Performance Prediction for Large Scale Parallel Systems

Yuhong Wen, Geoffrey C. Fox
Northeast Parallel Architecture Center
Syracuse University
111 College Place
Syracuse, NY, U.S.A
{wen,gcf}@npac.syr.edu

Abstract

In both the design of parallel computer systems and the development of applications, it is very important to have good performance prediction tools. This paper describes a new approach -- PetaSIM, which is designed for the rapid prototyping stage of machine or application design. Computers, networks and applications are described as objects in a Java IDL (Interface Definition Language) with special attention to the proper representation of caches and hierarchical memories. PetaSIM represents a prototype for a performance specification language or PSL. We present encouraging initial results for a set of data-intensive applications from the University of Maryland. We discuss the extension of PetaSIM to support applications of the type found in distributed collaborative engineering.

Keywords: Performance Prediction, Performance Specification Language, Multi-Domain Performance Model

1. Introduction

Large-scale data-intensive parallel applications have become the leading scientific computing problems on the massively parallel machines, because of their complexity and time-consuming processing. It will be great help to the application design and the new computer architecture design of petaflop performance if we can give the performance prediction before physically running the applications on the parallel machines. There are two kinds of performance prediction approaches, *concept design level* and *detailed performance prediction*. At the concept design level performance prediction, the goal is to provide a quick and roughly correct performance prediction at the early stage of the model design of the application and/or the new computer architecture design. While the detailed performance prediction is aimed to provide the detailed information of a given application running on a specific computer system, normally, we call it performance simulation.

In this paper, we will present a performance estimator which address to the conceptual performance prediction, called PetaSIM. It is designed also with particular attention to easy interactive comparison of different system design. It is quite convenient to change application structure in PetaSIM but we have chosen to focus on cases where one has a relatively fixed application suit and wish to rapidly explore a range of system designs. A Java applet front-end is used to optimize interactive estimation.

The performance estimator PetaSIM is built around the performance prediction process sketched in Fig.1 [1]. The distinctive feather of our approach is the use of machine and problem abstractions which although less accurate than detailed complete representations, can be expected to be more robust and further quite

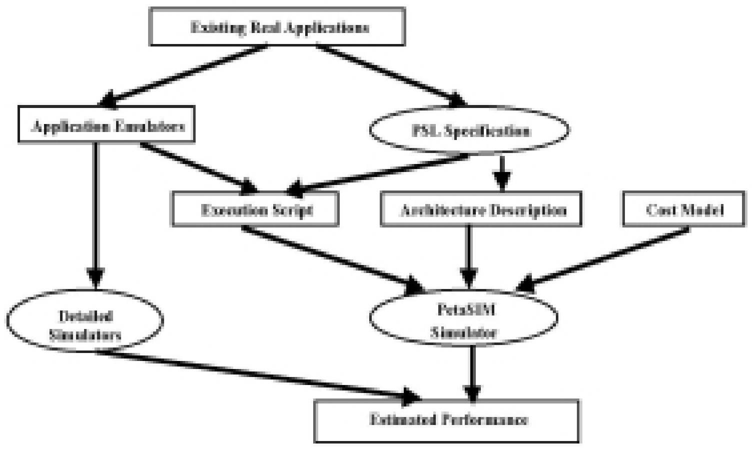

Fig.1: The Performance Prediction Process

appropriate for the rapid prototype needed in the design of new machines, software and algorithms. The hearts of this performance prediction process are two technologies - PSL (Performance Specification Language) and PetaSIM [1], [3]. In this paper, we will address the design of PetaSIM, which will take three key inputs from PSL, which describe respectively the target machines, application, script specifying execution of the application on the machine, to get an estimation of the performance of the application running on the machine. All kinds of research have showed that the performance prediction / estimation has been a very difficult problem, because of different kinds of applications, and different kinds of computer systems. It is important to design a general performance specification language (PSL) to support the performance estimation. In this paper, we will also show that PetaSIM is an initial step to suggest the characteristics of such a Performance Specification Language.

The rest of the paper is organized as follows. Section 2 presents the general concepts of the performance prediction model in our approach, and the characteristics of performance specification language (PSL). In Section 3, we will introduce the implementation of performance estimator-PetaSIM, and in Section 4, some preliminary results on real applications. Related work and further work is discussed in Section 5.

