RAIRO. RECHERCHE OPÉRATIONNELLE

PIERRE HOUEIX
MONIQUE BECKER

Modélisation d'une architecture parallèle pour le traitement d'images

RAIRO. Recherche opérationnelle, tome 22, n° 1 (1988), p. 45-65
<http://www.numdam.org/item?id=RO_1988__22_1_45_0>

Recherche opérationnelle/Operations Research
(vol. 22, n° 1, 1988, p. 45 à 65)

MODÉLISATION D'UNE ARCHITECTURE PARALLÈLE POUR LE TRAITEMENT D'IMAGES (*)

par Pierre HOUEIX (1) et Monique BECKER (2)

Résumé. — *Une architecture parallèle pour la gestion d'une base de données images est décrite. Un des traitements les plus classiques est étudié, il est réparti entre les processeurs.*

Un modèle de cette architecture est réalisé, dans le but d'évaluer le temps de réponse des traitements.

Une solution approchée est donnée: elle est obtenue en considérant une file multi-serveur avec des arrivées groupées. Cette méthode est d'un intérêt général.

Mots clés : Modélisation; simulation; parallélisme; fork-join; Markov; réseau de files d'attente.

Abstract. — *A parallel architecture is described for the management of an image database. One of the most usual processing is studied, it is distributed among the processors.*

In order to get the response time, a model of the system is made.

An approximate solution is given: a fork-join with p processors and n arrivals (where n is different from p) is approximated by the join of the response times of a multiservor queue with bulk arrivals. This method is of general use.

Keywords : Modelisation; simulation; parallelism; fork-join; Markov; queueing network.

I. INTRODUCTION

Ce travail s'inscrit dans un projet de conception d'une machine base de données multi-media, qui est surtout destinée à la gestion d'images, notamment d'images satellites.

Cette architecture comprend plusieurs stations de travail et un serveur base de données, qui communiquent par un bus de type Ethernet (*voir* la figure 1).

(*) Reçu février 1987.
(1) L.C.S., I.M.A.G., I.N.P.G., 46, avenue F.-Viallet, 38000 Grenoble.
(2) I.N.T., 9, rue Charles-Fourier, 91011 Évry Cedex.
Ce travail a été réalisé au L.C.S., I.M.A.G., I.N.P.G., 46, avenue F.-Viallet, 38000 Grenoble.

Recherche opérationnelle/Operations Research, 0399-0559/88/01 45 21/$ 4.10

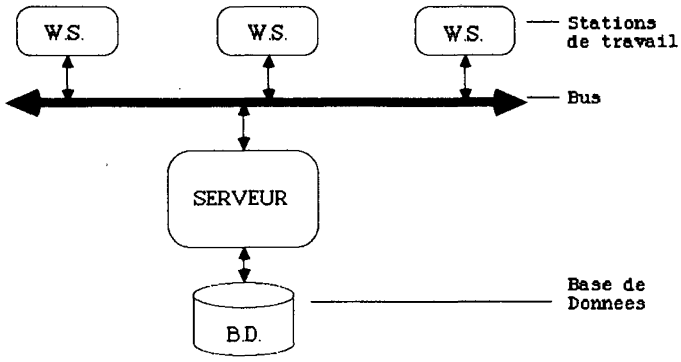

Figure 1. – L'architecture générale.

Les stations de travail recoivent des images du serveur base de donnée. Les images sont des images satellites de très grande dimension (SPOT). Ces images ont parfois besoin d'un traitement préalable, selon les demandes des utilisateurs. Ce traitement, dont le temps de réponse peut être très long est fait sur une architecture M.I.M.D. (voir *fig.* 2). Celle-ci comprend des processeurs spécialisés pour le traitement d'images, leur mémoire locale, et des controleurs disques. Ces unités sont reliées à un bus, ainsi que le calculateur hôte, le calculateur principal du serveur.

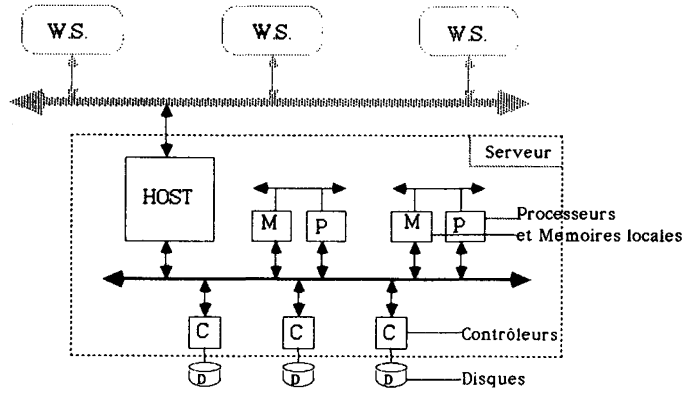

Figure 2. – Le serveur base de données.

Nous voulons estimer le temps d'accès à une image, y compris le temps nécessaire au serveur pour le traitement. Différents traitements peuvent être

nécessaires. Nous ne décrirons que le plus utilisé; la rectification géométrique simple d'une image SPOT.

Le modèle du système sous forme de réseau de files d'attente conduit à un problème de FORK-JOIN, qui est différent de ceux qui ont été étudiés dans [1], [2], ou [3]. On a un FORK lorsqu'un travail crée n tâches identiques qui seront traitées par p processeurs. Le JOIN est effectué lorsque les n tâches sont terminées. Ici, [4], il y a n tâches et n est beaucoup plus grand que p, à cause de la taille de la mémoire locale des processeurs spécialisés. De plus, les tâches ne sont pas préaffectées aux processeurs, mais sont traitées par les processeurs, lorsqu'ils sont libres, avec une priorité premier arrivé premier servi.

Signalons qu'une étude analogue, plus générale, mais moins complète, a été faite, par une autre méthode, simultanément par Nelson, Towsley et Tantawi [5].

Au paragraphe 2, nous décrirons le modèle de l'architecture et l'algorithme distribué de rectification. Au paragraphe 3, nous présenterons une estimation du temps de réponse de ce Fork-Join particulier.

