Characterizing Resource Allocation Heuristics for Heterogeneous Computing Systems

SHOUKAT ALI

Department of Electrical and Computer Engineering,
University of Missouri–Rolla, Rolla, MO 65409-0040, USA
shoukat@umr.edu

TRACY D. BRAUN

3rd Dental Battalion, US Naval Dental Center, Okinawa, Japan
tdbraun@computer.org

HOWARD JAY SIEGEL[1] AND
ANTHONY A. MACIEJEWSKI

Colorado State University, Electrical and Computer Engineering
Department, Fort Collins, CO 80523-1373, USA
hj@colostate.edu
aam@colostate.edu

NOAH BECK

Advanced Micro Devices, MS 83-29, 90 Central St.,
Boxboro, MA 01719, USA
noah.beck@amd.com

LADISLAU BÖLÖNI

Department of Computer Engineering,
University of Central Florida, Orlando, FL 32816-2362, USA
lboloni@cpe.ucf.edu

MUTHUCUMARU MAHESWARAN

School of Computer Science, McGill University,
Montreal, QC H3A 2A7 Canada
maheswar@cs.mcgill.ca

ALBERT I. REUTHER[2]

School of Electrical and Computer Engineering,
Purdue University, West Lafayette, IN 47907-1285, USA
reuther@ecn.purdue.edu

JAMES P. ROBERTSON

NVIDIA Corp, 12331 Riata Trace Pkwy, Austin, TX 78727, USA
jrobertson@nvidia.com

MITCHELL D. THEYS

Department of Computer Science (MC 152), University of Illinois
at Chicago, Chicago, IL 60607-7053, USA
mtheys@uic.edu

BIN YAO

Networking Platforms Research Department, Bell Laboratories,
Holmdel, New Jersey 07733, USA
byao@lucent.com

Abstract

In many distributed computing environments, collections of applications need to be processed using a set of heterogeneous computing (HC) resources to maximize some performance goal. An important research problem in these environments 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 constraints. This process of matching and scheduling is called mapping.

[1]Howard Jay Siegel holds a joint appointment in the Computer Science Department as well.
[2]Albert I. Reuther is currently with MIT Lincoln Laboratory, Lexington, MA.

To make meaningful comparisons among mapping heuristics, a system designer needs to understand the assumptions made by the heuristics for (1) the model used for the application and communication tasks, (2) the model used for system platforms, and (3) the attributes of the mapping heuristics. This chapter presents a three-part classification scheme (*3PCS*) for HC systems. The 3PCS is useful for researchers who want to (a) understand a mapper given in the literature, (b) describe their design of a mapper more thoroughly by using a common standard, and (c) select a mapper to match a given real-world environment.

1. Introduction

In today's interconnected world, many industry, laboratory, and military sites each use a shared set of networked computers of different types and ages. In such environments, there are large applications or collections of applications to be processed. The workload can be distributed over the set of heterogeneous computing resources to maximize some performance goal. To accomplish this, the system administrator or designer must include some method for determining which application tasks to assign to different machines in the network to exploit effectively the heterogeneity of the system platform. Such methods are applicable to the allocation of resources for many different types of computing and communication tasks in many types of environments, including parallel, distributed, cluster, grid, Internet, embedded, wireless, and reconfigurable systems.

The system designer may go to the literature to find information about techniques for assigning and scheduling (collectively referred to as "mapping") tasks in a network of heterogeneous machines. For example, in the literature, the system designer may find a technique A that is claimed to be better than another technique B. However, further investigation may show that A can only map independent (non-communicating) tasks while B can map both independent tasks and communicating subtasks within tasks. Thus, A may not be appropriate for the designer's system if the tasks therein include communicating subtasks (a workload model issue).

The system designer may find a third technique C that also handles communicating subtasks and is claimed to be better than B. It may be the case that C can incorporate information about both network contention and network link bandwidths to estimate communication times, whereas B (probably unreasonably) assumes a "fully connected" network, and is thus unable to use any contention information to estimate communication times. If a simulation study compares B and C for a system platform that is fully connected (i.e., no contention), it may turn out that B generates better mappings. However, if B and C are compared for a system platform that is not fully connected, we may find that B is not better than C, probably because B is not accounting for the network contention (a platform model issue).

Another important issue is the performance metric that the mapper tries to maximize. If both C and a fourth mapping technique D are appropriate for a particular HC environment, then different criteria may be used to determine which technique is "better." For example, if the performance metric is minimizing the average task execution time (not including the time waiting to begin execution), we might choose the technique (say D) which tries to select the fastest machine for each task. However, if the performance metric is minimization of the overall completion time of the workload, we may be more interested in selecting the technique (say C) that, for a given task, will choose an alternate machine if the fastest machine has a large queue of tasks waiting to execute (a mapping strategy issue).

Figure 1 is an illustration of the procedure used above to select a mapping heuristic based on information about the workload model, platform model, and mapping strategy. The figure uses the heuristics A, B, C, and D mentioned above.

From the above discussion, we observe that to make meaningful comparisons among mapping heuristics, a system designer needs to understand (a) the model used for workload, i.e., applications, (b) the model used for system platforms, and (c) the attributes of the mapping strategies. To accomplish this, we present a three-part classification scheme (*3PCS*) for heterogeneous computing systems (Figure 2). The 3PCS builds on the previous work done in [24,29,30,44,50,51,62,66,75,80]. All three parts of the 3PCS are needed to determine which resource allocation heuristics would be most appropriate for a given environment, and also to facilitate fair comparisons among different heuristics. We now describe some terms related to heterogeneous computing systems to set the stage for describing the 3PCS.

Performing computing and communication tasks on parallel and distributed *heterogeneous computing* (*HC*) systems involves the coordinated use of different types of machines, networks, interfaces, and other resources (e.g., [5,31,36,72]). An HC system may be used to perform a variety of different tasks that have diverse computational requirements. The resources should be allocated to the tasks in a way that

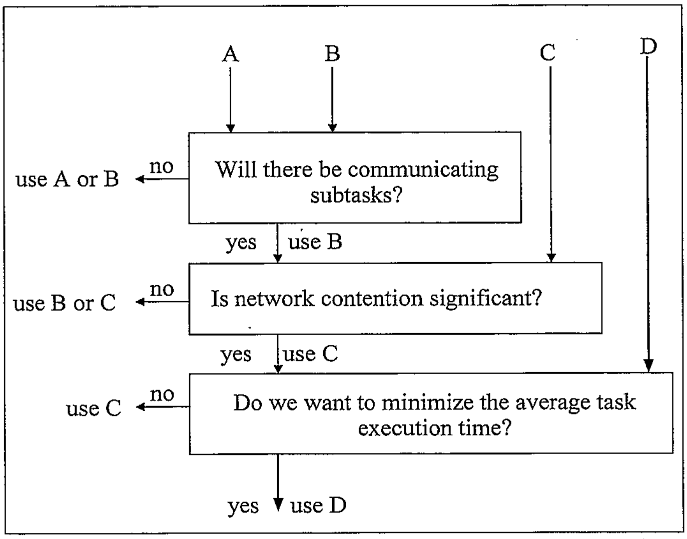
A
B
C
D
Will there be communicating subtasks?
no
use A or B
yes
use B
Is network contention significant?
no
use B or C
yes
use C
Do we want to minimize the average task execution time?
no
use C
yes
use D

FIG. 1. The choice of a mapping technique for a given system depends on the characteristics of the workload model, the characteristics of the platform model, and the mapping strategy. In this example, A, B, C, and D are four different mapping techniques discussed in the text.

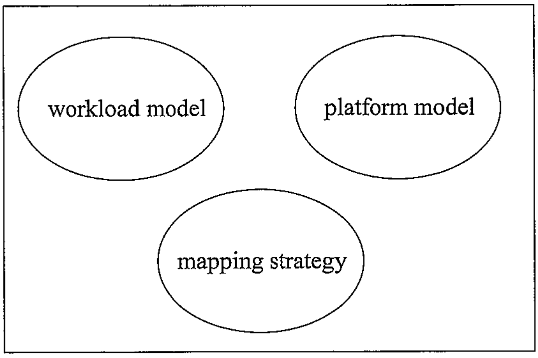

FIG. 2. The three parts of the 3PCS: workload model, platform model, and the mapping strategy.

exploits the heterogeneity to maximize some system performance measure. A cluster composed of machines of different ages and types is an example of an HC system. Alternatively, a cluster could be treated as a single machine in an HC suite. An HC system could also be part of a larger computational grid [34].

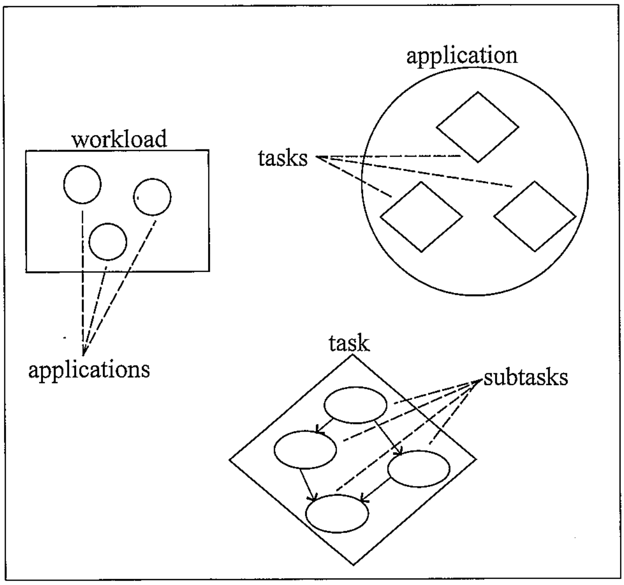

FIG. 3. The workload for a system consists of applications. Applications may be decomposed into tasks. Tasks may, in turn, be decomposed into two or more communicating subtasks. Some tasks may not have any subtasks.

The *workload* to be processed on an HC system consists of one or more *applications* (Figure 3). An application is assumed to be composed of one or more independent (i.e., non-communicating) *tasks*. It is also assumed that some tasks may be further decomposed into two or more communicating *subtasks*. The subtasks have data dependencies among them, but can be assigned to different machines for execution. If there are communicating subtasks within an application, inter-machine data transfers need to be performed when such subtasks are assigned to different machines.

A key factor in achieving the best possible performance from HC environments is the ability to effectively and efficiently *match* (assign) the applications to the machines and *schedule* (order the execution of) the applications on each machine (as well as ordering any inter-machine communications). The matching and scheduling of applications is defined as *mapping*. The mapping problem is also called resource management (or resource allocation) and a mapper is also called a resource management system.

