Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

PNA

Probability, Networks and Algorithms

Probability, Networks and Algorithms

Task allocation in a multi-server system

S.C. Borst, O.J. Boxma, J.F. Groote, S. Mauw

REPORT PNA-R0122 DECEMBER 31, 2001

CWI is the National Research Institute for Mathematics and Computer Science. It is sponsored by the Netherlands Organization for Scientific Research (NWO).
CWI is a founding member of ERCIM, the European Research Consortium for Informatics and Mathematics.

CWI's research has a theme-oriented structure and is grouped into four clusters. Listed below are the names of the clusters and in parentheses their acronyms.

Probability, Networks and Algorithms (PNA)

Software Engineering (SEN)

Modelling, Analysis and Simulation (MAS)

Information Systems (INS)

P.O. Box 94079, 1090 GB Amsterdam (NL)
Kruislaan 413, 1098 SJ Amsterdam (NL)
Telephone +31 20 592 9333
Telefax +31 20 592 4199

ISSN 1386-3711

Task Allocation in a Multi-Server System

Sem Borst*, Onno Boxma, Jan Friso Groote, Sjouke Mauw

CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

and

Department of Mathematics & Computer Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

ABSTRACT

We consider a slotted queueing system with C servers (processors) that can handle tasks (jobs). Tasks arrive in batches of random size at the start of every slot. Any task can be executed by any server in one slot with success probability α. If a task execution fails, then the task must be handled in some later time slot until it has been completed successfully. Tasks may be processed by several servers simultaneously. In that case, the task is completed successfully if the task execution is successful on at least one of the servers.

We determine the distribution of the number of tasks in the system for a broad class of task allocation strategies. Subsequently, we examine the impact of various allocation strategies on the mean number of tasks in the system and the mean response time of tasks. It is proven that both these performance measures are minimized by the strategy which always distributes the tasks over the servers as evenly as possible. Some numerical experiments are performed to illustrate the performance characteristics of the various strategies for a wide range of scenarios.

2000 Mathematics Subject Classification: 60K25, 68M20.
1998 ACM Computing Classification System: C.4.

Keywords & Phrases: task allocation, multi-server queues, response times.

Note: Work of the first two authors carried out in part under the project PNA2.1 "Communication and Computer Networks".

*Also with Bell Laboratories, Lucent Technologies, P.O. Box 636, Murray Hill, NJ 07974-0636, USA

1 Introduction

The problem of efficiently assigning tasks to processors or servers has been studied from many different points of view. Depending on the particular side constraints, the problem can take different shapes and can give rise to essentially different solutions. Approaches from the fields of job shop scheduling and distributed computer systems have been extensively studied.
Here, we study scheduling in the setting of *distributed heterogeneous computing*. The basic observation underlying this branch of computing is the fact that most low-end computers are often idle. Due to the increased connectivity of such computers it is now possible to aggregate these otherwise wasted CPU cycles and form a massively parallel computing resource (see [9]). Participating computers run a client application which on a regular basis receives new tasks from a central server and submits results of completed tasks.
The last few years, several initiatives were taken to use the idle time of computers linked to the Internet for solving specific compute-intensive problems. Most notably, the SETI@home project (see [17, 15]) is dedicated to the search for signs of extraterrestrial civilizations. Radio signals from outer space form a huge amount of (uniform) data which must be analyzed for the occurrence of special patterns. The tasks performed by the participating computers are uniform: after initialization the clients only receive new chunks of data to be searched through. Current capacity of SETI@home is about 15.7 Teraflops, which is much more than the largest supercomputer currently available. There are also more general global computing projects, such as Entropia and United Devices, which support varying tasks. Some of these projects even sell part of their computing power to commercial clients. From a more abstract point of view, the Internet and its connected computers form a giant software and hardware infrastructure, which, in analogy to the power grid, is termed *the grid* (see [6]). The grid should provide dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities. It is intended to support more classes of applications than the class of *high-throughput computing* [6] in which the SETI@home applications falls.
The task allocation strategies used by the central servers of these high-throughput computing projects are only described in very general terms. For SETI@home, it is stated that "The response to [a client's] request includes a work unit (...). Priority goes to those units that have not previously been sent or those that were sent but for which no results were received."
The purpose of our research is to analyze algorithms for task allocation in such a setting. Rather than analyzing or reengineering the strategies currently being used for initiatives such as SETI@home, we study algorithms in an idealized setting. As may be expected the problem of assigning an incoming stream of tasks to a fluctuating set of error-prone computers is not amenable to analysis in full generality. Therefore, we make some simplifications in modeling the system.

We assume a central application which receives a "stochastically distributed" stream of tasks that must be assigned to a collection of servers. We consider independent tasks, which means that execution of one task does not influence execution of another task. The number of servers, C, is assumed to be constant. Processors are error prone and the availability of computing power can vary per server and over time. Therefore, it cannot be predicted when a particular task will be completed, nor will a server report failures. It is, therefore, necessary for the central server to use time-outs or a similar mechanism to guarantee that every task will eventually be processed successfully. We model this behavior in a simplified way by assuming that the system operates in a slotted fashion. By this we mean that tasks are assigned to servers in the beginning of each time slot. It is possible to assign the same task to different servers. Therefore, we assume that tasks are idempotent, i.e. each task can be executed multiple times without negatively impacting the final result. At the end of each time slot every server is assumed to have completed its task. If this is not the case, the server is said to have failed on the task, and the task must be processed in some later time slot. This can be done by one or more different servers, until the task has been completed successfully. We model this failure behavior by assuming that any task can be executed by any server in one slot with success probability α.

In the present paper, we investigate the impact of various task allocation strategies on performance measures such as the mean response time of tasks. We focus on the class $\mathcal{S}$ of strategies that assign the servers to C different tasks whenever there are at least C tasks present. As a way of justification, we will show that the class $\mathcal{S}$ in fact contains the optimal strategy among *all* admissible strategies (in terms of minimum mean response time).

The strategies in $\mathcal{S}$ are basically free in their actions when there are less than C tasks present. For example, tasks may then be allocated to several servers simultaneously. In that case, a task is completed successfully if the task execution is successful on at least one of the servers. However, strategies in $\mathcal{S}$ may also opt to assign exactly one server to each task, thus leaving some servers unutilized, or even decide not to execute any tasks at all until the number of tasks in the system reaches the value C.

For the class of strategies $\mathcal{S}$, we determine the distribution of the number of tasks in the system at slot beginnings. The analysis yields expressions for the mean number of tasks in the system, and thus via Little's theorem [13] the mean response time. Subsequently, we focus on three specific strategies in the class $\mathcal{S}$, corresponding to three allocation rules for the situation with less than C tasks present: (i) The 'lazy' strategy S_0 executes *no* tasks at all; (ii) Strategy S_1 assigns exactly *one* server to each task; (iii) Strategy S^* allocates *all* servers, distributing them over the tasks as evenly as possible. It will be proven that the latter strategy is in fact optimal within the class of all admissible strategies, in the sense that it minimizes the number of tasks in the system (in distribution), and thus the mean

response time.

As stated before, scheduling problems have been studied extensively in many different settings. Our approach differs in several respects from related work. From a queueing angle our model may be viewed as a multi-server queue with geometrically distributed service times (see [13]), however with the unusual element that tasks can be run in parallel.

Most approaches from the area of distributed computing systems consider just a finite set of tasks, rather than a stream of incoming tasks. An example is the DO-ALL problem [5]. Research on the DO-ALL problem concentrates on finding efficient algorithms which can deal with different classes of server failures and restarts (see e.g. [3]). Kanellakis and Shvartsman [11, 12] studied the DO-ALL problem for the shared-memory model of computation, and called it WRITE-ALL. Also for the WRITE-ALL problem a variety of algorithms have been developed (see e.g. [7]). Our problem setting can be considered the natural extension of the DO-ALL problem to an unbounded number of incoming tasks.

Whereas in the WRITE-ALL problem setting execution of a task is assumed to take exactly one unit of time, several other approaches start from a stochastic distribution of task processing times. Bruno et al. [2] show that if the task processing times are independent, identically distributed random variables with some specific common distribution function, then the assignment that attempts to place an equal number of tasks on each machine achieves the stochastically smallest makespan among all assignments. This result is based on the assumption of a fixed number of tasks and error-free processes.

Other approaches relax our requirement that tasks shall be independent. Hsu et al. [10] consider e.g. a fixed set of tasks with precedence constraints that form a directed acyclic graph. Another extension is based on the assumption of extra structure within tasks or processors, such as the cost of a task and the load of a processor (see e.g. [8] for an advanced dynamic scheduling algorithm and an extensive overview). Both approaches are restricted to a finite number of tasks and do not consider server faults.

Maheswaran et al. [16] study scheduling of a fixed number of tasks with random arrival times. Their fault model includes machines that may go off-line and come on-line again. Tasks may have affinity for different machines, and tasks may age while waiting for execution.

There are several workbenches which support high-throughput computing. We mention Condor [1] and EcliPSe [14]. Task distribution in these systems is a complex phenomenon. In Condor, tasks are distributed according to dynamic negotiations between agents representing the customer and the resource owner. In this way constraints from both parties can be satisfied. These systems implement some level of fault-tolerance by means of check-pointing (i.e. periodic savings of data during a task's execution). In our setting, fault-tolerance is reached by duplication and repetition of tasks. Siegel and Ali [18] give an excellent overview of techniques for mapping tasks in heterogeneous computing systems.

