A Taxonomy for Describing Matching and Scheduling Heuristics for Mixed-Machine Heterogeneous Computing Systems

Tracy D. Braun†, Howard Jay Siegel†, Noah Beck†, Ladislau Bölöni‡, Muthucumaru Maheswaran†, Albert I. Reuther†, James P. Robertson†, Mitchell D. Theys†, and Bin Yao†
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907-1285 USA
†{tdbraun, hj, noah, maheswar, reuther, robertso, theys, yaob}@ecn.purdue.edu
‡boloni@cs.purdue.edu

Abstract

The problem of mapping (defined as matching and scheduling) tasks and communications onto multiple machines and networks in a heterogeneous computing (HC) environment has been shown to be NP-complete, in general, requiring the development of heuristic techniques. Many different types of mapping heuristics have been developed in recent years. However, selecting the best heuristic to use in any given scenario remains a difficult problem. Factors making this selection difficult are discussed. Motivated by these difficulties, a new taxonomy for classifying mapping heuristics for HC environments is proposed ("the Purdue HC Taxonomy"). The taxonomy is defined in three major parts: (1) the models used for applications and communication requests, (2) the models used for target hardware platforms, and (3) the characteristics of mapping heuristics. Each part of the taxonomy is described, with examples given to help clarify the taxonomy. The benefits and uses of this taxonomy are also discussed.

1. Introduction

Different portions of a computationally intensive application often have diverse computational requirements. In general, a high-performance machine may perform poorly on such an application because it is difficult for a single machine architecture to satisfy the computational requirements of the different portions equally well. A mixed-machine heterogeneous computing (HC) system alleviates this problem by utilizing a suite of different high-performance machines, interconnected with high-speed links. Such a system coordinates the execution of various portions of the application on different machines within the system to exploit the different architectural capabilities available and achieve increased application performance [12, 13].

This research was supported in part by the DARPA/ITO Quorum Program under NPS subcontract numbers N62271-98-M-0217 and N62271-98-M-0448, and in part by the DARPA/ISO BADD Program and the Office of Naval Research under grant number N00014-97-1-0804.

To take advantage of HC systems in this manner, an application task may be decomposed into subtasks, where each subtask is computationally homogeneous. Different subtasks, however, may require different architectural capabilities. These subtasks may share stored or generated data, creating the potential for inter-machine dependencies and data transfer overhead. Subtasks may be determined by: (a) user specification, (b) analysis by the mapping heuristic, or (c) a given separate program for each subtask. Once the application is decomposed into subtasks, each subtask is assigned to a machine (matching) and the subtasks assigned to a particular machine are ordered (scheduling) such that the overall execution time of the application is minimized. The combination of matching and scheduling subtasks to machines is defined as subtask mapping. While each subtask is a separate item to be mapped, the mapping decision for any one subtask may impact the mapping decision for others.

Another version of the mapping problem, meta-task mapping, deals with matching and scheduling a collection of tasks to the machines in an HC environment. The term meta-task has been used in different ways. In this paper, the tasks in the meta-task are independent, in that they have no data dependencies among them. A given task, however, may have subtasks and dependencies among the subtasks. (In some systems, all tasks and subtasks in a meta-task, as defined above, are referred to as just tasks.) An example of meta-task mapping is the mapping of an arbitrary set of independent tasks from different users waiting to execute on a heterogeneous suite of machines. Each task in a meta-task may have associated properties, such as a deadline and a priority.

In general, finding optimal solutions for the mapping problem and the scheduling of inter-machine communications in HC environments is NP-complete [7], requiring the development of near-optimal heuristic techniques. In recent years, numerous different types of mapping heuristics have been developed (e.g., see [1, 6, 8, 12, 13]). However, selecting a particular heuristic to use in a certain practical scenario remains a difficult problem. One of the reasons for this difficulty is that when one heuristic is presented and evaluated in the literature, typically, different assumptions are made about the underlying target platform than those used for earlier heuristics, (e.g., the degree to which the capabilities of machines differ in the HC suite) making comparisons problematic. Similarly, different assumptions about application models complicate comparisons (e.g, the variation among average task execution times). Moreover, the mapping heuristics themselves usually have different characteristics (e.g., different optimization criteria, different execution times). Therefore, a fair comparison of various heuristics is a challenging problem.