II. LE MODÈLE

II. 1. L'architecture

Nous ne modélisons que le serveur représenté sur la figure 2. Ce système comprend les éléments suivants:

II. 1. 1. *Le bus*

Ce bus relie les processeurs, leur mémoire locale, les supports de masse et le calculateur hôte. Chacun d'eux peut envoyer des demandes de bus et les réitérer si le bus n'est pas libre. Le bus ne travaille pas en mode partagé: une fois que le bus est alloué à un demandeur, tous les messages envoyés par celui-ci sont transmis.

II. 1. 2. *Les processeurs spécialisés*

Il s'agit de processeurs Matra spécialisés. Chacun d'eux est en réalité un ensemble comportant un processeur d'échange, un processeur arithmétique et une mémoire locale du processeur arithmétique, reliés à un bus interne. Sur la figure 2 cet ensemble est représenté par un seul bloc: P. Nous modélisons cet ensemble par une seule file d'attente.

Chacun reçoit de l'hôte un bloc de commandes qui correspondent à la tâche qui lui est affectée. Ce bloc comprend surtout des requêtes de calcul, de transferts de mémoire et de synchronisations avec le calculateur hôte.

II.1.3. *Le calculateur hôte*

Il reçoit des requêtes depuis les stations de travail, par le bus Ethernet. Il «découpe» le traitement en n tâches pour les processeurs spécialisés et distribue ces tâches en fonction de la disponibilité des processeurs. Pendant l'exécution d'une tâche, l'hôte peut aussi demander à un processeur d'échange de transférer des données sur la mémoire locale du processeur. Quand une image est traitée, c'est-à-dire quand les n tâches sont terminées, l'hôte renvoie l'image traitée à la station de travail qui la lui a demandée.

II.1.4. *Les mémoires locales*

Elles sont représentées par le bloc M sur la figure 2. Chacune d'elle a une capacité qui peut atteindre 2 Moctets. Elles sont organisées en 2 bancs de 1 Moctet chacun, pouvant fonctionner en FLIP-FLOP. On peut accéder à ces deux bancs simultanément, par exemple, quand le processeur accède à un banc, l'hôte peut accéder à l'autre.

II.1.5. *Les supports de masses*

Ils sont composés des disques proprement dits, soient optiques numériques soient des disques magnétiques, selon les versions. Ils sont représentés par les blocs D sur la figure 2. Nous avons déjà indiqué le rôle des processeurs d'échanges qui peuvent recevoir des blocs de requêtes du calculateur hôte, telles que des requêtes de lecture sur le disque, suivies de requêtes d'écriture sur un banc de la mémoire locale du processeur.

II.2. Un exemple de traitement

La rectification géométrique [6] est un des traitements les plus courants et les plus pénalisants en temps de réponse. Décrivons rapidement cet algorithme pour une image panchromatique.

II.2.1. *Principe de l'algorithme*

Il applique à l'image brute reçue du satellite, la transformation inverse d'un modèle de déformation de l'image due aux mouvements relatifs des capteurs par rapport à notre planète :

- rotation du globe,
- rotation du satellite,

– balayage du capteur,

– tangage et roulis du satellite...

Soit $(l, p)_c$ les coordonnées entières d'un point de l'image corrigée I_c, la rectification de l'image brute I_b est déduite d'une transformation T:

$$T: \quad (l, p)_c \rightarrow (x, y)_b$$

T transforme un point de l'image corrigée en sa «position» dans l'image brute, x et y sont des nombres réels (*voir* la figure 3).

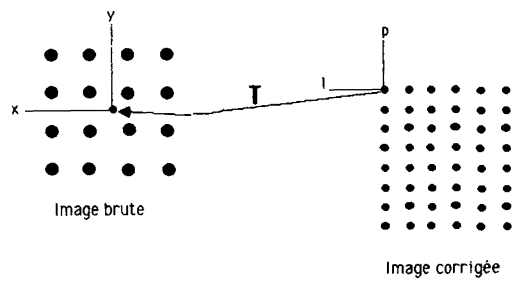

Figure 3. – Image brute et image corrigée.

La partie entière de x et y permet de calculer le voisinage de 4×4 points de l'image brute. La valeur du pixel dans l'image corrigée est calculée par interpolation à partir des valeurs de ces 4×4 pixels. Ces valeurs sont pondérées suivant la partie fractionnaire de x et y. T est approché par une estimation polynômiale bidimensionnelle.

$$x(l, p) = P(x, y)$$
$$y(l, p) = P'(x, y)$$

P et P' sont des polynômes des variables x et y. Le calcul de ces polynômes n'est pas développé ici. Il est différent suivant le niveau de correction demandé (niveaux 2, 3, S_1, S_2). L'application de ces polynômes pour tous les points de l'image corrigée serait trop onéreuse. Aussi les polynômes ne sont-ils appliqués qu'à des points d'une grille régulière: GIG. Cette grille est alors densifiée à l'ensemble des points par interpolation linéaire (ajoutons que pour une correction de niveau 3, deux grilles sont nécessaires: une pour l'altitude maximale et une pour l'altitude minimale, pour permettre par la suite de compenser par interpolation les déformations dues aux variations d'altitude,

à partir d'un modèle numérique du terrain). Ici, nous ne considérerons que des rectifications de niveau 2 et nous supposons que la GIG est préalablement calculée.

II.2.2. *Description rapide de l'algorithme*

Entrée: l'image brute est stockée sur disque par segments correspondant chacun à un bloc de points rectangulaire de taille (n_b, m_b).

Sortie: l'image corrigée est aussi calculée par blocs de points rectangulaires de taille (n_c, m_c).

début traitement

Pour chaque bloc de l'image corrigée *faire*

début pour

(*a*) détermination du domaine de l'image brute nécessaire (ensemble de blocs),

(*b*) chargement des pavés bruts requis et « désegmentation »,

(*c*) calcul des positions de rééchantillonnage de chaque point du bloc par interpolation bilinéaire dans la GIG,

(*d*) affectation d'une valeur radiométrique à chaque point du segment rectifié (balayage par ligne) par interpolation,

fin pour

fin traitement

Remarque: La désegmentation consiste à réordonner les points du bloc de l'image brute pour qu'ils soient adressables par ligne (*voir* la figure 4).

