Utilization-Based Heuristics for Statically Mapping Real-Time Applications onto the HiPer-D Heterogeneous Computing System

Shoukat Ali[†], Jong-Kook Kim[†], Yang Yu[*], Shriram B. Gundala[*], Sethavidh Gertphol[*], Howard Jay Siegel[‡§], Anthony A. Maciejewski[‡], and Viktor Prasanna[*]

[†]Purdue University
School of Electrical and Computer Engineering
West Lafayette, IN 47907-1285 USA
{alis, kim42}@ecn.purdue.edu

[*]University of Southern California
Department of Electrical Engineering
Los Angeles, CA 90089-2560 USA
{yangyu, gundala, gertphol, prasanna}@halcyon.usc.edu

Colorado State University
[‡]Department of Electrical and Computer Engineering
[§]Department of Computer Science
Fort Collins, CO 80523-1373 USA
{hj, aam}@colostate.edu

Abstract

Real-time applications continue to increase in importance as they are employed in various critical areas, such as command and control systems. These applications have traditionally required custom-made systems to execute them. Recently, with the widespread use of increasingly powerful commercial off-the-shelf (COTS) products, some real-time system designers have started a shift from custom development to COTS-based systems to achieve lower costs and more flexible systems. This research investigates the problem of allocating real-time applications to a set of COTS heterogeneous machines connected together by a COTS high-speed network. The heuristics were implemented on the High Performance Distributed Computing Program's (HiPer-D) Naval Surface Warfare Center (NSWC) testbed. At the specification of NSWC, the goal of this study is to design static resource allocation heuristics that balance the utilization of the computation and network resources in the HiPer-D system based on the system information provided. The broader goal is to maximize the time before dynamic reallocation is required for managing an increased workload at runtime.

This research was supported by the DARPA/ITO Quorum Program through the Office of Naval Research under Grant No. N00014-00-1-0599. Some of the equipment used was donated by Intel and Microsoft.

1. Introduction

The importance of real-time systems continues to grow as they are used in various critical areas, such as command and control systems, intensive care monitoring, flight control systems, process control systems, multimedia, and high-speed communication systems [35]. In the past, computational requirements for some real-time applications could only be met by custom-made systems. Such systems generally employed special purpose computers, interconnects, languages, and operating systems designed and built to execute those real-time applications. Typically, these custom solutions are expensive and have limited flexibility. Recently, however, with the widespread use of increasingly powerful commercial off-the-shelf (COTS) products, some real-time system designers have started a shift from custom development to COTS-based systems to achieve lower costs and more flexible systems [39].

To use COTS-based systems effectively as parts of a larger system, one needs to exploit the heterogeneity in processor speeds, memory structures, specialized hardware capabilities, etc., that most likely will be present in different COTS products. Heterogeneous computing (HC) is the coordinated use of different types of machines, networks, and interfaces to meet the requirements of widely varying application mixtures and to maximize the combined performance or cost-effectiveness, e.g., [10, 14, 19, 37]. A typical real-

0-7695-1573-8/02/$17.00 (C) 2002 IEEE

time HC system consists of heterogeneous sets of sensors, real-time applications, machines, and actuators. An important research problem is how to assign resources to applications (matching) and order the execution of the applications (scheduling) so as to maximize some performance criterion without violating any real-time constraints. This process of matching and scheduling is called mapping.

Two different types of mapping are static and dynamic. Static mapping is performed when the applications are mapped in an off-line planning phase [9], e.g., when a system is first started up and a mapping is needed to ensure that all real-time constraints will be met for a given initial workload (i.e., the set of initial sensor outputs). Dynamic mapping is performed when the applications are mapped in an on-line fashion [36], e.g., when an increase in the workload of a system causes quality of service violations to occur [25, 44], or when new applications arrive unpredictably. Both types of mapping problems have been shown, in general, to be NP-complete [13, 16, 26]. Thus, the development of heuristic techniques to find near-optimal mappings is an active area of research, e.g., [6, 5, 9, 10, 14, 17, 36, 38, 45].

MSHN (Management System for Heterogeneous Networks) is a collaborative research effort among Colorado State University, Purdue University, the University of Southern California, NOEMIX, and the Naval Postgraduate School [21]. It is supported by the DARPA/ITO Quorum Program. One objective of MSHN is to design and evaluate mapping heuristics for different types of HC environments, including the COTS-based High Performance Distributed Computing Program (HiPer-D) environment at the Naval Surface Warfare Center, Dahlgren Division (NSWCDD) [39].

The contribution of this research is the design and evaluation of five mapping heuristics for the initial allocation of resources to applications in the HiPer-D environment. The heuristics are compared by using simulation experiments, and it is shown that two heuristics, MIP* and HRA Max-min, are particularly suited for HiPer-D-like HC systems with high heterogeneity.

A general framework for a HiPer-D subsystem is shown in Figure 1 and a specific example is shown in Figure 2. In Figure 1, the output from a radar is processed by a group of applications, and a signal is then produced to control the firing of a missile. The nodes denote applications in the real-time system. In Figure 2, Fire Sim 21, OTH (Over the Horizon) Data Server, and ALT (Air Engagement Control Local Track) Data Server send periodic data to the applications. Tacfire, CFF (Call for Fire) Broker, Land Attack Engagement Server, Deconflict Server, Gun Sim, and Display Components are the applications to be mapped. The arrows denote communications, and the labels next to them denote the network protocols used for communications. The labels in parentheses next to the applications denote the types

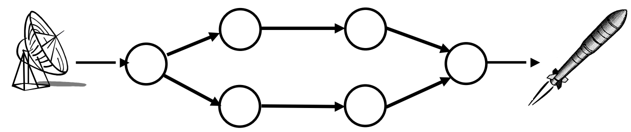


Figure 1. An example real-time system.

of machines on which those applications can execute. The HiPer-D system consists of a number of such systems.

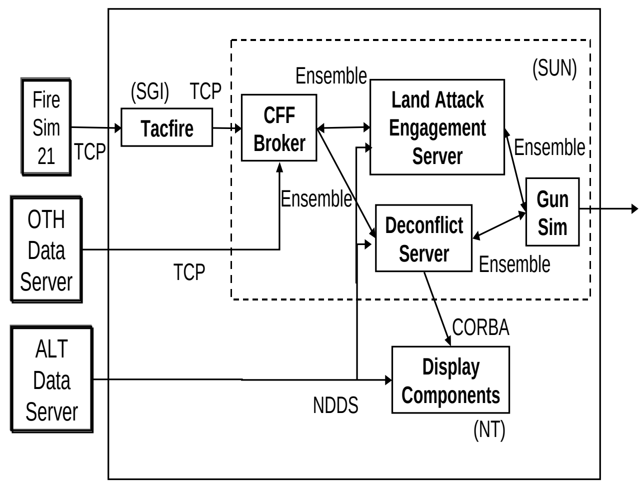


Figure 2. An example HiPer-D subsystem composed of heterogeneous COTS machines and networks.

At the specification of NSWCDD, the goal of this study is to design static mapping heuristics that minimize the utilization of the most utilized computation or network resource in the HiPer-D system. The broader goal is to maximize the time before dynamic re-mapping is required for managing an increased workload at runtime.

The rest of the paper is organized as follows. Section 2 outlines how this work is related to the previous work in this area. The system model is described in Section 3. Section 4 presents the static mapping heuristics designed for the NSWC system. The details of the simulation experiments are given in Section 5. Section 6 concludes the paper.

2. Related Work

Many research efforts in the literature concentrate on mapping real-time applications on a uniprocessor (e.g., [4, 7, 12, 20, 23, 28, 29, 30, 31, 32, 33, 34, 43, 46]). Even though these papers present good schemes to schedule real-time tasks on a uniprocessor, they could not be directly ap-

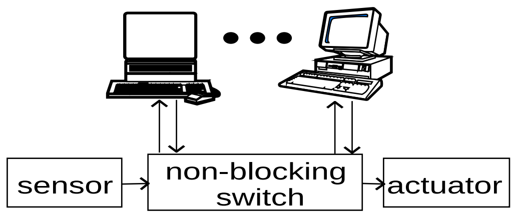


Figure 3. The hardware model. All machines have dedicated, full duplex Ethernet connections to a non-blocking switch.

plied to our work because our research assumes a very different environment that includes multitasking on multiple processors with multitasking communication links, continuously running applications, and heterogeneous distributed processors.

