High End Computing Technologies for Earth Science Applications: Trends, Challenges, and Innovations

Rupak Biswas, Jerry C. Yan, Walter F. Brooks

NASA Ames Research Center, Moffett Field, CA 94035

Thomas L. Sterling

NASA Jet Propulsion Laboratory, Pasadena, CA 91109

Abstract

Earth science applications of the future will stress the capabilities of even the highest performance supercomputers in the areas of raw compute power, mass storage management, and software environments. These NASA mission critical problems demand usable multi-petaflops and exabyte-scale systems to fully realize their science goals. With an exciting vision of the technologies needed, NASA has established a comprehensive program of advanced research in computer architecture, software tools, and device technology to ensure that, in partnership with US industry, it can meet these demanding requirements with reliable, cost effective, and usable ultra-scale systems. NASA will exploit, explore, and influence emerging high end computing architectures and technologies to accelerate the next generation of engineering, operations, and discovery processes for NASA Enterprises. This article captures this vision and describes the concepts, accomplishments, and the potential payoff of the key thrusts that will help meet the computational challenges in Earth science applications.

1. ESE Computational Technology Requirements

The mission of NASA's Earth Science Enterprise (ESE) is to understand the total Earth system from the vantage point of space, and the effects of natural and human-induced changes on the global environment. This will enable improved predictions of climate, weather, and natural hazards (the three specific disciplines within ESE) for present and future generations. However, in order to achieve this goal, Earth science applications of the future will stress the capabilities and capacities of even the highest performance supercomputers in the areas of raw compute power, throughput, mass storage management, and software environments. For example, the computing and storage requirements for a "sensorweb" of several EOS satellites will be enormous: handling petabytes of data per day. Advances in spaceborne computing capabilities will be necessary to enable on-board data processing and data compression. NASA will also need advances in software environments to allow on-ground high performance supercomputers to run coupled Earth system models, for example, to forecast weather to the theoretical limits of prediction.

1.1. Weather Forecasting

Let us look at the high end computing (HEC) requirements for each of the three ESE disciplines. In weather forecasting, we are currently able to provide 3-day forecasts with 93% accuracy and 7-day forecasts with only 62% accuracy. The ultimate goal is to provide 14-day weather predictions (theoretical limit) with more than 90% accuracy and understand its effects on the environment. The computational requirements for assimilating massive data volumes ($>10^{11}$ observations per day) in real-time could easily be more than 10^6 times the current capabilities. This requires orders of magnitude increases in raw and sustained computational power as well as in data handling capabilities; however, new filtering strategies will also be necessary to compact the information before data assimilation. The report from the ESE Computational Technology Requirements Workshop contains extrapolated estimates for computing, networking, and data storage in year 2010. The current capabilities and future requirements for weather forecasting are reproduced in Table 1.

Table 1: Current and future (in 2010) computational technology requirements for weather forecasting

	Current	In 2010
Capability Computing	10 Gflops	20–50 Tflops
Capacity Computing	100 Gflops	400–1000 Tflops
Input Data Volume (observations)	400 MB/day	1 TB/day
Output Data Volume (gridded)	2 TB/day	10 PB/day
Archival Storage	1 TB/day	10 PB/day
Internal Data Movement	4 TB/day	20 PB/day
External Data Movement	5 GB/day	10 TB/day

1.2. Climate Modeling

Climate research also relies heavily on HEC resources. For example, current seasonal-to-interannual (S-I) prediction systems are useful for climate applications if single image model performance is at least 1000 d/d (simulated days per wall-clock day). In addition, since climate forecasts are based on ensembles of runs, an aggregate throughput of 20,000-30,000 d/d is actually required. However, even if these performance goals remain steady over the next 10 years, the computational requirements will increase due to greater model complexity and higher resolution simulations. The requirements for decade-to-century predictions are similar, although more substantial in capacity computing because of increased complexity from the inclusion of atmospheric chemistry and resolution of the stratosphere. For instance, a single image of an atmospheric configuration of 1° resolution with 100 layers (including the stratosphere) and 40 on-line chemical tracers would require a throughput of about 5 Tflops; with a coupled ocean of 50 layers and 0.5° resolution, the requirement is 6.6 Tflops. The coupled model configuration with a job mix equivalent to 100 concurrent images (ensembles, parameter sensitivity sweeps) would then require 660 Tflops. The input and output data volume for S-I prediction is estimated to be 10 GB/day and 100 TB/day, respectively.

1.3. Solid Earth Science

Even though the area of solid Earth science is currently mission and data poor, advances in space technologies will enable new measurements resulting in a better understanding of complex, interacting solid Earth processes. High-resolution predictive models for forecasting natural hazards will have to be developed that process global observation streams in real time. For instance, radar interferometry missions such as ECHO will generate 100 GB of data per day that will have to be fully ingested for forecasting seismic activity, and require a sustained performance of 2 Tflops. Other sub-fields of solid Earth (e.g. volcanoes, magnetic fields, vegetation, ice modeling) have similar HEC requirements. An estimated capacity requirement is 100 Tflops and 10 PB/day of output data.

1.4. Capability and Capacity Needs

Overall, the sustained capability-computing requirement is a few teraflops for all three disciplines within ESE, and is primarily driven by model resolutions. Capacity requirements are significantly higher, but can be provided by a geographically distributed environment. However, the tradeoff between tightly-coupled systems capable of 100 Tflops sustained performance and loosely-coupled distributed systems with 1 Pflops throughput capacity is ESE discipline specific. For example, the latter platform may be sufficient for solid Earth applications but will create a data transport bottleneck for weather prediction codes.

2. HEC Technology Trends and Limitations

To meet these ambitious ESE application goals, significant improvements are required in NASA's ability to create, process, understand, store, and communicate data. This implies that dramatic advances are required in a wide spectrum of HEC technologies, including computer system architectures, storage technologies, and software environments. In this article, we will discuss current industry trends, research challenges, and innovations in each of these areas. However, to satisfy ESE requirements, major progress are also necessary in other arenas such as simulation models (greater complexity, higher resolution, multi-component coupling, etc.), algorithms (proper choice of algorithms, design of new architecture-appropriate new algorithms, data compression techniques, etc.), and data analysis and visualization (ability to handle extremely large data sets, real-time computational steering, knowledge extraction, etc.). These latter areas will not be addressed in this article.

2.1. Commodity MPP and Clusters

The dramatic advances in HEC performance achieved within the last decade are a product of semiconductor technology improvements (attributed to Moore's Law) and the exploitation of commodity devices in massively parallel processing (MPP) and cluster (e.g., Beowulf) system architectures. As reflected by the TOP-500 list of the world's fastest computers (www.top500.org), the overall rate of performance gain in the last ten years has been approximately 600X (80% average annual improvement). This sustained growth was derived from a combination of advances in processor architecture, clock speed, system scale, and software technology. The rapidly increasing peak performance and generality of superscalar cache-based microprocessors long led researchers to believe that competing architectures hold little promise for future large-scale computing systems. Due to their cost effectiveness, an ever-growing fraction of today's supercomputers employ commodity superscalar processors, arranged as systems of interconnected symmetric multiprocessor (SMP) nodes.

