# Load balancing in multiprocessor architecture

A. Greenberg, Philippe Jacquet

UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

N° 1370

*Programme 2*
*Calcul symbolique, Programmation et Génie logiciel*

# LOAD BALANCING IN MULTIPROCESSOR ARCHITECTURE

**Albert GREENBERG**
**Philippe JACQUET**

**Janvier 1991**

*RR_1370*

# LOAD BALANCING IN MULTIPROCESSOR ARCHITECTURE

Albert GREENBERG,
AT&T Bell Laboratories
Murray Hill, New Jersey
USA

Philippe JACQUET
INRIA
Le Chesnay
France

**Abstract.** In this note we describe a simple algorithm for load balancing in multiprocessor architecture. In classic multiprocessor architecture each job is resident at a processor and each processor processes its resident jobs according to processor sharing policy. Our algorithm simply consists in (1) placing each of the new jobs on the processor which has the smallest number of resident jobs, and (2) in periodically move heavy jobs from processor with the highest number of resident jobs to processors with the smallest number of resident jobs. Good initial placement of jobs (1) suffices as long as a stream of light jobs is present, and periodic rebalancing (2) fixes things when that stream is absent. Very simple models show how robust is the algorithm. For instance it is shown that optimal re-balancing period praticaly does not depend on activity parameters.

# EQUILIBRAGE DES CHARGES DANS LES ARCHITECTURES A PROCESSEURS MULTIPLES

**Résumé.** Nous décrivons dans cette note un algorithme simple pour l'équilibre des charges dans les architectures à processeurs multiples. Dans de telles architectures, chaque tâche est résidente chez un processeur et chaque processeur traite ses tâches résidentes selon le principe du partage de ressource. Notre algorithme consiste simplement en (1) placer chacune des nouvelles tâches sur le processeur qui a le plus petit nombre de tâches résidentes, et en (2) déplacer périodiquement les tâches lourdes du processeur qui a le plus grand nombre de tâches résidentes vers le processeur qui en a le moins. Le bon placement initial des tâches (1) suffit tant qu'un flot constant de tâches légères est présent, et le re-équilibrage périodique (2) maintient les choses dés que ce flot est absent. Des modèles très simples montrent la robustesse de l'algorithme. En particulier on montre que la période optimale pour le re-équilibrage des charges est largement indépendante des paramètres d'utilisation.

## 1 INTRODUCTION

*A. Resident jobs in multiprocessor architecture*

It is conventional that a computerized unit processes several jobs simultaneously (memory accesses, keyboard device, system management, window manager, *etc*). A job can be in the following alternative states: the active state, when it is directly processed, and the waiting state when it is waiting for an external memory access or any other input signal. When the processor is unique it usually processes its active jobs according to processor sharing policy. In multiprocessor architecture each job is assigned to a single processor a time. the processor to which is assigned a given job at a given time is called the *host* processor. The jobs that are assigned to a given processor are called the *resident* jobs. Usually a processor processes its resident active jobs according to processor sharing policy.

*B. Load balancing in multiprocessor architecture*

When no load balancing process is present jobs are permanent resident on their processor. Problems occur when too large distorsions in the number of resident active jobs permanently co-exist between processors: for example on the user point of view, the power of the multiprocessor architecture is very under-utilized when idle processors co-exists with very busy processors, furthermore job completion time largely suffers when jobs reside on too busy processor. Our solution to this problem consists in two things:

(1) Good initial placement: new active jobs are placed onto processor with the smallest number of active resident jobs.

(2) Periodic re-balancing moves jobs from processors with the highest number of active resident jobs to processors with the smallest.

The first statement (1) is natural and is unexpensive, given that a correct evaluation of the number of resident jobs per processor is available at each time. We may be more careful about the second statement (2). Multiprocessor architectures lead to memory sharing policy: internal memory of the computer is shared by all processors so that remote access to that central memory may be costly enough that makes a cache strategy worthy. In this perspective moving a job already in processing may momentary introduce some local distorsion in the cache utilization of its host processor that is resorbable (cache warming) but leads to momentary under-utilization of the processor (more remote memory accesses per time quantum). Therefore too frequent re-balancing moves may let the power of the architecture collapsing even if the resident populations per processor are perfectly equalized.

