CESDIS

TECHNICAL REPORT SERIES

NUMBER: TR-95-161

The Proceedings of The Petaflops Frontier Workshop

February 6, 1995

Associated with the
IEEE Sponsored Frontiers '95 Conference
on Massively Parallel Computations

Thomas Sterling
and
Michael J. MacDonald

Center of Excellence in Space Data and Information Sciences

Center of Excellence in Space Data and Information Sciences
Goddard Space Flight Center
Code 930.5
Greenbelt, MD 20771

(301) 286-4403

Internet: cas@cesdis1.gsfc.nasa.gov

Abstract

This report presents the proceedings of the The Petaflops Frontier (TPF) Workshop conducted at the 1995 Frontiers of Massively Parallel Processing in McLean, VA on February 6, 1995. A year after the first Pasadena Workshop on Enabling Technologies for Peta(FL)ops Computing, this workshop was held to extend the findings of the first workshop through wide coverage of related disciplines and involvement of a broader community. Over a hundred participants attended the one-day workshop at which 18 presentations were given on topics in technology, architecture, alogrithms and applications related to petaflops-scale computing.

The architecture and technology presentations included discussions of heterogeneous mixed-machine and mixed-mode computing systems, processor-in-memory (PIM) technology developments along with other approaches to combining logic functions with memory, and several that dealt with the potential of various optical technologies to ease the bandwidth bottleneck.

The applications and algorithms presentations focused primarily on the existing need for various applications for petaflops-level computing performance. These applications include the human genome project, modeling of physiological functions, drug design, ecological studies, and computational fluid dynamics (CFD), a discipline used in numerous applications.

The workshop showed progress in the thinking about petaflops architecture and technology requirements, reinforced the need for algorithmic research to enable effective management of petaflops-level computing systems, and, finally, reinforced the findings of the first petaflops workshop in Pasadena in 1994.

Executive Summary

Even as the federal HPPCIT program pushes the computing frontier to achieve teraflops computing capabilities, researchers across a wide variety of disciplines have been engaged in defining the generation beyond teraflops—petaflops. A petaflops equals a million billion (10^{15}) floating point operations per second—more than the entire computing power now existing in the U.S.

The first on Enabling Technologies for Peta(FL)OPS Computing workshop in Pasadena, California in 1994 called for continuing efforts to refine the workshop findings and to more accurately define the requirements for architecture, technology, applications and algorithms. Also, that workshop participants' concluded that developing a petaflops machine should be possible by the year 2013 and that a new architecture paradigm should not be needed to achieve petaflops-level performance. The Pasadena workshop was successful in bringing together an invited group of experts from many disciplines to address fundamental issues related to the feasibility of petaflop computing.

The Petaflops Frontier (TPF) Workshop was a continuation of the work started in Pasadena. The workshop was held in McLean,Virginia on February 6, 1995. This workshop was designed to bring together a wider community of researchers in architecture, technology, applications and algorithms who, using an open forum format, could present their work to a multidisciplinary audience for review and comment. Many of the presenters had not participated in previous petaflops workshops, but were actively working on parts of the problem. Their contributions were considered important to preserve what has been one of the fundamental motivations of the Frontiers series: to provide a forum for researchers who may not be part of large, organized research and development efforts. At the same time, the workshop was fortunate to have researchers who are participating in the efforts to develop teraflops computing capabilities and who contributed to the earlier petaflops workshops.

Of the 18 presentations given, about half were on the architecture and technology and half were on applications and algorithms.

The Petaflops Frontier Workshop clearly reaffirmed the results of previous gatherings that addressed petaflops computing. The discussions on architecture and technology demonstrated that conceptually the problems (or challenges) are known and that a dramatic paradigm shift will not be likely or needed. Possibilities in architecture range from heterogeneous systems as proposed by Siegel et al., to distributed instruction-set architectures. Several presenters, such as Kogge, Elliott and others made the case for exploiting processor-in-memory (PIM) technologies in architecture design. Others (Lukowicz, Qiao, and Dowd) addressed the potential for optical technologies to help break the I/O bandwidth bottleneck.

The presentations on applications and algorithms varied from Moore's discussion on data-intensive applications based on empirical data from the San Diego Supercomputer center, to a futuristic look at strategic applications by Stevens et al. The problems of virtual reality were addressed by Taylor et al. Presentations about current applications that need petaflops computing include medical and biological problems (Maizel); computational fluid dynamics (Deane); astrophysics (Fryxell and Olson); ecosystem simulation (Zeigler et al.) and "real" engineering and scientific problems (Robinson).

As a result of the workshop CESDIS along with NASA's Goddard Space Flight Center has designed and implemented at Goddard and CESDIS an on-line reference index accessible by means of the world wide web for petaflops enabling technologies and applications (PETA). The index will be a central point of information on petaflops research and development. (PETA can be accessed at the URL: http://cesdis.gsfc.nasa.gov/petaflops/peta.html) In addition to this report and the establishment of PETA, other follow-on activities to the workshop will include a more detailed examination of petaflops requirements for science, engineering, and information management applications, and added effort in reviewing, defining, and focusing the requirements for possible architectures for petaflops computing.

Contents

List of Figures

List of Tables

1 Introduction

Even as the Federal HPCC Program works towards achieving teraflops computing, policy makers and future research program planners in government, academia and industry concluded that teraflops-level computing systems will be inadequate to address many scientific and engineering problems that exist now, let alone applications that will, arise in the future. As a result, the high performance computing community is examining the feasibility of achieving petaflops-level computing over a 20-year period.

1.1 What is Petaflops?

A petaflops is a measure of computer performance equal to a million billion floating point operations per second. It is comparable to more than ten times all the networked computing capability in America, and it is ten thousand times faster than the world's most powerful massively parallel computer. In short, it is more than all the world's computing power today. A petaflops computer is so far beyond anything within contemporary experience that scientists and engineers confronting problems that are essentially intractable today may in their lifetimes see significant progress in problem solutions. Perhaps even more exciting about the concept of petaflops computing is that applications that now are little more than dreams may become realities.

Petaflops computing also offers the attraction of being the next far horizon on the landscape of high performance computing. It offers the incentive of working at the very boundaries of computing—of seeing the future and being part of its genesis. In the push towards future computing capabilities, however, it is important to understand how the vision of petaflops computing developed.

1.2 Historical Perspective

As early as December 1991 the challenge of petaflops computing was receiving serious consideration at the Purdue Workshop on Grand Challenges in Computer Architecture for the Support of High Performance Computing sponsored by the National Science Foundation. The workshop co-chairs identified achieving petaops performance as one of four grand challenge problems in computer architecture.[1] The authors noted that, "This . . . challenge [of achieving peta-ops computing] is

[1] Siegel, H. J. et al, "Report of the Purdue Workshop on Grand Challenges in Computer Architecture for the Support of High Performance Computing", J. Parallel and Distributed Computing, Vol. 16, No. 3, 1992, pp. 199–211.

to dramatically improve and effectively harness the base technologies into a future computer system that will provide usable peta-ops of computer performance to grand challenge application programmers."

Following the Purdue workshop, the issue of petaflops computing was addressed by the High Performance Computing, Communications and Information Technology Subcommittee (HPCCIT)[2]. The HPCCIT, comprised of representatives from the major government agencies involved in the HPCC program, proposed that enabling technologies for petaflops computing be addressed in a workshop in the near future.

Soon after the meeting, the Administrator of NASA convened a special initiative team to evaluate its existing and future high performance computing requirements. The NASA Supercomputing Special Initiative Team used a projected 10-year period to assess the implications of the computational aerosciences and Earth and space sciences grand challenges with respect to (1) established NASA requirements, (2) other U.S. government HPC activities, including advanced architectures, component technologies, and communications, (3) U.S. industry efforts, (4) activities in academia and other orgnaizations, and (4) the approach and progress of foreign efforts.

The team re-affirmed the findings of the earlier Pasadena workshop with respect to the requirements to achieve teraflops computing. The team also concluded that some NASA grand challenge problems would require petaflops computing performance. In their assessment the team identified seven major technology barriers to achieving petaflops-level performance:

1. Systems software
2. Memory speed
3. Aggregate I/O
4. Interprocessor speed
5. Processor speed
6. Packaging
7. Power management

[2]The HPCCIT is a subcommittee of the Committee on Physical, Mathematical, and Engineering Sciences (PMES), a committee of the Federal Coordinating Council on Science, Engineering and Technology (FCCSET)

Other government agencies, academia and industry were no less aware of the need to extend their horizons beyond the teraflops regime. The combination of this awareness, the HPCCIT meeting, and the report of NASA's Supercomputing Special Initiative Team, helped form the basis of the first workshop on petaflops computing.

In February 1994 in Pasadena, California, Caltech hosted the first major workshop to address petaflops computing. The Workshop on Enabling Technologies for Peta(FL)OPS Computing involved over 60 invited experts in all aspects of high performance computing technology who met to establish the basis for considering future research initiatives that could lead to the development, production, and application of petaflops scaled computing systems. The objectives of the Pasadena workshop were to (1) identify applications that require petaflops performance and determine their resource demands, (2) determine the scope of the technical challenge to achieving effective petaflops computing, (3) identify critical enabling technologies that lead to petaflops computing capability, (4) establish key research issues, and (5) recommend elements of a near-term research agenda.

Over a period of three days the Pasadena workshop focused on the following major and inter-related topic areas:

- Applications and Algorithms
- Device Technology
- Architecture and Systems
- Software Technology.

The workshop results reflected the potential opportunities and the daunting challenges that will confront designers and users of future petaflops computing systems.

Despite the expected challenges, the participants concluded that a petaflops computing system should be feasible in 20 years. This prediction was partly based on an assumption that during the 20 years the semiconductor industry would continue advancing in speed enhancement and in cost reduction through improved fabrication processes. And, although the workshop concluded that no paradigm shift would be needed in systems architecture, managing active latency would be essential and require a very high degree of fine-grain parallelism along with the mechanisms to exploit it. Also, a mix of technologies might be required, including semiconductor for main memory, optics for inter-processor (and possibly inter-chip) communications and secondary storage, and perhaps cryogenic (e.g., Josephson Junction) for very high clock rate and very low power processor logic. Finally, dramatic per device cost reduction and innovative approaches to system software and programming methodologies would be essential.

1.3 The Petaflops Frontier Workshop

The The Petaflops Frontier Workshop was held in McLean, Virginia on February 6, 1995. This first Frontiers petaflops workshop was part of a deliberate and ongoing process to define the long range future of high performance computing here in the U.S. and, practically speaking, the rest of the world. The one-day workshop included 18 presentations in architecture, technology, applications, and algorithms from particpants ranging from recognized leaders in their fields to individual researchers from government, academia and industry.

If the notion of petaflops is ambitious, the technical challenges promise to be staggering. The nature and scope of the challenges were identified by the Pasadena petaflops workshop. But, while the experts participating in that workshop were acknowledged experts, missing were individuals who could be thought of as working "on the edge": researchers working on bits and pieces of the technologies that eventually will be needed to advance beyond teraflops to petaflops.

Eliciting the participation of such individuals in the high performance computing community was and remains an overarching goal of the annual Frontiers sessions. In this respect the The Petaflops Frontier Workshop was at least partially successful because, while many participants and attendees had not been at Pasadena, an appreciable number had. One obvious reason for this is that interest in working towards a petaflops capability remains high.

1.3.1 Objectives

The Petaflops Frontier Workshop had five specific objectives

1. Help satisfy the recommendation of the working groups at the first Pasadena petaflops workshop for further near term studies and more detailed examination of system requirements and technology extrapolation.

2. Open the emerging discipline of petaflops computing to a much broader community of science and technology. (As indicated earlier, this also is one of the overarching goals of Frontiers.)

3. Focus on identifying (and extending) science, engineering and information management challenges that would be enabled by petaflops capabilities, and determine resource requirements.

4. Widen the scope of technologies and architectures being considered for petaflops computing.

5. Assess the feasibility and desirability of establishing PETA (Petaflops Enabling Technologies and Applications), an on-line reference index as a central point for cataloging information about petaflops-related research and development accessible via the world wide web.

1.3.2 Workshop Approach

The workshop was organized to ensure continuity with the Pasadena petaflops workshop and to ensure that anyone who could contribute in any way had the opportunity to do so. Consequently, an open invitation was extended to all interested colleagues. This approach distinguished the Frontiers workshop from the "by invitation only" participation at Pasadena.

A large number of brief presentations was scheduled with two preliminary categories considered:

- Architecture and Technology
- Applications and Algorithms

An open speaker forum format was used. This eased some of the constraints on participation encountered in more structured events. The focus of the discussions was the challenges, barriers, and opportunities to petaflops computing.

This report constitutes another aspect of the approach to the workshop. As a result, the goal of the report is to clearly and succinctly capture the workshop deliberations and present it in a context that puts it into perspective with the on-going HPCCIT program.

Finally, even before this report was published the petaflops database and electronic reference index, PETA, became operational. This index will considerably enhance communication among those working on the various technologies suitable for future petaflops development work.

1.4 Report Organization

This report is being published as a Technical Report of the Center of Excellence in Space Data and Information Systems, Universities Space Research Association, in cooperation with NASA's Goddard Space Flight Center.

Section 1 briefly describes several key events or activities that preceded the The Petaflops Frontier Workshop, the objectives and approach of the workshop and the report organization.

Section 2 summarizes the key issues of petaflops computing. Much of the discussion is based on the excellent work and report from the Workshop on Enabling Technologies for Peta(FL)OPS Computing in Pasadena in 1994. The discussion provides a synopsis of the important findings and conclusions from that workshop.

Section 3 includes the The Petaflops Frontier Workshop agenda and information about the organizers, the presenters and the participants.

Section 4 is a synthesis of the presentations at The Petaflops Frontier Workshop in McLean, VA February 6, 1995. Eighteen presentations addressed various aspects of architecture, technology, applications, and algorithms.

Section 5 consists of extended abstracts from the workshop presentations in the areas of architecture and technology, and Section 6 includes the extended abstracts from the applications and algorithms presentations. These are included both to ensure the technical content and to provide the reader with material directly by the participants.

Section 7 distills the workshop results and presentations in a comprehensive discussion of conclusions and recommendations for follow-on activities.

2 Issues for Petaflops Computers

2.1 Introduction

The first Pasadena petaflops workshop represented major potential constituencies for a long-term initiative to develop the enabling technologies that can lead to petaflops computing systems. Equally important, these representatives were, for the most part, active users, innovators, and researchers in high performance computing and related technologies. Their specific findings and the issues they identified, therefore, illuminate the important challenges facing the high performance computing community

2.2 Workshop Findings on Enabling Technologies for Peta(FL)OPS Computing

Perhaps the two most important overriding conclusions of the first workshop on enabling technologies for petaflops computing were that, (1) although teraflops computing has not yet been achieved, it was not too soon to assess the state-of-the-art with respect to petaflops computing, and (2) the implications of such a capability were too important to the nation's future to ignore.

The working groups of the workshop individually and collectively developed twelve separate findings that can be categorized as follows:

Feasibility

Potential Use

Cost

Reliability

MIMD Model

Latency Management and Parallelism

Semiconductor Technology

Memory

Software Paradigm Shift

Merging of Technologies

A Role for Superconductivity

Optical Logic Unlikely

The findings are summarized in the paragraphs that follow.

Feasibility. Assuming that key technologies maintain recent rates of advancement over the next 20 years, a petaflops computer is feasible. This also assumes that appropriate software technology develops as well. The 20-year development period will require appreciable work in several device technology areas to achieve working systems.

Potential Use. Applications that will need petaflops computing include science, engineering, economics, societal information infrastructure, and management to name a few. The availability of petaflops-level computing will also engender applications that cannot be predicted now, but will determine the technologies of the future. This cycle of technology push, requirements pull has historical precedence in the computing community and there is no reason to expect that the next 20 years will be different.

Cost. Cost will be the single most important factor affecting the development of petaflops computing. In today's dollars and technology such a system would cost over $100 billion. The best estimates for the cost of a petaflops system in 20 years range from $100 million to $1 billion.

Reliability. Cost will place an upper limit on component count of between a hundred thousand and a million for a petaflops computer and this will be achievable only with significant advances in device integration density. But, this estimate of components is within the range of the largest systems being manufactured today. Consequently, using known techniques, a petaflops computing system should have acceptable reliability; although some question exists concerning the reliability of extremely high density on-chip devices.

MIMD Model. Achieving petaflops computing should entail no fundamental paradigm shift in system architecture. A NUMA (Non-Uniform Memory Access) MIMD structure with possible SIMD elements can satisfy the key resources and control requirements. Many of the details, however, will be quite different from conventional multiprocessors. This finding is considered valid for today's classes of applications, but it is not clear what additional requirements may be imposed by new applications once petaflops computing becomes available.

Latency Management and Parallelism. Compared to conventional MPPs, a petaflops system will have a wide diameter[1]. Unless the resulting latency can be hidden, system efficiency will be unacceptable. Single-cycle context switching and communication pipelining are methods of managing the latency, but this will require exploiting vast amounts of parallelism. Two possible alternatives are very low latency systems with a few (1000) very high performance processors, and systems with millions of processors where almost all accesses are to local memory. For either, latency-hiding mechanisms would not be as critical.

Semiconductor Technology. The commercial semiconductor marketplace invests the most in technology development, far exceeding any potential augmentation from government sources. Specialty chips are more costly than mass produced commodity parts of equivalent complexity. The development of a petaflops computer will have to rely heavily (although not exclusively) on commercially available components. But the time to develop a petaflops computer will be determined by the "technology wave" which is largely controlled by market forces.

Memory. Memory will drive cost as well as many of the architectural decisions. The important findings relating to memory covered several topic areas that are summarized below:

- **Size**. The predominant component of a petaflops computer will be memory. Although it has been thought that memory should scale linearly with performance, a scaling factor of the $3/4$ root provides an upper bound for entire classes of scientific and engineering applications, and for many applications the storage requirement were considerably less. Consequently, the memory size will be between 10 and 40 terabytes. In some instances, however, more memory will be desirable.

- **Bandwidth**. Compared to today's standards, memory bandwidth will have to be significantly increased with respect to memory size. While on-chip memory bandwidth is very high, its interface to external devices inhibits its use. New memory organizations that expose this bandwidth will be essential in the design of a petaflops system.

- **Global Name Space**. Having a common name space across the machine can simplify both programming and implementing resource management software, and (through hardware support) minimize overhead for fine grain remote access requests. Support of a global name space is important for both human and machine effectiveness.

[1]Diameter is a measure of the number of clock cycles required for an access request to propagate across the machine

- **Petabyte Computer**. Many information management applications of the future may be more dependent on availability of main memory than on peak performance. While a different class of computer than the petaflops computer that was the focus of the workshop, such memory intensive systems will play a significant role.

Software Paradigm Shift. Current programming methodologies and resource management software are inadequate to exploit massive parallelism. An entirely new programming paradigm may be essential for effective use of these systems, and it must extend beyond the central computer to incorporate semantics that deal with the information infrastructure. Complicated compile time analysis algorithms, which have intense compute requirements, will benefit from a petaflops capability.

Merging of Technologies. Instead of the fairly exclusive and homogeneous use of semiconductor technology today, the merging of several technologies could give a stronger petaflops system. Likewise, superconductive devices could provide extremely high speed logic with very low power. Very low power optics will provide internal bandwidth, while semiconductor technology will be used for the high density mass main memory.

A Role for Superconductivity. Superconductor technology may be extremely useful in a petaflops computer. While the potential two orders of magnitude clock speed improvement over semiconductor speeds will be important, the low power requirements may be crucial. In the U.S., however, cryogenic digital technology is receiving relatively low support.

Optical Logic Unlikely. Optical technology is unlikely to provide a path to high density, high speed, low latency logic that can be used in petaflops computers. Such devices may play a part in very specialized hardwired computing systems, but their use as constituent elements in the class of systems considered is not tenable.

2.3 Important Issues and Implications

The first Pasadena Workshop on Enabling Technologies for Peta(FL)OPS Computing identified 14 important issues and implications that arose from the discussions within and among the various workgroups. These issues and implications can be categorized in three areas as follows:

- **Policy and National Need Issues**
  - Why Petaflops Now?
  - Role of a Petaflops Computer
  - Side-effect Products

- **Technology Issues**
  - SIA Predictions
  - Impact of Exotic Technologies
  - U.S. Capabilities in Memory Fabrication
  - Performance Versus Efficiency
  - Special Widgets, Where to Invest
  - Programming Paradigms
- **Architecture Issues**
  - A Range of Architectures
  - Far-side Architectures
  - Latency-hiding Techniques May Help Smaller Machines
  - Long Versus Short Latency Machines
  - I/O Scaling

Each of the categories is summarized in the paragraphs that follow.

Policy and National Need Issues. The report on the first petaflops workshop stated that the fundamental issue of the workshop was, "... the stakes are too high for the U.S. to leave this future technology in the hands of other nations."[2] For that reason the participants believed the subject of petaflops computing was both a serious and timely topic, especially since the U.S. appears to be lagging in certain technologies that could be critical to a petaflops computer.

A petaflops computer is unlikely to be restricted to the classical role of supercomputers, i.e., few users working on scientific or engineering applications. More likely, a petaflops computer will be shared by many and varied users working on diverse problems. Also it's possible that a petaflops computer may actually be a confederation of computers instead of a single computer. Such an arrangement would simultaneously support numerous small computing problems but make available the collective compute power when required for large problems. As more and more computing power becomes available, it's likely that non-scientific applications will avail themselves of the capability. These non-scientific applications could include massive information databases and multimedia presentations for simultaneous access by thousands of users, management services, etc.

[2]Thomas Sterling, Messina, P. C., Smith, P. H., *Enabling Technologies for Peta(FL)OPS Computing*, MIT Press, 1995, p. 159

Lastly, if history is any indicator, as the quest for more computing power heats up along the road to petaflops capabilities, the technology developed will be applied to "interim" products that will have dramatically increased computing power compared to what is available today. It's not inconceivable that before 20 years have passed a desktop workstation will be more powerful than today's most powerful computer.

Technology Issues. The technology issues and implications encompass both hardware and software. In the hardware sector there seems to be no question that semiconductor technology will reach the limits of physics long before petaflops capabilities can be achieved. At the same time the decades-long dominance of semiconductor technology has had a serious negative effect on the funding of other technologies: technologies that now seem to offer promise in solving some of the obstacles to developing a petaflops computer. Two of these technologies are superconductor and optical technologies.

Software technology also lags the needs of the high performance computing community. Indeed, this observation was the basis of the first Systems Software and Tools for High Performance Computing Environments workshop in Pasadena (hosted by Caltech) in 1992. Too much effort (time and dollars) is required to program, debug and optimize applications code for massively parallel machines. Without significant improvement on all software technology fronts, programming for a petaflops system may be unaffordable.

As a basis for its hardware technology discussions and projections, the workshop used raw extrapolations of existing SIA projections for the growth of semiconductors. The report noted that the workshop's petaflops projections shoull be used with caution because detailed studies to extend the SIA projections had not been done; thus, projections beyond the year 2000 are subject to considerable uncertainty. Also, at the limits of the SIA projections for chip fabrication, new quantum effects are expected. As a result, it's not at all clear how the semiconductor technology will evolve at lithographic resolutions finer than 0.1μ.

Optical technology, while offering promise for intercommunications, appears to be impractical for optical-only logic. But, the very high bandwidth available in optical devices can be used to manage the tremendous message traffic between high-speed processors and terabytes of global memory. Also, optical storage may be essential for the emerging databases that are accessible world-wide. It may be that superconducting technology can help solve the problems of massive power consumption and heat dissipation that could make a semiconductor-only computer.

If the U.S. expects to play a dominant role in the development of a petaflops computer, it must improve its capabilities to fabricate memory and processors and especially to integrate them. With integrated memory and processors, as described in one of the architectures postulated at the workshop, processor-to-memory bandwidth is maximized. A petaflops computer will be much more memory intensive than today's computers. Dependence on foreign suppliers for huge quantities of memory does not seem to be a reasonable alternative.

A consistent theme of the workshop was that with respect to hardware a petaflops computer could be affordable only if most of its components were commercially available. Yet, it also seems inevitable that design and fabrication of some custom components will be necessary. The issue then becomes the cost of such custom components as a cost-driver. It would take only a few such components to overwhelm the development cost curve. At the same time, to restrict component selection to those commercially available could also severely restrict system design and, ultimately, system performance.

Software technology issues are as important as hardware issues. Today's machines are built to maximize peak throughput—not time to solution. It's not unusual for an application programmer to need months to develop, debug and optimize code that eventually runs in just a few minutes or hours. Current systems are just too hard to program and optimize because the adequate software tools are not available and managing the distributed resources is too complex. Without a serious shift in emphasis in what is important in system design, a petaflops computer will be of little practical value in addressing real-world problems.

Change is slow in the world of programming languages. The scientific community is slowly recognizing that Fortran may not be the most suitable language to exploit the capabilities of massively parallel computers. But, if not Fortran, what? Few replacement candidates seem strong. A petaflops computer will demand a new language. And, the language must significantly ease the processes of programming, debugging, and optimizing.

Architecture Issues. Architecture is the sine qua non: it will ultimately drive (directly or indirectly) all other aspects of a petaflops computer. The Pasadena petaflops workshop examined three different architectures with respect to processor granularity, speed, and latency.

Each architecture imposed different requirements on technology and supported applications. Per processor speeds ranged from 10 gigaflops at the low end to teraflops at the high end. At the high end, latency was assumed to be low: at the other extreme, while local memory access and system-wide bandwidth was good due to processor-in-memory chips, hardware support of shared-memory would be difficult.

The workshop also considered a heterogeneous system composed of the three different architectures. The hybrid architecture achieved petaflops performance although each separate computer was capable of only teraflops-level computing. The architectures considered may, over time, prove to be the appropriate approach. But, other parallel architectures might be feasible, especially for specific kinds of applications.

Latency will continue to be a significant problem for massively parallel machines. One result of the quest for petaflops computing could be that latency-hiding techniques developed for the petaflops machine may be applicable to less powerful machines as well. And, as density increases, another possibility is to develop methods to further exploit the advantages of short latency machines combined with very high clock speeds.

Although clock speed and local memory or system memory access (depending on the architecture chosen) will be crucial, the importance of I/O cannot be overstated. While I/O may not scale linearly as once anticipated, it is and will continue to be a bottle neck until I/O bandwidth and bandwidth for secondary storage are increased appreciably.

2.4 Summary

Clearly, the challenges to developing a petaflops computer are formidable. And, that applies to the *known* challenges.The unknown will be confronted when they emerge. They may—and probably will—fall into most of the distinct areas listed earlier. Perhaps the most important point to be gleaned from this discussion is that working experts think that petaflops computing within 20 years is feasible.

3 Workshop Organization

The Petaflops Frontier Workshop was held in McLean,Virginia on February 6, 1995 at the McLean Hilton hotel. This section includes the workshop organizing committee and their affiliations, the workshop agenda in chronological order along with the presenters and affiliations, and the attendees.

3.1 Organizing Committee

- Thomas Sterling, USRA/CESDIS
- John Dorband, NASA Goddard Space Flight Center
- Michele O'Connell, USRA/CESDIS

3.2 Agenda

- Welcome—Michele O'Connell, USRA/CESDIS
- Why Petaflops? Why Now? Why Here?—Thomas Sterling, USRA/CESDIS
- Keynote Address—George Lake, University of Washington
- Review of Pasadena Petaflops Workshop—Thomas Sterling, USRA/CESDIS
- Introduction to Talks—John Dorband, NASA Goddard Space Flight Center

3.3 Workshop Presentations

Listed below in chronological order are the presentation topics, the presenters and their affiliations[1]

- Processors-In-Memory (PIM) Chip Architectures for Petaflops Computing, Peter Kogge, Notre Dame
- A Case Study of Interactive, Immersive Visualization for Scientific Environments, Rick Stevens, Valerie Taylor and Meena Kandaswamy, Argonne National Laboratory

[1]Justin Porter, University of British Columbia, and Guy Robinson, Syracuse University, who could not attend the workshop provided extended abstracts, "Non von Neuman Instruction Set Architecture" and, "Parallel Computations for Scientific, and Engineering Applications: What Could We Do With Petaflops", respectively, that are included in this report.

- Please Everyone, Take a Futuristic Look at Optical Processing Systems, Chunming Qiao, SUNY
- Strategic Applications for Peta(FL)OPS Computational Systems, Rick Stevens and Valerie Taylor, Agonne National Laboratory
- Taming Massive Parallelism: The Prospects of Opto-Electronic CRCW Shared Memory, Paul Likowicz, University at Karlsruhe
- Enabling Data-Intensive Applications through SDSC Petaflops Computing, Reagan Moore, San Diego Supercomputing
- Design of a Massively Parallel Computer Using Bit Serial Processing Elements, Maurice Aburdene, Bucknell University
- Hierarchical Distributed Genetic Algorithms Control of Simulation-based Optimization: The Need for Petaflops, George Ball, University of Arizona
- A PetaOp/s is Currently Feasible by Computing in RAM, Duncan Elliott, University of Toronto
- Buffet Lunch and Poster Session
- Heterogeneous Computing: One Approach to Sustained Petaflops Performance, H. J. Siegel, Purdue University
- Some Applications Demonstrating the Existing Need for Petaflops Computing in Biomedical Research, Jacob Maizel, National Cancer Institute
- Lightning: A Scalable Dynamically Reconfigurable Hierarchical WDM Network for High-Performance Clustering, Patrick Dowd, NASA Lewis Research Center/SUNY
- Petaflops and the Gravitational N-body Problem, Kevin Olson, George Mason University
- Computational Astrophysics Calculations on Petaflop Computers, Bruce Fryxell, NASA Goddard Space Flight Center
- Easing the Burden on Latency-Tolerance Mechanisms in Petaflops Computers, David Probst, University of Toronto

- Thoughts on Artificial Life in a Petaflop/Second Environment, John Thorp, Cray Research.[2]
- Petaflops Technology: Real Time Image Compensation, Rick Lyon, Hughes STX
- Computational Requirements for Hydrodynamic Turbulence on Petaflops Computers, Anil Deane, George Mason University
- Open Speakers Forum and Discussions
- Plenary Session with Workshops A and C

3.4 Workshop Attendees

The workshop attendees and their affiliations are shown below:

Maurice Aburdene (Bucknell University)
Robin Alford (CESDIS)
Von Backenstose (Department of Commerce)
David Bader (University of Maryland)
George Ball (University of Arizona)
F. D. Bedard (National Security Agency)
George Bell (Stanford University)
Simon Berkovich (George Washington University)
Mike Berry (Department of Defense/USAF)
Bruce Black (Cray Research Inc.)
Andrew Chien (University of Illinois)
Fabien Coelho (École des Mines)
Jarrett Cohen (NASA Goddard Space Flight Center)
John Conery (University of Oregon)
Bob Cox (Cray Computer Corporation)
David Crawford (Electronic Trend)
Dave Curkendall (Jet Propulsion Laboratory)
Anil Deane (George Mason University)
David DiNucci (Computer Science Corporation)
John Dorband (NASA Goddard Space Flight Center)
Patrick Dowd (State University New York)
Duncan Elliott (University of Toronto)
Walter Ermler (Department of Energy)

[2] An extended abstract of this presentation was not available for this report.