The integration of commodity off-the-shelf (COTS) components such as CPUs and memory in MPPs has provided good performance-to-cost ratios by exploiting the investment and advances made in complex semiconductor devices for much broader markets than that of the supercomputing arena. Commodity clusters have pushed the cost benefits even further by integrating full systems into cooperative ensembles via dedicated system area networks (SAN) for minimum hardware development time and maximum resource reuse (in both hardware and software). Conventional wisdom asserts that the COTS-based cluster and MPP strategy will enable petaflops-scale performance by 2010. Thus, according to this reasoning, the ESE application requirements will be fully satisfied by future systems evolved incrementally through this strategy.

2.2. Vector Processors and the Earth Simulator

There are two major factors that counter this perspective. First, conventional MPPs and commodity clusters deliver low sustained performance with respect to peak for many applications and are difficult to program, debug, and optimize. Second, a number of important opportunities are offered by innovative computer architectures that could achieve higher fractions of peak performance but are unavailable to scientists because of the constraint to use only conventional technologies. In short, supercomputers should be designed with HEC requirements and methods in mind to achieve the best in capabilities, capacities, and usability.

Perhaps a dramatic example demonstrating this view is the Japanese supercomputer referred to as the Earth Simulator (www.es.jamstec.go.jp). This system, based on NEC SX6 technology, comprises of 5120 vector processors (640 8-way nodes), each capable of 8 Gflops, and interconnected by a high-bandwidth low-latency 640x640 crossbar network. It is the fastest computer in the world with a LINPACK performance of 35.8 Tflops (five times the performance with half the number of processors of the IBM SP-based ASCI White, built using superscalar technology). More important than peak performance, however, is what this new capability entails for scientific communities that rely on modeling and simulation. For instance, the Earth Simulator was developed to provide unprecedented capabilities for solving Earth science applications including climate modeling, seismology, biosphere, plate tectonics, magnetosphere modeling, and planet formation. Its processors and network were specifically designed for supercomputing and the exceptional delivered performance (almost 65% of peak) on realistic applications demonstrates the effectiveness of this strategy. Of the Gordon Bell Prizes awarded at SC2002, scientists employing the Earth Simulator received the majority of the prizes (Sterling was Chair of the 2002 Gordon Bell Prize Committee). Note that other parallel vector systems, such as the Cray X1, also offer the potential to bridge the gap between sustained and peak performance for a significant number of scientific codes, and to increase computational power substantially.

2.3. Conventional System Architectures

In spite of dramatic performance gains demonstrated by conventional HEC systems that exploit commodity hardware, they have exhibited poor operational properties in many cases. It is certainly true that there are specific applications that run extremely well on such platforms (delivering better than 50% of peak); however, they are embarrassingly parallel requiring only limited inter-node communication and synchronization, have good memory access patterns yielding high cache hit rates, and are static and regular so that load balancing is fairly straightforward. But many important applications, including those from Earth science, are not so favorably structured; therefore conventional systems are typically ill equipped to efficiently handle their requirements.

Current HEC systems suffer from a number of limitations due to their architectural inadequacies. There are several areas of concern that contribute to the degradation of sustained system performance. These include memory access latency, inter-processor communication latency, overhead of managing parallel resources and tasks, contention for shared resources including networks and memory banks, and absence of sufficient concurrency to sustain useful throughput of critical resources. For instance, conventional microprocessors rely almost exclusively on cache hierarchies and temporal locality to avoid memory access latency (out-of-order execution and prefetching are sometimes used for latency hiding). There is no hardware support for hiding remote access request latencies across distributed systems, although some software techniques for locality management attempt to mitigate such latencies. Current architectures incorporate almost no mechanisms for supporting multiprocessor parallelism, except for cache coherence across a small number of processors within an SMP. There is usually some instruction set support for synchronization to control parallel execution; a software implementation can be very inefficient except for coarse-grained parallelism. Finally, two critical bottlenecks routinely experienced by conventional HEC systems are memory banks and interconnection networks (MPPs and clusters merely differ in the amount of bisection bandwidth that is provided).

2.4. Distributed Grid Computing

One possible mechanism to satisfy the capacity computing requirements of future HEC applications would be to use distributed grid technology [1]. Grids could also be utilized to provide seamless uniform access to federated databases, which are particularly important for ESE applications. In the last five years, computational grids have attracted a lot of worldwide interest, from both government agencies and computer industries, as a viable means to meet a variety of resource requirements.

The NASA Grid is one of many such grid testbeds, and is designed to ubiquitously harness the power of geographically distributed resources, many of which are specialized and cannot be replicated at all agency sites. It involves linking NASA's vast collection of heterogeneous and distributed computers, data archives, and scientific instruments to create a scalable, adaptive, robust, and transparent metacomputing environment. The interface to the NASA Grid will hide details of resource characteristics, such as location, nature, connectivity, and name, thereby presenting users with seamless access to these resources. As a valuable by-product, it will also enable remote collaboration among users to more easily model, simulate, and analyze large-scale realistic scientific problems in support of mission goals. It will therefore allow scientists and engineers to focus on making new discoveries in science rather than on the details of using specific hardware, software, and information resources. However, grid technology is still somewhat in its infancy, and significant strides in middleware, high-level services, information management systems, and application environments are necessary to realize its promised objectives. To that end, the Global Grid Forum co-founded by NASA is an international organization with the charter to standardize the different grid access and usage protocols (www.ggf.org). We will not discuss grid technology trends and challenges in this article.

2.5. Software Environments

We can think of innovative HEC system architectures as a means to improve computational productivity by increasing both peak and sustained performance. Computational productivity is also improved by the effective choice and/or the proper design of algorithms for these new architectures. But human productivity must also be enhanced if scientists are to routinely and efficiently utilize the petaflops-scale supercomputers of the future. In other words, the programmability of computer architectures must be improved to increase their utilization. For example, novel programming techniques must be developed for efficient and economical representation of algorithms and their optimal mappings to target architectures. This includes programming paradigms (languages, libraries, compilers, etc.) as well as user tools (parallelization, performance analysis, debugging, etc.) with the goal to improve parallel performance, scalability, portability, and interoperability. In addition, problem solving environments (PSEs) are necessary to make application scientists more productive. This encompasses the areas of workflow definition, specification, and management; data analysis and knowledge discovery; and application portals and graphical user interfaces (GUIs).

Apart from such programming tools and information environments, dramatic innovations in runtime execution systems (e.g. smart operating systems, dynamic adaptive environments to support cross-discipline interactions) are also required to harness the full potential of the highest performance supercomputers. Current runtime environments will be unable to handle the complication of managing very large ensembles of processing elements and overcome the generally costly service times routinely imposed by typical operating system calls. For example, Pthreads, the Unix mechanism for managing parallelism can take a long time to instantiate a new process while petaflops-scale machines will certainly require almost cycle-by-cycle context switching. In other words, to positively impact future NASA Earth science missions, we need flexible implementations of critical algorithms (e.g. fault-tolerant, latency-tolerant, steerable, adaptive, and incorporation of data provenance and uncertainty), coupled with relevant HEC technologies for formulating, mapping, executing, and managing them on new architectures.

3. Innovations in HEC Architecture

Thanks to improved VLSI technology, there are many opportunities to overcome the deficiencies of conventional HEC architectures through innovative system design. Let us look at some possibilities to improve sustained performance, while also increasing peak performance. Where once the arithmetic and logic unit (ALU) was considered the most precious resource of a computer system, it now occupies only a small fraction of the microprocessor's area and consumes little power. Thus there is a tremendous potential for increasing system performance by replicating ALUs, assuming that they can be controlled in a coordinated manner and kept usefully busy. It also implies that ALUs can be inserted into structures to more effectively utilize resource-critical units such as memory banks, thereby greatly increasing throughput and reducing latency. Another legacy is the separation between processing and memory. Historically, the technologies and optimizations for logic and memory were different. However, current fabrication processes are capable of implementing both DRAM memory cells and CMOS logic on the same die, enabling new classes of computing structures. Current VLSI technology can also achieve a high level of device density, allowing very complex structures. Multiple processors can therefore reside on a single die, delivering performance gains proportional to the increase in devices on the chip. Finally, some basic hardware mechanisms (e.g. Intel hyper-threading) are now available to support multithreading for rapid context switching and latency hiding.