This paper is organized as follows. The distribution of the number of tasks in the system for strategies in the class $\mathcal{S}$ is derived in Section 2. In Section 3, the analysis is specialized to the three strategies S_0, S_1, and S^* described above. In Section 4, we show that the allocation rule which distributes the servers over the tasks as evenly as possible maximizes the number of successful task completions. In Section 5, it is then proven that strategy S^*, which follows this rule in each time slot, minimizes the number of tasks in the system (in distribution). In Section 6, we study how the strategies perform if we split a large number of servers into a number of smaller pools. In Section 7, we present the results from some numerical experiments which we conducted to gain further insight into the (absolute and relative) performance of the various strategies for a wide range of scenarios. In Section 8, we make some concluding remarks.

Acknowledgement We thank Wil Kortsmit for his assistance with Mathematica.

2 Steady-state distribution of the number of tasks

We consider a slotted queueing system with C servers that can handle tasks. Let B_n denote the number of new tasks arriving in slot n, $n = 1, 2, \ldots$. We assume that $B_1, B_2, \ldots$ and the generic random variable B are independent, identically distributed random variables with proper probability distribution $\mathbf{P}(B = k)$, $k = 0, 1, 2, \ldots$, and with probability generating function

$$\mathbf{E}[r^B] = \sum_{k=0}^{\infty} \mathbf{P}(B = k) r^k, \quad |r| \leq 1.$$

Define X_n as the number of tasks present at the beginning of slot n, $n = 1, 2, \ldots$, just before the arrival of the B_n new tasks. Denote by A_n the number of successful task completions in slot n, $n = 1, 2, \ldots$.

If $X_n + B_n \geq C$, then C tasks are executed in slot n, each one being successful with probability α. In that case, A_n is binomially distributed with parameters C and α. If $X_n + B_n < C$, then some of the servers may be left idle in slot n, or some of the tasks may be processed by several servers simultaneously. For now, we do not make any specific assumptions regarding the allocation rule used in slot n when $X_n + B_n < C$; we simply assume that the number of successful task completions A_n is a random variable which only depends on $X_n + B_n$. Hence, the stochastic process $\{X_n, n = 1, 2, \ldots\}$ is a Markov chain which evolves as follows:

$$X_{n+1} = X_n + B_n - A_n, \quad n = 1, 2, \ldots. \tag{1}$$

In this section we determine the steady-state distribution $\mathbf{P}(X = k) := \lim_{n\to\infty} \mathbf{P}(X_n = k)$ and its generating function $\mathbf{E}[r^X] = \sum_{k=0}^{\infty} \mathbf{P}(X = k) r^k$. It can be easily verified that a necessary and sufficient condition for this steady-state distribution to exist is $\mathbf{E}[B] < \alpha C$, i.e., the mean number of arriving tasks per slot is strictly less than the processing capacity. Throughout the current paper, this stability condition is assumed to hold.
In the following lemma we give a relation from which $\mathbf{E}[r^X]$ can be obtained. Let $\mathrm{I}(A)$ denote the indicator function of the event A: $\mathrm{I}(A) = 1$ if A is true, and $\mathrm{I}(A) = 0$ otherwise.

Lemma 2.1
The generating function $\mathbf{E}[r^X]$ of the steady-state distribution of the number of tasks at slot beginnings satisfies the following relation, for $|r| \leq 1$:

$$\mathbf{E}[r^X] = \frac{r^C \mathbf{E}[r^{X+B-A} \mathrm{I}(X + B < C)] - (\alpha + (1-\alpha)r)^C \mathbf{E}[r^{X+B} \mathrm{I}(X + B < C)]}{r^C - (\alpha + (1-\alpha)r)^C \mathbf{E}[r^B]}. \quad (2)$$

Proof
It follows from the recurrence relation (1) that

$$\begin{aligned} \mathbf{E}[r^{X_{n+1}}] &= \mathbf{E}[r^{X_n + B_n - A_n}] \\ &= \mathbf{E}[r^{X_n+B_n-A_n} \mathrm{I}(X_n + B_n \geq C)] + \mathbf{E}[r^{X_n+B_n-A_n} \mathrm{I}(X_n + B_n < C)]. \end{aligned} \quad (3)$$

Observe that, for $j \geq C$, $i = 0, \ldots, C$:

$$\mathbf{P}(A_n = i | X_n + B_n = j) = \left(\begin{array}{c} C \\ i \end{array} \right) \alpha^i (1-\alpha)^{C-i},$$

and hence for $j \geq C$, using Newton's binomium:

$$\mathbf{E}[r^{-A_n} | X_n + B_n = j] = \sum_{i=0}^{C} r^{-i} \left(\begin{array}{c} C \\ i \end{array} \right) \alpha^i (1-\alpha)^{C-i} = (\frac{\alpha}{r} + 1 - \alpha)^C. \quad (4)$$

The first term in the right-hand side of (3) may now be rewritten as:

$$\begin{aligned} & \mathbf{E}[r^{X_n+B_n-A_n} \mathrm{I}(X_n + B_n \geq C)] \\ = & \sum_{j=C}^{\infty} r^j \sum_{i=0}^{C} r^{-i} \mathbf{P}(X_n + B_n = j, A_n = i) \\ = & \sum_{j=C}^{\infty} r^j \mathbf{P}(X_n + B_n = j) \sum_{i=0}^{C} r^{-i} \mathbf{P}(A_n = i | X_n + B_n = j) \\ = & \left(\mathbf{E}[r^{X_n+B_n}] - \mathbf{E}[r^{X_n+B_n} \mathrm{I}(X_n + B_n < C)] \right) (\frac{\alpha}{r} + 1 - \alpha)^C. \end{aligned} \quad (5)$$

Observe that X_n and B_n are independent, so that $\mathbf{E}[r^{X_n+B_n}] = \mathbf{E}[r^{X_n}]\mathbf{E}[r^{B_n}]$. In the steady-state situation, combination of (3) and (5) now yields (2).

□

Formula (2) expresses $\mathbf{E}[r^X]$ in terms of the two unknown functions $\mathbf{E}[r^{X+B-A}\mathrm{I}(X+B<C)]$ and $\mathbf{E}[r^{X+B}\mathrm{I}(X+B<C)]$. Let us concentrate on the first one, since the second one may be viewed as a special case of the first one. Using the independence of X_n and B_n, hence of X and B, we can write:

$$\mathbf{E}[r^{X+B-A}\mathrm{I}(X+B<C)] = \sum_{k=0}^{C-1} \mathbf{P}(X=k) \sum_{j=k}^{C-1} r^j \mathbf{P}(B=j-k) \sum_{i=0}^{j} r^{-i} \mathbf{P}(A=i|X+B=j).$$

Hence, the two above-mentioned unknown functions can both be expressed as weighted sums of C unknown probabilities $\mathbf{P}(X=0),\ldots,\mathbf{P}(X=C-1)$. Once the allocation rule is specified for $X_n+B_n<C$, the probabilities $\mathbf{P}(A=i|X+B=j)$ are known, and hence all the weight factors of the probabilities $\mathbf{P}(X=k)$ are known. We now show how the C unknown probabilities $\mathbf{P}(X=0),\ldots,\mathbf{P}(X=C-1)$ may be determined.

Lemma 2.2
The function $r^C-(\alpha+(1-\alpha)r)^C\mathbf{E}[r^B]$ has exactly C zeros $r_1,\ldots,r_C$ with $|r_i|\leq 1$, $i=1,\ldots,C$.

Proof
Consider the circle $C_{1+\epsilon}$ around zero with radius $1+\epsilon$, for small but positive ϵ. According to Rouché's theorem ([4], p. 652), the function $r^C-(\alpha+(1-\alpha)r)^C\mathbf{E}[r^B]$ has exactly as many zeros as r^C inside $C_{1+\epsilon}$ if $|(\alpha+(1-\alpha)r)^C\mathbf{E}[r^B]|<|r^C|$ on $C_{1+\epsilon}$. (We apply Rouché's theorem with the circle $C_{1+\epsilon}$ instead of with the unit circle, since $r=1$ is a zero that lies on the unit circle.) The right-hand term equals $(1+\epsilon)^C$. A routine calculation shows that the left-hand term is bounded by $(1+\epsilon)^C$ when $-\alpha C+\mathbf{E}[B]<0$, which is exactly the stability condition for the system which we assumed to hold. Since ϵ can be chosen arbitrarily small, the lemma follows.

□

Now observe that for $|r|\leq 1$ the probability generating function $\mathbf{E}[r^X]$ is a convergent power series, and hence an analytic function. So for $|r|\leq 1$, whenever the denominator of (2) equals 0, the numerator must also equal 0. For each of the zeros $r_1,\ldots,r_{C-1},r_C=1$, this gives one linear equation in the C unknown probabilities $\mathbf{P}(X=0),\ldots,\mathbf{P}(X=C-1)$. In the case of $r_C=1$, that equation is degenerate. The normalizing condition $\mathbf{E}[r^X]=1$ for $r=1$ provides the required extra equation. Via an application of l'Hôpital's rule to (2) it reads:

$$\alpha C\mathbf{P}(X+B<C)-\mathbf{E}[A\mathrm{I}(X+B<C)]=\alpha C-\mathbf{E}[B]. \tag{6}$$

From these equations, one can (in general only numerically) find the probabilities $\mathbf{P}(X=k)$ for $k=0,\ldots,C-1$. Therefore, these probabilities $\mathbf{P}(X=k)$ can and will be treated

as known constants in the remainder of this paper. In particular, the mean number of tasks at the beginning of a slot, $\mathbf{E}[X]$, can be expressed in terms of these probabilities. Differentiating $\mathbf{E}[r^X]$ w.r.t. r, and substituting $r = 1$, yields $\mathbf{E}[X]$. Write the righthand side of (2) as $T(r)/N(r)$. It is then easily seen, using l'Hôpital's rule and $T(1) = N(1)$ (this is exactly (6)), that

$$\mathbf{E}[X] = \frac{N(1)T''(1) - N''(1)T(1)}{2N(1)N'(1)} = \frac{T''(1) - N''(1)}{2N'(1)}.$$