These comparison problems are compounded by the fact that there exist no standard set of application benchmarks or target platforms for HC environments. Motivated by these difficulties, a new taxonomy for classifying mapping heuristics for HC environments is proposed. The Purdue HC Taxonomy is defined in three major parts: (1) the models used for applications and communication requests, (2) the models used for target hardware platforms, and (3) the characteristics of mapping heuristics. This new taxonomy builds on previous taxonomies (e.g., [2, 4, 5, 10]).

In Section 2, previous taxonomies from the fields of distributed computing and HC are reviewed. The proposed taxonomy for mapping heuristics is defined in Section 3. The benefits and possible uses of this new taxonomy are examined in Section 4.

This research was supported in part by the DARPA/ITO Quorum Program project called MSHN (Management System for Heterogeneous Networks). MSHN is a collaborative research effort among the Naval Postgraduate School, Noemix, Purdue University, and the University of Southern California. The technical objective of the MSHN project is to design, prototype, and refine a distributed resource management system that leverages the heterogeneity of resources and tasks to deliver requested qualities of service. This research is also supported in part by the DARPA/ISO BADD (Battlefield Awareness and Data Dissemination) Program. In the BADD Program, communications from a large number of heterogeneous information sources (e.g., databases, sensors) to a large number of heterogeneous destinations (e.g., warfighters' laptops, proxy servers) must be scheduled over a set of heterogeneous networks [16]. Thus, most of this taxonomy pertains to this environment also.

2. Previous Taxonomies

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

A taxonomy classifying scheduling techniques used in general-purpose distributed computing systems is presented in [2]. The classification of target platforms and application characteristics was outside the scope of this study. The taxonomy in [2] 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 mapping problem. The matching of tasks to machines also greatly affects execution schedules and system performance. Therefore, the taxonomy proposed in Section 3 also includes categories for platform characteristics and application characteristics, both of which influence matching (and scheduling) decisions.

Several different taxonomies are presented in [5]. The first is the EM^3 taxonomy, which classifies all computer systems into one of four categories, based on execution mode and machine model [4]. The taxonomy proposed here in Section 3 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 [2] is also presented in [5]. The modified taxonomy introduces new descriptors and is applied to heterogeneous resource allocation techniques. Aside from considering different parallelism characteristics of applications, target platform and application properties were not classified as part of the study.

A taxonomy for comparing heterogeneous subtask matching methodologies is included in [10]. The taxonomy focuses on static subtask matching approaches, and classifies several specific examples of optimal and sub-optimal techniques. This is a single taxonomy, without the three distinct parts of the Purdue HC Taxonomy presented in the next section. However, the "optimal-restricted" classification in [10] includes algorithms that place restrictions on the underlying program and/or multicomputer system.

The Purdue HC Taxonomy uses these studies as a foundation, and extends their concepts to the specific

HC mapping problem domain being considered. Relevant ideas from these studies are incorporated into the unique structure of the three-part taxonomy described in the next section, allowing for more detailed classifications of HC mapping heuristics.

3. Proposed Taxonomy

3.1. Introduction

As mentioned in Section 2, it is assumed that a mixed-machine HC system is composed of different machines, with possibly multiple execution models. The system is defined to be heterogeneous if any one or more of the following characteristics varies among machines enough to result in different execution performance among those machines: processor type, processor speed, mode of computation, memory size, number of processors (within parallel machines), interprocessor network (within parallel machines), etc.

The new Purdue HC Taxonomy for describing mapping heuristics for mixed-machine HC systems is defined by three major components: (1) application model and communication requests characterization, (2) platform model characterization, and (3) mapping strategy characterization. Previous taxonomies have focused only on the third item above. However, intelligent mapping decisions require information about both the hardware platform and the application being executed. Also, if there are special platform or application requirements (e.g., priorities associated with each task in a military environment), it is important that the mapping strategy be able to support these.