In response to this multitude of opportunities resulting from technology advances, new directions in HEC architecture are being explored. Three broad classes of innovative architectures (streaming, HTMT, and Gilgamesh) are derived in part from the ability to incorporate many ALUs and their control units on a die. In this section, we introduce these areas of new work and then describe specific architecture research at NASA that will enable efficient petaflops-scale processing.

3.1. Logic Intensive

A new class of processor design that might be called Logic Intensive Computer Architecture (LICA) is being pursued in different forms to exploit the opportunity to include many ALUs within a single microprocessor. With LICA structures, many data elements are simultaneously directed through multiple layers or arrays of arithmetic units, sometimes moving between successive ALUs without making intermediate stops at the general register set. For calculations that exhibit a large number of operations per unit data access from memory as might occur with many digital signal processing and graphics rendering problems, the performance benefits may be substantial. Two such projects include the Stanford Streaming Supercomputer (SSS) at Stanford and the Tera-op Reliable Intelligently adaptive Processing System (TRIPS) at UT Austin.

The streaming processor being developed at Stanford contains tens to hundreds of ALUs on each chip with performance achieved through pipelining, fused computations, and stream parallelism [2]. The SSS stream model exposes parallelism and locality in applications by passing streams of records (data) through computational kernels. Data parallelism is obtained across stream elements while pipeline (functional) parallelism exists across kernels. In addition to spatial and temporal locality within each kernel, a producer-consumer locality exists between neighboring kernels. This is achieved by avoiding unneeded memory traffic associated with intermediate loads and stores to main memory. A stream processor can therefore be simplistically viewed as a vector processor with local registers. The overall goal of the SSS project is to achieve a cost/performance ratio of 100X better than conventional MPP and cluster-based supercomputers on both arithmetic and memory bandwidth limited computations.

The goal of the UT Austin TRIPS project is to develop a computer architecture that achieves single-chip teraops performance and keeps pace with advances in semiconductor technology [3]. It uses a novel processor organization called Grid Processor Architectures (GPAs) that is composed of a tightly-coupled array of ALUs, each with limited control and connected via a thin network. Programs are run by mapping large blocks of statically-scheduled instructions onto the GPA and then executing them dynamically in dataflow order. To hide on-chip communication latencies, instructions are scheduled so that their critical dataflow paths lie along neighboring (or at least nearby) ALUs. The strategy is similar to that of the SSS where instruction blocks are executed on chains of ALUs without transmitting temporaries back to the registers, thereby avoiding situations that limit the scalability of conventional architectures. TRIPS is also reconfigurable to efficiently satisfy a range of possible workloads (e.g. control-bound integer codes, parallel threaded codes, compute-bound streaming codes) via on-chip sensors and a lightweight software layer called morphware.

But not all data has high temporal locality. For many applications, large sparse and time varying data sets govern the form and function of the application. The manipulation of metadata requires very little processing on a per word basis and exhibits poor locality. This, in turn, results in poor cache behavior and low processor utilization. The opportunity to employ many ALUs on a single chip and merge with DRAM memory on a single chip has resulted in another class of innovative architecture: processor-in-memory (PIM). PIM is one of the most important advanced device technologies being pursued by NASA as part of the HTMT and Gilgamesh architectures.

3.2. Hybrid Technology Multithreading

Historically, computer performance has grown in response to advances in device technology, computer architecture, programming methods and tools, and application algorithms. The Hybrid Technology Multithreading (HTMT) architecture project involved more than a dozen government, industry, and academic institutions, and sponsorship of four government agencies to explore the frontier of computer design. HTMT pushed several dimensions of system design and operation to accelerate and explore multiple high-risk, high-payoff opportunities to realize practical effective petaflops-scale computation [4]. The result of the three-and-a-half year project was to demonstrate the feasibility of implementing a petaflops-scale computer with size and power requirements less than that of conventional systems delivering one one-hundredth the performance. The HTMT structural diagram is shown in Figure 1.

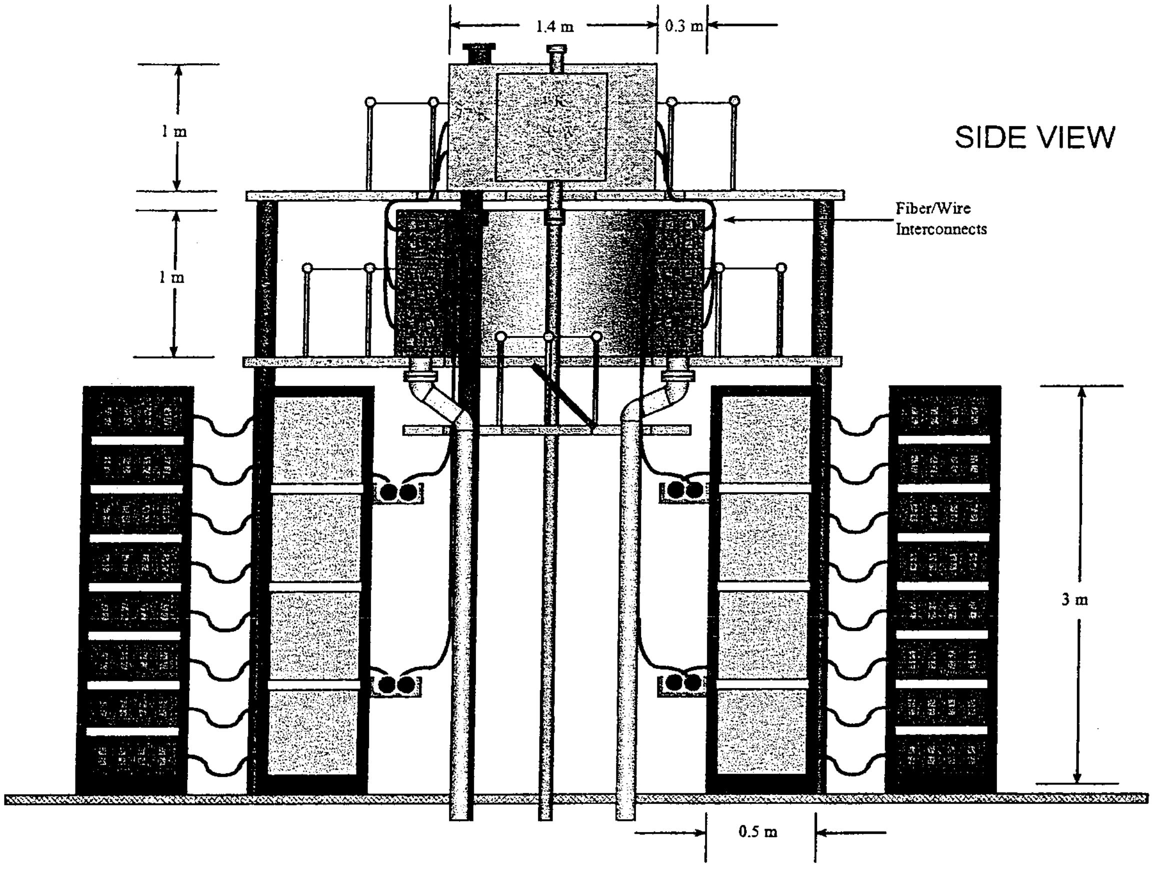

Figure 1: HTMT structural diagram

3.2.1. Advanced Device Technologies

The device technologies explored and developed as part of the HTMT architecture project are described below.

