On the Feasibility of Optical Circuit Switching for High Performance Computing Systems

Kevin J. Barker[2], Alan Benner[1], Ray Hoare[3], Adolfy Hoisie[2], Alex K. Jones[3,4], Darren J. Kerbyson[2], Dan Li[4], Rami Melhem[4,3], Ram Rajamony[1], Eugen Schenfeld[1], Shuyi Shao[4], Craig Stunkel[1], Peter Walker[1]

1 IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
eugen@us.ibm.com

2 Performance and Architecture Lab
Los Alamos National Laboratory
Los Alamos, NM 87545
djk@lanl.gov

3 Dept. of Electrical and Computer Eng.
4 Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15280
melhem@cs.pitt.edu

ABSTRACT

The interconnect plays a key role in both the cost and performance of large-scale HPC systems. The cost of future high-bandwidth electronic interconnects is expected to increase due to expensive optical transceivers needed between switches. We describe a potentially cheaper and more power-efficient approach to building high-performance interconnects.

Through empirical analysis of HPC applications, we find that the bulk of inter-processor communication (barring collectives) is bounded in degree and changes very slowly or never. Thus we propose a two-network interconnect: An Optical Circuit Switching (OCS) network handling long-lived bulk data transfers, using optical switches; and a secondary lower-bandwidth Electronic Packet Switching (EPS) network. An OCS could be significantly cheaper than an all electronic network as it uses fewer optical transceivers. Collectives and transient communication packets traverse the electronic network.

We present compiler techniques and dynamic run-time policies, using this two-network interconnect. Simulation results show that our approach provides high performance at low cost.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Circuit switching networks and network communications. B.4.3 [**Interconnections (Subsystems)**] fiber optics.

General Terms

Design, Performance

Keywords

Optical circuit switching, Network design, High Performance Computing, Performance Analysis.

SC|05 November 12-18, 2005, Seattle, Washington, USA

1. INTRODUCTION

High performance computing systems are being built out of ever-increasing numbers of processors. Blue Gene/L [1], which is the current leader in the TOP-500 supercomputer list as of June 2005, achieved that spot with a system containing 65,536 processors. The largest Blue Gene/L installation will be a 128K-processor system at Lawrence Livermore National Laboratory (LLNL) to be completed in 2005. Trends in microprocessor speeds indicate that systems with around 100K processors may be necessary in order to achieve petascale performance. A key component of these systems is the interconnect that connects together the system processors. In this paper, we examine a novel network structure with superior cost-performance benefits which we believe can be utilized in High Performance Computer systems: the Optical Circuit Switching (OCS) network.

Today's high performance computing systems use packet-switched networks to interconnect system processors. Inter-processor messages get broken into packets that are routed through network switches. InfiniBand, QsNet, switched Ethernet, and Myrinet networks are all examples of such interconnects. As systems get larger, a scalable interconnect can consume a disproportionately high portion of the system cost when striving to meet the twin demands of low-latency and high-bandwidth. The quest for cheap, packet-switched, low-latency, high-bandwidth networks to interconnect large numbers of processors is indeed worthwhile. However, it is reasonable to ask whether cheaper alternatives can be identified today to fulfill the needs of large-scale applications.

By empirically observing the communication patterns of a large number of high performance computing applications (eleven are described in this work), we find that two principles govern the bulk of the communication. First, barring collectives, the number of communicating partners for each computation

partition is bounded and typically small. Second, the set of partners with which a partition communicates either remains unchanged, or changes very slowly through the course of the execution. That these principles exist should not be surprising. Many high performance computing applications solve physical problems and model the problem domain using a graph-based structure (regular or irregular). Computation partitions communicate with partnering partitions in the graph domain. When techniques such as Adaptive Mesh Refinement (AMR) are used, new partitions are dynamically created and the communication partner set changes. However, the partner set only changes after a load balancing stage which typically occurs infrequently compared to the typical execution time of the application, as such a step is relatively expensive and may involve data or process migrations.

Based on this observation, we propose the use of two separate communication networks in a HPC system: one mechanism to accomplish long-lived transfers of large amounts of data, and another to accommodate collectives and short-lived data exchanges. Separating the communication classes in this manner enables us to target each class with the most appropriate network technology and operating mechanism.

For long-lived bulk data transfers, we propose an Optical Circuit Switching (OCS) interconnect, some components of which have been used in wide-area backbone networks and SONET cross-connects to support telecommunications traffic loads. These switches use optics at all elements of the data path. Setting up a circuit (or switching) is typically accomplished through the use of MEMS-based (Micro-Electro-Mechanical Systems) mirror arrays that physically move the light beam to establish an optical data path between any input and any output port. Once a circuit has been established, communication between the end-points occurs at very high bandwidth and very low latency. Furthermore, since all data paths are optical, the switch is nearly impartial to the distance between communication end-points. In other words, two distant end-points can communicate at equal bandwidth and near-equal latency as two neighboring ones.

We use the term *communication partner* throughout this paper instead of *communication neighbor*. The term *neighbor* connotes a physical closeness which our circuit switches are explicitly directed at eliminating - i.e., on a good circuit switch, everybody is a neighbor (physically equally close, essentially), and the communication partners can be changed on demand by re-configuring the circuits.

Optical circuit switches can be relatively inexpensive and power-efficient compared to their packet-switched counterparts. Because optical switches directly manipulate the light beams, without any electronic processing, no optical to electrical to optical (O-E-O) conversions are needed. This results in tremendous savings of very expensive optical transceivers which are needed to link high bandwidth electronic packet switches. Hence, optical fibers can be used to interconnect high bandwidth signals in between the cabinets of HPC processors and switches. All optical switches do not need such conversion, as they operate directly on the high bandwidth optical signal a fiber delivers and passively redirect such optical signal into another fiber. The main drawback of these MEMS-based switches is the relatively long time (order of a few milliseconds) they need to reconfigure their connections. However, since the circuits established using the OCS network will be used for long-lived data transfers, the slower switching speed will not be a performance impediment. For collectives and transient communication, we propose a secondary non-high-bandwidth electronic packet network. With its high switching speed, the secondary interconnect is able to handle collective and transient traffic with low latency. At the same time, we are able to keep the secondary interconnect from being overwhelmed by using it for only a small portion of the overall traffic. The cost-performance advantages of the combined interconnect come about because the components of each network are optimized to handle the kind of traffic for which they are best suited to handle.

For this paper, a detailed cost analysis is inappropriate since it would of necessity be too speculative. We have strong expectations that the costs of OCS will be significantly cheaper, but the uncertainties in the expected costs and cost take-down trends are large enough that it would distract from the paper's other results to even address it.

Finally, we would like to emphasize that this paper contains many ideas and areas of work that are separate research topics in their own right. However, we feel it is important to give a global picture of the OCS system architecture and how various research aspects relate to it. By no means is it intended to give complete details since this would be well beyond the scope of this paper. Further details in each research area including compiler work, performance

simulations, application analysis, and system architecture will be published in due course. We hope the reader will benefit from a description of the interdisciplinary mix of research areas that are considered when a new architecture is proposed, and this will help to illuminate the various inter-dependencies of the work in this paper.

1.1 Related and Previous Work

Perhaps the most well known example of a circuit switching based network is the NEC Earth Simulator [18]. The Earth Simulator (ES) network uses a huge electronic crossbar, with 640x640 ports. This crossbar is made of 128 independent data path switches forming circuits, each handling one byte of data, and a control unit that is used to setup these data path switches. Each processing node has 8 separate channels which are 13 bytes wide (data), and 3 bytes wide (ECC), that are used to communicate with other nodes. Thus each of the eight vector processing units in a processing node can share the 8 channels and the circuits formed by the data switches to communicate to other nodes.

The ICN (Interconnection Cache Network) [16,17] is similar to the ES in its one-to-one relationship between each processing node and one channel or circuit that it handles. As with the ES, in ICN the processing nodes are grouped into small clusters, such that a small electronic packet switch can be used to share any circuit already established to other clusters with any one of the group's processing nodes. In the ES case, the size of such an ICN cluster is 8 (for 8 vector processing units in a node) and 8 circuits that each cluster (node) can form to other clusters (nodes).