Some research efforts assume a multiprocessor (sometimes a multi-resource) environment (e.g., [11, 15, 22, 41, 47]). The research in [11, 15, 47] is not directly applicable because, unlike our research, it does not assume heterogeneous distributed multitasking processors. The work in [8, 24, 40] has the same system model for sensors, tasks, and actuators as the model used in our research. These research efforts are very different because in our research the processors and the communication links can perform multitasking.

3. Model

The system consists of heterogeneous sets of sensors, real-time applications, machines, and actuators. Each machine on the network has a full-duplex Ethernet connection to a non-blocking switch. The sensors and actuators have half-duplex connections (Figure 3). Each sensor produces data periodically, and these data streams are fed into applications. The applications process the data and send the output to other applications or to actuators (Figures 1 and 2).

Let $\underline{\mathcal{M}}$ be the set of machines in the system. Each machine in $\mathcal{M}$ is characterized by its floating point and integer SPEC values [42], and the "background load" on its CPU and input/output network links. The background loads on a machine are the utilizations of the CPU, input link, and output link before any applications are mapped on the system.

Let $\underline{\mathcal{A}}$ be the set of applications that need to be mapped. All applications in $\mathcal{A}$ execute continuously to process the periodic inputs that arrive from the sensors or to process input data from the predecessor applications. An application can start processing input data as soon as the data is available and the application has finished processing prior data inputs. Based on the problem definition provided by NSWCDD, an application is characterized by the CPU, input network link, and output network link utilizations that it requires on a given machine to finish within a given time constraint. Each utilization is an average value for the fraction of the resource required by the given application on a given machine to process a data set within its time period. Let $\underline{C(a_i, m_j)}$ be the CPU utilization that application $\underline{a_i}$ requires on machine $\underline{m_j}$, with the initial workload, to process a data set within the required time period (based on the rates of its associated sensors). Let $\underline{I(a_i)}$ and $\underline{O(a_i)}$ be the input network link and output network link utilizations, respectively, that application a_i requires on the machine it is executing to process a data set, at the initial workload, within the required time period. It is assumed that network link utilizations for an application do not depend on the machine where the application executes.

Including the background load on resources, the utilization of any resource on any machine cannot exceed 100%. Let $\underline{C_j^{\text{bg}}}$, $\underline{I_j^{\text{bg}}}$, and $\underline{O_j^{\text{bg}}}$ be the background utilizations on the CPU, input network link, and output network link, respectively, of machine m_j. Let $\underline{\mathcal{A}_j}$ be the set of applications already mapped on machine m_j. Let $\underline{C_j}$, $\underline{I_j}$, and $\underline{O_j}$ be the total utilizations on the CPU, input network link, and output network link, respectively, of machine m_j. Then,

$$C_j = C_j^{\text{bg}} + \sum_{a_i \in \mathcal{A}_j} C(a_i, m_j),$$

$$I_j = I_j^{\text{bg}} + \sum_{a_i \in \mathcal{A}_j} I(a_i), \text{and}$$

$$O_j = O_j^{\text{bg}} + \sum_{a_i \in \mathcal{A}_j} O(a_i).$$

The total utilization of the most heavily loaded resource (the CPU, input link, or output link) of machine m_j is given by $\underline{U_j} = \max(C_j, I_j, O_j)$. Ensuring that no resource is more than 100% utilized implies that, for $1 \leq j \leq |\mathcal{M}|$, $U_j \leq 100\%$.

The performance objective for the mappings in this study is the minimization of the utilization of the most heavily loaded resource, here called the maximum utilization, defined as $\underline{U_{\text{max}}} = \max_j(U_j)$.

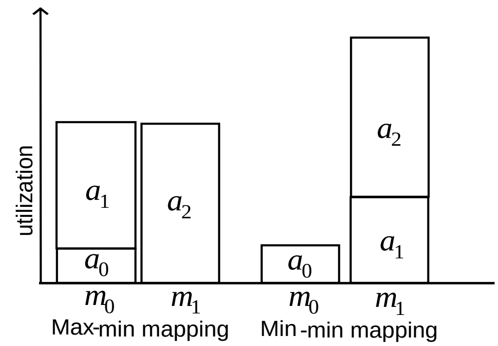
4. Mapping Heuristics

This study examines five mapping heuristics, namely: (i) the Min-min heuristic, (ii) the Max-min heuristic, (iii) the host-restriction-aware Min-min heuristic, (iv) the host-restriction-aware Max-min heuristic, and (v) the Mixed-Integer-Programming-based heuristic (referred to as MIP* in the following text).

The Min-min heuristic is based on [26], and is one of the heuristics implemented in SmartNet [18]. Some variants of

(1) **for** all machines m_j
(2) $\quad U_j = \max(C_j^{\rm bg}, I_j^{\rm bg}, O_j^{\rm bg})$
(3) **do** until all applications are mapped
(4) $\quad$ **for** each unmapped application a_r, find the machine m_k such that $U_{r,k} = \min_j U_{r,j}$ and $U_{r,k} \leq 1$
(5) $\quad$ **if** no such machine found, print "mapping not found" and return
(6) $\quad$ from the $(a_r,\ m_k)$ pairs found in step (4), select the pair $(a_x,\ m_y)$ for which $U_{x,y} = \min_{(a_r,\ m_k)} U_{r,k}$
(7) $\quad$ assign the application a_x to the machine m_y
(8) $\quad$ mark the application a_x as mapped
(9) $\quad$ update C_y, I_y, and O_y
(10) **end do**

Figure 4. The Min-min heuristic.



	m_0	m_1
a_0	2%	11%
a_1	18%	15%
a_2	40%	20%

Figure 5. A scenario where Max-min performs better than Min-min.

the Min-min heuristic were studied in, e.g., [3, 9, 36, 45], and were seen to perform well in many different environments.

Formally, the version of the Min-min heuristic designed for the NSWCDD model can be defined as follows. Let $\underline{U_{r,j}}$ be the total utilization of the most heavily loaded resource (the CPU, input link, or output link) of machine m_j, if the currently unmapped application a_r is mapped on machine m_j. That is,

$$U_{r,j} = \max(C_j + C(a_r, m_j), I_j + I(a_r), O_j + O(a_r)).$$

Let $\underline{U^*} = \min_{i:a_i \in (\mathcal{A} - \bigcup_j \mathcal{A}_j)} (\min_{j:m_j \in \mathcal{M}} U_{i,j})$. The outer minimum in the preceding expression is taken over all unmapped applications (i.e., $\mathcal{A} - \bigcup_j \mathcal{A}_j$). Figure 4 shows the pseudo-code used to implement Min-min for the NSWCDD model. Min-min selects the x, y for which $U_{x,y} = U^* \leq 1$, assigns a_x to m_y, adds a_x to $\mathcal{A}_y$, and updates U_y to reflect the assignment. The above process is repeated until all applications are mapped. If $U^* > 1$ in some iteration, a mapping cannot be found with this heuristic.

The Max-min heuristic is similar to the Min-min heuristic, and is also one of the heuristics implemented in SmartNet [18]. It differs from the Min-min heuristic in that now $U^* = \max_{i:a_i \in (\mathcal{A} - \bigcup_j \mathcal{A}_j)} (\min_{j:m_j \in \mathcal{M}} U_{i,j})$, and in Figure 4, "$U_{x,y} = \min_{(a_r,\ m_k)} U_{r,k}$" in step (6) is replaced with "$U_{x,y} = \max_{(a_r,\ m_k)} U_{r,k}$." Max-min is likely to do better than Min-min in scenarios like the one shown in Figure 5. Here, the system consists of three applications and two machines. The table in Figure 5 shows the $C(a_i, m_j)$ values. For all i, j, $I(a_i) = O(a_i) = C_j^{\rm bg} = I_j^{\rm bg} = O_j^{\rm bg} = 0$. Min-min maps a_0 and a_1 before mapping a_2, while Max-min maps a_2 first.