<u>Superconductor RSFQ Logic</u>

Superconductor components have enabled the fabrication of the highest speed logic of any device technology. Clock rates of over 770 GHz have been demonstrated in the laboratory (SUNY Stony Brook) using rapid single flux quantum (RSFQ) gates implemented with Josephson Junctions fabricated as Niobium on Silicon and cooled to 4 Kelvin. Also significant is that power dissipation per gate of approximately 0.1 microwatts can be three orders of magnitude less than that of CMOS gates running two orders of magnitude slower. The challenges of using RSFQ include data communication among components and the storage of temporary data in registers and cache. At an assumed processor clock rate of 100 GHz, signal propagation is less than a millimeter per cycle, requiring novel processor designs. It has been shown that data transfers between chips on a shared MCM substrate can operate at 30 Gb/sec point-to-point.

Optical Networking

Petaflops-scale architecture bisection bandwidth requirements range from 10^{15} to 10^{17} bits/sec, depending on underlying assumptions about system architecture and program locality. It could take tens of millions of wires to provide all of the global communication bandwidth required by such systems. Using modulated lasers, digital light signals can be operated at up to 10 Gb/sec per channel. Wave Division Multiplexing (WDM) may combine hundreds of separate wavelengths, enabling a communication medium capable of multiple Tb/sec per channel. Switching rates of 50 MHz is currently possible, with rates of 1 GHz using experimental devices a future prospect. This optical communication technology was developed at Princeton University with many of the required devices demonstrated on the bench. An innovative network topology referred to as the "Data Vortex" was devised at IDA to provide a very high degree interconnect with dynamic routing using optical butterfly switches. The Data Vortex appears as a series of concentric network cylinders, each similar to an Omega network with radial inward directed paths. Data enters the external surface and departs from the internal concentric cylinder. This topology exhibits exceptional properties of sustained bandwidth and latency, even under heavy loads.

Holographic Storage

Memory density, access time, bandwidth, and power consumption are critical in determining the sustained performance, generality, power, size, and cost of a computer system. The possibility of improved memory motivated an exploration of optical holographic storage based on photorefractive physics at Caltech and IBM Almaden. Photorefractive materials retain a holographic image of a page of binary data, of approximately a million bits, typically storing it in a small 1 cc cube of optically sensitive material. The entire page can be read out in a single cycle, providing very high data bandwidth. Multiple pages can be stored in the same device by altering the laser angle of incidence or aiming it at a different spatial position in the storage medium. Power consumption is limited to the single laser and data lifetimes approach many hours, minimizing the need for refresh.

Processor-in-Memory

Advances in semiconductor technology permit the merger of both DRAM cells and CMOS logic on the same die, manufactured together with the same fabrication process, enabling a number of new structures including processor-in-memory (PIM). PIM exploits the proximity of logic with memory to expose the entire row buffer directly to processing logic and permits a high degree of memory partitioning on the chip to greatly increase available memory bandwidth while reducing access latency and power consumption. All of these characteristics offer the potential for a new generation of highly efficient computing and the opportunity for employing smart memories in computer systems to effectively process data intensive computations with low temporal locality.

3.2.2. HTMT System Architecture

The HTMT architecture, shown in Figure 2, incorporates two major subsystems: the superconductor processor core and two-level smart memory system. An important revolutionary feature of HTMT is that the memory is smart while the processors are relatively primitive (conventional systems operate with smart processors and dumb memory). In other words, the memory system controls the processors. The interface between the two subsystems is via a high speed SRAM PIM layer that acts as a global L3 cache. This layer feeds the instructions and data to the processors, performing load balancing and all of the data gather and write-back scatter operations to the memory. The 4096 superconductor processors are multithreaded to hide internal propagation delays through the execution units and the latency to the SRAM PIM layer.

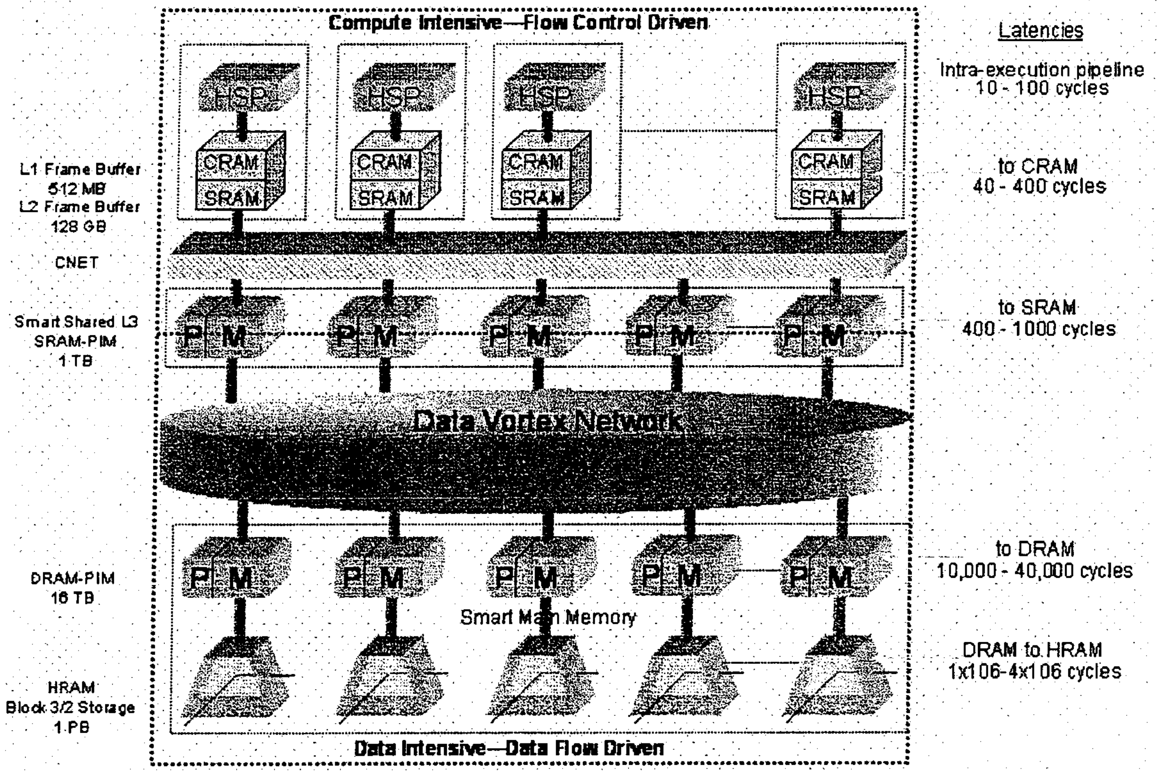
Compute Intensive—Flow Control Driven
HSP
CRAM
SRAM
L1 Frame Buffer 512 MB
L2 Frame Buffer 128 GB
CNET
Smart Shared L3 SRAM-PIM 1 TB
P M
Data Vortex Network
DRAM-PIM 16 TB
Smart Main Memory
HRAM Block 3/2 Storage 1 PB
Data Intensive—Data Flow Driven
Latencies
Intra-execution pipeline 10 - 100 cycles
to CRAM 40 - 400 cycles
to SRAM 400 - 1000 cycles
to DRAM 10,000 - 40,000 cycles
DRAM to HRAM 1x106-4x106 cycles

