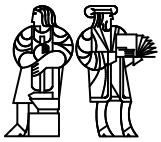
LABORATORY FOR
COMPUTER SCIENCE



MASSACHUSETTS
INSTITUTE OF
TECHNOLOGY

A Personal Supercomputer for Climate Research

Computation Structures Group Memo 425
August 6, 1999

James C. Hoe
Lab for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139
jhoe@lcs.mit.edu

Chris Hill, Alistair Adcroft
Dept. of Earth, Atmospheric and Planetary Sciences
Massachusetts Institute of Technology
Cambridge, MA 02139
{cnh,adcroft}@mit.edu

In Proceedings of SC'99

This work was funded in part by grants from the National Science Foundation and NOAA and with funds from the MIT Climate Modeling Initiative[23]. The hardware development was carried out at the Laboratory for Computer Science, MIT and was funded in part by the Advanced Research Projects Agency of the Department of Defense under the Office of Naval Research contract N00014-92-J-1310 and Ft. Huachuca contract DABT63-95-C-0150. James C. Hoe is supported by an Intel Foundation Graduate Fellowship. The computational resources for this work were made available through generous donations from Intel Corporation. We acknowledge Arvind and John Marshall for their continuing support. We thank Andy Boughton and Larry Rudolph for their invaluable technical input.

A Personal Supercomputer for Climate Research

James C. Hoe
Lab for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139
jhoe@lcs.mit.edu

Chris Hill, Alistair Adcroft
Dept. of Earth, Atmospheric and Planetary Sciences
Massachusetts Institute of Technology
Cambridge, MA 02139
{cnh,adcroft}@mit.edu

Abstract

We describe and analyze the performance of a cluster of personal computers dedicated to coupled climate simulations. This climate modeling system performs comparably to state-of-the-art supercomputers and yet is affordable by individual research groups, thus enabling more spontaneous application of high-end numerical models to climate science. The cluster's novelty centers around the Arctic Switch Fabric and the StarT-X network interface, a system-area interconnect substrate developed at MIT. A significant fraction of the interconnect's hardware performance is made available to our climate model through an application-specific communication library. In addition to reporting the overall application performance of our cluster, we develop an analytical performance model of our application. Based on this model, we define a metric, Potential Floating-Pointing Performance, which we use to quantify the role of high-speed interconnects in determining application performance. Our results show that a high-performance interconnect, in conjunction with a light-weight application-specific library, provides efficient support for our fine-grain parallel application on an otherwise general-purpose commodity system.

1 Introduction

Cluster computers constructed from low-cost platforms with commodity processors are emerging as a powerful tool for computational science[27]. These clusters are typically interconnected by standard local area networks, such as switched Fast Ethernet. Fast Ethernet is an attractive option because of its low cost and widespread availability. However, communication over Fast Ethernet, and even Gigabit Ethernet, incurs relatively high overhead and latencies[25]. This creates a communication bottleneck that limits the utility of these clusters and presents a significant challenge in taking advantage of this exciting hardware trend in climate research, where the dominant computation tools are not, in general, embarrassingly parallel.

General circulation models (GCMs) are widely used in climate research. These models numerically step forward the *primitive equations*[17] that govern the planetary-scale time evolution of the atmosphere (in an AGCM) and ocean (in an OGCM). Though the models possess abundant data-parallelism, they can only tolerate long latency and high-overhead communication in relatively coarse grain parallel configurations that have a low communication-to-computation ratio. In many cases, only numerical experiments that take months to complete have a sufficiently large problem size to achieve an acceptable ratio. This limitation precludes capitalizing on one of the most attractive qualities of affordable cluster architectures - their mix of responsiveness with high-performance that gives rise to a *personal supercomputer* suited to exploratory, spontaneous numerical experimentation. A specialized high-performance interconnect can substantially ameliorate communication bottlenecks and allows a fine-grain application to run efficiently on a cluster[8, 2, 9].

In this paper, we present Hyades, an affordable cluster of commodity personal computers with a high-performance interconnect. Hyades is dedicated to parallel coupled climate simulations. By developing a small set of communication primitives tailored specifically to our application, we efficiently support very fine-grain parallel execution on general purpose hardware. This provides a low-cost cluster system that has significant potential as a climate research tool. The architecture of the cluster is described in Section 2. The climate model we employ is described in Section 3. Section 4 discusses the mapping from the numerical

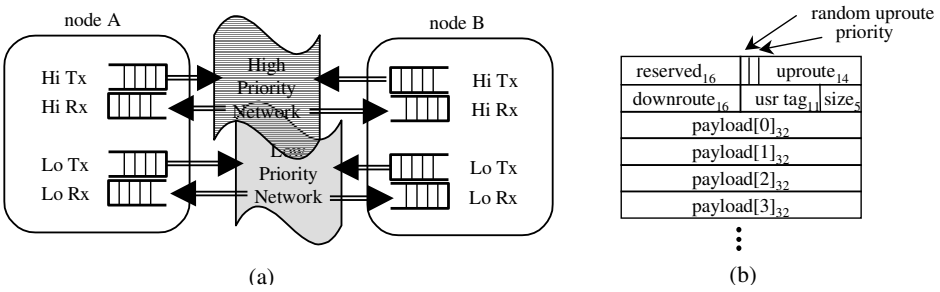


Figure 1: (a) PIO Mode Abstraction and (b) StarT-X Message Format

size ($byte$)	O_s (μsec)	O_r (μsec)	$\frac{T_{round-trip}}{2}$ (μsec)	$L_{network}$ (μsec)
8	0.4	2.0	3.7	1.3
64	1.7	8.6	11.7	1.4

Figure 2: LogP performance characteristics of PIO message passing for 8-byte payload messages and 64-byte payload messages.

model onto the cluster, and Section 5 analyzes the performance. We conclude with a discussion of the lessons learned.

2 Cluster Architecture

The Hyades cluster is comprised of sixteen two-way Symmetric Multiprocessors (SMP). Each SMP node is connected to the Arctic Switch Fabric through a StarT-X PCI network interface unit (NIU), yielding interprocessor communication that is an order-of-magnitude faster than a standard local area network. The total cost of the hardware is less than $100,000, about evenly divided between the processing nodes and the interconnect.

