Linear Programming Based Affinity Scheduling of Independent Tasks on Heterogeneous Computing Systems

Issam Al-Azzoni and Douglas G. Down, Senior Member, IEEE

***Abstract*— Resource management systems (RMS) are an important component in heterogeneous computing (HC) systems. One of the jobs of an RMS is the mapping of arriving tasks onto the machines of the HC system. Many different mapping heuristics have been proposed in recent years. However, most of these heuristics suffer from several limitations. One of these limitations is the performance degradation that results from using outdated global information about the status of all machines in the HC system. This paper proposes several heuristics which address this limitation by only requiring partial information in making the mapping decisions. These heuristics utilize the solution to a linear programming (LP) problem which maximizes the system capacity. Simulation results show that our heuristics perform very competitively while requiring dramatically less information.**

***Index Terms*— distributed systems, load balancing, heterogeneous processors, queueing theory.**

I. Introduction

Widespread availability of low-cost, high performance computing hardware and the rapid expansion of the Internet and advances in computing networking technology have led to an increasing use of heterogeneous computing (HC) systems. An HC system is constructed by networking various machines with different capabilities and coordinating their use to execute a set of tasks. Cluster computer systems are an example of HC systems. These are rapidly gaining acceptance as the preferred way to construct large computing platforms for applications with extensive computer needs (see Sterling *et al.* [1]). Such systems form the building blocks for grids which are becoming very successful in managing and organizing an institution's computing resources (see Foster *et al.* [2]).

An important component of an HC system is its resource management system (RMS) which is responsible for assigning resources to tasks in order to satisfy certain performance requirements. In the HC systems considered here, the RMS consists of a single mapper dedicated for mapping incoming tasks to machines. The mapper maps an arriving task as soon it arrives to a machine in the system. The mapper is assumed to only know the arrival rates of the task classes and the execution rates for the task classes on each machine.

The mapper uses mapping heuristics to map the arriving tasks onto the machines of the system. This paper considers dynamic mapping heuristics. Dynamic mapping heuristics map tasks as they arrive, *i.e.*, in an on-line fashion (Kim *et al.* [3]). On the other hand, static mapping heuristics map tasks to machines in an off-line fashion and know in advance which tasks are to be executed during a given interval of time.

It is necessary for any mapping heuristic to stabilize the system if the system can be stabilized. Furthermore, the mapping heuristic should minimize the mean task waiting time. In addition to guaranteeing stability and improving performance, the mapping heuristic should minimize the amount of state information required in mapping. In large systems, mapping heuristics that require full state information suffer from several limitations. First, there is a significant communication overhead since the mapper needs to communicate with a large number of machines. Also, the synchronization overhead that results from requiring full state information may degrade performance. Another important problem is that the supplied information can be out of date resulting in performance degradation. As observed by Mitzenmacher [4], this is a major limitation of heuristics which attempt to exploit global information to balance load too aggressively.

Motivated by these requirements, we propose several mapping heuristics that perform very competitively and verify their performance using simulation. In particular, the Linear Programming Based Affinity Scheduling (LPAS) heuristic achieves competitive performance and at the same time requires less state information than current heuristics. Furthermore, by solving an allocation LP, the LPAS heuristic provides an explicit method to compute the maximum capacity and to compute the allocation of machines to classes. Our initial work on the LPAS heuristic is presented in Al-Azzoni and Down [5] and He *et al.* [6]. This paper, aiming to be a sole reference for the LPAS heuristic, includes that initial work and extensions to it.

In this paper, we also describe other LPAS-related heuristics which attempt to reduce further the state information required in making mapping decisions. We also introduce the Guided-LPAS heuristic, a modification of the LPAS heuristic which guarantees stability of a stabilizable system. Although the LPAS heuristic does not suffer from the root cause for instability of other heuristics, we are unfortunately not able to prove its stability.

The organization of the paper is as follows. Section II gives the workload model in detail and describes several mapping heuristics. Section III introduces the LPAS heuristic and includes simulation results that compare the performance of various mapping heuristics. Other LPAS-related heuristics that attempt to reduce further the required state information for mapping are discussed in Section IV. The Guided-LPAS heuristic is introduced in Section V. The literature related to this work is discussed in Section VI. Appendix A contains detailed proofs for several results discussed in the paper.

II. Problem Statement

A. Workload Model

In a general HC system, there is a dedicated mapper for assigning incoming tasks to machines. Let the number of machines

in the system be M. It is assumed that the tasks are classified into N classes. Tasks that belong to class i arrive according to a renewal process with rate α_i. Furthermore, the execution time of a task on a machine depends on the class of the task and the machine. Let $\mu_{i,j}$ be the execution rate for tasks of class i at machine j, hence $1/\mu_{i,j}$ is the mean execution time for class i tasks at machine j. We allow $\mu_{i,j} = 0$, which implies machine j is physically incapable of executing class i tasks. Each task class can be executed by at least one machine. Let α be the arrival rate vector, where the ith element of α is α_i. Also, let μ be the execution rate matrix, having (i,j) entry $\mu_{i,j}$. We assume that $\alpha_i > 0$ for all $i = 1, \ldots, N$. Also, given any class i, we assume that there exists at least one machine j such that $\mu_{i,j} > 0$. There are further conditions on the arrival and execution time processes that are needed for the analytic results to hold (see Appendix A). Several techniques for classifying tasks and obtaining the arrival and execution rates in HC systems exist (see Kontothanassis and Goddeau [7]).

The mapping heuristics considered in this paper are immediate mode heuristics [3]. In such heuristics, a mapping decision is made by the mapper as soon as a task arrives. The tasks are assumed to be independent and atomic. Each new task arriving in the system is assigned to one of the machines immediately upon arrival, and after that, the task can only be executed by the machine to which it is assigned. It is assumed that there is no queueing at the mapper and tasks are queued at the machines to which they are assigned. With respect to local scheduling, we assume that each machine can use any policy as long as it is non-idling.

Dynamic mapping heuristics assume that the execution rates are known. A task's execution time is not known until its completion, though the task class and thus its execution rate is known to the mapper and the machines. Furthermore, in most of these heuristics, the mapper uses information supplied by the machines in making mapping decisions. Such information includes, for instance, the machine's expected completion time. Thus, when a task arrives to the system, the mapper contacts the machines whose information is needed, and subsequently, the machine supplies the mapper with the requested information.

B. Mapping Heuristics

A mapper using the MET (minimum execution time) heuristic assigns an incoming task to the machine that has the least expected execution time for the task (Maheswaran *et al.* [8]). Thus, when a new task of class i arrives, the mapper assigns it to a machine j such that $j \in \arg\min_{j'} 1/\mu_{i,j'}$. Ties are broken arbitrarily; for instance, a mapper could pick the machine with the smallest index j when more than one machine has the least expected execution time. The MET heuristic does not require the machines to send their expected completion times back to the mapper as tasks arrive, thus the MET heuristic has the advantage of requiring limited communication between the mapper and machines. However, this heuristic can cause severe load imbalance to a degree that the system is unstable. For example, consider a system with one arrival stream with rate $\alpha_1 = 6$, and two machines with execution rates $\mu_{1,1} = 5$ and $\mu_{1,2} = 3$, respectively. This system will suffer from load imbalance causing instability if the MET heuristic is used, as no tasks are sent to machine 2. It is easy to see that the system can be stabilized with the given value of α_1.

The MCT (minimum completion time) heuristic assigns an arriving task to the machine that has the earliest expected completion time [8]. Several existing resource management systems, *e.g.* NetSolve [9] and SmartNet [10], [11], use the MCT heuristic or other heuristics that are based on the MCT heuristic. The mapper examines all machines in the system to determine the machine with the earliest expected completion time.

There are several limitations for the MCT heuristic. First, the mapper requires full state information since it needs to obtain the queue lengths of all machines in the system. Second, the MCT heuristic suffers from its lack of any foresight about task heterogeneity. It might assign an arriving task to a poor machine which minimizes the task's completion time, yet causes problems for future arrivals. Consider the following system with $M = 7$ and $N = 4$. We will refer to this system as System H. The arrival and execution rates are given by $\alpha = [8.5\ 8.5\ 9.6\ 8.5]$ and

$$\mu = \begin{bmatrix} 5 & 5.02 & 4.95 & 0.001 & 4.7 & 5.2 & 5.25 \\ 0.001 & 5.09 & 4.9 & 4.92 & 5 & 5.13 & 5.14 \\ 4.45 & 5 & 0.001 & 4.45 & 4.9 & 5 & 5.1 \\ 5.02 & 4.95 & 5 & 5.02 & 5.25 & 4.75 & 0.001 \end{bmatrix}.$$

The system contains a few machines that have very poor performance when executing tasks that belong to particular classes. While such values would most likely not arise in practice, we have chosen these values to emphasize the point that assigning a task to a machine that is very poor executing its class may result in significant performance degradation. Since the MCT heuristic maps each arriving task to the machine that minimizes its expected completion time, it may assign an arriving task of class i to a machine j that is very poor executing class i tasks. Since the MCT heuristic does not prevent this from happening, it can result in very poor performance. Other heuristics, including the LPAS heuristic, perform much better than the MCT heuristic in such cases since they avoid mapping an arriving task to its minimum expected completion time machine that could do better for future task arrivals. Simulation results for System H are shown later.

