Un outil de recherche en système : application à la gestion des ressources

Bernard Maillot

THESE

présentée à

Institut National Polytechnique de Grenoble

pour obtenir le grade de

DOCTEUR ingénieur et 3ème cycle

Spécialité : GENIE INFORMATIQUE

par

Bernard MAILLOT

UN OUTIL DE RECHERCHE EN SYSTEMES INFORMATIQUES

APPLICATION A LA REPARTITION DE RESSOURCES

Thèse soutenue le 11 décembre 1978 devant la Commission d'Examen

Président	S. KRAKOWIAK
Examinateurs	R. BALTER
	J. BRIAT
	C. KAISER
	J. MOSSIERE
	J.P. VERJUS

P L A N

1. INTRODUCTION

Le but du projet OURS est la définition sur IRIS 80 d'un outil pour la construction et l'expérimentation de systèmes informatiques. L'outil de construction que nous avons développé s'appuie sur la décomposition d'ur système en niveaux de machines abstraites et sur la communication par messages.

Il se compose d'une machine abstraite de base qui correspond à l'IRIS 80 en mode C esclave (nous l'appelons "machine IRIS") et de mécanismes d'extension matérialisés par les trois opérations primitives suivantes :

- CREER-PROCESSUS (machine abstraite,
  programme).

Un nouveau processus est créé, correspondant à l'exécution du programme par la machine abstraite

- CREER-UNITE (machine abstraite,
  programme,
  degré de multiplication,
  nombre d'entrées,
  nombre de sorties).

Une nouvelle unité est créée par regroupement de plusieurs processus sur un même programme et une même machine abstraite.
Une unité est une boîte noire définie fonctionnellement par ses entrées et ses sorties.

- CREER-MACHINE (constituants,
  connexions,
  entrées,
  sorties).

Une nouvelle machine est créée par assemblage d'unités et/ou machines déj existantes. Les connexions entre les entrées et les sorties de ces différ constituants sont définies par un automate de transition.

L'exécution d'un processus par une machine abstraite se traduit enfin par le cheminement d'un message entre les constituants de cette machine, ce cheminement étant guidé par l'automate de transition de la machine.

Un noyau de système a été réalisé au moyen de cet outil : c'est la machir MAS (MAchine Système). Elle contient d'une part des unités construites sur la machine de base (machine IRIS) et d'autre part la machine de base elle-même.

L'outil de construction de système que nous venons d'esquisser, ainsi que la machine MAS, sont présentés en détail dans l'ouvrage principal de cette thèse [MAI].

Dans cet ouvrage, l'accent est mis sur la décomposition fonctionnelle en tant que méthode de réalisation d'un système informatique et nous y développons l'idée que l'architecture d'un système peut être définie dans les mêmes termes que celle d'une machine.

Nous abordons ici les problèmes de répartition de ressources et notre propos est de montrer que l'outil de construction de systèmes que nous avons défini est en même temps un outil pour la répartition de ressources

Le présent exposé comporte deux parties :

- dans la première partie, nous essayons de situer la notion de ressource par rapport aux notions d'unités et de machines définies dans notre modèle de décomposition. Ce faisant, nous montrons de quelle façon ces unités et ces machines peuvent être utilisées pour la mise en oeuvre de politiques décentralisées de répartition de ressources ;

- la seconde partie est consacrée à la machine MAS et à la répartiti des ressources physiques. Nous décrivons et nous discutons brièvement les stratégies choisies. Nous développons plus précisément les mécanismes qui permettent de la mettre en oeuvre.

2. OUTILS DE REPARTITION FOURNIS PAR LE MODELE

2.1. La notion de ressource

L'architecture d'une application étant constituée de machines, d'unités et de processus, il paraît intéressant de considérer que les ressources consommables par un élément quelconque de cette architecture sont les constituants de la machine sur laquelle cet élément est construit.

Deux schémas d'allocation sont alors envisageables :

a) La prise en compte d'un message par une unité représente l'alloc d'une ressource au processus qui est associé au message, et le renvoi du message représente la libération de la ressource. Sur une machine réelle par exemple, le chargement du contexte d'un processus sur l'unité centrale représente l'allocation de cette unité centrale au processus.

b) La ressource doit rester allouée au processus pendant plusieurs appels successifs. Des opérations d'allocation et de libération sont alo définies en plus des opérations d'accès par l'unité qui gère la ressourc Les unités périphériques sont gérées par exemple de cette façon dans la machine MAS : le schéma standard d'utilisation d'un périphérique est en effet le suivant (cf. [MAI] § IV.3.3)

- ALLOUER-PERIPH (nom symbolique du périphérique désiré)
{- TRANSFERER (magnum du périphérique, paramètres)}*
- LIBERER-PERIPH (magnum du périphérique).

L'allocation d'une ressource se traduit alors par le RENVOI d'un message sur une patte de sortie particulière de l'unité qui gère cette ressource.

Quel que soit le schéma d'allocation utilisé, il n'appartient pas aux unités qui gèrent ou qui représentent des ressources de contrôler cette allocation, c'est-à-dire de choisir parmi les demandes lesquelles elles vont prendre en compte ; elles n'en ont d'ailleurs pas les moyens. La seule politique qu'elles puissent appliquer dans ce domaine est celle dit du "premier arrivé, premier servi".

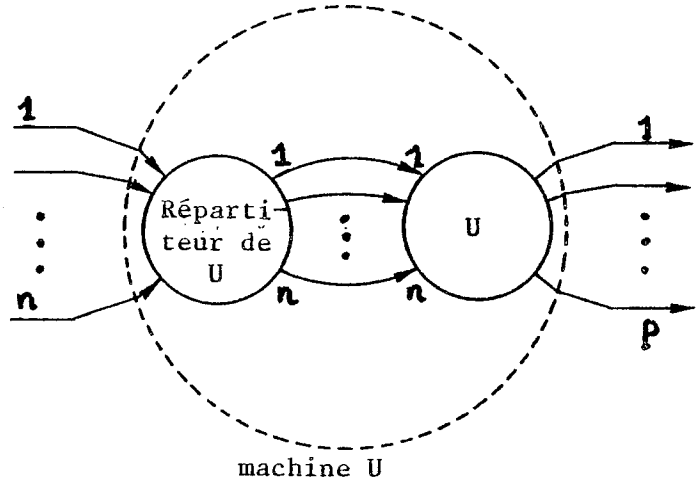
Si l'on veut pouvoir contrôler les avancements relatifs des processus dans une architecture, il apparaît donc nécessaire de fournir des outils permettant de définir pour chaque ressource une politique de répartition de cette ressource entre les demandeurs qui se présentent.

Nous allons voir que la communication par messages et le modèle de décomposition que nous avons présentés fournissent potentiellement ces outils.

2.2. Les répartiteurs de ressources

L'exécution d'un processus par une machine abstraite se traduit par le cheminement d'un message entre les unités de cette machine abstraite. Il est donc facile de contrôler les allocations et libérations de ressou en filtrant les messages avant qu'ils parviennent aux unités.

La solution la plus triviale consiste alors à remplacer chaque unité par une machine constituée de cette unité et d'une unité que nous appelor "répartiteur" et dont le rôle est de filtrer toutes les demandes.



machine U

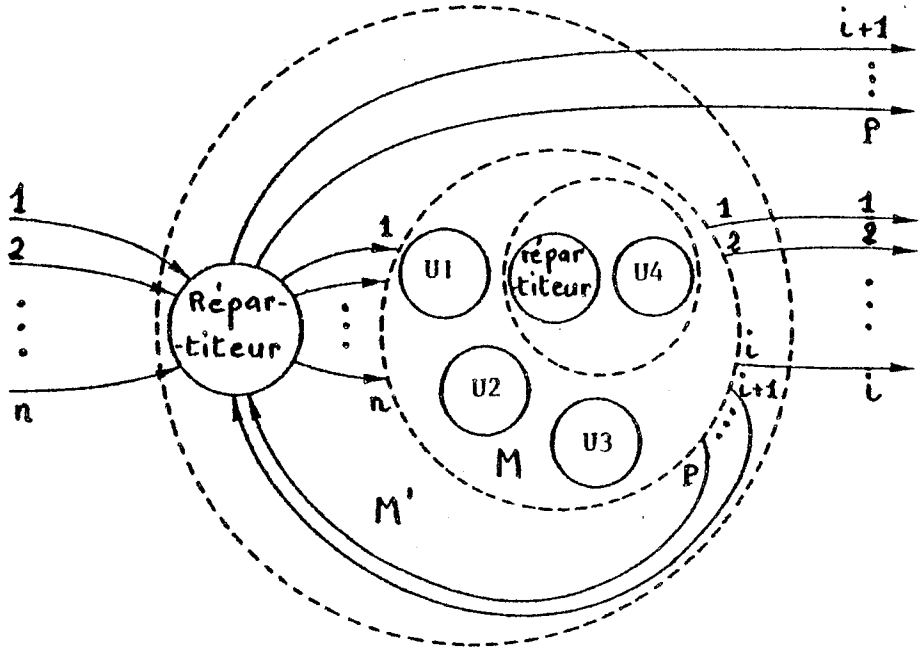
Le répartiteur de U est l'unité d'entrée de la machine U ; les pattes d'entrée et de sortie de la machine U sont les mêmes que celles de l'unité U.

Rien n'empêche que plusieurs unités possèdent le même répartiteur, si par exemple elles collaborent à la gestion d'une même ressource, mais on aura quand même autant de machines "réparties" que d'unités si l'on se limite à cette solution triviale.

Une solution plus économique consiste à définir une unité de répartition non pas pour chaque ressource, mais pour chaque groupe de ressources. On constate d'ailleurs que les ressources d'un système ne sont pas toujours indépendantes et que la définition d'une politique de répartition doit être faite à un niveau relativement global.

Le mécanisme que nous proposons se résume en définitive à associer un "répartiteur" à chaque machine abstraite et à placer ce répartiteur aux "endroits-clés" dans l'automate de transition de la machine.

L'utilisation de ce mécanisme conduit à la définition de machines répart ainsi constituées :



machine M' = (M répartie)

Le répartiteur doit être l'unité d'entrée de la machine M' et la machine ne doit jamais apparaître seule dans l'architecture, mais toujours englobée dans M', faute de quoi des processus pourraient s'exécuter sur M en échappant à son répartiteur.

Le répartiteur peut en même temps être intégré à la machine abstraite M selon le schéma proposé au début. Cela correspond au fait que la politiqu de répartition même si elle est définie globalement au niveau d'une machi doit pouvoir être mise en oeuvre plus finement au niveau de certains composants.

Dans la pratique, le répartiteur sera plutôt un module de répartition qui pourra regrouper plusieurs unités : nous verrons que dans la machine MAS, ce module de répartition regroupe les unités DISTRIBUTEUR et CONTROLEUR.

