UNIVERSITY OF MARYLAND
Department of Computer Science
A.V. Williams Building
College Park, MD 20742-3255

Announcement :
Methods and Tools for Performance Modeling, Measurement, Analysis, Evaluation and Prediction, BAA 97-12

Technical Topic Area :
Functional Specification Methods, Model Validation and Integration, Measurement and Abstraction, Performance Specification Languages

Proposal Title :
Performance Prediction and Modeling of Compute and Data Intensive Applications on Current and Future High Performance Architectures

Technical Points of Contact :

Joel Saltz
Alan Sussman
Anurag Acharya
Jeffrey Hollingsworth
Ashok Agrawala
Computer Science Department
University of Maryland
College Park, MD 20742-3255
Phone: 301-405-2669
Fax: 301-405-2744
{saltz,als,hollings,agrawala}@cs.umd.edu

Geoffrey Fox
Northeast Parallel Architectures Center
Syracuse University
Syracuse, NY
315-443-2163
315-443-4741
gcf@npac.syr.edu

Apostolos Gerasoulis
Department of Computer Science
Rutgers University
New Brunswick, NJ 08903
gerasoul@cs.rutgers.edu
Phone: 908-445-2725
Fax: 908-445-0537

Tao Yang
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106
tyang@cs.ucsb.edu
805-893-4384
805-893-8553

Administrative Point of Contact :

Barbara O'Malley
Office of Research Administration and Advancement
2100 Lee Building
University of Maryland

College Park, Maryland 20742-5141
Phone: (301) 405-6276
Fax: (301) 314-9569
bomalley@deans.umd.edu

Type of Business : Other Educational

Section II : Detailed Proposal Information

A: Innovative claims

- Development of novel technology (Application Emulators/HLAM/PetaSIM) to support rapid prototyping of new computer architectures and new system software. This process will make it possible to provide reasonable estimates of the performance that will result from running realistic applications on future very high end architectures, such as those contemplated in the PetaFlop initiative. In order to support the need for rapid prototyping, the process will be robust and relatively fast.

- Development of application emulators that will be used to predict how targeted classes of leading edge applications will perform on very high end architectures. Application emulators will be produced for two large and crucial application classes: 1) adaptive irregular computational science and engineering applications and 2) applications that carry out data analysis, data exploration and data fusion.

- Development and formal specification of Hierarchical High Level Application Modeling Framework (HLAM). The HLAM will include mechanisms to allow users to provide: a) hierarchical description of applications, b) high level specification of a target machine, c) procedures to define how the hierarchical application description is mapped to the machine model and d) cost model to be used in performance estimation.

- Development of a simulation framework (PetaSIM) to generate performance predictions for application emulators using information provided by the high level application modeling framework HLAM. Will include support for the generation and use of data parallel aggregates as building blocks for specifying modules. PetaSIM will achieve high performance even on large applications by using event driven simulations for the coarse grain task parallel parts of an application, and by exploiting the loosely synchronous structure of SPMD data parallel components. Parallel implementation of PetaSIM will make use of optimizations pioneered by the PIs under previous DARPA contracts.

- Detailed simulation tools will be used to define the cost model for HLAM (especially in I/O area) and to selectively validate the results of PetaSIM.

- The HLAM machine description will be a basis for future extensions to current parallel programming environments, such as HPF and MPI, which currently embody a simple two level (on-processor and off-processor) view of memory.

- The development of HLAM and PetaSIM performance prediction technologies will, in future work, make it possible to develop standardized and highly efficient computational steering systems.

- Technology transfer to several HPCC communities, including commercial and academic designers of new machines and software, the PetaFlop initiative and the large HPCC education and training enterprise, who will be able to use the pedagogical value of our high level descriptions. A General Dynamics/Raytheon/SAIC team will use project technology in developing the new ship design code LAMP.

B: Deliverables

The items listed as options will be delivered if the options part of the budget is funded.

Application emulators

- Construction of two application emulators motivated by loosely synchronous adaptive applications and by data exploration and data fusion applications. An application emulator is a suite of programs that, when run, exhibits computational and data access patterns that resemble the patterns observed in a particular type of application.

- The irregular scientific application emulator will be designed to simulate the behavior of coupled adaptive unstructured mesh codes, integro-differential equations solvers and particle codes.

- The data fusion application emulator will be designed to 1) simulate data intensive applications that run on a single multiprocessor platform and 2) simulate an additional computational step where results from individual data intensive calculations are combined. The data fusion application emulator will emulate operations such as image segmentation, image registration and compositing. The emulator will carry out I/O to disk arrays. Application emulators will be validated by comparing performance (on current multiprocessor architectures) of application emulators with performance of full applications.

Options:

- The irregular scientific application emulator will be extended to simulate behavior of Monte Carlo, multipole and structured adaptive codes.

- The data fusion application emulator will be extended to take tertiary storage into account.

HLAM/PetaSIM

- Development and formal specification of Hierarchical High Level Application Modeling Framework (HLAM). The HLAM will include mechanisms to allow users to provide: a) hierarchical description of applications, b) high level specification of a target machine, c) procedures to define how the hierarchical application description is mapped to the machine model and d) cost model to be used in performance estimation.

- Development of a simulation framework (PetaSIM) that is able to generate performance predictions for application emulators using information provided by the high level application modeling framework. PetaSIM will include support for the generation and use of aggregates as building blocks for specifying modules.

Options:

- Development of runtime/compile time techniques that support semi-automatic generation of HLAM/PetaSIM aggregates and modules.

- PetaSIM support for additional collective data movement primitives, which will increase the range of multi-phase loosely synchronous problems handled. This corresponds to optimized support of the advanced MPI calls, such as ones for collective communication.

- Development of optimized parallel implementation of PetaSIM. We will focus our efforts on paralleling the data parallel portions of PetaSIM but will tackle the event driven portions of the simulation as necessary. The computational demands associated with PetaSIM will vary with the structure of a problem's HLAM hierarchical graph. In an adaptive problem, the structure of the hierarchical graph will change as the program progresses. We consequently anticipate that an efficient PetaSIM implementation will require the use of adaptive load balancing methods.

Detailed Performance Simulation and Validation of HLAM/PetaSIM

- We will use the application emulators to produce application and machine specifications at varying levels of granularity and then use PetaSIM to estimate performance obtained on selected current and future architectures. We will use detailed simulation tools developed at Maryland and at other sites to characterize the performance of the application emulators on selected current architectures (e.g. IBM SP-2) and on a limited range of future architectures.

- We will use various techniques, including detailed simulation tools, instrumented static and runtime compilation and analytic models, to produce PetaSIM cost models.

C: Statement of Work

Our goal is to develop methodologies that will make it possible to provide approximate predictions of the performance that could be achieved by sophisticated new applications on new high performance architectures. While many of the techniques we develop will apply to any type of application, we will focus on two broad classes of applications. *Loosely synchronous adaptive applications* include adaptive structured or unstructured multigrid codes, particle methods and multipole codes. *Data exploration and data fusion applications* are codes that carry out processing, analysis and exploration of one or several very large data sets. This class includes codes that carry out analysis, exploration and fusion of sensor data from sensors located on different platforms (e.g. satellites, aircraft and ships), and codes that carry out sensor data analysis and fusion of data from conventional high power microscopy, confocal microscopy, electron microscopy, computerized tomography, and magnetic resonance imaging.

The groups involved in this proposal will leverage their extensive experience with high-end applications through a multi-level process. We list the following salient features of this process:

1. Construction of *application emulators* motivated by loosely synchronous adaptive applications and by data exploration and data fusion applications. An application emulator is a suite of programs that, when run, exhibits computational and data access patterns that resemble the patterns observed in a particular type of application.

2. Development of a *hierarchical High Level Application Modeling framework* (HLAM). The HLAM will include mechanisms to allow users to provide: a) a hierarchical description of an application, b) a high level specification of a target machine, c) procedures to define how the hierarchical application description is mapped to the machine model and d) a cost model to be used in performance estimation.

3. Development of a simulation framework (PetaSIM) that is able to generate performance predictions using information provided by the hierarchical high level application modeling framework

4. Performance optimization of HLAM and PetaSIM. PetaSIM will itself likely be a highly irregular parallel application and it may be necessary to develop an efficient parallel implementation of PetaSIM.

5. Validation of the HLAM/PetaSIM modeling process. We will use the application emulators to produce application and machine specifications at varying levels of granularity, and then use PetaSIM to estimate performance obtained on selected current and future architectures. We will use detailed simulation tools developed at Maryland and at other sites to characterize the performance of the application emulators on selected current architectures (e.g. IBM SP-2) and on a limited set of future architectures.

6. We will use various techniques, including detailed simulation tools, instrumented static and runtime compilation and analytic models, to produce PetaSIM cost models.

One of the components in the project is a hierarchical high level application modeling framework (HLAM), which is based on existing research in the group, but will be substantially generalized in this project, as well as being integrated into a powerful simulation system. We believe there are several advantages in using simplified application abstractions rather than the full application. First, it is expensive to simulate full applications - especially large problems on future high performance (PetaFlop) systems. Second, use of a well chosen abstraction can lead to better understanding of the key features that determine performance. Finally, abstractions can be generic and easily represent a broader class of applications than any one full application. In HLAM, we first hierarchically divide an application (sometimes called in this context a *meta-problem*) into modules. A sophisticated parallel application may be composed of several coupled parallel programs. Parallel programs can themselves be viewed as being composed of a collection of parallel modules. These modules may be explicitly defined by a user, or the modules may be generated by a semi-automatic process, such as an HPF compiler. Modules that represent distinct programs may execute on separate nodes of a networked meta-computer. An individual module may be sequential or data parallel. We might use a data parallel module to represent a multi-threaded task that runs on a multiprocessor node. HLAM will include a wide range of applications, including data-intensive applications (including I/O from remote sites) and migratory Web programs.

Another component of the proposal is a performance simulator, PetaSIM, which is aimed at supporting the (conceptual and detailed) design phases of parallel algorithms, systems software and hardware architecture. PetaSIM is aimed at a middle ground - 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, adaptivity and multiple program components which make even high level performance estimates hard. It uses a crucial simplification - dealing with data in the natural blocks (called *aggregates* in HLAM) 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 defines a general framework in which the user specifies the computer and problem architectures and the primitive costs of I/O, communication and computation. The computer and problem can in principle be expressed at any level of granularity - typically the problem is divided into aggregates which fit into the lowest interesting level of the memory hierarchy which is exposed in the user specified computer model. Note the user is responsible for deciding on the "lowest interesting level" and the same problem/machine mapping can be studied at different levels depending on what part of the memory is exposed for user(PetaSIM) control and which (lower) parts are assumed under automatic (cache) machine control. The computer and problem can both be described hierarchically, and PetaSIM will support both numeric and data intensive applications. Further, both distributed and shared memory architectures, and various combinations of those architectures, can be modeled.