Hassan Fallah-Adl (University of Maryland)
Robert Ferraro (Jet Propulsion Laboratory)
Charles Fiduccia (Supercomputing Research Center)
Jim Fischer (NASA Goddard Space Flight Center)
Ian Foster (Argonne National Laboratory)
Bruce Fryxell (George Mason University)
Eugene Gavrilov (Los Alamos National Laboratory)
Norman Glick (National Security Agency)
Peter Gulko (Rebus Technolgies)
Yang Han (George Washington University)
Jim Harris (NASA HQ, Office of Mission to Planet Earth)
R. Michael Hord (ERIM)
Fred Johnson (National Institute of Standards and Technology)
Kamal Khouri (Bucknell University)
David Kilman (Los Alamos National Laboratory)
Steve Knowles (Naval Space Command)
Peter Kogge (Notre Dame University)
John Korah (NASA, EOSDIS)
Joydip Kundu (University of Oregon)
H. T. Kung (Harvard University)
George Lake (University of Washington)
William Leinsberger (Computer Devices International)
Paul Lukowicz (University at Karlsruhe)
Lou Lome (Ballistic Missile Defense Organization)
Serge Lubenec (George Mason University)
Rick Lyon (Hughes STX)
Jacob Maizel (National Cancer Institute)
Yossi Matias (AT&T Bell Labortories)
William Mattus (Villanova University)
Thomas McCormick III (National Security Agency)
Al Meilus (George Washington University)
A. Ray Miller (National Security Agency)
Jose Milovich (Lawrence Livermore National Laboratory)
Samin Mohammed (George Mason University)
Reagan Moore (San Diego Supercomputing Center)
Z. George Mou (Brandeis University)
Samiu Muhammed (George Mason University)
Chrisochoides Nikos (Syracuse University)
Michele O'Connell (CESDIS)

Kevin Olson (George Mason University)
Behrooz Parhami (University of California)
Jeff Pedelty (NASA Goddard Space Flight Center)
Ivars Peterson (Science News)
Larry Picha (CESDIS)
Thierry Porcher (CEA)
David Probst (Concordia University)
Chunming Qiao (State University New York)
Donna Quammen (George Mason University)
Craig Reese (Supercomputing Research Center)
S. Repdauay (CPP)
Michael Rilee (Cornell University)
Allen Robinson (Sandia National Laboratory)
Subhash Saim (NASA Ames Research Center)
Subhash Saini (Computer Sciences Corporation)
Ray Sakardi (National Security Agency)
David Schaefer (George Mason University)
Judith Schlesinger (Supercomputing Research Center)
Vasili Semenov (State University New York)
Bruce Shapiro (National Cancer Institute)
H. J. Siegel (Purdue University)
Margaret Simmons (Los Alamos National Laboratory)
Burton Smith (Tera Computer)
Paul H. Smith (NASA HPCC Office)
Matteo Sonza-Reorda (Politecnico Di Torino)
Thomas Sterling (CESDIS)
Katja Stokley (George Mason University)
Valerie Taylor (Northwestern University)
John Thorp (Cray Research Inc.)
Joe Vaughn (Computing Devices International)
Chris Walter (WW Technology Group)
Pearl Wang (George Mason University)
Nancy Welker (National Security Agency)
Leonard Wisniewski (Dartmouth College)
Paul Woodward (University of Minnesota)
Bill Wren (Honeywell)
Richard Yentis (George Washington University)
Steve Zalesak (NASA Goddard Space Flight Center)
Bernard Zeigler (University of Arizona)

4 Overview of Presentations

This section provides an overview of the workshop presentations in two major categories: architecture and technology; and applications and algorithms. Extended abstracts of the workshop presentations are included in Sections 5 and 6.

4.1 Architecture and Technology Overview

The architecture and technology presentations encompassed a variety of topics and included subjects such as heterogeneous computing environments, developments in processor-in-memory, and methods of employing optical technologies. The extended abstracts of the architecture and technology presentations are included in Section 5.

The first article, by Siegel et al., discusses heterogeneous computing as one approach to sustained petaflops computing. In their presentation the authors discuss the goal of heterogeneous computing, mixed-mode heterogeneous computing and mixed-machine heterogeneous computing. They also present a conceptual model of automatic heterogeneous computing and discuss several "open problems" related to developing and implementing heterogeneous computing. In concluding, the authors make the case that reaching *sustained* petaflops computing will be difficult even if *peak* petaflops performance is achieved. They also note that because in heterogeneous computing each subtask of an application can be matched to the machine that can execute it most effectively, and multiple machines can be used concurrently to process a single application, heterogeneous computing is one possible approach to providing sustained petaflops computing.

Processor-in-chip (PIM) chip architectures are addressed by Kogge who discusses how they can contribute to the development of petaflops computing. Kogge provides details about the EXECUBE PIM chip that merged 100K custom circuits and 4.5 Mbits of dynamic random access memory onto a single die. In addressing the potential tradeoffs of such developments, Kogge discusses the SIA technology projections with respect to two CPU architectures. In concluding that the PIM-based architecture offers the potential to achieve "huge" levels of performance with far fewer chips and lower cost than other approaches, Kogge notes that a number of open questions "absolutely" must be answered before PIM-based systems can become a reality.

The feasibility of petaops computing in RAM is discussed by Elliott et al. by focusing on what might be possible using technology currently available. From their slightly different perspective they support Kogge's position regarding integrating processing power into memory. They suggest radical changes to memory architecture. And, while the majority of the computing power will reside in the SIMD processing elements in memory, they propose a hybrid MIMD-SIMD machine because of the flexibility of MIMD and the economics of SIMD.

Dorband, Aburdene et al. suggest that the shift in emphasis to MIMD machines has been based on "inappropriately drawn" conclusions and that the full potential of SIMD machines has not been fully explored. The objective of their work is to design a massively parallel SIMD architecture having over one million processors. They discuss the design of the processing element and simulating a 3×3 toroidal 2-D mesh architecture.

Porter contends that current superscalar/superpiplined architectures "sidestep" the von Neumann bottleneck of moving data between processors and memory. He addresses the concept by moving decoding and fetching from the processor to memory using a non von Neumann instruction set architecture. Porter also discusses his simulation of a dual processor model to perform the Cooley-Tukey algorithm Fast Fourier Transform benchmark to show the feasibility of distributed-instruction set architectures.

Lukowicz and Tichy discuss the prospects of opto-electronic concurrent-read, concurrent-write shared-memory. Their approach is to exploit the natural parallelism of optical data storage and integrate it into the memory hierarchy of a massively parallel computer. With opto-electronic components they project systems with gigabytes capacity with a latency of $\leq$ 1ns.

The theme of optical components is continued by Dowd who discusses *Lightning*, an architecture based on wavelength-division multiplexing (WDM). A hierarchical structure is used to enable scalability while avoiding the need for multiple wavelength-tunable devices per node. The communications requirements of the architecture are achieved using wavelength-, space- and time-division multiplexing. An advantage to this approach is that bandwidth can be reconfigured dynamically and reallocated to adapt to shifts in traffic patterns.

Optical interconnection networks also are addressed by Chunming Qiao who discussed free-space optical and fiber-optic interconnects. Chunming presents a futuristic topology, Multi-Plane Interconnected Cube (MPIC), but notes that a number of issues such as routing, electronic and optical implementations, and fault-tolerance are still being actively investigated. Chunming calls for increased acceptance of fiber-optic technology, a technology that so far has been largely fostered by the telecommunications industry.

Latency tolerance remains an issue with large-scale systems. Probst addresses the relationship between latency-tolerance and dependence on or independence from data locality. His goal is for neither locality dependence nor cache behavior to constrain available parallelism or degrade the quality of memory bandwidth.

4.2 Applications and Algorithms Overview

The applications and algorithms presentations included topics such as medical and biological requirements, computational fluid dynamics applications, image deblurring in real time, and astrophysics. The extended abstracts are provided in Section 6.

In the first article, Moore argues that with petaflops computational power, data assimilation will become as important to modeling as computational simulation is today. Using data from the San Diego Supercomputer Center, he projects I/O requirements for giga, tera, and petaflops execution rates. Moore suggests that a petaflops computer can be used to process archived data sets that by the year 2000 will be in the petabyte range.

Maizel shows in the next article that several classes of problems exist that have an existing need for petaflops computing capabilities. These problems include the DNA in Human Genome Project, drug design, and organ modeling. Beyond these existing needs, petaflops computing could contribute significantly to the development of virtual medicine and surgery, prosthesis design, and environmental research.

The existing need for petaflops-level computing power also is addressed by Zeigler et al. who discuss the intricacies of high resolution simulation of landscape ecosystems. They present results of their work to support the view that current technology cannot support the large data sets needed to model such systems, or accomplish the work in a usable period of time.

Deane discusses the need for petaflops computing from the perspective of computational fluid dynamics (CFD) problems. Deane discusses the cross-discipline dependence on CFD for such problems as aircraft simulation, atmospheric studies, oceanographic analyses, and studies of solar convection and supernova.

Fryxell's article on computational astrophysics provides additional definition to the problems involved in analyzing astrophysics phenomena. He suggests that in attempting to analyze the requirements for a petaflop computer (for astrophysics problem), it is better to use a general approach for the analysis than one designed for a specific problem

In the next article, Olson specifically addresses the gravitational N-body problem in cosmology. He approaches the problem from both the direct N^2 and tree algorithms to show how petaflops computing capabilities could be applied.

Stevens and Taylor address the need for petaflops computing from the perspective of conceptualizing strategic applications for early in the 21st century. Their applications range from molecular nanotechnology CAD systems to national-scale data mining engines for a wide variety of purposes, to Lunar and Mars bases for backing up a human knowledge repository. For each of the strategic applications, they provide a requirements listing and research issues.

Taylor et al. focus on the work being done in virtual reality that may someday lead to one of the Stevens-Taylor strategic applications. This work uses a Cave Automatic Virtual Environment (CAVE). A petaflops computer will be needed to satisfy the requirements for real-time, high fidelity, interactive and immersive 3-D virtual reality.

Robinson addresses the complications of discrete sets of equations representing physical systems such as an automobile engine or climate modeling, and discusses how a petaflops computer could be used to either improve the modeling or speed up the solution time or both.

5 Architecture and Technology Issues and Challenges

Introduction

This section includes the papers from participants who made presentations on architecture and technology issues and challenges of petaflops computing.

Listed below are the titles of the extended abstracts and their authors:

- Heterogeneous Computing: One Approach to Sustained Petaflops Performance, H.J. Siegel, John K. Antonio, Min Tan, Richard C. Metzger, Richard F. Freund, and Yan Alexander
- Processors-In-Memory (PIM) Chip Architectures for Petaflops Computing, Peter M. Kogge
- A Petaops is Currently Feasible by Computing in RAM, Duncan Elliott
- Design of a Masssively Parallel Computer Using Bit Serial Processing, John E. Dorband, Maurice F. Aburdene, Kamal S. Khouri, Jason E. Piatt, Jianqing Zheng
- Non von Neumann Instruction Set Architecture as an EnablingTechnology in Grand Challenge Systems, Justin S. M. Porter (Note: Porter was unable to make a presentation)
- Taming Massive Parallelism: The Prospects of Opto-Electronic CRCW-Shared Memory, Paul Lukowicz, Walter F. Tichy
- Lightning: A Scalable Dynamically Reconfigurable Hierarchical WDM Network for High Performance Clustering, Patrick W. Dowd
- PETAflops: PErhaps Take A Futuristic Look at Optical Processing Systems, Chunming Qiao
- Easing the Burden on Latency-Tolerance Mechanisms in Petaflops Computers, David K. Probst
- Petaflops Technology: Real Time Image Compensation, Richard G. Lyon

5.1 Heterogeneous Computing: One Approach to Sustained Petaflops Performance

Howard Jay Siegel[1] John K. Antonio[1], Min Tan[1]
Richard C. Metzger[2], Richard F. Freund*, and Yan Alexander Li[1]

[1]Parallel Processing Laboratory, School of Electrical Engineering
Purdue University, West Lafayette, IN 47907-1285, USA

[2]Software Engineering Branch (C3CB), Rome Laboratory
Griffiss AFB, NY 13441-5700, USA

*NCCOSC RDTE, Div. 4221, 53140 Gatchell Road
San Diego, CA 92152-7463, USA

5.1.1 Introduction

A single application task often requires a variety of different types of computation (e.g., operations on arrays versus operations on scalars). Numerous application tasks that have more than one type of computational characteristic are now being mapped onto high-performance computing systems. Existing supercomputers generally achieve only a fraction of their peak performance on certain portions of such application programs. This is because different subtasks of an application can have very different computational requirements that result in different needs for machine capabilities. In general, it is currently impossible for a single machine architecture to satisfy all the computational requirements of various subtasks in certain applications equally well [9]. Thus, a more appropriate approach for high-performance computing is to construct a heterogeneous computing environment.

A heterogeneous computing (*HC*) system provides a variety of architectural capabilities, orchestrated to perform an application whose subtasks have diverse execution requirements. Two types of HC systems are mixed-mode machines and mixed-machine systems [31]. A mixed-mode machine is a single parallel machine that is capable of operating in either the SIMD or MIMD mode of parallelism and

[1] Supported by Rome Laboratory under contract number F30602-94-C-0022 and by NRaD under contract number N68786-91-D-1799. Some of the research discussed used equipment supported by the National Science Foundation under grant number CDA-9015696.

[2] Supported by AFOSR under RL JON 2304F2TK.

can dynamically switch between modes at instruction-level granularity with generally negligible overhead. There are various trade-offs between the SIMD and MIMD modes of parallelism (see [12, 24, 25]), and mixed-mode machines can exploit these by matching each subtask with the mode that results in the best overall performance. Studies have shown that a mixed-mode machine can outperform a single-mode machine with the same number of processors for a given algorithm (e.g., [22]).

A number of prototype mixed-mode machines have been built that, to varying degrees, incorporate the mixed-mode approach. These include PASM (Purdue University, USA) [24, 26], TRAC (University of Texas at Austin, USA) [19], OPSILA (University of Nice, France) [4], Triton (University of Karlsruhe, Germany) [11, 21], and EXECUBE (IBM Federal Systems Division, USA) [15]. The reconfigurability of a mixed-mode system can increase the percentage of a machine's peak performance that an application can attain.

A *mixed-machine system* is a heterogeneous suite of different types of independent machines interconnected by a high-speed network. The goal is to match each subtask to the machine that results in the lowest overall task execution time. Unlike mixed-mode machines, switching execution among machines in a mixed-machine system can require measurable overhead because data may need to be transferred among machines. Thus, the mixed-machine systems considered in this paper are assumed to have high-speed connections among machines that make decomposition at the subtask level feasible. Also, in mixed-machine systems, the set of subtasks may be executed as an ordered sequence or concurrently.

To exploit HC systems, a task must be decomposed into subtasks, where each subtask is computationally homogeneous. As stated above, the subtasks are then assigned to and executed with the machines (or modes) that result in a minimal overall execution time for the task. Typically, users must specify this decomposition and assignment. One long-term pursuit in the field of heterogeneous computing is to do this automatically.

Figure 1 shows a hypothetical example of an application program whose various subtasks are best suited for execution on different machine architectures, i.e., vector, SIMD, MIMD, data-flow, and special purpose [7]. The execution time for the heterogeneous suite includes inter-machine communications. Percentages are based on 100% being the total execution time on the baseline serial system, but are not drawn to scale. Executing the whole program on a vector supercomputer only gives twice the performance achieved by a baseline serial machine. The vector portion of the program can be executed significantly faster. However, the non-vector portions of the program may only have a slight improvement in execution time due to the mismatch between each subtask's unique computational requirement and the machine architecture being used. Alternatively, the use of five different machines, each matched to the computational requirements of the subtasks for which it is used, can result in an execution 20 times as fast as the baseline serial machine, including the overhead time for inter-machine communication.

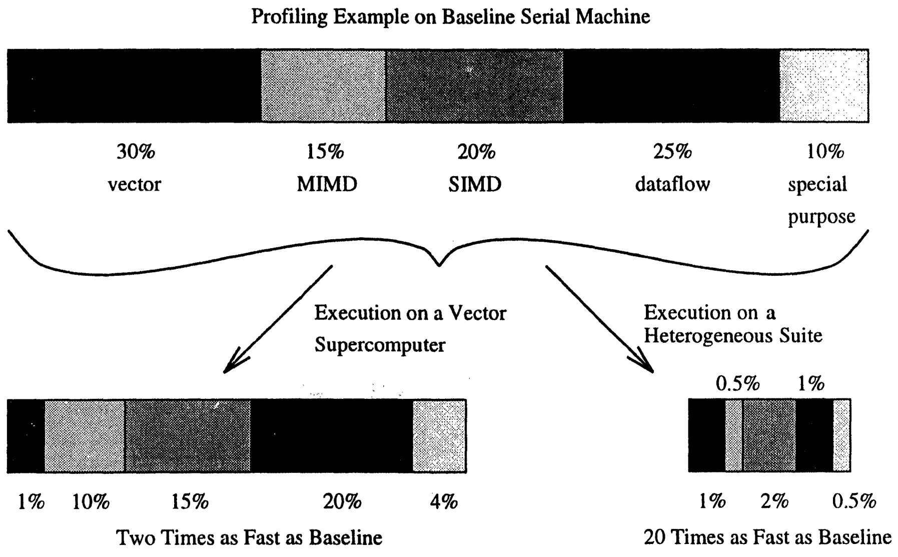

Profiling Example on Baseline Serial Machine
30%
vector
15%
MIMD
20%
SIMD
25%
dataflow
10%
special purpose
Execution on a Vector Supercomputer
Execution on a Heterogeneous Suite
1%
10%
15%
20%
4%
Two Times as Fast as Baseline
0.5%
1%
1%
2%
0.5%
20 Times as Fast as Baseline

Figure 1: Hypothetical Example of the Advantages of Heterogeneous Computing [7].

One of the issues discussed at the workshop on "Grand Challenges in Computer Architecture for the Support of High Performance Computing," reported in [23], was achieving petaflops performance. Among the problems discussed, one that was stressed was delivering "usable" (versus "peak") petaflops performance. Even if it is assumed that machines with "peak" petaflops performance can be constructed, reaching the goal of "sustained" (or usable) petaflops performance when executing real applications is still a very difficult problem.

The use of both types of HC systems to achieve sustained petaflops performance is proposed in this paper. Mixed-mode systems allow an application task to exploit those aspects of SIMD and MIMD modes that will reduce its execution time. This will help increase the average percentage of peak petaflops the application can sustain.

Based on the history of commercial supercomputers, it is reasonable to expect that a variety of different architectural designs will be represented in the set of machines constructed with peak petaflops performance in the future. Thus, mixed-machine HC can help provide sustained petaflops performance because each subtask of an application can be matched to a machine or mode that can execute it most effectively, and multiple machines can be used concurrently to process a single application. The matching will increase the average percentage of a machine's peak performance that can be achieved. The concurrent use of multiple machines will allow the sustained FLOPS provided by each of the machines to be added together when determining the total task sustained FLOPS during the overlap period. Thus, the mixed-machine HC approach can help provide sustained petaflops.

The rest of this paper will focus on mixed-machine HC systems. Examples of existing HC systems are presented in the next section. Following that a conceptual model for automatic HC is introduced. The paper concludes with a discussion of open problems in the field of HC. Much of the material given here is summarized from [25].

5.1.2 Examples of Mixed-Machine HC

Overview

Three examples of existing HC systems are very briefly introduced here. In the first two, the decomposition of tasks into subtasks and the assignment of subtasks to machines were user specified. The third, SmartNet, schedules tasks in an HC system. The long-term goal of automatic HC is discussed in the next section.

Simulation of Mixing in Turbulent Convection at the Minnesota Supercomputer Center

In [14], the usefulness of an HC system developed at the Minnesota Supercomputer Center is demonstrated through a particular application involving the simulation of mixing in turbulent convection in three dimensions. The HC system consists of Thinking Machines' CM-200 and CM-5, a CRAY 2, and a Silicon Graphics VGX workstation, all interconnected over a high-speed HiPPI (high-performance parallel interface) network. The required calculations for the simulation were divided into three phases: (1) calculation of velocity and temperature fields, (2) calculation of particle traces, and (3) calculation of particle distribution statistics and refinement of the temperature field.

The velocity and temperature fields associated with the phase 1 calculations are governed by two second order partial differential equations. These computations were done on the CM-5. The particle traces were calculated by solving a set of ordinary differential equations that are dependent on the velocity field solution computed in phase 1. Initially, this computation was attempted on the CM-200 by employing an Eulerian approach, but could not be done because a prohibitive amount of memory was required. Instead, the three-dimensional simulations were implemented using a vectorized Lagrangian approach on the CRAY 2, which required substantially less memory than the parallel Eulerian scheme. The CM-200 was used to calculate statistics of the particle distribution and to assemble a three-dimensional temperature field from the associated spline coefficients (phase 3). The final results were then sent to a SGI VGX workstation where they were visualized using an interactive volume renderer.

Interactive Rendering of Multiple Earth Science Data Sets on the CASA Testbed

The CASA testbed interconnects several remote sites including the California Institute of Technology, San Diego Supercomputer Center, Jet Propulsion Laboratory (*JPL*), and Los Alamos National Laboratory [2, 27]. The computational resources of the testbed consist of various parallel and vector machines including an Intel Touchstone Delta, Thinking Machines' CM-5 and CM-200, CRAY Y-MP8/864, Y-MP/264, and Y-MP/232, and a number of workstations and specialized visualization engines.

One of the applications developed on the CASA testbed involves interactive three-dimensional rendering of multiple Earth science data sets. Functional modules were identified and optimized for specific machines. Initially, raw data sets are transferred to one of the two-dimensional functional modules for processing. The two-dimensional modules manipulate image and/or elevation data via a number of different algorithms. Most of the two-dimensional modules were developed for the CRAY Y-MP/232 at JPL and the CRAY Y-MP8/864 at the San Diego Supercomputer Center. Two of the two-dimensional modules were implemented on the CM-5 and CM-200 located at Los Alamos. Output from the two-dimensional modules are sent over the network to the three-dimensional rendering process, which was implemented on the Intel Touchstone Delta located at the California Institute of Technology.

SmartNet

SmartNet is a "real-time" look-ahead/look-back, near optimal scheduler/planner for heterogeneous systems being designed and developed at NRaD (NCCOSC RDTE) [8]. It demonstrates marked improvement over opportunistic load-balancing and other traditional methods of scheduling, even when only partial information on "affinities" is available.

The basic concept of SmartNet may be described as solving a mathematical programming problem to minimize the objective function, total time on a task set, taking into account the times to compute task i on machine j, as well as the latency time for any needed data transfers involved in computing task i on machine j. The constraint in this formulation is the sum of the costs (of machines and networks). A matrix (called an ETC matrix or Expected Time to Complete) is assumed to contain the estimated time it would take task i to execute on machine j, including any latency time required for transfers of data. In addition, it is assumed that each machine may have previously scheduled tasks either executing or awaiting execution. Given this scenario, the goal is to minimize the objective function of the total time required for all new tasks to be finished on all machines. The objective function is not to minimize the compute time for any specific job, but rather to maximize the overall throughput.

SmartNet is currently operational and performs the functions discussed above. Various extensions to SmartNet are under development.

5.1.3 A Conceptual Model for HC

A conceptual model for the automatic assignment of subtasks to machines in an HC environment is shown in Figure 2. This model builds on the one presented in [9]. Figure 2 is referred to as a "conceptual" model because no complete automatic implementation currently exists. As stated earlier, automatic decomposition and assignment is a long-term goal in the field of HC.

In stage 1 of the model, a set of descriptive parameters is generated that is represented as the general characteristics of both the computational requirements of the applications and the machine capabilities of the HC system. These parameters define the multidimensional decision space to be used for describing and matching subtasks and machines. Information about the expected types of application tasks to be executed and about the machines that currently exist in the heterogeneous suite is used to generate these parameters.

For each parameter, a corresponding computational requirement and a corresponding machine architecture feature are derived. For example, considering the parameter "floating point operations," the computational requirements of the application tasks to be quantified are the number and types of floating point operations needed to perform the calculation.

The architecture feature of the machines in the heterogeneous suite to be quantified is the speed for these different types of floating point operations.

A particular parameter is included for further consideration in the following stages of the model only if both the architecture features and the related compu-

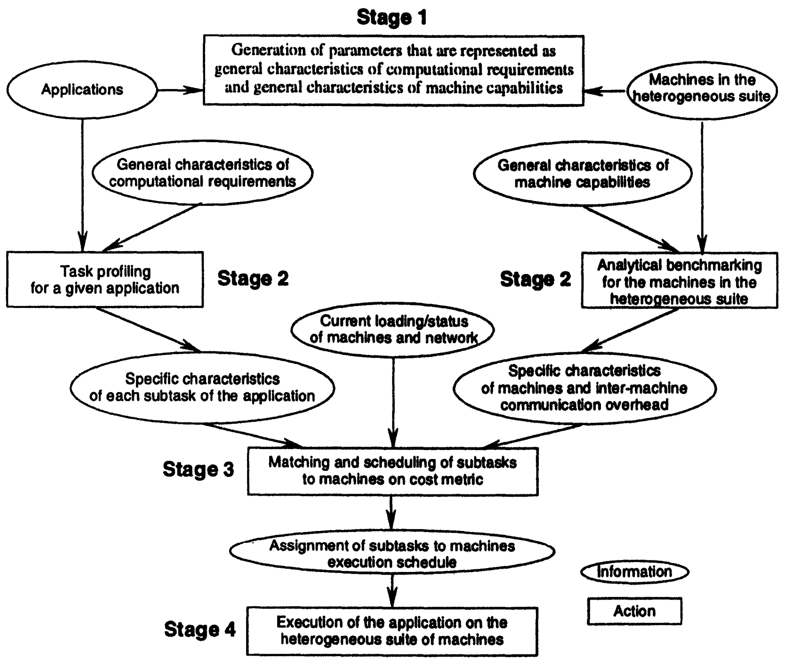

Stage 1
Generation of parameters that are represented as general characteristics of computational requirements and general characteristics of machine capabilities
Applications
Machines in the heterogeneous suite
General characteristics of computational requirements
General characteristics of machine capabilities
Task profiling for a given application
Stage 2
Stage 2
Analytical benchmarking for the machines in the heterogeneous suite
Current loading/status of machines and network
Specific characteristics of each subtask of the application
Specific characteristics of machines and inter-machine communication overhead
Stage 3
Matching and scheduling of subtasks to machines on cost metric
Assignment of subtasks to machines execution schedule
Information
Action
Stage 4
Execution of the application on the heterogeneous suite of machines

Figure 2: Conceptual Model of the Automatic Assignment of Subtasks to Machines in an HC environment [25]

tational requirements exist. For example, if the given applications have no floating point operations, then it is not necessary to evaluate the machine capabilities for executing floating point operations in stage 2. As another example, if there is no vector machine available in the heterogeneous suite, vectorizable code may be excluded from the set of computational requirements that must be considered.

After stage 1, a collection of corresponding features of the application tasks and machines in the heterogeneous suite can be enumerated. These features determine the dimensions of this automatic assignment problem for the given applications and the given HC system. Each of these dimensions represents a specific parameter, which characterizes computational requirements and the related machine capabilities that need to be considered in the later stages of the model. The total number of features enumerated determines the complexity of this automatic problem. An important aspect of the chosen parameters is that they evolve dynamically when new types of applications and/or new types of machines are added.

In stage 2, two characterization steps, task profiling and analytical benchmarking, are used to quantify the features and transform them into quantitative data. *Task* *profiling* is a method used to identify the types of computational requirements that are actually present in a specific application program. The task is decomposed into subtasks, such that the computational requirements for each subtask are homogeneous and quantified. The term often used for this characterization step in the existing literature is code profiling. The reason for using task profiling instead is that, to identify the types of computational requirements present in a specific task, both the code and characteristics of the data upon which the specified HC system will operate must be examined. *Analytical benchmarking* is a procedure that provides a measure of how effectively each of the available machines in the heterogeneous suite performs on each of the types of computations being considered. Examples of proposed approaches to task profiling and analytical benchmarking are in [10, 34, 35].

In stage 3, the results of previous stages and the information about the current loading and "status" of the machines and inter-machine network are used to generate an assignment of the subtasks to machines in the HC system based on certain cost metrics. The "status" could include such items as whether the machines and network are fully or partially functioning due to faults, and when other tasks using the machines/network are expected to complete. The most common cost metric for HC is minimization of the overall execution time (including the inter-machine communication time) of a given application task on a particular HC system. Another interesting metric is finding the most appropriate suite of heterogeneous machines for a given collection of applications, such that the cost of the corresponding HC system is minimized for a given set of execution time constraints [6]. A variety of techniques have been proposed in the literature for selecting a machine for each subtask based on certain cost metrics (e.g., [3, 5, 6, 16, 17, 18, 20, 29, 30, 31, 32]).

Stage 4 of the model is the execution of the given applications on the machines in the HC system. Because the loading of the machines and network may change and some faults may occur, it is sometimes necessary to reallocate machines for certain subtasks of the application program. Under such circumstances, the current loading and status of the machines and network are updated and stage 3 is reactivated to decide the new assignment of subtasks. Finding techniques for the actual migration of a subtask from one type of machine to another during execution is a difficult problem; one approach is described in [1].

It is important to note that the mathematical formulation and automatic assignment of subtasks to a heterogeneous suite of machines connected by high-speed links are relatively new fields in HC. Thus, most of the automatic methods that have been proposed for stages 2 and 3 are frameworks that require further research before they can be parts of a working system.

5.1.4 Open Problems

A great many open problems need to be solved before heterogeneous computing can be made available to the average applications programmer in a transparent way. Many (possibly even most) need to be addressed just to facilitate near-optimal practical use of real heterogeneous suites in a "visible" (i.e., user specified) way. Below is a brief discussion of some of these open problems; it is far from exhaustive, but it will convey the types of issues that need to be addressed. Others may be found in [13, 28].

Implementation of an automatic HC programming environment, such as envisioned in Section 3, will require a great deal of research for devising practical and theoretically sound methodologies for each component of each stage. A general open question that is particularly applicable to stages 1 and 2 of the conceptual model is: "What information should (must) the user provide and what information should (can) be determined automatically?" For example, should the user specify the subtasks within an application or can this be done automatically? Future HC systems will probably not completely automate all of the steps in the conceptual model. A key to the future success of HC hinges on striking a proper balance between the amount of information expected from the user (i.e., effort) and the level of performance delivered by the system.

To program an HC system, it would be best to have machine-independent programming languages [33] that allow the user to augment the code with compiler directives. The programming language and user specified directives should be designed to facilitate (a) the compilation of the program into efficient code for any of the machines in the suite, (b) the decomposition of tasks into homogeneous subtasks, and (c) the use of machine-dependent subroutine libraries.

Along with programming languages, there is a need for debugging and performance tuning tools that can be used across an HC suite of machines. This involves research in the areas of distributed programming environments and visualization tools.

Operating system support for HC is needed. This includes techniques applicable at both the local machine level and at the system-wide network level.

Ideally, information about the current loading and status of the machines in the HC suite and the network that is linking these machines should be incorporated into the matching and scheduling decisions. Many questions arise here: what information to include in the status (e.g., faulty or not, pending tasks), how to measure current loading, how to effectively incorporate current loading information into matching and scheduling decisions, how to communicate and structure the loading and status information in the other machines, how often to update this information, and how to estimate task/transfer completion time?

There is much ongoing research in the area of inter-machine data transport. This research includes the hardware support required, the software protocols required, designing the network topology, computing the minimum time path between two machines, and devising rerouting schemes in case of faults or heavy loads. Related to this is the data reformatting problem, involving issues such as data type storage formats and sizes, byte ordering within data types, and machines' network-interface buffer sizes.

Another area of research pertains to methods for dynamic task migration between different parallel machines at execution time. This could be used to rebalance loads or if a fault occurs. Current research in this area involves how to move an executing task between different machines and determining how and when to use dynamic task migration for load balancing.

Lastly, there are policy issues that require system support. These include what to do with priority tasks, what to do with priority users, what to do with interactive tasks, and security.

5.1.5 Conclusions

Even if it is assumed that machines with *peak* petaflops performance can be constructed, reaching the goal of *sustained* petaflops performance is still a very difficult problem. Heterogeneous computing can help through the reconfigurability of mixed-mode machines and the flexibility of mixed-machine systems.

Based on the history of supercomputers, it is reasonable to expect a variety of different architectural designs to be represented in the set of future petaflops machines. Thus, one possible approach to providing sustained petaflops performance is through the use of HC, where each subtask of an application can be matched to the machine that can execute it most effectively, and multiple machines can be used concurrently to process a single application.

There is clearly a gap between the state-of-the-art in practical HC (illustrated in Section 5.1.2) and automating all of the steps characterized by the conceptual model of Section 5.1.3. In particular, stages 1 through 3 of the conceptual model are typically done entirely by the user, while some aid is provided for the user for stage 4 by existing tools and environments. Thus, although the uses of existing HC systems demonstrate the significant potential benefit of HC, the amount of effort currently required to implement an application on an HC system can be substantial. Future research on the above open problems will improve this situation and make HC more viable.

5.1.6 Acknowledgments

The authors thank J. M. Siegel for her valuable comments.

5.1.7 References

[1] J. B. Armstrong, H. J. Siegel, W. E. Cohen, M. Tan, H. G. Dietz, and J. A. B. Fortes, "Dynamic Task Migration from SPMD to SIMD Virtual Machines," 1994 Int'l Conf. Parallel Processing, Vol. II, Aug. 1994, pp. 160– 169.

[2] L. Bergman, H-W. Braun, B. Chinoy, A. Kolawa, A. Kuppermann, P. Lyster, C. R. Mechoso, P. Messina, J. Morrison, D. Stanfill, W. St. John, and S. Tenbrink, "CASA Gigabit Testbed 1993 Annual Report: A Testbed for Distributed Supercomputing," Technical Report CCSF-33, Caltech Concurrent Supercomputing Facilities, Pasadena, CA, May 1993.

