Année 1997

THÈSE

Présentée par Maryse BÉGUIN

Pour obtenir le titre de

Docteur en Mathématiques Appliquées

DE L'UNIVERSITÉ JOSEPH FOURIER-GRENOBLE 1

(arrêtés ministériels du 5 juillet 1984 et du 30 mars 1992)

MODÈLES MARKOVIENS DE TRANSFERT DE CHARGE DANS LES RÉSEAUX INFORMATIQUES

DATE DE SOUTENANCE : 20 octobre 1997

COMPOSITION DU JURY :

Brigitte	PLATEAU	Présidente
Philippe	ROBERT	Rapporteur
Michel	TRÉHEL	Rapporteur
Jean-Marc	VINCENT	Examinateur
Bernard	YCART	Directeur de thèse

Thèse préparée au sein du Laboratoire LMC de l'Institut IMAG de Grenoble

Remerciements

Toute thèse qui réussit la longue traversée jusqu'à la soutenance a le plus souvent été élaborée grâce à la collaboration d'esprits et de mains autres que ceux de son seul auteur. Celle présentée ici, après de longues années d'inexpérience de la recherche et d'éloignement relatif des Mathématiques, accumule plus de dettes que la plupart des autres. Je tiens donc à remercier ici un certain nombre de personnes qui m'ont aidée et soutenue pour conduire ce travail à bon port contre vents et marées :

Bernard Ycart, tout d'abord, qui m'a suggéré l'idée de faire une thèse et m'en a proposé le sujet. J'ai pu mesurer au cours de ces années de recherche sa compétence et son professionnalisme. Ses cours, ses séminaires, sa curiosité et ses exigences restent pour moi un exemple.

Jean-Marc Vincent dont je ne saurais assez dire combien toutes les questions et les commentaires furent, en toute occasion, aussi pertinents qu'encourageants. Il a constamment dirigé nos résultats vers une interprétation explicitée en termes informatiques ouvrant ainsi largement le débat et les possibles applications des modèles proposés.

Olivier Francois dont les suggestions et les avis toujours aussi opportuns que généreux se sont révélés singulièrement efficaces. Nous partageons actuellement le même bureau et le voir réfléchir ou exposer des Mathématiques est un apprentissage permanent doublé d'un réel plaisir.

Monsieur Gray dont les conseils dans le dernier chapitre furent essentiels à l'achèvement de cette partie.

Marie-Christine Roubaud dont le précieux réconfort à des moments difficiles m'a rendu son amitié très chère.

Jean-Louis Soler qui m'a généreusement accueillie dans son laboratoire, voire dans son bureau pour quelques mois, et m'a encouragée à postuler le poste de PRAG à l'ENSIMAG que j'occupe actuellement.

Philippe Robert et Michel Tréhel qui ont accepté d'être les rapporteurs de cette thèse, tâche dont je mesure aisément l'ampleur, et dont j'ai pu apprécier les suggestions ou les remarques.

Brigitte Plateau qui me fait l'honneur de présider ce jury malgré des journées surchargées, et dont les encouragements ont été un précieux soutien dès le début de ce travail. Puis-je lui dire aussi qu'à mes remerciements d'universitaire, s'ajoutent mes remerciements de chercheuse fière de voir la présidence de son jury de thèse confiée à une femme, dans ce domaine où précisément le rôle des femmes reste encore statistiquement trop discret.

Ma profonde gratitude va également à l'ancienne et à la nouvelle équipe de thésards et plus particulièrement à Florence Forbes, Catherine Trottier, Mhammed El Aroui et Mohammed Bouatou, sans qui les nombreuses heures en salle machine auraient parfois été longues et monotones. Une mention spéciale pour Lamine (alias Mould) dont j'ai apprécié l'enthousiasme, la constante bonne humeur et les conseils appropriés.

Je voudrais encore exprimer ma reconnaissance sincère à tous ces proches, parents, amies ou amis qui, embarqués avec moi sur " le seul bateau qui ait tenu le coup" (comme le chante Brassens), ne m'ont pas tenu rigueur de mes fréquentes escapades et qui, attendant patiemment la fin de cette thèse, m'ont conservé leur amitié. C'est un baume au cœur extrêmement précieux lors de cette aventure souvent bien solitaire.

Enfin, ces remerciements seraient incomplets si je n'y associais mes "maîtres" d'antan, ceux là mêmes qui m'ont donné le goût, le respect des Mathématiques et, par dessus tout la vocation de les enseigner : Mme Potaux, Mme De Saintilan, Mr Bonvallet, Mr Minois, Mr Grappy, hélas disparu mais auquel je tiens à rendre hommage, Mr Chevalley, Mme Chauve, Mme El Karoui, Mr Revuz.

Je dédie cette thèse à Adam Képes, Agnès Ullmann, Yvonne Joyeux et Simone Béguin.

Table des matières

Chapitre 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. Ces processeurs sont reliés entre eux par des réseaux d'inter-connexions grâce auxquels ils peuvent communiquer, acheminer des messages, des données, des applications. Une zone mémoire est attachée à un processeur et l'ensemble constitué du processeur, de cette zone mémoire, et des possibilités de communication/synchronisation inter-processeurs forme un réseau de sites informatiques sur lequel vont s'exécuter des applications dites de grande taille. Du point de vue de ces applications, le problème crucial à résoudre est la gestion et le partage de ces ressources multiples. Cette gestion suppose la mise en œuvre d'un système d'exploitation "parallèle" permettant de contrôler l'exécution d'applications parallèles/distribuées en répartissant la charge de travail sur les différents sites. Ces répartitions impliquent des transferts de tâches d'un site à un autre, de sites dits "surchargés " à des sites dits "sous-chargés", et ces transferts soulèvent de nombreux problèmes liés aux contraintes technologiques et informatiques et relatifs aux objectifs que souhaitent atteindre les programmeurs ou les utilisateurs des logiciels. Ces objectifs sont quantifiés par des indices de performance, et selon le choix des indices de performance que l'on souhaite ou non optimiser, différentes politiques de partage de charge peuvent être testées et comparées. Nous proposons ici la modélisation et l'évaluation d'algorithmes de partage de charge basées sur une évolution markovienne de la configuration des charges de l'ensemble des processeurs.

Le but principal de cette étude est de quantifier numériquement l'incidence de la mise en œuvre d'une politique de transfert sur la valeur de certains indices de performance, et d'en déduire la pertinence ou non de la décision d'un transfert en fonction des valeurs des paramètres du modèle.

Dans le premier chapitre nous analysons les différentes situations de transfert ainsi que les différentes stratégies et modalités de ces transferts. La mise en place d'un système d'exploitation "parallèle" pose des problèmes inhérents à la technologie et à l'informatique, que nous détaillons.

Les différents indices de performance rencontrés dans la littérature (voir [12]) sont exposés. De nombreuses études ont été faites dans cette direction de recherche, et nous nous sommes appuyés sur les études de classifications (voir [21, 11]) ou des recherches plus récentes (voir [34, 46, 85]) pour analyser les résultats antérieurs. Nous détaillons et justifions le choix de notre modélisation stochastique du transfert de charge, et nous donnons une notation particulière aux indices de performance que nous retenons comme pertinents pour notre étude.
Dans le chapitre deux, nous développons l'étude obtenue lorsque nous restreignons notre modèle à un réseau composé de deux sites. Le résultat principal est l'obtention de valeurs critiques permettant de décider des plages de valeurs des paramètres pour lesquelles la politique de transfert apparaît pertinente, et d'autres ou elle s'avère non utile. Ces valeurs dépendent de l'indice de performance considéré. En particulier, l'incidence de certains algorithmes de transfert sur le temps de réponse (moyenne et variance) est détaillée et analysée.
Au chapitre trois, l'étude d'un cas particulier où le temps des communications et du transfert sont négligés devant les temps de calcul, est généralisé pour un nombre quelconque de sites homogènes de capacité mémoire finie, tous inter-connectés. Il s'agit donc d'un algorithme d'équilibrage de charge pour un réseau totalement connecté. Les résultats principaux sont de deux sortes :

Le résultat principal est le théorème de convergence qui permet de prévoir le comportement asymptotique des systèmes massivement parallèles. La comparaison des résultats théoriques obtenus avec les résultats obtenus pour des valeurs plausibles du nombre de sites (32,64,128) montre la pertinence et l'intérêt de ces valeurs asymptotiques.

D'autre part, cette étude permet de calculer des bornes supérieures sur les bénéfices éventuels que l'on peut attendre d'un réel transfert. Elle permet aussi d'envisager d'autres calculs d'optimisation comme ceux du taux d'arrivée quand l'objectif est de garantir un débit global constant, ou un dimensionnement optimal de la capacité mémoire selon un critère donné.

Les outils mathématiques requis pour la compréhension de ces deux derniers chapitres sont issus de la théorie des processus de Markov (voir [24]), et font essentiellement appel aux notions de processus agrégés (voir [78]), de processus de naissance et de mort (voir [5]), de processus de Poisson en environnement Markovien (voir [42]), et de la théorie des files d'attente (voir [54, 25, 99]).
Dans le chapitre quatre, nous prenons en compte la notion de voisinage entre les sites, et nous étendons le principe de l'algorithme de transfert étudié au chapitre 4 à des réseaux dont la topologie est régulière. L'hypothèse essentielle de cet algorithme est celle d'un transfert immédiat et instantané dès que la différence de charge entre deux sites voisins dépasse strictement la valeur 1. Le modèle obtenu est résumé sous l'appellation LTM.

L'explosion combinatoire de l'espace d'états pour des réseaux de grande taille conduit à l'utilisation d'outils mathématiques différents de ceux utilisés dans les chapitres précédents. Ceux-là dérivent de la théorie des systèmes de particules (voir [58, 31]) et utilisent, entre autres, les outils de comparaisons stochastiques (voir [58, 41, 40]). Notre résultat principal est le théorème d'ergodicité : le modèle LTM sur $\mathbb{Z}^d$ est ergodique et converge à vitesse exponentielle vers son régime stationnaire. Ce résultat justifie le recours aux techniques de simulation que nous utilisons (voir [102]) pour l'estimation de valeurs obtenues en régime stationnaire. Compte tenu de l'heuristique dite "du champ moyen" utilisée avec succès dans des domaines différents et voisins du nôtre (voir [2, 15, 91, 98, 33, 49]), nous avons également calculé ces valeurs par cette méthode. Pour certaines valeurs des paramètres de notre modélisation, les valeurs des probabilités stationnaires pour un site d'avoir une charge égale à i obtenues par ces deux méthodes sont extrêmement proches. Une comparaison et une analyse de ces résultats sont proposées. Enfin, l'incidence de cette politique de transfert sur la valeur des indices de performance est étudiée.

La conclusion récapitule les résultats essentiels d'ordre pratique qui découlent de notre étude et quelques perspectives possibles sont évoquées.

Chapitre 2

Transfert de charge dans les réseaux informatiques

2.1 Analyse

2.1.1 Situations de transfert

Le problème d'un transfert de charge en informatique se pose dès que plusieurs processeurs, reliés entre eux par un réseau de communication, sont utilisés par un ou plusieurs utilisateurs. Un processeur peut être considéré comme un site sur lequel s'exécutent des applications. L'origine de ces applications peut être diverse : édition, compilation, exécution d'une série d'instructions ... Chaque processeur est géré par un système d'exploitation qui génère en fonction de l'application un bloc d'instructions machines. Ce bloc sera appelé selon la littérature processus ou tâche informatique. La granularité de cette tâche peut certes être variable, et va dépendre de caractéristiques liées entre autres à la technologie. Dans la présente étude, une unité de tâche est supposée définie (nous y reviendrons au chapitre 2.5) et le problème de transfert peut alors schématiquement être présenté de la façon suivante.

Des tâches sont générées par un processeur, et celles-ci attendent, comme dans une file d'attente, que le processeur soit libre pour pouvoir être traitées.
Une difficulté importante consiste alors à évaluer la mesure de la charge d'un processeur. Cette mesure doit rendre compte de la quantité de travail que va devoir effectuer un processeur. Or cette quantité est difficile à anticiper, et nous verrons en section 2.2 les difficultés liées à l'évaluation de cet indice.
Néanmoins, et intuitivement, plus cet indice est grand et plus le processeur sera dit chargé. Il semble alors naturel d'imaginer les situations où certains processeurs sont surchargés tandis que d'autres sont sous-chargés. Grâce au réseau de communication, on peut envisager de pallier de telles situations en transférant une partie de la charge d'un processeur surchargé à

un processeur sous-chargé. Cette politique de transfert suppose la mise en oeuvre, au-dessus du système d'exploitation gérant chaque processeur (en temps partagé ou non), d'un système que l'on peut appeler "système d'ordonnancement", chargé de la répartition des tâches sur les différents processeurs.
Nous retiendrons l'appellation "système" pour désigner l'ensemble des processeurs reliés entre eux par un réseau de communication. Le système d'ordonnancement vise à améliorer les performances du système, ces performances pouvant être évaluées par différents indices que nous examinerons en section2.5.5 (*cf.* [11, 55]).

Pour effectuer les transferts, le système d'ordonnancement utilise des mécanismes de transfert de charge dont les protocoles sont fortement liés au contexte informatique. On peut distinguer principalement deux cas de figure.

La première situation possible est celle d'une machine parallèle où les différents processeurs sont homogènes, gérés par un même système d'exploitation. Les protocoles de transfert peuvent être assez simples et les temps de communication entre les différents processeurs sont faibles en regard des temps de calcul. Choisir le lieu d'exécution d'une tâche consiste à choisir un processeur parmi les processeurs sous-chargés. Certaines études ont été faites dans ce domaine, par exemple celle de Nicol et Saltz (*cf.* [73]), mais la façon dont sont utilisés les processeurs et les tâches induit des études très différentes. Par exemple, lorsque la topologie du système et le nombre de processeurs sont connus, un algorithme d'équilibrage de charge dans une situation particulière est étudié dans [13]. Pour des machines parallèles, les différents processeurs sont considérés comme une ressource globale et une application va générer des tâches qui vont utiliser tout ou partie de l'ensemble des processeurs du réseau pour exécuter des calculs en parallèle. Afin d'améliorer le speed-up de l'application, le système d'ordonnancement peut décider de transferts et cette décision doit être prise en cours d'exécution de l'application. Quand la topologie, le nombre de processeurs ou les systèmes d'exploitation de la machine parallèle peuvent changer avec le temps, l'exécution d'une même application change d'une fois sur l'autre, et la difficulté est de trouver une fonction de régulation de la charge qui soit néanmoins stable d'une exécution à une autre et portable d'une machine à une autre. Les décisions de transfert sont prises dynamiquement au cours de l'exécution de l'application.

La deuxième situation est celle des réseaux hétérogènes ou non. Dans le cas des réseaux non hétérogènes (systèmes faiblement couplés), les différents processeurs sont à des distances plus ou moins proches, ils n'ont pas forcément les mêmes carac-

téristiques matérielles, mais ils sont tous gérés par un même système d'exploitation. Les protocoles de transfert restent relativement simples à mettre en place, mais les temps de communication peuvent être plus ou moins longs selon la topologie du réseau et les protocoles utilisés. Cette situation est celle qui a été la plus étudiée dans la littérature (*cf.* [11]). La situation diffère par rapport à la machine parallèle car à chaque instant, une tâche utilise un processeur, et du point de vue du programmeur, les processeurs sont considérés comme différentes ressources grâce auxquelles s'exécutent différentes applications.

Dans le cas des systèmes hétérogènes, les caractéristiques des processeurs peuvent être différentes ainsi que les systèmes d'exploitation qui les gèrent. Pour de tels systèmes, les protocoles de transfert peuvent devenir compliqués et les communications sont globalement plus longues. Du fait de ces difficultés, ces systèmes ont été relativement peu étudiés dans la littérature. Citons néanmoins les études [101] et celles plus récentes [106, 66].

2.1.2 Partage, équilibrage de charge, placement et migration

La mise en place d'une politique de transfert entre processeurs nécessite le choix des instants où un tel transfert peut être envisagé. En effet, celui-ci peut être envisagé chaque fois qu'un événement se produit dans le système, c'est-à-dire à chaque fois qu'une tâche est générée et à chaque fois qu'une tâche est complètement traitée, ou bien selon une périodicité donnée.
Les instants possibles du transfert étant choisis, la décision du transfert suppose elle aussi le choix d'une politique de transfert. Deux politiques de transfert se retrouvent dans la littérature sous les termes de "partage de charge" et "d'équilibrage de charge". Les définitions exactes de ces politiques peuvent varier selon les auteurs, et nous retiendrons celles données dans [11].

Selon la politique de "partage de charge", un tranfert est décidé si la charge d'un processeur dépasse un certain seuil. Il n'y a donc pas de transfert systématique aux instants possibles puisqu'il faut que l'un des sites ait une charge supérieure à un seuil donné. Différents seuils peuvent d'ailleurs être utilisés pour classer les sites soit parmi les site sous-chargés, soit parmi les sites en charge normale, soit parmi les sites surchargés. Ainsi, cette politique tient compte des difficultés de transfert, mais permet d'éviter les phases de congestion où certains processeurs sont clairement surchargés tandis que d'autres sont sous-chargés. Des études sur cette politique de partage de charge dans certaines situations ont par exemple été faites par Wang et Morris dans [100], Zhou et Ferrari dans [104] ou Zhou dans [105].

Selon la politique "d'équilibrage de charge", la charge globale du système est répartie le plus équitablement possible entre tous les sites. Cette politique de transfert permet de niveler la

charge globale sur l'ensemble des processeurs et vise naturellement à améliorer les performances globales du système. Cette politique a par exemple été étudiée pour les systèmes distribués sous certaines conditions par Chou et Abraham dans [22].

Une fois le transfert décidé, sa mise en place nécessite le déplacement de blocs d'instructions d'un processeur à un autre. On peut là encore souligner deux distinctions possibles, selon que l'on effectue une migration de tâche, ou un placement de tâche.
La migration de tâche s'effectue sur une tâche en cours d'exécution et consiste à interrompre cette exécution pour aller finir le traitement de la tâche sur un autre site (immédiatement ou ultérieurement). C'est une opération difficile à mettre en oeuvre du point de vue du ou des systèmes d'exploitation, mais la politique autorisant les migrations a été étudiée dans la littérature (*cf.* [1]).
Le placement de tâche s'effectue lorsqu'une tâche vient d'être générée par un processeur, et est transférée sur un autre, ou lorsqu'une tâche attend pour être exécutée dans la file d'un processeur et est aussi transférée sur un autre.

Dans les deux cas, la tâche transférée sur le processeur cible pourra être traitée immédiatement ou ultérieurement. Entre l'instant où elle est générée et l'instant où elle est complètement traitée, une tâche peut donc être migrée ou placée sur plusieurs sites différents.

2.2 Problèmes

Néanmoins, l'analyse précédente ne permet pas de mesurer l'ampleur des problèmes rencontrés dans la pratique informatique lorsque l'on souhaite mettre en place une politique d'ordonnancement des tâches.
Quelle que soit la politique de transfert choisie, celle-ci suppose connues certaines informations dont le nombre et la précision quantitative sont très variables.
Les deux difficultés essentielles sont l'évaluation de la charge d'un processeur (*cf.* [37]) et la prise en compte des délais de communication entre les processeurs (*cf.* [22]).

2.2.1 Charge d'un processeur

L'évaluation correcte de la charge d'un processeur est plus complexe qu'il n'y paraît, et dans la littérature, les auteurs utilisent différents indicateurs.
La plupart des auteurs utilisent comme mesure de la charge d'un processeur la longueur de la file d'attente de l'unité cpu (*cf.* [34]). Mais cet indicateur ne prend pas en compte la durée variable du temps d'exécution des tâches, et pour cette raison, certains auteurs se réfèrent à la longueur moyenne de la file de l'unité cpu (*cf.* [4]). Ces indicateurs ne tiennent pas compte des

imperatifs associés aux instructions dites d'entrées/sorties sur les processeurs. Ces instructions ajoutent des contraintes et ralentissent le plus souvent la durée de traitement d'une tâche. Pour en tenir compte, certains auteurs préconisent comme indicateur de charge la longueur de la file cpu plus celle des entrées/sorties (*cf.* [37]).
Dans la situation d'un système hétérogène, les indicateurs de la charge de chaque processeur doivent aussi tenir compte des différences de vitesse des unités cpu des différents processeurs.
Dans la situation exceptionnelle où le temps d'exécution des tâches est connu à l'avance, le meilleur indicateur de charge est le temps d'exécution cumulé. Cette situation a été étudiée par exemple par Shin et Hou dans [81].

La politique de partage de charge suppose l'existence de seuils à partir desquels un processeur est considéré comme sous-chargé, en charge normale ou surchargé. Certains algorithmes utilisent même des politiques dites de doubles seuils et on peut imaginer plusieurs seuils de charge. Le choix de ces seuils reste déterminant et plusieurs politiques ont déjà pu être testées. En outre, ces seuils peuvent être statiques au cours du temps (*cf.* [34, 38]), ou dynamiques c'est-à-dire évoluant au cours du temps (*cf.* [75]).

2.2.2 Contention réseau

La prise en compte des délais de communication n'est pas toujours aisée non plus. En effet, les tractations entre processeurs pour échanger des informations via le réseau de communication et décider d'un transfert, sont plus ou moins longues selon des caractéristiques liées à la topologie du système et aux protocoles de communication utilisés. Ces caractéristiques sont fortement liées à la technologie, et évoluent avec elle. Elles sont donc plus ou moins bien maîtrisées sauf sur des machines multi-processeurs dont on connaît la topologie quand celle-ci reste statique (*cf.* [13]).
Dans le cas de machines parallèles, la charge globale du système est seulement estimée car la connaissance exacte de la charge de chaque processeur nécessiterait une synchronisation globale qui ralentirait considérablement le temps d'exécution de l'application (cf. [74]). En outre, ces communications représentent un coût financier qui n'est pas à négliger dans l'évaluation des performances. Ces coûts dépendent du matériel utilisé et des types de réseaux concernés (*cf.* [22]).

2.2.3 Filtres, transparence vis à vis de l'utilisateur, comportements précédents, effet domino

Compte tenu des temps de communication entre les processeurs un problème essentiel pour décider de l'opportunité d'un transfert, est lié à la durée variable du temps d'exécution des tâches. En effet, pour qu'un transfert soit justifié pour une tâche donnée, il faut que la durée de vie de celle-ci soit suffisamment longue et au moins supérieure au temps de communication.

Or des études montrent que ce n'est généralement pas le cas pour au moins 66% des tâches (*cf.* [90, 20]). Il semble donc souhaitable d'installer des filtres visant à éliminer, parmi les tâches candidates au transfert, celles ayant une durée de vie trop courte. Plusieurs filtres ont ainsi été proposés dans la littérature (*cf.* [20, 105, 90]).

L'utilisation de tels filtres pose le problème de la transparence ou non du transfert vis-à-vis de l'utilisateur. La non transparence permet une participation de l'utilisateur qui pourra, le cas échéant, donner des informations sur les tâches qui vont être générées par le système d'exploitation et indiquer si celles-ci sont candidates ou non au transfert. Cette méthode est très intéressante, mais elle n'est performante que si l'utilisateur est conscient des problèmes de coût liés au transfert. Elle permet, par contre, d'éliminer des candidates au transfert comme les tâches interactives pour lesquelles le transfert s'avère délicat, et le plus souvent non souhaitable (*cf.* [64, 79]).

La prise en considération d'un utilisateur donné par le système d'ordonnancement pose le problème du respect du caractère personnel d'un ordinateur mis sur un réseau. Un utilisateur peut en effet laisser des tâches provenant de sites distants s'exécuter sur l'unité cpu de son ou ses processeurs, et vouloir à un instant donné retrouver la priorité sur son ordinateur (*cf.* [71]). La prise en compte de cette priorité est une difficulté supplémentaire dans la mise en œuvre du système d'ordonnancement.

Le système d'ordonnancement peut aussi tenir compte des comportements précédents du système et obtenir ainsi des politiques de transfert de plus en plus performantes. Néanmoins, cet ajustement en fonction des comportements précédents nécessite de grosses capacités mémoires pour conserver l'historique de ces comportements (*cf.* [86]), et un tel investissement n'est utile que si le système traite fréquemment les mêmes tâches.

Enfin, la politique de transfert doit minimiser les situations d'inversion des rôles. Cette inversion a lieu lorsqu'un processeur A transfère vers un processeur B moins chargé qui à son tour, et avant la mise en place effective du transfert, peut devenir à son tour surchargé. Ainsi, il se peut que la tâche qui lui a été transférée le soit à nouveau. Pour éviter cet effet dit "effet domino", il faudrait tenir compte de ce que sera la charge d'un processeur, car le temps de réponse du processeur, c'est-à-dire sa capacité à traiter rapidement une tâche, dépend de sa charge future et non de sa charge actuelle (*cf.* [11]).

2.3 Indices de performance

L'objectif d'une politique de répartition des tâches est d'obtenir une amélioration des performances du système, mais les mesures de ces performances peuvent être multiples. En effet, un utilisateur donné peut souhaiter une réponse rapide aux tâches qu'il soumet indépendamment

des autres utilisateurs voire au détriment de ceux-ci, tandis qu'un administrateur du système souhaite la meilleure utilisation possible de l'ensemble des ressources, au détriment parfois de quelques utilisateurs. La politique de transfert induit des communications qui peuvent devenir onéreuses, et dans certaines situations, le gérant par exemple peut s'intéresser aux coûts financiers des différentes tractations dues à cette politique. Enfin, d'autres aspects très importants comme la fiabilité du système peuvent aussi rentrer en ligne de compte ...

Ces différentes approches ne sont donc pas toutes conciliables et nécessitent des indices précis pour évaluer et comparer les performances de différents systèmes. Parmi tous les indices proposés dans la littérature, on peut retenir les suivants.

Du point de vue de l'utilisateur, l'indice le plus important est le temps de réponse, c'est-à-dire le temps qui s'écoule entre l'instant où une tâche est générée par un processeur et l'instant où celle-ci est complètement exécutée. Ce temps de réponse est une variable aléatoire dont les valeurs doivent être les plus faibles possibles. La loi de probabilité de cette variable est souvent résumée et jugée d'après son espérance et sa variance (*cf.* [22, 14]).

Du point de vue des ressources *i.e.* de l'ensemble des processeurs disponibles, l'indice mesuré est le nombre de tâches traitées par unité de temps par l'ensemble du système. Ce nombre est aussi une variable aléatoire dont l'espérance caractérise le débit moyen du système. Cet indice est par exemple utilisé dans [12] et dans Zahorjan *et al.* [103].

Le nombre de tâches présentes à un instant t dans le système représente la charge de travail instantanée de ce système. C'est une variable aléatoire dont l'espérance est la charge moyenne du système, et cet indice caractérise aussi les performances d'un système (voir [12]).

Afin de minimiser le coût financier dû aux transferts, on peut mesurer le taux des communications sur le réseau *i.e.* le nombre de communications effectuées dans le système pendant une unité de temps. Cet indice a par exemple été étudié dans [50].

En terme de fiabilité du ou des systèmes, on peut par exemple mesurer le taux de perte des tâches du système. Mais, cette "perte" peut survenir pour plusieurs raisons qu'il convient de distinguer. Il peut en effet survenir des pertes de tâches à la suite de pannes de certains processeurs, et la tolérance à la panne est bien sûr, en terme de fiabilité, un indice déterminant (*cf.* [1]). Une autre perte peut survenir à la suite d'une surcharge de la capacité mémoire associée à un processeur. Une politique de transfert pertinente consiste à minimiser la proportion de tâches perdues par le système pour cette raison. D'autres pertes de tâches peuvent déclencher des pertes en cascade ou avoir des répercussions graves. En effet, beaucoup d'applications informatiques génèrent des tâches qui doivent parfois au cours de leur exécution être synchronisées. Lors de la synchronisation des tâches, la

perte d'une tâche est alors dommageable pour l'ensemble du système (*cf.* [73]). Enfin, pour assurer le fonctionnement correct de certains appareils gérés en temps réel, certaines tâches doivent impérativement avoir fini leur exécution avant une date limite, et on parle d'échec dynamique lorsque ces échéances ne sont pas respectées. Pour ces systèmes, il devient vital de minimiser la probabilité d'échec dynamique.

2.4 Résultats antérieurs

Les travaux de recherche dans le domaine du transfert de charge ont commencé il y a une vingtaine d'années et se sont amplement poursuivis depuis (*cf.* [72, 100, 89]). Mais il résulte de l'étude qui précède que le transfert de charge est un problème extrêment complexe donnant lieu à des approches et à des solutions très diverses selon les contextes et selon les priorités à respecter (*cf.* [55]). On peut toutefois schématiser l'étude d'une politique de transfert selon l'indice de performance à optimiser et selon la gestion du système d'ordonnancement des tâches. Cette gestion repose sur une politique d'information qui précise les informations dont on dispose et le lieu où elles sont stockées, une politique de transfert qui précise quel processeur prend la décision du transfert, par quelles méthodes (migration ou placement) et avec quelle stratégie (partage ou équilibrage de charge) et enfin une politique de localisation qui précise si la tâche est transférée à l'aveugle sur un autre processeur ou en tenant compte d'informations adéquates sur les autres processeurs (*cf.* [11]). Un certain nombre d'algorithmes et de modèles ont donc été proposés et testés afin d'améliorer les performances du système (*cf.* [80, 94, 19, 92, 93]). Des classifications de ces algorithmes ont été faites, et un certain nombre d'implémentations effectives d'algorithmes d'ordonnancement ont été répertoriées, et analysés par Casavant et Kuhl [21] et Bernard *et al.* [11]. Voici un résumé de leurs conclusions.

La recherche d'une amélioration des performances grâce à une politique de transfert semble justifiée puisque des études ont montré que la probabilité qu'un processeur soit inactif alors que des tâches sont en attente ailleurs dans le système, est en général assez élevée (*cf.* [105], [94]). Autrement dit, la probabilité d'être en situation de transfert est assez élevée, et justifie donc les travaux qui sont consacrés au problème du transfert de charge.

Une amélioration substancielle des performances peut être obtenue par le placement de tâches, sous réserve de filtrer les tâches candidates au placement dans les environnements où beaucoup de tâches ont des durées de vie courtes, et les meilleurs résultats sont obtenus avec les algorithmes exploitant une information globale (*cf.* [87]). Par exemple, le partage de charge par placements seuls, améliore non seulement la moyenne, mais aussi la variance du temps de réponse pour tous les algorithmes étudiés dans [105].

La migration de tâches n'apporte pas d'améliorations décisives, par rapport au placement, sur la moyenne et la variance du temps de réponse. Elle est très difficile à mettre en place (*cf.* [1]), et elle ne s'avère justifiée que dans les environnements où l'on souhaite priviligier le caractère personnel des sites (*cf.* [35]).

Plusieurs études (*cf.* [46, 56]) font des analyses comparatives des algorithmes de partage de charge en fonction du choix d'un indice de performance.

D'autres études analysent des modèles théoriques de partage de charge (*cf.* [81]), et lorsque certaines caractéristiques ou certaines informations concernant les tâches ou le système sont connues, certaines études montrent l'optimalité de certaines stratégies. Par exemple, la théorie des files d'attente a été utilisée par Chow et Kohler dans [23] avec des politiques de placement déterministes ou non déterministes pour optimiser la valeur d'un indice de performance dans le cas de systèmes simples. Une autre approche est proposée par Bryant et Agre dans [18] pour résoudre le problème du placement d'un module particulier. Zahorjan *et al.* dans [103] développent une méthode permettant d'obtenir à faible coût des bornes sur le débit moyen (throughput) et le temps de réponse moyen d'un système homogène séparable. De Souza E Silva et Gerla dans [27] présentent une méthode de placement optimal dans un environnement statique. Dans [34] Eager *et al.* utilisent les processus de naissance et de mort pour comparer 2 politiques de transfert de charge selon que le transfert est effectué à l'initiative des processeurs sur-chargés ou à celle des processeurs sous-chargés. Deux études proches de celle que nous présenterons au chapitre 4 sont celles réalisées dans [46] et [83] et celle réalisée par Spies dans [82], mais elles diffèrent par le choix des indices de performance à optimiser et les outils mathématiques utilisés. En particulier le cas des systèmes massivement parallèles n'est pas résolu.

Dans le cas des machines parallèles, le choix d'une modélisation du transfert de charge est crucial car l'on ne dispose d'aucune information *a priori*, ni sur les caractéristiques des tâches, ni sur ce que sera l'état global du système au moment de leur création. Or, la parallélisation d'une application de grande taille s'exécute le plus souvent en environnement aléatoire et devrait nécessiter la maîtrise de différentes sources d'irrégularité. Parmi ces sources d'irrégularité, nous pouvons citer les deux principales :

La première est celle du temps de calcul car les applications de grande taille nécessitent souvent la génération et la manipulation de données complexes (files de priorité, matrices creuses ou graphes par exemple), et la complexité des dépendances de données rend imprévisible le schéma de calcul (graphe de précédence des tâches et mouvements de données).

La deuxième est celle de l'irrégularité des supports d'exécution et des temps de commu-

nications entre les sites. En effet, l'émergence de supports hétérogènes très performants (réseaux de stations de travail pour le calcul intensif) conduit à une multiplicité des machines parallèles disponibles.

Des indications et des informations explicites peuvent être suggérées par le programmateur dans les applications afin d'orienter le système d'exploitation dans la décision, en environnement aléatoire, du choix ou non d'un transfert de tâches, mais ces indications ne sont pas aisées à cerner (*cf.* [77]).
Typiquement, cette situation se retrouve, par exemple, dans les applications utilisant des recherches arborescentes. La fonction de régulation de charge doit décider de paralléliser ou de séquentialiser une partie des calculs avec l'exigence que le choix de la parallélisation assure une plus grande efficacité que celle de la séquentialisation. Cette étude suppose la mise en place de plate-formes d'évaluation de la fonction de régulation dynamique de la charge et de ses répercussions (*cf.* [17, 39]). Afin d'aider à cette étude, nous proposons ici un modèle stochastique de transfert de charge. Les problèmes et les choix de cette modélisation seront abordés dans la section 2.5. Des études récentes présentent d'autres modèles possibles de la gestion de la répartition des tâches pour les machines parallèles. Citons par exemple l'étude de Nelson et Squillante [68] pour laquelle la politique de transfert est basée sur des politiques de seuils. Les outils d'analyse qu'ils utilisent dérivent de ceux présentés dans [69] et l'approche diffère donc radicalement de ce document.

2.5 Modélisation stochastique du transfert 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 transfert de charge se situe donc à l'interface entre l'architecture et l'application.

2.5.1 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 2.1). 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 par leur 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 transfert de charge.

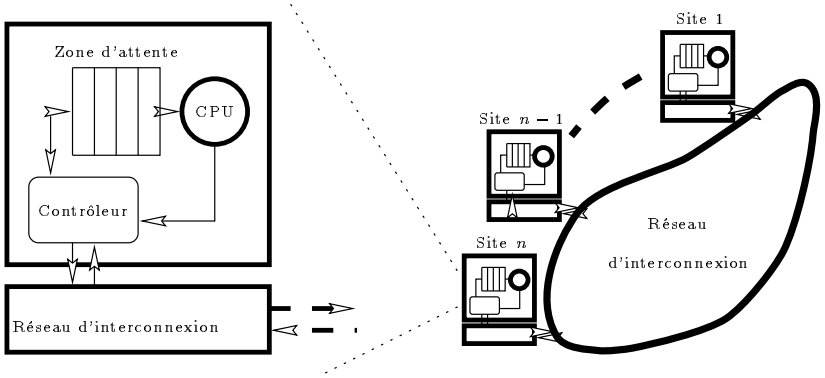

Figure 2.1 : Modèle d'un site et d'un réseau

2.5.2 Modélisation de la charge induite par des applications

De manière générale, une application distribuée/parallèle est constituée par un ensemble de processus ou tâches informatiques pouvant communiquer ensemble et 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 cette étude, nous retiendrons l'appellation de tâche pour modéliser ce grain de calcul. Le choix de ce grain reste un problème difficile, souvent résolu empiriquement. 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 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 [12], 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'attente avec synchronisations (*cf.* [3, 96]). 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 transférée, alors que cet état peut varier d'une exécution à une autre (*cf.* [76, 61]).