II.2.3. *Parallélisation de cet algorithme*

Les calculs correspondant à chaque bloc de l'image corrigée peuvent être faits indépendamment les uns des autres, (c'est-à-dire que les calculs de l'un n'ont pas besoin des résultats des calculs de l'autre, il n'est toutefois pas exclu qu'il y ait des conflits d'accès aux données ou à certaine ressource), la parallélisation de cet algorithme est alors évidente.

Une tâche sera le traitement des pixels de n_B blocs. Soit N_{BC} le nombre de blocs dans l'image corrigée à calculer. L'hôte aura à gérer n tâches si:

$$N_{BC} = n_B . n$$

L'hôte aura à répartir ces n tâches parmi p processeurs. Le traitement d'une image sera terminé lorsque les n tâches seront terminées. Remarquons que cet algorithme sera écrit une fois pour tous les traitements. Les problèmes

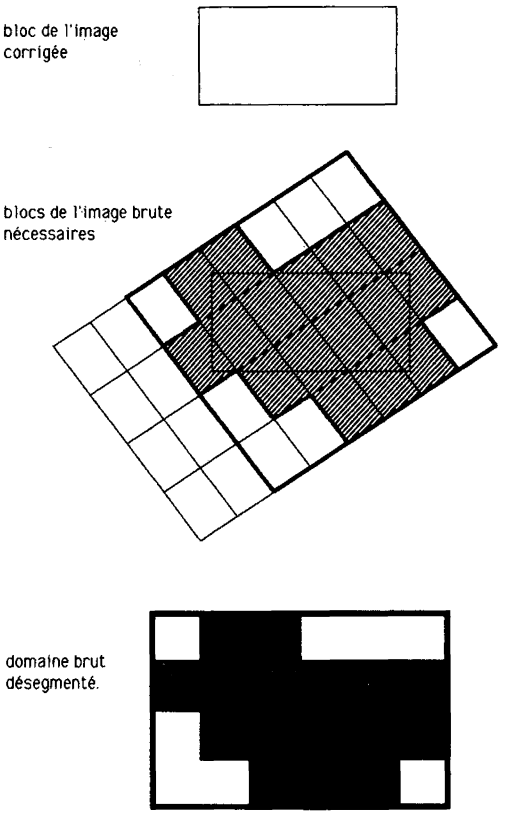

Figure 4. — La désegmentation.

géométriques sont difficiles et il n'est pas facile de changer dynamiquement la valeur de n_B afin de rendre le nombre de tâches identique au nombre de processeurs. Si n_B était un paramètre, cela introduirait de grandes difficultés de programmation. De plus pour des raisons de taille mémoire, n sera en général beaucoup plus grand que p.

III. ÉVALUATION DU TEMPS DE RÉPONSE POUR LE TRAITEMENT D'UNE IMAGE

III. 1. Le modèle

Un modèle du serveur a été réalisé (*voir* la figure 5). C'est un réseau de files d'attente. Il y a des théorèmes classiques pour les files d'attente [7], mais ils ne peuvent pas être appliqués ici, à cause de la synchronisation des tâches.

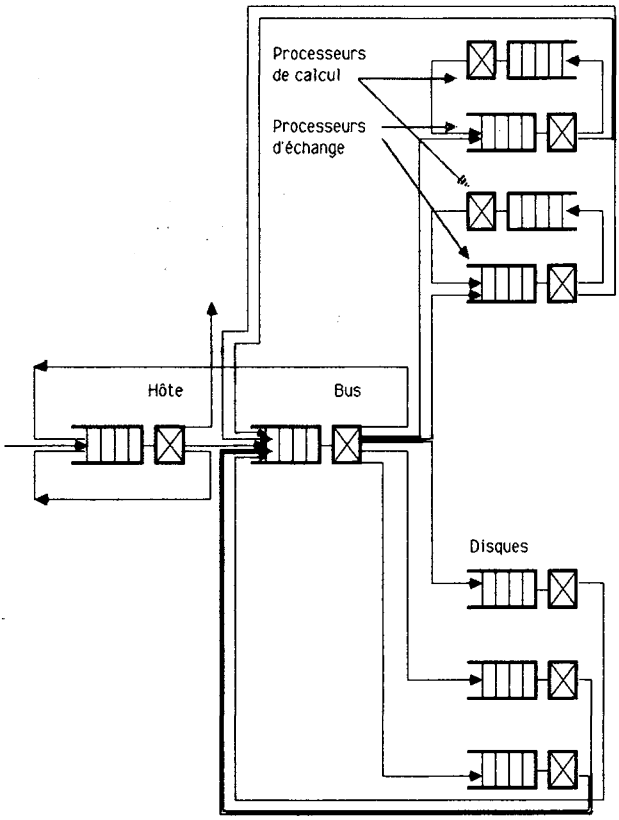

Figure 5. — Le modèle du serveur.

Dès qu'on rencontre des problèmes de synchronisations de ce type, les réseaux classiques de files d'attente ne sont pas assez puissants. Il faudrait introduire des réseaux de Pétri stochastiques [8]. Mais, alors, le nombre d'états de la chaîne de Markov devenant très grand, il n'est pas facile d'obtenir des résultats. Aussi, avons-nous réalisé une simulation. Nous avons utilisé un progiciel de modélisation: QNAP [9] en ajoutant des sous-programmes pour les synchronisations.

III.1.1. *Le temps de service*

Les blocs ayant une longueur constante, chacun des temps de service a été supposé de durée constante. Nous avons obtenu par simulation des estimations du temps de réponse pour une tâche, et bien sûr, il est apparu que ce

temps n'était plus constant. Une tâche a besoin du bus et de l'hôte, ces ressources sont partagées avec les autres tâches, aussi y-a-t'il des problèmes d'attente. Il apparaît qu'il n'est pas trop mauvais de faire pour une tâche l'hypothèse d'un temps de réponse exponentiellement réparti dont l'espérance est une valeur mesurée: $S = 1/\mu$. (Nous supposons que l'hypothèse exponentielle n'est pas trop mauvaise, en effet la variance de ce temps de réponse est de l'ordre du carré de la moyenne.) Nous allons considérer un modèle simplifié dans lequel les serveurs auront un temps de service exponentiel de taux μ.