We will provide both a C and Java version of the simulator, whereas the user interface will be developed as a Java applet. The visualization of the results will use a set of Java applets based on extensions to NPAC's current Java interface to Pablo performance monitoring system from the University of Illinois.

The final component in the proposal is a set of *application emulators*. An application emulator is a suite of programs that, when run, exhibits computational and data access

patterns that resemble the patterns observed in a particular type of application. We will construct two application emulators motivated by loosely synchronous adaptive applications and by data exploration and data fusion applications. As described earlier, application emulators will be used to validate the HLAM/PetaSIM modeling process. We will use the application emulators to produce application and machine specifications at varying levels of granularity and then use PetaSIM to estimate performance obtained on selected current and future architectures. We believe that our application emulators address some of the key applications targeted at future very-high end architectures. The application emulators will be shared with the performance modeling community. We believe that general availability of application emulators will help the community focus attention on crucial application classes.

We will develop an application emulator to model the performance characteristics of three classes of irregular adaptive scientific computations, along with coupled versions of multiple instances of any of these classes. The targeted computation classes are: (1) adaptive unstructured codes (e.g. unstructured multigrid solvers, integro-differential equation solvers and molecular dynamics codes), (2) structured adaptive codes, (e.g. adaptive multigrid algorithms), and (3) particle codes (e.g. Direct Simulation Monte Carlo methods, Rokhlin-Greengard or Barnes-Hut Fast Multipole codes, particle-in-cell codes).

We will also develop an application emulator that will reproduce application characteristics found in many defense and high-end civilian applications that involve sensor data analysis, sensor data fusion and real time sensor data processing. We are focusing on emulating application scenarios that will be of practical relevance in a 5 to 15 year time frame. The application emulator will be coded as a stripped-down application suite that runs on distributed collections of multiprocessors and networked workstations.

The application emulator will emulate this ambitious sensor data fusion application suite in a parameterized fashion. Parameter adjustment will make it possible to use the emulator for various application scenarios. The behavior we emulate will include computation, secondary storage accesses, tertiary storage accesses, remote object invocations, and program migration between processing nodes.

D: Results, Products and Technology Transfer

Results and Products

This project has the typical features of a significant experimental computer science activity. The work we propose is novel and as such has a major research component, which will be described in conventional journal and conference papers. However there are several major software system components that will produce artifacts of general value. These will consists of software realizations as well as innovative designs that will be documented by internal memos as well external publications.

All the software we develop will be in the public domain and highlights of our work will include:

- Development of two *application emulators*, which will emulate a total of 4-8 applications. The emulators will be simplified from real applications, but contain the essential communication, computation, synchronization and I/O characteristics that impact performance. The emulators will allow generic performance studies of important application classes.

- Development of a multi-level *hierarchical application modeling framework*, HLAM, where the products include design as well as a Java graphical user interface allowing interactive specification. HLAM will specify an application in terms of the smallest data blocks (called aggregates) needed to reliably understand performance.

- HLAM will also have an API to allow its generation from static and runtime compilation of application emulators.

- Development of a hierarchical machine specification capable of defining architectures expected in the next ten years, including those identified in the PetaFlop workshop studies.

- Development of a coarse grain simulator, PetaSIM, which will take HLAM as input, together with a machine specification, cost models and an execution script.

- Our activity will naturally produce reports describing and assessing the results of PetaSIM, which will validate both the approach and particular high level machine and application HLAM specifications.

- PetaSIM in its base form will be produced in both C and Java allowing standalone and applet execution. The system will be made available on a Web site with both the client download (Java) and higher performance (CGI or more modern servlet) server side implementation.

- Parallel implementation of PetaSIM is necessary for very large scale adaptive problems, and is a program option.

Technology Transfer

Electronic distribution and traditional scholarly activities will allow a broad based distribution. We pointed out that extensive use of Java will help maximize the ease of use and dissemination of our products. However we can also identify several special pro-active outreach (technology transfer) functions. First, we will work with DARPA and other funded projects from this BAA to establish connections with key users of our technology. These include commercial and academic designers of new hardware and software systems. We already have the connection with the PetaFlop studies, where Fox and Saltz have been ongoing leading contributors. Another interesting (and perhaps unexpected) area is the HPCC education and training community, as we believe our products will have significant pedagogical value stemming from their high level structure. Here we will target NSF, DoD modernization and DoE ASCI activities. We are well connected with these organizations already. The particular selected applications will directly benefit from our work and this is illustrated by SAIC for ship design, Johns Hopkins for the virtual microscope and the NSF Black Hole collaboration for their simulation.

In a different way, we expect our project will produce and validate a hierarchical machine description that we expect to have great value in other HPCC areas, including the design of extensions to parallel environments such as MPI and HPF.

Related Technology Transfer Activities

We have a good record for technology transfer with related HPCC activities. Syracuse's first DARPA award (1991-94) resulted in the Portland Group licensing the prototype HPF compiler produced and Portland Group is also using Maryland's advanced technology for compilation of irregular problems. We have also been active in the HPF forum for the initial (HPF1) and recent extension (HPF2) where again the Maryland work was critical in making clear that one can and should support irregular problems. Both Maryland and Syracuse are members of CRPC (Center for Research in Parallel Computation), which has an excellent record in technology transfer. We intend to incorporate the results of this performance project into our research compilers and as in the past we thereby will be able to transfer this technology to the commercial compiler industry. The Portland Group (PGI) and Applied Parallel Research (APR) have been leading third-party vendors for HPF compilers. Both PGI and APR found it straightforward to directly adapt the Maryland PARTI/CHAOS runtime techniques, thus supporting irregular problems in their compilers. The PGI compiler originally incorporated the actual PARTI source code from Maryland before they developed customized routines, still employing the inspector-executor paradigm.

Under the DARPA HPCD project, Rutgers has collaborated with the ship division of SAIC for the parallelization and improvement in robustness of the LAMP ship design codes. The parallel codes have been transferred to the Navy for their internal use. The LAMP system was recently used in the design of the revolutionary Arsenal Ship configuration developed by the General Dynamics/Raytheon/SAIC team and is a central application in this proposal. The proposed techniques can be used to assist this team and DARPA in selecting appropriate future machines based on the simulated performance study. The PYRROS system and related clustering techniques produced by Rutgers/UCSB under the HPCD project

have been used by several other research groups including MIT, Berkeley, UMD, LIP, Tennessee, CMU, and NASA. The NSF is sponsoring interactions between LIP (France) and Rutgers/UCSB for studying automatic task graph generation systems to extend PYRROS techniques. UCSB is working with Navy NRAD for transferring parallel systems techniques developed in the NSF/DARPA/NASA sponsored digital library project on Navy parallel machines. UCSB is also sponsored by SUN MICRO for developing next generation SCI-based workstation clusters. The proposed performance technology will be used by them to evaluate/select new architectures.

Given this excellent record, we believe that we will be able to effectively technology transition the results of this project in both specific (particular applications) and general cases (broad based technology such as PetaSIM and HLAM).

E: Other DARPA Contracts

Maryland

Scalable Runtime Support for Sparse and Adaptive Computations

- Contract Numbers NASA #NAG-1-1485 and ARPA #8874, through NASA Langley Research Center, DARPA/ITO

KQML-Accessible, High-Performance, Massive Knowledge Bases

- Contract Number ONR #N00014-94-1-0907, through Office of Naval Research/Advanced Research Projects Agency, DARPA/SISTO

Common Runtime Support for High Performance Parallel Languages

- Contract Number Hanscom AFB #F19628-94-C-0057, U. S. Air Force, through Syracuse University, DARPA/ITO

Scalable I/O Project - Performance Monitoring of I/O and Computer Intensive Parallel Applications

- ARPA Contract #DABT63-94-C-0049, through Caltech Subcontract #9503, DARPA/CSTO

Rutgers/UCSB

Hypercomputing and Design(HPCD)

- DABT-63-93-C-006, ARPA Order A801, DARPA/CSTO

A Scheduling and Data mapping system for high performance multiprocessing

- DABT-63-93-C-006, DARPA/ITO

Syracuse

Common Runtime Support for High Performance Parallel Languages

- US Air Force – Hanscom Air Force Base # F19628-94-C-0057, from DARPA/ITO

F: Cost, Schedule and Milestones

Year 1:

Initial development and formal specification of Hierarchical High Level Application Modeling Framework (HLAM).

Initial definition of PetaSIM and the delineation of the relationship between PetaSIM and HLAM. Construction of first HLAM and PetaSIM prototypes.

Initial versions of data fusion and adaptive application emulators. These first emulators will emulate only a single irregular or data parallel application. The adaptive application emulator will focus on emulation of unstructured mesh and particle codes. The data fusion application emulator will emulate processing and I/O associated with data exploration.

Year 2:

A variety of HLAM representations at different levels of granularity will be generated for both application emulators. Initial performance predictions will be carried out. Results will be validated on current multiprocessor architecture using detailed simulation techniques.

Application emulators extended to represent coupled applications running on network-connected collections of multiprocessors.

Application emulator performance validated through comparison with performance of real applications.

Year 3:

HLAM representations generated for application emulators running on network-connected collections of multiprocessors.

Irregular problem aggregation routines incorporated into HLAM/PetaSIM software environment.

Irregular adaptive application emulators extended to represent coupled adaptive unstructured mesh codes, integro-differential equations solvers and particle codes. Data fusion application emulators extended to emulate multiple coupled applications that carry out image segmentation, image registration and compositing.

Options:

Year 1:

The irregular scientific application emulator will be extended to simulate behavior of Monte Carlo, multipole and structured adaptive codes.

Year 2:

The data fusion application emulator will be extended to take tertiary storage into account.

PetaSIM support added for additional collective data movement primitives, which will increase the range of multi-phase loosely synchronous problems handled.

Begin development of optimized parallel implementation of PetaSIM.

Year 3:

PetaSIM cost function will be extended to take tertiary storage into account; results validated using virtual microscope application.

Development of runtime/compile time techniques that support semi-automatic generation of HLAM/PetaSIM aggregates and modules.

Optimized parallel implementation of PetaSIM.