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 ce document où la charge d'un site est définie comme étant égale au nombre de tâches stockées sur ce site.

2.5.3 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 (*cf.* [57]) ont montré que la distribution du temps d'exécution T_{exec} d'une tâche, après avoir éliminé les tâches dont la durée de vie est trop courte, pouvait raisonnablement être approché par une distribution de probabilité exponentielle, c'est-à-dire

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

Avec cette modélisation, 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 [51]).
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 est alors un processus de Poisson (*cf.* [67]).

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

2.5.4 Modélisation du protocole de transfert

Sans rentrer dans les détails d'une classification d'algorithmes de partage de charge ou d'équilibrage de charge, il faut noter que la plupart des protocoles de tranfert utilisent des informations sur l'état 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 [99] ou [70]. 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" (*cf.* [60]). 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 poli-

tiques on pourra consulter [99, 48]. Une application au partage de charge et à l'implémentation d'heuristiques basées sur cette approche est développée dans [36].

Les modélisations que nous allons étudier utilisent la connaissance de l'état instantané de chaque site. Connaissant la charge de chaque site, un transfert peut être envisagé dès que la différence de charge entre deux sites voisins excède un certain seuil. De telles hypothèses conduisent à des modèles markoviens (automates à états probabilisés), dont on trouvera des exemples et des applications à des systèmes informatiques, dans [95] ou [67]. Le but de cette étude est d'obtenir des résultats quantitatifs permettant de comparer les valeurs moyennes des indices de performance avec une telle politique de transfert et celles obtenues sans politique de transfert. Cette étude pourrait alors cibler les valeurs des paramètres du système pour lequel une telle politique est souhaitable et celles pour lesquelles elle ne s'avère pas indispensable, compte tenu des difficultés de son implémentation effective.

Les difficultés mathématiques apparaissent d'une part lorsque l'on introduit les temps nécessaires au transfert de tâches, d'autre part lorsque le système devient massivement parallèle et enfin lorsque l'on tient compte de la topologie du réseau d'interconnection.
La premiere partie sera traitée dans le chapitre 3 dans le cas particulier où le nombre de sites est restreint à 2 sites chacun de capacité 2.
La deuxième partie est traitée dans le chapitre 4 dans le cas particulier d'un transfert instantané et immédiat entre n sites tous inter-connectés, de capacité mémoire K finie.
La dernière partie est traitée dans le chapitre 5 pour une gamme de réseaux de topologies régulières.

2.5.5 Indices de performance : choix et notations

Dans la majorité des modèles étudiés dans la littérature, le choix des indices de performance dépend fortement de l'utilisateur et des spécifications quantitatives demandées. Parmi les indices exposés dans la section 2.3, les indices suivants apparaissent pertinents pour la plupart des problèmes posés, et pour les modèles étudiés dans ce document. En effet, d'une part ces indices répondent à la demande des évaluations de performance pour les machines parallèles et ces indices rendent comptent du bon fonctionnement d'un système. D'autre part ces indices restent accessibles par le calcul ou par des simulations pour les modèles que nous avons étudiés. Ces modélisations permettent donc de faire de réelles comparaisons sur les valeurs de ces indices.

Débit du système C'est le nombre moyen $\overline{T}$ de tâches traitées par unité de temps par l'ensemble du système (*cf.* [99]).

Saturation mémoire C'est 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. Cet indice est par exemple aussi utilisé dans un autre contexte dans [53].

Charge de travail C'est le nombre moyen $\overline{N}$ de tâches présentes sur l'ensemble des sites. Aisément calculable pour les modèles étudiés dans la suite de ce document, ce nombre moyen est surtout nécessaire pour estimer le temps de réponse moyen d'une tâche grâce à la formule de Little (*cf.* [59]).

Temps de réponse moyen C'est le temps moyen $\overline{R}$ 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 (*cf.* [54]).

Mesure stationnaire de charge C'est la probabilité P_i pour un site donné d'avoir une charge égale à i. Pour un nombre donné de sites, ces probabilités représentent la proportion de sites ayant une charge égale à i. Ces probabilités ne sont donc pas à proprement parler des indices de performance, mais leurs évolutions selon les paramètres du système, permettent de rendre compte du comportement global de celui-ci, et permettent de mieux comprendre comment opère le transfert de charge. De plus, dans tous les modèles étudiés dans ce document, ces probabilités s'avèrent suffisantes pour calculer les indices de performance précédemment décrits et elles représentent donc des valeurs caractéristiques du système étudié.

Cette présentation a été partiellement faite en français dans le rapport technique [6], et en anglais dans le rapport technique [9] et a été partiellement publiée dans [10].

Chapitre 3

Le transfert de charge entre deux sites

3.1 Description du modèle

Pour simplifier dans un premier temps notre approche, nous supposerons que notre système est constitué de 2 sites homogènes *i.e.* ayant le même temps d'exécution moyen d'une tâche. 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 3.1. Dans ce graphe, le codage 0 représente une sous-charge, 1 représente une charge normale, et 2 représente une surcharge.

La capacité K de chaque site est ici égale à 2, et les deux sites seront numérotés 1 et 2.

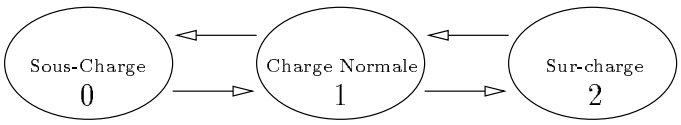

Figure 3.1 : Graphe d'état d'un site

Les hypothèses de modélisation de la dynamique du système sont les suivantes. Chaque processeur génère des tâches, et le temps séparant deux générations de tâches est modélisé par un temps aléatoire de loi exponentielle de paramètre λ. Le temps de service de chaque tâche sur un processeur est modélisé par un temps aléatoire de loi exponentielle de paramètre μ.
La charge de chaque site est définie par le nombre de tâches qu'accueille ce site. A chaque instant, ce nombre est un entier entre 0 et K, et une tâche générée par un processeur de charge K est rejetée.
Sans transfert, ce modèle se comporte comme 2 files $M/M/1/2$ indépendantes de taux d'arrivée

λ et de taux de départ μ (voir [99] p 38-87 pour une référence générale). Afin de prendre en compte les transferts possibles d'un site à l'autre, l'état global du système est modélisé par le couple des états (i, j) où i est l'état du site 1 et j l'état du site 2. La politique de transfert que nous souhaitons étudier dans ce modèle tient compte des délais des protocoles de routage entre les différents sites, et afin de prendre en compte le coût du transfert des tâches entre les sites, des états intermédiaires $(T, 0)$ et $(0, T)$ sont intoduits; l'état T d'un site indique sa possibilité de transférer, et le temps de séjour dans cet état modélise le temps des tractations possibles entre les deux sites. Pour prendre en compte l'évolution possible du site cible pendant le transfert (effet domino) deux stratégies peuvent être modélisées.

Stratégie sans interruption Le site initiateur ne contrôle pas le transfert qui s'effectue indépendamment de la surcharge du site cible. Le site cible devient chargé et la décision de transfert est maintenue. On introduit les états $(T, 1)$ et $(1, T)$ dans le modèle, traduisant cet événement.

Stratégie avec interruption Le site initiateur contrôle le transfert et le site cible devenant chargé, le transfert éventuel est interrompu. Le site initiateur reste donc surchargé.

Les graphes de transition des modèles de ces deux stratégies sont présentés dans la figure 3.2, et les différents paramètres de ces graphes s'interprètent de la manière suivante :

λ est le taux de génération de nouvelles tâches de chaque site. Le temps moyen qui s'écoule entre 2 générations de tâches est donc $1/\lambda$.

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

$1/x$ est le temps moyen de décision de transfert du site initiateur.

$1/y$ est le temps moyen de transfert entre le site initiateur et le site cible.

Détaillons la signification de ces graphes dans le cas d'un transfert. Supposons qu'une tâche est générée par le processeur numéro 2 alors que l'état global du système est $(0, 1)$. L'état du système devient $(0, 2)$. Le transfert peut alors être envisagé, ce qui dans la pratique peut prendre un certain temps puisque le système d'ordonnancement doit être informé que le site 2 est surchargé tandis que le site 1 est oisif. Pendant les tractations de communication, la premiere tâche du site numéro 2 peut se terminer et dans ce cas la décision de transfert est abandonnée et l'état global du système passe de $(0, 2)$ à $(0, 1)$. Ou bien, le site cible peut générer une tâche, et dans ce cas la possibilité de transfert est aussi abandonnée. L'état global du système passe alors de $(0, 2)$ à $(1, 2)$, et le site initiateur reste avec sa surcharge. Si aucune de ces deux éventualités ne s'est produite, le transfert est effectivement décidé et initialisé par le site surchargé; l'état global du système passe de $(0, 2)$ à $(0, T)$. Le transfert est décidé, et la dernière tâche arrivée sur

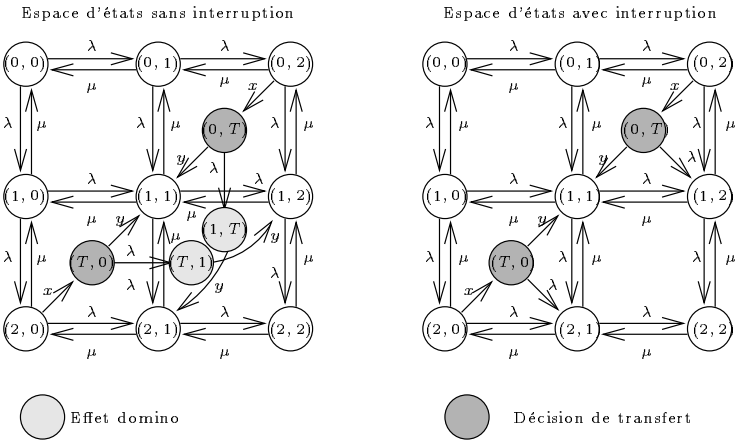

Figure 3.2 : Graphes d'états modélisant les 2 stratégies sans et avec interruption

le site numéro 2 va être transférée sur le site numéro 1. Le temps de ce transfert est une variable aléatoire dont la distribution est supposée exponentielle de paramètre y. Pendant ce transfert, l'état du site cible qui était 0 au début du transfert peut évoluer. Dans notre modélisation, le site initiateur lui est entièrement mobilisé par le transfert. Ce qui arrive ensuite est directement déduit des hypothèses. Si aucune tâche n'arrive sur le site cible, quelle que soit la stratégie, avec ou sans interruption, le prochain état du système devient $(1,1)$. Si une tâche est générée par le site cible pendant le transfert, les deux stratégies se traduisent de la manière suivante.
Pour la stratégie sans interruption, malgré l'arrivée de cette tâche, la décision de transfert est maintenue et l'état du système devient $(1,T)$. Dans notre modélisation, les deux processeurs sont alors mobilisés par le transfert et la tâche est transférée sur le site numéro 1. Cette tâche sera traitée après celle qui a été générée pendant le transfert. L'état global du système après le transfert devient alors $(2,1)$.
Pour la stratégie avec interruption, le système d'ordonnancement prend en compte l'évolution possible du site cible et quand l'état de celui-ci change (passe de 0 à 1) les mécanismes de transfert sont interrompus. L'état du système passe donc de $(0,T)$ à $(1,2)$ et le site initiateur garde sa surcharge.

La comparaison de ces deux modèles par rapport à deux files indépendantes $M/M/1/2$ permet d'évaluer la mise en œuvre de tels algorithmes de transfert. La comparaison de ces deux modèles entre eux permet d'évaluer la mise em œvre de la politique d'interruption. Afin d'éviter toute confusion, le modèle se référant à la première stratégie sera nommé modèle sans interruption, et celui se référant à la seconde sera nommé modèle avec interruption.

3.2 Résultats formels

Dans les deux modèles, avec et sans interruption, le processus $\{X_t, t \geq 0\}$ qui à chaque instant t associe le couple des états de chacun des deux sites est un processus markovien irréductible admettant donc une unique mesure stationnaire. Pour résoudre ce type de modèles markoviens, les outils d'agrégation forte (*cf.* [78]) sont utiles. Rappelons que l'agrégation forte d'un processus de Markov $\{X_t, t \geq 0\}$ dont l'espace d'états est E est possible s'il existe une partition $\mathcal{B} = (\mathcal{B}_1 \ldots \mathcal{B}_l)$ de E telle que :

$$\forall\, i \neq j \quad , \forall x \neq y \in \mathcal{B}_i \quad ,$$

le taux de transition de x vers $\mathcal{B}_j$ est égal au taux de transition y vers $\mathcal{B}_j$.
De telles méthodes d'agrégations et de décompositions peuvent être trouvées dans [25]. 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 plus aisés. Appliquée au modèle avec n sites présenté dans le chapitre 4, cette méthode devient indispensable.

Les deux études correspondant à chacun des deux modèles sont semblables, et seule celle correspondant au modèle sans interruption sera détaillée dans ce document. Afin d'utiliser la symétrie du système, on construit le processus agrégé $\{Y_t, t \geq 0\}$ de la manière suivante :

$$\begin{aligned}
Y_t &= 0 \quad si \quad X_t = (0,0), \\
Y_t &= 1 \quad si \quad X_t \in \{(0,1),(1,0)\}, \\
Y_t &= 2 \quad si \quad X_t \in \{(0,2),(2,0)\}, \\
Y_t &= e \quad si \quad X_t = (1,1), \\
Y_t &= T \quad si \quad X_t \in \{(0,T),(T,0)\}, \\
Y_t &= T1 \quad si \quad X_t \in \{(1,T),(T,1)\}, \\
Y_t &= 3 \quad si \quad X_t \in \{(1,2),(2,1)\}, \\
Y_t &= 4 \quad si \quad X_t = (2,2)
\end{aligned}$$

Les conditions d'agrégabilité sont vérifiées et d'après les résultats obtenus dans [78] le processus $\{Y_t, t \geq 0\}$ est un processus de Markov homogène dont le graphe de transition est donné dans la figure 3.3. Par exemple, le taux de transition de l'état 0 à létat 1 du processus $\{Y_t, t \geq 0\}$ s'obtient aisément. Le temps de séjour dans l'état 0 est le minimum de deux temps aléatoires modélisés par des lois exponentielles de paramètre λ. Ce temps suit donc une loi exponentielle de paramètre 2λ. Soit $(p_0, p_1, p_e, p_T, p_{T1}, p_2, p_3, p_4)$ la mesure stationnaire du processus $\{Y_t, t \geq 0\}$.

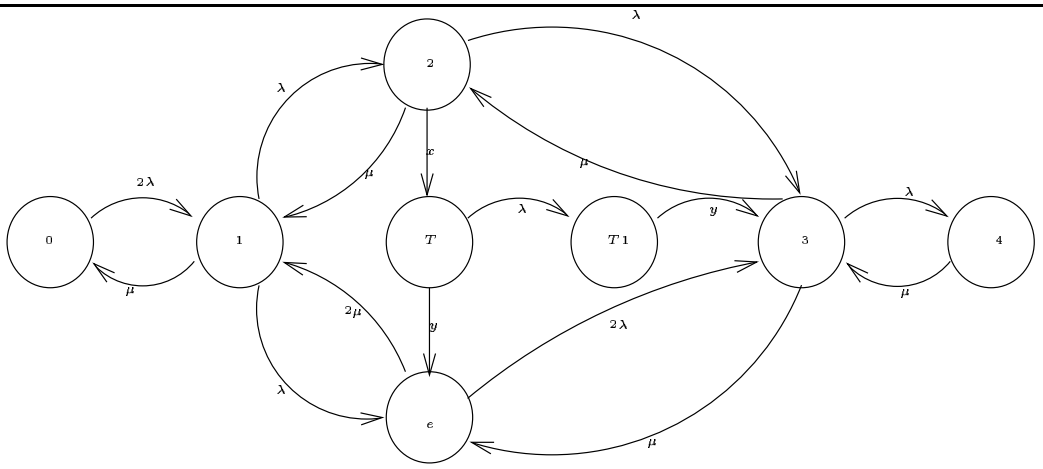

Figure 3.3 : Graphe d'états du processus agrégé du modèle sans interruption

D'après les équations de Kolmogorov, cette mesure est solution des équations :

$$\begin{aligned}
2\lambda p_0 &= \mu p_1, \\
(\mu + 2\lambda)p_1 &= 2\mu p_e + 2\lambda p_0 + \mu p_2, \\
(\lambda + \mu + x)p_2 &= \lambda p_1 + \mu p_3, \\
2(\lambda + \mu)p_e &= \lambda p_1 + \mu p_3 + y p_T, \\
(\lambda + y)p_T &= x p_2, \\
\lambda p_T &= y p_{T1}, \\
(2\mu + \lambda)p_3 &= 2\mu p_4 + 2\lambda p_e + y p_{T1} + \lambda p_2, \\
2\mu p_4 &= \lambda p_3 \quad .
\end{aligned}$$

Notons p_{ij}, la probabilité, en régime stationnaire, de l'état (i, j) du processus $\{X_t,\, t \geq 0\}$ où $(i, j) \in \{0, 1, 2, T\}^2$.
En utilisant la symétrie du système, ces valeurs s'obtiennent aisément à partir de celles obtenues pour le processus $\{Y_t,\, t \geq 0\}$.
Ces valeurs permettent alors de faire explicitement l'évaluation des indices de performances en régime stationnaire. En effet, ceux-ci s'expriment par les relations algébriques suivantes :

Proposition 1. *Avec 2 sites de capacité 2, et pour le modèle sans interruption, la valeur des indices de performance est*

$$\begin{aligned}
P_{sat} &= p_{20} + p_{21} + p_{22} + p_{T0} + p_{T1} + p_{1T}, \\
\overline{T} &= 2\lambda(1 - P_{sat}), \\
\overline{N} &= 2p_{01} + 4p_{02} + 2p_{11} + 6p_{12} + 4p_{22} + 4p_{T0} + 6p_{T1},
\end{aligned}$$

$$\overline{R} = \frac{\overline{N}}{\overline{T}}$$

Démonstration

La première équation utilise la symétrie du système et s'obtient aisément compte tenu de la définition de la probabilité de saturation mémoire.

La troisième équation donne le nombre moyen de tâches présentes dans le système à un instant donné et sa justification ne présente pas de difficulté.

La deuxième et la quatrième équation nécessitent quelques développements.

- Nombre de tâches traitées par unité de temps :
  Ce résultat est intuitivement clair et sa démonstration ne sera donnée que dans le cas de ce modèle avec 2 sites.
  A l'instant t, définissons les variables aléatoires suivantes :

  A_t représente le nombre de tâches qui ont été générées par le système dans l'intervalle de temps $[0,t]$. $\{A_t,\, t \geq 0\}$ est un processus de Poisson d'intensité 2λ.

  T_t représente le nombre de tâches effectivement acceptées par le système.

  X_t représente l'état du système à l'instant t. Le processus $\{X_t,\, t \geq 0\}$ est un processus de Markov dont le taux de transition d'un état x à un état y sera noté λ_{xy}. Ce processus reste dans chaque état x un temps exponentiel dont le paramètre sera noté λ_x.

  $N_{x,y}(t)$ représente le nombre de fois où a eu lieu la transition de x vers y dans l'intervalle $[0,t]$.

  Z_t le processus défini sur l'ensemble des transitions de X_t c'est-à-dire sur l'ensemble des couples (x,y), où (x,y) représente une transition possible du processus $\{X_t,\, t \geq 0\}$. A l'instant t, Z_t indique la dernière transition effectuée par X_t.

  Soit (x,y) la dernière transition effectuée. Le processus Z_t reste un temps exponentiel de paramètre λ_y dans cet état et, à l'issue de ce séjour, la transition suivante, c'est-à-dire l'état de Z_t sera (y,z) avec la probabilité $\frac{\lambda_{xy}}{\lambda_y}$.
  Le processus $\{Z_t,\, t \geq 0\}$ est donc un processus de Markov, et le taux de transition d'un état (x,y) à un état (y,z) est λ_{yz}. Ce processus est

irréductible et admet une unique mesure stationnaire.

Soit q_{xy} la probabilité stationnaire de l'état (x, y) du processus $\{Z_t,\, t \geq 0\}$. D'après le théorème ergodique appliqué au processus $\{Z_t,\, t \geq 0\}$, q_{xy} représente la proportion de temps moyen passé dans l'état (x, y). Or à chaque séjour dans l'état (x, y) le processus $\{Z_t,\, t \geq 0\}$ y reste en moyenne $\frac{1}{\lambda_y}$. D'où l'équation

$$\lim_{t \to +\infty} \frac{N_{x,y}(t)}{t} = \lambda_y q_{xy} \quad ,$$

Soit Δ l'ensemble des transitions possibles du processus $\{X_t,\, t \geq 0\}$ qui augmentent de 1 le nombre total de tâches dans le sytème. On obtient alors les égalités suivantes.

$$\overline{T} = \lim_{t \to +\infty} \frac{N_t}{t} = \sum_{(x,y) \in \Delta} \frac{N_{x,y}(t)}{t} = \sum_{(x,y)\, \in\, \Delta} \lambda_y q_{xy} \quad .$$

Désignons par p_x la probabilité stationnaire de l'état x du processus $\{X_t,\, t \geq 0\}$.

On vérifie alors sur le processus $\{Z_t,\, t \geq 0\}$ l'égalité,

$$\sum_z \lambda_{yz} q_{xy} = \sum_u \lambda_{xy} q_{ux} \quad ,$$

D'où

$$\lambda_y q_{xy} = \lambda_{xy} p_x \quad .$$

Dans les deux modèles présentés, avec et sans interruption, on a :

$$(x, y) \in \Delta \Rightarrow \lambda_{xy} = \lambda \quad .$$

Par suite,

$$\overline{T} = \lambda \sum_{(x,y)\, \in\, \Delta} p_x \quad .$$

Soit Δ_1 l'ensemble des transitions possibles du processus $\{X_t,\, t \geq 0\}$ qui augmentent de 1 le nombre total de tâches dans le sytème en augmentant de 1 la charge du site numéro 1. Par symétrie des rôles des sites 1 et 2, on obtient :

$$\overline{T} = 2\lambda \sum_{(x,y)\, \in\, \Delta_1} p_x \quad ,$$

D'où le résultat

$$\overline{T} = 2\lambda(1 - P_{sat}) \quad .$$

- Temps de réponse moyen

  Les conditions de validité de la formule de Little explicitées dans l'ouvrage de Walrand [99] p. 100 sont ici vérifiées puisque

$$\lim_{t \to +\infty} \frac{T_t}{t} = \overline{T} \quad ,$$

  d'une part, et d'autre part, le processus $\{X_t, t \geq 0\}$ est ergodique donc la variable aléatoire égale au temps de séjour d'une tâche acceptée par le système admet une espérance.

  D'où l'égalité

$$\overline{R} = \frac{\overline{N}}{2\lambda(1 - P_{sat})} \quad .$$

■

3.3 Obtention de valeurs critiques dans certains cas particuliers

Pour alléger les notations et sans restriction de généralité, on supposera que le temps moyen d'exécution d'une tâche est unitaire *i.e.* que le paramètre μ est égal à 1. Les calculs ont été effectués grâce au logiciel *Mathematica*. Parmi toutes les situations possibles, celle où le temps de décision avant transfert est supposé négligeable devant le grain de calcul (x très supérieur à 1) semble intéressante à étudier.

Le résultat principal est l'existence de valeurs critiques. Pour un taux de génération de tâches fixé λ, il existe une valeur du temps de transfert moyen $D_c(\lambda)$ au dessus de laquelle le transfert n'améliore plus les performances, et cette valeur critique dépend de la performance que l'on souhaite améliorer. Dans les deux modèles avec et sans interruption, on peut évaluer par exemple, la valeur critique $D_c^{\overline{R}}(\lambda)$ optimisant le temps de réponse moyen $\overline{R}$, obtenu avec une politique de transfert par rapport à celle obtenue sans transfert (*i.e.* avec le temps de réponse moyen d'une $M/M/1/2$).

Pour la politique sans interruption on obtient

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

et pour la politique avec interruption on obtient

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

La figure 3.4 met en évidence cette situation et corrobore l'idée intuitive selon laquelle plus le

taux de génération des tâches est grand plus le temps de transfert doit être petit pour obtenir un gain sur le temps de réponse moyen d'une tâche.

En étudiant les autres indices de performance, on obtient d'autres valeurs critiques, ce qui

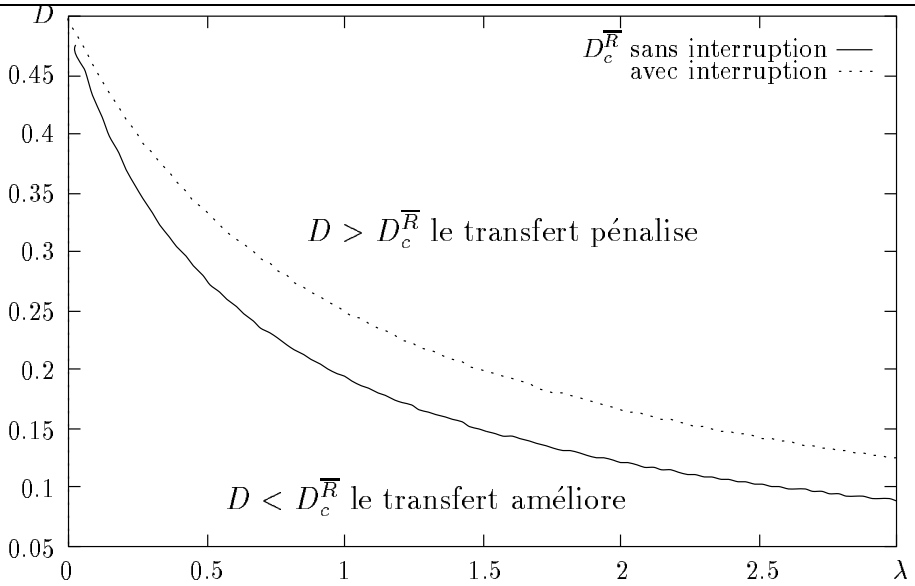

Figure 3.4 : Temps de transfert critique en fonction du débit d'entrée

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 $D_c^{P_{sat}}(\lambda)$.

Pour le modèle sans interruption, cette valeur est différente de celle obtenue pour le temps de réponse moyen.

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

et pour le modèle avec interruption

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

3.4 Calculs des bénéfices maxima et relatifs obtenus grâce au transfert

Le bénéfice obtenu grâce à la politique de transfert est calculé à partir des valeurs des indices obtenus avec une politique de transfert, et celles obtenues sans politique de transfert. Les courbes obtenues en section 3.3 montrent que la possibilité d'interrompre le transfert améliore le bénéfice possible grâce au transfert. Cela dit, les mécanismes qui permettent une telle interruption sont très complexes et les écarts entre les bénéfices obtenus ne semblent pas justifier leur mise en

œuvre. L'étude de cette possibilité sera donc écartée dans la suite de ce document.

Pour les distinguer des valeurs obtenues avec une politique de transfert, les valeurs des indices obtenues dans la situation sans transfert seront notées avec un astérisque "*".

Lorsque le temps de transfert est aussi négligé devant le temps de calcul (y très supérieur à 1), la modélisation obtenue avec la politique de transfert est celle d'une situation idéale où le transfert est immédiat et instantané. Les mécanismes d'interruption n'ont alors plus de sens et les valeurs optimales des bénéfices maxima et relatifs que l'on peut attendre d'un transfert réel peuvent alors être calculées. Les gains relatifs sont calculés par rapport à la situation sans transfert. Ces valeurs optimales des bénéfices sont obtenues pour des valeurs particulières respectives λ_{max}^{abs} et λ_{max}^{rel} du flux d'entrée, et ces valeurs dépendent elles aussi de l'indice de performance que l'on souhaite optimiser. Nous pouvons par exemple donner les valeurs de λ_{max}^{abs} et λ_{max}^{rel} obtenues pour le temps de réponse moyen et pour la probabilité de saturation mémoire et donner, pour ces indices, les valeurs des gains absolus maxima et des gains relatifs maxima.

Temps de réponse moyen. Le gain absolu optimum est 0.192 et est obtenu pour la valeur $\lambda_{max}^{abs} = 0.565$. Pour cette valeur, on obtient sans transfert $\overline{R}^* = 1.361$ et avec transfert $\overline{R} = 1.169$. 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 avec une politique de transfert effectuée dans des conditions idéales.

Le gain relatif optimum est 0.143 et est obtenu pour la valeur $\lambda_{max}^{rel} = 0.455$. Pour cette valeur, on obtient sans transfert $\overline{R}^* = 1.313$ et avec transfert $\overline{R} = 1.125$. Notons que les valeurs critiques du taux d'arrivée sont différentes, mais elles peuvent s'interpréter comme proches de la moitié du temps d'exécution moyen d'une tâche.

Probabilité de saturation mémoire. Le gain absolu maximum est 0.089 et est obtenu pour la valeur $\lambda_{max}^{abs} = 0.767$. Pour cette valeur on obtient sans transfert $P_{sat}^* = 0.25$ et avec transfert $P_{sat} = 0.161$. Dans notre modélisation et pour ce cas particulier restreint à deux sites de capacité mémoire égale à 2, l'incidence d'une politique de transfert sur la probabilité de saturation reste, en absolu, assez faible. Le gain relatif optimum tend vers 1.0 et est obtenu quand $\lambda \, tend \; vers \; 0$. En effet, pour des valeurs du taux d'arrivée très proches de 0, par xemple 0.01, on obtient sans transfert $P_{sat}^* = 0.0001$ et avec transfert $P_{sat} = 0.0$. Pour de faibles valeurs du taux d'arrivée, la politique de transfert permet d'avoir une utilisation optimale des capacités mémoires. Le gain relatif est donc proche de 1, mais même dans la situation sans transfert, la probabilité de saturation mémoire est déjà très faible.

3.5 Incidence du transfert sur la variance du temps de réponse

3.5.1 Calculs formels

Une autre quantité intéressante à calculer, quand cela est possible, est la variance du temps de réponse. La méthode utilisée ici consiste à suivre la trajectoire éventuelle d'une tâche générée par le système. Cette méthode permet de calculer d'une autre façon que par la formule de Little l'espérance du temps de réponse puis sa variance. En régime stationnaire, la loi du temps de séjour d'une tâche dans le système ne dépend pas de l'instant t où la tâche est générée par le système. En utilisant la symétrie des rôles des sites numéro 1 et numéro 2, nous pouvons, sans perte de généralité supposer que la tâche est générée par le site numéro 1.
Les notations suivantes seront utilisées dans la suite de ce développement.
Soit

Ω l'ensemble des états x possibles du système.

$\Omega^{'}$ l'ensemble des états x du système pour lesquels une tâche générée sur le site x est effectivement acceptée par le système.

R la variable aléatoire égale au temps de séjour dans le système de cette tâche.

X la variable aléatoire donnant l'état x du système quand la tâche est générée,

A l'événement : "la tâche est acceptée par le système".

p_x la probabilité de l'état x en régime stationnaire.

Le temps de séjour moyen d'une tâche acceptée par le système est

$$\overline{R} = \mathbb{E}(\, R \,|\, A\,) = \sum_{x \in \Omega} \mathbb{E}(\, R \,|\, A \cap (X = x))\mathbb{P}(X = x \,|\, A\,) \quad ,$$

d'où

$$\mathbb{E}(\, R \,|\, A\,) = \sum_{x \in \Omega'} \mathbb{E}(\, R \,|\, X = x))\frac{p_x}{\mathbb{P}(A)} \quad .$$

Notons alors,

$$q_x = \frac{p_x}{\mathbb{P}(A)} \quad .$$

Nous obtenons alors,

$$\overline{R} = \mathbb{E}(\, R \,|\, A) = \sum_{x \in \Omega'} \mathbb{E}(\, R \,|\, X = x))q_x \quad . \tag{3.1}$$

De la même manière, nous obtenons,

$$\mathbb{E}((R-\overline{R})^2 \mid A) = \sum_{x\in\Omega'} \mathbb{E}((R-\overline{R})^2 \mid X=x))q_x \quad .$$

Donc

$$Var\ (\ R \mid A) = \sum_{x\in\Omega'} \mathbb{E}(R^2 \mid X=x)q_x - \overline{R}^2 \quad . \tag{3.2}$$

Pour obtenir l'espérance et la variance du temps de réponse, il suffit donc de calculer les quantités $\mathbb{E}(\ R \mid X=x)$ et $\mathbb{E}(\ R^2 \mid X=x)$ pour tous les états x de Ω'.

Lorsqu'une tâche est générée et est acceptée par le site numéro 1 alors que le système est dans l'état x, l'état du processus $\{X_t,\ t \geq 0\}$ change et passe de façon univoque à un état y. Parvenue dans l'état y, cette tâche se retrouve la n-ième dans la file d'attente du site où elle se trouve. Afin de pouvoir suivre la trajectoire possible de cette tâche dans le système, donnons alors les notations suivantes. Soit

$T_{y,n}$, le temps pour cette tâche, restant à passer dans le système immédiatement après son arrivée.

Ω_y , l'ensemble des états accessibles depuis y
Autrement dit $\Omega_y = \{z \in \Omega \quad tel\ que\ \lambda_{yz} \neq 0\}$ avec les notations de la preuve donnée pour la proposition 1.

p_{yz} , la probabilité de passer de l'état y à l'état z.

y_0 le seul état accessible depuis y correspondant à un service sur le site numéro 1.

Ω_y^* , l'ensemble des états, hormis l'état y_0, accessibles depuis y.

Nous avons les relations évidentes suivantes :

$$\Omega_y = \Omega_y^* \cup \{y_0\} \quad ,$$

$$\sum_{z\ \in\ \Omega_y} p_{yz} = 1 \quad .$$

De plus, le temps $T_{y,n}$ est la somme du temps résiduel T_y à passer dans l'état y et du temps restant à passer dans le système après la prochaine transition.
D'après les hypothèses, T_y suit une loi exponentielle de paramètre $\lambda_y = \sum_x \lambda_{yx}$.
Remarquons que pour un état y donné, il n'y a que deux sortes de transitions possibles. Il y a celle qui correspond à un service sur le site où se trouve la tâche et qui diminue n de 1, et celles

qui ne modifient pas n.

