/m*	*n/m*

TAB. 5.8 – Comparaison de NFS et de NFS/XW pour des opérations courantes d'entrées/sorties (en secondes).

5.6 Conclusions

Nous avons montré que notre plate-forme de calcul global peut virtualiser des services standards de type « cluster », basés sur des protocoles standards. Cette virtualisation n'a nécessité aucun changement des logiciels mis en œuvre pour ces services (il n'a pas été besoin de recompiler les logiciels) ; seule une configuration adéquate des ressources mises en œuvre a été nécessaire. La sécurité de cette virtualisation, basée sur les mécanismes intrinsèques aux services eux mêmes, est globalisée grâce à la définition d'utilisateurs inter-domaines ainsi qu'aux capacités de journalisation de notre plate-forme

Cette virtualisation a toutefois un coût non négligeable en termes de performance. Ce coût vient de l'architecture trois tiers de notre plate-forme qui multiplie par quatre le nombre de communications nécessaires ; des protocoles de communication de notre plate-forme qui sont en mode déconnectés et qui sont exclusivement en XML, ce qui demande un temps de traitement relativement lourd ; et des mécanismes de sécurité et de journalisation. Enfin, notre plate-forme, entièrement écrite en langage Java qui ne saurait rivaliser avec aucun code natif, se situe au niveau applicatif sur chaque ressource, ce qui est sans commune mesure en comparaison aux applications qui se situent au niveau «système» sur les ressources utilisées.

Chapitre 6

Applications : Auger, une expérience pour la détection de rayons cosmiques à très haute énergie

6.1 Introduction

L'observatoire Pierre Auger a été conçu pour la détection et l'étude, avec une capacité de prise de données sans précédent, des rayons cosmiques dont les énergies sont autour et au-dessus de la coupure spectrale de GZK, c'est-à-dire supérieures à 10E+19eV. Dans la limite des connaissances actuelles, il n'existe aucune explication conventionnelle des mécanismes qui sont à l'origine de la production et l'accélération des particules à de telles énergies macroscopiques. Le projet Pierre Auger est l'unique moyen proposé par la communauté scientifique pour résoudre cette énigme astrophysique vieille de plus de 30 ans. La détection et l'étude de ces gerbes devraient permettre de déterminer la composition, l'énergie et l'origine de ces rayons cosmiques. Les mesures s'appuient sur les propriétés des particules détectées au niveau du sol (nombre, position, nature et date d'arrivée) ainsi que sur l'étude de la fluorescence nitrogénique produite par l'interaction de la gerbe avec l'atmosphère.

L'installation d'un détecteur de particules au niveau du sol ne permet pas de détecter directement les particules entrant dans l'atmosphère ; on ne peut que détecter les résultats des interactions multiples de ces particules primaires avec les noyaux atomiques présents dans l'atmosphère. Ces interactions se font en cascade à partir de la première interaction de la particule à haute énergie elle même avec la première particule rencontrée dans l'atmosphère (cette interaction est dite « primaire »). Cette cascade d'interactions génère une *gerbe* de particules. Le laboratoire Pierre Auger peut détecter cette gerbe de particules et la reconstruire, en suivant les modèles de physique standards, jusqu'à l'interaction primaire.

Les champs d'étude de physique ne sont ni exhaustifs ni figés dans la mesure où ils dépendent des données qui seront collectées ainsi que des améliorations techniques à venir au niveau du détecteur.

Comprendre l'origine des rayons cosmiques à ultra hautes énergies *RCUHE*. Les rayons cosmiques à ultra haute énergie qui ont pu être détectés jusqu'à présent semblent venir de sources proches ; on suppose que leurs sources sont peu nombreuses dans notre voisinage. La recherche de sources est un des enjeux du programme Auger grâce à sa vision globale du ciel, à sa bonne résolution angulaire ainsi que de sa capacité à collecter un grand nombre de données statistiques. Notons qu'à ce jour, seul l'observatoire de l'hémisphère sud a été installé.

Les théories conventionnelles d'astrophysique ne peuvent expliquer, jusqu'à présent, les énergies détectées. Ces théories se regroupent en deux catégories pour prédire l'existence de RCUHE : les modèles « bottom-up » d'accélération dans des objets inter-galactiques par un processus itératif d'accélération, et les modèles « top-down » de désintégration de particules super lourdes produites par l'univers primordial (défauts topologiques). Les données d'Auger permettront d'ouvrir de nouveaux champs de physique en confirmant ces théories, ou en en découvrant peut être de nouvelles.

La physique des particules. Les possibilités d'études dans le domaine de la physique des hautes énergies et de la compréhension des interactions fondamentales concernent les défauts topologiques et les particules super lourdes. De telles études permettent de déterminer l'état de l'univers quelques secondes après sa naissance, un temps pendant lequel toutes les interactions de particules ont probablement été unifiées en une seule et simple interaction.

Les particules et leurs interactions que le laboratoire Auger s'apprête à détecter sont à des niveaux d'énergie sans commune mesure à ce que l'on peut reproduire artificiellement et devront permettre de valider (ou d'infirmer) les modèles et les théories des mécanismes d'interaction.

Informatique et astrophysique. Dans la communauté de la physique des hautes énergies, les moyens mis en œuvre, humains et matériels, ne sont par divisés par le nombre d'expériences. Les moyens techniques, par exemple, peuvent dépendre de la difficulté des problèmes à résoudre ; les moyens informatiques, de la complexité des simulations et des dépouillements, ainsi que de la quantité des données à collecter.

Les expériences autour des collisionneurs sont dites « des grands instruments » ; elles portent particulièrement bien ce nom, car elles sont très gourmandes en moyens. Il leur faut des mécaniciens, des opticiens, des électroniciens, des informaticiens... pour créer des collisionneurs, machines hautes de plusieurs étages. Ces machines collectent des péta-octets de données qu'il faut stocker puis dépouiller. Les besoins informatiques sont tellement importants, que cette communauté s'est naturellement engagée dans les grilles (Cf chapitre 2.4).

A contrario, les expériences d'astrophysique ne sont généralement pas incluses dans la famille des expériences « des grands instruments », car elles sont relativement plus faciles à mettre en œuvre, coûtent moins chers et nécessitent moins de personnel. Bien qu'elles aient évidemment d'importants besoins de simulation et de dépouillement, elles ont parfois du mal, face aux mastodontes comme le LHC, à accéder aux ressources de la communauté de la physique des hautes énergies. Ceci est d'autant plus vrai avec la mise en service et la montée en puissance du LHC.

C'est dans ce contexte que la collaboration Auger s'est tournée vers les grilles de PC, qu'une collaboration LAL-LRI s'est mise en place, et que Auger a été partie prenant de l'ACI « GRID CGP2P »[8]. Un des desseins de ces collaborations est d'étudier la mise en œuvre d'une telle plate-forme pour la physique des hautes énergies. Nous présentons, dans ce chapitre, les résultats de ces études et leurs mises en œuvre.

6.2 Les besoins informatiques d'Auger

Les besoins informatiques d'Auger peuvent se décomposer en deux parties distinctes : la simulation de gerbes et la simulation du détecteur. Les physiciens en ont grande utilité car ils ont besoin de beaucoup de simulations afin de pouvoir faire des analyses statistiques et de vérifier les prédictions des modèles de physique. Le temps de calcul d'une simulation est proportionnel à l'énergie souhaitée. Dans la mesure où les simulations intéressantes sont celles à très haute énergie, chaque simulation demande plusieurs heures de calcul.

6.2.1 Simulation de gerbes

Les programmes de simulation de gerbes atmosphériques par méthode dites de *« Monte Carlo »* permettent de faire des analyses statistiques et de vérifier la validité des modèles de physique pour lesquels il n'existe pas de consensus à de tels niveaux d'énergie. Les programmes de simulation utilisés par la collaboration sont Aires [96] (Air Shower Extended Simulations) et Corsika [60] (COsmic Ray SImulations for KAscade), tous deux basés sur *MOCCA* [59]. La simulation de l'interaction d'un rayon cosmique à très haute énergie avec un noyau de l'atmosphère génère une cascade de plusieurs milliards de particules secondaires au niveau du sol. Les programmes de simulation permettent d'obtenir une description détaillée de cette cascade de particules secondaires générées, à partir de la connaissance en physique des propriétés des interactions fondamentales. Les résultats obtenus dépendent des paramètres fournis aux programmes de simulation. Les expériences réalisées sur des accélérateurs de particules, à des énergies beaucoup plus basses, ont permis de mesurer certains de ces paramètres :
- les sections efficaces des interactions ;
- les multiplicités des interactions ;
- l'inélasticité des interactions.

[8] Action Concertée Incitative « Calcul Global Pair à Pair » ; http ://www.lri.fr/~fci/CGP2P.html

Les modèles d'interaction implémentés dans les programmes de simulation permettent de prédire la valeur de ces paramètres à haute énergie. Il permettent aussi de prédire les profils longitudinal et latéral des gerbes. Ainsi les expérimentateurs peuvent directement comparer observations et prédictions.

Les programmes Aires et Corsika implémentent les mêmes modèles de physique (*QSJET* et *SIBYLL*), mais diffèrent dans leurs algorithmes et leurs implémentations.

Aires est un programme de simulation développé au département de physique de l'université de la Plata en Argentine. La version utilisée est la version 2.6.0. Aires est un ensemble de programme et de routines permettant de simuler des gerbes de particules dues à l'entrée dans l'atmosphère terrestre de particules à hautes énergies. La simulation est basée sur une procédure d'échantillonnage statistique (le *thinning*) afin de simplifier les simulations. Il est en effet impossible de simuler complètement une gerbe de particules car le nombre de particules secondaires créées par interaction en cascade est trop élevé (une gerbe de particules générée par une particule à 10E+20eV contient de l'ordre de 10E+11 particules secondaires).

Les besoins en ressources informatiques de Aires sont dépendants des paramètres de simulation. Les temps de calcul varient de quelques minutes à plusieurs jours. Les besoins en espace mémoire et en espace disque se comptent en méga-octets et sont directement corrélés aux nombre de particules de la gerbe simulée. L'espace disque utilisé en cours de simulation peut atteindre la centaine de méga-octets.

L'exécution d'un de ces deux programmes nécessite un fichier en entrée et génère un ensemble de fichier en sortie qui peuvent avoir des tailles non négligeables, au delà de la centaine de méga-octets.

La simulation de gerbes nécessite donc les actions suivantes :

- la compilation du programme de simulation avec les paramètres adéquats : modèle de physique, type de données etc ;
- le déploiement du programme de simulation sur les machines souhaitées ; que ce soit sur le centre de calcul, sur un cluster propriétaire ou sur des machines individuelles ;
- la génération d'un fichier d'entrée par simulation. Notons qu'une simulation peut générer autant de gerbe qu'on le souhaite ; le temps de calcul est donc aussi proportionnel au nombre de gerbes demandées.

La compilation du programme de simulation peut être tout simplement impossible selon la plate-forme visée. Par exemple Corsika ne peut être compilé avec les librairies *FLUKA* pour Windows, ni pour Mac OS X car seule la version Linux est fournie et les codes sources ne sont tout simplement pas disponibles. Nous nous cantonnerons donc aux environnements Linux pour ce programme.

Aires, de son côté, n'a posé aucun problème de recompilation sur les trois plates-formes : Linux, Windows et Mac OS X.

6.2.2 Reconstruction et analyse de données

Les solutions logicielles d'Auger sont relativement hétérogènes et ne proposent pas de chaîne complète permettant de simuler et de reconstruire des événement de physique du laboratoire. Ces solutions sont développées par les collaborateurs d'Auger ; elles s'appuient sur différentes librairies de physique des hautes énergies, dont *ROOT* [9] et *GEANT4* [10].

Les programmes d'analyse et de reconstruction de données d'Auger permettent de simuler les détecteurs et de reconstruire des évènements de physique. Ils sont écrits de manière à accepter des modules utilisateurs permettant de les spécialiser. Ces programmes sont regroupés sous l'appellation *Development Physics Analysis (DPA)*.

Le DPA, écrit en *C++*, définit une structure permettant de décrire un évènement, qu'il soit simulé ou réel. Différents modules peuvent être appliqués séquentiellement à cette structure afin de réaliser différentes tâches de physique, comme la simulation de tout ou partie du détecteur, ou encore la reconstruction d'évènements (simulés ou non) dans différents modes : détecteur de surface ; détecteur de fluorescence ; ou en mode hybride. Les différents paramètres à fournir au DPA peuvent être des fichiers XML, ou éventuellement une base de données MySQL ; ils concernent la configuration du détecteur et des algorithmes. Les séquences de modules à appliquer sont traités par un *Run Controler (RC)* ; elles doivent aussi être décrites dans un fichier XML. Il est de la responsabilité de l'utilisateur de fournir les modules qui doivent être des programmes exécutables compatibles avec la boîte à outils du DPA. Il faut finalement fournir des données calculées par Aires ou Corsika pour la simulation du détecteur, ou des données à reconstruire.

Les résultats du DPA sont stockés dans des *Data Summary Tape (DST)* qui peuvent être stockés dans différents format afin de pouvoir les traiter avec par programmes de physique (par exemple, avec ROOT).

La reconstruction et l'analyse de données nécessitent donc les actions suivantes :
- la compilation de ROOT et Geant4 ;
- la compilation du DPA en lui même ;
- la compilation des modules utilisateurs avec les paramètres adéquats ;
- l'installation des librairies ROOT, Geant4 et DPA sur les machines visées ;
- l'installation des modules utilisateurs ;
- la génération des fichiers d'entrée.

La compilation des librairies ROOT et Geant4 peuvent être tout simplement impossible selon la plate-forme visée. Nous n'avons en effet pas réussi à les compiler pour Windows ni pour Mac OS X. Ici encore, nous nous cantonnerons aux environnements Linux.

[9] ROOT : une librairie d'analyse de données orientée objet `http ://root.cern.ch`

[10] GEANT4 : un logiciel pour la simulation de passage de particules à travers la matière `http ://geant4.web.cern.ch/geant4/`

6.3 Auger et les ressources de calcul

L'expérience d'astrophysique « Auger » est une expérience de la physique des hautes énergies. A ce titre, elle bénéficie des ressources du CCIN2P3 [11]. Les collaborateurs de Auger peuvent utiliser le système de soumission (ou système *batch*) du centre de calcul, ainsi que de ses capacités de stockage. Par contre Auger n'entre pas dans la famille de la physique « des grands instruments » et ne participe pas à la recherche du LHC. Auger ne bénéficie donc pas des ressources de la grille LCG, ni d'aucune autre.

Nous montrons ici l'utilisation pratique de notre plate-forme afin de proposer un ensemble de nouvelles ressources à la collaboration Auger. Nous détaillons tout d'abord les tests préliminaires prouvant l'apport d'une telle plate-forme. Nous décrivons ensuite les utilisations pour les simulations, ainsi que pour la reconstruction et l'analyse des données.

6.3.1 Tests et validation de la plate-forme XtremWeb pour la physique des hautes énergies : simulations de gerbes pour Auger

Nous proposons ici d'utiliser la plate-forme Condor-XW décrite au chapitre 4 pour effectuer des calculs de simulation avec Aires, afin de montrer l'utilisation d'une telle plate-forme par la communauté de physique des hautes énergies.

Nous avons mené des expérimentations avec notre plate-forme XtremWeb avant de la proposer pour la génération de gerbes simulées pour Auger. Chaque expérimentation consiste à soumettre 1024 tâches Aires identiques afin de déterminer le comportement du système dans son intégralité. Nous avons inséré la version Linux de l'application Aires dans la plate-forme car ces expérimentations n'utiliseront que des ressources de calcul de ce type. Chaque tâche est composée d'un fichier d'entrée dont la taille est considéré comme négligeable relativement à la taille des résultats.

Le fait d'insérer des tâches identiques permet de rendre possible les comparaisons entre les différentes tâches, tant au niveau de leur prise en charge que de leur exécution ou encore de la disponibilité de leurs résultats. Notre machine de référence, un mono processeur *Pentium III* à 733 Mhz, exécute une de ces 1024 tâches en 18 minutes. Ce temps d'exécution est du aux paramètres choisis pour notre simulation de gerbe. L'ensemble des 1024 tâches nécessiterait donc 307 heures.

Nous avons déployé XtremWeb sur trois sites, deux en France, un au LRI et un à Grenoble, ainsi qu'un site au Wisconsin, USA. Ce déploiement s'inscrit dans le cadre de l'interaction de plates-formes de calcul, comme présenté au chapitre 4. La figure 6.1 schématise ce déploiement.

Le service de coordination est pris en charge par une machine dédiée au LRI.

[11] Centre de Calcul de l'Institut National de Physique Nucléaire et de Physique des Particules `http ://cc.in2p3.fr`

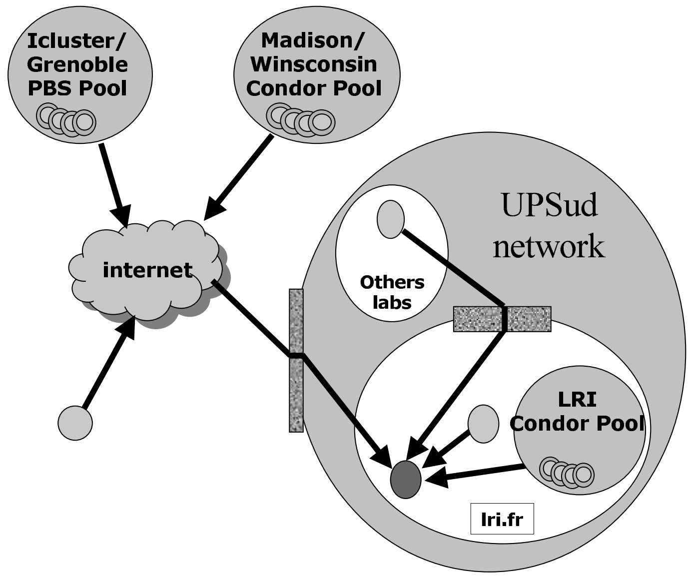

FIG. 6.1 – Déploiement test pour la physique des hautes énergies.

