Gestion de données dans les grilles de calcul:

Support pour la tolérance aux fautes et la cohérence des données

Sébastien Monnet

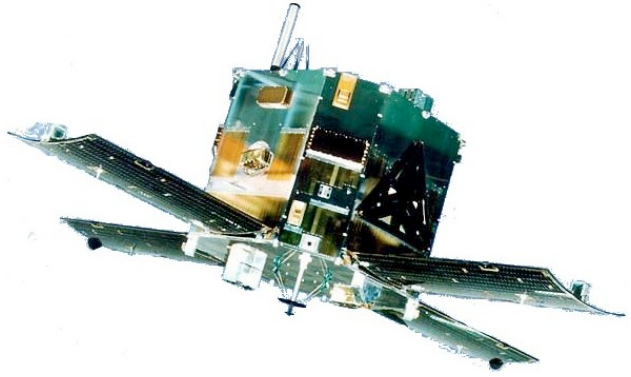
Projet PARIS/IRISA
Université de Rennes 1
France

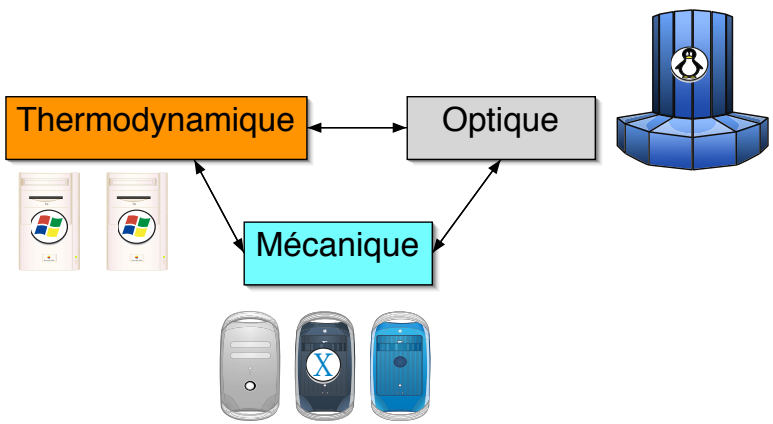
30 novembre 2006

Directeurs de thèse :
Luc Bougé et Gabriel Antoniu

De nouveaux besoins

- Simulations numériques
  - Grande précision
  - Phénomènes physiques complexes
- Importants besoins
  - Efficacité d'interaction
  - Exploitation des résultats
  - Puissance de calcul
  - Espace de stockage

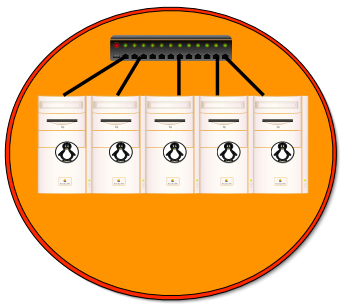


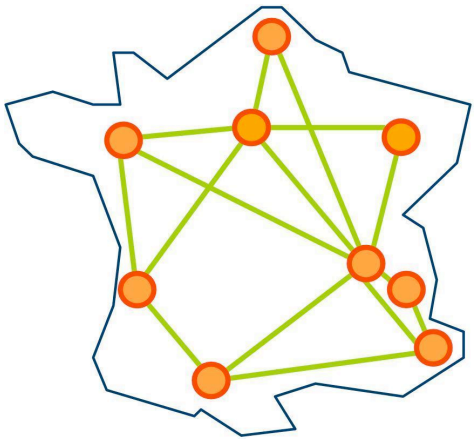


Une solution : les grilles de calcul

- Une puissance de calcul virtuellement infinie
  - Mutualisation de ressources (calcul, transport, stockage)
  - Ressources distribuées dans différentes organisations
  - Organisation virtuelle

- Un cas particulier : les fédérations de grappes
  - À grande échelle $10^3 - 10^4$ nœuds
  - Hétérogènes
  - Dynamiques
  - Hiérarchiques

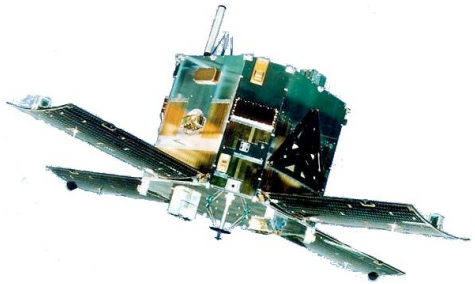


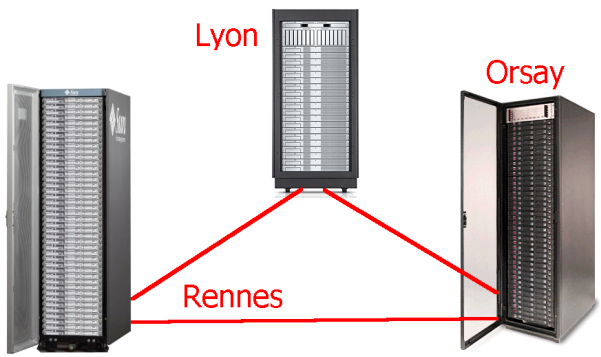


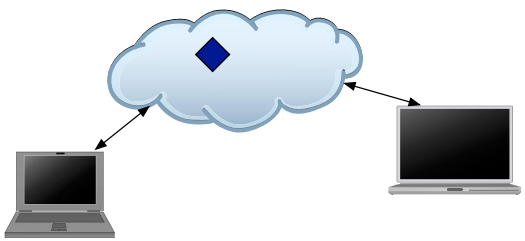
Grid'5000

Problème : la gestion des données

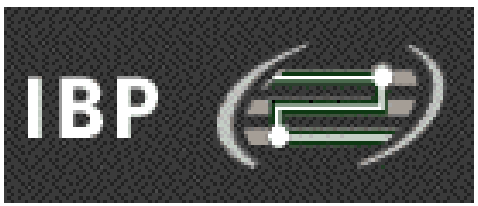
- Besoins
  - Applications distribuées sur plusieurs nœuds
  - Données partagées
- Propriétés attendues
  - Transparence
    - Localisation, transfert, etc.
  - Persistance
    - Dépendance de données entre plusieurs calculs







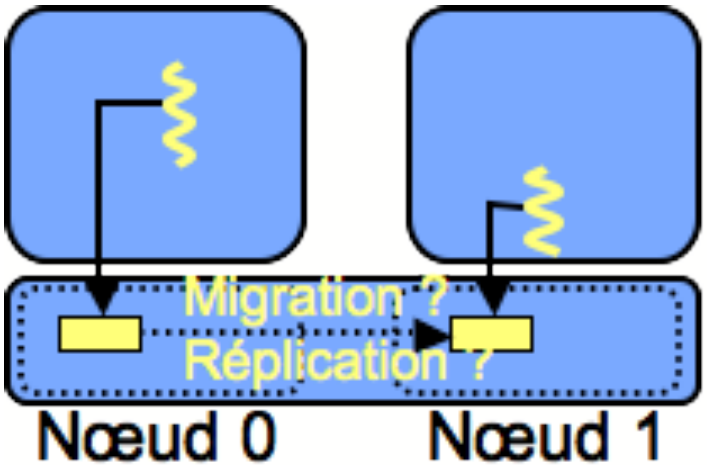
Systèmes existants



- Limites : gestion explicite des données par les applications
  - Localisation
  - Cohérence des copies
- Défi : gestion transparente du partage des données

Les systèmes à mémoire virtuellement partagée (MVP)

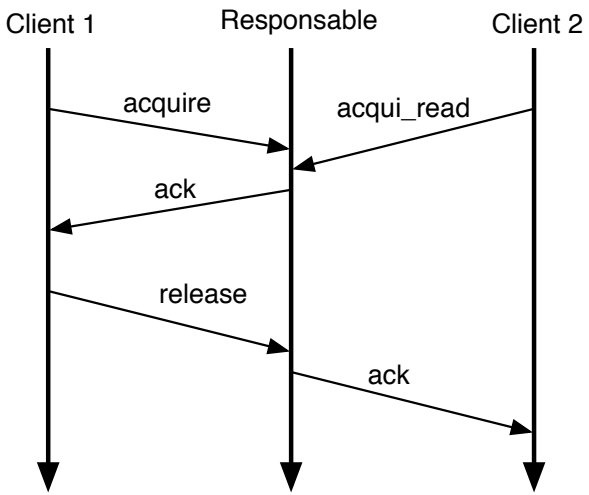
- Propriétés
  - Accès aux données uniforme via des identifiants globaux
  - Localisation et transfert transparents
  - Modèles et protocoles de cohérence



Exemple de protocole de cohérence

- Cohérence à l'entrée
  - Modèle relâché
  - Association explicite de verrous aux données
  - Lectures parallèles
    - acquire
    - acquire_read
- Utilisation d'une copie de référence

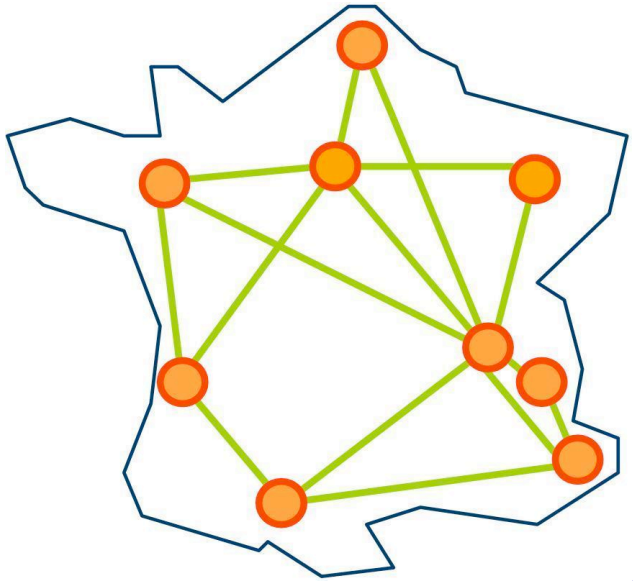
acquire(data)
write(data)
release(data)



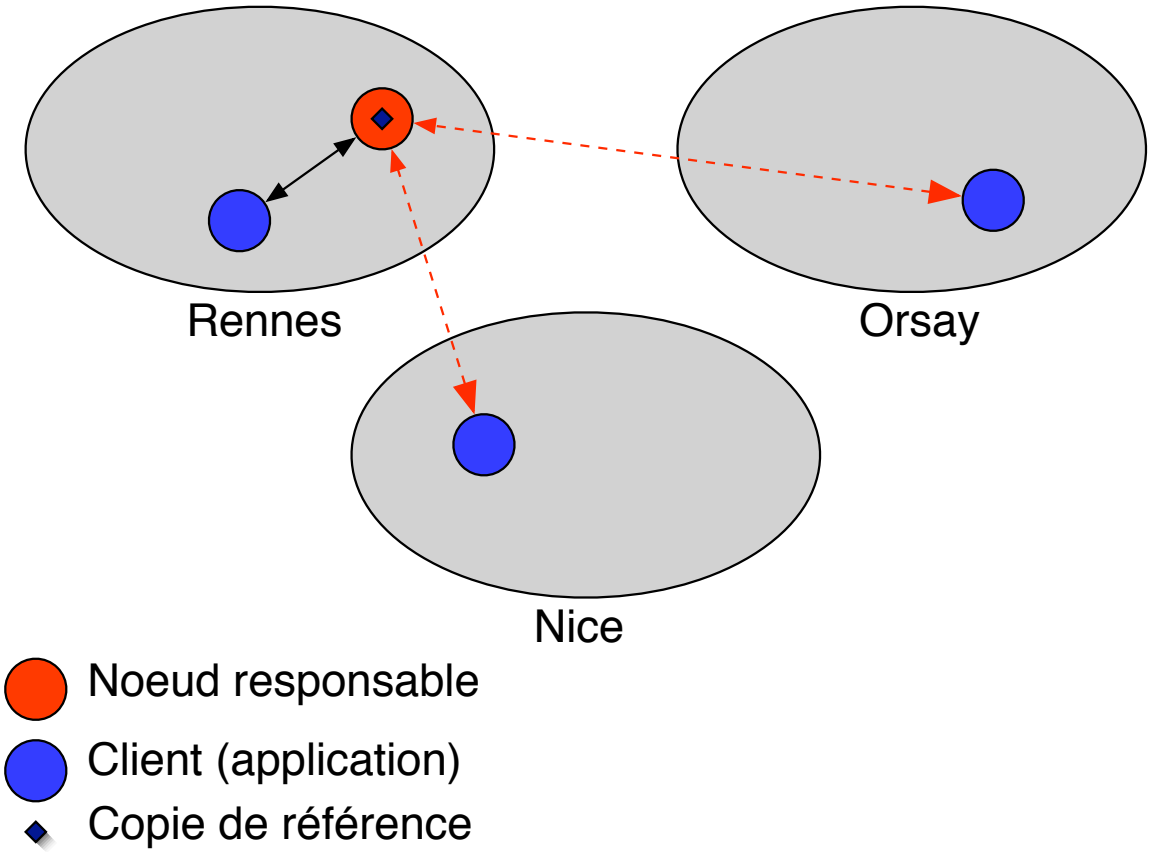
acquire_read(data)
read(data)

Défis dans les grilles

- Protocoles de cohérence conçus pour des architectures
  - Statiques
  - De petite taille
- Défi : intégrer de nouvelles hypothèses !
  - *Grande échelle*
  - *Nature dynamique*

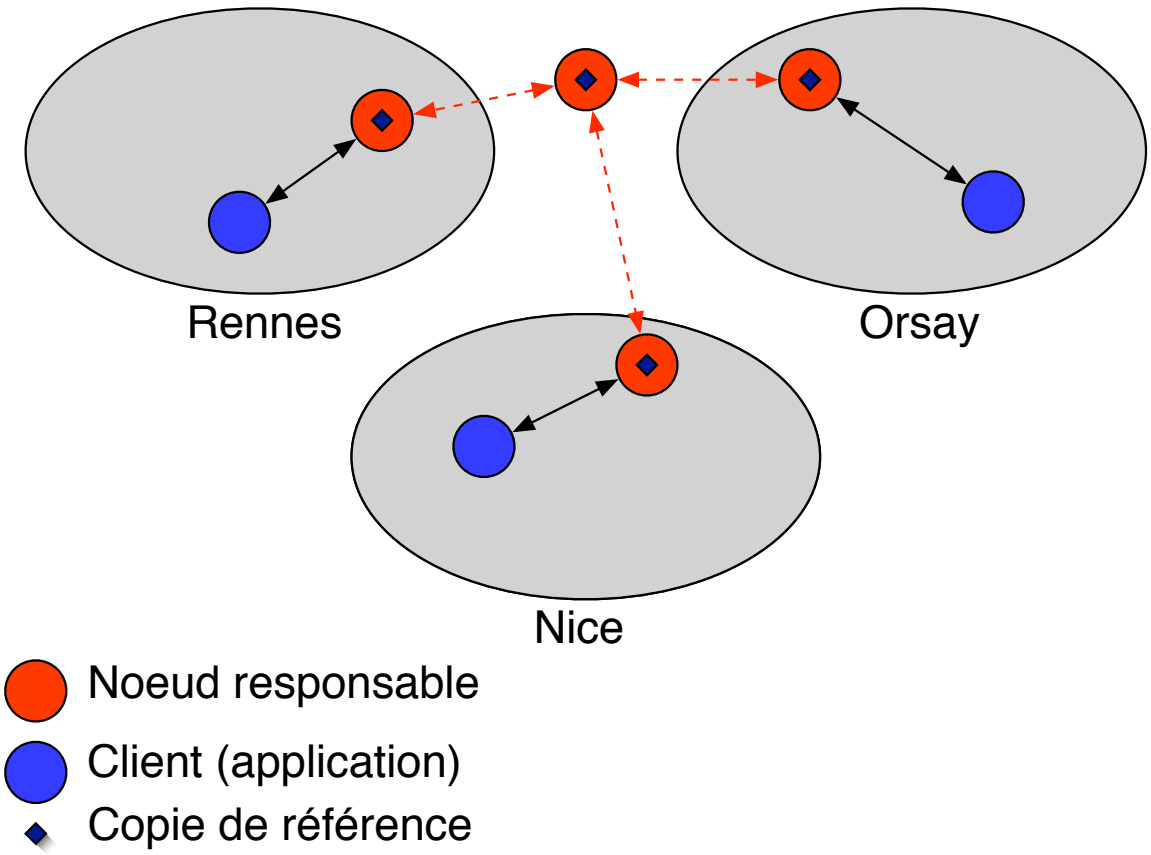


Premier défi : grande échelle



- Différence de latence d'un facteur 1000

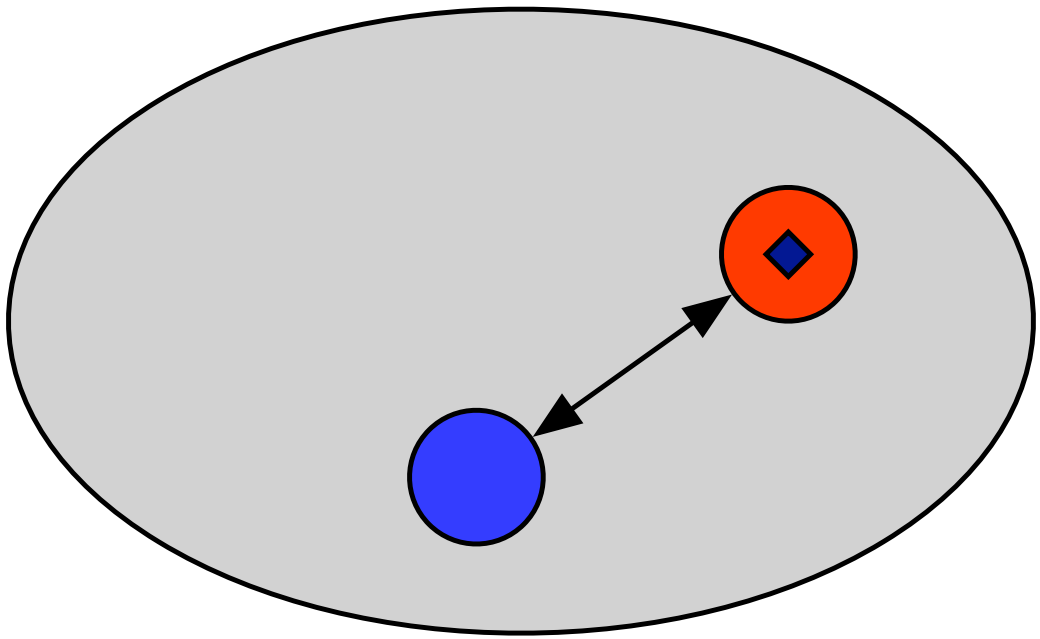
Solution : une approche hiérarchique



- Basé sur CLRC [LIP6] et H2BRC [IRISA]