The host-restriction-aware Min-min heuristic (HRA Min-min) considers the fact that in many systems a given application may not be able to execute on all machines in the system. This may arise because the application is not compiled for all machines or it requires specialized capabilities available only on select machines. In such systems, the Min-min or Max-min heuristics may fail to find "obvious" mappings for some cases. One such case is shown in Table 1 where both Min-min and Max-min fail to find the obvious mapping (a_0 on m_1, a_1 on m_2, and a_2 on m_0). In Table 1, for all i, j, $I(a_i) = O(a_i) = C_j^{\rm bg} = I_j^{\rm bg} = O_j^{\rm bg} = 0$. The symbol ∞ for an entry $C(a_i, m_j)$ indicates that application a_i cannot execute on machine m_j. Min-min assigns a_1 to m_1 in the first iteration, thereby depriving a_0 of the only machine on which it could execute. Similarly, Max-min assigns a_1 to m_0 in the second iteration, thereby depriving a_2 of the only machine on which it could execute. The HRA Min-min heuristic, described next, does find the obvious mapping.

The HRA Min-min heuristic is shown in Figure 6. In each iteration, the heuristic splits the unmapped applications into two sets, and tries to map those sets separately. Let $\underline{U_1^p}$ and $\underline{U_2^p}$ be the two sets of unmapped applications in iteration p. Let $\underline{\mathcal{S}(k)}$ be the set of applications that can map on exactly k machines. In the first iteration, the heuristic splits all applications such that $U_1^1 = \mathcal{S}(1)$ and $U_2^1 = \mathcal{A} - \mathcal{S}(1)$. Then it attempts to map the applications in U_1^1 onto their respective machines. If that partial mapping is not successful, then no mapping exists. If this partial map-

	m_0	m_1	m_2
a_0	∞	60%	∞
a_1	60%	45%	70%
a_2	50%	∞	∞

Table 1. An example scenario showing $C(a_i, m_j)$ values for a system of three applications and three machines where HRA Min-min finds the obvious mapping, but Min-min and Max-min do not find any feasible mapping.

ping is successful, it tries to map U_2^1 by using Min-min. If the mapping of U_2^1 fails, the heuristic undoes any changes it made to the system while trying to find the mapping of U_2^1, and then moves to the second iteration. In the second iteration, $U_1^2 = \mathcal{S}(2)$ and $U_2^2 = \mathcal{A} - U_1^2 - U_1^1$. In general, in the p-th iteration, $U_1^p = \mathcal{S}(p)$ and $U_2^p = \mathcal{A} - \bigcup_{k=1}^{p} U_1^k$. For all p except $p = 1$, HRA Min-min uses Min-min to map U_1^p. For the case of U_1^1, performing Min-min is equivalent to assigning each application in U_1^1 to the only machine on which it can execute.

The maximum number of iterations is equal to the total number of machines in the system. In the best case, a complete mapping is found in the first iteration; in the worst case, a complete mapping is not found until the last iteration.

The host-restriction-aware Max-min heuristic (HRA Max-min) is similar to the HRA Min-min except that it uses the Max-min heuristic instead of Min-min. That is, in steps (4) and (6) in Figure 6, Max-min is used instead of Min-min.

The MIP* heuristic is based on the well-researched mixed integer programming (MIP) mathematical technique for optimization [1]. A mathematical programming formulation based on the model in Section 3 is developed to map the applications onto machines. The set $\{x_{ij}\}$ defines a mapping of applications onto machines such that

$$x_{ij} = \begin{cases} 1 & \text{if application } a_i \text{ is mapped onto machine } m_j \\ 0 & \text{otherwise} \end{cases}$$

(1) $N = 1$
(2) **while** ($N \leq |\mathcal{M}|$)
// if there are any applications that can map to
// only N machines, map those applications first
(3) **if** $(|\mathcal{S}(N)| > 0)$
(4) use Min-min to find a mapping for $\mathcal{S}(N)$
(5) **if** a mapping for $\mathcal{S}(N)$ is not found
print “no feasible mapping found”
exit
(6) use Min-min to find a mapping for all of the remaining applications, marking each assignment as “speculative”
(7) **if** a complete mapping is not found in step (6)
(8) **for each** speculative assignment (a_i, m_j) made in step (6)
// roll back - undo all changes to
// the system data structures
(9) undo the mapping of a_i on m_j
(10) mark application a_i as unmapped
(11) undo the increases in the CPU, input link, and output link utilizations of machine m_j that were caused by speculative mapping of a_i on m_j
(12) $N = N + 1$
(13) **else** // matches the “if” in step (7)
(14) print “mapping found”
(15) return mapping
(16) **else** // matches the “if” in step (3)
(17) $N = N + 1$
(18) **end while**

Figure 6. The HRA Min-min heuristic.

given $\mathcal{M}, \mathcal{A}, \{C(a_i, m_j)\}, \{I(a_i)\}, \{O(a_i)\}$
find x_{ij} and U, where $x_{ij} \in \{0, 1\}$
and U is a real number
to
minimize

$$U$$

subject to

$$
\begin{aligned}
& U \leq 1 \\
& \forall m_j \in \mathcal{M},\ C_j \leq U \\
& \forall m_j \in \mathcal{M},\ I_j \leq U \\
& \forall m_j \in \mathcal{M},\ O_j \leq U \\
& \forall a_i \in \mathcal{A}, \sum_{j: m_j \in \mathcal{M}(a_i)} x_{ij} = 1 \\
& \forall a_i \in \mathcal{A}, \sum_{j: m_j \notin \mathcal{M}(a_i)} x_{ij} = 0
\end{aligned}
$$

Figure 7. The mixed integer programming formulation.

where $1 \leq i \leq |\mathcal{A}|$ and $1 \leq j \leq |\mathcal{M}|$. In terms of x_{ij},

$$C_j = C_j^{\mathrm{bg}} + \sum_{1 \leq\ i \leq |\mathcal{A}|} (x_{ij} \times C(a_i, m_j))$$

$$I_j = I_j^{\mathrm{bg}} + \sum_{1 \leq\ i \leq |\mathcal{A}|} (x_{ij} \times I(a_i))$$

$$O_j = O_j^{\mathrm{bg}} + \sum_{1 \leq\ i \leq |\mathcal{A}|} (x_{ij} \times O(a_i))$$

Let $\mathcal{M}(a_i)$ be the set of hosts onto which application a_i can be mapped. Figure 7 shows the MIP formulation, where U is an auxiliary variable that will equal the minimum value of $U_{\max}$ when the optimization is complete. In this paper, the objective of the MIP formulation is to minimize $U_{\max}$ based on the constraints that both CPU and network utilizations of each machine are less than or equal to 100%. (However, this approach can be extended to optimize more complex metrics.) The last two constraints in Figure 7 force application a_i to be mapped onto exactly one machine in $\mathcal{M}(a_i)$.

Because the above objective function minimizes the maximum utilization (CPU or network) among all machines, the mapping of applications on the less utilized machines may not be necessarily optimized. To achieve system-wide optimization, the MIP* heuristic uses an iterative way to solve the problem. The mapping is described as a set, $\mathcal{T}$, of $|\mathcal{A}|$ tuples, where $\mathcal{T} = \{T_1, \ldots, T_{|\mathcal{A}|}\}$. Each tuple T_i is in the form (a_i, m_j), where $a_i \in \mathcal{A}$ and $m_j \in \mathcal{M}$. Note that there is a tuple (a_i, m_j) in $\mathcal{T}$ iff $x_{ij} = 1$. The complete pseudo-code is shown in Figure 8.