[3] S. Chen, M. M. Eshaghian, A. Khokhar, and M. E. Shaaban, "A Selection Theory and Methodology for Heterogeneous Supercomputing," Workshop on Heterogeneous Processing, Apr. 1993, pp. 15–22.

[4] P. Duclos, F. Boeri, M. Auguin, and G. Giraudon, "Image Processing on a SIMD/SPMD Architecture: OPSILA," 9th Int'l Conf. Pattern Recognition, Nov. 1988, pp. 14–17.

[5] M. M. Eshaghian and R. F. Freund, "Cluster-M Paradigms for High-Order Heterogeneous Procedural Specification Computing," Workshop on Heterogeneous Processing, Mar. 1992, pp. 47–49.

[6] R. F. Freund, "Optimal Selection Theory for Superconcurrency," Supercomputing '89, Nov. 1989, pp. 699–703.

[7] R. F. Freund, "SuperC or Distributed Heterogeneous HPC," Computing Systems in Engineering, Vol. 2, No. 4, 1991, pp. 349–355.

[8] R. F. Freund, "The Challenges of Heterogeneous Computing," Parallel Systems Fair at the 8th Int'l Parallel Processing Symp., Apr. 1994, pp. 84–91.

[9] R. F. Freund and H. J. Siegel, "Guest Editors' Introduction: Heterogeneous Processing," IEEE Computer, Vol. 26, No. 6, June 1993, pp. 13–17.

[10] A. Ghafoor and J. Yang, "Distributed Heterogeneous Supercomputing Management System," IEEE Computer, Vol. 26, No. 6, June 1993, pp.78–86.

[11] C. G. Herter, T. M. Warschko, W. F. Tichy, and M. Philippsen, "Triton/1: A Massively-Parallel Mixed-Mode Computer Designed to Support High Level Languages," Workshop on Heterogeneous Processing, Apr. 1993, pp. 65–70.

[12] L. H. Jamieson, "Characterizing Parallel Algorithms," in The Characteristics of Parallel Algorithms, L. H. Jamieson, D. B. Gannon, and R. J. Douglass, eds., MIT Press, Cambridge, MA, 1987, pp. 65–100.

[13] A. Khokhar, V. K. Prasanna, M. Shaaban, and C. L. Wang, "Heterogeneous Computing: Challenges and Opportunities," IEEE Computer, Vol. 26, No. 6, June 1993, pp. 18–27.

[14] A. E. Klietz, A. V. Malevsky, and K. Chin-Purcell, "A Case Study in Metacomputing: Distributed Simulations of Mixing in Turbulent Convection," Workshop on Heterogeneous Processing, Apr. 1993, pp. 101–106.

[15] P. M. Kogge, "EXECUBE-A New Architecture for Scalable MPPs," 1994 Int'l Conf. Parallel Processing, Vol. I, Aug. 1994, pp. 77–84.

[16] C. Leangsuksun and J. Potter, "Problem Representations for an Automatic Mapping Algorithm on Heterogeneous Processing Environments," Workshop on Heterogeneous Processing, Apr. 1993, pp. 48–53.

[17] Y. A. Li, J. K. Antonio, H. J. Siegel, M. Tan, D. W. Watson, "Estimating the Distribution of Execution Times for SIMD/SPMD Mixed-Mode Programs," Heterogeneous Computing Workshop, April 1995, to appear.

[18] D. J. Lilja, "Experiments with a Task Partitioning Model for Heterogeneous Computing," Workshop on Heterogeneous Processing, Apr. 1993, pp. 29–35.

[19] G. J. Lipovski and M. Malek, Parallel Computing: Theory and Comparisons, John Wiley & Sons, New York City, NY, 1987.

[20] B. Narahari, A. Youssef, and H. A. Choi, "Matching and Scheduling in a Generalized Optimal Selection Theory," Heterogeneous Computing Workshop, Apr. 1994, pp. 3–8.

[21] M. Philippsen, T. Warschko, W. F. Tichy, and C. Herter, "Project Triton: Towards Improved Programmability of Parallel Machines," 26th Hawaii Int'l Conf. System Sciences, Jan. 1993, pp. 192–201.

[22] G. Saghi, H. J. Siegel, and J. L. Gray, "Predicting Performance and Selecting Modes of Parallelism: A Case Study Using Cyclic Reduction on Three Parallel Machines," J. Parallel and Distributed Computing," Vol. 19, No. 3, Nov. 1993, pp. 219–233.

[23] H. J. Siegel, S. Abraham, W. L. Bain, K. E. Batcher, T. L. Casavant, D. DeGroot, J. B. Dennis, D. C. Douglas, T. Feng, J. R. Goodman, A. Huang, H. F. Jordan, J. R. Jump, Y. N. Patt, A. J. Smith, J. E. Smith, L. Snyder, H. S. Stone, R. Tuck, and B. W. Wah "Report of the Purdue Workshop on Grand Challenges in Computer Architecture for the Support of High Performance Computing," J. Parallel and Distributed Computing, Vol. 16, No. 3, pp. 199–211, Nov. 1992.

[24] H. J. Siegel, J. B. Armstrong, and D. W. Watson, "Mapping Computer-Vision-Related Tasks onto Reconfigurable Parallel Processing Systems," IEEE Computer, Vol. 25, No. 2, Feb. 1992, pp. 54–63.

[25] H. J. Siegel, J. K. Antonio, R. C. Metzger, M. Tan, and Y. A. Li, "Heterogeneous computing," in Handbook of Parallel and Distributed Computing, edited by A. Y. Zomaya, McGraw-Hill, 1995, to appear (also Purdue University, EE School, Technical Report TR-EE 94-37, Dec. 1994).

[26] H. J. Siegel, T. Schwederski, W. G. Nation, J. B. Armstrong, L. Wang, J. T. Kuehn, R. Gupta, M. D. Allemang, D. G. Meyer, and D. W. Watson, "The Design and Prototyping of the PASM Reconfigurable Parallel Processing System," to appear in Parallel Computing: Paradigms and Applications, A. Y. Zomaya, ed., Chapman and Hall, London, U.K., 1995.

[27] "Special Report: Gigabit Network Testbeds," IEEE Computer, Vol. 23, No. 9, Sept. 1990, pp. 77–80.

[28] V. S. Sunderam, "Design Issues in Heterogeneous Network Computing," Workshop on Heterogeneous Processing, revised edition, Mar. 1992, pp. 101–112.

[29] M. Tan, J. K. Antonio, H. J. Siegel, and Y. A. Li, "Scheduling and Data Relocation for Sequentially Executed Subtasks in a Heterogeneous Computing System," Heterogeneous Computing Workshop, April 1995, to appear.

[30] L. Tao, B. Narahari, and Y. C. Zhao, "Heuristics for Mapping Parallel Computations to Heterogeneous Parallel Architectures," Workshop on Heterogeneous Processing, Apr. 1993, pp. 36–41.

[31] D. W. Watson, J. K. Antonio, H. J. Siegel, and M. J. Atallah, "Static Program Decomposition Among Machines in an SIMD/SPMD Heterogeneous Environment with Non-Constant Mode Switching Costs," Heterogeneous Computing Workshop, Apr. 1994, pp. 58–65.

[32] M. Wang, S.-D. Kim, M. A. Nichols, R. F. Freund, H. J. Siegel, and W. G. Nation, "Augmenting the Optimal Selection Theory for Superconcurrency," Workshop on Heterogeneous Processing, Mar. 1992, pp. 13–22.

[33] C. C. Weems, G. E. Weaver, and S. G. Dropsho, "Linguistic Support for Heterogeneous Parallel Processing: a Survey and an Approach," Heterogeneous Computing Workshop, Apr. 1994, pp. 81–88.

[34] J. Yang, I. Ahmad, and A. Ghafoor, "Estimation of Execution Times on Heterogeneous Supercomputer Architecture," 1993 Int'l Conf. Parallel Processing, Vol. I, Aug. 1993, pp. 219–225.

[35] J. Yang, A. Khokhar, S. Sheikh, and A. Ghafoor, "Estimating Execution Time for Parallel Tasks in Heterogeneous Processing (HP) Environment," Heterogeneous Computing Workshop, Apr. 1994, pp. 23–28.

5.2 Processors-In-Memory (PIM) Chip Architectures for Petaflops Computing

Peter M. Kogge
McCourtney Chair in Computer Science & Engineering
IEEE Fellow, IBM Fellow (retired)
384 Fitzpatrick, University of Notre Dame
Notre Dame, IN 46556

5.2.1 Introduction

One of the three architectures proposed at the February 1994 Petaflops workshop in Pasadena, California [1] revolved around placing multiple complete CPUs and interconnect logic on a high density memory chip. This single-chip type Processor-In-Memory (PIM) MPP-building block chip could then be cascaded as often as needed to meet the particular performance levels desired. For petaflops levels, systems of as little as a few thousand chips appeared feasible. This was upwards of two orders of magnitude fewer chips than for estimates of the other two architectures.

This paper will develop in more detail than at the [Pasadena] Workshop the more important tradeoffs involved with PIM architecture chips. Technology trends over the next twenty years will be coupled with both variations in architecture and ratios of processing speed to memory. Also addressed are the most important technological challenges that still need to be resolved to make PIM chips a wide spread reality.

As background, the "proof of concept" for this single chip type MPP block is a recently announced PIM chip termed EXECUBE [2, 3]. A 5V 0.8 μm CMOS technology merged 100K custom circuits and 4.5 Mbits of DRAM onto a single die. The logic implemented eight complete 16-bit CPUs, plus four DMA channels each that interconnected the CPUs internally into a 3-dimensional binary hypercube of eight processing elements (PEs). The DRAM memory is partitioned, giving each CPU its own independent 64 KB memory, as pictured in Figure 1. One of the DMA channels from each CPU leaves the chip, permitting larger arrays of chips to be tied together directly, without any external interface glue logic or chips. Arrays of up to 64 such chips are currently in operation.

The instruction set of the CPU was optimized to both minimize gates, perform a variety of computational operations, and work in either an MPP SIMD or MIMD mode. We kept each CPU as simple as possible, and avoided silicon-consuming speedup techniques, such as caches and extensive pipelining, that do not return performance gains proportional to their cost in silicon. Significant thought was given to communications patterns, and the inter-PE DMA channels included a wide variety of circuit, packet, and multiplex/demultiplex functions. Also, thought was given on how to utilize the huge memory bandwidth available to each CPU at its own row buffer on it's DRAM interface and several new instructions were added. At 25 MHz, each CPU can peak at 6.25 MIPS, for a 50 MIPS total per chip.

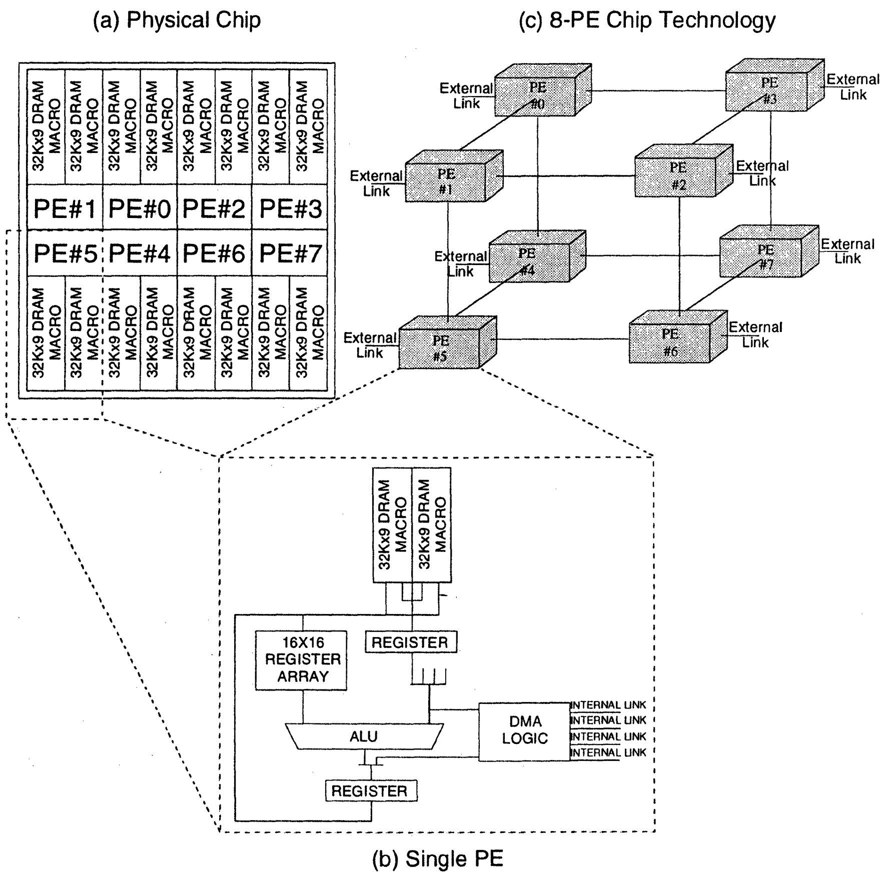


Figure 1: The EXECUBE Processor-In-Memory Architecture

Estimates comparing the chip with other MPP architectures [2] indicate that it is upwards of 10× more efficient in terms of computation per square of silicon than any other existing approach. Similar numbers hold in terms of how it uses the native bandwidth available from the DRAM macros, how the off- chip contacts are used to enhance communications for MPP applications, and even in power dissipation per million instructions.

Given the huge advantage this type of chip architecture has for MPP applications, one can ask what are the potentials for future generations of it, and what would be realistic design points to choose if a petaflop was the ultimate goal?

To get some insight into the tradeoffs possible, we can use the SIA projections [4] on technology, and sketch out trend lines for various possible configurations.

5.2.2 SIA Projections and CPU Architecture

To do this we assume two different CPU architectures. The first, based on the EXECUBE experience, assumes that each CPU is designed simply, and is optimized for fixed point computations. For this we assumed an EXECUBE-like 12K circuit CPU which executes an average instruction in about 2.5 clock cycles. The second CPU assumes a design optimized for floating point, but as with EXECUBE, a simpler (but more efficient in terms of FLOPS/silicon) design point is chosen than what is common in high end microprocessors today. We assume a 100K circuit CPU that can operate on the average at 1 FLOP per clock.

The other major assumption we make is that in a mixed DRAM/logic configuration and at any projected point in time, we can smoothly vary the transistor usage on one chip from 100% logic (using the maximum projected logic density) to 100% DRAM (assuming the maximum projected DRAM density). Thus, we can look at different numbers of CPUs on a chip, with different amounts of memory available to them.

The reason for this latter tradeoff is that during the workshop it became apparent that the major economic constraint on reaching a petaflops system was in the cost of the memory system to support it. Based on typical rules of thumb, a petaflop would require about a petabyte of memory, which even with very dense DRAM, would be in the order of 10,000s of chips. When this was realized, the application workgroup at the workshop came to the conclusion that there were reasonable petaflops applications where at least an "$N^{3/4}$" rule would apply, meaning that perhaps only about 32 terabytes of memory might be need for some applications. Instead of the typical "1 byte per FLOP" rule, this translates into a "0.03 byte/FLOP" rule.

Figure 2 rolls these design assumptions, together with the SIA projections, into a spectrum of potential chip and system configurations assuming an EXECUBE-like largely fixed point CPU macro. (Note that this chart assumes extending the 1992 SIA projections out through 2010 and 2013). Figure 3 does the same for the assumed floating point CPU macro. The calculations behind the (a) chart in each

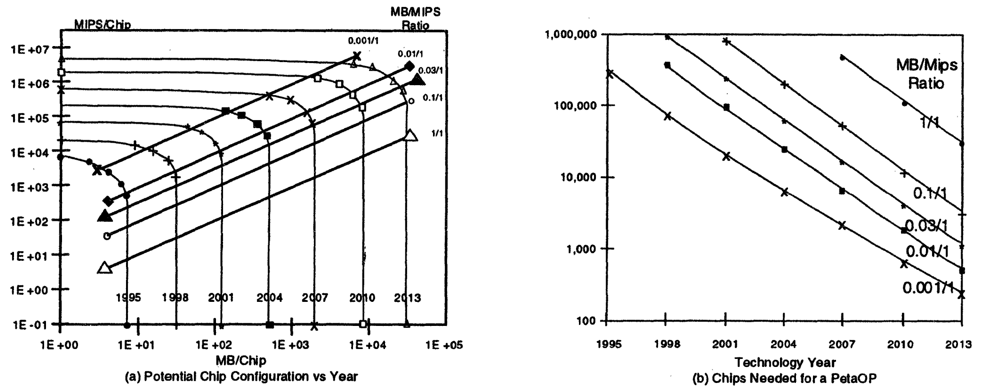


Figure 2: PIM Configurations for a PetaOP

figure were performed at several different year points, and took the projected logic density to determine how many CPUs might fit on different percentages of a chip. From this, and the projected on chip clock speeds, we determined a projected per chip performance number. This was plotted against the amount of memory that could be placed in the remainder of the chip (the "knee-shaped" curves). Through these curves were then drawn straight lines that represent different ratios of storage to performance, to match the above discussion.

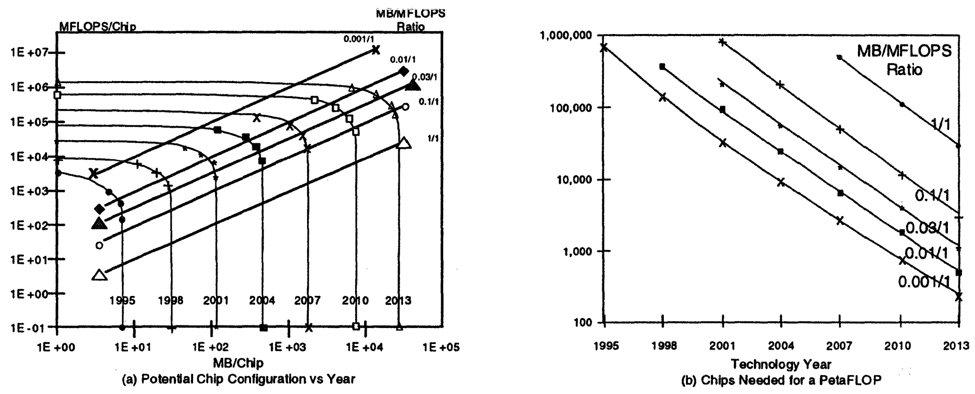


Figure 3: PIM Configurations for a Petaflop

The (b) charts in each figure then use the intersections of these pairs of curves to determine how many chips would be needed to reach a petaflops system, again for different ratios of memory to performance. The numbers agree with the feeling of the Pasadena workshop, namely that a PIM-based architecture has the potential to achieve huge levels of performance with far fewer chips (and thus cost) than the other approaches.

5.2.3 Open Issues

As encouraging as these results are, there are several open questions that absolutely must be answered before such systems can become a reality, and which ought to be the subject of ongoing research:

1. Selecting an ISA and CPU organization that together offers the best possible performance per square of silicon. (This includes developing architectural techniques to utilize the tremendous memory bandwidth available when the CPU is physically right next to its primary memory on the same chip.)
2. Controlling power in the design of each CPU so that the overall power does not exceed our ability to cool the systems in a cost- efficient manner
3. Developing 3-dimensional packaging schemes that permit multiple PIM chips to be packaged close together, thus minimizing the interconnection costs and communications time of flight inherent in building networks of 1000s of chips
4. Selecting an inter-PE topology, and fast enough interconnection paths, to permit this performance to be efficiently utilized on real problems
5. Developing software tools which permit easy construction of programs that can utilize 100,000s of PEs simultaneously

5.2.4 References

[1] Sterling, T., P. Messina, and P. Smith., Enabling Technologies for Peta(FL)ops Computing, Cal Tech Report CCSF-45, July 1994.

[2] Kogge, P. M., "The EXECUBE Approach to Massively Parallel Processing," 1994. Int. Conf. on Parallel Processing, Chicago, IL, August, 1994.

[3] Kogge, P. M., T. Sunaga, E. Retter, "Combined DRAM and Logic Chip for Massively Parallel Applications," accepted for 1995 IEEE Conf. on Advanced Research in VLSI, Raleigh, NC, March, 1995.

[4] Semiconductor Industries Association, Semiconductor Technology, Working Group Report, 1992. 1994 update in progress.

5.3 A Petaops is Currently Feasible by Computing in RAM

Duncan Elliott
Martin Snelgrove[†]
Christian Cojocaru[†]
Michael Stumm

Department of Electrical and Computer Engineering
University of Toronto, Canada M5S 1A4

[†]Department of Electronics
Carleton University, Ottawa, Canada K1S 5B6

5.3.1 Introduction

We focus on what would be required, *given technology now available*, to make a petaops computer system. In particular we address memory bandwidth, packaging, and power consumption.

A petaops computer system would necessarily be large, and would have to be implemented with the most reliable and highly integrated technology available. Now, and for the next few years, that is silicon CMOS memory technology running at room temperature. Of memory technologies, DRAM has higher density, lower cost per bit and moderate speed with cycle times under 100ns. SRAM, on the other hand, is more expensive, but is available with cycle times under 10ns. Ultimately, the memory size/speed requirements of the set of applications to be run will favor SRAM or DRAM.

We assume the presence of significant amounts of memory and memory traffic in a petaops architecture. And, we discount specialized architectures like "datapath-only" systolic arrays (being free of RAM) on the assumption that they would not be broadly enough applicable to make the system economic.

The memory bandwidth requirement for a petaops machine (floating point or integer) demands that the bottle-neck of the memory chip pins be circumvented. For this example, we use a 32-bit functional-unit word and 16Mb, 10ns-cycle SRAM chips with 16 data pins. If only one memory access is made in every eight operations, 2.5 million memory chips are needed to supply the 0.5 petabytes/s memory bandwidth. We optimistically assume that registers and caches local to the functional-unit chip make up the other sources and destinations. Off-chip caches don't solve the bottleneck of bringing data through memory pins. If instead, the functional units are connected directly to the memory columns and each operation requires 3 memory accesses, as few as 15 thousand chips are required to deliver the necessary 12PB/s, provided that: memory columns are 256 bits long, the IC geometries will permit connection to the columns or sense amplifiers, and that all columns can be made active during one cycle (at the expense of extra power pins). Even when an order of magnitude is lost due to compromises in the above assumptions, the bandwidth gains of putting processing in the memory are still tremendous. With the higher densities of DRAM, the gains are greater. Assuming a 256Mb, 100ns-cycle DRAM with 16 data pins, the aforementioned bandwidths require 25 million chips for off-chip access and 9200 chips for integrated processors.

Even the busing between memory and off-chip functional units is overwhelming. Half a petabyte/s transferred at a sustained clock rate of 500MHz still requires 8 million bus wires.

The memory bandwidth requirements of a petaops machine make a strong argument for integrating the processing power into the memory. These are approximate lower bounds on the number of memory chips necessary for the required memory bandwidth. We will see that differences in memory geometry and speed, as well as the efficiency of the processor, affect the amount of memory needed for a petaops.

5.3.2 Power Limitations

For an overall power consumption of 10KW, the energy consumed per operation in a petaops system would have to be 10^{-11}J. A power consumption in the tens of kilowatts is a comfortable limit for an air-cooled machine.

CMOS consumes energy CV_{DD}^2 during one cycle of charging and discharging a circuit node between ground and a power supply V_{DD}. The minimum practical value of V_{DD} for room-temperature operation appears to be about 1V [1, 2], since going below $V_{DD} = 4V_t$ doesn't improve energy per operation and where MOS threshold voltage V_t has to be greater than about 0.2V to control leakage currents that are determined by the room-temperature value $kT/q \approx 26mV$. Assuming that an average operation involves reading and changing roughly 100 bits and that circuit nodes are precharged to $V_{DD}/2$, we require

$$10^{-11}J > 100bitsC(0.5V)^2 \tag{1}$$

which limits the capacitance to an average of 400fF.

While adiabatic computing [3] attempts to reduce CV^2f power, it still has switching overheads of the same general form but with a diode voltage (around 0.7V) replacing the V_{DD} term. At present, adiabatic computing appears to consume roughly the same power as 1V CMOS, and hence it does not appear to offer a solution with present technology.

A 400fF capacitance is barely greater than the bit-line pair capacitance of a typical modern DRAM [4] which is charged for every memory access. A cache doesn't help power consumption. Even though a high hit rate reduces the number of accesses to main memory, the cache RAM charges similar capacitances on each access.

A conventional memory architecture also wastes almost all bit reads because only a small fraction of the bits read by the sense amplifiers on a given cycle are actually used. This is obviously unacceptable, since we already have a tight power-budget when assuming that bits are used with perfect efficiency. Driving signals off-chip also comes with the expense of charging many pF per wire.

We therefore claim that bits should usually be processed on-chip with the memory, and in fact very close to the sense amplifiers of the memory chip. It follows that the processors used must be compact and simple, or their sheer size will consume energy in routing signals.

Radical changes to memory architecture, such as breaking up memory arrays and introducing extra row decoders to perform independent addressing, will reduce memory density and increase the power cost of communications. The long internal memory words are too long for a uniprocessor and difficult for a MIMD multiprocessor to utilize, unless the application can benefit from processors autonomously executing their own instruction streams but performing loads and stores to the same address in local memory at the same time. The shared memory address stream suggests the use of a shared instruction stream as well. For these reasons, we use SIMD processing elements (PEs) in the memory.

Attempting to speed up the cycle time is also wrong. The best energy/operation is obtained at low V_{DD} and hence relatively slow switching. Faster cycle times would also make power supply transients worse, and we already propose to activate many more sense amplifiers at once than is typically done.

5.3.3 Computing in RAM

We have shown the feasibility of placing one-bit SIMD PEs in memory adjacent to the sense amplifiers in both SRAM and DRAM. Our first proof-of-concept design of "Computational RAM" (C•RAM) [5] was fabricated in a 1.2μm CMOS process and had 64 PEs with 128 bits of SRAM per PE. This minimalistic PE, shown in Figure 1, requires 77 transistors and fits in the width of a memory column. Together the PEs occupy 9% of the chip area. The second generation of C•RAM [6] is being fabricated in a 0.8μm BiCMOS process and has 512 PEs by 480 bits per PE, as well as some new features. A DRAM version of C•RAM has been designed (but not fabricated) with a PE for every 2Kb of memory.

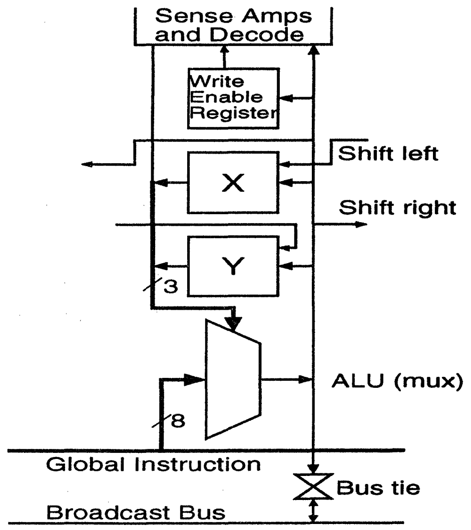


Figure 1: C•RAM Processing Element

Other architectures include a 3.8 GIPS chip from NEC [7] with 64 eight-bit processors attached to 2 MB of SRAM, a SRAM based 64-processor chip with 1-bit PEs from the Supercomputer Research Center [8], and the DRAM based "Serial Video Processor" with 1024-PEs TI [9] using one-bit processors. These chips demonstrate that computing can be done very close to the memory, and that this can be done with light-weight SIMD PEs, simple buses, and one-dimensional nearest-neighbor communication to keep size (and hence power) under control.

These simple one-bit processors can deliver between 0.1 MIPS and 1 MIPS each for 32-bit integer operations. A terabyte of SRAM based C•RAM could deliver a petaops (more memory than our lower bound). Assuming one PE per 512 bits with a 20ns read/ALU/write cycle, 32-bit additions can be performed at a rate of 6.7 petaops. An 8-bit multiply with 16-bit accumulate achieves 2.6 petaops. Yet, single precision floating point multiplies are only performed at a rate of 0.4 petaflops.

In the above examples of performance, C•RAM falls short of producing 1 bit of result for every memory write operation. Since C•RAM is register-poor, intermediate results are stored in memory. In the case of multiplication, partial sums are ferried to and from memory. If multiplies are really needed at the full rate, our processor would have to be redesigned with more registers (to avoid rereading a multiplicand and partial product for each bit of the multiplier from the relatively long and power-hungry bit-lines). Normalization support for floating-point adds would also be expensive. We have developed a variable-width processor [6] that contributes improved power and performance on multiply for roughly a factor of two increase in processor area.

5.3.4 Overall Architecture

Even though the majority of the computing power will come from the SIMD PEs in memory, we do not propose a pure SIMD machine but rather a MIMD-SIMD hybrid. Combining C•RAM, a high-performance microprocessor like the DEC Alpha [10] for every 256 MB of memory, and a scalable interconnect [11], would make a multicomputer MIMD machine with each of its processors in turn having a massively parallel SIMD machine as its memory. With a terabyte of total system memory, each of 4096 microprocessors would have 128 SRAM- (or 8 DRAM-) based C•RAM chips. The MIMD-SIMD hybrid combines the flexibility of MIMD with the economics (especially the power economics) of SIMD.

5.3.5 Conclusions

A petaops system is obviously an extremely aggressive target, but a C•RAM design that focuses on power consumption and bandwidth makes it plausible. While the technologies we propose are far from "proven", they are within the bounds of the imaginable with present fabrication processes and system engineering.

5.3.6 Acknowledgments

The authors are grateful for assistance from MOSAID Technologies, Northern Telecom Electronics, Canadian Microelectronics Corporation; and support from the Natural Sciences and Engineering Research Council of Canada and MICRONET. Address email to dunc@eecg.toronto.edu

5.3.7 References

[1] Dake Liu and Christer Svensson. Trading Speed for Low Power by Choice of Supply and Threshold Voltages. IEEE Journal of Solid-State Circuits, Vol 8, No. 1, January 1993, pp. 10–17.

[2] Anantha P. Chandrakasan, Randy Allmon, Anthony Statakos, and Robert W. Brodersen. "Design of Portable Systems". In Custom Integrated Circuits Conference, May 1994, pp. 12.1.1–12.1.8.

[3] Alex G. Dickinson and John S. Denker. "Adiabatic Dynamic Logic." In Custom Integrated Circuits Conference, May 1994, pp. 12.6.1–12.6.4.

[4] Betty Prince, "Semiconductor Memories: A Handbook of Design, Manufacturing and Application," 2d ed. Wiley, 1991.

[5] Duncan G. Elliott, W. Martin Snelgrove, and Michael Stumm. "Computational RAM: A Memory-SIMD Hybrid and its Application to DSP." In Custom Integrated Circuits Conference, Boston, MA, May 1992, pp. 30.6.1–30.6.4.

[6] Christian Cojocaru. "Computational RAM: Implementation and Bit-Parallel Architecture." Master's thesis, Carleton University, January 1995.

[7] Nobuyuki Yamashita, Tohru Kimura, Yoshihiro Fujita, Yoshiharu Aimoto, Takashi Manaba, Shin'ichiro Okazaki, Kazuyuki Nakamura, and Masakazu Yamashina. "A 3.84GIPS Integrated Memory Array Processor LSI with 64 Processing Elements and 2Mb SRAM." In International Solid-State Circuits Conference, San Francisco, February 1994, pp. 260–261.

[8] Maya Gokhale, Bill Holmes, Ken Iobst, Alan Murray, and Tom Turnbull. "A Massively Parallel Processor-in-Memory Array and its Programming Environment." Technical Report SRC-TR-92-076, Supercomputer Research Center, Institute for Defense Analyses, 17100 Science Drive, Bowie, Maryland, November 1992.

[9] Jim Childers, Peter Reinecke, and Hiroshi Miyaguchi. "SVP: A Serial Video Processor." IEEE 1990 Custom Integrated Circuits, May 1990 Conference, pp 17.3.1–17.3.4.

[10] Daniel Booberpuhl, Richard Witek, Randy Allmon, Robert Anglin, and Sharon Britton. "A 200MHz 64b Dual-Issue CMOS Microprocessor." In International Solid-State Circuits Conference, San Francisco, February 1992, pp 106–107.

[11] Zvonko G. Vranesic, Michael Stumm, David M. Lewis, and Ron White. "Hector: A Hierarchically Structured Shared-Memory Multiprocessor Computer," Vol 24, No.1, January 1991, pp.72–79.