Figure 2: HTMT system architecture

The main memory system consists of large DRAM PIM with the optical holographic devices serving as a high-speed backing-store (3/2 memory). All PIM components, both DRAM and SRAM, are interconnected by the Data Vortex optical communication network, which also provides I/O ports to external support including mass storage and user interfaces. Each DRAM PIM chip is subdivided between 16 and 64 processor node pairs. In ensemble, the main memory system in cooperation with the SRAM PIM layer conduct a strategy of data and task prestaging (referred to as "percolation") in which the flow control of the program execution is managed by the memory. As functions are ready to be executed, all of the required information is migrated to the SRAM PIM layer without intervention by or intrusion of the superconductor processors. Once there, the SRAM PIMs feed the work to the ultra high-speed processors, which perform only those operations required for the actual execution of the application, leaving all of the overhead tasks to the PIMs. Thus, the major sources of performance degradation, including latency, contention, overhead, and starvation are managed by the low-cost PIMs, ensuring excellent efficiency of the high-cost superconductor processors.

3.2.3. Summary Findings

- *Performance:* HTMT demonstrated that a petaflops-scale computer could be implemented with technologies either already tested in the laboratory or for which there was a clear development path. Through incremental advances and the scaling properties of the architecture, it was anticipated that future generations of HTMT could realize an overall performance approaching 100 Pflops before intrinsic bottlenecks and fundamental technology limitations inhibited further extensions of its processing capability.
- *Cost:* While it would be impossible to put a dollar value on the replacement price of an HTMT machine, one important metric is parts complexity, i.e. the number of distinct devices with which it would be assembled. The total number of chips was about 200,000 that, while large, is certainly manageable.
- *Space:* The core system itself, excluding support equipment and mass storage, was only 400 square feet; ten times less than a 1 Tflops conventional system which delivers only a thousandth the performance. A more honest number including all mass storage, power conditioning, and cooling equipment was 12,000 square feet, including 20 PB of hard disks.
- *Power:* The core computer would consume 500 watts internally, and 100 KW after cooling to 4 Kelvin. The additional power required for the optical network, high-speed semiconductor memory, PIM DRAM, and holographic storage would consume less than 1 MW (about that of the ASCI Red machine which has one thousandth the computing capability).
- *Efficiency:* HTMT incorporated mechanisms that performed all of the overhead in the pervasive and cheap PIM processors, executed almost all zero locality data oriented operations directly in the memory by the PIM processors, and allowed the compute processors to work exclusively out of high-speed SRAM. It thus avoiding the latency to main memory by prestaging all data associated with a task to be performed in the SRAM by means of the PIM processors prior to the RSFQ processors engaging in the work.
- *Programmability:* HTMT demonstrated that shared memory, multithreaded parallelism, hardware supported latency management, and automatic

scheduling and load balancing eliminates many of the critical aspects of programming conventional parallel computers that make the task so arduous.

3.3. Gilgamesh MIND

Gilgamesh (Billions (Giga) of Logic-Gate Assemblies with MESH integration) is an advanced scalable computer architecture for both ground-based and spaceborne HEC systems based on an innovative PIM design [5,6]. Gilgamesh is a computing system that integrates the MIND (Memory, Intelligence, and Network Device) PIM architecture in potentially large arrays for a continuum of computational capabilities from tens of gigaflops to multiple petaflops. Gilgamesh systems may be used in a standalone structure (sea of PIMs) or in combination with high-speed external processors and backing storage components.

MIND supports a virtual name space with multiple concurrent contexts. A distributed address translation method and underlying mechanisms allow virtual pages of data to be efficiently located within the distributed physical memory while supporting virtual page migration for active locality management. MIND supports message-driven computation using a class of active message called parcels (PARellel Communication ELements). A parcel can invoke remote tasks (those on another MIND chip) as simple as a memory read or as complex as instantiating a new object and set of functions. Parcels permit work to be moved to the data when it is more effective rather than the conventional strategy of always moving data to work. MIND supports multithreading for dynamic adaptive management of local (on-chip) resources and latency hiding. Multithreading proves to be a powerful unifying mechanism that greatly simplifies the control logic of the MIND processors while achieving greater utilization of the memory and I/O pin bandwidths.

3.3.1. Features of PIM technology

PIM was introduced briefly in Section 3.2.1. The ability to co-locate and integrate CMOS logic and DRAM cell arrays on the same die provides the potential for an unprecedented degree of coupling between these two historically segregated digital devices. A number of advantages compared to conventional practices are implied by this new strategy to devising digital structures; to what degree they are exploited depends on the specific architecture devised and the operational execution model employed. The salient features of PIM technology are:

1. Dramatic increase in available memory bandwidth,
2. Significant reduction in memory access latency,
3. Efficient operation in the absence of temporal and spatial locality of memory access,
4. Substantial reduction in gate count per processor,
5. Very low power consumption per operation,
6. Major decrease in system size and weight,
7. Opportunity for high availability through replication, and
8. Technology tracking by optimally using Moore's Law.

3.3.2. Gilgamesh System Architecture

The Gilgamesh system architecture exhibits a hierarchical structure of functional elements and their interconnection. Different implementations may vary dramatically in their actual structure depending on scale, functionality, and relationship to other elements of the global system in which they are embedded. Nonetheless, all Gilgamesh systems may be devised within a three-level framework. At the system level, the Gilgamesh architecture is defined in terms of the number of MIND modules employed, their interconnect topology and network components, and the external devices attached to it. At the MIND module level, the MIND chip has an internal structure that includes memory, processing, and communication functionality, and is capable of fully independent operation or as a cooperating element in a highly parallel structure. At the MIND node architecture level, all the functionality required to perform core calculations and manage physical and logical resources is incorporated.

3.3.3. MIND Module Structure

The MIND module (usually a single chip) is designed to serve both as a complete stand-alone computational element and as a component in synergistic cooperation with other like modules. The MIND module subsystems are therefore devised to support both its internal functionality and its cooperative relationship. The major functional elements of the MIND module are shown in Figure 3 and described below.