It thus follows that

$$\begin{aligned}
\mathbf{E}[X] &= \frac{1}{2(\alpha C - \mathbf{E}[B])}\{2(1-\alpha)C\mathbf{E}[B] + \mathbf{E}[B(B-1)] \\
&- (2\alpha - \alpha^2)C(C-1)\mathbf{P}(X+B \geq C) + 2\alpha C\mathbf{E}[(X+B)\mathrm{I}(X+B<C)] \\
&- 2C\mathbf{E}[A\mathrm{I}(X+B<C)] + \mathbf{E}[A^2\mathrm{I}(X+B<C)] \\
&- \mathbf{E}[A(2X+2B-1)\mathrm{I}(X+B<C)]\}. \qquad (7)
\end{aligned}$$

Remark 2.3

From a performance perspective, a crucial characteristic is the response time W, i.e., the amount of time that a task spends in the system before it is successfully completed. The mean response time immediately follows from equation (7) via Little's formula [13]:

$$\mathbf{E}[X] + \mathbf{E}[B] = \mathbf{E}[B]\mathbf{E}[W], \qquad (8)$$

or

$$\mathbf{E}[W] = 1 + \frac{\mathbf{E}[X]}{\mathbf{E}[B]}. \qquad (9)$$

Relation (8) may be heuristically derived using the following 'bookkeeping' argument, cf. [20]. If each task receives one dollar for every slot that it spends in the system, then the total average cost per slot may be evaluated in two different ways which both should yield the same result: the mean number of arriving tasks per slot multiplied with the mean number of slots that a task spends in the system: $\mathbf{E}[B]\mathbf{E}[W]$; or as the mean number of tasks in the system during a slot: $\mathbf{E}[X] + \mathbf{E}[B]$.

Remark 2.4

From equation (7), one immediately obtains a simple expression for $\mathbf{E}[X]$ in a heavy-traffic regime, i.e., when $\alpha C - \mathbf{E}[B] \downarrow 0$. Let us fix the integer number of servers, C, and assume that $\mathbf{E}[B]/\alpha \to C$. In that case, $\mathbf{P}(X + B < C) \downarrow 0$ for any strategy in $\mathcal{S}$. Hence, after an elementary calculation,

$$\lim_{\mathbf{E}[B]/\alpha \to C}(\alpha C - \mathbf{E}[B])\mathbf{E}[X] = \frac{1}{2}[\mathrm{Var}[B] + \alpha(1-\alpha)C]. \qquad (10)$$

Note that the heavy-traffic approximation (10) holds for any strategy in $\mathcal{S}$, regardless of its actions when less than C tasks are present. In other words, in heavy traffic, $\mathbf{E}[X]$ grows asymptotically at the same rate for all strategies in $\mathcal{S}$. This may be understood as follows. As observed when deriving (10), in heavy traffic, there are almost always at least C tasks present. In that situation, the actions of all strategies in $\mathcal{S}$ coincide. As a result, the differences in operation when there are less than C tasks present have no effect. As will be shown later, the class $\mathcal{S}$ includes the optimal strategy among all admissible strategies (in terms of minimum $\mathbf{E}[X]$ and $\mathbf{E}[W]$). Thus, we conclude that in heavy traffic all strategies in $\mathcal{S}$ are in fact asymptotically optimal in that respect.

3 Three candidate strategies

In the previous section, we derived the distribution of the number of tasks in the system for the class of strategies $\mathcal{S}$. The corresponding probability generating function in (2) still contained the term $\mathbf{E}[r^{X+B-A}\mathrm{I}(X+B<C)]$, which may be determined explicitly once the allocation rule is specified for $X+B<C$. In this section, we focus on three specific strategies in the class $\mathcal{S}$, corresponding to three such allocation rules: (i) The 'lazy' strategy S_0 executes *no* tasks at all; (ii) Strategy S_1 assigns exactly *one* server to each task; (iii) Strategy S^* allocates all servers, distributing them over the tasks as evenly as possible. In the next section, we will show that the latter strategy is in fact optimal within the class of all admissible strategies, in the sense that it minimizes the number of tasks in the system (in distribution), and thus the mean response time. Some numerical results for the strategies S_0, S_1 and S^* are presented in Section 7.

The 'lazy' strategy S_0

Strategy S_0 executes no tasks at all until the number of tasks in the system reaches the value C, so that $A=0$ when $X+B<C$. Formula (2) then reduces to

$$\mathbf{E}[r^X] = \frac{\{r^C - (\alpha+(1-\alpha)r)^C\}\mathbf{E}[r^{X+B}\mathrm{I}(X+B<C)]}{r^C-(\alpha+(1-\alpha)r)^C\mathbf{E}[r^B]}, \tag{11}$$

and Formula (7) becomes

$$\begin{aligned}\mathbf{E}[X] &= \frac{1}{2(\alpha C-\mathbf{E}[B])}\{2(1-\alpha)C\mathbf{E}[B]+\mathbf{E}[B(B-1)] \\ &- (2\alpha-\alpha^2)C(C-1)\mathbf{P}(X+B\geq C)+2\alpha C\mathbf{E}[(X+B)\mathrm{I}(X+B<C)]\}. \end{aligned}\tag{12}$$

Strategy S_1

Strategy S_1 assigns exactly one server to each task when there are less than C tasks present.

Thus, A is binomially distributed with parameters $X+B$ and α when $X+B < C$. Formula (2) reduces to

$$\mathbf{E}[r^X] = \frac{r^C \mathbf{E}[(\alpha+(1-\alpha)r)^{X+B}\mathrm{I}(X+B<C)] - (\alpha+(1-\alpha)r)^C \mathbf{E}[r^{X+B}\mathrm{I}(X+B<C)]}{r^C - (\alpha+(1-\alpha)r)^C \mathbf{E}[r^B]}, \quad (13)$$

and Formula (7) becomes

$$\begin{aligned} \mathbf{E}[X] &= \frac{1}{2(\alpha C - \mathbf{E}[B])}\{2(1-\alpha)C\mathbf{E}[B] + \mathbf{E}[B(B-1)] \\ &- (2\alpha-\alpha^2)C(C-1)\mathbf{P}(X+B \geq C) \\ &- (2\alpha-\alpha^2)\mathbf{E}[(X+B)(X+B-1)\mathrm{I}(X+B<C)]\}. \end{aligned} \quad (14)$$

Although duplication of tasks increases the number of successful task completions in a particular slot, it cannot improve the long-term throughput, which is obviously bounded by the mean number of arriving tasks per slot $\mathbf{E}[B]$. Viewed that way, duplication of tasks increases the server utilization without improving the long-term throughput. The server utilization is evidently minimized by the class of 'economic' strategies that never duplicate tasks. A little thought shows that strategy S_1 minimizes the number of tasks in the system among all 'economic' strategies.

Strategy S^*

Strategy S^* always allocates all servers, distributing them over the tasks as evenly as possible. The term $\mathbf{E}[r^{X+B-A}\mathrm{I}(X+B<C)]$ in (2) may thus be determined as follows. Let $m_1(j) := C \bmod j$ and $m_2(j) := C \mathrm{div} j$. Under strategy S^*, if there are $X+B=j<C$ tasks present, then there are $j - m_1(j)$ tasks allocated to $m_2(j)$ servers, and $m_1(j)$ tasks allocated to $m_2(j)+1$ servers. The former ones are completed with success probability

$$\beta(j) := 1-(1-\alpha)^{m_2(j)+1},$$

and the latter ones with success probability

$$\gamma(j) := 1-(1-\alpha)^{m_2(j)}.$$

Similar to the calculation in (4), we have

$$\mathbf{E}[r^{-A}|X+B=j] = (\frac{\beta(j)}{r}+1-\beta(j))^{m_1(j)}(\frac{\gamma(j)}{r}+1-\gamma(j))^{j-m_1(j)}, \quad (15)$$

so that

$$\mathbf{E}[r^{X+B-A}\mathrm{I}(X+B<C)] = \sum_{j=0}^{C-1}\mathbf{P}(X+B=j)(\frac{\beta(j)}{r}+1-\beta(j))^{m_1(j)}(\frac{\gamma(j)}{r}+1-\gamma(j))^{j-m_1(j)}.$$

Substitution in (2) gives $\mathbf{E}[r^X]$, expressed in the probabilities $\mathbf{P}(X+B=j)$, $j=0,\ldots,C-1$, which in their turn can be expressed in the probabilities $\mathbf{P}(X=k)$, $k=0,\ldots,C-1$. In a

similar fashion, $\mathbf{E}[X]$ may be evaluated using (7) and (15). We specify the last three (and most difficult) terms, using (15) each time:

$$\begin{aligned}\mathbf{E}[A\mathrm{I}(X+B<C)] &= \sum_{j=0}^{C-1}\mathbf{P}(X+B=j)\mathbf{E}[A|X+B=j] \\ &= \sum_{j=0}^{C-1}\mathbf{P}(X+B=j)[m_1(j)\beta(j)+(j-m_1(j))\gamma(j)],\end{aligned}$$

$$\begin{aligned}\mathbf{E}[A^2\mathrm{I}(X+B<C)] &= \sum_{j=0}^{C-1}\mathbf{P}(X+B=j)\mathbf{E}[A^2|X+B=j] \\ &= \sum_{j=0}^{C-1}\mathbf{P}(X+B=j)([m_1(j)\beta(j)+(j-m_1(j))\gamma(j)]^2 \\ &+ m_1(j)\beta(j)(1-\beta(j))+(j-m_1(j))\gamma(j)(1-\gamma(j))),\end{aligned}$$

$$\mathbf{E}[A(2X+2B-1)\mathrm{I}(X+B<C)] = \sum_{j=0}^{C-1}\mathbf{P}(X+B=j)\mathbf{E}[A(2X+2B-1)|X+B=j]$$

$$= \sum_{j=0}^{C-1}\mathbf{P}(X+B=j)(2j-1)[m_1(j)\beta(j)+(j-m_1(j))\gamma(j)].$$