Our approach has two distinctive improvements. First, the ratio of the number of circuits to the number of processors in a node can be greater than one-to-one (i.e. each node may have an independent network interface controller (NIC) handling multiple links and thus multiple circuits). Secondly, we add a separate low-bandwidth network to handle small-traffic communication needs and collective operations. Although, an Electronic Circuit Switch (ECS) has faster circuit setup and release times, for some AMR codes and other irregular communication patterns with a switching degree need larger than 8, the ES may perform poorly. An example of a study of four applications is given in [28]. However the authors are yet to investigate the performance of AMR type applications which may not present a good match for the ES network, for the above reasons.

The Blue Gene/L machine also uses a coarse form of circuit switching intended to help with system partitioning and job scheduling [2,3]. Mid-plane sections having 512 nodes, connected as a 3-D torus, use X, Y and Z reconfigurable switches. These switches are used to partition up to 64K nodes (with 128K Processors) into smaller partitions that maintain a 3-D torus topology. The switches are reconfigured per job allocation forming a partition, and are not changed while a job is running. This is a coarse, per job, type of circuit switching, and has limitations in matching to the more complex communication patterns that, for example, HPC applications exhibit.

Previous work has also attempted to split traffic and carry it over different networks, however with a different architecture and for a different use. For example the Gemini and Clint projects [8,14] both use multiple networks for communication between processor nodes.

The Gemini architecture consists of a dual multistage network fabric. One such fabric is made of optical 2×2 cross connect switches, made to operate in a circuit switching mode, while the second network is made of regular 2×2 electronic packet switches, and is also used to setup the optical circuits of the first network. Hence, each node has two ports. The optical circuit switching port is used for long lasting transfers, but due to its circuit switching nature, has a relative higher setup time. Since each node has only one port of each (optical and electrical) into the dual network structure, setting up an optical circuit will tie that port to only one specific destination. If there are long messages that need to be transferred to only one specific destination, such a circuit will be advantageous. However, as we will see in Section 3 HPC applications require more than one destination per node (rather a partner set of such nodes).

The Clint architecture is similar to Gemini, in that two types of switches are used, although no optical technology is used. Both switches are implemented using VLSI technology but their structure is different. The "Bulk Switch" is made for electronic circuit switching, and thus has no buffers or per packet arbitration. The 2×2 switch is setup centrally, establishing a circuit from one processing node to another. As with Gemini, the Clint system uses the Bulk Switch for long messages that are not sensitive to latency. Latency is considered higher for these circuit switching paths, since it takes a long time to establish a circuit and then tear it down. In the Clint system, as with Gemini, there is no sharing of already

established circuits. Each node will have to setup its own circuit and hopefully will have enough data to send through this circuit before it needs to tear it down and setup another circuit to another destination.

Finally, related distributed computing work attempts to use the Internet as the network between large processing sites. The computing performed over large distances communicates through optical fibers, with potentially different TCP/IP protocols. Examples are the Cheetah and OptIPuter projects. [11,33]. Both Cheetah and OptIPuter are aimed at very-widely-distributed computing (i.e., 100's of km - between sites), and result in different network costs than those in HPC clusters. Cheetah's architecture is aimed at solving the problem of transferring huge files (Terabytes to Petabytes) between geographically separated research labs. A circuit is setup as a means to enjoy better performance than the use of standard TCP/IP for remote file transfers. OptIPuter's architecture is aimed at linking together clusters of various types (particularly storage clusters and visualization clusters with compute clusters) over similarly long-distances. Neither allows, for example, the running of fine-grained HPC applications which are of interest here.

1.2 Paper Outline

In order to be viable, optical circuit switching must overcome several nontrivial challenges. What circuits should be established in the switch and when? How should *random* communication patterns be handled? What happens when the communication degree of an application exceeds the degree of connectivity provided by the OCS? We address all of these questions in this paper.

An architectural description of a high performance computing system incorporating optical circuit switching technologies is given in Section 2. In Section 3 we provide an application analysis that makes the case for using an optical circuit switching approach. Section 4 describes how optical circuit switching can be incorporated into a high performance computing system, and the software support necessary in order for seamless integration. Conclusions on this work are given in Section 5.

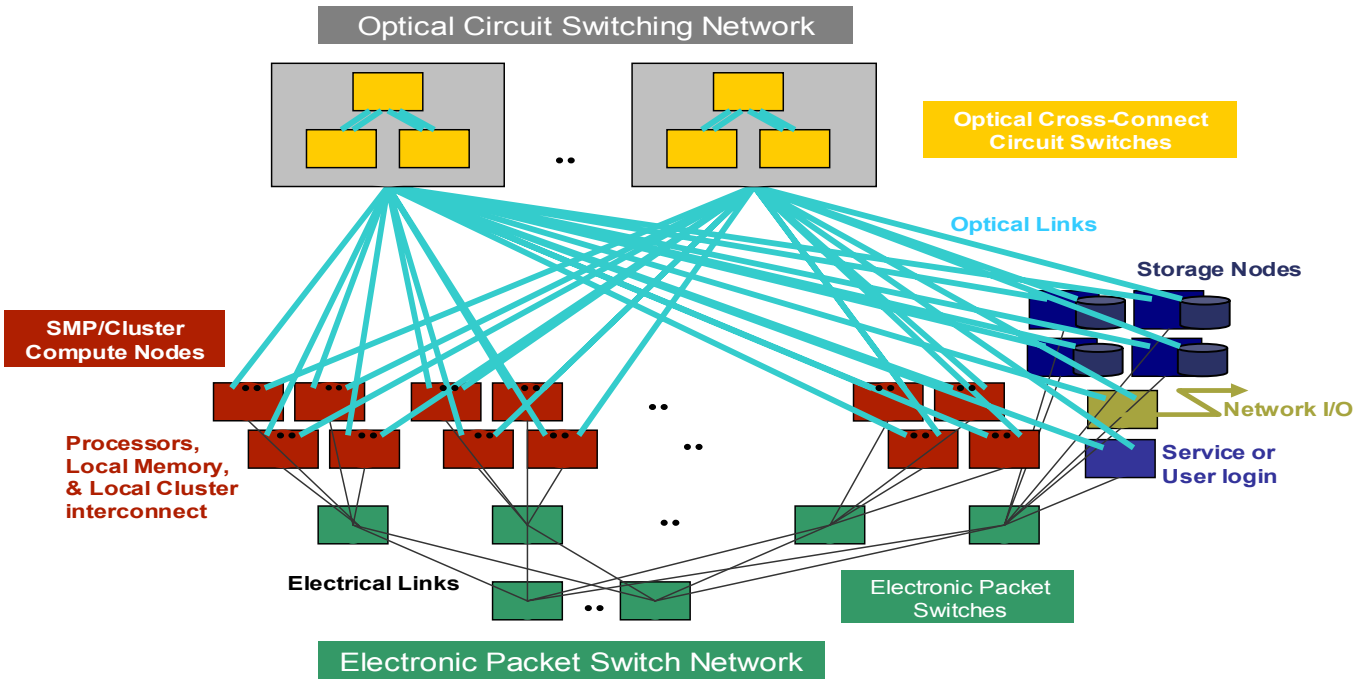


Figure 1. System design containing two complementary networks that handle i) small-degree high-bandwidth communications (Optical) and ii) large-degree low-bandwidth communications as well as collectives (Electronic).

2. OCS-BASED SYSTEM DESIGN

The system considered here is for HPC applications that currently execute on large-scale clusters or parallel-processing machines, such as those used at national laboratories or supercomputing centers. In these systems, thousands of processors, and their associated memory and disk storage subsystems, are tightly coupled together across a high-performance network. Typically, the network delivers a bandwidth up to tens of Gb/s to each processor, and the latency across a network with thousands of nodes is several microseconds. As systems' throughput requirements steadily grow, the bandwidth delivered will approach 100's of Gb/s per processor across networks supporting systems in the order of 100,000 processors.