En utilisant la propriété de Markov du système, l'homogénéité dans le temps et l'invariance par translation, nous obtenons les relations récursives suivantes :

$$\forall y \in \Omega \quad ,$$

$$\mathbb{E}(T_{y,2}) = \frac{1}{\lambda_y} + \sum_{z \in \Omega_y^*} \mathbb{E}(T_{z,2}) p_{yz} + \mathbb{E}(T_{y_0,1}) p_{yy_0} \quad , \tag{3.3}$$

$$\mathbb{E}(T_{y,1}) = \frac{1}{\lambda_y} + \sum_{z \in \Omega_y^*} \mathbb{E}(T_{z,1}) p_{yz} + \mathbb{E}(T_{y_0,0}) p_{yy_0} \quad , \tag{3.4}$$

$$\mathbb{E}(T_{y,0}) = 0 \quad . \tag{3.5}$$

Appliquées au cas particulier $y = (2,0)$ ces équations donnent par exemple les relations suivantes.

$$\mathbb{E}(T_{(2,0),2}) = \frac{1}{\lambda + x + \mu} + \mathbb{E}(T_{(2,1),2}) \frac{\lambda}{\lambda + x + \mu} + \mathbb{E}(T_{(T,0),2}) \frac{x}{\lambda + x + \mu} + \mathbb{E}(T_{(1,0),1}) \frac{\mu}{\lambda + x + \mu} \quad ,$$

$$\mathbb{E}(T_{(1,0),1}) = \frac{1}{2\lambda + \mu} + \mathbb{E}(T_{(1,1),1}) \frac{\lambda}{2\lambda + \mu} + \mathbb{E}(T_{(2,0),1}) \frac{\lambda}{2\lambda + \mu} \quad .$$

L'écriture de la totalité des équations est relativement lourde, et elle est donnée dans le rapport technique [6].
Les valeurs des espérances $\mathbb{E}(T_{y,n})$ apparaissent donc comme la solution d'une équation matricielle que l'on résoud, par exemple avec le logiciel *Mathematica*.

De façon analogue, cette méthode permet de raisonner sur les espérances des carrés des temps de séjour successifs dans le système. On obtient ainsi :

$$\mathbb{E}(T_{y,2}^2) = \sum_{z \in \Omega_y^*} \mathbb{E}(T_y + T_{z,2})^2) p_{yz} + \mathbb{E}(T_y + T_{y_0,1})^2) p_{yy_0} \quad ,$$

En utilisant l'indépendance des temps de séjour, on obtient,

$$\begin{aligned} \mathbb{E}(T_{y,2}^2) \quad = \quad & \sum_{z \in \Omega_y} \mathbb{E}(T_y^2) p_{yz} + 2\mathbb{E}(T_y) \left(\sum_{z \in \Omega_y^*} \mathbb{E}(T_{z,2}) p_{yz} + \mathbb{E}(T_{y_0,1}) p_{yy_0} \right) \\ & + \sum_{z \in \Omega_y^*} \mathbb{E}(T_z^2) p_{yz} + \mathbb{E}(T_{y_0,1}^2) p_{yy_0} \quad , \end{aligned}$$

D'où, en utilisant l'équation 3.3

$$\mathbb{E}(T_{y,2}^2) - \sum_{z \in \Omega_y^*} \mathbb{E}(T_z^2) p_{yz} - \mathbb{E}(T_{y_0,1}^2) p_{yy_0} = \frac{2}{\lambda_y^2} + \frac{2}{\lambda_y} \left(\mathbb{E}(T_{y,2}) - \frac{1}{\lambda_y} \right) \quad ,$$

On obtient alors sur les espérances des carrés des temps de séjour une équation semblable à l'équation 3.3

$$\mathbb{E}(T_{y,2}^2) - \sum_{z \in \Omega_y^*} \mathbb{E}(T_{z,2}^2) p_{yz} - \mathbb{E}(T_{y_0,1}^2) p_{yy_0} = \frac{2}{\lambda_y} \left(\mathbb{E}(T_{y,2}) \right) \quad . \tag{3.6}$$

De même, on obtient sur les espérances des carrés des temps de séjour, une équation analogue à l'équation 3.4.

$$\mathbb{E}(T_{y,1}^2) - \sum_{z \in \Omega_y^*} \mathbb{E}(T_{z,1}^2) p_{yz} = \frac{2}{\lambda_y} \left(\mathbb{E}(T_{y,1}) \right) \quad . \tag{3.7}$$

Les espérances des carrés des temps de séjour $T_{y,n}^2$ sont donc aussi solution d'une équation matricielle et cette équation utilise les résultats obtenus pour les espérances $\mathbb{E}(T_{y,n})$.
Pour alléger ici les notations le temps de séjour $T_{y,n}$ pour $y = (0,1)$ sera noté $T_{01,1}$.

En appliquant alors l'équation 3.1 l'espérance du temps de réponse d'une tâche acceptée par le système se calcule par la formule

$$\overline{R} = \frac{p_{00}\mathbb{E}(T_{10,1}) + p_{01}\mathbb{E}(T_{11,1}) + p_{02}\mathbb{E}(T_{12,1}) + p_{10}\mathbb{E}(T_{20,2}) + p_{11}\mathbb{E}(T_{21,2}) + p_{12}\mathbb{E}(T_{22,2}) + p_{0T}\mathbb{E}(T_{1T,1})}{1 - P_{sat}} \; .$$

Et de façon analogue, la variance du temps de réponse d'une tâche acceptée par le système s'obtient alors grâce à l'équation 3.2.

3.5.2 Résultats numériques

De même que pour les études précédentes, les valeurs de la variance peuvent être comparées dans les situations avec transfert et dans la situation sans transfert. La variance du temps de réponse d'une tâche acceptée par le système sans transfert sera notée *Vrt** et celle obtenue avec la politique de transfert sans interruption sera notée *Vrt*.
Ces variances ont été comparées dans la situation d'un transfert immédiat et instantané. La table 3.1 donne les valeurs de ces variances obtenues pour des valeurs intéressantes du taux d'arrivée λ.

Dans les situations extrêmes, quand le taux d'arrivée est très faible ou très grand par rapport au temps d'exécution moyen d'une tâche, la politique de transfert n'induit pas d'amélioration significative sur la variance du temps de réponse. Par contre, pour les valeurs d'intérêt (qui maximisent le gain maximum ou relatif obtenu sur le temps de réponse moyen), la politique de transfert permet, de plus, de diminuer la variance du temps de réponse d'une tâche. Dans tous les algorithmes étudiés dans [105], la politique de partage de charge améliore aussi la moyenne

λ	Vrt^*	$\overline{R}^*$	Vrt	$\overline{R}$
0.01	1.02	1.01	1.0	1.0
0.455	1.53	1.31	1.14	1.13
0.565	1.59	1.36	1.19	1.17
100	1.99	1.99	1.98	1.99

Table 3.1 : Table des valeurs de la variance du temps de réponse

et la variance du temps de réponse.

Les résultats de ce chapitre ont partiellement été publiés dans [10].

Chapitre 4

Le transfert de charge entre n sites, graphe complet

Le modèle établi pour deux 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, utilisée dans le chapitre 3 permet dans ce cas de réduire considérablement la complexité des calculs, ce qui rend le modèle analytiquement résoluble. On observera, de la même manière les phénomènes de valeurs critiques et de basculement de comportement.

4.1 Description du modèle

Pour améliorer la pertinence des modèles précédents, les hypothèses de modélisation sont généralisées de la façon suivante.
On dispose de n sites ayant chacun une capacité mémoire de K tâches. Chaque site gère sa file d'attente de tâches selon une politique FIFO.
Dans ce modèle, chaque site communique avec l'ensemble des autres.
Les hypothèses de modélisation de la dynamique du système sont les suivantes. Chaque processeur génère des tâches, et le temps séparant deux générations de tâches est modélisé par un temps aléatoire de loi exponentielle de paramètre λ. Le temps de service de chaque tâche sur un processeur est modélisé par un temps aléatoire de loi exponentielle de paramètre μ.
La charge de chaque site est définie par le nombre de tâches qu'accueille ce site. A chaque instant, ce nombre est un entier entre 0 et K, et une tâche générée par un processeur de charge K est rejetée.
Deux sites se transfèrent instantanément des tâches dans les deux situations suivantes. Quand une tâche arrive sur un site dont la charge devient j, cette tâche est transférée sur un site de charge $j-2$ s'il existe. Parmi les sites de charge $j-2$, le site qui reçoit la tâche supplémentaire

est choisi au hasard. Quand une tâche se termine sur un site dont la charge devient j, une tâche en provenance d'un site de charge $j+2$ est transférée. Parmi les sites de charge $j+2$, celui qui transfère sa dernière tâche arrivée est choisi au hasard.

Pour que le transfert ait un sens, la contrainte sur la capacité mémoire K est d'être supérieure à 2.

Cette modélisation traduit une politique d'équilibrage de charge (*cf.* [11]) où les tâches sont transférées dynamiquement à chaque changement d'état du système, et utilise les techniques de placement de tâches. La différence avec une file d'attente $M/M/n/Kn$ (*cf.* [16]) tient au respect du comportement local de chaque processeur. En effet, dans notre modélisation, il n'y a pas de file d'attente partagée par l'ensemble des processeurs et un site de charge K n'accepte plus de nouvelles tâches.

La différence entre les deux situations est illustrée par les figures 4.1 et 4.2. Quand il y a

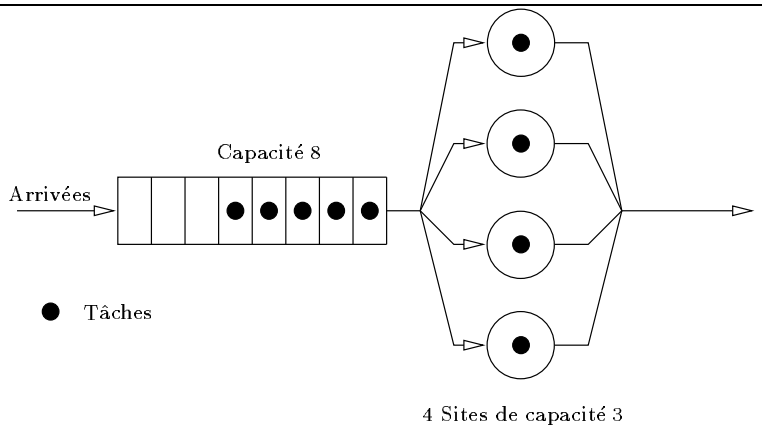

Figure 4.1 : File d'attente $M/M/4/12$ modélisant 4 sites se partageant un espace mémoire de 12 tâches.

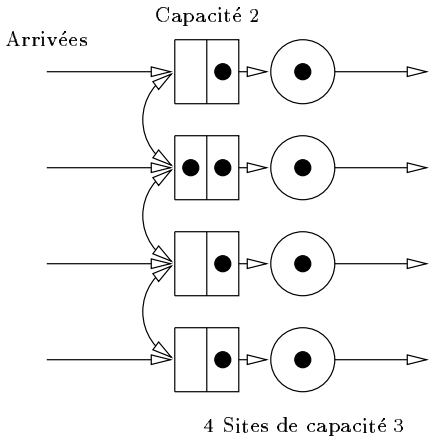

Figure 4.2 : Quatre files d'attente $M/M/1/3$ en parallèle avec partage de charge .

partage de l'espace mémoire, le protocole de transfert instantané de la surcharge sur des sites sous-chargés conduit à un modèle de file d'attente multiserveurs (figure 4.1), ayant n serveurs et 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 (*cf.* [54]).

Comme nous l'avons constaté au chapitre 3, un transfert immédiat et instantané entre tous les processeurs dès que la différence de charge dépasse 2, ne traduit pas toute la complexité d'un réel transfert de charge. Néanmoins, cette modélisation traduit une situation idéale généralisée à n processeurs de capacité quelconque où le comportement autonome de chaque processeur est respecté, et le but de cette étude est de comparer cette situation idéale avec la situation opposée où aucune politique de transfert n'est mise en place.
Dans la situation sans transfert, les n processeurs se comportent comme n files indépendantes $M/M/1/K$ de taux d'arrivée λ, et de taux de service μ. Dans la situation avec transfert, le processus étudié est le processus $\{X_t, t \geq 0\}$, qui à chaque instant t fait correspondre le n-uplet dont la i-ème coordonnée représente la charge du $i-$ème processeur. Les hypothèses de modélisation permettent d'étudier l'évolution de la charge de l'ensemble des processeurs au cours du temps, puisque $\{X_t, t \geq 0\}$ est un processus de Markov. Ce processus admet une unique mesure stationnaire et tous les indices de performance étudiés peuvent être calculés en régime stationnaire. Pour chacun de ces indices, la différence des valeurs obtenues avec une politique de transfert idéale avec celles obtenues sans politique de transfert peut donc être interprétée comme la borne supérieure du bénéfice que l'on peut attendre d'un réel transfert.

4.2 Résultats exacts sans transfert et avec cette politique de transfert

Le processus $\{X_t, t \geq 0\}$ défini à la section 4.1 est un processus de Markov admettant une unique mesure stationnaire. Tous les indices explicités dans la section 2.5.5 sont calculés en régime stationnaire dans les situations avec et sans transfert. Pour les distinguer, toutes les quantités relatives à la situation sans transfert seront notées avec un astérisque " $*$ ".

Sans perte de généralité et, pour des raisons de symétrie, la valeur de n est supposée paire, égale à $2q$. Ceci est une contrainte extrêmement faible, puisque les architectures parallèles disposent le plus souvent d'un nombre pair de processeurs.

4.2.1 Résultats valables dans les deux situations

Dans un premier temps, nous nous intéressons aux probabilités stationnaires pour un processeur donné d'avoir une charge i. Compte tenu des rôles symétriques joués par chacun des

processeurs, ces quantités ne dépendent pas du processeur. Pour chaque niveau de charge i, nous noterons $P_i(\frac{\lambda}{\mu})$ et $P_i^*(\frac{\lambda}{\mu})$ les probabilités stationnaires de chaque processeur d'avoir une charge i avec et sans transfert respectivement. Dans les deux situations, avec et sans transfert, des équations de nature algébrique relient certaines valeurs des $P_i(\frac{\lambda}{\mu})$.

Proposition 2. *Les équations suivantes sont vérifiées dans les deux situations avec et sans transfert :*

$$1) \quad \forall\, i \in \{0 \ldots K\} \quad P_i(\frac{\lambda}{\mu}) = P_{K-i}(\frac{\mu}{\lambda}) \quad ,$$

$$2) \qquad \lambda(1 - P_K(\frac{\lambda}{\mu})) = \mu\left(1 - P_0(\frac{\lambda}{\mu})\right) \quad .$$

La première partie de cette proposition revient à l'observation suivante. Remplacer i par $K - i$ revient à remplacer la capacité mémoire occupée d'un processeur par sa capacité mémoire libre. Le processus qui à chaque instant t fait correspondre le n-uplet dont la i-ème coordonnée représente la capacité mémoire libre du $i-$ème processeur a exactement la même dynamique que le processus $\{X_t,\, t \geq 0\}$, en inversant les rôles de λ et μ.
La deuxième partie de la proposition est une équation d'équilibre global. La partie gauche représente le nombre moyen de tâches acceptées par chaque processeur par unité de temps, tandis que la partie droite représente le nombre moyen de tâches traitées par chaque processeur par unité de temps.
Ces deux équations font immédiatement apparaître les caractéristiques dues à la symétrie du système d'une part et celle traduisant la balance des flux du système d'autre part. Néanmoins, la démonstration de ces résultats avec et sans transfert, anticipe sur les notations utilisées et les résultats obtenus dans la section 4.2.3 et la section 4.2.2 que nous supposons acquis dans la preuve qui suit.

Démonstration

Voici une démonstration de la proposition 2 dans les deux situations sans et avec transfert. Pour alléger les notations, et sans perte de généralité, cette démonstration est obtenue avec un temps moyen d'exécution égal à 1, ce qui revient à fixer l'unité de temps $\frac{1}{\mu} = 1$.

Sans transfert : L'équation est immédiate pour $\lambda = 1$.

Soit $\lambda \neq 1$. On obtient

$$\begin{aligned} \lambda(1 - P_K^*(\lambda)) &= \lambda\left(1 - \lambda^K \frac{1-\lambda}{1-\lambda^{K+1}}\right) \quad , \\ &= \lambda\left(\frac{1-\lambda^K}{1-\lambda^{K+1}}\right) \quad . \end{aligned}$$

$$\begin{aligned} 1 - P_0^*(\lambda) &= 1 - \frac{1-\lambda}{1-\lambda^{K+1}} \quad , \\ &= \lambda \left(\frac{1-\lambda^K}{1-\lambda^{K+1}} \right) \quad . \end{aligned}$$

D'où le résultat sans transfert.

Avec transfert : On a, d'après les calculs faits en 4.2.3,

$$P_K(\lambda) = \rho \left[G(n, \frac{1}{\lambda})(1 - \frac{1}{\lambda}) + \frac{n^n}{n!}\lambda^{Kq-n+1} \right] \quad .$$

des calculs similaires donnent

$$P_0(\lambda) = \rho \left[G(n, \lambda)(1 - \lambda) + \frac{n^n}{n!}\frac{1}{\lambda^{Kq-n+1}} \right] \quad .$$

En utilisant l'égalité 4.2, on obtient

$$\lambda(1-P_K(\lambda)) = \rho \left[\lambda G(n, \lambda) + G(n, \frac{1}{\lambda}) + \lambda\frac{n^n}{n!}(1 + F(n, \lambda) + F(n, \frac{1}{\lambda}) - \lambda^{Kq-n+1} \right] \quad ,$$

et de même,

$$(1-P_0(\lambda)) = \rho \left[\lambda G(n, \lambda) + G(n, \frac{1}{\lambda}) + \frac{n^n}{n!}(1 + F(n, \lambda) + F(n, \frac{1}{\lambda}) - \frac{1}{\lambda^{Kq-n+1}} \right] \quad ,$$

l'équation d'équilibre est donc équivalente à

$$(1 - \lambda) \left(1 + F(n, \lambda) + F(n, \frac{1}{\lambda}) \right) = \frac{\lambda^{n-1}}{\lambda^{Kq}} - \lambda\frac{\lambda^{Kq}}{\lambda^{n-1}}$$

ce qui se vérifie aisément à partir de l'égalité 4.1.

■

La détermination de la valeur des $P_i(\frac{\lambda}{\mu})$ suffit pour calculer les valeurs de tous les indices de performance pertinents dans ce modèle. En effet, ces indices sont liés aux probabilités stationnaires pour un processeur d'avoir une charge i par des relations algébriques données dans la proposition 3. Pour alléger les notations, et sans perte de généralité, ces équations sont écrites avec un temps moyen d'exécution égal à 1, ce qui revient à fixer l'unité de temps $\frac{1}{\mu} = 1$.

Proposition 3. *Avec un temps moyen d'exécution de tâche égal à 1, les probabilités stationnaires et les autres indices de performance sont liés de la façon suivante :*

$$\begin{aligned} P_{sat} &= P_K(\lambda) \quad , \\ \overline{T} &= n\lambda(1 - P_K(\lambda) \quad , \\ \overline{N} &= n\left[\sum_{j=1}^{K} jP_j(\lambda)\right] \quad , \\ \overline{R} &= 1 + \frac{\sum_{j=2}^{K}(j-1)P_j(\lambda)}{\lambda(1 - P_K(\lambda))} \quad . \end{aligned}$$

La première et la troisième équation sont aisément obtenues. La deuxième équation donne le nombre moyen de tâches entrant dans le système par unité de temps. La démonstration de cette formule donnée dans le cas particulier $n = 2$, et $K = 2$ dans le chapitre 3, section 3.2 se généralise à ce système. La dernière équation exprime le temps de réponse moyen d'une tâche comme la somme du temps de service moyen (ici, $\frac{1}{\mu} = 1$) et du temps d'attente moyen, et cette équation généralise celle obtenue dans la section 3.2.

Démonstration

La dernière formule de la proposition 3 s'obtient de la façon suivante. On vérifie que les hypothèses de validité de la formule de Little (*cf.* [59] démontrée par exemple dans [99] p. 100) sont effectivement vérifiées dans ce contexte. Ceci nous permet d'écrire

$$\overline{R} = \frac{\overline{N}}{\overline{T}} \quad ,$$

Or,

$$\begin{aligned} \frac{\overline{N}}{n} &= \sum_{j=1}^{K} P_j(\lambda) + \sum_{j=2}^{K}(j-1)P_j(\lambda) \quad , \\ &= 1 - P_0(\lambda) + \sum_{j=2}^{K}(j-1)P_j(\lambda) \quad , \\ &= \lambda(1 - P_K(\lambda)) + \sum_{j=2}^{K}(j-1)P_j(\lambda) \quad . \end{aligned}$$

D'où le résultat. ■

La suite de cette section utilise les résultats classiques sur les processus de naissance et de mort pour lesquels une référence est par exemple Barucha-Reid [5].

4.2.2 Résultats sans transfert

Sans transfert, les n processeurs se comportent comme n files $M/M/1/K$ indépendantes. Les techniques habituelles permettent d'obtenir la mesure stationnaire du processus $\{X_t,\, t \geq 0\}$, et les résultats suivants en découlent.

Proposition 4. *Pour le modèle n sites de capacité K de taux d'arrivée λ, les valeurs des indices de performance sans politique de transfert sont les suivantes*

$$\begin{array}{lrcll}
\forall\, j \in \{0 \ldots K\} & P_j^*(\lambda) & = & \lambda^j \frac{1-\lambda}{1-\lambda^{K+1}} & si\ \lambda \neq 1, \quad = \frac{1}{K+1} \quad sinon \quad , \\
& P_{sat}^* & = & \lambda^K \frac{1-\lambda}{1-\lambda^{K+1}} & si\ \lambda \neq 1, \quad = \frac{1}{K+1} \quad sinon \quad , \\
& \overline{T}^* & = & n\lambda(\frac{1-\lambda^K}{1-\lambda^{K+1}}) & si\ \lambda \neq 1, \quad = n\frac{K}{K+1} \quad sinon \quad , \\
& \overline{N}^* & = & n\lambda(\frac{1}{1-\lambda} - \frac{(K+1)\lambda^K}{1-\lambda^{K+1}}) & si\ \lambda \neq 1, \quad = n\frac{K}{2} \quad sinon \quad , \\
& \overline{R}^* & = & 1 + \frac{\lambda}{1-\lambda} - \frac{K\lambda^K}{1-\lambda^K} & si\ \lambda \neq 1, \quad = \frac{1+K}{2} \quad sinon \quad .
\end{array}$$

4.2.3 Résultats avec transfert

Pour étudier le système avec transfert, il est opportun d'utiliser les procédés d'agrégation (*cf.* [78]). L'observation clé est de constater que le nombre total L_t de tâches présentes à l'instant t dans le système évolue comme un processus de naissance et de mort sur $\{0, \ldots, Kn\}$ avec des taux de naissance (de j à $j+1$)

$$\lambda(j) = \begin{cases} n\lambda & \text{pour } j = 0, \ldots, (K-1)n \\ (Kn-j)\lambda & \text{pour } j = (K-1)n, \ldots, Kn-1. \end{cases}$$

et des taux de mort (de j à $j-1$)

$$\mu(j) = \begin{cases} j & \text{pour } j = 1, \ldots, n \\ n & \text{pour } j = n+1, \ldots, Kn. \end{cases}$$

Pour vérifier ces résultats, il suffit pour les taux de naissance de faire les remarques suivantes. Tant qu'il y a moins de $n(K-1)$ tâches dans le système, tous les processeurs ont moins de $K-1$ tâches et donc toute nouvelle tâche générée par le système est acceptée.
S'il y a plus de $n(K-1)$ tâches, tous les processeurs ayant une charge égale à K provoquent des rejets.
Soit $(p_j)_{j=0\ldots Kn}$ la mesure stationnaire du processus de naissance et de mort $\{L_t,\, t \geq 0\}$. Cette mesure stationnaire suffit à déterminer celle du processus $\{X_t,\, t \geq 0\}$. En effet les valeurs des $(p_j)_{j=0\ldots Kn}$ sont liées aux valeurs des $(P_i(\lambda))_{i=0\ldots K}$ par les relations algébriques données dans la proposition 5.

Proposition 5. *Les probabilités* $(P_i(\lambda))_{i=0\ldots K}$ *sont reliées aux* $(p_j)_{j=0\ldots Kn}$ *de la façon suivante :*

$$P_0(\lambda) = \sum_{i=0}^{n-1} \frac{n-i}{n} p_i \quad , \quad P_K(\lambda) = \sum_{i=0}^{n-1} \frac{n-i}{n} p_{Kn-i} \quad ,$$

$$\forall\ 0 < j < K\ , \quad P_j(\lambda) = \sum_{i=0}^{n} \frac{i}{n} p_{(j-1)n+i} + \sum_{i=1}^{n-1} \frac{n-i}{n} p_{jn+i} \quad .$$

Démonstration

Ces équations découlent de la constatation suivante. Lorsqu'il y a $(j-1)n+i$ tâches dans le système, i processeurs ont j tâches, et $n-i$ ont $j-1$ tâches. ■

Il est donc nécessaire d'exprimer la valeur de chaque p_j en fonction des paramètres λ, $n = 2q$ et K du système.
En respectant la symétrie du système, et par les techniques habituelles, on obtient les formules algébriques suivantes.

$\forall \quad 0 \leq j \leq n$

$$\begin{aligned} p_j &= \frac{n^j}{j!}\lambda^j p_0 = \frac{n^j}{j!}\frac{1}{\lambda^{Kq-j}}\lambda^{Kq} p_0 \quad , \\ p_{Kn-j} &= \frac{n^j}{j!}(\frac{1}{\lambda})^j p_{Kn} = \frac{n^j}{j!}\lambda^{Kq-j}\frac{1}{\lambda^{Kq}} p_{Kn} \quad , \end{aligned}$$

$\forall \quad n \leq j \leq Kq$

$$\begin{aligned} p_j &= \frac{n^n}{n!}\lambda^j p_0 = \frac{n^n}{n!}\frac{1}{\lambda^{Kq-j}}\lambda^{Kq} p_0 \quad , \\ p_{Kn-j} &= \frac{n^n}{n!}(\frac{1}{\lambda})^j p_{Kn} = \frac{n^n}{n!}\lambda^{Kq-j}\frac{1}{\lambda^{Kq}} p_{Kn} \quad , \end{aligned}$$

Posons

$$\rho = p_{Kn}\frac{1}{\lambda^{Kq}} = \lambda^{Kq} p_0 \quad .$$

Notons $G(n,\lambda)$ et $F(n,\lambda)$ les sommes

$$G(n,\lambda) = \sum_{j=0}^{n-2} \frac{n^j}{j!}(\frac{1}{\lambda})^{Kq-j} \quad ,$$

$$F(n,\lambda) = \sum_{j=n-1}^{Kq-1} \frac{1}{\lambda^{Kq-j}} \quad = \frac{1}{\lambda^{Kq}}\left(\frac{\lambda^{n-1} - \lambda^{Kq}}{1-\lambda}\right) \quad . \tag{4.1}$$

En utilisant les relations évidentes $\sum_{j=0}^{j=Kn} p_k = 1$ et $\frac{n^n}{n!} = \frac{n^{n-1}}{(n-1)!}$, on obtient,

$$\rho \left[G(n,\lambda) + G(n,\frac{1}{\lambda}) + \frac{n^n}{n!} \left(1 + F(n,\lambda) + F(n,\frac{1}{\lambda}) \right) \right] = 1 \quad . \tag{4.2}$$

D'où ρ, puis p_0 puis la mesure stationnaire, puis les probabilités $P_j(\lambda)$.
Notons également $H(n,\lambda)$ la quantité suivante

$$H(n,\lambda) = \frac{1}{n}(\sum_{j=n}^{Kq-1} \frac{j}{\lambda^{Kq-j}} + (Kn-j)\lambda^{Kq-j}) \quad .$$

Tous les indices de performance peuvent alors s'exprimer directement en fonction des paramètres λ, $n = 2q$ et K du système et des fonctions G, F et H précédemment définies.

Proposition 6. *Pour le modèle à n processeurs avec transfert, les valeurs des indices de performance en fonction de $n = 2q$, λ et K sont les suivantes*

$$\begin{aligned}
P_{sat} &= \left[G(n,\frac{1}{\lambda})(1-\frac{1}{\lambda}) + \frac{n^n}{n!}\lambda^{Kq-n+1} \right] \rho \quad , \\
\overline{T} &= n\lambda(1-P_{sat}) \quad , \\
\overline{N} &= \rho n \left[G(n,\frac{1}{\lambda})(K-\frac{1}{\lambda}) + \frac{n^n}{n!}(\frac{K}{2} + K\lambda^{Kq-(n-1)}) + H(n,\lambda) \right] \quad , \\
\overline{R} &= \frac{\lambda G(n,\lambda) + G(n,\frac{1}{\lambda})(K-\frac{1}{\lambda}) + \frac{n^n}{n!}(K\lambda^{Kp-n-1} + \frac{K}{2} + H(n,\lambda))}{\lambda G(n,\lambda) + G(n,\frac{1}{\lambda}) + \frac{n^n}{n!}(1 + F(n,\lambda) + F(n,\frac{1}{\lambda}) - \lambda^{Kp-n+1})} \quad . \\
&= \frac{\overline{N}}{\overline{T}} \quad .
\end{aligned}$$

Démonstration

Voici la démonstration du calcul des indices en fonction des paramètres $n = 2q$, λ et K. On a :

$$\begin{aligned}
P_{sat} &= \sum_{j=0}^{n} \frac{n-j}{n} p_{Kn-j} \quad , \\
&= \sum_{j=0}^{n-2} p_{Kn-j} + p_{Kn-n+1} + p_{Kn-n} - \sum_{j=1}^{n} \frac{j}{n} p_{Kn-j} \quad , \\
&= \rho \left[\sum_{j=0}^{n-2} \frac{n^j}{j!}\lambda^{Kq-j} + \frac{n^n}{n!}(\lambda^{Kq-n+1} + \lambda^{Kq-n}) - \frac{1}{\lambda}\sum_{j=0}^{n-1} \frac{n^j}{j!}\lambda^{Kq-j} \right] \quad , \\
&= \rho \left[G(n,\frac{1}{\lambda}) + \frac{n^n}{n!}\lambda^{Kq-n+1} - \frac{1}{\lambda}G(n,\frac{1}{\lambda}) \right] \quad , \\
&= \rho \left[G(n,\frac{1}{\lambda})(1-\frac{1}{\lambda}) + \frac{n^n}{n!}\lambda^{Kq-n+1} \right] \quad .
\end{aligned}$$

De même, on a

$$\begin{aligned}\overline{N} &= \sum_{j=1}^{n-1} jp_j + \sum_{j=0}^{n-1}(Kn-j)p_{Kn-j} \\ &\quad + Kqp_{kn} + \sum_{j=n}^{Kq-1} jp_j + \sum_{j=n}^{Kq-1}(Kn-j)p_{Kn-j} \quad ,\end{aligned}$$

La deuxième ligne du membre de droite de cette équation vaut

$$\rho\frac{n^n}{n!}\left[\sum_{j=n}^{Kq-1}\frac{j}{\lambda^{Kq-j}} + (Kn-j)\lambda^{Kq-j} + Kq\right] \quad .$$

Tandis que la première ligne de cette équation vaut

$$\begin{aligned}&\rho\left[\sum_{j=1}^{n-1}\frac{n.n^{j-1}}{(j-1)!}\frac{\lambda}{\lambda^{Kq-j+1}} + \sum_{j=0}^{n-1}Kn\frac{n^j}{j!}\lambda^{Kq-j} - \sum_{j=1}^{n-1}\frac{n.n^{j-1}}{(j-1)!}\frac{\lambda^{Kq-j+1}}{\lambda}\right] \quad , \\ = \;& \rho\left[n\lambda G(n,\lambda) + Kn\left[G(n,\frac{1}{\lambda}) + \frac{n^n}{n!}\lambda^{Kq-n+1}\right] - \frac{n}{\lambda}G(n,\frac{1}{\lambda})\right] \quad .\end{aligned}$$

D'où

$$\overline{N} = \rho n\left[\lambda G(n,\lambda) + G(n,\frac{1}{\lambda})(K-\frac{1}{\lambda}) + \frac{n^n}{n!}(K\lambda^{Kq-n+1} + \frac{K}{2} + H(n,\lambda)\right] \quad .$$

Enfin,

$$\overline{R} = \frac{\overline{N}}{n\lambda(1-P_{sat})} \quad ,$$

or,

$$\begin{aligned}1 &= \rho\left[G(n,\lambda) + G(n,\frac{1}{\lambda}) + \frac{n^n}{n!}(1+F(n,\lambda)+F(n,\frac{1}{\lambda})\right] \quad , \\ P_{sat} &= \rho\left[G(n,\frac{1}{\lambda})(1-\frac{1}{\lambda}) + \frac{n^n}{n!}\lambda^{Kq-n+1}\right] \quad ,\end{aligned}$$

D'où

$$\overline{R} = \frac{\lambda G(n,\lambda) + G(n,\frac{1}{\lambda})(K-\frac{1}{\lambda}) + \frac{n^n}{n!}(K\lambda^{Kq-n-1} + \frac{K}{2} + H(n,\lambda))}{\lambda G(n,\lambda) + G(n,\frac{1}{\lambda}) + \frac{n^n}{n!}(1+F(n,\lambda)+F(n,\frac{1}{\lambda}) - \lambda^{Kq-n+1})} \quad .$$

■

Afin de rendre compte de l'influence de chacun des paramètres sur les indices de performance, plusieurs situations semblent intéressantes à étudier. Pour un nombre de processeurs n fixé de capacité mémoire K fixée, on peut par exemple examiner le comportement des indices de performance quand le taux d'arrivée des tâches λ varie. Cette étude est détaillée dans la section 4.3.

Inversement, pour un taux d'arrivée λ fixé et une capacité K fixée, on peut observer le comportement du système quand le nombre n de processeurs augmente. Cette étude sera détaillée en section 4.4.

Enfin, pour des valeurs de n et λ appropriées, on peut s'interroger sur des valeurs pertinentes de la capacité K à choisir. Cette étude sera abordée en section 4.5.3.

4.3 Comparaison pour n fixé

Dans cette section nous allons donner, pour fixer les idées, les résultats obtenus pour des valeurs particulières du nombre de processeurs et de la capacité mémoire K, par exemple $n = 8$, $n = 32$ ou $n = 128$ et $K = 2,\ 3\ ou\ 6$. L'observation du comportement du sytème pour ces valeurs particulières permet d'extrapoler celui obtenu pour les autres valeurs. Les étude numériques ont été effectuées avec *Mathematica*.

4.3.1 Comparaison des probabilités stationnaires

Pour des valeurs de n et K fixées, les probabilités stationnaires $P_i(\lambda)$ évoluent en fonction de la valeur du taux d'arrivée λ, et les valeurs de ces probabilités stationnaires sont très différentes selon qu'une politique de transfert est mise ou non en place. Les figures 4.3, 4.4, et 4.5 par exemple représentent l'évolution des probabilités stationnaires $P_0(\lambda), P_1(\lambda), P_2(\lambda)$ sans transfert et avec transfert pour $K = 6$, $n = 8$, $n = 32$ et $n = 128$, et montrent graphiquement ces différences de comportement. Lorsque le taux d'arrivée des tâches λ est de l'ordre du taux de service moyen, c'est-à-dire autour de $\lambda = 1$, notons le brusque changement de comportement de ces probabilités dans la situation avec transfert.

Les valeurs respectives de chacune de ces probabilités les unes par rapport aux autres sont également intéressantes à examiner. Les figures 4.6, 4.7 et 4.8 montrent les évolutions relatives des probabilités P_0, P_1, P_2, P_3 avec transfert pour $n = 8$, $n = 32$ et $n = 128$.

La valeur de ces probabilités est modifiée autour de $\lambda = 1$ et cette modification est d'autant plus accentuée que le nombre n de processeurs est grand. Pour plus de précision nous présentons des tables donnant les valeurs numériques de ces probabilités obtenues pour $n = 32$, $K = 6$ et deux valeurs de λ représentatives du comportement du système. En effet, compte tenu de la proposition 2, les valeurs obtenues pour $\lambda \leq 1$ suffisent, et nous avons choisi les valeurs $\lambda = 0.8$ et $\lambda = 1$.