5.4 Design of a Masssively Parallel Computer Using Bit Serial Processing Elements

John E. Dorband
Goddard Space Flight Center
Greenbelt, MD 20171

Maurice F. Aburdene
Kamal S. Khouri
Jason E. Piatt
Jianqing Zheng
Department of Electrical Engineering
Bucknell University
Lewisburg, PA 17837

5.4.1 Introduction

Recent trends for the design of massively parallel computers have shifted from emphasis on the design of single-instruction, multiple-data (SIMD) machines to multiple instruction-multiple data (MIMD) machines. However, it is our contention that the trend is based on inappropriately drawn conclusions. Because of the early rush to build SIMD computers, inadequate attention was devoted to exploring the full potential of these architectures. Current MIMD machines that are based on commercially available processors lack the scalability originally intended for massively parallel computers. These computers employ processors on the order of a hundred or a thousand rather than a million, and yet consume large amounts of power and space.

5.4.2 Massively Parallel SIMD Architecture

The objective of our project is to design a massively parallel SIMD architecture with greater than a million processors. A bit serial processor specifically designed for a SIMD architecture will be utilized. The processing element [1] consists of a bit serial arithmetic and logic unit (ALU), a distributed bit serial RAM, and a two-dimensional router. The ALU is further divided into computational logic, three registers (an accumulator register, a carry register, and a mask register), and transmission gates, as shown in Figure 1.

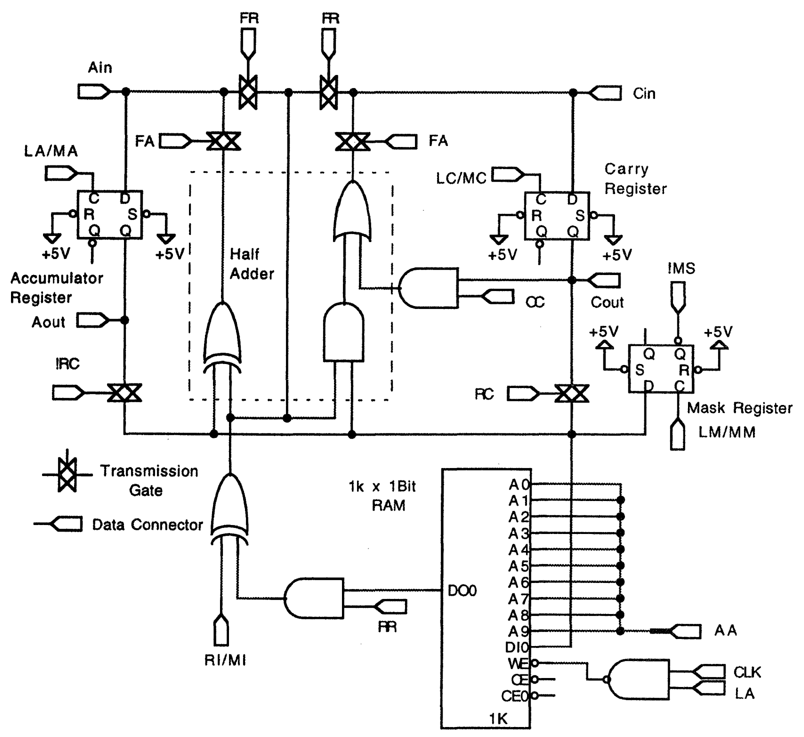


Figure 1: Registers and ALU of PE

In this design, a half adder is the fundamental component of the ALU. The results of such operations may be stored in either the accumulator or carry register. The use of a shift register may considerably reduce the execution time for functions such as multiplication. However, to reduce the size of the processing element by at least a factor of five, a shift register is not used. The mask register allows certain processors to perform a given operation while the others are "masked" and therefore do not execute that operation.

Designed for CMOS implementation, the processor uses transmission gates to guarantee that there are no conflicting signals, and that no combination of control signals will produce a short from the power supply to the ground of any device.

The processing element may receive and transmit data from three levels: (1) local RAM and ALU, (2) router network, and (3) global operation network. Each processing element may only communicate with one level at a time. The first level is used to perform operations on data stored in the bit-serial RAM or any one of the registers.

The router network allows processors to transmit and receive information from each other. Any data received may be stored locally and operated on. The router is a two-dimensional mesh as shown in Figure 2. This means that each element may communicate directly with any of its four closest neighbors—north, south, east or west.

Since two registers are available to store data, we may use both for communication purposes. The accumulator register is chosen to accommodate east-west data transfer, while the carry register is used for north-south data transfer. This scheme

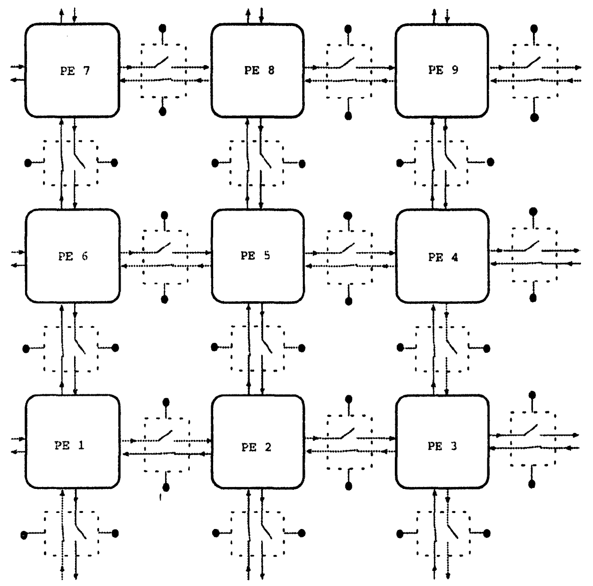


Figure 2: 3×3 Toroidal 2-D Mesh of Bit-Serial PEs

optimizes the use of a two-dimensional router, and if need be, can allow for transmitting two sets of data at a time. For example, one command may be "send carry register contents north" or it may be "send carry register contents north and accumulator register contents west."

Each processing element is connected to two routing switches—one switch for north-south communication and another for east-west communication. The switchs consists of four transmission gates in a square formation, where opposing gates are controlled by a common signal. The corners of each switch are connected to four data transmission lines: input of a register, output of a register, and the switches from the two adjacent processors. The control signals dictate which transmission gates are on and hence which direction the data is transmitted. In addition, a series of external data lines have been added to allow for external I/O. When the external I/O is activated, the router network remains on, but the toroidal connect is removed to allow data to enter from only one side of the mesh.

Often it is necessary to inquire about the status of all the processors. These types of operations are described as global operations, and require the use of an independent network. This is the third level network for data transmission that consists of a linear mesh of OR gates. The register contents of each PE are ORed and the result of the operation placed in each processor.

5.4.3 Summary

The group has successfully simulated a 3×3 toroidal mesh of processing elements using circuit design software. The simulation included all local operations. In addition, the router and global networks have been designed, and we are currently in the process of simulating them. Plans are to simulate a larger network and begin to develop a VLSI prototype.

5.4.4 Acknowledgment

Partial support for this work was provided by NASA Grant NAG 5-2509.

5.4.5 References

[1] Dorband, John E., "Processing Element Description", NASA Goddard Space Flight Center, 1989.

5.5 Non von Neumann Instruction Set Architecture as an Enabling Technology in Grand Challenge Systems

Justin S. M. Porter
University of British Columbia
Department of Electrical Engineering

5.5.1 Introduction

To attain significant performance gains, system bottlenecks need to be removed. High-end processors currently achieve large throughput by superscalar and superpipelined architectures, register set layouts and caching systems. These well established tools all sidestep the von Neumann bottleneck of moving data between processors and memories. This paper examines performance details of a distributed-instruction set architecture as an enabling technology in grand challenge systems. Performance gains will be shown from the movement of decoding and fetching units from the processor to memory systems.

In order to obtain the performance goal of petaflops speed, identified as a crucial factor by working groups in grand challenge systems [1], the fundamentals of computational style will have to be re-examined. Performance gains must be realized for this class of computing power in a manner that will not compromise efforts in software development or manufacturing. Previous work in the area of active memory have been pursued [8] in fields related to database searches, network routing, and storage management but this technology has yet to be realized for computation-intensive scientific applications.

The von Neumann architecture has been adhered to by a majority of supercomputer developers, the exceptions being dataflow [5] and symbolic language machines. Traditionally, operational cycles have continued to be composed of fetch, decode, execute and writeback stages. Both vector supercomputers and massively parallel systems have been built using this paradigm. Each approach has, at appropriate times, been the underlying methodology by which the fastest computer systems in the world have been built. Expansion of computational power by orders of magnitude must free itself from technology dependent factors which have been utilized in coarse-grained parallel vector processors such as the Cray YMP [2] and NEC SX-2 [3] or on the dependency of parallel problem spaces which massively parallel systems such as the Connection Machine [4] have been constructed around.

The use of cutting edge physical technology such as gallium arsenide chips, high mobility heterojunctions, optical devices, and superconductors may not be incorporated into grand challenge systems on the basis that there are factors such as expense, manufacturing difficulty, and maintenance difficulties. Software advances in highly parallel decomposition via graph techniques, loop decomposition techniques and vectorizing compilers[6] are important advances, may not adequately address the large range of problems presented as grand challenges.

The distributed instruction set architecture embodied in this paper starts with the movement of fetch and decode units from processor elements to the memory system. Additionally, memory related instructions such as loading and moving data have been implemented inside the memory system in order to save bus cycles. In light of current trends in multithreaded vector processing, architectural tools such as caching, pipelining and register sets may only continue to yield performance gains until the end of the century. Distributed-instruction set architecture does not rely on these architectural features.

In order to demonstrate the feasibility of distributed instruction set architectures, a machine simulation was constructed and used to perform the Fast Fourier Transform benchmark to determine execution lengths. The remainder of this paper discusses features of the instruction set used, simulator results, and future directions.

5.5.2 Distributed Instruction Set Architecture

For decades, computer designers have provided extra functionality and performance in systems by use of co-processors for math and graphical functions. To take this concept to an extreme form, this type of distribution of functionality can be applied to the basic operating cycle in the von Neumann architecture. To show the feasibility of this type of architecture for scientific applications, a machine simulator was constructed. Two instruction sets were used in the simulations: one representing a traditional von Neumann architecture, and the other representing a distributed-instruction set. The instruction formats for both simulators are presented in Table 1 and Table 2. Bracketed values are indirect addressing, literal indicates immediate mode addressing.

Both instruction sets were designed with basic tenets of reduced-instruction set computers in mind. The fundamental difference between the two sets is that the distributed case takes into account a savings of one cycle for memory instructions. This derives from memory instructions being executed entirely within the memory system without involvement of a processor element. The execution delay introduced by the fetch, decode, execute, writeback cycle for each instruction set are

Table 1: Sample Instruction Set for von Neumann System

Instruction	**Format**	**Cycles to Execute**
Integer Add	**INT ADD** (src1)(src2)(dst)	3
Integer Multiply	**INT MULT** (src1) (src2) (dst)	4
Integer Add	**INT ADD** (src1) (src2) (dst)	3
Integer Subtract	**INT SUB** (src1) (src2) (dst)	3
Integer Divide	**INT DIV** (src1) (src2) (dst)	4
Floating Point Add	**FP ADD** (src1) (src2) (dst)	3
Floating Point Multiply	**FP MULT** (src1) (src2) (dst)	4
Logical And	**AND** (src1) (src2) (dst)	1
Logical Or	**OR** (src1) (src2) (dst)	1
Logical Not	**NOT** (src) (dst)	1
Load	**LOAD** (src) (dst)	2
Load Immediate	**LOADI** (dst) value	2
Move	**MOVE** (src) (dst)	2
Branch Equal to	**BREQ** (src1) (src2) address	2
Branch Greater than	**BRGT** (src1) (src2) address	2
Branch Less than	**BRLT** (src1) (src2) address	2
Jump	**JMP** address	2

Table 2: Sample Instruction Set for Distributed System

Instruction	**Format**	**Cycles to Execute**
Integer Add	**INT ADD** (src1)(src2)(dst)	3
Integer Multiply	**INT MULT** (src1) (src2) (dst)	4
Integer Add	**INT ADD** (src1) (src2) (dst)	3
Integer Subtract	**INT SUB** (src1) (src2) (dst)	3
Integer Divide	**INT DIV** (src1) (src2) (dst)	4
Floating Point Add	**FP ADD** (src1) (src2) (dst)	3
Floating Point Multiply	**FP MULT** (src1) (src2) (dst)	4
Logical And	**AND** (src1) (src2) (dst)	1
Logical Or	**OR** (src1) (src2) (dst)	1
Logical Not	**NOT** (src) (dst)	1
Load	**LOAD** (src) (dst)	1
Load Immediate	**LOADI** (dst) value	1
Move	**MOVE** (src) (dst)	1
Branch Equal to	**BREQ** (src1) (src2) address	2
Branch Greater than	**BRGT** (src1) (src2) address	2
Branch Less than	**BRLT** (src1) (src2) address	2
Jump	**JMP** address	2

summarized inTable 3.

Table 3: Processor Cycle Delays

von Neumann System	Delay	Distributed System	Delay
Fetch	1	Fetch	0
hline Decode	1	Decode	0
Execute	As Per Table 1	Execute	As Per Table 2
Write	1	Write	1

Again, the distributed system has an advantage since the fetch and decode operations can be accomplished without usage of the system bus. Even in the most powerful computers, the majority of system bottlenecks are currently attributed to memory system design and memory-processor interaction.

5.5.3 Simulation Results

The simulator was designed to model a dual processor system executing in SIMD fashion using the above instruction set parameters. The benchmark used was the Cooley-Tukey algorithm for the Fast Fourier Transform which is a representative scientific benchmark. The results of simulated executions are presented in Table 4.

Moving the fetch, decode, and load/move instructions to memory yielded a 16.6% savings in execution time, allowing for a speedup factor of 1.19. Although one benchmark does not completely prove the case that distributed-instruction sets would yield higher throughput, the distributed architectures performance on a basic test such as this suggests that further simulation studies would have similar findings.

The simulator did not incorporate models for distributed memory access, dynamic instruction scheduling, interprocessor data dependencies, or usage of pipelines or caches. It could be argued that these results are inaccurate because pipelining is insensitive to total length of operation execution, however the large amount of context information captured in pipelines and caches presents a large problem when operating in a multithreaded environment. Current superpipelined designs usually

Table 4: Simulation Results

Instruction Set	Execution Time (cycles)
von Neumann	9657560
Distributed	7731990

incorporate a dozen pipeline sections on average, thus presenting a large performance bottleneck on context-switching activities. The impact of these factors on system performance is beyond the scope of this paper.

The major advantage of distributed instruction sets would be that the benchmark code for the distributed system can run unchanged from that of the von Neumann system. No recoding of software would be required and there would not be a burden placed on compilers to determine optimal code transformations based on this change in architecture[6].

One obstacle to implementing distributed instruction set machines would be in the wafer fabrication process. Different fabrication techniques are used in the manufacture of both high performance processors and memory chips, and the integration of techniques to create memory chips with instruction units on them will require additional work.

5.5.4 Conclusions

In order to facilitate an order of magnitude increase in performance in cutting edge supercomputers, a change in the fundamentals of execution style has shown to give a 16.6% savings in execution length. In exposing the distributed nature of basic instructions, techniques such as multithreading and vector processing[7] may be combined in the future for higher performance gains.

5.5.5 References

[1] Seigel, H.J. and Abraham, S. “Summary of the Report of the NSF-Sponsored Purdue Workshop on Grand Challenges in Computer Architecture for the Support of High Performance Computing”, 4th Symposium on Frontiers of Massively Parallel Processing, 1992, p. 76.

[2] Vajapeyam, S., Sohi, G. And Hsu, W.C. “An Empirical Study of the CRAY Y-MP Processor using the PERFECT Club Benchmarks”, 18th International Symposium on Computer Architecture, p. 170.

[3] Fatoohi, R. “Vector Performance Analysis of the NEC SX-2”, International Conference on Supercomputing, 1990, p. 389.

[4] Tucker, L.W. and Robertson, G.G., “Architecture and Applications of the Connection Machine”, IEEE Computer, Vol. 21, No. 8, August 1988, pp. 26–38.

[5] Yuba, T. et al., “Dataflow Computer Development in Japan”, International Conference on Supercomputing, 1990, p. 140.

[6] Tsuda and Kunieda, Y., "V-Pascal: An Automatic Vectorizing Compiler for Pascal with No Language Extensions", The Journal of Supercomputing, Vol 4, No. 3, September 1990, p. 251.

[7] Chiueh, T. "Multi-Threaded Vectorization", 18th International Symposium on Computer Architecture, p. 352.

[8] Asthana, A., Cravatts, M. and Kryzanowski, P. "An Experimental Active Memroy Based I/O Subsystem", Computer Architecture News, ACM SIGARCH, Vol 22, No. 4, September 1994, p. 29.

5.6 Taming Massive Parallelism: The Prospects of Opto-Electronic CRCW-Shared Memory

Paul Lukowicz
Walter F. Tichy
Fakultät für Informatik
Universität Karlsruhe
D–76128 Karlsruhe, Germany
lukowicz@ira.uka.de, tichy@ira.uka.de

5.6.1 Introduction

Much would be gained on the way beyond the teraflop and towards the petaflop if massively parallel computers (MPCs) could be utilized more efficiently. Currently most non-trivially parallel applications achieve only a fraction of the theoretical peak performance once the number of processors goes into thousands. Much better results could be obtained if it were possible to build a MPC with a uniform access time concurrent-read, concurrent-write-shared memory (CRCW-SM). Unfortunately, due to fundamental physical limitations of conventional electronic memory all such attempts have so far yielded unsatisfactory results. This has motivated us to look at a novel, promising approach: using the natural parallelism provided by optical data storage to build an opto-electronic CRCW-SM (OCRCW-SM) suitable for MPCS.

5.6.2 Related Work

So far, research on the use of optical technology in parallel computers has concentrated on the improvement of interconnection networks [10, 3]. Optically, extremely tightly (or even fully) connected networks were suggested as a potential platform for the efficient simulation of shared memory (e.g., [11]). We go a step further by proposing a direct implementation of the full functionality of a CRCW-SM and its integration into the memory hierarchy of a MPC. Our work is based on the research in the area of magnetooptic memory [5], holographic optical memory [4], and ferroelectric liquid crystal memory [1].

5.6.3 OCRCW-SM

A detailed description of our proposed system and elaborate performance estimates can be found in [8] and [7]. Here we give a compact overview of the overall structure and the performance.

5.6.4 Principles

The idea behind the OCRCW-SM is that information stored using optically controlled, variable light absorption of memory pixels can be accessed concurrently by distinct light beams. Each bit is stored in a light modulator (LM) that can be either transparent (1) or light absorbing (0). It is read out by illuminating it with a beam of light (readout beam) and using a light detector to check whether it has been absorbed or transmitted (Figure 1d). To allow optical, parallel write access an electro-optic modulator is constructed by combining an electrically controlled LM with two photodetectors (PDs). When illuminated with a light beam (write beam) one PD (set PD) causes the modulator to become transparent (bit=1) , while the other one (clear PD) makes it absorbing (bit=0). For a parallel write operation with some processors trying to set the bit and others trying to clear it, the set and clear PDs can be wired to accept a majority decision.

To build an M-bit OCRCW-SM we need a plane of M LMs, for data storage. Each of the P accessing processors must have a device to direct a light beam towards the desired memory location (beam deflector) placed in front of the MP and a detector device placed behind it. An optical system must be devised that makes sure that all beams emitted by a particular processor find their way to its detector device.

Major requirements on any concrete OCRCW-SM design are the limitation of the overall size of the memory plane and the size and complexity of the beam deflector (the size of the beam deflectors must be much smaller then the MP to allow for a large number of accessing processors). Furthermore it should be possible to implement the device in the form of integrated chip-like components that can easily be mass produced and can operate reliably under every day conditions.

5.6.5 Memory Architecture

To limit the size and complexity of the beam deflectors the memory is divided into$p$ pages and addressed in two stages. In the first optical stage a processor projects the selected page on an array of light detectors using the beam deflection mechanism (Figure 1b). Thus, the beam deflector need only resolve p points. It can be realized by p miniature laser diodes, each equipped with appropriate micro-optics. In the second stage the desired bit is retrieved from the detector array by a matrix addressing mechanism. In a similar two-stage addressing scheme, an appropriately set sample page is projected on the memory page containing a given memory location to perform a write operation (Figure 1c). The sample page has two electrically controlled LMs for each bit of a memory page. One LM corresponds to the set PD (set LM), the other one to the clear PD (clear LM). To write a selected bit of page the LMs of the sample page corresponding to the other bits are all off (absorbing). Of the LMs corresponding to the selected bit either the set LM or the clear LM is on, depending on whether the bit is to be cleared or set (Figure 1e).

To limit the overall size of the MP we divide it into m modules $M_{mod} = \frac{M}{m}$ bits each. The modules are connected to the processors by a tree of height h of $P \times D$ multiplexers with $h = log_m(D)$ (Figure 1a). The memory modules are located at the leaves and have P IO channels, one for each PE. Thus each PE has an independent connection to each memory module.

Each of the $n_{mult} = \sum_{i=0}^{h-1} D^i \leq D^h = m$ multiplexers maps the input of each PE onto one of D alternative outputs. This is essentially a simplified version of the functionality contained in a memory module, where the input of each PE has to be directed to one of p pages with $p > D$. Thus, at worst the number of components in the system is doubled. On the other hand the overall complexity is greatly reduced since we are now dealing with a system modularly built of identical, constant-size components.

5.6.6 Technological Feasibility

The basic components of the OCRCW-SM are light sources, light modulators, light detectors and passive optical components (lenses, holograms etc). Such devices allowing miniaturization and integration in semiconductor technology and operating at GHz frequencies have all been demonstrated in laboratory experiments (e.g., [9]). They were used to implement complex systems in optical bench experiments [2]. The integration of simple systems in compact modules has also been studied and reported [6].

5.6.7 Performance

There is no relevant theoretical limit to the number of processors that can access the OCRCW-SM. For practical purposes the concurrency level is limited by the system size, the optical cross-talk and energy considerations. Up to the order of magnitude of 10^5 processors can be shown to be a realistic estimate. The information density depends on the resolution of the optical system and the size the LMs. Both approach the wave length of the light meaning that densities of $\frac{10^6 bits}{mm^2}$ and beyond could be reached. The capacity of our modular design is limited only in as far as the latency is determined by the physical dimensions of the system. Taking into account the performance of the opto-electronic components, systems with GBytes capacity could be built with a latency $\leq$1ns.

5.6.8 System Architecture

The speed (due to physical dimensions), the feasibility (a reduced density device is easier to build) and the cost of an OCRCW-SM system depend on the capacity of the SM. At the same time, experience shows that most applications require only a fraction of storage to be sharable. It thus makes sense to integrate the OCRCW-SM into the memory hierarchy instead of using it exclusively. This also opens the way for further performance improvement through local caching.

We are currently working on theoretical estimates and simulations to determine the optimal ration of OCRCW-SM to conventional distributed memory and the best way to integrate it. The most promising strategies are (a) software distinction between private and shared memory (efficient and simple, but burdensome to the programmer), (b) using a small amount of OCRCW-SM for special system purposes (e.g., maintaining global cache coherence), (c) using OCRCW-SM as a fast-shared cache for the private memories (desirable, but difficult to implement), and (d) connecting the OCRCW-SM to a large conventional memory that is used as a "swap" device.

5.6.9 Conclusion and Future Work

We have shown that recent rapid progress in opto-electronic technology offers the possibility of building a true scalable CRCW-SM. We have presented a design that could be implemented in integrated semiconductor technology with memory density, latency and access speed equal to or better than what is possible with state-of-the-art conventional memory. Such a device would allow us to utilize massive parallelism of up to 10^5 processors with a high degree of efficiency.

We are using arrays of S-SEED MQW modulators and VCSEL laser chips to build an optical bench proof-of-principle model. Furthermore, experiments to test the scalabibty and the performance limits of different critical system components are planned. We are also working on simulations comparing the performance of machines using OCRCW-SM varying in size and performance to more conventional parallel machines.

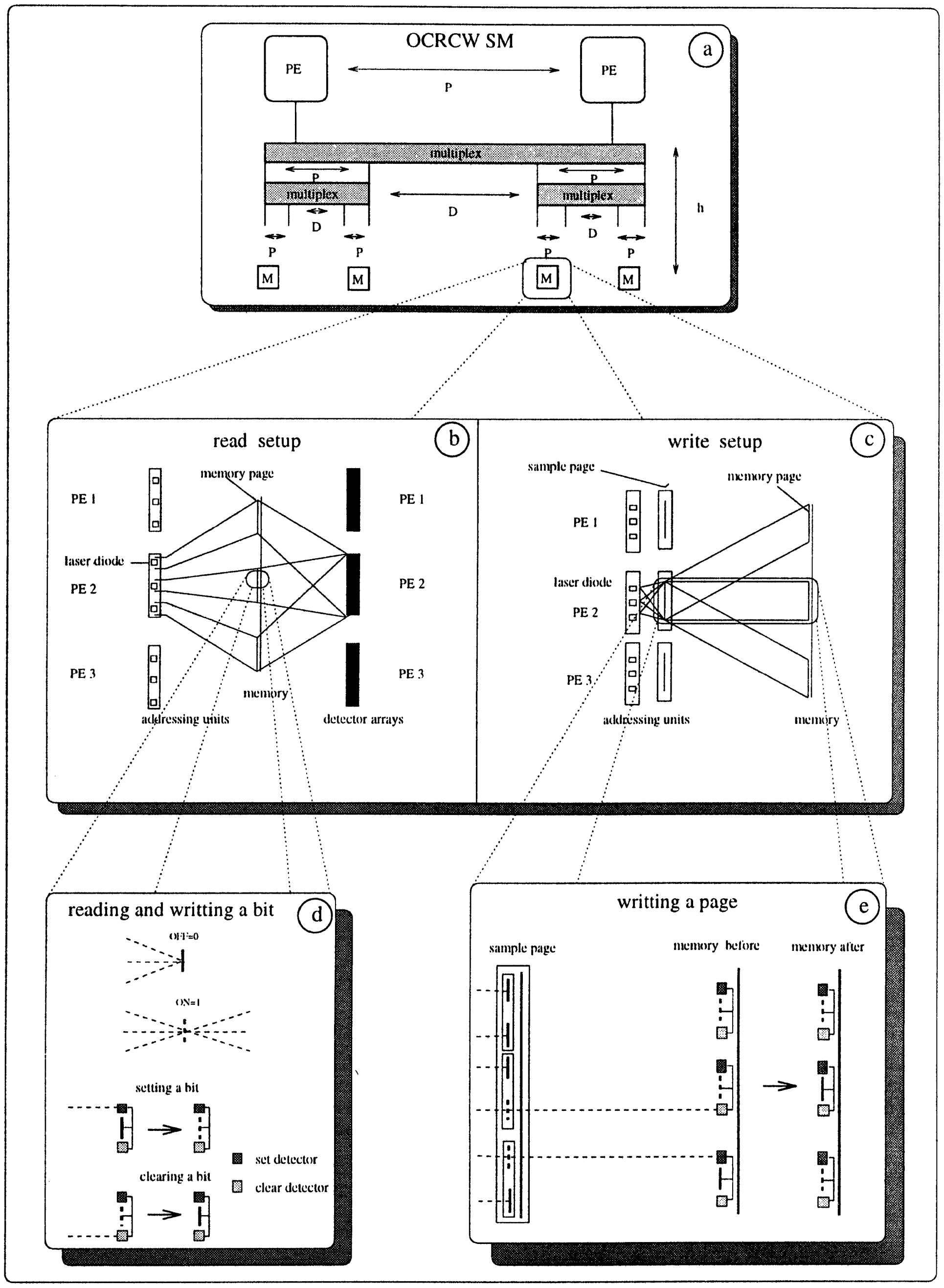


Figure 1: Architecture of the Proposed Opto-Electronic Shared Memory System

5.6.10 References

[1] J. L. de Bougrenet del la Tocnaye and J.R. Brocklehurst. "Parallel Access Read/Write Memory Using an Optically Addressed Ferroelectric Spatial Light Modulator." Applied Optics, Vol. 30, No. 2, January 1991, pp. 179–180.

[2] F. B. McCormic et al. "Five-Stage Free-Space Optical Switching Network With Field-Effect Transistor Delf-Electro-Optic-Effect-Device Smart-Pixiel Arrays." Applied Optics, Vol. , No. 8, March 1994, p. 1601.

[3] E. E. E. Friet,man, W. van Nifterick, L. Dekker, and T. J. M. Jongeling. "Parallel Optical Interconnects: Implementation of Optoelectronics in Multiprocessor Architectures." Applied Optics, Vol. 29, No.8, March 1990, pp. 1160–1177.

[4] J. Glanz. "Will Holograms Tame the Data Glut." Science, Vol. 265, No. 5, August 1994, p. 736.

[5] B. Hill. Optical Memory Systems. Digital Memory and Storage. Vieweg, 1978.

[6] J. Jahns. "Planar Packaging of Free-Space Optical Interconnections," Proceedings. of the IEEE, Vol. 82, No. 11, 1994.

[7] P. Lukowicz and W. F. Tichy. "On the Feasibility of a Scalable Opto-Electronic CRCW Shared Memory," Submission to the International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP-95).

[8] P. Lukowicz and W. F. Tichy. "Designing a Scalable Optoelectronic CRCW PRAM," Proceedings of the Lessach Workshop on Parallel Processing, University of Clausthal TR series, September 1994.

[9] A.D. McAulay. "Optical Computer Architectures," John Wiley & Sons, Inc., 1991.

[10] T. M. Pinkston. "The GLORI Strategy for Multiprocessors: Integrating Optics into the Interconnect Architecture," Technical Report CSL-TR-92-552, Stanford University, Department of Computer Science, December 1992.

[11] C. Waterson and K. Jenkins. "Shared-Memory Optical/Electronic Computer Architecture and Control," Applied Optics, Vol. 33, No. 8, March 1994, p. 1559.

5.7 Lightning: A Scalable Dynamically Reconfigurable Hierarchical WDM Network for High Performance Clustering

Patrick W. Dowd
Department of Electrical and Computer Engineering
State University of New York at Buffalo
201 Bell Hall
Buffalo, NY 14260
dowd@eng.buffalo.edu.

5.7.1 Introduction

This paper describes a hierarchical optical structure for high-performance processor clustering. The architecture is based on wavelength division multiplexing (WDM) which enables multiple multi-access channels to be realized on a single optical fiber. The objective of the hierarchical architecture is to achieve scalability yet avoid the requirement of multiple wavelength-tunable devices per node. In addition, single-hop, all-optical communication is achieved: a packet remains in the optical form from source to destination and does not require opto-electronic conversion or intermediate routing. The wavelength multiplexed hierarchical structure features wavelength channel re-use at each level, allowing scalability to very large system sizes. It partitions the traffic between different levels of the hierarchy without electronic intervention in a combination of wavelength- and space-division multiplexing. A significant advantage of the proposed structure is its ability to dynamically vary the bandwidth provided to different levels of the hierarchy.

5.7.2 Scalable Technique for Clustering

The objective is to develop a scalable technique for clustering: a strategy that is effective (in performance and price/performance) for high performance supercomputer system-level interconnection. The interconnection strategy needs to be flexible to adapt to cost constraints at the low-end and high performance requirements at the high-end. This paper defines the general architecture and then defines more specifically an experimental testbed currently under construction known as "Lightning" that is targeted to supercomputer interconnection. This is a group project including researchers from the State University of New York at Buffalo (network and system architecture), University of Maryland at College Park (WDM devices), University of Maryland at Baltimore County (WDM devices), the Supercomputing Research Center (memory interface and operating system), and the Laboratory for Physical Sciences (optical materials processing).

The goal of the experimental testbed being developed is to support a (weakly coherent) distributed shared memory environment between the interconnected processors. This is accomplished through a combination of the network architecture, operating system, and network/memory interface design. The design objective is not just to provide a high-bandwidth network, but to deliver that bandwidth to the application rather than wasting it in overhead at the operating system level. Our approach is to achieve a tighter integration in the optical, electronic and software technologies within the system. For example, the design of our memory interface is being over-seen by the system architect, an operating systems designer, the primary application developer, and the designers of the optical components. This collaboration has resulted in a 0-copy memory/network interface card where an incoming memory block is injected directly to its final location in memory without intermediate buffer copies.