Therefore the challenge is the balance between how costly may be too frequent job moves, and how costly may be too badly equalized number of resident jobs per processor. In a first remark, it is unuseful to move light jobs (which do not demand too much time quantum to be completed), since their initial placement suffice to maintain good load balancing as it will be illustrated by the model of section 2. Therefore only heavy jobs (which demands a lot of time quanta to be completed) may be affected by periodic re-balancing moves, that may slightly increase the re-balancing period (and therefore decrease the balancing frequence), since heavy jobs must occur much less often than light jobs in stationary utilization of the architecture. Model of section 3 will illustrate the fact that the rebalancing period in fact can slightly exceed the heavy jobs occurrence period.

*C. The algorithm*

The algorithm we suggest is very simple and directly follows statements (1) and (2).

(i) New active jobs are assigned to one of the processors which, at that time, has the smallest number of resident active jobs.

(ii) Periodically a heavy job is chosen from the processor with the highest number of active resident heavy jobs, at that time, and is moved to one of the processors with the smallest number of active resident heavy jobs, given that the discrepancy between these highest and lowest numbers be greater than or equal to 2.

The problem may be in how to recognize heavy jobs from light jobs. In fact we can replace heavy jobs by old jobs and introduce a threshold $D$: jobs which has already completed more than $D$ time quanta in processor are considered to be heavy, the other ones are considered to be light jobs.

## 2 Model for good initial placement

In this model we suppose that all jobs are light jobs, therefore statement (ii) is not taken in account. We don't do the distinction between new active jobs from re-activated jobs after their waiting state. Thus we consider a closed model similar to the model of memory conflict [1]. There are $N$ jobs circulating from active state to waiting state, there are $M$ identical processors, we suppose $N > M$. The interest of the close model compared to an open one is in the fact that it appropriately modelizes the behavior of the system when close to congestion without suffering some unstable divergence.

The set of active resident jobs at a given processor is the *queue* at this processor. Each queue is served within processor sharing policy. For each job, the number of time quanta needed for the completion of an active period is Poisson of rate $\mu$. After the completion of their active period each job enters waiting period of mean length $1/p$ (let us suppose that waiting periods are i.i.d Poisson random variables with parameter $p$). At the end of the waiting period the job is re-activated and immediately placed on the processor with the shortest queue.

*A. The steady state*

Let $\alpha = N/M$, and suppose that $N$ and $M$ are large. Let us consider the steady state of the system. We suppose that the model leads to the existence of an integer $\ell$ such that *most* of the queues are of length $\ell$ or $\ell+1$. By "most of the queues are of length $\ell$ or $\ell+1$" we mean that when $M$ and $N$ tend both to infinity, the number of queues of length distinct than $\ell$ and $\ell+1$ remains finite and bounded (in probability). We call $\ell$, this *waterline* level, and $r_0$ and $r_1$ respectively the proportion of processor with queue length $\ell$ and $\ell+1$. Thus $r_0 + r_1 = 1 - O(1/N)$ in steady state.

In the following, we suppose that $N/M - \mu/p > 1$ (alternative case leads to $\ell = 0$ which is more simple and does not describe congested architecture).

Since every new placement occurs on the shortest queue, let $m(t)$ denotes the length of the shortest queue(s) at time $t$, quantity $m(t)$ is a *random variable*. In the sequel we rescale the time in such a way that $M$ new time units equal one old time unit.

Conjecture 1

*When $N$ and $M$ tend to be large, and considering steady state, quantities $\ell$, $r_0$ and $r_1$ are constant.*

COROLLARY 2
*We have the expression $\ell = \lfloor \alpha - \mu/p \rfloor$ and $r_1 = \alpha - \mu/p - \ell$; $\lfloor x \rfloor$, when $x$ is real, is the integer part of $x$.*
*Proof:* The number of active jobs is $(\ell+r_1)M+O(1)$, therefore the number of waiting jobs is $N-(\ell+r_1)M+O(1)$. The process of job re-activation is Poisson of rate $p(\alpha-\ell-r_1)M+O(1)$ per old time unit. Assuming by now a new scaled time, and dropping $O(1/M)$ terms, the rescaled re-activation rate is $\lambda = p(\alpha - \ell - r_1)$ per new time unit. If the waterline remains above zero, the departure process from processors is a Poisson process of rate $\mu$ per new time unit. Identity $\lambda = \mu$ is directly provided by conjecture 1 when equalizing arrival and departure rates. Therefore $\ell + r_1 = \alpha - \mu/p$.