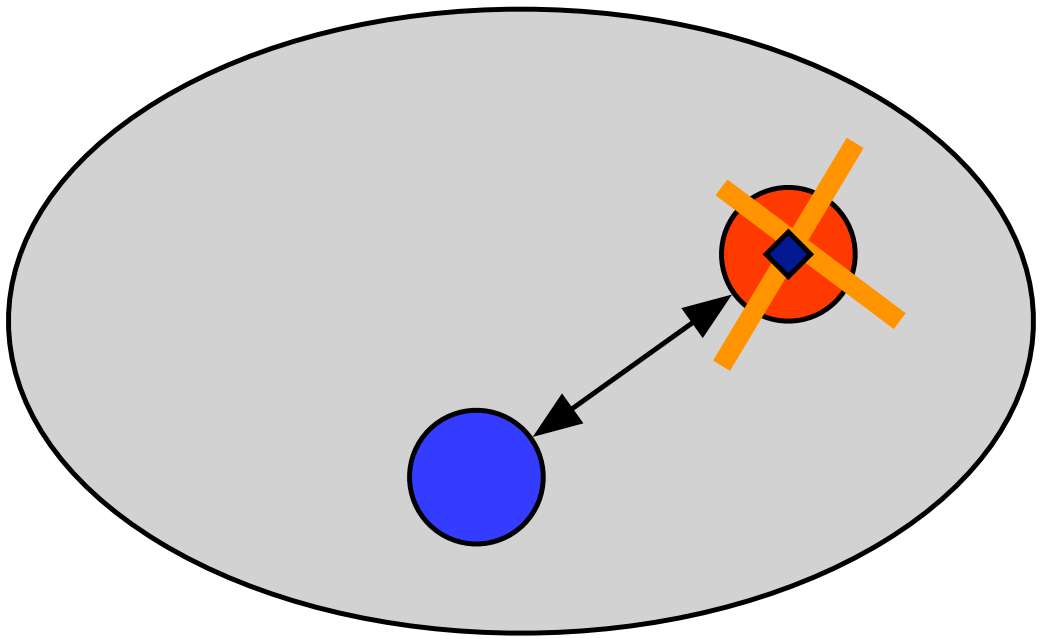
Deuxième défi : nature dynamique

- Hypothèses : fautes (crashs) et déconnexions
- Problème : disponibilité des données
- Solution : réplication



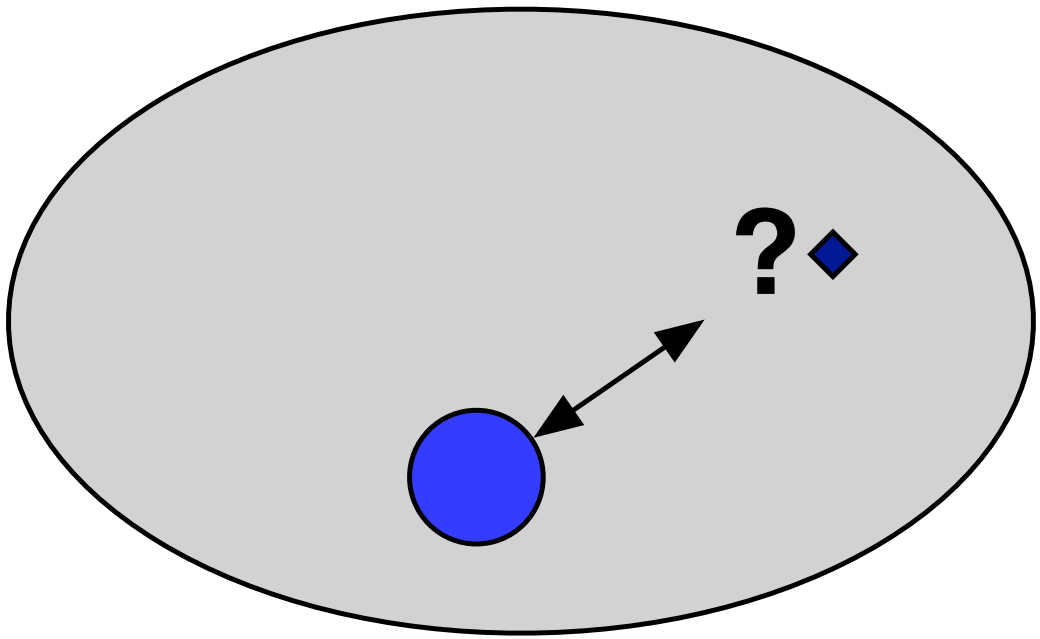
Deuxième défi : nature dynamique

- Hypothèses : fautes (crashs) et déconnexions
- Problème : disponibilité des données
- Solution : réplication

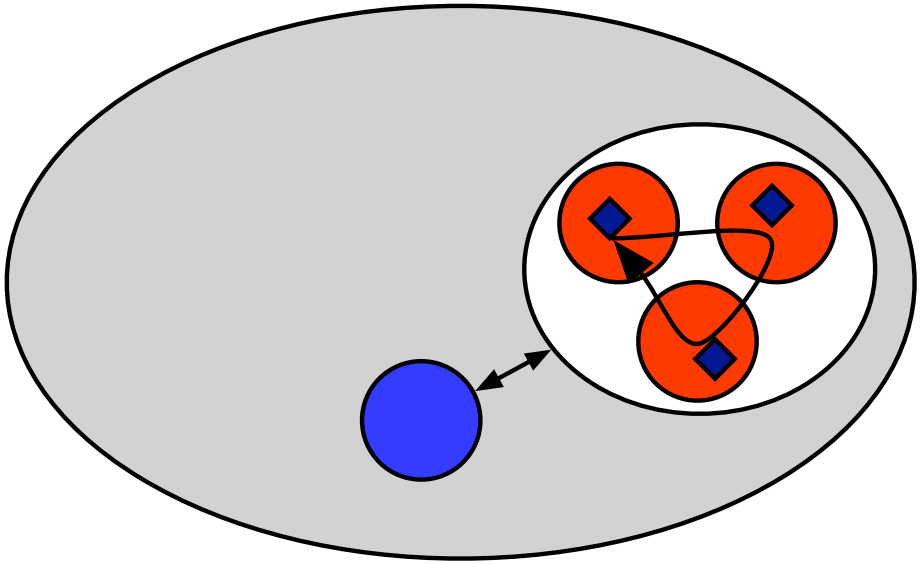


Deuxième défi : nature dynamique

- Hypothèses : fautes (crashs) et déconnexions
- Problème : disponibilité des données
- Solution : réplication

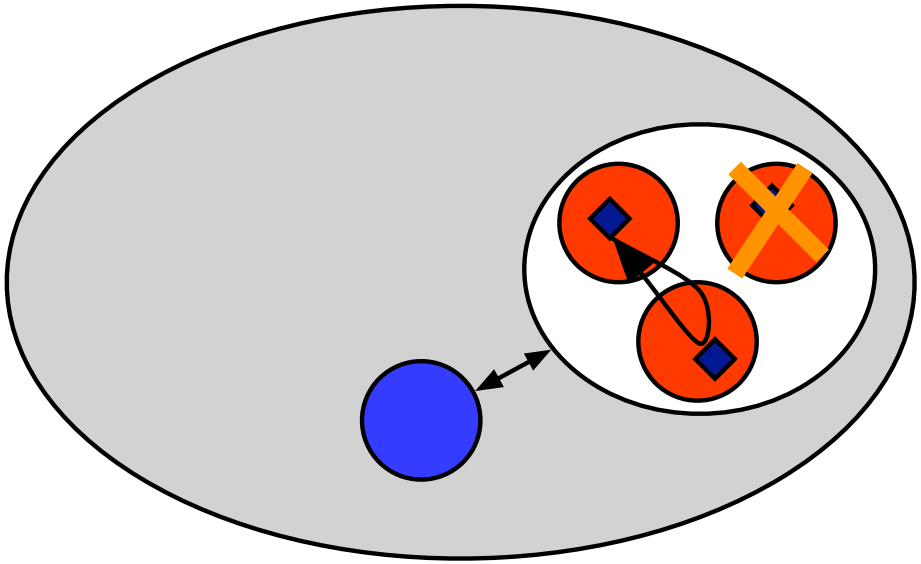


Solution : réplication des entités critiques



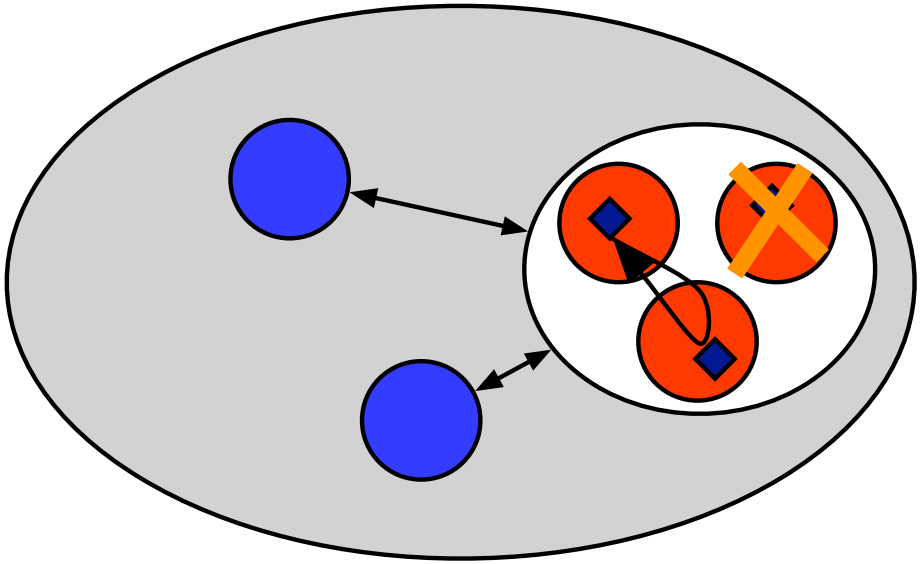
- Solution utilisée pour fiabiliser de nombreux systèmes
  - DARX [LIP6], HORUS [Cornell], LegionFS [U.Va], etc.
- Besoins
  - Notion de groupe : composition de groupes
  - Communications de groupes : diffusion atomique

Solution : réplication des entités critiques



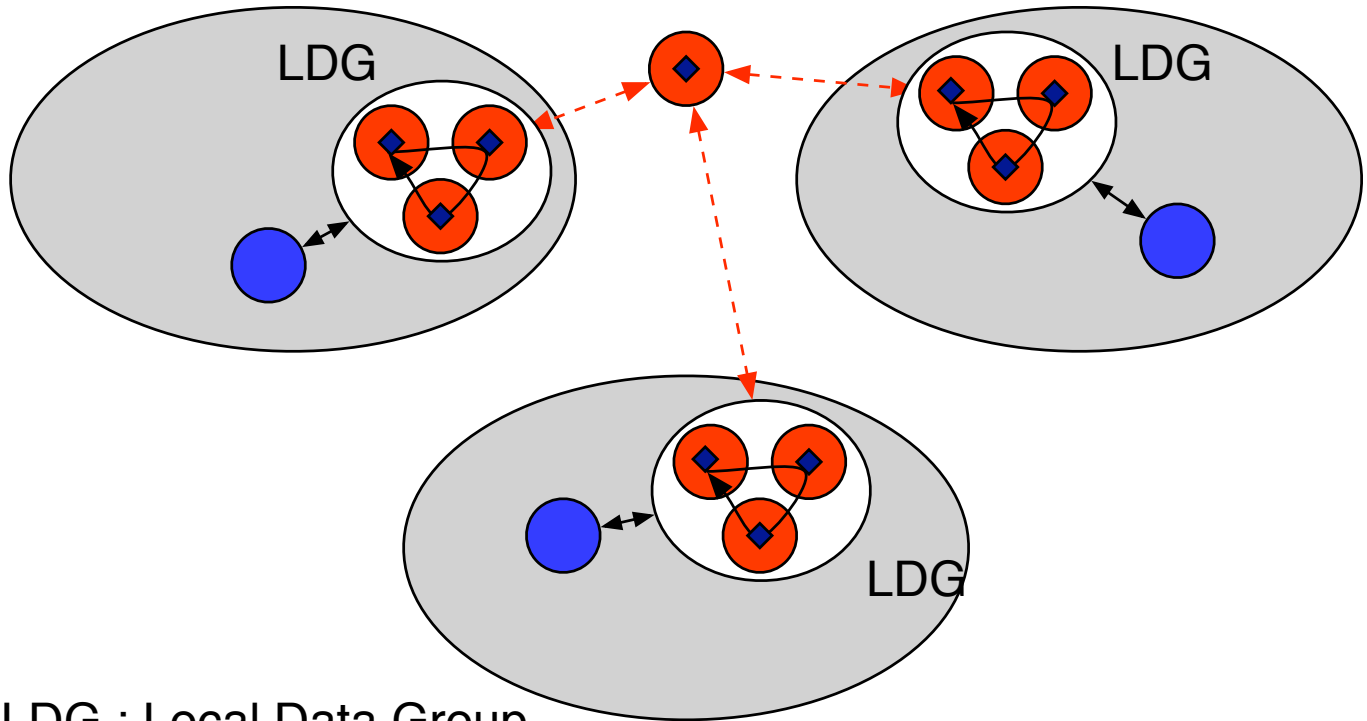
- Solution utilisée pour fiabiliser de nombreux systèmes
  - DARX [LIP6], HORUS [Cornell], LegionFS [U.Va], etc.
- Besoins
  - Notion de groupe : composition de groupes
  - Communications de groupes : diffusion atomique

Solution : réplication des entités critiques



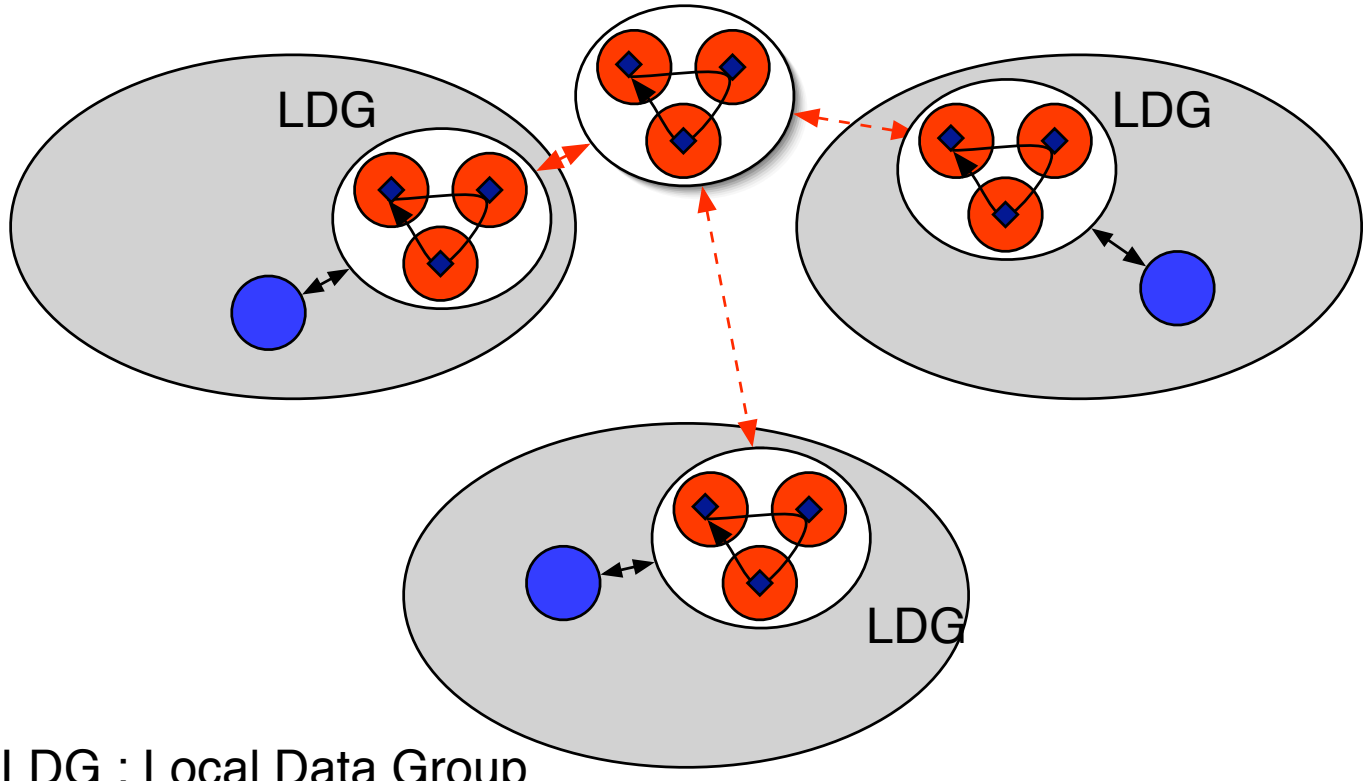
- Solution utilisée pour fiabiliser de nombreux systèmes
  - DARX [LIP6], HORUS [Cornell], LegionFS [U.Va], etc.
- Besoins
  - Notion de groupe : composition de groupes
  - Communications de groupes : diffusion atomique