5.7.3 Lightning Network Architecture

Lightning network architecture uses wavelength-, space-, and time-division multiplexing to achieve the communication requirements of the system. In particular, wavelength re-use is achieved at each level of the hierarchy to both magnify the usefulness of WDM channels and reduce the number of required WDM channels. A major objective of the architecture was to relax the constraints placed on the optical devices—strong constraints usually translate into high cost. In particular, one objective is to reduce the number of required WDM channels to preserve practical feasibility.

This architecture has an advantage in that bandwidth can be dynamically reallocated throughout the system to adapt to shifts in traffic patterns. The bandwidth assigned to different levels of the hierarchical system can be dynamically increased or decreased, based on the reference patterns and file activity of the system. However, in comparison to traditional reconfigurable architectures, the reconfiguration of the system is automatic and hidden from the user. The user does not have to logically map an application to a specific topology or inform the operating system at compile- or run-time of its intended communication patterns. Note that this system is not designed to reconfigure on an instruction-by-instruction basis, due to the geographically distributed design constraint, and reconfiguration is viewed as more a process-by-process level. Lightning senses the traffic patterns inherent to an application and adapts itself accordingly. This function is hidden from the user and the operating system and avoids placing the burden of understanding the specifics of the system on the programmer. The programmer views the system as a pool of processors and is not involved with process placement.

5.7.4 Summary

Lightning accomplishes the dynamical reconfiguration capability by monitoring the traffic intensities on channels at each level of the hierarchy. When an imbalance in traffic intensities is detected between two levels of the hierarchy, the system reconfigures to balance the traffic by pushing channels either up or down the hierarchy. Note that the reconfiguration is completely decentralized for fault tolerant reasons—there is no single watchdog monitoring and all nodes share in the responsibility of monitoring and initiating reconfiguration.

5.7.5 Acknowledgements

This work is supported by the Laboratory for Physical Sciences, College Park, Maryland.

5.8 PETAflops: PErhaps Take A Futuristic Look at Optical Processing Systems

Chunming Qiao
Department of Electrical and Computer Engineering
SUNY at Buffalo, Amherst, NY 14260
qiao@photon.eng.buffalo.edu

5.8.1 Introduction

Many studies have already been carried out to compare optical and electronic interconnects for high-speed digital systems [1, 2]. It has been found that based on power consumption, speed and integration density, the optical interconnects are more suitable than electronic counterparts for extremely high bandwidth (e.g. above GHz) systems [3]. In addition, the current VLSI technology is quickly approaching the physical limits due to thermal and/or quantum effects [4, 5].

Here, we consider a system capable of petaflops-range performance, from the viewpoint of interconnection networks. One possible model is a massively parallel machine and the other is a distributed cluster. We feel that free-space optical and fiber-optic interconnects are promising for massively parallel and distributed processing systems, respectively.

5.8.2 Free-space Optics for Massively Parallel Machines

A massively parallel machine would be suitable for relatively fine-grained data-parallel applications such as image processing or finite-element analysis. We wish to limit the number of nodes needed in order to achieve a reasonable connectivity. Limiting the number of nodes, however, implies that each node will have to be very powerful. We assume a system having 1 million nodes (10^6), each is capable of one gigaflops.

With electronic interconnects, no matter what logical topologies is used, the physical layout is usually two-dimensional: that is, processing elements and interconnects are on the same plane. A typical system is a 2-D mesh shown in Figure 1(a).

With free-space optic interconnects, it is possible to use the third dimension to interconnect two planes of processing elements as shown in Figure 1(b). However, the size of such a system is limited by the aberration and diffraction of the lens array

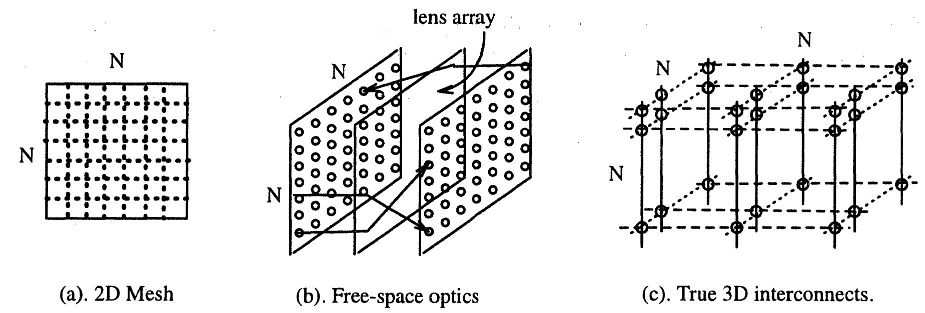
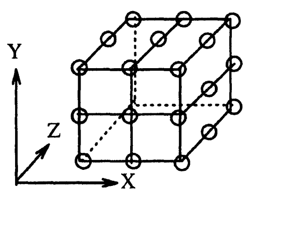


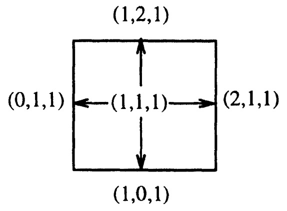
Figure 1: Possible Topologies

and the optical power detection limit of the detectors. In fact, studies [6] showed that the product of channel bandwidth (in Mbits/s) and system size is limited to about 5×10^5. Using extreme technologies of free-space optics, one may implement the interconnects in 3 or more dimensions as shown in Figure 1(c). It is important for such a system to have a fixed nodal degree and a localized connection pattern while maintaining a small diameter.

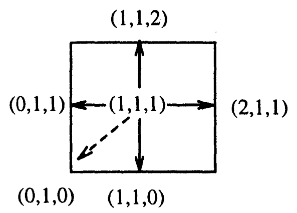
Here we present a futuristic topology which we call Multi-Plane Interconnected Cube (MPIC). To explain the interconnection pattern, we examine a 3-D block of 27 nodes, each represented in a triplet according to their x, y and z coordinates, as shown in Fig. 2(a). All nodes in an MPIC have the same fixed degree of 14 and connectivity. Figures 2(b) through (e) show the interconnection pattern from the viewpoint of the center node of the block, (1, 1, 1). It is interesting to observe that the diameter of an $N \times N \times N$ MPIC is $N - 1$ if N is odd and N otherwise, even when diagonal links such as the one shown in the dotted line in Figure 2(c) do not exist. Including those diagonal links would increase the nodal degree from 14 to 26, only to improve the diameter by 1 when N is even. In addition, as illustrated in Figure 2 (f), the minimal angle separation of any two light beams would be decreased from 55^o to 35^o. Many issues related to the MPIC topology such as routing, electronic and/or optical implementations, fault-tolerance and comparative performance study are still being actively investigated.



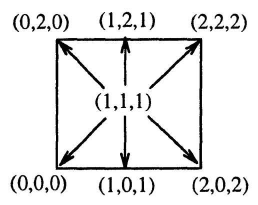
(a). A block of 27 node in an MPIC



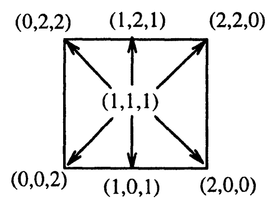
(b).Front view (Z=1)



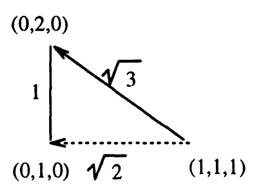
(c). Top view (Y=1)



(d). Diagonal view (FR)



(e). Diagonal View (FL)



(f). Minimal angle separation

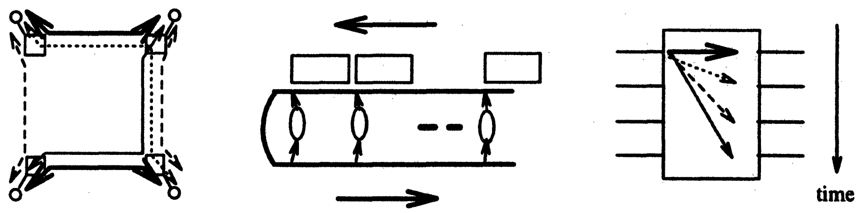
Figure 2: Interconnection Patterns in an MPIC

5.8.3 Fiber-optic Interconnects for Parallel and Distributed Systems

One may envision a distributed system of moderate size (about 1K nodes), in which each node is capable of 1 teraflops. Such a system would be suitable for a large-grain parallel and/or distributed computations. Issues related to the reduction and tolerance of communication latency are of major concern in such a system, but novel solutions using optical technology are being sought.

- **Increase connectivity.**The network diameter can be reduced by increasing its connectivities with both time and wavelength multiplexing. Multiple virtual channels can be created on a single physical link, so that the effective network connectivity is increased. Figure 3 (a) shows a ring of four nodes which is configured to a completely-connected network using four wavelengths.

- **Tolerate propagation latency**. When two communicating nodes are physically far apart, time-multiplexing techniques [7, 8] that can effectively achieve message pipelining on a single fiber [9] can be applied to tolerate the long latency. Figure 3 (b) shows that up to n packets can be transmitted and received within an end-to-end delay.



(a). WDM in a ring (b). Message pipelining (c). Reconfiguration in TDM

Figure 3: Fiber-optics in Distributed Computing

- **Eliminate intermediate O/E conversions**: Whenever feasible, intermediate O/E conversions should be eliminated Not only can this avoid delay, it also can make the communication network bit-rate transparent. Single-hop networks can be reconfigured in time-division multiplexed (TDM) fashion [10, 8] as in Figure 3 (c) to eliminate or amortize control. Multi-hop networks can either establish time-slot continuous connections [12], or use all-optical time-slot interchangers. Similar approaches have been studied in WDM systems.

Finally, we note that regardless of whether petaflops is feasible, or whether optical computing can replace electronic processing, we believe that optical interconnects will be the means in the future for high speed and high bandwidth communications. In the past, optical technology has received enormous attention from the telecommunication community but has been largely ignored by the computer community. Currently, fiber-optic technology is already a driving force to high performance communication and computing in local area network, although many issues remain open. It is hoped that more and more people in the computer community will embrace this new technology, which is going to play important roles in high performance parallel and distributed systems.

5.8.4 References

[1] D. Hartman, "Digital High Speed Interconnects: A Study Of the Optical Alternative," Optical Engineering, Vol. 25, Oct. 1986, pp. 1086–1102.

[2] P. Haugen, S. Rychovsky, and A. Husain, "Optical Interconnects for High Speed Computing," Optical Engineering, Vol. 25, Oct. 1986, pp. 1076–1984.

[3] M. Feldman, S. Esener, C. Guest, and S. Lee, "Comparison Between Optical and Electrical Interconnects Based on Power and Speed Considerations," Applied Optics, Vol. 27, May 1988, pp. 1742–1751.

[4] R. Keyes, "Physical Limits in Digital Electronics," IEEE Proceedings, Vol. 63, May 1975, pp. 740–760.

[5] Gooeman, F. Leonberger, S. Kung, and R. Athale, "Optical Interconnections for VLSI Systems," IEEE Proceedings, Vol. 72, 1984, pp. 850–866.

[6] T. Sakano, K. Noguchi, and T. Matsumoto, "Optical Limits for Spatial Interconenction Networks Using 2-D Optical Array Devices," Applied Optics, Vol. 29, Mar. 1990, pp. 1094–1100.

[7] H. Jordan, "Time Multiplexed Optical Computers," in Proceedings of Supercomputing, Nov. 1991, pp. 370–378.

[9] C. Qiao and R. Melhem, "Time-Division Optical Communications in Multiprocessor Arrays," IEEE Transactions on Computers, Vol. 42, May 1993, pp. 577–590.

[10] R. Melhem, D. Chiarulli, and S. Levitan, "Space Multiplexing of Waveguides in Optically Interconnected Multiprocessor Systems," The Computer Journal, Vol. 32, No. 4, 1989, pp. 362–369.

[11] R. Thompson, "The Dilated Slipped Banyan Switching Network Architecture for Use in an All-Optical Local Area Network," IEEE Journal of Lightwave Technology, Vol. 9, Dec. 1991, pp. 1780–1787.

[12] C. Qiao and R. Melhem, "Reconfiguration with Time-division Multiplexed MINs for Multiprocessor Communications," IEEE Transactions on Parallel and Distributed Systems, Vol. 5, No. 4, 1994, pp. 337–352.

[13] C. Qiao and R. Melhem, "Reducing Communication Latency with Path Multiplexing in Optically Interconnected Multiprocessor Systems," in Proceedings of the Int'l Symp. on High Performance Computer Architecture, Jan. 1995, pp. 34 –43.

5.9 Easing the Burden on Latency-Tolerance Mechanisms in Petaflops Computers

David K. Probst
Department of Computer Science
Concordia University
Montreal, Quebec
probst@crim.ca

5.9.1 Introduction

The large diameters of projected petacomputers push the envelopes of required mechanisms for latency-tolerance such as pipelining and multithreading. User/compiler-managed register caches that offer scalable latency avoidance (by neither storing large state nor generating coherence traffic) can ease some of this burden. In high performance parallel computing the fundamental question is whether scalability—at a given design level in the hierarchy—is best achieved by *dependence on* or *independence from* data locality.

5.9.2 Locality Independence

Here, we explore the design point of a multithreaded multiprocessor with locality independence (i.e., true shared memory) and enough bisection bandwidth to balance the machine. Our goal is that neither locality dependence nor cache behavior should be allowed to either constrain available parallelism or degrade the quality of memory bandwidth.

A register cache allows the *state* of a cache line to be partitioned between the compiler and the cache. The cache has associative lookup, presence bits, dirty bits, but no coherence protocol. For each datum, the compiler manages an additional five states (roughly, the Cartesian product of a set of three states and {this processor, not this processor}).

Compiler dependence analysis and user directives allow visible synchronization to directly coordinate transitions among the following three local "split" cache states of a datum:

1. *Private* (i.e., exclusive read/write)—One cached copy exists; the value in memory is not accessible.

2. *Sharable* (i.e., read only)—Several cached copies exist; the value in memory is not accessible

3. *Uncachable* (i.e., shared and writable)—No cached copy exists; access to the location is by "write through".

5.9.3 Summary

We propose replacing *program-independent* cache-coherence protocols by *program-specific* user/compiler control over (a) when to copy into data register caches after visible input synchronization points, and (b) when to uncopy out of data register caches before visible output synchronization points. "Uncopy" means self-invalidate, possibly writing back. Cache invalidation is performed locally— whenever data become shared and writable—by executing machine instructions expressly provided for this purpose.

The new cache instructions are: read/nocopy and uncopy; normal reads and writes may make copies in the data register cache. The semantics of cache instructions is an obvious function of the presence and dirty bits.

5.10 Petaflops Technology: Real Time Image Compensation

Richard G. Lyon
Hughes STX
NASA/GSFC Code 934
Greenbelt, MD 20771
lyon@phase.gsfc.nasa.gov
(301)-286-9768

5.10.1 Introduction

Optical systems designers currently design optics and associated hardware to meet system requirements. Petaflops computing would allow for relaxation of the design optical requirements and correction of the imagery in real time to meet these requirements. The system would encompass not only the optics and detector but the reconstruction algorithm and petaflops computational engine as well. No hardware feedback loop would be required.

The recent successes [2] in restoration of Hubble Space Telescope (HST) imagery has shown that image reconstruction algorithms are a potential alternative to costly optical systems. While algorithms cannot totally recover HST's design resolution due to photometric and noise limitations, optimal systems could be designed for some applications. Discussed briefly here will be two such "canonical systems", each based on petaflops technology. The first will be for real time compensation of a low cost optical system without off-line processing. The other will be for real time atmospheric compensation without the use of costly wavefront sensor and adaptive optical technology. Discussed in a generic fashion will be the computational requirements required to realize both systems.

5.10.2 Real Time Image Compensation

In order to track in real time a rapidly moving object we need a system capable of frames rate approaching 1000 Hz. The object radiance drives the size of the primary mirror and sensitivity increases as the square of the mirror diameter. Thus, for dim or space-based objects the mirror diameter must be quite large. Fabrication of high optical quality large monolithic mirrors is costly and time consuming.

An alternative approach would be to design and build large low cost mirrors with figure errors and compensate the imagery in real time at video frame rates. Realization of petaflops technology would make this an attractive alternative. The image compensation could be performed on a petaflops computational engine using an image deconvolution algorithm. The current state of the art in image deconvolution algorithms is generally considered the maximum entropy algorithm (MEM).

The MEM maximizes the information entropy of the recovered image subject to a set of constraints and is generally considered the best method for ill-posed problems. The algorithm is non-linear and iterative, converging asymptotically to a solution. For Hubble images excellent results are achieved after 1000 iterations. Each iteration requires 20 gigaflops consisting of four, 2-dimensional (1024×1024), 32-bit FFTs and a number of highly data-parallel, complex floating point operations. One thousand iterations would require 20 teraflops, and to do this at a frame rate of 1000 Hz would require 20 petaflops of compute power. The level of parallelism is driven by the number of pixels in the detector array (1024×1024), given a parallelism of approximately 1,000,000. This would require a computational engine with 1024×1024 32-bit processors each operating at 20 gigaflops sustained.

5.10.3 Real Time Atmospheric Compensation

The atmosphere typically changes on the order of a millisecond. A real time device would need to sample it at approximately 1000 Hz. Current atmospheric compensation schemes rely on a wavefront sensor with a delay line feeding back to an adaptive optic. Both the wavefront sensor and adaptive optic are complex and costly and can yet only partially compensate the atmospheric turbulence. Petaflops computing would allow for real time correction without a wavefront sensor or an adaptive optical system. Indeed only conventional optics would be used, and they could be relatively low quality. The imagery would be compensated in real time via phase [1, 3]. Phase diversity relies on simultaneous measurements of an image with known phase differences (e.g., focus). This is easily accomplished by splitting the converging focal beam prior to focus into multiple foci. Phase diversity uses an iterative non-linear algorithm to converge to the phase errors introduced by the atmosphere. From the recovered phase errors the object can be recovered. The algorithm uses $6N + 2$ FFTs per image recovery, where N is the number of iterations (typically 1000). Each FFT requires 5 gigaflops. In order to correct the atmosphere at 1000 Hz we need 30 petaflops of compute power. The level of parallelism is driven by the number or samples (1024×1024) yielding approximately 1,000,000. This would require a computer with 1024×1024 32-bit processors each operating at a sustained 30 gigaflops.

5.10.4 References

[1] R. A. Gonsalves, "Phase Retrieval and Diversity in Adaptive Optics", Opt Eng. Vol. 21, 1982, pp. 829–832.

[2] J. M. Hollis, J. E. Dorband, F. Yuseh-Zadeh, "Comparing Restored HST and VLA Imagery of R Aquar II", Astrophysical Journal, Feb 1992.

[3] R. G. Paxman, T. J. Schultz, and J.R. Feinup, "Joint Estimation of Object and Aberrations by using Phase Diversity", JOSA-A 9, 1992.

6 Applications and Algorithms: Issues and Challenges

This section includes the extended abstracts from participants who made presentations on applications and algorithms issues and challenges of petaflops computing.

Listed below are the titles of the extended abstracts and their authors:

- Enabling Data-intensive Applications through Petaflops Computing, Reagan W. Moore

- Some Applications Demonstrating the Existing Need for Petaflops Computing In Biomedical Research, Jacob V. Maizel, Jr.

- Hierarchical Distributed Genetic Algorithms Control of Simulation-based Optimization: The Need for Petaflops, Bernard P. Zeigler, George Ball, Doo Hwan Kim

- Computational Requirements for Hydrodynamic Turbulence on Petaflop Computers, Anil E. Deane

- Computational Astrophysics Calculations on Petaflop Computers, Bruce Fryxell

- Petaflops and the Gravitational N-body Problem, Kevin M. Olson

- Strategic Applications for Petaflops Computational Systems, Rick L. Stevens, Valerie E. Taylor

- A Case Study of Interactive, Immersive Visualization for Scientific Environments, Valerie E. Taylor, Meena Kandaswamy, Rick L. Stevens

- Parallel Computations for Scientific and Engineering Applications: What Could We Do With Petaflops? What Must We Consider If We Are To Exploit Petaflops?, Guy Robinson (Note: Robinson was unable to make a presentation)

6.1 Enabling Data-intensive Applications through Petaflops Computing

Reagan W. Moore
San Diego Supercomputer Center
San Diego, CA

6.1.1 Introduction

Computers capable of sustaining petaflops computation rates will also enable data-intensive analysis of massive data sets. Just the data archival storage requirements for a petaflops computer will require moving over a petabyte of data per hour. With an appropriate design, however, even faster manipulation of data will be feasible. It should be possible to dynamically manipulate petabyte data sets that are derived from exabyte-sized data archives. This will allow a new mode of science, in which data assimilation becomes as important to the development of predictive models as computational simulation is today.

6.1.2 Data Assimilation

Data assimilation can be viewed as combinations of data mining (in which correlations are sought in large sets of data) and data modeling (in which observational data are combined with a simulation model to provide an improved predictive system.) These approaches may require either locating a single data set within the data archive (data picking) or deriving a data subset from data that may be uniformly distributed throughout the archive. The latter requires supporting the streaming of data through data subsetting platforms to create the desired data set. The hardware architecture that can sustain this high data rate is based on the use of parallel I/O streams to multiple data subsetting platforms.

A petaflops computer will incorporate many of the features needed to support data-intensive problems. It will need to be a scalable parallel architecture for computation, I/O access, and data storage. Data will need to flow in parallel from the data storage devices to parallel nodes on the petaflops computer. A minimum I/O rate needed to sustain just the archiving of data can be estimated from current systems. Assuming that data archival storage requirements for computational modeling on a petaflops computer will be similar to those on a gigaflops computer, it is possible to estimate the amount of data that would be archived by scaling from data

Table 1: Projections of I/O Requirements

System Execution Rate	**I/O Rate to Disk (GB/s)**	**I/O Rate to Archive (GB/s)**	**Data Moved to Archive (PB/day)**
Gigaflops	2 x 10^{-2}	4 x 10^{-4}	35 x 10^{-6}
Teraflops	2 x 10^{1}	4 x 10^{-1}	35 x 10^{-3}
Petaflops	2 x 10^{4}	4 x 10^{2}	35 x 10^{0}

flow analyses of current CRAY supercomputers [1]. For the workload at the San Diego Supercomputer Center, roughly 14% of the data written to disk survives to the end of the computation and 2% of the generated data is archived. The amount of data that is generated is roughly proportional to the average workload execution rate, with about 1 bit of data transferred for every 6 floating point operations [2]. Using these characterizations it is possible to project the I/O requirements for gigaflops, teraflops, and petaflops computers as shown in Table 1

In practice, the usage model is expected to shift towards in-core regeneration of intermediate results on a petaflops computer. This will decrease the amount of data that is moved to disk. On the Cray Y-MIP8/864 at SDSC, a range of I/O generation rates were observed with as few as 1 bit moved for every 10 floating point operations. However, when data-intensive applications are analyzed, even higher I/O access rates can be required. Note that the nominal rate suggests that a petaflops computer will move over 30 petabytes of data per day to archival storage. Thus a petaflops computer should be capable of supporting analyses of petabyte-size data sets on an hourly basis.

6.1.3 Architecture Implications

The architecture design that will support this rate of data movement might be constructed from the following components:

— 10,000 nodes at 100 gigaflops/node

— 100 terabytes of memory at 10 gigabytes/node

— 400 I/O channels at 1 gigabyte/sec per channel connected to 400 storage devices that record at the same rate

— 2.5 petabytes of data stored per storage device for a total archival capacity of 1 exabyte

A terabyte of data distributed across all 400 storage devices could be read in 2.5 seconds. The time to read the entire exabyte-size archive would be 29 days. By increasing the degree of I/O parallelism (increasing the number of I/O channels and storage devices) even faster data processing rates would be achievable.

Petabyte data sets will exist by year 2000. An example is longitudinal clinical patient records (including x-ray and MRI images, video, etc.). The amount of data aggregated over the hospitals within a major city over a five-year interval is expected to exceed a petabyte. The ability to manipulate such a large data set within seconds will allow studies that could greatly improve health care. A second example is the NASA EOSDIS remote sensing data collection [3]. This collection also will be multiple petabytes in size by year 2000 and is expected to grow to 8 petabytes by year 2007. The incorporation of remote sensing data in weather prediction can lead to greatly improved predictive models.

The concept of a petaflops computer is intriguing because of the enhanced computational modeling that will be enabled. However, instead of thinking of such a system as a generator of simulation data, it is also possible to envision a petaflops computer as a processor of archived data. It should be possible to manipulate a petabyte of data per hour using the minimal design I/O capability. A petaflops computer that can process an exabyte-size data archive is even more intriguing, in that it will enable the solution of a significantly larger range of scientific problems that can benefit from data assimilation.

6.1.4 References

[1] Moore, Reagan W., "File Servers, Networking, and Supercomputers," Adv. Info Storage Systems, Vol. 4, SDSC Report GA-A20574, 1992.

[2] Vildibill, Mike, R. W. Moore, Henry Newman, "I/O Analysis of the CRAY Y-MP8/864," Proceedings, Thirty-first Semi-annual Cray User Group Meeting, Montreux, Switzerland, March 1993.

[3] Davis, F., W. Farrell, J. Gray, C. R. Mechoso, R. W. Moore, S. Sides, M. Stonebraker, "EOSDIS Alternative Architecture," Final Report submitted to HAIS, Sept. 6,1994.

6.2 Some Applications Demonstrating the Existing Need for Petaflops Computing In Biomedical Research

Jacob V. Maizel, Jr.
Chief, Laboratory of Mathematical Biology
National Cancer Institute
Building 469, Room 151
Frederick Cancer Research and Development Center
Frederick, MD 21702

6.2.1 Introduction

Experience at existing high performance computing centers shows large requirements for compute cycles in a wide range of problems. Genome analysis, structure-based drug discovery(referred to as Rdrug designS in vernacular terms), and modeling of organs and tissues are examples outlined below for which much foundation research is ongoing. Practical consequences are already envisioned in these areas. Furthermore, most scientists expect that in the coming decades the number of examples in these areas will increase dramatically. Such things as virtual medicine/surgery for teaching and research, prostheses for physical and cognitive remediation, biomaterials, environmental research, information handling and other imaginable areas will benefit from many orders more compute power.

6.2.2 Challenges from the Data of Genome Projects

The Human Genome project should produce the entire sequence of 3×10^9 bases of human DNA in the next decades. It is reasonable to expect that at that time it may be possible to do genomes of similar size, such as individual humans, experimental and domestic animals, crops, and others at the rate of one per year. Rates of 100 bases/sec may be routine. Full, element-by-element comparisons of new sequences with existing data bases would require more than 10^{11} to 10^{12} elementary comparisons per second, compounded as the data base grows. Clever heuristics can reduce the essential operations, but some algorithm discovered in the future could require complete re-analysis of all existing data. Full comparison of human and mouse would need 10^{18} to 10^{19} operations, for example. This kind of comparative sequence analysis is already proven to be one of the most powerful sources of knowledge about genes, and shows promise of becoming more important as the data and knowledge bases grow. These discoveries will drive the need for even more sophisticated analyses.

Genomic information completely determines the characteristics of the protein and nucleic acid molecules that express a living organism's form and function. One of the greatest challenges, in which computation is playing a major role, is the prediction of higher order structure from the one-dimensional sequence of genes. Rules for prediction of macromolecule folding are beginning to emerge. In the case of RNA there are some simple rules that partially predict the secondary interactions of distant parts of the polymer chain. A deterministic, dynamic programming code is in wide use. It scales as $\sim N^3$ in operations and $\sim N^2$ in memory. Other non-deterministic methods are also available. More complex methods for predicting three-dimensional structure are appearing. As an example, preliminary secondary structure predictions for sequences of HIV RNA (9,218 nucleotides) can be done on a 16K processor SIMD Maspar or an 8-processor Cray YMP in about six hours. There are 100,000s of sequences of potential interest ranging in size from 50 to 10,000 nucleotides.

Protein structure prediction is even of greater interest since proteins are the principal agents of expression for genetic information. Rules for prediction are more complex that for RNA, and are a research area of major concern. In its extreme the problem could be viewed as of n^N complexity, where reasonable values for n, the number of conformations amino acids may take, could be dozens, and N, the length of the polypeptide chain, ranges from 10s to 1000s. Exhaustive conformational search would not be feasible for many proteins, even with petaflops computers. However, even now there are a number of strategies to explore the problem. Exhaustive search on highly simplified lattice models, using simplified potential functions is partially successful. Statistics based assignments of structure from sequence similarities is another. In all cases refinement of final structures requires molecular mechanical and computational chemistry tools.

Much experimental work, involving heavy computation, is still needed in algorithm development for both the RNA and protein folding problem.

6.2.3 Drug Design

Current costs of developing a new drug are several hundred millions of dollars. Good candidates are scarce. For example, by screening 10s to 100s of thousands of natural products and other chemicals, new candidate anti-cancer agents emerge at less than one per year. Computation is anticipated as a powerful aid to designing and pre-screening new candidates. This approach requires fundamental structural data on potential targets and the agents that may affect them. High performance computer models are essential for analysis of experimental structural data to produce detailed molecular models, to molecular mechanical/dynamic exploration of target structures, to chemical structure of small drugs and to interaction between drugs and targets. All of these methods will require petaflops to achieve reliable prediction, design and testing.

Structure determination requires computation in solution of the phase problems of x-ray crystallography and in distance geometry calculation for magnetic resonance. Computational chemistry is needed to understand the structure of drugs and how they bind to targets at the atomic level and for details of electronic structure.

As examples, consider the protease of HIV and its inhibitors. This enzyme is required for mature, infectious virus to be produced, and is thus a candidate for drug targeting. It is being studied intensively. At this stage computing speeds are on the order of 10^{12} fold slower than desired. Typically, 10^{-9}seconds of real time requires 100 Cray YMP hours. We want to cover 1 ms of chemical time. Petaflops would permit electronic calculations at moderate levels of theory for the entire protease and very detailed calculations for drug-sized molecules.

The calculations currently scale as N^2 for number of atoms in molecular mechanics calculations, with various heuristics to reduce from this upper bound. Electronic structure requirements vary depending on the level of theory, but range from N^2 upward. Memory requirements scale as N^2 for molecular mechanics with additional off-line storage for molecular dynamics trajectories. Electronic structure codes vary in memory requirement depending whether storage and re-use of intermediate values or re-computation is chosen.

6.2.4 Realistic 3-D Heart Models

Development of a realistic 3-D model of the heart will benefit improved designs for prosthetic heart valves, modeling of cardiac diseases and understanding the functional anatomy of the heart. Petaflops are needed to allow current promising models to be scaled to realistic levels of detail.

The present level of development successfully models some portions of the fluid dynamics of a heartbeat. It models the heart as geodesic fiber paths on surfaces in 3-space, based on painstaking anatomical dissection of mammalian hearts. Fiber forces are transmitted to the blood by a special weighting function. Blood is represented currently by $128 \times 128 \times 128$ 3-D lattice of points on which fluid dynamics are calculated using versions of the Navier-Stokes equations. The problem scales in memory as slightly less then the grid size cubed and in computational complexity as more than N^4. Present requirements are a Cray C-90 cpu-week and 50M words of memory for a single beat. Realistic improvements would require petaflops; and could be utilized immediately to refine the many parameters, to achieve steady state dynamics and to introduce new features such as electrical activity. Methods developed for this work are applicable to problems of sperm motility, platelet aggregation, and other problems with flexible boundaries, such as blood vessels of lung and heart.

6.3 Hierarchical Distributed Genetic Algorithms Control of Simulation-based Optimization: The Need for Petaflops

Bernard P. Zeigler
George Ball
Doo Hwan Kim

High Performance Simulation Project
Department of Electrical and Computer Engineering
School of Renewable Natural Resources
University of Arizona, Tucson, AZ 85721

6.3.1 Introduction

This discussion addresses the computing resource needs for computing environments supporting simulation of landscape ecosystems at high levels of resolution and encompassing large areas such as forests and watersheds. We will argue that current technology is not adequate to support the large amounts of data necessary for representing such systems, the speed required of simulations to provide outputs in reasonable time, nor the control structure necessary to search through the associated model parameter spaces. Our position is based upon experience gained in an NSF/ARPA-sponsored Grand Challenge Application Group project.

The goals of the project are to (1) construct a modeling and simulation environment that employs massively parallel processing to simulate interactions of ecosystem processes at selectable scales of space and time, (2) integrate as intrinsic to the environment, geographical Information system (GIS) data bases to provide realistic descriptions of 3-D landscapes, and (3) support experimentation and interpretation through scientific visualization and automated optimization.

We first review our effort to extract the maximum performance from current technology. Then we state why, despite best efforts, current technology is inadequate and requires petaflop performance levels.