Currently, supercomputing systems are typically interconnected using fat-tree networks (indirect or multi-stage interconnection networks) of electronic packet switch chips or switching elements, or direct networks such as 2-D or 3-D mesh/torus topologies. There are many topology variations possible, depending on full system size and switching elements used.

Our proposed system is made of two complementary networks as shown in Figure 1. It has the following components:

Compute nodes. These contain multiple processors, their associated memory, storage, and network interface components. The organization of these is relatively opaque, as far as the OCS network is concerned. That is, they may consist of SMP (Shared Memory Processor) nodes sharing a single operating system, or clusters of SMPs, or clusters of single-processor nodes. In all cases, the compute nodes are assumed to contain at least one NIC (Network Interface Chip) or HCA (Host Channel Adapter), and some local electronic switching capability to allow traffic from the compute nodes to be distributed across at least one plane of electronic packet switches, and several planes of optical circuit switches. The compute nodes have optical transceivers to provide fiber interfaces to the network switches.

Optical Circuit Switching (OCS) network. This network is made of multiple planes of all-optical switches. Circuits are established between inputs and outputs of this network, inter-connecting two compute nodes via a link in their NICs. These circuits can be shared among the communications going from any processing element in one compute node to any processing element in another node. Latency through this network is lower than through the Electronic Packet Switch (EPS) network once circuits are established, since electronic buffering and O-E-O conversions are not required.

Electronic Packet Switch (EPS) network. This network is made of *standard* switches which do not have a particularly high bandwidth. When trying to increase both the radix (port count) and the bandwidth per port of each switch, the resulting electronic switch becomes too complex and expensive to make. However, making a high radix electronic switch, with low bandwidth per port is feasible. Such a switch could also contain support for collective operations, aiding HPC applications that have a critical performance dependence on collectives. Our initial study shows that using 10% of the OCS bandwidth, for the EPS is a good design point.

Details of our proposed system and technology are described in the following sections.

2.1 Optical Circuit Switches

An important distinction in switch design is that between packet switches (where the duration of an input-port-to-output-port connection lasts for the period of one packet), and circuit switches (also sometimes called optical cross-connects), where such a connection is maintained for the duration of many packets (typically millions).

Several optical packet switch architectures have been described and demonstrated. For example, [20] describes an optical packet switch based on using fiber- and wavelength-selection filters at output ports to select signals broadcast from each input, and [15] describes an optical packet switch based on wavelength-tunable transmitters and an array waveguide grating capable of routing various wavelengths to various output ports. Such architectures are capable of switching traffic on a per-packet basis, with switching times (i.e. the time for data from one input to be switched from one output to a different output) of less than 10 nanoseconds. Designing arbiters for such a packet switch, however, is extremely challenging – switch element arbiters capable of arbitrating across 64 ports within a packet duration of roughly 50ns (256 bytes at 40Gb/s) are near the limits of aggressive design. Also, designing

switching technologies capable of reconfiguring connections in less than 10ns or so is challenging.

For this work, however, we consider network architectures using optical circuit switches; switches where the duration of an input-port-to-output-port connection lasts for the period of typically millions of packets (at least several milliseconds). Use of circuit switches rather than packet switches dramatically simplifies the arbiter design. Such distinction may also widen the range of technology options available to build the interconnect.

The optical circuit switching technologies with the best promise of providing the highest switch radix and throughput are two-dimensional MEMS devices, as described in [29]. MEMS optical switches have been shown to scale to the order of 1024 input and output ports, and switches with 100s of ports are generally available. They could feasibly support wavelength division multiplexed data per port of up to 100's of Gb/s, with appropriate transceivers. The most promising devices operate on the principle that two arrays of tiltable mirrors can redirect the photonic signals from each input fiber to one of the output fibers.

It is generally true that switch elements with higher radix and throughput (bandwidth across switch ports) are preferable in order to reduce the number of switching stages and switch-to-switch links that use more expensive optical transceivers. Electronic packet switches are strongly limited by the capabilities of silicon integrated circuits. A single switch element chip with throughput of ~1.2 Tb/s (e.g. 512 signal I/O pins at 5 Gb/s per pin pair) would be a very aggressive chip design, and a supercomputer network providing, for example, 40 Gb/s to each of 8,000 nodes would require a multi-stage network with several stages and many inter-switch links substantially increasing the cost of such a network. Higher node counts, in the order of 32,000 or 64,000 nodes, could mushroom the overall cost of the network and the system, due to the use of expensive high bandwidth optical transceivers between the electronic switching stages.

2.2 Handling Collectives and Low-Bandwidth Communication Needs

Optical circuit switching could very well provide the primary communication path for high performance applications. However, there are types and patterns of communication which are not natural matches for OCS networks. Specifically, some high performance computing applications have an amount of random or global communication, and many also make heavy use of collective operations. These applications can benefit from embedded collective communication support in the electronic switching network (not merely at the end-points or network interfaces).

Here we briefly examine the issue of collective support. Within switches, this support can be broadly categorized into two basic types: replication and combining. Collective operations such as broadcast, barrier synchronization, reductions, and all-reduce (reduction followed by a broadcast) can be implemented by sequences of these two basic types. Multicast replication is an operation that is increasingly supported in commercial electronic computer system interconnects such as InfiniBand switches. Some optical switching technologies may also be natural matches for replication. However, to be inexpensive, OCS technologies are based on MEMS technology (movable mirrors) which are not inherently capable of replicating a beam of light. The other basic collective operation—combining—is an operation which requires logic processing, and no practical optical solutions exist. There are a number of examples of combining support within electronic switches. Perhaps the best-known is the Blue Gene/L machine [1], and other examples include the Cray T3E [30] and the Thinking Machines CM-5 [26]. Support for operations such as barrier synchronization and reductions can be made reliable without excessive difficulty [31].

Therefore, we propose to handle these global and collective communication requirements through the use of the secondary EPS network. This network is expected to be used sparingly, and is therefore constructed to provide low-latency communication to partnering processors at low-bandwidth levels.

The other use of this network is to handle communication *exceptions*. These are low-bandwidth communications that do not merit the overhead and expense of setting up a circuit in the OCS. Since we want to optimize the OCS and its ability to handle the main fire-hose data circuits, we would use the electronic network also for those cases of low-bandwidth communication among the processing nodes in the system. This will help bound the set of application communication partners to the number of links or circuits that can be accommodated by the OCS network.

2.3 OCS Network Latency

The cost to setup a circuit and tear it down in architectures such as Gemini [8] makes the latency cost higher than when using other packet switching networks. With OCS, multiple circuits are shared amongst groups of processing nodes, and circuits do not need to be setup and torn down as often. Thus the latency in our OCS circuit switching network can actually be lower than in a packet switching network, as is further described below.

Both the Gemini and Clint systems are limited in the ways a node can communicate with other nodes in the system. The sharing of many circuits among clusters of nodes in the OCS system allows the network to match the communication patterns of HPC applications, without the need to keep resetting established circuits between processing nodes. Once a circuit is setup, it can be repeatedly used by any member of a node to communicate with any member of the remote node. Hence, this mechanism actually results in a lower latency of communication for packets or messages passing through a circuit.

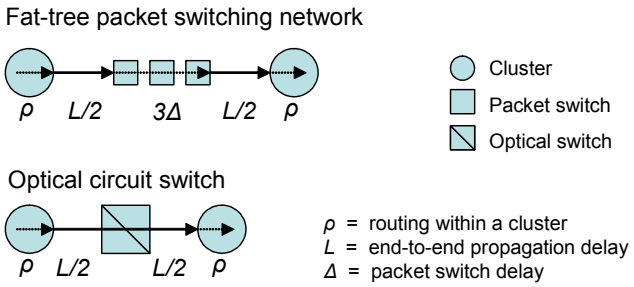


Figure 2. Comparison of latency cost in a fat-tree packet switch and optical circuit switch networks.

A comparison of the latency through a fat tree or a multi stage packet switching network and through an OCS network is shown in Figure 2. For a typical routing delay within a cluster of 100ns, a packet switch delay of 50ns, and an end-to-end propagation delay of 300ns (60m at 5ns/m), the fat-tree packet switch delay is 650ns. In comparison the delay for the optical circuit switch is only 500ns (the transmission time through the optical switch is negligible).

