Modélisation et évaluation d'algorithmes de partage de charge

Maryse Béguin, Jean-Marc Vincent & Bernard Ycart

LMC-IMAG, Projet MAI,
BP 53,
38041 GRENOBLE CEDEX
FRANCE
E-mail : {*Maryse.Beguin, Jean-Marc.Vincent, Bernard.Ycart*}*@imag.fr*

RÉSUMÉ. *L'étude développée dans cet article porte sur le comportement dynamique de protocoles de partage de charge dans des systèmes parallèles et/ou distribués. Après une synthèse des différentes approches utilisées pour l'évaluation quantitative de ces protocoles, nous proposons un modèle simplifié mettant en évidence les différents comportements possibles. De plus, ce modèle permet une estimation optimiste des gains que l'on peut espérer, en partageant la charge, en fonction des temps de transfert et de calcul. Les indices de performance étudiés sont le débit du système, la saturation de la mémoire, la charge de travail, le temps moyen de réponse...*

Ce modèle simplifié permet d'envisager de nouvelles méthodes d'évaluation de grands réseaux avec partage de charge basées sur des propriétés statistiques spatiales du réseau. Quelques images de simulations sont présentées.

MOTS CLÉS : *Transfert de charge, systèmes parallèles/distribués, évaluation de performances, processus markoviens*

ABSTRACT. *The study developped in this article focuses on the dynamical behavior of load sharing protocols in parallel and/or distributed systems.*

After a synthesis of the different approaches used for the quantitative evaluation of these protocols, we propose a simplified model which yields the different possible behavior. Besides, this model leads to an optimistic estimation of the benefits one can expect with a load sharing policy according to the transfer and the computation delays. The performance indexes under consideration are the throughput, the memory saturation, the workload, the mean response time.

This model gives rise to new evaluation methods on very large networks, endowed with a transfer policy, based on the regular topology of the network. Some images of simulation are presented.

KEYWORDS : *Load transfer, parallel/distributed systems, performance evaluation, Markov processes*

1 Introduction

Le développement récent des architectures multi-processeurs parallèles ou distribuées permet de traiter des problèmes complexes issus de domaines divers nécessitant de très grosses puissances de calcul. Se posent alors des problèmes cruciaux de gestion de ressources multiples (les processeurs). Des solutions consitent à contrôler l'exécution des programmes parallèles/distribués en répartissant la charge de travail sur les différentes ressources disponibles. La difficulté est alors de construire de "bons" algorithmes de contrôle permettant d'optimiser des indices de performance comme le taux d'utilisation des ressources, le débit du système... Il faut remarquer que ces algorithmes de contrôle s'exécutent en environnement aléatoire. En effet, il n'est pas toujours possible de supposer connus, a priori, les temps de calcul ou les temps de communication. Une méthode pour pallier à cette difficulté est de construire des algorithmes de contrôle à décision ponctuelle, c'est-à-dire que ces algorithmes prennent leurs décision en fonction de l'état instantané du système ou en fonction d'une partie du passé du système.

Dans la pratique, le transfert de charge d'un processeur à un autre soulève de nombreux problèmes liés aux contraintes informatiques et relatifs aux objectifs à atteindre. Ces problèmes et les techniques utilisées pour le transfert sont par exemple exposés dans [2]. Cette problématique reste un domaine actif de la recherche et, selon le point de vue adopté, de nombreuses solutions algorithmiques ont été proposées. Des classifications de ces solutions sont périodiquement mises à jour comme celles de [9] et [6]. Cette distribution des tâches est donc gérée par le système d'exploitation mais celui-ci peut aussi être aidé par des indications données par le programmeur [29]. Lorsque plusieurs stratégies de partage ou d'équilibrage de charge sont envisageables, les spécifications quantitatives en terme d'indices de performances conditionne largement le choix de la stratégie. Les indices évoqués dans la littérature sont nombreux [7] et différentes études de modélisation ou des analyses comparatives de plusieurs algorithmes de partage de charge ont été proposées. Par exemple, des algorithmes de partage de charge sont comparés dans [15] en fonction d'indices de performance pertinents.

D'autres études analysent des modèles théoriques de partage de charge. Par exemple, la théorie des files d'attente a déjà été utilisée dans [10] avec des politiques de placement déterministes ou non déterministes pour optimiser la valeur d'un critère de performance dans le cas d'architectures homogènes. Une autre approche est proposée dans [8] pour résoudre le problème du placement d'une tâche particulière. Dans [32] est développée une méthode permettant d'obtenir, à faible coût, des bornes sur le débit moyen (throughput) et le temps moyen de réponse d'un système homogène séparable. Dans [12] est présentée une méthode de placement optimal dans un environnement statique. Une modélisation basée sur les processus de naissance et de mort est utilisée dans [13] pour comparer 2 politiques de transfert de charge selon que le transfert est effectué à l'initiative des processeurs surchargés ou à celle des processeurs sous-chargés. Deux études proches de celle-ci sont celles réalisées dans [27] et [26], mais elles diffèrent par le choix des indices de performance et les outils utilisés.

En particulier le cas des systèmes massivement parallèles n'est pas résolu.