Furthermore, the tendency of the MCT heuristic to make mapping decisions based on the immediate marginal improvement in completion time for an arriving task may be problematic. In fact, using the MCT heuristic may result in having an unstable system even though the system can be stabilized. The instability of the MCT heuristic is demonstrated in Sharifnia [12] by considering the following system with $M = 2$ and $N = 4$. The arrival and execution rates are given by $\alpha = [10\ 10\ 25\ 40]$ and

$$\mu = \begin{bmatrix} 0.065 & 0 \\ 0.06 & 0.06 \\ 0 & 0.02 \\ 0.06 & 0.005 \end{bmatrix}, \text{ respectively.}$$

Simulation experiments show the instability of the MCT heuristic for such a system. Other heuristics, including the LPAS heuristic, do not suffer from this limitation. In fact, the LPAS heuristic does stabilize this system.

In order to address the limitations of the MCT heuristic, the k-percent best (KPB) heuristic [8] identifies for each class a subset consisting of the $(kM/100)$ best machines based on the execution times for the class, where $100/M \leq k \leq 100$. Let S_i^k be the set of the $\lfloor kM/100 \rfloor$ machines that have the smallest expected execution time for class i tasks. The mapper assigns an arriving class i task

to the machine in the subset S_i^k that has the earliest expected completion time. Define $\overline{k} = \lfloor kM/100 \rfloor$ to be the number of machines considered by the KPB heuristic.

The KPB heuristic does not attempt only to assign an arriving task to a superior machine based on execution times, it also attempts to avoid assigning an arriving task to a machine that could do better for tasks that arrive later. As discussed earlier, this foresight is not present in the MCT heuristic. Another advantage of the KPB heuristic is that the mapper needs only to communicate with a subset of the machines for each class, rather than with all machines in the system. Thus, the mapper requires less state information than the MCT heuristic.

While the KPB heuristic succeeds in reducing the required state information for mapping, setting its parameter (k) may be problematic and can only be done in an ad-hoc manner. Instability or severe performance degradation can result if inappropriate values for k are used. Also, the KPB heuristic maps each class to the same number of machines, which may not be desirable.

As will be discussed in the next section, the LPAS heuristic builds on the idea of the KPB heuristic. Instead of mapping each class to a fixed number of machines, the LPAS heuristic maps each class to a different set of machines based on the solution of an allocation LP. Furthermore, by solving an allocation LP, the LPAS heuristic provides an explicit method to compute the maximum capacity and to compute the allocation of machines to classes. This has the advantage of requiring dramatically less state information while at the same time achieving competitive performance levels. The LPAS heuristic maps each class to a much smaller number of machines than the MCT heuristic. Furthermore, the LPAS heuristic provides a systematic way to choose the machines.

III. The LPAS Heuristic

A. Overview

Our proposed heuristic is similar to the KPB heuristic in that the mapper needs only to consider a subset of the machines for each class, however, the determination of this subset requires solving a linear programming (LP) problem (Andradóttir *et al.* [13]). Then, the mapper assigns the task to the machine that has the earliest expected completion time in the subset.

The LPAS heuristic requires solving the following allocation LP, where the decision variables are λ and $\delta_{i,j}$ for $i = 1, \ldots, N$, $j = 1, \ldots, M$ (recall that $\mu_{i,j}$ and α_i are the execution rates and arrival rates for the system, respectively). The variables $\delta_{i,j}$ are to be interpreted as the proportional allocation of machine j to class i.

$$\begin{aligned}
&\max \ \lambda \\
&\text{s.t.} \ \sum_{j=1}^{M} \delta_{i,j}\mu_{i,j} \geq \lambda\alpha_i, \qquad \text{for all } i = 1, \ldots, N, && (1)\\
&\sum_{i=1}^{N} \delta_{i,j} \leq 1, \qquad \text{for all } j = 1, \ldots, M, && (2)\\
&\delta_{i,j} \geq 0, \qquad \text{for all } i = 1, \ldots, N, \text{ and } j = 1, \ldots, M. && (3)
\end{aligned}$$

The left-hand side of (1) represents the total execution capacity assigned to class i by all machines in the system. The right-hand side represents the arrival rate of tasks that belong to class i scaled by a factor of λ. Thus, (1) enforces that the total capacity allocated for a class should be at least as large as the scaled arrival rate for that class. This constraint is needed to have a stable system. The constraint (2) prevents overallocating a machine and (3) states that negative allocations are not allowed.

Let λ^* and $\{\delta^*_{i,j}\}$, $i = 1, \ldots, N$, $j = 1, \ldots, M$, be an optimal solution to the allocation LP. The allocation LP always has a solution, since no lower bound constraint is put on λ. However, the physical meaning of λ^* requires that its value be at least one. In this case, $1/\lambda^*$ is interpreted as the long-run utilization of the busiest machine.

The value λ^* can also be interpreted as the maximum capacity of the system. We define the maximum capacity as follows. Consider a system with given values for α_i ($i = 1, \ldots, N$) and λ^*. If $\lambda^* \leq 1$, then the system is unstable. Thus, the system will be overloaded and tasks queue indefinitely. If, however, $\lambda^* > 1$, then the system can be stabilized even if each arrival rate is increased by a factor less than or equal to λ^* (*i.e.*, the same system with arrival rates $\alpha'_i \leq \lambda^*\alpha_i$, for all $i = 1, \ldots, N$, can be stabilized). In this case, the values $\{\delta^*_{i,j}\}$, $i = 1, \ldots, N$, $j = 1, \ldots, M$, can be interpreted as the long-run fraction of time that machine j should spend on class i in order to stabilize the system under maximum capacity conditions. Let δ^* be the machine allocation matrix where the (i, j) entry is $\delta^*_{i,j}$.

The following theorems summarize these stability results (the proofs are provided in Appendix A):

Theorem 1: If $\lambda^* > 1$, then the system can be stabilized. More specifically, the workload process converges to a steady-state distribution as $t \to \infty$.

Theorem 2: If $\lambda^* < 1$, then the system can not be stabilized.

The LPAS heuristic can be stated as follows. Given a system, solve the allocation LP to find $\{\delta^*_{i,j}\}$, $i = 1, \ldots, N$, $j = 1, \ldots, M$. When a new task of class i arrives, let S_i denote the set of machines whose $\delta^*_{i,j}$ is not zero ($S_i = \{j : \delta^*_{i,j} \neq 0\}$). The mapper assigns the task to the machine $j \in S_i$ that has the earliest expected completion time among the subset of machines S_i. Again, ties are broken arbitrarily. Note that the LPAS heuristic does not use the actual values for $\{\delta^*_{i,j}\}$, beyond differentiating between the zero and nonzero elements. We must solve the allocation LP to know where the zeros are.

The LPAS heuristic considers both the arrival rates and execution rates and their relative values in deciding the allocation of machines to tasks. Furthermore, by solving the allocation LP, the LPAS heuristic provides a systematic approach for setting parameters that guarantee the stability of a stabilizable system. This is an advantage over the KPB heuristic where figuring the correct value for k may not be a trivial task. The KPB heuristic maps each class to $\overline{k}$ machines independent of the class, whereas the LPAS heuristic maps each class to a different subset of the machines based on the solution of the allocation LP. The following example clarifies this point and provides some intuition for the LPAS heuristic.

Consider a system with two machines and two classes of tasks ($M = 2$, $N = 2$). The arrival and execution rates are as follows:

$$\alpha = \begin{bmatrix} 2.45 & 2.45 \end{bmatrix} \text{ and } \mu = \begin{bmatrix} 9 & 5 \\ 2 & 1 \end{bmatrix}.$$

Solving the allocation LP gives $\lambda^* = 1.0204$ and

$$\delta^* = \begin{bmatrix} 0 & 0.5 \\ 1 & 0.5 \end{bmatrix}.$$

A mapper using the LPAS heuristic maps all arriving tasks that belong to class 1 to machine 2. At the times of their arrivals, tasks that belong to class 2 are mapped to the machine, either machine 1 or 2, that has the earliest expected completion time.

Even though machine 1 has the fastest rate for class 1, the mapper does not assign any class 1 tasks to it. Since the system is highly loaded, and since $\frac{\mu_{1,1}}{\mu_{2,1}} < \frac{\mu_{1,2}}{\mu_{2,2}}$ and $\alpha_1 = \alpha_2$, the performance is improved significantly if machine 1 only executes class 2 tasks. In fact, the performance of the LPAS heuristic is better than that of the MCT heuristic. For this particular system, both the MET heuristic and the KPB heuristic (with $\overline{k} = 1$) result in unstable systems. This is because both heuristics map class 2 tasks to machine 1 only, and the system will be unstable since $\alpha_2 > \mu_{2,1}$.

In the LPAS heuristic, the mapper considers a subset of the machines for each class. Ideally, the size of each subset should be much smaller than M. Similar to the KPB heuristic, this has the advantage of requiring less communication between the mapper and the machines. Furthermore, degradation in performance due to outdated information should be minimized. To achieve this, the δ^* matrix should contain a large number of elements that are equal to zero. In fact, there could be many optimal solutions to an allocation LP, and an optimal solution with a larger number of zeros in the δ^* matrix is preferred. The following proposition gives the number of zero elements in the δ^* matrix that could be achieved (the proof can be found in [13]):

Proposition 1: There exists an optimal solution to the allocation LP with at least $NM+1-N-M$ elements in the δ^* matrix equal to zero.