Notre approche : réplication hiérarchique



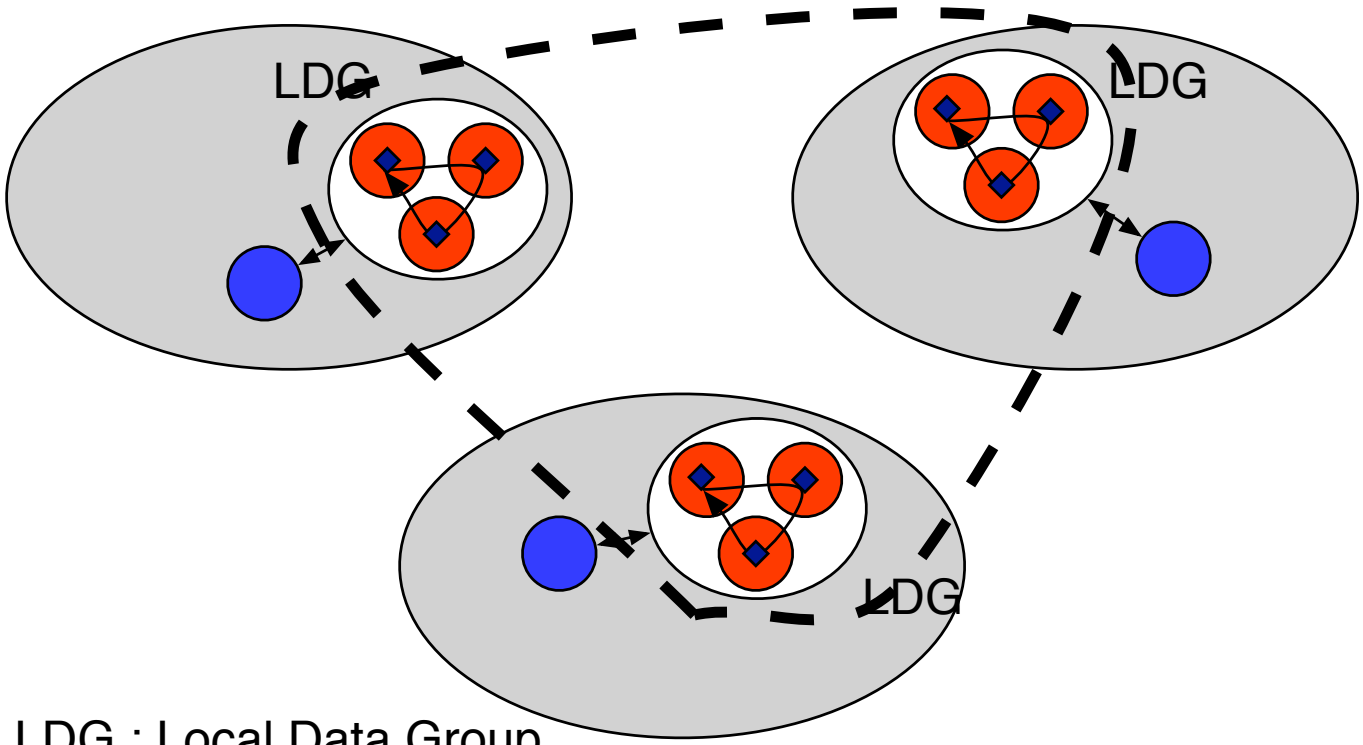
LDG : Local Data Group

Notre approche : réplication hiérarchique



LDG : Local Data Group

Notre approche : réplication hiérarchique



LDG : Local Data Group
GDG : Global Data Group

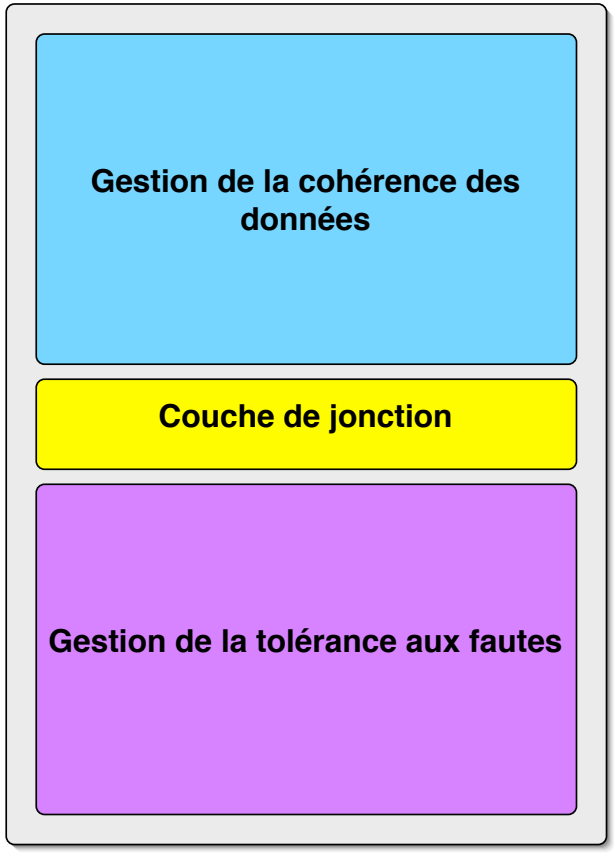
Double intérêt de la réplication

- Localité des accès aux données
  - Copies créées par le protocole de cohérence pour améliorer les performances des accès
- Disponibilité des données
  - Copies créées par les mécanismes de tolérance aux fautes

- Deux types de copies
  - *Gestion intégrée*
    - Faible nombre de copies
    - Grande complexité
  - *Gestion disjointe*
    - Grand nombre de copies
    - Tolérance aux fautes et cohérence gérées séparément

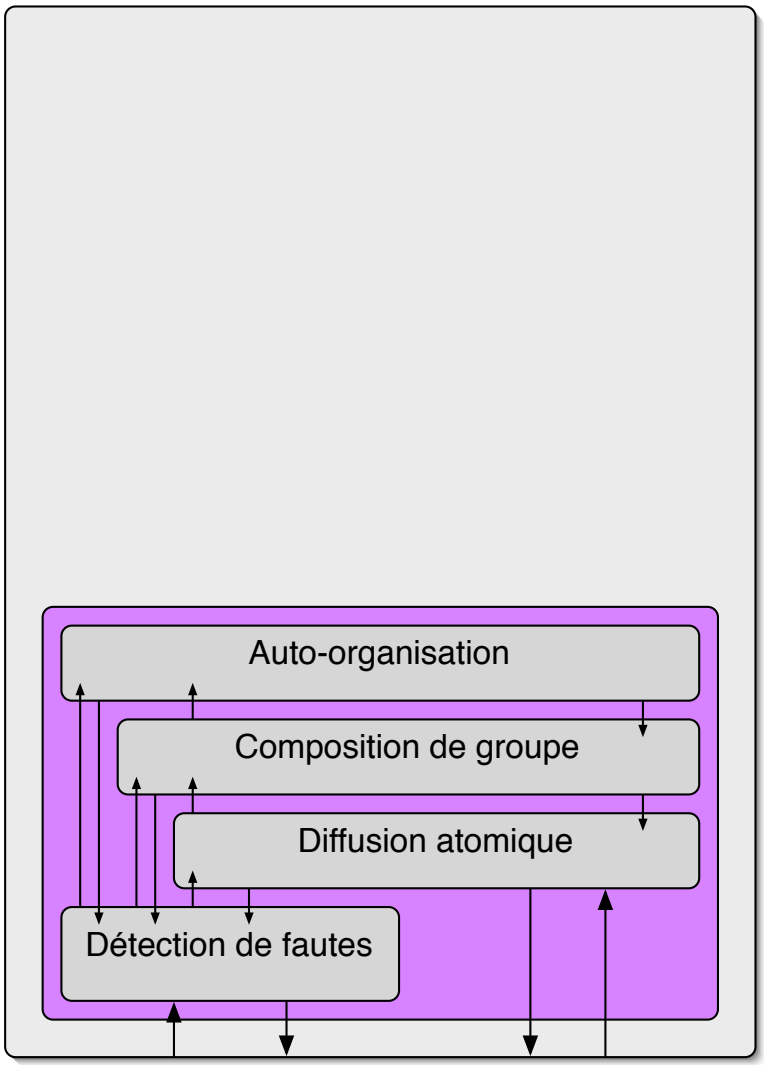
Une approche intermédiaire : gestion conjointe

- Gestion découplée
  - Cohérence des données
  - Tolérance aux fautes
- Avantages :
  - Conception séparée
  - Multiples mises en œuvre possibles
  - Architecture multi-protocole
- Gestion fine des interactions
  - Couche de jonction



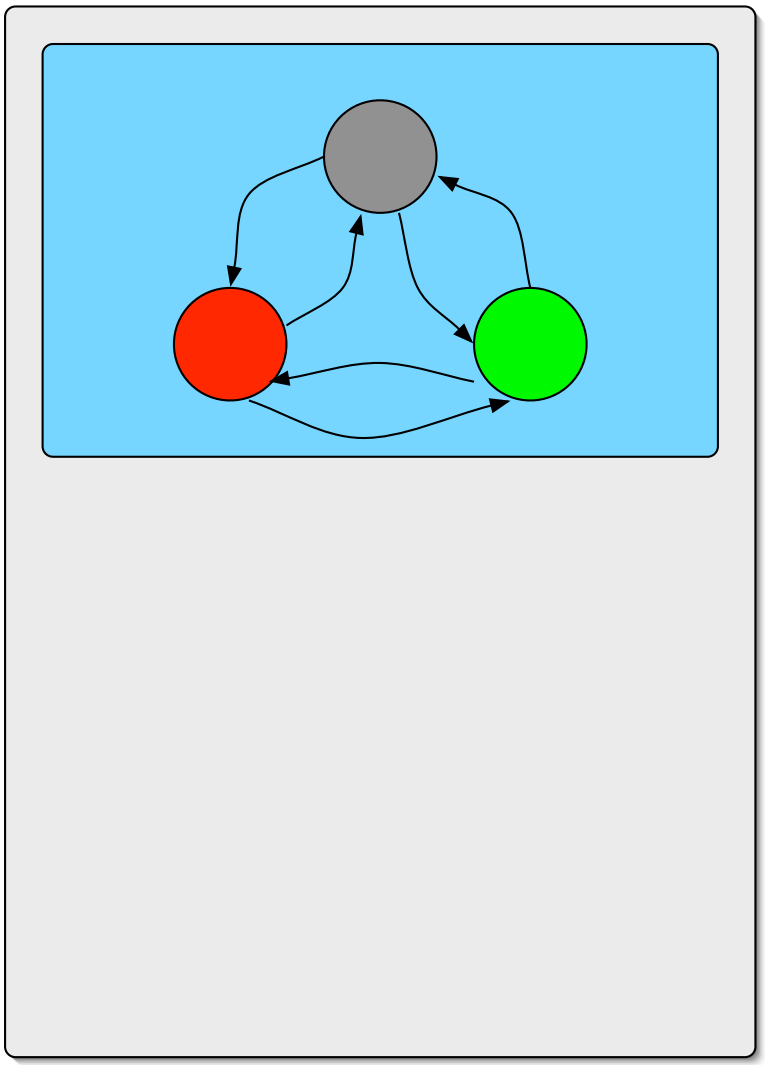
Gestion de la tolérance aux fautes

- But : mise en place des groupes de réplication
- Mécanismes :
  - Détection de fautes
  - Diffusion atomique
  - Composition de groupe
  - Auto-organisation
    - Conservation du degré de réplication



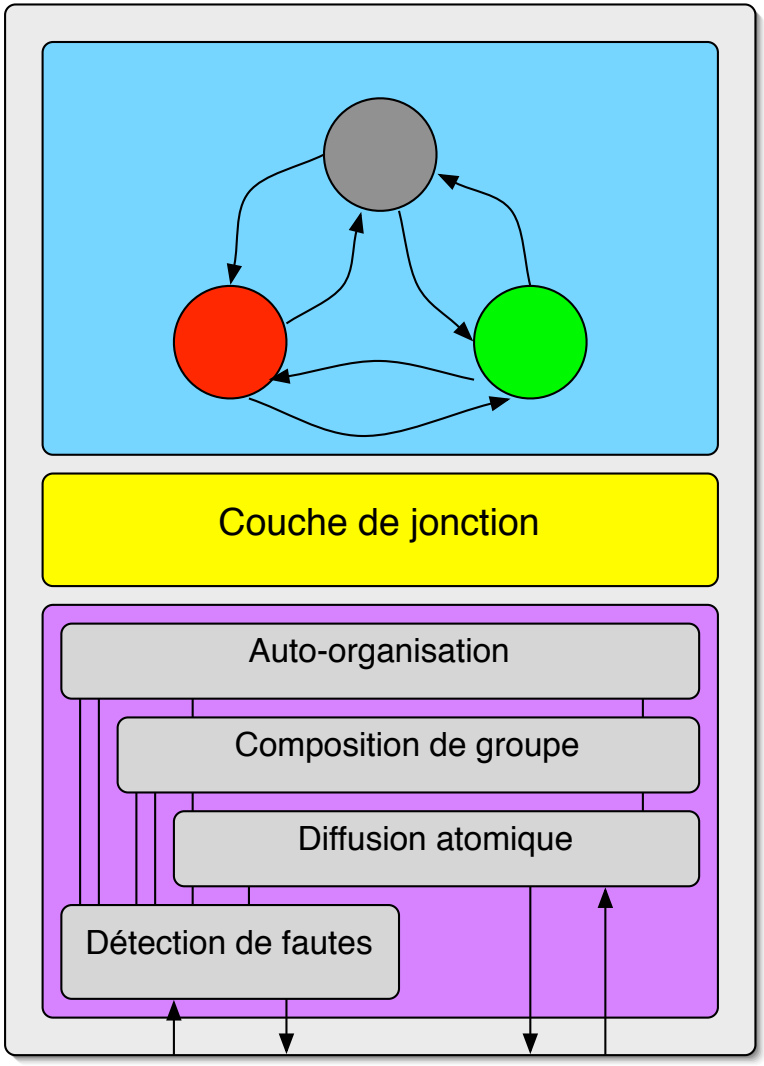
Gestion de la cohérence des données

- But : gestion de la cohérence des données
- Propriétés :
  - Selon le modèle de cohérence
- Mise en œuvre d'un protocole de cohérence

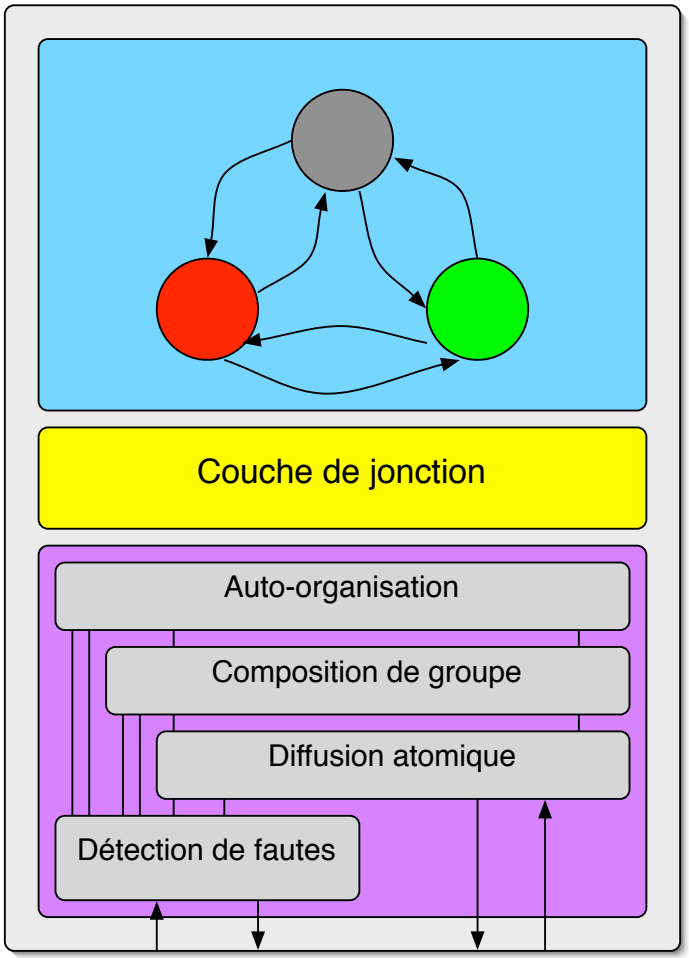


Couche de jonction

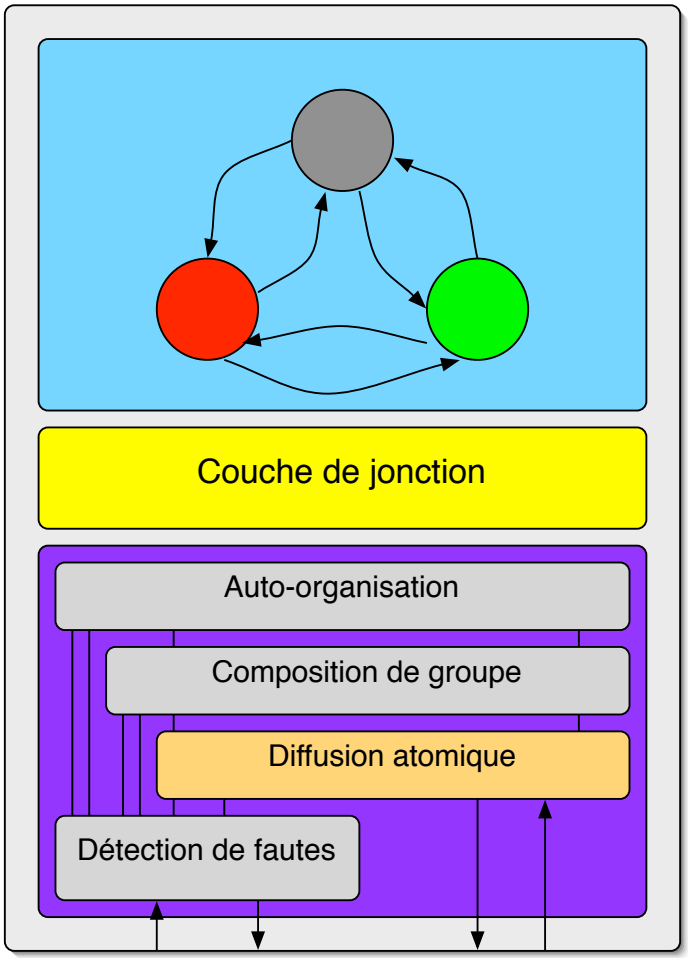
- Rôle
  - Offrir des interfaces génériques
  - Concentrer les interactions entre les couches