Remark 3.1

In Remark 2.4 we obtained a simple heavy-traffic result for the mean number of tasks at slot beginnings, $\mathbf{E}[X]$; and this result was seen to be valid for any strategy in $\mathcal{S}$. Let us now consider the light-traffic situation. We let $C \to \infty$ so that $\mathbf{E}[B]/\alpha C \downarrow 0$. From equation (7), one can derive an expression for $\mathbf{E}[X]$ in this light-traffic scenario. In light traffic, $X+B$ will usually be less than C. Hence, the probabilities $\mathbf{P}(X=j)$ for $j=0,\ldots,C-1$ now play a crucial role, which makes it hard to derive an explicit expression for $\mathbf{E}[X]$ along these lines. However, it is very easy to obtain expressions for $\mathbf{E}[X]$ for the special strategies S_0, S_1 and S^* by intuitive arguments. Under the lazy strategy S_0, the number of tasks in the system will vary between C and $(1-\alpha)C$ according to a saw-tooth pattern. Hence

$$\mathbf{E}[X] \approx \frac{2-\alpha}{2}C. \tag{16}$$

For strategy S_1 in light traffic, $\mathbf{E}[X]$ should approach the mean batch size times the mean number of slots required by a server to handle a task:

$$\mathbf{E}[X] \downarrow \frac{(1-\alpha)\mathbf{E}[B]}{\alpha}. \tag{17}$$

Hence, from (9) we find $\mathbf{E}[W] \downarrow \frac{1-\alpha}{\alpha}$; indeed, under strategy S_1 the response time approaches the service time, which has a geometric distribution with parameter α. Finally, for strategy S^*, all tasks will be successfully handled in their first slot by at least one server:

$$\mathbf{E}[X] \downarrow 0. \tag{18}$$

Note that the (relative) performance of the three strategies drastically differs in light-traffic conditions, in contrast to the heavy-traffic regime where in fact *all* strategies in $\mathcal{S}$ asymptotically coincide. This may be intuitively explained as follows. In light traffic, there are almost always much less than C tasks present. In that situation, the various strategies in $\mathcal{S}$ may differ arbitrarily in their actions.

4 Maximizing the number of task completions

In this section we identify the allocation rule which maximizes the number of successful task completions in a particular slot. As it turns out, the optimal rule distributes the servers over the tasks as evenly as possible. In the next section, we will then prove that strategy S^* which follows this rule in each slot, minimizes the number of tasks in the system and thus the mean response time among all admissible strategies.

In fact, we establish a somewhat more general result which shows that a 'more balanced' allocation yields a larger number of successful tasks. In particular, it follows that the 'most balanced' allocation maximizes the number of successful tasks, and that no duplication is optimal in that respect when there are at least C tasks present.

The desirability of a well-balanced allocation may be heuristically motivated as follows. Assigning additional servers increases the probability that a task will be completed successfully. However, for every extra server that is assigned, the marginal increase in the success probability decreases. Formally speaking, the success probability is a concave increasing function of the number of servers that are being assigned. Thus, the marginal return of assigning additional servers is diminishing. As a result, it is optimal to distribute the servers over the tasks as evenly as possible. In order to measure the degree of 'balancedness', it is useful to adopt the following partial ordering.

Definition 4.1

Let p and q be two M-dimensional vectors. Let $(p_{(1)}, \ldots, p_{(M)})$ and $(q_{(1)}, \ldots, q_{(M)})$ be the components of p and q, respectively, arranged in non-decreasing order. Define $P_m := \sum_{l=1}^{m} p_{(l)}$ and $Q_m := \sum_{l=1}^{m} q_{(l)}$ as the m-th ordered partial sum of the vectors p and q, respectively. Then p is said to majorize q, denoted as $p \succeq q$, if $P_m \geq Q_m$ for all $m = 1, \ldots, M-1$, and $P_M = Q_M$.

Thus, $p \succeq q$ may be interpreted as saying that the vector p is 'more balanced' than q, the average value of the components being equal.

Because of the randomness involved in the execution of tasks, one can only hope to maximize the number of successful task completions in a *stochastic* sense. In order to formalize that notion, we use the following definition of stochastic majorization [19].

Definition 4.2
Let F and G be two non-negative integer-valued random variables. Then F is said to stochastically majorize G, denoted as $F \geq_{\text{st}} G$ (or also as $G \leq_{\text{st}} F$), if $\mathbf{P}(F \geq n) \geq \mathbf{P}(G \geq n)$ for all $n = 1, 2, \ldots$, or equivalently, $\mathbf{P}(F \leq n) \leq \mathbf{P}(G \leq n)$ for all $n = 1, 2, \ldots$.

The following three facts follow directly from the above definition.

Fact 4.3
If $F \geq_{\text{st}} G$, then $\mathbf{E}[F^k] \geq \mathbf{E}[G^k]$ for all $k \geq 1$.

Fact 4.4
Let F and G be two random variables with $F \geq_{\text{st}} G$, both independent of a third random variable H. Then $F + H \geq_{\text{st}} G + H$.

Fact 4.5
Let F, G, and H be three random variables with $F \geq_{\text{st}} G$ and $G \geq_{\text{st}} H$. Then $F \geq_{\text{st}} H$.

Let us now consider a particular slot with M tasks present. Let p_m be the number of servers assigned to the m-th task, with $\sum_{m=1}^{M} p_m \leq C$. Let $S(p)$ be a 0–1 random variable indicating whether or not a particular task is completed successfully (0 for failure, 1 for success) when allocated to p servers, $p = 0, 1, \ldots, C$. Note that $\mathbf{P}(S(p) = 0) = (1 - \alpha)^p$ and $\mathbf{P}(S(p) = 1) = 1 - \mathbf{P}(S(p) = 0)$.
The number of successful task completions may then be formally expressed as

$$T(p_1, \ldots, p_M) = \sum_{m=1}^{M} S(p_m).$$

Since the random variables $S(p_m)$ in the sum are all mutually independent, the distribution of $T(p_1, \ldots, p_M)$ is completely determined by the marginal distribution of the $S(p_m)$ as specified above. Thus, the problem may be phrased as maximizing the quantity $T(p_1, \ldots, p_M)$ (in the sense of Definition 4.2), subject to the capacity constraint $\sum_{m=1}^{M} p_m \leq C$. Note that optimality requires that the latter constraint is satisfied with equality, since assigning additional servers increases the number of successful task completions (strictly, unless $\alpha = 1$).
Denote by $\mathcal{P} := \{p \in \mathbb{N}^M : \sum_{m=1}^{M} p_m = C\}$ the set of non-dominated feasible allocation vectors. Define the 'most balanced' allocation vector p^* with $p^* \succeq q$ for all $q \in \mathcal{P}$ (which is unique up

to a permutation) by $p_1^*, \ldots, p_{m_1}^* = m_2 + 1$ and $p_{m_1+1}^*, \ldots, p_M^* = m_2$, with $m_1 := C \mathrm{mod} M$ and $m_2 := C \mathrm{div} M$.

The next theorem states the main result of this section.

Theorem 4.6
If $p \succeq q$, then $T(p) \geq_{\mathrm{st}} T(q)$. In particular, $T(p^*) \geq_{\mathrm{st}} T(q)$ for all $q \in \mathcal{P}$, with p^* the 'most balanced' allocation vector defined above.

The above theorem states that the 'more balanced' the allocation is, the larger the number of successful tasks (in the sense of Definition 4.2).
In order to prove the above theorem, we first consider the case of $M = 2$ tasks. As it turns out, this case already reveals the main proof ingredients for the case of $M \geq 2$ tasks.

Lemma 4.7
If $p_1 \leq p_2 - 2$, then $T(p_1 + 1, p_2 - 1) \geq_{\mathrm{st}} T(p_1, p_2)$. It follows inductively that, if C is even, then the optimal allocation is $p_1^* = p_2^* = C/2$, while if C is odd, then $p_1^* = (C+1)/2$, $p_2^* = (C-1)/2$.

Proof
Note that $T(p_1, p_2) \leq 2$ for all values of p_1, p_2. Therefore, it suffices to prove that if $p_1 \leq p_2 - 2$, then (i) $\mathbf{P}(T(p_1 + 1, p_2 - 1) = 0) \leq \mathbf{P}(T(p_1, p_2) = 0)$, and (ii) $\mathbf{P}(T(p_1 + 1, p_2 - 1) = 2) \geq \mathbf{P}(T(p_1, p_2) = 2)$.
These two inequalities may be verified through a simple calculation. (As an alternative, a probabilistic coupling argument may be used.)

(i) $$\mathbf{P}(T(p_1, p_2) = 0) = \mathbf{P}(S(p_1) + S(p_2) = 0) = \mathbf{P}(S(p_1) = 0, S(p_2) = 0) =$$

$$\mathbf{P}(S(p_1) = 0)\mathbf{P}(S(p_2) = 0) = (1-\alpha)^{p_1}(1-\alpha)^{p_2} = (1-\alpha)^C$$

for all p_1, p_2 with $p_1 + p_2 = C$.

The above calculation shows that the probability of zero successful task completions is always $(1-\alpha)^C$, irrespective of the allocation (p_1, p_2), which of course may also be seen directly.