La table 4.1 donne ces différentes probabilités pour $\lambda = 0.8$, $n = 32$, $K = 6$, ainsi que la charge moyenne par site notée CM dans cette table.

$\lambda = 0.8$	P_0	P_1	P_2	P_3	P_4	P_5	P_6	CM
sans transfert	0.253	0.202	0.162	0.13	0.104	0.083	0.066	2.143
avec transfert	0.2	0.78	0.02	0.0	0.0	0.0	0.0	0.82
	P_6	P_5	P_4	P_3	P_2	P_1	P_0	$\lambda = 1.25$

Table 4.1 : Table des probabilités stationnaires pour 32 sites de capacité $K = 6$ avec $\lambda = 0.08$ et $\lambda = 1.25$.

La table 4.2 donne ces mêmes quantités pour $\lambda = 1$, $n = 32$, $K = 6$.

$\lambda = 1$	P_0	P_1	P_2	P_3	P_4	P_5	P_6	CM
sans transfert	0.143	0.143	0.143	0.143	0.143	0.143	0.143	3.003
avec transfert	0.007	0.156	0.224	0.224	0.224	0.156	0.007	2.994

Table 4.2 : Table des probabilités stationnaires pour 32 sites de capacité $K = 6$ avec $\lambda = 1.0$.

Ces résultats appellent quelques commentaires. Pour $\lambda = 0.8$ et plus généralement pour $\lambda < 1$, les valeurs obtenues sans transfert sont toutes significatives tandis qu'avec transfert, seule une très faible proportion des processeurs supporte une charge supérieure ou égale à 2. Dans le cas particulier de $\lambda = 0.8$, 98% des processeurs ont une charge égale à 0 ou à 1. Autrement dit, le transfert tend à homogénéiser la charge des processeurs et une proportion de l'ordre de λ d'entre eux ont une charge égale à 1.
Pour $\lambda = 1$, toutes les probabilités sont également non nulles sans transfert, et elles sont toutes égales. Avec transfert, les charges extrêmes, 0 et K (dans l'exemple 0 et 6) sont très peu probables, les charges 1 et $K - 1$ (dans l'exemple 1 et 5) sont non négligeables, et toutes les autres charges de 2 à $K - 2$ (dans l'exemple de 2 à 4) sont équiprobables. Ces résultats seront confirmés dans la section 4.4.1.
La différence des comportements du système obtenus avec et sans politique de transfert semble donc assez nette et il semble intéressant d'en mesurer l'incidence sur les indices de performance particulièrement sensibles, la probabilité de saturation mémoire et le temps de réponse moyen.

4.3.2 Comparaison des probabilités de saturation mémoire

Pour des valeurs de n et K fixées, la probabilité de saturation mémoire P_{sat} évolue en fonction de la valeur du taux d'arrivée λ, et de même que précédemment, la valeur de cette probabilité

est très différente selon qu'une politique de transfert est mise ou non en place. La valeur de la capacité K n'altère pas les différences de comportement obtenues. La figure 4.9 montre pour une capacité mémoire $K = 2$ les évolutions des probabilités de saturation obtenues sans transfert et avec transfert pour $n = 8$, $n = 32$ et $n = 128$. et la figure 4.10 montre les mêmes évolutions obtenues pour une capacité $K = 6$.

Le transfert améliore significativement cet indice, et la différence $P^*_{sat} - P_{sat}$ est d'autant plus importante que le nombre n de processeurs augmente. Nous reviendrons sur l'étude de cette différence lors de l'étude du comportement asymptotique en section 4.4.

4.3.3 Comparaison des temps de réponse moyens

Rappelons que sans transfert, les processeurs sont indépendants et le temps de réponse moyen d'une tâche ne dépend pas du nombre n de processeurs. Avec transfert en revanche le temps de réponse moyen d'une tâche est fortement dépendant du nombre total de processeurs *i.e.* de n. De plus, pour un nombre n donné de processeurs, le comportement du temps de réponse moyen dépend de la capacité mémoire K de chaque site.
Il convient en effet de distinguer les valeurs $K = 2$, $K = 3$ et $K \geq 4$.
La figure 4.11 montre les évolutions des temps de réponse moyens de n sites de capacité $K = 2$ obtenues sans transfert d'une part et avec une politique de transfert d'autre part pour les valeurs $n = 8$ et $n = 32$.
La figure 4.12 montre les mêmes évolutions obtenues pour une capacité $K = 3$, et la figure 4.13 celles obtenues pour une capacité $K = 6$.

Ces graphiques appellent quelques commentaires. Le temps de réponse moyen d'une tâche n'est donc pas nécessairement amélioré par le transfert, et il convient ici d'expliciter les comportements observés selon les valeurs des paramètres λ et K.

Pour $\lambda < 1$. La politique de transfert diminue le temps de réponse moyen d'une tâche par rapport à celui obtenu dans la situation sans transfert et ce, pour toutes les valeurs de la capacité K. L'explication intuitive de ce comportement est simple. En effet, pour un taux d'arrivée des tâches inférieur (strictement) au taux de service, le système est non saturé et d'après les études faites sur la probabilité de charge d'un processeur, celui-ci a très probablement une charge égale soit à 0 soit à 1. Autrement dit, une tâche acceptée dans le système sera vraisemblablement transférée sur un processeur libre, et sera donc traitée plus rapidement par le système doté d'une politique de transfert que par celui sans politique de transfert.

Pour $\lambda = 1$. Pour toutes les valeurs de la capacité K et du nombre n de processeurs, la politique de transfert diminue le temps de réponse moyen d'une tâche. La table 4.3 donne

les valeurs de différents temps de réponse obtenus sans transfert et avec une politique de transfert pour les différentes valeurs pertinentes des paramètres K et n.

$\lambda = 1$	$K = 2$	$K = 3$	$K = 6$
$n = 8$	$\overline{R}^* = 1.5$	$\overline{R}^* = 2.0$	$\overline{R}^* = 3.5$
	$\overline{R} = 1.154$	$\overline{R} = 1.604$	$\overline{R} = 3.078$
$n = 32$	$\overline{R}^* = 1.5$	$\overline{R}^* = 2.0$	$\overline{R}^* = 3.5$
	$\overline{R} = 1.074$	$\overline{R} = 1.533$	$\overline{R} = 3.021$

Table 4.3 : Table des valeurs du temps de réponse moyen pour $\lambda = 1$.

Pour $\lambda > 1$. Il convient dans ce cas de distinguer les différents comportements du système selon la valeur de la capacité mémoire K.

Pour $K = 2$. Bien que le système soit plutôt en surcharge, la politique de transfert diminue le temps de réponse moyen d'une tâche. Intuitivement, cela s'explique par le fait que le transfert d'une tâche ne peut être effectué que pour une tâche générée par un processeur de charge 1 qui la transfère sur un processeur libre, et la politique de transfert ne modifie le comportement du système que dans cette situation. La valeur moyenne du temps de réponse d'une tâche est donc plus faible avec la politique de transfert que celle obtenue sans transfert. Notons néanmoins que lorsque la valeur du taux d'arrivée λ augmente cette situation devient de moins en moins probable car la proportion de processeurs oisifs devient voisine de 0, et dans ce cas la politique de transfert ne peut plus apporter de gain de temps notoire.

Pour $K > 2$. Le système est toujours en surcharge, et la politique de transfert dans ce cas augmente le temps de réponse moyen d'une tâche. Autrement dit, le temps de réponse moyen d'une tâche générée et acceptée par le système est pénalisé par la politique de transfert. Pour lever cet apparent paradoxe, il faut noter que dans la situation sans transfert, une tâche acceptée par le système a une probabilité non nulle d'être dans les premières à être traitée. La politique de transfert permet en fait d'augmenter le nombre total de tâches acceptées par le système, mais chaque tâche acceptée ne sera pas transférée sur un site sous-chargé et attendra que toutes les tâches qui la précède soient traitées. La politique de transfert permet donc de privilégier et d'augmenter l'indice $\overline{T}$, mais le temps de réponse d'une tâche particulière est, en moyenne, pénalisé.

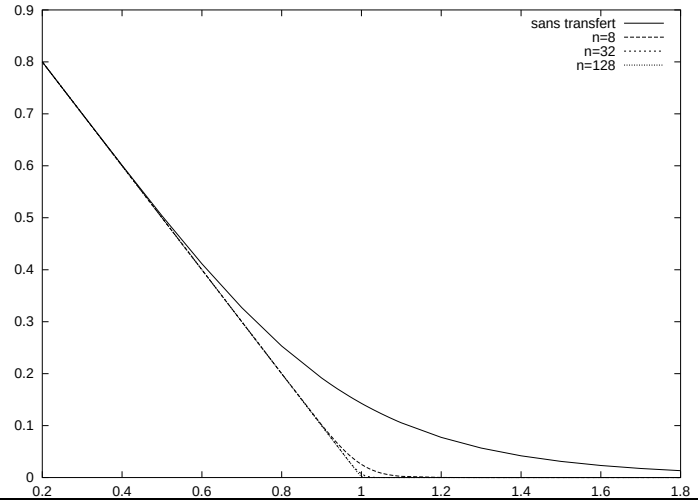

Figure 4.3 : Évolution de $P_0(\lambda)$ fonction de λ, pour n sites de capacité $K = 6$.

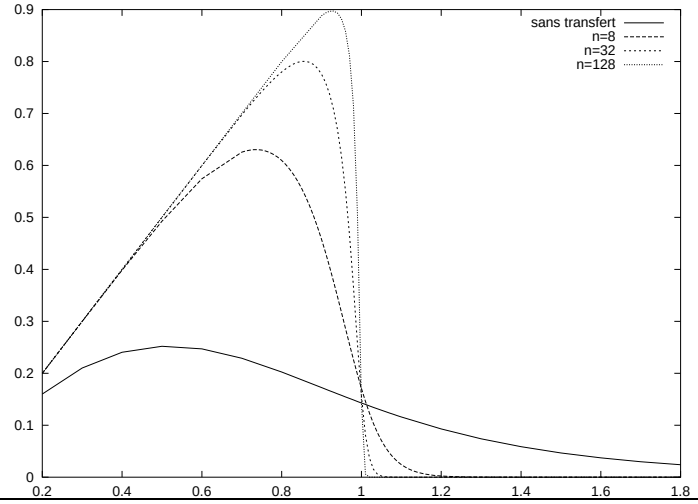

Figure 4.4 : Évolution de $P_1(\lambda)$ fonction de λ, pour n sites de capacité $K = 6$.

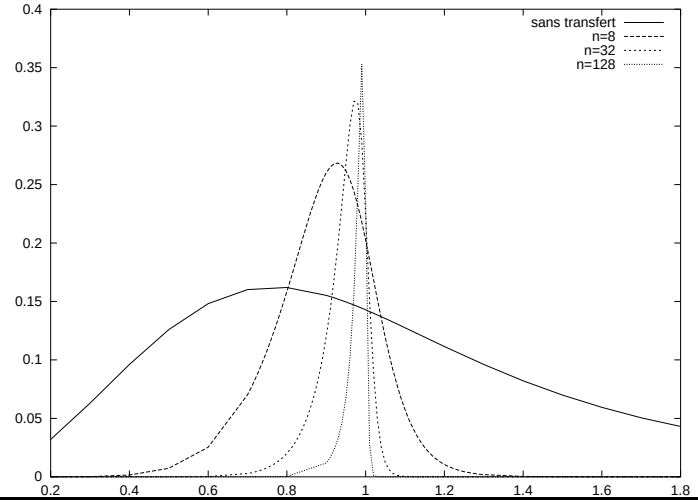

Figure 4.5 : Évolution de $P_2(\lambda)$ fonction de λ, pour n sites de capacité $K = 6$.

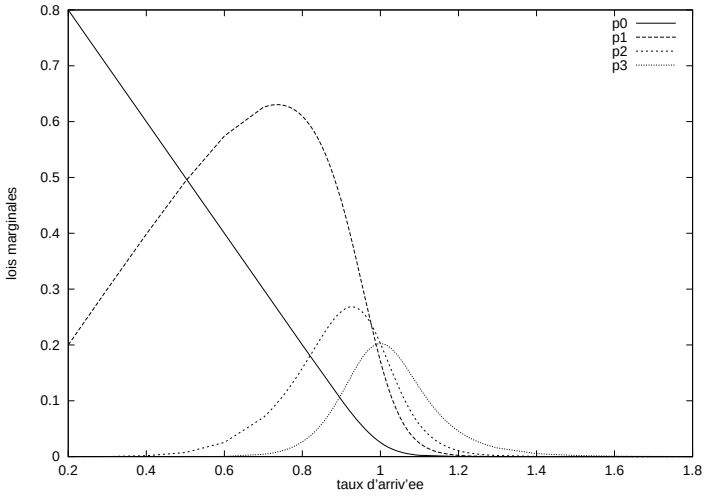

Figure 4.6 : Évolutions relatives de $P_0(\lambda), P_1(\lambda), P_2(\lambda), P_3(\lambda)$ pour $n = 8$ sites de capacité $K = 6$.

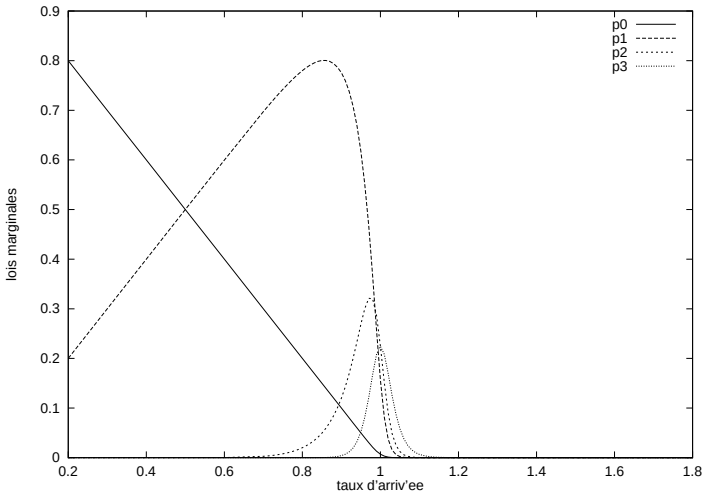

Figure 4.7 : Évolutions relatives de $P_0(\lambda), P_1(\lambda), P_2(\lambda), P_3(\lambda)$ pour $n = 32$ sites de capacité $K = 6$.

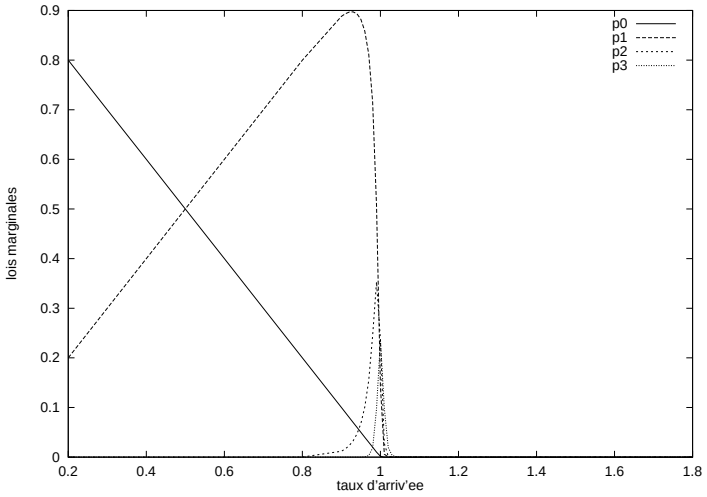

Figure 4.8 : Évolutions relatives de $P_0(\lambda), P_1(\lambda), P_2(\lambda), P_3(\lambda)$ pour $n = 128$ sites de capacité $K = 6$.

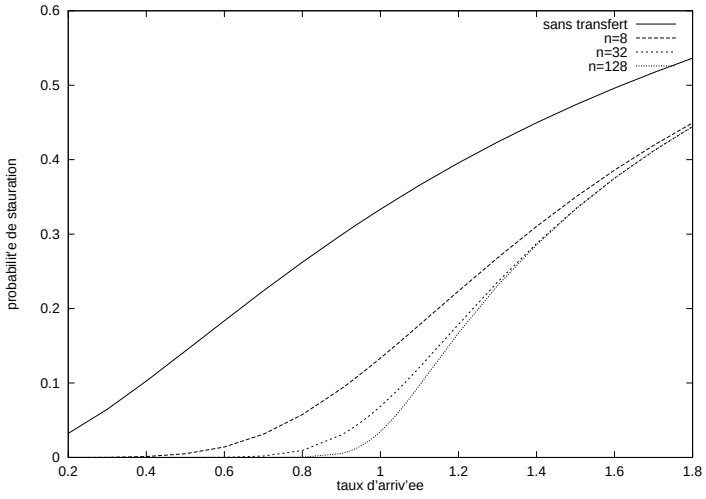

Figure 4.9 : Évolutions de la probabilité de saturation de n sites de capacité $K = 2$.

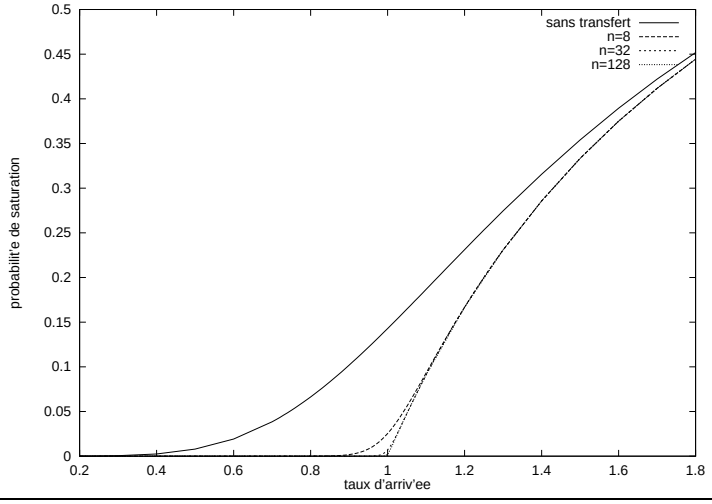

Figure 4.10 : Évolutions de la probabilité de saturation de n sites de capacité $K = 6$.

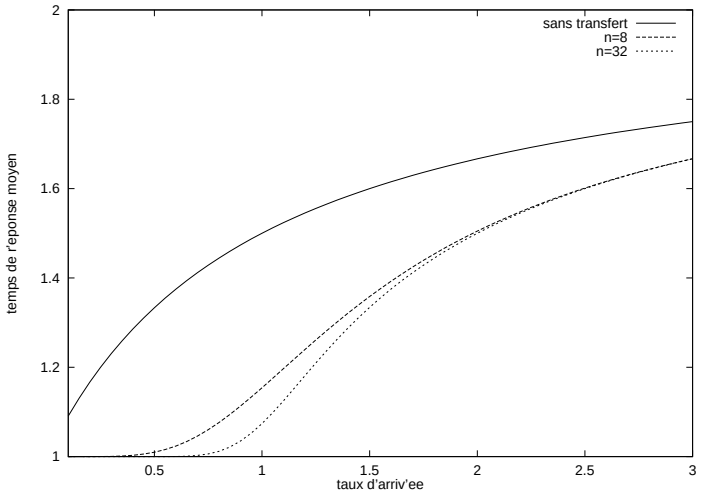

Figure 4.11 : Évolutions des temps de réponse moyens sur n sites de capacité $K = 2$.

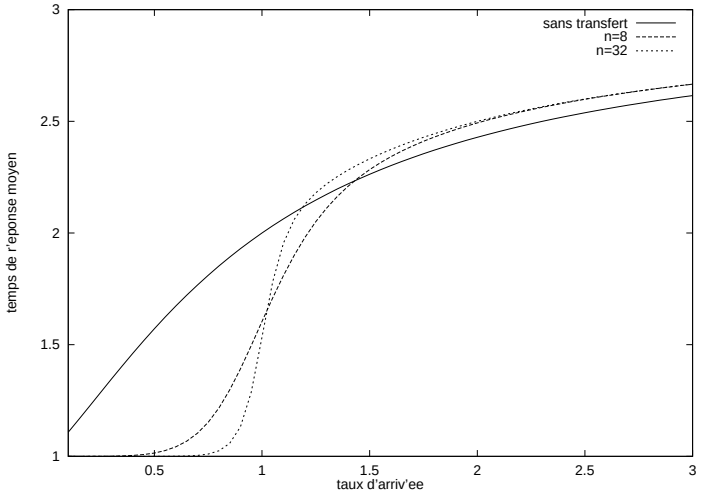

Figure 4.12 : Évolutions des temps de réponse moyens sur n sites de capacité $K = 3$.

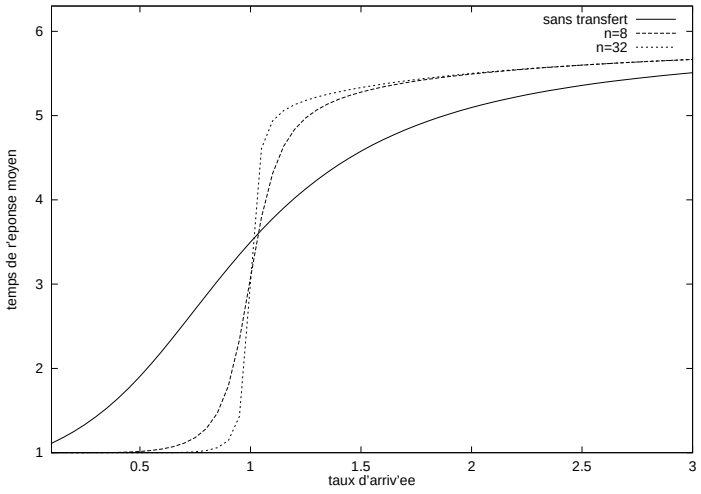

Figure 4.13 : Évolutions des temps de réponse moyens sur n sites de capacité $K = 6$.

4.4 Comportement asymptotique des systèmes massivement parallèles

Le fait le plus marquant que l'on peut observer sur les courbes obtenues dans la section 4.3 est que le passage de 8 à 32 sites n'améliore pas de manière significative les résultats sur les indices de performances P_{sat} et $\overline{R}$. Ceci laisse à penser qu'en ayant des contraintes de localité faible (par exemple quand chaque site est connecté à 8 autres sites seulement) des performances similaires à celle d'un réseau totalement connecté peuvent être espérées. De telles situations seront étudiées au chapitre 5. Il est néanmoins très intéressant de s'interroger sur le comportement du système totalement connecté lorsque le nombre de sites devient grand. Les formules obtenues analytiquement permettent d'obtenir le comportement asymptotique de ces systèmes dits massivement parallèles. La section 4.4.1 donne les théorèmes de convergence et les valeurs limites des indices de performance obtenues analytiquement pour n tendant vers l'infini. La section 4.4.2 compare les résultats numériques obtenus pour ces indices pour des valeurs classiques de n (par exemple $n = 32, 64, 128$) avec les valeurs asymptotiques précédemment calculées.

4.4.1 Théorème de convergence

Pour un taux d'arrivée λ donné, et une capacité mémoire K pour chaque processeur, les probabilités stationnaires $P_i(\lambda)$ tendent vers une limite quand n tend vers l'infini.

Théorème 1. *Soit $K \geq 3$ la capacité de chaque processeur, λ le taux d'arrivée des tâches sur chaque processeur. Quand n tend vers l'infini, les probabilités de chaque niveau de charge du modèle complet à n sommets admettent les limites suivantes.*

$\lambda < 1$			$\lambda = 1$			$\lambda > 1$		
$P_0(\lambda)$	$\to$	$1-\lambda$	$P_0(\lambda)$	$\to$	0	$P_0(\lambda)$	$\to$	0
$P_1(\lambda)$	$\to$	λ	$P_1(\lambda)$	$\to$	$1/(2K-4)$	$P_1(\lambda)$	$\to$	0
	$\vdots$			$\vdots$			$\vdots$	
$P_i(\lambda)$	$\to$	0	$P_i(\lambda)$	$\to$	$1/(K-2)$	$P_i(\lambda)$	$\to$	0
	$\vdots$			$\vdots$			$\vdots$	
$P_{K-1}(\lambda)$	$\to$	0	$P_{K-1}(\lambda)$	$\to$	$1/(2K-4)$	$P_{K-1}(\lambda)$	$\to$	$1/\lambda$
$P_K(\lambda)$	$\to$	0	$P_K(\lambda)$	$\to$	0	$P_K(\lambda)$	$\to$	$1-1/\lambda$

Pour $K = 2$, ces résultats restent vrais sauf pour la valeur $\lambda = 1$, auquel cas $P_1(\lambda)$ tend vers 1.

Pour $K > 2$, ces limites présentent une discontinuité en $\lambda = 1$. Un tel comportement irrégulier a déjà été observé dans un modèle différent étudié dans [62].

Démonstration

Voici une démonstration du théorème 1. Compte tenu de la proposition 2, il suffit de démontrer ce théorème pour $\lambda \leq 1$. Rappelons que le nombre n de processeurs est supposé pair et que $n = 2q$ (voir 4.2).

- Pour $P_0(\lambda)$, on obtient,

$$\begin{aligned} P_0(\lambda) &= \sum_{i=0}^{n-1} \frac{n-i}{n} p_i \quad , \\ &= \sum_{i=0}^{n-1} \frac{n-i}{n} \frac{n^i}{i!} \lambda^i p_0 \quad , \\ &= \sum_{i=0}^{n-1} \frac{n^i}{i!} \frac{1}{\lambda^{Kq-i}} \rho - \sum_{i=1}^{n-1} \frac{n^{i-1}}{(i-1)!} \frac{1}{\lambda^{Kq-i+1}} \rho \quad , \\ &= \frac{(1-\lambda) G(n,\lambda) + \frac{n^n}{n!} (\frac{1}{\lambda^{Kq-n+1}})}{G(n,\lambda) + G(n,\frac{1}{\lambda}) + \frac{n^n}{n!}(1 + F(n,\lambda) + F(n,\frac{1}{\lambda}))} \quad . \end{aligned}$$

Il s'agit donc de déterminer le comportement asymptotique de $G(n,\lambda)$. Ecrivons

$$\lambda^{Kq} e^{-n\lambda} G(n,\lambda) = \sum_{j=0}^{j=n-2} \frac{(n\lambda)^j}{j!} e^{-n\lambda} \quad ,$$

Ceci est la valeur au point $n-2$ de la fonction de répartition d'une loi de Poisson de paramètre λn. D'après le théorème central limite, il vient

$$\begin{aligned} \lim_{n \to +\infty} \lambda^{Kq} e^{-n\lambda} G(n,\lambda) &= 1 \quad si \quad \lambda < 1 \\ &= \frac{1}{2} \quad si \quad \lambda = 1 \\ &= 0 \quad si \quad \lambda > 1 \quad . \end{aligned}$$

Il convient donc de distinguer les 2 situations possibles dans cette preuve.

Pour $\lambda = 1$, on obtient (formule de Stirling)

$$\frac{n^n}{n!} \times \frac{1}{G(n,1)} \sim \frac{1}{2\sqrt{2\pi n}} \quad ,$$

Pour $\lambda < 1$, on obtient

$$\frac{G(n,1)}{G(n,\lambda)} \sim \frac{1}{2} \left(\frac{\lambda^{\frac{K}{2}}}{e^{\lambda - 1}} \right)^n \quad .$$

D'où

$$\lim_{n\to+\infty} \frac{G(n,1)}{G(n,\lambda)} = 0 \quad ,$$

et par suite

$$\lim_{n\to+\infty} \frac{\frac{n^n}{n!}\lambda^{Kq-n+1}}{G(n,\lambda)} = 0 \quad .$$

Or, pour $\lambda < 1$,

$$G(n,\frac{1}{\lambda}) \leq G(n,1) \quad ,$$

Donc

$$\lim_{n\to+\infty} \frac{G(n,1/\lambda)}{G(n,\lambda)} = 0 \quad .$$

Ces résultats permettent de conclure :

Pour $\lambda = 1$: Le terme dominant de $P_0(1)$ est $G(n,1)$ au dénominateur, et par suite

$$\lim_{n\to+\infty} P_0(1) = 0 \quad .$$

Pour $\lambda < 1$: Le terme dominant du numérateur et du dénominateur de $P_0(\lambda)$ est $G(n,\lambda)$. D'où

$$\lim_{n\to+\infty} P_0(\lambda) = 1 - \lambda \quad .$$

- Pour $P_1(\lambda)$, on obtient,

$$\begin{aligned} P_1(\lambda) &= \sum_{i=0}^{n} \frac{i}{n} p_i + \sum_{i=1}^{n-1} \frac{n-i}{n} p_{n+i} \quad , \\ &= \frac{\lambda G(n,\lambda) + \frac{n^n}{n!}(\frac{1}{\lambda^{Kq-n+1}})}{G(n,\lambda) + G(n,\frac{1}{\lambda}) + \frac{n^n}{n!}(1 + F(n,\lambda) + F(n,\frac{1}{\lambda}))} \\ &\quad + \sum_{i=1}^{n-1} \frac{n-i}{n} \frac{n^n}{n!} (\frac{1}{\lambda^{Kq-(n+i)}})\rho \quad . \end{aligned}$$

En utilisant les mêmes arguments, on obtient,

Pour $\lambda < 1$: Le terme dominant des numérateurs et du dénominateur de $P_1(\lambda)$ est $G(n,\lambda)$. D'où

$$\lim_{n\to+\infty} P_1(\lambda) = \lambda \quad .$$

Pour $\lambda = 1$: Après calculs, pour $K \neq 2$, on obtient

$$P_1(1) = \frac{G(n,1) + \frac{n^n}{n!}(\frac{n}{2} + \frac{1}{2})}{2G(n,1) + \frac{n^n}{n!}[(K-2)n + 2]} \quad .$$

Or,

$$\lim_{n \to +\infty} \frac{G(n,1)}{n\frac{n^n}{n!}} = 0 \quad ,$$

Et par suite, pour $K \neq 2$,

$$\lim_{n \to +\infty} P_1(1) = \frac{1}{2(K-2)} \quad .$$

- Autres résultats :
  les autres résultats obtenus pour $\lambda < 1$ découlent de ceux obtenus pour $P_0(\lambda)$ et $P_1(\lambda)$.
  Enfin, pour $\lambda = 1$, la symétrie du graphe de transition du processus de naissance et de mort du processus agrégé $\{L_t,\, t \geq 0\}$ et les relations algébriques de la proposition 3 permettent d'écrire les équations suivantes,

$$\begin{aligned} P_0(1) &= P_K(1) \quad , \\ P_1(1) &= P_{K-1}(1) \quad , \\ P_2(1) &= P_3(1) = \ldots = P_{(K-2)n}(1) \quad . \end{aligned}$$

  D'où les autres résultats obtenus pour $\lambda = 1$ avec $K > 2$ ou $K = 2$.

■

Une justification plus intuitive de ce théorème peut être donnée de la façon suivante.
Le processus de naissance et de mort $\{L_t,\, t \geq 0\}$ se comporte sur l'intervalle $\{0 \ldots, n\}$ comme une file $M/M/\infty$, avec un taux d'arrivée $n\lambda$ et un taux de service 1. La mesure stationnaire de cette file suit une loi de Poisson de paramètre $n\lambda$. Pour $\lambda < 1$, la probabilité que $\{L_t,\, t \geq 0\}$ dépasse n tend vers 0 quand n augmente. Ainsi la mesure stationnaire du processus $\{L_t,\, t \geq 0\}$ tend vers une distribution de Poisson dont l'espérance est $n\lambda$. Pour des systèmes massivement parallèles (n grand), la plupart des processeurs ont une charge égale à 0 ou à 1. En moyenne, une proportion λ d'entre eux ont une charge égale à 1. D'où les résultats dans le cas des systèmes non saturés ($\lambda < 1$). Dans le cas $\lambda = 1$, le même argument concernant la distribution de Poisson permet de déduire que la probabilité qu'un site soit libre tend vers 0. Les mêmes arguments que ceux utilisés dans la démonstration permettent alors de conclure.

Les indices de performance d'intérêt P_{sat} et $\overline{R}$ étant reliés aux probabilités stationnaires par les relations de la proposition 3 ces indices admettent aussi des valeurs limites dont les notations et les valeurs numériques sont données dans le corollaire 1.

Corollaire 1. *Pour n très grand, le système avec transfert, présente trois types de comportement selon les valeurs du taux d'arrivée λ :*
Système non saturé : $\lambda < 1$

$$\lim_{n\to+\infty} P_{sat} = P_{sat}^{lim} = 0\,, \qquad \lim_{n\to+\infty} \overline{R} = \overline{R}^{lim} = 1\,,$$

Valeur critique : $\lambda = 1$

$$\lim_{n\to+\infty} P_{sat} = P_{sat}^{lim} = 0\,, \qquad \lim_{n\to+\infty} \overline{R} = \overline{R}^{lim} = \frac{K}{2}\,,$$

Système saturé : $\lambda > 1$

$$\lim_{n\to+\infty} P_{sat} = P_{sat}^{lim} = 1 - \frac{1}{\lambda}\,, \qquad \lim_{n\to+\infty} \overline{R} = \overline{R}^{lim} = K - \frac{1}{\lambda}\,.$$

Commentaires :
Ce comportement asymptotique révèle donc une transition de phase intéressante. Lorsque le taux de service est supérieur au taux d'arrivée, une tâche arrivant dans le système est presque toujours acceptée. Elle est presque toujours générée ou transférée sur un site sous-chargé, et son temps de réponse moyen est donc égal à son temps de service moyen, ici égal à 1.
Lorsque le taux de service est égal au taux d'arrivée, une tâche arrivant dans le système est également presque toujours acceptée, mais elle va, en moyenne, être proche de la moitié de la file d'attente, et son temps de réponse moyen est $\frac{K}{2}$.
Lorsque le taux de service est inférieur au taux d'arrivée, une tâche peut être refusée, et quand elle est acceptée, elle va presque toujours avoir à attendre beaucoup de temps avant d'être complètement traitée. Plus le taux d'arrivée λ est grand plus le temps de réponse moyen d'une tâche est proche du temps maximum K et, dans ce cas la politique de transfert n'apporte aucune amélioration par rapport à celle sans transfert.
Démonstration

Voici la démonstration du corollaire 1. Les résultats obtenus pour la probabilité de saturation mémoire découlent immédiatement du théorème 1 puisque $P_{sat} = P_K(\lambda)$. Pour le temps de réponse, les résultats s'obtiennent à partir des réultats obtenus dans le théorème 1 et dans la proposition 3.
On obtient alors la limite souhaitée de façon évidente pour $\lambda < 1$.

Pour $\lambda = 1$ et $K \neq 2$, on obtient

$$\lim_{n \to +\infty} \overline{R} = 1 + \frac{1}{K-2} + \frac{2}{K-2} + \ldots + \frac{K-3}{K-2} + \frac{1}{2} \quad ,$$

D'où,

$$\lim_{n \to +\infty} \overline{R} = \frac{K}{2} \quad .$$

Pour $\lambda = 1$ et $K = 2$, on obtient

$$\lim_{n \to +\infty} \overline{R} = 1 = \frac{K}{2} \quad .$$

Pour $\lambda > 1$, on obtient

$$\lim_{n \to +\infty} \overline{R} = 1 + (K-2)\frac{1}{\lambda} + (K-1)\left(1 - \frac{1}{\lambda}\right) \quad ,$$

D'où,

$$\lim_{n \to +\infty} \overline{R} = K - \frac{1}{\lambda} \quad .$$

■

4.4.2 Comparaison entre les résultats numériques et les résultats asymptotiques

Les résultats numériques de cette section ont été obtenus avec *mathematica*. Le but de ces calculs est de montrer que les résultats théoriques obtenus dans le théorème 4.4.1 donnent une bonne approximation du comportement du système pour les nombres classiques de sites mis en réseau dans les machines parallèles, à savoir $n = 32, 64, 128, 256$ ou plus. En effet, pour ce qui concerne les probabilités stationnaires, les valeurs limites obtenues ne sont effectivement approchées que pour un nombre très important de processeurs. Par contre, les valeurs limites des indices de performance P_{sat} et $\overline{R}$ sont obtenues très rapidement *i.e.* pour un nombre raisonnable de sites.