They key difference in achieving a lower latency in the OCS is in the reuse of already established circuits. If a compute node has only one circuit switch port it can access, such a reuse will not be practical. The sharing of many circuits in the OCS architecture by a compute node allows a match to what many HPC applications need (as presented later). Hence, our OCS system achieves a better latency than regular packet switching, and avoids the contention, queuing and arbitration at the packet switches that needs to happen on a per packet basis. Such queuing delays could add latency especially with large packets and messages. Latency is one of the important factors affecting the overall performance of HPC applications which makes grid or internet based approaches, such as the OptIPuter or Cheetah, impractical.

3. COMMUNICATION REQUIREMENTS

There are several aspects of the communication pattern exhibited by an application that are important when considering the utilization of a dual OCS and EPS network architecture. The most important of these are:

Bounded communicating partner set. The set of partners from each processor should be bounded. Ideally the communication degree should be small, where the communication degree here is taken to be the maximum size of the set of partners across all processors. Bounding the communication degree facilitates a limited number of circuits that need to be established within the OCS network.

Slowly changing communication partner set. The set of partners should ideally be slowly changing. Establishing an optical circuit can take several milliseconds when using MEMS technology due to the mechanical rotation of mirrors. This cost is amortized over the time in which the communicating partner set persists, and thus is negligible if the partner set slowly changes – in the order of seconds.

Mapping of communicating partners to the dual networks. Communicating partners with high-bandwidth requirements should utilize the high-bandwidth OCS. Similarly, communicating partners with low-bandwidth requirements should utilize the secondary low-bandwidth electronic network. The assignment of communications in this way will also aid in bounding the size of the communicating partner set on the OCS and hence the overall effectiveness of the OCS.

The "degree of communication" of the application is an important concept here. In the OCS architecture, it matters how many communication partners a node has (whereas it doesn't matter so much in other network architectures such as Gemni or Clint [8,14]). One of the key points of the OCS architecture is that, if application requirements are such that a node needs to communicate with k partners, then the node will

need to have access to at least k planes of optical circuit switches to achieve good performance.

There are actually two possible definitions for the "degree of communication": 1) the number of partners a node sends data to (unidirectional circuits) or 2) the number of partners a node exchanges data with (bidirectional circuits). In this paper we assume the later definition (i.e., every circuit we set up is actually a bidirectional link), since (a) we need a return channel for flow control and acknowledgements of intact data transmission (although these could pass through the electronic network in the case of a unidirectional setting) and (b) most applications require bidirectional data exchanges – there are exceptions which may not have a symmetric bandwidth requirement per link. However, our simulator and method used to setup a circuit, handles each direction independently (hence a circuit will be setup from one cluster to another will be unidirectional, so it may take two setups, one on each side and could be on a different switching planes).

3.1 Analysis Approach

In order to analyze the communication requirements of an application, dynamic call-graph information and message passing activity is collected from executing the application. Logged events include: subroutine entry and exit, as well as communication events such as collectives and point-to-point messaging. Subroutine entry and exit events are produced from a source-to-source translation with instrumentation calls placed at all subroutine entry and exit locations, and the PMPI (MPI Profiling) interface is used to seamlessly instrument the communication calls. Point-to-point communications simply record the source, destination, size, and data-type of the messages. Similarly, the sizes and data-type of collective communications are also recorded.

All events are time-stamped, and in a similar manner to Paradyn [6], the concepts of inclusive and exclusive metrics are used to record the duration within a subroutine. Trace files are collected per processor, and are made available in separate files for post-processing. The instrumentation and profiling method has been described [32] and was originally used to compare differences in processing flow across processing threads. The source-to-source instrumentation is similar as that used in other tools such as TAU [4]. Communication patterns can be constructed using this trace data as presented in [24] and are further analyzed here to examine the three issues listed at the beginning of Section 3.

A communication matrix is considered at each communication call-point in an application. A communication call-point includes the prior call-stack to the communication point. From our experience, this appears to be an appropriate level of detail at which to analyze the communication patterns. It has proved to be sufficient to reveal all the different communication patterns that occur across a range of scientific applications.

A communication call-stack is defined to be

$$R_1 \text{ -> } R_2 \text{ -> } \ldots \text{ -> } R_n \text{ -> } C_{CP} \qquad (1)$$

where R_i denotes a specific routine in the application, C_{CP} denotes the communication call-point, and routine R_i calling routine R_j is denoted by R_i -> R_j.

The use of the call-stack for each communication call-point is important as many applications employ the use of a communication layer which in turn is used to call the message passing library such as MPI. In addition, a communication matrix can be constructed for sub-sets of an application's entire execution. For instance, if the application's communication requirements change dynamically, separate matrices can be constructed for each iteration.

3.2 Sample Applications Analyzed

The communication requirements of a broad range of applications have been analyzed, many of which have their origins in either the Department of Defense (used in the HPC modernization program), or the Department of Energy (the Accelerated Strategic Computing program, and from the Office of Science). Most applications are in active production use on large-scale machines. Many are also being used within the DARPA High Productivity Computing Systems (HPCS) program. It is not the intention of this paper to provide details on the applications themselves, but rather to analyze their suitability to the use of a dual OCS / EPS network architecture. The applications analyzed are:

CAM – the atmosphere component of the Community Climate System Model (CCSM), used in both stand-alone and coupled simulations [9].

CICE – the sea ice component of the CCSM simulating multiple layers of ice and snow [5].

HYCOM – the Hybrid Coordinate Ocean Model implements a general circulation model which accurately represent depth changes from stratified

open ocean, to shoreline regions. It is available from NRL [19].

KRAK – a Lagrangian hydrodynamics application that originates from Los Alamos National Laboratory (LANL).

LBMHD – this application simulates a charged fluid moving in a magnetic field using a Lattice Boltzmann formation of the magneto-hydrodynamics equations. It originates from Lawrence Berkeley National Laboratory (LBNL).

PARTISN – this application is a comprehensive implementation of S_N transport, the solution of the Boltzmann equation using the discrete ordinates method, on structured meshes. This code originates from LANL.

POP – the Parallel Ocean Program is the ocean circulation component of CCSM. POP is used extensively in coupled simulations. It originates from LANL [22,25].

RF-CTH2 – an adaptive mesh shock-dynamics application that originates from Sandia National Laboratory (SNL) [10]. It is a part of the DOD technology insertion benchmark suite.

SAGE – an adaptive mesh (AMR) hydrodynamics application used for the simulation of shock-waves. Its performance characteristics have been extensively studied [23].

SWEEP3D – A kernel application implementing the main processing involved in deterministic S_N transport calculations. It originates from LANL [21].

UMT2K – A further implementation of S_N transport but on unstructured meshes. It is available from LLNL. Details on the performance of UMT2K can be found in [27].

3.3 Communication Degree Analysis

Each application was instrumented and trace files collected from a 128 processor execution apart from HYCOM (124 processors), and CICE (100 processors). The data was collected on a 32 node, 4-way, ES40 AlphaServer with a QSNet-1 fat-tree network. A summary of the communication patterns, whether they are static (i.e. unchanged throughout the application execution) or dynamic, and their communication degree is listed in Table 1.

Table 1. Details of the sample applications analyzed and their communication patterns.

Application	Static / Dynamic	Communication Patterns Observed	Degree	Comments
CAM	Static	1-D partitioning	6	Communicates with first and second partners in X
CICE	Static	2-D partitioning, Master/Slave	4 *P-1*	Cyclic in X
HYCOM	Static	2-D partitioning modified, Row / Column reduction	6 12	Land only sub-grids are removed resulting in a *modified* 2-D partitioning
KRAK	Static	Irregular	8	Irregular mesh whose structure can change but infrequently
LBMHD	Static	2-D partitioning	4	Cyclic in X and Y
Partisn	Static	2-D partitioning	4	Direction of communications differs across application phases
POP	Static	2-D partitioning	4	Cyclic in X, land only sub-grids are ignored in the computation.
RF-CTH2	Static / Dynamic	3-D partitioning modified	6 / 45	Communication is local partner. Some communications are phased.
SAGE	Static / Dynamic	1-D partitioning modified, reduction/broadcast	6 / 34	Pattern can change in each iteration when using AMR and load-balancing
Sweep3D	Static	2-D	4	Direction of communications differs across
UMT2K	Static	Irregular	16	Unstructured mesh partitioned.