(ii) $$\mathbf{P}(T(p_1, p_2) = 2) = \mathbf{P}(S(p_1) + S(p_2) = 2) = \mathbf{P}(S(p_1) = 1, S(p_2) = 1) =$$

$$\mathbf{P}(S(p_1) = 1)\mathbf{P}(S(p_2) = 1) = (1 - \mathbf{P}(S(p_1) = 0))(1 - \mathbf{P}(S(p_2) = 0)) =$$

$$(1 - (1-\alpha)^{p_1})(1 - (1-\alpha)^{p_2}) = 1 - (1-\alpha)^{p_1} - (1-\alpha)^{p_2} + (1-\alpha)^C.$$

Thus, it needs to be shown that if $p_1 \leq p_2 - 2$, then

$$(1-\alpha)^{p_1+1} + (1-\alpha)^{p_2-1} \leq (1-\alpha)^{p_1} + (1-\alpha)^{p_2},$$

which follows directly from the convexity of the function $(1-\alpha)^p$ in p.

□

We now turn to the case of $M \geq 2$ tasks.

Lemma 4.8
If $p_i \leq p_j - 2$, then $T(p_1, \ldots, p_i + 1, \ldots, p_j - 1, \ldots, p_M) \geq_{\text{st}} T(p_1, \ldots, p_i, \ldots, p_j, \ldots, p_M)$. It follows inductively that the optimal allocation vector is p^* defined above. In case $M \geq C$, we find $p_1^*, \ldots, p_C^* = 1$ and $p_{C+1}^*, \ldots, p_M^* = 0$, i.e., no duplication is optimal.

Proof
Using Fact 4.3 and Lemma 4.7,

$$T(p_1, \ldots, p_i + 1, \ldots, p_j - 1, \ldots, p_M) = \sum_{m \neq i,j} S(p_m) + S(p_i + 1) + S(p_j - 1) =$$

$$\sum_{m \neq i,j} S(p_m) + T(p_i + 1, p_j - 1) \geq_{\text{st}} \sum_{m \neq i,j} S(p_m) + T(p_i, p_j) =$$

$$\sum_{m \neq i,j} S(p_m) + S(p_i) + S(p_j) = T(p_1, \ldots, p_i, \ldots, p_j, \ldots, p_M).$$

□

Note that the above lemma already implies that the 'most balanced' allocation maximizes the number of successful tasks. However, in order to complete the proof of Theorem 4.6, it remains to prove the more general result that a 'more balanced' allocation produces a larger number of successful tasks.

Proof of Theorem 4.6
Define $\hat{\mathcal{P}} := \{p \in \mathcal{P} : p_1 \leq \cdots \leq p_M\}$ as the set of feasible allocation vectors whose components are in non-decreasing order. Note that for $p \in \hat{\mathcal{P}}$, $p_{(m)} = p_m$ for all $m = 1, \ldots, M$, and hence $P_m = \sum_{l=1}^{m} p_l$. Because of symmetry, $T(p_1, \ldots, p_M)$ is invariant (in distribution) under permutation of $p_1, \ldots, p_M$. Therefore, it suffices to prove the statement of the theorem for vectors $p, q \in \hat{\mathcal{P}}$.
For any two vectors $p, q \in \hat{\mathcal{P}}$, define $\Delta(p, q) := \sum_{m=1}^{M} (P_m - Q_m)$. By definition, $p \succeq q$ means that $P_m \geq Q_m$ for all $m = 1, \ldots, M$. Hence, $p \succeq q$ implies $\Delta(p, q) \geq 0$, with equality iff $P_m = Q_m$ for all $m = 1, \ldots, M$, i.e., $p = q$.

The proof is by induction on $n = \Delta(p,q)$. We first consider the case $n = 0$, i.e., $p = q$. Then $T(p) \stackrel{d}{=} T(q)$, with $\stackrel{d}{=}$ denoting the equality in distribution, i.e., $\mathbf{P}(T(p) = n) = \mathbf{P}(T(q) = n)$ for all $n = 0, 1, 2, \ldots$, so that the statement is trivially true.

Now suppose that the statement is true for some $n \geq 0$. Let $p, q \in \hat{\mathcal{P}}$ be two vectors with $p \succeq q$ and $\Delta(p,q) = n+1$. Let $l^* := \max\{l : p_l \geq q_l + 1\}$ and $m^* := \min\{m : p_m \leq q_m - 1\}$. The fact that $\sum_{m=1}^{M} p_m = \sum_{m=1}^{M} q_m$ and $\Delta(p,q) \geq 1$ so that $p \neq q$ ensures that l^* and m^* are well-defined. Also, the fact that $p_1 \leq \cdots \leq p_M$ as well as $q_1 \leq \ldots \leq q_M$, and $P_m \geq Q_m$ for all $m = 1, \ldots, M$ implies that $l^* \leq m^* - 1$, $p_m = q_m$ for all $m = l^* + 1, \ldots, m^* - 1$ and $P_m \geq Q_m + 1$ for all $m = l^*, \ldots, m^* - 1$. Moreover, $q_{l^*} \leq q_{l^*+1} - 1$, $q_{m^*} \geq q_{m^*-1} + 1$, and in particular $q_{l^*} \leq q_{m^*} - 2$.

Now define the allocation vector r as follows: $r_{l^*} = q_{l^*} + 1$, $r_{m^*} = q_{m^*} - 1$, and $r_m = q_m$ for all $m \neq l^*, m^*$. According to Lemma 4.8, we have $T(r) \geq_{\text{st}} T(q)$.

Also, note that $r_1 \leq \cdots \leq r_M$, and that $R_m = Q_m$ for all $m = 1, \ldots, l^* - 1$, $R_m = Q_m + 1$ for all $m = l^*, \ldots, m^* - 1$, and $R_m = Q_m$ for all $m = m^*, \ldots, M$. Thus, $p \succeq r$, and $\Delta(p,r) \leq \Delta(p,q) - 1 = n$. Hence, by the induction hypothesis $T(p) \geq_{\text{st}} T(r)$.

Combining the above two stochastic inequalities, and using Fact 4.5, we find that $T(p) \geq_{\text{st}} T(q)$.

□

5 Minimizing the mean response time

In the previous section we identified the allocation rule which maximizes the number of successful task completions in a particular slot. It was shown that the optimal rule distributes the servers over the tasks as evenly as possible. In this section we prove that strategy S^* which follows this rule in each slot minimizes the number of tasks in the system and thus the mean response time among all admissible strategies.

Remark 5.1

At first sight, it may seem completely obvious that always following the rule which maximizes the number of successful task completions also minimizes the number of tasks in the system. Note that maximizing the number of successful tasks indeed minimizes the number of tasks remaining at the end of the slot, and thus the number of tasks at the beginning of the next slot. However, minimizing the number of remaining tasks also reduces the potential for successful task completions in the next slot. Hence, the subtlety lies in proving that the total effect is still favorable, which indeed turns out to be the case.

To illustrate that the latter fact is not entirely trivial, it is worth considering the 'lazy' strategy S_0, which essentially does exactly the opposite and *minimizes* the number of successful

tasks (that is, among strategies in the class $\mathcal{S}$ which are required to come into full action when at least C tasks are present; never processing any tasks would obviously be even worse). Therefore, it may seem equally plausible that strategy S_0 maximizes the number of tasks in the system (among all strategies in $\mathcal{S}$). Surprisingly however, this turns out *not* to be the case.

Like any strategy in the class $\mathcal{S}$, strategy S_0 is required to come into action when the number of tasks in the system reaches the level C, and will then necessarily generate a substantial number of successful tasks, and hence significantly reduce the number of tasks present. Instead, one could imagine a more perverse strategy which processes just a few tasks when the number of tasks approaches the level C. Thus, the strategy prevents that the number of tasks ever reaches the level C so as to avoid being forced into full action. That way, the strategy keeps the number of tasks in the system close to the level C without ever hitting it. In particular, for large values of C, the number of tasks will hover relatively close to the level C. In comparison, recall that under the naively 'lazy' strategy, the number of tasks will oscillate between the levels $(1-\alpha)C$ and C, and thus be lower by a margin $\alpha C/2$ for large values of C.

We first state an auxiliary lemma.

Lemma 5.2

Let X_n and Y_n be two Markov chains with $X_{n+1} = X_n + B_n - F_n$, $Y_{n+1} = Y_n + B_n - G_n$, $n = 1, 2, \ldots$, and D some constant. If $X_1 \geq_{\text{st}} Y_1 + D$, and $y + F_n|(X_n + B_n = x) + D \leq_{\text{st}} x + G_n|(Y_n + B_n = y)$ for all $x \geq y + D$, then $X_n \geq_{\text{st}} Y_n + D$ for all $n = 1, 2, \ldots$.

Proof

The proof is by induction on n. By assumption, the statement is true for $n = 1$. Now suppose that the statement is true for some $n \geq 1$. Then there exist random variables X'_n and Y'_n such that $X'_n \geq Y'_n + D$, $X'_n \stackrel{d}{=} X_n$, and $Y'_n \stackrel{d}{=} Y_n$, cf. [19]. Also, for all $x \geq y + D$, there exist random variables $F'_n(x)$ and $G'_n(y)$ such that $y + F'_n(x) + D \leq x + G'_n(y)$, $F'_n(x) \stackrel{d}{=} F_n|(X_n + B_n = x)$ and $G'_n(y) \stackrel{d}{=} G_n|(Y_n + B_n = y)$. Then $X_{n+1} = X_n + B_n - F_n \stackrel{d}{=} X'_n + B_n - F'_n(X'_n + B_n) \geq Y'_n + D + B_n - G'_n(Y'_n + B_n) \stackrel{d}{=} Y_n + D + B_n - G_n = Y_{n+1} + D$, so that $X_{n+1} \geq_{\text{st}} Y_{n+1} + D$. □

The next lemma demarcates the performance range of the class $\mathcal{S}$ in the form of simple stochastic lower and upper bounds that coincide up to a constant term. Let $D_1, D_2, \ldots$ be a sequence of independent random variables, each binomially distributed with parameters C and α. Let $\tilde{X}_n$ be a random walk with step sizes $B_n - D_n$, reflected at zero, i.e., $\tilde{X}_{n+1} = \max\{\tilde{X}_n + B_n - D_n, 0\}$, with $B_1, B_2, \ldots$ the random batch sizes defined earlier.