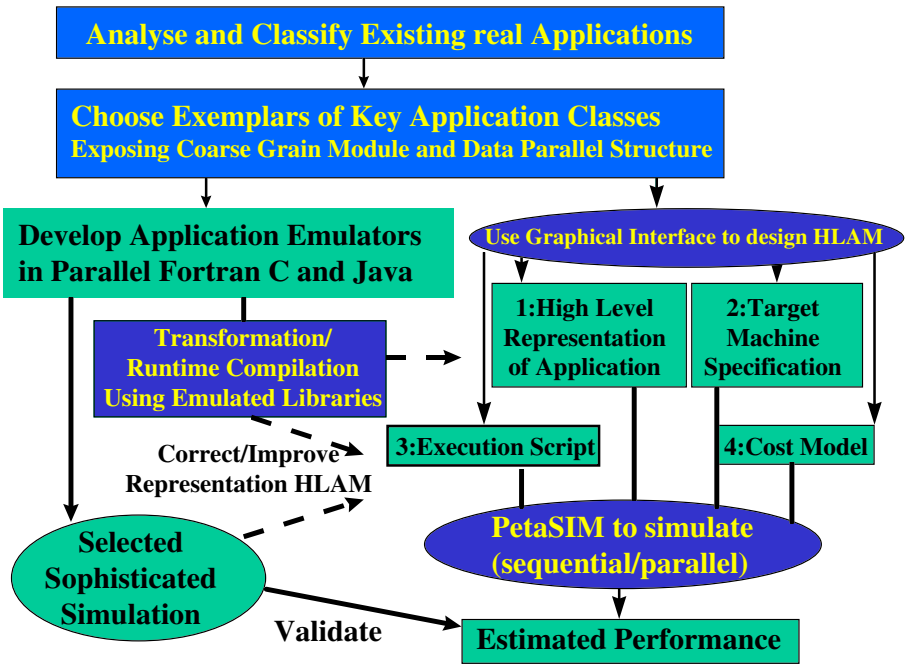

Figure 1: The Performance Prediction Process

G: Technical Rationale and Approach

1 Overview

Our goal is to develop methodologies that will make it possible to provide approximate predictions of the performance that could be achieved by sophisticated new applications on new high performance architectures. While many of the techniques we develop will apply to any type of application, we will focus on two broad classes of applications. *Loosely synchronous adaptive applications* include adaptive structured or unstructured multigrid codes, particle methods and multipole codes. *Data exploration and data fusion applications* are codes that carry out processing, analysis and exploration of one or several very large data sets. This class includes codes that carry out analysis, exploration and fusion of sensor data from sensors located on different platforms (e.g. satellites, aircraft and ships), and codes that carry out sensor data analysis and fusion of data from conventional high power microscopy, confocal microscopy, electron microscopy, computerized tomography, and magnetic resonance imaging.

The groups involved in this proposal will leverage their extensive experience with high-end applications through the process depicted in Figure 1. We list the following salient features of this process:

1. Construction of *application emulators* motivated by loosely synchronous adaptive applications and by data exploration and data fusion applications. An application emulator is a suite of programs that, when run, exhibits computational and data access patterns

that resemble the patterns observed in a particular type of application.

2. Development of a *hierarchical High Level Application Modeling framework* (HLAM), as shown in Figure 2. The HLAM will include mechanisms to allow users to provide: a) a hierarchical description of an application, b) a high level specification of a target machine, c) procedures to define how the hierarchical application description is mapped to the machine model and d) a cost model to be used in performance estimation.

3. Development of a simulation framework (PetaSIM) that is able to generate performance predictions using information provided by the hierarchical high level application modeling framework

4. Performance optimization of HLAM and PetaSIM. PetaSIM will itself likely be a highly irregular parallel application and it may be necessary to develop an efficient parallel implementation of PetaSIM.

5. Validation of the HLAM/PetaSIM modeling process. We will use the application emulators to produce application and machine specifications at varying levels of granularity, and then use PetaSIM to estimate performance obtained on selected current and future architectures. We will use detailed simulation tools developed at Maryland and at other sites to characterize the performance of the application emulators on selected current architectures (e.g. IBM SP-2) and on a limited set of future architectures.

6. We will use various techniques, including detailed simulation tools, instrumented static and runtime compilation and analytic models, to produce PetaSIM cost models.

In the next section we describe this performance prediction process in more detail while Section 3 explains the application emulation approach and our approach to choosing applications which will test and motivate our project. In Section 4, we describe how we will integrate the various components of our activity.

2 The Performance Prediction Process

This project is built around the performance prediction process sketched in Figure 1. The distinctive feature 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 also quite appropriate for the rapid prototyping needed in the design of new machines, software and algorithms. At the heart of this performance prediction process are two technologies - HLAM and PetaSIM. These originate in the work of the Rutgers/UCSB and Syracuse respectively but will be generalized and fully integrated in this project. Further the HLAM/PetaSIM performance prediction system will interface to the application emulators developed at Maryland using a combination of static and runtime compilation techniques from all three groups. We refer generically by HLAM to the four key inputs to PetaSIM that describe the target machine (Sec. 2.1), application (Sec. 2.2), script specifying execution of the application on the machine (Sec. 2.3) and finally the cost model for basic communication, I/O and computation primitives (Sec. 2.4). Section 2.3 describes PetaSIM in some detail, including both basic concept and potential implementation.

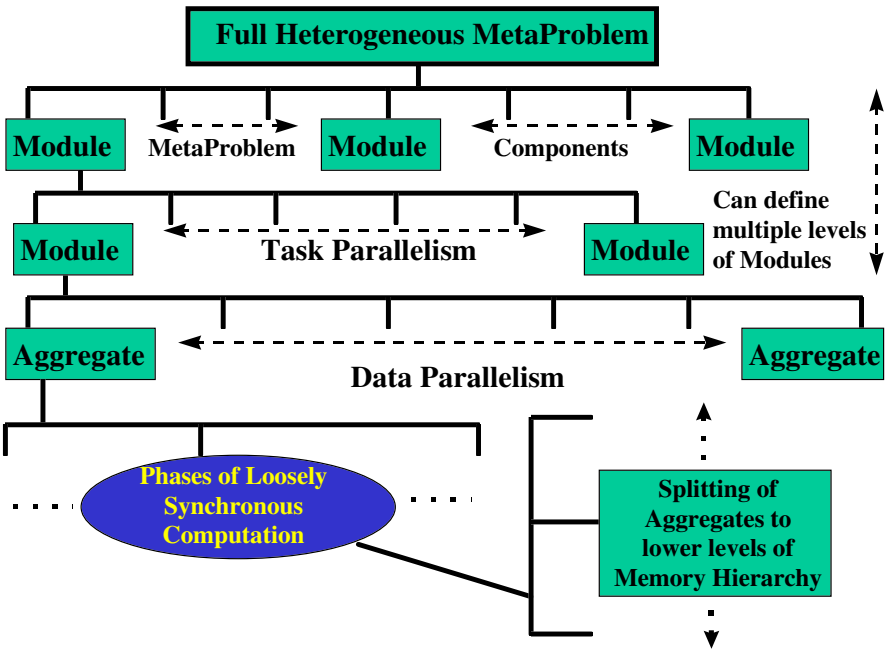

Figure 2: HLAM - the Hierarchical Application Model

2.1 Target Machine Specification

There is a close relationship between performance modeling and the description of applications needed to allow either the user or compiler to properly map an application onto parallel systems. In general, reliable performance estimates need the same level of machine description as is needed to specify parallel programs in a way that allows good performance to be obtained when executing on the target machine. This machine description can either be explicit (as in MPI [57]) or implicit as in an automatic parallelizing compiler which must essentially use such a machine description to define its internal optimizations of data placement and movement. Therefore to be effective in estimating performance on a target machine, PetaSIM must take as input an architectural description at the same level needed by parallel programming environments. The PetaSoft meetings identified the need for such architectural descriptions as essential in defining future extensions to parallel languages whether they be message or data parallel. MPI and HPF [42] implicitly treat parallel systems as a three level memory hierarchy (local processor memory, remote memory and disk). This model is inadequate for some current and nearly all future expected high performance systems. Thus an important product of our project will be such a machine description that targets both today's (distributed shared memory) machines and future designs typified by those examined in the PetaFlop process. That process looked at *extrapolated conventional*, superconducting and Processor in Memory (PIM) designs, and our proposed specification is appropriate for these three alternatives [23]. As was discussed earlier, this machine description will be helpful in developing future parallel programming environments. We expect experiences from our project to drive new developments in this field, as we will determine which features of

application and machine are performance critical and will use (reliable) models of expected complex memory hierarchies, not waiting for new hardware to become available.

Our proposed machine description in HLAM will allow specification of the number of levels in the memory hierarchy, their sizes and data movement (latency and bandwidth) times. These primitive machine operations will include collective, as well primitive, operations and cover both data movement and data replication (as in messaging and cache operation).

2.2 HLAM – Hierarchical High-Level Application Modeling Framework

Critical to our project will be a hierarchical high level application modeling framework (HLAM), which is based on existing Rutgers/UCSB research but will be substantially generalized in this project, as well as being integrated into a powerful simulation system. We believe there are several advantages in using simplified application abstractions rather than the full application. First, it is expensive to simulate full applications - especially large problems on future high performance (PetaFlop) systems. Second, use of a well chosen abstraction can lead to better understanding of the key features that determine performance. Finally, abstractions can be generic and easily represent a broader class of applications than any one full application. This is not to say detailed simulations of full applications are not of value – rather we argue that our high-level abstractions have intrinsic complementary value. HLAM is illustrated in Figure 2, which shows that we first hierarchically divide an application (sometimes called in this context a *meta-problem*) into modules. A sophisticated parallel application may be composed of several coupled parallel programs. Parallel programs can themselves be viewed as being composed of a collection of parallel modules. These modules may be explicitly defined by a user, or the modules may be generated by a semi-automatic process, such as an HPF compiler. Modules that represent distinct programs may execute on separate nodes of a networked meta-computer. An individual module may be sequential or data parallel. We might use a data parallel module to represent a multi-threaded task that runs on a multiprocessor node. HLAM will include a wide range of applications, including data-intensive applications (including I/O from remote sites) and migratory Web programs. Some key features of the proposed HLAM are:

1. A hierarchical graph representation.
2. Symbolic specification of problems, so that the system can be used to test arbitrarily large problem instances.
3. Use of *aggregates* as building blocks for specifying application modules in a hierarchical/multi-level manner that includes both task and data parallelism.
4. *Aggregates* are chosen as the largest possible unit of data parallelism that can specify the problem at the level needed to model performance with the required precision and grain size.
5. Model various types of data interaction (loose synchronization and other dependencies such as pipelining) between program components. Explicit support of the loosely synchronous structure present in essentially all large scale data parallel applications.

6. Modeling of dynamic relationships between program components for runtime adaptive prediction/optimization.

One initial focus will be to develop a hierarchical behavioral representation for MPI-based SPMD parallel code. Previous research on representing parallelism in sequential code will benefit our project. For example, control and data dependence can be modeled through program dependence graphs [22]. Hierarchical task graphs [33] were developed for modeling functional parallelism and loop parallelism. Such a graphical structure has also been studied in the SISAL project for functional parallelism [20]. For modeling MPI-based parallel programs, we will abstract not only hierarchical control structures, but also important multiprocessing events such as message sends and receives, reduction operations, global communication and barrier synchronizations. Thus the graphical structure of an MPI program will consist of basic computation components, communication and I/O primitives, and multi-level control over these components. A basic computation is a code segment that does not involve I/O and interprocessor communication. Basic computation blocks are modeled at a coarse-grain level if possible so that the performance impact of the multi-level memory hierarchy can be studied at a block level. Computation primitives from software building blocks such as the BLAS and LAPACK math subroutines can be used to abstract basic computations.

