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

August 28, 2012

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 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. Subsequently, we determine the distribution of the number of tasks in the system for a broad class of task allocation strategies, which includes the above optimal strategy as a special case. Some numerical experiments are performed to illustrate the performance characteristics of the various strategies.

Key Words & Phrases: Task allocation, Multi-server queues, Response times.

1 Introduction

We study assigning tasks to processors or servers in the setting of *distributed heterogeneous computing*. The basic observation underlying this branch of computing is the fact that most 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 [7]. 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

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

[†]Corresponding author: Jan Friso Groote, Eindhoven University of Technology, Department of Computer Science, P.O. Box 513, 5600 MB Eindhoven, The Netherlands, tel: +31-40-2473549/4416/5003, fax: +31-40-2468508, e-mail: jfg@win.tue.nl. Thanks go to Wil Kortsmit for his assistance with Mathematica.

project [10] 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. 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 [5]).

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 "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. Tasks may be processed by several servers simultaneously. In that case, a task is completed successfully if the task execution is successful on at least one of the servers. In Section 2, 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 addition, it will be proven that the strategy S^*, which follows this rule in each slot, minimizes the number of tasks in the system jointly across time (in distribution), and thus the mean response time among all admissible strategies.

In Section 3, we determine the distribution of the number of tasks for the class of strategies $\mathcal{S}$ which assign the C servers to C different tasks whenever there are at least C tasks in the system. Observe that the class $\mathcal{S}$ includes the optimal strategy S^* as an important special case. For comparison purposes, we also briefly consider the 'lazy' strategy S_0 which executes

no tasks at all when there are fewer than C tasks, and strategy S_1 which assigns exactly *one* server to each task in that situation. The distributional analysis yields expressions for the mean number of tasks in the system, and thus via Little's theorem [9] for the mean response time. In Section 4, we specialize the distributional analysis to the optimal strategy S^*.

The motivation for considering the class $\mathcal{S}$ is that the performance of all these strategies is within a fixed margin from that of the optimal strategy S^*. In particular, the performance will be proven to be asymptotically optimal in a heavy-traffic regime. Besides, the strategies in $\mathcal{S}$ may avoid duplication of tasks. Note that duplication of tasks increases the utilization of the servers and thus processing cost without improving the long-term throughput.

Section 5 shows how the various strategies perform if we split a large number of servers into a number of smaller pools. In Section 6, we present the results from some numerical experiments which we conducted to gain further insight into the (absolute and relative) performance of the strategies S_0, S_1 and S^*.

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 [9]), 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 [4], which consists of performing t tasks reliably in a message-passing synchronous system of p fault-prone processors. 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]). Our problem setting can be considered the natural extension of the DO-ALL problem to an unbounded number of incoming tasks.

Whereas in the DO-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, error-free processes, and tasks must be assigned to exactly one machine.

Other approaches relax our requirement that tasks be independent. Hsu et al. [8] 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. [6] 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.

2 Derivation of the optimal strategy S^*

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 addition, it will be proven that the strategy S^*, which follows this rule in each slot, minimizes the number of tasks in the system jointly across time (in distribution), 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 therefore 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 to a task increases the probability that the task will be completed successfully. However, for every extra server that is assigned to the same task, the marginal increase in the success probability decreases. Formally speaking, the success probability for a task 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 [11].

Definition 2.1

Let c and d be two M-dimensional vectors. Let $(c_{[1]}, \ldots, c_{[M]})$ and $(d_{[1]}, \ldots, d_{[M]})$ be the components of c and d, respectively, arranged in non-increasing order. Define $C_m := \sum_{l=1}^{m} c_{[l]}$ and $D_m := \sum_{l=1}^{m} d_{[l]}$ as the m-th ordered partial sum of the vectors c and d, respectively. Then c is said to be majorized by d, denoted as $c \prec d$, if $C_m \leq D_m$ for all $m = 1, \ldots, M-1$, and $C_M = D_M$.

Thus, $c \prec d$ may be interpreted as saying that the vector c is 'more balanced' than d, 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 [12].

Definition 2.2

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

The following three facts follow directly from the above definition.

Fact 2.3

If $X \geq_{\rm st} Y$, then $\mathbf{E}[X^k] \geq \mathbf{E}[Y^k]$ for all $k \geq 1$.

Fact 2.4

Let X and Y be two random variables with $X \geq_{\rm st} Y$, both independent of a third random variable Z. Then $X + Z \geq_{\rm st} Y + Z$.