The problem of mapping applications and communications onto multiple machines and networks in an HC environment has been shown, in general, to be NP-complete [33], requiring the development of near-optimal heuristic techniques. In recent years, numerous studies have been conducted on the mapping problem and mapping heuristics (e.g., [1,9,10,12–14,16,17,19,20,31,37,49,59,63,65,81,84,87]).

A mapper can be used in different ways. For example, we can use it to optimize a performance metric given a particular configuration of the system (i.e., particular sets of machines, networks, and protocols). Conversely, we can also use a mapper to optimize the operational *cost* of the system that is needed to reach a target value of performance. For example, a mapper could be used to select particular sets of machines, networks, and protocols that meet a certain level of performance while minimizing the dollar cost of the system (e.g., [27,64]).

The 3PCS for HC systems is useful for understanding a mapper. We propose that when one studies a mapper given in the literature, one should try to "check" the mapper in the context of the 3PCS. One should ask the questions raised in the 3PCS regarding the characteristics of the workload model, the platform model, and the mapping strategy.

The 3PCS is also useful for researchers who want to describe their mapper more thoroughly by using a common standard. When describing a mapper, one should indicate the appropriate values for all features in the three parts of the 3PCS. In this regard, the 3PCS may also help researchers see design and environment alternatives that they might not have otherwise considered during the development of new heuristics.

Additionally, the 3PCS is useful when one wants to select a mapper to match a given real-world environment. One can use the 3PCS to characterize the environment in terms of the platform model, workload model, and the mapping strategy. Then one can characterize different available mappers using the 3PCS, and select one that is most appropriate for the given environment (possibly with some modification).

Lastly, in the future, the 3PCS for HC systems could focus research towards the development of a standard set of benchmarks for evaluating resource allocation heuristics for HC environments. Classes for benchmarks could be defined based on specified workload, platform, and mapping strategy characteristics.

The work in this chapter was supported in part by the DARPA Quorum Program project *MSHN* (Management System for Heterogeneous Networks), by the DARPA *AICE* (Agile Information Control Environment) Program, and by the DARPA *BADD* (Battlefield Awareness and Data Dissemination) Program. One technical objective of the MSHN project was to design, prototype, and refine a mapping system as part of a distributed resource management system that leverages the heterogeneity of resources and workload to deliver the requested qualities of service [20,59]. One aspect of the AICE and BADD programs involved designing a scheduling system for forwarding data items prior to their use as inputs to a local application in a wide area network distributed computing environment [79,78]. The AICE, BADD, and MSHN

environments are similar in that, in some situations, not all applications or communication requests can be completed with their most preferred qualities of service by their deadlines. Thus, the goal of the mapper in these environments is to satisfy a set of application tasks or communication requests in a way that has the greatest collective perceived value. The 3PCS pertains to both the task environment in MSHN, and the communication request environment in BADD and AICE. In some cases, an application mentioned in this chapter may also refer to a communication request in the AICE and BADD context.

The rest of the chapter is organized as follows. Section 2 describes the 3PCS. Section 3 characterizes six heuristics from the literature in terms of the 3PCS. Section 4 gives an overview of some related taxonomy studies. A summary of the chapter is presented in Section 5.

2. Proposed Characterization Scheme

2.1 Overview

A "mixed-machine" HC system is composed of different machines, with possibly multiple execution models (as in the *MEMM* classification [30]). Such a system is defined to be *heterogeneous* if any features vary among machines enough to result in different execution performance among those machines. Such features could be processor type, processor speed, mode of computation, memory size, cache structure, number of processors (within parallel machines), inter-processor network (within parallel machines), etc.

The proposed 3PCS for describing mapping heuristics for mixed-machine HC systems is defined by three major components: (1) workload model, (2) platform model, and (3) mapping strategy. To properly analyze and compare mapping heuristics for current and future HC environments, information about all three parts is needed.

The 3PCS attempts to *qualitatively* define aspects of the environment that can affect mapping decisions and performance. (Doing this *quantitatively* in a thorough, rigorous, complete, and uniform manner is a long term goal of the HC field.) The 3PCS design was based on the existing mapping heuristic literature, as well as our group's previous research and experience in the field of HC. Each part of the scheme can, of course, be investigated in more detail.

2.2 Workload Model Characterization

The first category of the 3PCS defines the model used for the workload characterization. In our context, workload characterization is limited to defining application computational and communication characteristics that may impact mapping decisions and relative mapper performance. This workload characterization defines a model for the applications to be executed on the HC system, and for the communications to be scheduled on the inter-machine network. Furthermore, the 3PCS includes application traits that may not be realistic, but do correspond to assumptions a given researcher may have made when designing and analyzing mapping heuristics in the literature. Typically, such assumptions are made to simplify the mapping problem in some way. For example, many researchers assume a given subtask must receive all of its input data from other subtasks before it can begin executing, when in reality the subtask may be able to begin with only a subset of data. As another example, some researchers may assume that the workload is divisible into arbitrary size portions that could be allocated to different machines [81]. The goal of the 3PCS is to reflect the environment assumed by the mapping heuristic, so that the workload model can capture any assumptions made (even if they are unrealistic). The defining traits of such an application model are explained below and illustrated in the organization chart given in Figure 4.

workload composition: Does the workload consist of any communicating entities, or, in other words, do the tasks within a given application have (communicating) subtasks? A workload that consists of (non-communicating) tasks only is sometimes termed as a "bag-of-tasks" or "meta-task" (e.g., Braun et al. [20], Maheswaran et al. [59]) and has a less complicated mapping problem associated with it.

Are the applications continuously executing, as in some real-time systems (e.g., [6,7])? Continuously executing applications are different from "one-shot" applications that process just one data set and then stop executing in that the former are usually part of a sensor-fed computing system. For example, the system in [6] consists of heterogeneous sets of sensors, subtasks, machines, and actuators. Each sensor produces data periodically at a certain rate, and the resulting data streams are input into subtasks. The subtasks process the data and send the output to other subtasks or to actuators (see Figure 5). Some of the items mentioned later in this characterization scheme apply only to the case where the workload contains communicating subtasks (whether continuously executing or "one-shot"). Such items will be indicated with the letter "S" written next to them.

Note that the mapping problem for a system that consists entirely of continuously executing subtasks often reduces to a problem of allocating subtasks to machines, i.e., scheduling is not involved. For example, in [6], each machine is capable of multi-tasking, executing the applications assigned to it in a round robin fashion. Similarly, a given network link is multi-tasked among all data transfers using that link.

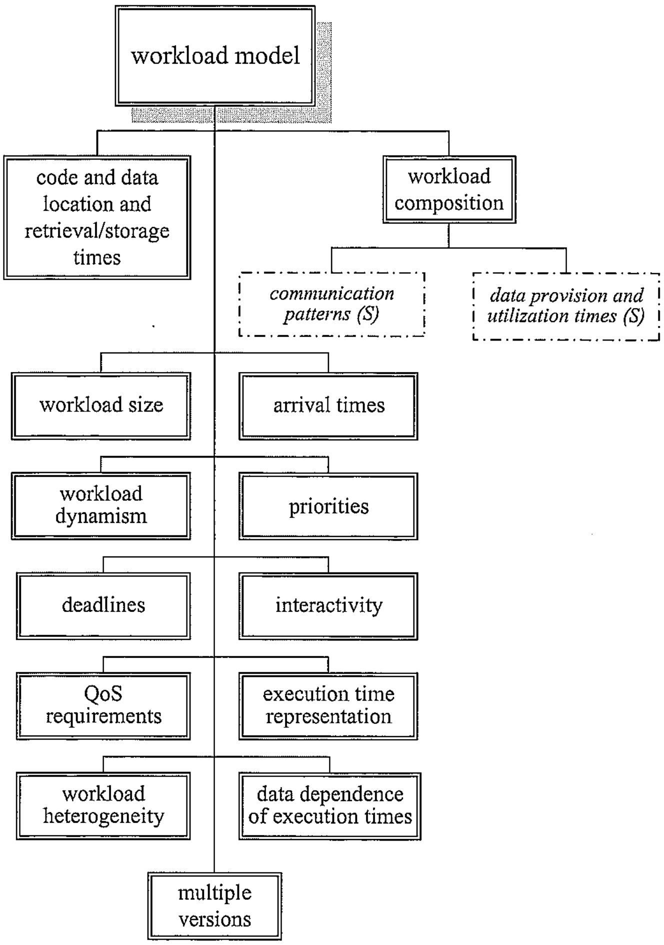
workload model
code and data location and retrieval/storage times
workload composition
communication patterns (S)
data provision and utilization times (S)
workload size
arrival times
workload dynamism
priorities
deadlines
interactivity
QoS requirements
execution time representation
workload heterogeneity
data dependence of execution times
multiple versions

FIG. 4. The different features of the workload model. The items "communication patterns" and "data provision and utilization times" are shown under "workload composition" because these items are applicable only if the workload is composed of communicating subtasks.

communication patterns (S): Does the application have any particular data communication pattern with respect to the source and destination subtasks for each data item to be transferred? Can the workload be represented as a directed acyclic graph, or a graph with loops, possibly including a set of inter-communicating coroutines? For example, assume that for a given application with n subtasks, $n - 1$ subtasks communicate only with a "master" subtask, and otherwise do not com-

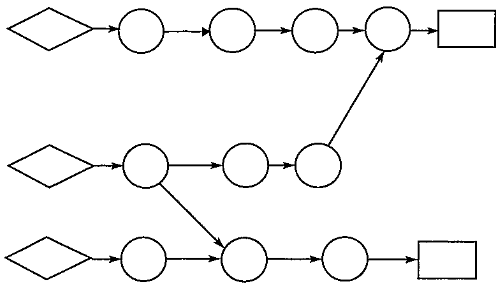

FIG. 5. The subtasks (denoted by "circles") in this system are continuously executing to process the data streams being generated by the sensors (denoted by "diamonds"). The output from subtasks is used to control the actuators (denoted by "rectangles").

municate with each other. Then, the communication times among subtasks can be reduced much more by mapping this application on a parallel machine with a "star" network topology than a machine with a hypercube network topology.

data provision and utilization times (S): Can a source subtask release data to consumer subtasks before it completes? Can a consumer subtask begin execution before receiving all of its input data? For example, the clustering non-uniform directed graph heuristic [32] assumes that a consumer subtask has to wait for the completion of the parent subtask if there is data dependency. The time at which input data needed by a subtask or output data generated by a subtask can be utilized may vary in relation to subtask start and finish times, and can help the mapper overlap the execution of inter-dependent subtasks.

code and data location and retrieval/storage times: Do tasks require data from special servers? Are data retrieval and storage times considered? Is the time required to fetch task code and initial data, as well as storing the final results of a task, considered part of the task execution time?

workload size: If a mapping heuristic was evaluated by simulation or experimentation, what size workload was used? The workload size is quantified by

(a) the number of "mappable" entities (tasks and subtasks) present in the workload (often considered relatively to the number of machines in the HC suite),
(b) the average size of "service demand," e.g., average machine utilization required by the mappable entities, average computation time, and
(c) the rate of arrival, i.e., the rate at which the mappable entities become available for assignment.