Probabilité de saturation

Pour toutes les valeurs du taux d'arrivée λ données, la probabilité de saturation mémoire est une fonction décroissante du nombre n de processeurs comme l'indiquent les figures 4.14, 4.15, 4.16 obtenues pour différentes valeurs de λ. Pour la valeur $\lambda = 0.8$, on constate une convergence très rapide vers 0 obtenue dès les valeurs $n = 8$. Cette même rapidité de convergence est obtenue

pour $\lambda = 1.25$, mais vers la valeur 0.2.

Autrement dit, avec la politique de transfert, ajouter un processeur dans un réseau totalement connecté diminue toujours la probabilité de saturation du système et tend à la stabiliser vers sa valeur limite P_{sat}^{lim}. Les résultats numériques montrent que la convergence vers cette valeur limite est très rapide. La table 4.4 donne par exemple les valeurs $P_{sat} - P_{sat}^{lim}$ pour un taux d'arrivée $\lambda = 1$, pour n sites de capacité $K = 6$ avec $n = 16, 32, 64, 128$.

Or la convergence la plus lente correspond à un taux d'arrivée égal au taux de service, ici

λ	1
$n = 16$	0.0134
$n = 32$	0.0070
$n = 64$	0.0036
$n = 128$	0.0018

Table 4.4 : Table de la différence $P_{sat} - P_{sat}^{lim}$.

$\lambda = 1$, et dans cette situation la valeur de P_{sat} peut être estimée par sa valeur limite P_{sat}^{lim} avec l'approximation donnée par cette table pour les systèmes dont le nombre n de processeurs est supérieur à 32. Autrement dit, quand on dispose de 32 processeurs, ajouter un processeur augmente encore la probabilité qu'une tâche soit acceptée dans le système, mais de façon moins significative, puisque le gain obtenu est inférieur à 0.007.

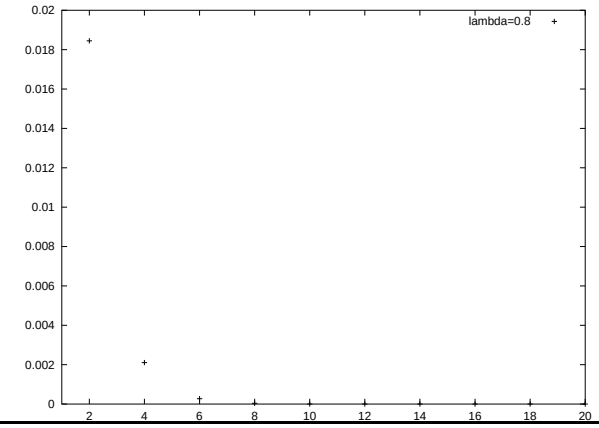

Figure 4.14 : Probabilité de saturation pour un taux d'arrivée $\lambda = 0.8$ fonction du nombre $n = 2q$ de sites de capacité $K = 6$.

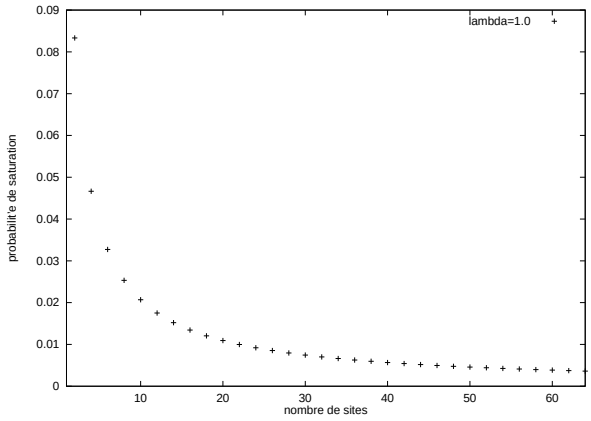

Figure 4.15 : Probabilité de saturation pour un taux d'arrivée $\lambda = 1$ fonction du nombre $n = 2q$ de sites de capacité $K = 6$.

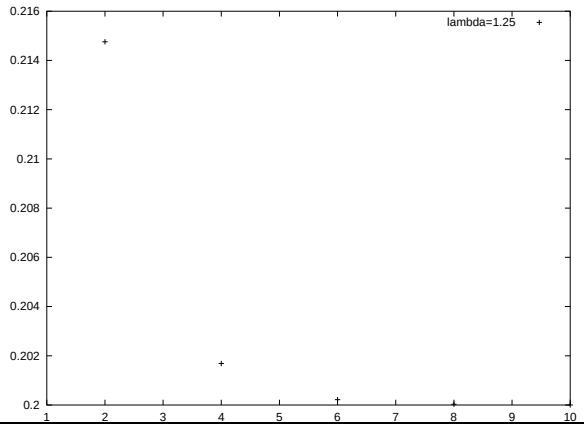

Figure 4.16 : Probabilité de saturation pour un taux d'arrivée $\lambda = 1.25$ fonction du nombre $n = 2q$ de sites de capacité $K = 6$.

Temps de réponse moyen

Le temps de réponse moyen pour un taux d'arrivée λ et une capacité K donnés est une fonction monotone de n, dont le sens dépend des valeurs de λ et de K.

Pour $\lambda < 1$, augmenter le nombre de sites diminue le temps de réponse moyen et ce temps moyen tend rapidement vers 1 qui correspond au temps de service moyen.
La figure 4.17 montre, par exemple, la courbe obtenue pour un taux d'arrivée $\lambda = 0.8$ avec des sites de capacité $K = 6$.

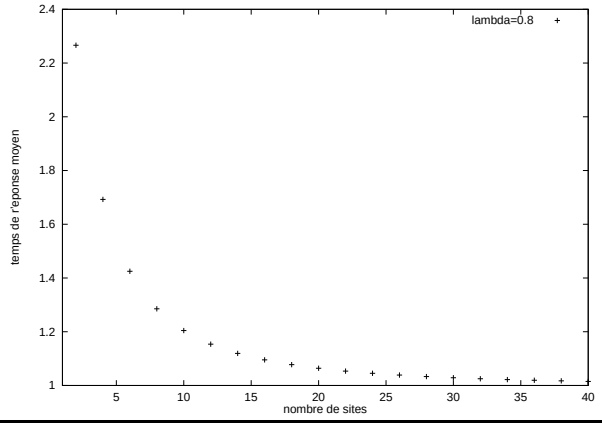

Figure 4.17 : Temps de réponse moyen pour un taux d'arrivée $\lambda = 0.8$ fonction du nombre $n = 2q$ sites de capacité $K = 6$.

Pour $\lambda = 1$, le temps de réponse moyen diminue quand le nombre n de sites augmente, et tend vers $\frac{K}{2}$. La figure 4.18 montre, par exemple, la courbe obtenue pour un taux d'arrivée $\lambda = 1$ avec $n = 2q$ sites de capacité $K = 6$.

Pour $\lambda > 1$, comme il a déjà été mentionné au 4.3.3, l'évolution du temps de réponse moyen en fonction de n dépend de la capacité mémoire K. Le temps de réponse moyen est une fonction monotone de n qui est décroissante si la capacité mémoire est $K = 2$ et est croissante dans tous les autres cas. Les figures 4.19, 4.20 et 4.21 donnent par exemple les résultats obtenus pour un taux d'arrivée $\lambda = 1.25$ avec les capacités $K = 2$, $K = 3$ et $K = 6$.

Dans tous les cas, la stabilité autour de la limite est aussi rapidement atteinte. La table 4.5 donne les valeurs $|\overline{R} - \overline{R}^{lim}|$ pour un nombre n de processeurs de capacité $K = 6$ et un taux

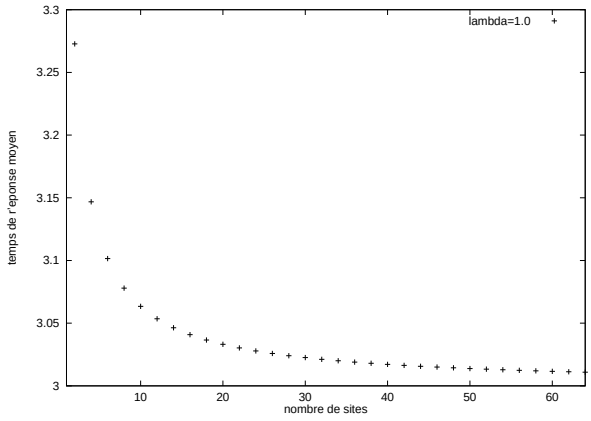

Figure 4.18 : Temps de réponse moyen pour un taux d'arrivée $\lambda = 1$ fonction du nombre $n = 2q$ sites de capacité $K = 6$.

d'arrivée λ donné, avec $n = 16, 32, 64, 128$.

De même que pour la probabilité de saturation mémoire, la valeur $\lambda = 1$ est celle pour laquelle

λ	0.6	0.8	1	1.2	1.4
$n = 16$	0.0065	0.0953	0.0409	0.121	0.0232
$n = 32$	0.004	0.0251	0.0212	0.0369	0.0038
$n = 64$	0	0.0044	0.0109	0.0084	0.0003
$n = 128$	0	0.0004	0.0056	0.0011	0

Table 4.5 : Table de la différence $\mid \overline{R} - \overline{R}^{lim} \mid$.

le système converge le plus lentement, mais même pour cette valeur et pour un système ayant au moins 32 sites, le temps de réponse moyen d'une tâche peut être estimé, avec l'approximation obtenue dans la table 4.5, par sa valeur asymptotique $\overline{R}^{lim}$.

4.5 Calculs d'optimisation

4.5.1 Bornes supérieures sur le bénéfice obtenu avec le transfert

Intuitivement, le transfert tend à améliorer les performances du système, et pour mesurer l'incidence, en termes de gain, de la politique de transfert sur ces indices, nous nous sommes restreints à l'étude de la probabilité de saturation mémoire P_{sat} et du temps de réponse moyen $\overline{R}$, l'étude des autres indices que nous avons mentionnés à la section 2.5.5 pouvant s'en déduire.

Probabilité de saturation mémoire,

Le gain obtenu sur la probabilité de saturation mémoire grâce à la politique de transfert

est $P^*_{sat} - P_{sat}$ où P_{sat} dépend du nombre n de processeurs, de la capacité K et du taux d'arrivée λ. Pour des sites de capacité K donnée, et pour un taux d'arrivée λ donné, le meilleur gain est obtenu pour n infini, et la valeur de ce meilleur gain est

$$\begin{aligned} Gain^{P_{sat}}_{max}(\lambda, K) &= \lambda^K \frac{1-\lambda}{1-\lambda^{K+1}} \quad si\ \lambda \leq 1 \quad , \\ &= \frac{1}{K+1} \quad si\ \lambda = 1 \quad , \\ &= \lambda^K \frac{1-\lambda}{1-\lambda^{K+1}} + \frac{1}{\lambda} - 1 \quad si\ \lambda > 1. \end{aligned}$$

La figure 4.22 montre l'évolution de ce gain en fonction du taux d'arrivée λ, pour des sites de capacité $K = 6$.

Pour une capacité donnée, le meilleur gain est donc obtenu pour $\lambda = 1$ et vaut

$$Gain^{P_{sat}}_{max}(K) = \frac{1}{K+1} \quad .$$

C'est lorsque le taux de service est égal au taux d'arrivée que la politique de transfert permet en moyenne de mieux gérer l'ensemble des mémoires du système. Le gain obtenu sur la probabilité de saturation mémoire est d'autant plus important que la capacité K des sites est petite. Ce gain est donc au mieux égal à $1/3$, lorsque la capacité mémoire est égale à 2.

Temps de réponse moyen,

Comme nous l'avons déjà constaté, le gain obtenu pour le temps de réponse moyen grâce à la politique de transfert dépend de λ de n, et de K, et il convient de distinguer les résultats selon les valeurs de ces paramètres.

Pour $\lambda < 1$ et une capacité K donnés, le gain obtenu est $\overline{R}^* - \overline{R}$, et le meilleur gain, obtenu pour n infini, est

$$Gain^{\overline{R}}_{max}(\lambda, K) = \frac{\lambda}{1-\lambda} - \frac{K\lambda^K}{1-\lambda^K} \quad ,$$

Le meilleur gain est donc obtenu pour $\lambda \to 1$ et vaut

$$Gain^{\overline{R}}_{max}(K) = \frac{K-1}{2} \quad .$$

Autrement dit, plus la capacité K est grande, plus le gain sur le temps de réponse moyen d'une tâche est important. Ce résultat est intuitif car pour λ très proche de 1 mais en régime non saturé ($\lambda < 1$), une tâche générée dans le système sans transfert

de capacité K va attendre beaucoup, tandis qu'elle va en moyenne être traitée la première quand la politique de transfert est mise en œuvre.

Pour $\lambda = 1$, le gain obtenu pour une capacité K est toujours $\overline{R}^* - \overline{R}$, et le meilleur gain obtenu pour n infini est

$$Gain^{\overline{R}}_{max}(1, K) = \frac{1}{2} \quad .$$

Notons que cette limite est indépendante de la capacité mémoire K. Dans la situation sans transfert, la file $M/M/1/K$ est saturée. Le transfert améliore donc le temps de réponse moyen d'une tâche, mais ne permet pas de gagner plus de 50% du temps de service moyen, quelle que soit la capacité K des sites du système.

Pour $\lambda > 1$, le temps de réponse moyen d'une tâche générée par le système avec une politique de transfert peut être supérieur à celui d'une tâche générée par le système sans transfert. Pour une capacité K donnée et un nombre infini de sites, la différence $\overline{R} - \overline{R}^*$ est

$$K - \frac{1}{\lambda} - \frac{\lambda}{1-\lambda} + \frac{K\lambda^K}{1-\lambda^K} - 1 \quad ,$$

et cette différence peut être positive ou négative selon les valeurs de la capacité mémoire K de chaque site.

Pour souligner ces différences de comportement, la figure 4.23 montre les évolutions de la différence $\overline{R}^* - \overline{R}$ en fonction du taux d'arrivée λ, pour un nombre infini de sites de capacité mémoire $K = 2$, $K = 3$ et $K = 6$.

D'après ces graphiques, il apparaît que pour $K = 2$, le transfert diminue toujours le temps de réponse moyen tandis que pour $K > 2$, le temps de réponse moyen avec une politique de transfert devient supérieur au temps de réponse moyen sans transfert dès que le taux d'arrivée devient supérieur au taux de service (ici $\lambda = 1$). Dans cette situation, la différence entre les temps de réponse moyen obtenus sans transfert et avec une politique de transfert accuse donc un brusque changement de comportement en $\lambda = 1$, et ce changement est d'autant plus significatif que la capacité mémoire K est grande.

Le meilleur bénéfice possible sur le temps de réponse moyen d'une tâche grâce à la politique de transfert est donc obtenu pour des valeurs du taux d'arrivée proche du taux de service, en régime non saturé. Dans cette situation, le bénéfice maximum est $\frac{K-1}{2}$, et celui-ci est donc d'autant plus important que la capacité K est grande, et le temps d'attente d'une tâche est alors considérablement réduit grâce à la politique de transfert.

4.5.2 Optimisation du taux d'arrivée

Une application possible de cette modélisation est par exemple, de comparer les valeurs minimales que doit avoir le taux d'arrivée λ pour obtenir un débit global fixé pour chaque processeur.
Ce débit global (throughput dans la littérature anglaise) pour chaque processeur est le nombre moyen de tâches traitées par unité de temps (*cf.* [12]). Sans ou avec une politique de transfert, lorsque le temps de service moyen est pris comme unité de temps, ce débit global, que nous noterons δ est :

$$\delta = 1 - P_0(\lambda) \quad .$$

Afin de garantir une efficacité maximum ou un rendement donné important de chaque processeur, il faut assurer un taux d'arrivée des tâches suffisamment important, et ces valeurs diffèrent selon qu'une politique de transfert est mise en place ou non.
Pour quelques débits particuliers, la table 4.6 donne les valeurs minimales que doit avoir le taux d'arrivée λ pour garantir ce débit, dans les situations sans transfert et avec transfert avec 32 sites.

débit δ souhaité	λ minimale sans transfert	λ minimale avec transfert, n=32
0.95	1.34201	0.95
0.90	1.11712	0.90
0.85	0.98369	0.85

Table 4.6 : Valeurs minimales du taux d'arrivée λ pour un débit δ souhaité

Afin d'assurer un débit moyen constant donné pour chaque processeur, proche de l'efficacité maximum de ce processeur ($\mu = 1$), il est donc nécessaire, d'assurer un taux d'arrivée significativement élevé (par rapport au taux de service) sans transfert, tandis qu'avec transfert, un taux d'arrivée égal au débit moyen souhaité est suffisant. Sans politique de transfert, il y a donc beaucoup de pertes de tâches par rapport au nombre de tâches générées, tandis qu'avec une politique de transfert, la perte de tâches est infime, même pour des valeurs du taux d'arrivée proches du taux de service moyen.

4.5.3 Dimensionnement de la capacité mémoire

Une autre problématique intéressante est de trouver une valeur de la capacité mémoire permettant de garantir une probabilité de saturation mémoire faible pour les valeurs du taux d'arrivée inférieures au taux de service (ici pour $\lambda \leq 1$). Pour un nombre n de processeurs et une valeur du taux d'arrivée λ donnés, la probabilité de saturation mémoire est une fonction

décroissante de la capacité mémoire K. Pour obtenir une probabilité de saturation mémoire inférieure à un seuil donné la capacité mémoire requise est d'autant plus grande que le taux d'arrivée est proche du taux de service.
Dans les deux situations, avec et sans politique de transfert, et dans le cas où le taux de service λ est égal au taux de service de moyen, ici $\lambda = 1$, la table 4.7 donne les valeurs de la capacité mémoire nécessaire pour obtenir une probabilité de rejet inférieure à 0.001.

λ	K sans transfert	K avec transfert, $n = 8$	K avec transfert, $n = 32$
1.0	999	130	33

Table 4.7 : Dimensionnement de la capacité mémoire K pour obtenir $P_{sat} < 0.001$ en régime non saturé ($\lambda < 1$).

Pour obtenir une probabilité de saturation mémoire raisonnablement proche de zéro, la politique de transfert permet donc de réduire considérablement la taille de la capacité mémoire de chaque site.

4.6 Estimation du coût des transferts

4.6.1 Nombre moyen de transferts par unité de temps

Dans la modélisation étudiée dans ce document, le temps des protocoles des communications qui précèdent le transfert et celui du transfert lui-même ont été négligés par rapport au temps de service moyen. Il est toutefois opportun, pour un système doté de cette politique de transfert d'estimer le nombre moyen de transferts effectués par unité de temps. Avec d'autres hypothèses de modélisations, certaines études permettent de tenir explicitement compte de certains délais de communications (*cf.* [65]).

Soit N_t la variable aléatoire égale au nombre de transferts intervenus entre l'instant initial 0 et l'instant t.
Le processus $\{N_t,\, t \geq 0\}$ est un processus de Poisson à environnement Markovien noté dans la littérature MMPP (Markov modulated Poisson process). Quand i processeurs ont une charge égale à j et les $n - i$ autres processeurs ont une charge égale à $j - 1$ ($1 \leq j \leq K$), le taux de passage de n à $n + 1$ de ce processus est (toujours avec un temps d'exécution moyen égal à 1)

$$\lambda i + (n - i) \quad .$$

Le théorème exposé dans la section "the counting function" de [42] permet d'obtenir le nombre

moyen de transferts par unité de temps en fonction des probabilités stationnaires $(p_j)_{j=0...Kn}$ du processus $\{L_t,\, t \geq 0\}$ (avec les notations de 4.2.3). Ce nombre moyen de transferts par unité de temps, $\overline{N}_{tra}$ s'exprime par la relation suivante.

$$\overline{N}_{tra} = \lim_{t \to +\infty} E(\frac{N_t}{t}) = \sum_{j=1}^{K} \left(\sum_{i=1}^{n-1} \lambda i p_{(j-1)n+i} \right) + \sum_{j=1}^{K} \left(\sum_{i=1}^{n-1} (n-i) p_{(j-1)n+i} \right) \quad .$$

Les tables 4.8 et 4.9 donnent les valeurs obtenues pour ce nombre de transferts moyen pour des capacité mémoire $K = 2$ et $K = 6$ avec des valeurs variables du taux d'arrivée λ et du nombre n de processeurs.

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

Table 4.8 : Nombre moyen de transferts par unité de temps pour une capacité mémoire $K = 2$ pour différentes valeurs du taux d'arrivée λ et du nombre n de sites.

λ	0.7	0.8	0.9	1	1.1	1.2	1.3
$n = 8$	6.10	6.34	6.64	6.99	7.34	7.73	8.14
$n = 16$	12.55	13.20	14.03	14.92	15.52	16.15	16.87
$n = 32$	25.25	26.75	28.71	30.88	31.79	32.83	34.11
$n = 64$	50.56	53.72	57.96	62.84	64.23	66.03	68.40
$n = 128$	101.12	107.51	116.33	126.82	128.96	132.24	136.85

Table 4.9 : Nombre moyen de transferts par unité de temps pour une capacité mémoire $K = 6$ pour différentes valeurs du taux d'arrivée λ et du nombre n de sites.

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

4.6.2 Nombre moyen de transferts pour une tâche donnée

Avec cette politique de modélisation de la politique de transfert, une tâche générée et acceptée par le système, peut être transférée plusieurs fois d'un site à un autre. Il est donc utile d'estimer le nombre moyen de transferts que cette tâche va subir avant d'être traitée par un processeur. Ce nombre, que nous noterons $\overline{N}_{tts}$, peut s'obtenir à partir du nombre moyen de transferts $\overline{N}_{tra}$ et du temps moyen de réponse d'une tâche $\overline{R}$, et le débit du système $\overline{T}$. En effet, le nombre moyen de tâches qui entrent dans le système par unité de temps est $\overline{T}$. Le nombre moyen de transferts par tâche par unité de temps est donc $\frac{\overline{N}_{tra}}{\overline{T}}$. De plus, pendant son séjour dans le système, le temps pendant lequel une tâche est susceptible d'être transférée, est le temps d'attente moyen de cette tâche soit $(\overline{R}-1)$. Le nombre moyen de transferts $\overline{N}_{tts}$ que subit une tâche pendant son séjour dans le système est donc

$$\overline{N}_{tts} = \frac{\overline{N}_{tra}}{\overline{T}}(\overline{R}-1)$$

Les tables 4.10 et 4.11 donnent les valeurs obtenues pour ce nombre moyen de transferts que subit une tâche pendant son séjour pour des capacités mémoire $K=2$ et $K=6$ avec des valeurs variables du taux d'arrivée λ et du nombre n de processeurs.

λ	0.7	0.8	0.9	1	1.1	1.2	1.3
$n=8$	0.05	0.08	0.11	0.15	0.20	0.25	0.30
$n=16$	0.02	0.04	0.07	0.11	0.15	0.21	0.27
$n=32$	0.00	0.01	0.04	0.07	0.13	0.19	0.25
$n=64$	0.00	0.00	0.02	0.05	0.11	0.18	0.25
$n=128$	0.00	0.00	0.01	0.04	0.10	0.17	0.25

Table 4.10 : Nombre moyen de transferts d'une tâche donnée pendant son séjour pour n sites de capacité mémoire $K=2$ avec un taux d'arrivée λ.

Avec une capacité mémoire égale à 2, rappelons qu'une tâche arrivant dans le système est transférée si et seulement si elle arrive sur un site de charge 1 tandis qu'il existe un processeur libre. Dans une telle situation, une charge subit donc au plus un transfert, et le nombre de transferts que subit une tâche semble donc acceptable. Notons la décroissance de ce nombre particulièrement quand $\lambda > 1$, mais rappelons que dans cette situation, même pour de petites valeurs de n, la probabilité qu'un processeur soit oisif est voisine de zéro, et la situation de transfert d'une tâche est d'autant moins probable que n est grand.

Pour une capacité mémoire $K=6$, une tâche entrant dans le système peut subir entre 0 et 5 transferts avant d'être traitée. Pour des valeurs du taux d'arrivée inférieures au taux de

λ	0.7	0.8	0.9	1	1.1	1.2	1.3
$n = 8$	0.13	0.28	0.73	1.8617	3.05	3.71	4.14
$n = 16$	0.03	0.10	0.36	1.93	3.60	4.08	4.41
$n = 32$	0.00	0.03	0.14	1.96	3.91	4.24	4.50
$n = 64$	0.00	0.00	0.05	1.98	4.05	4.29	4.52
$n = 128$	0.00	0.00	0.01	1.99	4.11	4.30	4.52

Table 4.11 : Nombre moyen de transferts d'une tâche donnée pendant son séjour pour n sites de capacité mémoire $K = 6$ avec un taux d'arrivée λ.

service, elle va subir en moyenne très peu de transferts. Pour un taux d'arrivée égal au taux de service, ici $\lambda = 1$, le nombre moyen de transferts de cette tâche reste raisonnable de l'ordre de deux. Par contre, pour des valeurs du taux d'arrivée supérieures au taux de service, ici $\lambda > 1$, ce nombre moyen de transferts d'une tâche est important, de l'ordre de 4, voire 5 transferts, c'est-à-dire proche du maximum de transferts possibles. Notons enfin que pour cette capacité $K = 6$, ce nombre augmente quand le nombre de processeurs augmente car la probabilité de transfert d'une tâche augmente. Le coût de la politique de transfert dans cette situation peut alors s'avérer important.

4.7 Comportement asymptotique pour une capacité mémoire non limitée

Il peut être intéressant d'étudier le comportement du système lorsque la capacité mémoire de chaque processeur est idéalement non limitée. Le régime d'équilibre n'a alors de sens que lorsque le taux d'arrivée est strictement inférieur au taux de service, ici $\lambda < 1$.
Sans politique de transfert, on retrouve alors les résultats classiques de n files $M/M/1$ indépendantes.
Avec une politique de transfert, le système se comporte alors comme une file $M/M/n$ pour laquelle le temps inter-arrivées suit une loi exponentielle de paramètre $n\lambda$. La probabilité de saturation mémoire n'a plus de sens dans ce contexte, et le temps de réponse moyen s'exprime de la façon suivante : (avec les notations du 4.2.3)

Proposition 7. *Pour une capacité mémoire $K = \infty$, le temps de réponse moyen en fonction du taux d'arrivée λ et du nombre de processeurs n est :*

$$\overline{R} = \frac{G(n,\lambda) + \frac{n^n}{n!}\left(\frac{\lambda^{n-1}}{1-\lambda} + \frac{\lambda^n}{n(1-\lambda)^2}\right)}{G(n,\lambda) + \frac{n^n}{n!}\frac{\lambda^{n-1}}{1-\lambda}} \quad ,$$

Pour un système massivement parallèle, on obtient :

$$\lim_{n \to +\infty} \overline{R} = 1.$$

Une tâche arrivant dans un tel système va donc presque sûrement être traitée immédiatement. Cela correspond à un comportement limite idéal pour lequel la politique de transfert est éminemment bénéfique.

Ces résultats ont été partiellement décrits dans le rapport de recherche [7].

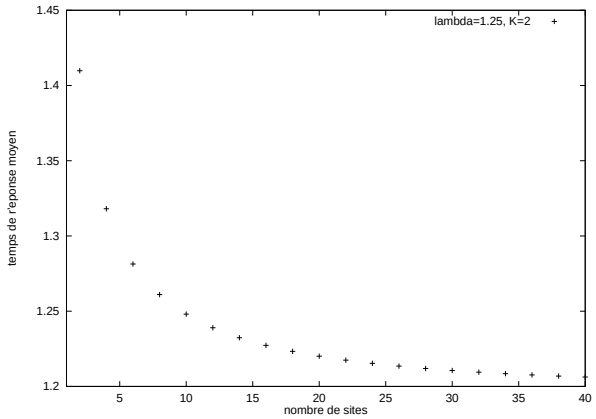

Figure 4.19 : Temps de réponse moyen pour un taux d'arrivée $\lambda = 1.25$ fonction du nombre $n = 2q$ de sites de capacité $K = 2$.

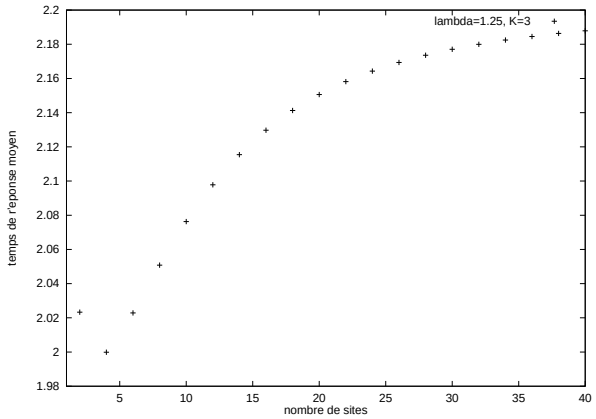

Figure 4.20 : Temps de réponse moyen pour un taux d'arrivée $\lambda = 1.25$ fonction du nombre $n = 2q$ de sites de capacité $K = 3$.

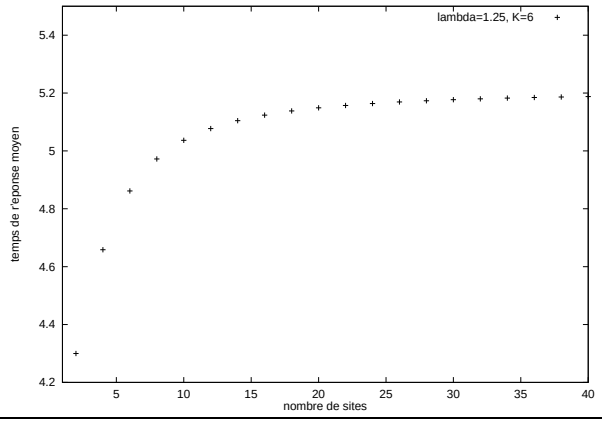

Figure 4.21 : Temps de réponse moyen pour un taux d'arrivée $\lambda = 1.25$ fonction du nombre $n = 2q$ de sites de capacité $K = 6$.

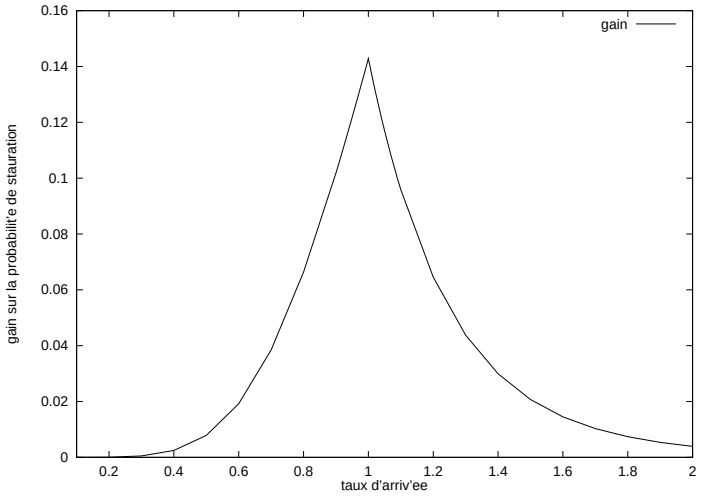

Figure 4.22 : Gain maximum sur P_{sat} en fonction de λ pour une infinité de sites de capacité $K = 6$.

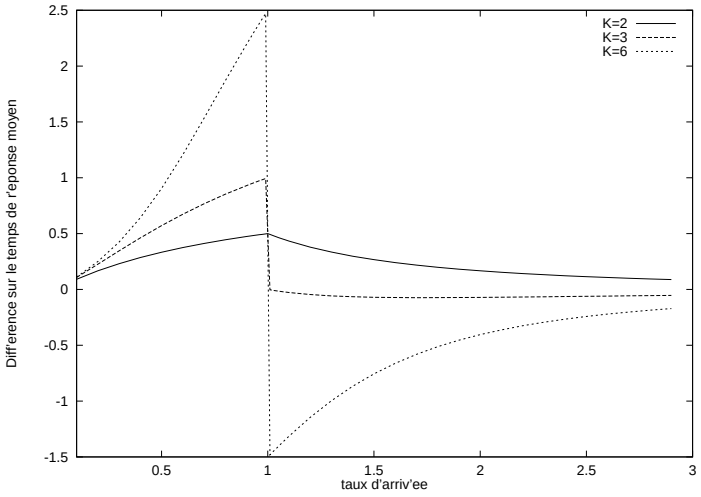

Figure 4.23 : Évolution de la différence $\overline{R}^* - \overline{R}$, pour une infinité de sites de capacité mémoire $K = 2$, $K = 3$ et $K = 6$, en fonction du taux d'arrivée λ.

Chapitre 5

Le transfert de charge dans les réseaux à topologies régulières

5.1 Introduction à la modélisation pour des architectures régulières

Dans ce chapitre, les sites ne sont pas tous interconnectés et la modélisation diffère de celles données dans les chapitres précédents car elle tient compte des contraintes locales de l'architecture d'un réseau. Deux sites seront dits voisins s'ils sont directement connectés. Le réseau formé par l'ensemble des sites et les communications physiques qui les relient peut alors être modélisé par un graphe non orienté dont les sommets représentent les sites et dont les arêtes représentent les connexions physiques entre ces sites. Comme dans toute structure de graphe, la distance entre deux sites est définie comme égale au nombre minimum d'arêtes nécessaires pour relier ces 2 sites.
Comme dans les modélisations des chapitres précédents, nous supposons qu'un transfert immédiat et instantané d'une tâche est effectué dès qu'une différence de charge entre deux sites voisins dépasse strictement 1. Pour que la situation de transfert ait un sens, la condition sur la capacité mémoire de chaque site $K \geq 2$ déjà requise dans les chapitres précédents reste toujours valable.

Cette politique de transfert peut entraîner des transferts en cascade. Par exemple, supposons que le site x ait une charge égale à $i < K$ et que l'un de ses voisins ait une charge strictement inférieure à i. Si une nouvelle tâche est générée sur le site x, ce voisin devient la cible possible d'un transfert du site x vers ce voisin. Ainsi, toute nouvelle tâche générée sur le site x sera immédiatement transférée sur l'un des sites voisins possibles. Parmi les sites voisins possibles, celui sur lequel est transférée cette tâche est choisi au hasard. Mais ce voisin, à son tour, peut lui aussi avoir des voisins ayant une charge strictement plus petite que la sienne. La tâche générée au site x peut ainsi subir des transferts en cascade jusqu'à un site plus éloigné. Il faut

néanmoins souligner que, compte tenu de l'hypothèse d'une capacité limitée K, le nombre de transferts successifs que peut subir une tâche est au plus égal à $K-1$. Ce nombre représente aussi la distance maximum séparant le site où la tâche a été générée et celui où elle sera acceptée (d'un site de charge $K-1$ vers un site de charge 0). Ainsi, selon cette règle, la charge d'un site ne peut augmenter que lorsque celle-ci est inférieure ou égale à celle de ses voisins.

Les figures 5.1 et 5.2 montrent le transfert en cascade après la génération d'une tâche sur un site pour une architecture en cycle.