■

PROPOSITION 3
*Quantity $m$ oscillates between $\ell$ and $\ell - 1$ and the number of processors with queue length $\ell - 1$ follows a M/M/1 system with creation rate $\mu r_0$ and annihilation rate $\mu$ per new time unit.*
*Proof:* Let us consider that all queues are greater than or equal to $\ell$, namely $m(t) = \ell$. If a new arrival (new re-activated job) occurs it will be placed on a queue of length $\ell$, and this won't change the proportion $r_0$ and $r_1$ and therefore the value of $m(t)$. Similarly, nothing changes when a service completion occurs on a processor of length $\ell + 1$. Interesting things happen when a job departs from a processor of length $\ell$, which happens like a Poisson process of rate $\mu r_0$ (per new time unit). Therefore $m(t)$ takes the value $\ell - 1$ and there is now one queue of length $\ell - 1$. Such a queue can be seen like a "hole" in a sequence of queues which are all of length equal to $\ell$ or $\ell + 1$.

It is unlikely that a new service termination occurs in this very queue in a finite time (since time is rescaled the mean time between two service terminations on the same queue is now $M/\mu$). But if a new arrival occurs, which happens like a Poisson rate $\lambda = \mu$, it will fill the hole, since this queue is now the only shortest queue, of length $\ell - 1$. Therefore holes are created with Poisson rate $\mu r_0$ and canceled with Poisson rate $\mu$; thus the number of holes follows a M/M/1 like process.

■

COROLLARY 4
*The mean value of $m(t)$, $\mathrm{E}[m] = \alpha - \mu/p - 1$.*
*Proof:* At steady state the probability that $m(t) = \ell - 1$, i.e the number of holes is non zero, is $\mu r_0/\mu$ as a classical result about M/M/1 system.

■

PROPOSITION 5
*Let $\beta(s)$ be the Laplace Stieljes of the duration of one active period of a random job; we have*

$$\beta(s) = \frac{(\ell + r_0)(\frac{1}{r_0} + \frac{s}{\mu})}{(\ell + 1)(\frac{r_1 \ell}{r_0} + 1 + \frac{s\ell}{\mu})(1 + \frac{s}{\mu}) - \frac{r_1 \ell^2}{r_0}}$$

*Proof:* Let us look at a fixed active job and consider the behavior of its resident queue as long as this very job is active.

If the total length of the resident queue is $\ell + 1$ no new arrival is expected, since the queue has not the shortest length. Therefore the only expected event is the termination of the active period of any other job in the queue, event that occurs according to a Poisson process of rate $\mu_1 = \mu\ell/(\ell+1)$ per old unit of time.

When the length of the queue is $\ell$ two events can occur: 1) another job termination, with Poisson rate $\mu(\ell-1)/\ell$ per old time unit; 2) a new arrival, if the queue is the shortest one. It is already seen that if a new job termination occur, the queue will be immediately the shortest one and the hole will be filled in $O(1/N)$ old time unit. Dropping these $O(1/N)$ perturbation in the behavior of the queue, the next event will be the arrival of a new active job that will raise the queue length back to $\ell + 1$. This event will occur according to a Poisson process of rate $\mu_0 = \mu r_1/r_0$ (since a proportion of $r_0$ of the arrival rate is absorbed filling the holes).

In conclusion, omitting the time-negligible incursion to length $\ell - 1$, any given queue steadily commutes from length $\ell+1$ to length $\ell$ with Poisson rate $\mu_1$, and from length $\ell$ to length $\ell + 1$ with Poisson rate $\mu_0$. When a queue of length $n$ ($n \geq 1$) is processor shared the service rate per job is $1/n$. Therefore our job receives a service rate of $\theta_1 = 1/(\ell+1)$ when its resident queue length is $\ell + 1$, the service rate is $\theta_0 = 1/\ell$ when the queue length is $\ell$.