Lemma 5.3
For any strategy $S \in \mathcal{S}$, if $\tilde{X}_1 \leq_{\text{st}} X_1^S \leq_{\text{st}} \tilde{X}_1 + C - 1$, then $\tilde{X}_n \leq_{\text{st}} X_n^S \leq_{\text{st}} \tilde{X}_n + C - 1$ for all $n = 1, 2, \ldots$, and in particular $\tilde{X} \leq_{\text{st}} X^S \leq_{\text{st}} \tilde{X} + C - 1$.

Proof
By definition, $X_{n+1}^S = X_n^S + B_n - A_n^S$ and $\tilde{X}_{n+1} = \tilde{X}_n + B_n - \tilde{A}_n$, with $\tilde{A}_n = \min\{D_n, \tilde{X}_n + B_n\}$.
Note that $\tilde{A}_n | (\tilde{X}_n + B_n = y) = \min\{D_n, y\}$ for all y.
Also, $A_n^S | (X_n^S + B_n = x) \leq_{\text{st}} \min\{D_n, x\}$ for all x.
By virtue of the fact that $S \in \mathcal{S}$, we have that $A_n^S | (X_n^S + B_n = x) \stackrel{d}{=} D_n$ for all $x \geq C$.
Thus, for all $x \geq y$, we find that $A_n^S | (X_n^S + B_n = x) \leq_{\text{st}} \min\{D_n, x\} \leq x - y + \min\{D_n, y\} = x - y + \tilde{A}_n | (\tilde{X}_n + B_n = y)$.
In addition, for all $x \geq y + C - 1$, we have $A_n^S | (X_n^S + B_n = x) \geq_{\text{st}} \tilde{A}_n | (\tilde{X}_n + B_n = y)$.
Applying Lemma 5.2, once with $D = 0$ and once with $D = C - 1$, then completes the proof.
□

The next theorem states the main result of this section, demonstrating that strategy S^* minimizes the number of tasks in the system among all admissible strategies.

Theorem 5.4
For any admissible strategy S, if $X_1^S \geq_{\text{st}} X_1^{S^*}$, then $X_n^S \geq_{\text{st}} X_n^{S^*}$ for all $n = 1, 2, \ldots$, and in particular $X^S \geq_{\text{st}} X^{S^*}$.

Proof
By definition, $X_{n+1}^S = X_n^S + B_n - A_n^S$ and $X_{n+1}^{S^*} = X_n^{S^*} + B_n - A_n^{S^*}$.
For all $x \geq y$, we have that $A_n^S | (X_n^S + B_n = x) = T(p_1^S, \ldots, p_x^S) = \sum_{m=1}^{x} S(p_m^S) \leq \sum_{m=1}^{y} S(p_m^S) + x - y \leq_{\text{st}} \sum_{m=1}^{y} S(p_m^{S^*}) + x - y = x - y + T(p_1^{S^*}, \ldots, p_y^{S^*}) = x - y + A_n^{S^*} | (X_n^{S^*} + B_n = y)$.
Applying Lemma 5.2 with $D = 0$ then completes the proof.
□

Using Fact 4.3, we have the following corollary.

Corollary 5.5
For any admissible strategy S, if $X_1^S \geq_{\text{st}} X_1^{S^*}$, then $\mathbf{E}[(X_n^S)^k] \geq \mathbf{E}[(X_n^{S^*})^k]$ for all $k \geq 1$, $n = 1, 2, \ldots$, and in particular $\mathbf{E}[(X^S)^k] \geq \mathbf{E}[(X^{S^*})^k]$ for all $k \geq 1$.

Taking $k = 1$ in the above corollary, and using equation (9), we obtain a similar optimality result for the mean response time.

Corollary 5.6
For any admissible strategy S, $\mathbf{E}[W^{S^*}] \leq \mathbf{E}[W^S]$.

The above corollary confirms that strategy S^* minimizes the mean response time among all admissible strategies.

There are two caveats. First of all, as pointed out in Remark 2.4, in heavy traffic the mean response times grow at the same rate for all strategies in $\mathcal{S}$. Thus, in heavy traffic the mean response time for strategy S^* cannot be significantly smaller than for any other strategy in $\mathcal{S}$. Also, in Remark 3.1 we observed that in light traffic, the mean response times may differ substantially in a relative sense, but will still be moderate in absolute terms for most (sensible) strategies.

Second, the optimality result in terms of the distribution of the number of tasks as stated in Theorem 5.4 does in general not extend to the distribution or even higher moments of the response time. In some situations however, the variance in the response time, or the probability that the response time violates some deadline may be equally important performance measures as the mean response time.

In order to minimize the variance or the violation probabilities, one should presumably give some sort of priority to relatively old tasks or tasks that approach their deadline. To some extent, one can realize priorization while adhering to strategy S^* by selecting older tasks whenever there is a choice. To achieve a strong degree of priority however, one should assign even more servers to the older tasks. On the other hand, if the goal is to minimize a deadline violation probability, then once a task has exceeded its deadline, one should not assign any servers to it anymore until the system has been cleared from all tasks whose deadline has not yet expired.

Thus, in order to optimize these sorts of performance measures, one would occasionally have to deviate from the optimal balanced allocation rule that is followed by strategy S^*. In deviating from the optimal allocation rule however, one would reduce the number of successful task completions, and thus increase the number of tasks in the system, at the risk of a total performance collapse. This suggests that there may be a rather delicate balance between these two conflicting objectives.

As described above, the optimality result for strategy S^* in terms of the distribution of the number of tasks does in general not extend to the distribution or higher moments of the response time. As a further illustrative example, let us suppose we wish to find a strategy which minimizes the second moment of the response time. In fact, let us suppose that there is a cost $2t+1$ incurred when a task is not successfully completed within t slots from its arrival. Then the total cost incurred for a task with response time w is $\sum_{t=1}^{w}(2t-1) = w(w+1)-w = w^2$, so that the average cost rate per unit of time is $\mathbf{E}[B]\mathbf{E}[W^2]$. Thus, minimizing the second moment of the response time is equivalent to minimizing the average cost rate per unit of time.

A natural heuristic is a myopic strategy which simply minimizes the expected cost at the start of the next slot. Observe that this strategy is not necessarily optimal, since it ignores long-term repercussions.

Let us now consider a particular slot with M tasks present, with ages $V_1, \ldots, V_M$. As before, let p_m be the number of servers assigned to the m-th task. Let $U(p) = 1 - S(p)$ be a 0–1 random variable indicating whether or not a particular task fails (0 for success, 1 for failure) when allocated to p servers, $p = 0, 1, \ldots, C$. Then the cost at the start of the next slot may be formally expressed as

$$R(p_1, \ldots, p_M) = 2 \sum_{m=1}^{M} V_m U(p_m).$$

(For convenience, we exclude the cost associated with newly arriving tasks, since these do not depend on the allocation strategy.) In particular, the expected cost at the start of the next slot is

$$\mathbf{E}[R(p_1, \ldots, p_M)] = 2 \sum_{m=1}^{M} V_m \mathbf{E}[U(p_m)] = 2 \sum_{m=1}^{M} V_m (1-\alpha)^{p_m}.$$

Thus, the problem may be phrased as minimizing the latter quantity, subject to the capacity constraint $\sum_{m=1}^{M} p_m = C$. If the integrality constraints are relaxed, then the optimal solution is given by $p_m^* = K \log V_m$ with $K = C / \sum_{m=1}^{M} \log V_m$. This suggests that the number of servers assigned to a task should be roughly proportional to the logarithm of its age in order to minimize the second moment of the response time.

6 Scaling properties

As mentioned earlier, the number of servers, C, may potentially be quite large. It is interesting therefore to understand the scaling properties of the system when the offered traffic and the processing capacity grow large. Specifically, let us compare a system with KC servers and batch sizes $B_n^K = \sum_{k=1}^{K} B_{k,n}$ with K independent systems, each with C servers, and batch sizes $B_{k,n}$ in the k-th system, all distributed as the generic batch size B. Let the other quantities be indexed similarly. For example, X^K is the number of tasks in the aggregated system, and X_k is the number of tasks in the k-th isolated system. Intuitively, one would expect the performance of the aggregated system to be better due to scaling efficiencies. In a similar fashion as in Section 5, it may be shown that that is indeed the case, in the sense that $X^{K,S} \leq_{\rm st} \sum_{k=1}^{K} X_k^S$ for any strategy S such that

$$\sum_{k=1}^{K} T(p_{k1}^S, \ldots, p_{kx_k}^S) \leq_{\rm st} T(p_1^{K,S}, \ldots, p_{\sigma(x)}^{K,S}) \tag{19}$$

for all $(x_1,\ldots,x_K) \in \mathbb{N}^K$, with $\sigma(x) := \sum_{k=1}^{K} x_k$. The above condition is satisfied for strategy S_1:

$$\sum_{k=1}^{K} T(p_{k1}^{S_1},\ldots,p_{kx_k}^{S_1}) = \sum_{k=1}^{K} D(\min\{x_k,C\},\alpha) \stackrel{d}{=} D(\sum_{k=1}^{K} \min\{x_k,C\},\alpha) \leq_{\rm st}$$

$$D(\min\{\sum_{k=1}^{K} x_k, KC\},\alpha) = T(p_1^{K,S_1},\ldots,p_{\sigma(x)}^{K,S_1})$$

for all $(x_1,\ldots,x_K) \in \mathbb{N}^K$, with the terms $D(q_k,\alpha)$ representing independent binomially distributed random variables with parameters q_k and α. For strategy S^* condition (19) is satisfied as well: using Theorem 4.6,

$$\sum_{k=1}^{K} T(p_{k1}^{S^*},\ldots,p_{kx_k}^{S^*}) = T(p_{k1}^{S^*},\ldots,p_{Kx_K}^{S^*}) \leq_{\rm st} T(p_1^{K,S^*},\ldots,p_{\sigma(x)}^{K,S^*}),$$

since $(p_1^{K,S^*},\ldots,p_{\sigma(x)}^{K,S^*}) \succeq (p_{k1}^{S^*},\ldots,p_{Kx_K}^{S^*})$ for all $(x_1,\ldots,x_K) \in \mathbb{N}^K$. Note however that condition (19) does not hold for the 'lazy' strategy S_0.