III.1.2. *Le n, p FORK-JOIN*

Pour avoir des résultats analytiques pour le temps de réponse du traitement d'une image, nous proposons d'évaluer approximativement le taux μ, soit par simulation, soit à l'aide d'un modèle analytique en négligeant dans un premier temps les problèmes de synchronisations, puis de résoudre le problème suivant:

Considérons l'arrivée des requêtes de traitement et la création des n tâches (FORK). Ces tâches seront exécutées par p processeurs. Elles auront besoin de plusieurs ressources; agrégeons ces ressources et supposons qu'il y a p serveurs exponentiels de taux μ. Le traitement de l'image sera terminé quand chacune des n tâches sera terminée, on a le JOIN de ces tâches. Le temps de réponse du traitement sera le maximum du temps de réponse de chacune des tâches.

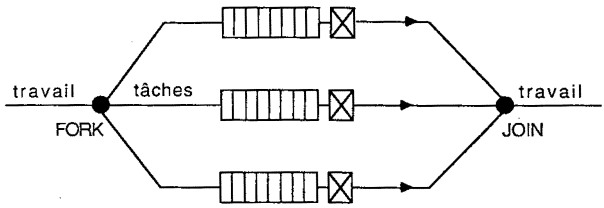

Figure 6. — Le FORK-JOIN classique.

Dans le FORK-JOIN classique [1, 2, 3], il y a p tâches avec des arrivées groupées et chacune d'elles est affectée à un des p processeurs. Ici, il y a n tâches et elles peuvent être servies par n'importe lequel des serveurs, quand il est libre. (Même si $n=p$, s'il se trouve que l'un des p serveurs est moins chargé qu'un autre, il est possible que deux tâches du même travail soient servies par le même serveur, l'une après l'autre.)

La figure 6 représente le problème classique du FORK-JOIN, la figure 7 représente le cas que nous traitons ici.

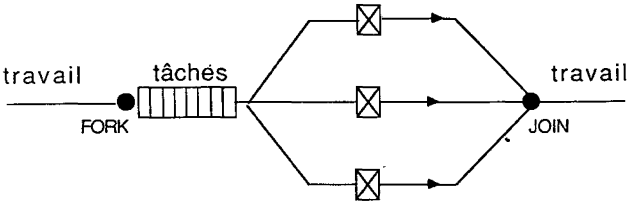

Figure 7. — Le *n*, *p* FORK-JOIN.

Les travaux se succèdent donc dans la file FORK on peut trouver des tâches correspondant à des travaux différents. Pour ce qui est des tâches correspondant à des travaux différents, la priorité est premier arrivé premier servi. (Remarque: cette loi de priorité est couramment appelée FIFO, ceci serait en fait incorrect dans le cas présent: il y a une différence entre premier arrivé, premier servi et premier arrivé, premier sorti, dans le cas où il y a plusieurs serveurs.)

Ce cas particulier de FORK-JOIN correspond à *une file multi-serveur comportant p serveurs en parallèle, exponentiels, avec le même taux de service* μ *et des arrivées groupées, par groupe de n et priorité premier arrivé premier servi. Nous voulons calculer le temps de réponse du Join des n tâches de ce travail, nous l'appelerons temps de réponse du groupe.*

III. 2. Solution du *n*, *p* FORK-JOIN

Considérons l'arrivée d'un groupe. Le temps de réponse R du groupe ne dépend que du nombre k de tâches qui précèdent le groupe dans la file FORK.

$$R = \sum_{k=0}^{\infty} P(k) R(k)$$

$R(k)$ est le temps de réponse du groupe quand il est précédé par k tâches. $P(k)$ est la probabilité d'avoir k tâches dans la file FORK.

III.2.1. *Calcul des P (k)*

Le calcul des $P(k)$ est classique [10]. La chaîne de Markov est représentée sur la figure 8.

Rappelons les équations d'état:

$n>p$

$$i=0, \quad \lambda P(0)=\mu P(1)$$

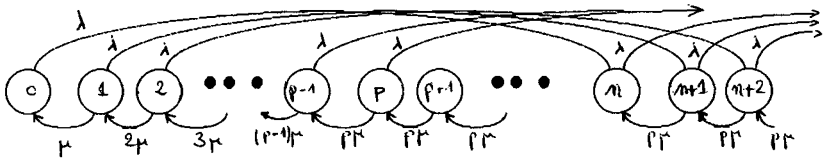

Figure 8. – Chaîne de Markov pour $n>p$.

$$0<i<p, \quad (i\mu+\lambda)P(i)=(1+i)\mu P(i+1)$$
$$p\leqq i<n, \quad (p\mu+\lambda)P(i)=p\mu P(i+1)$$
$$i\geqq n, \quad (p\mu+\lambda)P(i)=p\mu P(i+1)+\lambda P(i-n)$$

$n\leqq p$

$$i=0, \quad \lambda P(0)=\mu P(1)$$
$$0<i<n, \quad (i\mu+\lambda)P(i)=(1+i)\mu P(i+1)$$
$$n\leqq i<p, \quad (\lambda+i\mu)P(i)=(1+i)\mu P(i+1)+\lambda P(i-n)$$
$$i\geqq p, \quad (\lambda+p\mu)P(i)=p\mu P(i+1)+\lambda P(i-n).$$

Introduisons la fonction génératrice:

$$F(x)=\sum_{i=0}^{\infty} x^i P(i).$$

En multipliant chaque équation par x^i et en additionnant on obtient:

$$\lambda\sum_{i=0}^{\infty} x^i P(i)+\sum_{j=1}^{p}\left(\sum_{i=j}^{\infty} x^i \mu P(i)\right)=\sum_{j=1}^{p}\left(\sum_{i=j}^{\infty} x^{i-1}\mu P(i)\right)+\lambda\sum_{i=n}^{\infty} x^i P(i-n)$$