Comme complément au mécanisme que nous venons de suggérer, citons également deux primitives permettant à un répartiteur de connaître d'une part l'avancement d'un processus dans une machine et d'autre part de connaître la charge d'un élément quelconque de cette machine.

LIRE-ETAT (id-processus) → pile d'exécution du processus.

Cette primitive est également utilisée par l'unité opérateur d'une machi

LIRE-ACTIVITE ({id-machine / id-unité})

Cette primitive fournit en retour les renseignements suivants :
pour une unité : degré de multiplication
nombre de processus actifs
nombre de processus en attente
pour une machine : nombre d'unités et de processus supportés
nombre de processus actifs (en cours de traitement pa
un élément de la machine).

2.3. Quelques stratégies de répartition

Il appartient au constructeur d'une machine "répartie" de définir les tâches du répartiteur de sa machine et de le programmer en conséquence.

Nous donnons ici à titre d'exemple quelques stratégies applicables au niveau d'un répartiteur.

a) La transmission d'un message est différée ou refusée lorsque l'unité destinataire est surchargée, la surcharge étant mesurée par le nombre de messages que l'unité n'a pas encore renvoyés. Cette stratégie est utilisée dans la machine MAS pour réguler la charge de l'unité de pagination. La surcharge provient ici du fait que trop de processus se partagent la mémoire ; la régulation permet alors d'éviter le phénomène d'écroulement.

b) Des priorités sont affectées aux processus supportés par une machine abstraite donnée.
Les processus associés aux unités de MAS sont par exemple prioritaires sur tous les autres pour la ressource unité centrale.

c) Avant de transmettre un message, le répartiteur y insère des comm des ou des paramètres pour l'unité destinataire. Prenons l'exemple d'une unité capable d'abandonner un traitement en cas de dépassement d'une limite de consommation (nombre de lignes imprimées, durée d'exécution, .. passée en paramètre dans le message. Le répartiteur peut fixer la valeur de ces paramètres ou bien délivrer la valeur demandée par tranches successives pour effectuer un multiplexage de l'unité.

d) Le répartiteur collecte des statistiques sur l'emploi des ressources ou sur les flux des messages.

e) Dans le cas où le multiplexage des unités entraîne un risque d'interblocage, le répartiteur exécute un algorithme de prévention. La définition d'un outil centralisé de synchronisation tel que celui décrit dans ([MAI] § III.1.22) peut faciliter grandement la prévention des interblocages.

Les attributions des répartiteurs sont donc potentiellement très variées Elles dépendent essentiellement des applications supportées.

2.4. Les niveaux de répartition

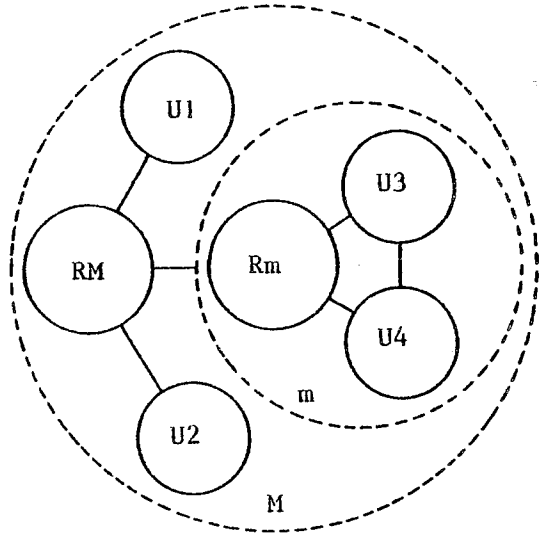
2.4.1. Répartition décentralisée

Le mécanisme que nous avons suggéré permet de décentraliser la fonction de répartition de ressources puisqu'un répartiteur différent peut être défini pour chaque machine abstraite.

Une politique de répartition peut donc être définie localement pour chaque machine abstraite. Cette possibilité est voisine, à première vue, de celle qui est offerte dans le projet HYDRA par les "Policy Modules" [LEV], mais en réalité nos différents répartiteurs ne permettent pas, contrairement aux "policy modules", une véritable décentralisation de la répartition. En effet, s'il est vrai que l'utilisateur peut définir sa propre politique de répartition au dessus de celle qui est définie pour les ressources de base, il est vrai également que la politique mise en oeuvre pour ces ressources de base est fixée entièrement par le répartiteur de la machine MAS et ne saurait être modifiée par les utilisateurs.

En résumé, il n'y a décentralisation que dans la mesure où les ressources sont à des niveaux différents du niveau de base.

Cette remarque ne vaut d'ailleurs pas seulement pour les ressources de base, elle reste valable lorsqu'il y a inclusion d'une machine dans une autre.



RM : Répartiteur de la machine M

Rm : Répartiteur de la machine m.

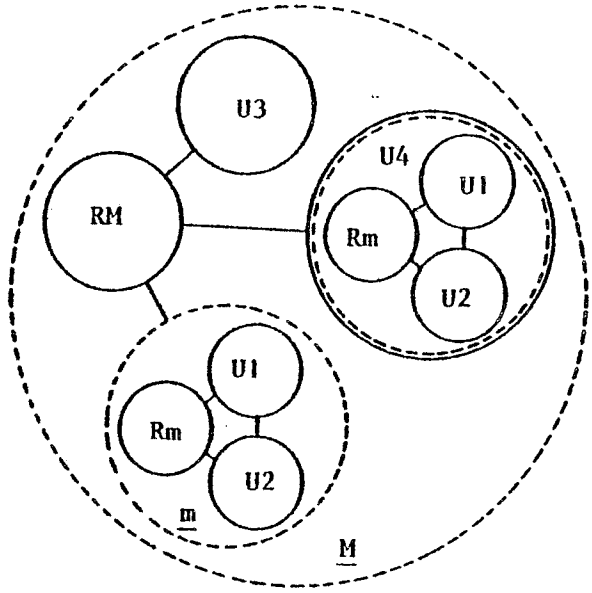
La machine M est composée des unités RM, U1 et U2 et de la machine m. Si l'on suppose que les ressources de la machine m sont réparties par Rm entre tous les processus qui se présentent, alors la politique de répartition fixée par Rm ne saurait être influencée par RM. Cependant, l'ensemble des ressources de m constitue une ressource dans la machine M et une politique de répartition de cette ressource peut être mise en oeuvre par RM.

Il est possible, cependant, que les machines incluses et la machine englobante possèdent le même répartiteur. C'est le cas notamment lorsque ces machines sont créées pour les besoins d'une même application. La machine MAS et la machine IRIS en sont un exemple.

2.4.2. Interprétation et répartition

L'architecture d'une application peut être composée de plusieurs niveaux de machines abstraites et mettre donc en oeuvre des niveaux d'interpréta-tion. On peut alors se demander si les mécanismes de répartition que nous proposons permettent de refléter la hiérarchie des niveaux d'interprétat:

Prenons un exemple pour fixer les idées :



U4 construite su

La machine m est composée des unités U1 et U2 et du répartiteur Rm. La machine M est composée des unités U3 et U4, du répartiteur RM et de la machine m. L'unité U4 est elle-même construite sur m.

Le problème que nous soulevons ici est le suivant :
Les processus exécutés par m sont pris en charge par Rm du point de vue de la répartition de ressource. Supposons que cette répartition se fasse dans m sur la base d'un mécanisme de priorités.

Parmi les processus exécutés par m, il y a d'une part ceux qui sont associés à l'unité U4 et d'autre part ceux qui viennent de la machine M.

La priorité des premiers est normalement fixée à la création de l'unité U
La priorité des autres peut être redéfinie à chaque transition vers la machine m.

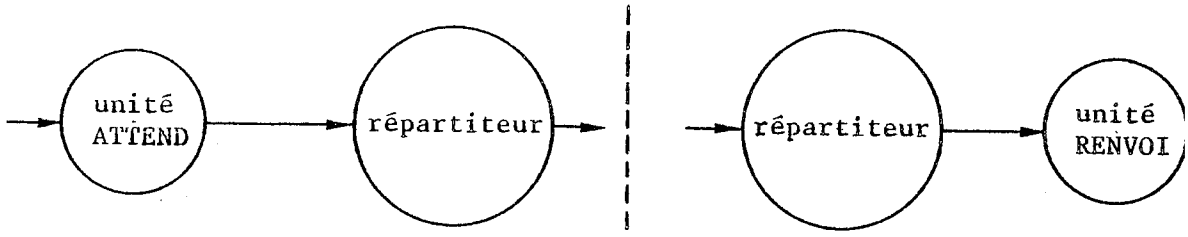
Peut-on alors faire en sorte qu'un processus de M qui est prioritaire pour les ressources U1 et U2 conserve la même priorité pour ces ressources quand il est traité par l'unité U4 ?

Nous proposons ici trois façons de résoudre ce type de problème :

a) La première consiste à réserver systématiquement les priorités maxima aux processus serveurs lors de la création des unités. C'est la solution que nous avons adoptée pour les unités de MAS en ce qui concerne la ressource unité centrale.
Cette solution est assez grossière et pénalise d'emblée le libre-service. Elle présente néanmoins l'avantage de la simplicité et elle est assez bien adaptée au cas où les services sont de courte durée. Il est en

effet habituel de privilégier les traitements brefs par rapport aux traitements longs.

b) Il est tout à fait possible de faire en sorte qu'un processus serveur acquiert dynamiquement les caractéristiques de répartition de son client pour les ressources dont ils ont un besoin commun. Cette acquisition doit logiquement se faire sous le contrôle du répartiteur ; suggérons ici de réaliser ce contrôle en utilisant les unités ATTEND et RENVOI de la façon suivante :



Le répartiteur attribue au serveur les caractéristiques du client

Le répartiteur restitue au serve ses caractéristiques propres ava le RENVOI

L'utilisation de cette solution nécessite que d'une part les caractéristi de répartition des processus soient accessibles aux unités, que d'autre part un inventaire de toutes les ressources "réparties" d'une architecture puisse être établi et qu'enfin les caractéristiques de répartition de toutes ces ressources puissent s'exprimer dans les mêmes termes, ce dernier point supposant que les politiques de répartition utilisent des techniques répertoriées.

c) La troisième solution consiste à déterminer l'architecture de la machine de façon à éviter tout simplement que le type de problème évoqué puisse se poser.
Plus précisément, on remplacera la création d'unités par une gestion de contextes analogue à celle faite dans le noyau GEMAU [BRI].
Une machine sera alors constituée d'un répartiteur et d'un ensemble de machines incluses sur lesquelles le répartiteur enverra les processus clients s'exécuter avec le contexte désiré.
Cette solution est à rapprocher d'un certain nombre de projets de multi-microprocesseurs dans lesquels un contrôleur centralisé distribue le

travail à des processeurs. Dans le cas des multi-micro, cette distributi peut être coûteuse car les programmes associés aux services à effectuer doivent être transférés vers les mémoires locales des processeurs. Dans notre cas, ces transferts sont peu coûteux car la banque de segments MAS est le support de toutes ces mémoires locales (cf. [MAI] § III.1.3). On peut donc raisonnablement envisager de construire une architecture avec des machines incluses (processeurs banalisés) plutôt qu'avec des unités (processeurs spécialisés) dans le seul but de supprimer des relations d'interprétation.