Les services workers sont exécutés sur différentes machines des trois sites ; ils sont pris en charge par les systèmes d'ordonnancement de ces sites. A Grenoble, l'ordonnanceur utilisé est *PBS* [42] ; au LRI et au Wisconsin, c'est Condor [67]. Aucune prédiction d'ordonnancement ne peut être faite, dans la mesure où chaque site met en œuvre sa propre politique d'allocation de ressources. La figure 6.1 montre aussi certains services worker hors de tout site (des machines personnelles individuelles connectées à l'Internet) afin de rappeler qu'XtremWeb n'a pas besoin de prise en charge « cluster » pour ces services. Nous avons utilisé ce type de déploiement afin de faciliter l'expérimentation.

Nous avons mené cinq expériences : WISC-97, 97 processeurs au Wisconsin ; WL-113, 113 processeurs au Wisconsin et au LRI ; G-146, 146 processeurs à Grenoble ; WLG-270, 270 processeurs au Wisconsin, au LRI et à Grenoble ; et enfin WLG-451, 451 processeurs au Wisconsin, au LRI et à Grenoble.

La figure 6.2 montre les temps d'exécution par ordre décroissant pour chaque tâche. La courbe*G-146* a une pente nulle, ce qui dénote une stabilité du cluster de Grenoble ; cette uniformité est due à l'exécution des services worker sur un ensemble de machines homogènes (mémoire, CPU...). La courbe *WISC-97*, quant à elle, à deux paliers clairement distincts correspondant aux deux types de machines utilisées : des CPU à 533 Mhz et 900 Mhz. Les deux plateaux ne sont toutefois pas plats à cause de l'ordonnancement de Condor, qui tient compte des politiques d'activation des ressources. Contrairement au cluster de Grenoble où les ressources sont dédiées au calcul, les ressources utilisées par l'ordonnanceur Condor sont des ressources volontaires que l'ordonnanceur ne peut utiliser à sa guise. Enfin, les ressources réseaux ne sont pas les mêmes entre le service de coordination et Grenoble, et entre le service de coordination et le Wisconsin.

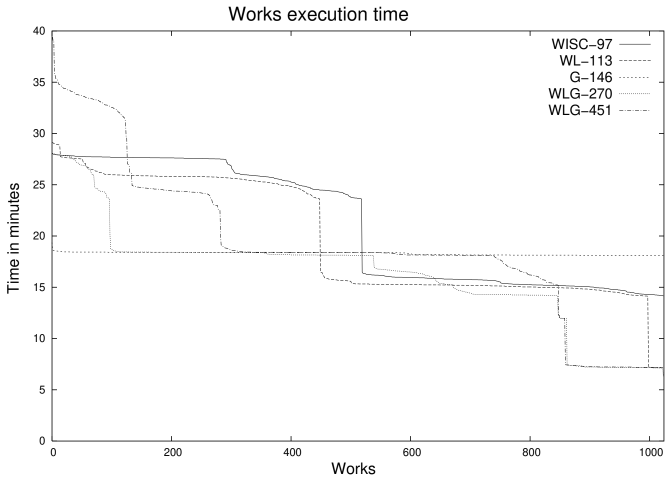

FIG. 6.2 – Temps d'exécution des tâches.

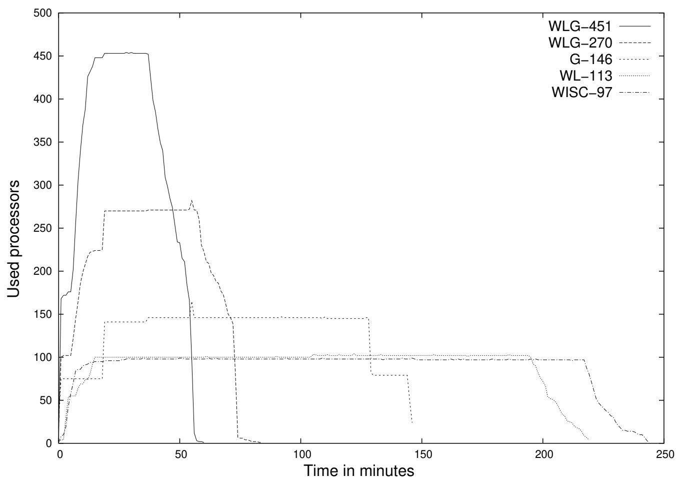

FIG. 6.3 – Utilisation des processeurs.

La figure 6.3 montre l'utilisation des ressources pour chaque expérience. Chaque courbe se décompose en trois étapes. La première correspond à la montée en puissance où les ressources sont allouées par les ordonnanceurs. On voit ensuite le palier où les ressources sont allouées et où la plate-forme donne toute sa puissance. Les courbes finissent par une phase descendante correspondant à la fin des expériences et à l'arrivée des derniers résultats.

Finalement, nous pouvons conclure que la plate-forme a rempli son rôle et que des tâches de physique peuvent être correctement calculées par notre plate-forme. Nous pouvons donc maintenant la mettre en production.

6.3.2 Simulation de gerbes

La mise en production de simulations pour Auger nécessite la prise en compte des habitudes des utilisateurs qui soumettaient, jusqu'à présent, leurs jobs sur des systèmes de type *batch*.

Nous avons inséré les binaires Linux et Windows de Aires dans notre plate-forme de production au LAL afin de pouvoir utiliser ces deux types de ressources de calcul. Les utilisateurs doivent fournir le fichier d'entrée pour chaque tâche. Plusieurs paramètres doivent être positionnés dans le fichier d'entrée, pour chaque simulation. On peut toutefois regrouper ces dernières par familles dont on différencie les membres en fonction de leur *RandomSeed* (valeur initiale du générateur aléatoire). Les autres paramètres, tel que la particule primaire, son énergie, son angle, étant figés pour une même famille. Un deuxième paramètre est le nom de la tâche (*TaskName*), qui permet aux utilisateurs de gérer leurs résultats. Enfin, le troisième paramètre utilisé est le nombre de gerbes souhaitées. Afin de faciliter l'utilisation de la plate-forme par nos utilisateurs, le client a été spécialisé et permet de générer automatiquement le nombre de tâches voulues, pour une famille de tâche donnée. L'utilisateur fourni un fichier d'entrée Aires qui définit les paramètres de la famille de tâches et le client génère automatiquement le *RandomSeed* pour chaque simulation qu'il insère dans la plate-forme, au titre de l'utilisateur. Pour des raisons d'équité d'utilisation de la plate-forme, le nombre de gerbes par tâche est forcé à une.

Certaines ressources sont fournies de manière « standard », par des centres de calcul traditionnels ; d'autres sont fournies par notre plate-forme XtremWeb. Quelles que soient les ressources utilisées, les simulations sont préparées grâce à la base de données d'Auger (*AugerDb*) gérées au CCIN2P3, à Lyon, et leurs résultats sont stockés dans les espaces Auger de ce même centre.

Le déploiement d'XtremWeb pour Auger s'architecture ainsi :
- la soumission de tâches est faite par les physiciens qui utilisent le service client d'XtremWeb à cet effet ;
- le service de coordination est installé au LAL ;
- les services worker sont déployés au LAL et à l'IPN0[12] (ainsi que quelques machines personnelles à domicile) ;
- un service client est installé à Lyon afin de récupérer les résultats obtenus et de les stocker dans les espace de stockage d'Auger. Dès que les résultats sont copiés en lieu sûr, les tâches sont automatiquement retirées de notre plate-forme.

La figure 6.4 montre la configuration des ressources de calcul utilisées pour les simulations *Monte Carlo*.

La figure 6.5 compare l'utilisation de XtremWeb dans la configuration LAL/IPNO

[12] L'Institut de Physique Nucléaire d'Orsay est partie prenante de XtremWeb pour Auger : http ://ipnweb.in2p3.fr

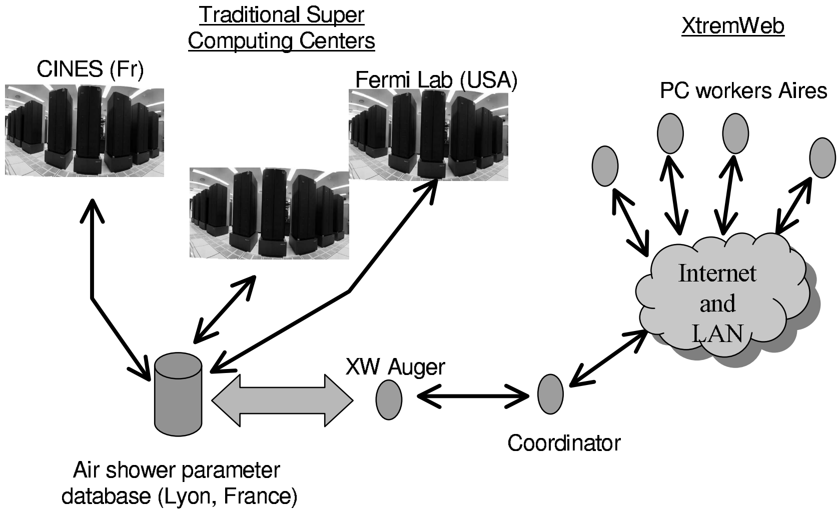

FIG. 6.4 – Configuration XtremWeb pour Auger.

comprenant une douzaine de machines avec le *CCIN2P3* pour générer des gerbes à partir de particules de fer ou de protons. Le programme de simulation utilisé ici est Aires. Cette figure montre le nombre de gerbes générées par heure pour un sous ensemble du total d'heures utilisées démontrant l'utilité d'une plate-forme comme XtremWeb. On voit qu'avec une douzaine de machines, XtremWeb génère quatre gerbes en moyenne, ce qui est bien sûr sans commune mesure avec les capacités de calcul du CCIN2P3, mais fournit malgré tout 10% du total des calculs par heure. XtremWeb est donc une source non négligeable de ressources, d'autant plus quand le CCIN2P3 est indisponible, comme pendant la période des heures 113 à 150, où seul la plate-forme de calcul global continue de fournir des résultats.

6.3.3 Reconstruction et analyse de données

Les programmes de reconstruction et d'analyse (*DPA*) ne peuvent bénéficier que de ressources de calcul bien spécifique, dans la mesure où ils nécessitent Linux avec les librairies *gcc3*. Notre plate-forme sait distinguer les ressources de calcul en fonction de leur système d'exploitation, mais ne possède aucune information quant aux librairies installées. Tenter d'exécuter une tâche sur une ressource dont on sait qu'elle n'offre pas l'environnement spécifique nécessaire à son exécution utiliserait une ressource de calcul inutilement ; d'autre part, une telle erreur n'est pas prise en compte par la plate-forme et les tâches ne seraient alors réellement jamais exécutées. Ce problème n'est pas une erreur en soit, ni dans le cadre d'XtremWeb, ni dans le cadre de n'importe quel environnement d'exécution (cluster, batch, grille...) Il a toujours été de la responsabilité des utilisateurs de suivre les recommandations des contraintes de la plate-forme, dans la mesure où ils ont le droit d'insérer leurs propres binaires (dans le cas inverse, ce problème ne se pose pas, bien sûr).

Dans le cadre d'un environnement hétérogène comme XtremWeb, nous avions

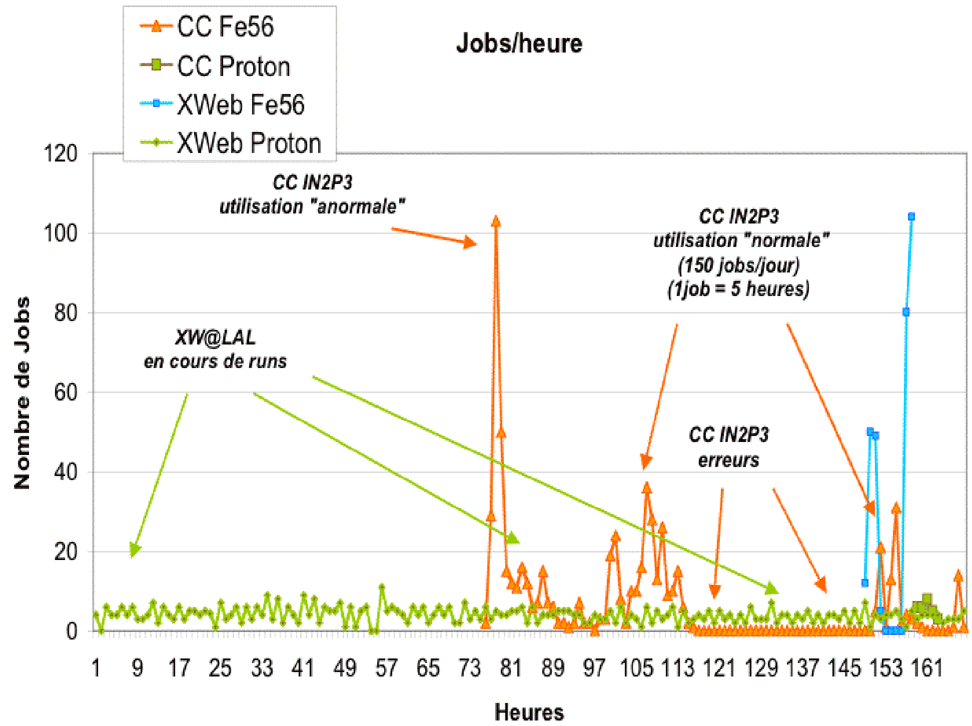

Fig. 6.5 – Tâches *Aires* calculées par XtremWeb et au CCIN2P3

deux possibilités afin d'essayer de résoudre ces problèmes liés aux contraintes applicatives. La première est de compiler différentes versions du programmes afin d'augmenter les chances de pouvoir utiliser différentes ressources de calcul. Après analyse de la situation par les physiciens/développeurs eux mêmes, celle solution n'a pas été retenue. La solution retenue a alors été de n'utiliser que les ressources susceptibles de pouvoir exécuter du code DPA.

Un autre problème à résoudre pour le déploiement de ces logiciels est leur taille, de l'ordre de 1.6Go (400Mo une fois compressés). Ils ont donc préalablement été installés sur les ressources de calcul afin d'éviter d'engorger le réseau.

Enfin, troisième problème, la reconstruction et l'analyse de données dépendent d'un nombre conséquents de paramètres qui doivent être générés pour chaque calcul ; ces paramètres font eux mêmes quelques centaines de méga-octets par tâche. Notre plate-forme n'est pas prévue pour gérer les fichiers utilisateurs, mais seulement des tâches. Elle n'est de toute façon pas dimensionnée pour contenir autant de données. Pour chaque tâche, les fichiers sont donc installés sur une ressource de stockage accessible par le Web. Les ressources utilisées doivent donc être configurées avec les droits d'accès au réseau adéquats, afin que les tâches distribuées puissent récupérer leurs fichiers.

En résumé :

1. nous déployons au préalable les logiciels nécessaires sur les ressources de calcul

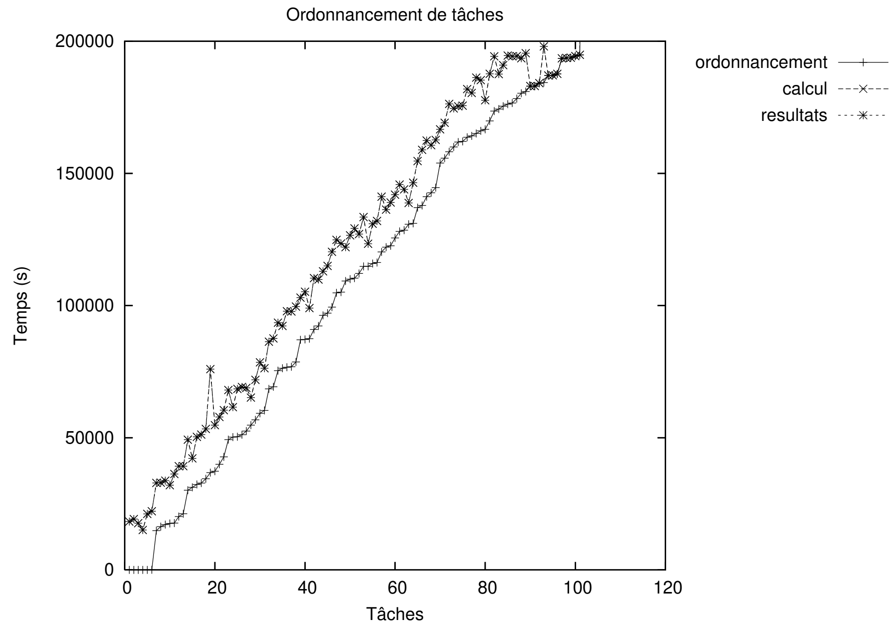

FIG. 6.6 – Simulation du détecteur sur 20 machines : ordonnancement des tâches.

compatibles (Linux avec gcc3) que nous utiliserons pour ce type de tâche. Cet ordonnancement précis des tâches sur un nombre limité de ressources est automatiquement opéré par la plate-forme elle même.

2. les fichiers de paramètres des tâches doivent être préalablement générés ; ce point n'est pas imposé par la plate-forme (que ce soit la notre, ou tout autre grille, ou encore un centre de calcul traditionnel) mais par la solution logicielle elle même.
3. ces fichiers sont stockés sur un serveur Web afin que les tâches qui seront déployées puissent les récupérer.
4. Enfin, les tâches sont ordonnancées par notre plate-forme ; elles téléchargent les fichiers de paramètres depuis le serveur Web prévu à cet effet et utilisent les logiciels pré-installés.

Nous montrons finalement la distribution d'une centaine de tâches sur douze machines équivalentes en terme de puissance (CPU et mémoire) ; les figures 6.6 et 6.7 montrent une prise en charge satisfaisantes et une bonne distribution de ces tâches.