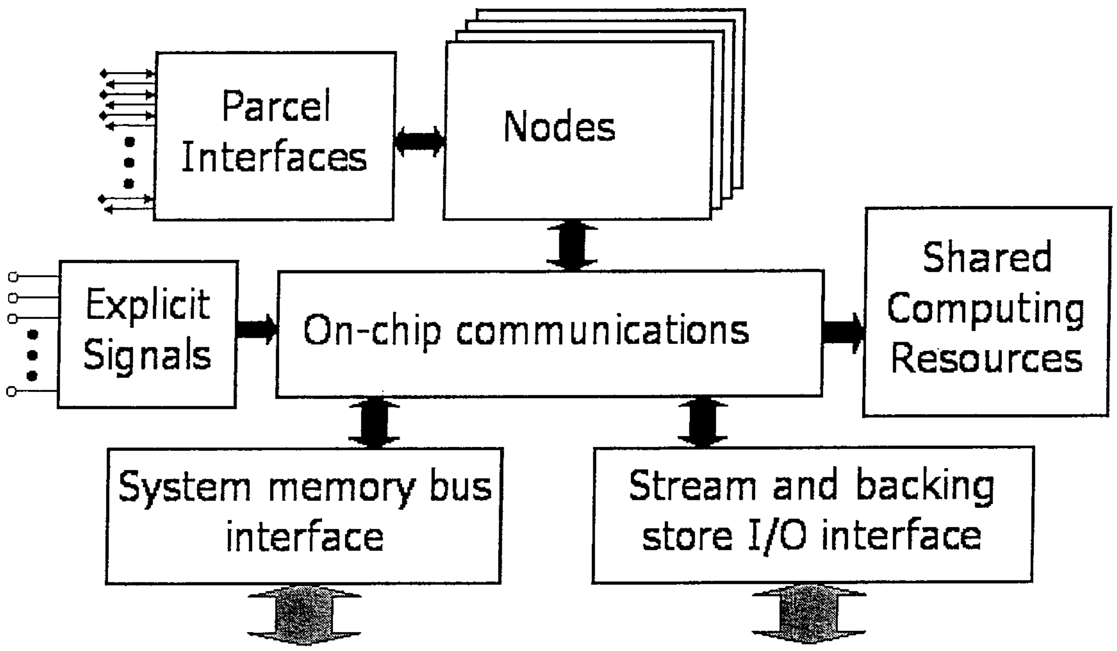
Parcel Interfaces
Nodes
Shared Computing Resources
Explicit Signals
On-chip communications
System memory bus interface
Stream and backing store I/O interface

Figure 3: MIND module organization

MIND Nodes

The MIND node is the principal execution unit of the MIND architecture. Multiple nodes are incorporated on a single chip; the exact number is dictated by fabrication technology, chip real estate, and various design considerations (e.g. the number of gates per node). The node consists of the node memory block, the wide-word multithreaded processor, and connections to the parcel message interface and the MIND chip internal bus.

Shared Function Units

MIND provides the necessary logical and physical infrastructure to permit the addition of separate functional units that can be accessed by all MIND nodes as well as through a master-slave external interface. These can be pipelined, thereby supporting multiple requests concurrently and have their own dedicated access arbitration controllers such as vector floating point multiply-add functional units.

Internal Shared Communications

The majority of node operations employ local resources, but some operational functionality is provided through subsystems on the MIND module but external to the specific node. Some examples are the shared function units and the external interfaces. Another important resource to which every node must have access is the combined memory blocks of the other nodes on the same MIND module. To support the sharing of function units, control of external interfaces, and access to chip-wide memory blocks, an internal shared communication mechanism is incorporated as part of every MIND module.

Master-Slave External Interface

A Gilgamesh ensemble of MIND modules may operate as an independent system or cooperate with other computing elements. The MIND chip therefore incorporates an external interface that services the necessary communications, command, and control functions for interoperability with these remote components (not including other MIND chips). One or a collection of MIND modules may be slaved and responsive to the commands of one or more external master microprocessors.

Streaming I/O External Interface

The external streaming interface provides a direct high bandwidth connection between external remote devices and the MIND memory blocks. The interface can support full direct memory access (DMA) rate of data transfer in or out of the chip. It can be used for such input devices as real-time digital cameras or output stereoscopic projectors at full frame rate. Using this interface, MIND units can be used as post sensor processors for digital signal processing tasks such as passive sonar or radar return data.

Parcel Interface

Inter-MIND chip communication is supported by the parcel packet transport layer. Each MIND chip includes multiple parcel interfaces to an external network providing access to all MIND nodes comprising a Gilgamesh system. Parcels support variable format packets for a wide array of sophisticated remote operation invocation. The parcel interface is capable of interpreting basic operations to perform the simplest tasks without demanding full thread scheduler support or can invoke a new thread for general software controlled function execution.

3.3.4. MIND Node Architecture

The MIND node provides the primary storage and operational resources of a Gilgamesh system. It manages the DRAM main memory, providing both local and remote access to its stored data. It also performs basic operations on multiple fields of data simultaneously. It initiates fine grain tasks, carries them out, and completes them, interleaving operations from separate but concurrent tasks to achieve high efficiency through hardware supported multithreading. The node assimilates parcels and instantiates new tasks in response, as well as generates outgoing parcels. Figure 4 shows a block diagram of the MIND node architecture. Each important component is described below.

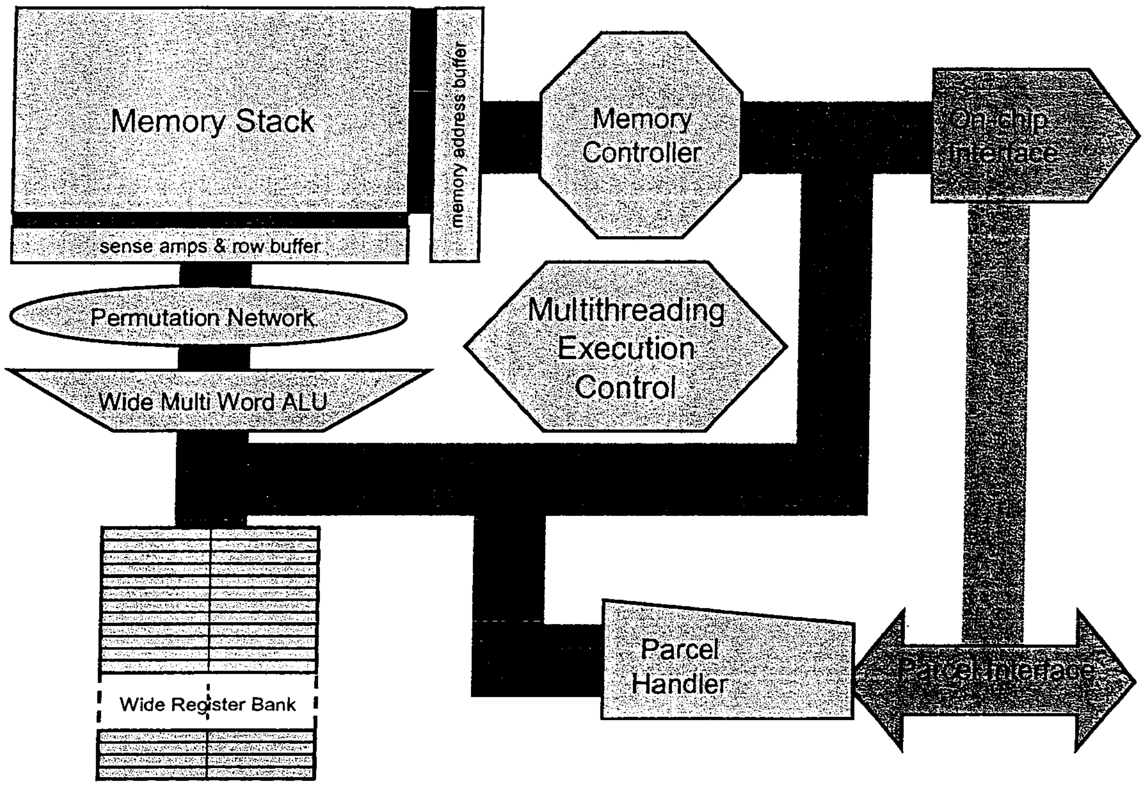

Figure 4: MIND node architecture

<u>Memory Block</u>

The node memory block has one or more conventional stacks of DRAM cells. Each row may typically contain 2048 such cells. There are one-eighth as many output bus lines and sense amps as row cells. Thus a row, once addressed, is read in a succession of eight 256-bit groups (for 2048 cells per row). Access to the memory block is managed by the memory controller, which selects among a number of requests to determine the next memory access cycle and performs the memory operation that may include some simple logical function as part of a compound atomic operation on the designated memory cells.

