WARMStones: Benchmarking

Wide-Area Resource Management Schedulers

Steve J. Chapin
University of Virginia

DRAFT White Paper

Abstract

Researchers have proposed hundreds of algorithms for solutions to the distributed scheduling, or matching, problem. However, few of these algorithms have been implemented, and the primary analysis has either been purely theoretical or with limited simulation. There has been little or no performance analysis of these algorithms under realistic workloads. The result is that it is impossible to compare algorithms and to determine which work well in practice.

We propose to build a benchmarking system for matching algorithms with the following properties: (1) the system will use real, existing high-performance computing applications as a basis for the benchmark suite, (2) we will provide a scheduler implementation toolkit, allowing researchers to implement their algorithms in isolation from dependencies on particular scheduling support systems, and (3) we will simulate the execution of the matching algorithm on a variety of heterogeneous distributed system architectures. The result will be that we can assess the strengths and weaknesses of individual scheduling algorithms in a uniform manner, thus allowing comparison of algorithm performance under realistic workloads.

1. Introduction

For at least the last two decades, computer scientists have been developing *task placement* algorithms[1] for heterogeneous distributed systems. For our purposes, *heterogeneous systems* consist of multiple independent processors of varying architectures connected by a communications medium, and without shared memory or a global clock. Examples of such systems include small-scale systems such as a local-area network with a mixture of workstations from several vendors, and at the other extreme, the entire Internet.

The development of gigabit long-haul networks and high-powered commodity microprocessors has led to the development of metasystems: large-scale, wide-area distributed systems composed of heterogeneous systems from multiple administrative domains. Metasystems contain a wide range of computing and communication infrastructure: workstations, PCs, and supercomputers connected by networks ranging from Ethernet and Myrinet to gigabit ATM. Users of metasystems will write programs that can take advantage of this heterogeneity. For example, climate-modeling programs might have several modules, each best suited to run on different architectures (e.g. an ocean model running on a COTS-components-based multicomputer, an

[1] Also called *global scheduling*, *matching*, and *mapping* in the literature.

atmospheric model running on a shared-memory multiprocessor, and a visualization module running on a high-end graphics workstation).

However, properly running complex applications on the metasystem requires support for *task placement*, or *scheduling*, which is largely lacking in today's systems. The challenge of task placement in such systems can be phrased as, "Given a set of tasks T, and the set of available processors P, assign each of the tasks in T to a processor in P, subject to a set of optimizing criteria." Once a processor acquires a set of tasks, it must choose which of those tasks to run at any given time. This selection is called *local scheduling*. Figure 1 depicts both local and global scheduling operations.

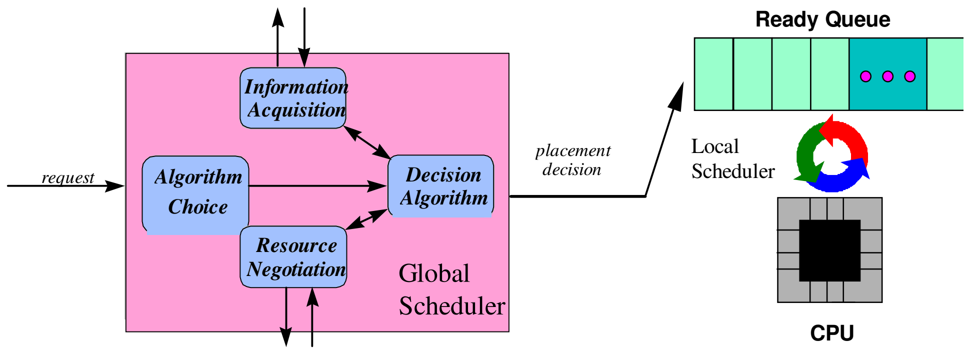

Figure 1: Global and Local Scheduling

Hundreds of algorithms have been proposed to solve the matching problem [1][4], but few of these algorithms have been implemented in real systems. This problem is aggravated by the difficulty in comparing scheduling algorithms from differing researchers. Published results are frequently based on custom simulations or data sets that favor particular schedulers; in practice, it is nearly impossible to make accurate comparisons between scheduling algorithms. Until such comparisons can be made, metasystem implementers will have difficulty choosing which algorithms to support. Worse, users will find it difficult to select among a set of supported algorithms for their particular application.