6.4 Conclusion

Nous avons vu dans ce chapitre la complexité des besoins applicatifs de la physique des hautes énergies. Les programmes sont lourds en temps de calcul ainsi qu'en utilisation mémoire et disque. Ces programmes n'ont généralement pas été

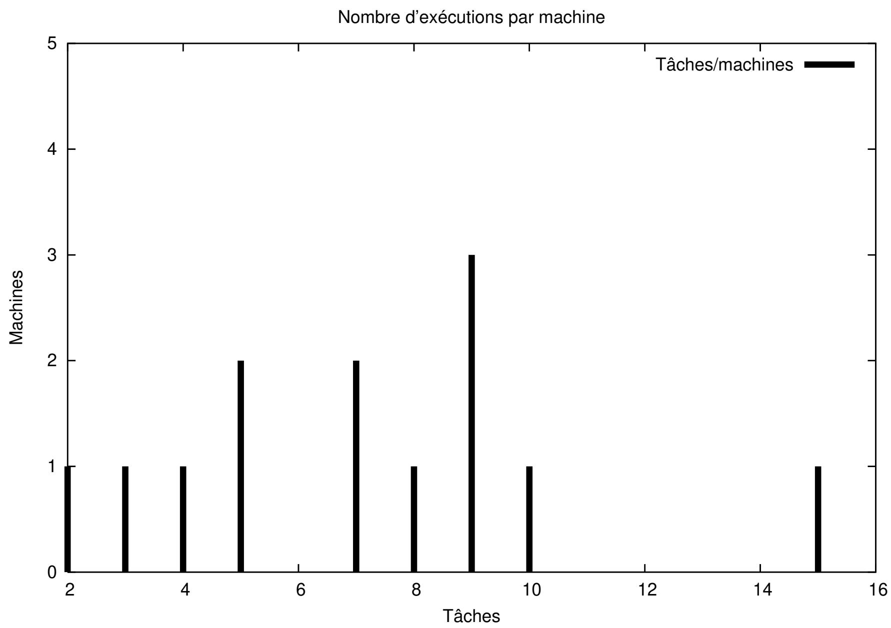

FIG. 6.7 – Simulation du détecteur sur 20 machines : distribution des tâches par machine.

prévus pour des contextes hétérogènes et sont extrêmement dépendants des types de ressources (système d'exploitation, dépendances inter-librairies). Il en résulte une certaine inefficacité d'utilisation de notre plate-forme que nous avons dû limiter à un seul type de ressource. Nous n'avons pas pu exploiter le potentiel de plusieurs dizaines d'ordinateur individuels Windows et Mac OS X dessiminés dans le laboratoire, parce que nous n'avons pas pu obtenir d'application de physique candidate.

Quoi qu'il en soit, malgré ces limitations, nous pensons que le concept est prometteur. Nous en voulons pour preuve les traces de monitoring des figures 6.8 et 6.9 qui montrent respectivement les tâches calculées par la plate-forme et la disponibilité des ressources, depuis décembre deux mille cinq, date d'installation du système de monitoring Ganglia[13]. On remarquera que les compteurs de Ganglia ne compte que ceux pour les machines connectées ; ceci explique le pic au delà des six milles tâches à la mi juin qui retombe ensuite dans un creux pour revenir aux seuil des six milles : entre le vingt juin et mi juillet, des machines n'étaient pas connectées et Ganglia n'a pas tenu compte de leurs compteurs pendant cette période. Nous voyons, avec ces figures, la puissance du concept de calcul global, malgré le nombre très limité de ressources mises à notre disposition.

Si les utilisateurs ont pu être dérouté par ce nouvel outil, certains n'utilisent plus qu'exclusivement cette ressource de calcul ; ils mettent alors en avant deux raisons principales. La première est que la complexité de mise en œuvre pour les programmes de physique des hautes énergies est similaire à tout autre système de calcul (cluster, grille institutionnelle). Sur notre plate-forme, comme sur les autres, il est nécessaire

[13]http ://ganglia.sourceforge.net

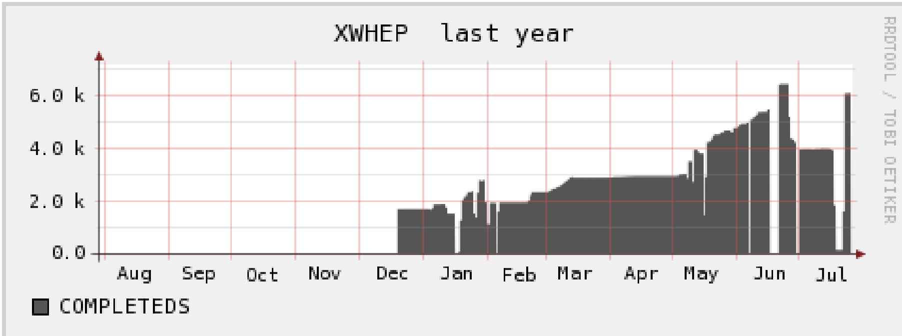

FIG. 6.8 – Utilisation de la plate-forme sur 6 mois.

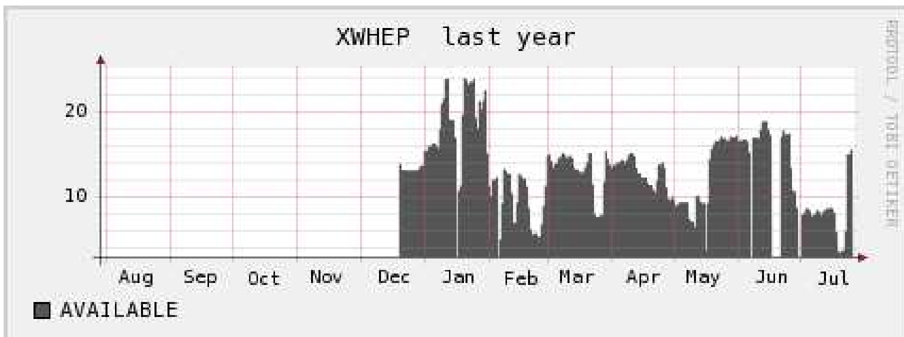

FIG. 6.9 – Disponibilité des ressources de la plate-forme sur 6 mois.

de définir un protocole de déploiement et de mise en route afin de résoudre, par exemple, l'accès aux données nécessaires aux exécutions. La seconde est relative à la facilité d'utilisation de la plate-forme grâce à son architecture basée sur les services offrant intrinsèquement la publication et l'instanciation à distance. Les utilisateurs apprécient particulièrement de soumettre et suivre à distance leurs travaux, ainsi que de pouvoir vérifier leurs résultats de la maison comme du bureau.

Chapitre 7

Conclusion

Les expériences et les mises en production des grilles ont montré la puissance et l'utilité du concept. Nous avons vu qu'il existe différents types de grilles de calcul : les plate-formes institutionnelles, pair à pair ou de calcul global. Toutes ces grilles ont le même but, mutualiser des ressources à grande échelle. Elles utilisent cependant des caractéristiques orthogonales pour y parvenir.

Le calcul global pair à pair tire son originalité d'intégrer les erreurs en tant que composante à part entière de ses pré-requis ; les ressources sont volatiles, l'environnement non sécurisé, les fautes massives. Nous considérons toutefois que ce modèle a maintenant fait ses preuves ; les résultats obtenus à ce jour sont très prometteurs et ils ouvrent le champ à de nombreuses études expérimentales à venir.

7.1 Perspectives

Dans la lignée directe des travaux présentés, et grâce à l'expérience acquises, nous considérons quatre champs d'étude.

Le modèle de programmation et architecture. Le langage Java, utilisé à la fois comme langage de programmation et environnement d'exécution, fait preuve de souplesse satisfaisante et de grande robustesse. Il nous a toutefois joué de nombreux tours, causes de pertes de temps non négligeables, tout particulièrement dans la gestion des applications natives du côté du service de calcul. Cet environnement n'étant pas prévu pour de telles opérations, il ne propose ni ne standardise rien de ce point de vue. Il a donc été entièrement à notre charge de développer et de mettre au point une gestion des process natifs, standardisée et indépendante des types de ressources. Cette gestion est aujourd'hui propre et entièrement satisfaisante.

Un autre point est la gestion de la mémoire utilisée. Bien que les environnements de programmation et d'exécution prévoient une gestion automatisée de la mémoire, l'usage a montré qu'il reste à la charge du programmeur de rester sur ses gardes de ce point de vue. De nombreux problèmes sont en effet apparus, notamment du côté du service de coordination.

Il conviendrait d'autre part, maintenant que notre plate-forme a fait ses preuves, de découpler les différents services inclus au niveau de ce dernier. Ainsi, les ressources en seraient d'autant allégées et les problèmes mieux séparés. Cela permettrait aussi de comparer une approche entièrement centralisée avec une autre, où les services seraient distribués.

Malgré ces petits problèmes, nous recommandons de garder Java comme environnement pour la suite de ce projet. Nous recommandons aussi d'utiliser le plus possible les technologies existantes et de ne se concentrer que sur le cœur de métier de la plate-forme : la distribution de services applicatifs natifs, ce qui inclut l'ordonnancement et le service de calcul. Il nous semble inutile de continuer à maintenir un serveur de résultats, un répertoire de services, ainsi que le service client, qui pouvent être avantageusement pris en charge par des technologies standards (serveur « apache » ou « TomCat », browser Web etc.).

Les connexions inter plates-formes. Les travaux présentés au chapitre 4 ont constitué un premier pas vers une collaboration constructive des différents types de plate-forme de calcul. Les mêmes expériences, avec les mêmes succès, ont été menées sur *EGEE*.

Nous pensons maintenant en mener d'autres afin que notre plate-forme puisse proposer une ressource de calcul à la grille EGEE. Il ne s'agira donc plus ici d'utiliser les ressources d'une grille pour prendre en charge un service d'une autre grille, mais d'utiliser conjointement, de manière standardisée et transparente, les ressources des deux grilles. La grille EGGE utilise différents systèmes d'ordonnancement (Condor, PBS, OAR...) afin de pouvoir utiliser différentes ressources de calcul (pool Condor, cluster de calcul...) Il conviendra donc de présenter le service client d'XtremWeb afin que la grille puisse utiliser notre plate-forme comme ressource de calcul. Les expérimentations à mener devront permettre de déterminer les performances possibles, mais aussi les types de décisions à prendre, relativement aux qualités intrinsèques de notre plate-forme où la faute d'une ressource n'est pas une erreur mais bien un pré-requis.

Le transport de protocoles standards. L'utilisation de protocoles (et donc de services applicatifs) standards est une demande forte de la communauté des utilisateurs. Les travaux présentés au chapitre 5 sont très prometteurs et devront être étendus et généralisés à tous les protocoles standards. Nous considérons que l'exemple présenté (SunRPC) est parmi ceux des plus lourds à mettre en place. D'autres, plus généraux, tels que les sockets TCP ou les packets UDP devraient nécessiter moins d'ingénierie.

Généraliser ce concept de transport de protocole permettra de prendre en charge n'importe quel type d'application de la famille client/serveur.

La mise en production. Nous avons vu la complexité des besoins applicatifs de la physique des hautes énergies. Les programmes sont lourds en temps de calcul ainsi qu'en utilisation mémoire et disque. Ces programmes n'ont généralement pas été prévus pour des contextes hétérogènes et sont extrêmement dépendants des types de ressources (système d'exploitation, dépendances inter-librairies).

Il en résulte une certaine inefficacité d'utilisation des plate-formes de grilles. Ces problèmes ne se limitent pas à notre plate-forme. Nombre d'utilisateurs utilisent la grille LCG (par exemple) en copiant des centaines de méga-octets d'archives de librairies diverses et variées téléchargées sur les ressources de calcul au moment de l'exécution de leurs tâches. Il en résulte une utilisation non optimum des espaces de stockage ainsi que de la bande passante. Ceci passe relativement inaperçu pour le moment, au vu des capacités mises en œuvre (téra-octets de stockage, milliers de processeurs), d'une part, et parce que cette grille n'est pour le moment pas pleinement utilisée (les expériences du LHC ne débuteront qu'en deux mille sept). De telles utilisations, si elles risquent toutefois d'être problématiques avec la montée en charge de l'utilisation des ressources, permettent de répondre dans l'immédiat aux besoins des utilisateurs.

Dans le cadre de ces contraintes applicatives, notre plate-forme peut répondre à ces mêmes besoins, avec les mêmes artifices d'utilisation. Ils sont toutefois plus visibles au vu de ses capacités : peu de processeurs et pas d'espace de stockage (la plate-forme n'intègre pas la notion de données). Les gains que notre plate-forme sont d'autre part amputés des ressources de type Windows et Mac OS X dans la mesure où ces plate-formes ne sont pas prises en compte dans les solutions logicielles d'Auger.

Un déploiement général au niveau du LAL de notre plate-forme reste toujours à faire, afin d'utiliser les ordinateurs de bureau (Windows et Apple) pour créer une ressource de calcul. Ce déploiement n'a pas eu lieu pour au moins deux raisons. La première est que nous n'avons donc pas trouvé de logiciels de physique compatibles avec des ressources de calcul de type PC de bureau. Ensuite, sans réelle volonté politique, on peut considérer qu'un tel déploiement est presqu'impossible. Les utilisateurs ne veulent pas, à priori, proposer leur ordinateur de bureau, pour des raisons principalement « psychologiques ». Ils se posent des questions sur l'intégrité de leurs ordinateurs qu'ils ne se posent en général pas pour installer des logiciels inconnus, payants (ou pire...). Nous avons animé un séminaire de présentation où sont librement venus les collaborateurs du LAL, pour leur présenter notre plate-forme et pour faire appel aux volontaires. On peut imaginer que le public présent était intéressé par cette solution, et pourtant, personne ne s'est porté volontaire.

On peut alors légitimement se poser la question de comprendre comment et pourquoi les systèmes pair à pair ont pu se développer jusqu'à agréger des millions de ressources. La piste principal pour répondre à cette question est celle du marketing et des gains pour chaque utilisateur. La plate-forme SETI@Home inclue la notion de point de participation et chacun peut voir les siens ; cette notion est tellement présente qu'on a même vu des challenges de participation entre les propriétaires des ressources. Quant aux plate-formes de distribution de fichier (comme Gnutella ou eMule), chaque utilisateur a un gain immédiat à se connecter.

Pour obtenir une plate-forme XtremWeb intéressante en termes de ressources agrégées, nous pouvons imaginer au moins deux pistes à explorer. La première est celle du marketing, en effet. Après avoir découvert la plate-forme et ses capacités, les utilisateurs se portent tous naturellement volontaires ; c'est là une vraie satisfaction et il conviendra donc de proposer notre solution au plus grand nombre. Cette approche impose d'investir dès le début dans un minimum de machines afin que tout nouvel utilisateur puisse obtenir des résultats immédiatement. Ainsi, satisfait et rassuré, il proposera volontiers d'ajouter ses propres machines dans le pool de ressources XtremWeb.

Une autre piste est celle du couplage avec les grilles institutionnelles dont les travaux préliminaires ont été présentés au chapitre 4.

7.2 Conclusion

Nos travaux, notamment la ré-architecture de la plate-forme, ont permis de valider XtremWeb en tant que plate-forme de calcul global d'expérimentation et de production.

Nous avons été amenés à développer et mettre au point cette plate-forme parce qu'il n'en existe aucune autre ayant les mêmes caractéristiques originales ; Condor ne peut être modifié car les sources ne sont pas disponibles ; BOINC (Seti@Home) ne permet pas de distribuer des services applicatifs sans les modifier (les recompiler avec l'API). Enfin, notre plate-forme contient tous les mécanismes nécessaires à la sécurité des applicatifs et de leurs résultats, d'une part, ainsi qu'à celle des ressources participantes d'autres part.

Nous avons montré par nos travaux, ainsi qu'avec la présentation succincte de travaux d'autres équipes de recherche, que la généricité et la dynamicité de notre plate-forme peut être avantageusement utilisées afin de mener à bien des études de recherche sur les plates-formes de calcul. Enfin, la mise en production pour le Laboratoire de l'Accélérateur Linéaire a clairement montré l'attrait qu'XtremWeb a pour des utilisateurs exigeants, bien que nous n'ayons pas pu le déployer à une échelle plus intéressante.

Chapitre 8

Annexes.

Le projet XtremWeb est composé d'un ensemble de paquets décrivant les différentes couches de son architecture. Il est disponible sur le site http ://www.xtremweb.net, sous forme d'un paquet contenant l'ensemble des fichiers sources (fichiers *Java* et *C++*), ou sous forme de paquets binaires prêts à l'utilisation.

Le paquet contenant les fichiers sources inclut les fichiers et les librairies nécessaires à la reconstruction des paquets binaires.

Il existe une famille de paquets binaires par type de service : une pour le service de coordination, une pour le service client et une pour le service de calcul. Chaque famille inclut un paquet binaire par plate-forme supportées qui sont au nombre de trois : *Linux* [14], *Windows* [15] de Microsoft et *Mac OS X* [16] d'Apple. Une plate-forme est dite *supportée* si l'ensemble des trois services décrivant le projet peut s'exécuter correctement sur cette plate-forme.

8.1 Structure du projet

Le projet est structuré suivant l'ensemble des répertoires suivants :

- `build` : contient les fichiers nécessaires à la reconstruction des paquets binaires (Cf chapitre 8.2) ;
- `classes` : contient les librairies utilisées par le projet ;
- `doc` : contient la documentation du projet ;
- `misc` : contient un ensemble de fichiers divers (les icônes, les fichiers de configuration etc.) ;
- `php` : contient la structure des pages web ;
- `src` : contient l'ensemble des fichiers sources (Java et C++).