Fact 2.5

Let X, Y, and Z be three random variables with $X \geq_{\rm st} Y$ and $Y \geq_{\rm st} Z$. Then $X \geq_{\rm st} Z$.

Let us now consider a particular slot with M tasks present. Let c_m be the number of servers assigned to the m-th task, with $\sum_{m=1}^{M} c_m \leq C$. Let $S(c)$ 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 c servers, $c = 0, 1, \ldots, C$. Note that $\mathbf{P}(S(c) = 0) = (1-\alpha)^c$ and $\mathbf{P}(S(c) = 1) = 1 - \mathbf{P}(S(c) = 0)$.

The number of successful task completions may then be formally expressed as

$$T(c_1, \ldots, c_M) = \sum_{m=1}^{M} S(c_m).$$

Since the random variables $S(c_m)$ in the sum are all mutually independent, the distribution of $T(c_1, \ldots, c_M)$ is completely determined by the marginal distribution of the $S(c_m)$ as specified above. Thus, the problem may be phrased as maximizing the quantity $T(c_1, \ldots, c_M)$ (in the sense of Definition 2.2), subject to the capacity constraint $\sum_{m=1}^{M} c_m \leq C$. 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 $\mathcal{C} := \{c \in \mathbb{N}^M : \sum_{m=1}^{M} c_m = C\}$ the set of non-dominated feasible allocation vectors. Define the 'most balanced' allocation vector c^* with $c^* \prec d$ for all $d \in \mathcal{C}$ (which is unique up to a permutation) by $c^*_1, \ldots, c^*_{m_1} = m_2 + 1$ and $c^*_{m_1+1}, \ldots, c^*_M = m_2$, with $m_1 := C \mathrm{mod} M$ and $m_2 := C \mathrm{div} M$.

The next theorem states the main result of this section saying that the 'more balanced' the allocation is, the larger the number of successful tasks (in the sense of Definition 2.2).

Theorem 2.6
If $c \prec d$, then $T(c) \geq_{\mathrm{st}} T(d)$. In particular, $T(c^*) \geq_{\mathrm{st}} T(d)$ for all $d \in \mathcal{C}$, with c^* the 'most balanced' allocation vector defined above.

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 2.7
If $c_1 \leq c_2 - 2$, then $T(c_1 + 1, c_2 - 1) \geq_{\mathrm{st}} T(c_1, c_2)$.

Proof
Note that $T(c_1, c_2) \leq 2$ for all values of c_1, c_2. Therefore, it suffices to prove that if $c_1 \leq c_2 - 2$, then (i) $\mathbf{P}(T(c_1+1, c_2-1) = 0) \leq \mathbf{P}(T(c_1, c_2) = 0)$, and (ii) $\mathbf{P}(T(c_1+1, c_2-1) = 2) \geq \mathbf{P}(T(c_1, c_2) = 2)$.

These two inequalities may be verified through a simple calculation. (As an alternative, a probabilistic coupling argument may be used.)

(i) $\mathbf{P}(T(c_1, c_2) = 0) = (1-\alpha)^C$ for all c_1, c_2 with $c_1 + c_2 = C$.

(ii) $\mathbf{P}(T(c_1, c_2) = 2) = 1 - (1-\alpha)^{c_1} - (1-\alpha)^{c_2} + (1-\alpha)^C$.

Thus, it needs to be shown that if $c_1 \leq c_2 - 2$, then

$$(1-\alpha)^{c_1+1} + (1-\alpha)^{c_2-1} \leq (1-\alpha)^{c_1} + (1-\alpha)^{c_2},$$

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

It follows inductively that, if C is even, then the optimal allocation is $c_1 = c_2 = C/2$, while if C is odd, then $c_1 = (C+1)/2$, $c_2 = (C-1)/2$. This is exactly the most balanced allocation vector c^* defined above.

□

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

Lemma 2.8
If $c_i \leq c_j - 2$, then $T(c_1, \ldots, c_i + 1, \ldots, c_j - 1, \ldots, c_M) \geq_{\text{st}} T(c_1, \ldots, c_i, \ldots, c_j, \ldots, c_M)$.