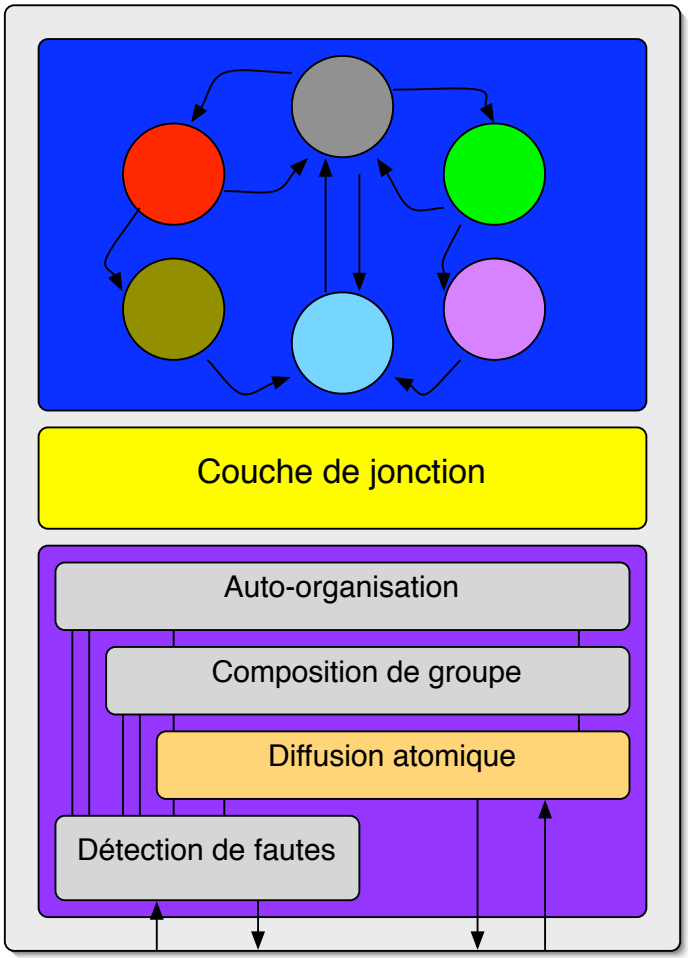
Une architecture logicielle multi-protocole



Une architecture logicielle multi-protocole



Une architecture logicielle multi-protocole



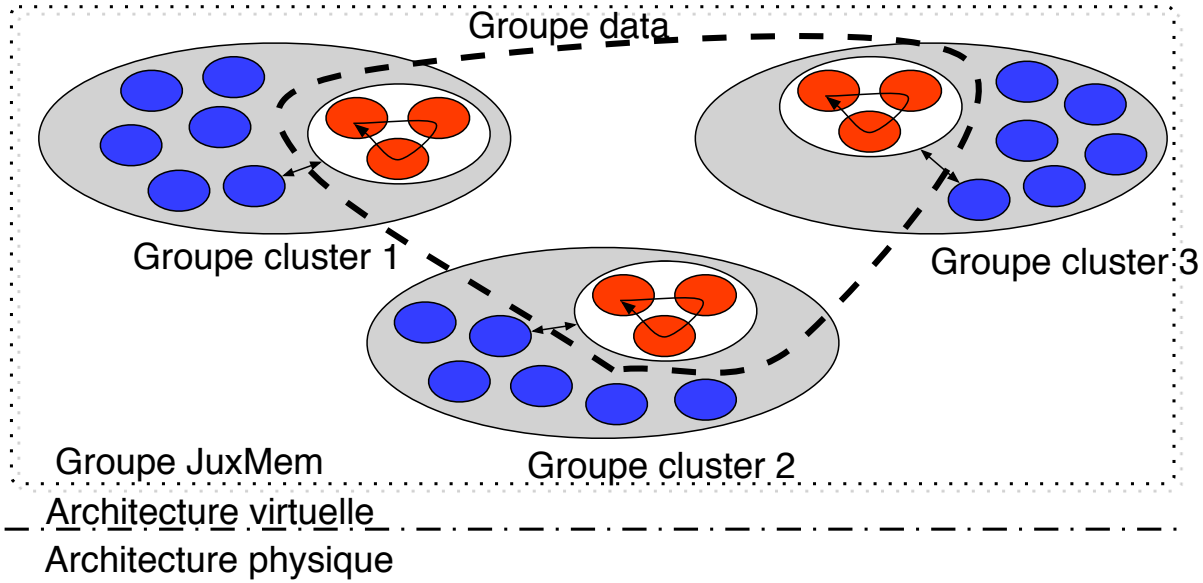
Cadre de mise en œuvre : JuxMem

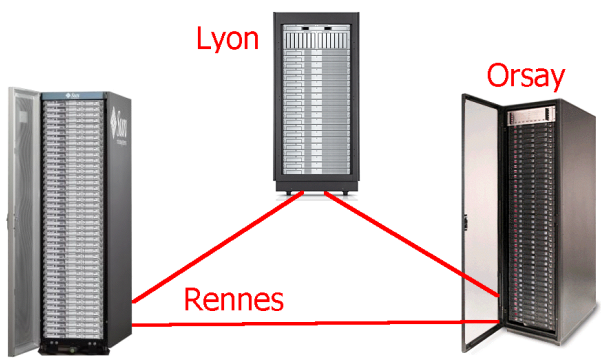
- Un service de partage de données pour la grille
- Développé au sein du projet PARIS à l'IRISA
  - Cadre : projet GDS de l'ACI Masse de Données
- Objectifs
  - Partage de données transparent
    - Localisation, transfert, cohérence
  - Persistance de données
    - Tolérance aux fautes
- Une approche hybride
  - Systèmes pair-à-pair
  - Systèmes à mémoire virtuellement partagée
  - Systèmes distribués tolérants aux fautes

JXTA - Sun Microsystems

- Notion de pair
- Notion de groupe de pairs
- Mécanismes de communication
  - Point-à-point
  - Diffusion
- Notion d'annonce
- Mécanismes de publication/recherche
- Groupes JXTA différents des groupes de réplication

Architecture générale

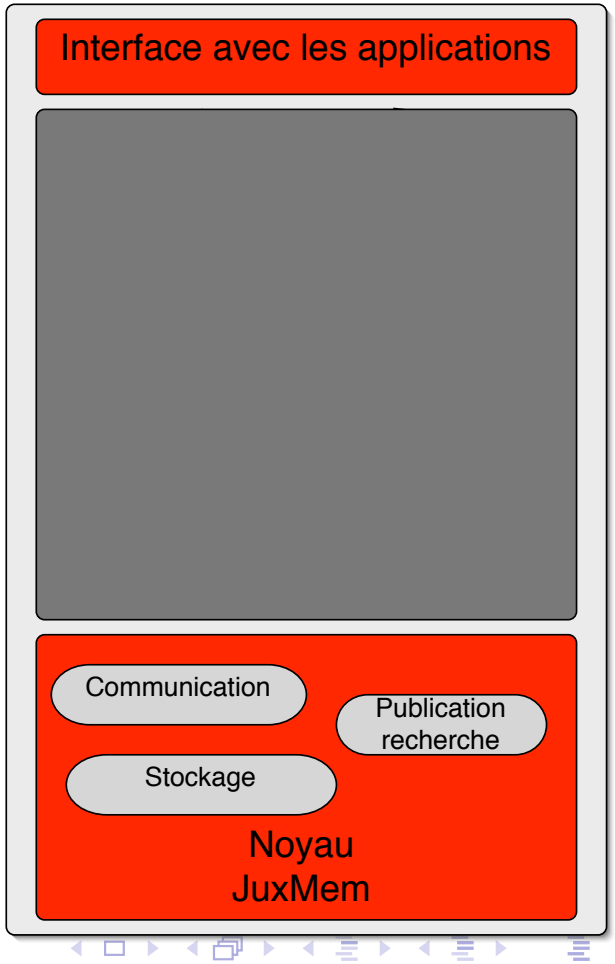




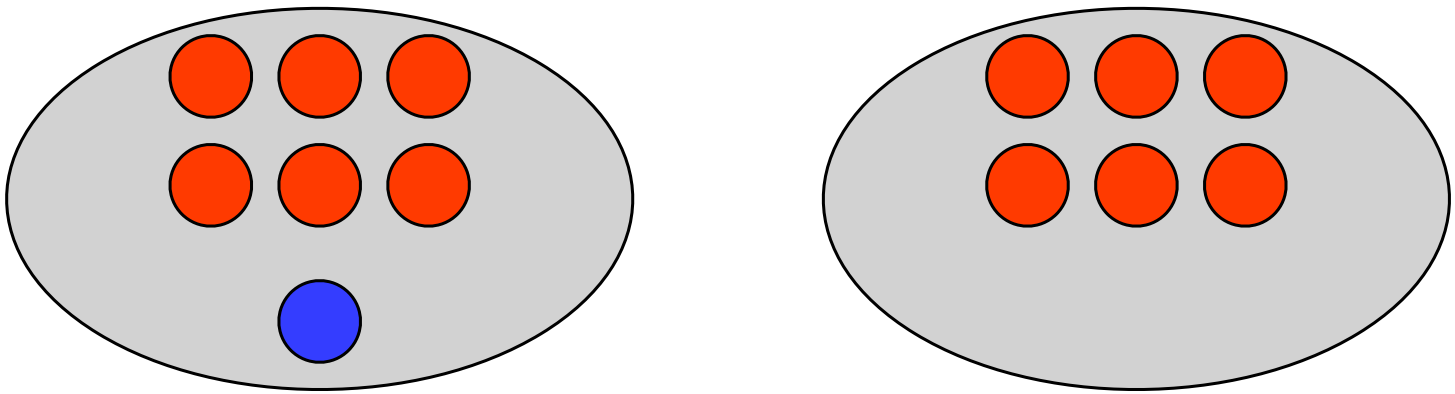
Le noyau JuxMem : *juk*

Réalisé au cours de la thèse de Mathieu Jan

- Permet de s'abstraire de JXTA
- Fonctionnalités
  - Communication
  - Publication/Recherche
    - Locale
    - Globale
  - Stockage

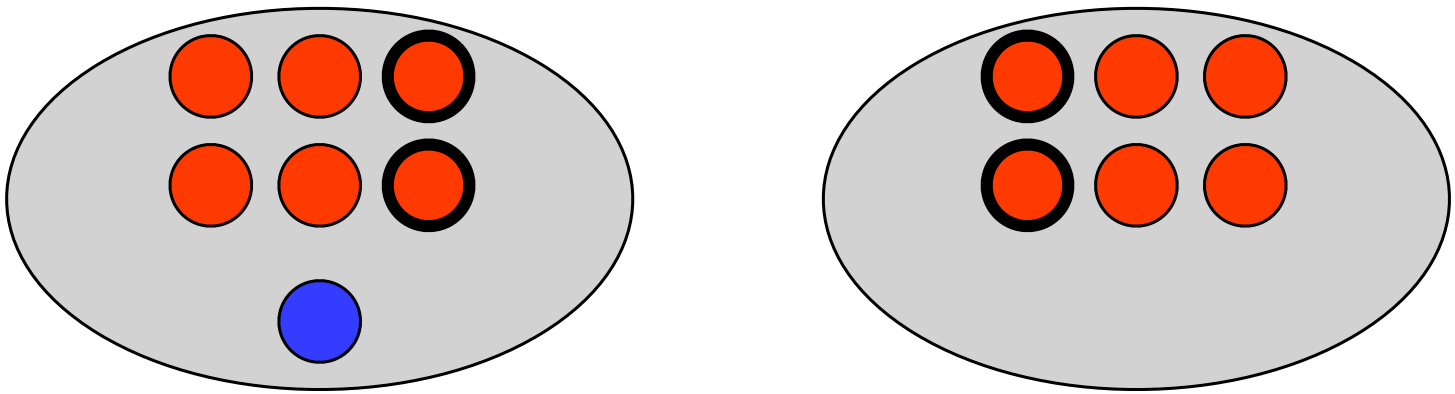


Exemple de scénario



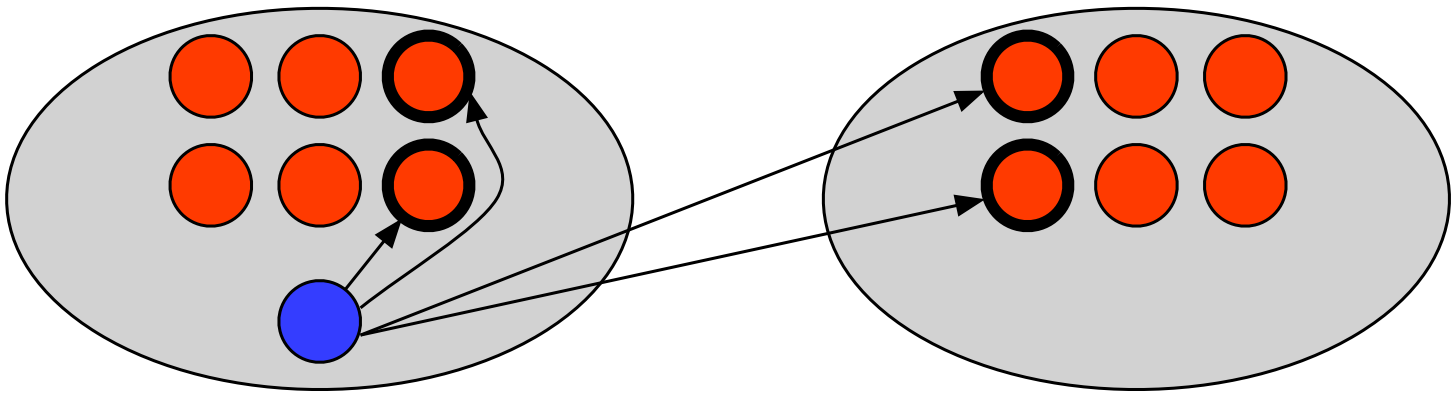
data_id = juxmem_malloc(size, 2, 2)

Exemple de scénario



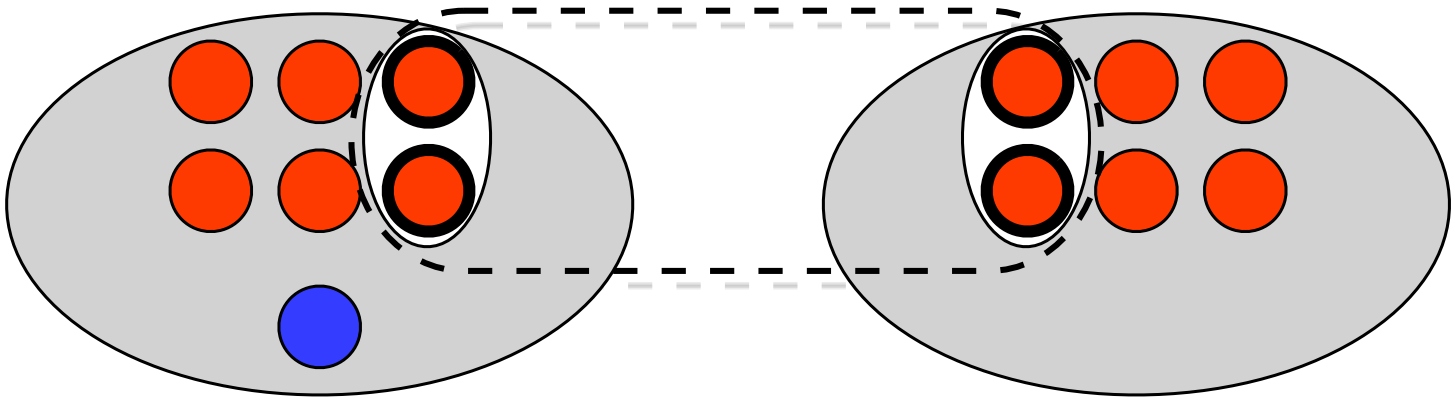
data_id = juxmem_malloc(size, 2, 2)