[14] `http ://www.linux.org`
[15] `http ://www.microsoft.com`
[16] `http ://www.apple.com/macosx`

8.2 Reconstruction et installation

Il est possible de reconstruire les paquets binaires grâce au paquet contenant les fichiers sources. Cette reconstruction est indépendante de la plate-forme utilisée; on peut indifféremment reconstruire les paquets sur n'importe quelle plate-forme supportée.

Il est probable qu'on puisse aussi bien le faire sur d'autres plates-formes (*FreeBSD*, *Solaris*, *OSF* etc.) ; aucun test complet n'a toutefois été pratiqué et nous nous limitons ici à la reconstruction sur un des OS supportés.

La reconstruction se fait depuis le répertoire `build` qui contient les sous répertoires suivants :

- `installers` contient les fichiers nécessaires à la génération de paquets d'installation en fonction des plates-formes visées.
- `platforms` contient les libraires natives pour différentes plates-formes. Ces librairies sont automatiquement incluses dans les paquets XtremWeb.
- `rpm` contient les fichiers de spécification nécessaires à la génération des paquets d'installation RPM pour Linux.
- `stand-alone` est utilisé lors de la génération des paquets XtremWeb pour inclure toutes les librairies nécessaires à la génération des paquets XtremWeb.

La reconstruction nécessite les outils suivants :

- le J2SDK : l'environnement de développement *Java* de *Sun*. La version 1.4 ou supérieure est nécessaire;
- l'outil *make*;
- un compilateur *C++*;
- un interpréteur de commande (ou *shell*).

Parmi les plates-formes supportées, Linux et Mac OS X possèdent ces outils; il n'est donc pas nécessaire de les installer spécifiquement. La plate-forme Windows, quant à elle, ne les possède pas; il convient donc de les installer. L'interpréteur de commande fourni avec Windows ne peut être utilisé pour reconstruire le projet. Il est nécessaire d'installer cygwin [17] qui propose un environnement complet, semblable à ceux qu'on trouve sous Linux.

8.2.1 Préparation

Pour construire les paquets binaires, les quatre étapes suivantes sont nécessaires :

1. installer Java. Pour la suite nous supposons que Java est installé dans $/opt/j2sdk1.4.2$;

2. positionner la variable d'environnement `JAVA_HOME` : elle doit contenir le répertoire où Java a été installé. Avec l'interpréteur de commandes *bash*, il faut

[17] `http ://www.cygwin.com`

exécuter la commande :

```
export JAVA_HOME=/opt/j2sdk1.4.2
```

Avec *csh*, il faut exécuter la commande :

```
setenv JAVA_HOME /opt/j2sdk1.4.2
```

3. remonter au répertoire `build` dans lequel le paquet sources a été installé ; en supposant qu'il soit installé dans `/opt/XtremWeb`, il faut exécuter la commande :

```
cd /opt/XtremWeb/build
```

4. éditer le fichier `build.conf` et positionner les variables suivantes :

   4.1 `xtremweb.admin.login` décrit le nom de l'administrateur de la plate-forme. Grâce à ce login, on pourra gérer la plate-forme : insérer et supprimer des utilisateurs, des applications etc. Ce login peut être librement choisi.

   4.2 `xtremweb.admin.password` contient le mot de passe de cet administrateur. Ce mot de passe peut être librement choisi.

   4.3 `xtremweb.admin.email` contient l'adresse électronique de cet administrateur.

   4.4 `xtremweb.admin.fname` contient le prénom de cet administrateur.

   4.5 `xtremweb.admin.lname` contient le nom de cet administrateur.

   4.6 `dispatcher.host` décrit la liste des serveurs XtremWeb formant le service de coordination. Cette variable contient une liste d'adresses de serveurs séparées par des virgules ; ces adresses peuvent être résolues ou non (adresses IP). Voici un exemple :

```
dispatcher.host=www.lri.fr,192.168.0.1,xtremweb.lri.fr
```

   Cette variable est nécessaires à tous les services pour construire une plate-forme XtremWeb. C'est grâce à elle que le service de coordination peut se mettre en place, et que les services client et worker peuvent se connecter au service de coordination.

   4.7 `db.host` contient l'adresse (résolue ou non) du serveur de base de données (*DB*).

   4.8 `db.system` décrit le type de DB. Les valeurs possibles sont `mysql`, `hsqldb` et `hsqldb :mem`. Pour utiliser **mysql** [18] il faut que ce système DB soit installé sur le serveur de DB. Le système **hsqldb** [19] est entièrement écrit en Java ; les librairies nécessaires sont incluses dans le paquet source ainsi que dans le paquet binaire du service de coordination. L'option **hsqldb :mem** décrit l'utilisation du système **hsqldb** avec le moteur `MEMORY` (Cf *db.engine*).

[18] `http ://www.mysql.org`
[19] `http ://www.hsqldb.org`

4.9 `db.engine` décrit le moteur de la DB MySQL. Les valeurs possibles sont celles décrites dans la documentation MySQL ; on y trouve entre autre **MEMORY**, **MyISAM**, **InnoDB**... Le moteur **MEMORY** est un cas très spécial à utiliser avec précaution : ce moteur n'enregistre rien sur aucun espace de stockage non volatile (disque...). L'arrêt du serveur entraîne donc la perte totale des données.

4.10 `db.su.login` contient le login de l'administrateur de la DB. Grâce à ce login, on peut gérer la DB : insérer des utilisateurs, des bases et des tables de données.

4.11 `db.su.password` contient le mot de passe de cet administrateur.

Ces informations sont nécessaires afin de pouvoir installer la base de données. A défaut de les obtenir, l'intervention d'un administrateur est nécessaires afin que celui-ci installe une DB et un utilisateur ayant les droits d'administration utiles sur celle ci (ajout/suppression de tables etc.)

Ces deux variables (`db.su.login` et `db.su.password`) définissent alors cet utilisateur et son mot de passe.

4.12 `db.name` contient le nom de la DB. Ce nom peut être choisi librement.

4.13 `install.dir` contient le répertoire d'installation des paquets binaires. Les droits d'écriture dans ce répertoire sont nécessaires à l'installation.

4.14 `install.www.dir` contient le répertoire d'installation des pages PHP. Les droits d'écriture dans ce répertoire sont nécessaires à l'installation. L'accès à la plate-forme par le Web est optionnel. Toutefois ce répertoire est utilisé comme répertoire de travail pour le service de coordination : il y stocke les fichiers binaires des applications, ainsi que les fichiers relatifs aux tâches.

4.15 `ganglia.www.dir` contient le répertoire d'installation des pages PHP de `Ganglia` [20]. Les droits d'écriture dans ce répertoire sont nécessaires à l'installation. `Ganglia` est un système de monitoring. Son utilisation avec XtremWeb est optionnelle. Elle permet de surveiller les services worker ; le service de coordination est configuré pour fournir les informations de monitoring à `ganglia`.

4.16 `xw.passwordPass` contient une phrase pour crypter les données. Cette phrase peut être librement choisie.

4.17 `debug` contient mode de compilation. Les valeurs possibles sont `on` et `off`.

4.18 `logger.level` contient le mode de journalisation. Les valeurs possibles sont `debug` (le mode le plus verbeux), `info`, `warn` et `error` (le mode le moins verbeux). Attention, le mode `debug` peut générer des fichiers de journalisation de grosse taille.

[20] `http ://ganglia.sourceforge.net`

Le tableau 8.1 résume les différentes variables.

Nom de la variable	Description	Valeur et type	Service concerné
debug	mode de compilation	`on`, `off`	compilation
install.dir	répertoire d'installation		installation
db.su.login	administrateur de la DB et son mot de passe.		installation et coordination
db.su.password			
db.system	type de base de données	`mysql`, `hsqldb`, `hsqldb :mem`	coordination
db.engine	moteur de la DB	`MEMORY`, `MyISAM`, `InnoDB` etc.	
db.host	serveur de la DB		
db.name	nom de la DB		
install.www.dir	répertoire des fichiers		
dispatcher.host	serveurs XtremWeb	liste à virgules	client, worker et coordination
logger.level	mode de journalisation	`debug`, `info`, `warn`, `error`	
xtremweb.admin.login	nom de l'administrateur		client
xtremweb.admin.password	son mot de passe		
xtremweb.admin.email	son adresse électronique		
xtremweb.admin.fname	son prénom		
xtremweb.admin.lname	son nom		
xw.passwordPass	une phrase pour crypter		
ganglia.www.dir	répertoire `ganglia`		ganglia

TAB. 8.1 – Les variables de reconstruction et d'installation du fichier `build.conf`.

8.2.2 Reconstruction

Après avoir défini les variables du fichier `build.conf` (Cf tableau 8.1), les paquets XtremWeb peuvent être construits. Ces actions sont facilitées grâce à l'utilisation de l'outil `make`. Cet outil permet de gérer les paquets grâce à une description de cibles (ou actions) fournies dans le fichier `Makefile`. Les différentes cibles possibles pour reconstruire et installer les paquets XtremWeb sont décrites ci-après :

- `distrib` permet de construire les paquets.
- `clean` permet d'effacer les paquets. Une construction complète sera alors nécessaire.
- `install` permet d'installer les paquets

- `rpmworker` créer un paquet d'installation `RPM` pour le worker.
- `rpmserver` créer un paquet d'installation `RPM` pour le serveur.
- `doc` génère la documentation *JavaDoc*.
- `installDB` créer et installe la base de données.
- `removeDB` efface la base de données. Seules les tables sont effacées, la base de données elle même doit être effacée "à la main".
- `uninstal` désinstalle les paquets.
- `uninstallAll` désinstalle les paquets et la base de données.

En résumé, la compilation complète des paquets binaires nécessitent les actions suivantes (dans l'ordre) ;

1. make clean
2. make installDB
3. make distrib

Le tableau 8.2 résume les dépendances entre les différentes cibles.

Action	Pré-requis
doc	aucun
uninstal	aucun
uninstallAll	aucun
clean	aucun
removeDB	aucun
installDB	aucun
distrib	installDB
install	distrib
rpmworker	install
rpmserver	install

TAB. 8.2 – Les dépendances des phases de reconstruction.

8.2.3 Installation

Une fois les paquets binaires reconstruits, l'installation se fait avec la commande :

```
make install
```

Les différents services peuvent être immédiatement démarrés sur la machine où ils ont été installés. L'exécution des services sur d'autres machines est présentée dans le paragraphe 8.3 relatif au déploiement.

En supposant que votre fichier `build.conf` contienne la ligne suivante ;

```
install.dir=/opt
```

Les paquets ont alors été installés dans :

```
/opt/XtremWeb-1.7.0
```

(si une autre version a été téléchargée , il faut remplacer “1.7.0” par le numéro de la version téléchargée.)

Ce répertoire contient les quatre sous répertoires suivants :

1. `bin` contient les scripts *bash*, *perl* et *win32*, permettant de lancer, contrôler et arrêter les services :

   1.1 `xtremwebconf.sh` est un script *bash* utilisé par tous les autres scripts ; il détermine la configuration de chaque service (il retrouve le fichier de configuration pour le service demandé, la version de *Java* etc.).

   1.2 `xtremweb` est un script *bash* utilisé par tous les autres scripts ; c'est un outil générique de contrôle des services.

   1.3 `xtremweb.server` est un script *bash* contrôlant l'exécution du service de coordination.

   1.4 `xtremweb.worker` est un script *bash* contrôlant l'exécution du service worker.

   1.5 `xtremweb.client` est un script *bash* contrôlant l'exécution du service client.

   1.6 `xtremweb.rpc` est un script *bash* contrôlant l'exécution du service SunRPC sur XtremWeb.

   1.7 `xwapps` est un script *bash* utilisant le service client pour récupérer la liste des applications installées sur la plate-forme.

   1.8 `xwgui` est un script *bash* ouvrant la fenêtre graphique du service client.

   1.9 `xwresult` est un script *bash* utilisant le service client pour récupérer la liste des résultats des jobs ayant été correctement exécutés.

   1.10 `xwrm` est un script *bash* utilisant le service client pour supprimer des jobs.

   1.11 `xwstatus` est un script *bash* utilisant le service client pour récupérer la liste des jobs.

   1.12 `xwtasks` fait la même chose, mais affiche aussi le service de calcul qui a pris en charge les jobs.

   1.13 `xwsubmit` est un script *bash* utilisant le service client pour soumettre un nouveau job.

   1.14 `xwusers` est un script *bash* utilisant le service client pour récupérer la liste des utilisateurs installées sur la plate-forme.

   1.15 `xwworkers` est un script *bash* utilisant le service client pour récupérer la liste des services workers de la plate-forme.

   1.16 `xtremweb.monitor` est un script *bash* contrôlant l'exécution de `xtremweb.monitor.pl`.

1.17 `xtremweb.monitor.pl` est un script *perl* surveillant l'utilisation de la mémoire utilisée par le service worker. S'il consomme plus de 90% de la mémoire totale (*swap* compris), il est arrêté.

1.18 `xtremweb.ganglia` est un script *bash* contrôlant l'exécution de `xtremweb.gmond.pl`.

1.19 `xtremweb.gmond.pl` est un script *perl* récupérant les informations de monitoring relatives aux services worker de la plate-forme. Ce services est accessible par connexion *TCP* sur le port 8694, par défaut. Ce service doit être installé sur une machine ayant accès à la base de données XtremWeb.

1.20 `xwrmdb.sql` est un script *SQL* effaçant la base de données.

1.21 `xwsetupdb.sql` est un script *SQL* configurant une nouvelle base de données.

1.22 `InstallXWDispatcher-NT.bat` est un script *win32* installant le service de coordination en tant que service NT.

1.23 `UninstallXWDispatcher-NT.bat` est un script *win32* désinstallant le service de coordination.

1.24 `InstallXWWorker-NT.bat` est un script *win32* installant le service worker en tant que service NT.

1.25 `UninstallXWWorker-NT.bat` est un script *win32* désinstallant le service worker.

1.26 `Wrapper.exe` est un programme *win32* contrôlant l'exécution de service NT.

2. `conf` contient les fichiers de configuration.

   2.1 `wrapper-server.conf` et `wrapper-worker.conf` sont les fichiers de configuration de `Wrapper.exe`.
   Ils ne doivent pas être modifiés.

   2.2 `xtremweb.server.conf` est le fichier de configuration du service de coordination.
   Il est détaillé au chapitre 8.4.1.

   2.3 `xtremweb.worker.conf` est le fichier de configuration du service worker.
   Il est détaillé au chapitre **??**.

   2.4 `xtremweb.client.conf` est le fichier de configuration du service client.
   Il est détaillé au chapitre **??**.

3. `doc` contient la documentation.
4. `lib` contient les fichier des librairies.

8.3 Déploiement

Le déploiment, dans son aspect général, a été abordé au chapitre 3.3.7. Nous allons ici nous intéresser à la génération des paquets binaires utilisés pour le déploiement.

Le déploiement concerne les trois services : coordination, worker et client. Le déploiement est fortement lié aux environnements sur lesquelles ces services doivent être déployés. Nous nous concentrerons sur les environnements supportés dans la version actuelle d'XtremWeb : Linux, Mac OS X et Windows.

Chacune de ces trois plates-formes proposent leurs propres systèmes de déploiement et de gestion de paquets. Il n'est malheureusement pas possible de générer des paquets inter plates-formes, ni de faire de la cross génération (par exemple de générer des installeurs Windows sur une station Linux).

Nous détaillons donc la génération des installeurs de paquets, en fonction de la plate-forme visée, dans les trois sections suivantes.

Les services worker et coordinateur sont plus que deux applications pour la ressource sur laquelle ils s'exécutent : ils s'intègrent en tant que services de la ressource elle même. La ressource gère donc ces services comme n'importe quel service local.

A cette fin, ces deux services nécessitent une installation *système* sur la ressource. Le paquet source d'XtremWeb contient les fichiers nécessaires à la génération des paquets d'installation pour ces deux services, et ce, pour les trois environnements supportés.

Le service client, pour sa part, ne s'intègre pas en tant que service sur les ressources ; c'est une application comme les autres et il reste à la charge de l'utilisateur de le gérer en tant que tel. Le paquet source d'XtremWeb ne contient donc aucune définition de paquet d'installation pour ce service.

8.3.1 Linux

Le système de déploiement et de gestion de paquets choisi pour déployer XtremWeb sur des stations linux est RPM [21].

Deux cibles du `Makefile` permettent de construire les fichiers *RPM* pour le service de coordination et le service worker afin de faciliter et d'automatiser leur installation. Les fichiers de spécification RPM se trouvent dans `build/rpm`.

Pour les générer, il est nécessaire de construire et d'installer les paquets XtremWeb sur la machine utilisée à la construction des paquets. Après reconstruction et installation des paquets, les commandes suivantes peuvent être exécutées (attention, la génération de fichiers RPM nécessite certains droits ou configuration. La documentation de RPM est disponible sur le Web pour plus d'informations) :

- make rpmserver
- make rpmworker

Ces deux commandes génèrent les fichiers suivants (dans le répertoire `build`) :

- rpm/RPMS/noarch/XtremWeb-server-1.7.0-1.noarch.rpm

[21] `http ://www.rpm.org`

– rpm/RPMS/noarch/XtremWeb-worker-1.7.0-1.noarch.rpm

Si une autre version a été téléchargée, il faut remplacer “1.7.0” par le numéro de la version téléchargée. Le dernier chiffre (ici, ”1”) est le numéro de *release* de votre fichier RPM.

Ces fichiers *RPM* contiennent les librairies et les scripts nécessaires à l'installation, la gestion et le lancement automatique des services. Ces derniers sont lancés et disponibles dès leur installation.

8.3.2 Mac OS X

Les paquets pour Mac OS X doivent être générés sur une machine *Apple*.

Génération du paquet Mac OS X pour le service worker. Le paquet source d'XtremWeb contient un répertoire `build/installers/macosx` décrivant la définition des paquets d'installation pour Mac OS X (Figure 8.1).