2. Performance Prediction Model

In order to give the performance prediction of large-scale complex applications on parallel architectures, we need to provide a model of their execution behavior. The main idea in our approach of the conceptual framework is to design a general model of parallel computation and the use of associative objects as the representation basis for components. The models developed in our system span three domains: *application*, *software / operating system*, and *hardware*. The application domain represents parallel computation and gives the abstract data movement and computation behavior of the applications. The software/operating system domain provides the models for task process and memory management, inter-process communication, and parallel file systems. The hardware domain provides models for the processor and memory components, where the latter includes models for cache memory as well as shared memory hierarchies.

2.1 General Model of Parallel Performance Prediction

PetaSIM is aimed at a conceptual level performance prediction - half way between detailed instruction level machine simulation and simple "back of the envelope" performance estimates. It takes care of the complexity - memory hierarchy, latencies, adaptability and multiple program components, which make even high-level performance estimates hard. It uses a crucial simplification - dealing with data in the natural blocks (or aggregates) suggested by memory systems which both speeds up the performance simulation and in many cases will lead to greater insight as to the essential issues governing performance.

PetaSIM came out to be a three-domain performance prediction model: application, software/operating system, and hardware in Fig.2. In the hardware domain, we will try to represent the detailed computing and communication capability, and the memory hierarchy of the parallel architectures, which include CPU speed, memory size, cache size, components link relationship and the inter-communication features, etc. While in the software/operating system domain, we will address the software level

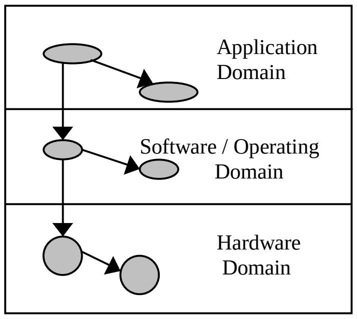

Fig.2: Multi Domain Model

memory and cache management, the task schedule approach problems. And in the application domain, we will extract the data distribution model, data movement between the memory hierarchy, and the data computation behavior.

2.2 Performance Specification Language

Because of the complexity of the various different kinds of applications, and different kinds of architecture of the parallel machines, the performance prediction of such application execution becomes rather difficult. Much research has shown that the performance prediction or estimation is a very difficult problem, because of the different kinds of applications, and multitude of distinct computer systems. It is important to design a general performance specification language to support the performance estimation. In the following of this paper, we will introduce the design and implementation of PetaSIM, and we will also show that PetaSIM is an initial step to suggest the characteristics of such a Performance Specification Language (PSL).

2.2.1 Application Domain

In the application domain, the research is focused on how to extract the data blocks, the program execution pattern and data processing behavior of the applications. Because we are aiming at the conceptual performance prediction, we will deal with data in the natural blocks (or aggregates) instead of the detailed data computing. This will benefit the performance prediction and make it more robust to provide rapid processing. Thus, we need to extract the following information from the applications.

- the size of each data block
- the number of data blocks
- the amount of data operations in the data block
- data distribution model
- data processing sequence/flow of the data blocks -- the application algorithm

2.2.2 Software / Operating system Domain

In the software/operating system domain, it is aimed to provide the information of how the parallel operating system to schedule the parallel tasks and process the data. In this level, we need to provide all the system features of the parallel machines in the performance specification language:

- the memory management approach
- the cache management approach
- parallel task schedule method

2.2.3 Hardware Domain

In the hardware domain, the problem is how to present the hardware features of the underlying parallel machines, which includes the architecture of the processing node and the topology of the inter-connection between the nodes. In this level, we need the language to provide the following hardware information.

- computing capability of each processor, which include the CPU speed and the bandwidth
- memory size and cache Size
- architecture of each processing node
- inter-connection topology of the parallel machines, which is to provide the information of communication between the processors

3. Performance Estimator -- PetaSIM

In this section, we will discuss the design and implementation of our performance estimator PetaSIM for parallel memory hierarchy system. PetaSIM takes three key inputs, which describe respectively the target machines, application, script specifying execution of the application on the machine, to get an estimation of the performance of the application running on the machine. PetaSIM takes both inputs from the application emulators (such as University of Maryland Emulators)[4] and hand written codes.