Proof
Using Fact 2.4 and Lemma 2.7,

$$T(c_1, \ldots, c_i + 1, \ldots, c_j - 1, \ldots, c_M) = \sum_{m \neq i,j} S(c_m) + S(c_i + 1) + S(c_j - 1) =$$

$$\sum_{m \neq i,j} S(c_m) + T(c_i + 1, c_j - 1) \geq_{\text{st}} \sum_{m \neq i,j} S(c_m) + T(c_i, c_j) =$$

$$\sum_{m \neq i,j} S(c_m) + S(c_i) + S(c_j) = T(c_1, \ldots, c_i, \ldots, c_j, \ldots, c_M).$$

Again, it follows inductively that the optimal allocation is the most balanced allocation vector c^* defined above. In case $M \geq C$, we have $c_1^*, \ldots, c_C^* = 1$ and $c_{C+1}^*, \ldots, c_M^* = 0$, i.e., no duplication is optimal. We define $\mathcal{S}$ as the class of strategies which satisfy the latter optimality property and assign the C servers to C different tasks whenever there are at least C tasks in the system.

□

In order to complete the proof of Theorem 2.6, it remains to prove the more general result that a 'more balanced' allocation produces a larger number of successful tasks, and in particular that the 'most balanced' allocation maximizes the number of successful tasks. The latter statements follows directly from the above lemma, using Fact 2.5 combined with Muirhead's lemma [11], which shows that if $c \prec d$, then c can be derived from d by successive applications of a finite number of 'transfers' as considered in the above lemma.

We now proceed to prove that the strategy S^*, which selects the most balanced allocation in each slot, minimizes the number of tasks in the system jointly across time, and thus the mean response time among all admissible strategies.

Remark 2.9
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. 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 as formalized in Lemma 2.10 below. To illustrate that the latter fact is not entirely trivial, it is

worth considering the 'lazy' strategy S_0 which executes *no* tasks at all when there are fewer than C tasks in the system, and thus *minimizes* the number of successful task completions in every given slot (among the strategies in $\mathcal{S}$). Therefore, it may seem equally plausible that strategy S_0 maximizes the number of tasks in the system at any time (among the strategies in $\mathcal{S}$). Surprisingly however, this turns out *not* to be the case. For example, the number of tasks in the system is larger for the somewhat perverse strategy which processes just a few tasks when the number of tasks approaches the level C so as to avoid ever being forced into full action.

Let B_n be the number of new tasks arriving in slot n. Let X_n and X_n^* be the number of tasks at the beginning of slot n under some arbitrary strategy S and the optimal strategy S^*, respectively, just before the arrival of the new tasks.

Lemma 2.10
If $X_n^* \leq X_n$, then $X_{n+1}^* \leq X_{n+1}$ (under appropriate coupling).

Proof
By definition, $X_{n+1}^* = X_n^* + B_n - T(c_1^*, \ldots, c_{X_n^*+B_n}^*)$ and $X_{n+1} = X_n + B_n - T(c_1, \ldots, c_{X_n+B_n})$. We may write $T(c_1^*, \ldots, c_{X_n^*+B_n}^*) = \sum_{m=1}^{X_n^*+B_n} S(c_m^*)$ and $T(c_1, \ldots, c_{X_n+B_n}) = \sum_{m=1}^{X_n+B_n} S(c_m) \leq \sum_{m=1}^{X_n^*+B_n} S(c_m) + X_n - X_n^*$ since $X_n^* \leq X_n$ and $S(c) \leq 1$ for any c. According to Theorem 2.6, $\sum_{m=1}^{X_n^*+B_n} S(c_m) \leq_{\text{st}} \sum_{m=1}^{X_n^*+B_n} S(c_m^*)$, so there exists a coupling [12] such that $X_{n+1}^* \leq X_{n+1}$.
□

Using a forward induction argument [13], the above lemma readily yields the next theorem, demonstrating that strategy S^* minimizes the number of tasks in the system jointly across time among all admissible strategies.

Theorem 2.11
If $X_1^* \leq_{\text{st}} X_1$, then $\{X_n^*\}_{n=1}^{\infty} \leq_{\text{st}} \{X_n\}_{n=1}^{\infty}$.

Inspection of the proof shows that the above optimality result extends to the situation where the number of available servers varies over time according to some exogenous stochastic process (possibly modeling additional random fluctuations in server availability). Using Fact 2.3, we have the following corollary.

Corollary 2.12
If $X_1^* \leq_{\text{st}} X_1$, then $\mathbf{E}[(X_n^*)^k] \leq \mathbf{E}[(X_n)^k]$ for all $k \geq 1$, $n = 1, 2, \ldots$.