Let $T$ be the duration of the active period of our fixed job. Let, for $x \geq 0$, $A_1(x,s) = \mathrm{E}[e^{-Ts}]$ be the Laplace Stieljes transform of the distribution of the random variable $T$ given that the job was initially placed on a queue of length $\ell$ and that the total service time needed by the job is exactly $x$. We have the identity borrowed from straighforward combinatorial analysis:

$$\begin{aligned}\int_0^\infty A_1(x,s)e^{-xt}\mathrm{d}x =& \frac{1}{\mu_1/\theta_1 + t + s/\theta_1} + \\ &+ \frac{\mu_1/\theta_1}{(\mu_1/\theta_1 + t + s/\theta_1)(\mu_0/\theta_0 + t + s/\theta_0)} + \\ &+ \frac{\mu_1\mu_0/\theta_1\theta_0}{(\mu_1/\theta_1 + t + s/\theta_1)^2(\mu_0/\theta_0 + t + s/\theta_0)} + \cdots \\ =& \frac{\mu_1/\theta_1 + \mu_0/\theta_0 + t + s/\theta_0}{(\mu_1/\theta_1 + t + s/\theta_1)(\mu_0/\theta_0 + t + s/\theta_0) - \mu_1\mu_0/\theta_1\theta_0}\,.\end{aligned}$$

Let $\beta_1(s)$ be the Laplace Stieljes transform of the distribution of $T$ only given the fact that the job was initially placed on a queue of total length $\ell$. Since the total service time needed by the job is Poisson of rate $\mu$, we have

$$\begin{aligned}\beta_1(s) &= \int_0^\infty A_1(x,s)e^{-x\mu}\mu\mathrm{d}x \\ &= \mu\frac{\mu_1/\theta_1 + \mu_0/\theta_0 + \mu + s/\theta_0}{(\mu_1/\theta_1 + \mu + s/\theta_1)(\mu_0/\theta_0 + \mu + s/\theta_0) - \mu_1\mu_0/\theta_1\theta_0}\,.\end{aligned}$$

Let $\beta_0(s)$ the alternative of $\beta_1(s)$ given that the job was initially placed on a queue of length $\ell - 1$, we have

$$\beta_0(s) = \mu \frac{\mu_0/\theta_0 + \mu_1/\theta_1 + \mu + s/\theta_1}{(\mu_0/\theta_0 + \mu + s/\theta_0)(\mu_1/\theta_1 + \mu + s/\theta_1) - \mu_0\mu_1/\theta_0\theta_1} .$$

The probability that the job be initially placed on a queue of length $\ell$ is $r_1$, and the probability to be placed on a queue of length $\ell - 1$, is $r_0$. Therefore $\beta(s) = r_0\beta_0(s) + r_1\beta_1(s)$.

■

COROLLARY 6
*The mean active period for a random job is* $(\ell + r_1)/\mu$, *the second moment is*

$$2\frac{(\ell+1)(r_1 r_0 + \ell(\ell+1))}{\mu^2(\ell + r_0)} .$$

*Proof*: The moments of $T$ are directly computed from the derivative of $\beta(s)$ at $s = 0$. Note that the expression of the first moment was expected by Little, since the mean number of queued job per processor is $\ell + r_1$ and $\mu$ is the mean number of arrival of reactivated job per processor and time unit.

■

Figure 1 shows respective mean active periods for good initial placement and random initial placement. It is readily that the longer the active period is, the worst the behaviour of the multiprocessor architecture is, since it stretches too much the time needed for the completion of each quantum of work on a job. The evaluation of the active period for random initial placement is a direct application of the model analyzed in [1] and [2]. We borrow the expression $(1 + \alpha - \frac{\mu}{p} - \sqrt{(1 + \alpha - \frac{\mu}{p})^2 - 4(\frac{\mu}{p})^2})/2\mu$ for the mean active period in random initial placement that we have to compare with $(\alpha - \frac{\mu}{p})/\mu$ for good initial placement, when $\alpha \geq 1 + \frac{\mu}{p}$. Quantity $\frac{\mu}{p}$ is the ratio between the average duration in time quanta of a job inactive period $(1/p)$ and the average number of time quanta that a job receives from its host processor during an active period $(1/\mu)$. We show the plot for case $mu = p = 1$ and $\alpha$ ranging from 2 to 5.

## 3 MODEL FOR OPTIMAL RE-BALANCING

In this model we suppose that all jobs are heavy jobs. We consider an initial state of the system where heavy jobs are distributed among the $M$ processors according to a Poisson distribution of mean $\lambda$ jobs per processor. This situation is relevant when a burst of job arrivals occurs, for example $N \times M$ jobs, $N$ being large, with, among them, $\lambda M$ very heavy jobs, the other being very light jobs. Good initial placement leads to the fact that just after the bursty arrivals each processor queues almost exactly $N$ jobs. After a while light jobs evaporate and only remain the heavy jobs at processors. Since initial placement is independent of whether jobs are heavy or light, the number of heavy jobs remaining at each processor is the result of a Bernoulli trial with probability $\lambda/N$ over the $N$ initial jobs. This distribution is suitably approximated by a Poisson distribution of parameter $\lambda$.