Therefore, we propose to develop WARMstones,[2] a benchmarking system for scheduling algorithms. Our system will build upon earlier work in simulation, distributed scheduling, and benchmarking. The resulting system will allow scheduling researchers to implement their algorithms on a common platform, run a standard benchmark suite over a defined range of heterogeneous distributed systems, and produce standardized performance measurements indicating the strengths and weaknesses of their algorithms.

2. Background

While development of scheduling algorithms has been an active area of research, there has been little effort devoted to implementation and full analysis of these algorithms. In most cases, analysis is limited to a theoretical "big O" evaluation of the algorithm, without regard for its applicability to real-world heterogeneous systems and application programs. In the cases where deeper analysis was done, this analysis was usually limited to a small simulation with severely limited system models and task suites artificially generated from statistical models rather than from actual job traces.

[2] WARM = Wide-Area Resource Management; "stones" follows the naming convention of other benchmarking suites.

As a result, it is nearly impossible to compare the efficacy of differing scheduling algorithms, not only from the point-of-view of evaluating research to determine its scientific merit, but also to select algorithms for implementation in scheduling systems. We propose to remedy this by developing a benchmarking system for scheduling algorithms comprising the following elements. We will develop

- a standard task representation based on well-understood concepts from the area of compilers and run-time systems. We plan to use a flow graph representation such as that found in Legion [9], where nodes represent computation and arcs represent communication and data and control-flow dependencies.

- a benchmark suite of applications that span the computing space of heterogeneous systems, including

  - scientific applications
  - multimedia applications
  - soft real-time applications
  - database and data mining applications

  The important property of these applications is that they be chosen with parameters that span the application space (e.g. application structure, ratio of computation to communication, and I/O requirements), so that we can determine the structure of applications for which a given matching algorithm is likely to perform well. We will consult with our external research collaborators, who have expertise in these areas, to determine representative applications of each type. Once we have selected these applications, we will then use our standard task representation to model them as scheduling benchmarks.

- a *scheduler's toolkit* for implementing scheduling algorithms. This will include facilities for obtaining state information for the underlying system and the task suite to be scheduled, as well as expressing various portions of a scheduling policy (the transfer, location, selection, and information policies [16]). This will be flexible enough to implement all classes of scheduling algorithms from the existing literature, and will be extensible so as to accommodate future algorithms. The toolkit will also facilitate implementation of local scheduling policies.

- a representation of the underlying heterogeneous system suitable for describing systems ranging from a small workstation cluster to a wide-area distributed system consisting of thousands of machines connected by heterogeneous communications networks. Important aspects of each component system include its number of processors, internal interconnection, memory hierarchy, local scheduling policy, operating system, and processor type and speed. This description will accommodate machines ranging from single-CPU PCs or workstations to SPMD/SIMD/MIMD multiprocessors, and will also accommodate aggregations via intermediary management software such as Condor flocks.

- a simulator that will accommodate dynamic arrival of tasks, dynamic system state, and execution of the scheduling algorithm for a particular point in the problem space defined by the input task suite and a range of underlying system descriptions. As part of its execution, the simulator will track the performance of the scheduler and the efficacy of the computed schedule. This data will form the basis of our algorithm evaluation. The simulator will also have the ability to "age" the state description information before supplying it to the scheduling algorithm, so that we can better evaluate the sensitivity of policies to delays inherent in real-world state dissemination.

In each case, we intend to build upon prior work, where applicable. To rate a given algorithm, we will iterate across the task suite and heterogeneous system space, evaluating the algorithm at each point. The resulting profile will help us to understand the strengths and weaknesses of each algorithm, and will form a basis of comparison between algorithms. It is our hope that when a researcher proposes a new matching algorithm, he will implement and rate it in WARMstones, much as other benchmarking suites (Whetstones, Dhrystones, SPEC) are used to rate new computers and compilers.

3. Module Development

We will now describe each module in greater detail, and examine related work from which we can build.