Nous proposons ici un modèle stochastique de transfert de charge basé sur l'idée intuitive que l'évolution de la charge d'un processeur dépend uniquement de sa charge présente et du hasard et non de sa charge passée. L'objectif de ce modèle est de calculer les valeurs des indices de performance dans la situation où l'on met en œuvre une politique de transfert et dans la situation sans transfert afin d'estimer l'impact du transfert sur ces indices. Ces calculs permettent d'obtenir la valeur de ces indices pour des systèmes massivement parallèles (c'est-à-dire ayant un grand nombre de processeurs) et d'en déduire les fourchettes de valeur des paramètres pour lesquelles le transfert apporte en moyenne des améliorations significatives.

2 Modélisation du partage de charge

Pour analyser les systèmes informatiques se partageant la charge de travail, la phase de modélisation reste délicate. En effet, le modèle doit prendre en compte d'une part l'architecture du système et d'autre part les caractéristiques de l'application. Le protocole de partage de charge se situe à l'interface entre l'architecture et l'application.

2.1 Modélisation de l'architecture

Nous modélisons une architecture parallèle ou distribuée par un ensemble de n sites (processeurs) capables d'effectuer des tâches de calcul (voir figure 1). Les processeurs sont caractérisés d'une part par leur puissance de calcul (débit maximal en nombre de tâches de calcul effectuées par seconde) et d'autre part la capacité K en espace mémoire (nombre de tâches pouvant être stockées simultanément sur le site).

Les différents sites sont connectés par l'intermédiaire d'un réseau de communication par messages. Des protocoles de communications sont chargés de la collecte/diffusion d'information à l'intérieur du réseau. On supposera, sans restriction de généralité que le réseau est connexe et fiable (la gestion éventuelle de pertes de messages se faisant au niveau des protocoles de communication). Le réseau de communication et les algorithmes de routage associés conduisent à la notion de voisinage physique d'un site. Cette topologie de réseau peut être prise en compte par le protocole de partage de charge.

2.2 Modélisation de la charge induite par des applications

De manière générale, une application distribuée/parallèle est constituée par un ensemble de processus communiquant implantés sur une architecture distribuée/parallèle. Le grain de l'application sera défini comme étant l'unité élémentaire de quantité de calculs à exécuter sur un processeur. Dans cet article, nous appellerons tâche ce grain de calcul. Le choix de ce grain reste un problème difficile, souvent résolu empiriquement (voir les autres exposés). Nous supposerons dans ce document que le grain de l'application est fixé par l'utilisateur et ne varie pas au cours du temps (homogénéité temporelle). Dans la plupart des modèles basés sur les files d'attente le grain de calcul est modélisé par le client, les processeurs étant modélisés par des serveurs et l'espace mémoire par la capacité des files d'attente. La difficulté de modélisation réside

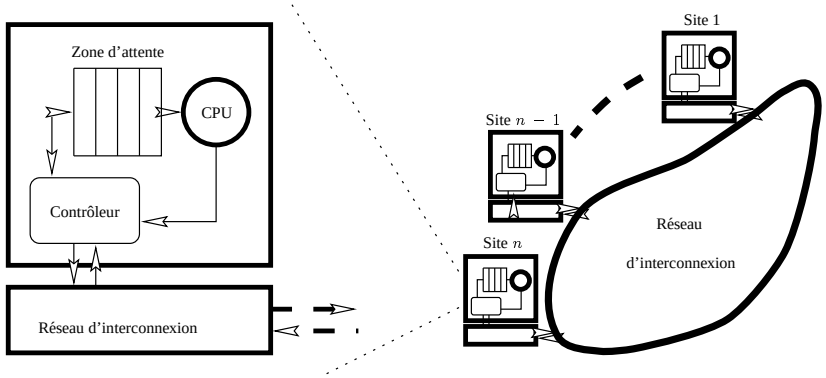


FIG. 1 – *Modèle d'un site et d'un réseau*

dans la prise en compte ou non des dépendances logiques entre les différentes tâches s'exécutant sur le système.

Un premier point de vue, considéré par exemple dans [7], consiste à modéliser l'application par un graphe de précédence. Ce graphe, connu au préalable ou construit dynamiquement, sera ensuite ordonnancé sur une architecture multi-processeurs. Les modèles correspondants en évaluation de performance sont construits à partir de réseaux de files d'attentes avec synchronisations [1, 30]. Il faut noter que la plupart de ces études supposent que l'allocation d'une tâche à un site est déterministe, c'est-à-dire ne prend pas en compte l'état du site sur lequel la tâche est exécutée [24, 22].

Un autre point de vue consiste à supposer que l'application que l'on modélise admet un comportement "moyen", c'est à dire que les contraintes de précédence sont noyées dans l'ensemble des tâches qui sont exécutées sur le système. Cela revient à représenter une application par un flot de tâches générées sur les différents sites. Dans ce cas, le système apparaît comme un réseau de files d'attente et le transfert des tâches dans ce réseau dépend alors de l'état de ces sites. C'est l'approche retenue dans la suite de cet article.

2.3 Hypothèses de nature statistique

Le modèle de l'application est donc décrit d'une part par le flot de tâches générées par chacun des sites et d'autre part par les temps d'exécution de ces tâches sur les sites où elles ont été placées dynamiquement.