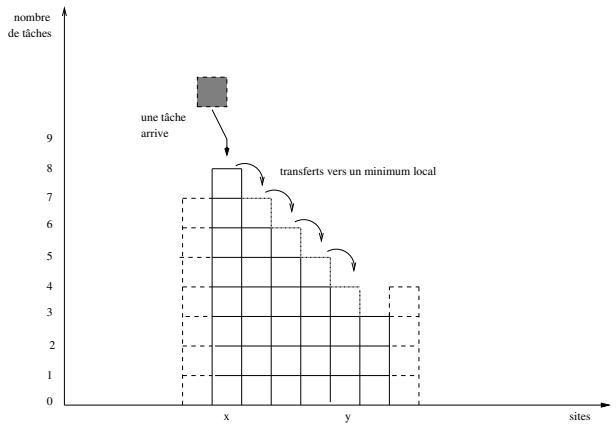

Figure 5.1 : Arrivée d'une tâche et transfert ...

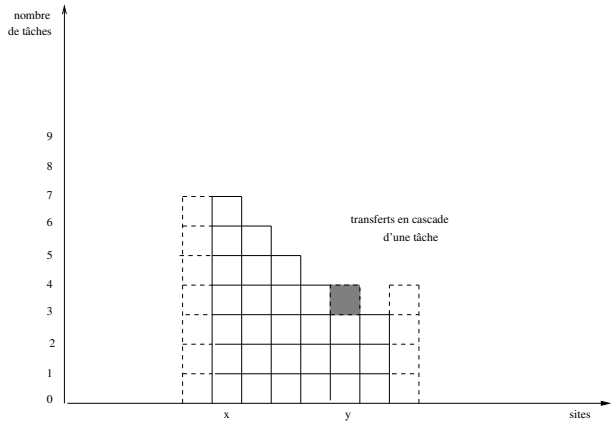

Figure 5.2 : Résultat du transfert après cette arrivée.

Jusque-là, le modèle ainsi décrit est semblable aux modèles de "tas de sable" largement décrits dans la littérature de physique. (voir Dhar [28] comme référence récente ou McNamara et Wiesenfeld [63] pour un modèle plus proche de celui considéré ici). On peut aussi faire une analogie entre ce modèle et les modèles de croissance de cristaux de Gates et Westcott [43], dans le sens où le taux de naissance d'une configuration à un site donné est fonction de la différence de charge entre celle de ce site et celles de ses voisins.

Contrairement aux modèles de "tas de sable" ou de croissance des cristaux, les tâches peuvent aussi quitter le système après la fin de leur exécution. Lorsque le traitement d'une tâche s'achève, la dynamique possible du système est totalement symétrique à celle obtenue lorsqu'une tâche est générée. Supposons qu'une tâche s'achève sur le site x et que certains voisins de x aient une charge supérieure à celle de x. Le départ de cette tâche va immédiatement être compensé par le transfert d'une tâche d'un de ces voisins plus chargé, chacun de ceux-ci ayant la même probabilité de transférer sur x leur dernière tâche. Ce transfert peut à son tour provoquer d'autres transferts en cascade. Comme précédemment, après la complétion d'une tâche, le nombre de transferts en cascade entre le site x et le dernier site où se produit un transfert est au plus égal à $K-1$. Compte tenu de l'obligation de transfert dès que cela est possible, la charge d'un site ne peut effectivement décroître que si la charge de ce site est supérieure ou égale à celle de ses voisins. Pour une architecture en cycle, les figures 5.3 et 5.4 montrent le transfert en cascade après la fin de l'exécution d'une tâche sur un site.

Le modèle ainsi décrit est un système de spins à valeurs dans $\{0,\ldots,K\}$ (*cf.* Liggett [58] pour une référence globale) et sa description formelle, ainsi que ses principales propriétés seront données dans la section 5.2.

Dans le cas des architectures parallèles, plusieurs dizaines voire des milliers de processeurs sont connectés. Pour des raisons technologiques les architectures les plus couramment développées sont les architectures en tores. Il est donc utile d'étudier la politique de transfert pour une architecture modélisée par $\mathbb{Z}^2$ ou $\mathbb{Z}^d$. D'autres modèles ont été étudiés sur des réseaux de taille infinie. Par exemple, Hajek (*cf.* [47]) étudie un modèle similaire, mais avec des modélisations différentes de la charge et de sa répartition. L'étude de notre modèle présente aussi un intérêt mathématique. La section 5.3 donne notre principal résultat : le modèle de transfert est un système ergodique pour toutes les valeurs de la capacité mémoire K et toutes les valeurs du taux d'arrivée λ, et le régime d'équilibre est atteint à vitesse exponentielle. L'ergodicité de systèmes de spins est souvent difficile à démontrer et peut utiliser des techniques ou des outils différents (*cf.* [44, 45]). La preuve donnée ici pour ce résultat repose sur un couplage spécifique et les techniques classiques de comparaisons stochastiques.

5.2 Description formelle du modèle LTM de transfert de charge

Le modèle présenté à la section 5.1 est un système de spins à valeurs dans $\{0,\ldots,K\}$, et pour ce qui concerne la terminologie et les notations utilisées, nous resterons aussi proches que possible de celles utilisées dans le chapitre III de [58].

L'ensemble des sites représentant les sommets du graphe sera noté S, et l'ensemble des arêtes

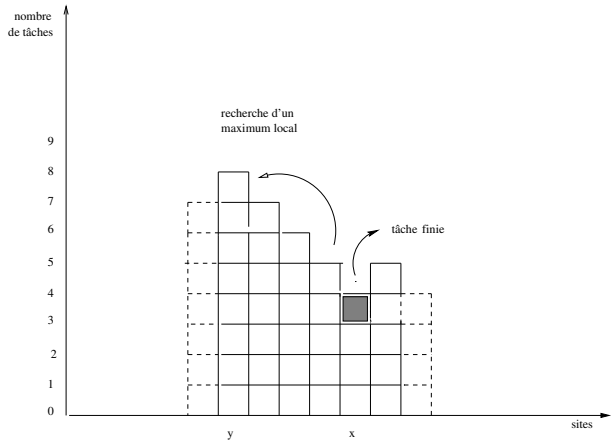

Figure 5.3 : Départ d'une tâche et transfert ...

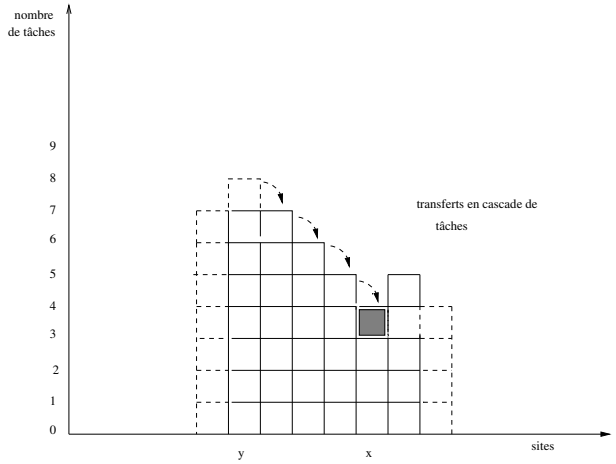

Figure 5.4 : Résultat du transfert après ce départ.

représentant les connections physiques entre les sites sera noté E. Si x est un site, ses voisins successifs sont définis de la manière suivante.

$$\mathcal{N}(x) = \mathcal{N}_1(x) = \{x\} \cup \{y \in S; \{x,y\} \in E\} \ , \quad \text{et}$$

$$\forall k \geq 2 \quad \mathcal{N}_k(x) = \bigcup_{y \in \mathcal{N}_{k-1}(x)} \mathcal{N}(y) \ .$$

Une configuration η est une application de S dans $\{0 \ldots K\}$ qui décrit la charge de chaque site.

$$\forall x \ \in S \ , \quad \eta(x) \in \{0, \ldots, K\} \ .$$

L'ensemble de toutes les configurations possibles est , $X = \{0, \ldots, K\}^S$. Ce modèle est donc défini *a priori* sur l'ensemble X de toutes ces configurations possibles mais ce modèle admet un ensemble absorbant $\mathcal{A}$ inclus dans X. Ce sous ensemble $\mathcal{A}$ est l'ensemble de toutes les configurations telles que la différence de charge entre deux sites voisins est au plus égale à 1.

$$\mathcal{A} = \{\eta \in X \quad \text{t.q.} \quad \forall \{x,y\} \in E \ , \ |\eta(x) - \eta(y)| \leq 1\} \quad .$$

Nous réduirons donc l'espace d'état à ce sous ensemble A. Dans notre modèle, l'état d'une configuration ne peut changer que sur un seul site à la fois (dynamique d'un système de spins). Un tel changement ne peut correspondre qu'au départ ou à l'arrivée d'une tâche à un site x (éventuellement après une cascade de transferts se terminant au site x), et la configuration sera incrémentée ou décrémentée de 1 au site x. Ceci correspond à une transition de la configuration η à l'une des deux configurations notées η_x^+ et η_x^- et définies par

$$\begin{array}{rclcrcl} \eta_x^+(x) & = & \eta(x) + 1 & , & \eta_x^-(x) & = & \eta(x) - 1 \\ \eta_x^+(y) & = & \eta(y) \ \forall \, y \neq x & , & \eta_x^-(y) & = & \eta(y) \ \forall \, y \neq x \ . \end{array}$$

Le taux de transition de η à η_x^+ (arrivée au site x) sera noté $a(x,\eta)$, et celui de η à η_x^- (départ au site x) par $d(x,\eta)$.

Les valeurs de $a(x,\eta)$ et $d(x,\eta)$ doivent prendre en compte les transferts. L'arrivée d'une tâche au site x ne peut se produire que si x est un minimum local de la configuration η, mais cela peut correspondre soit à la génération d'une tâche au site x, soit à une génération sur un autre site dans $\mathcal{N}_{K-1}(x)$, d'une tâche qui subit des transferts en cascade jusqu'au site x.
Pour alléger les notations, l'unité de temps sera égale au temps d'exécution moyen d'une tâche et tous les sites seront supposés homogènes.

Nous noterons $\overline{n}(x,\eta)$ (respectivement $\underline{n}(x,\eta)$) le nombre de sites voisins de x ayant, dans la configuration η, une charge strictement supérieure (respectivement inférieure) à celle de x.

Pour toute configuration η, et pour tout site x, nous définissons *le taux virtuel d'arrivée* $\alpha(x,\eta)$ de la façon récursive suivante.

- Si $\overline{n}(x,\eta) = 0$ alors

$$\begin{aligned} \alpha(x,\eta) &= \lambda \quad \text{si} \quad \eta(x) < K \\ &= 0 \quad \text{si} \quad \eta(x) = K \ . \end{aligned}$$

- Si $\overline{n}(x,\eta) > 0$ alors

$$\alpha(x,\eta) = \lambda + \sum_{\substack{y \in \mathcal{N}(x) \\ \eta(y) > \eta(x)}} \frac{\alpha(y,\eta)}{\underline{n}(y,\eta)} \quad .$$

Traduisons la dynamique du modèle selon laquelle le taux réel d'arrivée au site x est non nul si et seulement si x est un minimum local pour η. Ceci donne les relations suivantes.

$$\begin{aligned} a(x,\eta) &= \alpha(x,\eta) \quad \text{si} \quad \underline{n}(x,\eta) = 0 \\ &= 0 \qquad\quad \text{si} \quad \underline{n}(x,\eta) > 0 \ . \end{aligned}$$

Les précédentes relations peuvent être interprétées de la façon suivante. Si x est un maximum local pour la configuration η, $(\overline{n}(x,\eta) = 0)$ alors aucune tâche générée par le système ne peut être transférée sur x. Par contre, une tâche peut être directement générée sur x si sa charge n'est pas égale à K. Si x n'est pas un maximum local $(\overline{n}(x,\eta) > 0)$, la charge du site x est strictement inférieure à K, et une tâche peut être directement générée sur le site x. De plus, au moins un site y parmi ses voisins a une charge supérieure à la sienne. Toute tâche reçue par le site y (soit directement, soit après des transferts successifs) sera transférée sur le site x avec la probabilité $\frac{1}{\underline{n}(y,\eta)}$. Cette probabilité dépend du nombre de voisins de y susceptibles de recevoir cette tâche. Ce nombre est supérieur à 1 car le site x est au moins un candidat possible.

En considérant la symétrie de la dynamique du modèle en ce qui concerne les départs et les arrivées, on obtient une définition analogue pour *les taux de départ*. Pour *tout* site x dans la configuration η, nous définissons d'abord *le taux virtuel de départ* de la façon récursive suivante.

- Si $\underline{n}(x,\eta) = 0$ alors

$$\begin{aligned} \delta(x,\eta) &= 1 \quad \text{si} \quad \eta(x) > 0 \\ &= 0 \quad \text{si} \quad \eta(x) = 0 \ . \end{aligned}$$

- Si $\underline{n}(x,\eta) > 0$ alors

$$\delta(x,\eta) = 1 + \sum_{\substack{y \in \mathcal{N}(x) \\ \eta(y) < \eta(x)}} \frac{\delta(y,\eta)}{\overline{n}(y,\eta)} \quad .$$

De façon analogue, le taux réel de départ au site x est non nul si et seulement si x est un maximum local pour η. Du point de vue formel, ceci se traduit par les relations suivantes.

$$\begin{aligned} d(x,\eta) &= \delta(x,\eta) \quad \text{si} \quad \overline{n}(x,\eta) = 0 \\ &= 0 \qquad\quad \text{si} \quad \overline{n}(x,\eta) > 0 \ . \end{aligned}$$

De même, l'interprétation de ces relations est la suivante. Si x est un minimum local ($\underline{n}(x,\eta) = 0$), aucune tâche du site x ne peut être transférée sur un de ses voisins. Si la charge du site x est supérieure ou égale à 1 ($\eta(x) > 0$), une tâche peut terminer son exécution sur le site x avec taux 1. Si x n'est pas un minimum local, ($\underline{n}(x,\eta) > 0$) alors une tâche peut quitter le site x, et de plus au moins un site y parmi ses voisins a une charge inférieure à la sienne. Si une tâche quitte ce site y, soit parce que son exécution est terminée soit après des départs en cascade, le site x pourra transférer sa dernière tâche sur le site y avec la probabilité $\frac{1}{\overline{n}(y,\eta)}$. Cette probabilité dépend du nombre de sites voisins de y pouvant bénéficier de ce départ.

Les taux qui viennent d'être définis correspondent à un champ d'interactions de type fini car les taux de départ et d'arrivée au site x ne dépendent des valeurs de la configuration η que dans le voisinage $\mathcal{N}_K(x)$. Dans le cas particulier où $S = \mathbb{Z}^d$, muni d'une structure de réseau, les taux sont invariants par translation d'un site à l'autre et le processus ainsi décrit peut être explicitement construit d'après l'argument de Harris (*cf.* [31] p. 119).

Le système de particules interactives ainsi défini sera désigné par LTM (pour *Load Transfer Model*).

Deux propriétés de base du modèle LTM peuvent immédiatement se déduire de la définition des taux de départ et d'arrivée.

- La première de ces propriétés rend compte des rôles symétriques des départs et des arrivées dans la dynamique du système. Supposons qu'à chaque site x on associe, non sa charge $\eta(x)$ c'est-à-dire le nombre de tâches que ce site accueille, mais son espace mémoire libre, soit $K - \eta(x)$. Ce nouveau processus garde une dynamique semblable à celle du processus initial et cette dynamique est obtenue en inversant les rôles des taux d'arrivée et de départ. Une fois prouvée l'existence de ces quantités, la première partie de la proposition 2 du chapitre 4 reste donc valable ($P_i(\frac{\lambda}{\mu}) = P_{K-i}(\frac{\mu}{\lambda})$).

- La deuxième de ces propriétés rend compte de la monotonie stochastique. Considérons sur $\mathcal{A}$ l'ordre composante par composante suivant

$$\eta \leq \zeta \Longleftrightarrow \forall x \in S \quad \eta(x) \leq \zeta(x) \ .$$

Proposition 8. *Le modèle LTM, restreint à l'ensemble absorbant $\mathcal{A}$, est attractif au sens*

suivant

$$\forall\, \eta \leq \zeta \in \mathcal{A}\,, \quad \eta(x) = \zeta(x) \Longrightarrow \left\{ \begin{array}{rcl} a(x,\eta) & \leq & a(x,\zeta) \\ d(x,\eta) & \geq & d(x,\zeta)\,. \end{array} \right. \tag{5.1}$$

En d'autres termes, la dynamique du modèle LTM tend à homogénéiser la charge d'un site avec celles de ses voisins.

Démonstration

Soit deux configurations ζ et η de A telles que $\eta \leq \zeta$. Soit x un site de S tel que $\eta(x) = \zeta(x)$. Nous allons vérifier ce résultat uniquement dans le cas où une tâche est générée, la situation après la fin d'exécution d'une tâche étant symétrique. Si x n'est pas un minimum local por η, l'inégalité est trivialement satisfaite. Si x est un minimum local pour η, alors c'est aussi un minimum local pour ζ. Appelons *chemin descendant* tout chemin au sens de la structure de graphe de S au long duquel la valeur de η décroît exactement de un à chaque étape. Le long de tout chemin descendant se terminant au site x, les valeurs de ζ et de η coïncident (car ζ $\in$A). Le taux d'arrivée peut s'exprimer sous la forme du produit par λ de la somme sur tous les chemins descendants se terminant au site x de la probabilité qu'une tâche arrive au sommet de ce chemin, le suive par des transferts succeessifs jusqu'au site x. Si γ est un tel chemin, la probabilité qu'une tâche suive un tel chemin est supérieure pour la configuration ζ que pour la configuration η car il y a dans la configuration ζ au moins autant de chemins descendants se terminant au site x que dans la configuration η. On obtient ainsi l'inégalité souhaitée. ■

On peut alors étendre le théorème 2.2 p.134 de [58] et vérifier que être attractif au sens (5.1) ci-dessus est équivalent à la monotonie stochastique pour les systèmes de spins à valeurs dans $\mathcal{A}$ ([58] définition 2.3 p.72). On peut aussi obtenir ce résultat comme un cas particulier du théorème (3.3) in Forbes *et al.* [41]. La monotonie stochastique sera l'un des arguments clés de la preuve de l'ergodicité dans le théorème 2.

5.3 Résultat d'ergodicité du modèle LTM pour les architectures modélisées par $\mathbb{Z}^d$

Tout au long de cette section, l'ensemble des sites est modélisé par $S = \mathbb{Z}^d$. La seule restriction posée sur la structure de graphe est d'être invariante par translation, et localement finie. Par exemple,

$$\forall x \in S \quad \mathcal{N}(x) = \{y \text{ t.q. } ||x - y|| \leq r\}\,,$$

où r est un entier donné, et $||\,.\,||$ est une norme parmi toutes les normes équivalentes de $\mathbb{Z}^d$.

Théorème 2. *Le modèle LTM sur $\mathbb{Z}^d$ est ergodique et converge à vitesse exponentielle vers son régime stationnaire.*

Démonstration

Pour un système de spins attractif à valeurs dans $\{0,1\}$, la technique classique de la preuve de l'ergodicité est décrite dans [58] chap. III Théorème 2.3 et corollaire 2.4 p.136. Ce résultat peut s'étendre à des systèmes de spins attractifs à valeurs dans $\{0,\ldots,K\}$.
L'idée est la suivante. Considérons deux copies du processus, l'une initialisée au temps zéro à la configuration minimale, $(\forall\, x \in\ S\,,\ \eta(x) = 0)$, et l'autre initialisée à la configuration maximale, $(\forall\, x \in\ S\,,\ \eta(x) = K)$, couplées de telle sorte que l'ordre initial reste préservé à chaque instant. Si la différence entre les deux copies tend vers zéro en loi, alors le processus est ergodique. Le couplage traditionnellement efficace pour ce genre de démonstration est le couplage dit de Vaserhstein ([58] p.124). Nous allons utiliser un couplage plus fort, qui préserve non seulement l'ordre entre les deux configurations, mais qui est tel que la différence totale de charge entre les deux copies ne peut que décroître avec le temps. De plus, notre couplage a l'avantage de suivre la description intuitive du modèle donnée dans la section 5.1 (transferts en cascade d'un site x vers un site y).

A chaque site, associons deux processus de Poisson, $\{A_t(x)\,;\,t \geq 0\}$ et $\{D_t(x)\,;\,t \geq 0\}$ et d'intensités respectives λ et 1, tous ces processus étant indépendants. Le processus $\{A_t(x)\}$ recense les arrivées de tâches au site x (acceptées ou non) et le processus $\{D_t(x)\}$ recense les fins d'exécution de tâches au site x (effectives ou non). Nous voulons construire un processus de Feller $\{(\eta_t, \zeta_t)\,;\ t \geq 0\}$ sur $X \times X$ initialisé au couple de configurations suivantes

$$\forall x \in S \quad (\eta_0(x), \zeta_0(x)) = (0, K)\ .$$

Les deux configurations initiales sont dans l'ensemble $\mathcal{A}$ et cet ensemble est absorbant pour la dynamique du modèle LTM, donc à chaque instant, les deux coordonnées vont rester dans l'ensemble $\mathcal{A}$. Ainsi, dans la construction qui va suivre, nous pouvons supposer que toutes les configurations sont telles que la différence de charge entre sites voisins est au plus égale à 1. La contrainte principale est de préserver à chaque instant l'ordre initial *i.e.*

$$\forall x \in S \quad \forall t \geq 0 \quad \eta_t(x) \leq \zeta_t(x)\ .$$

Nous allons seulement décrire l'évolution de chacune des coordonnées après une génération de tâches au site x, *i.e.* un instant correspondant à un saut pour $A_t(x)$. L'évolution obtenue après un départ est exactement symétrique, et elle est obtenue en remplaçant $\eta(x)$ par $K-\eta(x)$ et $A_t(x)$ par $D_t(x)$.

Supposons pour alléger les notations, qu'à un instant t, $\eta_t = \eta$ et $\zeta_t = \zeta$ avec $\eta \leq \zeta$, et qu'une tâche est générée au site x. Trois cas sont alors possibles.

- Si $\eta(x) = K$ et $\zeta(x) = K$: la tâche est rejetée et les deux configurations restent identiques.
- Si $\eta(x) < K$ et $\zeta(x) = K$: la tâche est rejetée par la deuxième coordonnée, et la configuration ζ reste inchangée. La tâche est acceptée dans la première configuration et provoque à un certain site une augmentation de la valeur de la configuration η. Ce site est x lui-même si c'est un minimum local. Sinon, l'augmentation de la valeur de η se produit sur un autre site y dans $\mathcal{N}_K(x)$, choisi en respectant les règles de la dynamique du transfert du modèle LTM données dans l'introduction. Pour que ce cas de figure puisse se produire, il faut qu'il existe un chemin (au sens du graphe) reliant x à y au long duquel la valeur de η décroît exactement de 1 à chaque étape. Le long de ce même chemin, la configuration ζ peut au plus décroître de 1 à chaque étape. Comme $\eta(x)$ est strictement inférieure à $\zeta(x)$, la même inégalité doit rester vraie pour $\eta(y)$ et $\zeta(y)$. Ainsi, l'augmentation de 1 de la valeur de η au site y préserve l'ordre initial des configurations.
- Si $\eta(x) < K$ et $\zeta(x) < K$: la tâche est acceptée et provoque pour chaque configuration une augmentation sur un certain site dans $\mathcal{N}_K(x)$. Deux cas sont alors possibles.

  1. Supposons $\eta(x) < \zeta(x)$.
     Selon que le site x est ou non un minimum local pour η, la tâche générée va rester au site x ou subir des transferts successifs indépendants les uns des autres jusqu'à une destination finale y. D'après l'argument déjà employé dans le cas précédent, la valeur $\eta(y)$ est nécessairement strictement inférieure à la valeur $\zeta(y)$. Ainsi, l'augmentation de la valeur de η au site y préserve l'ordre initial des configurations. Enfin, l'augmentation de la valeur de ζ en un autre site dans $\mathcal{N}_K(x)$ préserve toujours l'ordre initial.
  2. Supposons $\eta(x) = \zeta(x)$.
     Afin de préserver l'ordre initial, les transferts vont devoir être choisis le plus judicieusement possible. Examinons les différentes situations possibles.

Si x est un minimum local pour η , alors c'est aussi un minimum local pour ζ. Dans ce cas la valeur des deux configurations augmente de 1 au site x, et l'ordre initial des configurations est préservé.

Si x est un minimum local pour ζ et non pour η , alors la configuration ζ augmente de 1 au site x, et la configuration η augmente de 1 sur un autre site $y \neq x$. Toujours par le même argument, la valeur $\eta(y)$ est nécessairement strictement inférieure à celle de $\zeta(y)$, et dans cette situation, l'ordre initial des configurations reste préservé.

La dernière situation possible est celle où le site x n'est un minimum local ni pour η, ni pour ζ. Puisque $\eta(x) = \zeta(x)$ et $\eta \leq \zeta$, la tâche générée a nécessairement au moins les mêmes possibilités de transfert dans la configurations η que dans la configuration ζ. Choisissons au hasard (parmi les sites possibles) le site y où la tâche aboutit après le premier transfert possible, dans la configuration η. Deux cas sont alors à nouveau possible :

(a) Si ce site y correspond aussi à un choix possible pour la configuration ζ, alors la tâche est aussi transférée sur y dans la configuration ζ.

(b) Si ce site y n'est pas un choix possible pour la configuration ζ alors y reste la première étape du transfert de la tâche dans la configuration η et la tâche est transférée quelque part dans $\mathcal{N}_1(x)$ pour la configuration ζ, indépendamment du premier choix. Dire que y est un choix possible pour η et ne l'est pas pour ζ signifie que $\eta(y) < \zeta(y)$, et ainsi l'augmentation possible de la valeur de η de 1 au site y préserve l'ordre initial.

Nous itérons alors cette procédure si les choix des transferts possibles peuvent concorder et nous laissons évoluer les deux processus indépendamment sinon. Ainsi, la tâche générée au site x subit une cascade de transferts et va ou bien aboutir sur le même site pour les deux configurations ou bien aboutir sur un site z dans la configuration η tel que $\eta(z) < \zeta(z)$. Dans tous les cas l'augmentation de 1 de la valeur de la configuration η à la destination finale de la tâche générée au site x préserve l'ordre initial.

Le couplage ainsi construit préserve donc l'ordre initial. Montrons que la différence totale de charge entre les deux copies ne peut que décroître avec le temps.
Soit $R \subset S$ un sous-ensemble fini de sites. Définissons la *différence de charge* sur R à l'instant t, comme la somme des différences de charge de tous les sites de R entre les deux configurations η_t et ζ_t. Notons $C_t(R)$ cette différence.

$$C_t(R) = \sum_{x \in R} (\zeta_t(x) - \eta_t(x)) \ .$$

D'après la construction précédente, il apparaît que après la génération d'une tâche au site x, la différence de charge sur tout sous-ensemble contenant $\mathcal{N}_K(x)$ ne peut que décroître de 1, lorsque la tâche est rejetée dans la configuration ζ_t et acceptée dans la configuration η_t, ou rester la même dans tous les autres cas. Le couplage est construit de façon symétrique dans le cas de la fin d'exécution d'une tâche au site x. Ainsi cette propriété de décroissance de la différence de charge reste vraie à chaque instant.

Soit $\{(\eta_t, \zeta_t)\ , t \geq 0\}$ le couplage précédemment défini. Pour prouver le théorème 2, nous devons trouver une constante positive δ telle que pour toute fonction f de X dans $\mathbb{R}$, ne dépendant que d'un nombre fini de coordonnées, il existe une constante positive γ telle que

$$|\mathbb{E}\left[f(\zeta_t) - f(\eta_t)\right]| \leq \gamma e^{-\delta t} \ .$$

Par un argument standard résultant de l'invariance par translation, il est en fait suffisant de prouver

$$\mathbb{P}[\zeta_t(x) - \eta_t(x) > 0] = \mathbb{P}[C_t(\{x\}) > 0] \leq \gamma e^{-\delta t} \tag{5.1}$$

pour des constantes positives γ et δ.

Soit x un site pour lequel $\zeta_t(x) > \eta_t(x)$. L'étape suivante dans la démonstration du théorème 2 consiste à montrer que, après une unité de temps, la différence de charge au site x va effectivement décroître de 1 avec une probabilité strictement positive :

$$\mathbb{P}[C_{t+1}(\{x\}) - C_t(\{x\}) = -1] \geq \delta > 0 \ .$$

Pour montrer ceci, nous allégeons les notations et nous supposons qu'à un instant t, $\eta_t = \eta$ et $\zeta_t = \zeta$, et nous considérons deux configurations $\eta \leq \zeta$ avec $\eta(x) < \zeta(x)$. Nous considérons alors un événement dépendant de x, η et ζ selon lequel une tâche générée au site x est rejetée dans la configuration ζ et pas dans la configuration η,

tous les sites à l'extérieur de $\mathcal{N}_{2K}(x)$ n'étant pas affectés. Pour obtenir cet événement, l'idée est d'avoir suffisamment de tâches générées dans $\mathcal{N}_K(x)$ et aucun autre événement dans $\mathcal{N}_{2K}(x)$ afin

1. d'élever la valeur de la configuration ζ jusqu'à K au site x.
2. de maintenir la valeur de la configuration η strictement inférieure à K au site x.

Alors une tâche supplémentaire générée au site x sera rejetée dans la configuration ζ, et acceptée dans la configuration η.
Nous appelons *chemin descendant* tout chemin dans le graphe des sites reliant le site x à un minimum local de la configuration ζ, le long duquel la valeur de la configuration ζ décroît strictement. Il n'existe qu'un nombre fini de tels chemins. Ce nombre est majoré par $|\mathcal{N}(x)|^K$, et tous les sites du chemin descendant appartiennent à $\mathcal{N}_K(x)$. Faisons d'abord arriver des tâches à toutes les extrémités de ces chemins descendants (minima locaux pour ζ). Après ces arrivées, la configuration ζ est devenue une autre configuration où la longueur de tous les chemins descendants a diminué d'une unité. Après au plus $K-1$ itérations de cette procédure le site x lui-même est devenu un minimum local pour la configuration ζ, mais la valeur $\zeta(x)$ n'a pas changé. Pendant ce temps, la première coordonnée du couplage η n'a été modifiée que sur des sites inclus dans $\mathcal{N}_{2K}(x)$. Certaines des tâches arrivant de cette manière ont pu être transférées sur le site x lui-même, mais ceci ne peut se produire que s'il existe un chemin descendant du site où l'arrivée se produit jusqu'au site x. Ceci implique que la différence $\zeta(x) - \eta(x)$ est strictement supérieure à un. Ainsi, augmanter la valeur de la configuration ζ de sorte que le site x devienne un minimum local pour ζ préserve l'inégalité : $\zeta(x) - \eta(x) > 0$. Si après ces transformations $\zeta(x) = K$, alors une nouvelle tâche générée au site x est rejetée dans la configuration ζ et acceptée dans la configuration η et l'événement désiré est obtenu.
Sinon, une nouvelle tâche générée au site x augmente la valeur des configurations au site x et ceci ne modifie pas la relation $\zeta(x) > \eta(x)$. En répétant alors toute la procédure au plus $K-1$ fois, on obtient l'événement considéré. Pour toutes les configurations η et ζ la probabilité pour que cet événement ait lieu entre les instants t et $t+1$ est strictement supérieure à une valeur $\delta > 0$.

Pour prouver l'équation 5.1, considérons maintenant le cube $R = [-n, +n]^d \cap \mathbb{Z}^d$, et notons R' sa "fermeture" c'est-à-dire l'ensemble de tous les sites situés à une distance inférieure à $2K$ d'un site de R.

$$R' = \bigcup_{x \in R} \mathcal{N}_{2K}(x) \ .$$

La différence de charge sur R', $C_t(R')$, évolue en fonction des arrivées et des départs. D'après ce qui précède, cette différence ne peut que décroître pendant une unité de temps sur les sites de R tels que $\zeta_t(x) - \eta_t(x) > 0$, mais elle peut aussi augmenter sur les autres sites en raison des effets de bords. Ces augmentations ne peuvent survenir qu'à la suite de départs ou d'arrivées à des sites n'appartenant pas à R, et dont la distance à la frontière de R est au plus égale à K. Le nombre de ces sites est majoré par $(n+K)^d - n^d$ donc par une constante multipliée par n^{d-1}.
Pour chaque site x dans R, la probabilité d'une réelle décroissance due au rejet d'une tâche générée au site x dans la configuration ζ et non dans la configuration η est supérieure à

$$\delta \mathbb{P}\left[\zeta_t(x) - \eta_t(x) > 0\right] \ .$$

Par suite

$$\mathbb{E}\left[C_{t+1}(R) - C_t(R)\right] \leq -(2n+1)^d \delta \mathbb{P}\left[\zeta_t(x) - \eta_t(x) > 0\right] + o(n^d) \ .$$

Mais, en utilisant l'invariance par translation, la partie gauche de l'inégalité ci-dessus est aussi

$$\mathbb{E}\left[C_{t+1}(R) - C_t(R)\right] = (2n+1)^d \mathbb{E}\left[C_{t+1}(\{x\}) - C_t(\{x\}\right] \ .$$

Pour n suffisamment grand, on obtient

$$\mathbb{E}\left[C_{t+1}(\{x\}) - C_t(\{x\})\right] \leq -\delta' \mathbb{P}\left[C_t(\{x\}) > 0\right] \quad .$$

Mais $C_t(\{x\})$ est une variable aléatoire à valeurs dans $\{0, \ldots, K\}$.
Par suite

$$\frac{1}{K}\mathbb{E}\left[C_t(\{x\})\right] \leq \mathbb{P}\left[C_t(\{x\}) > 0\right] \ .$$

Ainsi

$$\mathbb{E}\left[C_{t+1}(\{x\})\right] - \mathbb{E}\left[C_t(\{x\})\right] \leq -\frac{\delta'}{K}\mathbb{E}\left[C_t(\{x\})\right] \quad .$$

Par suite, pour tout entier positif m, on obtient

$$\mathbb{E}\left[C_m(\{x\})\right] \leq \left(1 - \frac{\delta'}{K}\right)^m \mathbb{E}(C_0\{x\}) \ ,$$

et par suite, il existe deux constantes positives γ et δ'' telles que pour tout $t \geq 0$

$$\mathbb{E}\left[C_t(\{x\})\right] \leq \gamma e^{-\delta'' t} \ ,$$

et aussi

$$\mathbb{P}\left[C_t(\{x\}) > 0\right] \leq \gamma e^{-\delta'' t} \quad .$$

Ceci termine la démonstration du théorème 2. ■

Comme il a été démontré dans les chapitres précédents, tous les indices de performance pertinents pour ce modèle peuvent s'exprimer en fonction des probabilités $P_i(\lambda)$ donnant la probabilité pour chaque site d'avoir une charge égale à i en régime stationnaire (*cf.* proposition 3 chapitre 3). Ces relations restent valides pour cette nouvelle modélisation, et il est donc essentiel d'obtenir ces valeurs pour comparer les performances du système sans transfert avec celles obtenues avec transfert. Nous commencerons par exposer la méthode du champ moyen dans la section 5.4 puis nous détaillerons les procédures utilisées pour obtenir ces valeurs par des simulations dans la section 5.5 et nous comparerons les résultats obtenus dans la section 5.6. Très curieusement, dans certains cas, les valeurs obtenues par la méthode du champ moyen sont très proches de celles obtenues par les simulations. Enfin, l'incidence du transfert sur les indices de performance P_{sat} et $\overline{R}$ sera étudié dans la section 5.7.