2.3 PetaSIM – *A Performance Simulator for parallel hierarchical memory computers*

2.3.1 Introduction and Motivation

Central to this proposal is a performance simulator, PetaSIM, which is aimed at supporting the (conceptual and detailed) design phases of parallel algorithms, systems software and hardware architecture. Originally this was designed as a result of two week-long workshops - PAWS and PetaSoft - aimed at understanding hardware and software architectures ten years from now when Petaflop (10^{15}) scale computing can be expected. It was clear from these meetings that the community needed better aids for performance estimation, as the 8 groups (and 8 different machine designs) present found it difficult to compare designs and in particular differed by a factor of one million in estimating the performance of a set of extremely simple algorithms - the PetaKernels – on their new designs. The most sophisticated PetaKernel was a regular finite difference problem solved by simple Jacobi iteration. These workshops emphasized the need for tools to allow the description of complex memory hierarchies (present in all future and most current machine designs) and the mapping of problems onto them in a way that allows reliable (if high level) performance estimates in the initial design and rapid prototyping stages.

PetaSIM is aimed at a middle ground - 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, adaptivity and multiple program components which make even high level performance estimates hard. It uses a crucial simplification - dealing with data in the natural blocks (called *aggregates* in HLAM) 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. We motivate and illustrate

the design of PetaSIM by the well known formulas for parallel performance of simple regular applications on nodes without significant memory hierarchy. Then (Chapter 3 of [24]) one finds,

SpeedUp = Number of Nodes/(1 + Overhead)

where the Overhead is proportional to $(Grain\ Size)^{-g}(t_{comm}/t_{calc})$,

where in this case the natural data block size is the $Grain\ Size$, or number of data points located in each node. The power g measures edge over area effects and is $1/d$ for a system of geometric dimension d. (t_{comm}/t_{calc}) represents a ratio of communication to compute performance of the hardware. Such a formula shows the importance of identifying natural data blocks, and how such high level analysis allows one to understand the relation of performance to the memory size, I/O and computation capabilities of the architecture. PetaSIM generalizes this "back of the envelope" estimate to more complex problems and machines. It also includes not just primitive messaging performance (node to node as used in the above estimate), but also collective (such as multi-cast) mechanisms that are present in most applications but ignored in many simple analyses. Note that the simple performance estimate above is valid (with straightforward generalizations) on machines with a simple two level distributed memory hierarchy - namely memory is either on or off processor - which is essentially the model built into the current generation of parallel programming systems as typified by HPF or MPI. As we described in Section 2.1, it is essential to generalize this machine model whether we want to provide input to either parallel program generation tools or to performance estimation systems. Thus we believe our experience with HLAM and PetaSIM will be very valuable in helping to design the new generation of parallel programming environments needed for the increasingly complex high performance systems coming online.

Fortunately we have good evidence that we can generalize this naïve analysis to more complex problems and machines, since the Rutgers/UCSB group has studied granularity issues [31, 64] to identify natural data block sizes and computation clusters based on computation/communication ratios in more general hierarchical memories. They have developed preliminary performance prediction and optimization tools (PYRROS [63], D-PYRROS [41], RAPID [25]) based on task graphs in which the impact of a single-processor memory hierarchy is addressed at the intra-task level and the impact of interprocessor communication delay is identified at the inter-task level. These techniques have been shown to be effective for a number of adaptive and static applications [29, 28, 21, 26] and will be integrated into PetaSIM, as this technology is the basis of HLAM, as described in Section 2.2.

As well as a machine description, PetaSIM requires a problem description, and will use the HLAM described in Section 2.2. Application emulators (further described in Section 3 also use the same "aggregate" description of applications needed by PetaSIM and will also be used as input. It is not clear yet if we will convert the application emulators directly to the HLAM or design a separate API for this style of program description.

2.3.2 Operation of PetaSIM

PetaSIM defines a general framework in which the user specifies the computer and problem architectures and the primitive costs of I/O, communication and computation. The computer and problem can in principle be expressed at any level of granularity - typically the

problem is divided into aggregates which fit into the lowest interesting level of the memory hierarchy which is exposed in the user specified computer model. Note the user is responsible for deciding on the "lowest interesting level" and the same problem/machine mapping can be studied at different levels depending on what part of the memory is exposed for user(PetaSIM) control and which (lower) parts are assumed under automatic (cache) machine control. The computer and problem can both be described hierarchically, and PetaSIM will support both numeric and data intensive applications. Further, both distributed and shared memory architectures, and various combinations of those architectures, can be modeled.

We assume the HLAM model shown in Figure 2, where an application is viewed as a meta-problem that is first divided into modules that are task parallel. Each module is either sequential or data parallel and further divided into aggregates defined by the memory structure of the target machine. The multi-level module structure in HLAM will be treated conventionally as an event driven simulation in PetaSIM. However the large scale data parallel parts (collections of aggregates) will be treated differently by a novel mechanism that exploits the loosely synchronous SPMD structure of this part of the problem.

Synchronous and loosely synchronous (and the typically easier embarrassingly parallel) data parallel modules will be supported. As shown in Figure 2, these are divided into phases - in each phase all aggregates are computed - with synchronization (typically I/O and/or interprocessor communication) at the end of each phase. Modules must have a fixed distribution into aggregates during each phase but can be redistributed at phase boundaries. Aggregates are the smallest units into which we divide problem and could be, for example, a block of grid points or particles, depending on the application. The basic idea in PetaSIM is to use a sophisticated spreadsheet. Each cell of the spreadsheet represents a memory unit of the computer and they are generated from the hierarchical machine description described in Section 2.1. For each phase of the module simulation, the user provides a strategy – called the execution script in Figure 1 – that moves the module aggregates through the cells. The simulator accumulates the corresponding communication and computation costs. Some special cells (e.g. those corresponding to the lowest memory hierarchy level exposed) have CPU's attached to them and aggregates are computed when they land on a CPU cell. The phase terminates when all aggregates have been computed. Note that a "real computer" would determine this cell stepping strategy from the software supplied by the user and the action of the hardware. The simulator will supply a framework but the user is responsible for understanding both the problem and the machine well enough to supply this cell stepping execution script. We expect to gradually automate parts of this process in future activities. In general, there are many more memory cells than aggregate objects. Usually the number of module aggregates is greater than or equal to the number of cells with CPU's. A distributed memory machine is just a special case with no memory hierarchy at each node. For that configuration, the number of cells, CPU's and module aggregates are all equal and the cell stepping strategy involves one step in which all objects are computed. In this case, PetaSIM will reproduce results like the simple "back of the envelope" results quoted earlier.

We will provide both a C and Java version of the simulator, whereas the user interface will be developed as a Java applet. The visualization of the results will use a set of Java Applets based on extensions to NPAC's current Java interface to Pablo performance monitoring system from the University of Illinois [18].

It is possible to use any level of fidelity, but the user is responsible for estimating communication costs and the compute cost when an aggregate arrives at a CPU cell. These estimates require determining the effects of lower memory levels in the hierarchy, which are not modeled directly. As was described earlier and illustrated in Figure 2, modules and computers are both specified as a set of units labeled by a hierarchy level and a position (labeled by a one or multi-dimensional index) within a given level. The user must specify the linkage between these units with an appropriate associated function (or simply a weight) that calculates the communication performance between units in the computer model and the message traffic required by the problem model. Section 2.4 gives more details on the estimation of performance cost functions for the different components needed by PetaSIM. The initial system will support "flat" problems but general hierarchies for problems and computers is essential and will be implemented in the second year of the project.

2.4 Estimation of Performance Cost Functions

2.4.1 General Approach

The cost abstraction for performance prediction will be conducted for each primitive operation. For computation components, an average cost function will be estimated using the parameters of the processor and cache/memory. For example, BLAS and LAPACK primitive performance cost functions will be studied in detail, and Rutgers/UCSB has done some preliminary studies on this topic. For standard communication and I/O primitives, cost functions will be determined based on device/networking performance parameters and the size of data communicated.

We will model the data communication structure between the modules (of Figure 2) seen in multidisciplinary and distributed applications. This coarse grain structure is typically not too sensitive to architectural details and we do not expect major difficulties in that area.

2.4.2 Analytic Estimation and Compiler Generation

The cost modeling for individual components requires either user insight or the simple measurements described earlier or symbolic inference through use of compilation of appropriate application emulators. For example, average program cost estimations are studied in [55]. Computation and communication cost estimation for a loop program is studied in [14], based on symbolic integer point counting techniques. None of this work has addressed the impact of multi-level caching and further research will be needed. For some parts of the applications, it may be difficult for a compiler or a user to provide accurate prediction. As described in the following section, runtime profiling and full simulation for some critical parts of code will lead to a relatively accurate performance abstraction for these parts to improve the accuracy of the overall performance prediction. Sometimes the worst case performance estimate from a static compiler analysis can be overly pessimistic in predicting performance in practice. On the other hand, average performance can be very difficult to obtain using symbolic analysis. Runtime profiling based on a set of carefully selected instances can be used to study the average-case performance of components where symbolic analysis is unduly pessimistic.

As shown in Figure 1, we will also use runtime compilation to generate input to PetaSIM. This will help both in defining appropriate aggregates and in estimating the associated cost

functions. For this activity we will use results from the ARPA sponsored Parallel Compiler Runtime Consortium in which both Maryland and Syracuse participate. This effort has generated a set of libraries implementing essential data movement and computation primitives used by parallel C++, HPF and also preliminary versions of parallel Java compilers. These libraries include those for both regular (HPF1) and adaptive irregular problems (HPF2 extensions). We intend to use the PCRC libraries in the application emulation path of Figure 1 to both motivate the data movement primitives in HLAM and provide cost functions by runtime profiling of their execution.

2.4.3 Use of Hardware Modeling and Simulation

For a large-scale tera/petaflop-level application, we do not intend to conduct complete simulations for entire applications on current or future parallel machines. Instead, we plan to use detailed simulation and/or runtime profiling only for the performance-critical segments of the application. We plan to use the results of detailed simulation for two purposes: (1) to develop the cost functions used by PetaSIM and (2) to verify the predictions generated by PetaSIM.