Des études statistiques [19] ont montré que la distribution du temps d'exécution d'un processus, après avoir éliminé les processus courts, pouvait raisonnablement être approchée par une distribution de probabilité exponentielle, c'est-à-dire

$$\mathbb{P}(T_{exec} \leq t) = 1 - e^{-\mu t}.$$

La moyenne et l'écart-type de ce temps d'exécution d'une tâche sont tous deux égaux à $\frac{1}{\mu}$. On peut également remarquer que le choix d'une telle distribution minimise

la quantité d'information contenue dans le modèle sous la contrainte d'une valeur moyenne observée (pour ces techniques de modélisation voir par exemple [17]). La modélisation des flots de tâches générées sur chaque site se fera de la même manière : le temps séparant deux générations de tâches est modélisé par un temps aléatoire de loi exponentielle de paramètre λ. Le processus aléatoire qui compte le nombre de tâches générées au cours du temps s'appelle un processus de Poisson [23].

Lorsque cela sera nécessaire, en section 3, on modélisera les temps de transfert de tâches sur le réseau par d'autres variables aléatoires de loi exponentielle.

De telles hypothèses conduisent à des modèles markoviens (automates à états probabilisés), on en trouvera des exemples et des applications à des systèmes informatiques, dans les ouvrages [28] ou [23].

2.4 Modélisation du protocole de partage

Sans rentrer dans les détails d'une classification d'algorithmes de partage de charge, il faut noter que la plupart des protocoles de partage utilisent des informations sur létat global du système pour répartir la charge de travail sur les sites de calcul.

Dans le cas du placement dynamique de tâches sans migration, des évaluations de performances et des preuves d'optimalité ont été obtenues, voir par exemple [31]. Lorsque l'information de charge des sites n'est pas connue au moment de l'affectation de la tâche au site, on montre l'optimalité de politiques de type "round-robin" [21]. Lorsque l'information de charge des sites est connue, la politique d'allocation au site le moins chargé, sans migration entre les sites, est appelée "Join the Shortest Queue". Pour évaluer les performances de telles politiques on consultera [31, 16]. Une application au partage de charge et à l'implémentation d'heuristiques basées sur cette approche est développée par [14].

Le protocole de transfert instantané de la surcharge sur des sites sous-chargés conduit à un modèle de file d'attente multiserveurs (figure 2), ayant n serveurs, une capacité globale de nK tâches. Ce type de files d'attente ($M/M/n/nK$) s'analyse avec des techniques classiques de traitement des processus aléatoires de naissance et de mort [18].

Dans notre modélisation, lorsqu'il n'y a pas de file d'attente globale partagée par l'ensemble des sites, l'équilibrage de charge s'effectue par transfert de tâches entre sites voisins (voir figure 3). La différence avec une file d'attente $M/M/n/Kn$ tient au respect du comportement local de chaque site, un site de charge K n'accepte plus de nouvelles tâches et n'en génère plus.

Ce type de modélisation a été partiellement étudié par [5, 26]. Les difficultés mathématiques apparaissent d'une part lorsque l'on introduit les temps nécessaires au transfert des tâches, d'autre part lorsque le système devient massivement parallèle et enfin lorsque l'on tient compte de la topologie du réseau d'interconnection.

2.5 Indices de performance

Dans la majorité des modèles le choix des indices de performance à étudier dépend fortement de l'utilisateur et des spécifications quantitatives demandées. Les indices

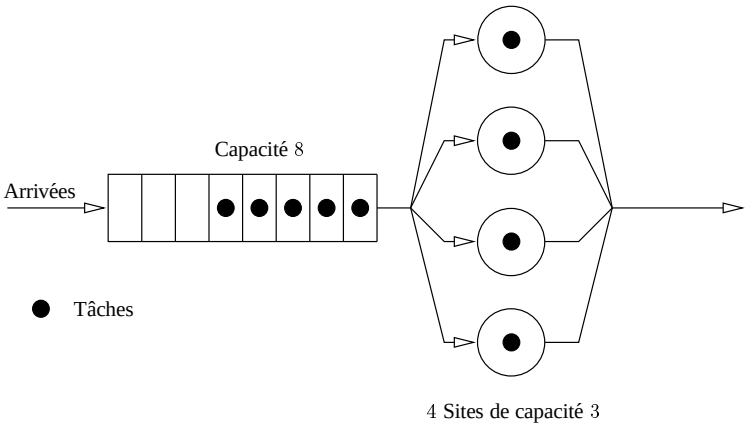


FIG. 2 – *File d'attente* $M/M/4/12$ *modélisant* 4 *sites se partageant un espace mémoire de* 12 *tâches.*

suivants apparaissent pertinents pour la plupart des problèmes posés.

Débit du système C'est-à-dire, le nombre moyen de tâches traitées par unité de temps $\overline{T}$. Cet indice permet de rendre compte du comportement global du système et est largement utilisé dans la littérature (throughput) [7, 32].

Saturation mémoire C'est-à-dire la probabilité P_{sat} que l'espace mémoire d'un site soit saturé (K tâches présentes) et que, par conséquent, toute nouvelle génération de tâche par l'application sur ce site soit rejetée par le système. Cette probabilité peut être interprétée comme une mesure de la dégradation du système. En régime non saturé, cette probabilité doit être minimisée pour garantir une utilisation optimale de l'ensemble des mémoires du système.