Note that communication patterns listed are those implemented in the applications by point-to-point communications. The main communication pattern for CICE, LBMHD, Partisn, POP, and Sweep3D is logically two-dimensional resulting from a 2-D partitioning of a 3-D data domain (except for LBMHD which partitions a 2-D data domain). This results in at most four communication partners per processor and hence the applications have a communication degree of four. Note that in CICE and POP, the communication is cyclic in the logical X dimension, and in LBMHD is cyclic in both the logical X and Y dimensions. CAM uses a 1-D partitioning of a 3-D data domain and each processor communicates with at most six partners. For these communication patterns, the communication degree is low and easily handled by an OCS network.

CICE has a master-to-slave (one-to-all) pattern having communication degree *P-1* (the number of processors minus one). However, this operation is infrequent and could be mapped to the electronic network.

The communication pattern for HYCOM results from a 2-D partitioning of a 3-D data domain, but sub-grids representing only land are removed from the computation. This arrangement results in 'holes' in the 2-D processing array which are subsequently filled by shunting adjacent sub-grids. HYCOM has a low communication degree for the general boundary exchanges. However, a two stage software reduction is performed across rows and then up the root column of the sub-grids. Thus the communication degree in these stages is the order of $\sqrt{p}$.

SAGE and RF-CTH2 contain the only dynamically changing communication patterns – all other applications have a static communication pattern with low degree. We will concentrate further analysis on these two applications. Both can be used in two modes – with AMR (Adaptive Mesh Refinement), or without. When using AMR, grid-points in the physical simulation can be generated and/or removed dynamically, and load-balancing is employed to keep the number of grid-points across processors constant. However, the change in the mapping of grid-points to processors affects the logically partnering processors that communicate and thus alters the communication pattern on an iteration-to-iteration basis.

The main communication pattern in SAGE is based on a 1-D partitioning of a 3-D data grid but, due to the AMR, the communication is not always nearest neighbor [23]. The main communication pattern in RF-CTH2 is based on a 3-D partitioning of a 3-D data grid. The time between load-balancing in both SAGE and RF-CTH was the order of tens of seconds on the measurement system. Thus the cost of establishing the necessary optical circuits in the OCS after each load-balancing stage, of several milliseconds, would be negligible. The communication degree listed in Table 1 is the maximum over all iterations of two input decks to SAGE, and similarly is the maximum over all iterations on one input deck to RF-CTH.

The communication degree per iteration is depicted in Figure 3. It can be seen that the communication degree varies on a per application cycle basis and is between 11 and 24 for SAGE when using the two AMR input decks TimingB and TimingC (which differ in the physical simulation processed), and between 8 and 127 for an AMR input deck to RF-CTH. Note that there is an initialization in RF-CTH which has a high communication degree requirement. This occurs early in the processing and could be mapped to the low-bandwidth electronic network without substantially affecting the overall application execution time.

3.4 Partitioning Traffic between OCS and EPS

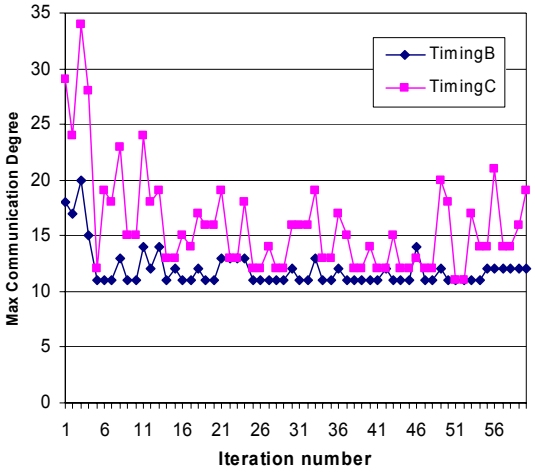
The calculation of the communication degree as shown in Figure 3 for both SAGE and RF-CTH is based on all communicating partners in each iteration. However, in the dual OCS and EPS architecture, only communicating partners with high-bandwidth requirements need use the OCS, and those with low-bandwidth requirements will use the EPS network.

In order to analyze the partitioning of the messaging across the two networks, a threshold filtering is applied to the communication pattern as follows:

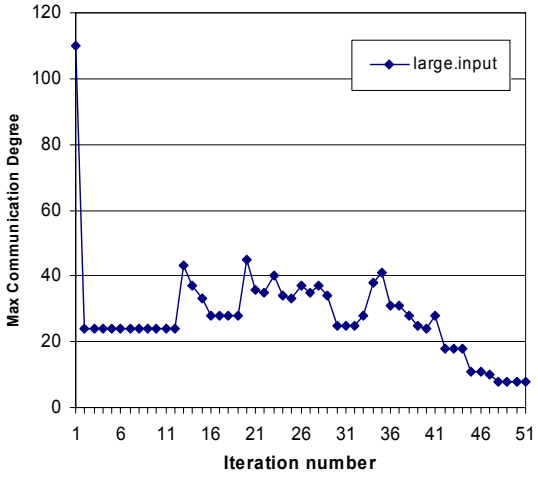
1) For processor P_i calc total bytes sent (T_{Bi})
2) For P_i, order communications P_i -> P_j in terms of bytes sent
3) Remove the lowest set of communicating pairs, P_i->P_j, whose combined communication volume is < T_{Bi} * THRESHOLD from the communication degree calculation

where THRESHOLD is in the range 0.0 to 0.2 but will typically be equal to 0.1 since the relative bandwidth of the OCS to the EPS network is expected to be 10:1. Note that this filtering partitions the communication traffic from each processor so that only a given percentage will be eliminated from the OCS and is mapped to the electronic network.

The affect on the communication degree after applying this threshold on SAGE are shown in Figure 4. Note that a threshold of 0.0 does not eliminate any partners from the communication degree calculation.



(a) SAGE



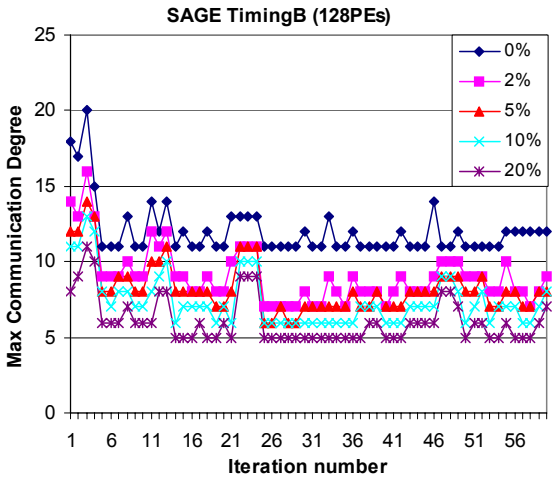
(b) RF-CTH

Figure 3. Variation in communication degree.

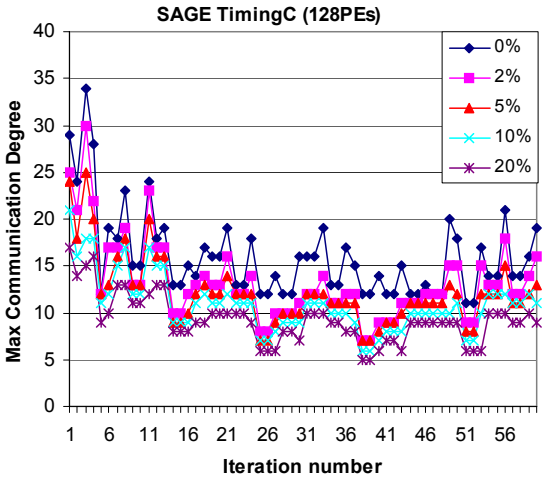
It can be seen that the communication degree of SAGE can be substantially reduced using this threshold filtering method.