Exemple de scénario



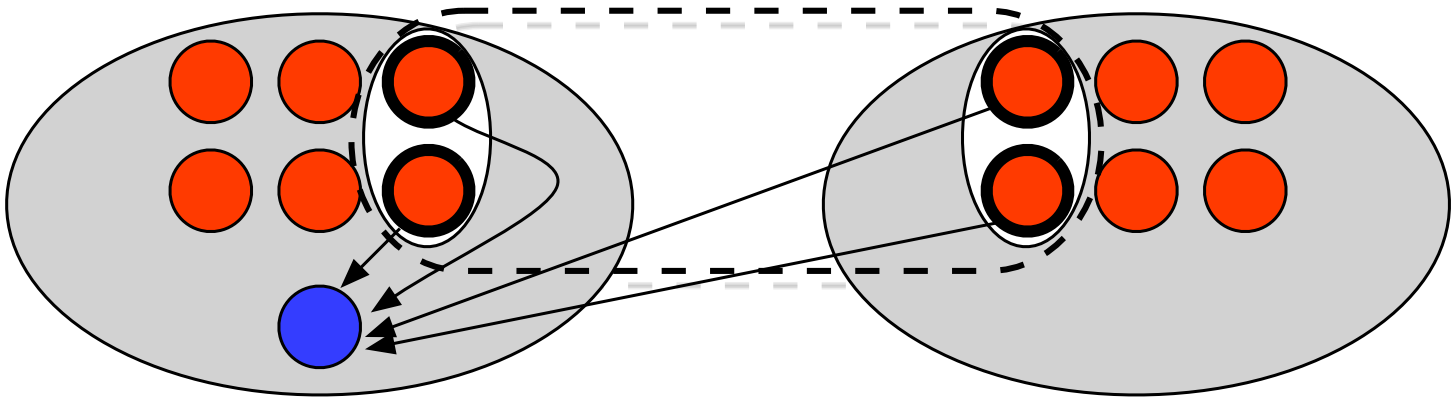
data_id = juxmem_malloc(size, 2, 2)

Exemple de scénario



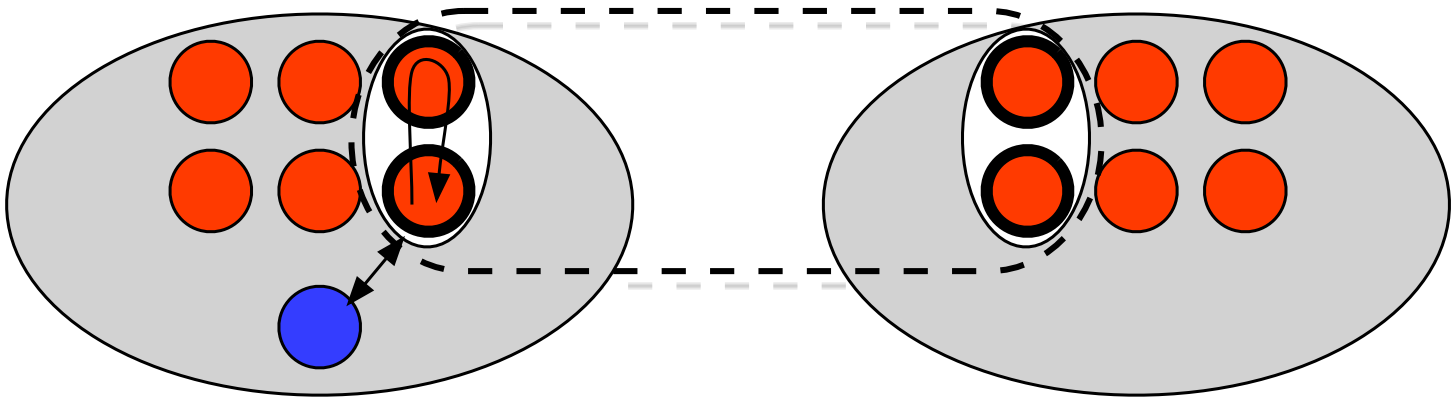
data_id = juxmem_malloc(size, 2, 2)

Exemple de scénario



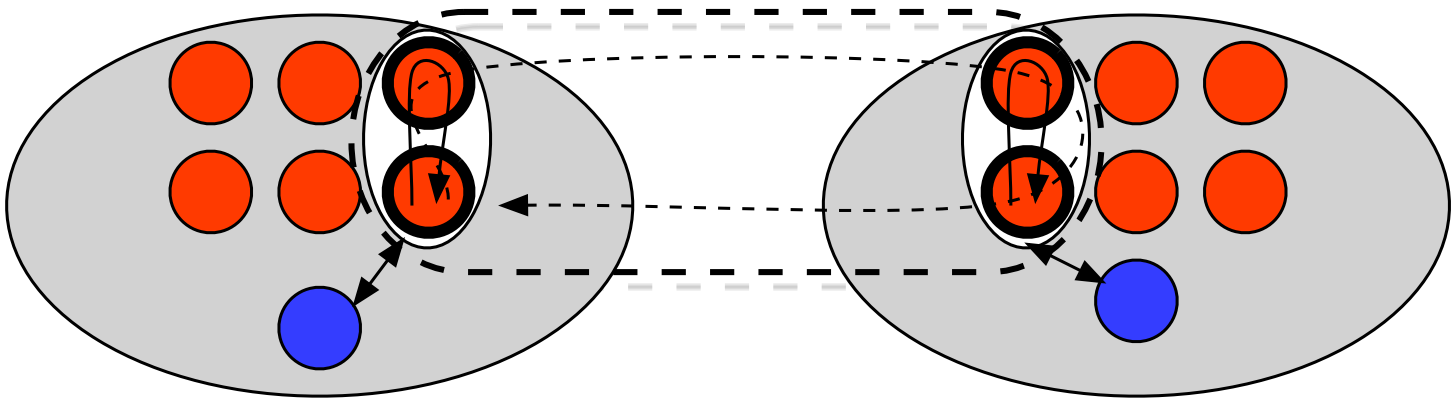
data_id = juxmem_malloc(size, 2, 2)

Exemple de scénario



juxmem acquire(data_id)
juxmem write(data_id)
juxmem release(data_id)

Exemple de scénario

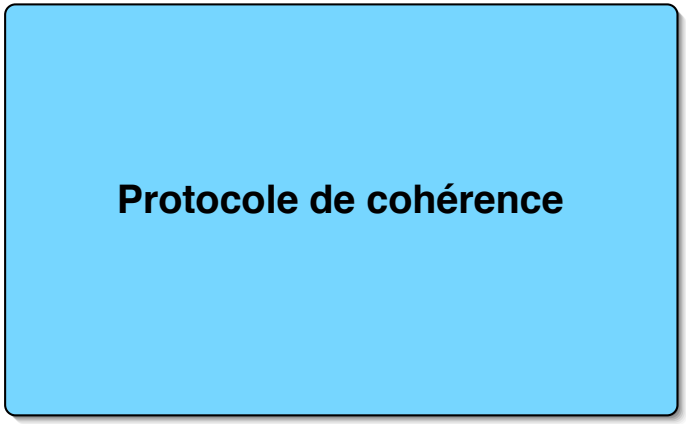


juxmem acquire(data_id)
juxmem write(data_id)
juxmem release(data_id)

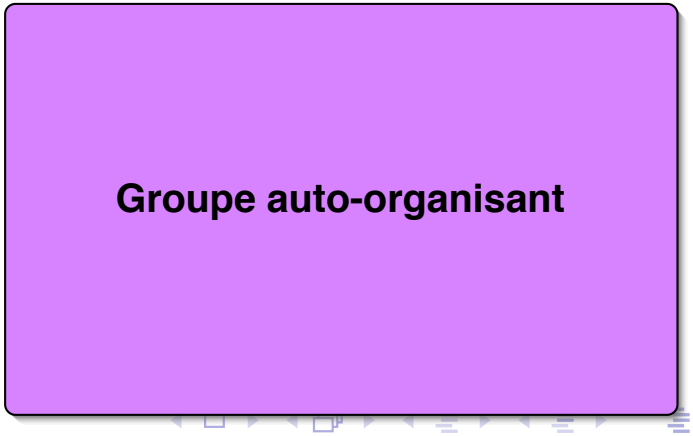
juxmem acquire_read(data_id)
juxmem read(data_id)
juxmem release(data_id)

Mise en œuvre des couches

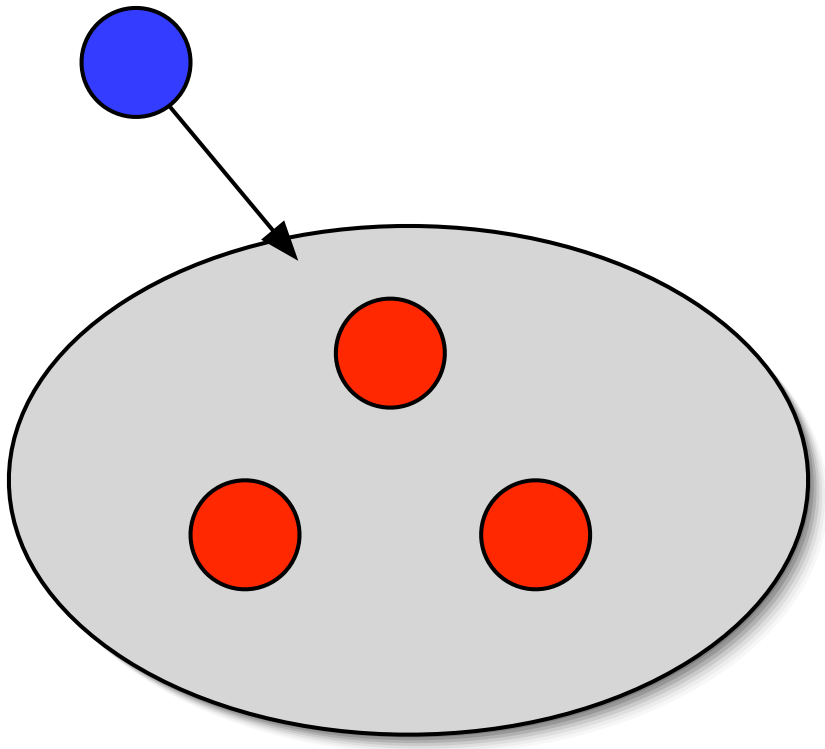
- Mise en œuvre d'un protocole de cohérence
  - Module client
  - Module local
  - Module global



- Mise en œuvre d'un groupe auto-organisant
  - Module client
  - Module fournisseur

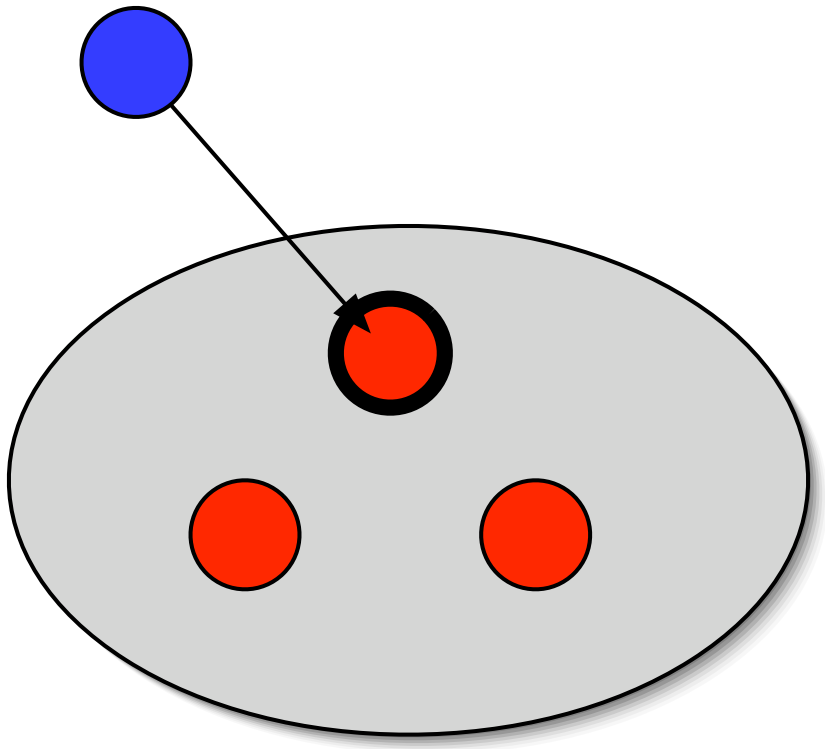


Mise en œuvre des groupes auto-organisants



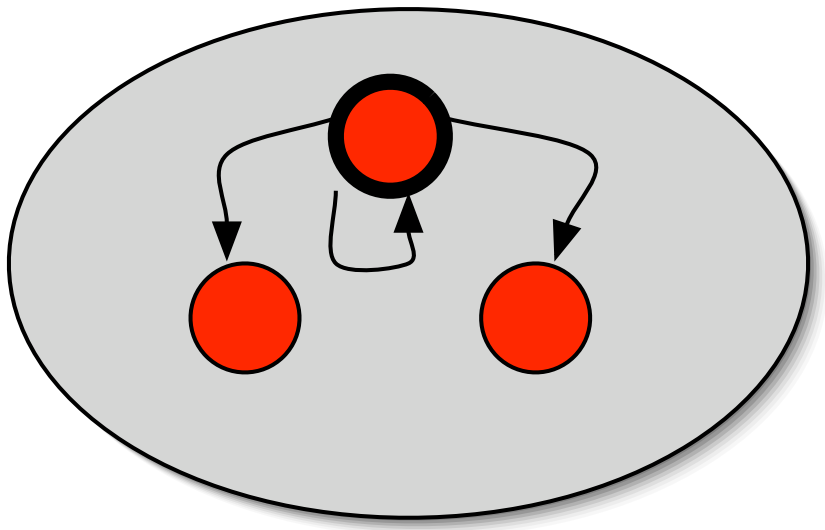
- Approches pour la diffusion atomique
  - Protocole d'accord
  - Nœud séquenceur

Mise en œuvre des groupes auto-organisants



- Approches pour la diffusion atomique
  - Protocole d'accord
  - Nœud séquenceur

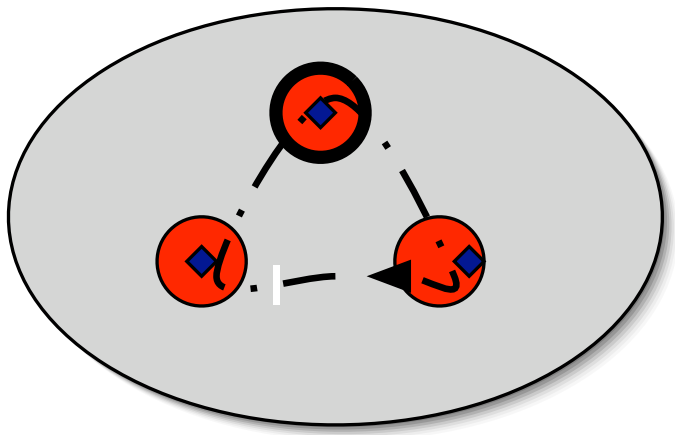
Mise en œuvre des groupes auto-organisants



- Approches pour la diffusion atomique
  - Protocole d'accord
  - Nœud séquenceur

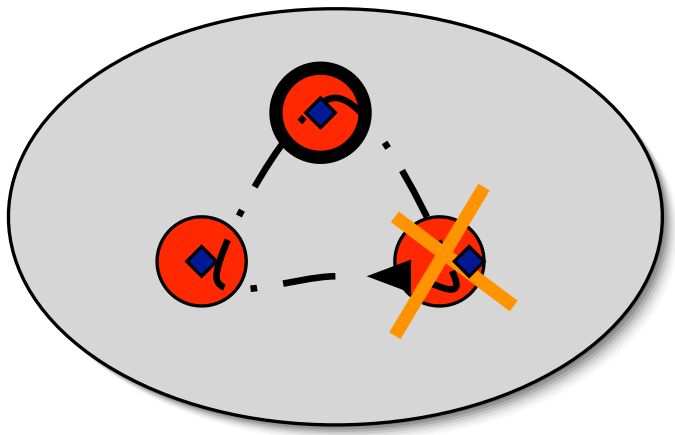
Auto-organisation des groupes

- Détection de fautes
- Politique
  - conservation du degré de réplication
- Faute d'un fournisseur non-séquenceur
  - Recherche d'un remplaçant
  - Gel du protocole
  - Mise à jour du remplaçant
  - Mise à jour du groupe
- Faute du fournisseur séquenceur
  - Choix du séquenceur suivant (élection)



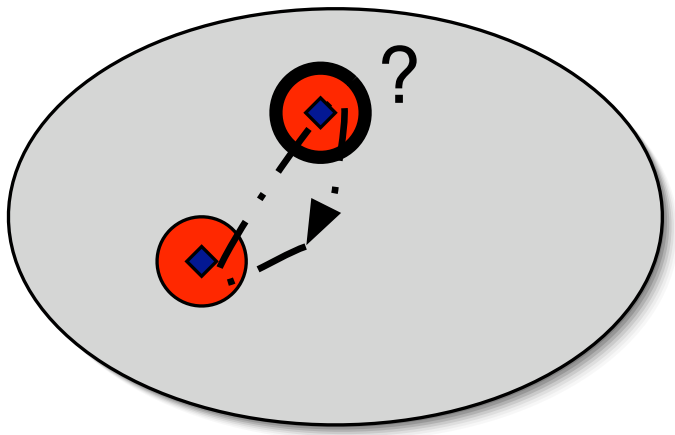
Auto-organisation des groupes

- Détection de fautes
- Politique
  - conservation du degré de réplication
- Faute d'un fournisseur non-séquenceur
  - Recherche d'un remplaçant
  - Gel du protocole
  - Mise à jour du remplaçant
  - Mise à jour du groupe
- Faute du fournisseur séquenceur
  - Choix du séquenceur suivant (élection)