Charge de travail C'est le nombre moyen de tâches $\overline{N}$ présentes sur l'ensemble des sites. Aisément calculable pour les modèles étudiés dans la suite de cet article, ce nombre moyen est surtout nécessaire pour estimer le temps moyen de réponse d'une tâche grâce à la formule de Little.

Temps moyen de réponse $\overline{W}$ est le temps moyen écoulé entre l'instant où la tâche est générée par l'application, et l'instant où la tâche est terminée. Du point de vue de l'utilisateur, et en l'absence de toute autre spécification, cet indice doit être minimisé [18].

Mesure stationnaire de charge p_i est la probabilité pour un site donné d'avoir i tâches présentes (charge de i tâches). Pour un nombre donné de sites, ces probabilités représentent la proportion de sites ayant une charge de i. Ces probabilités

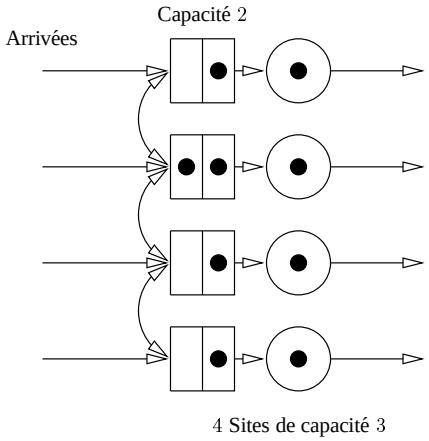


FIG. 3 – 4 *files d'attentes* $M/M/1/3$ *en parallèle avec partage de charge*

ne sont pas à proprement parler des indices de performance, mais leurs évolutions selon les paramètres, permettent de rendre compte du comportement global du système, et permettent de mieux comprendre comment opère le transfert de charge.

Ces différents indices de performance sont liés par les relations algébriques suivantes :

Proposition 1
Calculés en régime stationnaire, avec un temps moyen d'exécution de tâche égal à 1*,*

$$P_{sat} = p_K(\lambda) \quad , \tag{1}$$

$$\overline{T} = n\lambda(1 - p_K(\lambda)) \quad , \tag{2}$$

$$\overline{N} = n\left[\sum_{j=1}^{K} j p_j(\lambda)\right] \quad , \tag{3}$$

$$\overline{W} = 1 + \frac{\sum_{j=2}^{K}(j-1)p_j(\lambda)}{\lambda(1 - p_K(\lambda))} \quad . \tag{4}$$

La première et la troisième équation sont aisément obtenues. La deuxième donne le nombre moyen de tâches entrant dans le système par unité de temps. Un exemple de démonstration de cette formule pour un tel système est donné dans [2]. La dernière équation exprime le temps moyen de réponse d'une tâche comme la somme du temps moyen de service et du temps moyen d'attente. Les hypothèses de validité de la formule de Little, démontrée dans [31], sont vérifiées dans ce contexte.

3 Le transfert de charge entre 2 sites

3.1 Modélisation

Pour simplifier, dans un premier temps, notre approche, nous supposerons que notre système est constitué de 2 sites. Pour rendre compte des politiques de partage de charge basées sur des seuils de sous-charge, de charge normale et de surcharge, l'évolution de l'état d'un site sera donc modélisé par un graphe à 3 états, représenté en figure 4. La capacité K du site en nombre de tâches est ici égale à 2.

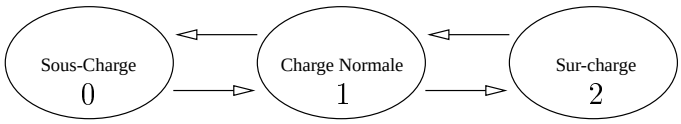


FIG. 4 – *Graphe d'état d'un site*

L'état global du système est alors modélisé par le couple des états (i, j) où i est l'état du site 1 et j l'état du site 2 (avec le codage 0= sous-charge, 1= charge normale et 2= surcharge).

Afin de prendre en compte le coût du transfert des tâches entre les sites, on introduit des états intermédiaires $(T, 0)$ et $(0, T)$; l'état T d'un site indique sa volonté de transférer. Pour prendre en compte l'évolution possible du site cible pendant le transfert (surcharge du site cible) deux stratégies peuvent être modélisés.

Sans interruption Le site initiateur ne contrôle pas le transfert qui s'effectue indépendamment de la surcharge du site cible. On introduit les états $(T, 1)$ et $(1, T)$ dans le modèle, traduisant l'effet "domino" du transfert.

Avec interruption Le site initiateur contrôle le transfert et reste surchargé, le transfert éventuel est interrompu.

Les graphes de transition des modèles de ces deux stratégies sont présentés dans la figure 5.
Les différents paramètres de ces graphes s'interprètent de la manière suivante :

λ taux de génération de nouvelles tâches par un site non surchargé. $1/\lambda$ est le temps moyen entre 2 générations de tâches.

μ taux de service des tâches. $1/\mu$ est le temps moyen d'exécution d'une tâche.

$1/x$ temps moyen de décision de transfert.

$1/y$ temps moyen de transfert.