The workload size for which a given heuristic is evaluated can impact the relative performance of different heuristics for a given metric. For example, research in [59] shows that a certain class of mapping heuristics performs increasingly better than another class for larger workload sizes. Another example is given in the discussion for workload heterogeneity below.

workload dynamism: Is the complete workload to be mapped known *a priori* (static workload), or do the applications arrive in a real-time, non-deterministic manner (dynamic workload), or is it a combination of the two? Depending on this classification of the workload, we may need a "static" or a "dynamic" mapping strategy (defined in the mapping strategy characterization).

arrival times: Are arrival times for applications taken from a real system, or generated artificially for simulation studies? When taken from a real system, arrival times will be known exactly. For simulation studies, arrival times can be sampled from an anticipated probability distribution.

deadlines: Do the applications have deadlines? This property could be further refined into soft and hard deadlines, if required (e.g., [23,48]). Applications completed by a soft deadline provide the most valuable results. An application that completes after a soft deadline but before a hard deadline is still able to provide some useful data. After a hard deadline has passed, the output from the application is useless. The "worth" of the execution of a task may decrease as its completion time increases from the soft to hard deadline.

priorities: Do the applications have priorities (e.g., [48])? Environments that would require priorities include military systems and machines where time-sharing must be enforced. Priorities are generally assigned by the user (within some allowed range), but the relative weights given to each priority are usually determined by another party (e.g., a system administrator). Priorities and their relative weights are especially important if the mapping strategy is preemptive (defined in the mapping strategy characterization).

multiple versions: Do the applications have multiple versions, with different resource requirements, that could be executed (e.g., [23,70])? If yes, what are the relative "values" to the user of the different versions? Many applications have different versions. For example, an application that requires a Fast Fourier Transform might be able to perform the Fast Fourier Transform with either of two different procedures that have different precisions, different execution times, and different resource requirements.

QoS requirements: Do any applications specify any Quality of Service (QoS) requirements (other than deadlines and priorities mentioned above)? Most QoS requirements, like security, can affect mapping decisions (e.g., not mapping a particular application onto a machine of the wrong security classification).

interactivity: Are applications user-interactive (i.e., do they depend on real-time user input)? Such applications must be executed on machines for which the user has access or clearance.

workload heterogeneity: For each machine in the HC suite, how greatly and with what properties (e.g., probability distribution (e.g., [11])) do the execution times of the different tasks (or subtasks) vary for any given machine? For subtasks, the above question applies to message sizes as well.

Workload heterogeneity can affect the failure rate of a heuristic (failure rate is defined in the mapping strategy characterization). It has been shown in [4] that, for a particular system where the goal was to design static resource allocation heuristics that balance the utilization of the computation and network resources, an increase in workload heterogeneity increased the difference in failure rates among some of the heuristics being considered.

Workload heterogeneity also can impact the suitability of a performance metric for a given system. For example, the research in [4] shows that the need to measure system robustness increases as the system "complexity" increases. Based on [57,58,67,76], system complexity is a function of the system heterogeneity and the size. Figure 6 shows how the graph of robustness against slack changes in appearance as the system complexity increases. Slack is a proxy measure of robustness, and is simpler to calculate than the robustness measure introduced in [4]. The underlying systems for the first two graphs (from the left) are identical in size but different in workload heterogeneity. The third system (right-most) has the same heterogeneity as the second system but is bigger in size. It can be seen that for the first system, robustness is tightly correlated with slack. For this system, there is almost no need to calculate robustness values because the slack values can be used to approximate robustness. For the second system, however, the need to measure robustness explicitly increases. For the third system, the need is even more acute.

execution time representation: How are the estimated execution times of tasks and subtasks on each of the different machines in the HC suite determined? Most mapping techniques require an estimate of the execution time of each task and subtask on each machine (e.g., [53,85,86]).

The two choices most commonly used for making these estimates from historic or direct information are deterministic and distribution modeling. Deterministic modeling uses a fixed (or expected) value (e.g., [35]), e.g., the average of ten previous executions of a task or subtask. Alternatively, the application developer or the application user could specify the estimated execution time of the applications' tasks and subtasks on each machine in the suite. Distribution modeling statistically processes historic knowledge to arrive at a probability distribution for task or subtask execution times. This probability distribution is then used to make mapping decisions (e.g., [12,55]).

The execution time for a given application may be determined by task profiling. Task profiling specifies the types of computations present in an application based on the code for the task (or subtask) and the data to be processed (e.g., [38,60]).

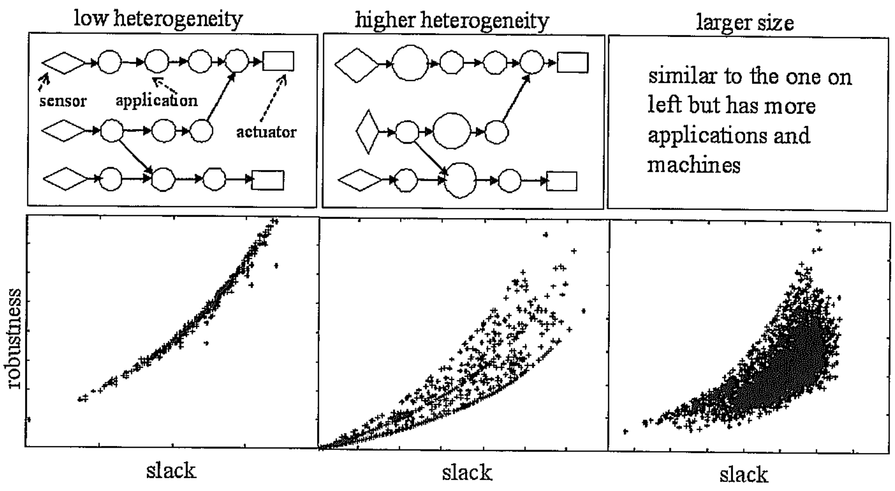

FIG. 6. The need to measure the system robustness increases as the system complexity increases. For the system with small heterogeneity, the robustness and slack are tightly coupled, thereby suggesting that robustness measurements are not needed if slack is known. As the system heterogeneity increases, the robustness and slack become less correlated, indicating that the robustness measurements can be used to distinguish between mappings that are similar in terms of the slack. As the system size increases, the correlation between the slack and the robustness decreases even further.

This information may be used by the mapping heuristic, in conjunction with analytical benchmarking (defined in the platform model characterization), to estimate task or subtask execution time.

In simulation studies, estimated execution times can be derived from probability distributions with a given mean and a given degree of heterogeneity (e.g., Ali et al. [11]).

The same issues apply to subtask communications (e.g., [79]).

data dependence of execution times: Is the execution time of a given application on a given machine independent of the data content of the application inputs (e.g., in image smoothing) or is it data dependent (e.g., in object recognition) (e.g., [73])?

2.3 Platform Model Characterization

The second category of the 3PCS defines the models used for target platforms available within HC systems. The target platform traits listed are those that may impact mapping decisions and relative mapper performance. The target platform is defined by the hardware, network properties, and software that constitute the

HC suite. Several existing heuristics make simplifying (but unrealistic) assumptions about their target platforms (e.g., assuming that an infinite number of machines are available [74]). Therefore, the 3PCS is not limited to a set of realistic target platforms. Instead, a framework for classifying the *models* used for target platforms is provided. Such a framework allows the 3PCS to reflect the environment assumed by a mapping heuristic (even if the environment is unrealistic). The defining traits of the platform model are explained below and illustrated in the organization chart given in Figure 7.

number of machines: Is the number of machines finite or infinite? Is the number of machines fixed or variable (e.g., new machines can come on-line)? Furthermore, a given heuristic may treat a finite, fixed number of machines as a parameter that can be changed from one mapping to another, e.g., when trying to find a minimum dollar-cost set of machines to meet a performance requirement (e.g., [27,64]).

system control: Does the mapping strategy control and allocate all resources in the environment (dedicated), or are external users also consuming resources (shared)?

application compatibility: Is each machine in the environment able to perform each application, or, for some applications, are there any special capabilities that are only available on certain machines? These capabilities could involve issues such as database software, I/O devices, memory space, and security.

machine heterogeneity and architecture: For each task or subtask, how greatly and with what properties (e.g., probability distribution (e.g., [11])) do the execution times vary across different machines in the HC suite? For subtasks, the above question applies to communication link speeds as well. For each machine, various architectural features that can impact performance must be considered, e.g., processor type, processor speed, external I/O bandwidth, mode of computation (e,g., shared memory, distributed shared memory, NUMA, UMA), memory size, number of processors (within parallel machines), and inter-processor network (within parallel machines). The machines in the HC suite may be evaluated on analytical benchmarks to aid in later estimating task or subtask execution times. Analytical benchmarking provides a measure of how well each available machine in the HC platform performs on each given type of computation [60]. This information may be used by the mapping heuristic, in conjunction with task profiling, to estimate task or subtask execution times. Machine heterogeneity may affect the performance of a mapping heuristic in some cases. For example, it is shown in [39] that a particular algorithm, CDA, is outperformed by another algorithm, PSP, when resource heterogeneity is increased.

code and data access and storage times: How long will it take each machine to access the code and input data it needs to execute a given task or subtask? How long will it take each machine to store any resulting output data for a given task or