Shown in Figure 5 is the percentage of traffic that is actually eliminated from the OCS and mapped to the electronic network after applying the threshold filter. Since the criteria for communication elimination takes the set of processors whose traffic is less than a threshold of the total, the exact amount of traffic eliminated will be less than the threshold. The amount of traffic eliminated will also depend on the distribution of traffic amongst the communicating partners, but it is guaranteed to be less than T_{Bi} *THRESHOLD for each processor.

A similar analysis is shown in Figure 6 for RF-CTH. It can again be seen that the threshold filtering



(a) SAGE (TimingB input)



(b) SAGE (TimingC input)

Figure 4. Communication degree after threshold filtering.

operation significantly reduces the communication degree by mapping the communications with low-bandwidth requirements to the EPS network.

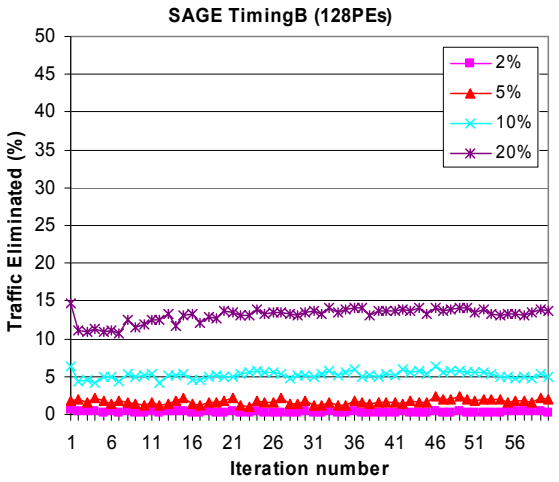
3.5 Communication Degree Summary

The communication behavior of these applications is summarized by considering both the communication degree and its dynamic behavior. This is shown in the spatio-temporal graph in Figure 7. Figure 7(a) shows the overall communication degree of the applications prior to threshold filtering, and Figure 7(b) shows the communication degree after threshold filtering. The

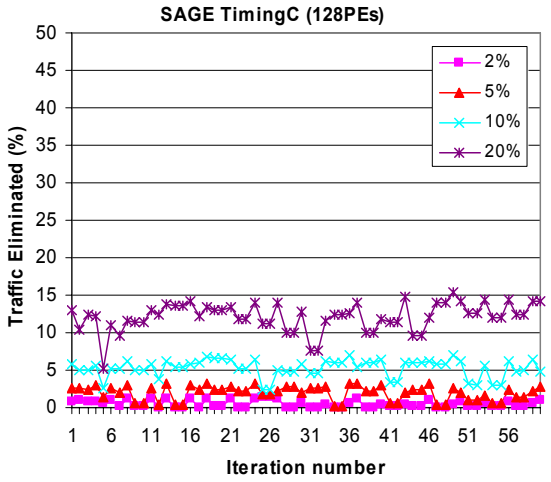
metric of rate-of-change is used to indicate the frequency of communication pattern change on the measurement system. This is plotted in Hz and the time between changes is simply the reciprocal. A high rate of change will require an increased number of optical circuits to be established during run-time and thus increase the overhead of the OCS. The overhead of establishing the OCS circuits, in terms of the percentage increase in application run-time, is shown on the right-hand Y-axis in Figure 7 based on a MEMS set-up time of 3 milliseconds.

For the applications of CAM, CICE, LBMHD, Partisn, POP, Sweep3D, HYCOM, and UMT2K, the rate of change is zero – i.e. there is no change in the communication pattern, and the communication degree varies between 4 and 16. This can be seen by the points on the X-axis in Figure 7(a). KRAK has a very slowly varying pattern and it's rate-of-change is slightly above zero. For SAGE, and RF-CTH the communication pattern is dynamic. The range in communication degree from its minimum to maximum and the range in the rate-of-change over application iterations are shown in Figure 7 by the extent of the bars. For instance the communication degree in SAGE varies between 11 and 34, while the rate-of-change varies between 0.004Hz and 0.029Hz (or time between load-balancing steps varies between 34.5s and 250s).

Figure 7(b) shows the same analysis for SAGE and RF-CTH but after the threshold filtering, at a level of

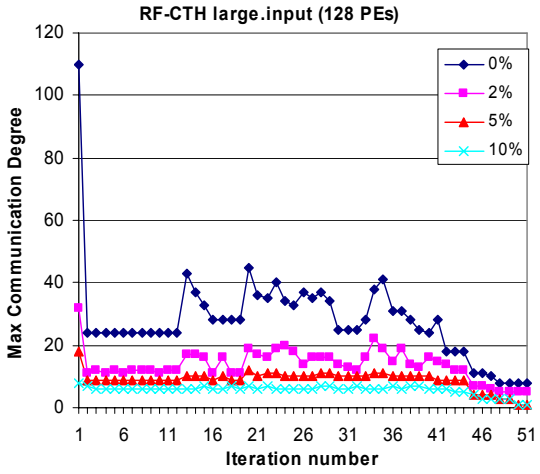


(a) SAGE (TimingB input)

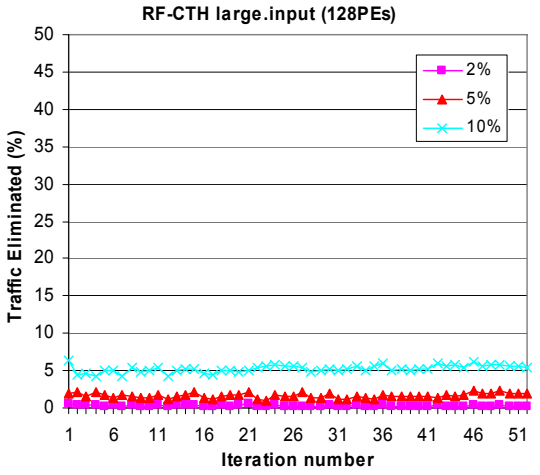


(b) SAGE (TimingC input)

Figure 5. traffic eliminated from the OCS (mapped to the EPS network) after threshold filtering.



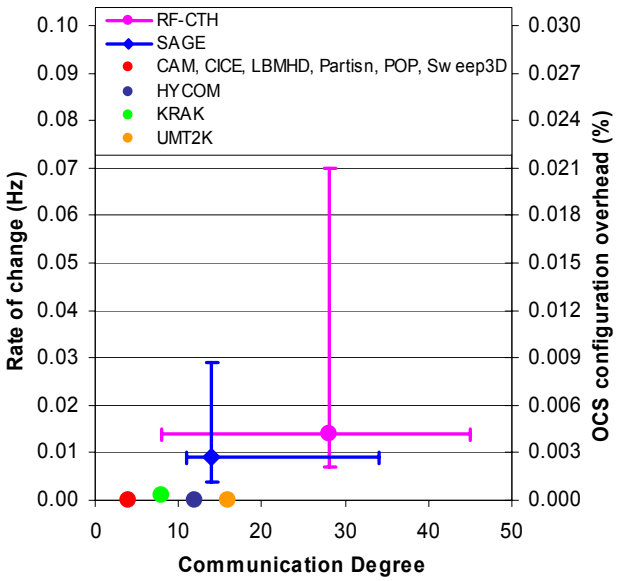
(a) Communication degree after threshold



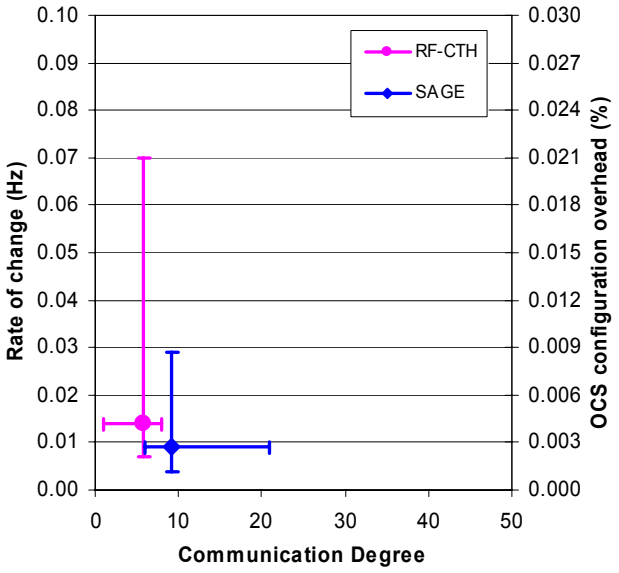
(b) Traffic eliminated from the OCS

Figure 6. Filtering analysis (RF-CTH)

10%, has been applied. The threshold filtering reduces the communication degree as shown earlier in Figures 4 and 6(a). Note however that the rate-of-change does not change. The set of communicating partners still changes dynamically at the same rate.