Le problème que nous avons évoqué est analogue à ceux qui se posent pour la mise en place d'un service de facturation automatique (cf. [MAI] § II.4.3) et nous y avons apporté une solution analogue bien que moins développée :

- A chaque processus on associe ses caractéristiques de répartition sur la machine MAS (priorité et numéro de classe).

- Les caractéristiques de répartition MAS d'un processus client ne sont pas attribuées automatiquement au processus serveur dans l'unité ATTEND, mais les répartiteurs disposent de deux opérations qui permettent de le faire :

LIRE-QUOTA (processus) → classe, priorité
MODIF-QUOTA (processus, nouvelle classe, nouvelle priorité).

2.5. Résumé

En résumé, le modèle de décomposition que nous proposons contient des mécanismes qui permettent de définir des politiques de répartition propre à chaque application. Une politique peut être définie localement à chaque machine abstraite en termes des ressources logiques fournies par cette machine. Mais il est en même temps possible de traduire dans la répartition des ressources une relation d'interprétation mise en oeuvre par des niveaux de machines abstraites.

3. REPARTITION DES RESSOURCES DANS LA MACHINE MAS

Les mécanismes de répartition présentés au chapitre précédent ont été utilisés pour la répartition des ressources physiques unité centrale et mémoire centrale.

Afin de montrer clairement comment ces mécanismes ont été mis en place dans la machine MAS, nous commencerons par rappeler la structure de cette machine en ne conservant que les éléments qui ont une signification fonctionnelle.

Puis nous présenterons brièvement la politique de répartition choisie et les techniques utilisées pour la mettre en oeuvre.

Pour terminer nous mettrons progressivement en place les mécanismes de répartition et nous donnerons la structure finale de la machine MAS.

3.1. Structure fonctionnelle de MAS

Les ressources physiques gérées par le noyau MAS sont :

- les périphériques
- les disques MAS qui supportent la banque de segments
- la mémoire centrale et son extension (disque de pagination) qui sont accessibles aux programmes via les espaces virtuels
- les deux processeurs IRIS 80.

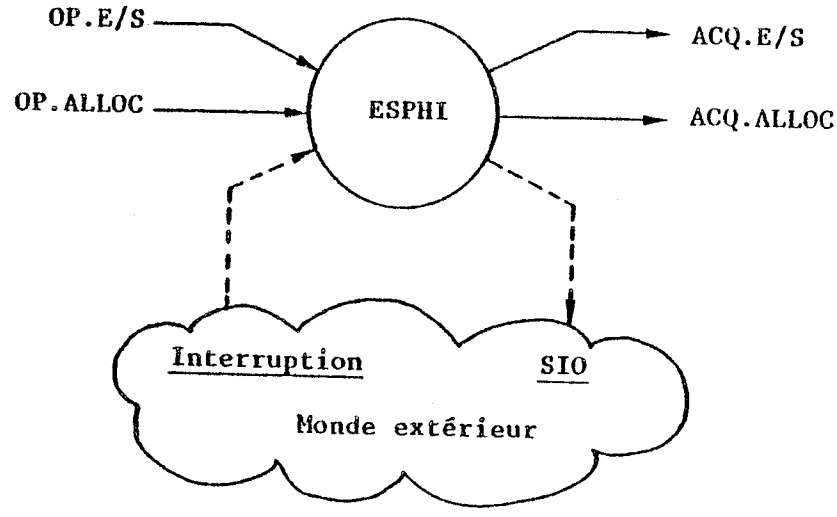
La gestion de ces ressources est effectuée par un certain nombre de programmes qui sont regroupés en cinq modules.
Chaque module contient les données qui décrivent la ou les ressources gérées, les fonctions d'accès à ces données et les contextes d'utilisatic de ces données par les processus (cf. [MAI] § III.1.23). Selon les ressources qui sont gérées, une ou plusieurs unités ont été créées par module. Par ailleurs la gestion des périphériques se décompose en deux niveaux (cf. [MAI] § IV.3) : un niveau logique pour la traduction des adresses virtuelles en adresses réelles, et un niveau physique pour l'exécution des entrées-sorties et le traitement des interruptions d'entrée-sortie. A chacun de ces deux niveaux correspondent un module et une unité.

Nous allons décrire brièvement les constituants de la machine MAS puis nous établirons les relations qui existent entre ces constituants et nous donnerons la structure de la machine construite en fonction de ces relations.

3.1.1. Les composants élémentaires de la machine MAS

Pour simplifier nous ne faisons pas figurer sur les schémas les sorties correspondant aux erreurs.

Gestion des entrées-sorties physiques : unité ESPHI



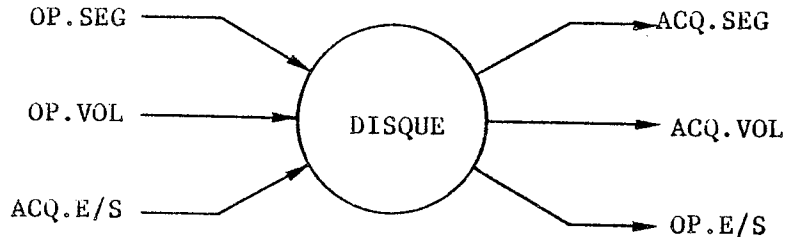
OP.E/S : demande d'entrée-sortie

OP.ALLOC : demande d'allocation ou de libération d'un périphérique

ACQ.E/S : acquittement d'entrée-

ACQ.ALLOC : acquittement d'une demande d'allocation de libération de péri

Gestion des disques MAS : unité DISQUE



OP.SEG : opération sur les segments MAS (création,lecture,sauvegarde, destruction)

OP.VOL : opération sur les disques MAS (volumes) (formatage,protection,banalisation)

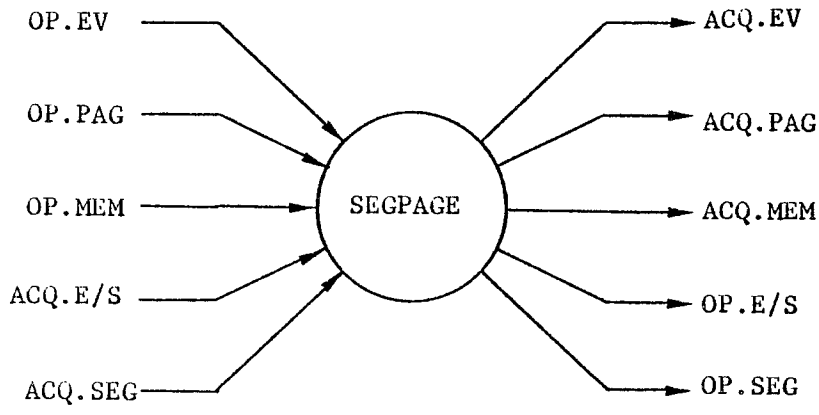
ACQ.E/S : acquittement d'entrée/sortie

ACQ.SEG : acquittement d'opérations sur les segments

ACQ.VOL : acquittement d'opérations sur les disques MAS

OP.E/S : demande d'entrée-sortie sur un disque MAS

Gestion de la mémoire réelle, de l'espace de pagination et des espaces virtuels : unité SEGPAGE



OP.EV : opérations sur les espaces virtuels (création, destruction, LIER, DELIER)

OP.PAG : faute de page, blocage et déblocage de pages

OP.MEM : demande et libération de mémoire réelle (activation et désactivation de processus)

ACQ.E/S : acquittement d'entrée-sortie

ACQ.SEG : acquittement d'opérations sur les segments

ACQ.EV : acquittement d'opération les espaces virtuels

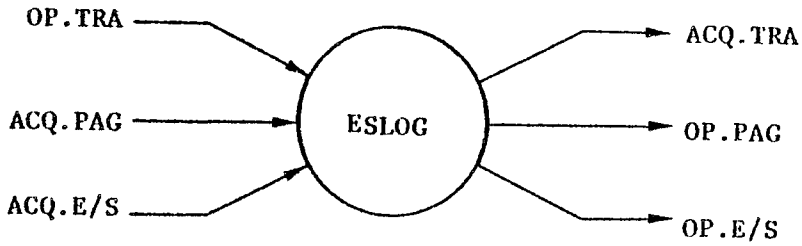
ACQ.PAG : acquittement d'opératio de pagination

ACQ.MEM : acquittement de demande sur la mémoire

OP.E/S : demande de transfert de

OP.SEG : opérations sur les segme MAS (lecture, sauvegarde destruction)

Gestion des entrées-sorties logiques : unité ESLOG



OP.TRA : opération de transfert (E/S)

ACQ.PAG : acquittements d'opération de pagination (blocage, déblocage)