2.1 Processing Nodes

Each SMP contains two 400-MHz Intel PII processors with 512 MBytes of 100-MHz SDRAM. These SMPs, based on Intel's 82801AB chipsets, have significantly better memory and I/O performance in comparison to previous generations of PC-class machines. The SMPs are capable of sustaining over 120 MByte/sec of direct memory accesses (DMA) by a PCI device. The latency of an 8-byte read of an uncached memory-mapped (*mmap*) PCI device register is 0.93 μsec while the minimum latency between back-to-back 8-byte writes is 0.18 μsec. These I/O characteristics directly govern the performance of interprocessor communication, and hence, have a significant impact on the performance of our parallel application.

2.2 The Arctic Switch Fabric

Inter-node communication is supported through the Arctic Switch Fabric[5, 6], a system area network designed for *T massively-parallel processors[24, 3]. This packet-switched multi-stage network is organized in a fat-tree topology. The latency through a router stage is less than 0.15 μsec. Each link in the fat-tree supports 150 MByte/sec in each direction. In an N-endpoint full fat-tree configuration, the network's bisection bandwidth is $2 \times N \times 150$ MByte/sec.

In addition to high-performance, Arctic provides features to simplify software communication layers. First, Arctic maintains a FIFO ordering of messages sent between two nodes along the same path in the fat-tree topology. Second, Arctic recognizes two message priorities and guarantees that a high-priority message cannot be blocked by low-priority messages. Lastly, Arctic's link technology is designed such that the software layer can assume error-free operations. The correctness of the network messages is verified at

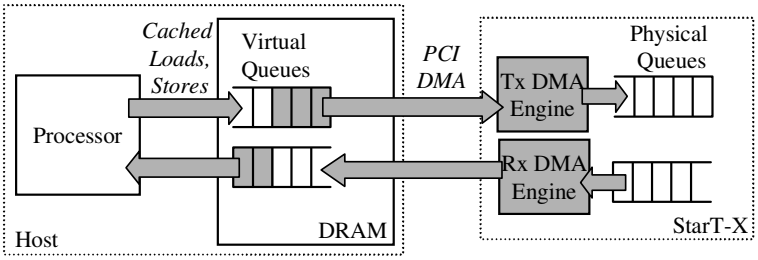


Figure 3: Cacheable Virtual Interface (VI) Abstraction

every router stage and at the network endpoints using CRC. The software layer only has to check a 1-bit status to detect the unlikely event of a corrupted message due to a catastrophic network failure.

2.3 The StarT-X NIU

The StarT-X network interface unit (NIU) provides three simple but powerful message passing mechanisms to user-level applications. Its operation is designed to minimize the need for elaborate software layers between the network and the application. The StarT-X message passing mechanisms are implemented completely in hardware, rather than using an embedded processor, so the peak performance can be attained easily and predictably under a variety of workloads. The StarT-X NIU has been used in clusters consisting of SUN E5000 SMPs as well as previous generations of Intel PCs. In each case, StarT-X's performance has only been limited by the peak performance of the particular host's 32-bit 33-MHz PCI environment.

The different StarT-X communication mechanisms are optimized to support different granularities of communication patterns. Elsewhere, we have described StarT-X's operation in detail[16]. Here, we briefly present StarT-X's Programmed I/O Interface (PIO) and Cacheable Virtual Interface (VI), the two mechanisms employed by the GCM code.

PIO Mode: To support fine-grain message-passing applications, the PIO interface presents a simple FIFO-based network abstraction similar to the CM-5 data network interface[29]. Processes on different nodes communicate by exchanging messages which contain two 32-bit header words followed by a variable size payload of between 2 and 22 32-bit words. The PIO mode abstraction and packet format are depicted in Figure 1. Due to the relative high cost of the uncached *mmap* accesses, we can reliably estimate the performance of PIO-mode communication by summing the cost of the *mmap* accesses (given in Section 2.1). For example, when passing an 8-byte message, the sender and the receiver each perform two 8-byte (header plus payload) *mmap* accesses to the NIU registers, and thus, we can estimate the overhead of sending and receiving a message to be 0.36 μsec and 1.86 μsec, respectively. The experimentally determined LogP characteristics[10] of StarT-X's PIO mechanism, summarized in Figure 2, corroborate these estimates.

VI Mode: In modern architectures, cached accesses to main memory are orders-of-magnitude faster than PIO accesses. To take advantage of this performance disparity, the VI mode uses DMA to extend the physical transmit and receive queues into the memory system. Figure 3 illustrates this abstraction. The user interacts with StarT-X indirectly through memory, and hence, avoids costly PIO accesses. The VI mode makes use of a pinned, contiguous physical memory region for DMA. The VI memory region is mapped into a cacheable virtual memory region of the user process. To send a VI message, instead of enqueuing directly to the hardware transmit queue, the user process writes the message to a cacheable VI buffer. The user then invokes StarT-X's DMA engine to enqueue the message into the physical transmit queue. On the receiver end, StarT-X delivers the message directly to a pre-specified buffer in the receiving node's VI memory region. Because DMA invocation and status polling require *mmap* accesses to StarT-X registers, the VI mode is most efficient when multiple out-bound messages are queued consecutively in memory and are transmitted with a single DMA invocation. The peak payload transfer bandwidth in VI mode is 110 MByte/sec.

3 The MIT General Circulation Model

The climate model used on Hyades is the MIT general circulation model[21, 20, 15, 14]. The model is a versatile research tool that can be applied to a wide variety of processes ranging from non-hydrostatic rotating fluid dynamics[15, 22] to the large-scale general circulation of the atmosphere and ocean[14, 19]. The model is implemented to exploit the mathematical isomorphisms that exist between the equations of motion for an incompressible fluid (the ocean) and those of a compressible fluid (the atmosphere), allowing atmosphere and ocean simulations to be performed by the same basic model code[14].

3.1 Numerical Procedure

At the heart of the model is a numerical kernel that steps forward the equations of motion for a fluid in a rotating frame of reference using a procedure that is a variant on the theme set out by Harlow and Welch[13]. The kernel can be written in semi-discrete form to second order in time, Δt, thus:

$$\frac{\underline{v}^{n+1} - \underline{v}^{n}}{\Delta t} = \underline{G}_{v}^{n+\frac{1}{2}} - \nabla p^{n+\frac{1}{2}} \tag{1}$$

$$\nabla . \underline{v}^{n+1} = 0 \tag{2}$$

Equations (1) and (2) describe the time evolution of the flow field ($\underline{v}$ is the three-dimensional velocity field) in response to forcing due to $\underline{G}$ (representing inertial, Coriolis, metric, gravitational, and forcing/dissipation terms) and to the pressure gradient ∇p. For brevity we do not write the equations for thermodynamic variables which must also be stepped forward to find, by making use of an equation of state, the buoyancy b.