6.3.2 Watershed Simulation

To prototype the application domain, we have developed a watershed simulation with simplified hydrology. The watershed is modeled as a bounded 2-D discrete event cell space with each cell representing a square whose dimensions are determined by the chosen resolution. A column of cells typically starts with bedrock at the bottom, moves up through several layers of soil with differing hydrologic parameters, reaching a surface layer, and continuing on to several cells of air. Rain will be represented as inputs to the top boundary layer of cells and will infiltrate downwards and sidewards to the watershed basin. Each cell is represented by a local model and the cellular heterogeneity requires that each such model have parameters specifying its unique wind, soil, or bedrock characteristics. This information is stored in the GIS and downloaded into each model initially and at specified points in the simulation run.

A major advantage of GIS-integrated simulation is that many of the parameters needed to "situate" a model in the desired landscape are obtained directly from satellite- or ground-based measurements. Still, a large simulation model typically has many more parameters that are unknown and in need of adjustment to tune the model to real world observed behavior or to optimize its performance of a desired behavior. Searching through such large parameter spaces for optimal, or even acceptable, points is a daunting task, especially in multiple process (e.g., hydrology, sedimentation, vegetation) models where each simulation run may require hours or days to complete. The more that automated optimizers can relieve human modelers of this search task, the faster will be the pace of advance in the modeling or design effort. Therefore, optimization-based control of simulation is a key feature of our high performance environment.

6.3.3 Hierarchical Distributed Genetic Algorithms

We are developing Hierarchical Distributed Genetic Algorithms (GAs) that tackle high complexity by performing successive approximation searches. In such a multilevel approach, higher level GA processes work in a wide, but abstract, search spaces whereas a lower level GAs search narrow, high resolution, spaces which seem to hold promise of finding the global optimum. An architecture for a heterogeneous, distributed computing environment is being developed to support the evolution of Hierarchical GAs (HGAs) and the simulation experiments they generate. The HGA consists of GA clusters that are dynamically created hierarchically to work on subproblems at different levels of abstraction. HGAs differ from the conventional distributed parallel GAs in which multiple GAs are constructed at a single level and work on the problem at the same level of abstraction. In the HGA, lower level GA nodes are created dynamically by higher level GA nodes based upon search performance. A GA node which has discovered superior points in its search space receives greater resources to start more refined searches in subnodes. Judicious allocation of resources in a heterogeneous, distributed computing environment is critical to search success. It must be based on knowledge of the current state and on appropriate allocation policies. The emphasis of this domain, compared with standard processor allocation studies, is that the environment is dedicated to work on a single (albeit multiply resolved and distributed) job rather than an assortment of independent jobs.

The computing environment is built on top of parallel virtual machine (PVM)-networked platforms of various powers including workstations and parallel systems (e.g., a locally owned XPlorer transputer and the CM-5 via Internet at NCSA). The architecture is conceived of as consisting of a light processing layer and a heavy processing layer. Decision making operations are performed concurrently in the light processing layer, with the computation intensive simulations in the heavy processing layer. Each layer consists of a large number of processes forming a process pool. The heavy layer pool is centered in the massively parallel components (CM-5 and XPlorer); the light layer consists of threaded processes within the workstations. Information is exchanged between processes via PVM-based message passing. One major advantage of the architecture is that the communication traffic in the light processing layer can be fully overlapped with the "number crunching" in the heavy processing layer in order to achieve a high degree of parallel activity.

6.3.4 Performance Limits and Petaflop Potential

Experiments to date with the environment have been largely aimed at sizing the problem requirements in terms of time and space.

Time

Figure 1 shows the increase in execution time as the number of cells per node increases (the number of nodes is constant at 512). We take as our execution time estimate the time measure for 1,300 cells, the largest number of cells that can be accommodated.

Execution time for one run: 10^{-1} day (2.4 hours)

We also have data for the number of evaluations and search times observed for different platforms when running a representative optimization search problem. From this data, we assume, conservatively:

Number of runs to find optimum$= 10^6$

Since on the 512-node CM-5 the total search time for 10^8 evaluations is less than a day, we can ignore the per run GA processing time. Hence, in the following:

Search time for find optimum$= 10^6$ runs$\times 10^{-1}$ days/run$= 10^5$ days

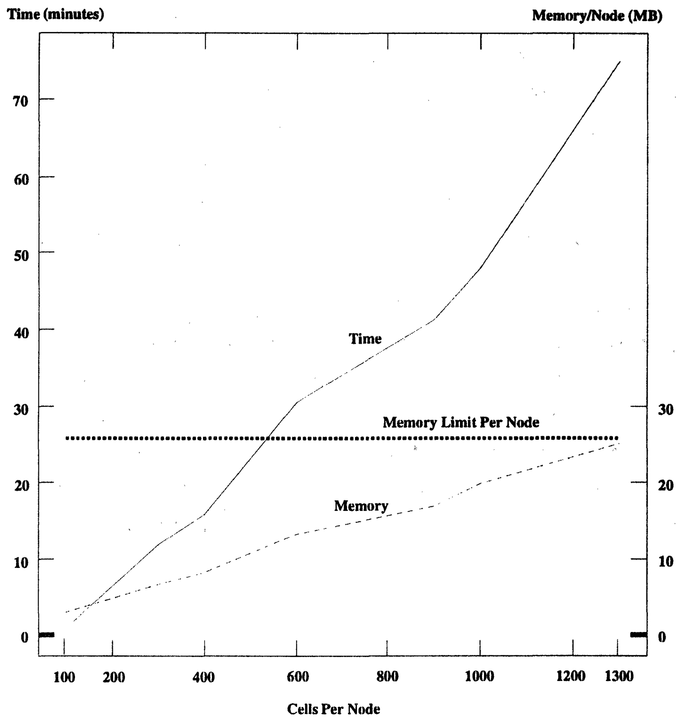


Figure 1: Limit in Memory Per Node of the CM-5

To reduce search time to one day requires a speed-up of 10^5. Since the CM-5 peak realized performance is approximately 10^{10} flops, the peak performance required is $10^{10} \times 10^5 = 10^{15}$, which is the petaflop level.

Space

The size of a simulation is measured in terms of the number of cells it contains. Figure 1 shows the limitations imposed on the number of cells that can be accommodated on CM-5 nodes. The 32 megabyte RAM of the CM-5 node is filled with approximately 3 megabytes of system code. The remaining space can accommodate about 1,300 cells containing one state variable each:

The memory limit per node= 1.3×10^3 cells

Assuming a 1024 node CM-5,

The capacity of 10^3 nodes= 1.3×10^6 cells with one attribute per cell

A spatial resolution of 20 feet (e.g., nominal cell size is 20 $\times$ 20 feet), requires approximately 420,000 cells per square mile. A small watershed can easily encompass 5 square miles, requiring 2.1 million cells to store basic information. Applications of smaller area but with higher resolution (e.g., a stand dynamics model of forest growth) will yield similar requirements. At the extreme end is a problem such as simulating the water runoff from the Grand Canyon to the Colorado river. This would require (at 200 meter resolution) 21 million cells per attribute layer (e.g., elevation).

Considering the six attributes of elevation, slope, aspect, soil, vegetation, and rainfall yields a storage requirement of 126 million attributes or approximately 100 times the storage currently available (assuming code increases linearly with each attribute—an assumption we haven't tested).

In conclusion, the speed required for simulation-based optimization of landscape-level ecosystems scale in the neighborhood of a 10^5 increase over today's technology, and the space scales in the neighborhood of a 10^2 increase.

6.4 Computational Requirements for Hydrodynamic Turbulence on Petaflop Computers

Anil E. Deane
Institute for Computational Science and Informatics
George Mason University
Fairfax, VA
and
High Performance Computing Branch
NASA Goddard Space Flight Center
Greenbelt, MD

6.4.1 Introduction

Most fluid flow is turbulent; be it household pipe flow, smokestack emission, atmospheric and oceanographic flows, flow around flight vehicles or astrophysical flows. To make progress these hydrodynamics subdisciplines have made numerous approximations, rooted in the physics of the phenomena, that make numerical simulation feasible. Thus, the limitations of the model are tied up in the constraints of the available computational resources. There is no single answer that is sought, but a suite of descriptions that trade a better description in one area with a poorer description in another. In this paper we will visit these approximations in some key hydrodynamics subdisciplines, helped by (albeit largely incomplete) theory and considerations of the dimension of the underlying (chaotic or strange) attractor. We are also guided in our speculations by others who have delved into the current practice and future trends of turbulence simulations [1].

At the outset we state, and argue later, the well known fact that turbulent flow is sufficiently complicated that petaflops computing, or even exaflops computing, will not enable the direct simulation of all the relevant length and time scales in most flows of scientific interest. Thus we will explore here what petaflops computing does enable. Hydrodynamics enjoys a rare privilege in the physical sciences in that the equations for hydrodynamics are known. This privilege, however, has not meant that it has been particularly easy to understand turbulence either theoretically or via simulation. In fact the contributions of hydrodynamicists to the theory and to algorithms for the numerical computation of partial differential equations have been seminal and are well known.

The study of hydrodynamics is chiefly divided into compressible and incompressible flows distinguished by the Mach number M of the flow, and viscous and inviscid flows, distinguished by the Reynolds number Re of the flow. It is important to note that incompressible flows $M \equiv 0$ and inviscid flows $Re \equiv \infty$ are singular limits of the equations. Thus the solutions to the viscous equations when the Reynolds number is finite, but arbitrarily large, are not the solutions to the inviscid equations. In particular, as far as viscosity is concerned, the difference between the two solutions confines itself to boundary layers whose size scales inversely with Re. (What the scaling exponent is constitutes asymptotic theory). Thus flows with very high Reynolds numbers, such as those past flight vehicles and most astrophysical flows, are best computed as inviscid flows. The thin boundary layers that form and the complex physics associated with them is then often ignored. The aforementioned complex physics can include non-continuum behavior, particularly when high Mach numbers are involved, that violate the assumptions of the Navier-Stokes equations themselves and new, different, treatments become necessary. In many situations however the Navier-Stokes equations do remain as a valid description, for instance in aerodynamic and atmospheric flows.

Another important point to note concerns the actual mechanics of numerical approximation. The singular limit $Re \equiv \infty$ leads to equations that lack viscous terms. However due to discretization of the equations the error terms that are introduced contain terms that are diffusive in nature. Thus, even though formally the equations being solved are inviscid, their numerical approximation contain what is termed *numerical viscosity*. Yet, in other inviscid systems, such terms are deliberately introduced in order to smear discontinuities leading to *artificial viscosity*. These types of viscosity can and do influence the simulated turbulence and form an effective Reynolds number even when there is none formally present.

6.4.2 Length and Time Scales

Turbulent hydrodynamics involves consideration of length and time scales of the irregular motions. Referring to Figure 1, there are the macroscopic length scales termed energy injection or integral scales which are those of the geometry in which the flow is confined, and at the other end of the spectrum, those scales that are microscopic scales at which viscosity damps the motion, dissipating the energy. In the intermediate region there is a broad range of scales that are too large for viscous dissipation and yet smaller than any geometrical scale. There are good theoretical reasons and ample empirical evidence that in this range, termed the inertial range there is scaling behavior in the motions. For instance the energy dissipation has the behavior [2], $E(k) \sim k^{-5/3}$.

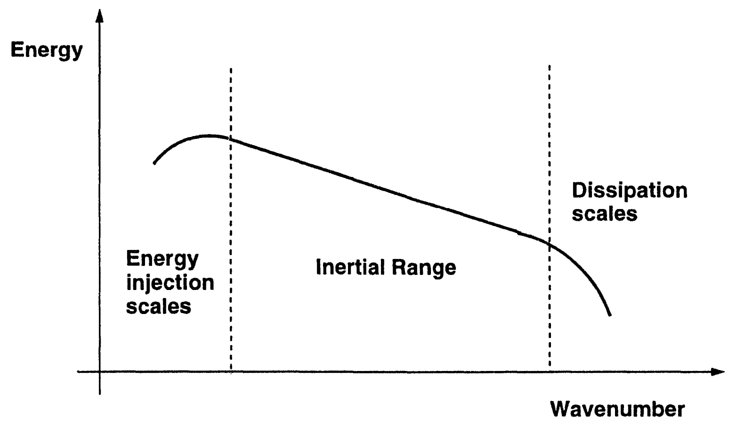


Figure 1: Range of scales in turbulence. Wavenumber is inverse length so that length scales decrease to the right

A direct numerical simulation (DNS) of turbulence serves to capture these three ranges of scales. If the Navier-Stokes equations are a correct description of hydrodynamic turbulence, and they have served remarkably well to date, the complete physical description is available within numerical approximation. Because of the immense resources required, current DNS simulations are limited to $Re_L < 5,000$, where we have used the 'L' suffix to denote the macroscopic length scale, such as the dimension of a body immersed in the fluid. A more useful length scale for turbulence is the Taylor microscale, λ, corresponding to the "beginning" of the dissipation range. Current simulations are limited to $Re_\lambda \simeq 200$ [3, 4].

An alternative to DNS is that of large eddy simulations (LES). Here the largest scales are computed directly. Scales smaller than the grid size of the computational mesh are modeled in this approach. Some turbulence model—and there are a variety of strategies that can be followed—then provides an approximation to these non-computed scales. Current LES simulations are limited to $Re_L < 50,000$.

Most practical systems require Re_L that are orders of magnitude greater than these capabilities. To perform simulations that account for the effects of turbulence in this regime of high Reynolds numbers, three strategies are available. The first is to solve turbulence averaged equations rather than the original equations; this is the practice for flight vehicle design. A second strategy is to arrive at some ad hoc turbulence parameterizations; this is commonly done in atmospheric and oceanographic models. The third strategy is to abandon any hope of doing a simulation at the requisite Reynolds number, and instead either perform a DNS or LES at significantly lower Re and intellectually extrapolate the results to the actual system or to only solve the $Re \equiv \infty$ system. This is common in astrophysical situations.

While capturing length scales constitutes the basis of a turbulence simulation, time scales are an important consideration as well. It is a feature of turbulence that

Table 1: Parameters of a Variety of Hydrodynamic Problems

Regime	*Re*	*M*	Other Physics
Aircraft[1]	10^9	< 5	-
Atmosphere[2]	10^{12}	~ 0	Rotation ($Ro \simeq 0.1$)
Ocean[3]	10^{11}	~ 0	Rotation ($Ro \simeq 0.1$)
Earth Magnetosphere[4]	$> 10^6$	< 5	Magnetic Fields ($Re_m > 10^6$)
Solar Convection[5]	10^{12}	> 1	Magnetic Fields ($Re_m = 10^7$) Rotation ($Ta = 10^{12}$)
Supernova[6]	10^6	10^3	Nuclear Burning

Notes: Values quoted are extreme
[1]Generic Values
[2]For weather patterns of scale 10^3 km. *Ro* is Rossby number.
[3]For major currents of scale 500 km
[4]J. Raeder (private communication), C. Mobarry (private communication).
[5]Ta is Taylor number, Re_m is magnetic Reynolds number; see [5]
[6]Parameters quoted are for debris. *Re* in the core is about 10^4 and in the neutrino sphere $Re \approx 10^{12}$; B. Fryxell and A. Burrows, private communication.

smaller scales adjust on time scales faster than large scales. The largest scales, on the order of the geometry size, need to adjust such that material has had a chance to traverse these scales a few times. The flow is then said to have had sufficient time to adjust during some number of "eddy turnover times". A simulation is typically carried over $O(100)$ eddy turnover times to reach a statistically steady state. This constrains the minimum length of the simulation to this time.

Table 1 summarizes some regimes of scientific interest in terms of the Reynolds and Mach numbers. Since a review of all these diverse areas is impractical, we do not consider the important areas of atmospheric and ocean modeling further. In these calculations hydrodynamics constitutes a fraction of the total computational cost, the major contributor being other physics (e.g., radiation). Thus different scalings hold than for hydrodynamics alone and we do not assess the impact of petaflop computing. Table 2 summarizes some recent large scale simulations, indicating current practical limits.

6.4.3 Minimum Modes

In the modern view of turbulence, turbulence occurs due to presence of a *chaotic* or *strange attractor* that contains the long-term dynamics. This attractor is embedded in the phase space of the system and is of fractional dimension. An initial condition of the system forms a trajectory in phase space that quickly moves onto an *inertial manifold*. Once on this manifold the trajectory goes onto the chaotic attractor. Associated with each of these subsystems is the concept of dimension. (Figure 2 summarizes the situation with regard to the hierarchy of dimensions.) For our pur-

Table 2: Large-scale Computations and Their Parameters

Problem	**Grid Points**	Re	M	**Other Physics**
Harrier Jet[1]	$2.8 \cdot 10^6$	$5 \cdot 10^6$	~ 1	-
Earth Magnetosphere[2]	$2 \cdot 10^6$	∞	< 5	Magnetic Fields
Solar Convection[3]	$2 \cdot 10^6$	$Ra = 2.3 \cdot 10^6$	< 5	Magnetic Fields, Rotation
Supernova[4]	$1000^2(10^6)$	∞	16	Nuclear Burning
Homogeneous Turbulence[5]	$512^3(1.3 \cdot 10^8$	$Re_\lambda = 200$	0	-
Homogeneous Turbulence[6]	$1024^3(10^9)$	∞	1.1	-

Notes
[1] Harrier YAV-8B aircraft. Parameters estimated; see [6]
[2] J. Raeder (private communication)
[3] Ra is the Raleigh number, a more appropriate parameter for convection. See [7].
[4] The Mach number quoted is that of the outwards propagating shock; see [8].
[5] See [3, 4]
[6] The Mach number quoted is the initial M; see [9, 10]

poses it is sufficient to think of dimension in this concept as number of degrees of freedom (DOF). Each DOF can require a separate evolution equation—an ordinary differential equation. The chaotic attractor cannot usually be graphed so a coordinate system is not available to project the governing equations on to give the absolute minimum number of modes that would still contain the attractor. An (approximate) inertial manifold *can* be graphed and hence a projection onto this basis can provide an orders of magnitude reduction of the total number of DOF. The best theoretical estimates so far have not constrained the minimum required modes to be less than the classical results discussed in the workload section that follows, although constructions based on projections have been made [11, 12, 13].

These kinds of projections are, however, in their infancy for hydrodynamics and large scale turbulence simulations still rely on the discretization of the original equations. In this author's opinion however, in the time frame of the availability of petaflop computing such projection methods will come increasingly into play in the simulation of turbulent flows, hence their inclusion in the discussion here.

It is instructive to consider what has been found for attractor dimensions for turbulent flows which are the (unrealizable) absolute minimum number of modes required. Even low Reynolds number flows have attractor dimensions of several hundred. Thus the phenomena of real-world turbulence will not be representable in a small number of equations [14]. In an example system of convection, the dimension of the numerical approximation is 1.4×10^4, while the dimension (Lyapunov dimension) of the chaotic attractor is 120 [15]. In another example, Poiseuille flow, the dimension of the numerical approximation is $\sim 10^5$, while the dimension of the chaotic attractor is 780 [16]. In neither of these examples was it possible to estimate

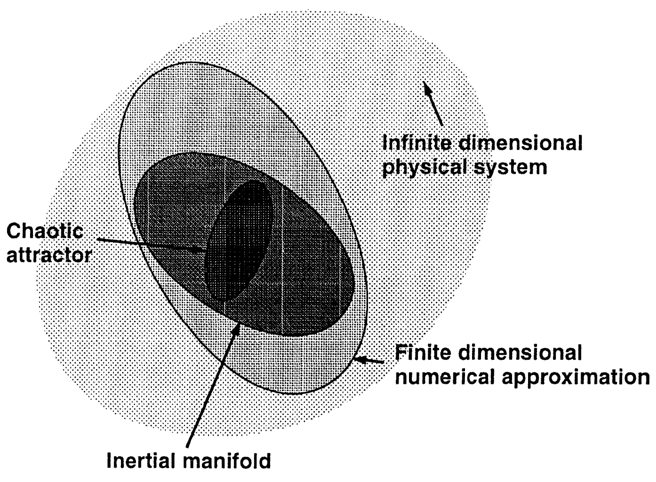

Infinite dimensional physical system
Chaotic attractor
Finite dimensional numerical approximation
Inertial manifold

Figure 2: Hierarchy of Dimensions in Turbulence

the dimension of the inertial manifold. However, by projection techniques such as in [11] an order of 10^2 reduction is possible. Thus projected simulation capabilities may become considerably higher in that the same computational capability buys a bigger simulation.

6.4.4 Workload

A hydrodynamics DNS consists of computing all relevant scales from the integral length scale L to the smallest dissipation scales l, which is then synonymous with the grid size. Turbulence theory gives (e.g., [17])

$$L/l \sim Re^{3/4}$$

so that for a three dimensional simulation $Re^{9/4}$ points are required. The number of floating point operations resulting from the increase in Reynolds number is $Re^{9/4}$. In addition, due to restriction of the time-step, an order of $Re^{3/4}$ number of time steps is required making the total number of floating point operations scale as

$$work \sim Re^3 \qquad (1)$$

Thus writing

$$r \equiv \frac{Re_2}{Re_1}$$

we have

$$\frac{work_2}{work_1} = r^3$$

Now we wish to keep the wall clock time constant,

$$\frac{work_2}{(machinespeed)_2} = \frac{work_1}{(machinespeed)_1}$$

$$\frac{work_2}{work_1} = \frac{(machinespeed)_2}{(machinespeed)_1} \equiv s = r^3$$

$$r = s^{1/3} \tag{2}$$

Memory use is calculated as follows. Let $N_{arr}(\sim 10)$ be the number of arrays (variable + auxiliary) required for calculation. The memory will be

$$memory = N_{arr}N^3 \sim N_{arr}Re^{9/4}$$

where N is the number of grid points in one direction, giving

$$\frac{memory_2}{memory_1} \equiv m = r^{9/4} \tag{3}$$

so that

$$m = s^{3/4} \tag{4}$$

Also note that if n represents the ratio of grid points in one direction,

$$m = n^3$$

Disk storage is proportional to the memory used,

$$d = fmN_{dump}$$

where d is the ratio of disk storage at two machine speeds, N_{dump} is the number of data dumps over the length of the simulation, while f is the fraction of variables to be stored as compared to those needed for computation. For incompressible turbulence three velocities need to be stored while arrays number about 12, making the ratio 0.25. For compressible flows 4–5 variables need storage, with $N_{arr} \approx 30$, with the ratio about 0.2. Other ancillary quantities such as vorticity may also be computed and stored for analysis and a value $f = 0.3$ appears reasonable.

To work out disk I/O rate we proceed as follows. The number of words that must be operated on in N_{steps}, is $N_{alg}N^3N_{steps}$, where N_{alg} is the operation count for the algorithm (for turbulence simulations N_{alg} is 500–2000). Since this many operations must be performed by the code on a machine of speed (ratio), s, the time taken is$N_{alg}N^3N_{steps}/s$ so that the disk I/O rate is,

$$\frac{fms}{N_{alg}N^3N_{steps}} \tag{5}$$

6.4.5 Petaflops

In Table 3 we list extrapolations of various hydrodynamics problems for computers capable of petaflops. In arriving at these estimates, equations (3) and (4) have been used. In estimating the disk storage, the assumption is that order 10^3 data dumps are typical (current practice is 10^2–10^3) over the time of the simulation. In addition, for storage purposes 0.3 times the number of memory is a canonical number for stored variables (e.g., only the velocity fields are stored). For purely hydrodynamic problem N_{arr} = 10 is used, while for additional physics such as magnetic fields N_{arr} = 20 is used. Both these values are underestimates.

Table 3: Scaling to Teraflop and Petaflop Computing

Problem	Current (10GF)	Teraflop	Petaflop
Harrier Jet	$Re_L = 5 \cdot 10^6$	$2.3 \cdot 10^7$	$2.3 \cdot 10^8$
memory (ratio)	$1.3 \cdot 10^7$	$4 \cdot 10^8$	$7.2 \cdot 10^{10}$
Earth Magnetosphere			
memory	$4 \cdot 10^7$	$1.3 \cdot 10^9$	$2 \cdot 10^{11}$
disk storage	$1.2 \cdot 10^{10}$	$4 \cdot 10^{11}$	$6 \cdot 10^{14}$
Solar Convection			
memory	$4 \cdot 10^7$	$1.3 \cdot 10^9$	$2 \cdot 10^{11}$
disk storage	$1.2 \cdot 10^{10}$	$4 \cdot 10^{11}$	$6 \cdot 10^{14}$
Supernova			
memory	10^7	$3.2 \cdot 10^8$	$5.6 \cdot 10^{10}$
disk storage	$3 \cdot 10^9$	10^{11}	$2 \cdot 10^{13}$
Homogeneous Turbulence	$Re_\lambda = 200$	930	9300
memory	$1.3 \cdot 10^9$	$4.3 \cdot 10^{10}$	$7.4 \cdot 10^{12}$
disk storage	$3.9 \cdot 10^{11}$	$1.3 \cdot 10^{13}$	$2.2 \cdot 10^{15}$
memory I/O rate	10^{11}	10^{13}	10^{15}
disk II/O rate	$3 \cdot 10^7$	$3 \cdot 10^9$	$3 \cdot 10^{12}$
Homogeneous Turbulence			
memory	10^{10}	$3.2 \cdot 10^{11}$	$5.6 \cdot 10^{13}$
disk storage	$3 \cdot 10^{13}$	10^{15}	$3 \cdot 10^{16}$

Note:
All projections, except those of the supernova (B. Fryxell, see [18]), are this author's estimates. The current capacity of 10GF is an approximation and is not really true across the simulations. Memory and disk storage are in words. Memory and disk I/O rates are in words/sec across the full machine, I/O rates must therefore be divided by number of I/O processors to arrive at rates in terms of words/sec/processor.

For memory I/O we have used the argument that in order to obtain the machine speed, s, the cache/registers must be supplied by memory at the rate $10s$. This is certainly true for today's machines, and could remain true for future machines. For disk I/O, we have used (5) with $N_{steps} = 1$, which is a worst case value.

Figure 3 shows the scaling for incompressible turbulence simulations. Similar graphs hold, of course, for the other cases—the starting points vary. As the graph indicates, with petaflop computers grids of $(9000)^3$ calculations should become possible, given adequate memory. From Table 3 we see that the simulated Reynolds numbers will reach $Re_\lambda \approx 9300$ which are enormous.

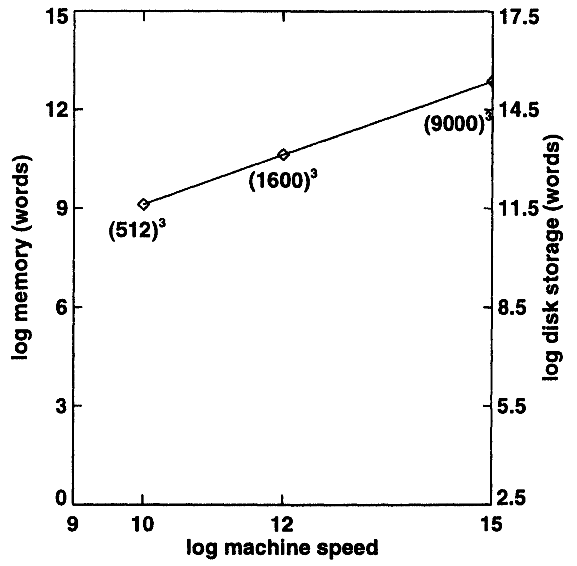


Figure 3: Memory and Storage Requirements for Incompressible Turbulence Vs. Machine Speed. (Shown also are the equivalent box sizes, i.e., number of grid points, that can be realized.)

We have resisted the temptation to break the memory, I/O, and disk requirements into per processor requirements, since N_p, the number of processors for petaflop computing is a technology-driven number. Also, it is not clear how well turbulence simulations can use a very large number of processors (10^3–10^6). Certainly good (with spectral codes) and excellent (with finite-difference/volume) efficiencies can be obtained for $\leq 10^3$ processors [1]. Spectral codes with in-processor 1-D FFTs will probably be completely replaced by across-processor multidimensional FFTs to obtain reasonable efficiency for very large numbers of processors.

High-order schemes (e.g., compact differences, high-order polynomial basis) that have high data locality will be the techniques to beat.

6.4.6 Conclusion

Turbulence simulations will benefit immensely from sustained petaflop performance. Not only will the range of simulations be increased, as measured by the Reynolds and Mach numbers, but so will the quality of the simulations, as measured by the included physical effects. Full flight vehicle simulations with modeling of subgrid scale turbulence (LES) at realistic parameters should become possible. The balance will be found between increasingly sophisticated modeling which will give better answers and will drive the simulated Re up, but which will drive up the cost of the simulation. Hence a decreased grid resolution will become necessary which will in turn drive the simulated Re down. It is likely that the DNS of full flight vehicles will be possible only with exaflop computing (or more).

The greatly increased computational capacity will in all probability be used in many simulations not to drive up the basic parameters, but to introduce more complex physics and simulated regime (and in some cases full three-dimensionality), whose effects are at present unknown, or only known qualitatively. For example, solar convection models currently include compressibility, magnetic fields, turbulence models, rotation and ionization effects. These models will likely see greatly increased parameter values. In addition multiple layer models and sphericity will be included. The coupling of solar evolution models to detailed convection codes should become possible.

Others areas where the current practice is limited to inviscid calculations such as the magnetosphere and supernova (to select two among space science/astrophysics areas) will likely see the explicit introduction of turbulence models. Then small scale turbulence effects which can greatly affect mixing and energy transfer will improve the overall simulation and bring it closer to the physical system. It is, of course, unlikely that a DNS of these flows will be possible in the foreseeable future. Similar remarks apply to atmospheric and ocean models, where grossly parameterized turbulent fluxes in global simulations will (by necessity, because of the finer resolutions) include turbulence models (in the sense of LES), but DNS will not be possible.

As we have mentioned previously, novel projection methods that are based on dynamical systems theory will play a role in the numerical computation of turbulent flows. How big a role depends on theoretical developments that are immensely complicated and need fundamental breakthroughs: hence they are unpredictable. These methods could change the values stated in Table 3.

Finally, the study of turbulence as (computational) laboratory flows for theoretical reasons will see the greatest increase in simulated parameter ranges. A well developed inertial range, valid over many decades in wavenumber, is important for many theoretical models and to the further development of turbulence models.

6.4.7 References

[1] Karniadakis, G. E. and Orszag, S. A., Physics Today, 46, 34 (1993).

[2] Kolmogorov, A. N., Dokl. Akad. Nauk SSSR, 30, 301 (1941).

[3] Jiménez, J., Wray, A. A., Saffman, P. G., and Rogallo, R. S., J. Fluid Mech., 255, 65 (1993).

[4] Chen, S., Doolen, G. D., Krachnan, R. H., and She, Z. S., Phys. Fluids A, 5, 458 (1993).

[5] Priest, E. R., *Solar Magnetohydrodynamics*, D. Reidel, 1982.

[6] Smith, Al. H., Chawla, K., and Dalsem, W. R. V., "Numerical Simulation of a Complete STOVL Aircraft in Ground Effect," in *AIAA Paper 91-3293, AIAA 9th Applied Aerodynamics Conference, Baltimore, MD*, 1991.

[7] Nordlund, A., Galsgaard, K., and Stein, R. F., "Magnetoconvection and Magnetoturbulence," in *Solar Surface Magnetism*, edited by Rutten, R. J. and Schrijver, C. J., Kluwer, 1995.

[8] Müller, E., Fryxell, B., and Arnett, D., Astron. and Astrophys., 251, 505 (1991).

[9] Porter, D. H., Pouquet, A., and Woodward, P. R., Theor. Comput. Fluid Dynamics, 4, 13 (1992).

[10] Porter, D. H., Woodward, P. R., Anderson, S., Chin-Purcell, K., Hessel, R., Perro, D., Zacharov, I., Ryan, J., Widra, L., and Galles, M., "Attacking a Grand Challenge in Computational Fluid Dynamics on a Cluster of Silicon Graphics Challenge Machines," unpublished.

[11] Aubry, N., Holmes, P., Lumley, J. L., and Stone, E. F., J. Fluid Mech., 192, 115 (1988).

[12] Sirovich, L., Knight, B. W., and Rodriguez, J. D., Quar. Appl. Math., 48, 535 (1990).

[13] Batcho, P. F. and Karniadakis, G. E., J. Comput. Physics, 115, 121 (1994).

[14] Temam, R., Proc. R. Soc. Lond., A 434, 23 (1991).

[15] Sirovich, L. and Deane, A., J. Fluid Mech., 222, 251 (1991).

[16] Keefe, L., Moin, P., and Kim, J., J. Fluid Alech., 242, 1 (1992).