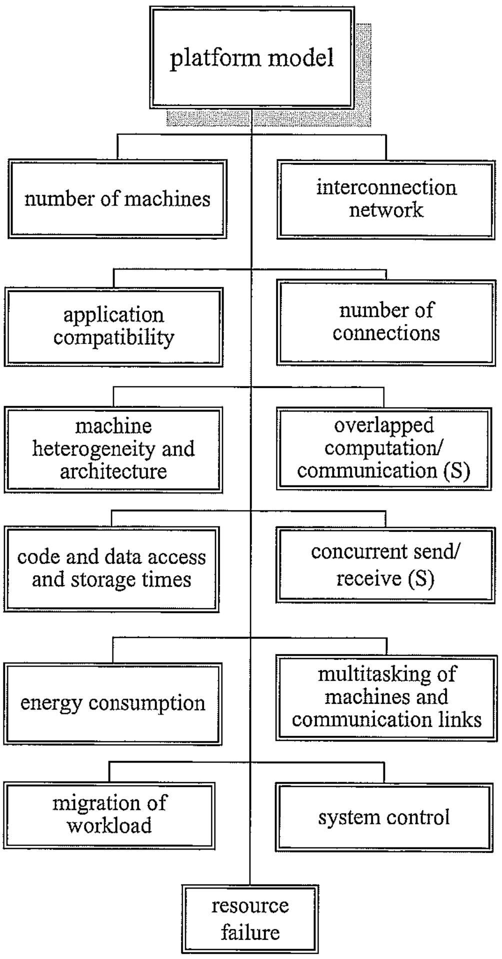
platform model
number of machines
interconnection network
application compatibility
number of connections
machine heterogeneity and architecture
overlapped computation/ communication (S)
code and data access and storage times
concurrent send/ receive (S)
energy consumption
multitasking of machines and communication links
migration of workload
system control
resource failure

FIG. 7. The different features of the platform model. The items marked with an "S" next to them are only applicable when the workload contains communicating subtasks.

subtask? For subtasks, the above questions do not apply to communication activity to/from another subtask.

interconnection network: What are the various properties of the inter-machine network? Many network characteristics can affect mapping decisions and system performance, including bandwidth, latency, switching control, and topology. Most of these network properties are also functions of the source and destination ma-
)

chines. Volumes of literature (e.g., [26,28,56,71]) already exist on the topic of interconnection networks, therefore, interconnection networks are not classified here.

number of connections: How many connections does each machine have to the interconnection network structure or directly to other machines?

concurrent send/receive (S): Can each machine perform concurrent sends and receives of data to other machines (assuming enough network connections)?

overlapped computation/communication (S): Can machines overlap computation and inter-machine communication?

energy consumption: How is consumption of energy in battery-based systems calculated? At what rate is energy consumed when performing computation, sending data, receiving data, or when system is idle (e.g., [41,46,69,70])? Does the system allow conservation of energy by reducing the clock speed (e.g., [46])? How many different voltage levels are allowed and how does each level impact clock rate and energy consumption (e.g., [83])? These questions are especially relevant for *ad hoc* grid computing systems, where some of the devices are mobile, battery-based, and use wireless communications (e.g., [61,69]).

multitasking of machines and communication links: Does the system allow machines and communications links to be multi-tasked (e.g., [10])? If so, how does the multitasking impact the execution time of tasks and subtasks, and the communication time of subtasks?

migration of workload: Do the machines support the migration of applications, tasks, or subtasks? Migration may be used by a "dynamic" mapping strategy (explained in the mapping strategy characterization) to re-map an application (or a task or a subtask) from a machine that has failed or is overloaded to some other suitable machine. Migration affects the communication patterns among subtasks, and may reduce the advantage of any mapping decision based on pre-migration communication patterns.

resource failure: Is the failure of machines, links, storage devices, and other resources modeled?

2.4 Mapping Strategy Characterization

The third category of the 3PCS defines the characteristics used to describe mapping strategies. Because the general HC mapping problem is NP-complete, it is assumed that the mapping strategies being classified are heuristics that attempt to produce near-optimal mappings. The different features of the mapping strategy characterization are explained below and illustrated in the organization chart given in Figure 8.

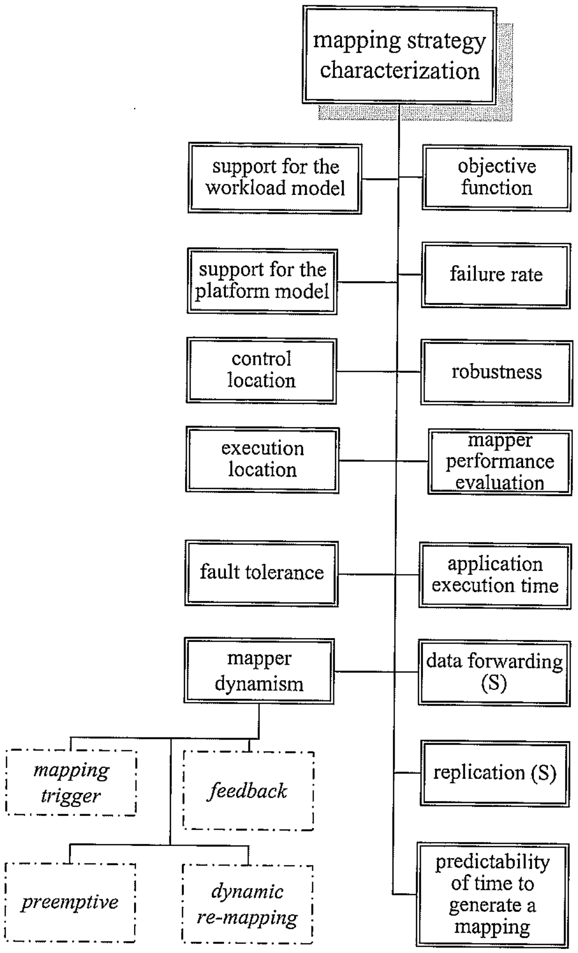
mapping strategy characterization
support for the workload model
objective function
support for the platform model
failure rate
control location
robustness
execution location
mapper performance evaluation
fault tolerance
application execution time
mapper dynamism
data forwarding (S)
mapping trigger
feedback
replication (S)
preemptive
dynamic re-mapping
predictability of time to generate a mapping

FIG. 8. The different features of the mapping strategy characterization. The items marked with an "S" next to them are only applicable when the workload contains communicating subtasks. The items in dotted boxes under "mapper dynamism" are only applicable for a dynamic or on-line mapping strategy.

support for the workload model: Can the mapping strategy use information about a given trait of the application (as modeled in the section on workload model characterization)? For example, a mapping strategy that cannot make use of the fact that a given task has multiple versions may be outperformed by one that does use this information in making mapping decisions.

support for the platform model: Can the mapping strategy use information about a given trait of the platform (as modeled in the section on platform model characterization)? For example, can the mapping strategy take advantage of any support that the platform provides for the migration of subtasks or tasks?

control location: Is the mapping strategy centralized or distributed? Distributed strategies can further be classified as cooperative or non-cooperative (independent) approaches.

execution location: Can a machine within the suite be used to execute the mapping strategy, or is an external machine required?

fault tolerance: Is fault tolerance considered by the mapping strategy? This may take several forms, such as assigning applications to machines that can perform checkpointing, or executing multiple, redundant copies of an application [1].

objective function: What quantity is the mapping strategy trying to optimize? Are there associated QoS constraints, e.g., minimizing average energy used by a mobile device while still completing the application in a given time (e.g., [69])? This varies widely among strategies, and can make some approaches inappropriate in some situations. The objective function, i.e., performance metric, can be as simple as the total execution time for the workload, or a more complex function that includes priorities, deadlines, QoS, etc. [47].

failure rate: What is the failure rate for the mapping heuristic? Before a mapping heuristic is employed in a real system, one would often evaluate its performance using simulations. In such a case, an important property of a mapping heuristic is its failure rate. A mapping heuristic failure occurs if the heuristic cannot find a mapping that allows the system to meet its QoS constraints (e.g., fails to find a resource allocation that completes a set of applications within a requested time constraint). One way of determining the failure rate is to repeat the simulation for a number of trials, where each trial uses the same probability distributions for simulation parameters (but re-samples execution times, arrival times, data sizes, etc.). The failure rate is then given by the ratio of the number of trials in which the heuristic could not find a mapping to the total number of trials. It has been shown in [10] that some heuristics may produce mappings that perform very similarly with respect to a given performance metric, but these heuristics differ very significantly in their failure rates.

robustness: For a given mapping, what is the smallest departure from the assumed conditions of system operation that will cause the objective function to degrade below some acceptable threshold? That is, what is the robustness of the mapping [9]. Parallel and distributed systems may operate in an environment where certain system performance features degrade due to unpredictable circumstances, such as sudden machine failures, higher than expected system load, or inaccuracies in the estimation of system parameters (e.g., [18,19,40,42,43,54,68]). Therefore, designing heuristics that produce robust mappings is an important design issue.

mapper performance evaluation: How is the mapper being evaluated? This issue is different from the evaluation of a mapping, which is done using the objective function. The mapper evaluation is performed before the mapper is employed in a real system, and is an attempt to find how close to optimal is the mapping found by the mapper. Because all non-trivial mapping problems are likely to be NP-hard, the mapper evaluation is as hard as finding the optimal mapping! One frequent approach is to determine an upper bound on performance. Of course the upper bound should be as tight as possible (being equal to optimal in the tightest case). Another approach is to simulate a system where a particular artificial condition makes it easy to find either the optimal performance or a tight upper bound. In yet another approach, the mapper is simply compared with another well-known mapper from literature. As a special case of the last approach, an on-line mapper may be evaluated against a static mapper with full *a priori* knowledge of the information not available to the on-line mapper. As another special case, a fault tolerant mapper may be evaluated against an otherwise identical fault intolerant mapper [1]. The relative amount of time it takes different mappers to generate mappings also may be a consideration.

application execution time: How are execution times determined by the mapper, e.g., are they estimated or produced from a probability distribution? (This was discussed in workload model and platform model characterizations.)

mapper dynamism: Is the mapping technique dynamic or static? Dynamic mapping techniques operate in real-time (as workload arrives for immediate execution), and make use of real-time information (e.g., [25,59,84,87]). Dynamic techniques require inputs from the environment, and may not have a definite end. For example, dynamic techniques may not know the entire workload to be mapped when the technique begins executing; new applications may arrive at unpredictable intervals. Similarly, new machines may be added to the suite. If a dynamic technique has feedback, the application executing on a machine may be reassigned because of the loss of the machine. Similarly, the applications waiting to be executed on a machine may be reassigned because of the currently executing applications on various machines taking significantly longer or shorter than expected.