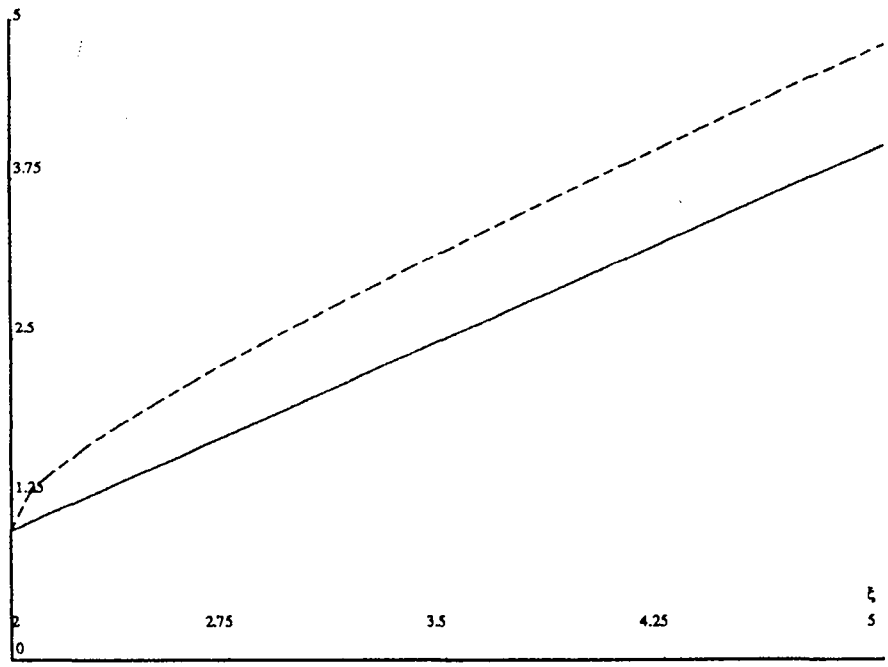


Figure 1: mean active period *versus* $\alpha$ for good initial placement and random initial placement (dashed)

We want to estimate the *completion time*, *i.e* the time needed to complete queued jobs. No new arrival (or reactivation) is expected during the completion time, therefore we measure the time needed to empty the system, jobs departing the system without coming back.

We suppose that the service time needed by each job is a Poisson random variable of parameter $\mu$. Calls to re-balancing algorithm occur according to a Poisson process of parameter $M\tau$, $\tau$ being non-negative. A call, at time $t$, consists in detecting queues with the shortest length $m(t)$ and with the largest, $v(t)$, and making a job moving from the second one to the first one only in the case where the discrepancy between $v(t)$ and $m(t)$ exceeds or equals 2. Job moves are considered instantaneous and without cost (we don't take in account cache warming at host processor, for example).

*A. Equations with fluid approximation*

For this analysis, we adopt old time unit. Matching methodology of section 2, we conjecture the existence of a waterline level $\ell(t)$ whose value is supposed stable during $O(1)$ periods. Almost all queue lengths are supposed greater than $\ell(t)$. Quantity $m(t)$ oscillates between values $\ell(t)-1$ and $\ell(t)$ in $O(1/M)$ periods and the proportion of queues of length $\ell(t)-1$ remains $O(1/M)$. Quantity $v(t)$ is stable during $O(1)$ periods and play a similar role than $\ell(t)$, quantity $v(t)$ can be referred as a *top-line* level.

Let $n(t) = v(t) - m(t)$ be the discrepancy between top-line level and shortest queue length. For integer $i$, between $\ell(t)$ and $n(t)$ (ends included), let $q_i^t$ be the proportion of queues of length $i$ at time $t$. Let $q(z,t) = \sum_{i=\ell(t)}^{v(t)} q_i^t z^i$ for $z$, informal complex valued

variable. According to our conjecture, $q(1) = 1$, dropping $O(1/M)$ terms.