Ideally, the number of zero elements in the δ^* matrix should be $NM + 1 - N - M$. If the number of zero elements is greater, the LPAS heuristic would be significantly restricted in shifting workload between machines resulting in performance degradation. Also, solutions that result in degenerate cases should be avoided. For example, if the δ^* matrix contains no zeros at all, then the LPAS heuristic reduces to the MCT heuristic. Throughout the paper, we use the unique optimal solution in which the δ^* matrix contains exactly $NM + 1 - N - M$ zeros.

The LPAS heuristic can be considered as a dynamic mapping heuristic. As the heuristic only involves solving an LP, it is suited for scenarios when machines are added and/or deleted from the system. On each of these events, one needs to simply solve a new LP and continue with the new values.

B. Simulation Results

1) Overview: We use simulation to compare the performance of the mapping heuristics. The task arrivals are modeled by independent Poisson processes, each with rate α_i, $i = 1, \ldots, N$. Several distributions are used for execution times. Unless otherwise stated, the execution times are exponentially distributed with rates $\mu_{i,j}$, where $1/\mu_{i,j}$ represents the mean execution time of a task of class i at machine j, $i = 1, \ldots, N$, $j = 1, \ldots, M$.

Each simulation experiment models a particular system, characterized by the values of M, N, α_i, and $\mu_{i,j}$, $i = 1, \ldots, N$, $j = 1, \ldots, M$. Each experiment simulates the execution of the corresponding system for 20,000 time-units. Each experiment is repeated 30 times. All confidence intervals are at the 95%-confidence level.

There are several performance metrics that could be used to compare the performance of the mapping heuristics [8]. We have chosen the long-run average number of tasks in the system, $\bar{Q}$, as a metric for performance comparison. This includes the tasks that are waiting for execution at a particular machine and tasks that are executing.

Table 1 lists simulation results for different systems (these results appear in [5]). These systems are discussed in Sections III-B.2 to III-B.6. For each system, the table shows the 95%-confidence interval for $\bar{Q}$ when the corresponding mapping heuristic is used. If a system becomes unstable due to the mapping heuristic used by its mapper, the table just indicates that the system is unstable. Since the MET heuristic results in unstable systems in most of the systems in Table 1, we do not include it here. The table also shows the results of using the KPB heuristic with different values for $\overline{k}$.

In these simulation experiments, we assume that a First-Come-First-Serve (FCFS) scheduling policy is used by the machines. Thus, in this case, the expected completion time of a class i arrival at machine j is given by

$$\frac{1}{\mu_{i,j}} + \sum_{i'=1}^{N} \frac{Q_{i',j}}{\mu_{i',j}},$$

where $Q_{i',j}$ is the number of tasks of class i that are waiting or executing at machine j, at the time of the task arrival.

2) Small Systems: System A in Table 1 is the system discussed in Section III-A. This is a highly loaded system as shown by the large values for $\bar{Q}$. As shown in the table, the MCT heuristic performs poorly with respect to the LPAS heuristic. This is because the MCT heuristic assigns some class 1 tasks to machine 1, although it is more advantageous to dedicate machine 1 for class 2 tasks.

System B is another small system, where $M = 2$ and $N = 2$. The arrival and execution rates are as follows:

$$\alpha = \begin{bmatrix} 5 & 8 \end{bmatrix} \text{ and } \mu = \begin{bmatrix} 8 & 3 \\ 4 & 10 \end{bmatrix}.$$

Solving the allocation LP gives $\lambda^* = 1.3333$ and

$$\delta^* = \begin{bmatrix} 0.8333 & 0 \\ 0.1667 & 1 \end{bmatrix}.$$

As indicated by the nonzero elements of the δ^* matrix, the LPAS heuristic assigns all class 1 tasks to machine 1. Thus, machine 2 becomes dedicated to execute class 2 tasks. This results in improved performance since, in this case, class 2 tasks have a higher arrival rate and they run much faster on machine 2 than on machine 1.

3) Large Systems: System $C1$ is a large system with $M = 30$ and $N = 3$. The machines are grouped into four groups, and each group consists of machines with identical performance. Thus, if two machines are in the same group, then they have the same execution rates. Table 2 shows the execution rates of the groups.

Table 2. Execution rates for System $C1$

Task	Group			
	P	Q	R	S
1	8	4	4	4
2	1	4	1	2
3	4	2	8	4

Groups P, Q, R, and S, consist of 10 machines, 9 machines, 6 machines, and 5 machines, respectively. As Table 2 shows, the groups vary in performance. For instance, a machine in group

Table 1. Comparison of the Mapping Heuristics

System	MCT	KPB	LPAS
A	$(85.68, 110.23)$	$\overline{k} = 1$ Unstable	$(62.56, 82.01)$
B	$(20.05, 21.10)$	$\overline{k} = 1$ $(5.65, 5.73)$	$(5.21, 5.26)$
C1	$(53.99, 54.98)$	$\overline{k} = 14$ $(75.26, 79.13)$	$(47.39, 47.72)$
D	$(22.68, 23.21)$	$\overline{k} = 2$ $(14.75, 14.89)$ $\overline{k} = 3$ $(11.00, 11.04)$	$(10.55, 10.59)$
E	$(27.71, 28.20)$	$\overline{k} = 5$ $(51.65, 55.60)$	$(36.54, 37.07)$
F1	$(19.09, 19.44)$	$\overline{k} = 4$ $(20.77, 21.07)$	$(28.71, 29.05)$
F2	$(46.36, 49.49)$	$\overline{k} = 4$ $(73.44, 81.75)$	$(34.27, 34.89)$
G	$(37.91, 40.43)$	$\overline{k} = 4$ $(42.21, 43.54)$	$(42.05, 43.09)$
H	$(3648.48, 4086.54)$	$\overline{k} = 5$ $(888.62, 1319.97)$	$(131.08, 150.15)$
I1	$(64.20, 66.32)$	$\overline{k} = 14$ $(86.65, 94.15)$	$(50.83, 38)$
I2	$(41.56, 41.82)$	$\overline{k} = 14$ $(53.69, 55.19)$	$(40.57, 40.69)$

P is twice as fast as a machine in group S on tasks of class 1, however, for tasks of class 2, the opposite is true. The arrival rates are given by $\alpha = [45\ 45\ 40]$.

Since System $C1$ consists of some identical machines, there are an infinite number of optimal solutions to the allocation LP. To better capture machine homogeneity of the system, it is desirable to use the unique optimal solution in which machines that belong to the same group have identical values for $\delta^*_{i,j}$. To do this, we solve the allocation LP corresponding to the following system:

$$N = 3, M = 4, \alpha = [45\ 45\ 40], \text{and } \mu = \begin{bmatrix} 80 & 36 & 24 & 20 \\ 10 & 36 & 6 & 10 \\ 40 & 18 & 48 & 20 \end{bmatrix}.$$

Solving the modified allocation LP gives $\lambda^* = 1.1146$ and

$$\delta^* = \begin{bmatrix} 0.6270 & 0 & 0 & 0 \\ 0.3730 & 1 & 0.0712 & 1 \\ 0 & 0 & 0.9288 & 0 \end{bmatrix}.$$

Thus, for System $C1$, we use the δ^* matrix represented in Table 3. In this particular solution, machines that belong to the same group have identical values for $\delta^*_{i,j}$. Note that the δ^* matrix has 44 elements that are equal to zero. However, note that based on Proposition 1, there exists another optimal solution to the allocation LP with 58 elements in the δ^* matrix that are equal to zero.

Table 3. The machine allocation matrix for System $C1$

	Group			
Task	P	Q	R	S
1	0.6270	0	0	0
2	0.3730	1	0.0712	1
3	0	0	0.9288	0

As shown in Table 1, the LPAS heuristic achieves the best results. Note that the KPB heuristic is unstable for $\overline{k} < 14$.

4) Task and Machine Heterogeneity: Systems D through G model different kinds of system heterogeneity. Machine heterogeneity refers to the average variation along the rows, and similarly task heterogeneity refers to the average variation along the columns (see Armstrong [14]). Heterogeneity can be classified into high heterogeneity and low heterogeneity. Based on this, we simulate the following four categories for heterogeneity [14], [8]: (a) high task heterogeneity and high machine heterogeneity (HiHi), (b) high task heterogeneity and low machine heterogeneity (HiLo), (c) low task heterogeneity and high machine heterogeneity (LoHi), and (d) low task heterogeneity and low machine heterogeneity (LoLo).

System D models a HiHi system with $M = 7$ and $N = 4$. The arrival and execution rates are given by $\alpha = [12.5\ 12\ 12.5\ 12]$ and

$$\mu = \begin{bmatrix} 4.5 & 2 & 9.5 & 6.2 & 10.25 & 2.25 & 3.95 \\ 6.2 & 4.5 & 6 & 2 & 4.2 & 5.9 & 10.25 \\ 9.5 & 6.5 & 4 & 10 & 5.9 & 2.25 & 3.95 \\ 2.25 & 10 & 2 & 3.95 & 1.75 & 10 & 1.75 \end{bmatrix}.$$

Solving the allocation LP gives $\lambda^* = 1.3449$ and

$$\delta^* = \begin{bmatrix} 0 & 0 & 0.6907 & 0 & 1 & 0 & 0 \\ 0.2830 & 0 & 0.3093 & 0 & 0 & 0.3861 & 1 \\ 0.7170 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0.6139 & 0 \end{bmatrix}.$$

For System D, the LPAS heuristic outperforms the other heuristics. It maps the tasks of each class to at most two machines, except for class 2 tasks that are mapped to four machines. The LPAS heuristic exhibits better performance than the KPB heuristic with $\overline{k} = 3$.