(1) initialize $\mathcal{T}$ to $\emptyset$
(2) let $\mathcal{M}^*$ and $\mathcal{A}^*$ denote the set of machines
and applications that need to be mapped
(3) initialize $\mathcal{M}^*$ to $\mathcal{M}$ and $\mathcal{A}^*$ to $\mathcal{A}$
(4) **repeat**
(5) Using $\mathcal{M}^*$ and $\mathcal{A}^*$, construct a MIP problem instance
(based on the MIP formulation shown in Figure 7)
(6) solve the MIP problem instance using an MIP solver
(7) find out the machine m_x that has the
highest CPU or network utilization
(8) for each application $a_i \in \mathcal{A}_x$
// record the mapping information
// regarding m_x in $\mathcal{T}$
(9) add (a_i, m_x) into $\mathcal{T}$
(10) delete a_i from $\mathcal{A}^*$
(11) delete m_x from $\mathcal{M}^*$
(12) **until** $\mathcal{M}^* = \emptyset$

Figure 8. The MIP* heuristic.

In addition to the five heuristics mentioned above, this study also examined a fast greedy heuristic, a random allocation heuristic, and a lower bound (LB) on the maximum utilization. The fast greedy heuristic and the random allocation heuristics are shown in Figure 9. Note that, unlike the Min-min or Max-min heuristics, the fast greedy and the random allocation heuristics iterate through the set of applications only once. The lower bound on the maximum utilization is calculated by assuming that for all applications $I(a_i)$ and $O(a_i)$ are zero, that each application a_i is mapped on the machine m_j where $C(a_i, m_j)$ is minimum over all machines, and that the sum of the utilizations can be divided equally over all of the machines (which, in general, may not be physically realistic). Specifically, $\mathrm{LB} = (\sum_i \min_j C(a_i, m_j) + \sum_j C_j^{\mathrm{bg}})/|\mathcal{M}|$. An example of when this lower bound situation could occur is: (1) all applications that communicate with each other are mapped to the same machine, (2) each application is mapped to its best machine, and (3) the set of applications is such that all machines are equally utilized.

5. Simulation Experiments and Results

In this study, several sets of simulation experiments were conducted to evaluate and compare the heuristics. For all experiments, the number of machines in the system was fixed at ten. Also, it was assumed that every application could execute on at least one machine. That machine was chosen randomly from among all of the machines in the system. For any other machine, the probability that a given application could execute on it was 50%.

```
(1) for all machines m_j
(2)     U_j = max(C_j^bg, I_j^bg, O_j^bg)
    // iterate through the applications
    // in an arbitrary order
(3) for i = 1 to |A|
(4)     find the machine m_k such that
            U_{r,k} = min_j U_{r,j} and U_{r,k} ≤ 1
(5)     if no such machine found,
            print "mapping not found" and return
(6)     assign the application a_r to the machine m_k
(7)     update C_k, I_k, and O_k
(8) end for
```

(a)

```
(1) for all machines m_j
(2)     U_j = max(C_j^bg, I_j^bg, O_j^bg)
    // iterate through the applications
    // in an arbitrary order
(3) for i = 1 to |A|
(4)     identify the set, L, of machines such that
            if a_i is mapped on m_j ∈ L, U_j ≤ 1
(5)     if L is empty,
            print "mapping not found" and return
(6)     assign a_i to a randomly chosen
            machine from L
(7)     update C_j, I_j, and O_j
(8) end for
```

(b)

Figure 9. (a) The fast greedy heuristic. (b) The random allocation heuristic.

The $C(a_i, m_j)$ matrix was generated by sampling a probability distribution $\underline{\mathbf{D^C}}$. The entries in the $C(a_i, m_j)$ matrix were generated to have a mean $\underline{M^{\mathrm{C}}}$, a "task heterogeneity" $\underline{H^{\mathrm{C}}_{\mathrm{task}}}$ (heterogeneity is the standard deviation divided by the mean), and a "machine heterogeneity" $\underline{H^{\mathrm{C}}_{\mathrm{mach}}}$. See [2] for a description of the method used in this study for generating random numbers with given mean and heterogeneity values. The $I(a_i)$ and $O(a_i)$ values were generated by sampling a probability distribution, $\underline{\mathbf{D^{IO}}}$, with a mean $\underline{M^{\mathrm{IO}}}$ and heterogeneity $\underline{H^{\mathrm{IO}}}$. In this study, $\mathbf{D^C}$ and $\mathbf{D^{IO}}$ were either both gamma distributions or both uniform distributions.

The C_j^{bg} values were generated by sampling a uniform distribution with a mean $\underline{M^{\mathrm{C}}_{\mathrm{bg}}}$ and a heterogeneity $\underline{H^{\mathrm{C}}_{\mathrm{bg}}}$. The I_j^{bg} and O_j^{bg} values were sampled from a uniform distribution with a mean $\underline{M^{\mathrm{IO}}_{\mathrm{bg}}}$ and a heterogeneity $\underline{H^{\mathrm{IO}}_{\mathrm{bg}}}$.

The simulation experiments were conducted for the parameter values as shown in Table 2. Each experiment was repeated enough number of times to give, for $U_{\max}$, a 95% confidence interval with a "precision" (i.e., the ratio of the half-width of the confidence interval to the mean [27]) of 10% or better. Call each repetition of a given experiment a trial. In each trial, $C(a_i, m_j)$, $I(a_i)$, $O(a_i)$, C_j^{bg}, I_j^{bg}, and O_j^{bg} values were re-generated by sampling their respective distributions.

The results for a selected set of representative experiments for inconsistent HC environments are shown in Figures 10 to 13. These figures also show the value of failure ratio (FR) for each heuristic, where FR is the ratio of the number of trials in which the heuristic could not find a mapping to the total number of trials. Note that the notion of a failure ratio does not apply to LB (therefore FR for LB is always shown to be zero in the results given here).

For each heuristic, at most three bars are shown in these figures. The first bar (from the left) shows the average value of $U_{\max}$ for that heuristic with a 95% confidence interval and a 10% (or better) precision. The $U_{\max}$ shown in the first bar is based on only those trials where the given heuristic was successful at finding a mapping; thus, the $U_{\max}$ values for two different heuristics may be based on different subsets of the trials. The second bar shows $\underline{u_{\max}}$ - the $U_{\max}$ value averaged only for those trials for which every heuristic successfully found a mapping. Finally, the third bar shows FR for the given heuristic. When FR for a heuristic is zero, the third bar is not shown.

Consider the significance of the performance metric $u_{\max}$. When FR is zero for all heuristics in a given experiment, then $u_{\max}$ equals the average value of $U_{\max}$, because no trials are excluded for the purpose of calculating $u_{\max}$. For the sake of discussion, assume that, in a certain experiment comparing two heuristics Alg-A and Alg-B, FR is non-zero for Alg-A and is zero for Alg-B. Then, for Alg-B, $u_{\max}$ may differ from the average value of $U_{\max}$ because some trials will be excluded for the purpose of calculating $u_{\max}$. If Alg-B had performed as well in the excluded trials as in the included trials, the difference between $u_{\max}$ and $U_{\max}$ would be zero. However, if Alg-B had performed poorly in the excluded trials, $u_{\max}$ would be smaller than $U_{\max}$. Also, note that by definition, $u_{\max} = U_{\max}$ for Alg-A.