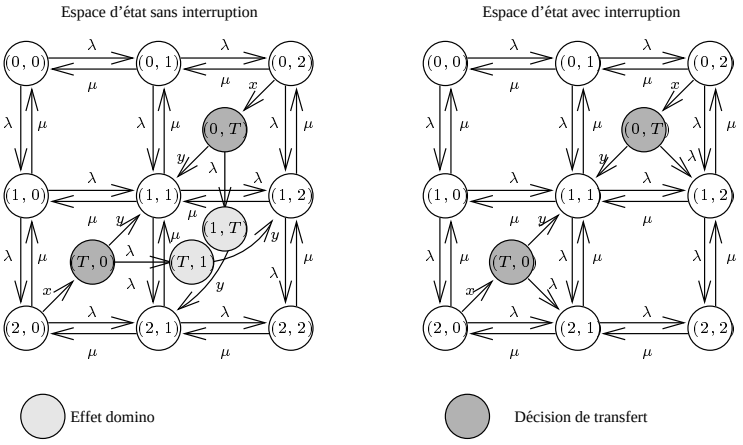


FIG. 5 – *Graphes d'états modélisant les* 2 *stratégies sans et avec interruption*

3.2 Résultats

Pour résoudre ce type de modèles markoviens, les outils d'agrégation forte [25] sont très utiles. Des méthodes d'agrégations et de décompositions peuvent être trouvées dans l'ouvrage de [11]. Ces méthodes s'appliquent dans la plupart des cas où l'on peut détecter des symétries dans le comportement du système. Dans notre cas les sites 1 et 2 jouent des rôles symétriques. Cette agrégation permet de réduire la taille de l'espace d'états et de rendre les calculs analytiques possibles. Cette méthode sera appliquée au modèle avec n sites présenté dans la section 4. Elle permet de faire explicitement l'évaluation numérique des indices de performance.

On supposera, sans restriction de généralité que le paramètre μ est égal à 1 (le temps moyen d'exécution d'une tâche est unitaire). Le temps de décision avant transfert est supposé négligeable devant le grain de calcul $(x >> 1)$.

Le résultat principal est l'existence de valeurs critiques. Pour un débit de tâche fixé λ, il existe une valeur du temps moyen de transfert $D_c(\lambda)$ en dessous de laquelle le transfert améliore effectivement les performances. On peut évaluer la valeur critique $D_c^{\overline{W}}(\lambda)$ dans les deux modèles optimisant le temps moyen de réponse $\overline{W}$. Pour la politique sans interruption on obtient

$$D_c^{\overline{W}}(\lambda) = \frac{1}{3\lambda}\left[\sqrt{1+\frac{3\lambda}{1+\lambda}}-1\right],$$

et pour la politique avec interruption on obtient

$$D_c^{\overline{W}}(\lambda) = \frac{1}{2(1+\lambda)} \quad .$$

Dans le cas d'un transfert instantané sans interruption, ce qui correspond à la situation idéale, le gain optimum 0.192 est obtenu pour une valeur $\lambda_{max} = 0.565$. Pour

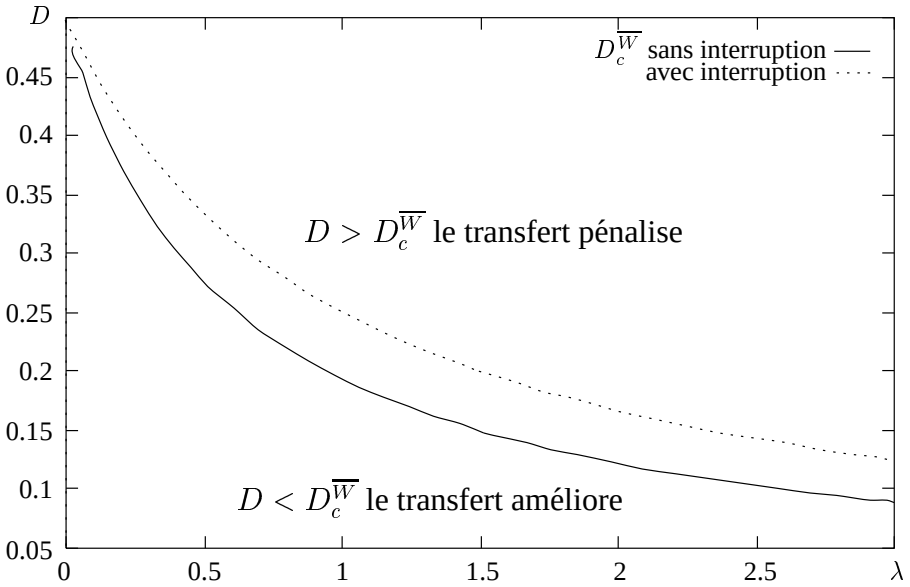


FIG. 6 – *Temps de transfert critique en fonction du débit d'entrée*

cette valeur, sans transfert $\overline{W} = 1.361$ et avec transfert $\overline{W} = 1.169$. En remarquant que le temps moyen d'exécution d'une tâche est égal à 1, ceci signifie que le temps moyen d'attente est divisé par 2 en autorisant le transfert.