The pressure field, p, required in equations (1) and (2) is found by separating the pressure into hydrostatic, surface and non-hydrostatic parts. The flow in the climate scale simulations presented here is hydrostatic, yielding a two-dimensional elliptic equation for the surface pressure p_s that ensures non-divergent depth integrated flow:

$$\nabla_h . H \nabla_h p_s = \nabla_h . \overline{\underline{G}_{v_h}^{n+\frac{1}{2}}}^{H} - \nabla_h . \overline{\nabla_h p_{hy}}^{H} \tag{3}$$

($\overline{}^{H}$ indicates the vertical integral over the local depth, H, of the fluid and the subscript $_h$ denotes horizontal). In the hydrostatic limit the non-hydrostatic pressure component is negligible and vertical variations in p are computed hydrostatically from the buoyancy, b, to yield p_{hy}.

3.2 Spatial Discretization

Finite volume techniques are used to discretize in space, affording some flexibility in sculpting of the model grid to the irregular geometry of land masses[1]. In this approach the continuous domain (either ocean or atmosphere) is carved into "volumes" as illustrated in Figure 4. Discrete forms of the continuous equations 1–3 can then be deduced by integrating over the volumes and making use of Gauss's theorem. Finite-volume discretization produces abundant data parallelism and forms the basis for mapping the algorithm to a multi-processor cluster.

4 Mapping to the Cluster

The numerical kernel of the GCM code is written in sequential Fortran to work on a computational domain that has been decomposed horizontally into tiles that extend over the full depth of the model. The tiles form the basic unit on which computation is performed and over which parallelism is obtained. As shown in Figure 5, tiles include a lateral overlap or "halo" region in which data from adjacent tiles are duplicated. For some stages of the calculation computations are duplicated (overcomputed) in the halo region, so that

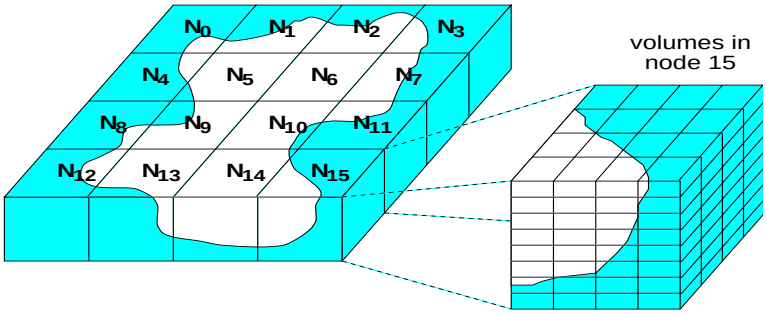


Figure 4: Finite volume spatial discretization. This schematic diagram illustrates how an ocean basin might be mapped to a 16-node parallel computer. The cutout for node 15 ($\mathbf{N}_{15}$) shows the volumes within a subdomain. The dark regions indicate land. The finite volume scheme allows both the face area and the volume of a cell that is open to flow to vary in space, so that the volumes can be made to fit irregular geometries. The vertical dimension stays within a single node.

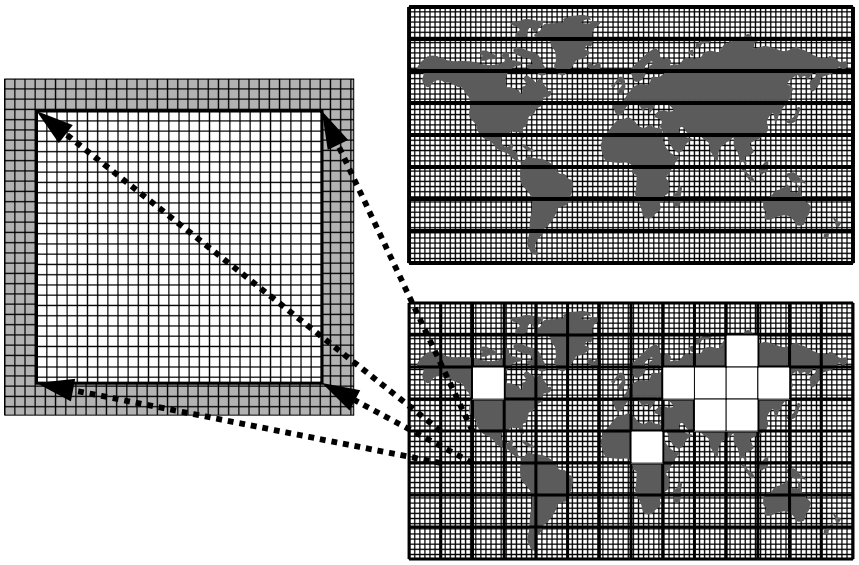


Figure 5: Flexible tiled domain decomposition. Tile sizes and distributions can be defined to produce long strips consistent with vector memories (upper panel). Alternatively small, compact blocks can be created which are better suited to deep memory hierarchies (lower panel). Connectivity between tiles can be tuned to reduce the overall computational load. The left panel shows a close-up of a single tile. The shaded area indicates the "halo" region. The halo region surrounds the tile interior and holds duplicate copies of data "belonging" to the interior regions of neighboring tiles.

INITIALIZE. Define topography, initial flow and tracer distributions
FOR each time step n DO

PS

Step forward state.	$\underline{v}^n = \underline{v}^{n-1} + \Delta t(\underline{G}_v^{n-\frac{1}{2}} - \nabla p^{n-\frac{1}{2}})$
Calculate time derivatives.	$\underline{G}_v^{n+\frac{1}{2}} = \mathbf{gv}(\underline{v}, b)$
Calculate hydrostatic p.	$p_{hy}^{n+\frac{1}{2}} = \mathbf{hy}(b)$

DS

Solve for pressure.	$\nabla_h . H \nabla_h p_s^{n+\frac{1}{2}} = \nabla_h . \overline{\underline{G}_{v_h}^{n+\frac{1}{2}}}^H - \nabla_h . \overline{\nabla_h p_{hy}^{n+\frac{1}{2}}}^H$

END FOR

Figure 6: Outline of the GCM algorithm. The model iterates repeatedly over a time-stepping loop comprised of two main blocks, **PS** and **DS**. A numerical experiment may entail many millions of time-steps. In **PS**, time tendencies (G terms) are calculated using the model state at previous time levels ($^{n,n-1,n-2}$ etc..). **DS** involves finding a pressure field p_s such that the flow field $\underline{v}$ at the succeeding time level will satisfy the continuity relation in equation (2). For clarity the calculation of G terms for thermodynamic variables (temperature and salinity in the ocean, and temperature and water vapor in the atmosphere) has been omitted from the **PS** outline. These terms are calculated by functions that have a similar form to the **gv**() function and yield the buoyancy, b.

data does not need to be fetched from neighboring tiles. This "overcomputation" is employed to reduce the number of communication and synchronization points required in a model time-step.