(a) without threshold filtering



(b) with threshold filtering (at 10%)

Figure 7. Dynamic analysis of the communication degree.

4. SYSTEM INTEGRATION

As described in Section 2, the proposed communication structure consists of two networks with different characteristics; an OCS which has a very high-bandwidth but requires a relatively large time to establish connections (circuits), and an EPS network that has a relatively low bandwidth. Due to the large delay for establishing connections in the OCS network, connections should be established only if they are used extensively for an extended period of time. This implies that communication over short-lived connections should be realized through the EPS network.

As described in Section 3, many applications exhibit a pattern of temporal locality which allows the overhead of establishing circuits in the OCS networks to be amortized over the life-time of the connection. However, other applications may exhibit poor temporal locality, and the system should be able to efficiently deal with such cases. In general, communications in high performance applications can be classified into three different classes:

1) Communications that exhibit good temporal locality and can be determined statically before execution.
2) Communications that exhibit good temporal locality but cannot be determined statically before execution.
3) Communications that exhibit poor temporal locality.

We will first describe a scheme in which the second class of communications can be dealt with. We then show, in Section 4.1, how to improve communication performance by discovering the communication patterns at compile time. The case of poor temporal locality is dealt with in Section 4.4.

As described earlier, the reason for having two networks is to use the OCS to establish connections between pairs of nodes that communicate heavily, while routing traffic between lightly communicating nodes through the EPS network. We assume that there is no a priori knowledge about the traffic volume between any two specific nodes, and will rely on a run-time system to determine which connections to establish in the OCS.

In the scheme, when a NIC receives a message from an attached node, it has to determine whether to send the message through the EPS or the OCS network. Initially all messages from a given source node, *s*, are routed through the EPS network. However, by monitoring the traffic out of *s*, it is possible to detect

when the traffic between *s* and a given destination, *d*, justifies the establishment of a dedicated circuit in the OCS. When such a decision is made, the NIC generates a request to the OCS to establish a connection between *s* and *d*. Different hardware and software traffic monitoring techniques can be used, and different policies can be devised to determine when a dedicated connection between two nodes is justified. We call such a policy a *migration policy*, and an example for a simple migration policy is given in Section 4.3. After the NIC is notified that the connection between *s* and *d* is established in the OCS, all subsequent traffic from *s* to *d* can be routed on the circuit established in the OCS.

Given that the OCS cannot accommodate all possible connections simultaneously, some provision has to be taken to tear-down an OCS connection when the migration policy decides to add a new connection to an OCS which is already fully populated by existing connections. A *replacement policy* is thus needed to determine which existing connection(s) to tear-down in order to make room for the new connection determined by the migration policy.

In addition to establishing long-lived connections in the OCS, a given logical topology may also be realized and used to route messages. For example, a logical ring may be embedded in an OCS by establishing a connection between every two adjacent nodes *n* and *n+1*. Of course, more efficient logical topologies, such as tori, hypercubes, and trees can be embedded in an OCS and used for routing messages between arbitrary sources and destinations. For multi-hop connections, each hop will traverse an optical connection in the OCS with the optical signal being converted to electronic and optical again at each NIC that also acts as a router.

Hence, the traffic monitor should be equipped with the capability of detecting when the communications in a running application do not exhibit temporal locality. Upon making this determination, the migration policy should embed a given predetermined logical topology in the OCS and use multi-hop routing in the OCS, in addition to using the EPS network for communication.

4.1 Finding Communication at Compile-time

The efficiency of the scheme described above can be improved if the communication requirements of an application are known at compile-time. Specifically, through compiler analysis, it may be possible to predict the connections that will exhibit heavy traffic at run-time, and to insert code in the executable that will be sent to the NIC, to establish these connections in the OCS. This *compiled communication* technique has been previously proposed for different networks [7,12,13,34]. Note that it is possible to combine compiled communication with run-time circuit setup migrations in the OCS system. Specifically, the instructions inserted in the code assist the migration and replacement policies with the time to establish and tear-down the statically determined connections. The run-time monitoring system will be left to make decisions concerning the connections that could not be determined statically in this way.

Different applications have different types of communication patterns. These communication patterns can be classified into three categories: *static*, *persistent* and *dynamic*.

Static — A communication pattern is static if it is completely known through compile-time analysis.

Persistent — A persistent communication exists if although it cannot be determined statically, it is set and does not change for a relatively long time.

Dynamic — A dynamic communication pattern cannot be determined until run-time when the communication operations actually occur.

The same terminology can be used to describe individual communications—*static*, *persistent* and *dynamic*. To represent these in the compiler, statically known communications are represented with constants, persistent connections are represented with symbolic expressions that will be resolved at run-time, and dynamic communications are represented with an "unknown" symbol. While many static and persistent communications may be selected for routing through the OCS network, some static and persistent communications will not use the OCS if their data sizes are not large enough or if they do not occur often enough to exceed a beneficial threshold determined by the technology. Those communications under such a threshold will be handled by the EPS network.

This classification scheme allows a determination of whether or not there are opportunities for (1) pre-establishing connections in the OCS prior to use for reducing or eliminating the relatively long connection establishment overhead, and (2) determining which connections are candidates for use in the OCS based on static information about the message sizes and frequency of communication.

Many previous efforts to analyze applications' communication characteristics are based solely on trace analysis. However, traces can provide the communication information for only a single execution instance of an application on a particular platform. To better formulate communication pattern

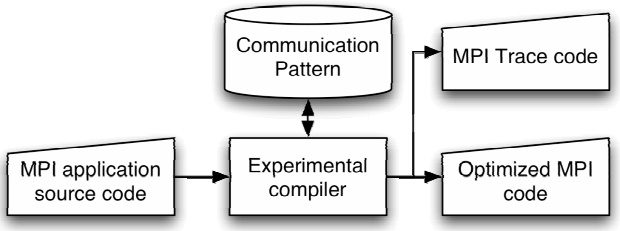


Figure 8. Compiler system paradigm.

needs, we may require a new communication pattern representation scheme. The flow shown in Figure 8 compiles C or Fortran + MPI applications and determines the communication patterns contained in the code. The compiler builds a communication graph based on a traditional control flow graph to represent the communications within the application. The communication graph is used to build a communication abstraction, which is the basis for the compile-time communication pattern. The compiler also has the capability to generate *smart traces*, or traces that correlate features of the code such as conditional statements and loops with communication operations. These traces can be used for further study of the applications or simply to verify the communication pattern discovered at compile-time.

Once the communication pattern is determined, the compiler inserts communication instructions into the application for the configuration of the OCS. Two types of communication instructions are designed in our prototype compiler:

1) Network configuration setup instructions
2) Network configuration flush instructions

Network configuration setup instructions are used to pre-establish network connections in order to reduce the setup overhead of the OCS. Network configuration flush instructions are used to flush the current network configuration effectively de-allocating the circuits that are no longer used. This is helpful when the application moves into a different phase and a run-time prediction scheme is employed to configure the OCS. This reduces the number of network conflicts that would otherwise be necessary without the compiler's information.

As described earlier, a compiler can be used to determine the different phases of execution and thus, different communication patterns. By simply adding a Flush command to the software execution, the Predictor can be reset to an unbiased state. For example, during the load balancing phase of SAGE, the communicate pattern can be completely changed. In these cases, it is best if the predictor is not biased by communications in the prior phase.