ACQ.E/S : acquittement d'opérations d'entrée-sortie

ACQ.TRA : acquittement de transfe

OP.PAG : blocage et déblocage de

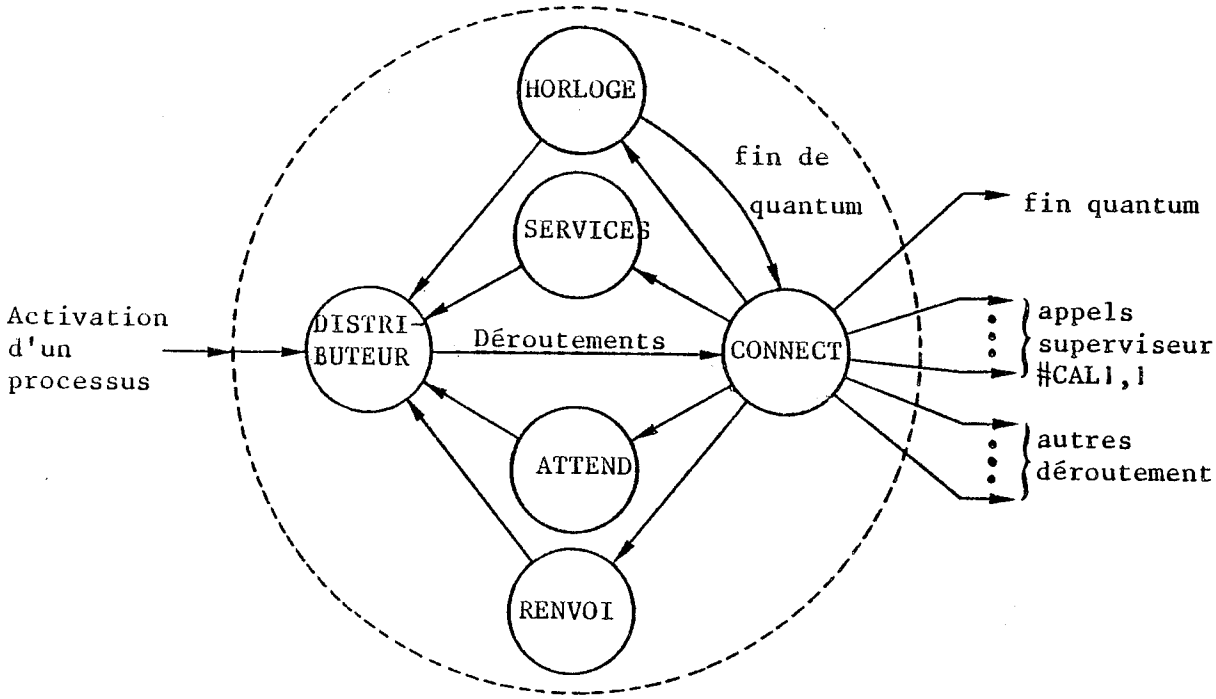
OP.E/S : opération d'entrée-sorti

Gestion des processeurs IRIS 80 : la machine IRIS

Pour des raisons historiques qui tiennent à la façon dont les tâches ont été réparties dans le projet, toutes les fonctions réalisées par le noyau de base HARDEXT (cf. [MAI] § II.3) ont été regroupées dans une machi abstraite que nous avons appelée "machine IRIS". Ces fonctions peuvent être classées en deux catégories, auxquelles correspondent les deux modules de HARDEXT :

- gestion des processeurs IRIS 80 et des contextes IRIS 80 associés aux processus
- gestion de tous les objets et mécanismes associés au modèle de décomposition.

Le premier module est constitué des unités "DISTRIBUTEUR", "SERVICES" et "HORLOGE". Le second module est constitué des unités "ATTEND", "RENVOI", "CONNECT" et "SERVICES" (cf. [MAI] § V.1).



Les demandes de services et les opérations ATTEND et RENVOI sont associées à l'appel superviseur CAL1,1.

Le distributeur alloue les processeurs IRIS 80 aux processus par tranches élémentaires de durée t.

Un processus qui demande l'unité centrale pour un temps T (quantum) doit donc tourner n fois (n = T/t) entre les unités DISTRIBUTEUR, CONNECT et HORLOGE (plus éventuellement les autres unités s'il demande des services) avant de sortir de la machine IRIS (fin de quantum).

3.1.2. Assemblage des composants élémentaires

Un certain nombre de relations de sous-traitance existent entre les composants que nous avons présentés.
Par exemple une demande de transfert arrivant sur l'unité ESLOG nécessit logiquement un blocage de page puis une demande d'entrée-sortie et enfin une demande de déblocage de page.

Afin de définir plus commodément l'automate de la machine MAS, nous isolons toutes ces relations de sous-traitance ou de coopération dans des machines abstraites intermédiaires. Chacune de ces machines intermédiair correspond alors à un sous-ensemble des services offerts par MAS, c'est-à-dire à un sous-ensemble du langage défini par la machine MAS.

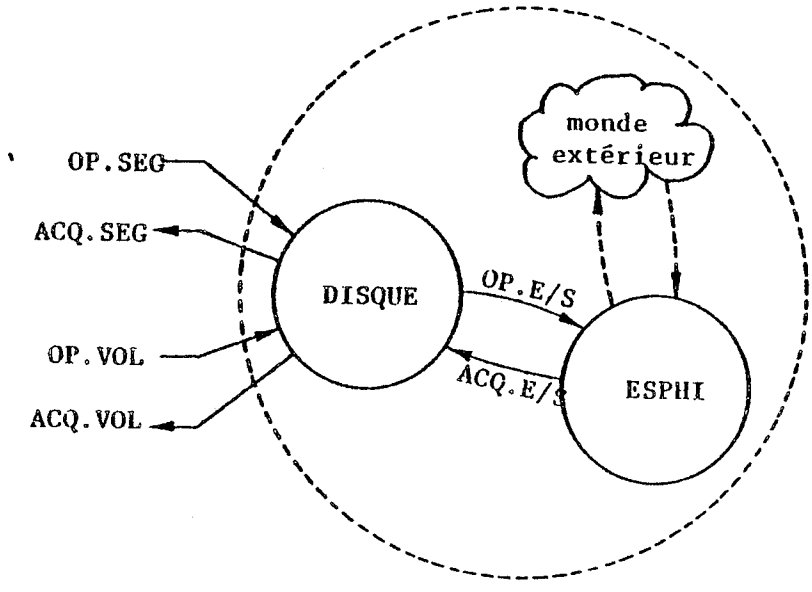
Nous obtenons en définitive les composants fonctionnels suivants :

Machine IRIS

Elle exécute les instructions IRIS 80 mode esclave C, ainsi que toutes les opérations relatives aux mécanismes de connexion et d'extension, et celles relatives à la gestion des processus.

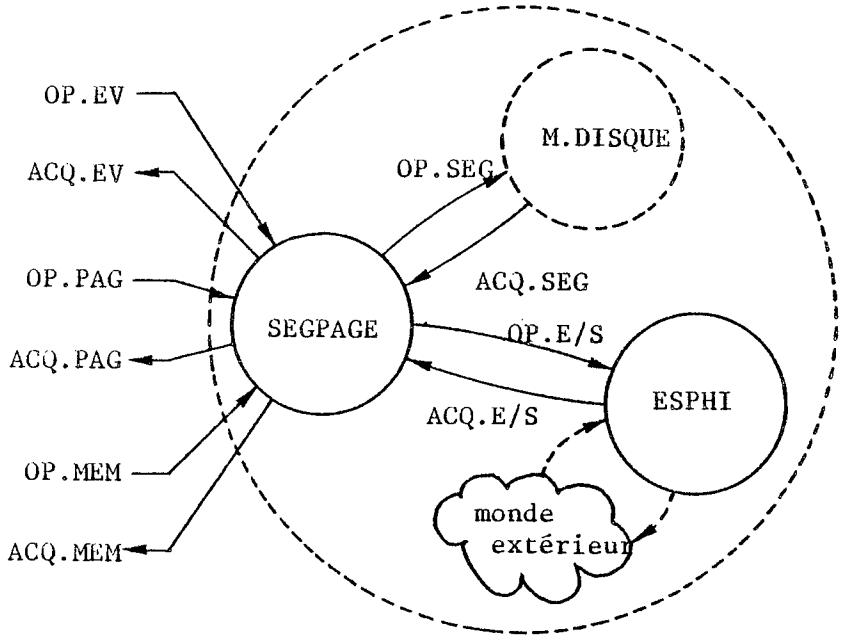
Machine M.DISQUE

Elle exécute toutes les opérations relatives à la gestion de la banque de segments :



Machine M.SEGPAGE

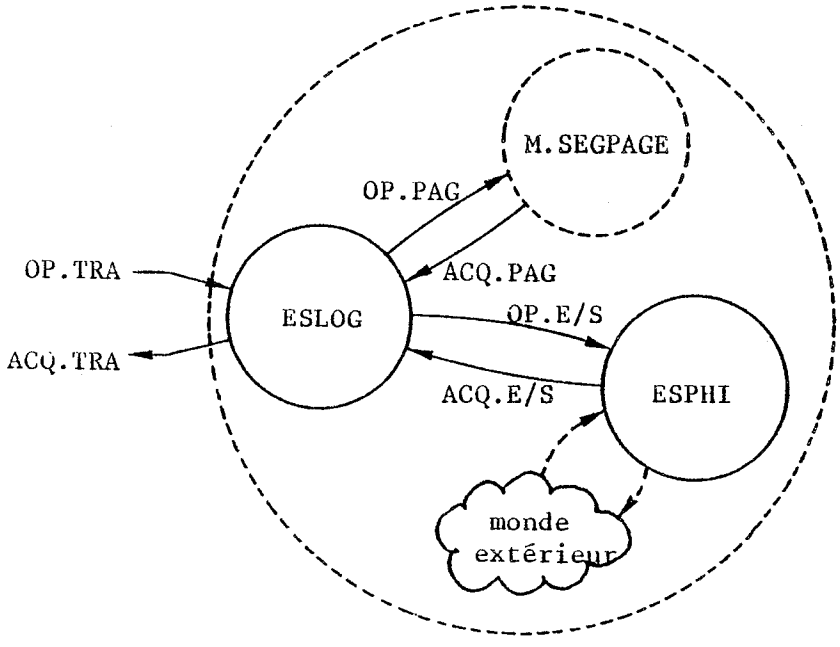
Elle exécute toutes les opérations relatives aux espaces virtuels (segmentation et pagination) et à la mémoire réelle.



M.SEGPAGE

Machine M.ESLOG

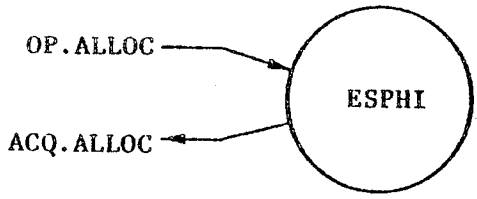
Elle exécute toutes les opérations du protocole appareil standard sur les périphériques.