Le système d'exploitation Mac OS X propose un système de déploiement et de gestion de paquets. L'environnement de développement contient une application, le *PackageMaker*, qui permet de créer des paquets Max OS X. On doit donc générer le paquet Mac OS X grâce à la définition fournie dans le paquet source d'XtremWeb.

FIG. 8.1 – Le projet de paquet Mac OS X pour le service worker.

En double cliquant sur l'icône de la figure 8.1, l'application *PackageMaker* s'ouvre ; il convient alors de configurer le paquet avec les paramètres souhaités (Cf figure 8.2).

La génération crée un paquet Mac OS X (Cf figure 8.3) qu'il convient alors de distribuer sur les machines de ce type afin de déployer le service worker de la plate-forme.

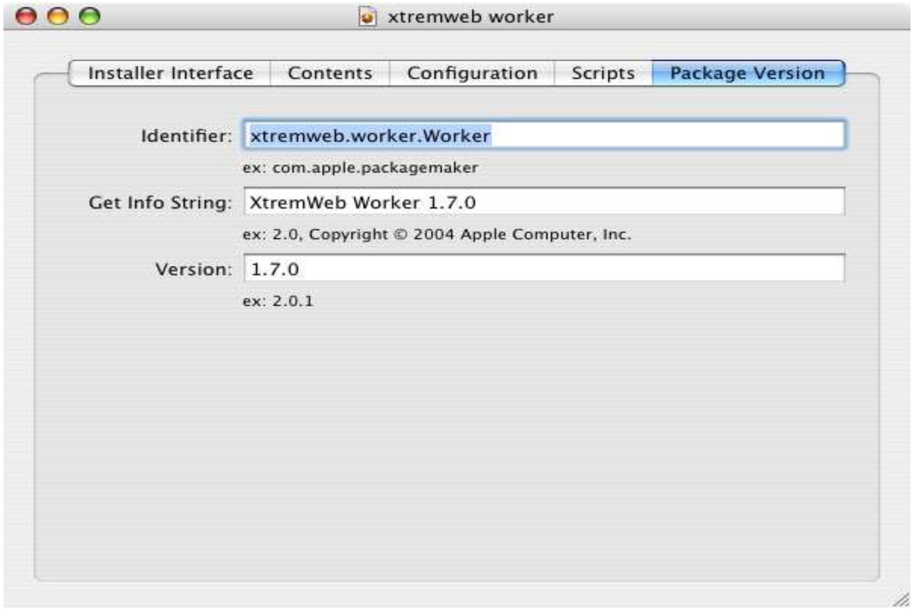

FIG. 8.2 – Le versionning nécessaire à la génération du service worker.

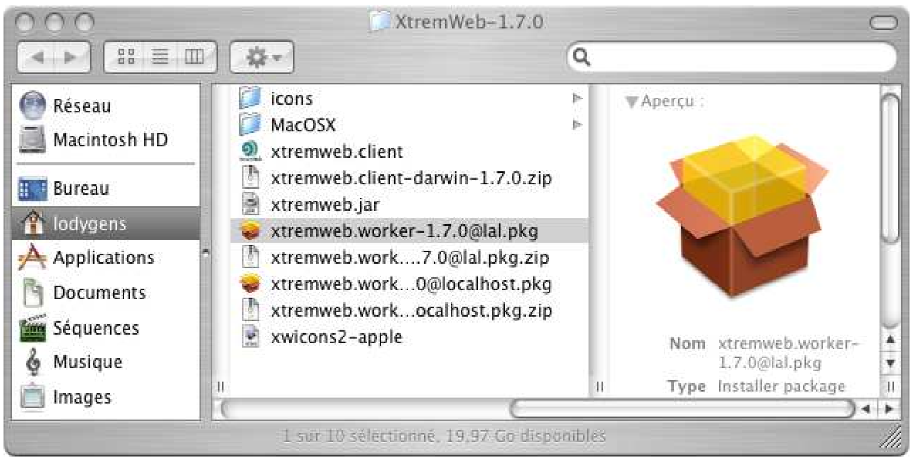

FIG. 8.3 – Le paquet Mac OS X pour le service worker.

Génération de l'application Mac OS X pour le service client. Le service client ne nécessite pas de paquet d'installation spécifique. La construction des paquets binaires (en faisant `make install`) a automatiquement crée une application Mac OS X native dans le répertoire *build/installers/macosx/installer/PckRoot/Applications* ainsi qu'une archive (*build/installers/macosx/installer/PckRoot/Applications/xtremweb.client-macosx-1.7.0.tgz*). Cette archive a été installée dans les pages de téléchargement (définies par la variable `install.www.dir`).

Ainsi, les utilisateurs souhaitant utiliser le service client peuvent télécharger ce fichier archive, l'installer sur leur ressource locale et lancer le service.

8.3.3 Windows

Les paquets pour Windows doivent être géréres sur une machine *Windows*.

Le système d'exploitation Windows propose un système de déploiement et de gestion de paquets.

Le paquet source d'XtremWeb contient un répertoire `build/installers/win32` décrivant la définition des paquets d'installation pour Windows. Cette définition doit être fournie à une application de génération de paquets afin de générer les paquets d'installation des services XtremWeb pour cet OS.

Génération du paquet Windows pour le service worker. L'application *Wyse* est ici utilisée pour illustrer notre propos.

Les figures 8.4, 8.5 et 8.6 montrent la configuration nécessaire à la génération du paquet d'installation Windows pour le service worker.

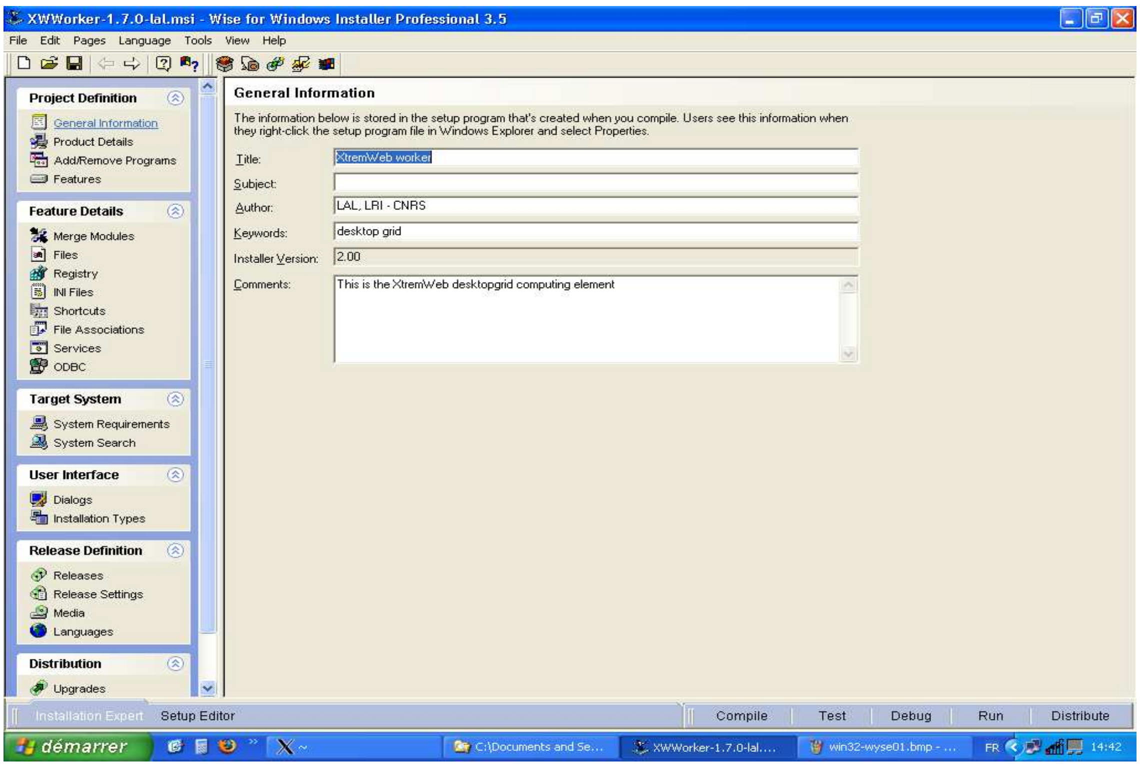

FIG. 8.4 – Informations générales nécessaires à la génération du service worker pour Windows.

Afin que le service worker soit automatiquement démarré et arrêté, il faut configurer sa gestion en cliquant sur *"service"* dans le menu *"feature details"* à gauche de l'écran. Les figures 8.7 et 8.8 illustrent la configuration de la création et du contrôle du service worker. La création du service worker exige la détermination de l'appli-

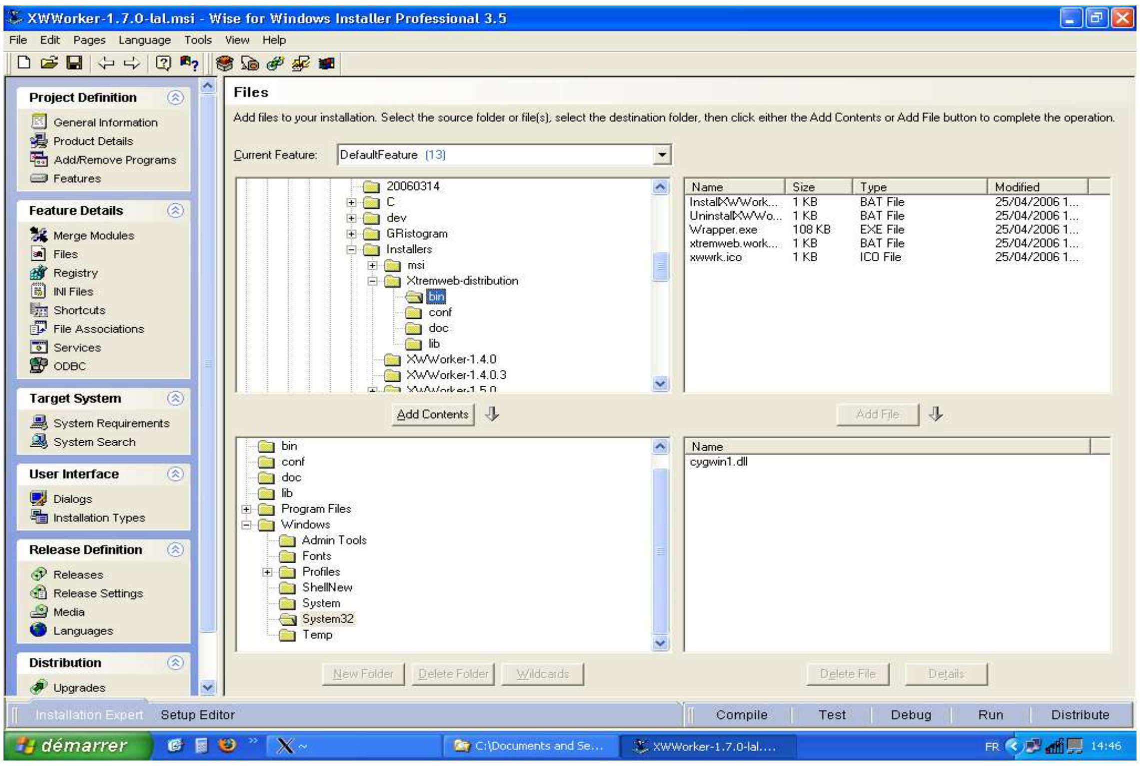

FIG. 8.5 – Fichiers nécessaires à la génération du service worker pour Windows.

cation du service : il faut sélectionner l'application *Wrapper.exe* dans la liste des fichiers du service (Cf figure 8.5).

Génération de l'application Windows pour le service client. Le service client ne nécessite pas de paquet d'installation spécifique. La construction des paquets binaires (en faisant `make install`) a automatiquement crée une application Windows native dans le répertoire `build/installers/win32`, ainsi qu'une archive (`build/installers/win32/xtremweb.client-win32-1.7.0.zip`). Cette archive a été installée dans les pages de téléchargement (définies par la variable `install.www.dir`).

Ainsi, les utilisateurs souhaitant utiliser le service client peuvent télécharger ce fichier archive, l'installer sur leur ressource locale et lancer le service.

8.4 Utilisation

L'utilisation des services est standardisée selon les environnements des ressources sur lesquelles ils sont installés. Les services worker et coordinateur sont gérables comme tout services de la ressource ; à cette fin, ils doivent suivre les standards de gestion des services selon la ressource.

Le service client n'étant pas inséré comme service système, mais comme une

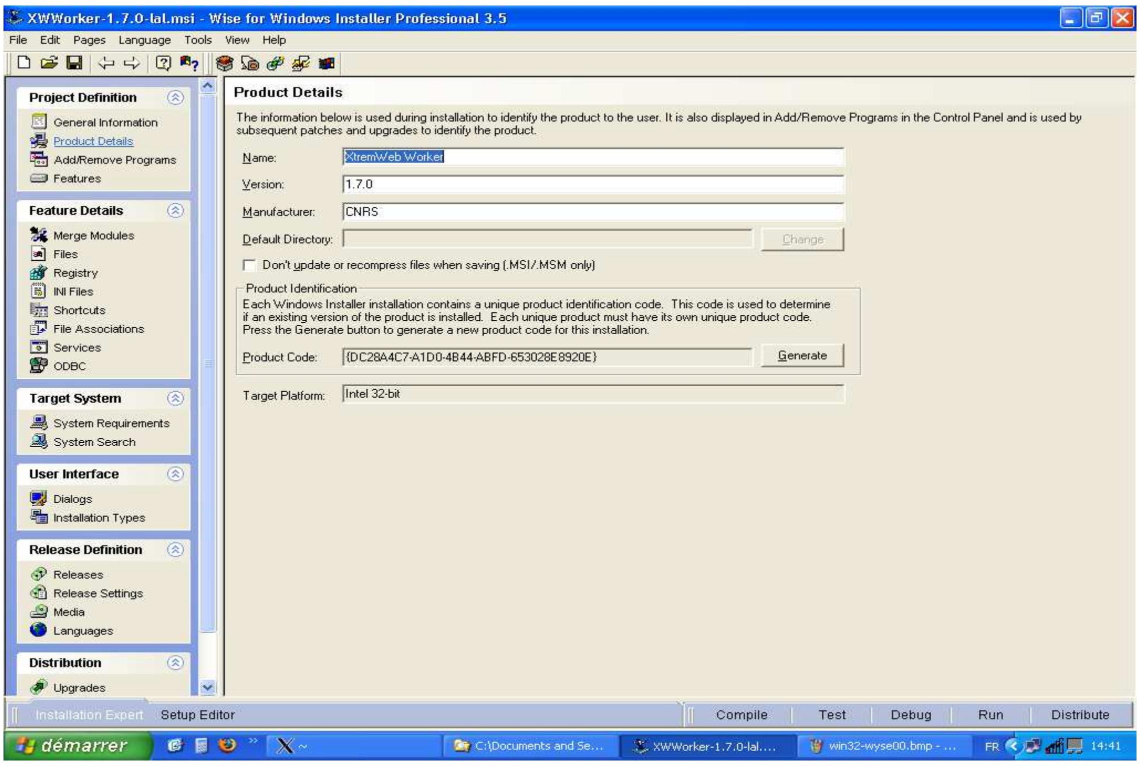

FIG. 8.6 – Versionning nécessaire à la génération du service worker pour Windows.

simple application, son utilisation se résume à l'utilisation d'une application en fonction des caractéristiques de la ressource.

8.4.1 Le service de coordination

Le service de coordination est indépendant de la plate-forme. Toutefois dans la version actuelle, seule la plate-forme *Linux* est supportée : le paquet binaire du service de coordination ne fournit que les scripts de gestion pour cette plate-forme.

Le paquet d'installation pour le service de coordination installe les scripts dans `/etc/init.d/`. Le script de gestion du service de coordination est `xtremweb.server`. Il accepte les paramètres décrits dans le tableau 8.3.

Paramètre	Action
console	démarre le service en premier plan
start	démarre le service en arrière plan
stop	arrête le service
restart	redémarre le service en arrière plan
status	affiche l'état du service

TAB. 8.3 – Les paramètres de gestion des services de coordination et worker sous Linux.

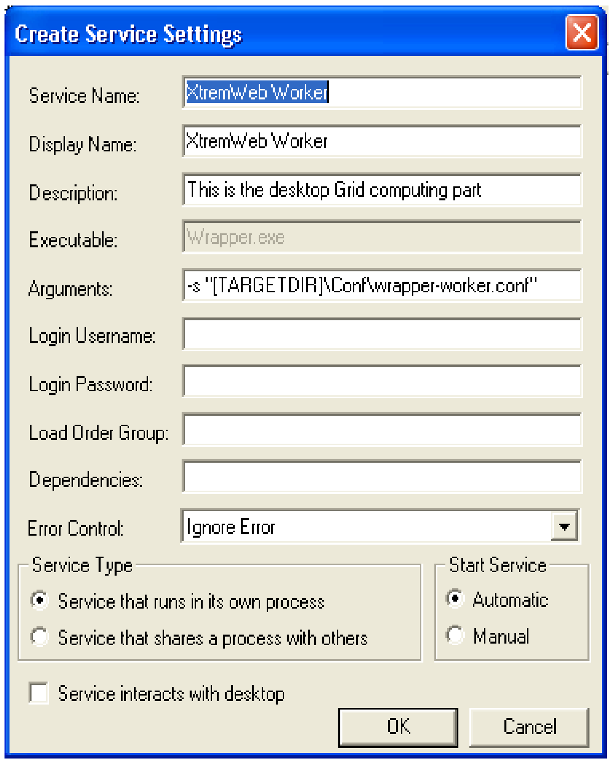

FIG. 8.7 – Création du service worker en tant que service Windows.