System E is a LoHi system. The system contains 7 machines and 4 classes. The arrival and execution rates are given by $\alpha =$

[10 10 8 8] and

$$\mu = \begin{bmatrix} 2.2 & 7 & 10.25 & 1 & 5.7 & 0.5 & 12 \\ 1.95 & 7.05 & 9.78 & 0.95 & 5.65 & 0.56 & 11.85 \\ 2 & 7.25 & 10.02 & 0.98 & 5.75 & 0.67 & 11.8 \\ 2.05 & 6.75 & 9.99 & 1.02 & 5.82 & 0.49 & 12.05 \end{bmatrix}.$$

Solving the allocation LP gives $\lambda^* = 1.0844$ and

$$\delta^* = \begin{bmatrix} 1 & 0 & 0.8433 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.9151 \\ 0 & 1 & 0.0754 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0.0813 & 1 & 1 & 0 & 0.0849 \end{bmatrix}.$$

The MCT heuristic has the best performance for System E. This is not an unexpected result as suggested by the following argument. Due to the very low task heterogeneity of system E, one can think of it as a system with one class of arriving tasks ($\alpha = [36]$) and the execution rate of each machine is the average of the execution rates of the four classes in System E on the machine, $\mu = [2.05\ 7.0125\ 10.01\ 0.9875\ 5.73\ 0.555\ 11.925]$. In this case, assigning an arriving task to the machine that has the minimum expected completion time (the MCT heuristic) is the best strategy. In fact, solving the allocation LP corresponding to the modified system above results in the following value for δ^*: [1 1 1 1 1 1 1]. Thus, in this case, the LPAS heuristic reduces to the MCT heuristic.

Even though the MCT heuristic is the best heuristic for System E, the LPAS heuristic has the advantage of mapping each class to a smaller number of machines. The LPAS heuristic performs much better than the KPB heuristic even for $\overline{k} = 5$. The KPB heuristic is unstable for $\overline{k} < 5$.

Systems $F1$ and $F2$ are HiLo systems ($M = 7$, $N = 4$). Both have the same execution rates and only differ in the arrival rate vector α. The arrival rate vector for System $F1$ is $\alpha = [4\ 8\ 10\ 10]$, and for System $F2$ it is given by $\alpha = [7\ 7\ 7\ 7]$. For both systems, the execution rate matrix is given by

$$\mu = \begin{bmatrix} 2 & 2.5 & 2.25 & 2 & 2.2 & 1.75 & 2.25 \\ 4.5 & 4 & 4.2 & 4 & 3.8 & 3.9 & 3.95 \\ 6 & 6.2 & 6.25 & 6 & 5.75 & 5.9 & 6.05 \\ 10 & 10.25 & 10.5 & 9.5 & 10.25 & 10.25 & 10 \end{bmatrix}.$$

For System $F1$, solving the allocation LP gives $\lambda^* = 1.1331$ and

$$\delta^* = \begin{bmatrix} 0 & 1 & 0 & 0 & 0.8946 & 0 & 0.0285 \\ 1 & 0 & 1 & 0.0911 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.9089 & 0 & 0 & 0.9715 \\ 0 & 0 & 0 & 0 & 0.1054 & 1 & 0 \end{bmatrix}.$$

For System $F2$, solving the allocation LP gives $\lambda^* = 1.0798$ and

$$\delta^* = \begin{bmatrix} 0 & 1 & 0.2704 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0.7282 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0014 & 1 & 0 & 0.2626 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.7374 & 0 \end{bmatrix}.$$

Due to the very low machine heterogeneity of both systems, one can think of them as consisting of identical machines. The LPAS heuristic achieves the best performance in many such systems as in $F2$.

System G is a LoLo system with $M = 7$ and $N = 4$. The arrival and execution rates are given by $\alpha = [8\ 9\ 7\ 10]$ and

$$\mu = \begin{bmatrix} 5 & 5.05 & 4.95 & 4.98 & 4.7 & 5.2 & 5.25 \\ 5.25 & 5.09 & 4.9 & 4.92 & 5 & 5.13 & 5.14 \\ 4.45 & 5 & 4.9 & 4.45 & 4.9 & 5 & 5.1 \\ 5.02 & 4.95 & 5 & 5.02 & 5.25 & 4.75 & 5 \end{bmatrix}.$$

Solving the allocation LP gives $\lambda^* = 1.0557$ and

$$\delta^* = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0.6182 \\ 1 & 0.8352 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1648 & 0.9426 & 0 & 0 & 0 & 0.3818 \\ 0 & 0 & 0.0574 & 1 & 1 & 0 & 0 \end{bmatrix}.$$

For System G, the MCT heuristic has slightly better performance than the other heuristics. The KPB heuristic ($\overline{k} = 4$) has performance close to that of the LPAS heuristic, however, the mapper is required to obtain the expected completion times from four machines at each task arrival as compared to at most three machines in the case of the LPAS heuristic.

5) Special Systems: Consider System H defined in Section II-B. As explained earlier, since the MCT heuristic does not have any foresight on task heterogeneity, it may assign an arriving task to a machine that minimizes the task's expected completion time, yet it is very poor executing the task's class. This results in significant performance degradation as shown in Table 1. The LPAS heuristic is the best heuristic for System H. The KPB heuristic performs poorly and is only stable for $\overline{k} \geq 5$. For $\overline{k} < 5$, instability results. For $\overline{k} \geq 5$, the system becomes stable, however the performance is poor.

6) Other Execution Time Distributions: To test the effect of execution time distribution on the performance of the mapping heuristics, all of the previous experiments were re-run with non-exponential execution time distributions. In particular, two distributions were used to study lower and higher variances than the exponential case: the first is a constant execution time of size $\frac{1}{\mu_{i,j}}$ for machine j executing class i tasks, and the second is a hyper-exponential distribution with mean $\frac{1}{\mu_{i,j}}$ for the execution times and twice the variance as the exponential case.

Our results indicate that the relative performance of the heuristics is not affected by the execution time distribution. System $I1$ has the same configuration as system $C1$, but with a hyper-exponential execution time distribution. System $I2$ also has the same configuration as system $C1$, but with constant execution times. Table 1 shows the performance of the different mapping heuristics for Systems $I1$ and $I2$. For the KPB heuristic, both systems are unstable for $\overline{k} < 14$.

IV. Other LPAS-Related Heuristics

A. Overview

In this section, we describe other LPAS-related heuristics which attempt to reduce further the state information required in making mapping decisions.

To compare mapping heuristics in terms of state information required for mapping, we use the discount metric defined in [6]. Let N_s be the average number of machines from which a heuristic acquires information for each arrival. The discount of the average required state information (over full information) for a mapping decision is then defined by

$$Discnt = \left(1 - \frac{N_s}{M}\right) \times 100\%. \quad (4)$$

Now, consider the heuristics introduced in Section II-B. For the MET heuristic, the mapper need not contact any machine in making mapping decisions and thus $N_s = 0$. For the MCT heuristic, assuming that all $\mu_{i,j}$ are positive, the mapper needs to acquire state information from all of the machines and thus $N_s = M$. For the KPB heuristic, the mapper needs to acquire state information from $\overline{k}$ machines (assuming that all $\mu_{i,j}$ are positive). For the LPAS heuristic, the average number of machines from which the mapper acquires information for each arrival is

$$N_s = \sum_{i=1}^{N} \frac{\alpha_i}{\tilde{\alpha}} |S_i|, \tag{5}$$

where $\tilde{\alpha} = \sum_{i=1}^{N} \alpha_i$.

1) The LPAS$-2/k$ Heuristic: One way to reduce further the state information required in making mapping decisions is to choose for each arrival of class i just two machines from the set S_i and then compare that pair in terms of the expected completion times. The LPAS$-2/k$ heuristic is stated as follows: A class i arrival is mapped to one of the two machines (j_1, j_2) chosen from S_i which has shorter expected completion time. If $|S_i| > 2$, the two machines j_1 and j_2 are chosen with probabilities $p_{j_1} = \frac{\delta^*_{i,j_1}\mu_{i,j_1}}{\lambda^*\alpha_i}$ and $p_{j_2} = \frac{\delta^*_{i,j_2}\mu_{i,j_2}}{\lambda^*\alpha_i - \delta^*_{i,j_1}\mu_{i,j_1}}$, respectively.

The average number of machines from which the LPAS$-2/k$ heuristic acquires state information for each arrival is given by

$$N_s = 2 \sum_{i:|S_i|>1} \frac{\alpha_i}{\tilde{\alpha}} + \sum_{i:|S_i|=1} \frac{\alpha_i}{\tilde{\alpha}}.$$

It is noted that, in the worst case, the LPAS$-2/k$ heuristic acquires state information from two machines for each arrival, and thus the discount of the average required state information is $(M-2)/M \times 100\%$. This implies that the discount increases as the number of machines grows, independently of the structure of δ^*. Note, however, that even though the LPAS$-2/k$ heuristic requires less state information than the LPAS heuristic, one needs to know the values for $\delta^*_{i,j}$ (rather than just whether they are zero or not).