Theorem 2.11 immediately implies that strategy S^* also maximizes the number of successful task completions jointly across time. With all departures occurring earlier, it follows that strategy S^* also minimizes the total flow time up to any given time as well as the mean response time among all admissible strategies, as may also be concluded by taking $k = 1$ in the above corollary and using equation (7) given in the next section.

There are two caveats. First of all, in heavy traffic the mean response times grow at the same rate for all strategies in $\mathcal{S}$, as will be demonstrated in Remark 3.4 in the next section.

Thus, in heavy traffic the mean response time for strategy S^* cannot be significantly smaller than for any other strategy in $\mathcal{S}$. This is corroborated by Theorem 2.14 below which shows that the difference between any two strategies in $\mathcal{S}$ is bounded by a constant term. Also, 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 as will be illustrated in Remark 4.1.

Second, the optimality result in terms of the distribution of the number of tasks as stated in Theorem 2.11 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 prioritization 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 cleared 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.

We finally provide simple stochastic lower and upper bounds for all strategies in $\mathcal{S}$ that coincide up to a constant term. Let $D_1, D_2, \ldots$ be a sequence of independent random variables, 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.

The next lemma is a counterpart of Lemma 2.10.

Lemma 2.13

If $\tilde{X}_n \leq X_n \leq \tilde{X}_n + C - 1$, then $\tilde{X}_{n+1} \leq X_{n+1} \leq \tilde{X}_{n+1} + C - 1$ (under appropriate coupling).

Proof

By definition, $\tilde{X}_{n+1} = \max\{\tilde{X}_n + B_n - D_n, 0\}$ and $X_{n+1} = X_n + B_n - T(c_1, \ldots, c_{X_n+B_n})$. Noting that $T(c_1, \ldots, c_{X_n+B_n}) \leq X_n + B_n$, we may write $X_{n+1} = \max\{X_n + B_n - T(c_1, \ldots, c_{X_n+B_n}), 0\}$. Further observe that $T(c_1, \ldots, c_{X_n+B_n}) \leq_{\text{st}} D_n$ and $T(c_1, \ldots, c_{X_n+B_n}) \stackrel{d}{=} D_n$ when $X_n + B_n \geq C$. We conclude that there exists a coupling [12] such that $\tilde{X}_{n+1} \leq X_{n+1}$ and $X_{n+1} \leq \tilde{X}_{n+1} + C - 1$ when $X_n + B_n \geq C$. That completes the proof, since the inequality $X_{n+1} \leq \tilde{X}_{n+1} + C - 1$ is trivially satisfied when $X_n + B_n \leq C - 1$.

□

The above lemma directly results in the next theorem, demarcating the performance range of the strategies in $\mathcal{S}$.

Theorem 2.14
If $\tilde{X}_1 \leq_{\rm st} X_1 \leq_{\rm st} \tilde{X}_1 + C - 1$, then $\{\tilde{X}_n\}_{n=1}^{\infty} \leq_{\rm st} \{X_n\}_{n=1}^{\infty} \leq_{\rm st} \{\tilde{X}_n + C - 1\}_{n=1}^{\infty}$.

3 Steady-state distribution of the number of tasks

We now determine the steady-state distribution of the number of tasks for the class of strategies $\mathcal{S}$. As we proved in Theorem 2.14, the performance of all the strategies in $\mathcal{S}$ is within a fixed constant from that of the optimal strategy S^*. As a further justification for considering the class S, we will show the performance to be asymptotically optimal in a heavy-traffic regime.

As before, let B_n be the number of new tasks arriving in slot n. 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, just before the arrival of the B_n new tasks, $n = 1, 2, \ldots$. Denote by A_n be the number of successful task completions in slot n, $n = 1, 2, \ldots$.

If $X_n + B_n \geq C$, then C distinct 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. In the previous section we proved that this last decision is optimal in order to maximize the number of successful task completions in a given slot as well as to minimize the number of tasks in the system over time. 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, 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 3.1
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 $|r| \leq 1$:

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

Proof

It follows from the recurrence relation (1) that

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

The last equality follows since $A_n \sim \mathrm{Bin}(C,\alpha)$ if $X_n + B_n = j \geq C$. 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, (3) 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 via an application of Rouché's theorem (see [1]).