Figure 6 shows the high-level structure of the GCM code. The two main stages of each model time-step are the Prognostic Step (**PS**) and the Diagnostic Step (**DS**). Both stages employ finite volume techniques and are formulated to compute on a single tile at a time, but the two stages differ in the amount and style of parallel communication that they require. All terms in **PS** can be calculated from quantities contained within a local stencil of 3×3 grid points. Accordingly **PS** is formulated to employ "overcomputation", so that all **PS** terms for a given tile can be calculated using only data within that tile and its halo region. In this way, the communication and synchronization cost for **PS** is isolated to one occurrence within a time step, and the ratio of on-node computation to inter-node communication is relatively high. In contrast, the form of equation (3) implies global connectivity between grid points during **DS**. A pre-conditioned conjugate-gradient[21, 15, 26] iterative solver is employed in this phase. This procedure reflects the inherent global connectivity of equation (3), and thus does not lend itself to overcomputation. Accordingly the ratio of on-node computation to inter-node communication is low for **DS**. In climate modeling scenarios, **DS** is also distinct from **PS** because **DS** operates on the vertically integrated model state (a two-dimensional field) whereas **PS** operates on the full, three-dimensional model state.

Maximizing the parallel performance of the model requires optimizing the two key primitives that communicate data amongst tiles in **PS** and **DS**. The first of these primitives is *exchange* which brings halo regions into a consistent state. In **DS**, the iterative solver requires an exchange to be applied to two fields at every solver iteration. **DS** exchanges update a halo region that is only one element wide and operate on a two-dimensional field. In **PS**, an exchange must be performed for each of the model three-dimensional state variables over a halo width of at least three points. The second key primitive is *global sum*. This primitive orchestrates the summing of a single floating-point number from each tile and returns the result to every tile. Two global sum operations are required at every solver iteration in **DS**; **PS** does not require any global sum operations.

The GCM software architecture isolates the communication code from the numerical code. Non-critical communication is implemented in a portable way using MPI or shared memory, but performance critical communication, exchange and global sum, can be customized for the specific hardware. High-performance implementations of these two primitives are vital to fine-grain parallel performance.

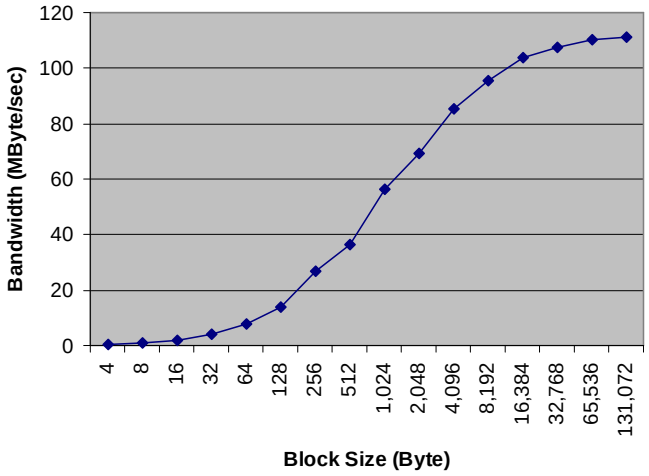


Figure 7: Transfer bandwidth as a function of block size.

4.1 Optimized Exchange

An exchange operation on Hyades is implemented as two separate VI-mode transfers in opposite directions. The two transfers are carried out sequentially because a single transfer alone can saturate the PCI bus. Since the StarT-X NIU does not have address translation facilities, it can only DMA to and from a pinned and contiguous physical memory region. To utilize the VI mode, the sending processor must copy the data from the source buffer into a special VI memory region. For efficiency, the sender copies the data in several small chunks and initiates DMA on a chunk immediately after each copy to overlap the DMA transfer with the next round of copying. Similarly, the receiver copies the data, in chunks, from the hardware inbound messages queue to the destination buffer as soon as the messages arrive. When the transfer in one direction of the exchange completes, the sender and receiver reverse roles and continue in the opposite direction.

A node can sustain 110 MByte/sec of peak data transfer bandwidth during an exchange. However, there is a one-time 8.6 μsec overhead to negotiate a transfer between two nodes. This small overhead is important because, for fine-grain problem sizes, the exchanges in **DS** only transfer a few kilobytes of data. Without a long transfer to amortize the overhead, this 8.6 μsec overhead reduces the perceived transfer bandwidth to only 56.8 MByte/sec for a 1-KByte transfer. The perceived bandwidth reaches 90% of the peak 110 MByte/sec for 9-KByte transfers. The perceived transfer bandwidth is plotted as a function of the transfer block size in Figure 7. Arctic's fat-tree interconnect can handle multiple simultaneous transfers with undiminished pair-wise bandwidth.

Hyades uses one StarT-X NIU in each two-processor SMP node. The discussion above pertains to operating a single processor at each network endpoint. This usage offers the maximum ratio of communication to computation performance. When multiple processors per SMP participate in the application, the exchange primitive operates in a mix-mode fashion which uses shared memory to handle intra-SMP communication. In this mode, one processor on each SMP is designated as the communication master who has sole control of the NIU and processes remote exchanges on behalf of slave processors. The slaves post remote exchange requests to the master through a shared-memory semaphore. The slave-to-slave exchange bandwidth is about 30% lower than a master-to-master exchange.

4.2 Optimized Global Sum

With ample network bandwidth, our implementation of global sum minimizes latency at the expense of more messages. For an N-node global sum where N is a power of two, $N \times log_2 N$ messages are sent over $log_2 N$ rounds. The global sum algorithm computes N reductions concurrently such that after i rounds, every node has the partial sum for the group of nodes whose node identifiers only differ in the lowest i bits. The communication pattern and the partial sum for each of the three rounds in a 8-way global sum is shown in Figure 8.