En étudiant les autres indices de performance, on obtient d'autres valeurs critiques, ce qui montre que le choix d'un "operating point" dépend fortement de l'indice que l'on cherche à maximiser. Par exemple, si l'on étudie la probabilité de saturation P_{sat} on obtient la valeur critique

$$D_c^{P_{sat}}(\lambda) = \frac{1}{4\lambda}\left[\sqrt{1+\frac{8\lambda}{1+\lambda}}-1\right],$$

pour le modèle sans interruption et pour le modèle avec interruption

$$D_c^{P_{sat}}(\lambda) = \frac{1}{1+\lambda}.$$

Le gain maximum est 0.089 pour $\lambda_{max} = 0.767$ et dans ce cas, sans transfert $P_{sat} = 0.25$ et avec transfert $P_{sat} = 0.161$.

4 Système de n sites totalement interconnectés

Le modèle établi pour 2 sites se transférant la charge immédiatement avec un temps de transfert négligeable, se généralise à un nombre quelconque de sites totalement connectés. La technique de l'agrégation d'états, évoquée dans la section 3,

permet dans ce cas de réduire considérablement la complexité des graphes de transition, ce qui rend le modèle analytiquement résoluble. On observera de la même manière les phénomènes de valeurs critiques et de basculement de comportement. Les résultats exposés dans cette partie sont extraits de [3].

4.1 Modélisation

Pour améliorer la pertinence du modèle précédent, chaque site sera caractérisé par K niveaux de charge. La politique de partage sera telle que la différence de charge entre 2 sites ne pourra pas excéder 2 unités, ceci correspond à une politique d'équilibrage de charge car les tâches sont transférées dynamiquement à chaque changement d'état du système. On supposera de plus que les temps de transfert de tâches sont négligeables devant le temps d'exécution des tâches.

En reprenant les hypothèses statistiques précédentes de la section 3 (λ est le débit moyen d'arrivées de tâches de temps d'exécution moyen 1 sur chaque site), on peut remarquer une symétrie entre les systèmes sous-chargés $\lambda < 1$ et surchargés $\lambda > 1$:

Proposition 2 *Les probabilités stationnaires* $(p_0(\lambda), \cdots, p_K(\lambda))$ *vérifient les égalités suivantes*

$$p_i(\lambda) = p_{K-i}(\frac{1}{\lambda}) \quad 1 \leq i \leq K-1.$$

De plus les probabilités $p_0(\lambda)$ et $p_K(\lambda)$ satisfont l'équation de balance globale :

$$\lambda(1 - p_K(\lambda)) = 1 - p_0(\lambda).$$

4.2 Résultats numériques

De même que précédemment, mais avec encore plus d'efficacité, la technique d'agrégation d'états permet d'obtenir la valeur numérique des probabilités stationnaires des sites et d'en déduire les indices de performance. Par exemple, pour une capacité $K = 6$ et un nombre de sites égal à 32, on calcule la distribution stationnaire de charge sur un site, dont les valeurs sans et avec transfert sont données dans la table 1.

$\lambda = 0.8$	p_0	p_1	p_2	p_3	p_4	p_5	p_6	
sans	0.253	0.202	0.162	0.13	0.104	0.083	0.066	
avec	0.2	0.78	0.02	0.0	0.0	0.0	0.0	
	p_6	p_5	p_4	p_3	p_2	p_1	p_0	$\lambda = 1.25$

TAB. 1 – *Table des probabilités stationnaires pour des sites de capacité* $K = 6$ *avec* $\lambda = 0.8$ *et* $\lambda = 1.25$.

Ces probabilités stationnaires permettent également d'évaluer la charge du réseau de communication en mesurant le nombre moyen de tâches en transfert par unité de temps. Par exemple, la table 2 donne le nombre moyen de transferts par unité de temps pour une capacité $K = 2$.

λ	0.7	0.8	0.9	1	1.1	1.2	1.3
$n = 8$	6.09	6.32	6.6	6.93	7.29	7.68	8.12
$n = 16$	12.51	13.08	13.75	14.46	15.20	15.95	16.76
$n = 32$	25.23	26.63	28.24	29.80	31.22	32.59	34.01
$n = 64$	50.55	53.65	57.41	60.86	63.53	65.85	68.36
$n = 128$	101.12	107.50	115.87	123.54	128.32	132.17	136.85

TAB. 2 – *Nombre moyen de transferts par unité de temps pour $K = 2$ fonction de λ et de la taille n du système.*

A condition que le temps de transfert d'une tâche soit petit devant sa durée moyenne d'exécution, on remarque que le nombre moyen de transferts par unité de temps est de l'ordre de la taille du système. Par suite, la surcharge du réseau de communication due à la politique de transfert est acceptable. Les contentions dans le réseau n'influeront que très peu sur les performances globales.

4.3 Résultats asymptotiques

Le fait le plus marquant est que le passage de 8 à 32 sites n'améliore pas de manière significative les résultats sur les indices de performance $\overline{W}$, P_{sat} et p_0. C'est-à-dire qu'en ayant des contraintes de localité faible (par exemple chaque site est connecté à 8 autres sites) des performances similaires à celles d'un réseau totalement connecté seront obtenues. Lorsque le nombre de sites devient grand, les indices de performance convergent rapidement vers des valeurs limites. Par exemple, les courbes sur le temps de réponse moyen (figure 7) confirment ce fait.