Lemma 3.2

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

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:

$$\alpha C\mathbf{P}(X+B<C) - \mathbf{E}[A\mathrm{I}(X+B<C)] = \alpha C - \mathbf{E}[B]. \quad (4)$$

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 right hand side of (2) as $N(r)/D(r)$. It is then easily seen, using l'Hôpital's rule and $N(1) = D(1)$ (this is exactly (4)), that

$$\mathbf{E}[X] = \frac{D(1)N''(1) - D''(1)N(1)}{2D(1)D'(1)} = \frac{N''(1)-D''(1)}{2D'(1)}.$$

It thus follows that

$$\begin{aligned}\mathbf{E}[X] &= \frac{1}{2(\alpha C - \mathbf{E}[B])}\{2(1-\alpha)C\mathbf{E}[B] + \mathbf{E}[B(B-1)] \\ &- (2\alpha - \alpha^2)C(C-1)\mathbf{P}(X+B \geq C) + 2\alpha C\mathbf{E}[(X+B)\mathrm{I}(X+B<C)] \\ &- 2C\mathbf{E}[A\mathrm{I}(X+B<C)] + \mathbf{E}[A^2\mathrm{I}(X+B<C)] \\ &- \mathbf{E}[A(2X+2B-1)\mathrm{I}(X+B<C)]\}. \end{aligned} \tag{5}$$

Remark 3.3
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 (5) via Little's formula [9]:

$$\mathbf{E}[X] + \mathbf{E}[B] = \mathbf{E}[B]\mathbf{E}[W], \tag{6}$$

or

$$\mathbf{E}[W] = 1 + \frac{\mathbf{E}[X]}{\mathbf{E}[B]}. \tag{7}$$

Remark 3.4
From equation (5), 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,

$$\lim_{\mathbf{E}[B]/\alpha \to C}(\alpha C - \mathbf{E}[B])\mathbf{E}[X] = \frac{1}{2}[\mathrm{Var}[B] + \alpha(1-\alpha)C]. \tag{8}$$

Note that the heavy-traffic approximation (8) holds for any strategy in $\mathcal{S}$, regardless of its actions when less than C tasks are present.

4 Steady-state distribution for strategy S^*

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 the optimal strategy S^*, which is included as a special case in the class $\mathcal{S}$ and always allocates all servers, distributing them over the tasks as evenly as possible. In [1] we also consider the 'lazy' strategy S_0 which executes no tasks at all when $X+B<C$, and strategy S_1 which assigns exactly one server to each task when $X+B<C$. In both these cases, it is fairly easy to evaluate Formula (2). Numerical results for S^*, S_0 and S_1 are presented in Section 6. 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 $m_1(j) := C \mathrm{mod} j$ 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 (3), we have

$$\mathbf{E}[r^{-A}|X+B=j] = (\frac{\beta(j)}{r}+1-\beta(j))^{m_1(j)}(\frac{\gamma(j)}{r}+1-\gamma(j))^{j-m_1(j)}, \tag{9}$$

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 (5) and (9). We specify the last three (and most difficult) terms, using (9) each time to obtain moments under the condition that $X+B=j$:

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

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

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

Remark 4.1

In Remark 3.4 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 (5) 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, intuitive arguments readily yield expressions for $\mathbf{E}[X]$ for S^*, and also for the strategies S_0 and S_1. 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

$$\mathbf{E}[X] \approx \frac{2-\alpha}{2}C. \tag{10}$$

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:

$$\mathbf{E}[X] \downarrow \frac{(1-\alpha)\mathbf{E}[B]}{\alpha}. \tag{11}$$

Hence, from (7) 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:

$$\mathbf{E}[X] \downarrow 0. \tag{12}$$

Note that, as might be expected, 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.

5 Scaling properties

As mentioned earlier, the number of servers, C, may potentially be quite large. It is therefore interesting 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. Using similar stochastic coupling techniques as in the proofs of Lemmas 2.10 and 2.13, it may be shown that the above intuition is indeed correct, in the sense that $X^{K,S} \leq_{\rm st} \sum_{k=1}^{K} X_k^S$ for any strategy S such that

$$\sum_{k=1}^{K} T(c_{k1}^S, \ldots, c_{kx_k}^S) \leq_{\rm st} T(c_1^{K,S}, \ldots, c_{\sigma(x)}^{K,S}) \tag{13}$$

for all $(x_1, \ldots, x_K) \in \mathbb{N}^K$, with $\sigma(x) := \sum_{k=1}^{K} x_k$. It is easily verified that the above condition is satisfied for strategies S_1 and S^*, but not 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 (8),

