Software Support for Heterogeneous Computing

HOWARD JAY SIEGEL and HENRY G. DIETZ

Purdue University, West Lafayette, Indiana ⟨hj@ecn.purdue.edu⟩, ⟨hankd@ecn.purdue.edu⟩

JOHN K. ANTONIO

Texas Tech University, Lubbock ⟨antonio@cs4sun.cs.ttu.ed⟩

As a result of advances in high-speed digital communications, researchers have begun to use collections of different high-performance machines in concert to execute computationally intensive application tasks. Existing high-performance machines typically achieve only a fraction of their peak performance on certain portions of such application programs; that is, there is a gap between average sustained performance and the machine's peak performance. One reason for this is that different subtasks of an application can have different computational requirements that are best processed by different types of machine architectures. Thus, an important approach to high-performance computing is to construct a heterogeneous computing (HC) environment, consisting of a variety of machines interconnected by high-speed links, orchestrated to perform an application whose subtasks have diverse execution requirements.

In addition to how well a subtask matches a machine, many factors must be considered to exploit optimally the power of an HC suite of machines. These include the time to move data shared by subtasks executed on different machines, the operating system overhead involved in stopping a task on one machine and restarting it on another, the ability to execute subtasks concurrently on all or some subset of the machines in the suite, and the machine and intermachine network load caused by other users of the HC system.

There are many instances of successful implementations of application tasks across suites of heterogeneous machines. Typically, however, current users of HC systems must decompose the application task into appropriate subtasks themselves, decide on which machine to execute each subtask, code each subtask specifically for its target machine, and determine the relative execution schedule for the subtasks. The automation of this process is a long-term goal in the field of HC, but research conducted toward this goal should produce tools that will aid the users of HC systems until full automation is possible.

Even though the field of HC is relatively new, it is very active. This paper is a brief outline of the software support challenges for HC addressed in Siegel et al. [1996], and readers interested in more details are referred to Eshaghian [1996], Freund and Siegel [1993], Freund and Sunderam [1994], Siegel et al. [1996], and Sunderam [1995].

The first step in using an HC system is to construct the application program. A programming language used in an HC environment must be compilable into efficient code for any machine in the HC suite, and the program specification should facilitate the decomposition of an application task into appropriate subtasks. One model for automated HC consists of four stages: (1) determina-

tion of characterization parameters, (2) task profiling and analytical benchmarking, (3) matching and scheduling, and (4) task execution.

Stage 1 generates a set of parameters relevant to both the computational requirements of the applications and the machine capabilities of the HC system using information about the expected types of application tasks and the machines in the HC suite. Categories for computational requirements and categories for machine capabilities are derived for each parameter. For example, for "floating-point operations," the computational requirements to be quantified are the number of each type of floating-point operation needed to perform the calculation, and the capabilities of the machines to be quantified are the speeds for these different types of floating-point operations.

Stage 2 involves task profiling and analytical benchmarking. Task profiling decomposes the application task into subtasks, each of which is in some sense homogeneous with respect to computational requirements. Both the code and the data upon which the specified HC system will operate must be profiled. Analytical benchmarking quantifies how effectively each of the machines in the HC suite performs each of the categories of computations being considered.

Stage 3 can use the information generated by stage 2 to derive the estimated execution time for a given subtask on a given machine and the intermachine communication overhead associated with a given assignment of subtasks to machines. These static results and dynamic information about the current loading and "status" of the machines and intermachine network allow stage 3 to create an assignment of the subtasks to machines and an execution schedule. The "status" could include whether the machines/network are fully or partially functioning due to faults, and when other tasks using the machines/network are expected to complete.

Finally, stage 4 is the execution of the given applications on the HC suite. Because the loading/status of the machines/network may change, sometimes it is necessary to reselect machines for certain subtasks by reactivating stage 3.

The study of automatic HC is a relatively new field with many open research problems. A portable HC programming language and each of the stages described above need much more research before they can be implemented in a practical way. There is also a need for debugging and performance tuning tools that can be used across an HC suite of machines. HC-specific operating system support is needed for rapidly stopping a task on one machine, transferring state information and data as needed, and then restarting the task on another machine. Research is needed in the area of software support for intermachine data transport, such as software protocols and data reformatting. Ideally, information about the current loading and status of the machines in the HC suite and the intermachine network should be incorporated into the matching and scheduling decisions. Research is needed to determine methods for collecting such information, ways to distribute it, how to organize the information, and how often to update it. There are numerous administrative issues that require software support, including what to do with priority tasks, what to do with priority users, what to do with interactive tasks, and what to do about security. Progress needs to be made on these "automation" problems (and others) just to generate tools to facilitate near-optimal practical use of HC systems in a user-specified way.

In summary, while the uses of existing HC systems demonstrate the benefit of HC, the amount of effort currently required to implement an application on an HC system can be substantial. Future research on open problems in the

area of software support for HC will improve this situation, will make the use of HC more viable, and will allow HC to realize its full potential.

REFERENCES

ESHAGHIAN, M. M., Ed. 1996. *Heterogeneous Computing*. Artech House, Norwood, MA (to appear).

FREUND, R. AND SIEGEL, H. J., Guest Ed. 1993. Special Issue on Heterogeneous Processing. *IEEE Computer 26*, 6 (June).

FREUND, R. AND SUNDERAM, V., Guest Ed. 1994. Special Issue on Heterogeneous Processing. *J. Parallel Distrib. Comput. 21*, 3 (June).

SIEGEL, H. J., ANTONIO, J. K., METZGER, R. C., TAN, M. AND LI, Y. A. 1996. Heterogeneous computing. In *Handbook of Parallel Distributed Computing*, A. Y. Zomaya Ed. McGraw-Hill, New York (to appear).

SIEGEL, H. J., DIETZ, H. G., AND ANTONIO, J. K. 1996. Software support challenges for heterogeneous computing. In *Handbook of Computer Science and Engineering*. CRC Press.

SUNDERAM, V., Ed. 1995. Proceedings of the Heterogeneous Computing Workshop (April). IEEE Computer Society Press, Los Alamitos, CA.