For convenience of notation let $v(t) = v$, $\ell(t) = \ell$ and $q(z,t) = q(z)$. In the general case when $\ell > 0$ and $v - \ell \geq 2$, holes in the waterline are created with rate $M\mu q_\ell$ and resorbed with rate $M\tau$. Therefore our conjecture about $m(t)$ implies that $\tau > \mu q_\ell$ (event that unconditionally happens when $\tau > \mu$).

When $\ell > 0$, $v - \ell \geq 2$ and $\tau > \mu q_\ell$, fluid approximation leads to the following differential equation.

$$\frac{\partial q(z)}{\partial t} = \underbrace{\mu \frac{q(z) - q_\ell}{z}(1-z)}_{\text{(i)}} + \underbrace{\tau(z^{v-1} - z^v)}_{\text{(ii)}} + \underbrace{(\tau - \mu q_\ell)(z^{\ell+1} - z^\ell)}_{\text{(iii)}} . \tag{3.1}$$

(i) Departure process from processor sharing, note that we only consider departure above the waterline level $\ell$. Holes below are resorbed through periodic rebalancing (rate $M\tau$) introducing $O(1/M)$ perturbation, that we drop.

(ii) Periodic job moves starting from queue of length $v$ (rate $M\tau$).

(iii) Periodic job moves arriving to queue of length $\ell$ with rate $M(\tau - \mu q_\ell)$, the other job moves, rate $M\mu q_\ell$, resorb queue of length $\ell - 1$.

Below are the specific remaining cases.

When $v = \ell + 1$, $\ell > 0$ and $\tau > \mu q_\ell$, job move only occurs when the shortest queue is of length $\ell - 1$ (the discrepancy with $v$ is therefore exactly 2), which occurs at rate $M\mu q_\ell$. Adding to that rate the erosion rate by service termination, $M\mu q_{\ell+1}$, the queues of length $\ell$ disappear at global rate $M\mu$. Thus, much simpler equation.

$$\frac{\partial q(z)}{\partial t} = \mu(z^{\ell+1} - z^\ell) . \tag{3.2}$$

Another specific case is when $\ell = 0$ and $v \geq 2$: no hole can be created in the waterline, therefore job moves all reach destination queue of length 0. The differential equation is simpler.

$$\frac{\partial q(z)}{\partial t} = \mu \frac{q(z) - q_0}{z}(1-z) + \tau(z^{v-1} - z^v + z - 1) . \tag{3.3}$$

Last case, $\ell = 0$ and $v = 1$: no job moves are possible (the discrepancy between shortest and largest queue is not big enough):

$$\frac{\partial q(z)}{\partial t} = \mu \frac{q(z) - q_0}{z}(1-z) . \tag{3.4}$$

*B. Resolution of equations, transition and stopping points, results*

We evaluate the completion time, by resolving the differential equations step by step from the initial state at $t = 0$. Each step describe a period during which parameters $\ell(t)$ and $v(t)$ are constant. The integration of the differential equations, within a given step, is explicit, although tedious. The end of a step is marked by a transition within these parameters. Let $t$ be the end of such of a step and therefore called transition point. Transitions occur according to the following rules.