Thus, the Purdue HC Taxonomy classifies all three components of an HC environment, and attempts to *qualitatively* define aspects of the environment that can affect mapping decisions and performance. (Doing this *quantitatively* in a thorough, rigorous, complete, and "standard" manner is a long term goal of the HC field.) This taxonomy is based on existing mapping heuristics found in the literature, as well as previous research and experience within the field of HC.

Because research on mapping heuristics is an active and growing field, this taxonomy can only capture features of the current state of research at a certain level of detail. It is assumed that this taxonomy will be refined and expanded over time to serve as an evolving standard for describing HC mapping heuristics and their assumed environments.

3.2. Application model characterization

The first category of the taxonomy defines the *models* used for the applications to be executed on the HC system and for the communications to be scheduled on the inter-machine network. The applications themselves are not classified by functionality, but rather by the traits that define application computational characteristics that may impact mapping decisions. Furthermore, the taxonomy is able to include application traits that are subject to simplifying implementation assumptions (which may not reflect the most effective implementations), e.g., a subtask that is capable of beginning execution with a partial set of data is instead forced to wait until all input data arrives. The defining characteristics of the applications (which can be tasks or subtasks) are listed below. Many of the characteristics are also relevant to communication requests in BADD-like environments, including deadlines, mulitple versions, priorities, QoS requirements, and temporal distribution.

application size: How many tasks are in the meta-task and/or how many subtasks are in each task?

application type: What type of applications are to be mapped? If all tasks are independent, meta-task mapping is being performed. If there is a single task decomposed into subtasks (recall subtasks have dependencies), it is subtask mapping. One can also have the situation where a meta-task has independent tasks, but some of the tasks have subtasks. In this case, both meta-task and subtask mappings would be necessary.

communication patterns: What are the source and destination subtasks for each data item to be transferred?

data availability: The time at which input data needed by a subtask or output data generated by a subtask can be utilized varies in relation to subtask start and finish times: (a) is data available (to be forwarded) before a subtask completes, and (b) can a subtask begin execution before receiving all of its input data? As an example, the clustering non-uniform directed graph heuristic in [6] assumes that a subtask cannot send data to other waiting subtasks until it completely finishes executing.

deadlines: Do the applications have deadlines? This property could be further refined into hard and firm deadlines, if required. Applications completed by a firm deadline provide the most valuable results. An application that completes after a firm deadline but before a hard deadline is still able to provide some useful data. After a hard deadline has passed, data from the application is useless.

execution time model: Most mapping techniques require an estimate of the execution time of each application on each machine. How are the estimated execution times modeled? The two choices most commonly used are probabilistic and deterministic modeling. Probabilistic modeling uses a probability distribution for application execution times when making mapping decisions [1, 11]. Deterministic modeling uses a fixed (or expected) value [8], e.g., the average of ten previous executions of an application.

meta-task heterogeneity: For each machine in the HC suite, how greatly and with what properties (e.g., probability distribution) do the execution times of the different tasks in the meta-task vary?

multiple versions: Do the applications have multiple versions that could be executed? For example, an application that requires an FFT might be able to perform the FFT with either of two different procedures that have different precisions, different execution times, and different resource requirements. What are the relative "values" of the different versions to the user?

priorities: Do the applications have priorities? 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 weightings given to each priority are usually determined by another party (e.g., a system administrator). Priorities and their relative weightings are required if the mapping strategy is preemptive (Subsection 3.4).

QoS requirements: Certain application specific Quality of Service (QoS) requirements may need to be considered, such as security level.

subtask heterogeneity: Similar to meta-task heterogeneity above.

task profile: 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 [9, 13]. This information may be used by the mapping heuristic, in conjunction with analytical benchmarking (Subsection 3.3), to estimate task (or subtask) execution time.

temporal distribution: Is the complete set of tasks of a meta-task to be mapped known *a priori* (static applications), or do the tasks arrive in a real-time, non-deterministic manner (dynamic applications), or is it a combination of the two?

Because the characteristics defined above are largely independent of each other, these would all be considered flat characteristics in a taxonomy, not hierarchical characteristics with dependencies. Each of the previous taxonomies listed in Section 2 used a hierarchical structure to show relationships. The "checklist" format above is necessary to capture all of the aspects of applications that can influence mapping decisions.