[17] Jackson, E., She, Z.-S., and Orszag, S. A., J. Sci. Comput., 6, 27 (1991).

[18] Fryxell, B., "Computational Astrophysics Calculations on Petaflop Computers," in The Petaflop Frontier Workshop, 1995 (this report).

6.5 Computational Astrophysics Calculations on Petaflop Computers

Bruce Fryxell
Institute for Computational Sciences and Informatics
George Mason University
Fairfax, VA
and
NASA Goddard Space Flight Center
Greenbelt, MD

6.5.1 Introduction

Computational astrophysics is such a diverse field that it is impossible to summarize it in a single short paper such as this. Nevertheless, a number of aspects common to many (if not all) areas of the field require the use of very large computational resources. It would be incorrect to state that a certain level of performance (such as petaflops) is necessary to solve a given problem. Instead, some crude information can usually be obtained about a given astrophysical system by making approximations so that the system can be simulated on a less powerful computer. Such approximations might include assuming a geometrical symmetry to the problem to reduce the number of spatial dimensions or ignoring certain physical effects which may change the solution quantitatively but not qualitatively.

As computer power increases, however, more and more realistic simulations can be performed by improving the numerical resolution of the calculation and by using more realistic input physics. The numerical accuracy of a calculation is usually determined by checking to see if the solution has "converged", i.e., if the answer does not change significantly when the resolution is increased. The performance required to reach this desired accuracy will vary significantly depending on the system being simulated and the approximations used. Because of the nonlinearity of the equations, the details of the solution and, therefore, the resolution needed to resolve all the important behavior are very difficult to predict in advance. However, it is clear that the level of computing performance required for many problems is significantly beyond what is currently available and for some problems even petaflops-level computing may be insufficient. Although in some cases a complete solution may not be attainable on a petaflop computer, such a large increase in computer power would certainly be accompanied by a much clearer understanding of these systems.

6.5.2 General Properties of Astrophysical Simulations

Astrophysics simulations are challenging for several reasons. First is the wide range of length and time scales that must be resolved. Astrophysics calculations cover length scales from the size of the universe to a small fraction of the radius of an individual star, and time scales from the age of the universe to a tiny fraction of a second. Fortunately, most simulations need to take into account only a small fraction of this range, but it is not uncommon for length and time scales in a single simulation to vary by many orders of magnitude. In order to resolve all the relevant length scales, one must either use computational grids with a huge number of points, use adaptive mesh techniques, which are much more expensive and complex, or both. One way to deal with the large range of time scales is to use an implicit technique, which usually runs less efficiently on parallel machines. However, if important physics is occurring on the smallest time scales, there is no substitute for using a time step value less than the smallest important time scale in the problem. In this case an enormous number of time steps may be required in order to evolve the system to its final state.

A second reason for the challenge of astrophysics simulations is that to obtain the complete solution to many astrophysics problems, a full three-dimensional simulation is required. Many systems, e.g., spiral galaxies, are inherently three dimensional. The evolution of some systems, such as individual stars, that appear roughly spherical, actually depends critically on non-spherical motions such as convective flows and circulation currents. Another good example is the collision of two stars. A head-on collision can be treated using a two-dimensional approximation. However, the chance that an exactly head-on collision will occur in nature is extremely small. The more common case of a grazing collision requires the use of all three spatial dimensions. In addition, for any system involving turbulent flow—a very common situation given the low viscosity of most astrophysical gases—two-dimensional simulations will give the wrong qualitative behavior because large scale structures, rather than small scale structures will dominate.

Astrophysics calculations also stress the I/O capabilities and storage capacity of computers. The reason for this is that most systems are time-dependent and do not evolve to a steady state. It is necessary to study every stage of the evolution to fully understand the behavior of the object being simulated. As a result, the entire state of the system must be stored a large number of times during the calculation. Each phase of the evolution can then be examined individually, or they can be combined to form a movie of the evolution.

A significant complication in trying to characterize the computational requirements for astrophysics calculations is that a wide variety of computational techniques is needed. For example, simulation of individual stars requires gas dynamics codes, while simulation of star clusters or galaxies is usually done using particle methods. Each technique will place different demands on computer architectures. There is probably not a single architecture that will be optimal for each situation. To make matters worse, simulation of the most complex systems will require hybrid codes that combine two or more computational techniques. Trying to predict what type of architecture is best in such a situation or guessing the computational requirements for such a simulation is extremely difficult. In order to make the task more manageable, the remainder of this paper will concentrate on systems which use only continuum physics rather than particle methods. For hybrid codes, some of the computational requirements quoted may be significant underestimates.

6.5.3 Requirements for a Petaflop Computer

In a field that is changing as rapidly as astrophysics, it is difficult to predict what problems will be of interest at the time a petaflop computer becomes available, or what techniques will be required to solve them. Thus, instead of analyzing the requirements to solve a given problem, it seems more appropriate to proceed in a more general way. The number of grid points required to obtain a believable answer will vary considerably from simulation to simulation, depending on how much fine-scale structure exists in the object being studied. However, experience with current two-dimensional codes has shown that 1000 grid points per spatial dimension is adequate to obtain reasonable results for many (but not all) problems. As more complex physics is added to the simulation, this number may no longer be sufficient, but for lack of a better estimate, it will be used in the remainder of this section. Thus, for three-dimensional simulations, 10^9 grid points will be required.

As mentioned above, there is probably no ideal architecture for all problems in computational astrophysics. However, there is usually a high degree of parallelism in the techniques. For much of the calculation, each processor can be assigned to update an individual grid point (or a group of grid points). In many cases, exactly the same operations are performed at each grid point, except for an occasional *if* test. These types of simulations can be performed equally well on SIMD or MIMD computers. However, if many different types of physical processes are important, and each is taking place in only a small portion of the computational grid, MIMD computers may have an advantage. For explicit methods, communication between grid points is local and is usually confined to nearest neighbors.

Many complex physical processes, such as reaction networks and equations of state calculations, require no communication at all. When such techniques are included in the simulation, communication overhead may be completely dominated by the computational costs. In this case, efficient simulations could be performed by letting each processor work on a single zone. For a 1000^3 simulation, for example, one could use up to 10^9 processors effectively. However, in general it would probably be more practical to assign perhaps 1000 zones to each processor so that only 10^6 processors could be used.

Simulations also exist that use techniques requiring global communication. The simulations include implicit hydrodynamics, radiative transfer, and calculation of gravitational potentials by solution of Poisson's equations. Non-local communication will also become important if unstructured grids or adaptive mesh techniques are used. The maximum degree of parallelism for these methods is still a matter of research, but it is likely that it will be much easier to make them work efficiently on computers with a smaller number (< 1000) of faster processors. It is still unclear at this time if MIMD architectures will have an advantage over SIMD for these types of techniques.

The amount of core memory needed is relatively easy to estimate, especially for simulations which involve only gas dynamics. In order to solve Euler's equations for compressible gas dynamics in three spatial dimensions using an explicit finite difference method, it is necessary to store at least the gas density, the total energy, and the three components of velocity at each grid point. The temperature and pressure of the gas are frequently stored also, although for simple equations of state, these could be calculated when needed without significantly increasing the CPU time of the simulation. If magnetic effects are important, then the three components of the magnetic field must also be stored. Thus, at least five to ten variables are required for each grid point. A few additional variables may be required for the gravitational potential, electric fields, electric currents, and so forth.

For some systems, the number of variables required can be considerably more. Frequently, additional variables must be stored to keep track of the composition of the gas. In stellar evolution, for example, the gas is composed of a mixture of a large number of nuclear species. A separate variable is required for the fraction of each of these species at each grid point to determine the rate of energy generation from nuclear reactions and to compute the pressure from the equation of state. In some cases, the fraction of the gas in each ionization state must also be stored. Thus, it is possible that the number of variables required could grow to a few hundred per grid point, or more.

Some additional memory will be required for scratch arrays. The exact number needed will depend on the algorithm and programming style. For the present estimate, it will be assumed that the amount of extra storage is a small fraction of the total. Assuming 10^9 grid points, the total memory required for various simulations should be in the range of $5\text{–}1000 \times 10^9$ words. Using 64-bit words, which should be sufficient for most applications, the memory requirement will range from 0.025 to 10 terabytes.

The amount of memory can also be estimated in a completely different way by scaling up from current computer configurations. The amount of memory required does not scale linearly with the processor speed for applications such as this. If the number of grid points per dimension is doubled, the total memory required for a three-dimensional simulation increases by a factor of 8. Likewise, the time to advance the solution one time step increases by a factor of 8. However, due to time step restrictions for numerical stability, twice as many time steps must be used to advance the solution the same amount of physical time, causing the total CPU time to increase by a factor of 16. Because of this, the memory should scale as the 3/4 power of the processor speed. If the starting point for scaling is taken as a gigaflop computer with a gigabytes of memory, a petaflop computer should need on the order of 10 terabytes of memory, the same as the estimate obtained above.

The CPU time required for a simulation with a given number of grid points will depend on the time required to update each grid point and the total number of time steps needed to reach the desired physical time. Both these numbers will depend on which system is being simulated and the complexity of the physics being tested and will vary over a wide range. Frequently, the complexity of the physics and the grid size are adjusted so that the simulation can be performed in a reasonable length of time (usually overnight). Thus, a typical calculation will be assumed to use on the order of 10 hours of processor time. Even calculations that take a much longer time to complete are usually run as a series of overnight jobs. This makes it possible to determine the amount of on-line storage and I/O bandwidth required. Assuming that the current state of the system being simulated can be represented completely by 10 terabytes of data and that it will be stored 1000 times during the simulation, 10 petabytes of fast on-line storage will be required. The I/O bandwidth required to dump this much data during the 10-hour calculation is 1 petabyte per hour or about a quarter of a terabyte per second. The amount of off-line mass storage will have to be scaled up from that available on current computers by a similar amount. Of course, these estimates assume that the computer is dedicated to a single user. In a multiuser environment the numbers will have to be scaled up accordingly.

6.6 Petaflops and the Gravitational N-body Problem

Kevin M. Olson
NASA Goddard Space Flight Center
Greenbelt, MD
and
Institute for Computational Sciences and Informatics
George Mason University
Fairfax, VA

6.6.1 Introduction

In this paper I consider what impact the development of a petaflop computer could have on the gravitational N-body problem in astrophysics. Both the direct N^2 algorithm as well as tree algorithms are considered. A petaflop computer should be able to solve the direct problem for particle numbers several orders of magnitude larger than is currently feasible, and realistic simulations with particle numbers matching real systems where a high force resolution is required should be possible (e.g., globular clusters with 10^6 stars). For tree codes the same should be true, but the scaling arguments are not as straight forward. For most astrophysical systems a solution of the N-body problem is not enough, and new numerical techniques (already under development) and greater physical understanding will be required to accurately model these systems. The implications of the development of a petaflops computer would have for simulations of two specific N-body systems are considered: globular clusters with$\sim 10^6$ stars and disk galaxies with $\sim 10^{11}$ stars.

6.6.2 The N^2 Problem

The force on particle i in a system of N gravitationally interacting particles is given by

$$\vec{F_i} = \sum_{j=1}^{N} \frac{G m_i m_j \vec{r}_{ij}}{(r_{ij}^2 + \epsilon^2)^{3/2}} \tag{1}$$

where G is the universal gravitational constant, m_i and m_j are the masses of the particles i and j, $\vec{r_{ij}}$ is the vector separating them, and ϵ is a smoothing length which can be nonzero and serves to eliminate diverging values in $\vec{F_i}$ when $\vec{r_{ij}}$ is small. This parameter also serves to define a resolution limit to the problem. This equation also shows that the problem scales as N^2.

To perform the above sum on the Maspar MP-2 requires roughly 3–4 seconds for 16,384 particles. The peak performance of the Maspar MP-2 is $\sim$ 5 Gflops. If we assume that the same algorithm will scale linearly with the performance of the machine and that a similar fraction of the peak speed is attained, the time to perform all the sums for all the particles in a system is then given by

$$t_{force} = 3 - 4 \times \frac{N^2}{16,384^2} \times \frac{5Gflops}{10^6 Gflops} seconds \tag{2}$$

A typical simulation requires $\sim 1,000$ to 10,000 time steps and if $N = 10^6$, a simulation would require a running time of $\sim$ 75–750 seconds (assuming the time required for the integration of the equations of motion for each particle is small). Further, the amount of memory required for this problem is small. Nine numbers are required for each particle (3 positions, 3 velocities, and 3 accelerations) so that the total required memory is $9N \times 8$ bytes = 72 Mbytes for $N = 10^6$. Hence, realistic simulations of systems such as globular star clusters would easily be possible simply by increasing the processing speed to a petaflops level. The amount of disk space, I/O band width, and mass storage available today should be adequate for this specific problem.

We note that simulations of other N-body systems such as galaxies or clusters of galaxies would also be possible to a degree of resolution equal to that achieved in some of the largest of today's N-body simulations which currently use approximate techniques.

6.6.3 Tree Codes

Tree codes are a collection of algorithms which approximate the solution to equation 1. In these algorithms the particles are sorted into a spatial hierarchy which forms a tree data structure. Each node in the tree then represents a grouping of particles and data which represents average quantities of these particles (e.g., total mass, center of mass, and high-order moments of the mass distribution) are computed and stored at the nodes of the tree. The forces are then computed by having each particle search the tree and pruning subtrees from the search when the average data stored at that node can be used to compute a force on the searching particle below a user-supplied accuracy limit. Since the tree search for any one particle is not known *a priori* and the tree is unstructured, frequent use is made of indirect addressing. Therefore, in order for this algorithm to operate at reasonable efficiency on any machine, the time to make an indirect address and receive the requested data from the most distant memory location should be of the same order as the time to perform one monopole force calculation or $\sim$ 30 floating point operations.

For a fixed level of accuracy this algorithm scales as $Nlog(N)$ although $O(N)$ algorithms are also possible. However. if the number of particles are increased we also wish to run the algorithm to a higher degree of accuracy so that the approximation in the force calculation will not spoil the higher degree of spatial resolution we wish to achieve by using larger numbers of particles. To appreciate the magnitude of this effect, we note that the typical acceleration in a structure of mass M and size R is GM/R^2. If we allow the softening length to be a measure of the resolution we wish to attain in a simulation, then an upper limit to the tolerable accuracy in the acceleration will be $\sim GM/\epsilon^2$. Further, the softening length is normally chosen to be on the order of a typical interparticle separation and the mass M is the mass of a single particle. With these assumptions$M \sim 1/N$ and $\epsilon \sim N^{-1/3}$. Therefore, the accuracy with which we need to run a typical simulation scales as $N^{-1}/N^{-2/3} = N^{-1/3}$. Empirically, the running time of a typical tree algorithm scales with the chosen accuracy roughly as $1/log^{2-3}(\delta/a_{typ})$ where δa is the error in the acceleration and a_{typ} is a typical acceleration in the system. Therefore, the time for a tree search scales with N as $log^{2-3}(N^{1/3}) \times Nlog(N) \sim Nlog^{3-4}(N)$.

On the Maspar MP-2 a typical tree search takes roughly 10 seconds when run at an accuracy of 1% using N=65,536. Assuming that the peak performance of the Maspar MP-2 is 5 Gflops and that the same fraction of this peak would be attained by the hypothetical petaflop machine we arrive at the following relation for the time to search the tree and compute forces in a typical N-body system,

$$t_{tree} = 2.8 \times 10^8 \times 10sec. \times (5Gflops/10^6Gflops) \times Nlog^4(N)sec. \quad (3)$$

If $n = 10^6$ this would result in a tree search time of $\sim 2 \times 10^{-3}$ seconds. For $N = 10^9$ and $N = 10^{11}$ the search times would be ~ 10 seconds and $\sim 2,000$ seconds, respectively. It seems reasonable that simulations of this type will be possible with several billion particles.

The memory requirements for simulations using trees scale with N, but the scaling constant is larger than in the above discussed N^2 algorithm and a simulation with $N = 10^9$ would require ~ 2 terabytes of core memory. Further, we wish to save the particle data (positions and velocities) to disk for later analysis. If a typical simulation is run for 10,000 time steps and we save the data every 10 time steps we would require at least $\sim 1,000 \times 6 \times 8$ bytes $\times N$ or ~ 50 terabytes of disk space for one simulation. Further, if we demand that the I/O does not significantly interfere with processing, and we restrict the time spent in I/O to be less than a tenth of the total processing time, I estimate the required I/O bandwidth to be ~ 5 gigabytes per second.

6.6.4 Disk Galaxies

We consider next what this would mean for the specific example of a disk galaxy. Modeling of disk galaxies requires that a certain fraction ($\sim$ 70% to 80%) of the mass of the system be distributed in a spherical halo to simulate the so called dark matter and to make the disk globally stable. Hence, if we used 10^9 particles to simulate a disk galaxy roughly 7×10^8 particles would be distributed in this halo with the rest of them distributed in the disk. The average distance between particles in the disk would then be $\sim 10^{-4}$ of the disk radius. This distance effectively defines a lower limit to the resolution of the problem and is a few times smaller than a typical interstellar gas cloud for the typical dimensions of a disk galaxy.

With the addition of a gas dynamical model (e.g., Smooth Particle Hydrodynamics) to the basic N-body model we could model the evolution of a disk galaxy in some of its properties down to the resolution of some its more detailed features (a resolution not possible today). The addition of such a gas dynamical model, however, would add roughly a factor of 10 to the running time as well as to the memory requirements. This lower limit to the resolution also means that interesting effects known to operate on smaller scales (e.g., star formation) could not be included in an explicit way in these simulations. But, it would almost certainly be possible to model a single interstellar gas cloud which does include such effects using a petaflop computer.

6.7 Strategic Applications for Petaflops Computational Systems

Rick L. Stevens[1]
Argonne National Laboratory
MCS Division
Argonne, IL

Valerie E. Taylor
Northwestern Univ.
EECS Dept.
Evanston, IL

6.7.1 Introduction

In this paper we outline our assumptions for a hypothetical petaflops computer system and give a number of large-scale applications that we believe such systems would enable. The applications described in this paper are only a few of the many types of applications that we believe would become possible with peta-scale computing environments. We are interested in continuing a detailed analysis of these applications requirements and to understand the implications these requirements have for systems architecture, high performance networking and software environments. We have specifically focused on applications that do not have existing counterparts running on 10–100 gigaflops machines to force the "peta" discussion to expand to include new types of applications, as such these applications are speculative but we believe in the spirit of the original petaflops workshop held in Pasadena in 1994. We also note that several of these applications would have nontrivial social implications if built. This paper is meant to spur discussion.

6.7.2 Global Assumptions

We assume a target date of 2012 for an integrated petaflops system. The typical high-end user will by this time have a 10 gigaflops workstation with OC–3c ATM network connection to the global Internet. The global Internet will have backbones

[1]This research supported in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under contract W-31-109-Eng-38.

of terabit speeds, perhaps constructed of many logically trunked OC–192 lines. Portable and head mounted VR interfaces will support multiple HDTV resolution devices via OC–3c network connections. Latency in wide area networking applications is assumed to be limited by speed of light propagation delays and not significantly influenced by network congestion schemes. Due to high-latency we think most highly interactive applications will be primarily targeted to North America. The peta-scale computer system is estimated to cost in the $100 Million to $1 Billion range. Therefore all the applications are aimed at strategic topics and interests that potentially could support investments of that scale.

Assumed characteristics of petaflops computing systems:

- 10K–100K processors (10–100 gigaflops /processor) 10–20-way internal parallelism with .1–1ns clock (1 GHz–10 GHz)
- 256–1024 TB/RAM (primary memory) (.256–1 GB/GF)
- 20–100 PB Disk (secondary memory)
- 10–100 EB Tape (tertiary memory)

Node Memory Bandwidth and Hierarchy Needed

- 20 instructions per cycle with each instruction 32-bit OPS with 64-bit addresses
- Word size = 128 bits single/256 bits double
- Mem. bandwidth= 1280 bytes/cycle$*$10 GHz= $12,800\text{GB}/s = 13\ \text{TB}/s$

We assume a seven- (perhaps six-) level memory hierarchy roughly like this (per processor):

1. Registers = 10 TB/s, size = 1 MB
2. L1 cache = 1 TB/s, size = 10 MB
3. L2 cache = 100 GB/s, size = 100 MB
4. L3 cache = 10 GB/s, size = 1 GB
5. Primary = 1 GB/s, size = 10 GB

6. Secondary = 100 MB/s, size = 100 GB

7. Tertiary = 10 MB/s, size(shared) = 100 EB

Internal Communications (internode communication)

- 10–200ns communications latency (internode)
- 10–100 GB/s communications bandwidth (internode)

External Communications (between the petaflop systems and external world)

10 TB/s aggregate external network I/O bandwidth is assumed. For reference: VR video data rates (15 MB/s), NTSC video data rates 5 Mb/s (e.g., MPEG–2).

- 1 GB/s per processor (OC–192 ATM for example)
- (66 VR feeds per processor) 660K–6,600K VR feeds total
- (2,000 full motion video feeds per processor) 20M video feeds total

Storage Environment

10–100 EB of tertiary storage; $10^{15} bytes * 10^5 - -10^6 processors$

6.7.3 Molecular Nanotechnology CAD Systems

Description of Application: Molecular nanotechnology CAD (NanoCAD) systems are the software environments for designing atomically precise structures consisting of between 100K and 100 Million atoms. The structures designed and simulated in a NanoCAD system will be constructed by special computer controlled molecular scale assembly machines that will use scalable self-replication to produce macroscopic quantities of desired products.

The complex specification of systems containing millions of parts, assembly sequences of potentially millions of steps and the simulation and modeling needed to insure reliable, safe and efficient construction of these devices will stress even petaflops computer systems. None of these software systems exists today but we hope to estimate the requirements of such systems by extrapolating from today's molecular modeling and CAD systems.

Applications Requirements:

Much new software needs to be developed to model and simulate the processes of nanomechanical assembly. Some of the modeling systems needed are

— Molecular modeling of 100 million atoms

— Simulation of ensembles of molecular nano devices

— Simulation of assembly sequences

— Searching for process technologies

— Proving safety and reliability of structures

— Visualization of structure models and assembly sequences

— Failure mode analysis and diagnostic and testability analysis

Research Issues

- Scalable Quantum/MM/Continuum models
- Scalable Molecular CAD techniques
- Molecular Space Filing Algorithms
- VR for Assembly sequence visualization
- VR for model building and CAD interfaces
- Discovery of Process Technologies
- Scalable algorithms for verification and testability

Comments

Short range simulations of 100 M particles have been done. Hierarchical methods and scalable electronic structure codes are under development. VR for molecular model building and reaction pathway planning will be needed. Biggest challenge is a 10^6–10^9-fold speedup in modeling capability need to simulate the entire assembly sequence of a large (100 mega-atom) nano-object.

6.7.4 Multiuser Shared Immersive Environments

Description of Application: Providing a shared virtual environment for many hundreds of people to collaborate by interacting in a shared virtual space that provides shared data, shared virtual objects, live audio and stereo high-resolution video and access to interfaces to computer controlled devices, such as instruments, machine tools, surgical systems, robots, and even entire factories or spacecraft. The shared space will enable collaboration and interaction unrestricted by distance to enable commerce, entertainment and virtual presence for essentially any purpose. A petaflops system with the configuration from above could provide this virtual world to perhaps 100,000 people simultaneously and allow a virtually unlimited number of interactions between those connected.

Applications Requirements:

— Split rendering systems (partially rendered on central system, partially on local system)

—Hardware support for rendering and t-mesh manipulation, texture mapping etc.

—Ability to sustain a large fraction of external network bandwidth to users

—Ability to move user loads throughout the machine

—Ability to simultaneously model the virtual world and support interactions

—Support for wide variety of video and graphics primitives

—Order of a million simultaneous connections

—For reliability need RAID-like scalable processor to insure survivability.

Research Issues:

- Network performance,
- I/O performance,
- Modeling and sustained I/O simultaneously,
- Network connections scalability,
- Graphics hardware support for fast volume rendering, ray-tracing, VR, radiosity algorithms,

- Programming models for distributed, wide area, collaborative VR
- Recording and playback models for VR systems.

Comments

We need numbers for graphics performance, ray-tracing and radiosity models; numbers for the types of data that people would have access to on-line (images, text, books, movies, sports). Several projects are underway at ANL/NWU to building performance models of shared interactive VR spaces driven by supercomputing simulation (see Taylor, Stevens abstract this report).

6.7.5 National Scale Data Mining Engines

Description of Application: An aggregate of publicly available data and simulation resources that are updated continuously to support a variety of computational query environments enabling complex simulation based queries. This resource would most likely be linked dynamically into distributed applications, thereby provided a continuously accessible global database providing decision support for national activities (government, business and economics modeling, education and research). Databases would include, land use, agriculture, meteorological, economic, social and census data for example.

Applications Requirements:

— Large, integrated databases/knowledge bases

— Common simulation environment backend for distributed applications

— Support for hundreds of thousands of on-demand network connections

— Supporting cross correlations and arbitrary data mining operations

— Support multiple levels of access and security

— Scalable data intensive simulation environment to meet real-time goals

— Reliability

— Support for large-data movements

— Support for multimedia data types and indexing, searching and organizational structures

— Support for unstructured “exploratory” browsing and experimental engines

Research Issues:

- Scalable multimedia indexing and knowledge extraction
- Knowledge translation and semantic filtering
- Scalable simulations,
- Support for audio and video queries
- Fast passes through tertiary storage
- Indexing, browsing, and clustering techniques
- Portability and scalable data structures
- Reliability

Comments

We need good performance models and quantitative information for query types, indexing, simulation-based queries, video and audio, transactions and user interfaces.

6.7.6 Integrated Global Earth Systems Simulation Environment

Description of Application: Centralized database and simulation environment for real-time accumulation of remote sensing data and support of interactive earth systems simulations (global climate, ecosystems, agriculture, watershed, land use, wildlife management and fisheries etc.). Data would be qualified, “productized”, and made available in both data form and “assimilated” form (the result of incorporating into a continuous faster than real-time, or FTRL, simulation). This system would support economic forecasting, government planning and regulation, crop management, ecosystem management and education and research.

Applications Requirements:

— Support for many simultaneous FTRL simulations

— Potentially intercommunicating, real-time database data feeds

— Real-time output, many thousands of users (mostly automated wide-area simulation front ends)

Research Issues

- Resource management in FTRL simulations
- Assimilation enabled databases and interfaces to simulations
- Communications and I/O requirements for remote sensing data feeds
- Dynamic process management and process migration

6.7.7 Global Lifelong Education and Training Resource

Description of Application: Essentially a global university in a box, this application would contain complete educational and training materials for all known careers and profession/technical jobs. The environment would make available interactive VR-based instruction, multimedia-based, self-paced learning environments, interactive computer simulations, distance learning specialists on-line, and comprehensive aptitude and self-assessment capability. The system would also maintain and automatically update profiles for automated jobs search and placement processes. From learning a new language, to learning to play the guitar or to cook Ethiopian food the system will provide examples, instructors, resources and feedback on demand.

Applications Requirements

Systems support needs to be provided for

— User (or user-process proxy) initiated simulations

— Interactive user-to-user communication

— Intelligent multifunction agents

— Multimedia and VR interactive courseware

— Flexible interfaces to environmental systems (sensors, robotics)

Research Issues

- Effective models for distance learning
- Remote teaching and digital libraries
- Capture and enhancement of master teacher technology
- Databases (interactive MM databases)
- Dynamics linking to remote sensors and telepresence hardware

6.7.8 International Design and Modeling Resource

Description of Application: In this application we create an international design and product modeling resource (IDMR). The IDMR would contain a complete CAD specification of products and the manufacturing facilities necessary to fabricate and assemble them. The range of products could include all items necessary to support a modest-scale first-world economy from first principles. This repository would be a major design and engineering resource that would allow small startup organizations to build on the expertise and legacy systems developed over the last 20 years of use of digital systems. It would also support the rapid deployment of environmentally sustainable manufacturing technology in the third world. By simply installing adequate communications and display technology, an instant information and knowledge infrastructure can be created and projected. Communications technology could be land based optical or satellite based narrow beam. This design repository would contain the following types of data: CAD files, simulation models, video and audio documentation on techniques and instructional materials, manufacturing and fabrication knowledge, bootstrapping pathways and structured systems for rapid deployment of capability. It would also contain a complete database on distribution and financing, thereby opening instant markets for products. The database would allow startup ventures to immediately attack open market niches from any place on the planet.

Applications Requirements

The IDMR would require the ability to automatically compute industrial development trajectories (For example, given initial resource and environmental conditions what are the next steps that can be taken to rapidly develop new manufacturing and design resources?) Industrial trajectories require the analysis of perhaps millions of potential development strategies and to optimize each scenario to local and temporal conditions. While individual product specifications are likely to increase in size, complexity and computational requirements, the primary peta-systems requirement will be from the desire to search for development paths rather than investigate individual product development options. We think that a development trajectory would cover complete market sectors and consist of hundreds of process steps, thousands of products/components and involve the simulation of dozens of factory siting and construction steps. The integrated model should be capable of providing information at near real-time performance.

Research Issues

Principle issues are scalability and integration plus new modeling frameworks (i.e., just what exactly is a development trajectory model) Also scalable integrated CAD and simulations need to be developed.

- Industrial engineering models
- Integrated resource/environmental models
- Integrated CAD/mechanics modeling systems
- Generalized planning and robotics systems
- Scalable manufacturing capability (desktop to factory)
- Multidisciplinary optimization modeling systems
- Integrated database/simulations environments
- Enterprise trajectory modeling systems
- Economic and financial systems models

6.7.9 Human Knowledge Repository

Description of Application: Build the 21 century equivalent of the library of Alexandria, an internationally accessible digital library. The Human Knowledge Repository (HKR) would contain a record of all human knowledge (books, video, audio, models, images, 3-D representation of artworks, computer programs, digital design files and complete life recordings, etc.). Reference materials would be cross indexed and maintained in a state to support research and study. Intelligent assistant technology would provide search and browse proxies and would allow anyone to construct educational and enrichment composite data sources. The primary advantage of a close coupling of HKR with a petaflops system is the prospect of making the data available via multiple languages, in multiple modalities and with dynamic automated indexing and correlation agents. Accessing the HKR will be through video, audio, text, VR and other data streams (e.g. computer to computer binary formats).

Applications Requirements

Assuming 100 exabytes of external storage the system needs to be able to do the following operations on new data as it is being added to the system:

— Index

— Translate

— Compress

— Search

— Composite

— Integrate

— Verify

At 10 terabytes per second aggregate I/O bandwidth the system would take 100 seconds to move 1 petabyte; 100,000 seconds ($\sim$1 day) to move 1 exabyte; and 10–100 days to make a pass over the entire database.

Research Issues

- Access modality conversion
- Digital representation and presentation
- Rapid digitization and capture of knowledge assets
- Indexing/searching/correlation techniques
- Efficient compression and retrieval (hybrid systems)
- Automated syntactic and semantic translation systems (especially for domain specific knowledge bases)

Comments

We need to estimate the time needed to convert modalities (language, format) for various data types. For example, how long would it take to automatically translate 1 million digital books into 50 languages? Could one construct concordances of millions of books and improve automatic translation by having giant lookup tables?

6.7.10 Lunar and Mars Bases HKR Backup Station

Description of Application: The objective here is to provide a HKR in a box that would be portable and highly reliable to be placed in orbit, on the Moon and on Mars for redundancy and to serve the emerging space based human population. The primary challenges for this use of the HKR revolve around trying to make the system compact, self-repairing, power efficient (even power self contained) and highly reliable. We estimate that a remote HKR system should have a 10-year design lifetime (i.e., design to be upgradable for 10 years) and a 25-year MTBF. The system will need to be capable of robotic maintenance and be largely self-maintaining. The system should provide incremental upgrade capability and be able to be linked to other systems like itself for replication, updates, consistency checking and backup.

Applications Requirements

- Ultra reliability—internal process redundancy
- Robotic serviceability—self-maintenance
- Low power consumption
- Incremental upgradability
- Compact tertiary storage (goal would be 10^{20} bytes/cubic meter)
  $100GB/mm^2 \rightarrow$ 1 bit per 100 cubic microns

Research Issues

- Compact radiation hardened processors and storage systems
- Incremental upgradability
- Acceleration resistant components
- Parallel interfaces to wide band communications

- Flexible systems administration systems
- Self- maintaining software environments

6.7.11 Conclusions

We have outlined several information intensive applications that could use peta-scale computer systems. In fact we believe these are likely to be the most interesting class of emerging applications for the next 20 years. These applications are speculative (they don't and can't yet exist) but are likely to be built in one form or another during the next century. We believe they need to be studied and understood from a quantitative and software and systems architecture standpoint, and that they are an important class of applications projects that could be potential users of peta-flop computers. We hope these sketches will encourage people to think of new and even exotic uses of the next generations of large-scale computers.