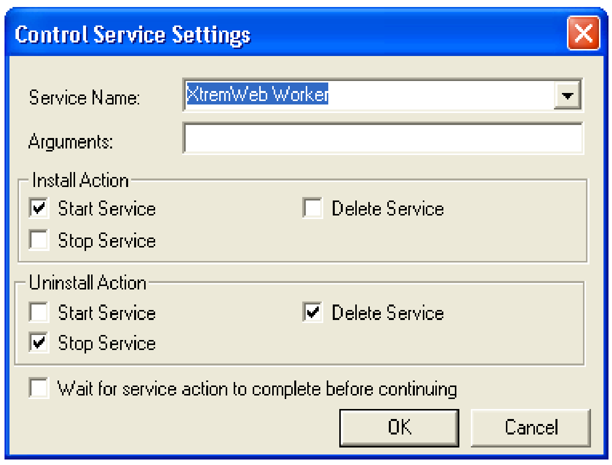

FIG. 8.8 – Contrôle du service worker en tant que service Windows.

8.4.2 Le service worker

Le service worker doit être installé comme un service système. Avec les environnements Linux et Mac OS X, le script `xtremweb.worker` permet de gérer le service worker. Ce script accepte les paramètres décrits dans le tableau 8.3.

Sous Windows, la gestion des services systèmes se fait par la "Gestion de l'ordinateur". On accède à cette gestion en cliquant avec le bouton droit de la souris, sur l'icône du "Poste de Travail" ; il faut choisir l'option "Gérer" (Cf figure 8.9)

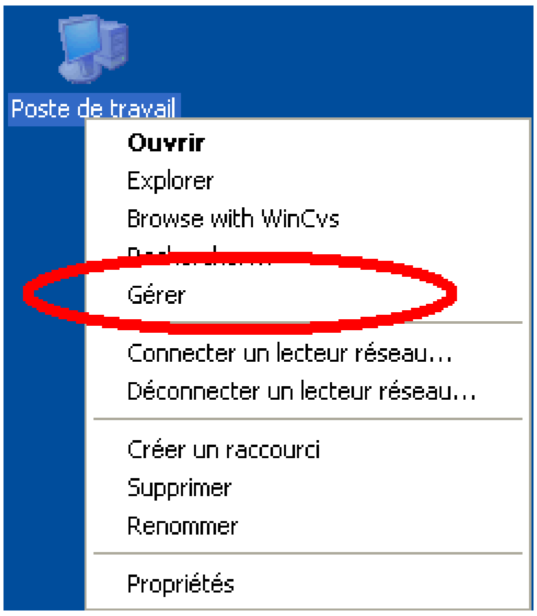

FIG. 8.9 – Gestion des services Windows.

Il faut aller chercher le service XtremWeb Worker dans la liste des services et cliquer sur "démarrer", "arrêter" ou "redémarrer" en fonction de l'action voulue (Cf figure 8.10).

8.4.3 Le service client

Le service client n'est pas installé comme service système, mais comme application. Le service client est indépendant de la plate-forme et les action suivantes sont donc applicables quelque soit le type de ressource utilisée.

La fenêtre principale du service client est présentée sur la figure 8.11. Elle possède sept pages d'affichage ; les pages *Jobs*, *Apps*, *Users*, *Usergroups*, *Hosts*, *Tasks* et *Works*. On ne peut afficher qu'une seule page à la fois en sélectionnant l'onglet correspondant. Ces pages servent respectivement à gérer les jobs de l'utilisateur, les services applicatifs et les utilisateurs du service de coordination, les ressources worker, les *travaux* et les *tâches* correspondant aux jobs. Les pages *Tasks* et *Works* ne sont pas affichées, par défaut ; il faut pour cela utiliser le menu "View" comme expliqué plus bas.

Les pages d'affichages sont toutes séparées en trois parties. Une partie en haut qui contient une barre de boutons permettant d'effectuer certaines actions ; une partie en bas contient une jauge qui affiche l'état d'avancement de l'action en cours. Le reste

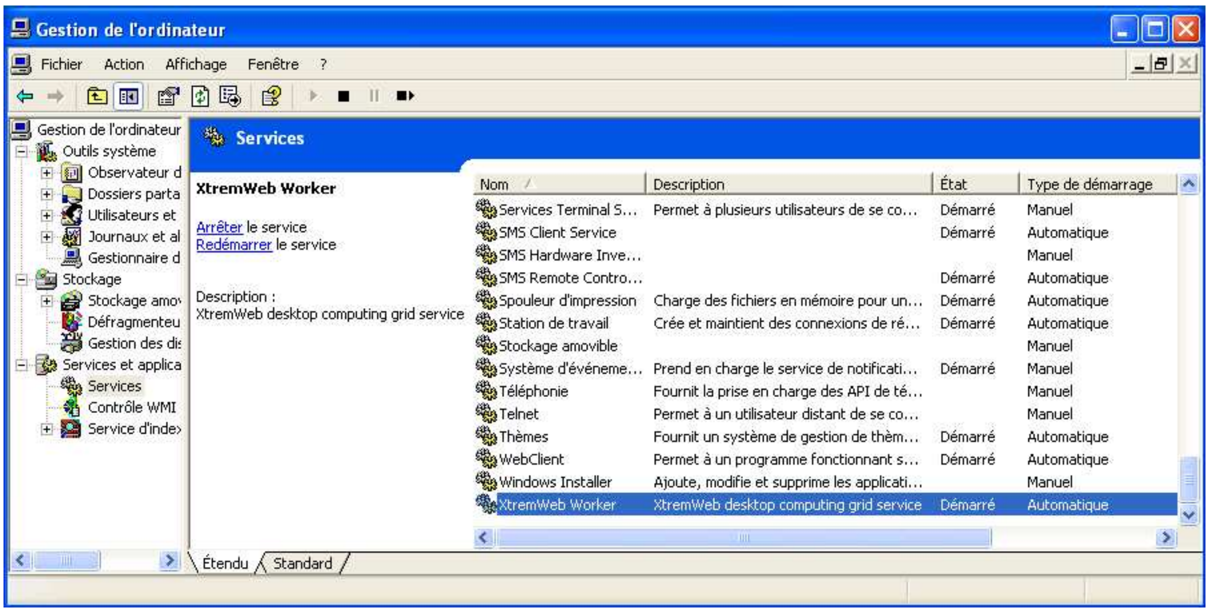

FIG. 8.10 – Contrôle du service worker en tant que service Windows.

de la page, entre la barre des boutons et la jauge d'avancement, est dédié à l'affichage des objets. Les action des boutons de la page sont applicables aux objets gérés par la page affichée. Par exemple, si la page *Jobs* est affichée, les actions de boutons s'appliquent aux jobs. De la même manière, l'affichage d'une page ne concerne que les objet de la page en cours ; la page *Jobs* n'affiche que les jobs de l'utilisateur.

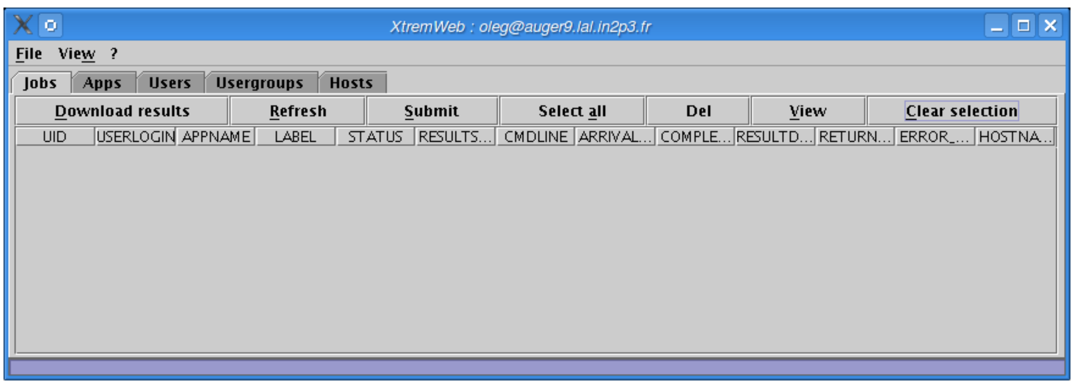

FIG. 8.11 – La fenêtre principale du client.

La fenêtre du client a trois menus : “file”, “view” et “ ?” qui contiennent les options détaillées dans le tableau 8.4.

Connexion au service de coordination. L'option `Login as...` permet de choisir le service de coordination auquel l'utilisateur souhaite se connecter. Elle ouvre une boîte de dialogue dans laquelle on trouve trois boutons qui sont communs à

Menu	Option	Action
file	Login as...	permet de se connecter sur un service de coordination
file	Close	arrête le service client
view	show works	affiche les works correspondant aux jobs
view	show tasks	affiche les tâches correspondant aux jobs
?	version	affiche la version du service
?	log level	sélectionne le niveau de journalisation

TAB. 8.4 – Les menus de la fenêtre du service client.

toute boîte de dialogue du service client : les boutons *OK* et *Cancel* pour valider ou annuler l'action, et le bouton *Help* pour afficher une aide contextuelle relative à l'action en cours.

La boîte de dialogue de connexion contient trois champs à remplir comme le montre la figure 8.12 : le champ *Server* dans lequel on doit spécifier le nom (ou l'adresse) du service de coordination ; les champs *Login* et *Password* dans lesquels l'utilisateur doit entrer ses identifiants.

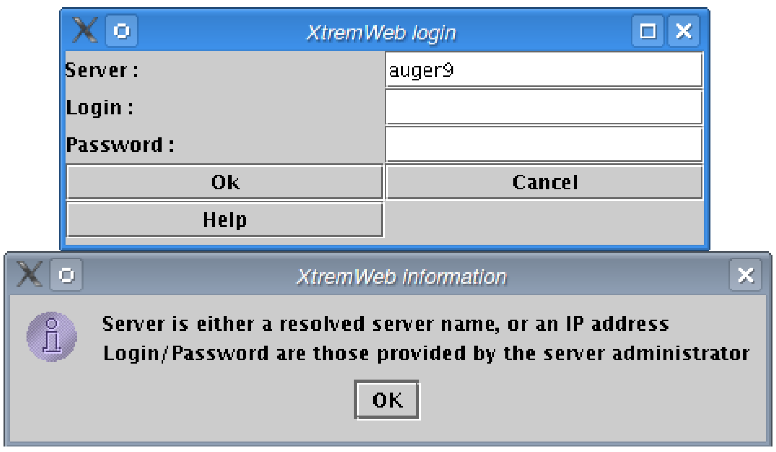

FIG. 8.12 – La fenêtre de connexion du client et son aide contextuelle.

Si le service de coordination n'est pas disponible, le message de la figure 8.13 s'affiche.

Si l'utilisateur est inconnu, le mot de passe erroné, ou encore si l'utilisateur n'est pas autorisé à se connecter, le message de la figure 8.14 s'affiche.

Quand l'utilisateur a réussi à se connecter, mais qu'il n'a pas les droits d'administration du service de coordination, le message de la figure 8.15 s'affiche. Il indique que certaines actions ne sont pas autorisées pour cet utilisateur (modifier les services applicatifs, par exemple).

Une fois connecté, l'utilisateur peut effectuer différentes actions grâce au service

FIG. 8.13 – La fenêtre d'erreur de connexion : le serveur est inconnu ou indisponible.

FIG. 8.14 – La fenêtre d'erreur de connexion : l'utilisateur est inconnu ou le mot de passe est invalide.

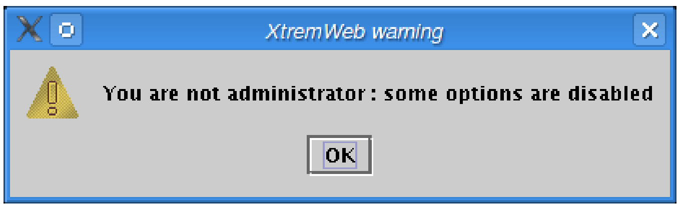

FIG. 8.15 – La fenêtre du mode de connexion : l'utilisateur n'est pas administrateur, certaines actions sont interdites.

client. Les actions possibles sont divisées en cinq familles : les actions sur les jobs, les services applicatifs, les utilisateurs, les groupes d'utilisateurs et les services worker.

On sélectionne la famille d'actions souhaitées en ouvrant la page correspondante ; par exemple, si on souhaite gérer les jobs, on ouvre la page *Jobs*. Chaque famille contient un certains nombre d'actions qui sont matérialisées par les boutons de la page affichée ; elles comprennent toutes, au minimum, les actions suivantes : *Refresh*, *Add*, *Del*, *View*, *Select all* et *Clear selection* décrites dans le tableau 8.5.

La première télécharge la liste des objets depuis le service de coordination et l'affiche. La seconde demande la création d'un nouvel objet au service de coordination, puis l'affiche. La troisième demande au service de coordination d'effacer les objets sélectionnés ; cette action est irréversible.

Les trois dernières sont des actions d'affichage, elles n'entraînent aucune action au niveau du service de coordination ; *View* affiche le détail de l'objet sélectionné ; *Select all* sélectionne tous les objets de la liste en cours ; *Clear selection* annule toute sélection.

Bouton	Action	Commentaire
Refresh	Télécharger la liste d'objets	
Add	Ajouter un objet	Intitulé "Submit" pour les jobs
Del	Effacer un ou plusieurs objets	Efface les objets sélectionnés dans la liste
View	Afficher le détail d'un objet	Affiche l'objet sélectionné dans la liste
Select all	Sélectionner tous les objets	Pas d'action avec le service de coordination
Clear selection	Efface la sélection en cours	

TAB. 8.5 – Les actions de base du service client.

Gestion des jobs. La page *Jobs* permet de gérer les jobs de l'utilisateur. Le bouton *Refresh* permet de récupérer la liste des jobs soumis au service de coordination (Cf figure 8.16) ; cette action est matérialisée par la jauge d'avancement.

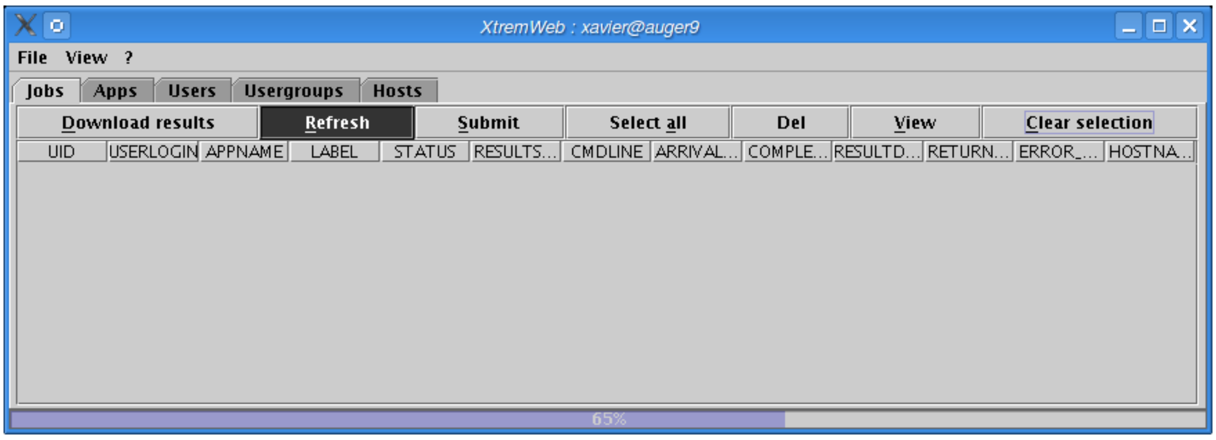

FIG. 8.16 – La fenêtre des jobs : récupération de la liste des jobs.

La liste des jobs récupérés s'affiche dans la partie centrale de la page des jobs (figure 8.17).

Les listes récupérées peuvent être triées en cliquant sur les entêtes des colonnes. Le choix du tri est cyclique : au premier click, le tri est ascendant ; au deuxième, il est descendant ; un troisième click annule tout tri. Le tri choisi est représenté par une petite icône sur l'entête de la colonne choisie pour le tri ; si la liste n'est pas triée, il n'y a pas d'icône. La figure 8.18 présente les jobs triés par *label*.

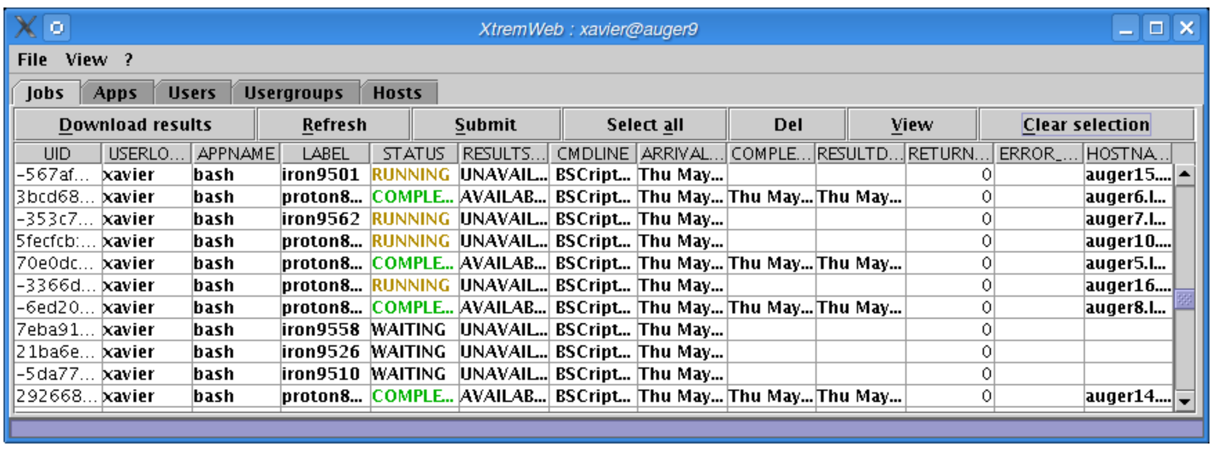

FIG. 8.17 – La fenêtre des jobs : la liste des jobs à été récupérée.