Figure 10 shows, for one set of parameters, the relative values of $U_{\max}$ and FR for the heuristics discussed in this paper. It can be seen that MIP*, Max-min, and HRA Max-min give almost the same $U_{\max}$ value, outperforming all other heuristics. The running time of HRA Max-min was smaller than that of MIP* by a factor of about 260. It should be noted that the mapping generation time for MIP* was lim-

$\lvert\mathcal{A}\rvert$	M^{C}	$H^{\mathrm{C}}_{\mathrm{task}}$	$H^{\mathrm{C}}_{\mathrm{mach}}$	M^{IO}	H^{IO}	$\mathbf{D^C} = \mathbf{D^{IO}}$	$M^{\mathrm{C}}_{\mathrm{bg}}$ = $M^{\mathrm{IO}}_{\mathrm{bg}}$	$H^{\mathrm{C}}_{\mathrm{bg}}$ = $H^{\mathrm{IO}}_{\mathrm{bg}}$
40	12 or 18	1.4 or 1.8	0.4	7 or 10	0.3 or 0.5	uniform or gamma	7 or 10	0.3
80	6 or 9	1.4 or 1.8	0.4	3.5 or 5	0.3 or 0.5	uniform or gamma	3.5 or 5	0.3

Table 2. A tabulation of parameter values for which the experiments were performed. An entry that contains an "or" indicates that separate sets of experiments were performed for each value.

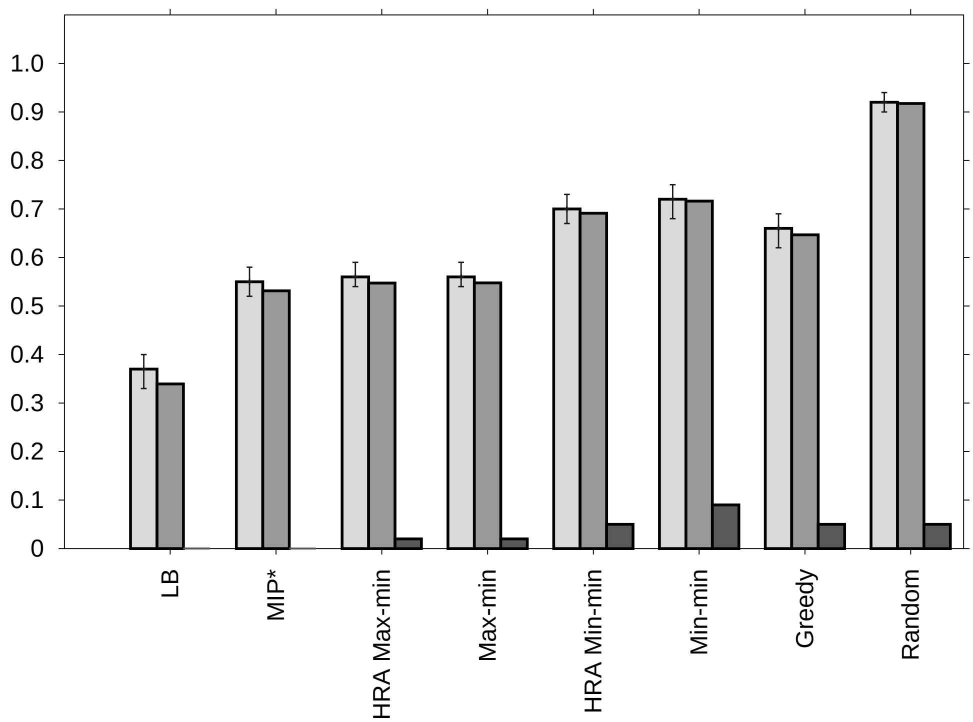


Figure 10. The variation of $U_{\max}$, $u_{\max}$, and FR for different heuristics. The bar with confidence intervals shows $U_{\max}$, the second bar shows $u_{\max}$, and the third bar if present shows FR. $|\mathcal{A}| = 40$, M^{C} = 12%, $H^{\mathrm{C}}_{\mathrm{task}} = 1.4$, $M^{\mathrm{IO}} = 7\%$, $H^{\mathrm{IO}} = 0.3$, $M^{\mathrm{C}}_{\mathrm{bg}} = M^{\mathrm{IO}}_{\mathrm{bg}} = 7$, and $\mathrm{D^C} = \mathrm{D^{IO}} =$ gamma. A total of 55 trials were performed.

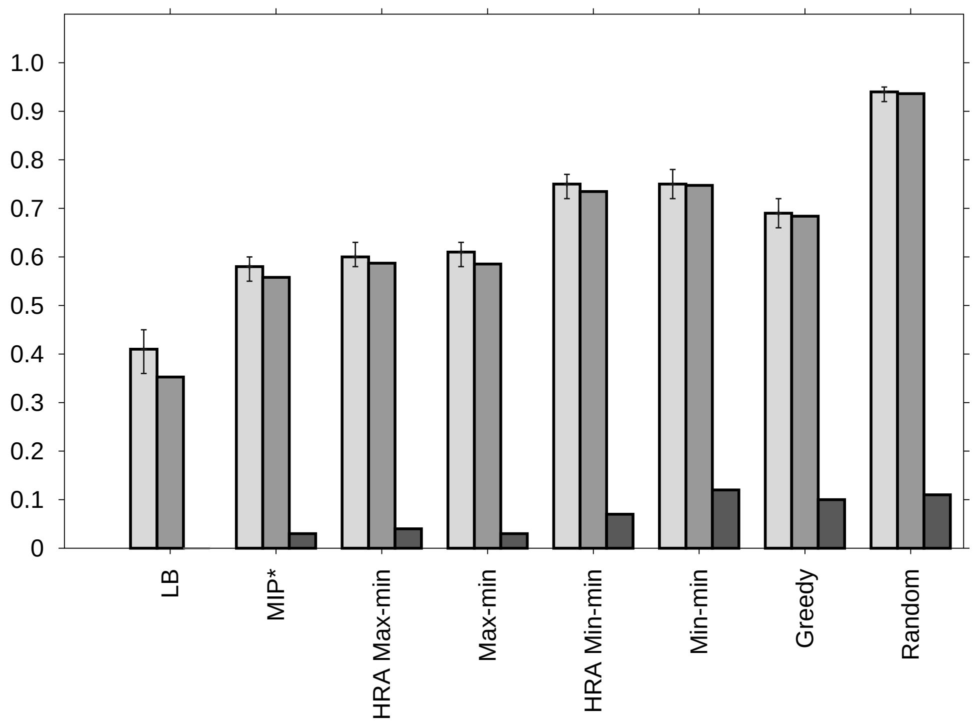


Figure 11. The variation of $U_{\max}$, $u_{\max}$, and FR for different heuristics. The bar with confidence intervals shows $U_{\max}$, the second bar shows $u_{\max}$, and the third bar if present shows FR. All parameters are the same as in Figure 10 except $H^{\mathbf{C}}_{\mathbf{task}}$ which has increased to 1.8 and $H^{\mathbf{IO}}$ which has increased to 0.5. A total of 90 trials were performed.

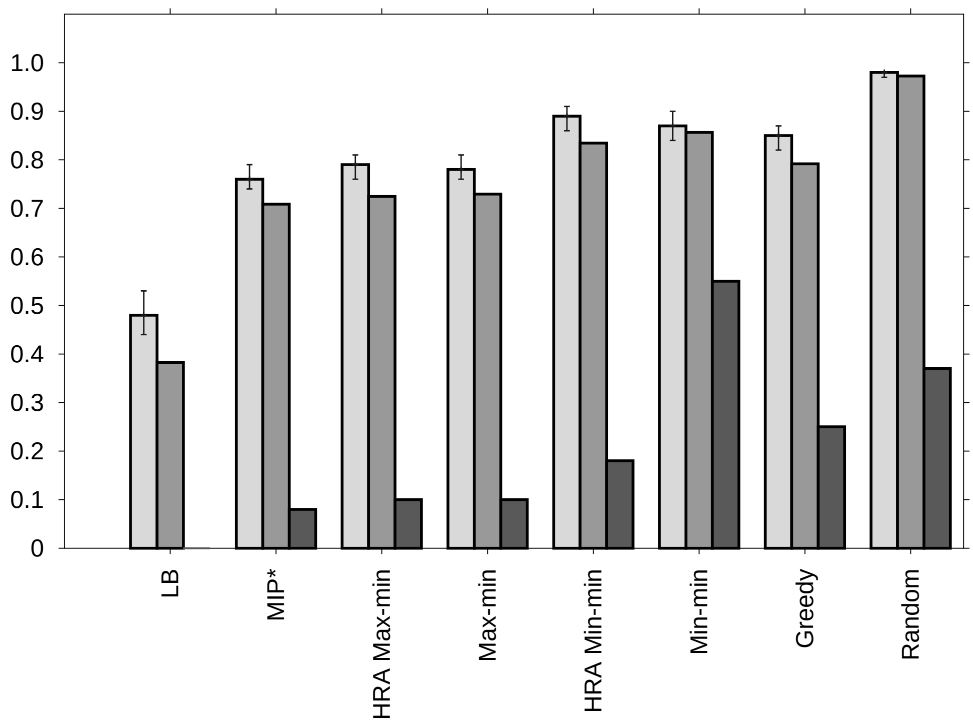


Figure 12. The variation of $U_{\max}$, $u_{\max}$, and FR for different heuristics. The bar with confidence intervals shows $U_{\max}$, the second bar shows $u_{\max}$, and the third bar if present shows FR. All parameters are the same as in Figure 10 except M^{C} which has increased to 18 and M^{IO} which has increased to 10. A total of 60 trials were performed.