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

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 (7),

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

		Strategy S_0				Strategy S_1				Strategy S^*			
$\mathbf{E}[B]$	C	0.55	0.75	0.95	0.99	0.55	0.75	0.95	0.99	0.55	0.75	0.95	0.99
1	2	0.56	1.00	5.00	25.00	0.10	0.67	4.74	24.75	0.01	0.33	4.26	24.25
2	4	1.64	2.26	6.34	26.35	0.20	1.11	5.41	25.46	0.02	0.54	4.73	24.77
4	8	3.82	4.84	9.09	29.12	0.40	2.05	6.89	27.02	0.04	1.01	5.91	26.07
8	16	8.18	10.09	14.71	34.78	0.80	4.01	10.06	30.37	0.08	2.00	8.60	29.05
16	32	16.91	20.69	26.12	46.29	1.60	8.00	16.72	37.40	0.16	4.00	14.46	35.54
32	64	34.36	42.00	49.22	69.57	3.20	16.00	30.49	51.98	0.32	8.00	26.79	49.31
1	4	2.47	3.26	9.32	39.33	1.21	2.28	8.55	38.60	0.20	0.94	7.13	37.17
2	8	5.50	6.47	12.62	42.64	2.40	4.15	10.89	41.01	0.34	1.48	8.50	38.74
4	16	11.62	12.97	19.31	49.36	4.80	8.05	15.88	46.15	0.61	2.59	11.94	42.73
8	32	23.96	26.13	32.84	62.94	9.60	16.01	26.30	56.90	1.17	4.81	19.79	51.89
16	64	48.68	52.64	60.14	90.33	19.20	32.00	47.76	79.07	2.31	9.28	36.73	71.78

Table 1: $\mathbf{E}[X]$ for $\mathbf{E}[B]/\alpha C = 0.55, 0.75, 0.95, 0.99$

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

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

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

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.

6 Numerical experiments

We have performed some numerical experiments to obtain further insight in the (absolute and relative) performance of the various strategies for a range of parameter values. Table 1 displays $\mathbf{E}[X]$ for strategies S_0, S_1 and S^* respectively, for various numbers of servers. We took $C/\mathbf{E}[B]$ equal to 2 and 4. We varied the ratio $\mathbf{E}[B]/\alpha C$ from 0.55 to 0.99 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 random, the value of $\mathbf{E}[X]$ is expected to increase.

The results in the table neatly match the results mentioned throughout the paper. For instance, heavy-traffic behavior corresponds to the rightmost column in the table. According to Remark 3.4, $\mathbf{E}[X]$ becomes large and is independent of any particular strategy. For $\mathbf{E}[B]$= 1, $C = 4$ and $\mathbf{E}[B]/\alpha C = 0.99$, Formula (8) would lead to the heavy-traffic approximation $\mathbf{E}[X] \approx 37.37$ which clearly matches the values in the table.

For light traffic, Remark 4.1 causes us to expect substantial differences. For $\mathbf{E}[B]$= 16, $C = 64$ and $\mathbf{E}[B]/\alpha C = 0.55$, Remark 4.1 would predict $\mathbf{E}[X] \approx 49.45$ (according to Equation (10)), $\mathbf{E}[X] \approx 19.20$ (according to Equation (11)) and $\mathbf{E}[X] \approx 0$ (according to Equation (12)) for the respective strategies, which are quite close to the values in the table.

Further notice that the table is in accordance with Theorem 2.14 which implies that the differences among $\mathbf{E}[X]$ for the different strategies never exceed $C - 1$.

Finally, observe that the table illustrates very well the scaling properties described in Section 5, stating that for light traffic scaling up has little effect on $\mathbf{E}[W]$, whereas for heavy traffic scaling up by a factor K reduces $\mathbf{E}[W]$ by the same factor.

Acknowledgment The authors are indebted to the associate editor for various useful suggestions.

References

[1] S. C. Borst, O. J. Boxma, J. F. Groote, and S. Mauw. Task allocation in a multi-server system. CWI Report PNA-R0122, 2001.

[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] 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.

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

[6] 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.

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

[8] 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.

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

[10] 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.

[11] A. W. Marshall, I. Olkin. *Inequalities: Theory of Majorization and Its Applications*. Academic Press, New York, 1979.

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

[13] J. Walrand. *An Introduction to Queueing Networks*. Prentice Hall, 1988.