5.4 Heuristique du champ moyen

L'idée de l'heuristique du champ moyen est de considérer que la mesure stationnaire d'un système de particules interactives peut s'obtenir comme la mesure produit de la distribution obtenue sur chaque site. Cette idée est largement utilisée dans la littérature de physique. Pour des modèles semblables à celui considéré ici, des exemples peuvent être trouvés dans [49] et [33]. De Masi *et al.* [26] donne une justification rigoureuse de cette heuristique en prouvant que si des échanges très rapides sont rajoutés aux interactions du système de particules, alors la distribution obtenue devient proche de la mesure produit (voir [32] pour des définitions précises). Cette méthode a été aussi largement utilisée dans la littérature informatique comme par exemple dans les études [34, 35, 65, 66, 84] et dans [85] pour des modèles semblables au nôtre, mais avec des capacités mémoire infinies sur chaque site. Les équations de champ moyen ont été également utilisées et justifiées pour obtenir les mesures stationnaires de larges réseaux dans les modèles de Bramson, Durrett et Swindle (*cf.* [15, 91]), et d'autres modèles adaptés à l'utilisation du champ moyen ont été proposés dans [29, 88, 30, 2, 98]. L'utilisation de cette méthode semble encore un domaine très actif de la recherche dans les réseaux de serveurs (*cf.* [97]).

Considérons le modèle LTM sur un ensemble fini de sites ou sur $\mathbb{Z}^d$. Dans les deux cas, il existe une unique mesure stationnaire. Cette mesure sur X sera notée π et toutes les espérances

relatives à cette mesure seront notées $\mathbb{E}_\pi$.
Si R est un sous-ensemble de S, l'ensemble des configurations restreintes à R seront notées $X(R)$ et la restriction de la mesure π à R sera notée $\pi_{|R}$:

$$\forall \zeta \in X(R) \ , \quad \pi_{|R}[\zeta] = \pi[\{\eta \in X \ t.q. \ \eta(x) = \zeta(x) \ \forall x \in R\}] \ .$$

Nous supposerons qu'un groupe de transformations opère transitivement sur (S, E), ce qui est vrai pour toutes les architectures classiques à topologies régulières. Alors

$$\forall x \in S \ , \ \forall i \in \{0, \ldots, K\} \ , \quad P_i(\lambda) = \pi_{|\{x\}}[i] \ .$$

Soit x un site de S. En appliquant la définition du générateur du modèle LTM à la fonction indicatrice, $\mathbb{1}_{\eta(x)=i}$, du site x dans l'état i, nous obtenons les équations d'équilibre local suivantes.
Pour $i = 0$,

$$\mathbb{E}_\pi \left[a(x,\eta) \left(\mathbb{1}_{\eta(x)=0} \right) - d(x,\eta) \left(\mathbb{1}_{\eta(x)=1} \right) \right] = 0 \quad ,$$

$\forall \ 1 \leq i \leq (K-1)$,

$$\mathbb{E}_\pi \left[a(x,\eta) \left(\mathbb{1}_{\eta(x)=i-1} - \mathbb{1}_{\eta(x)=i} \right) + d(x,\eta) \left(\mathbb{1}_{\eta(x)=i+1} - \mathbb{1}_{\eta(x)=i} \right) \right] = 0 \ ,$$

Pour $i = K$,

$$\mathbb{E}_\pi \left[a(x,\eta) \left(\mathbb{1}_{\eta(x)=(K-1)} \right) - d(x,\eta) \left(\mathbb{1}_{\eta(x)=K} \right) \right] = 0 \quad .$$

Pour une configuration η, les taux ne dépendent que des valeurs des sites dans $\mathcal{N}_K$ et l'espérance ci-dessus peut donc s'écrire comme une somme finie portant sur les configurations $X(\mathcal{N}_K(x))$. Pour ne pas alourdir les notations, nous noterons cet ensemble de configurations X'. On obtient alors les équations suivantes.
Pour $i = 0$,

$$\sum_{\substack{\zeta \in X' \\ \zeta(x)=0}} a(x,\zeta) \, \pi_{|\mathcal{N}_K(x)}[\zeta] = \sum_{\substack{\zeta \in X' \\ \zeta(x)=1}} d(x,\zeta) \, \pi_{|\mathcal{N}_K(x)}[\zeta] \ , \tag{5.1}$$

$\forall \ 1 \leq i \leq (K-1)$,

$$\begin{aligned} \sum_{\substack{\zeta \in X' \\ \zeta(x)=i}} a(x,\zeta) \, \pi_{|\mathcal{N}_K(x)}[\zeta] \ + \ & \sum_{\substack{\zeta \in X' \\ \zeta(x)=i}} d(x,\zeta) \, \pi_{|\mathcal{N}_K(x)}[\zeta] \\ = \ & \sum_{\substack{\zeta \in X' \\ \zeta(x)=i-1}} a(x,\zeta) \, \pi_{|\mathcal{N}_K(x)}[\zeta] \ + \ \sum_{\substack{\zeta \in X' \\ \zeta(x)=i+1}} d(x,\zeta) \, \pi_{|\mathcal{N}_K(x)}[\zeta] \ , \end{aligned} \tag{5.2}$$

Pour $i = K$,

$$\sum_{\substack{\zeta \in X' \\ \zeta(x)=K-1}} a(x,\zeta) \, \pi_{|\mathcal{N}_K(x)}[\zeta] = \sum_{\substack{\zeta \in X' \\ \zeta(x)=K}} d(x,\zeta) \, \pi_{|\mathcal{N}_K(x)}[\zeta] \ . \tag{5.3}$$

L'heuristique du champ moyen consiste à remplacer π par la mesure produit dans les équations (5.1, 5.2, 5.3).

$$\forall \zeta \in X' , \quad \pi_{|\mathcal{N}_K(x)}[\zeta] \leftrightarrow \prod_{y \in \mathcal{N}_K(x)} P_{\zeta(y)}(\lambda) .$$

Les valeurs $P_i(\lambda)$ pour $i \in \{0 \dots K\}$ apparaissent alors comme solution d'un système de $K+1$ équations non linéaires. Pour de grandes valeurs de K, ce système ne sera pas résoluble aisément. Pour $K = 2$ le système est plus simple, et on peut alors résoudre explicitement ces équations. Dans ce cas, les équations dépendent du cardinal du voisinage de chaque site. Pour les réseaux en tore dont la topologie est régulière, nous pouvons supposer que chaque site a exactement k voisins.

L'équation obtenue pour $i = 0$ est alors

$$P_1(\lambda)(1 - P_2(\lambda))^k \ = \ \lambda P_0(\lambda) + \lambda k P_1(\lambda) \sum_{\ell=1}^{k} \frac{1}{\ell} \binom{k-1}{\ell-1} (1 - P_0(\lambda))^{k-\ell} P_0(\lambda)^{\ell} .$$

Pour interpréter la partie gauche de cette équation, il suffit de vérifier qu'un site ne peut passer d'une charge égale à 1 à une charge égale à 0 que lorsqu'une tâche termine son exécution sur ce site (avec taux $\mu = 1$) et que aucun de ses voisins ne peut bénéficier de ce départ danc n'a pas une charge égale à 2. Pour interpréter la partie droite, il faut observer que la charge d'un site x peut passer de 0 à 1 soit après une génération de tâche sur ce site avec taux λ, soit après l'arrivée d'une tâche sur l'un de ses k voisins, nommons le y, si celui-ci a une charge égale à 1. Ce site y peut lui-même avoir ℓ possibilités de transferts, et il va choisir x avec probabilité $1/\ell$. Comme x fait au moins partie des voisins sur lequel y peut transférer, les $\ell - 1$ unes autres possibilités doivent être choisies parmi les $k - 1$ autres voisins.

En utilisant le lemme,

$$\sum_{k=0}^{n} \frac{1}{k+1} \binom{n}{k} x^k y^{n-k} \ = \ \frac{1}{(n+1)x} \left((x+y)^{n+1} \ - y^{n+1} \right) \quad ,$$

l'équation précédente peut se transformer en la suivante :

$$P_1(\lambda)(1 - P_2(\lambda))^k \ = \ \lambda(1 - P_2(\lambda)) - \lambda P_1(\lambda)(1 - P_0(\lambda))^k .$$

L'équation obtenue pour la valeur $i = 2$ est alors similaire et on obtient:

$$\lambda P_1(\lambda)(1 - P_0(\lambda))^k \ = \ (1 - P_0(\lambda)) - P_1(\lambda)(1 - P_2(\lambda))^k .$$

En combinant ces deux équations, on obtient

$$\lambda(1 - P_2(\lambda)) \ = \ (1 - P_0(\lambda)) \ .$$

On retrouve l'équation du système déjà obtenue pour les autres modélisations (voir chapitre 4 section 2).

Posons alors $x = 1 - P_0(\lambda)$. La valeur x est solution de l'équation suivante :

$$(\lambda + \lambda^{-k})x^{k-1}(x(1 + 1/\lambda) - 1) \ = \ 1 \ .$$

Cette équation a une solution unique $x_k(\lambda)$ dans l'intervalle $]0, 1[$, qui peut être obtenue numériquement. Les calculs ont été fait avec *mathematica* pour différentes valeurs du nombre de voisins k pour des sites de capacité $K = 2$. La totalité des équations a également été écrite dans le cas d'une architecture en cycle avec des sites de capacité $K = 6$ et les valeurs des $P_i(\lambda)$ ont été calculées. Ces valeurs peuvent être consultées dans des tables données dans la section 5.6.

Quand k tend vers l'infini, cette valeur $x_k(\lambda)$ tend vers λ si $\lambda < 1$ et vers 1 si $\lambda > 1$. En effet, les valeurs des $P_i(\lambda)$ obtenues grâce aux équations de champ moyen quand le nombre de voisins devient infini sont les mêmes que celles obtenues pour le cas d'une architecture de graphe complet massivement parallèle.

5.5 Estimations par simulation : méthode de Monte-Carlo

Des simulations ont été faites pour trois types de graphes, le tore à une dimension ou autrement dit le cyle, le tore à deux dimensions et l'hypercube. Pour chacun de ces types, des résultats ont été obtenus pour différentes valeurs des paramètres K (capacité mémoire) et λ (taux de génération des tâches sur chaque site) et n (nombre de sites mis en réseau). Pour la simulation des systèmes de particules, on pourra se référer à Ycart [102].

5.5.1 Temps d'accès à l'équilibre

Dans la pratique, pour une simulation donnée, il est d'abord nécessaire de décider quand le régime d'équilibre est atteint. Nous allons définir *le temps d'accès à l'équilibre* comme étant la limite supérieure de l'intervalle de confiance à 99% de la moyenne d'une variable aléatoire T. La définition de cette variable aléatoire utilise largement la propriété de monotonie stochastique (section 5.2). Considérons deux copies indépendantes du modèle LTM $\{\eta_t \, , \, t \geq 0\}$ et $\{\zeta_t \, , \, t \geq 0\}$, l'une initialisée à la configuration où tous les sites ont une charge égale à 0 (appelée configuration nulle ou minimale), et l'autre initialisée à la configuration où tous les sites ont une charge égale

à K (appelée configuration pleine ou maximale). Autrement dit,

$$\forall x \in S\ , \quad \eta_0(x) = 0 \quad , \quad \zeta_0(x) = K\ .$$

Puisque le modèle LTM est attractif, la charge totale (somme des coordonnées) du processus η_t augmente stochastiquement dans le temps, tandis que celle du processus ζ_t décroît. Nous pouvons alors donner la définition suivante

Définition 1. *La variable aléatoire T est égale au premier instant où les charges totales sont égales.*

$$T \ = \ \inf\{t > 0 \ t.q. \ \sum_x \eta_t(x) = \sum_x \zeta_t(x)\}\ .$$

Dans les simulations, les processus étudiés correspondent à la chaîne incluse de ces processus. A chaque itération, la probabilité qu'une tâche soit générée sur l'un des sites est $\frac{\lambda}{\lambda+\mu}$, et la probabilité qu'une tâche termine son exécution est $\frac{\mu}{\lambda+\mu}$. Le site x sur lequel cet événement arrive est choisi au hasard, et si la configuration le permet, celle-ci est changée en un site y avec les règles de transfert énoncées dans la section 5.1. A chaque itération, la valeur de la configuration change au plus en un point.
L'espérance et l'écart-type de T ont d'abord été estimés sur 1000 simulations indépendantes. Pour différents types de graphes (tores à une ou deux dimensions, hypercubes), les courbes donnant le temps d'accès moyen à l'équilibre en fonction de la valeur du taux d'arrivée λ apparaissent toutes semblables pour les mêmes valeurs de la capacité mémoire K. Elles sont bien sûr inchangées par la transformation $\lambda \leftrightarrow 1/\lambda$. Ces courbes s'incurvent de manière abrupte autour de la valeur $\lambda = 1$.
La figure 5.5 montre le temps d'accès moyen à l'équilibre pour l'hypercube à 32 sommets de capacité mémoire $K = 5$.
La figure 5.6 montre le temps d'accès moyen à l'équilibre pour le tore en deux dimensions de 2000 sites de capacité mémoire $K = 6$.

Nous retrouvons le fait que la vitesse de convergence vers le régime d'équilibre se fait à vitesse exponentielle en fonction du taux d'arrivée λ. Le temps d'accès moyen à l'équilibre est d'autant plus rapide que le nombre de sites est petit et que la capacité K est plus petite, mais ce temps moyen dépend également du nombre de voisins de chaque site.
A titre de comparaison la table 5.1 donne les temps d'accès moyens à l'équilibre pour une même architecture, le tore de 2000 sites à deux dimensions, et des capacités mémoires différentes $K = 2$, et $K = 6$ pour les valeurs du taux d'arrivée $\lambda = 0.8$ et $\lambda = 1$, et la table 5.2 donne ces mêmes temps d'accès moyens, en dimension 1 pour le cycle de 250 sites. Les valeurs de ces tables sont données en milliers d'itérations.

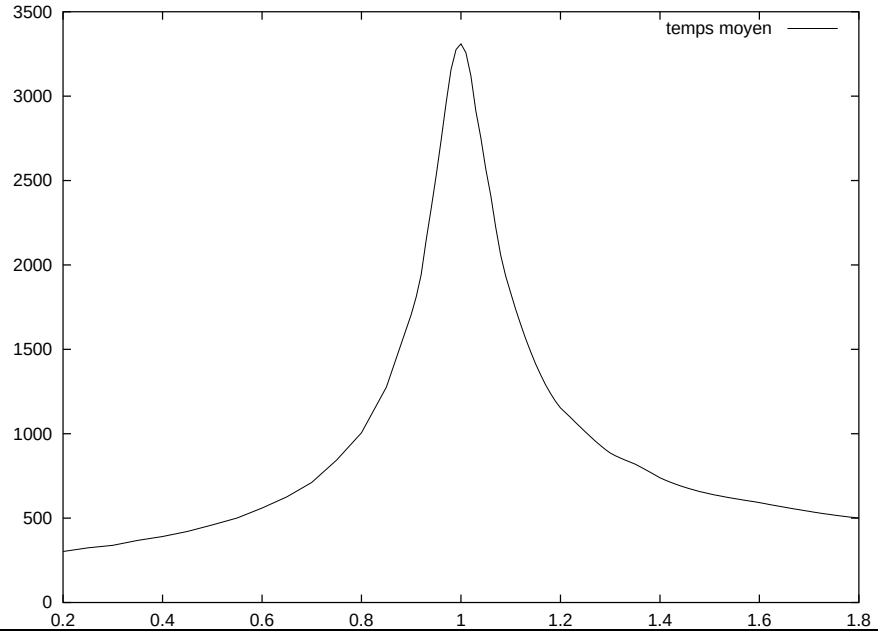

Figure 5.5 : Temps d'accès moyen à l'équilibre en fonction du taux d'arrivée λ pour un hypercube de 32 sites de capacité $K = 5$, une unité de temps = une itération.

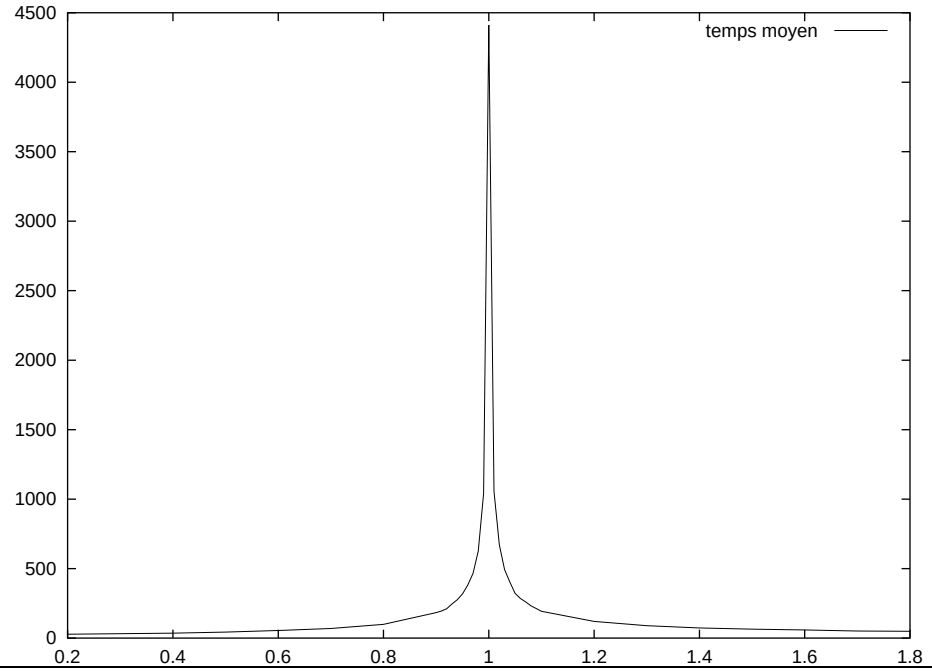

Figure 5.6 : Temps d'accès moyen à l'équilibre en fonction du taux d'arrivée λ pour un tore de 2000 sites (40×50) de capacité $K = 6$, une unité de temps = 1000 itérations.

Tore λ 2000 sites	0.8	1.0
K=2	19.3	19.8
K=6	96.6	4412.0

Table 5.1 : Influence de la capacité en dimension 2 : Temps d'accès moyen à l'équilibre pour le tore à 2000 sites et des capacités mémoires différentes $K = 2$, $K = 6$ avec les taux d'arrivée $\lambda = 0.8$ et $\lambda = 1$ (1 unité de temps = 1000 itérations).

Cycle λ 250 sites	0.8	1.0
K=2	1.9	2.0
K=6	12.7	27.3

Table 5.2 : Influence de la capacité en dimension 1 : Temps d'accès moyen à l'équilibre pour le cycle à 250 sites et des capacités mémoires différentes $K = 2$, $K = 6$ avec les taux d'arrivée $\lambda = 0.8$ et $\lambda = 1$ (1 unité de temps = 1000 itérations).

Autrement dit, augmenter la capacité mémoire augmente fortement le temps d'accès moyen à l'équilibre, et ce d'autant plus que le nombre de voisins de chaque site est grand. Pour une architecture en cycle où chaque site a deux voisins, passer d'une capacité $K = 2$ à une capacité $K = 6$ multiplie au plus le temps moyen d'accès par 16. Pour une architecture en tore, où chaque site a quatre voisins, passer d'une capacité $K = 2$ à une capacité $K = 6$, multiplie le temps d'accès moyen par au plus 225.

Pour comparer l'influence de l'architecture pour une même capacité mémoire, la table 5.3 donne les temps d'accès moyens à l'équilibre pour une même capacité mémoire $K = 2$ et un nombre de sites différents sur des architectures différentes (250 et 3000 pour le cycle, 2000 et 50000 pour le tore) avec des taux d'arrivée $\lambda = 0.8$ et $\lambda = 1$, et la table 5.4 donne les temps d'accès moyens à l'équilibre pour une même capacité mémoire $K = 6$ et un nombre de sites différents sur des architectures différentes (32 et 250 pour le cycle, 32, 2000 et 50000 pour le tore) avec des taux d'arrivée $\lambda = 0.7$, $\lambda = 0.8$ et $\lambda = 0.9$. On pourra remarquer que pour un même nombre de sites (ici $n = 32$), le temps d'accès moyen à l'équilibre augmente lorsque le nombre de voisins de chaque site augmente, et on pourra comparer ces valeurs avec celles obtenues dans la figure 5.5.

De même, pour une même architecture et une même capacité mémoire, augmenter le nombre de sites augmente le temps d'atteinte de l'équilibre. Par exemple, pour une capacité $K = 2$,

$K = 2 \setminus \lambda$	0.8	1.0
32 sites en cycle	0.198	0.227
32 sites en tore	0.209	0.230
250 sites en cycle	1.9	2.0
3000 sites en cycle	30.2	30.0
2000 sites en tore	19.3	19.8
50000 sites en tore	613.3	630.0

Table 5.3 : Influence de l'architecture : temps d'accès moyen à l'équilibre pour une même capacité $K = 2$ et un nombre de sites différents (1 unité de temps = 1000 itérations).

$K = 6 \setminus \lambda$	0.7	0.8	0.9
32 sites en cycle	0.990	1.273	1.975
32 sites en tore	0.986	1.513	2.640
250 sites en cycle	8.6	12.7	20.0
2000 sites en tore	68.6	96.6	173.5
50000 sites en tore	1856	2658	4826

Table 5.4 : Influence de l'architecture: temps d'accès moyen à l'équilibre pour une même capacité $K = 6$ et un nombre de sites différents (1 unité de temps = 1000 itérations).

passer du cycle de 250 sites au cycle de 3000 sites multiplie par 16 au plus ce temps d'accès moyen. Pour une capacité $K = 6$, passer du tore à 2000 sites au tore à 50000 sites multiplie ce temps par au plus 32.

5.5.2 Probabilités stationnaires

Les estimations des probabilités $P_i(\lambda)$ en régime stationnaire ont été estimées de la manière suivante. Le temps d'accès moyen à l'équilibre ayant été défini dans la section 5.5.1, nous avons choisi pour les expériences ultérieures visant à estimer les probabilités stationnaires, de majorer encore ce temps d'accès moyen à l'équilibre et de le prendre égal à la borne supérieure T_{eq} de l'intervalle de confiance de risque 0.01% de la moyenne $\mathbb{E}[T]$. Nous exécutons alors une simulation jusqu'à ce que ce temps d'équilibre soit atteint puis nous calculons la proportion de sites ayant une charge égale à i et nous refaisons cette mesure 30 fois à des intervalles réguliers de temps dont la longueur est égale à T_{eq}. La valeur estimée des $P_i(\lambda)$ et leurs écarts-types sont alors calculés à partir de ces mesures. Pour les grandes valeurs du nombre de sites, pour le cycle et les tores, la valeur des $P_i(\lambda)$ peut aussi être considérée comme la valeur obtenue pour un nombre infini de sites.

5.6 Comparaisons des valeurs obtenues

De nombreux résultats ont été obtenus par ces simulations, et pour comparer les différents modèles et méthodes étudiés dans ce document, nous nous limitons ici à donner quatre tables. Ces tables 5.9 , 5.10, 5.11, et 5.12, donnent les résultats obtenus pour $P_0(\lambda)$ pour les différents modèles et différentes architectures pour différentes valeurs du taux d'arrivée λ et deux valeurs de la capacité mémoire $K = 2$, et $K = 6$. Des tables analogues permettent de comparer les valeurs obtenues pour les autres $P_i(\lambda)$.Ces tables sont données à la fin de cette section. Pour mieux synthétiser ces résultats, et afin de quantifier la différence des valeurs obtenues par la méthode du champ moyen et par les simulations, nous choisissons de calculer la charge moyenne par site, , $\sum_{i=1}^{i=K} iP_i(\lambda)$, et nous définissons alors la différence relative ΔML de la façon suivante

Définition 2. *Nous appelons différence relative de la charge moyenne par site la quantité*

$$\Delta ML = \frac{|ML_{mc} - ML_{mf}|}{ML_{mf}} ,$$

où ML_{mc} (respectivement : ML_{mf}) est la charge moyenne calculée avec les valeurs de $P_i(\lambda)$ obtenues par la méthode de Monte-Carlo (respectivement par les solutions des équations de champ moyen).

La table 5.5 montre les valeurs de ΔML pour des sites de capacité $K = 2$ dans le cas du cycle de 250 sites, du tore à deux dimensions avec 2000 sites et de l'hypercube de dimension 12, pour des valeurs du taux d'arrivée λ comprises entre 0.1 et 1.

$K = 2 \setminus \lambda$	.1	.2	.3	.4	.5	.6	.7	.8	.9	1.
cycle de 250 sites	0.001	0.011	0.001	0.000	0.005	0.000	0.006	0.003	0.003	0.004
tore de 2000 sites	0.000	0.005	0.016	0.002	0.005	0.003	0.005	0.003	0.005	0.000
hypercube de 4096 sites	0.017	0.010	0.008	0.004	0.007	0.004	0.006	0.0025	0.000	0.001

Table 5.5 : Valeurs de la différence relative ΔML pour quantifier la différence obtenue par les méthodes de Monte-Carlo et les solutions des équations de champ moyen pour un grand nombre de sites de capacité $K = 2$.

Pour des sites de capacité $K = 2$ et n'ayant que 32 sites, la table 5.6 donne le résultat de ces calculs pour le cycle et le tore.

Dans le cas d'une capacité mémoire égale à 6, ces calculs n'ont été faits que dans le cas du cycle de 250 et de 32 sites. La table 5.7 donne les résultats obtenus.

Ces différences relatives pour de grandes valeurs du nombre de sites et pour la capacité mémoire $K = 2$ sont extrêmement faibles. Plusieurs explications peuvent être proposées. La différence

$K = 2 \setminus \lambda$	.1	.2	.3	.4	.5	.6	.7	.8	.9	1.
cycle de 32 sites	0.020	0.027	0.076	0.038	0.052	0.007	0.005	0.030	0.015	0.027
tore de 32 sites	0.042	0.083	0.003	0.049	0.047	0.012	0.044	0.023	0.030	0.030

Table 5.6 : Valeurs de la différence relative ΔML pour quantifier la différence obtenue par les méthodes de Monte-Carlo et les solutions des équations de champ moyen pour 32 sites de capacité $K = 2$.

$K = 6 \setminus \lambda$	.1	.2	.3	.4	.5	.6	.7	.8	.9	1.
cycle de 250 sites	0.059	0.021	0.017	0.046	0.073	0.072	0.088	0.021	0.077	0.332
cycle de 32 sites	0.040	0.144	0.051	0.006	0.036	0.133	0.108	0.030	0.140	0.341

Table 5.7 : Valeurs de la différence relative ΔML pour quantifier la différence obtenue par les méthodes de Monte-Carlo et les solutions des équations de champ moyen pour 250 sites de capacité $K = 6$.

entre les équations (5.1, 5.2, 5.3) et les équations de champ moyen peuvent s'exprimer en termes des corrélations suivantes

$$\pi_{|\mathcal{N}_K(x)}[\zeta] - \prod_{y \in \mathcal{N}_K(x)} P_{\zeta(y)}(\lambda) \ .$$

Le théorème 2 prouve que le régime stationnaire est atteint à vitesse exponentielle. Dans le cas d'un système de particules interactives invariant par translation, la distribution asymptotique (pour un grand nombre de sites) a des corrélations qui décroissent exponentiellement ([58], Théorème 4.20 p. 41). Néanmoins, ceci n'explique pas pourquoi ces corrélations devraient être petites dans un voisinage fini. En régime stationnaire, les sites ne sont clairement pas indépendants, et en particulier ils doivent obéir à la condition que la différence de charge entre deux sites voisins est inférieure ou égale à 1. Dans les équations (5.1, 5.2 et 5.3), tous les termes qui correspondent à des configurations qui ne vérifient pas cette condition sont nuls. Pour des valeurs du taux d'arrivée λ non comprises dans un petit intervalle autour de la valeur critique $\lambda = 1$, le système est soit en sous-charge et peu de sites ont une charge supérieure à 2, soit surchargé et peu de sites ont une charge inférieure à $K - 2$.

Pour des sites de capacité $K = 6$, la table 5.8 donne la somme $P_2(\lambda) + \cdots + P_6(\lambda)$, pour le cycle (250 sites), le tore (2000 sites) et l'hypercube (128 sites) pour des valeurs du taux d'arrivée λ comprises entre 0.90 et 0.99. Quand le système est en sous-charge, peu de sites ont une charge supérieure à 2 et très peu de transferts interviennent. Sur de vastes intervalles de temps, le système se comporte donc comme si les sites étaient indépendants. L'approximation de la valeur des $P_i(\lambda)$ par les valeurs obtenues par l'approximation du champ moyen est donc d'autant plus valide que le taux d'arrivée λ est distant de 1, que le nombre de sites n est grand que le

λ	.90	.91	.92	.93	.94	.95	.96	.97	.98	.99
cycle	0.510	0.540	0.577	0.638	0.662	0.684	0.727	0.759	0.807	0.839
tore	0.210	0.225	0.249	0.277	0.313	0.350	0.382	0.450	0.542	0.653
cube	0.086	0.106	0.112	0.133	0.160	0.192	0.234	0.310	0.405	0.600

Table 5.8 : Valeur $P_2(\lambda) + \cdots + P_6(\lambda)$ pour des sites de capacité $K = 6$ pour différentes architectures.

nombre de voisins de chaque site est élevé, et que la capacité mémoire est petite.

Comparaisons possibles pour P_0, $K = 2$

$0.1 \leq \lambda \leq 0.9$

P_0 ⋱ λ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
modèle sans transfert	0.9009	0.806	0.719	0.641	0.571	0.510	0.457	0.410	0.369
avec transfert graphe complet $n = 40$	0.9	0.8	0.7	0.6	0.5	0.4	0.3006	0.2050	0.1218
$n = 2000$	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
avec transfert cycle	0.902	0.796	0.723	0.627	0.533	0.422	0.351	0.269	0.223
σ écart-type (32 sites)	0.050	0.078	0.098	0.0108	0.095	0.0130	0.101	0.096	0.085
avec transfert cycle	0.897	0.800	0.708	0.603	0.509	0.424	0.349	0.285	0.239
σ écart-type (250 sites)	0.021	0.028	0.037	0.037	0.034	0.032	0.037	0.046	0.026
avec transfert cycle	0.9	0.798	0.702	0.604	0.510	0.427	0.354	0.286	0.233
σ écart-type (3000 sites)	0.005	0.008	0.011	0.011	0.012	0.012	0.012	0.010	0.01
champ moyen (k=2)	0.90	0.8002	0.7015	0.6054	0.5142	0.4302	0.3552	0.2904	0.2359
avec transfert tore (k=4)	0.904	0.783	0.700	0.619	0.477	0.403	0.348	0.264	0.158
σ écart-type (32 sites)	0.059	0.073	0.075	0.086	0.139	0.126	0.119	0.129	0.095
avec transfert tore (k=4)	0.900	0.799	0.695	0.600	0.499	0.407	0.318	0.242	0.174
σ écart-type (2000 sites)	0.008	0.010	0.008	0.013	0.017	0.015	0.013	0.014	0.012
champ moyen (k=4)	0.9	0.8	0.7	0.601	0.50255	0.4078	0.3195	0.2414	0.1764
avec transfert hypercube (k=12)	0.90170	0.8021	0.70236	0.60178	0.50346	0.40275	0.30526	0.20557	0.11630
champ moyen (k=12)	0.9	0.8	0.7	0.6	0.5	0.400057	0.300502	0.203223	0.11539

Table 5.9 : Comparaison de la probabilité P_0 pour $K = 2$, $0.1 \leq \lambda \leq 0.9$ pour les différents modèles pour des sites ayant k voisins

$$0.91 \leq \lambda \leq 1.0$$

P_0 ... λ	0.91	0.92	0.93	0.94	0.95	0.96	0.97	0.98	0.99	1.0
modèle sans transfert	0.3652	0.3615	0.3578	0.3541	0.3506	0.3470	0.3435	0.340	0.3367	0.3333
avec transfert graphe complet $n = 40$ $n = 2000$	0.1146 0.09	0.1077 0.08	0.1010 0.07	0.0946 0.06	0.0884 0.05	0.0825 0.04	0.0769 0.03	0.0716 0.02	0.0665 0.01	0.0617 0.0
avec transfert cycle σ (32 sites)	0.253 0.087	0.218 0.109	0.194 0.104	0.195 0.084	0.232 0.089	0.198 0.085	0.185 0.069	0.183 0.096	0.176 0.086	0.199 0.062
avec transfert cycle σ (250 sites)	0.223 0.018	0.224 0.04	0.215 0.034	0.223 0.033	0.216 0.038	0.207 0.034	0.195 0.034	0.190 0.029	0.184 0.035	0.183 0.029
avec transfert cycle σ (3000 sites)	0.226 0.012	0.225 0.011	0.218 0.009	0.215 0.008	0.211 0.011	0.204 0.009	0.198 0.011	0.198 0.008	0.189 0.009	0.186 0.011
champ moyen (k=2)	0.2310	0.2262	0.2215	0.2168	0.2123	0.2079	0.2035	0.1992	0.1951	0.1910
avec transfert tore (k=4) σ (32 sites)	0.162 0.094	0.154 0.091	0.158 0.0119	0.152 0.108	0.164 0.104	0.133 0.079	0.134 0.101	0.152 0.088	0.114 0.075	0.139 0.089
avec transfert tore (k=4) σ (2000 sites)	0.166 0.013	0.168 0.012	0.160 0.015	0.151 0.010	0.146 0.011	0.145 0.011	0.140 0.010	0.131 0.010	0.128 0.009	0.126 0.009
champ moyen (k=4)	0.1708	0.1652	0.1598	0.1545	0.1493	0.1444	0.1395	0.1348	0.1302	0.1258
avec hypercube (k=12)	0.11123	0.10349	0.09571	0.08938	0.08347	0.07730	0.06950	0.06473	0.05952	0.05161
champ moyen (k=12)	0.10767	0.10022	0.09304	.086169	0.07960	0.07336	0.06744	0.06186	0.05661	0.05171

Table 5.10 : Comparaison de la probabilité P_0 pour $K = 2,\ 0.91 \leq \lambda \leq 1.0$ pour les différents modèles pour des sites ayant k voisins

Comparaisons possibles pour P_0, $K = 6$

$0.9 \leq \lambda \leq 0.9$

P_0 ⋱ λ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
modèle sans transfert	0.9	0.80	0.70	0.60	0.5039	0.4115	0.3269	0.2531	0.1917
avec transfert graphe complet $n = 40, 2000$	0.9	0.80	0.70	0.60	0.50	0.40	0.30	0.20	0.10
avec transfert cycle	0.903	0.823	0.700	0.575	0.475	0.423	0.308	0.222	0.094
σ écart-type (32 sites)	0.054	0.081	0.081	0.111	0.121	0.124	0.097	0.1	0.071
avec transfert cycle	0.905	0.791	0.690	0.590	0.494	0.396	0.310	0.186	0.107
σ écart-type (250 sites)	0.016	0.027	0.039	0.029	0.036	0.047	0.039	0.039	0.037
champ moyen approximé voisins immédiats	0.8996	0.7971	0.6918	0.5842	0.4761	0.3669	0.2657	0.1660	0.0730
champ moyen exact (cycle,k=2,K=6)	0.8996	0.7970	0.6915	0.5831	0.4727	0.3622	0.2550	0.1579	0.0772
avec transfert tore	0.896	0.785	0.724	0.578	0.499	0.448	0.286	0.209	0.104
σ écart-type (32 sites)	0.059	0.108	0.097	0.127	0.111	0.117	0.121	0.144	0.109
avec transfert tore	0.90	0.803	0.701	0.596	0.502	0.395	0.302	0.20	0.099
σ écart-type (2000 sites)	0.009	0.010	0.017	0.014	0.017	0.012	0.016	0.010	

Table 5.11 : Comparaison de la probabilité P_0 pour $K = 6,\ 0.1 \leq \lambda \leq 0.9$ pour les différents modèles pour des sites ayant k voisins

$$\boxed{0.91 \leq \lambda \leq 1.0}$$

P_0 ... λ	0.91	0.92	0.93	0.94	0.95	0.96	0.97	0.98	0.99	1.0
modèle sans transfert	0.1862	0.1809	0.1757	0.1707	0.1657	0.1609	0.1562	0.1517	0.1472	0.1429
avec transfert graphe complet $n = 40$	0.09	0.08	0.07	0.06	0.05	0.04	0.0301	0.0206	0.0121	0.0057
$n = 2000$	0.09	0.08	0.07	0.06	0.05	0.04	0.03	0.02	0.01	0.00
avec transfert cycle	0.075	0.078	0.081	0.065	0.044	0.040	0.057	0.036	0.032	0.014
σ écart-type (32 sites)	0.070	0.083	0.067	0.066	0.054	0.037	0.074	0.047	0.059	0.026
avec transfert cycle	0.089	0.083	0.058	0.056	0.051	0.044	0.035	0.024	0.022	0.018
σ écart-type (250 sites)	0.030	0.030	0.022	0.020	0.020	0.020	0.014	0.015	0.014	0.011
champ moyen approximé voisins immédiats	0.0643	0.0559	0.0477	0.0397	0.0320	0.0247	0.0177	0.0112	0.0052	0.0005
champ moyen exact (cycle, $k = 2, K = 6$)	0.0700	0.0630	0.0561	0.0492	0.0422	0.0351	0.0277	0.0193	0.0084	0.0
avec transfert tore	0.108	0.117	0.073	0.064	0.041	0.030	0.049	0.021	0.025	0.014
σ écart-type (32 sites)	0.096	0.098	0.084	0.081	0.063	0.076	0.079	0.070	0.053	0.043
avec transfert tore	0.091	0.084	0.071	0.058	0.050	0.042	0.030	0.020	0.011	0.0
σ écart-type (2000 sites)	0.013	0.011	0.010	0.012	0.010	0.010	0.007	0.006	0.006	0.0

Table 5.12 : Comparaison de la probabilité P_0 pour $K = 6$, $0.91 \leq \lambda \leq 1$ pour les différents modèles pour des sites ayant k voisins

5.7 Comportement des indices de performance

Comme il a déjà été mentionné, la proposition 3 du chapitre 4 reste valable pour les architectures étudiées dans la section 5.6. Il est donc tout à fait possible de calculer la valeur des indices de performance pour cette modélisation.