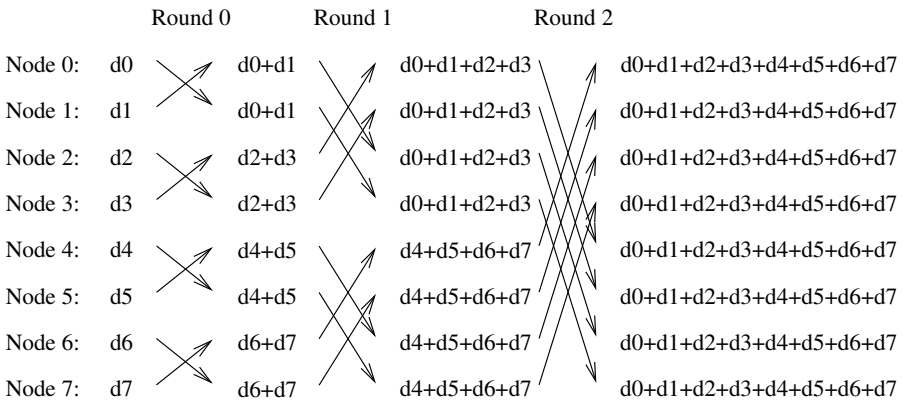


Figure 8: Communication pattern in an eight-way global sum.

The critical path in this algorithm involves successive sending and receiving of $log_2 N$ messages. Ignoring the second-order effect of varying network transit time, the elapsed time for an N-way global sum can be approximated by the simple formula:

$$t_{gsum} = C log_2 N$$

where the constant C corresponds to the user-to-user latency of an 8-byte payload message plus the CPU processing latency in each round. The measured latencies for 2-way, 4-way, 8-way and 16-way global sums are 4.0 μsec, 8.3 μsec, 12.8 μsec and 18.2 μsec, respectively. A least-squares fit to these measurements is $t_{gsum} = (4.67 \cdot log_2 N - 0.95)$ μsec.

When multiple processors per SMP are participating in the application, the processors on each SMP first generate a local global sum using shared-memory semaphores. Next, a master processor from each SMP enters into the system-wide global sum operation. Finally, the master processor distributes the overall global sum to local processors using shared memory. The local summing operation adds about 1 μsec to the global sum latency. On our two-way SMPs, the measured latencies for 2x2-way, 2x4-way, 2x8-way and 2x16-way global sums are 4.8 μsec, 9.1 μsec, 13.5 μsec and 19.5 μsec, respectively.

5 Performance

To demonstrate the utility of the cluster, a representative climate research atmosphere-ocean simulation was analyzed. Figure 9 shows a typical output from this simulation. The atmosphere and ocean simulations run at 2.8125° horizontal resolution (the lateral global grid size is 128×64 points). This experiment uses an intermediate complexity atmospheric physics package[14, 12] which has been designed for exploratory climate simulations. The configuration is especially well suited to predictability studies of the contemporary climate and to paleo-climate investigations. Running this configuration profitably on a cluster system is a challenging proposition. The total size of the model domain is not that large, so tiles arising from the domain decomposition illustrated in Figure 4 are small. As a result, processors must communicate relatively frequently, and parallel speed-up is very sensitive to communication overheads.

5.1 Overall Performance

In coupled simulations, the ocean and atmosphere isomorphs must run concurrently, periodically exchanging boundary conditions. During full-scale production runs, each isomorph occupies half of the cluster, sixteen processors on eight SMPs. In Figure 10, we compare Hyades to contemporary vector machines in terms of their performance on the GCM code. As the table shows, GCM performs competitively on all platforms, and the performance on sixteen processors of our cluster is comparable to a one-processor vector machine. On Hyades, for each of the component models, the sustained Flop rate on sixteen processors is fifteen times higher than the single processor rate. For a full-scale production run, the sustained combined floating-point performance of both the atmosphere and the ocean isomorphs is between 1.6–1.8 GFlop/sec.