M.ESLOG

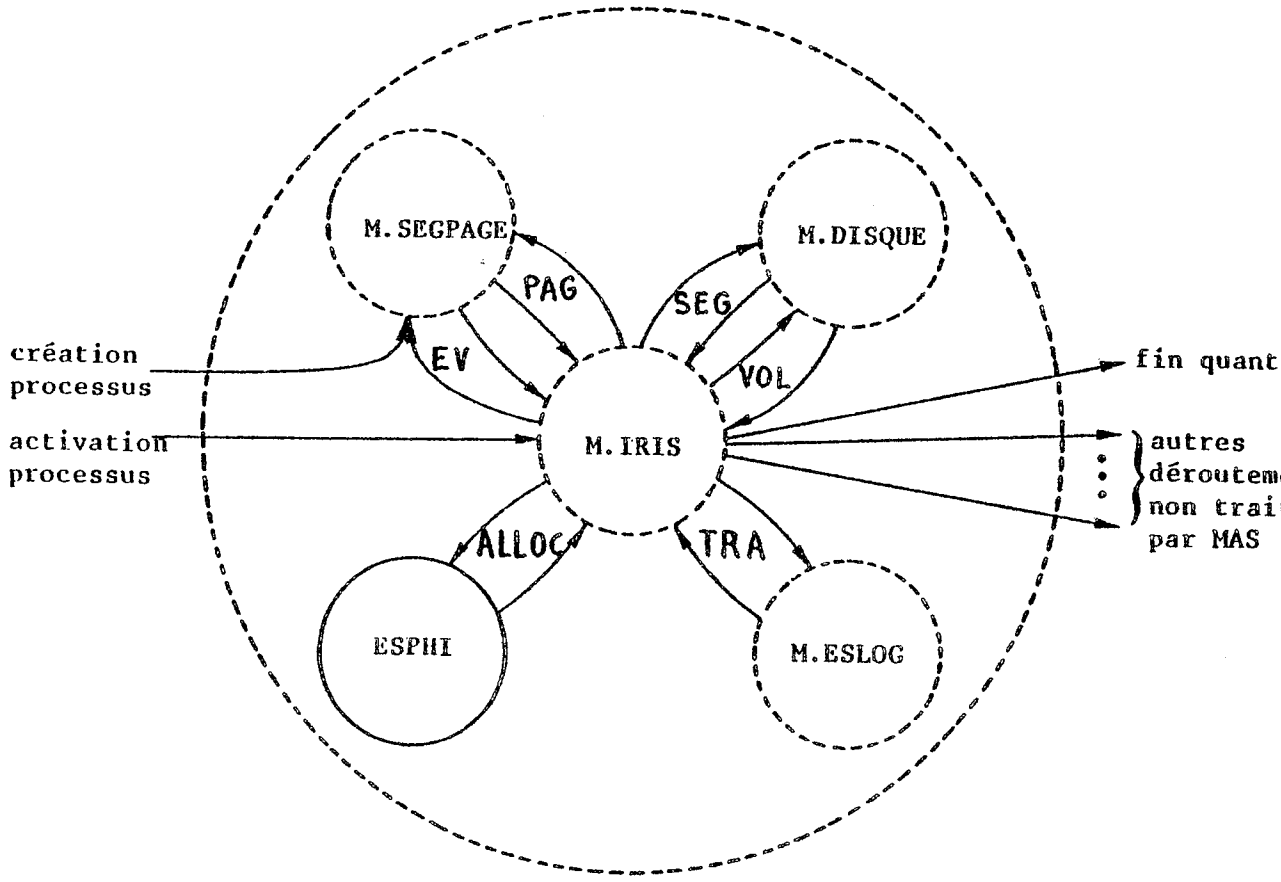
Unité ESPHI

Elle exécute les opérations d'allocation et de libération des périphériqu



3.1.3. La machine MAS

L'interprétation du langage machine MAS est effectué par les composants fonctionnels que nous avons présentés. Le séquencement est assuré par la machine IRIS (cf. [MAI] § III.1.1).



MAS

Remarque :

L'entrée initiale de la machine MAS correspond à la primitive créer-espa virtuel. Une table de segments est alors créée et initialisée par la mac SEGPAGE et son adresse est mise dans le contexte IRIS du processus grâce à l'opération MODIF-IRIS (cf. [MAI] § II.3).

La création d'un processus sur la machine MAS s'opère alors en trois temps :

- l'unité SERVICES crée un processus dans le module qui gère les processus, les unités et les machines ; un nouvel identificateur de processus est donc créé ;

- l'unité SERVICES crée un contexte IRIS dans le module associé au multiplexage des processeurs IRIS 80. Ce contexte est rempli avec des valeurs fournies en paramètres de la demande de création de processus (registres, registres de base) ;

- l'unité SERVICES dépose le reste des paramètres sous forme d'un message sur l'entrée initiale de la machine MAS. Ce message porte un en-tête identifiant le processus nouvellement créé. Il contient normalem une description d'espace virtuel.

3.2. Politiques de répartition des ressources de MAS

De façon générale une politique de répartition doit pouvoir tenir compte des contraintes caractéristiques des applications supportées. Ces contra ne sont souvent connues qu'au niveau utilisateur. Le rôle d'un noyau de système "universel" est donc de permettre la définition de plusieurs politiques, en dehors du noyau, dont chacune soit adaptée à un type donné de travaux. Ce point est l'objet des deux premières parties du présent paragraphe.

Un noyau de système a aussi un autre rôle, qui est de garantir la meilleure utilisation possible des ressources physiques dont il a la charge. On peut espérer de cette façon améliorer le débit global du systè Ce deuxième point est traité dans la troisième partie de ce paragraphe.

3.2.1. Les classes de processus

L'expérience des autres systèmes, heureusement, nous donne une idée des différents types d'applications qu'un noyau "universel" doit supporter : conversationnel, calculs scientifiques, recherche documentaire "en ligne' chaînes d'exploitation de gestion "batch", ...

De façon très schématique, une puissance "idéale" de machine correspond à chaque type d'applications en fonction de ses besoins en ressources physiques. Cette puissance idéale peut être définie de la façon suivante

- vitesse de calcul de l'unité centrale
- taille de la mémoire centrale
- vitesse d'accès aux fichiers (nombre de canaux, vitesse de transfert, temps d'accès).

Nous avons introduit la notion de classe de processus pour caractériser les différents types d'applications du point de vue de leurs besoins en ressources MAS.

Dans un premier temps nous avons défini cinq classes de processus. Chaque classe a droit à un pourcentage donné des ressources physiques.

Ainsi les classes paraissent disposer chacune d'une machine semblable à la machine réelle, mais plus lente, moins bien pourvue en mémoire et moins performante pour les entrées-sorties fichier.

Tout processus exécuté par la machine MAS doit appartenir à l'une des cinq classes que nous avons définies.

Les autres processus ne nous intéressent pas puisqu'ils ne consomment pas eux-mêmes des ressources MAS.

Les ressources que nous répartissons entre les classes sont :

- la mémoire centrale paginée
- le temps d'exécution sur les unités centrales.

Les accès aux fichiers ne sont que partiellement répartis entre les classe comme nous allons le voir maintenant.

Dans notre cas, la banque de segments MAS constitue le support des fichier et bases de données qui pourront être développés au dessus de MAS [FOR]. Cela signifie que les accès aux fichiers se traduiront par des accès en espace virtuel, lesquels pourront nécessiter des opérations de pagination.

La rapidité des accès aux fichiers est alors conditionnée par le temps d traitement des fautes de page ainsi que par leur fréquence, celle-ci éta directement liée à la taille de la mémoire centrale.

La mémoire centrale paginée est répartie entre les classes mais pas le dispositif de pagination, c'est-à-dire que les fautes de page sont trait dans l'ordre chronologique de leur arrivée sur l'unité SEGPAGE. C'est sans doute là une lacune mais elle peut être comblée facilement en utilisant la même technique de répartition que pour l'unité centrale.

La répartition de l'unité centrale entre les classes est effectuée par l'affectation de chaque tranche élémentaire à une classe. Les processus candidats à l'unité centrale dans une classe donnée se voient globalemen attribuer cette unité centrale selon une probabilité qui est définie par la classe (probabilité égale au pourcentage d'u.c. réservé à la classe).

La répartition de la mémoire entre les classes est effectuée en fonction du pourcentage du degré de multiprogrammation que l'on veut "affecter" à chaque classe. Le degré de multiprogrammation est le nombre de processi qui se partagent la mémoire à un instant donné.

Si l'on admet qu'il est possible d'estimer périodiquement le degré de multiprogrammation optimal, les pourcentages introduits ci-dessus déterm: le nombre de processus admis à consommer de la mémoire dans chacune des classes à un moment donné.

Notons que cette façon de limiter le degré de multiprogrammation n'est pa une technique de répartition de la mémoire entre les classes, mais corres plutôt au souci d'allouer "efficacement" cette mémoire. Nous reviendrons sur ce point au paragraphe suivant.

Remarque : Si une classe n'utilise pas tous ses droits, les autres classe peuvent en profiter momentanément.

Il va de soi que si cette situation devient trop fréquente, c'est que la définition des pourcentages de ressources allouées aux classes est mal faite. Ces pourcentages doivent donc pouvoir être ajustés dynamiquement par l'opérateur du système en fonction des réalités de l'exploitation.

Nous avons fixé à cinq le nombre de classes, mais ce nombre peut être augmenté. Du point de vue de l'exploitation d'un système, il peut être utile en effet d'avoir toujours quelques classes en "réserve", tout

comme on réserve certaines priorités pour des travaux d'exploitation.

3.2.2. Les priorités

Une classe regroupe un certain nombre de processus. Nous pensons qu'il es nécessaire de pouvoir départager ces processus par un mécanisme simple. Le mécanisme que nous avons choisi est celui des priorités.

Alors que nous avions réservé pour chaque classe un pourcentage différent pour chaque ressource MAS, nous définissons pour chaque processus une priorité unique pour l'ensemble des ressources MAS.

La priorité d'un processus est cependant locale à la classe dont il fait partie. Il n'est pas calculé de priorités globales : elles n'auraient aucun sens et fausseraient le partage des ressources entre les classes.

Il y a cependant une exception à cette règle : les processus qui sont associés aux unités de MAS (processus dits "primitifs") sont toujours prioritaires pour l'unité centrale. En outre, le temps qu'ils demandent n'est pas découpé en tranches élémentaires mais alloué à volonté. Ils n'appartiennent à aucune des cinq classes définies précédemment. Enfin, ils ne sont pas consommateurs de mémoire paginée car les programmes et leurs données propres sont résidents.

La mise en oeuvre des priorités à l'intérieur d'une classe est simple, comme nous allons le voir :

- la répartition de la mémoire revient à choisir les processus les plus prioritaires dans la limite du nombre fixé par le pourcentage du degré de multiprogrammation ;
- la répartition de l'unité centrale revient à choisir le processus qui attend depuis le plus longtemps parmi ceux qui ont la priorité maximum dans la classe. Si ce processus s'exécute sans déroutement pendant une tranche élémentaire, il est remis en attente (il devient alors le plus jeune dans sa priorité). Ce mécanisme permet d'écouler en priorité les processus prioritaires, tout en conservant les principes du multiplexage.

3.2.3. Utilisation efficace du matériel

Le partage des ressources matérielles ne doit pas introduire un overhead excessif.

En ce qui concerne le découpage du temps d'exécution en tranches élémentair la valeur des tranches doit être suffisamment petite pour que le "temps

de réponse" de l'unité centrale reste proportionnel à la fois au quantur demandé et à la charge globale du système. Mais cette valeur doit être également suffisamment petite pour qu'on ne passe pas son temps à faire des commutations de contexte.