De sorte que:

$$\lambda F(x)+\mu(F(x)-P(0))+\mu(F(x)-P(0)-P(1)\times x)+\ldots$$

$$+\mu(F(x)-P(0)-xP(1)-\ldots-x^{p-1}P(p-1))$$

$$=\frac{\mu}{x}(F(x)-P(0))+\frac{\mu}{x}(F(x)-P(0)-xP(1))+\ldots$$

$$+\frac{\mu}{x}(F(x)-P(0)-xP(1)-\ldots-x^{p-1}P(p-1))+\lambda x^n F(x)$$

et

$$F(x)\left(\lambda+p\mu-\frac{p\mu}{x}-\lambda x^n\right)=\mu\left(\sum_{i=0}^{p-1}(p-i)P(i)x^i\right)-\frac{\mu}{x}\left(\sum_{i=0}^{p-1}(p-i)P(i)x^i\right)$$

$$F(x)=\frac{\mu\left(\sum_{i=0}^{p-1}(p-i)P(i)x^i\right)}{\mu p-\lambda x((x^n-1)/(x-1))}.$$

$n>p$:

Pour $0<i<p$ on a:

$$P(i)=\frac{(i-1)\mu+\lambda}{i\mu}.P(i-1)=\frac{1}{i!}\left(\prod_{k=0}^{i-1}\left(\frac{\lambda}{\mu}+k\right)\right).P(0)$$

$n\leqq p$:

Pour $0<i<n$ on a:

$$P(i)=\frac{(i-1)\mu+\lambda}{i\mu}.P(i-1).$$

Pour $n\leqq i<p$ on a:

$$P(i)=\left[\frac{(i-1)\mu+\lambda}{i\mu}P(i-1)\right]-\frac{\lambda}{i\mu}.P(i-n-1).$$

On obtient:

$$P(i)=A_iP(0)$$

en posant:

$$n>p,\quad 0<i<p:\qquad A_i=\frac{1}{i!}\prod_{k=0}^{i-1}\left(\frac{\lambda}{\mu}+k\right)$$

$$n \leqq p, \quad 0 < i < n: \quad A_i = \frac{1}{i!} \prod_{h=0}^{i-1} \left(\frac{\lambda}{\mu} + k \right)$$

$$n \leqq i < p, \quad A_i = \frac{(\lambda/\mu) + i - 1}{i} A_{i-1} - \frac{\lambda}{\mu i} A_{i-n-1}$$

$$A_0 = 1.$$

$$F(x) = \frac{\mu \sum_{i=0}^{p-1} (p-i) A_i x^i}{\mu p - \lambda x (x^n - 1)/(x-1)} \cdot P(0) = \frac{\mu \sum_{i=0}^{p-1} (p-i) A_i x^i}{\mu p - \lambda x \sum_{i=0}^{n-1} x^i} \cdot P(0).$$

$P(0)$ est obtenu à partir de:

$$F(1) = 1 = \sum_{i=0}^{\infty} P(i).$$

On a:

$$P(0) = \frac{\mu p - n\lambda}{\mu \sum_{i=0}^{p-1} (p-i) A_i}$$

à condition que: $n\lambda < \mu p$.

Ainsi:

$$\forall i, \quad 0 < i < p, \qquad P(i) = \frac{\mu p - n\lambda}{\mu \sum_{k=0}^{p-1} (p-k) A k} A_i$$

Plus tard nous aurons besoin de:

$$F'(1) = \sum_{i=0}^{\infty} i P(i)$$

$$F'(1)=\frac{\sum_{i=1}^{p-1}(p-i)\,i\,A\,i}{\sum_{i=0}^{p-1}(p-i)\,A\,i}+\frac{\lambda n(n+1)}{2(\mu p-n\lambda)}$$

$$F'(x)=\left(\frac{\mu\sum_{i=1}^{p-1}(p-i)\,i\,A_i\,x^{i-1}}{\left(\mu p-\lambda x\sum_{i=0}^{n-1}x^i\right)}+\frac{\mu\left(\sum_{i=0}^{p-1}(p-i)\,A_i\,x^i\right)\times\lambda\sum_{i=0}^{n-1}(i+1)\,x^i}{\left(\mu p-\lambda x\sum_{i=0}^{n-1}x^i\right)^2}\right)\times P(0)$$

$$F'(1)=\left(\mu\frac{\left(\sum_{i=1}^{p-1}(p-i)\,i\,A_i\right)}{\mu p-n\lambda}+\lambda\mu\frac{n(n+1)\left(\sum_{i=0}^{p-1}(p-i)\,A_i\right)}{2(\mu p-n\lambda)^2}\right)P(0).$$

III.2.2. *Calcul des R (k)*

A chaque instant, l'état du groupe sera caractérisé par le couple: (i,j). i est le nombre de tâches précédant le groupe dans l'ensemble de la station (ceci signifie que quelques-unes de ces tâches sont éventuellement en train

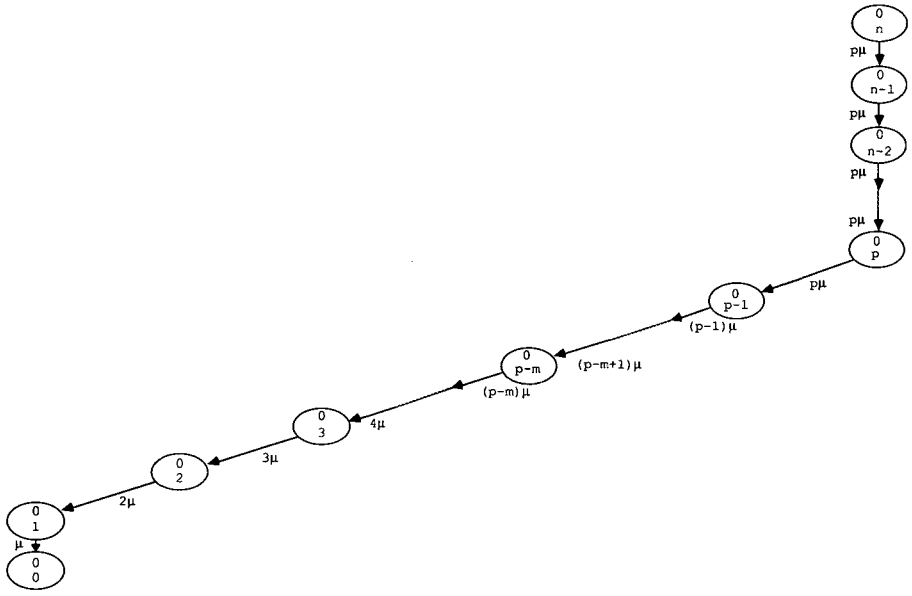

Figure 9. — **La chaîne de Markov.** $k=0$, $n>p$.

d'être servies, d'ailleurs si i est inférieur à p on est sûr que les i tâches sont en train d'être servies). Les i tâches commenceront à être servies avant chacune des tâches du groupe, mais comme il y a p serveurs, elles peuvent être terminées après que quelques-unes des tâches du groupe soient terminées. j est le nombre de tâches du groupe qui ne sont pas terminées.