Consider a system with $N = 1$ and M arbitrary. Assume that the execution times are exponentially distributed and $\mu_{1,j} = 1$ for all $j = 1, \ldots, M$. Also, assume that the arrival process is Poisson with rate $M\alpha_1$, where $\alpha_1 < 1$. In this case, $\delta^*_{1,j} = 1$ for all $j = 1, \ldots, M$. Thus, using the LPAS$-2/k$ heuristic, an arrival randomly (with equal probabilities) chooses two of the machines and joins the queue of the machine with the shorter queue length. Mitzenmacher [15] analyzed such a system and found that when α_1 approaches 1, there is an exponential improvement in the mean waiting time (over choosing only one machine randomly), while increasing the number of choices for an arrival results in only a constant improvement over two choices. This suggests that a similar degree of improvement might be expected for the LPAS$-2/k$ heuristic over a static mapping heuristic, although the "power of two choices" has not been analyzed rigorously for heterogeneous systems [6].

2) The LP–Static Heuristic: The LP–Static heuristic requires no state information in making mapping decisions. We define it here to compare against other heuristics which take into account state information. The heuristic is stated as follows. Class i tasks are mapped to machine j with probability

$$p_{i,j} = \frac{\delta^*_{i,j}\mu_{i,j}}{\lambda^*\alpha_i}. \tag{6}$$

The LP–Static heuristic maximizes system capacity in the long term, but may suffer from poor performance since it does not do any short-term shifting of workload among the machines. In Appendix A, it is proven that the LP–Static heuristic is guaranteed to stabilize a stabilizable system. However, the heuristic generally achieves poor performance. In Section V, we introduce the Guided-LPAS heuristic and prove that it is guaranteed to stabilize a stabilizable system. The Guided-LPAS heuristic achieves competitive performance levels with the LPAS heuristic.

B. Simulation Results

We use System $C2$ which models a real cluster system [7] (for details, see He [16]) to compare the LPAS-related heuristics. System $C2$ is a medium size system with 5 task classes and 30 machines. The machines are partitioned into 6 groups, machines within a group are identical. Groups T, U, V, W, X, and Y, consist of 2 machines, 6 machines, 7 machines, 7 machines, 4 machines, and 4 machines, respectively. The execution rates are shown in Table 4. The arrival rate vector is $\alpha = [204.10\ 68.87\ 77.63\ 5.01\ 10.43]$.

Table 4. Execution rates for System $C2$

Task	Group T	U	V	W	X	Y
1	16.7	24.8	24.2	29	25.6	48.3
2	30.4	48.3	77.7	83.6	135.9	144.9
3	18.9	24.2	48.3	45.8	72.5	72.5
4	3	3	7.6	7.6	8.3	8.7
5	1	1.1	3	2.9	3	3

As done for System $C1$ (Section III-B.3), we solve the allocation LP corresponding to the following system:

$$N = 5, M = 6, \alpha = [204.10\ 68.87\ 77.63\ 5.01\ 10.43], \text{ and}$$

$$\mu = \begin{bmatrix} 33.4 & 148.8 & 169.4 & 203 & 102.4 & 193.2 \\ 60.8 & 289.8 & 543.9 & 585.2 & 543.6 & 579.6 \\ 37.8 & 145.2 & 338.1 & 320.6 & 290 & 290 \\ 6 & 18 & 53.2 & 53.2 & 33.2 & 34.8 \\ 2 & 6.6 & 21 & 20.3 & 12 & 12 \end{bmatrix}.$$

Solving the modified allocation LP gives $\lambda^* = 2.4242$ and

$$\delta^* = \begin{bmatrix} 1 & 1 & 0 & 0.5881 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0.3071 & 0 \\ 0 & 0 & 0 & 0 & 0.6489 & 0 \\ 0 & 0 & 0 & 0.2009 & 0.0439 & 0 \\ 0 & 0 & 1 & 0.2111 & 0 & 0 \end{bmatrix}.$$

Thus, for System $C2$, we use the δ^* matrix in Table 5. In this particular solution, machines that belong to the same group have identical values for $\delta^*_{i,j}$. Note that the number of nonzero elements in the δ^* matrix is 52. Using the LPAS heuristic, the discount of the average required state information for a mapping decision is 58%. On the other hand, using the LPAS–$2/k$ heuristic the discount is 94%.

Table 5. The machine allocation matrix for System $C2$

Task	Group T	U	V	W	X	Y
1	1	1	0	0.5881	0	1
2	0	0	0	0	0.3071	0
3	0	0	0	0	0.6489	0
4	0	0	0	0.2009	0.0439	0
5	0	0	1	0.2111	0	0

Table 6 shows the simulation results for System $C2$. As the table shows, the LPAS heuristic achieves the best results. The LPAS$-2/k$ heuristic has worse performance than that achieved by the LPAS heuristic, yet it uses less state information. The performance degradation is not large (significantly better than the LP–Static heuristic).

Table 6. Simulation Results for System $C2$

LP–Static	$(24.25, 24.29)$
MCT	$(11.45, 11.46)$
LPAS	$(11.32, 11.33)$
LPAS$-2/k$	$(14.01, 14.02)$

V. The Guided-LPAS Heuristic

Consider the MCT heuristic. Stolyar [17] showed that it does not minimize system workload in heavy traffic. Sharifnia [12] showed that it may not stabilize the system even if the system can be stabilized. He attributed this to its greedy use of information resulting in assigning tasks to the "wrong" machines persistently and thus causing instability. An important question is: with the restrictions of the LPAS heuristic, is it true that the LPAS heuristic is guaranteed to stabilize a stabilizable system (*i.e.*, a system where the solution to the allocation LP is $\lambda^* > 1$)?

Even though our simulation experiments have failed to find a stabilizable system that is not stabilized by the LPAS heuristic, we are not able to prove the stability of the LPAS heuristic. This is because of the difficulty of finding an expression for the actual machine allocations achieved by the LPAS heuristic. However, we are confident of its stability as it avoids assigning tasks to the "wrong" machines by using task heterogeneity to provide foresight. Thus, it does not suffer from the root cause for the instability of the MCT heuristic. If one is still concerned about stability, we give the Guided-LPAS heuristic and give a proof for its stability.

The Guided-LPAS heuristic is guaranteed to stabilize a stabilizable system. It is a modification of the LPAS heuristic such that, over time, target (reference) execution capacities allocated for individual task classes on each machine are achieved. These targets are found from the solution of the allocation LP. In particular, the target execution capacity allocated by machine j for class i is $\frac{\delta^*_{i,j}\mu_{i,j}}{\lambda^*}$.

Let $\pi_{i,j}$ be the target mapping ratio of class i tasks to machine j such that the target execution capacity reference levels are achieved (*i.e.*, $\pi_{i,j} = \frac{\delta^*_{i,j}\mu_{i,j}}{\lambda^*\alpha_i}$). The Guided-LPAS heuristic uses the LPAS heuristic as long as the actual rate at which class i tasks are mapped to machine j is not too far from its target level $\pi_{i,j}$, or equivalently, the actual execution capacity levels are not far from their targets.

The Guided-LPAS heuristic can be stated as follows. Let $\alpha_{i,j}(t)$ denote the number of class i tasks assigned to machine j in $[0, t]$. Let $\alpha_i(t)$ denote the number of class i tasks that arrived during $[0, t]$. An arrival of a class i task at time t is mapped to a machine j for which: (i) the task's expected completion time is minimized, (ii) $\delta^*_{i,j} \neq 0$, and (iii) $\alpha_{i,j}(t^-) < \pi_{i,j}\alpha_i(t) + C_{i,j}\sqrt{t}$, where $C_{i,j}$ is a nonnegative but otherwise arbitrary constant. Note that since at any class i arrival time t, $\alpha_{i,j}(t^-) < \alpha_i(t)$ and $C_{i,j} \geq 0$, $j = 1, \ldots, M$, there is always at least one machine satisfying condition (iii), and therefore the heuristic is well defined.

The Guided-LPAS heuristic attempts to achieve the short-term advantages attained by the LPAS heuristic. However, it employs an oversight control that achieves target execution capacity reference levels in the long run. This ensures the stability of the heuristic while achieving good performance levels. The stability result for the Guided-LPAS heuristic is stated in the following theorem (the proof is provided in Appendix A):

Theorem 3: The Guided-LPAS heuristic stabilizes a stabilizable system. More specifically, if the system is stabilizable and the mapper uses the Guided-LPAS heuristic, then the workload process converges to a steady-state distribution as $t \to \infty$.

Our simulation experiments indicate that the oversight control mechanism is seldom used. For instance, consider the system defined in Section II-B to show the instability of the MCT heuristic. Setting $C_{i,j} = 1$, $i = 1, \ldots, N$, $j = 1, \ldots, M$, and simulating the system under the Guided-LPAS heuristic, we observe that the number of times condition (iii) is violated is zero.