Le bouton *Download results* permet de récupérer les résultats des jobs sélectionnés dans la liste. On peut sélectionner un ou plusieurs jobs. Une boîte de dialogue

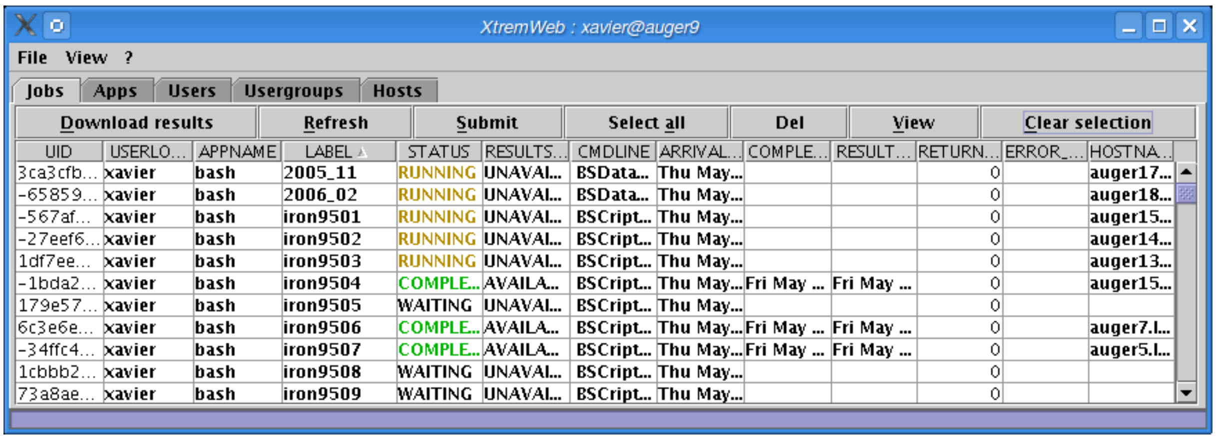

FIG. 8.18 – La fenêtre des jobs : les jobs sont triés par label.

s'affiche alors, dans laquelle il faut sélectionner le répertoire dans lequel seront stockés les résultats des jobs sélectionnés (figure 8.19). Les résultats sont stockés par service applicatif et par job, avec un répertoire unique par service applicatif dans lequel se trouvent un répertoire par job.

Une fois les résultats récupérés, une fenêtre de confirmation s'affiche : elle rappelle le nombre de résultats sauvegardés et le répertoire de sauvegarde (figure 8.20).

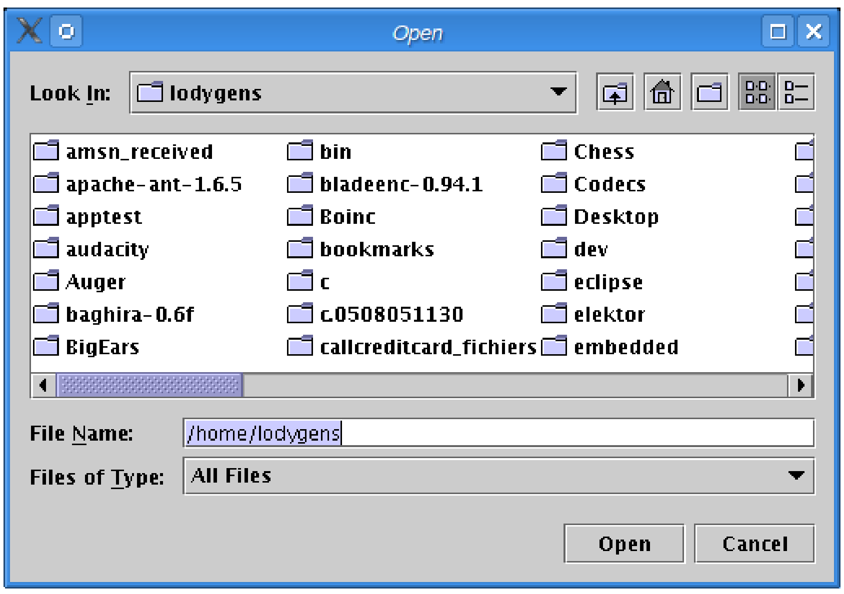

FIG. 8.19 – Récupération des résultats d'un job : la fenêtre de sélection du répertoire de stockage.

Le bouton *Del* permet d'effacer les jobs sélectionnés dans la liste. On peut sélectionner un ou plusieurs jobs. Cette action est définitive : il n'y aura plus aucun

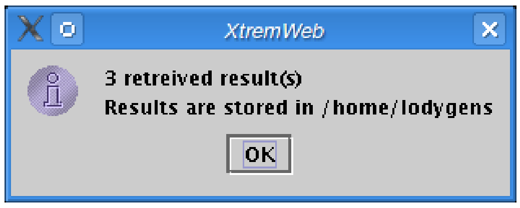

FIG. 8.20 – Récupération des résultats d'un job : la fenêtre de confirmation du stockage.

moyen de travailler ni de récupérer les éventuels résultats des jobs ainsi effacés. C'est pourquoi une fenêtre de confirmation s'affiche (figure 8.21).

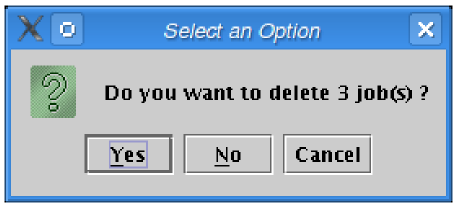

FIG. 8.21 – Une fenêtre de message pour confirmer une action «dangereuse».

Le bouton *Submit* permet de soumettre un nouveau job. Dans la fenêtre de soumission, le menu `APPNAME` contient la liste des service applicatifs installés, dans laquelle il faut choisir le service souhaité pour ce nouveau job (figure 8.22).

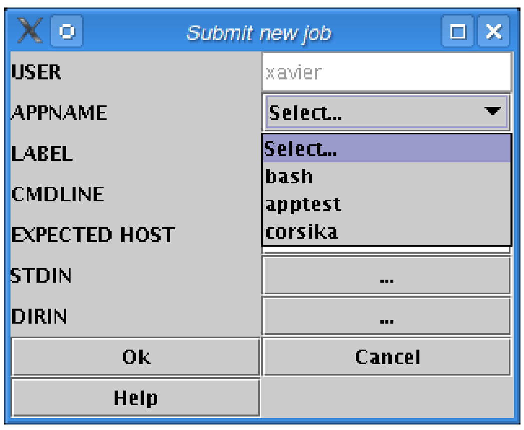

FIG. 8.22 – Soumission de job : sélection d'une application.

Un job peut avoir besoin d'un fichier d'entrée, ainsi qu'un fichier d'environnement.

Le fichier d'entrée est fourni sur l'entrée standard du job ; on le choisit avec le bouton *STDIN*.

Le fichier d'environnement est un fichier archive *ZIP* qui sera installé localement sur le service worker qui exécutera le job. Il faut sélectionner un fichier ou un répertoire. Si le fichier n'est pas une archive *ZIP*, ou si un répertoire a été sélectionné, une archive sera automatiquement créée. Pour sélectionner un fichier d'entrée, il faut cliquer sur le bouton *STDIN*. Pour sélectionner un fichier d'environnement, il faut cliquer sur le bouton *DIRIN*. Les boutons n'affichent rien si aucun fichier n'est sélectionné ; en cliquant, une fenêtre de sélection s'ouvre. La figure 8.23 montre la boîte de dialogue de sélection d'un fichier d'environnement. Une fois le fichier sélectionné, son nom s'affiche dans le bouton de sélection (figure 8.24). Pour annuler la sélection d'un fichier, il faut cliquer de nouveau sur le bouton et cliquer sur *cancel* dans la boîte de dialogue de sélection de fichier.

Des options de commande (*CMDLINE*) et un label peuvent aussi être associés au job (figure 8.25).

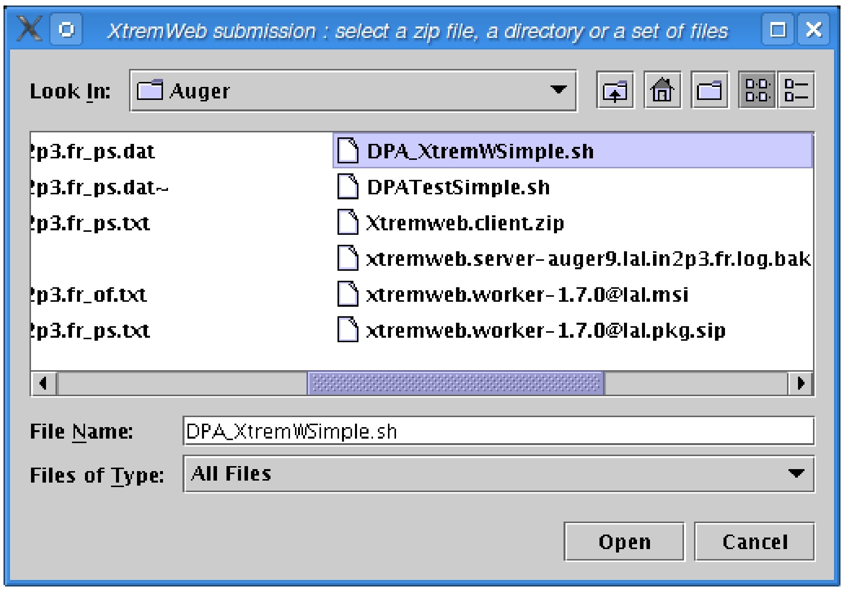

FIG. 8.23 – Soumission de job : sélection d'un fichier d'environnement.

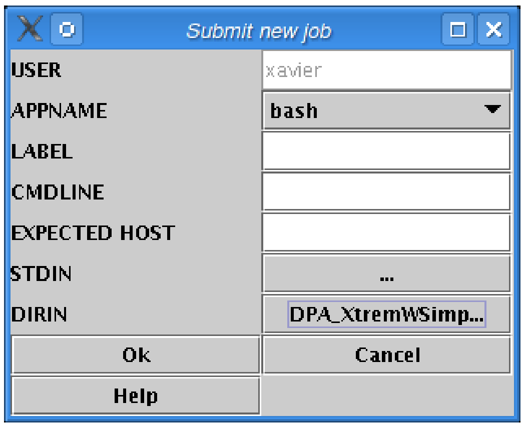

FIG. 8.24 – Soumission de job : un fichier d'environnement a été sélectionné.

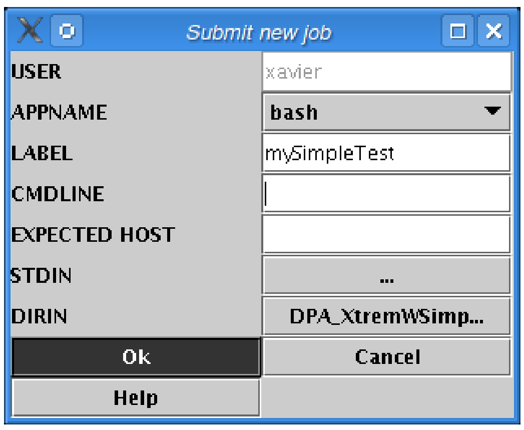

FIG. 8.25 – Soumission de job : définition d'un label et soumission.

Gestion des services applicatifs. La page d'affichage *Apps* gère les services applicatifs de la plate-forme.

Les actions *Add* et *Del* sont désactivées si l'utilisateur n'a pas les droits nécessaires. L'action*Refresh* permet d'obtenir la liste des services applicatifs installés.

Pour ajouter un service applicatif, il faut être connecté avec les droits nécessaires. En cliquant sur le bouton *Add*, la boîte de dialogue de la figure 8.26 s'ouvre. Il faut indiquer le nom du service a crée. Ce nom doit être unique ; l'utilisation d'un nom de service applicatif déjà installé modifie celui-ci. Ce point est abordé un peu plus bas.

Il faut sélectionner le type d'OS et de CPU souhaités. Enfin, le fichier binaire du service doit être sélectionné grâce au bouton *Select binary*.

Cliquer sur le bouton *OK* pour créer le nouveau service.

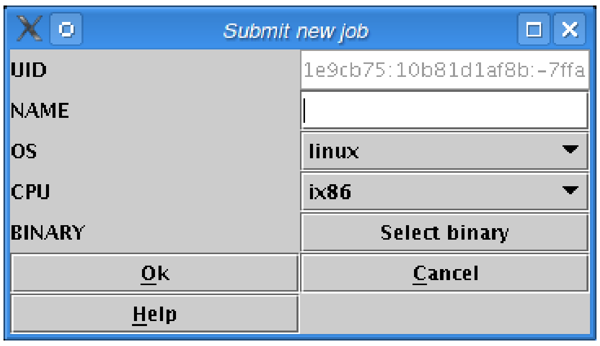

FIG. 8.26 – Création d'un service applicatif.

On peut souhaiter modifier un service applicatif déjà installé ; par exemple, si on souhaite que le service soit déployé sur différents types d'OS, il faut fournir un fichier binaire par OS. Dans la page *APPS*, cliquer sur le bouton *Add*. Dans la boîte de dialogue8.26, indiquer un nom du service à modifier ; sélectionnez le type d'OS et de CPU compatibles avec le fichier binaire fourni grâce au bouton *Select binary*. Cliquer sur *OK* pour modifier le service.

Gestion des utilisateurs. La page d'affichage *Users* gère les utilisateurs de la plate-forme.

Les actions *Add* et *Del* sont désactivées si l'utilisateur n'a pas les droits nécessaires. L'action*Refresh* permet d'obtenir les informations de l'utilisateur connecté. Cette action retrouve la liste de tous les utilisateurs référencés si elle est demandée avec les droits suffisants.

Pour ajouter un utilisateur, il faut être connecté avec les droits nécessaires. La boîte de dialogue de la figure 8.27 apparaît en cliquant sur le boutons *Add*. Il faut remplir les champs et cliquer sur le bouton *OK*.

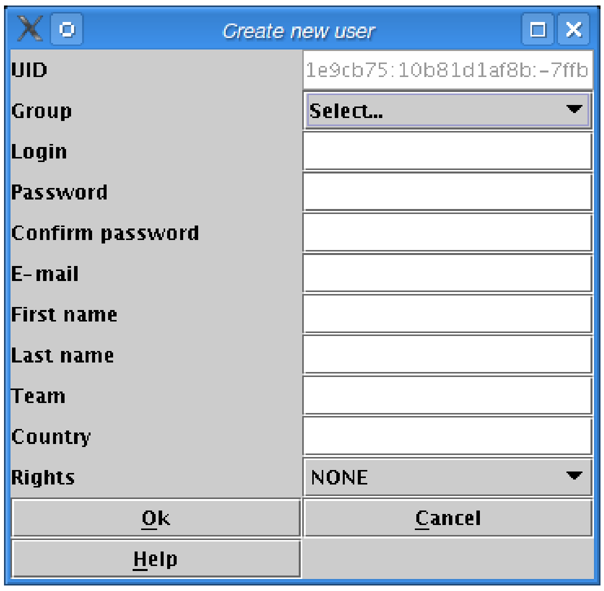

FIG. 8.27 – Création d'un nouvel utilisateur.

Bibliographie

[1] The jabber web site. www.jabber.org.

[2] The kazaa web site. www.kazaa.com.

[3] Web Services. www.webservices.org.

[4] The physiology of the grid : An open grid services architecture for distributed systems integration. In *Open Grid Service Infrastructure WG, Global Grid Forum*, 2002.

[5] D. Abramson, J. Giddy, and L. Kotler. High performance parametric modeling with nimrod/g : killer application for the global grid ? In *International Parallel and Distributed Processing Symposium (IPDPS)*, pages 520–528, 2000.

[6] D. Abramson, R. Sosic, J. Giddy, and B. Hall. Nimrod : A Tool for Performing Parameterised Simulations using Distributed Workstations. In *Proceedings of 4th IEEE Symp. High Performance Distributed Computing, Virginia, August 1995.*, Virginia, August 1995.

[7] A. Acharya and M. Raje. Mapbox : using parameterized behavior classes to confine applications. In *Technical report TRCS99-25,University of California, Santa Barbara*, 1999.

[8] D. Adams, D. Barberis, C. Bee, R. Hawkings, S. Jarp, R. Jones1, D. Malon, L. Poggioli, G. Poulard, D. Quarrie, and T. Wenaus. The atlas computing model. Technical report, CERN, January 2005. CERN-LHCC-2004-037/G-085.

[9] K. Aida, U. Nagashima, H. Nakada, S. Matsuoka, and A. Takefusa. Performance evaluation model for job scheduling in a global computing system. In *7th IEEE Int. Symp on High Performance Distributed Computing*, pages 352–353, 1998.

[10] A. Alexandrov, P. Kmiec, and K. Schauser. Consh : a confinied execution environment for internet computations. In *Proceedings of the Usenix annual technical conference, http ://www.usenix.org/events/usenix99/*, 1999.

[11] A. D. Alexandrov, M. Ibel, K. E. Schauser, and C. J. Scheiman. Superweb : Towards a global web-based parallel computing infrastructure. In *Proceedings of the 11 th IEEE International Parallel Processing Symposium (IPPS)*, April 1997.

[12] D. Anderson, S. Bowyer, J. Cobb, D. Gedye, W. T. Sullivan, and D. Werthimer. A New Major SETI Project Based on Project Serendip Data and 100,000 Personal Computers. In *Astronomical and Biochemical Origins and the Search for Life in the Universe, Proc. of the*

Fifth Intl. Conf. on Bioastronomy, 1997.

[13] David Anderson. BOINC : A System for Public-Resource Computing and Storage. In *proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, Pittsburgh, USA, November 2004.

[14] Nazareno Andrade, Walfredo Cirne, Francisco Brasileiro, and Paulo Roisenberg. OurGrid : An Approach to Easily Assemble Grids with Equitable Resource Sharing. In *Proceedings of the 9th Workshop on Job Scheduling Strategies for Parallel Processing*, June 2003.