In contrast, static mapping techniques take a fixed set of applications, a fixed set of machines, and a fixed set of application and machine attributes as inputs and generate a single, fixed mapping (e.g., [10,15,20,22]). These heuristics typically can use more time to determine a mapping because it is being done off-line, e.g., for production environments; but these heuristics must then use estimated values of some parameters such as when a machine will be available. Static mapping techniques have a well-defined beginning and end, and each resulting mapping is not modified due to changes in the HC environment or feedback. These techniques may be used to plan the execution of a set of tasks for a future time period (e.g., the production tasks to execute on the following day). Static mapping also is used in "what-if" predictive studies. For example, a system administrator might want to know the benefits of adding a new machine to the HC suite before purchasing it. By generating a static mapping (using estimated execution times for tasks and subtasks on the proposed new machines), and then deriving the estimated system performance for the set of applications, the impact of the new machine can be approximated. Static mapping also can be used to do a post-mortem analysis of dynamic mappers, to see how well they are performing. Dynamic mappers must be able to process applications as they arrive into the system, without knowledge of what applications will arrive next. When performing a post-mortem analysis, a static mapper can have the knowledge of all of the applications that had arrived over an interval of time (e.g., the previous day). Also static mappers derive mappings off-line, and thus can take much more time to determine a mapping than a dynamic mapper can. Therefore, the system performance for a static mapping should provide a good way to evaluate the system performance for a dynamic mapping of the same applications.

Some of the items mentioned later in this section apply only to dynamic mapping. They will be indicated with the letter "D" written next to them.

dynamic re-mapping (D): Can the mapping heuristic re-map an initial mapping? What event triggers the re-mapping? For example, a dynamic heuristic with feedback can re-map a previous mapping. The previous mapping could be a static or a dynamic mapping. Re-mapping is usually needed when the state of the system has changed enough so that the mapping found initially performs unacceptably low under the changed scenario. The trigger in this case could be the performance metric falling below a pre-declared threshold. Re-mapping can involve dynamically deriving a new mapping or selecting from previously stored mappings that were determined statically for different situations [52].

mapping trigger (D): What event triggers the dynamic mapping heuristic? Does it map a task as soon as it arrives (immediate mode dynamic mapping) or does it collect arriving tasks into a small batch and then perform the mapping (batch mode dynamic mapping)? Batch mode and immediate mode dynamic mapping techniques perform differently under different conditions (e.g., task arrival rate [59]). The mapping trigger could also depend on the status of the system, e.g., when the workload input queue of any machine falls below a certain threshold.

preemptive (D): What assumptions does the mapping strategy make about preemption of applications (e.g., can applications be interrupted and restarted)? Preemptive mapping strategies can interrupt applications that have already begun execution to free resources for more important applications. Applications that were interrupted may be reassigned (i.e., migrated), or may resume execution upon completion of the more important application. Preemptive techniques must be dynamic by definition. Application "importance" must be specified by some priority assignment and weighting scheme, as already discussed in the section on workload model characterization.

feedback (D): Does the mapping strategy incorporate real-time feedback from the platform (e.g., machine availability times) or applications (e.g., actual task execution times of completed tasks) into its decisions? If yes, how is the state estimated in a distributed system? In other words, as given in Rotithor [66], is state estimation:

(1) Centralized or decentralized? That is, which machines are responsible for collecting state information and constructing an estimate of the system state?
(2) Complete or partial? That is, during any state information exchange, how many machines are involved?
(3) Voluntary or involuntary? That is, how does a machine choose to disseminate state information?
(4) Periodic or aperiodic? That is, at what instant does a machine choose to initiate information dissemination.

data forwarding (S): Is data forwarding considered during mapping? That is, could a subtask executing on a given machine receive data from an intermediate machine sooner than from the original source (e.g., [77,82])? For example, assume that subtasks A, B, and C are mapped on machines X, Y, and Z, respectively, and that A needs to send the same data item to B and C. Further assume that this data item from subtask A on machine X has already been sent to subtask B on machine Y. For subtask C to receive the data item on machine Z, it may choose to receive the data item from machine X (the original source) or intermediate machine Y (the forwarder).

replication (S): Can a given subtask be duplicated and executed on multiple machines to reduce communication overhead? In a replication-based mapping, a subtask may be redundantly executed on more than one processor so as to eliminate the communication time [2,63]. Note that replication also can be used for fault tolerance, and as such has been covered under "fault tolerance" in the mapping strategy characterization.

predictability of time to generate a mapping: Is the time taken by the mapping strategy to generate a mapping predictable? For some heuristics, the mapping generation time can be accurately predicted. For example, in the greedy approaches in [10], the mapping heuristics perform a fixed, predetermined number of steps with a known amount of computation in each step before arriving at a mapping decision. In contrast, some heuristics are iterative in the sense that the mapping is continually refined until some stopping criterion is met, resulting in a number of steps that is not known *a priori*, or in a known number of steps with an unknown amount of work in each step (e.g., genetic algorithms [74,82]). The mapping generation times of different mapping strategies vary greatly, and are an important property during the comparison or selection of mapping techniques. For example, the choice between two mapping heuristics whose performance is comparable may be made based on the heuristics' mapping generation time [7,20].

3. Example Mapper Evaluations

This section characterizes six heuristics from the literature in terms of the 3PCS. The characterizations are presented in Tables I through IX. Each table contains two heuristics. The selection of heuristics to pair in each table was based on a desire to

TABLE I
WORKLOAD MODEL EXAMINATION FOR DFTS [1] AND THE BOTTOMS UP HEURISTIC [69]

Characteristic	DFTS	Bottoms up
workload composition	set of applications decomposable into tasks	one task with communicating subtasks
communication patterns (S)	N/A	directed acyclic graph
data provision and utilization times (S)	N/A	N/S
data retrieval/storage	N/S	N/C
workload size	on-line load specified with arrival and resource demand distributions	1024 subtasks with specified resource demand distribution
workload dynamism	dynamic	static
arrival times	simulated	N/A
deadlines	N/C	N/C
priorities	N/C	N/C
multiple versions	N/C	N/C
QoS requirements	N/C	task must be completed within a given time constraint
interactivity	N/C	N/C
workload heterogeneity	hyper-exponential distribution	Gamma distribution
execution time representation	distribution derived empirically	assumed distribution
data dependence of execution times	N/C	N/C

TABLE II
PLATFORM MODEL EXAMINATION FOR DFTS [1] AND THE BOTTOMS UP HEURISTIC [69]

Characteristic	DFTS	Bottoms up
number of machines	64	8
system control	shared	dedicated
application compatibility	no restrictions	no restrictions
machine heterogeneity and architecture	N/S	modeled
code and data access and storage times	N/A	assumed zero
interconnection network (S)	N/A	wireless
number of connections (S)	N/A	N/A
concurrent send/receives (S)	N/A	1 send/receive
overlapped computation/ communication (S)	N/A	yes
energy consumption	N/C	modeled
multitasking of machines and communication links	yes	N/C
migration of workload	N/C	N/C
resource failure	considered	N/C

show contrasting characteristics. The heuristics have been examined with respect to each of the three parts of the 3PCS, i.e., the workload model, platform model, and mapping strategy characterization. Readers should see the references for detailed descriptions of the heuristics themselves. The following notation is used in Tables I through IX. A field is marked "N/A" if that particular feature is not applicable to the heuristic being examined. A field is marked "N/S" if that particular feature is applicable to the heuristic but its value is not specified in the paper. Finally, "N/C" stands for "not considered," and refers to a feature that has not been considered in the given paper.

4. Related Work

Taxonomies related in various degrees to this work have appeared in the literature. In this section, overviews of some related taxonomy studies are given.

Ahmad et al. [3] survey algorithms that allocate a parallel program represented by an edge-weighted directed acyclic graph (DAG) to a set of homogeneous processors, with the objective of minimizing the completion time. They propose a taxonomy of scheduling with four groups. The first group includes algorithms that schedule the DAG to a bounded number of processors directly. These algorithms are called the

TABLE III
MAPPING STRATEGY EXAMINATION FOR DFTS [1] AND THE BOTTOMS UP HEURISTIC [69]

Characteristic	DFTS	Bottoms up
support for the workload model	yes	yes
support for the platform model	yes	yes
control location	centralized	centralized
execution location	N/S	N/S
fault tolerance	yes, through task replication	N/C
objective function	mean response time	energy consumption
failure rate	N/C	N/C
robustness	N/C	N/C
mapper performance evaluation	comparison with a fault intolerant heuristic	comparison with a lower bound
dynamism	on-line	static
dynamic re-mapping (D)	triggered when number of healthy replicas falls below a threshold	N/A
mapping trigger (D)	task arrival	N/A
preemptive (D)	yes	N/A
feedback (D)	yes	N/A
data forwarding (S)	N/A	N/C
replication	yes	N/C
time to generate a mapping	predictable	predictable

bounded number of processors scheduling algorithms. The algorithms in the second group "cluster" the subtasks, where a cluster is a set of subtasks formed to reduce or eliminate communication times. Because the number of clusters can be unbounded, these algorithms are called the unbounded number of clusters scheduling algorithms. The algorithms in the third group schedule the DAG using task duplication and are called the task duplication based scheduling algorithms. The algorithms in the fourth group perform allocation and mapping on arbitrary processor network topologies. These algorithms are called the arbitrary processor network scheduling algorithms. The authors discuss the design philosophies and principles behind these algorithms classes, and then analyze and classify 21 scheduling algorithms.

A scheme for classifying static scheduling techniques used in general-purpose distributed computing systems is presented in [24]. The classification of workload and platform was outside the scope of this study. The taxonomy in [24] does combine well-defined hierarchical characteristics with more general flat characteristics to differentiate a wide range of scheduling techniques. Several examples of different scheduling techniques from the published literature are also given, with each classified by the taxonomy. In HC systems, however, scheduling is only half of the

TABLE IV
WORKLOAD MODEL EXAMINATION FOR THE OASS ALGORITHM [45] AND THE HRA MAX–MIN HEURISTIC [8]

Characteristic	OASS	HRA max–min
workload composition	communicating subtasks	communicating subtasks, continuously executing
communication patterns (S)	undirected graph	DAG
data provision and utilization times (S)	N/S	N/S
data retrieval/storage	N/S	N/S
workload size	28 subtasks, resource demands calculated with a devised procedure	40 subtasks, resource demands sampled from a Gamma distribution
workload dynamism	static	static
arrival times	N/A	N/A
deadlines	N/C	N/C
priorities	N/C	N/C
multiple versions	N/C	N/C
QoS requirements	N/C	N/C
interactivity	N/C	N/C
workload heterogeneity	N/S	Gamma distribution
execution time representation	N/S	assumed distribution
data dependence of execution times	N/C	N/C

mapping problem. The matching of tasks to machines also greatly affects execution schedules and system performance. Therefore, the 3PCS also includes categories for platform and workload models, both of which influence matching (and scheduling) decisions.