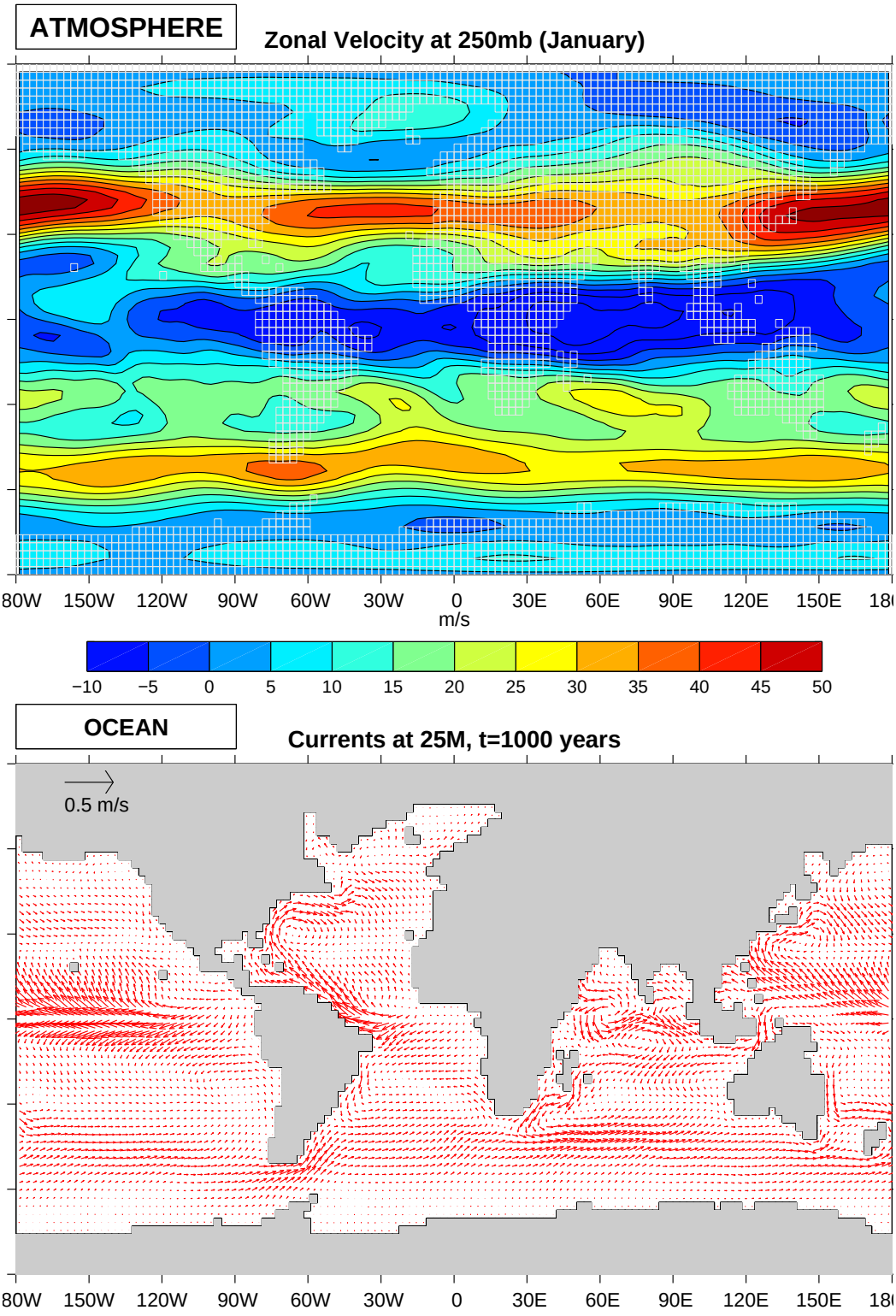


Figure 9: Currents obtained from the atmosphere and ocean isomorphs of the MIT general circulation model.

Processor Count	Machine	Sustained performance ($\frac{10^9 \text{flop}}{\text{sec}}$)
1	Cray Y-MP	0.4
4	Cray Y-MP	1.5
1	Cray C90	0.6
4	Cray C90	2.2
1	NEC SX-4	0.7
4	NEC SX-4	2.7
1	Hyades	0.054
16	Hyades	0.8

Figure 10: Performance of ocean isomorph of our coarse resolution climate model. Because it is based on the same kernel, the atmospheric counterpart has an almost identical profile.

5.2 Performance Model

A simple performance model can be used to examine the roles that floating-point capability and network capability play in setting the overall performance of this simulation. The performance model formula is derived by breaking down the numerical model phases **PS** and **DS** into constituent computation and communication stages.

An approximation for the time $t_{\mathbf{ps}}$ taken by a single pass through the **PS** phase is

$$t_{\mathbf{ps}} = t_{ps_compute} + t_{ps_exch} \tag{4}$$

where

$$t_{ps_compute} = \frac{N_{ps} n_{xyz}}{F_{ps}} \tag{5}$$

$$t_{ps_exch} = 5 t_{exchxyz} \tag{6}$$

The **PS** phase processor compute time $t_{ps_compute}$ is the total number of floating-point operations performed by each processor divided by the processor's floating-point operation rate. The total number of floating-point operations per processor in the **PS** phase is the product of N_{ps} and n_{xyz}. The term N_{ps} is the number of floating-point operations per grid cell in a single **PS** phase, and it can be determined by inspecting the model code. n_{xyz} is the number of grid cells in the 3-D volume assigned to a single processor. F_{ps} is the measured floating-point performance on the **PS** phase single-processor kernel.

The communication time t_{ps_exch} for the **PS** phase is the time for applying the three-dimensional exchange primitive to five separate model fields. For a given grid decomposition, the exchange block size is known, and $t_{exchxyz}$ can either be measured experimentally from a stand-alone benchmark or be estimated from the bandwidth curve in Figure 7. In **PS**, the exchange primitives are typically called with data blocks ranging from tens to hundreds of kilobytes.

The time $t_{\mathbf{ds}}$ taken by a single iteration of the **DS** phase solver can be expressed as

$$t_{\mathbf{ds}} = t_{ds_compute} + t_{ds_exch} + t_{ds_gsum} \tag{7}$$

where

$$t_{ds_compute} = \frac{N_{ds} n_{xy}}{F_{ds}} \tag{8}$$

$$t_{ds_exch} = 2 t_{exchxy} \tag{9}$$

$$t_{ds_gsum} = 2 t_{gsum} \tag{10}$$

The **DS** phase processor compute time $t_{ds_compute}$ is the total number of floating-point operations performed by each processor ($N_{ds} n_{xy}$) divided by the processor's floating-point performance F_{ds}. n_{xy} is the

PS phase parameters (Atmosphere)			
N_{ps}	n_{xyz}	$t_{exchxyz}$ (μsecs)	F_{ps} (MFlop/sec)
781	5120	1640	50

PS phase parameters (Ocean)			
N_{ps}	n_{xyz}	$t_{exchxyz}$ (μsecs)	F_{ps} (MFlop/sec)
751	15360	4573	50

DS phase parameters				
N_{ds}	n_{xy}	t_{gsum} (μsecs)	t_{exchxy} (μsecs)	F_{ds} (MFlop/sec)
36	1024	13.5	115	60

Figure 11: Performance model parameters of a coupled ocean-atmosphere simulation at 2.8125° resolution. Each isomorph occupies sixteen processors on eight SMPs. For the **PS** phase the atmosphere and the ocean model configurations have different vertical resolutions and contain slightly different numerical computations. This results in slightly different performance model parameters for the **PS** phase. The **DS** phase parameters are the same for both components.

number of vertical columns assigned to a single processor. Again, F_{ds} can be measured using a stand-alone single-processor **DS** kernel. The communication time t_{ds_exch} for the **DS** phase is the time for applying the two-dimensional exchange primitive to two separate model fields plus the time for applying the global sum primitive twice. The data blocks exchanged in the **DS** phase are typically an order of magnitude smaller than those in **PS**.

The total runtime T_{run} for a numerical experiment with N_t time steps and with a mean number of solver iterations, N_i, in **DS**, is

$$T_{run} \quad = \quad N_t t_{\mathbf{ps}} + N_t N_i t_{\mathbf{ds}} \tag{11}$$

5.3 Validation of the Performance Model

We tested the performance ourmodel against a one-year atmospheric simulation running on sixteen processors over eight SMPs. The performance model parameters corresponding to this simulation are given in Figure 11. The exchange and global sum cost is determined using stand-alone benchmarks. The same is true for F_{ps} and F_{ds}. The one-year simulation requires 183 minutes of wall-clock time.