Nous avons considéré la chaîne de Markov des états du travail. Différents cas de figure sont possibles. Nous donnons les trois figures (*fig.* 9, 10 et 11), dans le cas où n est supérieur à p. Chaque état de la chaîne est caractérisé par 2 nombres : le nombre supérieur, sur le schéma, correspond au nombre i de tâches non terminées parmi celles qui précèdent celles du travail considéré, et le nombre inférieur correspond au nombre j de tâches non terminées parmi celles du travail considéré. On calcule les temps de séjour dans les diverses places et les probabilités des divers chemins possibles et on obtient les temps moyens $R(k)$ du JOIN des n tâches du travail considéré en fonction de k.

Quand $n>p$ la figure 9 montre le cas où la station est vide, quand le groupe arrive. L'espérance du temps de réponse du groupe est alors :

$$R(0)=\frac{n-p+1}{p\mu}+\frac{1}{(p-1)\mu}+\frac{1}{(p-2)\mu}+\ldots+\frac{1}{2\mu}+\frac{1}{\mu}=\frac{n-p+1}{p\mu}+\sum_{i=1}^{p-1}\frac{1}{i\mu}.$$

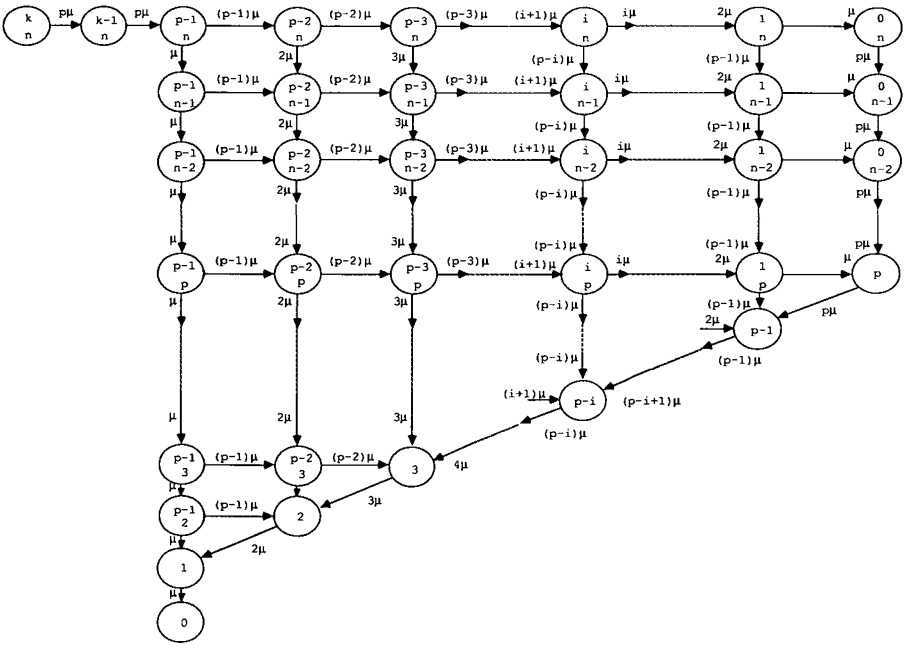

Figure 10. — La chaîne de Markov. $k>p$, $n>p$.

La figure 10 montre le cas où quand le groupe arrive il y a $k\,(k \geqq p)$ tâches dans la station. La figure 11 montre le cas où $k<p$. Il apparaît que pour tous les k, tels que $k \geqq p$

$$R(k)=R(p-1)+\frac{k-p+1}{p\mu}.$$

Aussi ne reste-t'il qu'à estimer $R(k)$ pour $0<k<p$.

Revenons aux chaînes de Markov. Rappelons que l'état du groupe est représenté par (i,j).

Quand $i>p-1$ seules les tâches des groupes précédents sont en service, et avec un taux $p\mu$ le groupe passe dans l'état $(i-1,n)$.

Quand $i \leqq p-1$ quelques-uns des serveurs vont servir quelques-unes des tâches du groupe. Avec un taux $i\,\mu$, une des i tâches précédentes sera terminée et le groupe passera dans l'état $(i-1,j)$ et avec un taux $(p-i)\,\mu$ une des tâches du groupe sera terminée et il passera dans l'état $(i,j-1)$.

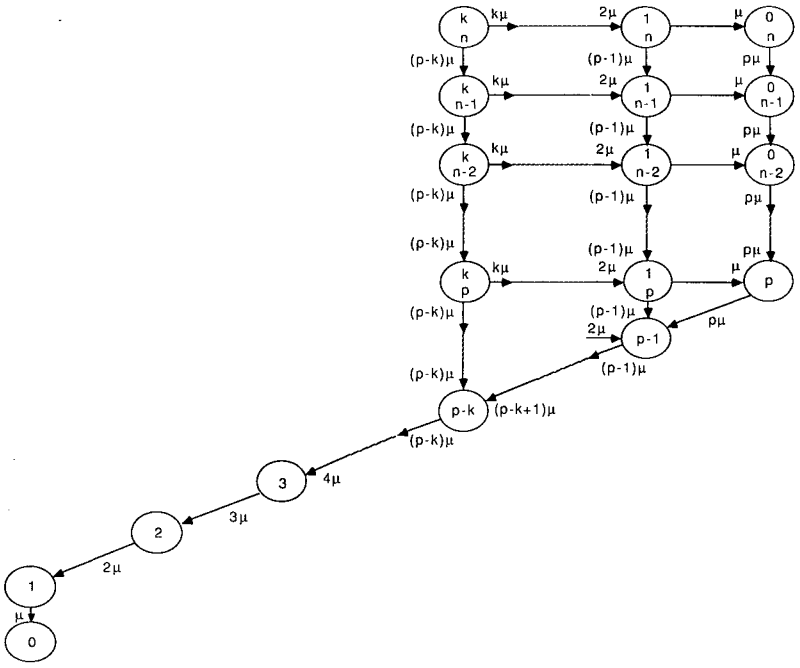

Figure 11. — La chaîne de Markov. $0<k<p$, $n>p$.