<u>Parcel Handler</u>

The MIND node architecture is message driven; simple actions or entire threads may be invoked by the incidence of a complex message (parcel) from an external source. The parcel handler is responsible for accepting incoming messages and transmitting outgoing messages. It may also perform as an intermediate router, accepting a message at one port and sending it out a second port, depending on the parcel's destination address. The parcel handler may directly access hardware buffers in response to a simple physical address parcel, access memory locations in response to a simple virtual address parcel, or instantiate a thread if the parcel identifies a function to be performed. Parcels can also support block moves within physical address space or virtual address space.

Thread invocation is the parcel handler's most powerful feature. It initializes the contents of a wide register including pointers to the code to be executed and sets the flag that designates that register as representing the state of an active thread. The thread supervisor then schedules the operations from the new thread, interleaving them with those of other active threads as resources become available. When parcels arrive at a rate greater than the node can process them, they are buffered in main memory and retained until the workload subsides and the buffered parcels can be processed. When this mechanism becomes over subscribed, a parcel is bounced back into the Gilgamesh fabric and grabbed by the first MIND module that has buffer space, to be released later.

Wide Row Register Bank

A bank with registers as wide as the row buffer is used for a number of purposes. They may be used to hold intermediate data, serve as a thread state register, act as an instruction cache, serve as a vector register, provide the equivalent of a translation look-aside buffer (TLB), provide data caching, buffer parcels, and store node control state. The registers are dynamically renamed and allocated by hardware.

Thread Controller

The thread controller determines which wide registers are thread state registers and selects those that are ready to perform an operation. It detects when the different functional elements of the node are ready and matches the thread operation classes of the pending threads with the capabilities of the available function units. These may include resources at other nodes or shared function units on the MIND chip.

Internal Interface

This provides direct access to the other nodes on the MIND module as well as to the shared function units and external interfaces. Data may be moved through this interface or threads may be invoked in either direction.

4. Innovations in Storage Technology

Many Earth science applications are heavily dependent on persistent storage, the capability of storing large volumes of data indefinitely and reacquiring them quickly upon demand. The demand for mass storage within ESE is driven by data acquisition from real-time remote sensing, secondary data products derived from analysis of raw data, or from simulations of physical phenomena such as climate modeling. Data requirements from these domains extend from terabytes to exabytes for long term archiving. Through much of NASA's history, these requirements were met with reel-to-reel tapes stored offline and requiring operator assistance to retrieve. Under certain circumstances, this could take days depending on the nature of the data repository. More recently, large robotic tape drives have been employed that could provide requested data within minutes. However, tape technology is neither ideal, nor is it growing in capacity and performance as fast as its competing technology, namely disk drives.

The mass market of personal computing has forced disk storage technology prices down while mobile computing has driven storage densities up and power, size, and weight characteristics lower. Although trends are highly sporadic, the average rate of capacity increases approaches a factor of two every 18 months. The future of mass storage is being defined around the migration to very large disk farms, with redundant disks for reliability. It is anticipated that tape storage is a sunset technology and that all data archiving will migrate to online cheap disks in the next five years. Today, the cost of a gigabyte of spinning storage is estimated at $1 for the drives themselves, with various degrees of markup depending on systems integrators. This year, EIDE drives will be available with capacities of 320 GB and bandwidths of 50 MB/sec using serial ATA interface and PCI 66 MHz with 64-bit interfaces. A rack may hold as much as 25 TB, with a throughput of 2 GB/sec. A petabyte will consume 25 racks; well within the range of even moderate machine rooms.

But even with the continued gains, data archiving requirements may outpace technological growth for Earth science requirements. The next major advance will be in rapid data compression and decompression. For read-only data files, which dominate most archiving purposes, emerging data compression techniques for many classes of data may yield a compression factor of up to two orders of magnitude. However, the methods still in development are compute intensive. Fortunately, microprocessor performance and cost is such that they can provide the necessary support. Future mass storage systems will be characterized as much by their data compression algorithms as they are by their specific spindle specifications.

5. Innovations in Software Environments

While advances in HEC architectures and distributed grid environments promise higher performance both in terms of capability and capacity computing, their software environments must keep pace in order to assist scientists and engineers deal with the complexity induced by system size and heterogeneity. Driven by the challenges described in Section 2.5, innovations in software environments must occur in two broad areas: programming tools and runtime systems. In this section, we describe important ongoing research at NASA in each arena that will eventually allow users to more easily harness the computational power of future petaflops-scale supercomputers.

5.1. Programming Tools

Efficient parallelization is the key to faster codes that enable scientists to conduct higher fidelity simulations of their problems on state-of-the-art parallel computers. However, this mapping of algorithms to architectures—a key to performance, and ultimately science productivity—remains an extremely tedious and error prone process. Furthermore, inflexible, highly-coupled implementations translate to ever-increasing computing requirements and human effort. In fact, the formulation of large-scale multi-disciplinary simulations becomes intractable as the complexity of the model increases. With the goal to eventually automate the mapping of applications to architectures, researchers at NASA Ames, in collaboration with groups at the University of Greenwich and the European Center for Parallelism, are developing a number of parallelization and other computer-aided programming support tools, and testing them on a suite of NASA scientific applications. Their multi-level parallelization and optimization environment (shown in Figure 5) integrates three prototype tools to enable the rapid transformation of serial codes into efficient, correctly functioning, multi-level parallel codes.