3.1 Task Representation and Benchmark Suite

There has already been extensive work in program representation in the areas of compiler implementation [9], run-time system development [7][9], and partitioning and scheduling algorithms [15]. In most cases, programs are represented as described earlier, with weighted nodes (computation) and edges (dependencies). We will start with the Legion graph representation, and augment it so that nodes can contain estimates of execution time on a variety of architectures and operating systems. Our initial augmentation will be based on the MESSIAHS task description vector, with adaptation and extension as needed.

It is important to note that we will not actually be executing the applications when we are benchmarking scheduling algorithms. Rather, we will extract their computational form, represent it in Legion graphs, and feed these graphs as input to the simulator. This is proven technology, as our MPL compiler for Legion generates graphs from C++ code. This will allow us to emulate execution of the benchmark programs in faster-than-real time while retaining the essential nature of the applications.

3.2 Scheduling Support Systems

As noted, there have been hundreds of scheduling algorithms proposed. However, there have been relatively few *scheduling systems*, which implement these algorithms, completed. Of those, the majority either encapsulate a single, inflexible algorithm which can only be slightly modified by altering input parameters [12][17], or provide a few pre-defined policies but no method of implementing general policies. Only a few systems [5][8][9] are flexible enough to allow the general expression of multiple policies—MESSIAHS was the first of these.

MESSIAHS provides two interfaces for scheduler implementation: an interpreted language for rapid prototyping (the MESSIAHS Interface Language, or MIL), and a library of C functions for higher efficiency [6]. We will use the knowledge gained in the earlier MESSIAHS work as a basis for developing our scheduler implementation toolkit. Because we are primarily interested in the quality of the schedule generated by many policies, we anticipate using an interpreted toolkit. This will not affect the relative cost of computing a schedule, and we do not expect the difference in absolute cost of generating the schedule between a compiled and an interpreted scheduler to be of importance.

3.3 System Representation

Earlier simulation projects, in both parallel and distributed systems, have developed machine/system representations (e.g., K9 [1], Osculant [13], PP-MESS-SIM [14], and Vint [10]). Scheduling systems also typically include abstract system representation, but not necessarily to the level of detail necessary for our purposes [5][11][17]. Therefore, we will devise a system description template that allows us to express system architecture, operating system, installed software, and connectivity, among other characteristics. We will start with the MESSIAHS system description vector and extend it as necessary. For example, we are currently working with the Application Level Schedulers (AppLeS [2]) group at UCSD to develop a complete basis of scheduling information.

To facilitate comparisons of scheduling algorithms, we will generate a default set of system configurations ranging from a single-CPU workstation to a metasystem composed of hundreds or thousands of machines. To create these configurations, we will select ranges of points across various axes of freedom (e.g. network characteristics, CPU characteristics, local scheduling algorithms, etc.) and generate the cross products of these spaces.

3.4 Simulator

The simulators mentioned in section 3.3 all simulate various aspects of a heterogeneous distributed system—K9 and PP-MESS-SIM simulate parallel processors, Vint can simulate general networking activity, and Osculant includes a simulator for general distributed systems. Osculant is a DARPA-funded research project at the University of Florida, and on first inspection comes closest to fulfilling our needs. However, the scope of the Osculant project is broader, and possibly overly general, than we need for a scheduling simulator. Therefore, we expect to track the development of Osculant closely as the system matures, and to lever the advances made there.

A complete run of the simulator will iterate across both the benchmark suite and the default set of system configurations. This will entail thousands of simulations for each algorithm, which will require substantial computational power. Fortunately, we have two resources upon which we can draw to accomplish this. The first is Centurion, a 128-node DEC Alpha-based multicomputer funded by the NSF and the ONR, which will provide more than 100 gigaflops of computing power. The second resource is the nationwide Legion system we will be bringing on-line in CY 1998, in conjunction with NPACI and our DOE and DOD research partners.

4 Plan of Research

We have already discussed our plans to base representations on work that has come before. The planned steps in our research are:

1. Thoroughly review the related work, including Osculant, K9, PP-MESS-SIM, and Vint. The purpose of this review is to determine the salient features of each system that will improve our design for WARMstones. Where possible, we will reuse existing designs to speed our implementation process.
2. Design the various components (system representation, simulator core, and scheduler's toolkit) which comprise the simulation system, based on the results of step 1.
3. Implement the core components (most likely in C++, so that we may build upon our earlier work with Legion).
4. Select benchmark programs and translate them into Legion graphs. This is orthogonal to the two prior steps, and our approach to this will be to start with simple tasks used primarily for system verification and add more complex tasks later.