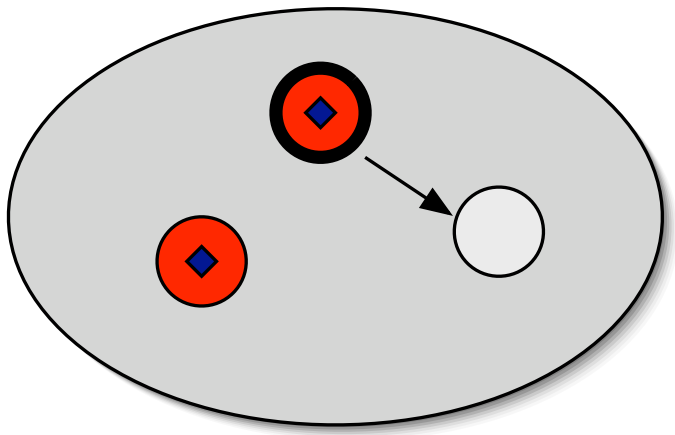
Auto-organisation des groupes

- Détection de fautes
- Politique
  - conservation du degré de réplication
- Faute d'un fournisseur non-séquenceur
  - Recherche d'un remplaçant
  - Gel du protocole
  - Mise à jour du remplaçant
  - Mise à jour du groupe
- Faute du fournisseur séquenceur
  - Choix du séquenceur suivant (élection)



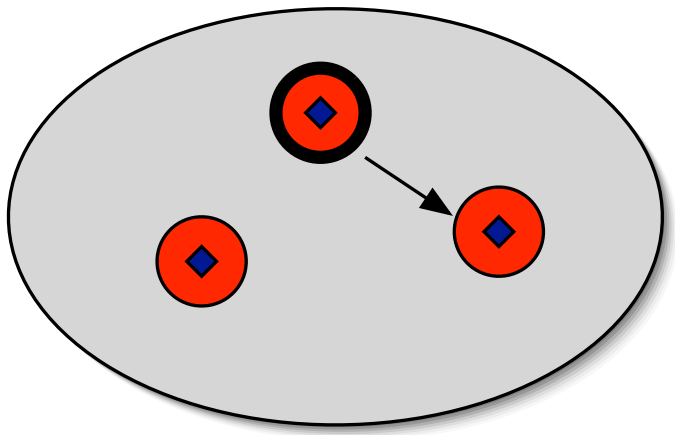
Auto-organisation des groupes

- Détection de fautes
- Politique
  - conservation du degré de réplication
- Faute d'un fournisseur non-séquenceur
  - Recherche d'un remplaçant
  - Gel du protocole
  - Mise à jour du remplaçant
  - Mise à jour du groupe
- Faute du fournisseur séquenceur
  - Choix du séquenceur suivant (élection)



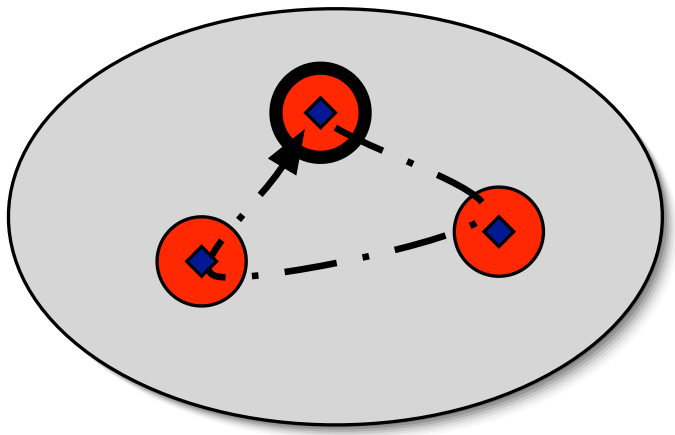
Auto-organisation des groupes

- Détection de fautes
- Politique
  - conservation du degré de réplication
- Faute d'un fournisseur non-séquenceur
  - Recherche d'un remplaçant
  - Gel du protocole
  - Mise à jour du remplaçant
  - Mise à jour du groupe
- Faute du fournisseur séquenceur
  - Choix du séquenceur suivant (élection)

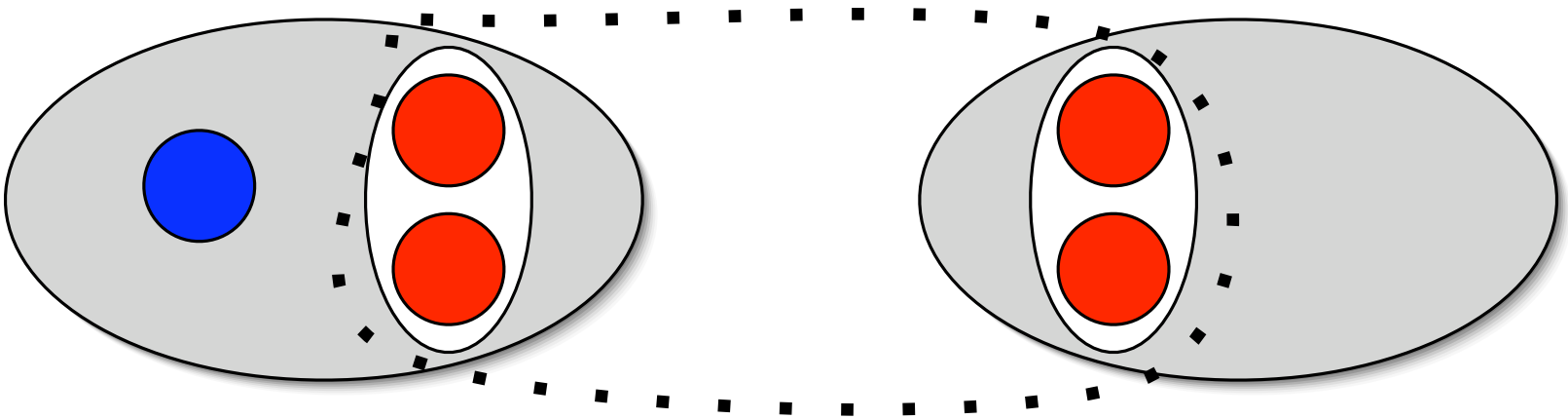


Auto-organisation des groupes

- Détection de fautes
- Politique
  - conservation du degré de réplication
- Faute d'un fournisseur non-séquenceur
  - Recherche d'un remplaçant
  - Gel du protocole
  - Mise à jour du remplaçant
  - Mise à jour du groupe
- Faute du fournisseur séquenceur
  - Choix du séquenceur suivant (élection)

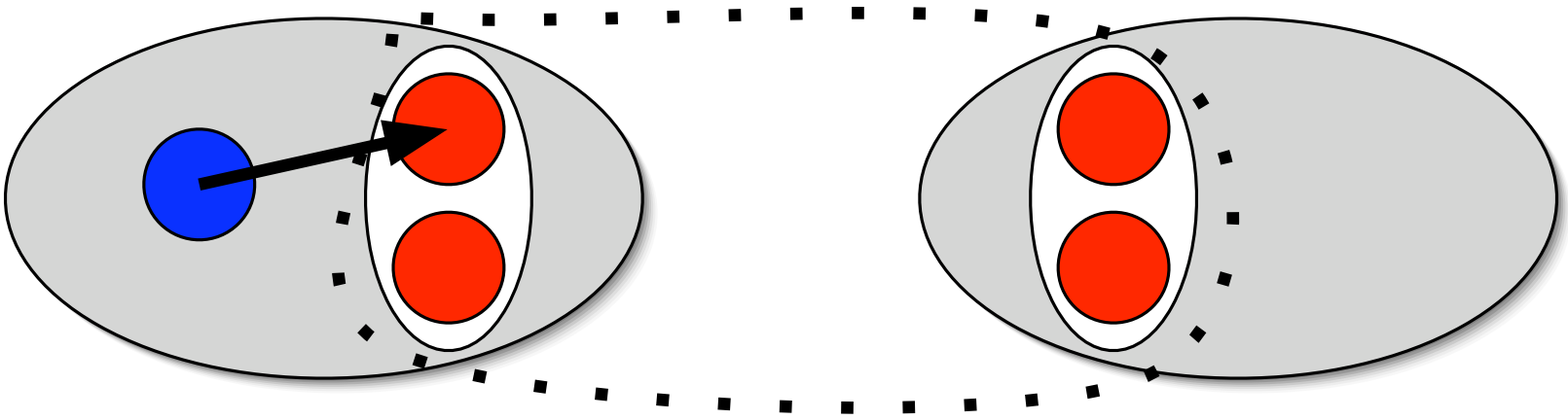


Fonctionnement général



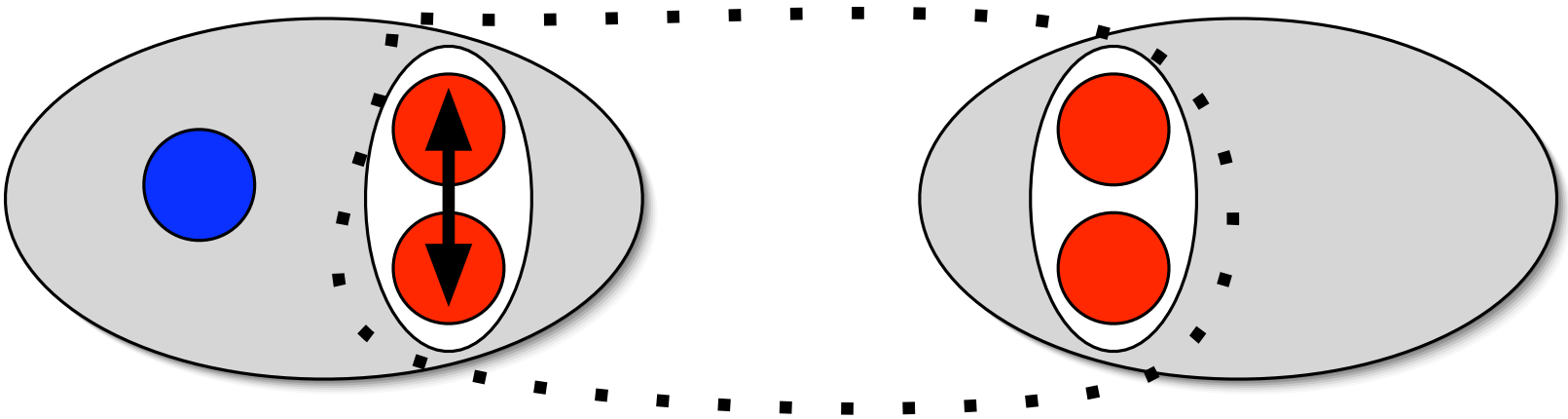
juxmem_acquire(data_id)

Fonctionnement général



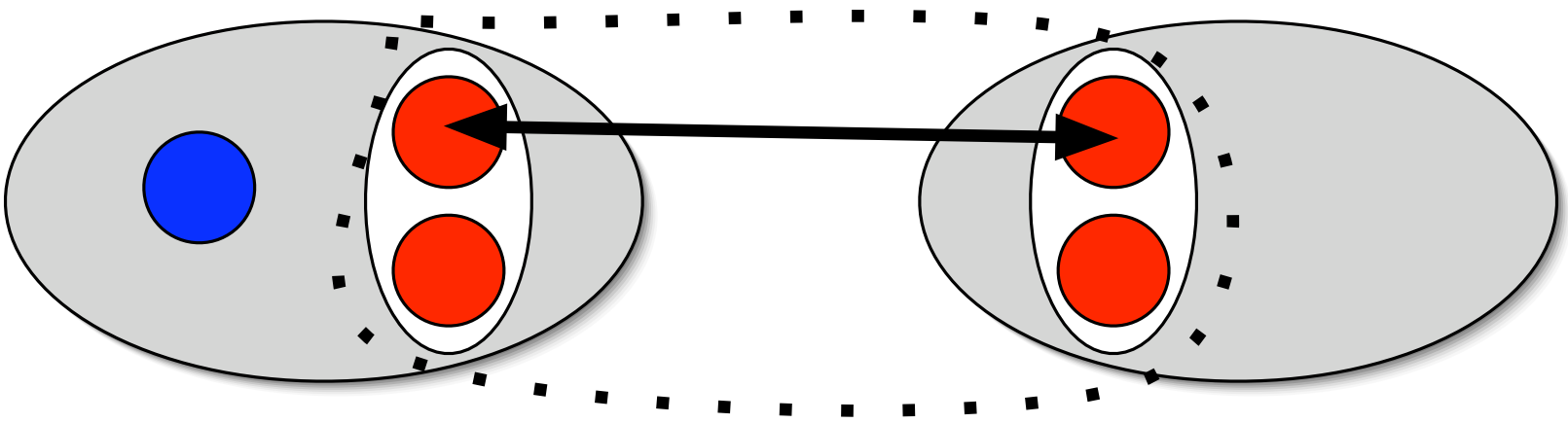
juxmem_acquire(data_id)

Fonctionnement général



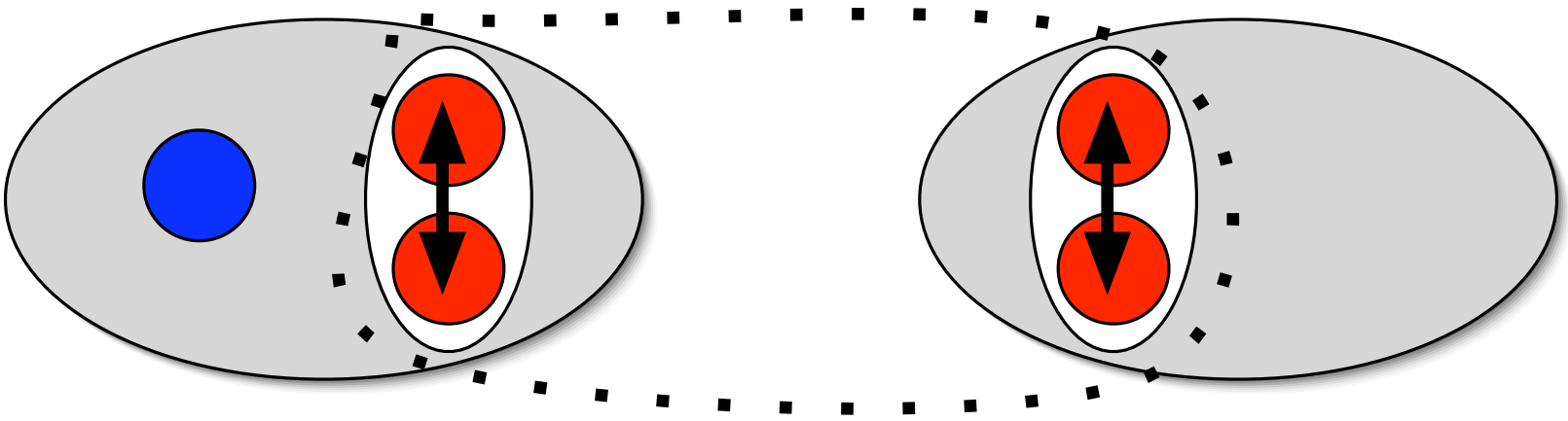
juxmem_acquire(data_id)

Fonctionnement général



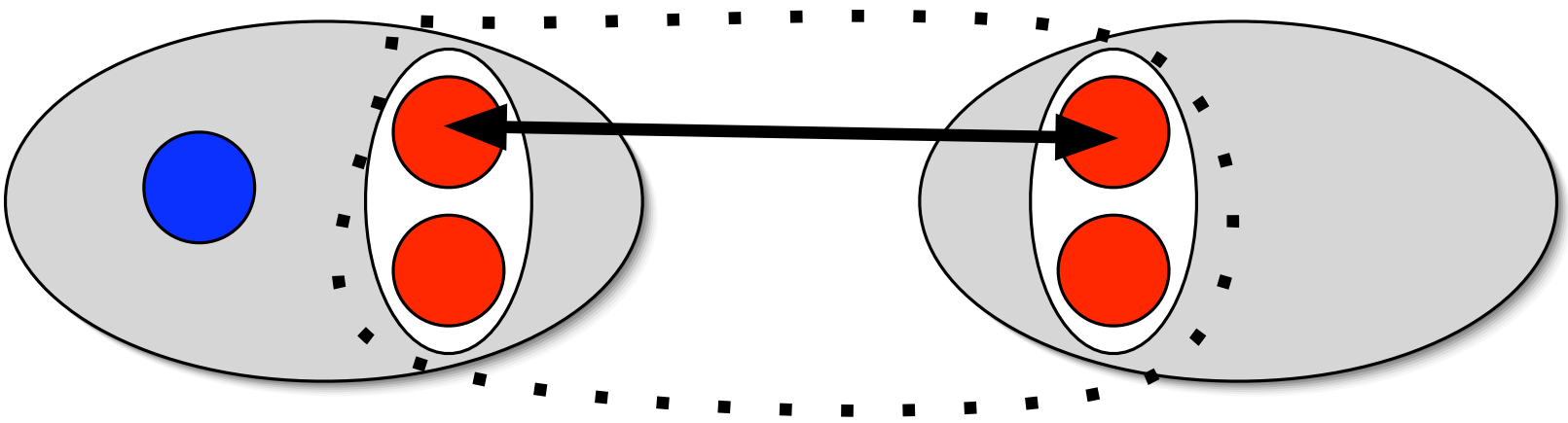
juxmem_acquire(data_id)