Quand $i \leqq p-j$ toutes les tâches du groupe sont en train d'être servies et le temps de réponse dépend seulement des serveurs qui servent ces tâches, on ne s'intéresse plus aux tâches précédentes, aussi avons-nous agrégé tous les états correspondants aux différentes valeurs de i, $(i \leqq p-j)$ en un état noté (j). Avec un taux $j\mu$ le groupe passe dans l'état $(j-1)$.

L'espérance du temps de réponse d'un groupe de n tâches, conditionné par le fait que lorsqu'il arrive il y a déjà k tâches dans la station, est noté $R(k)$. C'est l'espérance du temps mis par le groupe pour passer de l'état (k, n) à l'état (0). Notons $R(i,j)$ l'espérance du temps de passage de l'état (i,j) à l'état (0).

a. $R(p-1, 1) = 1/\mu$.

b. $\forall j \mid 1 < j \leqq p$

$$R(p-j,j) = \frac{1}{j\mu} + R(p-j+1, j-1).$$

c. $\forall j \mid p < j \leqq n$

$$R(0,j) = \frac{1}{p\mu} + R(0, j-1).$$

d. $\forall\, i, j \mid i+j > p,\ p > i > 0,\ 1 < j \leqq n$

$$R(i, j) = \frac{i}{p}\left(\frac{1}{p\mu} + R(i-1, j)\right) + \frac{p-i}{p}\left(\frac{1}{p\mu} + R(i, j-1)\right)$$

$$R(i, j) = \frac{1}{p}\left(\frac{1}{\mu} + i\,R(i-1, j) + (p-i)\,R(i, j-1)\right)$$

A partir de ces relations il est possible de calculer par récurrence,

$$R(k) = R(k, n)$$

pour $0 < k < p$.

Le cas où $n \leqq p$ est étudié sur les figures 12, 13 et 14, desquelles on déduit:

a. $R(p-1, 1) = 1/\mu$.

b. $1 < j \leqq n$

$$R(p-j,j) = \frac{1}{j\mu} + R(p-j+1, j-1)$$

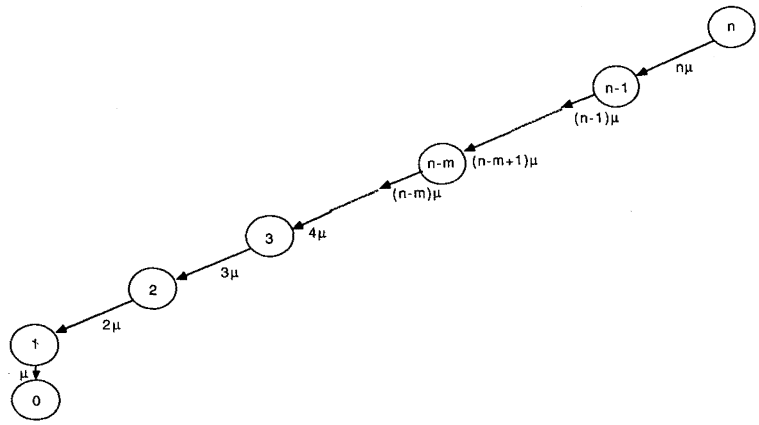

Figure 12. – $k \leqq p-n$, $n \leqq p$.

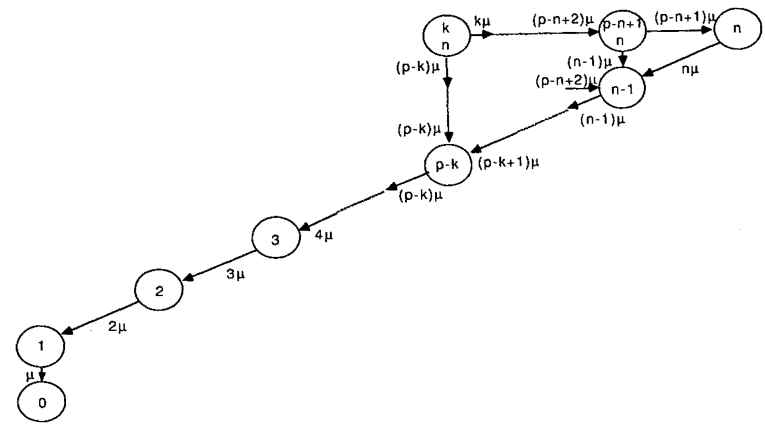

Figure 13. – $p-n<k<p$, $n \leqq p$.

c. $0 \leqq i < p-n$

$$R(i, n) = R(p-n, n) = R(i) = R(p-n)$$

d. $p-n+1 \leqq i \leqq p-1$, $i+j>p$, $1<j \leqq n$

$$R(i, j) = \frac{1}{p}\left(\frac{1}{\mu} + i\,R(i-1, j) + (p-i)\,R(i, j-1)\right)$$

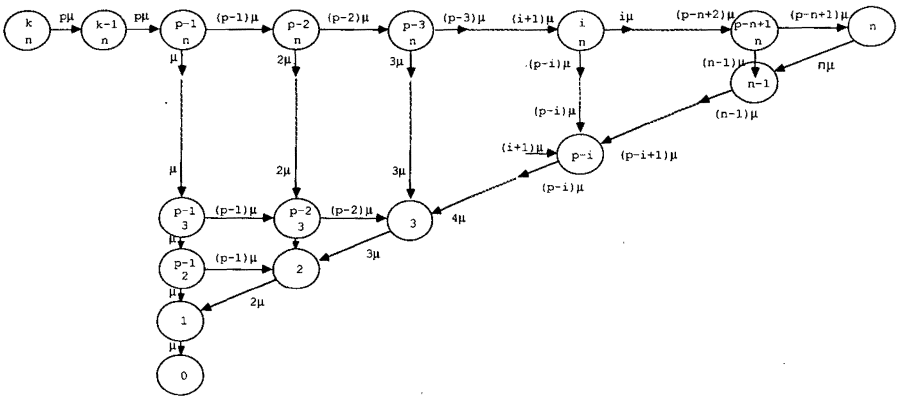

Figure 14. — $k \geqq p$, $n \leqq p$.