3.1 Emulators

An application emulator is a program to extract computational and data access patterns that closely resemble the patterns observed when executing a particular class of applications. In practice, an emulator is a simplified version of the real application, but contains all the necessary communication, computation and I/O characteristics of the application required for the performance prediction study. Using an emulator result in less accurate performance estimations than using full application, but it is more robust and enables fast performance predictions for rapid prototyping of new machines. An application emulator models the computation and data access patterns of the full application in a parameterized way. Adjusting the values of the parameters makes it possible to generate various application scenarios within a single class of applications.

In a simulation-based performance prediction framework, application emulator provides a specification of the behavior of the application to the simulator. Using an application emulator has several advantages over using traces from actual program runs or running the full application on the simulator. First, a trace is static and represents the behavior of the application for a single run on a particular configuration of the machine. Since an application emulator is a program that can be run on the simulator, it can model the dynamic nature of an application and can be used for different machine configurations. Second, running a real application may complicate the task of the simulator unnecessarily. By abstracting away parts of the application that are not critical to predicting performance, an application emulator can allow an efficient simulation without getting bogged down in the unimportant details of the application. Third, execution of a complete application requires the availability of real input data. Since the application behavior is only emulated, an application emulator does not necessarily require real input data, but can also emulate the characteristics of the actual data. This can enable performance studies of applications on large machine configurations with large data sets. An application emulator without a great amount of detail can be used for rapid prototyping of the performance of the application on a new machine configuration; while a highly detailed emulator can be used, for instance, to study different parallelization strategies for the application.

3.2 Petasim

There are three parts of descriptions in the *PetaSIM* performance estimation system, architecture description and application description, and the system/software description. The most general computer architectures can be specified using the PetaSIM *nodeset* and *linkset* objects while the applications can be specified using *dataset* and *distribution* objects. While the system description represent the software feature of the parallel machines.

A *nodeset* is a collection of entities with current types allowed as:

- *memory* with cache (with flushing rules) as special case
- *disk* which is essentially same as a memory.
- *CPU* where results can be calculated
- *pathway* such as a bus, switch or network cable which concentrates data

A *linkset* connects *nodesets* together in various ways. *distributions* specify the horizontal (geometrical) connectivity of *nodesets* and *linksets*. Typically these are arranged in a natural default for the classic homogeneous architectures. The default mapping is inferred from sizes of *nodesets* and done in a simple one-dimensional block fashion. The vertical (flow of information) connectivity in the architecture is specified in the execution script with defaults implied in architecture specification.

The application is specified by the *dataset* objects, whose implementation are controlled by the *distribution* objects that specify classic HPF style geometric decomposition across memories and CPU objects. The computation is specified by the execution script, which also specifies data movement.

nodeset, *linkset*, *dataset* and *distribution* are Java classes that are subclassed as necessary to give particular special cases with particular capabilities. They have methods that are defaulted for simple cases but can be overridden for complicated cases. Thus we are essentially Java as IDL (Interface Definition Language) for these core PetaSIM objects.

Much of the execution is controlled by methods in *nodeset*, *linkset* and *dataset* objects. Some typical additional commands that implicitly invoke these methods to represent the data movement and computing are:

- **send** DATAFAMILY **from** MEM-LEVEL-L **to** MEM-LEVEL-K
- Here DATAFAMILY is a *dataset* specified by name
- MEM-LEVEL-K, MEM-LEVEL-L are *nodesets* labeled by name which must be linked by a *linkset.*
- **move** DATAFAMILY **from** MEM-LEVEL-L **to** MEM-LEVEL-K
- **Use** distribution DISTRIBUTION **on** NODESET1,…,LINKSET1,…,DATASET1
- **compute** DATAFAMILY-A, DATAFAMILY-B .. **on** MEM-LEVEL-L
- **synchronize** synchronizes all processors (loosely synchronous barrier). Pipelining which is normally assumed, is stopped by this.

One of the most important motivations of PetaSIM is to provide a performance estimation tool for the new computer architecture designer to get a fast and accurate performance prediction during the conceptual architecture design. This requires that PetaSIM should be easy to operate and that it be convenient to modify features of the computer architecture. As shown in fig. 1, we chose a Java applet front end to meet this requirement in the design of PetaSIM.

The heart of the PetaSIM is a C++ simulator which takes the computer architecture description and application description to give the performance estimation. A multi-user Java Server provides the service to different Java Applet clients from the global Internet. The designers can download the applet from web site (http://kopernik.npac.syr.edu:4096/PetaSIM/V1.0/PetaSIM.html) to get easy access to the PetaSIM. PetaSIM supports both inputs from the emulators (such as University of Maryland Emulators) and hand written codes.