Fonctionnement général



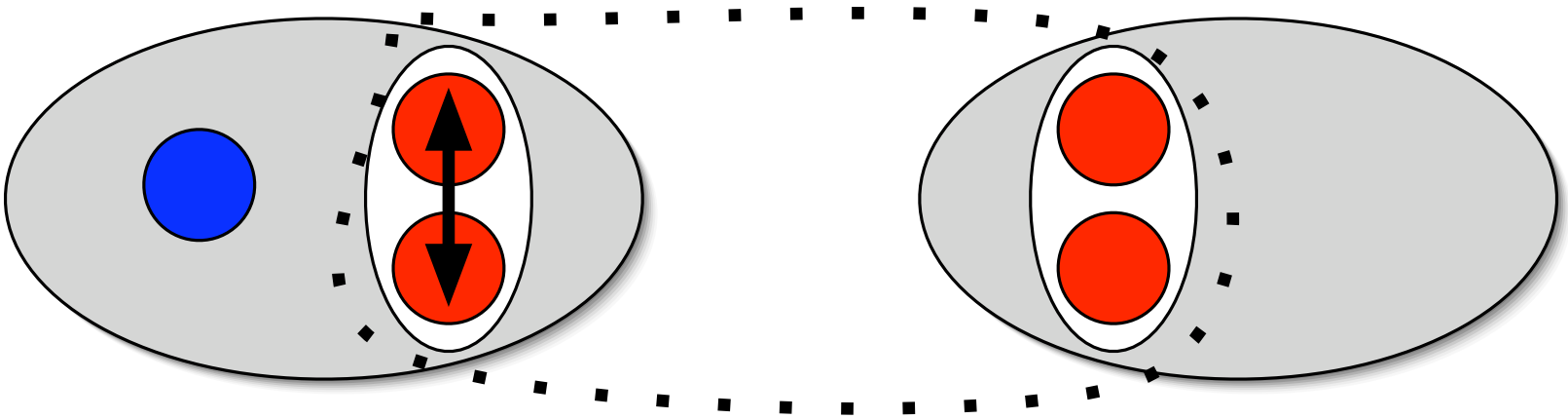
juxmem_acquire(data_id)

Fonctionnement général



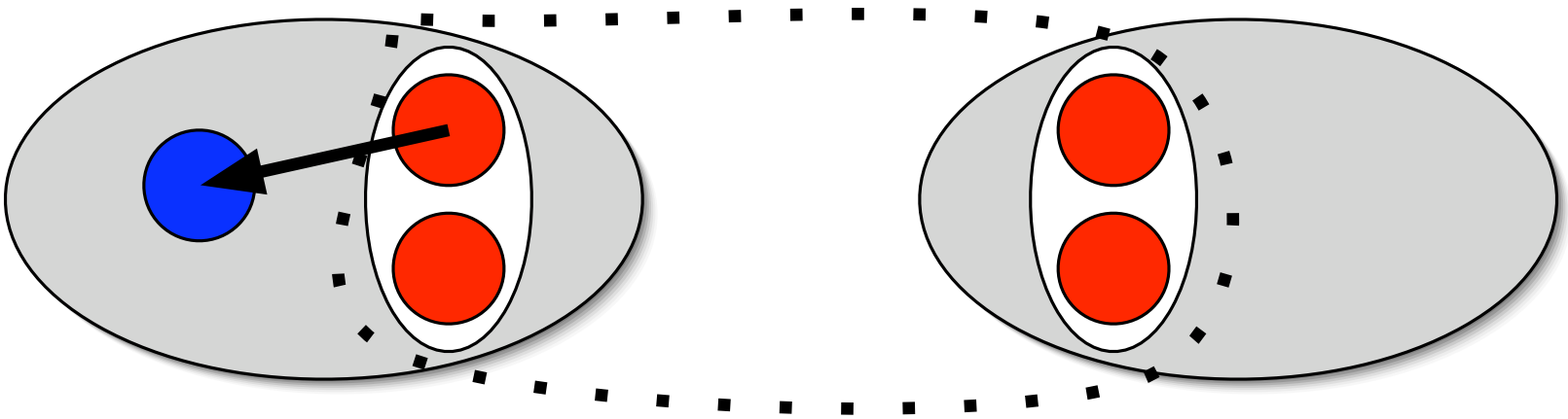
juxmem_acquire(data_id)

Fonctionnement général



juxmem_acquire(data_id)

Fonctionnement général

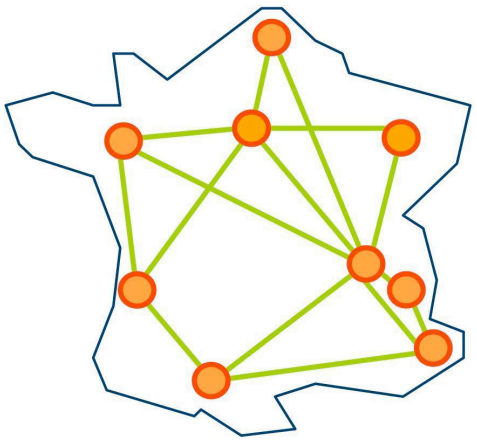


juxmem_acquire(data_id)

Méthodologie

Expérimentation sur des architectures réelles

- Multiples types d'évaluation
  - Preuves formelles
  - Simulations
  - Expérimentations
- Expérimentations sur grilles
  - Grid'5000
  - Réservation : OAR/OARGrid [IMAG]
  - Déploiement : ADAGE [IRISA]
  - Récupération des résultats



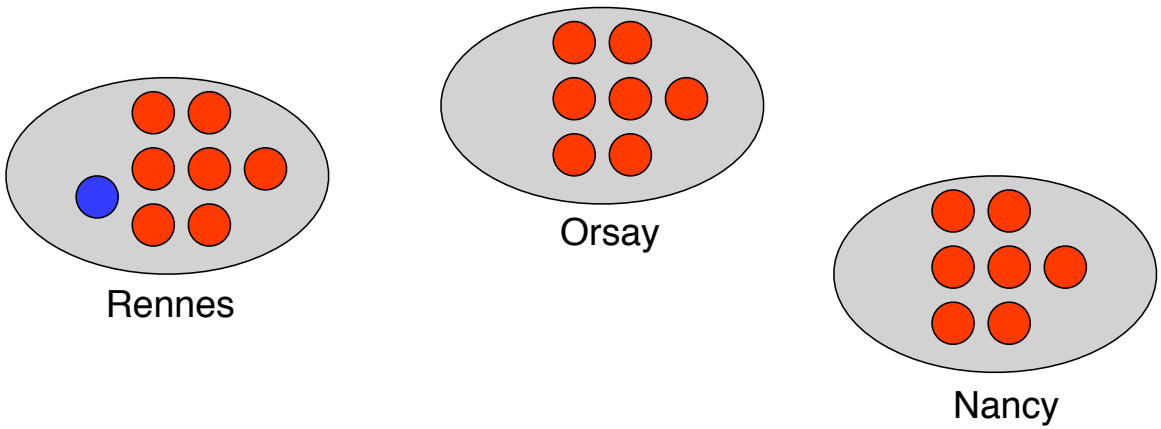
Grid'5000

Évaluations

1. Coût des mécanismes de réplication
2. Bénéfice de l'approche hiérarchique
3. Impact des fautes

1 : Coût de la réplication

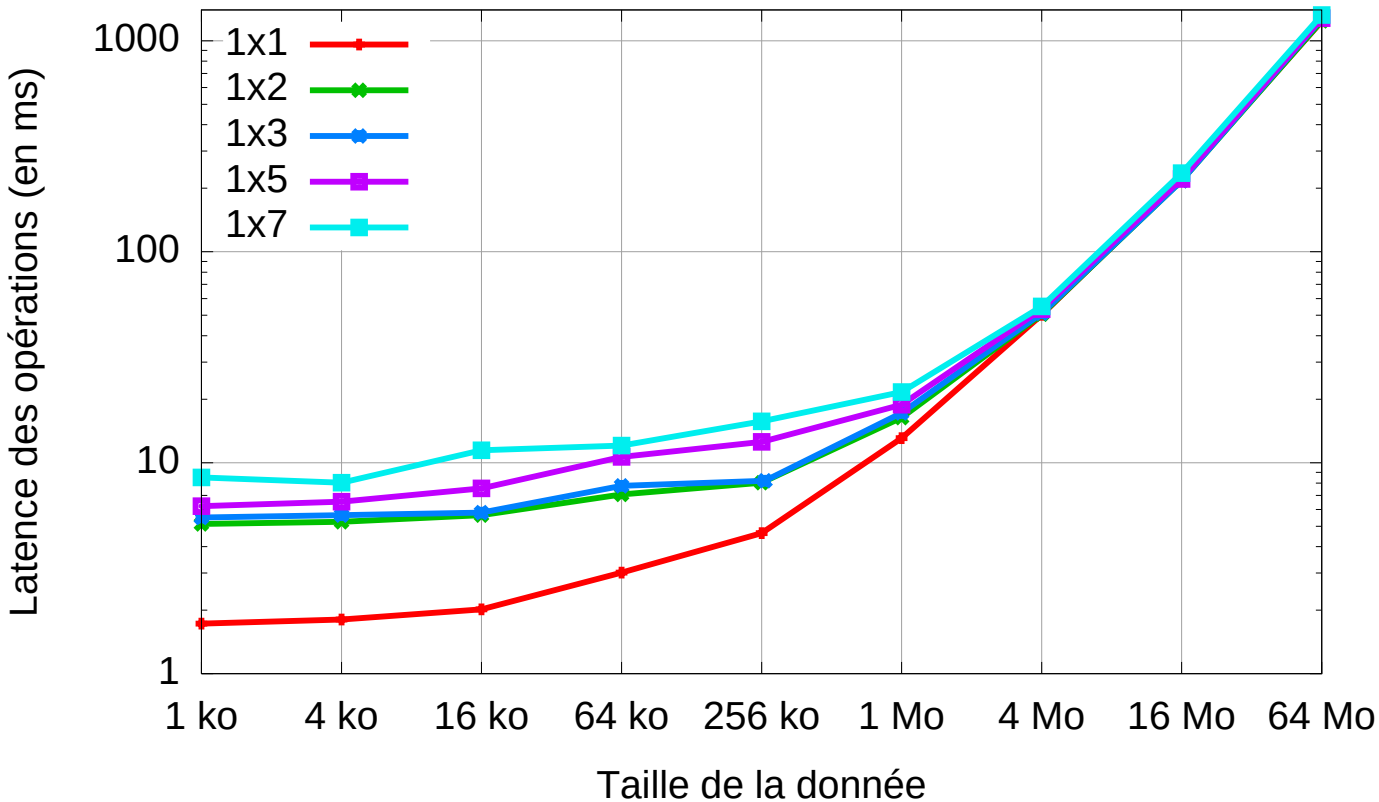
Description des tests



- 3 grappes géographiquement réparties
- 3×7 fournisseurs
- 1 client

1 : Coût de la réplication

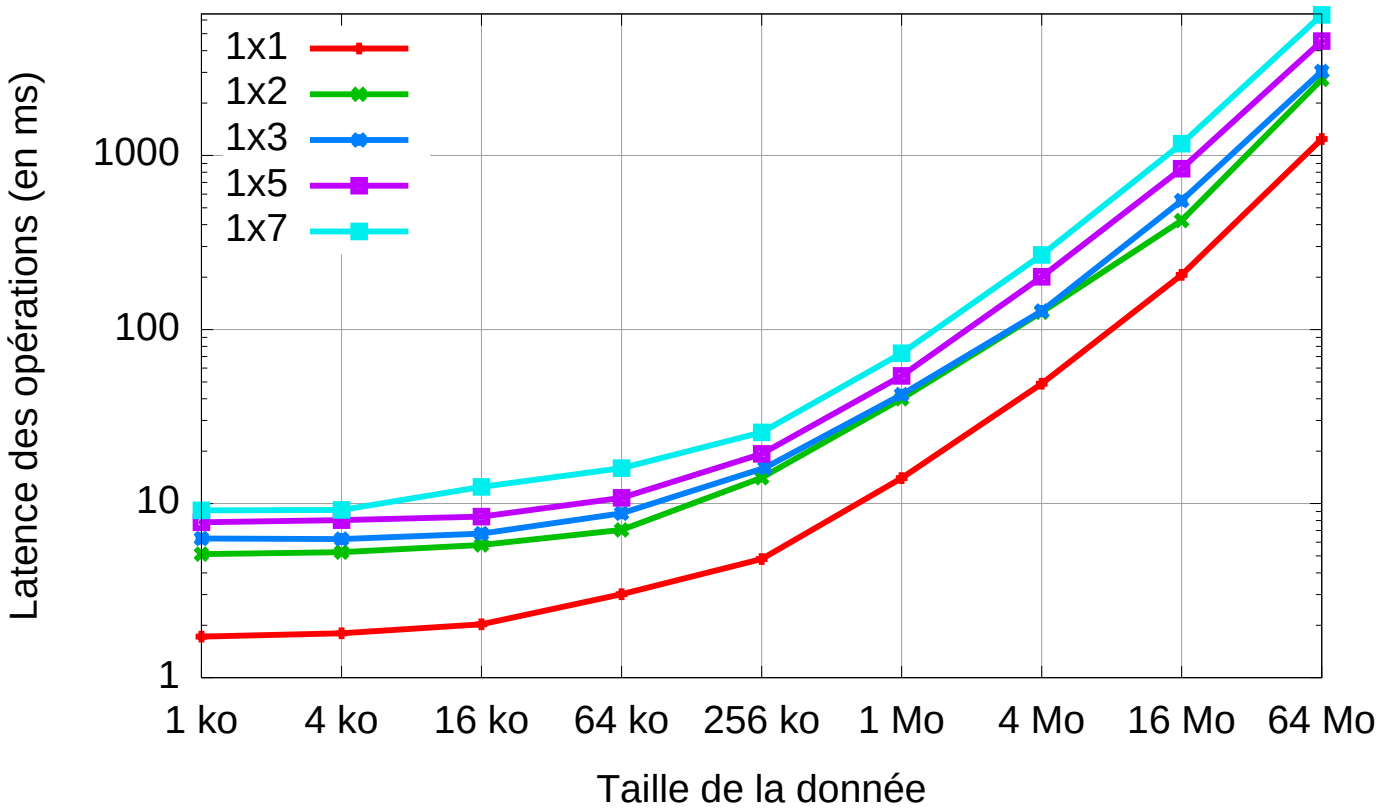
Dans une grappe - coût des opérations de lecture



$$Lecture = acquire\_read + read + release$$

1 : Coût de la réplication

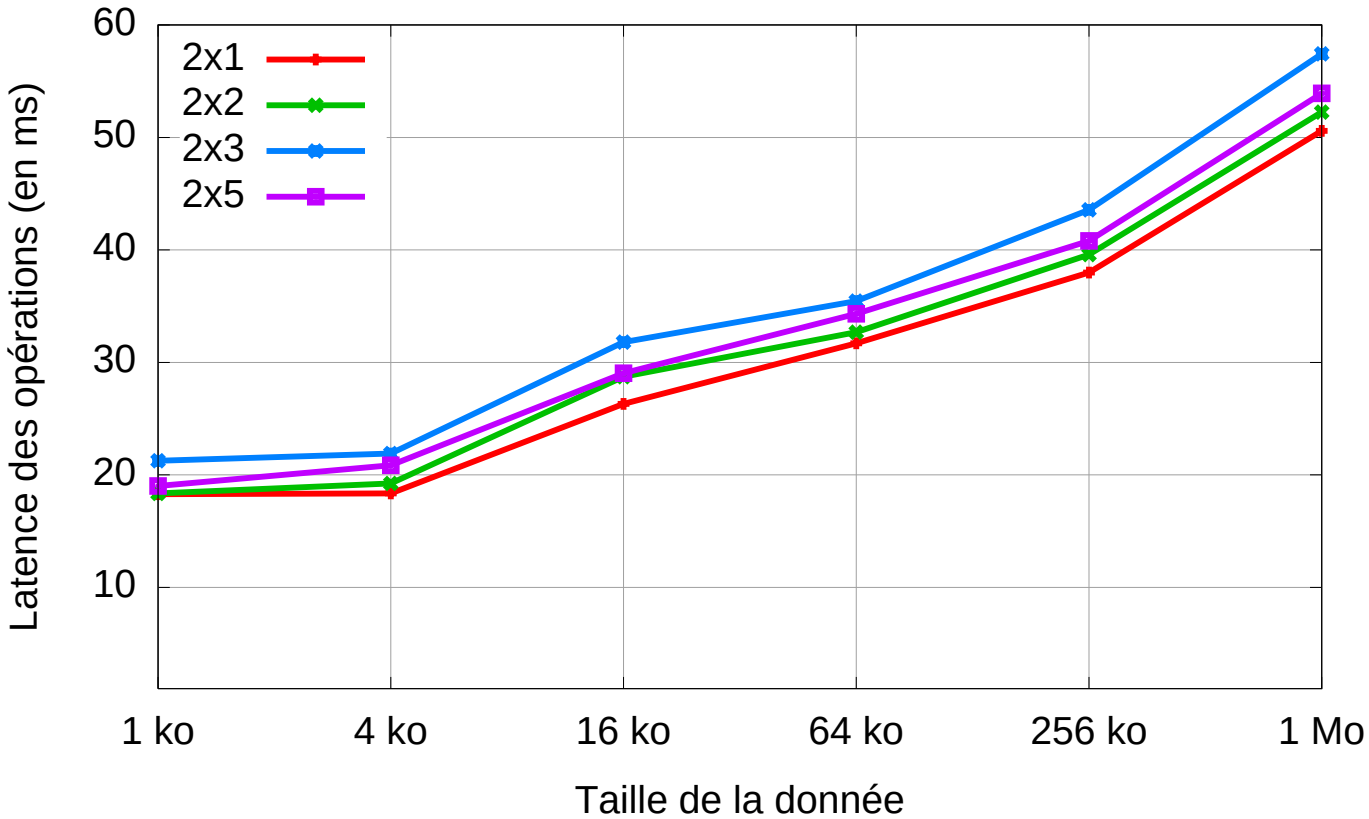
Dans une grappe - coût des opérations d'écriture