Probabilité de saturation mémoire Pour des sites de capacité $K = 2$, la figure 5.7 donne les courbes de la probabilité de saturation mémoire comme une fonction du taux de génération des tâches λ. La courbe dont les valeurs sont les plus élevées correspond aux valeurs obtenues pour un système sans politique de tranfert et celle dont toutes les valeurs sont les plus faibles correspond aux valeurs obtenues pour le modèle du graphe complet avec la politique de transfert de charge. Les courbes intermédiaires correspondent aux valeurs obtenues pour les architectures de cycle avec 250 sites, de tores avec 2000 sites et de l'hypercube avec 4096 sites.

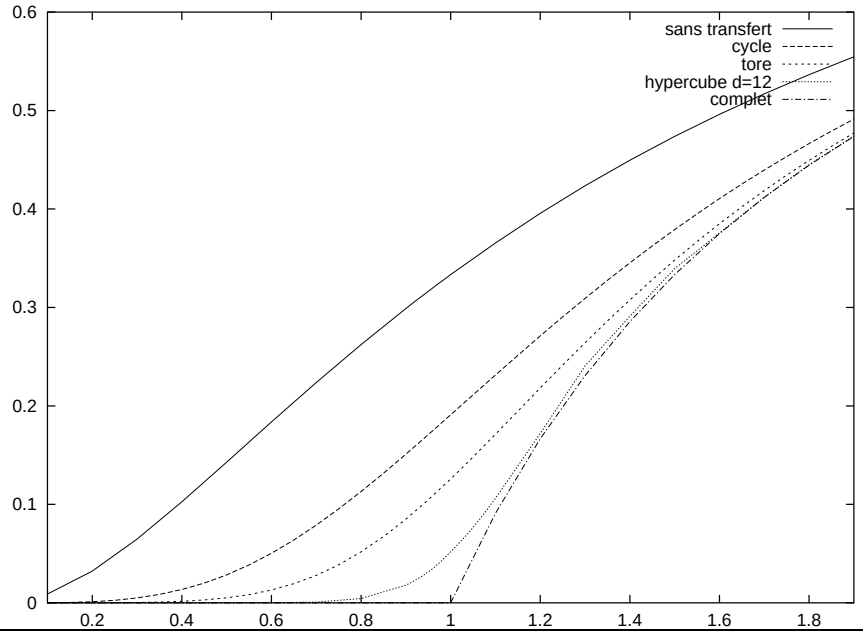

Figure 5.7 : Probabilité de saturation mémoire pour des sites de capacité $K = 2$ pour différentes architectures.

Presque toutes les courbes coïncident pour les faibles et les grandes valeurs de λ pour les raisons précédemment expliquées. Pour les valeurs du taux d'arrivée proche de celle du taux de service, l'architecture des sites influe sur la valeur de la probabilité de saturation,et celle-ci est d'autant plus proche de celle obtenue avec un graphe complet que le nombre de voisins de chaque site est grand. Du point de vue de la probabilité de saturation

mémoire, une architecture en hypercube est donc préférable à une architecture en tore, elle même préférable à une architecture en cycle, et le réseau totalement connecté est le plus souhaitable.

Pour des sites de capacité $K = 6$, la figure 5.8 donne les courbes de la probabilité de saturation mémoire comme une fonction du taux de génération des tâches λ, pour le système sans transfert, pour une architecture en tore de 2000 sites, pour un hypercube de 128 sites, et pour le modèle du graphe complet. Il apparaît relativement peu de différence sur les valeurs de cet indice entre les différentes architectures. Autrement dit, dans cette situation, dès que le nombre de voisins est suffisant (supérieur ou égal à quatre) les variations dues à l'architecture influent peu sur la qualité de la probabilité de saturation mémoire, et celle-ci est alors très proche de celle obtenue pour le graphe complet.

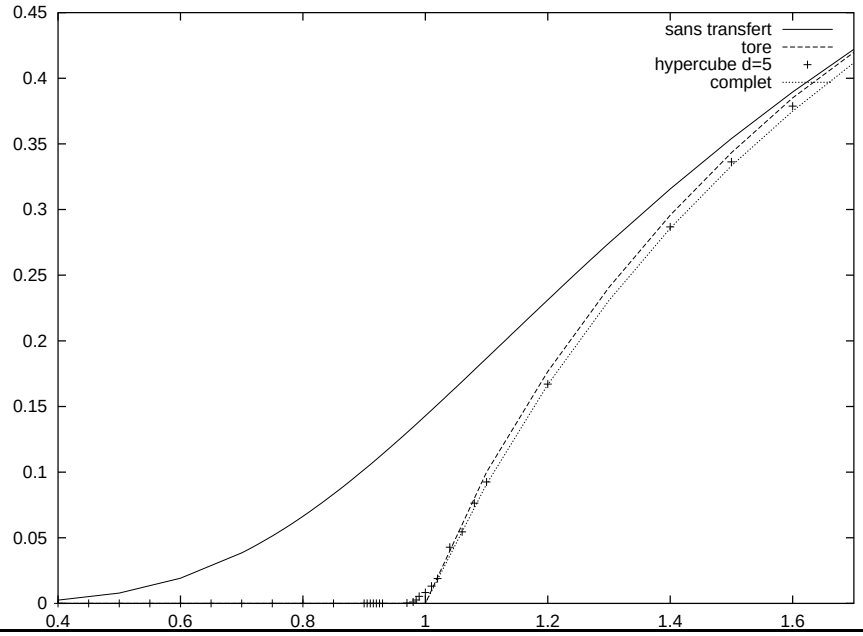

Figure 5.8 : Probabilité de saturation mémoire pour des sites de capacité $K=6$.

Temps de réponse moyen Pour des sites de capacité $K = 2$, la figure 5.9 donne les évolutions du temps de réponse moyen comme une fonction du taux de génération des tâches λ pour ces mêmes architectures. Comme nous l'avons déjà remarqué, pour une valeur de la capacité mémoire égal à deux, le temps de réponse moyen d'une tâche sans politique de transfert est toujours supérieur à celui obtenu avec transfert, et le bénéfice est d'autant plus important que le nombre de voisins de chaque site est grand.

Enfin, la figure 5.10 donne le temps de réponse moyen pour des sites de capacité

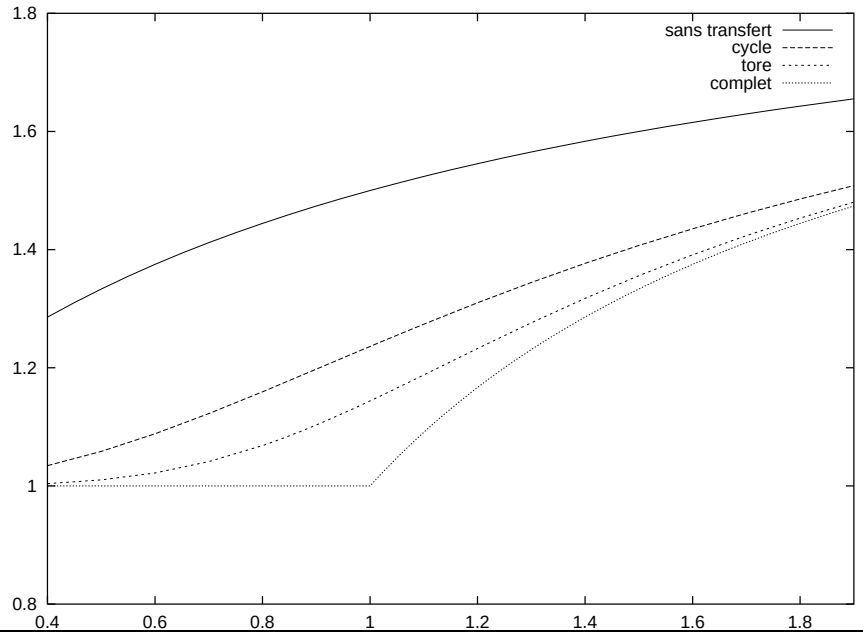

Figure 5.9 : Temps de réponse moyen pour des sites de capacité $K = 2$ pour différentes architectures.

$K = 6$ obtenu avec différentes architectures. De même que pour les courbes obtenues pour la probabilité de saturation mémoire, les courbes extrêmes correspondent aux valeurs obtenues avec le système sans politique de transfert, et avec celles obtenues avec la politique de transfert sur le graphe complet. Les courbes situées entre les courbes extrêmes correspondent à celles obtenues pour des architectures intermédiaires, et nous constatons le même basculement de comportement pour ces architectures que celui observé dans le cas du graphe complet lorsque le taux d'arrivée est proche du taux de service.

Ces résultats ont été partiellement décrits dans le rapport de recherche [8].

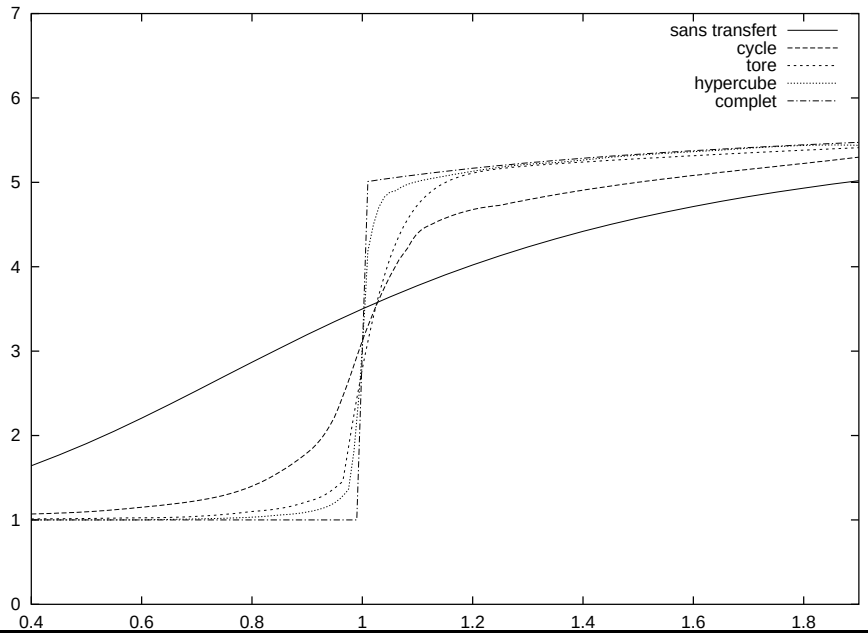

Figure 5.10 : Temps de réponse moyen pour des sites de capacité $K = 6$.

Chapitre 6

Conclusion-Perspectives

Quelques conclusions d'ordre pratique peuvent être déduites de ces études théoriques et expérimentales. Pour toutes les architectures à topologies régulières, les bénéfices que l'on peut attendre de la mise en œuvre d'une politique de transfert peuvent être bornés ou estimés par les méthodes du champ moyen ou par des méthodes de Monte-Carlo, en utilisant le modèle LTM. Ces bénéfices sont maximum quand le taux de service est supérieur au taux d'arrivée, et que la valeur du taux de service de chaque site est proche de celle du taux d'arrivée sur ce site. Lorsque, par le choix de telles architectures, le nombre de voisins par site augmente, les bénéfices possibles augmentent également. Les différences de bénéfices obtenus en fonction du choix des différentes architectures sont d'autant plus faibles que la capacité mémoire K de chaque site augmente. Ceci peut d'ailleurs s'expliquer intuitivement puisque une capacité mémoire plus grande implique un champ d'interaction d'autant plus grand entre les sites et donc plus de possibilités de transfert, ce qui atténue la différence du nombre de voisins liée directement à l'architecture. Les principaux bénéfices que l'on peut espérer d'une politique de transfert sont significatifs sur la probabilité de saturation mémoire et ce, d'autant plus que la capacité mémoire est faible. Pour une valeur du taux d'arrivée de tâches égale ou proche de celle du taux de service, cette probabilité peut passer de 0.33 à 0.001 pour des réseaux de 32 sites de capacité mémoire égale à 33 seulement. Le bénéfice obtenu sur le temps de réponse moyen peut atteindre 50% du temps de service moyen pour les valeurs du taux d'arrivée inférieures ou égales à la valeur du taux de service.

Pour ces valeurs, la politique de transfert diminue considérablement le temps d'attente moyen d'une tâche. Par contre, il se produit un brutal changement de comportement lorsque la valeur du taux d'arrivée excède celle du taux de service. Le débit global du système est alors considérablement augmenté par la politique de transfert au détriment du temps de réponse moyen individuel d'une tâche donnée. La pertinence ou non de la politique de transfert étudiée dépend alors uniquement des objectifs souhaités et des priorités à respecter.

L'étude effectuée ici ne traduit pas toute la complexité du problème de transfert de charge,

mais cette approche et 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. La parallélisation d'applications irrégulières reste un domaine actif de la recherche et la mise au point de systèmes d'exploitation de machines parallèles de haute performance est encore un problème ouvert. Le développement de ces applications irrégulières justifie l'étude de modèles de transfert de charge comme ceux présentés dans ce document. D'autres modèles pourraient être étudiés avec des taux d'arrivée de tâches différents selon les sites, tout au moins pour un nombre raisonnable de sites. Des développements ultérieurs devraient prendre en compte les temps de transfert pour des modèles plus généraux que ceux étudiés dans le chapitre 3. En s'inspirant des techniques utilisées dans le chapitre 5 et des travaux abordant des problématiques semblables dans d'autres domaines, ceci pourrait conduire à de nouveaux modèles de systèmes de particules.

Bibliographie

[1] Y. Artsy et R. Finke. Designing a process migration facility. The Charlotte Experience. *IEEE Trans. on Comp.* 22(9) : 47-56, Septembre 1989.

[2] F. Baccelli, F.I. Karpelevich, M. Ya. Kelbert, A. A Puhalski, A. N. Rybko, Y. M. Suhov. A mean field limit for a class of queueing networks. *Journ. of Statistical Physics*, 6(3/4) : 803–825, 1992.

[3] F. Baccelli et A. Makowski. Queueing models for systems with synchronization constraints. *in Special issue on dynamics discrete events systems*, 77(1) : 138–161, 1989.

[4] A. Barak et A. Shiloh. A distributed load balancing policy for a multi-computer. *Software practice and experience*, 15(9) : 901–913, 1985.

[5] A.T. Barucha-Reid. *Elements of the theory of Markov processes and their Applications.* Mc Graw-Hill, Londres, 1960.

[6] M. Béguin. Transfert de charge dans les machines parallèles. Rapport technique 6, MAI-IMAG, Grenoble, Octobre 1994.

[7] M. Béguin. Transfert de charge dans un réseau de processeurs totalement connectés. Rapport technique 28, MAI-IMAG, Grenoble, Juin 1996.

[8] M. Béguin, L. Gray, et B. Ycart. The load transfer model. A paraître dans Ann. of Appl. Probab, 1996.

[9] M. Béguin, J.M. Vincent, et B. Ycart. Markovian models for load transferring. Rapport technique 14, MAI-IMAG, Grenoble, Mai 1995.

[10] M. Béguin, J.M. Vincent, et B. Ycart. Modélisation et évaluation d'algorithmes de partage de charge, *INRIA, Ecole Française de parallélisme, réseaux et systèmes, support de cours. Placement dynamique et répartition de charge : application aux systèmes parallèlles et répartis*, 215–230, Giens, France, Juillet 1996.

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

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

[13] J. Boillat et F. Bruge et P. Kropf. A dynamic load balancing algorithm for molecular dynamics simulation on multi-processor system. *Journal of Computational Physics*, 96 : 1–14, 1991.

[14] F. Bonomi et A. Kumar. Adaptative optimal load balancing in a heterogeneous multiserver system with a central job scheduler. *in Proc 8th Int. Conf. on Dist. Comp. Syst., San Jose, Californie*, 500-508, 1988.

[15] M. Bramson, R. Durrett et G. Swindle. Statistical mechanics of crabgrass. *Ann. Probab.*, 17(2) : 444–481, 1989.

[16] Brémaud, P. Introduction aux chaînes de Markov et aux files d'attente. Polycopié de l'ENSTA, 1982.

[17] J. Briat et S. Kannat et J. Kitajima et E. Morel. A platform to study dynamic load balancing functions for parallel logic systems. *IWPP'94: IEEE 1st International Workshop on Parallel Processing*, 580-585, Bangalore, Inde, 1994.

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

[19] R.M. Bryant et R. Finkel. A stable distributed sheduling algorithm. *in Proc 2nd Conf. Dist. Comp.*, 314-323, 1981.

[20] L.F. Cabrera. The influence of workload on load balancing strategies. *in Proc. of the USENIX summer 86 Technical Conference*, 446-458, Atlanta, Georgia, USA, 1986.

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

[22] T. Chou et J.A. Abraham. Load balancing in distributed systems. *IEEE Trans. on Soft. Eng.* , SE-28(5), 401-412, 1982.

[23] Y.C. Chow et W. Kohler. Models for dynamic load balancing in a heterogeneous multiprocessors system. *IEEE Trans. on Comp.*, C-28(5) : 334–361, 1979.

[24] E. Çinlar. *Introduction to stochastic processes* Prentice-Hall, New-York, 1975.

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

[26] A. De Masi, P.A. Ferrari, et J.L. Lebowitz. *Reaction-diffusion equations for interacting particle systems.* J. Stat. Phys., 44 : 589-644, 1986.

[27] E. De Souza E Silva et M. Gerla. Queueing networks models for load balancing in distributed systems. *Journal of Parallel and Distributed Computing*, 12(1) : 24–38, 1991.

[28] D. Dhar, P. Ruelle, S. Sen, et Verma D.N. Algebraic aspects of Abelian sandpile models. *J. Phys.*, 28 : 805–831, 1995.

[29] R.L. Dobrushin, Switching networks, Gibbson fields-interconnection. *Proc. of the 1-th World Congress of the Bernoulli Society*, 1986, VNS Sci. Press, Utrecht, v.1 : 377–393, 1987.

[30] R.L. Dobrushin et Y. M. Suhov. Asymptotic investigation of starlike message switching networks with a large number of rays. *Probl. of Inform. Transm.*, 12(1) : 70–90, 1976.

[31] R.T. Durrett. Ten lectures on particle systems. In *Ecole d'été de probabilité de Saint-Flour XXIII (P. Bernard ed.)*, number 1608 in L.N. in Math., 97–201. Springer, New York, 1995.

[32] R.T. Durrett et C. Neuhauser. Particle systems and reaction-diffusion equations. *Ann. Probab.*, 22(1) : 289–333, 1994.

[33] R.T. Durrett et C. Neuhauser. Coexistence results for some competition models. Soumis pour publication, 1996.

[34] D.L. Eager, E.D. Lazowska, et J. Zahorjan. A comparison of receiver-initiated and sender-initiated adaptive load sharing. *Performance evaluation*, 6(1) : 53–68, 1986.

[35] D.L. Eager, E.D. Lazowska, et J. Zahorjan. The limited performance benefits of migrating active processes for load sharing. *In Proceedings of the ACM SYGMETRICS Conference on Measurement and Modeling of Computer Systems*, 63-72, 1988.

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

[37] D. Ferrari et S. Zhou. A load index for dynamic load balancing. *In Proc. 1986 Fall Joint Computer Conference, Dallas, Texas*, pages 684–690, 1986.

[38] D. Ferrari et S. Zhou. An empirical investigation of load indices for load balancing applications. *In Proc. of International symposium on computer performances*, Bruxelles, Belgique, 515-528, 1987.

[39] B. Folliot, P. Sens et P.G. Raverdy. Plate-forme de répartition de charge et de tolérance aux fautes pour applications parallèles en environnement réparti. *Calculateurs parallèles*, 7(4) : 345–366, 1995.

[40] F. Forbes, O. François. Stochastic comparison for Markov processes on a product of partially ordered sets. A paraître dans Statistics and Probability letters.

[41] F. Forbes, O. François, et B. Ycart. Stochastic comparison for resource sharing models. *Markov Proc. Rel. Fields*, 2(4) : 581–606, 1996.

[42] Fisher, W. et Meier-Hellstern, K. The Markov-modulated Poisson process (MMPP) cookbook. *Performance Evaluation*, 18(2) : 149–171, 1993.

[43] D.J. Gates et M. Westcott. Markov models of steady crystal growth. *Ann. Appl. Probab.*, 3(2) : 339–355, 1993.

[44] G. Gielis et C. Maes. Ergodicity in disordered spin systems : stretched exponential relaxation. *Europhys. Lett.*, 31 : 1–5, 1995.

[45] G. Gielis et C. Maes. Percolation techniques in disordered spin flip dynamics : relaxation to the unique invariant measure. *Commun. Math. Phys.*, 177 : 83–101, 1996.

[46] H. Guyennet, B. Herrman, L. Philippe, et F. Spies. A performance study of dynamic load balancing algorithms for multicomputers. Proc. of the 1st International Conference on Massively Parallel Computing Systems, 538-541, IEEE Computer Society Press, 1994.

[47] B. Hajek. Balanced loads in infinite networks. *Ann. Appl. Probab.*, 6(1) : 48–75, 1996.

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

[49] S.A. Janowski et C.A. Laberge. Exact solution for a mean field Abelian sandpile. *J. Phys*,A 26 : 973–980, 1993.

[50] E. Jul et H. Levy et N. Hutchinson et A. Black. Fine grained mobility in the emerald system. *ACM Trans. on Computer Systems*, 6(1) : 109–133, 1988.

[51] J.N. Kappur et H.K. Kesavan. *Entropy optimization principles with applications.* Academic Press, 1992.

[52] M. Ya. Kelbert et Y. M. Suhov. Poisson limit theorem for hybrid star networks: mean field approximation. *Probl. of Inform. Transm.*, 25(1) : 78–87, 1989.

[53] T. Kimura. Optimal Buffer Design of an $M/G/s$ Queue with Finite Capacity. *In Communications in statistics-Stochastic models*, 12(1) : 165–180, 1996.

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

[55] P. Krueger et M. Livny. The diverse objectives of distributed sheduling policies. *in Proc. 7th Int. Conf. on distributed computing systems, Berlin*, 242-249, 1987.

[56] P. Krueger et M. Livny. A comparison of preemptive and non-preemptive load distributing. *in Proc. 8th Int. Conf. on distributed computing systems, San Jose, California*, 123-130, 1988.

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

[58] T.M. Liggett. *Interacting Particle Systems.* Springer-Verlag, New-York, 1985.

[59] J.D.C. Little. A proof of the queueing formula $L = \lambda W$. *Oper. Res.*, (9) : 383–387, 1961.

[60] Z. Liu et D. Towsley. Optimality of the round-robin routing policy. *J. Appl. Probab.*, 31(2) : 466–475, 1994.

[61] S. Madala et J.B. Sinclair. Performance of synchronous parallel algorithms with regular structures. *IEEE Trans. on Parallel and Distributed Systems*, 2(1) : 105–116, 1991.

[62] V. Malyshev et P. Robert. Phase transition in a loss load sharing model. *Ann. Appl. Probab.*, 4(4) : 1161–1176, 1994.

[63] B. McNamara et K. Wiesenfeld. Self-organized criticality in vector avalanche automata. *Phys. Rev.*, 41 : 1867–1873, 1990.

[64] K.I. Mandelberg et E. Sunderam. Process migration in UNIX networks. *In Proc. USENIX winter '88, Dallas, Texas*, 357-364, 1988.

[65] R. Mirchananey, D. Towsley, et J.A. Stankovic. Analysis of the effects of delays on load sharing. *IEEE Trans. on Computers*, C-38 : 1513–1525, 1989.

[66] R. Mirchananey, D. Towsley, et J.A. Stankovic. Adaptative load sharing in heterogeneous systems. *Journal of Parallel and distributed Computing*, 331-346, 1990.

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

[68] R.D. Nelson et M.S. Squillante. Stochastic Analysis of Affinity Scheduling and Load balancing in Parallel Processing Systems. *IBM Rapport de recherche RC 20145 (89158)*, Juin 1995.

[69] M.F. Neuts. *Matrix-geometric solutions in stochastic models: an algorithmic approach.* The Johns Hopkins University Press, 1981.

[70] L.M. Ni et K. Hwang. Optimal load balancing for a multiple processor system with many job classes. *IEEE Trans. on software engineering*, SE-11(5) : 491–496, 1985.

[71] D.A. Nichols. Using idle workstations in a shared computing environments *ACM Operating System Review*, 21(5) : 5–12, 1987.

[72] D.A. Nichols. Multiprocessing in a network of workstations *Phd thesis, Carnegie-Mellon University*, 1990.

[73] D. Nicol et J. Saltz. Dynamic remapping of parallel computations with varying resource demands. *IEEE Trans. on computers*, 37(9) : 1073–1087, 1988.

[74] B. Plateau. Apache : Algorithmique parallèle et partage de charge. Rapport technique, APACHE-IMAG, Novembre 1994.

[75] S. Pulidas et D. Towsley et J.A. Stankovic. Imbedding gradient estimation in load balancing algorithms. *in Proc. 8th Int. Conf. on distributed computing systems, San Jose, California*, 482-490, 1988.

[76] S. Rajsbaum et 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, 1994.

[77] J.L. Roch, A. Vermeerbergen et G. Villard. A new load prediction scheme based on algorithm cost functions *Lecture notes in Computer Science* , 0(854) : 878-890, 1994.

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

[79] M. Simatic. Placement de tâches interactives sur un réseau de stations de travail. *Rapport de DEA, Université de Paris VI*, Septembre 1990.

[80] M.H. Solomon et R.A. Finkel. The Roscoe distributed operating system. *In Proc 7th ACM Symp. operating systems principles*, 108-114, 1979.

[81] K.G. Shin et C.J. Hou. Analytic models of adaptative load sharing schemes in distributed real time systems. *IEEE Trans. on parallel and distributed systems*, 4(7) : 740-761, 1993.

[82] F. Spies. Modeling of optimal load balancing strategy using queueing theory. *Microprocessing and Microprogramming*, 41(8/9) : 555–570, 1996.

[83] F. Spies, H. Guyennet, et M. Trehel. Modeling and simulation of dynamic load balancing using queueing theory. *Parallel Algorithms and Applications*, Gordon and Breach science publishers, 5 : 199–218, 1995.

[84] M.S. Squillante. *Issues in shared-memory multiprocessor scheduling : a performance evaluation.* PhD thesis, Departement of Computer Science and Engineering, University of Washington, 1994.

[85] M.S. Squillante et R.D. Nelson. Analysis of task migration in shared-memory multiprocessors. *In Proceedings of the ACM SYGMETRICS Conference on Meassurement and Modeling of Computer Systems*, 143-155, 1991.

[86] J.A. Stankovic. Simulations of three adaptative, decentralized controlled, job scheduling algorithms. *Comput. networks*, 8(3) : 199–217, 1984.

[87] J.A. Stankovic. An application of Bayesian decision theory to decentralized control of job scheduling. *IEEE Trans. on computers*, 34(2) : 117–130, Février 1985.

[88] A. L. Stoliar. Asymptotics of a stationary distribution of a closed queueing system. *Probl. of Inform. Transm.*, 5(4) : 80–92, 1989.

[89] H.S. Stone. Critical load factors in two processor distributed systems. *IEEE Trans. on software engineering*, 4(3) : 254–258, 1978.

[90] A. Swensson. History, an intelligent load sharing filter. *In Proc. 10th Int. Conf. on distributed computing system, Paris*, 546-553, 1990.

[91] G. Swindle. A mean field limit of the contact process with large range. *Probab. Theory Relat. Fields*, 85(2) : 261–282, 1990.

[92] A.N. Tantawi et D. Towsley. Optimal static load balancing in distributed computer systems. *Journal of the ACM*, 32(2) : 445–465, 1985.

[93] D. Towsley. The allocation of programs containing loops and branches on a multiple processor system. *IEEE Trans. on Software Engineering*, 12(10) : 1018–1024, 1986.

[94] M.M. Theimer et K.A. Lantz. Finding idle machines in a workstation-based distributed system. *IEEE Trans. on software engineering*, 15(2) : 1444–1458, 1989.

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

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

[97] N. D. Vvedenskaya. Large queueing system where the selection of the shortest of several queues is allowed. *Séminaire sur la mécanique statistique des grand réseaux*, 21-25 octobre 96 : 19–21, INRIA Rocquencourt, 1996.

[98] N. D. Vvedenskaya, R. L. Dobrushin, et F. I. Karpelevich. Queueing systems with selection of the shortest of two queues: an asymptotic approach. *Probl. of Inform. Transm.*, 32(1) : 20–34, 1996.

[99] J. Walrand. *Introduction to Queueing Networks.* Prentice-Hall, 1989.

[100] Y. Wang and R. Morris. Load balancing in distributed systems. *IEEE Trans. on computers*, C-34(3) : 204-217, 1985.

[101] C.E. Wills. A service execution mechanism for a distributed environment. *In Proc. 9th Int. Conf. on distributed computing systems, Newport Beach, California*, 326-334, 1989.

[102] B. Ycart. Systèmes markoviens Polycopié de DEA de Mathématiques Appliquées, Université Joseph Fourier, 1993.

[103] J. Zahorjan, K.C. Sevcik, D.L. Eager, et B. Galler. Balanced job analysis of queueing networks. *Communications of the ACM*, 25(2) : 134–141, 1982.

[104] S. Zhou et D. Ferrari. An experimental study of load balancing performance. *In Proc. 7th Int. Conf. on distributed computing systems, Berlin*, 490-497, 1987.

[105] S. Zhou. A trace driven simulation study of dynamic load balancing. *IEEE Trans. on software engineering*, 14(9) : 1327–1341, 1988.

[106] S. Zhou, J. Wang, X. Zheng, et P. Delisle. Utopia : A load sharing facility for large, heterogeneous distributed computer systems. Technical Report CSRI-257, Université de Toronto, Ontario, 1992.

Liste des figures

Liste des tables

Résumé

Cette thèse porte sur la modélisation et l'évaluation d'algorithmes de transfert de charge dans des systèmes parallèles et/ou distribués. Après une synthèse des différentes approches possibles du transfert de charge et des problèmes rencontrés pour leurs mises en œuvre et leurs évaluations quantitatives, nous développons plusieurs modèles basés sur une évolution markovienne de la configuration des charges de l'ensemble des processeurs. Les indices de performance étudiés afin de comparer les valeurs obtenues avec transfert et sans transfert sont la saturation mémoire, le débit du système, la charge de travail et le temps de réponse moyen. Dans les deux premiers modèles seuls deux sites se transfèrent des tâches, mais les temps de communication et de transfert sont modélisés. Des valeurs critiques concernant la pertinence ou non du transfert sont obtenues. Lorsque les temps de communication et de transfert sont négligés devant les temps de calculs, deux modèles sont étudiés. Le premier permet d'évaluer un algorithme d'équilibrage de charge pour un nombre quelconque de sites homogènes totalement connectés, de capacité mémoire finie. Cette étude permet de prévoir le comportement de systèmes massivement parallèles et des bornes supérieures de bénéfices que l'on peut attendre d'un réel transfert sont explicitées. Le deuxième prend en compte l'architecture du réseau et l'algorithme induit un transfert dès que la différence de charge entre deux sites voisins excède un. Dans le cas de réseaux infinis dont la topologie est régulière, ce modèle est ergodique et converge à vitesse exponentielle vers son régime stationnaire. Des résultats de simulations sont presentés pour différentes architectures et comparés aux solutions des équations de champ moyen, qui donnent de très bonnes approximations dans la plupart des cas pour les quantités d'intérêt pratique. Enfin, l'incidence sur la valeur des indices de performance est étudiée et interprétée.

Mots-clés : transfert de charge, systèmes parallels et/ou distribués, évaluations de performances, processus markoviens, files d'attente, systèmes de particules interactives, ergodicité, champ moyen.

Abstract

The aim of this thesis is to model and to evaluate load transferring algorithms in parallel and/or distributed systems. After a synthesis of the different possible approaches of this load transfer and the main problems one gets both to install and to evaluate them with quantitative analysis, we study some models based on a Markovian evolution of the map of the total load of the whole set of processors. The performance indexes under consideration , in order to compare the values with and without transfer, are the memory saturation, the throughput, the workload and the mean response time. In the first two models, only two sites can transfer tasks on each other, but the communication and the transfer delays are taken into account. Critical values about

accuracy or not of the transfer are obtained. When the communication and the transfer delays are negligible in front of the computation delays, two models are studied. The first one allows to evaluate a global load balancing algorithm for any number of sites fully connected, with finite memory capacity. This study yields the asymptotic behavior of massively parallel systems and upperbounds of some benefits that one can expect from a real transfer are derived. The second one takes into account the architecture of the net and the algorithm assumes that transfers occur between neighboring processors as soon as the load difference between them exceeds one. In the case of infinite lattice the model is proved to be ergodic and to converge exponentially fast to its equilibrium. Experimental results for different types of architectures are presented and compared to the solution of the mean field equations, that turn out in most cases to give fairly good approximation to the quantities of practical interest. Incidences of this politic of transfer on the values of the performance indexes are presented.

Key-words: load transferring, parallel and/or distributed systems, performance evaluations, Markovian processes, queueing networks, interactive particle systems, ergodicity, mean field.