[15] R. Baldoni and C. Marchetti. Three-tier replication for ft-corba infrastructures. *Software Practice & Experience*, 33 :767–797, 2003.

[16] Amnon Barak, Shai Guday, and Richard G. Wheeler. The mosix distributed operating system : load balancing for unix. volume 672, page 221, New York, NY, USA, 1993. Springer-Verlag Inc.

[17] A. Baratloo, M. Karaul, Z. Kedem, and P. Wyckoff. Charlotte : Metacomputing on the web. In *Proceedings of the 9th Conference on Parallel and Distributed Computing Systems*, 1996.

[18] Adam L. Beberg and et al. Project monarch. http ://www.distributed.net/des/.

[19] Dennis A. Benson, Ilene Karsch-Mizrachi, David J. Lipman, James Ostell, Barbara A. Rapp, and David L. Wheeler. GenBank. *Nucl. Acids Res.*, 30(1) :17–20, 2002.

[20] George Bosilca, Aurélien Bouteiller, Franck Cappello, Samir Djilali, Gilles Fédak, Cécile Germain, Thomas Hérault, Pierre Lemarinier, Oleg Lodygensky, Frédéric Magniette, Vincent Néri, and Anton Selikhov. Mpich-v : Toward a scalable fault tolerant mpi for volatile nodes. In *SC02*, Baltimore USA, Novembre 2002.

[21] T. Brecht, H. Sandhu, M. Shan, and J. Talbot. Paraweb : Towards world-wide supercomputing. In *Proceedings of the Seventh ACM SIGOPS European Workshop on System Support for Worldwide Applications*, 1996.

[22] J.M. Bull, L.A. Smith, L. Pottage, and R. Freeman. Benchmarking java against c and fortran for scientific applications. In *ISCOPE Conference, LNCS volumes 1343, Springer*, pages 97–105, 2001.

[23] Franck Cappello, Samir Djilali, Gilles Fedak, Frédéric Magniette Thomas Hérault, Vincent Néri, and Oleg Lodygensky. Computing on Large Scale Distributed Systems : XtremWeb Architecture, Programming Models, Security, Tests and Convergence with Grid. *FGCS Future Generation Computer Science*, 2004.

[24] Henri Casanova and Jack Dongarra. NetSolve : A Network-Enabled Server for Solving Computational Science Problems. In *The International Journal of Supercomputer Applications and High Performance Computing*, volume 11, Number 3, pages 212–223. Sage Publications, 1997.

[25] Tushar Deepak Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems.

Journal of the ACM, 43(2) :225–267, 1996.

[26] Andrew Chien, Brad Calder, Stephen Elbert, and Karan Bhatia. Entropia : Architecture and Performance of an Enterprise Desktop Grid System. *Journal of Parallel Distributed Computing*, 63(5) :597–610, 2003.

[27] Bernd O. Christiansen, Peter Cappello, Mihai F. Ionescu, Michael O. Neary, Klaus E. Schauser, and Daniel Wu. Javelin : Internet-based parallel computing using Java. *Concurrency : Practice and Experience*, 9(11) :1139–1160, 1997.

[28] I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet : A distributed anonymous information storage and retrieval system, 2000. In Proc. of the ICSI Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, 2000. International Computer Science Institute.

[29] LCG Project. The LHC computing grid project LCG.

[30] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid information services for distributed resource sharing. In *High-Performance Distributed Computing (HPDC-10), IEEE Press*, 2001.

[31] Andrei Oliveira da Silva and Osmar Norberto de Souza. A framework for result handling in bioinformatics : an application to computer assisted drug design. In *SAC '05 : Proceedings of the 2005 ACM symposium on Applied computing*, pages 128–132, New York, NY, USA, 2005. ACM Press.

[32] D.Barkai. Peer to peer computing. In *Intel Press*, 2001.

[33] Distributed.net. Project rc5. http ://www.distributed.net/rc5/, 1997.

[34] S. Djilali. P2p-rpc : Programming scientific applications on peer to peer systems with remote procedure call. In IEEE Press, editor, *Proceedings of the 3rd International Symposium on Cluster Computing and the Grid.* Tokyo Japan, November 2003.

[35] J. Dongarra, H. Meuer, and E. Strohmaier. Top500 supercomputer sites. http ://www.top500.org/.

[36] J. Douceur. The sybill attack. *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS02)*, 2002.

[37] P. Eerola et al. The NorduGrid Architecture and Tools. In *Proceedings of Computing for High Energy Physics 2003*, 2003.

[38] Ian T. Foster et all. Paul Sheldon. Grid2. In *Grid2*, pages 236–245, 2004.

[39] Ian T. Foster et all. Paul Sheldon. The grid2003 production grid : Principles and practice. In *13th International Symposium on High-Performance Distributed Computing (HPDC-13 2004), 4-6 June 2004, Honolulu, Hawaii, USA*, pages 236–245, 2004.

[40] Shawn Fanning. Napster : a p2p file sharing. http ://www.napster.com, 1999.

[41] Gilles Fedak. *XtremWeb : une plate-forme pour l'étude expérimentale du calcul global pair-à-pair.* PhD thesis, LRI Université Paris Sud, 2003.

[42] Dror G. Feitelson, Larry Rudolph, Uwe Schwiegelshohn, Kenneth C. Sevcik, and Parkson Wong. Theory and practice in parallel job scheduling. In Dror G. Feitelson and Larry Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, pages 1–34. Springer Verlag, 1997.

[43] C. Fetzer and S. Mishra. Transparent TCP/IP based Replication. In *Proceedings of the 29th International Symposium on FaultTolerant Computing*, Madison, Wisconsin, June 1999.

[44] M. Fischer, N. Lynch, and M. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32 :374–382, April 1985.

[45] I. Foster. Service-Oriented Science. *Science*, 308(5723) :814 – 817, may 2005.

[46] I. Foster, J. Geisler, W. Nickless, W. Smith, and S. Tuecke. Globus : A metacomputing infrastructure toolkit. In *Proc. 5th IEEE Symposium on High Performance Distributed Computing*, 1997.

[47] I. Foster and C. Kesselman. The grid : Blueprint for a future computing infrastructure. Morgan Kaufmann Publishers, 1999. 8, 1999.

[48] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid : An open grid services architecture for distributed systems integration, 2002.

[49] Ian Foster. The Anatomy of the Grid : Enabling Scalable Virtual Organizations, IJSA, 2001. In *International Jounal on Supercomputer Applications*, 2001.

[50] Ian Foster and Adriana Iamnitchi. On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing. In *2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, Berkeley, 2003.

[51] Ian Foster and Adriana Iamnitchi. On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing. *2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, 2003.

[52] Ian Foster and Carl Kesselman. The grid : Blueprint for a new computing infrastructure – isbn=1-55860-475-8. Morgan Kaufman Publisher, 1998.

[53] Ian T. Foster, Carl Kesselman, Gene Tsudik, and Steven Tuecke. A security architecture for computational grids. In *ACM Conference on Computer and Communications Security*, pages 83–92, 1998.

[54] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design patterns : Abstraction and reuse of object-oriented design. *Lecture Notes in Computer Science*, 707 :406–431, 1993.

[55] Douglas P. Ghormley, David Petrou, Steven H. Rodrigues, Amin M. Vahdat, and Thomas E. Anderson. GLUnix : A Global Layer Unix for a network of workstations. *Software Practice and Experience*, 28(9) :929–961, 1998.

[56] I. Goldberg, D. Wagner, R. Thomas, and E. Brewer. A secure environment for untrusted help application – confining the wily hacker. In *Proceedings of the 6th Usenix Security Symposium*, 1996.

[57] L. Gong, M. Muller, H. Prafullchandra, and R. Schemers. Going

beyond the sandbox : an overview of the new security architecture in the java development kit 1.2. In *Usenix Symposium on Internet Technologies ans Systems*, 1997.

[58] Jean-Pierre Goux, Sanjeev Kulkarni, Jeff Linderoth, and Michael Yoder. An Enabling Framework for Master-Worker Applications on the Computational Grid. In IEEE Conputer Society, editor, *Proc. 9th IEEE Symp. on High Performance Distributed Computing*, pages 43–50, Pittsburgh, Pennsylvania, USA, 2000.

[59] A. M. Hillas. Shower simulation : Lessons from mocca. *Nucl. Phys. Proc. Suppl.*, 52B :29–42, 1997.

[60] And The Influence. Extensive air shower simulations with corsika.

[61] John Kubiatowicz, David Bindel, Yan Chen, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westly Weimer, Christopher Wells, and Ben Zhao. Oceanstore : An architecture for global-scale persistent storage. In *Proceedings of ACM ASPLOS*. ACM, November 2000.

[62] Tuecke S. Enfert D. Foster I. Thompson M. Pearlman L. and Kesselman C. Internet x.509 public key proxy certificate profile. In *IEFT*, 2001.

[63] Lamport, Shostak, and Pease. The byzantine generals problem. In *Advances in Ultra-Dependable Distributed Systems, N. Suri, C. J. Walter, and M. M. Hugue (Eds.), IEEE Computer Society Press*. 1995.

[64] J. R. Licklider. Ire transactions on human factors in electronics. In Institute of Electrical New York and Electronics Engineers, editors, *Proceedings of the 3rd International Symposium on Cluster Computing and the Grid*, 1963.

[65] M. J. Litzkow, M. Livny, and M. W. Mutka. Condor - A Hunter of Idle Workstations. In *Proceedings of the 8th International Conference on Distributed Computing Systems (ICDCS)*, pages 104–111, Washington, DC, 1988. IEEE Computer Society.

[66] M. J. Litzkow, M. Livny, and M. W. Mutka. Condor - A Hunter of Idle Workstations. In *Proceedings of the 8th International Conference on Distributed Computing Systems (ICDCS)*, pages 104–111, Washington, DC, 1988. IEEE Computer Society.

[67] Michael Litzkow, Miron Livny, and Matt Mutka. Condor - a hunter of idle workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems, IEEE Computer Society Press*, pages 104–111, Madison, Wisconsin, 1988.

[68] Qin Lv, Sylvia Ratnasamy, and Scott Shenker. Can heterogeneity make gnutella scalable ?

[69] M. Lyvin and al. Building reliable clients and services. In *Grid : Blueprint for a New Computing Infrastructure, second edition*, pages 297–299, 2004.

[70] J. MacLaren, R. Sakellariou, J. Garibaldi, and D. Ouelhadj. Towards service level agreement based scheduling on the grid, 2004.

[71] G. Mersenne and P. Search. Gimps discovers 36th known

mersenne prime. Great Internet Mersenne Prime Search. Press Release, Sept. 1997. http ://www.mersenne.org/2976221.htm., 1997.

[72] Callman Mike and Machek Pavel. Subterfugue : a framework for observing and playing with the reality of software. http ://subterfugue.org/.

[73] A. Natrajan, M. Humphrey, and A. Grimshaw. Grids : Harnessing Geographically-Separated Resources in a Multi-Organisational Context. In *Proceedings of the 15th Annual Symposium on High Performance Computing Systems and Applications(HPCS 2001)*, Ontario, Canada, June 2001.

[74] M. Neary, S. Brydon, P. Kmiec, S. Rollins, and P. Capello. Javelin++ : Scalability issues in global computing. In *Proceedings of the ACM Java Grande 1999 Conference*, San Francisco, California, June 1999.

[75] N. Nisan, S. London, O. Regev, and N. Camiel. Globally distributed computation over the internet-the popcorn project. In *Proceedings for the 18th International Conference on Distributed Computing Systems*, 1998.

[76] Delannoy O. and Petiton S. A peer to peer computing framework : Design and performance evaluation of yml. In *Parallel and Distributed Computing, 2004. Third International Symposium on/Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks, 2004*, pages 362–369, 2004.

[77] Vijay Pande. Genome at home. http ://genomeathome.stanford.edu/. Pande Group, Chemistry Department, Stanford University.

[78] Pasta - The LHC Technology Tracking Team for Processors, Memory, Architectures, Storage and Tapes Brief. http ://lcg.web.cern.ch/LCG/PEB/PASTAIII/pasta2002Report.htm.

[79] H. Pedroso, L. M. Silva, and J. G. Silva. Web-based metacomputing with jet. In *Proceedings of the ACM 1997 PPoPP Workshop on Java for Science and Engineering Computation*. ACM, June 1997.

[80] Colin Percival. Pihex, a distributed effort to calculate pi. http ://www.cecm.sfu.ca/projects/pihex/index.html, 1999.

[81] J. S. Plank. A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems. *Software – Practice & Experience*, 27(9) :995–1012, September 1997.

[82] James S. Plank, Henri Casanova, Micah Beck, , and Jack Dongarra. Deploying fault tolerance and task migration with netsolve. *Future Generation Computer Systems*, 15 :745–755, 1999.

[83] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A Scalable Content Addressable Network. Technical Report TR-00-010, Berkeley, CA, 2000.

[84] M. Ripeanu and I. Foster. Mapping the gnutella network : Macroscopic properties of large-scale peer-to-peer systems, 2002.

[85] Lawrence G. Roberts. Beyond moore's law : Internet growth

trends. *Computer*, 33(1) :117–119, 2000.

[86] A. L. Rosenberg. Guidelines for data-parallel cycle-stealing in networks of workstations. *Journal of Parallel and Distributed Computing*, 59 :31–53, 1999.

[87] Antony Rowstron and Peter Druschel. Pastry : Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218, 2001.

[88] Alain J. Roy, Ian T. Foster, William Gropp, Nicholas T. Karonis, Volker Sander, and Brian R. Toonen. Mpich-gq : Quality-of-service for message passing programs. In *Supercomputing*, 2000.

[89] L. F. G. Sarmenta. *Volunteer Computing*. PhD thesis, Cambridge, USA, June 2001. Massachusetts Institute of Technology.

[90] L. F. G. Sarmenta, S. Hirano, and S. A. Ward. Towards bayanihan : builing an extensible framework for volonteer computing using java. In *Proc. ACM Workshop on Java for High-Performance Network Computing*, 1998.

[91] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proceedings of Multimedia Computing and Networking 2002 (MMCN '02)*, San Jose, CA, USA, (citeseer.nj.nec.com/saroiu02measurement.html), January 2002.

[92] Mitsuhisa Sato, Motonari Hirano, Yoshio Tanaka, and Satoshi Sekiguchi. OmniRPC : A Grid RPC Facility for Cluster and Global Computing in OpenMP. In Springer, editor, *Proc. of Workshop on OpenMP Applications and Tools 2001*, volume LNCS 2104, pages 130–135, West Lafayette, IN, USA, July 2001.

[93] Mitsuhisa Sato, Hidemoto Nakada, Satoshi Sekiguchi, Satoshi Matsuoka, Umpei Nagashima, and Hiromitsu Takagi. Ninf : A Network Based Information Library for Global World-Wide Computing Infrastructure. In *Proc. of High-Performance Computing ans Networking, International Conference and Exhibition, HPCN Europe*, volume LNCS 1225, pages 491–502, Vienna, Austria, April 1997. Springer.

[94] Heiko A. Schmidt. Phylogenetic trees from large datasets. 2003.

[95] Detlef Schoder and Kai Fischbach. Peer-to-peer prospects. *Commun. ACM*, 46(2) :27–29, 2003.

[96] S. J. Sciutto. Aires : Air showers extended simulation. Department of Physics of the Universidad Nacional de La Plata, Argentina, 1995. http ://www.fisica.unlp.edu.ar/auger/aires/.

[97] Keith Seymour, Hidemoto Nakada, Satoshi Matsuoka, Jack Dongarra, Craig Lee, and Henri Casanova. GridRPC : A Remote Procedure Call API for Grid Computing. In *Technical report, Univ. of Tennesse*, ICL-UT-02-06, June 2002.

[98] J.F. Shoch and J.A. Hupp. Computing practices : The 'worm' programs - early experience with a distributed computation. *Comm. ACM*, 25(3) :172–180, 1982.

[99] Clay Sirky. What is p2p. dave-

net.scripting.com/2000/11/15/claySirkyOnP2P, 2000.

[100] Clip2 Distributed Search Solution. Gnutella protocol specification v0.4. http ://gnutella.wego.com/.

[101] Olivier Soyez. Us : Prototype de stockage pair à pair. In *RENPAR 2003, la Colle sur Loup, France*, pages 214–218, October 2003.

[102] P. Stelling, I. Foster, C. Kesselman, C.Lee, and Gregor von Laszewski. A Fault Detection Service for Wide Area Distributed Computations. In *Proceedings of the 7th IEEE International Symposium on High Performance Distributed Computing*, pages 268–278, Chicago, IL, 28-31 July 1998.

[103] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord : A scalable Peer-To-Peer lookup service for internet applications. Technical Report TR-819, MIT, 2001.

[104] I. Stokes-Rees, A. Tsaregorodtsev, V. Garonne, R. Graciani, M. Sanchez, M. Frank, and J. Closier. Developing LHCb Grid Software : Experiences and Advances. *Concurrency and Computation :Practice and Experience*, 18(1–18), 2005.

[105] Sun Microsystems Inc. RPC : Remote Procedure Call Protocol specification version 2. In *Tech. Rept. DARPA-Internet RFC 1057, SUN Microsystems, Inc.*, June 1988.

[106] E. James Whitehead, Jr. and Yaron Y. Goland. WebDAV : A network protocol for remote collaborative authoring on the web. In *Proc. of the Sixth European Conf. on Computer Supported Cooperative Work (ECSCW'99), Copenhagen, Denmark, September 12-16, 1999*, pages 291–310.

[107] B. Yang and H. Garcia-Molina. Improving search in peer-to-peer networks. In *28th Intl. Conf. on Distributed Computing Systems*, 2002.

[108] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry : An Infrastructure for Fault-tolerant Wide-area Location and Routing. Technical Report UCB/CSD-01-1141, UC Berkeley, April 2001.