Several different taxonomies are presented in [30]. The first is the EM^3 taxonomy, which classifies all computer systems into one of four categories, based on execution mode and machine model [30]. The 3PCS proposed here assumes heterogeneous systems from either the SEMM (single execution mode, multiple machine models) or the MEMM (multiple execution modes, multiple machine models) categories. A "modestly extended" version of the taxonomy from [24] is also presented in [30]. The modified taxonomy introduces new descriptors and is applied to heterogeneous resource allocation techniques. Aside from considering different parallelism characteristics of applications, [30] did not explicitly consider workload model characterization and platform model characterization.

A taxonomy for comparing heterogeneous subtask matching methodologies is included in [44]. The taxonomy focuses on static subtask matching approaches, and classifies several specific examples of optimal and sub-optimal techniques. This is a

TABLE V
PLATFORM MODEL EXAMINATION FOR THE OASS ALGORITHM [45] AND THE HRA MAX–MIN HEURISTIC [8]

Characteristic	OASS	HRA max–min
number of machines	finite, fixed	finite, fixed
system control	N/S	shared
application compatibility	no restrictions	restricted
machine heterogeneity and architecture	N/S	modeled
code and data access and storage times	N/C	N/C
interconnection network (S)	point-to-point	non-blocking switched network
number of connections (S)	1	1
concurrent send/receives (S)	N/S	N/S
overlapped computation/ communication (S)	yes	yes
energy consumption	N/C	N/C
multitasking of machines and communication links	N/S	N/S
migration of workload	N/C	N/C
resource failure	N/C	N/C

single taxonomy, without the three distinct parts of the 3PCS. However, the "optimal-restricted" classification in [44] includes algorithms that place restrictions on the underlying program and/or multicomputer system.

Krauter et al. [50] give a taxonomy and a survey of grid resource management systems. Their taxonomy covers resource and resource manager models. For example, they include classifications like resource discovery (query based or agent based), resource dissemination (periodic or on-demand), QoS support (soft, hard, or none), scheduler organization (centralized, decentralized, or hierarchical), and rescheduling (periodic or even driven). The 3PCS includes the workload model in addition to the models discussed in [50]. Furthermore, the platform model and mapping strategy characterizations in the 3PCS include parameters not included in [50].

Kwok and Ahmad [51] provide a taxonomy for classifying various scheduling algorithms into different categories according to their assumptions and functionalities. They also propose a set of benchmarks to allow a comprehensive performance evaluation and comparison of these algorithms. With very much the same motivation as the study given in this chapter, Kwok and Ahmad [51] argue that while many scheduling heuristics proposed in literature are individually reported to be efficient, it is not clear how effective they are and how well they compare against each other, partially because these scheduling algorithms are based upon radically different assumptions.

TABLE VI
MAPPING STRATEGY EXAMINATION FOR THE OASS ALGORITHM [45] AND THE HRA MAX–MIN HEURISTIC [8]

Characteristic	OASS	HRA max–min
support for the workload model	yes	yes
support for the platform model	yes	yes
control location	centralized	centralized
execution location	N/S	N/S
fault tolerance	N/C	N/C
objective function	makespan	load balancing
failure rate	N/C	considered
robustness	N/C	N/C
mapper performance evaluation	comparison with an existing heuristic	comparison with a lower bound
dynamism	static	static
dynamic re-mapping (D)	N/A	N/A
mapping trigger (D)	N/A	N/A
preemptive (D)	N/A	N/A
feedback (D)	N/A	N/A
data forwarding (S)	N/A	N/A
replication	N/C	N/C
time to generate a mapping	predictable	predictable

Kwok and Ahmad [51] evaluate 15 scheduling algorithms, and compare them using the proposed benchmarks. They interpret the results and discuss why some algorithms perform better than the others. However, the taxonomy in [51] is based on the problem of scheduling a weighted directed acyclic graph to a set of homogeneous processors, whereas the 3PCS is for heterogeneous systems with a broader range of application types.

Noronha and Sarma [62] survey several existing intelligent planning and scheduling systems for providing a guide to the main artificial intelligence techniques. They give a taxonomy of planning and scheduling problems in an attempt to reconcile the differences in usage of the terms planning and scheduling between the AI and operations research communities. Some of the more successful planning and scheduling systems are surveyed, and their features are highlighted, e.g., deterministic versus stochastic, algorithm complexity (polynomial versus NP-hard), dynamism of the scheduler (on-line versus off-line), precedence constraints, resource constraints, objective function, scheduler type (optimizing versus feasible, i.e., satisfying the problem requirements).

Rotithor [66] presents a taxonomy of dynamic task scheduling schemes in distributed computing systems. The author argues that system state estimation and decision

TABLE VII
WORKLOAD MODEL EXAMINATION FOR THE KPB HEURISTIC [59] AND THE MIN–MIN HEURISTIC [21]

Characteristic	KPB	Min–min
workload composition	independent tasks	communicating subtasks
communication patterns (S)	N/A	DAG
data provision and utilization times (S)	N/A	consumer subtask must wait until all input data has been received
data retrieval/storage	N/C	N/C
workload size	2000 tasks, resource demands sampled from a truncated Gaussian distribution	1000 tasks and 1000 subtasks, resource demands sampled from a Gamma distribution
workload dynamism	dynamic	static
arrival times	Poisson distribution	N/S
deadlines	N/C	hard deadline on each task and subtask
priorities	N/C	tasks and subtasks classified in four priority classes
multiple versions	N/C	three versions per task
QoS requirements	N/C	N/C
interactivity	N/C	N/C
workload heterogeneity	truncated Gaussian distribution	Gamma distribution
execution time representation	assumed distribution	assumed distribution
data dependence of execution times	N/C	N/C

making are the two major components of dynamic task scheduling in a distributed computing system, and that the combinations of solutions to each individual component constitute solutions to the dynamic task scheduling problem. Based on this argument, the author presents a taxonomy of dynamic task scheduling schemes that is synthesized by treating state estimation and decision making as orthogonal problems. Solutions to estimation and decision making are analyzed and the resulting solution space of dynamic task scheduling is shown. The author shows the applicability of the proposed taxonomy by means of examples that encompass solutions proposed in the literature. The state estimation classification scheme sits under the "feedback" box in Figure 8, and is incorporated in our discussion of "feedback" in the mapping strategy characterization section.

Stankovic et al. [75] give an excellent discussion of a set of real-time scheduling results. The paper presents a simple classification scheme for the real-time scheduling theory, which it divides in two groups based on the platform model: uniprocessor and multiprocessor. For the uniprocessor scheduling, the authors further differentiate

TABLE VIII
PLATFORM MODEL EXAMINATION FOR THE KPB HEURISTIC [59] AND THE MIN–MIN HEURISTIC [21]

Characteristic	KPB	Min–min
number of machines	finite, variable	finite, fixed
system control	dedicated	dedicated
application compatibility	no restrictions	no restrictions
machine heterogeneity and architecture	modeled	modeled
code and data access and storage times	N/C	N/C
interconnection network (S)	N/A	N/S
number of connections (S)	N/A	N/S
concurrent send/receives (S)	N/A	N/S
overlapped computation/ communication (S)	N/A	N/S
energy consumption	N/C	N/C
multitasking of machines and communication links	N/C	N/C
migration of workload	N/C	N/C
resource failure	N/C	N/C

TABLE IX
MAPPING STRATEGY EXAMINATION FOR THE KPB HEURISTIC [59] AND THE MIN–MIN HEURISTIC [21]

Characteristic	KPB	Min–min
support for the workload model	yes	yes
support for the platform model	yes	yes
control location	centralized	centralized
execution location	external	N/S
fault tolerance	N/C	N/C
objective function	makespan	makespan
failure rate	N/C	N/C
robustness	N/C	N/C
mapper performance evaluation	comparison with an existing heuristic	comparison with a lower bound
dynamism	dynamic	static
dynamic re-mapping (D)	yes	N/A
mapping trigger (D)	task arrival	N/A
preemptive (D)	no	N/A
feedback (D)	yes	N/A
data forwarding (S)	N/A	N/S
replication	N/C	N/C
time to generate a mapping	predictable	predictable

between dedicated and shared resources. For the multiprocessor schedulers, the authors differentiate between static and dynamic techniques, and further examine each kind for preemptive and non-preemptive techniques. The focus of the effort in [75] is on real-time systems.

T'kindt and Billaut [80] present a survey of multi-criteria scheduling, and discuss some basic results of multi-criteria optimization literature. One of their aims is to contextualize the performance evaluation of a given multi-criteria scheduling scheme to be able to answer questions like: Are trade-offs among criteria allowed? Is it possible to associate a weight to each criterion? Is it possible to associate a particular goal value to each criterion? Does an upper bound exist for each criterion? In some of these respects, their work is similar to ours, but is intended for job shop community.

The 3PCS uses these studies as a foundation, and extends their concepts. Relevant ideas from these studies are incorporated into the unique structure of the 3PCS, allowing for more detailed classifications of HC mapping heuristics.

5. Summary

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 overall system performance or cost-effectiveness. An important research problem in HC is mapping, i.e., how to assign resources to applications, and schedule the applications on a given machine, so as to maximize some performance criterion without violating any quality of service constraints. This chapter proposes a three-part classification scheme (*3PCS*) for understanding, describing, and selecting a mapper for HC systems. The 3PCS allows for fair comparison of different heuristics.

The 3PCS for HC systems can help researchers who want to understand a mapper by giving them a "check list" of various characteristics of the workload model, platform model, and strategy attributes of the mapper. As such, it can help extend existing mapping work, and recognize important areas of research by facilitating understanding of the relationships that exist among previous efforts. The 3PCS also is useful for researchers who want to describe their mapper more thoroughly by using a common standard. In this regard, the 3PCS also may help researchers see the design and environment alternatives that they might not have otherwise considered during the development of new heuristics. Additionally, the 3PCS is useful when one wants to select a mapper to match a given real-world environment. All three parts of the 3PCS are needed to determine which heuristics would be appropriate for a given environment, and also to facilitate fair comparison of different heuristics.