3.3. Platform model characterization

The second category of the taxonomy defines the models used for target hardware platforms available within HC systems. Several existing heuristics make simplifying (but unrealistic) assumptions about their target platforms (e.g., [14] assumes an infinite number of machines are available). Therefore, this taxonomy is not limited to a set of realistic target platforms. Instead, a framework for classifying the *models* used for target platforms is provided below.

analytical benchmarks: Analytical benchmarking provides a measure of how well each available machine in the HC platform performs on each given type of computation [9, 13]. This information may be used by the mapping heuristic, in conjunction with task profiling (Subsection 3.2), to estimate task (or subtask) execution time.

communication time: How much time does it take to send data from any one machine to any other? This may be expressed as a function of path establishment time and bandwidth.

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

interconnection network: Volumes of literature already exist on the topic of interconnection networks, therefore, they are not classified here. (A general interconnection network taxonomy can be found in [3].) It is merely noted that many network characteristics can affect mapping decisions and system performance, including the following: bandwidth, ability to perform concurrent data transfers, latency, switching control, and topology. Most of these network properties are also functions of the source and destination machines.

machine architecture: 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., SIMD, MIMD, vector), memory size, number of processors (within parallel machines), and interprocessor network (within parallel machines).

machine heterogeneity: For each task (or subtask), how greatly and with what properties (e.g., probability distribution) do the execution times for this task vary across different machines in the HC suite?

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

number of machines: This property is defined by two subclasses, based on the quantity and variability of the number of machines: (a) finite or infinite, and (b) fixed or variable (e.g., new machines can come on-line). Furthermore, a given heuristic with a finite, fixed number of machines may treat this number as a parameter that can be changed from one mapping to another.

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

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

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

3.4. Mapping strategy characterization

The third category of the Purdue HC Taxonomy defines the characteristics used to describe the mapping strategies. Because the general HC mapping problem is NP-complete, it is assumed that the mapping strategies being classified are near-optimal techniques.

application model supported: See Subsection 3.2.

communication times: Are inter-subtask data communication times considered during subtask mapping?

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

data forwarding: Is data forwarding considered during mapping [15]? That is, could a subtask executing on a machine receive data from an intermediate machine sooner than from the original source?

dependencies: This property is closely related to the application type from Subsection 3.2. Meta-task mapping deals with an independent set of tasks. Subtask mapping handles the case where there are one or more tasks with subtasks and dependencies.

duplication: Can a given subtask be duplicated and executed on multiple machines to reduce communication overhead?

dynamic/static: Dynamic mapping techniques operate during (and possibly before) application execution time, and make use of real-time information. Dynamic techniques require inputs from the environment, and may not have a definite end. For example, dynamic techniques may not know the entire set of tasks to be mapped when the technique begins executing; new tasks may arrive at random intervals. Similarly, new machines may be added to the suite. If a dynamic technique has feedback, applications may be reassigned because of the loss of a machine, or application execution times taking significantly longer 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. 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.

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

execution times: The execution times of different mapping strategies vary greatly, and are an important property during the comparison or selection of mapping techniques. Can the execution time of the heuristic accurately be predicted, e.g., does the mapping heuristic perform a fixed, predetermined number of steps (e.g., greedy approaches [1]) before arriving at a mapping, or is the heuristic iterative in the sense that the mapping is continually refined until some stopping criteria is met, resulting in a number of steps that is not known *a priori* (e.g., genetic algorithms [13, 14])?

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.

feedback: Does the mapping strategy incorporate real-time feedback from the platform (e.g., machine availability times) or applications (e.g., actual task execution times) into its decisions? Strategies that utilize feedback are dynamic, but not all dynamic strategies have feedback.

objective function: The quantity that the mapping strategy is trying to optimize. This varies widely between strategies, and can make some approaches inappropriate in some situations.

platform model supported: See Subsection 3.3.

preemptive: 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, 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 discussed in Subsection 3.2.

remapping: Does the mapping heuristic require an initial mapping, which it then enhances? For example, a dynamic heuristic with feedback can remap a previous static mapping [13].