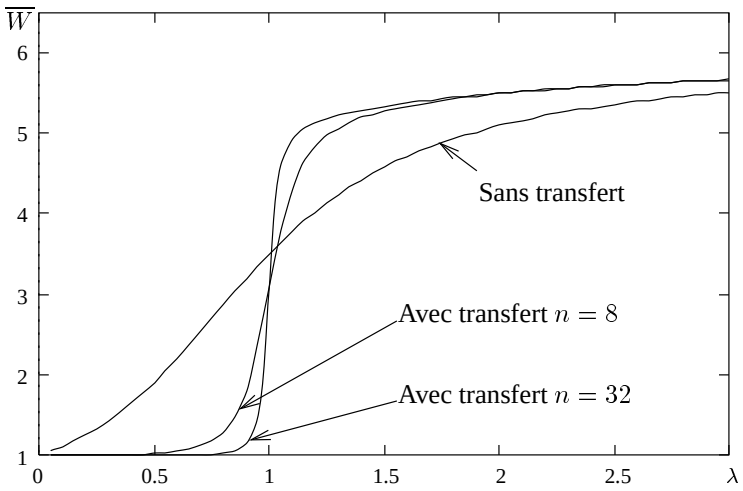


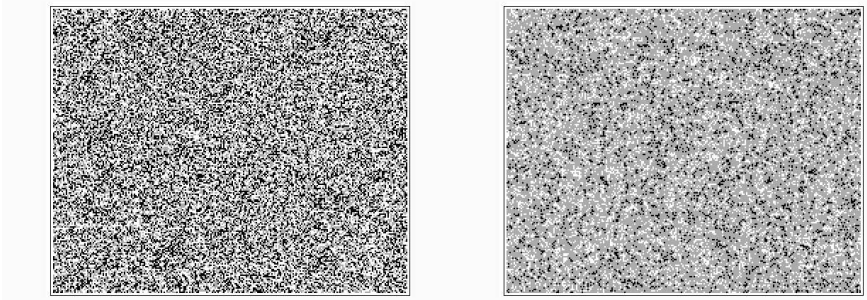
FIG. 7 – *Temps moyen de réponse $K = 6$ $n = 8,\ 32$ sans et avec transfert*

On peut également remarquer que la “transition de phase”, qui s’opère au voisinage de $\lambda = 1$, est d’autant plus brutale que n est grand. Le calcul analytique de ces asymptotiques pour tous les indices de performance est développé dans [3]. Cela permet de trouver de “bons points d’opération” optimisant des critères pertinents tels que la capacité K...

5 Vers de très grands réseaux à topologie régulière

A partir des résultats obtenus précédemment, le prolongement naturel est d’étudier le modèle de partage de charge en privilégiant la topologie du réseau d’interconnection. La théorie mathématique utilisée est celle des systèmes de particules interactives, pour laquelle la référence est [20]. Issue du domaine de la physique statistique, cette théorie permet d’envisager la modélisation et l’évaluation de très grands réseaux à topologie régulière (grilles, tores, hypercubes...). Le détail du modèle considéré et son traitement mathématique sont exposés dans [4]. La répartition des niveaux de charge sur chacun des processeurs est vue comme une configuration pour laquelle on définit une évolution markovienne. Cette évolution respecte la contrainte née de l’hypothèse de transfert immédiat, à savoir que la différence de niveau entre deux processeurs voisins ne dépasse jamais 1.

Pour ne pas surcharger cet article, nous présentons uniquement des résultats de simulations obtenus sur un ensemble de 50000 sites connectés en tore 250×200, avec $K = 2$. Dans la figure 8, l’image représente l’état du système avec le codage suivant : les sites sous-chargés sont en blanc, ceux en charge normale sont en gris et ceux surchargés sont en noir. Le transfert s’effectue dès qu’un site surchargé est voisin d’un site sous-chargé.



sans transfert avec transfert

FIG. 8 – *Exemple de configurations pour les politiques sans et avec transfert de charge,* $\lambda = 0.9$.

Cette simulation permet d’estimer les probabilités stationnaires. Par exemple, celles

obtenues pour $\lambda = 0.9$ sont présentées dans la table 3.

$\lambda = 0.9$	p_0	p_1	p_2
sans	0.365	0.335	0.299
avec	0.172	0.738	0.089

TAB. 3 – *Probabilités stationnaires pour K=2 associées aux configurations de la figure 8*

Ces résultats de simulation permettent d'estimer le gain dû à la politique de transfert sur le taux d'utilisation des sites, la charge moyenne d'un site et la probabilité de saturation mémoire. Ce type de simulation se généralise, sans trop de difficultés, à des systèmes ayant K niveaux de charge.

Il est important de noter que la modélisation de grands réseaux était non atteignable par les modèles markoviens classiques. Ceci est dû à l'explosion combinatoire de la taille de l'espace d'états. Cette nouvelle approche permet de réduire la complexité du modèle en utilisant la géométrie du réseau. De nouvelles méthodes analytiques couplées à des méthodes de simulation adaptées permettent actuellement d'envisager l'évaluation de politiques locales de partage de charge sur de très grands réseaux .