On peut envisager de modifier dynamiquement la valeur de la tranche élémentaire en fonction des réalités de l'exploitation : tranche "petite aux heures d'utilisation surtout conversationnelle, tranche "grande" aux heures prévues pour le traitement par lots.

En ce qui concerne le partage de la mémoire centrale, des études nombreu [BUL, DEN3, LEN] ont montré que dans l'exécution d'un programme apparais généralement des phases de localité pour l'accès à la mémoire. Une taill optimum de mémoire, inférieure à la taille du programme, peut alors être attribuée à chaque programme. En deça de cet optimum, le taux de fautes de pages du programme croît rapidement, au-delà par contre il décroît très peu.

Une utilisation optimum de la mémoire nécessite donc de limiter le nombr de programmes en mémoire afin d'assurer à chacun son nombre optimum de pages.

Nous utilisons deux techniques complémentaires pour arriver à cet optimum

- l'allocation "fine" de la mémoire est réalisée par une pagination à la demande qui égalise les fréquences des fautes de pages relatives à chaque segment (et non pas à chaque processus) : les segments de program auront donc comparativement plus de mémoire que les segments de données (fichiers) ;

- le taux global de pagination (nombre de transferts de pages par unité de temps) est mesuré en permanence par le module qui effectue les entrées-sorties physiques (ESPHI). La comparaison de ce taux avec un tau: de référence commande l'évolution du degré de multiprogrammation (donc indirectement le nombre de segments autorisés en mémoire) : si le taux mesuré dépasse sensiblement le taux de référence, on diminue le degré de multiprogrammation ; si au contraire il lui devient nettement inférieur on augmente le degré de multiprogrammation.

Cette méthode de régulation est inspirée étroitement du modèle développé par J. LEROUDIER [LER3]. Ce modèle est basé sur le taux d'utilisation du canal fictif équivalent à l'ensemble des canaux utilisés pour la paginati le degré de multiprogrammation optimum étant supposé obtenu pour un taux voisin de 50 %.

Dans notre cas cependant, il est impossible de déterminer un tel canal fictif, notamment à cause de l'utilisation d'un même canal pour la pagination et pour d'autres entrées-sorties. Le taux de référence et la fourchette de non-intervention doivent donc être déterminés par expérimentation.

Signalons pour terminer quelques problèmes liés à l'ajustement dynamique du degré de multiprogrammation :

- choix des processus à suspendre en cas de diminution ou à activer en cas d'augmentation,
- récupération des pages libérées par un processus "suspendu",
- élévation soudaine du taux global de pagination lors du vidage des pages appartenant à un processus suspendu.

3.3. Mise en place des politiques de répartition

Les différentes stratégies et politiques de répartition des ressources de MAS que nous avons présentées sont mises en place dans la machine MAS au moyen de deux répartiteurs qui partagent le même module. Ces deux répartiteurs sont :

- le DISTRIBUTEUR, qui assure le partage de l'unité centrale entre les classes et à l'intérieur de chaque classe ;
- le CONTROLEUR qui assure le partage du degré de multiprogrammation entre les classes. Il ajuste également ce degré lorsque c'est nécessaire en activant ou en suspendant des processus.

Nous allons décrire chacune de ces deux unités ainsi que leurs relations avec les composants fonctionnels de la machine MAS.

3.3.1. Le DISTRIBUTEUR

Pour mettre en oeuvre la politique de répartition que nous avons décrite, nous avons préféré modifier le DISTRIBUTEUR décrit en § 3.1.1 plutôt que d'introduire une unité supplémentaire dans la machine IRIS, en amont du DISTRIBUTEUR existant. De toute façon d'ailleurs la machine IRIS que nous avons décrite n'est qu'une modélisation du noyau HARDEXT car tous les appels entre les unités de cette machine sont des appels procéduraux.

Le DISTRIBUTEUR alloue en tourniquet les deux unités centrales à tous les processus qui se présentent, distribuant ainsi des tranches de temps fixes d'exécution que nous appelons tranches élémentaires.

Le pourcentage d'unité centrale qui doit être alloué à chaque classe est décrit par une structure de données appelée "barillet" à laquelle il n'accède qu'en lecture.

Le barillet est une liste circulaire d'identités de classes. Un élément de cette liste spécifie la classe dans laquelle le DISTRIBUTEUR doit prendre le processus à exécuter pendant une tranche élémentaire. L'éléme suivant spécifie l'affectation de la tranche élémentaire suivante.

La file d'entrée du DISTRIBUTEUR se décompose en six groupes de files. Le premier groupe correspond aux processus qui sont associés aux unités de MAS ; chacun des cinq autres groupes correspond à une classe de répartition.

Chaque groupe se décompose lui-même en autant de files d'attente qu'il y a de priorités (nous en avons défini 16). Chacune de ces files élémentai est gérée en "Premier Arrivé, Premier Servi".

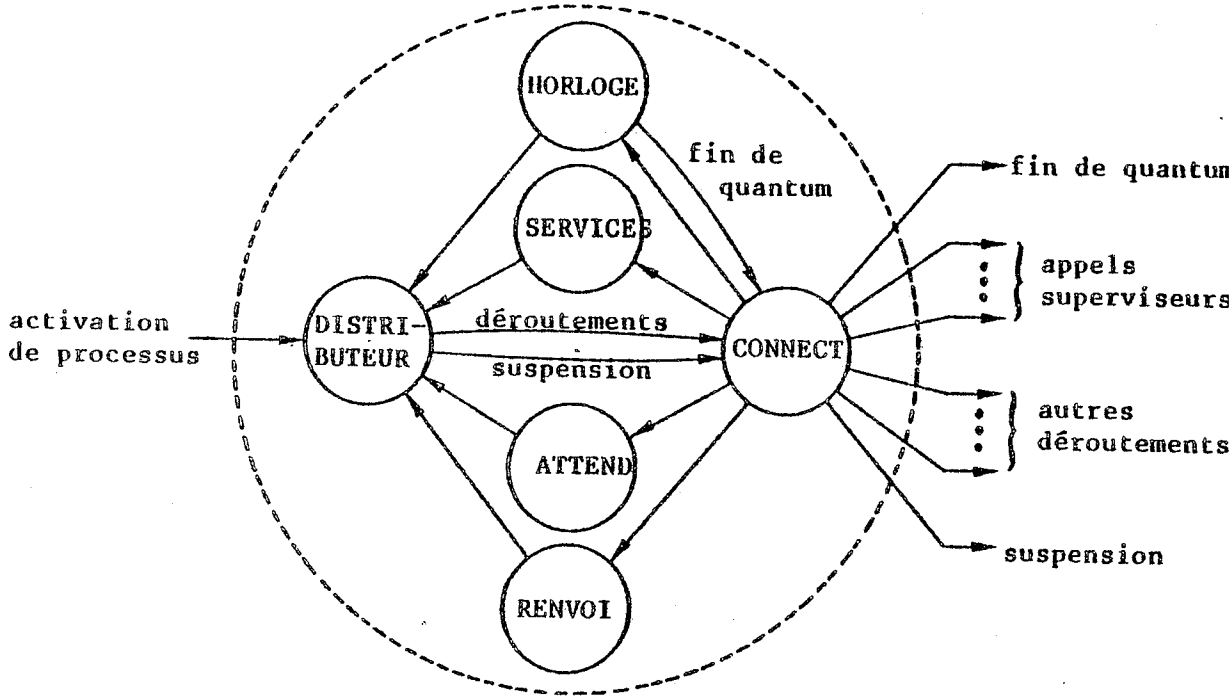
Les processus situés dans le premier groupe sont toujours servis en priorité (§ 3.22). Lorsqu'il n'y a pas de processus en attente dans ce groupe, le barillet permet donc d'élir une classe. Le DISTRIBUTEUR choisit alors dans le groupe correspondant, le premier processus de la f non vide de priorité maximum. Si ce processus s'exécute sans déroutement pendant une tranche élémentaire, il est placé en queue de sa file d'atte Si au contraire il est dérouté hors de la machine IRIS ou bien s'il effectue un ATTEND qui risque de le bloquer, le DISTRIBUTEUR alloue le reste de la tranche élémentaire à un autre processus de la même classe.

Ce mécanisme permet d'une part d'écouler en priorité les processus les plus prioritaires dans chaque classe, et d'autre part de garantir au mieux à chaque classe son pourcentage d'unité centrale.

Signalons pour terminer l'existence d'une liste supplémentaire appartenar au module de répartition et partagée donc par le DISTRIBUTEUR et par le CONTROLEUR. Cette liste est traitée par le DISTRIBUTEUR avant toutes les autres et elle ne contient que l'identité des processus que le CONTROLEUF a décidé de suspendre. Ces processus sont alors purement et simplement renvoyés de la machine IRIS sur une patte de sortie spéciale que nous appelons "suspension".

En résumé, le fonctionnement du DISTRIBUTEUR est assez simple. Toutes ses pattes d'entrée ont la même signification fonctionnelle, mais chacune est associée à une priorité et à une classe particulière.

Nous pouvons schématiser comme suit la machine IRIS obtenue (à rapprocher du schéma de la page 15). Pour simplifier nous avons regroupé sur une seu patte d'entrée toutes les pattes qui ne diffèrent que par le numéro de classe et de priorité.



3.3.2. Le CONTROLEUR

Les ressources unité centrale et mémoire ne sont pas indépendantes : lorsqu'un processus dispose de l'unité centrale, il est demandeur de mémoire. Par ailleurs, si un processus a fini d'utiliser l'unité centrale, il peut être intéressant de libérer la mémoire qu'il occupe pour la donner à d'autres processus.

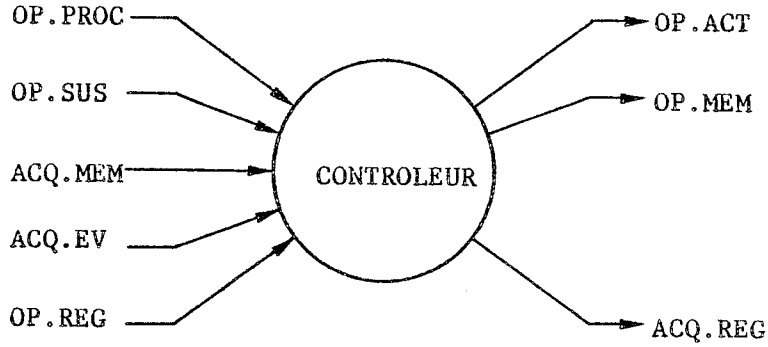
Le CONTROLEUR est chargé de filtrer toutes les demandes d'exécution sur l'IRIS 80. C'est lui qui assure le partage du degré de multiprogrammation entre les classes et c'est lui aussi qui prend la décision de modifier le degré de multiprogrammation et qui résoud les problèmes liés à cette

modification (cf. § 3.2.3).