4. Conclusions

The mapping of tasks and meta-tasks, and the scheduling of communications, in HC environments are active, growing areas of research. Based on existing mapping approaches in the literature, a three-part taxonomy was proposed. The Purdue HC Taxonomy classified characteristics within the application models, target hardware platform models, and mapping strategies that are used in HC research. By defining all three categories, heterogeneous mapping techniques can more accurately be classified.

The Purdue HC Taxonomy can be beneficial to researchers in several ways. Currently, it is difficult to meaningfully compare different mapping approaches. Similarly, it is difficult to extend existing work or recognize important areas of research without understanding the relationships that exist among previous efforts. The three-part classification system provided allows HC researchers to more easily describe mapping heuristics, and to see design and environment alternatives, during the development of new heuristics, they might not have otherwise considered. A researcher can also use the taxonomy to find the mapping heuristics that use similar target platform and application models. The mapping heuristics found for similar models can then possibly be adapted or developed further to better solve the mapping problem that is being considered. In the future, this taxonomy could focus research towards the development of a standard set of benchmarks for HC environments. It is expected, as research progresses, that the Purdue HC Taxonomy will be an evolving standard, that is refined and extended to incorporate new ideas and findings.

Acknowledgments - The authors thank Ishfaq Ahmad, Shoukat Ali, Richard F. Freund, Veljko Milutinović, and Janet Siegel for their comments.

References

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

[2] T. L. Casavant and J. G. Kuhl, "A taxonomy of scheduling in general-purpose distributed computing systems," *IEEE Transactions on Software Engineering*, Vol. 14, No. 2, Feb. 1988, pp. 141–154.

[3] J. Duato, S. Yalmanchili, and L. Ni, *Interconnection Networks: An Engineering Approach*, IEEE Computer Society Press, Los Alamitos, CA, 1997.

[4] I. Ekmečić, I. Tartalja, and V. Milutinović, "A taxonomy of heterogeneous computing," *IEEE Computer*, Vol. 28, No.12, Dec. 1995, pp. 68–70.

[5] I. Ekmečić, I. Tartalja, and V. Milutinović, "A survey of heterogeneous computing: Concepts and systems," *Proceedings of the IEEE*, Vol. 84, No. 8, Aug. 1996, pp. 1127–1144.

[6] M. M. Eshaghian, ed., *Heterogeneous Computing*, Artech House, Norwood, MA, 1996.

[7] D. Fernandez-Baca, "Allocating modules to processors in a distributed system," *IEEE Transactions on Software Engineering*, Vol. SE-15, No. 11, Nov. 1989, pp. 1427–1436.

[8] 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 multiuser, heterogeneous, computing environments with SmartNet," *7th Heterogeneous Computing Workshop (HCW '98)*, Mar. 1998, pp. 184–199.

[9] A. Ghafoor and J. Yang, "Distributed heterogeneous supercomputing management system," *IEEE Computer*, Vol. 26, No. 6, Jun. 1993, pp. 78–86.

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

[11] Y. A. Li and J. K. Antonio, "Estimating the execution time distribution for a task graph in a heterogeneous computing system," *6th Heterogeneous Computing Workshop (HCW '97)*, Apr. 1997, pp. 172–184.

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

[13] H. J. Siegel, M. Maheswaran, and T. D. Braun, "Heterogeneous distributed computing," *Encyclopedia of Electrical and Electronics Engineering*, J. Webster, ed., John Wiley & Sons, New York, NY, to appear.

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

[15] M. Tan, H. J. Siegel, J. K. Antonio, and Y. A. Li, "Minimizing the application execution time through scheduling subtasks and communication traffic in a heterogeneous computing system," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 8, No. 8, Aug. 1997, pp. 857–871.

[16] M. Tan, M. D. Theys, H. J. Siegel, N. B. Beck, and M. Jurczyk, "A mathematical model, heuristic, and simulation study for a basic data staging problem in a heterogeneous networking environment," *7th Heterogeneous Computing Workshop (HCW '98)*, Mar. 1998, pp. 115–129.