5. To validate the simulator, we will start with fairly simple tasks and systems (e.g. a small homogeneous workstation cluster running a handful of compute-bound jobs). We will use this information as feedback into an iteration of steps 2 and 3. When we are satisfied with the validity of the simulation system, we will add more complex system configurations and benchmark programs.
6. We will work with our research partners, including the AppLeS group at UCSD and the ELI group at Sandia National Labs, to refine the scheduler's toolkit and to obtain new algorithms for benchmarking.

5 Conclusions

We have proposed to build a simulation system for distributed schedulers, and thereby to facilitate benchmarking scheduling algorithms. This system, which we call WARMstones, will allow us to make meaningful comparisons between scheduling algorithms. We will generate benchmark numbers from a wide range of scheduling policies, and use these benchmarks to guide users in their choices of scheduling algorithms for use in the Legion system.

6 References

[1] P. Beadle, C. Pommerell, and M. Annaratone, "K9: A Simulator of Distributed-Memory Parallel Processors," *IEEE/ACM Supercomputing '89*, November 1989.

[2] F. Berman and R. Wolski, "Scheduling from the Perspective of the Application," *Proceedings of the 5th International Symposium on High-Performance Distributed Computing*, August 1996.

[3] T. L. Casavant and J. G. Kuhl, "A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems," *IEEE Transactions on Software Engineering*, 14(11), Nov., 1988, pp. 1578–1588.

[4] S. J. Chapin, "Distributed and Multiprocessor Scheduling," *ACM Computing Surveys*, 28(1), March, 1996, pp. 233–235.

[5] S. J. Chapin, "Distributed Scheduling Support in the Presence of Autonomy," *Proceedings of the 4th Heterogeneous Computing Workshop*, IPPS, 1995, pp. 22–29.

[6] S. J. Chapin and E. H. Spafford, "Support for Implementing Scheduling Algorithms Using MESSIAHS," *Scientific Programming*, vol. 3, 1994, pp. 325–340.

[7] A. Chien, Karamcheti, and Plevyak, "The Concert System: Compiler and Runtime Support for Fine-Grained Concurrent Object-Oriented Languages," University of Illinois at Urbana-Champaign DCS Tech Report R-93-1815, 1993.

[8] I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," *International Journal of Supercomputer Applications*, to appear.

[9] A. Grimshaw, W. Wulf, and the Legion Team, "The Legion Vision of a Worldwide Virtual Computer," *Communications of the ACM*, 40(1), January 1997.

[10] P. Huang, D. Estrin, and J. Heidemann, "Enabling Large-scale Simulations: Selective Abstraction Approach to the Study of Multicast Protocols," Submitted to the 6th International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems (MASCOTS '98). http://netweb.usc.edu/huang/publication/mascots.ps.

[11] P. Kreuger and R. Chawla, "The Stealth Distributed Scheduler," *Proceedings of the 11th International Conference on Distributed Computing Systems*, 1991, pp. 336–343.

[12] M. Litzkow, M. Livny, and M. W. Mutka, "Condor—A Hunter of Idle Workstations," ICDCS '88, pp. 104–111.

[13] Osculant home page: http://beta.ee.ufl.edu/Projects/Osculant/index.html.

[14] J. Rexford, W. Feng, J. Dolter, and K. Shin, "PP-MESS-SIM: A Flexible and Extensible Simlator for Evaluationg Multicomputer Networks," *IEEE Transactions on Parallel and Distributed Systems*, January 1997, pp. 25–40.

[15] V. Sarkar, *Partitioning and Scheduling Parallel Programs for Multiprocessors*, The MIT Press, 1989.

[16] M. Singhal and N. G. Shivaratri, *Advanced Concepts in Operating Systems*, McGraw–Hill, 1994, ISBN 0-07-13668-1.

[17] S. Zhou, X. Zheng, J. Wang, and P. Delisle, "Utopia: A Load Sharing Facility for Large, Heterogeneous Distributed Computer Systems," *Software—Practice & Experience*, 23(12), 1993, pp. 1305–1336.