In Section IV, we introduced a variant of the LPAS heuristic which results in a further reduction of the state information required for mapping. We referred to this heuristic as the LPAS$-2/k$ heuristic. Here, we modify the LPAS$-2/k$ heuristic such that stability is guaranteed for stabilizable systems. The resulting heuristic is referred to as the Guided-LPAS$-2/k$ heuristic and is defined as follows. Let $T_i(t) = \{j|\ \delta^*_{i,j} \neq 0$ and $\alpha_{i,j}(t^-) < \pi_{i,j}\alpha_i(t) + C_{i,j}\sqrt{t}\}$. A class i arrival at time t is mapped to one of the two machines (j_1, j_2) chosen from $T_i(t)$ such that the arrival joins the machine with the minimum expected completion time. If $|T_i(t)| > 2$, the two machines j_1 and j_2 are chosen with probabilities $p_{j_1} = \frac{\delta^*_{i,j_1}\mu_{i,j_1}}{\sum_{j\in T_i(t)}\delta^*_{i,j}\mu_{i,j}}$ and $p_{j_2} = \frac{\delta^*_{i,j_2}\mu_{i,j_2}}{(\sum_{j\in T_i(t)}\delta^*_{i,j}\mu_{i,j})-\delta^*_{i,j_1}\mu_{i,j_1}}$, respectively.

The following theorem states the stability result for the Guided-LPAS$-2/k$ heuristic (the proof is provided in Appendix A):

Theorem 4: The Guided-LPAS$-2/k$ heuristic stabilizes a stabilizable system. More specifically, if the system is stabilizable and the mapper uses the Guided-LPAS$-2/k$ heuristic, then the workload process converges to a steady-state distribution as $t \to \infty$.

VI. Related Work

The problem of mapping tasks onto machines in HC systems is an extremely active field (for example, see Braun *et al.* [18] and [3]). In the literature, several authors refer to the mapping of tasks onto machines as scheduling.

Several mapping heuristics are described and compared in [8]. The model assumptions in [8] and our assumptions for the HC system are identical. However, the authors in [8] do not group tasks into classes and they assume that the expected execution time of every arriving task is known on each machine. This can be unrealistic in typical HC systems. On the other hand, we assume that the tasks are grouped into classes and only the arrival rates of each class's tasks and the execution rates of each machine for each class are known. This assumption is made in several models of cluster and grid environments (such as Franke *et al.* [19] and [7]).

In [7], the performance of several scheduling algorithms is examined on a real-world workload. One of these algorithms is similar to the MCT (Minimum Completion Time) algorithm [8]. Another algorithm is a variation on the MCT algorithm that attempts to minimize completion time while taking affinity effects

into account. Experimental results show that varying the MCT algorithm to take affinity effects into account exhibits improved performance over the MCT algorithm [7].

Several dynamic mapping heuristics are proposed and compared in [3] for HC systems in which tasks have priorities and multiple soft deadlines. These heuristics are batch mode heuristics, as opposed to the immediate mode heuristics considered here. Immediate mode heuristics map an arriving task as soon as it arrives, whereas batch mode heuristics consider a subset of tasks for mapping. The workload model in [3] is identical to our workload model with the addition of priorities and deadlines associated with tasks.

Ansell *et al.* [20] develop a class of policies for systems having the same workload model. Such policies are based on the application of a policy improvement step to an optimal static policy. However, these policies are computationally intensive and may not scale well. In fact, only a small system with $N = 2$ and $M = 2$ is considered in their numerical study. Another limitation is that there is no attempt to reduce the amount of state information required in mapping. Thus, the mapper needs to obtain full state information at every mapping event.

Another heuristic is suggested in Glazebrook *et al.* [21]. The heuristic is applicable to systems having an identical workload model to the model considered here. However, the mean execution time of a task depends only on the machine (*i.e.*, for a machine j, $\mu_{i,j} = \mu_j, \ \forall i \in I$). Furthermore, machines may not be permanently available for service. Similar to [20], such a heuristic computes an index for each machine at every state of the system and thus may not scale to large systems.

Our model for an HC system has been studied in the context of queueing analysis. The MCT heuristic is a variation on the MinDrift rule which is shown to perform well in heavy traffic scenarios (see [17]). Wasserman *et al.* [22] introduce a processor allocation policy which corresponds to the MCT heuristic.

VII. Conclusion

The main contribution of the paper is the proposal of the LPAS mapping heuristic for heterogeneous computing systems. The LPAS heuristic utilizes the solution to an allocation LP in making mapping decisions. By solving an allocation LP, the LPAS heuristic provides an explicit method to compute the maximum capacity and to compute the allocation of machines to classes. This has the advantage of requiring dramatically less state information while at the same time achieving competitive performance levels. It does not suffer from the limitations of other mapping heuristics, namely the limited use of information about task heterogeneity (as in the case of the MCT heuristic) and the ad-hoc manner for setting parameters (as in the KPB heuristic). Furthermore, we have introduced two modifications to the LPAS heuristic. First, the LPAS$-2/k$ heuristic significantly reduces the state information required in mapping. Second, the Guided-LPAS heuristic is guaranteed to stabilize a stabilizable system.

We note that there has been little work done in characterizing actual HC system workloads. Hence, we believe that there is a need to develop a benchmark framework which characterizes actual workloads and can be used to compare the different heuristics in terms of several performance metrics (for example, see the work of Li *et al.* [23]). Also, the main issue addressed in the paper is dealing with heterogeneity of HC systems within the context of resource management. This work does not include other factors such as communication delay, data transfer costs, heterogeneous network bandwidths, and network topologies. We plan to incorporate some of these as part of our future work, however we believe that the basic framework presented here can be adapted to such settings.

A related open question is to analyze the robustness of the LPAS heuristic. Often, HC systems operate in an environment with a large degree of uncertainty (see Smith *et al.* [24]). In this context, robustness can be defined as the degree to which a system can function correctly in the presence of parameter values different from those assumed (Ali *et al.* [25]). A number of papers have studied robustness in HC systems, including [25], Mehta *et al.* [26], Shestak *et al.* [27], and [24]. We believe that the solution to the allocation LP is inherently robust and thus we expect the LPAS heuristic to have robustness advantages over other existing heuristics for HC systems.

Appendix A

Here, we apply the fluid limit methodology in proving several stability results. Our analysis will involve a formal limiting fluid model for the system. This is done by describing the system as a Markov process and performing a scaling in time and space that allows the use of law of large numbers results, leading to a deterministic model where the flow through the system is continuous (fluid) rather than discrete (tasks). The use of fluid model techniques for characterizing stability is a well established methodology: see, for example, the work of Chen [28], Chen and Yao [29], Dai [30], [31], and Dai and Meyn [32].

First, we define the system dynamics of the queueing network corresponding to our workload model. Class i tasks arrive via an arrival process with independent and identically distributed (i.i.d.) interarrival times $\{\xi_i(n)\}$ where $\alpha_i = 1/E[\xi_i(1)]$. Also, let $\eta_{i,j}(n)$ denote the execution time for the nth class i task executed at machine j, where $\mu_{i,j} = 1/E[\eta_{i,j}(1)]$ if machine j can execute class i tasks, and $\mu_{i,j} = 0$ otherwise. We assume that the sequence $\{\eta_{i,j}(n)\}$ is i.i.d. for each i and j. Let $A_i(t)$ be the residual interarrival time for class i tasks at time t. Then, the variable $E_i(t)$ is the number of class i tasks that arrive in $(0, t]$ and

$$E_i(t) = \max\{n \geq 0 : A_i(0) + \xi_i(1) + \cdots + \xi_i(n-1) \leq t\},$$

where the maximum of the empty set is defined to be zero.

Let $T_{i,j}(t)$ be the fraction of time that machine j spends executing class i tasks in $(0, t]$. Note that the functions $T_{i,j}(t)$ are determined by the mapping heuristic and the scheduling policy of machine j. Let $T_j(t) = \sum_{i=1}^{N} T_{i,j}(t)$ represent the total allocation of machine j (*i.e.*, the fraction of time it is busy).

Define $W_{i,j}(t)$ as the cumulative amount of time that it takes machine j to execute class i tasks present in its queue at time t. Thus, $W_{i,j}(t)$ represents the class i workload of machine j at time t. We also define $Q_{i,j}(t)$ as the total queue length of class i tasks at machine j at time t. Let $\alpha_{i,j}(t)$ be the total number of class i tasks assigned by the mapping heuristic to machine j in $(0, t]$. With the definitions above, we are now able to give an expression for the evolution of $W_{i,j}(t)$, $i = 1, \ldots, N, j = 1, \ldots, M$,

$$W_{i,j}(t) = \sum_{n=1}^{Q_{i,j}(0)+\alpha_{i,j}(t)} \eta_{i,j}(n) - T_{i,j}(t)$$

Second, we construct a Markov process X for the system. For any of the mapping heuristics discussed in this paper,

$$X(t) := (W_{i,j}(t), A_i(t), Y_j(t) : i = 1, \ldots, N, j = 1, \ldots, M)$$

is a Markovian state evolving on

$$\mathbb{R}_+^{N\times M} \times \mathbb{R}_+^{N} \times \mathbb{R}_+^{N\times M}.$$

The process X may be shown to have the strong Markov property.