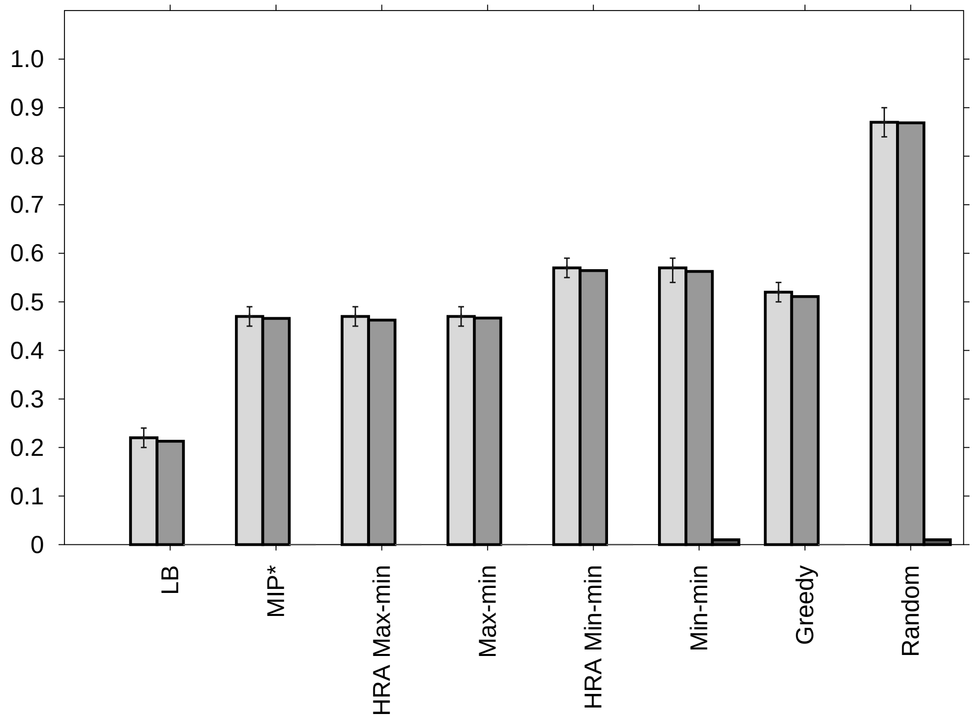


Figure 13. The variation of $U_{\max}$, $u_{\max}$, and FR for different heuristics. The bar with confidence intervals shows $U_{\max}$, the second bar shows $u_{\max}$, and the third bar if present shows FR. With respect to Figure 10, $|\mathcal{A}|$ has doubled and $M^{\mathbf{C}}$ and $M^{\mathbf{IO}}$ have halved. A total of 67 trials were performed.

ited to fifteen seconds (on an UltraSparc III 750MHz, one gigabyte memory machine running Solaris 5.8). The variation of the quality of mappings generated by MIP* as a function of the mapping generation time will be discussed later.

Figure 11 shows the change in relative performance when the heterogeneity of the application resource utilization increases. The value of $H^{\mathrm{C}}_{\mathrm{task}}$ increases to 1.8 from a value of 1.4 in Figure 10, and H^{IO} increases to 0.5 from a value of 0.3. The higher values of $H^{\mathrm{C}}_{\mathrm{task}}$ and H^{IO} increase the FR for all heuristics. The same effect can be seen in Figure 12, which shows the change in relative performance when the average application resource utilization increases. Here, the value of M^{C} increases to 18% from a value of 12% in Figure 10, and M^{IO} increases to 10% from a value of 7%. Figures 11 and 12 show that, for the parameters used in these experiments, the failure ratios for the greedy, random allocation, and Min-min techniques are higher in systems with higher application heterogeneity or higher average application resource utilization. These heuristics are, therefore, very undesirable for systems with limited resources and high application heterogeneity or high average resource requirement. In contrast, MIP*, Max-min, and HRA Max-min heuristics are appropriate for such systems. Low failure ratios can be very critical in HiPer-D-like systems.

Figure 13 shows the change in the relative performance of heuristics when the number of applications is doubled to 80 (from a value of 40 in Figure 10). At the same time, the average resource requirement of the applications is halved (M^{C} is halved to 6% and M^{IO} is halved to 3.5%). It is expected that, in general, when the resource requirements of the applications become smaller, the load imbalance generated by "bad" mapping decisions becomes smaller as well. This is seen in Figure 13 where the relative performance differences between different heuristics have dropped.

Note that the results given in this paper are for the simulation experiments. Future work will include the results that can be obtained from experiments on the NSWCDD testbed.

A discussion of how the length of mapping generation time affects the quality of the mapping generated by MIP* is now presented. Experiments were conducted for mapping generation times of one, two, five, fifteen, and 360 seconds on an UltraSparc III 750MHz, one gigabyte memory machine running Solaris 5.8. Each experiment was defined by the set of parameters in Figure 13, and was repeated for 30 trials. The solutions found with a mapping generation time of fifteen seconds were very close in quality (as measured by $U_{\max}$) to those found with a mapping generation time of 360 seconds. For these two mapping generation times, the maximum difference in $U_{\max}$ over all trials was less than 2%. For mapping generation times of one second and 360 seconds, the maximum difference in $U_{\max}$ over all trials was about 11%. In addition, no feasible solution was found for four trials with a mapping generation time of one second.

6. Conclusions

This paper presented five static heuristics designed to map applications onto machines in the NSWCDD platform. The heuristics were compared under a variety of simulated heterogeneous computing environments. The results from the simulation experiments show that MIP* and HRA Max-min are the heuristics of choice in HC environments with high application and machine heterogeneities, or with high average resource requirement. Both of these heuristics give comparable performance with either $U_{\max}$ or FR as the performance metric. The HRA Max-min heuristic takes a much shorter mapping generation time than MIP*. The results show that, among all heuristics compared in this study, MIP* is the best heuristic for mapping in the NSWCDD environment if the time to generate the mapping is not an issue. However, if the time to generate a mapping should be small, HRA Max-min is the best heuristic.

References

[1] A. Kaufmann. *Integer and Mixed Programming: Theory and Applications*. Academic Press, New York, NY, 1977.

[2] S. Ali, H. J. Siegel, M. Maheswaran, D. Hensgen, and S. Ali. Representing task and machine heterogeneities for heterogeneous computing systems. *Tamkang Journal of Science and Engineering*, Special 50th Anniversary Issue, 3(3):195–207, invited, Nov. 2000.

[3] R. Armstrong, D. Hensgen, and T. Kidd. The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions. In *7th IEEE Heterogeneous Computing Workshop (HCW '98)*, pages 79–87, Mar. 1998.