Enfin tout processus exécuté par la machine MAS doit être associé à une des cinq classes que nous avons définies. C'est le CONTROLEUR qui est chargé de vérifier que cette règle est bien respectée.

Dans ce paragraphe, nous allons décrire brièvement la mise en oeuvre de ces différents contrôles.

a) Définition fonctionnelle du CONTROLEUR



Les fonctions exercées par le CONTROLEUR sont les suivantes :

. Les demandes d'activation arrivent sur l'entrée OP.PROC. Notons qu'elles peuvent venir directement depuis l'extérieur de MAS (OP.PROC) ou bien elles peuvent avoir transité par la machine SEGPAGE (ACQ.EV) si un espace virtuel devait être préalablement créé ou modifié.
L'activation d'un processus se traduit pour le CONTROLEUR par une demand de mémoire (OP.MEM puis ACQ.MEM). Le message est ensuite acheminé directement sur la machine IRIS (OP.ACT).

. Lorsque le CONTROLEUR décide de suspendre un processus, il met l'identité de ce processus dans une liste qu'il partage avec le DISTRIBUTEUR. Au bout de la tranche élémentaire courante, le processus est renvoyé de la machine IRIS et récupéré par le CONTROLEUR sur la patte d'entrée OP.SUS.
Le CONTROLEUR demande alors à SEGPAGE la libération de la mémoire alloué à ce processus (OP.MEM puis ACQ.MEM).

. Le CONTROLEUR est averti d'un excès du taux de pagination par un message qui lui est envoyé sur l'entrée OP.REG par l'unité ESPHI.
Le CONTROLEUR renvoie immédiatement (ACQ.REG) le message puis prend la décision de modifier ou non le degré de multiprogrammation.

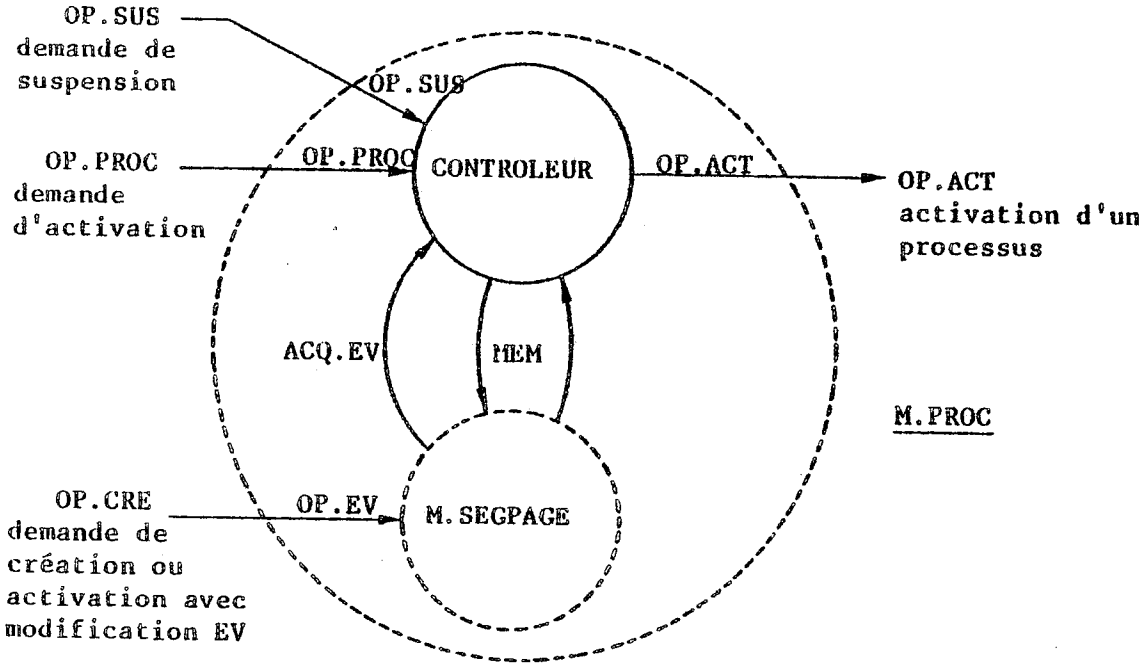
. Le CONTROLEUR peut modifier le barillet utilisé par le DISTRIBUTEL

b) Les machines de contrôle

Un certain nombre de fonctions de contrôle nécessitent la coopération entre le CONTROLEUR et les composants fonctionnels de MAS. Pour chacune de ces fonctions, nous définissons une machine intermédiaire dite de contrôle.

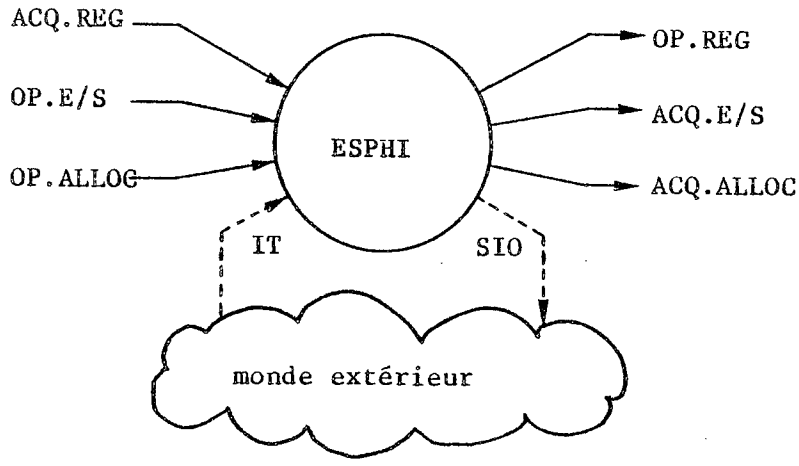
Machine de contrôle M. PROC

Elle reçoit toutes les demandes de création, d'activation et de suspensio des processus MAS.

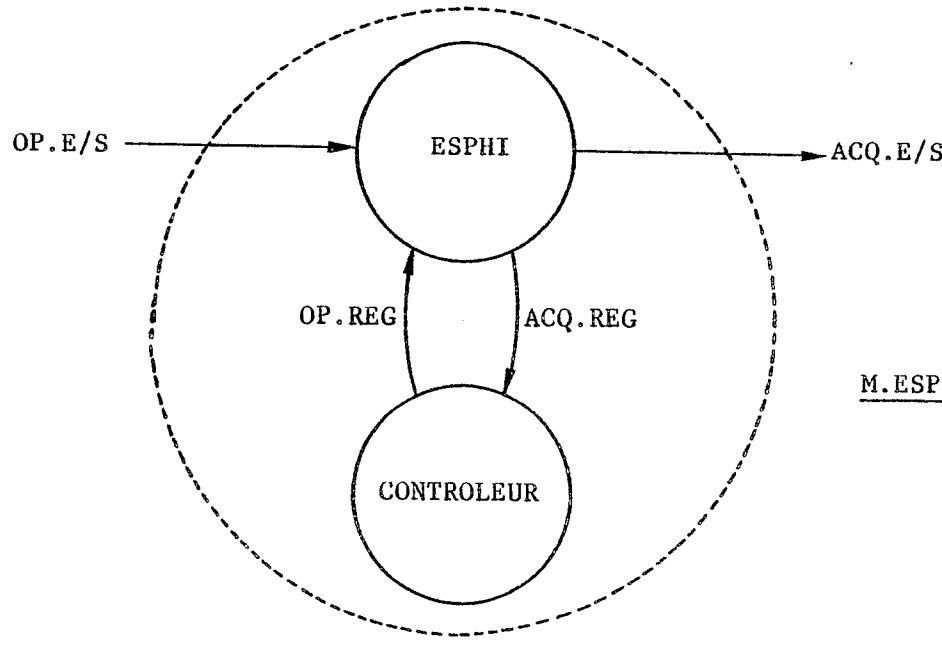


Régulation de la pagination : la machine M. ESPHI

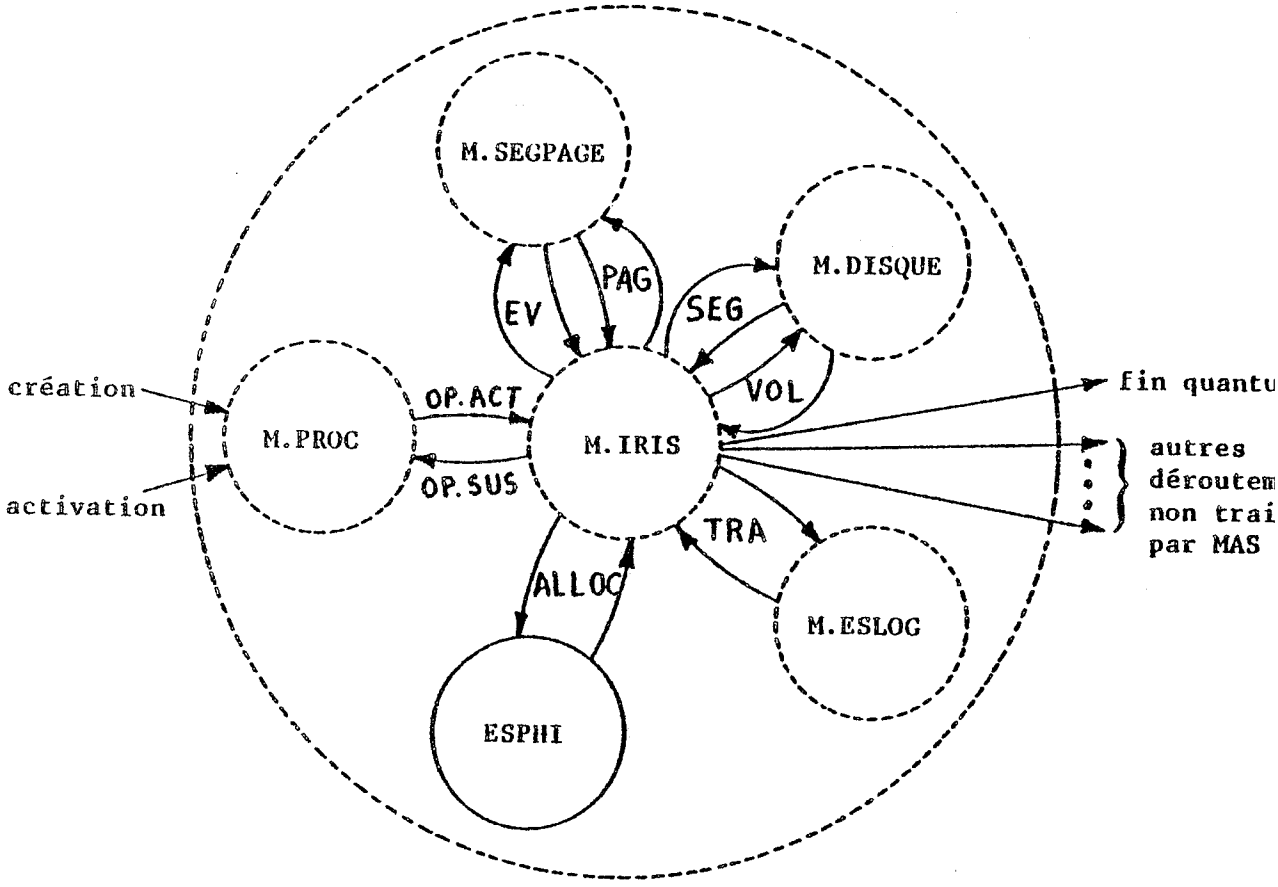
L'unité ESPHI doit avertir le CONTROLEUR dès que le taux de pagination sort d'une "plage" critique. L'unité ESPHI est donc légèrement modifiée afin de faire ce travail qui est nécessaire à la régulation.