4. Preliminary Results

In this section we summarize some preliminary results from PetaSIM with three data-intensive applications Pathfinder, Titan and Virtual Micro-Scope from the University of Maryland. The architecture and application description files are all automatically generated by University of Maryland's emulators[4].

The results are plotted against the number of parallel SP2 nodes except for one case (Pathfinder) where we also compare results plotted against number of I/O nodes in the system. From the benchmarks we can see that the PetaSIM estimate results are quite close to the measured application's running time on SP2 machine. We illustrate PetaSIM’s abstraction of the SP2 system used in the results presented in fig. 3.

These figures show the actual wall clock time used by PetaSIM to produce the estimates. As this runs on a sequential machine and the execution script is not explicitly data-parallel, the time to get an estimation increases linearly with the number of nodes. If one looks at the estimation time for simple data-parallel systems such as finite difference problems, PetaSIM would be much faster (as just a few not a few thousand lines of execution script) and take a time roughly independent of the number of nodes on the target machine. In fact we have recently abstracted the operations in the applications shown in fig. 3 to a “primitive data parallel operation” and have correspondingly drastically reduced the time needed to produce the estimate. One can expect to initially use a simple crude loop over parallel nodes for each new type of computation. If used enough, it can be implemented in data parallel fashion and added to PetaSIM’s library of operations.