$$\textit{Ecriture} = \textit{acquire} + \textit{write} + \textit{release}$$

1 : Coût de la réplication

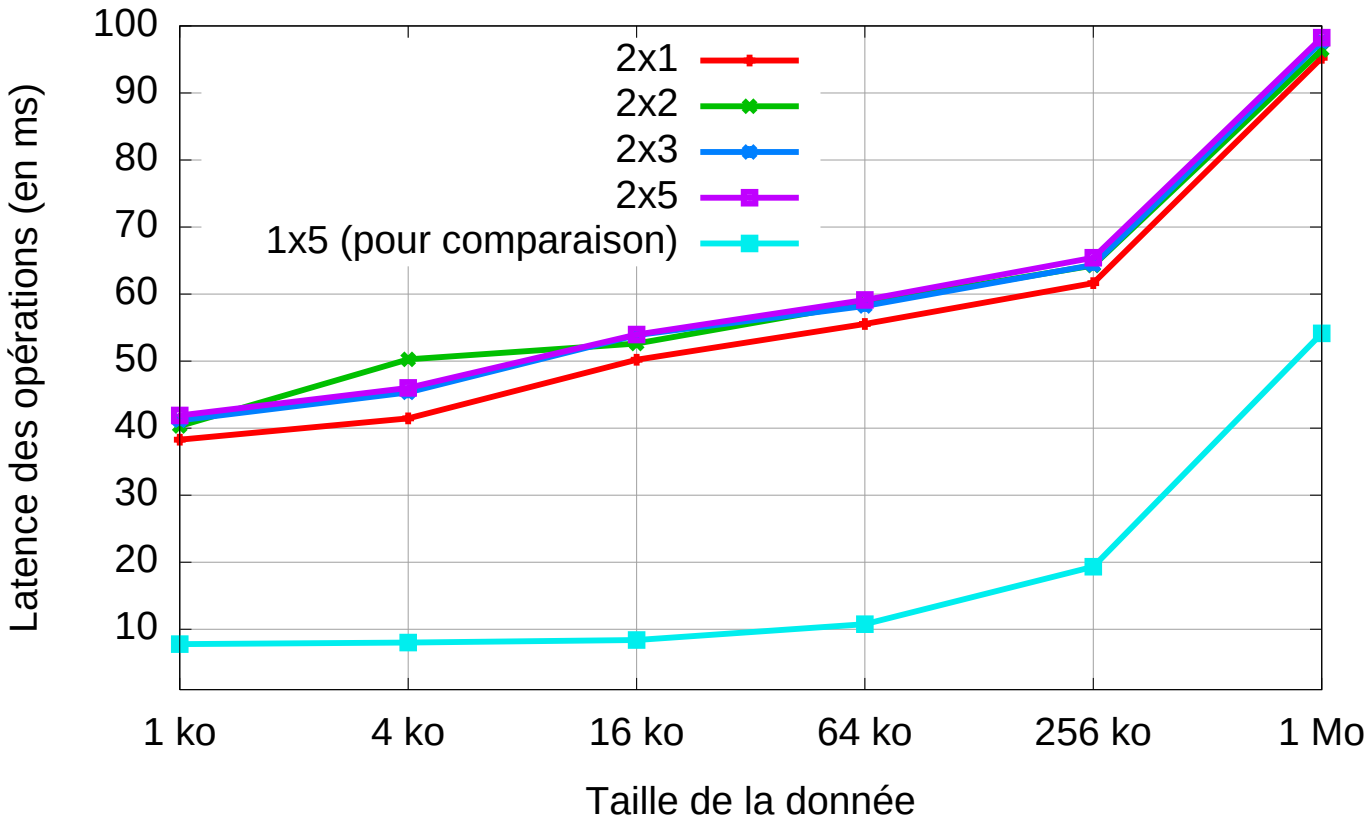
Dans deux grappes - coût des opérations de lecture



$$Lecture = acquire\_read + read + release$$

1 : Coût de la réplication

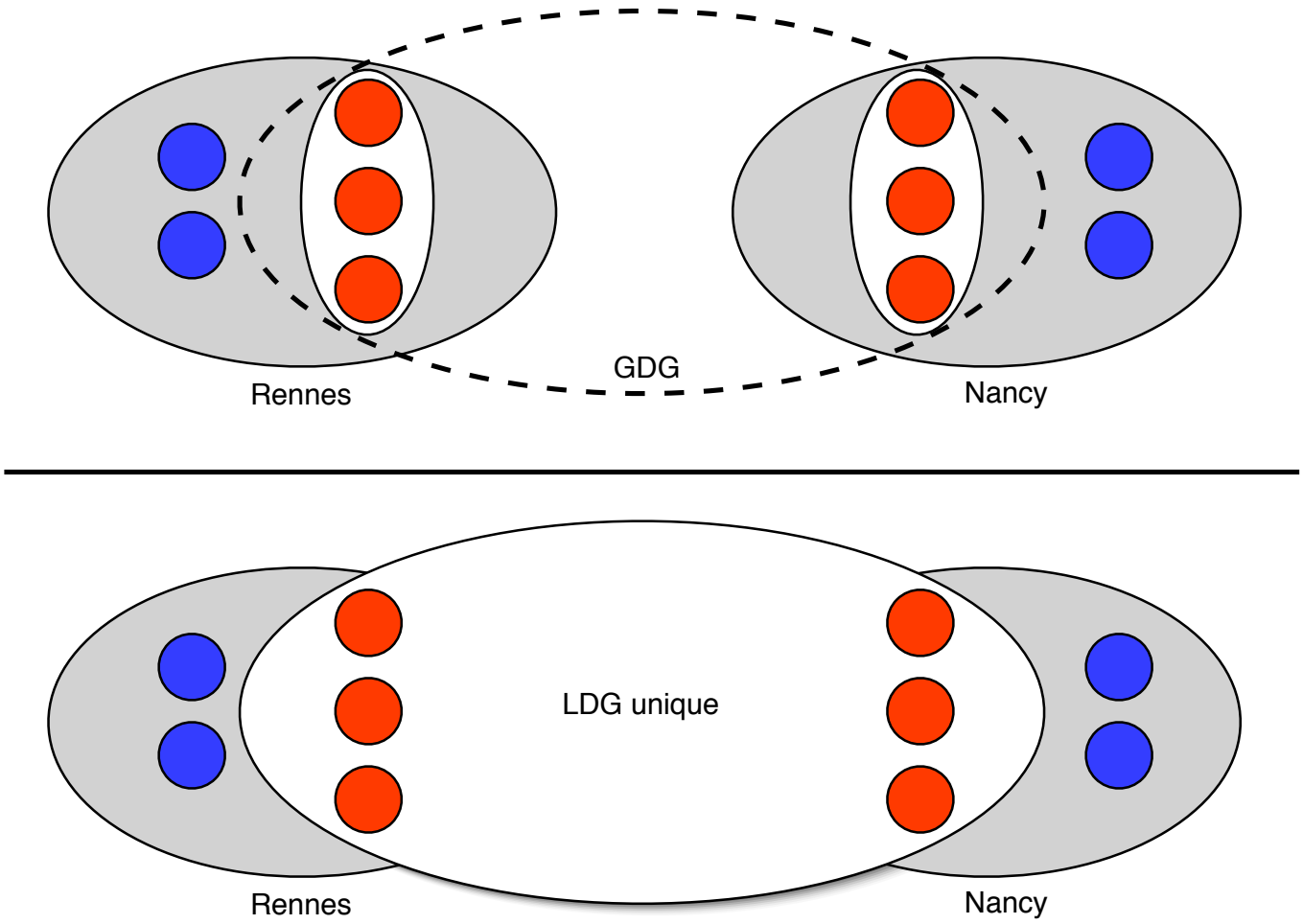
Dans deux grappes - coût des opérations d'écriture



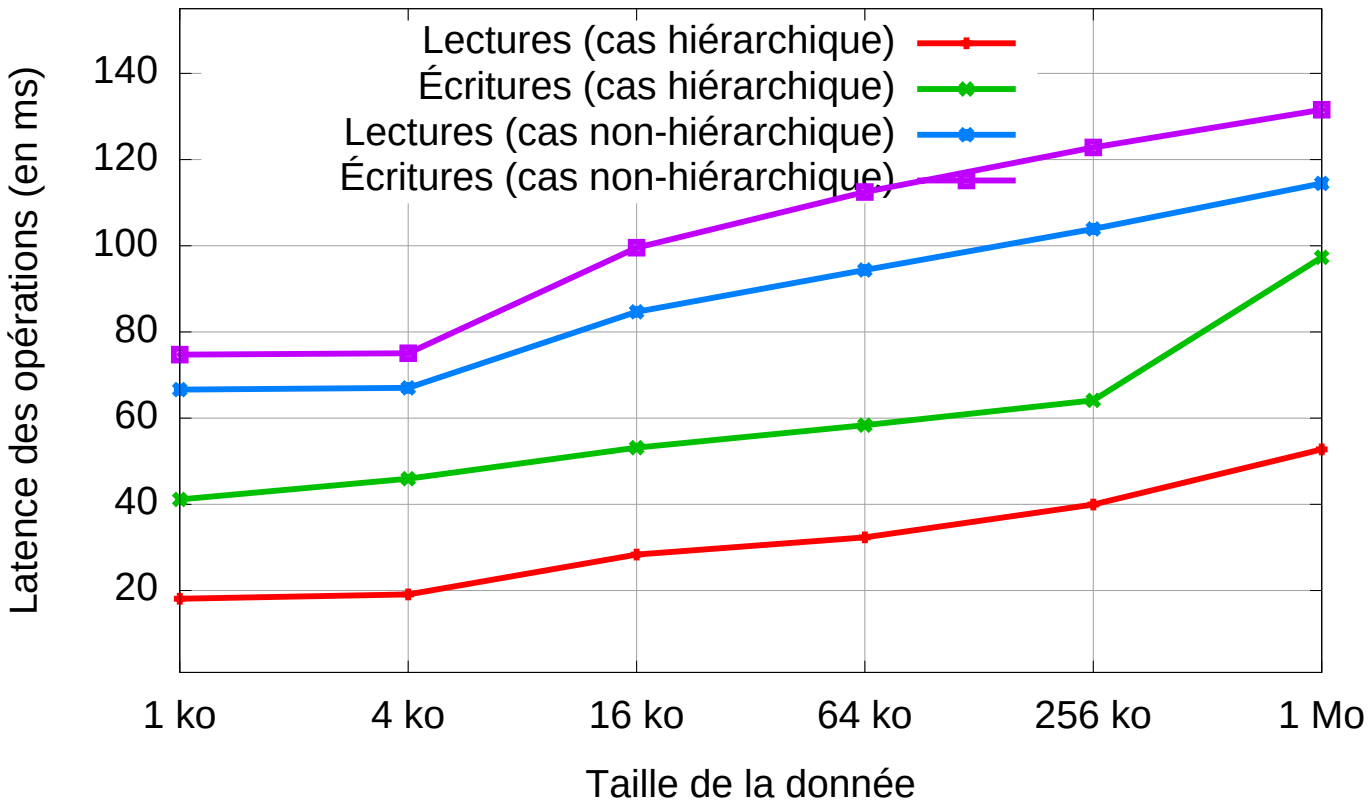
$$Ecriture = acquire + write + release$$

2 : Bénéfice de l'approche hiérarchique

Configuration des expérimentations



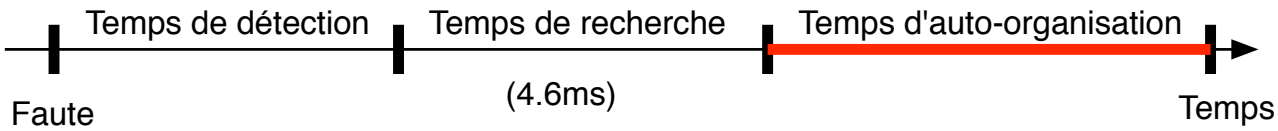
2 : Bénéfice de l'approche hiérarchique

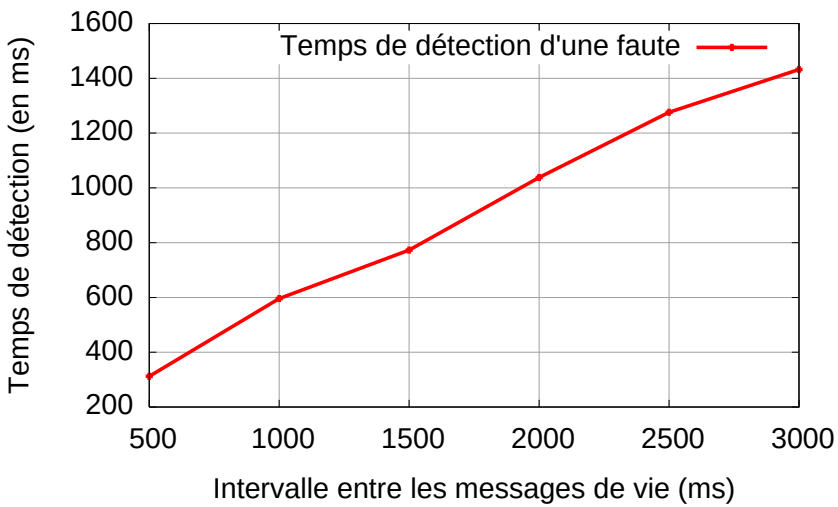


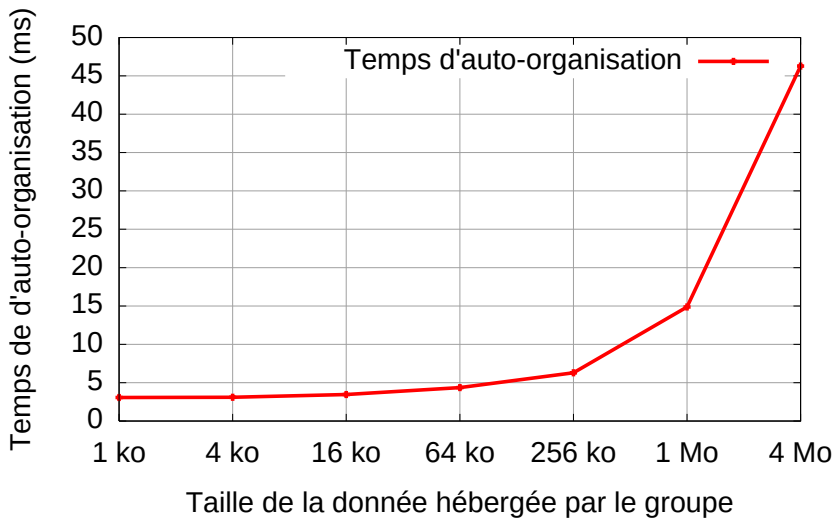
Méthodologie
Injection de fautes

- Expérimentations sans fautes
  - Coût de la tolérance aux fautes ?
- Expérimentations avec fautes
  - Résistance aux fautes ?
  - Impact des fautes sur les performances ?
- Outil d'injection de fautes
  - Besoin de reproductibilité, de précision et de passage à l'échelle
  - Génération d'un calendrier de fautes
  - Intégré à un outil de déploiement (JDF)

3 : Impact des fautes sur les performances







Contribution : architecture découplée générique

Gestion conjointe

- Protocoles de cohérence de données
- Mécanismes de tolérance aux fautes
- Architecture logicielle multi-protocole

Mise en œuvre et évaluation expérimentale

- JuxMem-c 13 500 lignes de code
  - dont 6 100 pour la cohérence des données et la tolérance aux fautes
- JuxMem-java 16 700 lignes de code
  - dont 7 300 pour la cohérence des données et la tolérance aux fautes
- Expérimentations multi-sites sur Grid'5000
- Publications : CCGrid 2006, HPDGrid 2006

Contribution : une approche hiérarchique
pour la tolérance aux fautes et la cohérence des données

Intégration dans une architecture hiérarchique commune

- Dans le cadre du projet GDS de l'ACI Masse de Données
  - Service de partage de données pour la grille

Mise en œuvre et évaluation expérimentale

- Intégration de GFD [LIP6]
- Expérimentations multi-sites sur Grid'5000
- Publications
  - WCGC 2006
  - le journal *international Concurrency and Computation : Practice and Experience* (2006)
  - le chapitre de livre *Future Generation Grids* (*CoreGRID series* 2006)

Contribution annexe

Réseau logique malléable

- Cadre : Collaboration avec l'UIUC
  University of Illinois at Urbana-Champaign
- Réseau logique pair-à-pair
- But : offrir des communications de groupe efficaces

Mise en œuvre et évaluation

- Conception d'un simutateur à évènements discrets
- Publications : SRDS 2006

Perspectives

- Données fragmentées
  - Découpage des données en blocs
  - Blocs répartis sur un ensemble de fournisseurs
- Support d'autres types d'applications
  - Application collaboratives
  - Projet RESPIRE de l'ANR MDMSA débuté en 2006
- Intégration à une plate-forme pour applications de fouille de données sur la grille
  - Collaboration avec l'université de Calabre débutée en 2006

Perspectives

- Mécanismes adaptatifs
  - Protocoles de cohérence
  - Mécanismes de tolérance aux fautes
- Fautes des clients
  - Applications tolérantes aux fautes
  - Couplage points de reprise/réplication