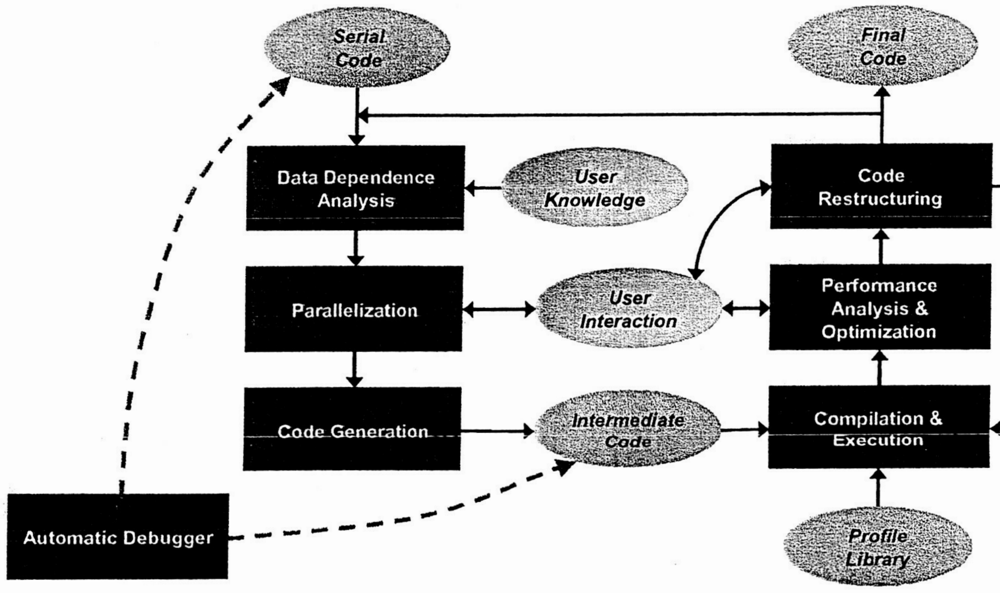

Figure 5: Multi-level parallelization and optimization environment

5.1.1. Automatic Parallelization

The first module of the integrated programming environment is called CAPO [7]. It is a parallelization assistant that includes sophisticated static program analysis, an informative and intelligent user interface, and portable parallel code generation. If necessary, CAPO is also able to restructure code with some user interaction, to improve performance. It first performs a data dependency analysis to determine how different variables depend on one another. All information is stored in a database that a user can examine and potentially utilize to remove parallelization obstacles. The other two modules also use this database in order to function efficiently. CAPO next conducts a loop level analysis, searching for repeated sequences of instructions in the code that can be parallelized. Users are then guided through a series of GUIs to view all instances where the code did not parallelize. The more obstacles a user is able to remove, the higher the level of parallelization that can be achieved. Once the user is satisfied, CAPO automatically inserts OpenMP directives and generates the parallel code for shared-memory systems. A similar tool, called CAPTools, which was developed at the University of Greenwich, can generate message-passing codes to run on distributed-memory machines. In fact, CAPO and CAPTools can interact seamlessly to produce multi-level parallel code. Recently, CAPO enabled researchers at NASA Goddard to run larger cloud simulations and more complex models in shorter periods of time, thereby providing additional insights into phenomena such as air-sea interactions and global climate changes [8].

5.1.2. Relative Debugging

Even when tools such as CAPO are used, porting serial codes to parallel form is still susceptible to errors. For example, the user might incorrectly indicate that a loop can be safely run in parallel. This could lead to the program using stale values in a distributed-memory version of the code. Finding such parallelization errors often requires the programmer to run the serial and parallel versions side-by-side in debuggers to try to find the first significant difference in execution. The Portable Parallel/Distributed Debugger (p2d2) group at NASA Ames and the CAPTools group at the University of Greenwich have collaborated to automate this process [9]. They use a combination of backtracking and re-execution in order to find the first difference in computation that may ultimately lead to an incorrect value that the user has indicated. In a prototype implementation, they use static analysis information from CAPTools in order to perform the backtracking as well as the mapping required between serial and parallel computations.

5.1.3. Performance Analysis

The process of tuning a parallelized code is also labor intensive. The programmer will typically go through several cycles of source code optimization and performance analysis. In collaboration with the European Center for Parallelism at Barcelona, researchers from NASA Ames are building an environment where the user can jointly navigate through program structure and performance data to make optimization decisions, automating tedious and error-prone tasks. They are interfacing CAPO with Paraver, a performance tool, so that the parallelization tool can query performance trace data and dynamically correlate the outcome with its loop analysis information [10]. The environment is currently able to support a multitude of parallel programming paradigms (MPI, OpenMP, and multi-level).

5.2. Runtime Systems

The conventional software architecture of a monolithic operating system and compiler-driven processing node is being replaced by a new logical organization that incorporates a third component, the runtime system, that serves as a virtual machine between systems software and the underlying hardware architecture. This layer of software abstraction is needed to manage large ensembles of processing elements and overcome costly operating system calls. Last generation MPPs and commodity clusters (e.g. Beowulf-class PC clusters) installed an operating system on each node and employed some protocol to manage them all. Distributed shared memory parallel systems such as the SGI Origin have a single system image, but the management of concurrent tasks and parallel resources is still relatively static. Runtime information about the status and operation of both the machine and the application is not exploited in the management of the system unless expressly incorporated into the application code. As advanced HEC architectures with hierarchies of processing elements and memories are realized, the role of the runtime system will become more pronounced in order to provide flexibility, ease of programming, generality, and performance efficiency.

A new generation of runtime systems will present the computing resources to a controlling agent and dynamically allocate them to the application as the program reveals its capability demands. Such runtime nodes will also provide low-level management of the local resources including fault tolerance, debugging, performance monitoring, and logical services transparently to both the operating system and the compilers. A general framework referred to as "introspection" monitors all aspects of the local resources and nearby environment to respond to changes and requirements. Agents, which are objects that track specific local and neighborhood state, provide the vehicle for implementing introspection. Systems embodying the concept of agent-supported introspection are scalable, potentially over millions of nodes, because they operate on a set of rules that do not require global information, although some global parameters may be part of the monitored state. In providing an intermediate for management between the single operating system strategy and the ensemble of independent and separate operating systems, runtime introspection will provide a robust and efficient abstraction for conducting time dependent parallel processing, simplifying the programming and management of very large high end computers.

6. Summary and Conclusions

This article provided an overview of emerging high end computing (HEC) technologies in the areas of computer system architecture, mass storage management, and software environments that will help meet some of the demanding requirements of future Earth science applications. We specifically described how innovations in computer system architecture (streaming, HTMT, and Gilgamesh) coupled with advances in software environments (automatic parallelization, performance analysis, and runtime systems) will be instrumental in satisfying the capability and capacity needs of weather forecasting, climate modeling, and solid Earth science. NASA is intimately involved in almost all of these research efforts with the goal to enable efficient petaflops-scale processing by the end of this decade.

References

1. I. Foster and C. Kesselman (Eds.), The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, San Francisco, CA, 1999.
2. B. Dally, P. Hanrahan, and R. Fedkin, "A Streaming Supercomputer," Unpublished report, Stanford University, Stanford, CA, Sep 2001 (available at http://graphics.stanford.edu/sss)
3. R. Nagarajan, K. Sankaralingam, D. Burger and S. Keckler, "A Design Space Evaluation of Grid Processor Architectures," 34th International Symposium on Microarchitecture, Austin, TX, Dec 2001, pages 40-51.
4. T. Sterling and L. Bergman, "A Design Analysis of a Hybrid Technology Multithreaded Architecture for Petaflops Scale Computation," 13th International Conference on Supercomputing, Rhodes, Greece, June 1999, pages 286-293.
5. T. Sterling, D. Katz, and L. Bergman, "High Performance Computing Systems for Autonomous Spaceborne Missions," International Journal of High Performance Computing Applications, Volume 15, Number 3, 2001, pages 282-296.
6. T. Sterling and H. Zima, "Gilgamesh: A Multithreaded Processor-In-Memory Architecture for Petaflops Computing," SC2002, Baltimore, MD, Nov 2002.
7. H. Jin, M. Frumkin, and J. Yan, "Automatic Generation of OpenMP Directives and its Application to Computational Fluid Dynamics Codes," 3rd International Symposium on High Performance Computing, Tokyo, Japan, Oct 2000, Springer LNCS Volume 1940, pages 440-456.
8. H. Jin, G. Jost, D. Johnson, and W.-K. Tao, "Experience on the Parallelization of a Cloud Modeling Code using Computer-Aided Tools," tech. report NAS-03-006, NASA Ames Research Center, Moffett Field, CA, 2003.
9. G. Matthews, R. Hood, S. Johnson, and P. Leggett, "Backtracking and Re-execution in the Automatic Debugging of Parallelized Programs," 11th International Symposium on High Performance Distributed Computing, Edinburgh, Scotland, July 2002, pages 150-160.
10. G. Jost, H. Jin, J. Labarta, and J. Gimenez, "Interfacing Computer Aided Parallelization and Performance Analysis," 3rd International Conference on Computational Science, Melbourne, Australia, June 2003, Springer LNCS, to appear.