The performance model predicts the total communication time T_{comm} should be given by

$$T_{comm} = 2N_t N_i t_{gsum} + 5N_t t_{exchxyz} + 2N_t N_i t_{exchxy} \tag{12}$$

For a one-year atmospheric simulation, $N_t = 77760$ and $N_i = 60$. The predicted total communication time, using parameters values from Figure 11, is 30.1 minutes. The performance model also predicts that the total computation time T_{comp} is given by

$$T_{comp} = N_t \frac{N_{ps} n_{xyz}}{F_{ps}} + N_t N_i \frac{N_{ds} n_{xy}}{F_{ds}} \tag{13}$$

Substituting values from Figure 11, the predicted T_{comp} is 151 minutes. T_{comm} and T_{comp} total to 181 minutes which agrees well with the observed 183 minutes of wall-clock time.

	t_{gsum} (μsec)	t_{exchxy} (μsec)	$t_{exchxyz}$ (μsec)	$P_{fpp,ps}$ (MFlop/sec)	$P_{fpp,ds}$ (MFlop/sec)	F_{ps} (MFlop/sec)	F_{ds} (MFlop/sec)
F.E.	942	10008	100000	8.0	1.6	50	60
G.E.	1193	1789	5742	139	6.2	50	60
Arctic	13.5	115	1640	487	143	50	60

Figure 12: Potential Floating-Point Performance of an 2.8125°-resolution atmospheric simulation on a sixteen-processor, eight-SMP cluster interconnected by Fast Ethernet (FE), Gigabit Ethernet (GE), and Arctic Switch Fabric (Arctic). t_{gsum}, t_{exchxy} and $t_{exchxyz}$ are determined using stand-alone benchmarks.

5.4 Analysis using the Performance Model

Based on this performance model, we introduce a performance metric which we call *Potential Floating-Point Performance*, P_{fpp}. This quantity is the per-processor floating-point performance a numerical application would have if the numerical computations took zero time. This metric quantifies, for a given application configuration and hardware, the role that the communication performance plays in determining overall performance. P_{fpp} not only can characterize the balance of a system, but it can also determine the direction for improvement. If P_{fpp} is significantly greater than current processor compute performance then straight-forward investments in faster or more processors are a viable route to improving overall application performance. Conversely, if P_{fpp} is several times smaller than current compute performance then there is little point in investing in hardware that only improves compute performance.

P_{fpp} is computed as the total number of floating-point operations per processor divided by the communication time. For the **PS** and the **DS** phase of the GCM algorithm P_{fpp} is

$$P_{fpp,ps} = \lim_{F_{ps}\to\infty} \frac{N_{ps} n_{xyz}}{t_{\mathbf{ps}}} = \frac{N_{ps} n_{xyz}}{5 t_{exchxyz}} \tag{14}$$

$$P_{fpp,ds} = \lim_{F_{ds}\to\infty} \frac{N_{ds} n_{xy}}{t_{\mathbf{ds}}} = \frac{N_{ds} n_{xy}}{2 t_{gsum} + 2 t_{exchxy}} \tag{15}$$

Figure 12 summarizes P_{fpp} achieved by a variety of architectures during the **PS** phase and the **DS** phase of an atmospheric simulation at 2.8125°. The table compares the results from simulations running on a sixteen-processor, eight-SMP cluster interconnected by Arctic, Fast Ethernet, and Gigabit Ethernet.[1] Given a reference floating-point performance of about 50 MFlop/sec, the Gigabit Ethernet cluster's $P_{fpp,ps}$ value suggests that this architecture is viable for coarse grain scenarios where communications are large and infrequent. However, the table also indicates that the performance in the fine-grain **DS** phase of the GCM code would be severely limited by the performance of Fast Ethernet and Gigabit Ethernet. To achieve $P_{fpp,ds}$ of 60 MFlop/sec, the sum of t_{gsum} and t_{exchxy} cannot exceed 306 μsec. The Gigabit Ethernet hardware is nearly a factor of ten away from this threshold. This number suggests that Gigabit Ethernet clusters are not suitable for this resolution of climate model.

6 Discussion

It has been noted elsewhere[7] that single message overheads of 100 microseconds or greater present a serious challenge to fine-grain parallel applications. As we illustrate here, eliminating this bottleneck enables the application of cluster technology to a much wider field. The GCM implementation we have analyzed is not limited to coupled climate simulations. The MIT GCM algorithm is designed to apply to a wide variety of geophysical fluid problems. The performance model we have derived is valid for all these scenarios. Therefore, our analysis suggests that, for many useful geophysical fluid simulations, commodity-off-the-shelf (COTS) processors significantly out perform COTS interconnect solutions. In these cases, advanced

[1] The GCM code uses MPI to communicate on the Ethernet clusters.

networking hardware and software can alleviate the performance disparity by decreasing the denominator terms in Equations (14) and (15).

Obviously, floating-point performance comparable with that of Hyades can be achieved on state-of-the-art supercomputers[11]. Big supercomputers, however, are typically shared resources where the CPU time can often be "dwarfed" by the amount of time spent in the job queue. In contrast, the affordability of our cluster makes it possible to build a system that can be dedicated to a single research endeavor such that the turn-around time is simply the CPU time. As a consequence the Hyades cluster is a platform on which a century long synchronous climate simulation, coupling an atmosphere at 2.8^o resolution to a 1^o ocean, can be completed within a two week period.

In a related work, the HPVM (High Performance Virtual Machine) communication suite[8] also allows a powerful cluster to be easily constructed from low-cost PCs and a high-performance interconnect, Myrinet[4]. HPVM provides a collection of communication APIs as well as a suite of system management packages that make the cluster suitable for a broad span of supercomputing applications. The Hyades cluster differs from HPVM in its purpose and approach. Our cluster has a much narrower focus that emphasizes supporting a single application as efficiently as possible, and can be viewed as an *application-specific supercomputer*. This perspective motivates us to stride for every last bit of performance by developing communication primitives that are tailored to the application. Although a HPVM cluster's hardware components have comparable peak performance as the Hyades cluster, a sixteen-way global barrier on a comparable HPVM cluster takes more than 50 μsec (more than 2.5 times longer than Hyades's context-specific primitive). Similarly, a HPVM cluster's transfer bandwidth for 1-KByte blocks is about 42 MByte/sec (25% slower than Hyades's exchange primitives).

The Hyades cluster does have general-purpose, high-level programming interfaces, like MPI-StarT[18] and Cilk[28], that can make use of the high-performance interconnect. However, in an application-specific cluster, there is little reason to give up any performance for an API that is more general than required. The one-time investment[2] in developing customized primitives can easily be recuperated over the life-time of the cluster. Much greater effort has already been spent on developing the science and the simulation aspects of the application. Any real application, like GCM, is going to be used repeatedly and routinely to produce meaningful and useful results. We believe developing customized primitives that only support what is needed is actually a powerful and efficient strategy for rapidly adapting an application to on-going innovations in networking hardware.