Acknowledgements

This research was supported by the DARPA/ITO Quorum Program under NPS subcontract numbers N62271-98-M-0217 and N62271-98-M-0448, GSA subcontract number GS09K99BH0250, and, the Office of Naval Research Grant No. N00014-00-1-0599; by the DARPA/ITO AICE Program under contract numbers DABT63-99-C-0010 and DABT63-99-C-0012; by the DARPA/ISO BADD Program and the Office of Naval Research under grant number N00014-97-1-0804; by the Colorado State University George T. Abell Endowment; and by the Colorado State University Center for Robustness in Computer Systems (funded by the Colorado Commission on Higher Education Technology Advancement Group through the Colorado Institute of Technology).

The authors thank Ishfaq Ahmad, Richard F. Freund, Veljko Milutinovic, Janet Siegel and Jay Smith for their comments. We presented a preliminary version of portions of this material at the Workshop on Advances in Parallel and Distributed Systems.

References

[1] Abawajy J.H., "Fault-tolerant scheduling policy for grid computing systems", in: *18th International Parallel and Distributed Processing Symposium (IPDPS 2004)*, April 2004, pp. 238–244.

[2] Ahmad I., Kwok Y.-K., "On exploiting task duplication in parallel program scheduling", *IEEE Transactions on Parallel and Distributed Systems* **9** (9) (September 1998) 872–892.

[3] Ahmad I., Kwok Y.-K., Wu M.-Y., "Analysis, evaluation, and comparison of algorithms for scheduling task graphs on parallel processors", in: *2nd International Symposium on Parallel Architectures, Algorithms, and Networks*, vol. 6, June 1996, pp. 207–213.

[4] Ali S., "Robust resource allocation in dynamic distributed heterogeneous computing systems", Ph.D. Thesis, School of Electrical and Computer Engineering, Purdue University, August 2003.

[5] Ali S., Braun T.D., Siegel H.J., Maciejewski A.A., "Heterogeneous computing", in: Urban J., Dasgupta P. (Eds.), *Encyclopedia of Distributed Computing*, Kluwer Academic Publishers, Norwell, MA, 2004, in press.

[6] Ali S., Kim J.-K., Yu Y., Gundala S.B., Gertphol S., Siegel H.J., Maciejewski A.A., Prasanna V., "Greedy heuristics for resource allocation in dynamic distributed real-time heterogeneous computing systems", in: *2002 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2002)*, vol. II, June 2002, pp. 519–530.

[7] Ali S., Kim J.-K., Yu Y., Gundala S.B., Gertphol S., Siegel H.J., Maciejewski A.A., Prasanna V., "Utilization-based techniques for statically mapping heterogeneous applications onto the HiPer-D heterogeneous computing system", in: *Parallel and Distributed Computing Practices*, 2004, in press.

[8] Ali S., Kim J.-K., Yu Y., Gundala S.B., Gertphol S., Siegel H.J., Maciejewski A.A., Prasanna V., "Utilization-based heuristics for statically mapping real-time applications onto the HiPer-D heterogeneous computing system", in: *11th IEEE Heterogeneous Computing Workshop (HCW 2002) in the Proceedings of the 16th International Parallel and Distributed Processing Symposium (IPDPS 2002)*, April 2002.

[9] Ali S., Maciejewski A.A., Siegel H.J., Kim J.-K., "Measuring the robustness of a resource allocation", *IEEE Transactions on Parallel and Distributed Systems* **15** (7) (July 2004) 630–641.

[10] Ali S., Maciejewski A.A., Siegel H.J., Kim J.-K., "Robust resource allocation for distributed computing systems", in: *2004 International Conference on Parallel Processing (ICPP 2004)*, 2004.

[11] Ali S., Siegel H.J., Maheswaran M., Hensgen D., Sedigh-Ali S., "Representing task and machine heterogeneities for heterogeneous computing systems", *Tamkang Journal of Science and Engineering* **3** (3) (November 2000) 195–207, invited.

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

[13] Aydin H., Melhem R., Mossé D., Mejía-Alvarez P., "Power-aware scheduling for periodic real-time tasks", *IEEE Transactions on Computers* **53** (5) (May 2004) 584–600.

[14] Bajaj R., Agrawal D.P., "Improving scheduling of tasks in a heterogeneous environment", *IEEE Transactions on Parallel and Distributed Systems* **15** (2) (February 2004) 107–118.

[15] Banino C., Beaumont O., Carter L., Ferrante J., Legrand A., Robert Y., "Scheduling strategies for master–slave tasking on heterogeneous processor platforms", *IEEE Transactions on Parallel and Distributed Systems* **15** (4) (April 2004) 319–330.

[16] Baskiyar S., SaiRanga P.C., "Scheduling directed a-cyclic task graphs on heterogeneous network of workstations to minimize schedule length", in: *2003 International Conference on Parallel Processing Workshops (ICPPW03)*, October 2003, pp. 97–103.

[17] Beaumont O., Legrand A., Robert Y., "Scheduling strategies for mixed data and task parallelism on heterogeneous clusters and grids", in: *11th Euromicro Conference on Parallel, Distributed and Network-Based Processing*, February 2003, pp. 209–216.

[18] Berry P.M., "Uncertainty in scheduling: Probability, problem reduction, abstractions and the user", IEE Computing and Control Division Colloquium on Advanced Software Technologies for Scheduling, Digest No. 1993/163, April 26, 1993.

[19] Bölöni L., Marinescu D.C., "Robust scheduling of metaprograms", *Journal of Scheduling* **5** (5) (September 2002) 395–412.

[20] Braun T.D., Siegel H.J., Beck N., Bölöni L.L., Maheswaran M., Reuther A.I., Robertson J.P., Theys M.D., Yao B., Hensgen D., Freund R.F., "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) (June 2001) 810–837.

[21] Braun T.D., Siegel H.J., Maciejewski A.A., "Heterogeneous computing: Goals, methods, and open problems", in: Monien B., Prasanna V.K., Vajapeyam S. (Eds.), *High Peformance Computing—HiPC 2001*, in: *Lecture Notes in Computer Science*, vol. 2228, Springer-Verlag, Berlin, 2001, pp. 307–318.

[22] Braun T.D., Siegel H.J., Maciejewski A.A., "Heterogeneous computing: Goals, methods, and open problems" (invited keynote presentation for the 2001 International Multiconference that included PDPTA 2001), in: *2001 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2001)*, vol. I, June 2001, pp. 1–12.
[23] Braun T.D., Siegel H.J., Maciejewski A.A., "Static mapping heuristics for tasks with dependencies, priorities, deadlines, and multiple versions in heterogeneous environments", in: *16th International Parallel and Distributed Processing Symposium (IPDPS 2002)*, April 2002.
[24] Casavant T.L., Kuhl J.G., "A taxonomy of scheduling in general-purpose distributed computing systems", *IEEE Transactions on Software Engineering* **14** (2) (February 1988) 141–154.
[25] Chandra A., Gong W., Shenoy P., "Dynamic resource allocation for shared data centers using online measurements", in: *2003 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, June 2003, pp. 300–301.
[26] Dally W., Towles B., *Principles and Practices of Interconnection Networks*, Morgan Kaufmann, San Francisco, CA, 2003.
[27] Dhodhi M.K., Hielscher F.H., Storer R.H., "SHEMUS: Synthesis of heterogeneous multiprocessor systems", *Microprocessor and Microsystems* **19** (6) (August 1995) 311–319.
[28] Duato J., Yalmanchili S., Ni L., *Interconnection Networks: An Engineering Approach*, IEEE Computer Society Press, Los Alamitos, CA, 1997.
[29] Ekmečić I., Tartalja I., Milutinović V., "A taxonomy of heterogeneous computing", *IEEE Computer* **28** (12) (December 1995) 68–70.
[30] Ekmečić I., Tartalja I., Milutinović V., "A survey of heterogeneous computing: Concepts and systems", *Proceedings of the IEEE* **84** (8) (August 1996) 1127–1144.
[31] Eshaghian M.M. (Ed.), *Heterogeneous Computing*, Artech House, Norwood, MA, 1996.
[32] Eshaghian M.M., Wu Y.-C., "A portable programming model for network heterogeneous computing", in: Eshaghian M.M. (Ed.), *Heterogeneous Computing*, Artech House, Norwood, MA, 1996, pp. 155–195.
[33] Fernandez-Baca D., "Allocating modules to processors in a distributed system", *IEEE Transaction on Software Engineering* **SE-15** (11) (November 1989) 1427–1436.
[34] Foster I., Kesselman C. (Eds.), *The Grid 2: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, San Francisco, CA, 2004.
[35] Freund R.F., Gherrity M., Ambrosius S., Campbell M., Halderman M., Hensgen D., Keith E., Kidd T., Kussow M., Lima J.D., Mirabile F., Moore L., Rust B., Siegel H.J., "Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet", in: *7th IEEE Heterogeneous Computing Workshop (HCW '98)*, March 1998, pp. 184–199.
[36] Freund R.F., Siegel H.J., "Heterogeneous processing", *IEEE Computer* **26** (6) (June 1993) 13–17.
[37] Gertphol S., Yu Y., Gundala S.B., Prasanna V.K., Ali S., Kim J.-K., Maciejewski A.A., Siegel H.J., "A metric and mixed-integer-programming-based approach for resource allocation in dynamic real-time systems", in: *16th International Parallel and Distributed Processing Symposium (IPDPS 2002)*, April 2002.

[38] Ghafoor A., Yang J., "Distributed heterogeneous supercomputing management system", *IEEE Computer* **26** (6) (June 1993) 78–86.

[39] Gomoluch J., Schroeder M., "Performance evaluation of market-based resource allocation for grid computing", *Concurrency and Computation: Practice and Experience* **16** (5) (April 2004) 469–475.

[40] Gribble S.D., "Robustness in complex systems", in: *8th Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, May 2001, pp. 21–26.

[41] Hong I., Qu G., Potkonjak M., Srivastava M., "Synthesis techniques for low-power hard real-time systems on variable voltage processors", in: *19th IEEE Real-Time Systems Symposium (RTSS '98)*, December 1998, pp. 95–105.

[42] Jen E., "Stable or robust? What is the difference?", Santa Fe Institute Working Paper No. 02-12-069, 2002.

[43] Jensen M., "Improving robustness and flexibility of tardiness and total flowtime job shops using robustness measures", *Journal of Applied Soft Computing* **1** (1) (June 2001) 35–52.

[44] Kafil M., Ahmad I., "Optimal task assignment in heterogeneous computing systems", in: *6th IEEE Heterogeneous Computing Workshop (HCW '97)*, April 1997, pp. 135–146.