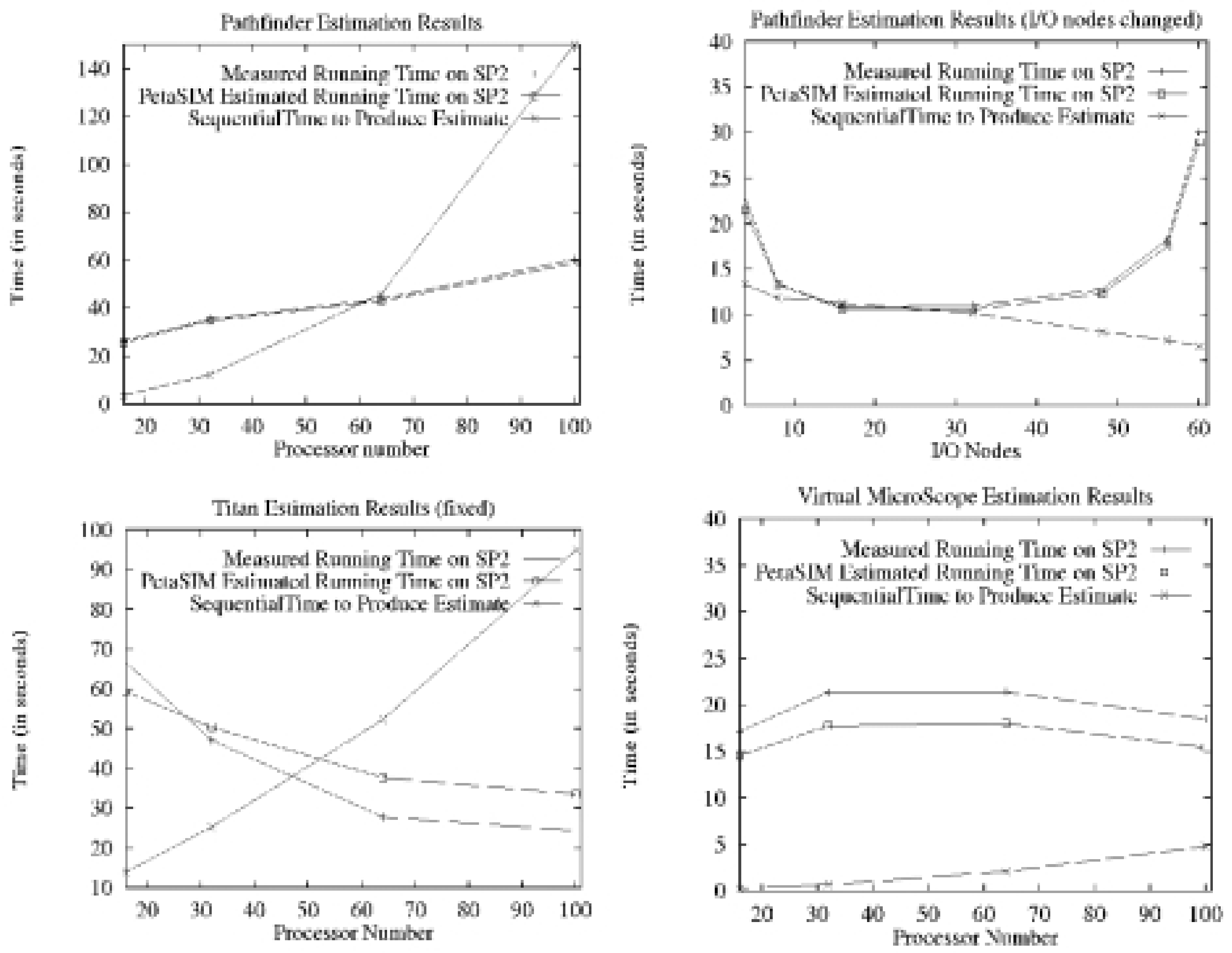

Fig.3: Measured Execution Time compared to the estimated execution time from PetaSIM for four examples. We also show the sequential execution time needed to produce the estimate.

PetaSIM provide the ability to easily modify the features of the architecture and application behavior, which helps greatly in the architecture conceptual design and get accurate performance estimation. And PetaSIM provide the interface for both inputs from the emulators (like our experience with the University of Maryland's emulators) and from the hand-written code of the system designers, which make it even more flexible.

5. Related Work and Further Work

PetaSIM bases its performance estimate on several inputs: namely the computer architecture description, *nodeset* and *linkset*, and application description, *dataset* and *distribution*, and the data operation description, *execution script*, of the application. This has similarities to the approach used by the POEMS group led by University of Texas at Austin. In the POEMS system, one divides the performance estimation into application domain, system and software domain, and hardware domain. In each domain, they provide a model to describe the features of both application and architecture. The performance estimation will be based on the information provided. [6]

Compared with some other performance estimators, PetaSIM has some special characteristics. Most of the other simulators, such as the University of Maryland's systems [4], [5], base their simulation on the task graph describing the application. PetaSIM instead uses an execution script for the application specified in ASCII format, which corresponds to a coarse grained description of the application. PetaSIM's approach appears to provide a more intuitive interface to both application and resource description, which naturally supports rapid prototyping studies over a wide range of computer architectures. In some cases PetaSIM's interpreted processing of the ASCII *execution script* [2], may be too slow and some pre-processing (compilation) of the execution script should be added as an option. Remember one of our goals was to support the study of a range of architectures for a fixed application suite and in this case, it makes sense to compile the execution script.

References:

[1] "The Petaflops Systems Workshops", Proceedings of the 1996 Petaflops Architecture Workshop (PAWS), April 21-25, 1996 and Proceedings of the 1996 Petaflops System Software Summer Study (PetaSoft), June 17-21, 1996, edited by Michael J. MacDonald (Performance Issues are described in Chapter 7).

[2] Kivanc Dincer and Geoffrey C. Fox, "Using Java in the Virtual Programming Laboratory: A web-Based Parallel Programming Environment", to be published in special issue of Concurrency: Practice and Experience on Java for Science and Engineering Computation.

[3] Geoffrey C. Fox and Wojtek Furmanski, Computing on the Web -- New Approaches to Parallel Processing-- Petaop and Exaop Performance in the Year 2007", submitted to IEEE Internet Computing, http://www.npac.syr.edu/users/gcf/petastuff/petaweb/

[4] Mustafa Uysal, Tahsin Kurc, Alan Sussman, Joel Saltz, Performance Prediction Framework for Data Intensive Applications on Large Scale Parallel Machines, University of Maryland Technical Report: CS-TR-3918 and UMIACS-TR-98-39, July 1998

[5] M. Uysal, A. Acharya, R. bennett, J. Saltz, "A Customizable Simulator for Workstation Networks", Proceedings of the International Parallel Processing Symposium, April 1997.

[6] Deelman, Bagrodia, Dube, Browne, Hoisie, Luo, Lubeck, Wasserman, Oliver, Teller, Sundram-Stukel, Vernon, Adve, Houstis, and Rice, POEMS: End-to-end Performance Design of Large Parallel Adaptive Computational Systems: Technical Report, August 1998