Let $w = W_{i,j}(0)$. Suppose that the function $(\overline{W}_{i,j}(t), \overline{T}_{i,j}(t) : i = 1,\ldots,N, j = 1,\ldots,M)$ is a limit point of the functions $(w^{-1}W_{i,j}(wt), w^{-1}T_{i,j}(wt) : i = 1,\ldots,N, j = 1,\ldots,M)$ when $w \to \infty$. We call $(\overline{W}_{i,j}(t), \overline{T}_{i,j}(t) : i = 1,\ldots,N, j = 1,\ldots,M)$ a fluid limit of the system.

We are now ready to describe the fluid model corresponding to our workload model. Let $(\overline{W}_{i,j}(t), \overline{T}_{i,j}(t) : i = 1,\ldots,N$ and $j = 1,\ldots,M)$ be a fluid limit for the system. Define $\alpha_{i,j}$ as $\lim_{t\to\infty} \frac{\alpha_{i,j}(t)}{t}$, $i = 1,\ldots,N, j = 1,\ldots,M$, assuming the limit exists (for the mapping heuristics we are concerned with, $\alpha_{i,j}$ does exist). For any mapping heuristic, every fluid limit satisfies the following set of conditions (for all $i = 1,\ldots,N$ and $j = 1,\ldots,M$):

$$\overline{W}_{i,j}(t) = \overline{W}_{i,j}(0) + \frac{\alpha_{i,j}t}{\mu_{i,j}} - \overline{T}_{i,j}(t); \tag{7}$$

$$\overline{W}_{i,j}(t) \geq 0; \tag{8}$$

$$\overline{T}_{i,j}(0) = 0 \text{ and } \overline{T}_{i,j}(\cdot) \text{ is nondecreasing}; \tag{9}$$

$$0 \leq \sum_{i=1}^{N} \frac{d}{dt}\overline{T}_{i,j}(t) \leq 1. \tag{10}$$

The derivatives above exist almost everywhere, as $\overline{T}_{i,j}(t)$ is Lipschitz for all i, j. From this point on, derivatives will be understood to be taken on the condition that they exist.

The conditions (7)-(10) do not completely specify the fluid limits, and there are other conditions on $\overline{T}_{i,j}(t)$. The complete set of conditions is known as the fluid model (see Theorem 2.3.2 of [30]). A fluid solution refers to any solution to the fluid model equations.

The fluid model is said to be stable if there exists a fixed time $t' > 0$ such that $\overline{W}_{i,j}(t) = 0$, $t > t'$, $i = 1,\ldots,N, j = 1,\ldots,M$, for any fluid solution. The fluid model is said to be (weakly) unstable if there exists a $t' > 0$ such that for every solution of the fluid model with $\sum_{i=1}^{N}\sum_{j=1}^{M}\overline{W}_{i,j}(0) = 0$, $\sum_{i=1}^{N}\sum_{j=1}^{M}\overline{W}_{i,j}(t') \neq 0$. Analyzing the stability region of the deterministic fluid model defined above allows us to characterize the maximum capacity of the actual system.

Proof: [Theorem 1]

Consider the LP–Static heuristic. If $\lambda^* > 1$, we show that the LP–Static heuristic is guaranteed to stabilize the system. The LP–Static heuristic randomly maps tasks to machines according to probabilities $p_{i,j} = \frac{\delta^*_{i,j}\mu_{i,j}}{\lambda^*\alpha_i}$, $i = 1,\ldots,N, j = 1,\ldots,M$.

Let $W_j(t)$ denote the total workload at machine j at time t. Define $\overline{W}_j(t)$ as a limit point of the function $w^{-1}W_j(wt)$ as $w \to \infty$, $j = 1,\ldots,M$. Then,

$$\overline{W}_j(t) = \sum_{i=1}^{N}\overline{W}_{i,j}(t).$$

Note that if $\overline{W}_j(t) > 0$, then it must be true that $\frac{d}{dt}\overline{T}_j(t) = 1$. Hence, if $\overline{W}_j(t) > 0$, then

$$\begin{aligned}\frac{d}{dt}\overline{W}_j(t) &= \frac{\alpha_{1,j}}{\mu_{1,j}} + \cdots + \frac{\alpha_{N,j}}{\mu_{N,j}} - \frac{d}{dt}\overline{T}_{1,j}(t) - \cdots - \frac{d}{dt}\overline{T}_{N,j}(t)\\ &= \frac{\alpha_{1,j}}{\mu_{1,j}} + \cdots + \frac{\alpha_{N,j}}{\mu_{N,j}} - 1.\end{aligned}$$

Since using the LP–Static heuristic as a mapping heuristic results in $\alpha_{i,j} = \alpha_i p_{i,j} = \frac{\delta^*_{i,j}\mu_{i,j}}{\lambda^*}$, $i = 1,\ldots,N, j = 1,\ldots,M$, it must be true that

$$\begin{aligned}\frac{d}{dt}\overline{W}_j(t) &= \frac{\delta^*_{1,j}}{\lambda^*} + \cdots + \frac{\delta^*_{N,j}}{\lambda^*} - 1\\ &= \frac{\sum_{j=1}^{M}\delta^*_{i,j}}{\lambda^*} - 1\\ &< 0 \quad \text{since } \sum_{j=1}^{M}\delta^*_{i,j} \leq 1 \text{ and } \lambda^* > 1.\end{aligned}$$

Thus, if $\overline{W}_j(t) > 0$, then $\frac{d}{dt}\overline{W}_j(t) < 0$ which implies that there exists a fixed time $t' > 0$ such that $\overline{W}_j(t) = 0$, and hence $\overline{W}_{i,j}(t) = 0$, $i = 1,\ldots,N, j = 1,\ldots,M$, for all $t > t'$. Hence, the fluid model is stable and the result follows from Theorem 4.2 in [31]. ∎

Proof: [Theorem 2]

Assume that the system can be stabilized. Hence, the corresponding fluid model is stable *i.e.,* there exists a fixed time $t' > 0$ such that $\overline{W}_{i,j}(t) = 0$, $t > t'$, $i = 1,\ldots,N, j = 1,\ldots,M$, for any fluid solution. Choose $s > t'$. Then, $\frac{d}{dt}\overline{W}_{i,j}(s) = 0$, $i = 1,\ldots,N, j = 1,\ldots,M$. Also, let $\frac{d}{dt}\overline{T}_{i,j}(s) = \delta_{i,j}$, $i = 1,\ldots,N, j = 1,\ldots,M$. Condition (7) implies (after taking the derivative of both terms and substituting s for t),

$$\frac{\alpha_{i,j}}{\mu_{i,j}} - \delta_{i,j} = 0, \qquad \text{for all } i = 1,\ldots,N, j = 1,\ldots,M.$$

Thus,

$$0 = \alpha_{i,j} - \delta_{i,j}\mu_{i,j}, \qquad \text{for all } i = 1,\ldots,N, j = 1,\ldots,M.$$

Summing over j,

$$0 = \alpha_i - \sum_{j=1}^{M}\delta_{i,j}\mu_{i,j}, \qquad \text{for all } i = 1,\ldots,N.$$

Thus, the following constraints hold ((12) and (13) follow from (10)):

$$\sum_{j=1}^{M}\delta_{i,j}\mu_{i,j} \geq \alpha_i, \qquad \text{for all } i = 1,\ldots,N, \tag{11}$$

$$\sum_{i=1}^{N}\delta_{i,j} \leq 1, \qquad \text{for all } j = 1,\ldots,M, \tag{12}$$

$$\delta_{i,j} \geq 0, \qquad \text{for all } i = 1,\ldots,N, \text{ and } j = 1,\ldots,M. \tag{13}$$

Thus, (11)-(13) provide a feasible solution for the allocation LP (1)-(3) with $\lambda^* = 1$ contradicting the assumption that $\lambda^* < 1$. Hence, the fluid model is weakly unstable and by Theorem 2.5.1 of [30], the system can not be stabilized. ∎

Proof: [Theorem 3]

Using the Guided-LPAS heuristic (introduced in Section V), we can show that

$$\pi_{i,j}\alpha_i(t) - M + 1 - \sum_{j'\neq j} C_{i,j'}\sqrt{t} \leq \alpha_{i,j}(t) < \pi_{i,j}\alpha_i(t) + C_{i,j}\sqrt{t} + 1. \tag{14}$$

First, let us show that

$$\alpha_{i,j}(t) < \pi_{i,j}\alpha_i(t) + C_{i,j}\sqrt{t} + 1. \quad (15)$$

Assume that the Guided-LPAS heuristic maps an arrival of class i at time t to machine j. Since the arriving task was mapped onto machine j, it must be true from the definition of the Guided-LPAS heuristic that $\alpha_{i,j}(t^-) < \pi_{i,j}\alpha_i(t) + C_{i,j}\sqrt{t}$. It then follows that $\alpha_{i,j}(t) = \alpha_{i,j}(t^-) + 1 < \pi_{i,j}\alpha_i(t) + C_{i,j}\sqrt{t} + 1$, proving (15).