[45] Kafil M., Ahmad I., "Optimal task assignment in heterogeneous distributed computing systems", *IEEE Concurrency* **6** (3) (July–September 1998) 42–51.

[46] Kim J.-K., "Resource management in heterogeneous computing systems: Continuously running applications, tasks with priorities and deadlines, and power constrained mobile devices", Ph.D. Thesis, School of Electrical and Computer Engineering, Purdue University, August 2004.

[47] Kim J.-K., Hensgen D.A., Kidd T., Siegel H.J., John D.S., Irvine C., Levin T., Porter N.W., Prasanna V.K., Freund R.F., "A flexible multi-dimensional QoS performance measure framework for distributed heterogeneous systems", in: *Cluster Computing*, 2005, in press. Special issue on *Cluster Computing in Science and Engineering*.

[48] Kim J.-K., Shivle S., Siegel H.J., Maciejewski A.A., Braun T., Schneider M., Tideman S., Chitta R., Dilmaghani R.B., Joshi R., Kaul A., Sharma A., Sripada S., Vangari P., Yellampalli S.S., "Dynamic mapping in a heterogeneous environment with tasks having priorities and multiple deadlines", in: *12th IEEE Heterogeneous Computing Workshop (HCW 2003) in the Proceedings of the 17th International Parallel and Distributed Processing Symposium (IPDPS 2003)*, April 2003.

[49] Ko K., Robertazzi T.G., "Equal allocation scheduling for data intensive applications", *IEEE Transactions on Aerospace and Electronic Systems* **40** (2) (April 2004) 695–705.

[50] Krauter K., Buyya R., Maheswaran M., "A taxonomy and survey of grid resource management systems", Tech. rep., University of Mannitoba, Canada and Monash University, Australia, TR 2000-80, November 2000.

[51] Kwok Y.-K., Ahmad I., "Static scheduling algorithms for allocating directed task graphs to multiprocessors", *ACM Computing Surveys* **31** (4) (1999) 406–471.

[52] Kwok Y.-K., Maciejewski A.A., Siegel H.J., Ghafoor A., Ahmad I., "Evaluation of a semi-static approach to mapping dynamic iterative tasks onto heterogeneous computing systems", in: *1999 International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN '99)*, June 1999, pp. 204–209.

[53] Leangsuksun C., Potter J., Scott S., "Dynamic task mapping algorithms for a distributed heterogeneous computing environment", in: *4th IEEE Heterogeneous Computing Workshop (HCW '95)*, April 1995, pp. 30–34.

[54] Leon V.J., Wu S.D., Storer R.H., "Robustness measures and robust scheduling for job shops", *IEE Transactions* **26** (5) (September 1994) 32–43.

[55] Li Y.A., Antonio J.K., Siegel H.J., Tan M., Watson D.W., "Determining the execution time distribution for a data parallel program in a heterogeneous computing environment", *Journal of Parallel and Distributed Computing* **44** (1) (July 1997) 35–52.

[56] Liszka K.J., Antonio J.K., Siegel H.J., "Problems with comparing interconnection networks: Is an alligator better than an armadillo?", *IEEE Concurrency* **5** (4) (October–December 1997) 18–28.

[57] Lloyd S., "Complex systems: A review, ESD-WP-2003-01.16", in: *ESD Internal Symposium, Working Paper Series*, Massachusetts Institute of Technology, May 2002.

[58] Magee C.L., de Weck O.L., "An attempt at complex system classification, ESD-WP-2003-01.02", in: *ESD Internal Symposium, Working Paper Series*, Massachusetts Institute of Technology, May 2002.

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

[60] Maheswaran M., Braun T.D., Siegel H.J., "Heterogeneous distributed computing", in: Webster J.G. (Ed.), *Encyclopedia of Electrical and Electronics Engineering*, vol. 8, John Wiley, New York, 1999, pp. 679–690.

[61] Marinescu D.C., Marinescu G.M., Ji Y., Bölöni L., Siegel H.J., "Ad hoc grids: Communication and computing in a power constrained environment", in: *Workshop on Energy-Efficient Wireless Communications and Networks 2003 (EWCN 2003) in the Proceedings of the 22nd International Performance, Computing, and Communications Conference (IPCCC 2003)*, April 2003.

[62] Noronha S.J., Sarma V.V.S., "Knowledge-based approaches for scheduling problems", *IEEE Transactions on Knowledge and Data Engineering* **3** (2) (June 1991) 160–171.

[63] Park C.-I., Choe T.-Y., "An optimal scheduling algorithm based on task duplication", *IEEE Transactions on Computers* **51** (4) (April 2002) 444–448.

[64] Prakash S., Parker A.C., "SOS: Synthesis of application-specific heterogeneous multiprocessor systems", *Journal of Parallel and Distributed Computing* **16** (4) (December 1992) 338–351.

[65] Ravindran B., Li P., "DPR, LPR: Proactive resource allocation algorithms for asynchronous real-time distributed systems", *IEEE Transactions on Computers* **53** (2) (February 2004) 201–216.

[66] Rotithor H.G., "Taxonomy of dynamic task scheduling schemes in distributed computing systems", *IEE Proceedings on Computer and Digital Techniques* **141** (1) (January 1994) 1–10.

[67] Schuster P., "How does complexity arise in evolution: Nature's recipe for mastering scarcity, abundance, and unpredictability", *Complexity* **2** (December 1996) 22–30.

[68] Sevaux M., Sörensen K., "Genetic algorithm for robust schedules", in: *8th International Workshop on Project Management and Scheduling (PMS 2002)*, April 2002, pp. 330–333.

[69] Shivle S., Castain R., Siegel H.J., Maciejewski A.A., Banka T., Chindam K., Dussinger S., Pichumani P., Satyasekaran P., Saylor W., Sendek D., Sousa J., Sridharan J., Sugavanam P., Velazco J., "Static mapping of subtasks in a heterogeneous ad hoc grid environment", in: *13th IEEE Heterogeneous Computing Workshop (HCW 2004) in the Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS 2004)*, April 2004.

[70] Shivle S., Siegel H.J., Maciejewski A.A., Banka T., Chindam K., Dussinger S., Kutruff A., Penumarthy P., Pichumani P., Satyasekaran P., Sendek D., Sousa J.C., Sridharan J., Sugavanam P., Velazco J., "Mapping of subtasks with multiple versions in a heterogeneous ad hoc grid environment", in: *3rd International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks (HeteroPar 2004) in the Proceedings of the 3rd International Symposium on Parallel and Distributed Computing in association with HeteroPar'04 (ISPDC/HeteroPar 2004)*, July 2004.

[71] Siegel H.J., *Interconnection Networks for Large-Scale Parallel Processing: Theory and Case Studies*, second ed., McGraw-Hill, New York, 1990.

[72] Siegel H.J., Dietz H.G., Antonio J.K., "Software support for heterogeneous computing", in: Tucker J.A.B. (Ed.), *The Computer Science and Engineering Handbook*, CRC Press, Boca Raton, FL, 1997, pp. 1886–1909.

[73] Siegel H.J., Siegel L.J., Kemmerer F., Mueller P.T. Jr., Smalley H.E. Jr., Smith S.D., "PASM: A partitionable SIMD/MIMD system for image processing and pattern recognition", *IEEE Transactions on Computers* **C-30** (12) (December 1981) 934–947.

[74] Singh H., Youssef A., "Mapping and scheduling heterogeneous task graphs using genetic algorithms", in: *5th IEEE Heterogeneous Computing Workshop (HCW '96)*, April 1996, pp. 86–97.

[75] Stankovic J.A., Spuri M., Natale M.D., Buttazzo G.C., "Implications of classical scheduling results for real-time systems", *IEEE Computer* **28** (6) (June 1995) 16–25.

[76] Sussman J.M., "Collected views on complexity in systems, ESD-WP-2003-01.01", in: *ESD Internal Symposium, Working Paper Series*, Massachusetts Institute of Technology, May 2002.

[77] Tan M., Siegel H.J., Antonio J.K., Li Y.A., "Minimizing the application execution time through scheduling of subtasks and communication traffic in a heterogeneous computing system", *IEEE Transactions on Parallel and Distributed Systems* **8** (8) (August 1997) 857–871.

[78] Theys M.D., Siegel H.J., Chong E.K.P., "Heuristics for scheduling data requests using collective communications in a distributed communication network", *Journal of Parallel and Distributed Computing* **61** (9) (September 2001) 1337–1366.

[79] Theys M.D., Tan M., Beck N.B., Siegel H.J., Jurczyk M., "A mathematical model and scheduling heuristics for satisfying prioritized data requests in an oversubscribed communication network", *IEEE Transactions on Parallel and Distributed Systems* **11** (9) (September 2000) 969–988.

[80] T'kindt V., Billaut J.-C., "Some guidelines to solve multicriteria scheduling problems", in: *1999 IEEE International Conference on Systems, Man, and Cybernetics*, vol. 6, October 1999, pp. 463–468.

[81] Veeravalli B., Min W.H., "Scheduling divisible loads on heterogeneous linear daisy chain networks with arbitrary processor release times", *IEEE Transactions on Parallel and Distributed Systems* **15** (3) (March 2004) 273–288.

[82] Wang L., Siegel H.J., Roychowdhury V.P., Maciejewski A.A., "Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach", *Journal of Parallel and Distributed Computing* **47** (1) (November 1997) 8–22.

[83] Weiser M., Welch B., Demers A., Shenker S., "Scheduling for reduced CPU energy", in: *USENIX Symposium on Operating Systems Design and Implementation*, November 1994, pp. 13–23.

[84] Xiao L., Chen S., Zhang X., "Dynamic cluster resource allocations for jobs with known and unknown memory demands", *IEEE Transactions on Parallel and Distributed Systems* **13** (3) (March 2002) 223–240.

[85] Xu D., Nahrstedt K., Wichadakul D., "QoS and contention-aware multi-resource reservation", *Cluster Computing* **4** (2) (April 2001) 95–107.

[86] Yang J., Ahmad I., Ghafoor A., "Estimation of execution times on heterogeneous supercomputer architecture", in: *1993 International Conference on Parallel Processing (ICPP '93)*, vol. I, August 1993, pp. 219–225.

[87] Zomaya A.Y., Teh Y.-H., "Observations on using genetic algorithms for dynamic load-balancing", *IEEE Transactions on Parallel and Distributed Systems* **12** (9) (September 2001) 899–911.