This is similar to the approach we have used in Howsim [59], a coarse-grain simulator for I/O-intensive tasks on workstation clusters developed at Maryland. We have developed Howsim for evaluation of architectural and OS policy alternatives for I/O-intensive tasks. Accordingly, Howsim simulates I/O devices (storage and network) and the corresponding OS software at a fairly detailed level and the processor at a fairly coarse level. To obtain the hardware and operating system cost functions needed for Howsim, we profiled a small set of micro-applications that exercised specific hardware and OS functionality on the IBM SP-2 and a cluster of Digital 4/2100 multiprocessor workstations. This approach has worked well for Howsim. For example, for the SP-2, Howsim was successfully able to model the application-level network bandwidth across a seven orders of magnitude difference in message size. The error for most message sizes was 2-6Howsim was able to model the application-level network bandwidth within an error of 10

We do not intend to develop or significantly extend detailed hardware simulators. Instead, we plan to use (and possibly integrate) existing hardware and complete system simulators such as SimOS [54], Proteus [11], Mint [60], Howsim and Trojan [49]. We expect to start with SimOS and integrate other simulators as needed. SimOS provides detailed models for shared-memory multiprocessors and can simulate highly realistic application workloads with acceptable slowdown. It provides a fairly good interface for adjusting the hardware configuration, such as the number of processors, clock speed, cache parameters, memory and disk system parameters. SimOS is geared towards shared-memory machines and does not emphasize network simulation. Other simulators, however, do provide good network models (e.g. Proteus provides detailed simulation of k-ary n-cube networks and Howsim simulates point-to-point networks). We plan to integrate these simulators (and possibly others) on an as-needed basis.

3 Emulating Families of High End Applications

An application emulator is a suite of programs that, when run, exhibits computational and data access patterns that resemble the patterns observed in a particular type of application. We will construct two *application emulators* motivated by loosely synchronous adaptive applications and by data exploration and data fusion applications. As described earlier, application emulators will be used to validate the HLAM/PetaSIM modeling process. We will use the application emulators to produce application and machine specifications at varying levels of granularity and then use PetaSIM to estimate performance obtained on selected current and future architectures. We believe that our application emulators address some of the key applications targeted at future very-high end architectures. The application emulators will be shared with the performance modeling community. We believe that general availability of application emulators will help the community focus attention on crucial application classes.

3.1 Irregular Adaptive Scientific Applications

We will develop an application emulator to model the performance characteristics of three classes of irregular adaptive scientific computations, along with coupled versions of multiple instances of any of these classes. The targeted computation classes are: (1) adaptive unstructured codes (e.g. unstructured multigrid solvers, integro-differential equation solvers and molecular dynamics codes), (2) structured adaptive codes, (e.g. adaptive multigrid algorithms), and (3)particle codes (e.g. Direct Simulation Monte Carlo methods, Rokhlin-Greengard or Barnes-Hut Fast Multipole codes, particle-in-cell codes).

The Maryland, Syracuse and Rutgers groups have extensive experience with each of these application classes. The three sites have long histories of developing libraries, compilers and runtime support for high performance parallel implementations for application codes from these areas [12, 15, 24, 29, 39, 46, 47]. Maryland has also taken the lead in developing systems software (Maryland's Meta-Chaos library [19]) to couple separately developed applications to carry out complex physical simulations, such as a combustion code coupled to a CFD code, to be able to model complex chemical reactions at fine scales using one application code and fluid flows at perhaps much coarser scales using another application code. The Syracuse group is playing a major role in both the physics and computer science aspects of an NSF Grand Challenge - the simulation of the collision of two black holes. This is built around adaptive mesh finite difference solutions of Einstein's equations. Our application emulators will leverage this extensive experience in a wide range of regular and irregular, static and dynamic applications.

The use of Maryland's Meta-Chaos software will greatly facilitate the construction of the application emulator. Our application emulator will make it possible to model the performance of multiple coupled parallel applications running on distributed sets of multiprocessors and/or networked workstations.

Another example of complex physical simulation is ship hull design. New advanced physics-based methods for the simulation of ship responses in a seaway and for the simulation of the viscous flow field around the hull have made it possible to explore completely new hull forms. DARPA has been supporting the development of a new Arsenal Ship design with an entirely new hull with superior resistance and seakeeping characteristics. The LAMP

codes (Large Amplitude Motions Program) are a class of multi-level physics codes for the simulation of wave body interactions of ship movements. The codes have been developed by the ship design division of SAIC. The LAMP codes are based on 3D integro-differential equation approximations and are iterative in time steps. The computations for the kernel approximations are dynamic and irregular.

The LAMP codes provide an excellent testbed application for future parallel systems and software. They have large memory requirements because the history of the ship movement needs to be stored to predict future movements and forces on the ship. Furthermore, the more physically accurate high level codes (LAMP-4 and above) require teraflop and petaflop performance to be used in an integrated design environment (LAMP-4 takes 1 CPU month on a CRAY C90-1 processor using 500 ship panels and 30 minutes of ship movement simulation. A typical 100 candidate design simulation requires about 10^6 Teraflops). Under the HPCD project Rutgers is working with SAIC on a scalable version written in MPI and on more robust LAMP codes. The LAMP codes are representative of a larger set of physical codes for multi-level ship design methods. For these classes of applications, the application emulators must model the varying quantities and patterns of computation from the application codes, including the data dependent nature of the computations. This means that the emulators must parameterize the behavior of the data access patterns of the applications, to model both the interprocessor communication within one application and the interprocessor communication generated through the coupling of multiple applications to form a complete distributed physical simulation.

3.2 I/O intensive applications

We will develop an application emulator that will reproduce application characteristics found in many defense and high-end civilian applications that involve sensor data analysis, sensor data fusion and real time sensor data processing. We are focusing on emulating application scenarios that will be of practical relevance in a 5 to 15 year time frame.

One of the motivating applications (the *Virtual Microscope*) under development at Johns Hopkins involves development of software that makes it possible to carry out a realistic digital emulation of a high power light microscope. Raw data is captured by scanning collections of full microscope slides under high power. For each microscope slide, high power images are captured at multiple focal planes. Once the image is captured, this application will make it possible to emulate the behavior of a microscope by allowing users to continuously move the stage and to simulate changing magnification and focus. The Virtual Microscope is being designed so that it will be possible to achieve an interactive level of response as the same dataset is simultaneously explored by multiple users. In the ongoing Johns Hopkins application effort, the Virtual Microscope will be coupled to computation modules that carry out: (1) three dimensional image reconstruction from data found in multiple focal planes and on multiple microscope slides, (2) image registration and compositing that takes into account data obtained using various special stains that reveal the presence or absence of biochemical markers, (3) image segmentation and pattern recognition to better characterize known malignancies, and (4) applications that aid the pathologist in screening for possible malignancy.

In the proposed effort, we will develop an applications emulator that prototypes the

performance characteristics associated with the current application suite along with the performance characteristics that will be associated with more ambitious future projects. The future projects will involve integration of additional sensor modalities along with associative retrieval of sensor data obtained from related cases. The sensor modalities that would be involved in a sensor data fusion effort include (1) different radiological imaging modalities such as CT, MRI and PET, (2) electron microscopy, (3) confocal microscopy, and (4) conventional high power light microscopy.

In the development of our application emulator, we will make the realistic assumption that sensor data is stored in multiple data repositories. Each data repository will consist of a multiprocessor architecture along with secondary and tertiary storage. The application emulator will carry out varying quantities and patterns of computation (computational patterns will be abstracted from our application codes). The processing carried out at a particular repository will result in a variable degree of data size reduction. Sensor data fusion will be emulated by prototyping processing algorithms that take as inputs processed data sets produced by several repositories.

The application emulator will emulate this ambitious sensor data fusion application suite in a parameterized fashion. Parameter adjustment will make it possible to use the emulator for various application scenarios. The behavior we emulate will include computation, secondary storage accesses, tertiary storage accesses, remote object invocations, and program migration between processing nodes.

4 Project Integration

In this section, we bring our activities together as the integrated approach outlined in Figure 1. We must make choices from the many different applications and target machines in order to provide a project focus. We will work with DARPA and the HPCC community in choosing exemplars that are synergistic with other activities. For the near future, we have as a natural baseline choice the machines chosen for the DOE ASCI program, and for a ten year horizon the target architectures from the PetaFlop process that were reported at the Frontiers'96 conference.

We expect that it will be straightforward to collaborate as a team, since we have great experience in doing this with other projects. As well as the standard (and increasingly sophisticated) electronic mechanisms, we expect that the team will need to hold meetings on a semi-annual basis. These will be especially important at the start of the project when more frequent get-togethers may be necessary – some of these will be at DARPA PI meetings.

H: Related Research

Researchers working on predicting performance of large-scale applications have taken one of two approaches: detailed simulation on specific architectures or high-level modeling. The SimOS simulation environment is the most ambitious of the detailed simulation efforts and is capable of modeling a complete computer system including a full operating system [54]. It is, however, focused exclusively on SGI multiprocessors and the Irix operating system and it is not suitable, in its current form, for simulating tera/peta-flop level applications. Furthermore, while SimOS does provide multiple levels of simulation detail, these levels are far too detailed for the scale of applications considered in this proposal. Other detailed simulation efforts have limited themselves to modeling application performance (e.g. [32, 52]). However, detailed simulation of future generation computing systems is difficult to perform. It is difficult to predict the exact sequence of technological developments across orders of magnitude improvements in performance. Even if one were able to, do so the computational resource requirements would be prohibitive.

High-level modeling is more suitable for the scale of applications and architectures envisaged in this proposal. Research on high-level modeling of large-scale applications has taken one of two approaches: algorithmic analysis to isolate performance-critical program segments or modeling of families of stylized applications.

An exemplar of the first approach is the iso-efficiency metric proposed by Grama et al [35] for evaluating the scalability of algorithm-architecture combinations. This metric relates the problem size with the number of processors based on information about processor speed, network speed and type of interconnection networks. This metric, and others like it [58, 61], is primarily suitable for the analysis of regular problems, meaning problems whose behavior is independent of the data and can be easily parameterized with respect to problem size. The metrics are not suitable for predicting the performance of irregular adaptive scientific applications. Furthermore, as they are currently formulated, they do not take I/O into consideration.

An exemplar of the second approach is the Modeling Kernel toolkit [56]. This toolkit focuses on computational fluid dynamics applications and analyzes an annotated parse tree representation of these programs. The annotations are attributes related to the program's execution time and include measured quantities like iteration counts, branch frequencies, message sizes and basic block execution times. It does not model I/O costs, nor does it deal with irregular problems. Another limitation is the lack of a model for the memory hierarchy. Performance of the memory hierarchy is an important determinant of overall performance in modern and future machines. The multi-level modeling framework we are proposing provides a hierarchical and symbolic representation of a class of application codes and can help identify performance-critical components and analyze their performance at an appropriate level of detail, including memory hierarchy performance.

A similar effort is the *Templates* compilation by Barrett et al [9]. That work focuses on iterative methods for solving large sparse systems of linear equations. Thus, besides providing templates, that work suggests how to choose and implement an effective method, and how to specialize a method to specific matrix types. The work focuses on selecting solution methods for particular problem types and quick turn-around time for developing parallel versions of the programs. The templates provided by Barrett et al are an important

resource for scientists but do not provide prediction for future computing systems (nor were they intended to).