La machine M.ESPHI est alors créée et remplace l'unité ESPHI dans les machines M.DISQUE, M.SEGPAGE et M.ESLOG. Cette substitution n'offre aucune difficulté car la machine M.ESPHI est fonctionnellement équivalen à l'unité ESPHI pour ce qui est des entrées-sorties.



3.3.3. La machine MAS "répartie"



3.3.4. Les machines MAS_i

Il reste un problème que nous n'avons pas traité : comment le CONTROLEUR peut-il vérifier ou affecter les numéros de classe aux processus qui arrivent sur MAS ?

La solution que nous suggérons consiste à définir cinq machines identiques à la machine MAS répartie et que nous appelons respectivement MAS1, MAS2, ..., MAS5.

Ces machines sont alors inscrites au catalogue général des objets offerts aux utilisateurs et la machine MAS est elle-même enlevée de ce catalogue, donc inaccessible directement.

Une machine construite par un utilisateur ne peut donc contenir que ces machines MAS_i (une ou plusieurs) et non pas MAS.

Lorsqu'un processus arrive sur une de ces machines MAS_i, c'est le CONTROLEUR qui reçoit en premier le message.
Il demande alors l'identité de la machine MAS_i pour laquelle il travaill au moyen de l'opération de l'opération primitive LIRE-ETAT (voir [MAI] § II.4.23).
L'identité de cette machine lui permet alors d'affecter le numéro de classe correspondant au processus.

Les machines MAS_i permettent de construire des applications dans lesquel un même processus utilisateur alterne différents types de comportements édition de texte, compilation, exécution interactive, mise à jour d'une base de données, ...

Pour chaque phase de travail, le répartiteur de la machine sur laquelle il a été créé peut le diriger sur telle ou telle machine MAS_i.

Les machines MAS_i sont donc un outil à la fois puissant et facile d'empl pour la mise en place de politiques de répartition des ressources physiq propres aux applications développées au-dessus de MAS.

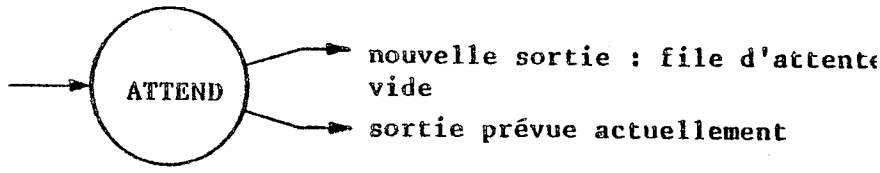
4. CONCLUSION

Le modèle de décomposition fournit des outils commodes pour la mise en place de politiques décentralisées des ressources. Ces outils sont les mêmes que ceux utilisés pour construire les applications.

Pour les ressources de base, mémoire et unité centrale, une première politique de répartition est mise en oeuvre par la machine MAS. Mais ell peut être modulée par dialogue opérateur en fonction des réalités de l'exploitation.

On ne peut que regretter que les politiques de répartition que nous avon développées ou adaptées à notre modèle n'aient pu être testées en vraie grandeur. Elles auraient permis de juger de la capacité du noyau à s'adapter à différents types de charges.

Néanmoins la mise en place de ces politiques nous a montré que les problè de répartition de ressources ne doivent pas être traités a posteriori, mais conditionnent sensiblement les choix d'architecture lorsqu'on construit une application multi-usagers. Nous citerons pour témoins une modification de notre modèle qui peut faciliter grandement la répartition des ressources physiques.



Cette modification consiste à introduire une instruction ATTEND non bloquante : la sortie "file vide" peut être alors récupérée par le CONTROLEUR, ce qui permet de libérer éventuellement les pages occupées par l'unité qui a fait ATTEND.
Nous citerons enfin une modification de la structure du superviseur d'entrées-sorties qui doit permettre d'améliorer la répartition des ressources physiques dans MAS. Cette modification consiste à éclater l'unité ESPHI en autant d'unités qu'il y a de périphériques ou de types de périphériques supportés : les demandes de transfert sur machines à écrire peuvent ainsi être facilement filtrés par le CONTROLEUR pour libére la mémoire occupée par le processus qui a demandé le transfert.

BIBLIOGRAPHIE

[ADA] J.C. ADAMS, G.E. MILLARD
Performance measurements on the Edinburgh Multiaccess System.
ICS 75 Antibes, Juin 1975.

[BEL1] L.A. BELADY, C.J. KUEHNER
Dynamic space sharing in computer systems.
CACM, Vol.12 n°5, 1969.

[BEL2] L.A. BELADY
A study of replacement algorithms for virtual storage computer.
IBM System Journal, Vol.5 n°2, 1966.

[BRI] J. BRIAT, S. GUIBOUD-RIBAUD
Espace d'adressage et espace d'exécution du système GEMAU.
IRIA 1974.

[BUL] P. BURGEVIN, J. LEROUDIER
Characteristics and models of program behavior.
ACM nat. conf., 1976.

[BUZ] J.P. BUZEN
Fundamental loase of computer system performance.
ACM Sigmetrics, Harvard 1976.

[CHU] W.W. CHU, H. OPDERBECK
Performance of the Page Fault Frequency replacement algorithm in a multiprogramming environment.
Information Processing 74 - North-Holland Publishing Company, 1974.

[COF1] E.G. COFFMAN
Analysis of drum input/output queue under scheduled operation in a paged computer system.
JACM Vol.16 n°1.

[COF2] E.G. COFFMAN, L.A. KLIMKO, RYAN
Analysis of scanning policies for reducing disk seek times.
SIAM Journal on Computing.

[DEN1] P.J. DENNING & al.
Optimal multiprogramming.
Acta Informatica, Vol.7 fasc.2.

[DEN2] P.J. DENNING
Thrashing : its causes and preventions.
Proceedings AFIPS, FJCC 1968.

[DEN3] P.J. DENNING
The Working Set model of program behavior.
CACM Vol.11 n°5, Mai 1968.

[FOR] J.M. FORESTIER
Machine relationnelle pour système segmenté.
Thèse 3ème Cycle, INPG, Septembre 1978.

[FRA] H. FRANK
Analysis and optimization of disk storage devices for time-sharing systems.
JACM Vol.16 n°4, Octobre 1969.

[FRAN] N. DE FRANCESCO, G. VAGLINI, M. VANNESCHI
On deadlocks in networks of asynchronous microprocessors.
Proc. EUROMICRO Symp., Venise, Octobre 1976.

[KAI] C. KAISER
Conception et réalisation de systèmes à accès multiple : gestion du parallélisme.
Thèse d'Etat, IRIA, 1973.

[KER] F. KERANGUEVEN
Conception et réalisation du noyau du système SAR : gestion du parallélisme.
Thèse 3ème Cycle, Rennes, Juillet 1974.

[KRA] S. KRAKOWIAK
Conception et réalisation de systèmes à accès multiple : allocation de ressources.
Thèse d'Etat, IRIA, 1973.

[LAF] P. LAFORGUE
La répartition des ressources.
OURS 1005, Novembre 1977.

[LEN] J. LENFANT
Comportement des programmes dans leur espace d'adresse.
Thèse d'Etat, Rennes, Décembre 1974.

[LER1] J. LEROUDIER, D. POTIER
Principles of optimality for multiprogramming.
ACM Sigmetrics, Harvard, 1976.

[LER2] J. LEROUDIER, D. POTIER, M. BADEL
Un modèle d'analyse des performances d'ordinateurs multiprogramm
à mémoire virtuelle.
IRIA - LABORIA.

[LER3] J. LEROUDIER
Systèmes adaptatifs à mémoire virtuelle.
Thèse d'Etat, Grenoble, 1977.

[LER4] J. LEROUDIER, D. POTIER
A two level control scheme for multiprogrammed virtual memory computer systems.
Rapport IRIA n° 141, Novembre 1975.

[LEV] R. LEVIN & al.
Policy / mechanism separation in HYDRA.
Proc. 5th symp. on OS principles, Austin, Novembre 1975.

[MAI] B. MAILLOT, A. TARABOUT, I. VATTON
Un Outil de Recherche en Systèmes Informatiques.
Thèse, INPG Grenoble, Décembre 1978.

[MER] D. MERLE, D. POTIER, M. VERAN
Un outil d'analyse des performances des systèmes informatiques.
Congrès AFCET 1978.

[MOR1] J.E. MORRISON
User program performance in virtual storage systems.
IBM system journal n° 3, 1973.

[MOR2] J.B. MORRIS
Demand paging through utilization of Working Sets on the MANIAC
CACM Vol.15 n°10, Octobre 1972.

[POT] D. POTIER
Modèles à files d'attente et gestion de ressources dans les systèmes informatiques.
Thèse d'Etat, INPG, 1977.

[STA] C. STANCESCU
Principes et propositions pratiques pour la réalisation d'un répartiteur de ressources.
Note interne. Equipe Technologie des Systèmes ENSIMAG.
Grenoble, Juillet 1976.

[TAR1] A. TARABOUT
ESLOGIQ et ESPHYSIQ. Réalisation des entrées/sorties.
OURS 2003, Juin 1976.

[TAR2] A. TARABOUT
Gestion optimale des entrées/sorties sur disques et tambours.
OURS 2002, Mai 1976.

[TEO1] T.J. TEOREY, T.B. PINKERTON
A comparative analysis of disk scheduling policies.
CACM Vol.15 n°3, Mars 1972.

[TEO2] T.J. TEOREY
Properties of disk scheduling policies in multiprogrammed computer systems.
Proc. AFIPS, FJCC, 1972.

[VAT] I. VATTON
Segmentation - Pagination.
OURS 2000, Novembre 1976.