[4] N. C. Audsley, A. Burns, M. F. Richardson, and A. J. Wellings. Hard real-time scheduling: The deadline monotonic approach. In *Real Time Programming*, W. A. Halang and K. Ramamritham, eds., pages 127–132. Pergamon, Oxford, UK (Proceedings of the IFAC/IFIP Workshop, May 1991), 1992.

[5] I. Banicescu and V. Velusamy. Performance of scheduling scientific applications with adaptive weighted factoring. In *10th IEEE Heterogeneous Computing Workshop (HCW 2001)*, Apr. 2001, in the CD-ROM "Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS 2001)," 2001, paper HCW_06.

[6] H. Barada, S. M. Sait, and N. Baig. Task matching and scheduling in heterogeneous systems using simulated evolution. In *10th IEEE Heterogeneous Computing Workshop (HCW 2001)*, Apr. 2001, in the CD-ROM "Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS 2001)," 2001, paper HCW_15.

[7] S. Baruah, A. Mok, and L. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *11th Real-Time Systems Symposium*, pages 182–190, Dec. 1990.

[8] R. Bettati and J. W.-S. Liu. End-to-end scheduling to meet deadlines in distributed systems. In *International Conference on Distributed Computing Systems*, pages 452–459, June 1992.

[9] T. D. Braun, H. J. Siegel, N. Beck, L. L. Boloni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen, and R. F. Freund. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *Journal of Parallel and Distributed Computing*, 61(6):810–837, June 2001.

[10] T. D. Braun, H. J. Siegel, and A. A. Maciejewski. Heterogeneous computing: Goals, methods, and open problems. In *The 2001 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '01)*, pages 1–12 (invited keynote paper), June 2001.

[11] B. M. Carlson and L. W. Dowdy. Static processor allocation in a soft real-time multiprocessor environment. *IEEE Transactions on Parallel and Distributed Systems*, 5(3):316–320, Mar. 1994.

[12] H. Chetto and M. Chetto. Some results of the earliest deadline scheduling algorithm. *IEEE Transactions on Software Engineering*, 15(10):1261–1269, Oct. 1989.

[13] E. G. Coffman, Jr. (ed.). *Computer and Job-Shop Scheduling Theory*. John Wiley & Sons, New York, NY, 1976.

[14] M. M. Eshaghian, editor. *Heterogeneous Computing*. Artech House, Norwood, MA, 1996.

[15] S. Faucou, A.-M. Deplanche, and J.-P. Beauvais. Heuristic techniques for allocating and scheduling communicating periodic tasks in distributed real-time systems. In *IEEE International Workshop on Factory Communication Systems*, pages 257–265, Sep. 2000.

[16] D. Fernandez-Baca. Allocating modules to processors in a distributed system. *IEEE Transaction on Software Engineering*, SE-15(11):1427–1436, Nov. 1989.

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

[18] R. F. 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 *7th IEEE Heterogeneous Computing Workshop (HCW '98)*, pages 184–199, Mar. 1998.

[19] R. F. Freund and H. J. Siegel. Heterogeneous processing. *IEEE Computer*, 26(6):13–17, June 1993.

[20] M. G. Harbour, M. H. Klein, and J. P. Lehoczky. Fixed priority scheduling periodic tasks with varying execution priority. In *12th Real-Time Systems Symposium*, pages 116–128, Dec. 1991.

[21] D. A. Hensgen, T. Kidd, D. S. John, M. C. Schnaidt, H. J. Siegel, T. D. Braun, M. Maheswaran, S. Ali, J.-K. Kim, C. Irvine, T. Levin, R. F. Freund, M. Kussow, M. Godfrey, A. Duman, P. Carff, S. Kidd, V. Prasanna, P. Bhat, and A. Alhusaini. An overview of MSHN: The Management System for Heterogeneous Networks. In *8th IEEE Heterogeneous Computing Workshop (HCW '99)*, pages 184–198, Apr. 1999.

[22] C.-J. Hou and K. G. Shin. Allocation of periodic task modules with precedence and deadline constraints in distributed real-time systems. *IEEE Transactions on Computers*, 46(12):1338–1356, Dec. 1997.

[23] R. R. Howell and M. K. Venkatrao. On non-preemptive scheduling of recurring tasks using inserted idle times. *Information and Computation*, 117(1):50–62, Feb. 1995.

[24] C.-W. Hsueh and K.-J. Lin. Scheduling real-time systems with end-to-end timing constraints using the distributed pinwheel model. *IEEE Transactions on Computers*, 50(1):51–66, Jan. 2001.

[25] E. Huh, L. R. Welch, B. A. Shirazi, B. Tjaden, and C. D. Cavanaugh. Accomodating QoS prediction in an adaptive resource management framework. In *Parallel and Distributed Processing*, J. Rolim *et al*., eds., volume 1800, pages 792–799. Lecture Notes in Computer Science, Springer-Verlag, New York, NY, 2000.

[26] O. H. Ibarra and C. E. Kim. Heuristic algorithms for scheduling independent tasks on nonidentical processors. *Journal of the ACM*, 24(2):280–289, Apr. 1977.

[27] R. Jain. *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, Inc., New York, NY, 1991.

[28] K. Jeffay, D. F. Stanat, and C. U. Martel. On optimal, non-preemptive scheduling of periodic and sporadic tasks. In *12th Real-Time Systems Symposium*, pages 129–139, Dec. 1991.

[29] J. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *11th Real-Time Systems Symposium*, pages 201–209, Dec. 1990.

[30] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In *10th Real-Time Systems Symposium*, pages 166–171, Dec. 1989.

[31] J. P. Lehoczky and S. Ramos-Thuel. An optimal algorithm for scheduling soft-aperiodic tasks in fixed-priority preemptive systems. In *13th Real-Time Systems Symposium*, pages 110–123, Dec. 1992.

[32] J. P. Lehoczky, L. Sha, and J. K. Strosnider. Enhanced aperiodic responsiveness in hard real-time environments. In *8th Real-Time Systems Symposium*, pages 261–270, Dec. 1987.

[33] J. Leung and M. Merrill. A note on preemptive scheduling of periodic, real-time tasks. *Information Processing Letters*, 11(3):115–118, Nov. 1980.

[34] C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1):46–61, Jan. 1973.

[35] J. W. S. Liu. *Real-Time Systems*. Prentice Hall, Upper Saddle River, NJ, 2000.

[36] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund. Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. *Journal of Parallel and Distributed Computing*, 59(2):107–131, Nov. 1999.

[37] M. Maheswaran, T. D. Braun, and H. J. Siegel. Heterogeneous distributed computing. In J. G. Webster, editor, *Encyclopedia of Electrical and Electronics Engineering, Vol. 8*, pages 679–690. John Wiley, New York, NY, 1999.

[38] Z. Michalewicz and D. B. Fogel. *How to Solve It: Modern Heuristics*. Springer-Verlag, New York, NY, 2000.

[39] Naval Surface Warfare Center,. In *Report of the HiPer-D C&CA Working Group*, `http://www.nswc.navy.mil/hiperd/reports/thrust/`, 1999.

[40] D.-T. Peng, K. G. Shin, and T. F. Abdelzaher. Assignment and scheduling of communicating periodic tasks in distributed real-time systems. *IEEE Transactions on Software Engineering*, 23(12):745–758, Dec. 1997.

[41] K. Ramamritham, J. A. Stankovic, and P.-F. Shiah. Efficient scheduling algorithms for real-time multiprocessor systems. *IEEE Transactions on Parallel and Distributed Systems*, 1(2):184–194, Apr. 1990.

[42] SPEC CPU '95 technical manual. Warrenton, VA, Aug., 1995. Standard Performance Evaluation Corporation.

[43] K. W. Tindell, A. Burns, and A. J. Wellings. An extendible approach for analysing fixed priority hard real-time tasks. *Real-Time Systems*, 6(2):133–151, Mar. 1994.