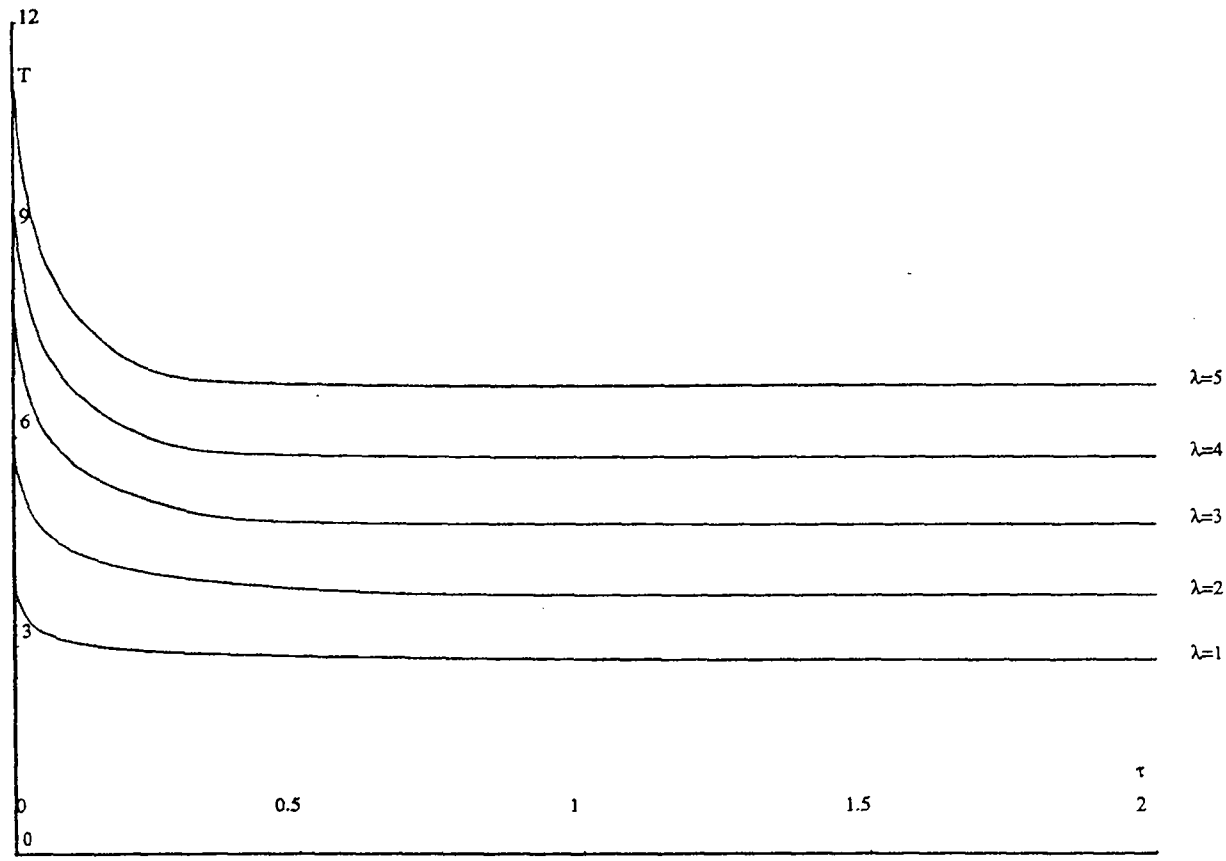


Figure 2: Completion time $T$ versus $\tau$ for different load parameter $\lambda$.

1. If $q_\ell = 0$ , then $(\ell, v) \to (\ell + 1, v)$, the waterline level is raised of one unit.
2. If $q_v = 0$, then $(\ell, v) \to (\ell, v - 1)$, the top-line level goes down of one unit.
3. If $q_\ell = \tau/\mu$ and $\ell > 0$, then $(\ell, v) \to (\ell - 1, v)$ and the new step starts with $q_{\ell-1} = 0$. This transition describes the case where the hole creation rate $\mu q_\ell$ in the waterline exceeds the hole cancelation rate $\tau$. In that case the number of holes will increase unlimitedly and rapidly reach an order $O(M)$ that makes the waterline actually dropping of one unit.

Quantity $T$ is the final stopping point, conventionally defined by $q'(1, T) = 1/M$, or equivalently when the average total number of active jobs is less than or equal to one. This convention is grounded by the fact that job moves do not change the instantaneous total number of jobs.

Below are the plots. We fix $\mu = 1$ and $M = 16$.

*C. Some comment about optimal re-balancing period*

In the previous model no cost was attached to job moves; in such context there is no optimal re-balancing period since the larger is the re-balancing rate, $\tau$, the smaller is the completion time, $T$. But it is an unexpected result of our analysis that the gain in $T$ tends

to be negligible as soon as $\tau > 0.2$, and this whatever be the value of parameter $\lambda$. If a cost were attached to job move it would appear an optimal re-balancing period corresponding to a rate $\tau$ between 0 and 0.2, independently of the value of parameter $\lambda$.

## References


[1] L. BOGUSLAVSKI, A. GREENBERG, P. JACQUET, C. KRUSKAL, A. STOLYAR, "Simple models of memory interference in multiprocessors, Part I: approximate analysis," submitted, 1989.

[2] L. BOGUSLAVSKI, A. GREENBERG, P. JACQUET, C. KRUSKAL, A. STOLYAR, "Simple models of memory interference in multiprocessors, Part II: asymptotics and limit theorems," submitted, 1990.




Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique.

ISSN 0249 - 6399