It is further interesting to examine the scaling properties in heavy-traffic or light-traffic conditions. In heavy traffic, noting that $\mathrm{Var}[B^K] = K\mathrm{Var}[B]$, we obtain from (10),

$$\lim_{\mathbf{E}[B]/\alpha\uparrow C} (\alpha C - \mathbf{E}[B])\mathbf{E}[X^K] = \frac{1}{2}(\mathrm{Var}[B] + \alpha(1-\alpha)C],$$

for any K and for all strategies in $\mathcal{S}$, so that

$$\lim_{\mathbf{E}[B]/\alpha\uparrow C} \frac{\mathbf{E}[X^K]}{\mathbf{E}[X]} = 1,$$

and hence using (9),

$$\lim_{\mathbf{E}[B]/\alpha\uparrow C} \frac{\mathbf{E}[W^K]}{\mathbf{E}[W]} = \frac{1}{K}.$$

Thus, in heavy traffic, the mean response time is reduced by a factor K when the system is scaled up a factor K.

In contrast, in light traffic, we find for all the three strategies S_0, S_1, and S^* that

$$\lim_{P\to\infty} \frac{\mathbf{E}[W^K]}{\mathbf{E}[W]} = 1.$$

Thus, in light traffic, the mean response time does not significantly improve when the system is scaled up. This may be understood by observing that in light traffic there are always plenty of servers available, so that there is little to be gained from sharing servers across independent isolated systems.

Instead of improving performance, the scaling efficiencies may also be exploited to increase the relative load on the system (i.e. increase the offered traffic relative to the processing capacity) while maintaining the performance at a fixed level. The question of course is exactly how the relative load should grow with the size of the system so as to achieve that. The heavy-traffic result (10) suggests that the mean response time $\mathbf{E}[W^K]$ will converge to some constant value τ as $K \to \infty$ if the relative load grows in such a way that the slack capacity $\mathbf{E}[B^K]/\alpha^K - KC$ remains at some fixed value δ, for example by taking $\alpha^K = \mathbf{E}[B]/(C+\delta/K)$. By varying the slack capacity δ, essentially any target value for τ may be achieved.

7 Numerical experiments

We performed some numerical experiments to obtain further insight in the (absolute and relative) performance of the various strategies for a wide range of parameter values.
Therefore, we considered each strategy with an increasing number of servers. We displayed $\mathbf{E}[B]$ in Tables 1, 3 and 5 for strategies S_0, S_1 and S^*, respectively, and similarly, $\mathbf{E}[W]$ in Tables 2, 4 and 6. In each table, we took $C/\mathbf{E}[B]$ equal to 2 and 4. We varied the ratio $\mathbf{E}[B]/\alpha C$ from 0.55 to 0.999 to see the effects from light to heavy traffic. In the experiments we assumed that a constant number of tasks arrived in each slot, i.e., $\mathrm{Var}[B] = 0$. In case B is taken according to some distribution, then the values of $\mathbf{E}[B]$ and $\mathbf{E}[W]$ are expected to increase.
The results in the tables match neatly with the results mentioned throughout the paper. For instance, heavy-traffic behavior corresponds to the rightmost columns in the tables. According to Remark 2.4, $\mathbf{E}[X]$ becomes large and is independent of any particular strategy. For $\mathbf{E}[B]= 1$, $C = 4$ and $\mathbf{E}[B]/\alpha C = 0.999$, Formula (10) would lead to the heavy-traffic approximation $\mathbf{E}[X] \approx 374.87$ which clearly matches the values in the tables.
For light traffic, which corresponds to the leftmost columns in the tables, Remark 3.1 causes us to expect substantial differences. For $\mathbf{E}[B]= 16$, $C = 64$ and $\mathbf{E}[B]/\alpha C = 0.55$, Remark 3.1 would predict $\mathbf{E}[X] \approx 49.45$ (according to Equation (16)), $\mathbf{E}[X] \approx 19.20$ (according to Equation (17)) and $\mathbf{E}[X] \approx 0$ (according to Equation (18)) for the respective strategies, which are quite close to the values in the tables.
Also observe that the tables are in accordance with Lemma 5.3 which implies that the differences between $\mathbf{E}[X]$ for the different strategies never exceed $C - 1$.
Finally, observe that the tables illustrate very well the conclusion from Section 6, stating that for light traffic scaling up has little effect on $\mathbf{E}[W]$, whereas for heavy traffic scaling up with a factor K reduces $\mathbf{E}[W]$ by the same factor.

		Strategy S_0										
$\mathbf{E}[B]$	C	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95	0.99	0.999
1	2	0.56	0.63	0.71	0.83	1.00	1.25	1.67	2.50	5.00	25.00	250.00
2	4	1.64	1.77	1.91	2.06	2.26	2.54	2.97	3.82	6.34	26.35	251.35
4	8	3.82	4.09	4.33	4.57	4.84	5.17	5.65	6.54	9.09	29.12	254.13
8	16	8.18	8.75	9.24	9.67	10.09	10.54	11.12	12.09	14.71	34.78	259.80
16	32	16.91	18.08	19.08	19.93	20.69	21.42	22.22	23.37	26.12	46.29	271.33
32	64	34.36	36.75	38.77	40.50	42.00	43.34	44.64	46.17	49.22	69.57	294.65
1	4	2.47	2.62	2.78	2.99	3.26	3.65	4.29	5.56	9.32	39.33	376.83
2	8	5.50	5.70	5.91	6.16	6.47	6.89	7.55	8.84	12.62	42.64	380.15
4	16	11.62	11.96	12.27	12.60	12.97	13.45	14.17	15.49	19.31	49.36	386.86
8	32	23.96	24.57	25.10	25.61	26.13	26.73	27.55	28.96	32.84	62.94	400.46
16	64	48.68	49.88	50.90	51.80	52.64	53.50	54.53	56.11	60.14	90.33	427.87

Table 1: $\mathbf{E}[X]$ for $\mathbf{E}[B]/\alpha C = 0.55, 0.60, \ldots, 0.95,\ 0.99,\ 0.999$

		Strategy S_0										
$\mathbf{E}[B]$	C	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95	0.99	0.999
1	2	1.56	1.63	1.71	1.83	2.00	2.25	2.67	3.50	6.00	26.00	251.00
2	4	1.82	1.89	1.95	2.03	2.13	2.27	2.49	2.91	4.17	14.18	126.68
4	8	1.95	2.02	2.08	2.14	2.21	2.29	2.41	2.64	3.27	8.28	64.53
8	16	2.02	2.09	2.15	2.21	2.26	2.32	2.39	2.51	2.84	5.35	33.48
16	32	2.06	2.13	2.19	2.24	2.29	2.34	2.39	2.46	2.63	3.89	17.96
32	64	2.07	2.15	2.21	2.27	2.31	2.35	2.39	2.44	2.54	3.17	10.21
1	4	3.48	3.62	3.78	3.99	4.26	4.65	5.29	6.56	10.32	40.33	377.83
2	8	3.75	3.85	3.96	4.08	4.23	4.44	4.78	5.42	7.31	22.32	191.07
4	16	3.91	3.99	4.07	4.15	4.24	4.36	4.54	4.87	5.83	13.34	97.72
8	32	4.00	4.07	4.14	4.20	4.27	4.34	4.44	4.62	5.11	8.87	51.06
16	64	4.04	4.12	4.18	4.24	4.29	4.34	4.41	4.51	4.76	6.65	27.74

Table 2: $\mathbf{E}[W]$ for $\mathbf{E}[B]/\alpha C = 0.55, 0.60, \ldots, 0.95,\ 0.99,\ 0.999$

		Strategy S_1										
$\mathbf{E}[B]$	C	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95	0.99	0.999
1	2	0.10	0.21	0.33	0.48	0.67	0.94	1.37	2.22	4.74	24.75	249.75
2	4	0.20	0.40	0.61	0.84	1.11	1.45	1.94	2.85	5.41	25.46	250.47
4	8	0.40	0.80	1.20	1.61	2.05	2.54	3.18	4.21	6.89	27.02	252.05
8	16	0.80	1.60	2.40	3.20	4.01	4.86	5.81	7.13	10.06	30.37	255.43
16	32	1.60	3.20	4.80	6.40	8.00	9.61	11.28	13.24	16.72	37.40	262.55
32	64	3.20	6.40	9.60	12.80	16.00	19.20	22.42	25.79	30.49	51.98	277.28
1	4	1.21	1.43	1.67	1.94	2.28	2.73	3.43	4.74	8.55	38.60	376.11
2	8	2.40	2.81	3.22	3.66	4.15	4.74	5.56	6.99	10.89	41.01	378.53
4	16	4.80	5.60	6.40	7.21	8.05	8.96	10.07	11.75	15.88	46.15	383.71
8	32	9.60	11.20	12.80	14.40	16.01	17.65	19.42	21.68	26.30	56.90	394.52
16	64	19.20	22.40	25.60	28.80	32.00	35.21	38.46	42.04	47.76	79.07	416.84