7 Acknowledgments

This work was funded in part by grants from the National Science Foundation and NOAA and with funds from the MIT Climate Modeling Initiative[23]. The hardware development was carried out at the Laboratory for Computer Science, MIT and was funded in part by the Advanced Research Projects Agency of the Department of Defense under the Office of Naval Research contract N00014-92-J-1310 and Ft. Huachuca contract DABT63-95-C-0150. James C. Hoe is supported by an Intel Foundation Graduate Fellowship. The computational resources for this work were made available through generous donations from Intel Corporation. We acknowledge Arvind and John Marshall for their continuing support. We thank Andy Boughton and Larry Rudolph for their invaluable technical input.

References

[1] A. Adcroft, C. Hill, and J. Marshall. Representation of topography by shaved cells in a height coordinate ocean model. *Mon. Wea. Rev.*, pages 2293–2315, 1997.

[2] T. E. Anderson, D. E. Culler, D. A. Patterson, and the NOW Team. Case for networks of workstations: NOW. *IEEE Micro*, February 1995.

[3] B. S. Ang, D. Chiou, D. L. Rosenband, M. Ehrlich, L. Rudolph, and Arvind. StarT-Voyager: A flexible platform for exploring scalable SMP issues. In *Proceeding of SC'98*, November 1998.

[2] It took less than one-man month to develop the two custom primitives.

[4] N. Boden, D. Cohen, R. Felderman, A. Kulawik, C. Seitz, J. Seizovic, and W. Su. Myrinet: A gigabit per second local area network. *IEEE Micro Magazine*, February 1995.

[5] G. A. Boughton. Arctic routing chip. In *Proceedings of Hot Interconnects II*, August 1994.

[6] G. A. Boughton. Arctic Switch Fabric. In *Proceedings of the 1997 Parallel Computing, Routing and Communications Workshop*, June 1997.

[7] P. Buonadonna, A. Geweke, and D. E. Culler. An Implementation and Analysis of the Virtual Interface Architecture. In *Proceedings of SC'98*, 1998.

[8] A. Chien, S. Pakin, M. Lauria, M. Buchanan, K. Hane, L. Giannini, and J. Prusakova. High performance virtual machines (HPVM): Clusters with supercomputing APIs and performance. In *Proceedings of the Eigth SIAM Conference on Parallel Processing for Scientific Computing*, March 1997.

[9] D. E. Culler, A. Arpaci-Dusseau, R. Arpaci-Dusseau, B. Chun, S. Lumetta, A. Mainwaring, R. Martin, C. Yoshikawa, and F. Wong. Parallel computing on the Berkeley NOW. In *Proceedings of 9th Joint Symposium on Parallel Processing*, 1997.

[10] D. E. Culler, L. T. Liu, R. P. Martin, and C. O. Yoshikawa. LogP: Performance assessment of fast network interfaces. *IEEE Micro*, Vol 16, February 1996.

[11] M. Desgagne, S. J. Thomas, R. Benoit, and M. Valin. Shared and distributed memory implementations of the canadian mc2 model. In *Proceedings of the Seventh ECMWF Workshop on the Use of Parallel Processors in Meteorology*, pages 155–181. World Scientific, 1997.

[12] F. Molteni. *Development of simplified pametrization schemes for a 5-level primitive-equation model of the atmospheric circulation.* CINECA, 1999. available from http://www.cineca.it/adamo/doc/pe5l_param.html.

[13] F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow. *Phys. Fluids*, 8:2182, 1965.

[14] C. Hill, A. Adcroft, D. Jamous, and J. Marshall. A strategy for tera-scale climate modeling. In *Proceedings of Eigth ECMWF Workshop on the Use of Parallel Processors in Meteorology*. World Scientific, 1999.

[15] C. Hill and J. Marshall. Application of a parallel navier-stokes model to ocean circulation. In *Proceedings of Parallel Computational Fluid Dynamics*, pages 545–552. Elsevier, 1995.

[16] J. C. Hoe. StarT-X: a one-man-year exercise in network interface engineering. In *Proceedings of Hot Interconnects VI*, August 1998.

[17] J. R. Holton. *An Introduction to Dynamic Meteorology*. Wiley, 1979.

[18] P. Husbands and J. C. Hoe. MPI-StarT: Delivering network performance to numerical applications. In *Proceeding of SC'98*, November 1998.

[19] J. Marotzke, R. Giering, K. Q. Zhang, D. Stammer, C. Hill, and T. Lee. Construction of the Adjoint MIT Ocean General Circulation Model and Application to Atlantic Heat Transport Sensitivity. *J. Geophys. Res.*, submitted 1999.

[20] J. Marshall, A. Adcroft, C. Hill, and L. Perelman. A finite-volume, incompressible Navier-Stokes model for studies of the ocean on parallel computers. *J. Geophys. Res.*, 102(C3):5753–5766, 1997.

[21] J. Marshall, C. Hill, L. Perelman, and A. Adcroft. Hydrostatic, quasi-hydrostatic and non-hydrostatic ocean modeling. *J. Geophys. Res.*, 102(C3):5733–5752, 1997.

[22] J. Marshall, H. Jones, and C. Hill. Efficient ocean modeling using non-hydrostatic algorithms. *J. Marine Systems*, 18:115–134, 1998.

[23] *MIT Center for Global Change Science Climate Modeling Initiative.* see http://web.mit.edu/cgcs/www/cmi.html.

[24] R. S. Nikhil, G. M. Papadopoulos, and Arvind. *T: A multithreaded massively parallel architecture. In *Proceedings of the 19th International Symposium on Computer Architecture*, May 1992.

[25] J. Salmon, C. Stein, and T. Sterling. Scaling of Beowulf-class Distributed Systems. In *Proceedings of SC'98*, 1998.

[26] A. Shaw, K. C. Cho, C. Hill, R. P. Johnson, J. Marshall, and Arvind. A comparison of implicitly parallel multithreaded and data parallel implementations of an ocean model based on the Navier-Stokes equations. *J. Parallel and Distributed Computing*, pages 1–51, 1998.

[27] T. Sterling, D. J. Becker, D. Savarese, J. E. Dorband, and U. A. Ranawake. BEOWULF: A parallel workstation for scientific computation. In *International Conference on Scientific Computation*, 1995.

[28] Supercomputing Technology Group, MIT Laboratory for Computer Science. *Cilk-5.1 Reference Manual*, September 1997.

[29] Thinking Machines Corporation. *Connection Machine CM-5 Technical Summary*, November 1993.