Second, we show

$$\pi_{i,j}\alpha_i(t) - M + 1 - \sum_{j' \neq j} C_{i,j'}\sqrt{t} \leq \alpha_{i,j}(t). \quad (16)$$

Consider a machine j. Clearly it is assigned the following number of class i tasks:

$$\alpha_{i,j}(t) = \alpha_i(t) - \sum_{j' \neq j} \alpha_{i,j'}(t).$$

where $j' \in \{1, \ldots, M\}$. Using the Guided-LPAS heuristic, (15) holds and it follows that

$$\begin{aligned}
\alpha_{i,j}(t) &= \alpha_i(t) - \sum_{j' \neq j} \alpha_{i,j'}(t) \\
&\geq \alpha_i(t) - \sum_{j' \neq j} (\pi_{i,j'}\alpha_i(t) + C_{i,j'}\sqrt{t} + 1) \\
&= \alpha_i(t) - \sum_{j' \neq j} \pi_{i,j'}\alpha_i(t) - \sum_{j' \neq j} C_{i,j'}\sqrt{t} - (M-1) \\
&= \pi_{i,j}\alpha_i(t) - M + 1 - \sum_{j' \neq j} C_{i,j'}\sqrt{t}.
\end{aligned}$$

This proves (16).

From (14), it follows that $\alpha_{i,j} = \alpha_i \pi_{i,j} = \frac{\delta^*_{i,j}\mu_{i,j}}{\lambda^*}$, $i = 1, \ldots, N, j = 1, \ldots, M$. Thus, the results follow as before (see the proof of Theorem 1). ■

Proof: [Theorem 4]

The proof that the Guided-LPAS$-2/k$ stabilizes a stabilizable system is similar to the proof given above for the Guided-LPAS heuristic. The Guided-LPAS$-2/k$ satisfies (14) and the results follow. ■

ACKNOWLEDGMENT

The first author is supported by an Ontario Graduate Scholarship in Science and Technology. This research is also supported by the Natural Sciences and Engineering Research Council of Canada. The authors wish to thank the anonymous reviewers for their helpful comments.

PLACE PHOTO HERE

Issam Al-Azzoni received his M. A. Sc. in Software Engineering from McMaster University. He is currently pursuing his Ph.D. degree in Software Engineering at McMaster University. His research interests include queueing networks, scheduling of parallel and distributed systems, and heterogeneous computing environments. His email address is *alazzoi@mcmaster.ca*.

PLACE PHOTO HERE

Douglas G. Down is an Associate Professor in the Department of Computing and Software at McMaster University. His email address is *downd@mcmaster.ca*.

REFERENCES

[1] T. Sterling, E. Lusk, and W. Gropp, Eds., *Beowulf Cluster Computing with Linux*. Cambridge, MA, USA: MIT Press, 2003.

[2] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the Grid: Enabling scalable virtual organizations," *International Journal of High Performance Computing Applications*, vol. 15, no. 3, pp. 200–222, 2001.

[3] J.-K. Kim, S. Shivle, H. J. Siegel, A. A. Maciejewski, T. D. Braun, M. Schneider, S. Tideman, R. Chitta, R. B. Dilmaghani, R. Joshi, A. Kaul, A. Sharma, S. Sripada, P. Vangari, and S. S. Yellampalli, "Dynamically mapping tasks with priorities and multiple deadlines in a heterogeneous environment," *Journal of Parallel and Distributed Computing*, vol. 67, no. 2, pp. 154–169, 2007.

[4] M. Mitzenmacher, "How useful is old information?" *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 1, pp. 6–20, 2000.

[5] I. Al-Azzoni and D. Down, "Linear programming based affinity scheduling for heterogeneous computing systems," in *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, 2007, pp. 105–111.

[6] Y.-T. He, I. Al-Azzoni, and D. Down, "MARO - MinDrift affinity routing for resource management in heterogeneous computing systems," in *Proceedings of the Conference of the Centre for Advanced Studies on Collaborative Research*, 2007, pp. 71–85.

[7] L. Kontothanassis and D. Goddeau, "Profile driven scheduling for a heterogeneous server cluster," in *Proceedings of the 34th International Conference on Parallel Processing Workshops*, 2005, pp. 336–345.

[8] 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*, 1999, pp. 30–44.

[9] D. Arnold, S. Agrawal, S. Blackford, J. Dongarra, M. Miller, K. Seymour, K. Sagi, Z. Shi, and S. Vadhiyar, "Users' Guide to NetSolve V1.4.1," University of Tennessee, Knoxville, TN, Innovative Computing Dept. Technical Report ICL-UT-02-05, June 2002.

[10] R. Freund, M. Gherrity, S. Ambrosius, M. Campbell, M. Halderman, D. Hensgen, E. Keith, T. Kidd, M. Kussow, J. D. Lima, F. Mirabile, L. Moore, B. Rust, and H. J. Siegel, "Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet," in *Proceedings of the 7th Heterogeneous Computing Workshop*, 1998, pp. 184–199.

[11] R. Freund, T. Kidd, and L. Moore, "SmartNet: a scheduling framework for heterogeneous computing," in *Proceedings of the 2nd International Symposium on Parallel Architectures, Algorithms and Networks*, 1996, pp. 514–521.

[12] A. Sharifnia, "Instability of the join-the-shortest-queue and FCFS policies in queuing systems and their stabilization," *Operations Research*, vol. 45, no. 2, pp. 309–314, 1997.

[13] S. Andradóttir, H. Ayhan, and D. G. Down, "Dynamic server allocation for queueing networks with flexible servers," *Operations Research*, vol. 51, no. 6, pp. 952–968, 2003.

[14] R. Armstrong, "Investigation of effect of different run-time distributions on SmartNet performance," Master's thesis, Naval Postgraduate School, 1997.

[15] M. Mitzenmacher, "The power of two choices in randomized load balancing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 10, pp. 1094–1104, 2001.

[16] Y.-T. He, "Exploiting limited customer choice and server flexibility," Ph.D. dissertation, McMaster University, 2007.

[17] A. Stolyar, "Optimal routing in output-queued flexible server systems," *Probability in the Engineering and Information Sciences*, vol. 19, no. 2, pp. 141–189, 2005.

[18] T. D. Braun, H. J. Siegel, and A. A. Maciejewski, "Heterogeneous computing: Goals, methods, and open problems," in *Proceedings of the 8th International Conference on High Performance Computing*, 2001, pp. 307–320.

[19] H. Franke, J. Jann, J. E. Moreira, P. Pattnaik, and M. A. Jette, "An evaluation of parallel job scheduling for ASCI Blue-Pacific," in *Proceedings of the ACM/IEEE Conference on Supercomputing*, 1999, pp. 11–18.

[20] P. S. Ansell, K. D. Glazebrook, and C. Kirkbride, "Generalised 'join the shortest queue' policies for the dynamic routing of jobs to multiclass queues," *Journal of the Operational Research Society*, vol. 54, pp. 379–389, 2003.

[21] K. D. Glazebrook and C. Kirkbride, "Dynamic routing to heterogeneous collections of unreliable servers," *Queueing Systems: Theory and Applications*, vol. 55, no. 1, pp. 9–25, 2007.

[22] K. Wasserman, G. Michailidis, and N. Bambos, "Optimal processor allocation to differentiated job flows," *Performance Evaluation*, vol. 63, no. 1, pp. 1–14, 2006.

[23] H. Li, D. Groep, and L. Wolters, "Workload characteristics of a multi-cluster supercomputer," in *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn, Eds. Springer Verlag, 2004, pp. 176–193, lect. Notes Comput. Sci. vol. 3277.

[24] J. Smith, L. Briceno, A. A. Maciejewski, H. J. Siegel, T. Renner, V. Shestak, J. Ladd, A. Sutton, D. Janovy, S. Govindasamy, A. Alqudah, R. Dewri, and P. Prakash, "Measuring the robustness of resource allocations in a stochastic dynamic environment," in *Proceedings of the International Parallel and Distributed Processing Symposium*, 2007.

[25] S. Ali, A. A. Maciejewski, H. J. Siegel, and J.-K. Kim, "Measuring the robustness of a resource allocation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 7, pp. 630–641, 2004.

[26] A. M. Mehta, J. Smith, H. J. Siegel, A. A. Maciejewski, A. Jayaseelan, and B. Ye, "Dynamic resource allocation heuristics that manage tradeoff between makespan and robustness," *The Journal of Supercomputing*, vol. 42, no. 1, pp. 33–58, 2007.

[27] V. Shestak, J. Smith, H. J. Siegel, and A. A. Maciejewski, "A stochastic approach to measuring the robustness of resource allocations in distributed systems," in *Proceedings of the International Conference on Parallel Processing*, 2006, pp. 459–470.

[28] H. Chen, "Fluid approximations and stability of multiclass queueing networks: Work-conserving disciplines," *Annals of Applied Probability*, vol. 5, pp. 637–655, 1995.

[29] H. Chen and D. Yao, *Fundamentals of Queueing Networks: Performance, Asymptotics and Optimization*. Springer-Verlag, 2001.

[30] J. G. Dai, *Stability of Fluid and Stochastic Processing Networks*, publication No. 9, 1999. Centre for Mathematical Physics and Stochastics. http://www.maphysto.dk/.

[31] ——, "On positive Harris recurrence of multiclass queueing networks: a unified approach via fluid limit models," *Annals of Applied Probability*, vol. 5, pp. 49–77, 1995.

[32] J. G. Dai and S. Meyn, "Stability and convergence of moments for multiclass queueing networks via fluid limit models," *IEEE Transactions on Automatic Control*, vol. 40, pp. 1889–1904, 1995.