Table 3: $\mathbf{E}[X]$ for $\mathbf{E}[B]/\alpha C = 0.55, 0.60, \ldots, 0.95,\ 0.99,\ 0.999$

		Strategy S_1										
$\mathbf{E}[B]$	C	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95	0.99	0.999
1	2	1.10	1.21	1.33	1.48	1.67	1.94	2.37	3.22	5.74	25.75	250.75
2	4	1.10	1.20	1.31	1.42	1.55	1.72	1.97	2.42	3.71	13.73	126.24
4	8	1.10	1.20	1.30	1.40	1.51	1.64	1.80	2.05	2.72	7.76	64.01
8	16	1.10	1.20	1.30	1.40	1.50	1.61	1.73	1.89	2.26	4.80	32.93
16	32	1.10	1.20	1.30	1.40	1.50	1.60	1.71	1.83	2.04	3.34	17.41
32	64	1.10	1.20	1.30	1.40	1.50	1.60	1.70	1.81	1.95	2.62	9.67
1	4	2.21	2.43	2.67	2.94	3.28	3.73	4.43	5.74	9.55	39.60	377.11
2	8	2.20	2.40	2.61	2.83	3.07	3.37	3.78	4.49	6.45	21.50	190.26
4	16	2.20	2.40	2.60	2.80	3.01	3.24	3.52	3.94	4.97	12.54	96.93
8	32	2.20	2.40	2.60	2.80	3.00	3.21	3.43	3.71	4.29	8.11	50.32
16	64	2.20	2.40	2.60	2.80	3.00	3.20	3.40	3.63	3.99	5.94	27.05

Table 4: $\mathbf{E}[W]$ for $\mathbf{E}[B]/\alpha C = 0.55, 0.60, \ldots, 0.95,\ 0.99,\ 0.999$

		Strategy S^*										
$\mathbf{E}[B]$	C	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95	0.99	0.999
1	2	0.01	0.04	0.10	0.19	0.33	0.56	0.96	1.78	4.26	24.25	249.25
2	4	0.02	0.08	0.18	0.33	0.54	0.83	1.30	2.18	4.73	24.77	249.78
4	8	0.04	0.16	0.36	0.64	1.01	1.48	2.13	3.19	5.91	26.07	251.11
8	16	0.08	0.32	0.72	1.28	2.00	2.89	3.98	5.48	8.60	29.05	254.14
16	32	0.16	0.64	1.44	2.56	4.00	5.76	7.85	10.39	14.46	35.54	260.77
32	64	0.32	1.28	2.88	5.12	8.00	11.52	15.68	20.51	26.79	49.31	274.81
1	4	0.20	0.31	0.45	0.65	0.94	1.35	2.02	3.32	7.13	37.17	374.68
2	8	0.34	0.51	0.75	1.06	1.48	2.06	2.93	4.46	8.50	38.74	376.29
4	16	0.61	0.92	1.33	1.87	2.59	3.58	4.99	7.18	11.94	42.73	380.39
8	32	1.17	1.76	2.51	3.50	4.81	6.60	9.22	13.12	19.79	51.89	389.81
16	64	2.31	3.41	4.86	6.77	9.28	12.62	17.57	25.38	36.73	71.78	410.21

Table 5: $\mathbf{E}[X]$ for $\mathbf{E}[B]/\alpha C = 0.55, 0.60, \ldots, 0.95,\ 0.99,\ 0.999$

		Strategy S^*										
$\mathbf{E}[B]$	C	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95	0.99	0.999
1	2	1.01	1.04	1.10	1.19	1.33	1.56	1.96	2.78	5.26	25.25	250.25
2	4	1.01	1.04	1.09	1.16	1.27	1.42	1.64	2.09	3.37	13.38	125.89
4	8	1.01	1.04	1.09	1.16	1.25	1.37	1.53	1.80	2.48	7.52	63.78
8	16	1.01	1.04	1.09	1.16	1.25	1.36	1.50	1.69	2.08	4.63	32.77
16	32	1.01	1.04	1.09	1.16	1.25	1.36	1.49	1.65	1.90	3.22	17.30
32	64	1.01	1.04	1.09	1.16	1.25	1.36	1.49	1.64	1.84	2.54	9.59
1	4	1.20	1.31	1.45	1.65	1.94	2.35	3.02	4.32	8.13	38.17	375.68
2	8	1.17	1.26	1.37	1.53	1.74	2.03	2.46	3.23	5.25	20.37	189.15
4	16	1.15	1.23	1.33	1.47	1.65	1.89	2.25	2.80	3.99	11.68	96.10
8	32	1.15	1.22	1.31	1.44	1.60	1.83	2.15	2.64	3.47	7.49	49.73
16	64	1.14	1.21	1.30	1.42	1.58	1.79	2.10	2.59	3.30	5.49	26.64

Table 6: $\mathbf{E}[W]$ for $\mathbf{E}[B]/\alpha C = 0.55, 0.60, \ldots, 0.95,\ 0.99,\ 0.999$

8 Conclusion

We have analyzed and clarified the task allocation problem in a slotted multi-server system with batch arrivals. The tasks can be assigned to different servers where each task can be processed by several servers. Each server deals with its task exactly within one slot with success probability α.

We were originally motivated by problems that occur when distributing tasks to computers connected via the Internet, as is for instance done in projects such as SETI@home. For our analysis, we simplified the setting by assuming that the tasks arrive and are processed in a slotted fashion, that servers always finish these tasks with probability α and that the number of servers is a priori fixed. We believe that all these parameters can and must be varied to get a more realistic analysis of this particular situation. Additionally, we think that it is interesting to extend our work by adding priorities to tasks that have waited relatively long.

References

[1] J. Basney and M. Livny. Deploying a high throughput computing cluster. In R. Buyya, editor, *High performance cluster computing*, volume 1, chapter 5. Prentice Hall PTR, 1999.

[2] J. Bruno, E. G. Coffman, and P. Downey. Scheduling independent tasks to minimize the makespan on identical machines. *Probability in the Engineering and Informational Sciences*, 9:447–456, 1995.

[3] B. S. Chlebus, R. De Prisco, and A. A. Shvartsman. Performing tasks on synchronous restartable message-passing processors. *Distributed Computing*, 14:49–64, 2001.

[4] J. W. Cohen. *The Single Server Queue*. North-Holland Publishing Company, Amsterdam, 2nd revised edition, 1982.

[5] C. Dwork, J. Y. Halpern, and O. Waarts. Performing work efficiently in the presence of faults. *SIAM Journal on Computing*, 27(5):1457–1491, 1998.

[6] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, San Fransisco, California, 1999.

[7] J. F. Groote, W. H. Hesselink, S. Mauw, and R. Vermeulen. An algorithm for the asynchronous Write-All problem based on process collision. *Distributed Computing*, 14:75–81, 2001.

[8] B. Hamidzadeh, L. Y. Kit, and D. J. Lilja. Dynamic task scheduling using online optimization. *IEEE Transactions on Parallel and Distributed Systems*, 11(11):1151–1163, 2000.

[9] B. Hayes. Collective wisdom. *The American Scientist*, 86(2):118–122, March/April 1998.

[10] T. Hsu, J. C. Lee, D. R. Lopez, and W. A. Royce. Task allocation on a network of processors. *IEEE Transactions on Computers*, 49(12):1339–1353, 2000.

[11] P. C. Kanellakis and A. A. Shvartsman. Efficient parallel algorithms can be made robust. *Distributed Computing*, 5(4):201–217, 1992. (A preliminary version appeared in Proceedings of the 8th ACM PODC, pages 211–222, 1989).

[12] P. C. Kanellakis and A. A. Shvartsman. *Fault-Tolerant Parallel Computation*. Kluwer Academic Publishers, Boston, 1997.

[13] L. Kleinrock. *Queueing Systems*, volume 1: Theory. John Wiley & Sons, New York, 1975.

[14] F. Knop, V. Rego, and V. Sunderam. Fail-safe concurrency in the EcliPSe system. *Concurrency: Practice and Experience*, 8(4):283–312, 1996.

[15] E. Korpela, D. Werthimer, D. Anderson, J. Cobb, and M. Lebofsky. SETI@home: Massively distributed computing for SETI. *Computing in Science and Engineering*, 3(1):78–83, 2001.

[16] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. In *Proceedings of the 8th heterogeneous computing workshop (HCW'99)*, pages 30–44, San Juan, Puerto Rico, 1999. IEEE Computer Society and Office of Naval Research.

[17] Seti@home – the search for extraterrestrial intelligence. http://setiathome.ssl.berkeley.edu/.

[18] H. J. Siegel and S. Ali. Techniques for mapping tasks to machines in heterogeneous computing systems. *Euromicro Journal of Systems Architecture*, 46(8):627–639, 2000.

[19] D. Stoyan. *Comparison Methods for Queues and Other Stochastic Models*. John Wiley & Sons, Chichester, 1983.

[20] H. C. Tijms. *Stochastic Models – An Algorithmic Approach*. John Wiley & Sons, New York, 1994.