6.8 A Case Study of Interactive, Immersive Visualization for Scientific Environments

Valerie E. Taylor
Meena Kandaswamy
EECS Dept., Northwestern Univ.
Evanston, IL

Rick L. Stevens
Argonne National Laboratory
MCS Division
Argonne, IL

6.8.1 Introduction

Interactive, immersive video combines real-time video with 3-D models of solids or physical systems. The users are physically immersed in this virtual environment and can navigate within an object in real time. This type of display is necessary for exploration of otherwise inaccessible environments such as hazardous and dangerous sites or remote space exploration. Systems for these environments would consist of an integrated array of cameras located at the site, a petaflop computer for volume reconstruction from the video data, and an immersive 3-D display such as a Cave Automatic Virtual Environment (CAVE) or head-mounted display. Numerical models would be integrated into the system to perform simulations of the various components of the remote site based upon the interactive feedback of the users.

This type of environment will come to fruition with the availability of Petaflops systems. It has been estimated that real-time 3-D pixel (or voxel) reconstruction requires on the order of 10^5–10^6 floating-point operations per second per voxel. For a 10^{12} voxel 3-D CyberScene object, 10^{17}–10^{18} floating-point operations must be calculated per second for real-time 3-D display. In addition, resources must be allocated for the numerical simulation.

The case study involves the development of a performance model of a scaled-down version of the aforementioned system. Our system consists of the 3-D interactive, immersive visualization of the results from a finite element simulation. The system is simple but captures many of the critical features of the system involving reconstruction in addition to simulation. In particular, we will use the simulation-only system to identify the compute, memory, I/O, and network requirements for a real-time interactive system. This model will be used to provide insight into the requirements for the reconstruction and simulation environment.

6.8.2 Environment

The interactive, immersive simulation environment consists of a 128-node IBM SP-1 system, a HIPPI switch, an SGI Onyx, and a CAVE. The CAVE is a 10-foot cube with display screens on two walls plus the floor. High-resolution video projectors are used to display the output of the SGI Onyx system with three RealityEngine graphics pipelines. Liquid crystal shutter glasses are used to present different views to each eye of the user, thereby providing a perception of depth. A head-mounted, electromagnetic tracking device allows the user to navigate within the virtual space. The 3-D rendering is generated based on the physical layout of the projection screens and the relative position of the user wearing the tracker. Interaction is achieved with a hand-held wand having three buttons, similar to a mouse.

The current application involves a small mechanical system modeled with 8-node hexagonal finite elements. It consists of a grinding wheel composed of 615 nodes and 433 elements used to grind a block composed of 96 nodes and 48 elements on a table represented by a single hexagonal element. This prototype is indicative of larger systems used to analyze complex mechanical systems, including the detection of contacting surfaces, friction at the interfaces, large rigid body motions, and thermal-mechanical analysis with finite elements.

A simulation event consists of running the finite element code on the SP system and transferring the data from the SP to the SGI via HIPPI sockets using TCP/IP writes. Future plans include using IPI writes to the SGI. The simulation data is then stored and processed by the three graphics pipelines; the graphics pipelines send the display data to the video projectors. A tracker event consists of a user navigating within the virtual space. These movements are relayed back to the Onyx resulting in a new display relative to the tracker; the update display is computed in real time. Changes to the table elevation or simulation reset is accomplished by pressing the specific wand buttons. This wand information is relayed back to the SP system for simulation of the modified system. Upon completion of the simulation, the results are transferred to the SGI for a new display.

6.8.3 Methodology

The performance model for the simulation-only system consists of the computational and memory requirements of the simulation and rendering codes in addition to the message size and frequency of the data transferred between the simulation, graphics engines, and the CAVE. The critical aspect of the analysis is the characterization of the data transfers, which is the least understand due to the recent availability of such a system. Hence our immediate goal is to identify the network traffic patterns of the simulation-only system.

Currently, we are instrumenting the graphics code to collect statistics on the traffic patterns of a tracker event (movement within the virtual space) and a simulation event (using the wand to change some aspect of the model). In addition, we are gathering data on the computational requirements of the various events. The experiments focus on the interactions between the following components:

- SP and SGI Onyx
- SGI and the CAVE projectors
- Wand and SGI
- Head tracker and SGI

We are also identifying limitations imposed on the transfer frequency, e.g., the maximum frequency at which the wand updates can occur. Further, we are investigating methods for recording the events of a design session. This will allow us to duplicate experiments to distinguish average patterns from anomalies.

When we understand the network traffic and required bandwidths, we will analyze the computational and memory requirements of the simulation and rendering codes. Significant work has been done in the area of models for parallel computations. We will tailor these models of communication and computation to the finite element code and the CAVE rendering codes.

The computational and memory models will be combined with the network traffic models to provide an understanding of the entire system. This combined model will be used to analyze bottlenecks in the simple system and provide insight on the computational and network requirements for large scale simulations. In particular, we hope to use this model to provide insight into the following important issues:

- Effective methods for shipping data from the SP system to the SGI Onyx
- Limitations on the frequency of wand updates
- Ratio of compute processors to graphics processors for a balanced system
- Memory requirements of the overall system
- Resource requirements for a particular frame rate

After we understand the simulation-only system, we will progress to more complicated systems involving:

- Many SP nodes for the simulation
- Many processors for the CAVE rendering code
- CAVE-to-CAVE displays
- Three or more CAVE displays

The CAVE-to-CAVE environment requires analysis of WAN traffic in addition to the LAN traffic of only one CAVE. A number of issues arise in this environment with the major issue being the latency in the transfer of data and the amount and frequency of data to be transferred. Given that features may be changed about the physical system being simulated and only one site may have a parallel machine, significant amounts of data must be shipped to the remote CAVE. Hence, an additional component would be added to the performance model to relate to remote transfers of data.

6.8.4 Summary

This paper describes the motivation and environment of the initial stages of a case study on the interactive, immersive visualization of scientific environments. The study is being conducted using the CAVE visualization environment at Argonne National Laboratory. The result of the study will be a performance model of a system involving simulations on the SP system and data visualization on the CAVE. This model will be used to provide insight into the inefficiencies of the existing systems and lead to recommendations and requirements for a real-time system involving large scale scientific simulations and CAVE-to-CAVE visualization.

6.9 Parallel Computations for Scientific and Engineering Applications: What Could We Do With Petaflops? What Must We Consider If We Are To Exploit Petaflops?

Guy Robinson
NPAC, Syracuse University
Syracuse, NY 13244-4100
robinson@npac.syr.edu

6.9.1 Introduction

At present the majority of large-scale scientific computations could easily be modified to embrace the teraflops-scale of computing resource. The barriers regarding petaflops computing will be much harder to breakdown—if we are to perform not just frontier science but real engineering computations.

Much of my research activity has been concerned with the solution of discrete sets of equations representing a physical system with a number of external boundary conditions. Often this area is dismissed as being relatively easily parallelized and that problems can simply expand to make use of the parallel resources available. However, we can show that this is not quite the case by the study of two representative examples such as the simulation of long transients for physical systems (e.g., nuclear reactors, offshore structures and aerodynamic systems) and global and climate modeling. Note that both these cases are in the production environment.

6.9.2 Typical Problem Magnitude

All scientists and engineers have a list of numerical experiments they would like to perform. Some are possible with existing codes whilst others require varying degrees of additional development work. Parallel systems capable of supporting these calculations and in returning results within the critical time span of the design cycle are not yet generally available. For example, some large scale transient calculations performed to support safety cases can take three months on a 5-gigaflops machine even with assumed symmetry and simplified boundary conditions reducing the problem. These simplifications often result in solutions that are of little relevance to specific systems. This discourages many scientists and engineers and companies from performing such computations or reduces such work to small studies to the side of main research interests. Such computations could easily be performed overnight on a teraflops machine, perhaps in a coffee break for a petaflops machine. Nor should we forget the human effort involved in establishing such a model. Nearly one month's work was spent developing and debugging the various supporting code and model concepts.

Many CFD and other engineering computations make assumptions about symmetry or apply a simple decoupled model of the complex physics involved. The study of large-scale steady states has been revolutionized by the availability of significant computing resources, systems with nearly one million nodes being commonplace. The predicted boom in available computing power should allow these computations to be expanded into transients. Many projects cannot be attempted, yet alone completed, without this magnitude of support. The use of Numerical Wind Tunnels [1,2] shows how there is a consistent change from experimental to numerical design work, partly due to the high cost of experimentation with today's expensive systems many of which are unique designs and the impossibility of performing many experiments. Estimates show that nearly 90% of research into some future projects will be numerical.

Several possible outlets exist for parallel computing to expand the nature of problems being tackled in science and engineering circles. The size of problems and the increasing degrees of freedom are continually expanding. Fully unstructured 3-D flows are now being resolved by unstructured adaptive meshed transient techniques with few constraints being introduced to reduce the computational demands. Many complete systems require on the order of 10–1000 million nodes for a complete geometrical representation with perhaps several orders more required for a good solution.

Codes which combine several physics models (e.g., CFD, stress and heat transfer) can be developed to predict the behavior of modern macroscopic objects within and without design parameters. For example, consider the simple automobile engine. The engine involves many physical processes and difficult numerical physics and *only* the ease of experimentation has allowed considerable development progress to be made. Ease of experimentation, however, is not possible in all cases. High speed flight, for example, is difficult to represent directly even in experiments. Some experiments will still be required, but these will be in a supporting role to validate our codes and specific design features. Numerical simulation will reduce, but not eliminate, the need for experimental studies, and these studies should be carefully selected so as to avoid building a "tower of babel".

Whilst it is encouraging to consider this line of development, there are problems. The majority of scientific codes can be considered as having two parts. An equation generation part from the application of various models that represent the laws of physics, chemistry, etc., and a solver that produces a solution to this. Both will need considerable revision if our goal of solving large, irregular fully 3-D problems is to be attained.

6.9.3 Problems With Equation Generation

Here we must look to increasing the order of our equation sets. Many codes today solve first or partly second-order nonlinear equations, second-order or higher methods may be required with the associated increase in computation.

At present there are few combined codes that model fluid structure interaction, and they often are designed for specific cases such as flight or marine structures. Chemistry can be found in many codes along with various other models. Often these physical processes are divided into two or more equation sets which are solved in isolation, one providing boundary conditions for another. For highly accurate solutions which are not prone to errors these equations must be coupled. In studying complex systems there is always the danger that poor solutions will be obtained, particularly if the system has any chaotic nature and the solution at any one point is poorly computed. Such coupled equations are necessary to solve transient problems with any degree of accuracy.

6.9.4 Problems With Solver Scaling

Coupled equation sets with high-order approximations and nonlinear relationships are difficult to solve. At present we have been able to solve only simple equations and have been able to use simple methods. Our existing parallel methods often scale poorly with the number of unknowns in the problem, even for simple regular linear problems (seeTable 1). This poor scaling is hidden by the higher performance of computing power available which has exceeded the growth in problem size: in short, we have been lazy. The breakthrough to larger problems will require the development of solvers which offer near linear scaling with problem size.

Table 1: Compute Time for a 3-D Problem

Solution Method	**Age**	**Run Time**	**Time to Solve n=1000 Problem**
sor	1960	$8N^4 log2(n)$	2.5 Years
cyclic red	1970	$8N^3 log2(n)$	22 Hours
mg	1978	$60N^3$	17 Hours

There is little interest in solving problems one order of magnitude larger if it requires the pain of using a machine with three orders of magnitude the cost. It is important to notice how current parallel systems reward locality of our solvers. For solvers to scale well requires the global interaction of multigrid techniques, which is not parallel friendly. Many simple steady state problems have been nearly impossible to solve without multigrid solvers [3, 4]. Perhaps it will be necessary to develop entirely new techniques such as sparse grid methods whereby high resolution is obtained by combining many coarse meshes together. As yet this is not fully understood for unstructured irregular problems.

6.9.5 Problems of Display and Interaction

These studies will generate a large amount of data. The storage and later, or even runtime, interpretation of these may require the development of new methods [5]. Multiple linked views may be required to convey even basic information regarding the information contained within output datasets. Mining may be necessary to extract the answers to specific questions and even expert systems with basic knowledge of the type of problem. Also similar systems may be required to specify, and perhaps verify, the problems initially before computation. Hopefully we can develop these systems gradually with the rise in computing power, but some effort must be expended in this area.

6.9.6 A Cautionary Note

We should be careful in just increasing the resolution and complexity of our models and expecting linear or even previously observed scaling. Higher resolution may resolve new physical features which might require additional computation to resolve correctly. Higher accuracy boundary conditions might be required in both spatial and temporal terms. Perhaps many calculations will need to be performed with slight perturbations in the initial conditions to see if the solution obtained is stable or just to find the range of solutions which might occur. Such methods are already under investigation by members of the meteorological community. Here, limits in the number and accuracy of the initial boundary conditions demand that greater attention be paid to the methods used.

Considerable additional computing is required by 4Dvar because temporal data is included in the initial conditions. This requires many cycles until the numerical solution fits the actual observations. Ensemble forecasting is simpler, requiring only many simple forecasts with slightly perturbed initial conditions. But, both techniques require large amount of computing effort that coupled with the time constraints results in a forecast computation having a limited lifetime. Indeed this time constraint limits the complexity of the physics model used. Similar constraints can be applied to climate studies where at present severe limitations are placed on the physics models used and conservation can be a major problem. This could be resolved with improved physics models which obviously require greater computation.

6.9.7 The Design Cycle

Today the design of high technology devices is supported by considerable computational effort. The construction of optimized, or in terms of safety or ecological behavior, preferred complex systems such as aircraft, power generation, and even automobiles has become a difficult task. These objects have been built from several isolated (or perhaps poorly constrained) systems. Power sources were designed with only limited interaction with body work or external structures. To further advance designs requires coupling the two processes. Perhaps this can only be achieved by taking great steps forward and attempting radical designs. The economics of any field however dictate that such bold steps would be thoroughly validated and proven in a computer study before construction was even considered. Help from genetic design methods may help. But, since evolution is a slow process and the assessment of each design is an expensive process, this also will require prodigious computing resources. The quality of the results in the design of solutions to difficult problems cannot be disputed, however, and often designers have stolen the best ideas from natural systems for inclusion in our modest attempts. Most of our early flight experience came through the observation of birds and their wing movements: observation that contributed to the design of control devices. Even today the sudden outbreak of vertical wingtips can be mapped onto similar features in natures widebody equivalents.

6.9.8 Petaflops, The Hardware and Software

If we can rely upon the hardware and software designers to deliver petaflops machines, then we must assist their endeavors. The exploitation of the current generation of parallel machines has limited the choice of algorithms. Indeed, we often see considerable competition between advanced methods on older vector shared-memory machines using the latest numerical methods and older methods on parallel machines. The true exploitation of petaflops will require the combination of both advanced solution techniques and parallel systems. To achieve this we must make the designers of these machines aware of the fundamental principles of these advanced methods, just as they must make us aware of the features of these parallel systems.

The field of QuantumChromoDynamics (QCD) has often constructed the world's most powerful machines, of the order of a teraflops at present. These have typically been "bespoke" machines designed for only one problem class. I propose that we really require highly modular machines where we do not decide to fix CPU power, communication bandwidth and topology. Users must be free to mix, match and configure as required. The time scales of machine development are too short to support anything else. We have already seen some good developments in this direction with vendors allowing CPU and communication networks to be exchanged during the lifetime of a single "box". Further efforts must be made in this direction. Of course, such highly variable systems will require accommodating software.

One of the often quoted rules of parallel processing is that it rewards locality. Some of the above methods can reward locality over an impressive number of processors. However, as we increase the non-linearity of the equations we wish to solve and the magnitude of the problems increase, this rule's alternative statement comes into play: use as few processors as possible.

6.9.9 Conclusion

In conclusion, there is little doubt that there are scientific and engineering computations to be done with petaflops-scale computing resources. The methods, both computational (in terms of the algorithms exploited) and scientific (in terms of obtaining useful and meaningful results from these machines) are a major point for considerable discussion. However, it is clear that we shall only progress further in any scientific discipline with the support of computation, both as an experimental and an analytical tool.

6.9.10 References

[1] "Integration of Numerical and Experimental Wind Tunnels," IofNEWT, Bailey, George, Koga, Bunning, Delzio, Kulfan. Supercomputing 94.

[2] "NAL Numerical Wind Tunnel," Miyoshi et al Supercomputing 94.

[3] "Parallel Algebraic Multigrid," Robinson, Parallel CFD92, New York, 92

[4] "Algebraic Multigrid Solver for the NS equations in the Discrete Second Order Approximation," Webster and Robinson (to be published)

[5] "Computation Models Applied to Problems in Medicine," Johnson and Parket, Supercomputing 94.

[6] European Centre for Medium Range Weather Forecasts Annual Report, 1994.

7 Discussion and Conclusions

This section addresses the motivation for the first Petaflops Frontier Workshop (TPF-1), several key consequences of the workshop, major findings of the presentations, and implications for future directions.

7.1 Motivating Factors for TPF-1

It was with some trepidation that the organizers embarked on this project to conduct The Petaflops Frontiers Workshop. An incipient field with a target likely to be 20 years in the future is vulnerable to issues of credibility. By its own design, it can not impact high performance computing in the near future. There are also not likely to be a large base of active researchers who would immediately identify with this field. Yet, there were strong reasons to coordinate such a meeting, in spite of the worrisome uncertainties.

Perhaps the most important reason is that there are many researchers engaged in work that directly relates to Petaflops computing, even if they themselves do not cast it as such. Because such research often does not have immediate near term application, it too may be more difficult to justify, especially to sponsoring agencies. By providing a petaflops computing system requirements conceptual framework, such research can be positioned in terms of its potential contribution to this larger whole. Also, the array of relevant research domains is so diverse, little cross fertilization of ideas and results occurs. Again, a petaflops conceptual framework enables such interdisciplinary exchanges. Finally, much of the basic research being conducted truly requires long term investment and must be pursued now to be ready in time to make its contribution. Identifying prospective candidates for enabling technologies and methods research now can only be achieved if the requirements of petaflops computing and the opportunities of such research products are matched. And this can only happen by bringing both together by among other means such a forum as The Petaflops Frontier Workshop.

Another important motivating factor for making TPF-1 a reality was the first Pasadena Workshop on Enabling Technologies for Peta(FL)OPS Computing. If the organizers of TPF-1 had some sense of being on the edge, imagine how the organizers of the Pasadena workshop felt. Yet, the response was exceptionally favorable and professional. The quality of the workshop, its participants, its findings, and the interest of the general high performance computing community to that undertaking can not be overstated. The very positive reaction to the Pasadena workshop was highly encouraging; without it, TPF-1 in all likelihood would not have occurred. Yet, the Pasadena workshop was only a first step with many limitations in fields considered and concepts addressed. The Pasadena workshop participation was by invitation only, assuring a high density of experts in the desired fields. While not a weakness, this was a self-biasing approach to attendance, representing those areas that were assumed to be premiere while permitting little opportunity for surprise or unanticipated possibilities. TPF-1 was left the opportunity to engage a second tier of participation by letting the petaflops community define itself. An open call for participation permitted all facets of the research community to contribute including those areas that might not otherwise have been represented in an "invitation-only" setting. As a result, the Pasadena and TPF-1 workshops complemented each other and were so successful it may prove to be the model for the future.

One of the recommendations of the Pasadena workshop was to provide complete coverage of relevant fields. By its nature, this was not possible at the Pasadena meeting. TPF-1 was organized to attract a diverse group of researchers and there by directly address this recommendation. A second recommendation of the Pasadena Petaflops workshop was to provide more detail and breadth in the area of application scaling. TPF-1 was organized explicitly to target applications programmers with problems whose requirements could scale many orders of magnitude beyond contemporary means and there by address this second recommendation as well.

Finally, it had been almost exactly one year since Pasadena workshop and a second effort was called for to engage this inchoate community, maintain the momentum, and examine results of the last year. The Frontiers of Massively Parallel Processing Conference, more than perhaps any other single conference series, has retained as its charter the domain of computation at the farthest conceivable extremes. This conference provided an ideal context for TPF-1 , both in concept and from the practical standpoint that many of the right people would likely be in attendance already.

It should be noted that throughout this endeavor, the organizers were encouraged and supported by NASA. The logistical details were professionally managed by the USRA Center for Excellence in Space Data and Information Sciences at the Goddard Space Flight Center. Without this backing and infrastructure, TPF-1 would not have been feasible.

7.2 Some High Points

The intellectual content of the TPF-1 workshop will be summarized in the next two subsections. Here, some key consequences of the workshop are highlighted.

The focus on petaflops computing structures did not parallel the balance of the architecture working group at Pasadena but rather found a somewhat distinct focus of its own. Most significant was the area of SIMD. The Pasadena petaflops workshop almost entirely ignored this computing model in preference for an almost exclusively MIMD perspective. More than one presentation at TPF-1 identified the SIMD approach to flow control as a viable way to capture a large class of highly parallel applications and to develop high density architectures towards petaflops capability. While the merging of storage and computing logic on single integrated circuits had been identified at Pasadena, this theme took on a more central role at TPF-1 as more than one paper addressed the means and implications of such technologies. Missing was any indication that very short latency architectures would be feasible. Instead, latency was exposed as the single dominant feasibility issue, with power requirements and cost a close second.

The Petaflops Frontier Workshop provided a forum for individual computational scientists to reflect on their own work and the implications of the potential of petaflops computing for it. Repeatedly, it was shown that science and engineering would be dramatically advanced through the availability of usable petaflops capability. But, the difficulties also were exposed. In many cases, applications do not scale linearly with system size in what can be achieved for the end problem science. As problem size grows, the computational requirements may be compounded to constrain errors as resolution is forced to increase. Other applications have definite limits in data set size and instead require more sequential steps. These benefit from architectures with higher clock speed but only to a certain point from architectures with unbounded parallelism. TPF-1 also directly examined some important non-numeric applications of petaflops computing, including the implications for virtual reality contexts.

The TPF-1 workshop was the setting for the announcement of a new facility to support the emerging petaflops research community. Petaflops Enabling Technologies and Applications (PETA) is an on-line reference index accessible by means of the world wide web. Its principle purpose is to provide a single source of links to the products of the diverse spectrum of research activities that make up the petaflops community. It provides a dynamic intellectual "watering hole" for all relevant work; one-stop shopping for everything you needed to know about the world of petaflops. Peter Kogge and Rick Stevens are Senior Editors with responsibilities in architecture/technologies and applications/algorithms, respectively. PETA can be accessed at the URL: http://cesdis.gsfc.nasa.gov/petaflops/peta.html

In spite of initial concerns, not to mention heating problems on a very cold winter day, The Petaflops Frontier Workshop was, by any measure, a success. Over a hundred colleagues participated and were actively involved. The structure of many short talks kept the meeting lively and provided many interesting concepts, issues, and results. Almost half of the total conference attendance was also at this workshop. All feedback from attendees was very positive and the organizers have been encouraged to host TPF-2 at Frontiers '96.

7.3 Applications and Algorithms

The presentations on applications and algorithms at the Petaflops Frontier Workshop provided ample evidence that petaflops computing cannot come soon enough. A wide spectrum of applications, many dealing with current scientific and engineering problems, *need* petaflops-level performance now. Equally true, once such performance is available, applications that are now just concepts will quickly become mainstays in science, engineering, medicine, pharmacology, education and training, and business.

One key issue for the current and future applications will be the enormous I/O requirements, both in speed and volume. Implicit in this, of course, is the need for data storage approaching exabyte levels. Significant advances in modeling will depend on a computing system's ability to manage (in addition to intensive computation) I/O streams of petabyte data sets per hour or minutes from archives in the exabyte range. Such I/O will demand new or innovative storage techniques as well. While not addressed during the TPF-1 workshop, the report on the first Pasadena petaflops workshop addressed substantive issues of storage technology.

Computational fluid dynamics (CFD), a discipline critical to a wide variety of applications (ranging from aerodynamics to medicine), could easily dominate usage of petaflops (and even exaflops) computational capabilities. Indeed, the computing needs of the CFD community are so extensive that it's likely that, to the extent that any community drives system architecture and algorithm development, the first teraflops and petaflops systems will be designed to satisfy those needs.

One example of the pervasiveness of CFD (provided by Maizel) comes from the field of medicine. Currently only portions of the fluid dynamics of a heartbeat have been modeled successfully. One heartbeat now requires 50M of memory and one cpu-week on a Cray C-90. Developing improved models along with a better understanding of the underlying science, will depend on increased computational power, i.e., moving from gigaflops to teraflops and petaflops.

Not all applications require petaflops-level computing to simply speed up execution time: increased problem scale (time and space) for improved accuracy and precision are important also in such areas as environmental studies and astrophysics. But, petaflops-level hardware must be developed in conjunction with algorithm development that allows the application to take full advantage of system capabilities.

Significant progress in algorithms can free applications programmers from burdensome concerns about system-level performance factors. Certainly this is at least one pressing need (true today—and even more important as the sizes of machines increase) about which all parts of the high performance computing community seem to agree.

As noted by several presenters, most applications could benefit from a heterogeneous computing environment (either mixed-machine or mixed-mode). The SIMD, MIMD, etc. chips and machines exist and some progress has been made in appropriate algorithms for running applications in a heterogeneous environment. But, moving from gigaflops machines to petaflops machines (via, presumably, teraflops machines) will pose a myriad of extremely difficult problems in resource and application management.

The situation presented above applies to applications with which the community has appreciable experience. What about applications that now are just concepts? These "future" applications, as presented by Stevens et al. include such possibilities databases with millions of users on-line simultaneously, off-world repositories of all of humankind's knowledge, and full-scale virtual reality interactive teaching environments. These and applications not yet conceived may be achievable, but not, as mentioned previously, without quantum improvements in algorithms, I/O and storage.

7.4 Architecture and Technology

By the close of the the Petaflop Frontier Workshop it was clear that issues of architectures for petaflops computing are still only broadly defined in terms consistent with the context of "architecture" as it is known today. The nature of an appropriate petaflops architecture (or several architectures) depends too much on technologies that are far from mature (or even technologies that ultimately may not "work"), algorithms that have yet to be developed, and applications that require further understanding with respect to their underlying science and their scalability.

Heterogeneous computing (HC) has been shown to be superior for certain problems classes than a purely vector supercomputer—in some cases up to ten times faster execution. Of the two approaches to HC, mixed-mode offers a finer granularity (instruction level) than mixed-machine.

Mixed-machine on the other hand is evolving more naturally as researchers devise and implement methods for managing the execution of complex applications among different classes of machines: machines that in some instances are geographically widely separated. These endeavors have depended on the availability of hardware and software to manage latency and inter-machine communications, and the development of decomposition methods that allow distributing elements of the problem to machines that can best execute them.

Mixed-mode (i.e., SIMD, MIMD) heterogeneous computing (HC) has been employed successfully by various researchers at Purdue University, the University of Texas at Austin, the University of Nice, France, the University of Karlsruhe, Germany, and by IBM Federal Systems Division. Mixed-mode HC, however, presents appreciably more challenges than mixed-machine HC, not the least of which is incomplete understanding of—if not disagreement about—which, SIMD or MIMD, is the better overall approach to problem solution.

The continuing rivalry between the SIMD and MIMD communities (which has certainly been exacerbated by the zero-sum approach to allocating research dollars) is exemplified by the contrast provided in the article by Dorband, Auburdene et al. Strong proponents of SIMD, they believe that a SIMD machine of 10^6 processors is not only achievable, but desirable. And, indeed, it may be just that. Certainly, one outcome of the research will be a better understanding of how to "manage" very large numbers of processors.

It is an understatement to assert that it is much too early to posit one architecture over another for a petaflops machine. Indeed, the conceptual model offered by Siegel et al. makes a strong case for an "architecture" that provides or makes available a variety of approaches depending on the computational requirements of the application and its subtasks, i.e., a system comprised of both mixed-machine and mixed-mode capabilities. The need for such flexibility is evident from the discussion of various applications in Section 6.

But, whether or not mixed-machine or mixed-mode HC receives the majority of support in the near term, formidable challenges exist in the development of systems that can attain sustained petaflops performance.

Many of the challenges involve conserving memory, reducing latency, and increasing bandwidth. A persistent theme of the workshop was that these issues can and will be addressed, at least in part, by marrying processors and memory, or as Kogge refers to it, processors-in-memory (PIM). Including computing logic in memory takes advantage of the large internal bandwidth and, for local memory purposes, reduces latency to nil. But, for such an architecture to be feasible (from a size and cost perspective), feature size has to be significantly reduced from today's levels while keeping costs reasonable. Reducing the feature size, of course, increases the density, thereby adding to the problem of power dissipation.

Maximum local bandwidth, though, can only reap some finite advantage; chip-to-chip, board-to-board, etc., communication (I/O) must be improved dramatically. Several optical schemes are being investigated that include guided, free-space and three-dimensional methods along with several multiplexing approaches (wavelength division, space division, frequency division), and concurrent read, concurrent write methods. But, to move from the realm of the theoretical, simulation or brassboard will necessitate considerable progress in hardware development. And, as Chunming Qiao points out, most of the investment in optical technologies has been from the telecommunications industries, not computing.

Noticeably absent from the workshop were any presentations on superconductivity technologies which received some attention at the first Pasadena petaflops workshop. This technology although offering considerable promise, also presents formidable obstacles to practical implementation, particularly in the interface to room temperature components. Indeed, in Pasadena I, the interface to room temperature was discussed in terms of the very optical technologies that this workshop showed need much more development. The need for optical development and low funding superconductivity technology indicate that for the time being architecture designers will reap better payoffs by focusing on processor-in-memory developments.

7.5 Implications for Future Directions

The principle contributions of The Petaflops Frontier Workshop are first, that it has increased the petaflops community knowledge and understanding of the field and its opportunities in precisely the way intended, and second, it has broadened the community by providing an open forum for a wider range of disciplines and increasing the involvement of researchers with relevant interests and contributions. But this is a community and a discipline that is inventing itself and the question remains as to its immediate future directions.

While TPF-1 provided new data related to application scaling and architecture approaches, it did not come near to completing the task. Therefore, these two goals must be pursued. The first is to be addressed at the Petaflops Summer Workshop at Bodega Bay, California in August 1995. This work is also being pursued by a small group of computational scientists in the domain of Earth and space science. The objectives of these studies are to (1) identify problems in science, engineering, and commerce that demand petaflops scale computing capability, and (2) determine the balance of resource demand by such applications on the petaflops computer system architectures.

The second goal, architecture, is being pursued, in part, by small sponsored studies to flesh out key architectures that show promise of feasibility. Among those for which on-going work is being conducted are Processor-in-Memory architectures, ensembles of SMP architectures, and Mixed technology, multi-threaded architecture. The intent of these studies is to expose the details and implications of the approaches both for applications programmers and technologists. The results of this work will be a basis for the Second Pasadena Workshop on Enabling Technologies for Petaflops Computing to be conducted in the Spring of 1996.

But an even broader range of approaches is possible and represented by current research not yet formally associated with the petaflops community. The approaches include a class of methods that could be referred to as "metaphorical computing" that employ various physical phenomena and isomorphic mapping of parameters between them and the computing problem to be solved. These are not von Neumann computer architectures, and in fact, usually not digital. One example is the old method of solving sets of first order differential equations using electronic analog computers. A more recent example is molecular computing where natural processes of organic molecules can be used to solve compute intensive applications such as the traveling salesman problem. This highly disparate work needs to be connected to the petaflops community so that adequate appreciation of their potential can be achieved in terms of petaflops-scale computing and applications requirements.

While technology was addressed, by chance, there were no formal presentations related to superconducting technologies. There were participants at TPF-1 from that community, however. Because of the perceived difficulties in using cryogenic technology and its importance for low power computation, a concerted effort must be undertaken to embrace that research community and bring out the results of its continuing advances.

It was felt by many that The Petaflops Frontier Workshop was an important component of the 1995 Frontiers of Massively Parallel Processing conference and that TPF-2 should be organized for Frontiers '96 with additional activities such as a panel session on the topic also conducted for the conference. This is being pursued by the conference program committee and will occur in Annapolis, Maryland in October of 1996.

Finally, PETA, the on-line reference index for the petaflops research community has been implemented and is accessible via the world wide web. Ideally, all relevant work will be linked to this site with adequate annotation so that anyone interested in this area has an immediate reference source. PETA is growing rapidly, but new solicitations and contributions are required on a sustained basis in order to achieve its objectives.