[44] L. R. Welch, B. A. Shirazi, B. Ravindran, and C. Bruggeman. DeSiDeRaTa: QoS management technology for dynamic, scalable, dependable, real-time systems. In *Distributed Computer Control Systems 1998,* F. De Paoli and I. M. MacLeod, eds., pages 7–12. Elsevier Science, Kidlington, UK (Proceedings of the 15th International Federation of Automatic Control (IFAC) Workshop, Sept. 1998), 1999.

[45] M.-Y. Wu, W. Shu, and H. Zhang. Segmented min-min: A static mapping algorithm for meta-tasks on heterogeneous computing systems. In *9th IEEE Heterogeneous Computing Workshop (HCW 2000)*, pages 375–385, May 2000.

[46] J. Xu and D. L. Parnas. Scheduling processes with release times, deadlines, precedence, and exclusion relations. *IEEE Transactions on Software Engineering*, 16(3):360–369, Mar. 1990.

[47] W. Zhao, K. Ramamritham, and J. A. Stankovic. Preemptive scheduling under time and resource constraints. *IEEE Transactions on Computers*, C-36(8):949–960, Aug. 1987.

Biographies

Shoukat Ali is pursuing a Ph.D. degree from the School of Electrical and Computer Engineering at Purdue University, where he is currently a Research Assistant. His main research topic is resource management in heterogeneous computing systems. He has held teaching positions at Aitchison College and Keyneysian Institute of Management and Sciences, both in Lahore, Pakistan. He was also a Teaching Assistant at Purdue. Shoukat received his MS degree in electrical engineering from Purdue University in 1999. His master's thesis was "A Comparative Study of Dynamic Mapping Heuristics for a Class of Independent Tasks onto Heterogeneous Computing Systems." He received his BS degree in electrical and electronic engineering from the University of Engineering and Technology, Lahore, Pakistan in 1996. His research interests include computer architecture, parallel computing, and heterogeneous computing. He has authored or coauthored ten technical papers in these areas. He is a member of the IEEE, ACM, and IEEE Computer Society.

Jong-Kook Kim is pursuing a Ph.D. degree from the School of Electrical and Computer Engineering at Purdue University, where he is currently a Research Assistant (from August 1998). Jong-Kook received his MS degree in electrical engineering from Purdue University in 2000. His master's thesis was " A Multi-Dimensional Performance Measure for Distributed Computing and Communication Systems." He received his B.S. degree in electronic engineering from Korea University, Seoul, Korea in 1998. His research interests include computer architecture, performance measure, resource management, and heterogeneous computing. He is a student member of the IEEE and IEEE Computer Society.

Yang Yu is a Doctoral Candidate at the Department of Electrical Engineering, University of Southern California, where he has been working as a research assistant since August 2000. Mr. Yu received both his B.Eng. and M. Eng. degrees of Computer Science from Shanghai Jiao Tong University in China. His research interests include heterogeneous computing, real-time systems, and computer architecture.

Shriram Bhargava Gundala is pursuing a Masters degree from the School of Electrical Engineering at the University of Southern California, where he is currently a Research Assistant. Shriram received his Bachelors degree in electronics and communications engineering from Andhra University, Visakhapatnam, India in 2000. His research interests include VLSI circuit design, ASIC design, computer architecture, and heterogeneous computing. He has authored or coauthored three technical papers in these areas.

Sethavidh Gertphol is pursuing a Ph.D. degree from the School of Electrical Engineering - Systems at the University of Southern California, where he is currently a Research Assistant. He received his M.S. degree in electrical engineering from University of Southern California in 1999 and B.Eng. degree in electrical engineering from Chulalongkorn University, Bangkok, Thailand in 1996. His main research topic is resource allocation in real-time heterogeneous systems. He has authored and co-authored three technical papers in the area.

Howard Jay Siegel holds the endowed chair position of Abell Distinguished Professor of Electrical and Computer Engineering at Colorado State University (CSU), where he is also a Professor of Computer Science. From 1976 to 2001, he was a Professor in the School of Electrical and Computer Engineering at Purdue University.

At CSU, Prof. Siegel is Chair of the Strategic Planning Committee for the university-wide Virtual College of Information Science & Technology. He teaches courses on computer architecture and heterogeneous computing

Prof. Siegel received a B.S. degree in electrical engineering and a B.S. degree in management from the Massachusetts Institute of Technology (MIT), and the M.A., M.S.E., and Ph.D. degrees from the Department of Electrical Engineering and Computer Science at Princeton University. He is a Fellow of the IEEE and a Fellow of the ACM. He is an international keynote speaker and tutorial lecturer, as well as a consultant for government and industry.

Prof. Siegel has co-authored over 290 published technical papers on parallel and distributed computing. His research interests include heterogeneous parallel and distributed computing, communication networks routing and security, parallel algorithms, parallel machine interconnection networks, and reconfigurable parallel computer systems.

Prof. Siegel was a Coeditor-in-Chief of the Journal of Parallel and Distributed Computing, and has been on the Editorial Boards of the IEEE Transactions on Parallel and Distributed Systems and the IEEE Transactions on Computers. He was Program Chair/Co-Chair of three major international conferences, General Chair/Co-Chair of four international conferences, and Chair/Co-Chair of four workshops.

Prof. Siegel was an "IEEE Computer Society Distinguished Visitor" and an "ACM Distinguished Lecturer," giving invited seminars about his research around the country. He is a member of the Eta Kappa Nu electrical engineering honor society, the Sigma Xi science honor society, and the Upsilon Pi Epsilon computing sciences honor society.

Anthony A. Maciejewski received the B.S.E.E, M.S., and Ph.D. degrees in Electrical Engineering in 1982, 1984, and 1987, respectively, all from The Ohio State University under the support of an NSF graduate fellowship. From 1985 to 1986 he was an American Electronics Association Japan Research Fellow at the Hitachi Central Research Laboratory in Tokyo, Japan where he performed work on the development of parallel processing algorithms for computer graphic imaging. From 1988 to 2001, he was a Professor of Electrical and Computer Engineering at Purdue University.

In 2001, he joined Colorado State University as a Professor of Electrical and Computer Engineering.

Prof. Maciejewski's primary research interests relate to the analysis, simulation, and control of robotic systems and he has co-authored over 100 published technical articles in these areas. He is an Associate Editor for the *IEEE Transactions on Robotics and Automation*, a Regional Editor for the journal *Intelligent Automation and Soft Computing*, and was co-guest editor for a special issue on "Kinematically Redundant Robots" for the *Journal of Intelligent and Robotic Systems*. He serves on the IEEE Administrative Committee for the Robotics and Automation Society and was the Program Co-Chair (1997) and Chair (2002) for the International Conference on Robotics and Automation, as well as serving as the Chair and on the Program Committee for numerous other conferences.

Viktor K. Prasanna (V. K. Prasanna Kumar) received his BS in Electronics Engineering from the Bangalore University and his MS from the School of Automation, Indian Institute of Science. He obtained his Ph.D. in Computer Science from the Pennsylvania State University in 1983. Currently, he is a Professor in the Department of Electrical Engineering as well as in the Department of Computer Science at the University of Southern California, Los Angeles. He is also an associate member of the Center for Applied Mathematical Sciences (CAMS) at USC. He served as the Division Director for the Computer Engineering Division during 1994-98. His research interests include parallel and distributed computation, computer architecture, VLSI computations, and high performance computing for signal and image processing, and vision. Dr. Prasanna has published extensively and consulted for industries in the above areas. He has served on the organizing committees of several international meetings in VLSI computations, parallel computation, and high performance computing. He is the Steering Co-chair of the International Parallel and Distributed Processing Symposium [merged IEEE International Parallel Processing Symposium (IPPS) and the Symposium on Parallel and Distributed Processing (SPDP)] and is the Steering Chair of the International Conference on High Performance Computing (HiPC). He serves on the editorial boards of the Journal of Parallel and Distributed Computing, IEEE Transactions on Computers, and the IEEE Transactions on Parallel and Distributed Systems. He was the founding Chair of the IEEE Computer Society Technical Committee on Parallel Processing (TCPP). He is a Fellow of the IEEE.