Several research efforts have focused on measurement of large, long-running programs, a leading example being Paradyn [44]. Paradyn attempts to provide detailed, flexible performance information without incurring the space (and time) overhead typically associated with trace-based tools by dynamically instrumenting the application and automatically controlling this instrumentation in search of performance problems. In addition to gathering data, it also includes a "Performance Consultant" which provides automated isolation of performance problems. Paradyn is suitable for isolating performance problems and for obtaining information needed for constructing accurate cost functions. Prediction of performance on future computing systems is outside the scope of Paradyn.

Given the scale and the multi-level nature of the simulation needed for performance prediction on tera/peta-flop systems, the ability to control the level of detail based on computational requirements and accuracy is essential. Several research groups have investigated computational "steering", which allows either users or monitoring programs to adapt an application during execution. An exemplar of such systems is Falcon [37], an online performance measurement tool that includes a "steering" interface. The available controls in Falcon, and similar systems, are restricted to "steering" options exported by the application programmer.

As we expect PetaSIM to be an irregular adaptive program, an important component of this research is parallelization of such programs. There is a large body of research in this area, including previous research by several of the PIs on this proposal [4, 3, 2, 12, 16, 17, 29, 45]. Important techniques in this regard are runtime preprocessing of access patterns to optimize communication and other operations (the inspector-executor scheme), graph transformations and scheduling suitable for unstructured task graphs and adaptive data partitioning. We expect to build on this expertise and further develop these techniques as needed.

I: Key Personnel

Joel Saltz received his MD, Ph.D. degrees from Duke University in 1985. He is currently an Associate Professor at the University of Maryland, College Park. He leads a research group that has developed methods that are making it possible to produce portable compilers and tools to map a broad range of challenging applications on high performance architectures. Dr. Saltz is currently developing techniques that will allow parallel compute and data servers to offer their services to remote clients and will make it possible to compose programs that execute on any combination of distributed memory, shared memory or networked machines. His computer science research program has been motivated by collaborations with biomedical, physical and earth scientists. Dr. Saltz has authored over 100 refereed journal articles, conference papers and book chapters.

Alan Sussman is a Research Associate in the Department of Computer Science at the University of Maryland, College Park. Previously, he was a staff scientist at Applications in Science and Engineering, ICASE, NASA LaRC. He received his Ph.D. in Computer Science from Carnegie Mellon University in 1991. He has worked on runtime libraries and compilers for distributed memory parallel machines. His other research interests include parallel supercomputer applications, parallel I/O systems, and object-oriented programming systems.

Jeffrey K. Hollingsworth is an Assistant Professor in the Computer Science Department at the University of Maryland, College Park. He also has an appointment in the University of Maryland Institute for Advanced Computer Studies. He received his PhD and MS degrees in computer sciences from the University of Wisconsin in 1994 and 1990, respectively. He received a B. S. in Electrical Engineering from the University of California at Berkeley in 1988. Dr. Hollingsworth's work seeks to develop a unified framework to understand the performance of large systems and focuses in three areas. First, he developed new approach, called dynamic instrumentation, to permit the efficient measurement of large parallel applications. Second, he has developed a methodology to partially automate the isolation of performance problems. Third, he is investigating the interactions between different layers of software and hardware to understand how they influence performance. His current projects also include parallel I/O systems and using commodity personal computers and software in conjunction with parallel systems.

Ashok K. Agrawala is a Professor of Computer Science at the University of Maryland, College Park. For the past twenty years he has been actively involved in research in various aspects of computer systems design, implementation and performance. He received the B.Sc. degree from Agra University in 1960, the B.E. degree in Electrical Technology, and the M.E. degree in Applied Electronics and Servomechanisms from the Indian Institute of Science in1963 and 1965, respectively. He earned the A.M. and Ph.D. degrees in Applied Mathematics from Harvard University, Cambridge, MA in 1970. From 1968 to 1970 Dr. Agrawala was a Senior Engineer in the Applied Research Department, Honeywell, Inc. Waltham, MA. In 1970 he joined the Honeywell Information Systems as Principal Engineer in the Optical Character Recognition Department. Since 1971 he has been on the Faculty of the Department of Computer Science, University of Maryland, College Park. He has been actively involved in research on several aspects of computer systems. He introduced the use of clustering for characterizing the workload of computer systems. He developed

techniques for transient analysis of queues. These techniques have been used to address several unsolved problems in the design and control of multicomputer systems. Recently, he has been working on the design problems for hard real-time systems and has developed the system, MARUTI, which addresses the needs of next generation real-time systems operating in distributed environments and supporting fault tolerant operations.

Geoffrey Fox is Director of NPAC and Professor of Computer Science and Physics at Syracuse University and an internationally recognized expert in the use of parallel architectures and the development of concurrent algorithms. He has published 300 papers and 3 books. He led a major project to develop first prototype high performance Fortran (Fortran90D) compilers with a follow-on ARPA project developing language independent runtime(PCRC). He has always emphasized the role of applications in driving and validating technology. This is illustrated by his recent book "Parallel Computing Works" which describes the use of HPCC technologies in 50 significant application examples. Fox directs InfoMall, which is focused on accelerating the introduction of high speed communications and parallel computing into New York State industry and developing the corresponding software and systems industry. Much of this activity is centered on NYNET with ISDN and ATM connectivity throughout the state including schools where Fox is leading developments of new K-12 applications that exploit Web technology. With Rome Laboratory CIV project, NPAC has developed the Web based command and control application indicating how this COTS technology can be effectively used in DoD applications.

Apostolos Gerasoulis is a Professor of Computer Science at Rutgers University. Professor Gerasoulis is a leading expert in scheduling heuristics and application of scheduling in physical and mathematical problems. He has published over forty publications in highly rated journals and conferences in the field. He has participated in the organization of many conferences and recently organized the workshop on scheduling and load balancing and gave the scheduling tutorial in EuroPar 96, the largest European conference in parallel computing. He is an editor of parallel processing letters(PPL), computer and mathematics with applications and applied numerical mathematics. He is a Co-PI and area co-coordinator for area III.3 of the HPCD DARPA supported project.

Tao Yang is an Assistant Professor in the Department of Computer Science, University of California, Santa Barbara. His research interests are algorithms and programming systems for parallel and distributed processing, scheduling and compilation, parallel scientific computing and digital libraries. He has published over forty refereed conference papers and journal articles on these topics. Dr. Yang served as a guest editor for a special issue on partitioning and scheduling in Journal Parallel Processing Letters, and as a program or organizing committee member for several parallel computing conferences and workshops. He is on the editorial boards of the CD-ROM Journal of Computing and Information, and Discrete Mathematics & Theoretical Computer Science. Dr. Yang received the Research Initiation Award from NSF in 1994, and UC Regents' Junior Faculty Award in 1994. He is being recommended by NSF for the CAREER award in 1997.

J: Previous Accomplishments

Previous work in Saltz's group at Maryland has focused on developing tools, compiler run time support and compilation techniques to help scientists and engineers develop high-speed parallel implementations of codes for irregular scientific problems (i.e. problems that are unstructured, sparse, adaptive or block structured). We have developed a series of run-time support libraries (PARTI, CHAOS, CHAOS++) that carry out preprocessing and data movement needed to efficiently implement irregular and block structured scientific algorithms on distributed memory machines and networks of workstations [50, 40]. Our compilation research has played a major role in demonstrating that it is possible to develop data parallel compilers able to make effective use of a wide variety of runtime optimizations. Over the past few years, much of my emphasis in this area has been to: (1) develop techniques that make it possible for data parallel compilers to optimize complex coding constructs found in many irregular application programs, and (2) develop interprocedural optimizations that optimize placement of irregular runtime support.

Maryland's approach to high performance I/O has been to work experimentally with applications to document the optimizations needed to obtain a given level of performance, and then to design compiler transformations and runtime support libraries in response to this experimental work. We have carried out a detailed study involving I/O-intensive applications from two areas: satellite-data processing (earth science) and out-of-core sparse-matrix factorization (scientific computation) [1]. Our primary experimental platform consisted of a 16-processor IBM SP-2 with six fast disks attached to every processor. For each program the objective was simple: make it run as fast as possible and keep track of what was required to achieve this. The results of this exercise are encouraging. Foremost, we were able to obtain application-level I/O rates of over 100 MB/s for three out of four applications.

Maryland has also been exploring ways to support interoperability between sequential and parallel programs written using different languages and programming paradigms. Successful techniques would facilitate the design of complex scientific applications that are composed of separately developed components, and provide the infrastructure required to make use of highly distributed data and computational resources. We have demonstrated an ability to compose parallel programs that have been written using different programming paradigms (e.g. High Performance Fortran, HPC++ and MPI) [19, 53]. We have developed a prototype "meta-library" called Meta-Chaos that makes it possible to integrate multiple data parallel programs (written using different parallel programming paradigms) within a single application. Meta-chaos also supports the integration of multiple data parallel libraries within a single program.

Maryland has two sets of projects in the area of detailed simulation of hardware and systems software. First, we have developed Howsim [59] a coarse-grain simulator for I/O-intensive tasks on workstation clusters. Howsim was developed for evaluation of architectural and OS policy alternatives for I/O-intensive tasks. Accordingly, Howsim simulates I/O devices (storage and network) and the corresponding OS software at a fairly detailed level and the processor at a fairly coarse level. Howsim has been applied to micro-applications on both an IBM SP-2 and a cluster of Digital Alpha SMPs, with very encouraging early results.

Second, Jeff Hollingsworth's prior research is also relevant to detailed performance modeling and simulation, and has been focused in the area of performance measurement tools,

attacking two sets of problems. First, he is developing techniques for efficient performance monitoring of large parallel applications. Second, he is trying to provide assistance to users to help them manage the collected data to reduce information overload. To date, dynamic performance monitoring has demonstrated the feasibility of efficiently monitoring the performance of large, long running applications. Measurements indicate that the approach to measurement reduces the volume of data gathered by two to three orders of magnitude compare to traditional event logging. His work on Dynamic Instrumentation provides an efficient way to collect performance data for parallel computations [38]. Dynamic Instrumentation Data collection is a critical problem for any parallel program performance measurement system. To understand the performance of parallel programs, it is necessary to collect data for full-sized data sets running on large numbers of processors. However, collecting large amounts of data can excessively slow down a program's execution, and distort the collected data. Dynamic Instrumentation takes a new approach to data collection that defers instrumenting the program until it is in execution. This approach permits dynamic insertion and alteration of the instrumentation during program execution. He has also developed a new data collection model that permits efficient, yet detailed measurements of a program's performance. The search model and dynamic instrumentation have been incorporated into the Paradyn Parallel Performance Measurement Tool [44].