III.2.3. *Calcul du temps de réponse du groupe*

Par sommation on obtient alors:

$$R = \sum_{i=0}^{\infty} R(i) P(i) = \sum_{i=0}^{p-1} R(i) P(i) + \sum_{i=p}^{\infty} \left(R(p-1) + \frac{i-p+1}{p\mu} \right) P(i)$$

$$= \sum_{i=0}^{p-1} R(i) A_i P(0) + \left(R(p-1) - \frac{p-1}{p\mu} \right) \sum_{i=p}^{\infty} P(i) + \sum_{i=p}^{\infty} \frac{i}{p\mu} P(i)$$

$$= \sum_{i=0}^{p-1} R(i) A_i P(0) + \left(R(p-1) - \frac{p-1}{p\mu} \right) \left(1 - \sum_{i=0}^{p-1} A_i P(0) \right)$$

$$+ \frac{1}{p\mu} \left(F'(1) - \sum_{i=1}^{p-1} i A_i P(0) \right)$$

Notons:

$$S_0 = \sum_{i=0}^{p-1} A i$$

$$S_1 = \frac{1}{p} \sum_{i=0}^{p-1} i A_i$$

$$S_2 = \frac{1}{p^2} \sum_{i=0}^{p-1} i^2 A_i$$

$$\rho = \frac{n\lambda}{\mu p}.$$

On a :

$$P(0) = \frac{1-\rho}{S_0 - S_1}$$

$$F'(1) = p\frac{S_1 - S_2}{S_0 - S_1} + \frac{n+1}{2}\frac{\rho}{1-\rho}$$

$$1 - \sum_{i=0}^{p-1} A_i P(0) = \frac{\rho S_0 - S_1}{S_0 - S_1}$$

$$\sum_{i=1}^{p-1} i A_i P(0) = p S_1 \times \frac{1-\rho}{S_0 - S_1}$$

et :

$$R = \sum_{i=0}^{p-1} \frac{R(i) A_i (1-\rho)}{S_0 - S_1} + \left(R(p-1) - \frac{p-1}{p\mu}\right) \times \left(\frac{\rho S_0 - S_1}{S_0 - S_1}\right) + \frac{1}{p\mu}\left(\frac{\rho S_1 - S_2}{(S_0 - S_1)} p + \frac{n+1}{2}\frac{\rho}{1-\rho}\right)$$

Ce temps de réponse du groupe peut facilement être calculé par programme (le programme est disponible à la demande).

Les formules sont relativement simples à calculer. Le temps de réponse est une somme de p termes, chacun d'eux peut être calculé par une récurrence dont la complexité est de l'ordre de n^2, en effet chacun des termes à calculer est un $R(k, n)$ et pour le calculer il faut connaître tous les $R(i, j)$ pour les i et j inférieurs à k et n respectivement.

IV. CONCLUSION

Nous avons donné une solution pour le temps de réponse du groupe dans ce cas particulier de FORK-JOIN, que nous avons appelé le n, p Fork-join.

Ces formules sont d'un intérêt général.

Rappelons la définition du n, p Fork-join. Quand le FORK génère n tâches qui sont traitées par p serveurs, nous avons calculé le temps de réponse du

JOIN des n tâches, en supposant les temps de service exponentiellement distribués et la loi de priorité premier arrivé premier servi.

Ce travail nous a déjà servi pour d'autres études d'architectures faiblement couplées, dans lesquelles le n, p Fork-join est un cas fréquent, notamment pour la modélisation de LCAP un réseau de supercalculateurs FPS reliés en étoile à un ordinateur IBM central (à Kingston, N.Y., U.S.A.).

REMERCIEMENTS

F. Baccelli (I.N.R.I.A.) nous a beaucoup aidé au cours de discussions très positives. Ce travail a été fait pendant que P. Houeix recevait une bourse de Matra DSEO (subvention Cifre). Nous remercions Matra et spécialement, G. Demathieu et M. Leroux et son groupe.

BIBLIOGRAPHIE

1. L. Flatto et S. Hahn, *Two Parallel Queues Created by Arrivals with two Demands I*, S.I.A.M. J. Appl. Math., vol. 44, 1984, p. 1041-1053.
2. F. Baccelli, A. M. Makowski et A. Schwartz, *Simple Computable Bounds and Approximations for the Fork-join Queue*, J.A.C.M., 1986.
3. A. Duda et T. Czachorski, *Performance Evaluation of the Fork and Join Synchronisation Primitives*, I.S.E.M. Research Report, janvier 1986.
4. P. Houeix et M. Becker, *Modelling a Parallel Architecture for Image Processing*, Rapport de recherches I.M.A.G., n° 6461, mars 1987.
5. R. Nelson, D. Towsley et A. N. Tantawi, *Performance Analysis of Parallel Processing Systems*, Livre du Congrès: 1987 A.C.M. SIGMETRICS Conference on Measurement and Modelling of Computer Systems, Banff, 11-14 mai 1987 (à paraître).
6. *Rectification des images SPOT*, Research Report, Matra DSEO.
7. F. Baskett, K. M. Chandy, R. R. Muntz et F. G. Palacios, *Open, Closed and Mixed Network of Queues with Different Classes of Customers*, J.A.C.M., vol. 22, 1975, p. 248-260.
8. G. Florin et S. Natkin, *One-Place Unbounded Stochastic Petri Nets: Ergodic Criteria and Steady-State Solutions*, J. of Syst. and Soft., vol. 6, n° 1-2, Special Issue on Modeling and Performance Evaluation of Parallel Systems, M. Becker éd., mai 1986.
9. M. Veran et D. Potier, *A Portable Environment for Queueing Networks Modelling*, Int. Conf. on Modeling Techniques and Tools for Performance Analysis, Paris, mai 1984.
10. L. Kleinrock, *Queueing Systems*, Wiley, 1975.