Références

[1] F. Baccelli and A. Makowski. Queuing models for systems with synchronization constraints. 77(1):138–161, January 1989.

[2] M. Béguin. Transfert de charge dans les machines parallèles. Technical Report 6, MAI-IMAG, Grenoble, October 1994.

[3] M. Béguin. Transfert de charge dans un réseau de processeurs totalement connectés. Technical Report 28, MAI-IMAG, Grenoble, June 1996.

[4] M. Béguin, L. Gray, and B. Ycart. The Load Transfer Model. Technical Report 31, MAI-IMAG, Grenoble, September 1996.

[5] M. Béguin, J.M. Vincent, and B. Ycart. Markovian models for load transferring. Technical Report 14, MAI-IMAG, Grenoble, May 1995.

[6] G. Bernard, D. Steve, and M. Simatic. Placement et migration de processus répartis faiblements couplés. *TSI*, 10(5):375–392, September 1991.

[7] D.P. Bertzekas and J.N. Tsitsiklis. *Parallel and distributed computation*. Prentice-Hall, 1989.

[8] R. Bryant and J. Agre. A queuing network approach to the module allocation problem in distributed systems. *Performance Evaluation Review*, 10(3):191–204, 1981.

[9] T.L. Casavant and J.G. Kuhl. A taxonomy of scheduling in general purpose distributed computing systems. *IEEE Trans. on Soft. Eng.*, 14(2):141–154, february 1988.

[10] Y.C. Chow and W. Kohler. Models for dynamic Load Balancing in a heterogeneous Multi-Processors System. *IEEE Trans. on Comp.*, c-28(5):334–361, 1979.

[11] P.J. Courtois. *Decomposability: Queueing and computer system applications.* Academic-Press, 1977.

[12] E. De Souza E Silva and M. Gerla. Queueing Networks Models for Load Balancing in Distributed Systems. *Journal of Parallel and Distributed Computing*, 12(1):24–38, 1991.

[13] D.L. Eager, E.D. Lazowska, and J. Zahorjan. A comparison of Receiver-Initiated and Sender-Initiated Adaptive Load Sharing. *Performance evaluation*, 6(1):53–58, 1986.

[14] D.J. Evans and W.U.N. Butt. Dynamic Load Balancing using Task-transfer Probabilities. *Parallel Computing*, 19:897–916, 1993.

[15] H. Guyennet, B. Herrman, L. Philippe, and F. Spies. A performance study of dynamic load balancing algorithms for computers. In *Int. Conference on Massively Parallel Computing*, Ischia, Italy, May 1994.

[16] S. Halfin. The Shortest Queue Problem. *J. Appl. Prob.*, 22:865–878, 1985.

[17] J.N. Kappur and H.K. Kesavan. *Entropy Optimization Principles with Applications*. Academic Press, 1992.

[18] L. Kleinrock. *Queuing systems : theory*, volume 1. J. Wiley & Sons, 1975.

[19] W. Leland and T. Ott. Load balancing heuristics and process behaviour. *ACM Perf. Eval Rev.*, 14:54–59, 1986.

[20] T.M. Liggett. *Interacting Particule Systems*. Springer-Verlag, 1985.

[21] Z. Liu and D. Towsley. Optimality of the Round-Robin Routing Policy. *J. Appl. Probab.*, 31(2):466–475, 1994.

[22] S. Madala and J.B. Sinclair. Performance of Synchronous Parallel Algorithms with Regular Structures. *IEEE Trans. on Parallel and Distributed Systems*, 2(1):105–116, 1991.

[23] R. Nelson. *Probability, stochastic processes and queueing theory*. Springer-Verlag, 1995.

[24] S. Rajsbaum and M. Sidi. On the Performance of Synchronized Programs in Distributed Networks with Random Processing Times and Transmission Delays. *IEEE Trans. on Parallel and Distributed Systems.*, 5(9):939–950, September 1994.

[25] M. Rosenblatt. Functions of a Markov process that are Markovian. *Journal of Mathematics and Mechanics*, 8(4):585–596, 1959.

[26] F. Spies. Modeling of optimal load balancing strategy using queueing theory. *Microprocessing and Microprogramming*, 41(8/9):555–, April 1996.

[27] F. Spies, H. Guyennet, and M. Trehel. Modélisation et simulation de la répartition de charge dynamique, à l'aide des réseaux de files d'attente. Technical report, Laboratoire d'Informatique, Besançon, France, 1994.

[28] K.S. Trivedi. *Probability and Statistics with Reliability, Queuing and Computer Science Applications*. Prentice-Hall, 1982.

[29] A. Vermeerbergen. Les poly-algorithmes et la prévision des coûts pour une expression portable et extensible du parallélisme. In Bougé, editor, *Renpar'6*, ENS-LYON, June 1994.

[30] J-M. Vincent. Stability condition of a service system with precedence constraints between tasks. *Performance Evaluation*, 12(1):61–66, 1991.

[31] J. Walrand. *Introduction to Queuing Networks*. Prentice-Hall, 1989.

[32] J. Zahorjan, K.C. Sevcik, D.L. Eager, and B. Galler. Balanced Job Analysis of Queuing Networks. *Communications of the ACM*, 25(2):134–141, 1982.