At Rutgers, Professor Gerasoulis' previous accomplishments are in many related areas that range from engineering to computer science and numerical mathematics. We present a sample of significant accomplishments: (1) His piecewise polynomial approximation method for singular integrals [27] has now become the standard in materials science research and crack analysis in engineering [8]]. (2) His fast algorithms for the multiplication of a singular matrix with a vector, similar to N-body computation, was the first algorithm to demonstrate the existence of faster than N^2 algorithms for the N-body problem. His work has generated significant research interest in this area and has become a standard reference in numerical computing and complexity theory [48, 34, 36]. (3) His work in scheduling and software systems has been extensive. Two of his papers with Tao Yang have been selected as significant research contribution in this area [30, 31]. His work in this area has been widely referenced. (4) He has been directing a major effort in the development of tools for mapping and scheduling applications on parallel machines. This work has been supported by ARPA under HPCD and has produced several PH.D. graduates, including Tao Yang, a collaborator on this proposal. The PYRROS, PLUSPYR and D-Pyrros systems are the outcome of this project. Clustering algorithms developed for these tools have been used by several leading institutions and other ARPA supported projects(MIT, Berkeley, NASA, UMD, CMU, RIACS, etc.). (5) Under the ARPA supported HPCD project he has collaborated with Norm Zabusky, Sandeep Bhatt and others in the application of 3D fast algorithms in vortex dynamics [21]. This collaboration has resulted in orders of magnitude performance improvements, making it possible to compute high resolutions that were not possible with the previous technology (going from 40000 particle to million particle simulations). (6) He has collaborated with the ship division of SAIC for the parallelization and improvement in robustness of the LAMP ship design codes [28]. The LAMP system was recently used in the design of the revolutionary Arsenal Ship configuration developed by the General Dynamics/Raytheon/SAIC team and is one of the applications cited in this proposal.

Tao Yang has conducted algorithm research and system development in the areas of

performance prediction, scheduling, runtime systems and sparse matrix computation, high performance WWW/digital library servers, parallel image processing tools and parallel radiosity. With Gerasoulis at Rutgers, he conducted research on granularity analysis and scheduling algorithms [31, 64], designed and implemented the PYRROS system [63], which models an application with task graphs, schedules and predicts performance and generates executable parallel code on nCUBE/Intel machines. The performance of the PYRROS scheduler is competitive with other well-designed algorithms but its complexity is one-order lower, which is critical for dealing with large-scale applications. Other research groups have used this software. For example, MIT and Maryland used it in a sparse triangular solver. PYRROS is also incorporated in an automatic task graph generation and cost abstraction system by French scientists [14] for Fortran. At UCSB, he also extended the task graph model and scheduling algorithms for dealing with iterative computations involving loop and task parallelism [62]. With his students, he has developed the RAPID runtime system [25] for parallelizing irregular scientific code. Using this system, an effective solution is provided for the semi-automatic parallelization of sparse triangular solvers, Cholesky and LU decomposition with partial pivoting. Note that sparse LU is an open problem for parallelization in the literature on distributed-memory machines with memory hierarchy. Using a new cache optimization and partitioning technique [26], the RAPID sparse LU code has achieved good speedups compared with a highly optimized sequential code recently developed at UC Berkeley. He has also developed a performance prediction and scheduling tool for modeling parallel image processing applications [43]. He is currently developing a nonlinear equation solver based on the parallel sparse LU algorithm. In collaborative work with the Alexandria project, he has developed prediction-based adaptive scheduling algorithms [6, 7] and a multiprocessor WWW server for distributed data-intensive digital library applications [5]. Those results are being used by Navy NRAD.

Geoffrey Fox has worked in this general area for 15 years at both Caltech and Syracuse. Recently his relevant related work includes the DARPA sponsored *Common Runtime Support for High Performance Parallel Languages* (PCRC) which is a collaboration including Syracuse, Maryland, Cooperating Systems, Florida, Indiana, Rochester and Texas. This is building common runtime to support Fortran C++ and Java (originally we had intended ADA as a third language) parallel compilers. This includes support for both regular applications (Syracuse) and irregular cases (Maryland, where the interoperable Meta-Chaos system was developed). This PCRC activity has also continued the development of the Syracuse HPF compiler [10, 13, 51], which was supported on a previous DARPA grant. This was probably the first HPF compiler to demonstrate the viability of the language and key technology ideas needed. This research prototype was licensed by the Portland Group, whose commercial product is highly regarded. We intend to build some of application emulator activity directly on the PCRC runtime support. Other relevant Syracuse activity is in applications which has always been Fox's focus. The book *Parallel Computing Works*, which essentially described Fox's work at Caltech, highlighted 50 separate significant parallel applications. The most relevant current Syracuse application project is an NSF funded Grand Challenge studying the collision of two black holes where they are playing a major role in both physics and computer science parts of the activity. We will use this as an adaptive mesh application emulator and expect Syracuse to be involved in other important applications during the period of this proposal.

K: Facilities

At the University of Maryland, our group has a network of Sun and Digital Alpha workstations, connected to the Internet.

Within the Department of Computer Science and UMIACS, we have:

- a 16 node IBM SP-2, with over 250 GBytes of secondary storage,
- a Digital Alpha workstation farm with 10 4-processor shared memory nodes, connected via OC-3 ATM links, and
- tertiary storage supplied by a Digital tape library, with total capacity of about 2.5 Tbytes.

The parallel machines, and many workstations, are networked via ATM OC-3 connections to other departments on the Maryland campus, and Maryland has also recently acquired an NSF vBNS ATM connection to the NSF supercomputing centers and other sites.

Rutgers has a high performance 8-node UltraSPARC cluster with Myrinet Switch, an 8-node IBM SP-2, a 512-node nCUBE2s, and a large heterogeneous workstation farm connected with Ethernet(Alpha, SGI, SUN and HP). Rutgers also has access to a large CRAY-T3E and IBM SP-2.

The University of California at Santa Barbara (subcontractors to Rutgers) has a 64-processor Meiko CS-2 distributed-memory machine supported by a major NSF CISE infrastructure grant, an SCI-based SUN UltraSPARC-1/2 workstation cluster supported by an NSF instrumentation grant along with matching funds from UC and SUN MICRO, a 4-node SGI CHALLENGE DM and a 32-node SGI ORIGIN2000 shared memory system supported by another NSF infrastructure grant. We have access to the Cray T3D/E at the San Diego Super Computing Center and an HP/Convex 32-node Exemplar shared memory machine at Navy NRAD.

Syracuse has a 12 node IBM SP-2, 8-node Sun UltraSPARC cluster with ATM connectivity and a 12 node Windows NT PC cluster with fast Ethernet connection. Syracuse also has an 8 node Silicon Graphics Power Challenge. We have excellent ATM networking with NSF vBNS as well a NYNEX experimental ATM link (NYNET). Syracuse systems are optimized for I/O and database support as part of our New York State funded InfoMall Industry outreach activity. Syracuse uses facilities at NSF centers for large scale computations such as those in the numerical black hole project.

Through the Internet, we have access to large parallel machines at several sites. Among these machines are:

- Intel Paragons at Rice, Caltech and Wright-Patterson AFB,
- IBM SP-2s at Argonne National Labs, Cornell Theory Center, NASA Ames, and the Maui High Performance Computing Center,
- the Cray T3D/Es at JPL and the San Diego Supercomputing Center.

L: Cost Breakdown

Section III : Additional Information

References

[1] Anurag Acharya, Mustafa Uysal, Robert Bennett, Assaf Mendelson, Michael Beynon, Jeffrey K. Hollingsworth, Joel Saltz, and Alan Sussman. Tuning the performance of I/O intensive parallel applications. In *Proceedings of the Fourth Annual Workshop on I/O in Parallel and Distributed Systems (IOPADS)*. ACM Press, May 1996.

[2] Gagan Agrawal, Guy Edjlali, Alan Sussman, Jim Humphries, and Joel Saltz. Runtime support for programming in adaptive parallel environments. In Boleslaw K. Szymanski and Balaram Sinharoy, editors, *Languages, Compilers and Run-Time Systems for Scalable Computers*, chapter 18, pages 241–252. Kluwer Academic Publishers, 1995.

[3] Gagan Agrawal and Joel Saltz. Interprocedural compilation of irregular applications for distributed memory machines. In *Proceedings Supercomputing '95*. IEEE Computer Society Press, December 1995. Also available as University of Maryland Technical Report CS-TR-3447.

[4] Gagan Agrawal, Alan Sussman, and Joel Saltz. Efficient runtime support for parallelizing block structured applications. In *Proceedings of the Scalable High Performance Computing Conference (SHPCC-94)*, pages 158–167. IEEE Computer Society Press, May 1994.

[5] D. Andresen, T. Yang, O. Egecioglu, O. Ibarra, and T. Smith. Scalability Issues for High Performance Digital Libraries on the World Wide Web . In *Proceedings of the 3rd Forum on Research and Tech. Advances in Digital Libraries (ADL96)*, pages 139–148, Washington D.C., May 1996.

[6] D. Andresen, T. Yang, V. Holmedahl, and O. Ibarra. SWEB: Towards a Scalable World Wide Web Server on Multicomputers . In *Proceedings of 10th IEEE International Parallel Processing Symposium*, pages 850–856, Hawaii, April 1996.

[7] D. Andresen, T. Yang, D. Watson, and A. Poulakidas. Dynamic Processor Scheduling with Client Resources for Fast Multi-resolution WWW Image Browsing . In *Proceedings of the Eleventh International Parallel Processing Symposium*, Geneva, April 1997.

[8] A. Azhdari and N. Nasser. Energy-release Rate and Crack Kinking Anisotropic Brittle Solids. *J. Mechanics and Physics of Solids*, 44(6):929–951, 1996.

[9] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. Philadelphia, PA, 1994.

[10] Z. Bozkus, A. Choudhary, G. Fox, T. Haupt, S. Ranka, and M.-Y. Wu. Compiling Fortran 90D/HPF for distributed memory MIMD computers. *Journal of Parallel and Distributed Computing*, 21(1):15–26, April 1994.

[11] E. Brewer, A. Colbrook, C. Dellarocas, and W. Weihl. Proteus: A high-performance parallel architecture simulator. *Performance Evaluation Review*, 20(1):247–8, June 1992.

[12] A. Choudhary, G. Fox, S. Hiranandani, K. Kennedy, C. Koelbel, S. Ranka, and J. Saltz. Software support for irregular and loosely synchronous problems. *Computing Systems in Engineering*, 3(1–4):43–52, 1992. Papers presented at the Symposium on High-Performance Computing for Flight Vehicles, December 1992.

[13] A. Choudhary, G. Fox, S. Hiranandani, K. Kennedy, C. Koelbel, S. Ranka, and C.-W. Tseng. Unified compilation of Fortran 77D and 90D. *ACM Letters on Programming Languages and Systems*, 2(1–4):95–114, 1993.

[14] M. Cosnard and M. Loi. Automatic Task Graph Generation Techniques. *Parallel Processing Letters*, 5(4):527–538, December 1995.

[15] R. Das, D. J. Mavriplis, J. Saltz, S. Gupta, and R. Ponnusamy. The design and implementation of a parallel unstructured Euler solver using software primitives. *AIAA Journal*, 32(3):489–496, March 1994.

[16] R. Das, R. Ponnusamy, J. Saltz, and D. Mavriplis. Distributed memory compiler methods for irregular problems - data copy reuse and runtime partitioning. In J. Saltz and P. Mehrotra, editors, *Languages, Compilers and Runtime Environments for Distributed Memory Machines*, pages 185–220. Elsevier, 1992.

[17] Raja Das, Mustafa Uysal, Joel Saltz, and Yuan-Shin Hwang. Communication optimizations for irregular scientific computations on distributed memory architectures. *Journal of Parallel and Distributed Computing*, 22(3):462–479, September 1994.

[18] Kivanc Dincer and Geoffrey 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, 1997.

[19] Guy Edjlali, Alan Sussman, and Joel Saltz. Interoperability of data parallel runtime libraries. In *Proceedings of the Eleventh International Parallel Processing Symposium*. IEEE Computer Society Press, April 1997. To appear.

[20] J. Feo, D. Cann, and R. Oldehoeft. A report on the Sisal language project. *J. of Parallel and Distributed Computing*, 10:349–365, 1990.

[21] V. M. Fernandez, N. J. Zabusky, P.-F. Liu, S. Bhatt, and A. Gerasoulis. Filament Surgery and Temporal Grid Adaptivity Extensions to a Parallel Tree Code for Simulation and Diagnosis in 3D Vortex Dynamics. In *Proceedings of 2nd International Workshop in Vortex Flows*, Montreal, Canada, August 1995.

[22] J. Ferrante, K. Ottenstein, and J. Warren. The Program Dependence Graph and its Use in Optimization. *ACM Trans. on Programming Languages and Systems*, 9(3):319–349, 1987.

[23] Geoffrey 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, available at *http://www.npac.syr.edu/users/gcf/petastuff/petaweb/*, 1997.

[24] Geoffrey C. Fox, Roy D. Williams, and Paul C. Messina. *Parallel Computing Works*. Morgan Kaufman, 1994.

[25] C. Fu and T. Yang. Run-time Compilation for Parallel Sparse Matrix Computations. In *Proceedings of the 1996 International Conference on Supercomputing*, pages 237–244, Philadelphia, May 1996.

[26] C. Fu and T. Yang. Sparse LU Factorization with Partial Pivoting on Distributed Memory Machines. In *Proceedings of Supercomputing'96*, Pittsburgh, November 1996.

[27] A. Gerasoulis. The use of piecewise quadratic polynomials for the solution of singular integral equations of Cauchy type, . *Computers and Mathematics with Applications*, 8:15–22, 1982.

[28] A. Gerasoulis, J. Jiao, W. Lin, M. Fritz, V. Fernandez, and N. Zabusky. Parallelization and Visiulization of the Ship Design Codes. In *Proceedings of International Mechanical Engineering Congress*, San Francisco, November 1995.

[29] A. Gerasoulis, J. Jiao, and T. Yang. Experience with graph scheduling for mapping irregular scientific computation. In *1995 International Parallel Processing Symposium, Proceedings of the First Workshop on Solving Irregular Problems on Distributed Memory Machines*, Santa Barbara, CA, April 1995.

[30] A. Gerasoulis and T. Yang. A comparison of clustering heuristics for scheduling DAGs on multiprocessors . *Journal of Parallel and Distributed Computing*, 16(2):276–291, 1992.

[31] A. Gerasoulis and T. Yang. On the Granularity and Clustering of Directed Acyclic Task Graphs . *IEEE Transactions on Parallel and Distributed Systems*, 4(6):686–701, June 1993.

[32] K. Ghosh, K. Panesar, R. Fujimoto, and K. Schwan. PORTS: A Parallel, Optimistic, Real-Time Simulator. In *Proceedings of 8th Workshop on Parallel and Distributed Simulation (PADS '94)*, pages 24–31, 1994.

[33] M. Girkar and C. D. Polychronopoulos. Automatic Extraction of Functional Parallelism from Ordinary Programs. *IEEE Transactions on Parallel and Distributed Systems*, 3(2):166–178, March 1992.

[34] I. Gohberg and V. Olshevsky. Fast Algorithms with preprocessing for Matrix-Vector Multiplication Problems . *Journal of Complexity*, 10:411–427, 1994.

[35] A. Grama, A. Gupta, and V. Kumar. Isoefficiency: Measuring the Scalability of Parallel Algorithms and Architectures. *IEEE Parallel and Distributed Technology*, 1(3):12–21, August 1993.

[36] L. Greengard. *The rapid evaluation of potential fields in particle systems* . PhD thesis, Yale University, 1987.

[37] W. Gu, G. Eisenhauer, E. Kraemer, K. Schwan, J. Stasko, J. Vetter, and N. Mallavarupu. Falcon: Online Monitoring and Steering of Large-scale Parallel Programs. In *Proceedings of Frontiers'95*, pages 422–9, Feb 1995.

[38] Jeffrey K. Hollingsworth, Barton P. Miller, and Jon Cargille. Dynamic program instrumentation for scalable performance tools. In *1994 Scalable High-Performance Computing Conf.*, May 1994.

[39] Yuan-Shin Hwang, Raja Das, Joel H. Saltz, Milan Hodoscek, and Bernard R. Brooks. Parallelizing molecular dynamics programs for distributed memory machines. *IEEE Computational Science & Engineering*, 2(2):18–29, Summer 1995.

[40] Yuan-Shin Hwang, Bongki Moon, Shamik D. Sharma, Ravi Ponnusamy, Raja Das, and Joel H. Saltz. Runtime and language support for compiling adaptive irregular programs. *Software–Practice and Experience*, 25(6):597–621, June 1995.

[41] J. Jiao. *Software Support for Parallel Processing of Irregular and Dynamic Computations* . PhD thesis, Dept. of Computer Science, Rutgers University, New Brunswick, October 1996.

[42] C. Koelbel, D. Loveman, R. Schreiber, G. Steele, Jr., and M. Zosel. *The High Performance Fortran Handbook*. MIT Press, 1994.

[43] C. Lee, T. Yang, and Y. Wang. Partitioning and scheduling for image processing operations. In *Proceedings of 7th IEEE Symposium on Parallel and Distributed Processing*, pages 86–90, Texas, October 1995.

[44] Barton P. Miller, Mark D. Callaghan, Jonathan M. Cargille, Jeffrey K. Hollingsworth, R. Bruce Irvin, Karen L. Karavanic, Krishna Kunchithapadam, and Tia Newhall. The Paradyn parallel performance measurement tools. *IEEE Computer*, 28(11), November 1995.

[45] B. Moon and J. Saltz. Adaptive runtime support for direct simulation Monte Carlo methods on distributed memory architectures. In *Proceedings of the Scalable High Performance Computing Conference (SHPCC-94)*, pages 176–183. IEEE Computer Society Press, May 1994.

[46] Bongki Moon, Gopal Patnaik, Robert Bennett, David Fyfe, Alan Sussman, Craig Douglas, Joel Saltz, and K. Kailasanath. Runtime support and dynamic load balancing strategies for structured adaptive applications. In *Proceedings of the Seventh SIAM Conference on Parallel Processing for Scientific Computing*, pages 575–580. SIAM, February 1995.

[47] R. P. Nance, R. G. Wilmoth, B. Moon, H. A. Hassan, and J. Saltz. Parallel Monte Carlo simulation of three-dimensional flow over a flat plate. *AIAA Journal of Thermophysics and Heat Transfer*, 9(3):471–477, July-Sept. 1995.

[48] V. Pan. Complexity of Computations with Matrices and Polynomials . *SIAM Review*, 34(2):225–262, 1992.

[49] D. Park and R. Saavedra. Trojan: A high-performance simulator for shared memory architectures. In *Proceedings of the 29th Annual Simulation Symposium*, pages 44–53, April 1996.

[50] R. Ponnusamy, J. Saltz, A. Choudhary, Y.-S. Hwang, and G. Fox. Runtime support and compilation methods for user-specified irregular data distributions. *IEEE Transactions on Parallel and Distributed Systems*, 6(8):815–831, August 1995.

[51] Ravi Ponnusamy, Yuan-Shin Hwang, Raja Das, Joel H. Saltz, Alok Choudhary, and Geoffrey Fox. Supporting irregular distributions using data-parallel languages. *IEEE Parallel & Distributed Technology*, 3(1):12–24, Spring 1995.

[52] S. Prakash and R. Bagrodia. Parallel Simulation of Data Parallel Programs . In *Proceedings of the 8th Workshop on Languages and Compilers for Parallel Computing*, Ohio, August 1995.

[53] M. Ranganathan, A. Acharya, G. Edjlali, A. Sussman, and J. Saltz. Runtime coupling of data-parallel programs. In *Proceedings of the 1996 International Conference on Supercomputing*, pages 229–236. ACM Press, May 1996.

[54] M. Rosenblum, S. Herrod, E. Witchel, and A. Gupta. Complete Computer Simulation: The SimOS Approach. *IEEE Parallel and Distributed Technology*, Fall 1995.

[55] V. Sarkar. Determining Average Program Execution Times and their Variance . In *Proceedings of ACM SIGPLAN'89*, pages 298–312, 1989.

[56] S. Sarukkai, P. Mehra, and R. Block. Automated scalability analysis of message-passing programs. *IEEE Parallel and Distributed Technology*, 3(4):21–32, Winter 1995.

[57] Marc Snir, Steve W. Otto, Steven Huss-Lederman, David W. Walker, and Jack Dongarra. *MPI: The Complete Reference*. Scientific and Engineering Computation Series. MIT Press, 1996.

[58] X-H. Sun and L. Ni. Another View of Parallel Speedup. In *Proceedings of Supercomputing'90*, pages 324–30, 1990.

[59] M. Uysal, A. Acharya, R. Bennett, and J. Saltz. A customizable simulator for workstation networks. In *Proceedings of the Eleventh International Parallel Processing Symposium*. IEEE Computer Society Press, April 1997. To appear.

[60] J. Veenstra and R. Fowler. MINT: A front-end for efficient simulation of shared-memory multiprocessors. In *Proceedings of International Workshop on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 201–7, February 1994.

[61] P. Worley. The Effect of Time Constraints on Scaled Speedup. *SIAM Journal on Scientific and Statistical Computing*, 11(5):838–58, 1990.

[62] T. Yang, C. Fu, A. Gerasoulis, and V. Sarkar. Mapping Iterative Task Graphs on Distributed-memory Machines . In *International Conf. on Parallel Processing*, pages 151–158, August 1995.

[63] T. Yang and A. Gerasoulis. PYRROS: Static Task Scheduling and Code Generation for Message-Passing Multiprocessors . In *Proc. of 6th ACM International Conference on Supercomputing*, pages 428–437, 1992.

[64] T. Yang and A. Gerasoulis. DSC: Scheduling Parallel Tasks on An Unbounded Number of Processors . *IEEE Transactions on Parallel and Distributed Systems*, 5(9):951–967, 1994.