If the compiler can predict the communication pattern, then this static information can be used in place of prediction. However, the compiler may determine that some of the communications are dynamic and not statically deterministic. In such cases, it would be beneficial for the compiler to be able to bias the Predictor with the knowledge of the static communications and rely on the Predictor to fill in for the dynamic communications.

Additionally, the information from the Predictor can be used in subsequent executions of the same program. In essence, the Predictor could also be used as a profiler during an initial execution to assist a second execution of the same code. This is due to the fact that the information required to perform prediction is the same information that is required to perform profiling. In prediction, the destination, frequency and average length of messages, for example, are used to estimate the benefit of setting up a circuit in the OCS network. In profiling, this information is collected but is statistically analyzed. This information is extremely valuable to the Predictor. For example, if the maximum number of destinations is known and there are sufficient circuits to handle all possible destinations, then the circuits could be allocated according to the estimated bandwidth and adjusted at run time to fit the actual distribution. However, if there are fewer circuits than destinations and it has been shown from prior runs that all destinations are used equally, then it would be best not to change the allocation dynamically. The profile information may also show a mixture of these two examples depending on the execution phase. Irrespective of the actual behavior pattern, prediction/profile data from prior runs could be used to bias the Predictor for increased performance.

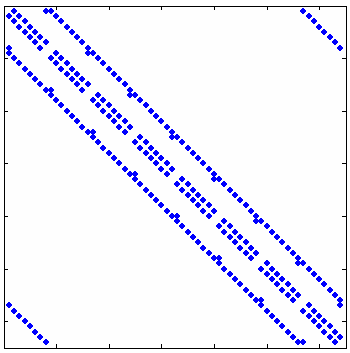
Figure 9 shows the communication pattern for the LBMHD application. This application consists of a single communication phase. In Figure 9(a) we show the communication partners for a processor of ID *myrank* as generated by our compiler. Each row contains a symbolic expression for one of the

communication partners. Each processor has a set of four other processors with which it communicates. Because the processor count, *P*, and the processor rank are not known at compile-time these symbols are resolved at run-time to generate the final communication matrix.

$$\begin{pmatrix} ((\lfloor myrank/P_x \rfloor + 1) \bmod P_x) P_x + myrank - \lfloor myrank/P_x \rfloor P_x \\ ((\lfloor myrank/P_x \rfloor - 1) \bmod P_x) P_x + myrank - \lfloor myrank/P_x \rfloor P_x \\ \lfloor myrank/P_x \rfloor P_x + myrank - (\lfloor (myrank+1)/P_x \rfloor P_x + 1) \bmod \lfloor myrank/P_x \rfloor \\ \lfloor myrank/P_x \rfloor P_x + myrank - (\lfloor (myrank+1)/P_x \rfloor P_x - 1) \bmod \lfloor myrank/P_x \rfloor \end{pmatrix}$$

where $P = P_x \, x \, P_y$ (or $P_x = P_y = \sqrt{P}$ for square processor counts)

(a) matrix generated by the compiler



(b) communication pattern from a 64 processor trace

Figure 9 – Communication pattern for LBMHD

However, in this example since *P* and *myrank* are the only symbols in the expression, this matrix is considered statically known as it may be entirely resolved at load-time prior to execution. Thus it is possible to entirely configure the OCS for LBMHD based on compiler analysis prior to execution. In Figure 9(b) we show the actual trace generated from a run on 64 processors. The colored pixels indicate one or more communications between source processors (vertical axis), and destination processors (horizontal axis). This trace confirms that each processor communicates with four other processors (seen by examining a single row or column in Figure 9b).

In addition to LBMHD, we used our experimental compiler to study the communications in the NAS parallel benchmark suit v2.4.1. This and other results concerning the OCS compiler work will be reported in a separate paper.

4.2 Network Adapter with Integrated Communication Predictor

Combining the OCS and the EPS networks requires an intelligent NIC. In Figure 10, we illustrate the various components of a candidate NIC to show that the various requirements of the OCS could also be efficiently handled by NIC hardware. It is possible to move some of these functions back into software in order to simplify the hardware but at the cost of some extra processing and time delay. However at this point we do not know exactly the benefit of having added functionality in the NIC verses doing it in software. In Figure 10, the host interface is represented as a set of directional queues between the host and the NIC. When point-to-point data is sent, its destination is used to determine the outgoing queue that the data is sent through. This simplifies and unifies the communication library as both networks utilize the same queues. It also enables run-time prediction of communication patterns and migration of communications from one network to the other.

When a Compute node sends a message, it is placed directly in an outgoing FIFO with the same index as the destination Compute node. These buffers do not have to be large nor do they imply significant latency. Our experiments with FPGAs show that a single FIFO adds only a single cycle of delay and operate at greater than 250MHz. Table 2 gives the performance of an *N*-destination Queue that is implemented using a single RAM and a register files for tracking *N head and tail* pointers and *N fully/empty* status bits. A 130nm FPGA was used and achieved over 3 Gbps. The size of *N* is expected to scale to well as we lost only 4MHz per doubling of *N*. A 65nm ASIC is expected to scale *N* to over 4096 at over 10Gbps.

Table 2. Performance of the NIC Queues for a 130nm FPGA (Altera EP1S25-5). ASIC performance is estimated to be 2.5 to 10x faster.

	Number Destinations per Queue			
	16	32	64	128
Area: Logic Cells (% device)	1439 (5.6%)	1988 (7.8%)	3939 (15.4%)	8010 (31.2%)
Memory in Bits (% device)	65,536 (3.4%)	131,072 (6.7%)	262,144 (13.5%)	524,288 (27.0%)
Max MHz	69	67	63	59
Throughput Gbps	4.4	4.3	4.0	3.8
Latency ns	14	15	16	17

The benefit of using the outgoing *N* destination Queue is that they provide details about the traffic pattern

without requiring additional software layering. By simply combining the "Empty" signals from each of the FIFOs into a single bit-vector, the real-time traffic requirements can be determined. In fact, the FIFOs can provide one or more "Almost Full" indicators that can be used to determine the amount of traffic that is pending. These FIFO status indictors are used by the Predictor to project the needs of the Compute node(s) to route communication through the OCS network.

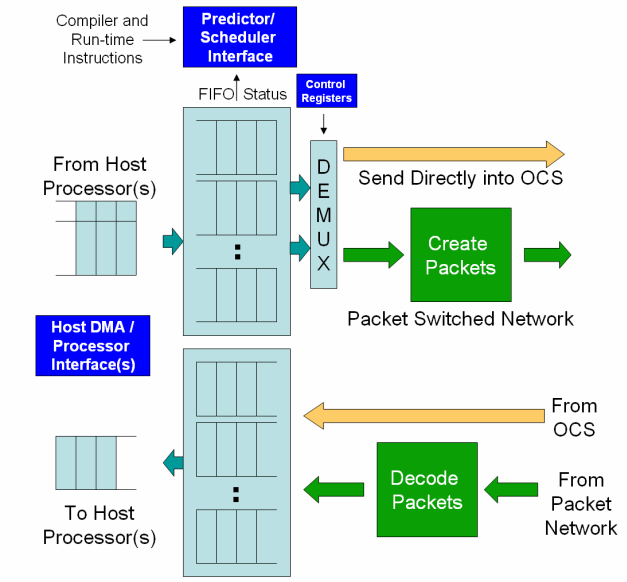


Figure 10. Example NIC with integrated prediction hardware for the dual OCS and EPS network.

The EPS network, in our candidate NIC, also uses the FIFO buffers but must first form the data into packets before it is sent into the EPS network. Even for a wormhole network, the data must be assigned a channel and the flits need to be created. For multiple communications between two compute nodes the EPS, if it is wormhole routed, must create the header of the worm. In each switch hop, the destination (or virtual channel) must be determined before the flit is routed. Thus, the EPS requires some level of buffering while the OCS does not. Similarly, the source of the incoming data from the OCS network is known a priori as a side effect of establishing the connection. The EPS data, however, can be from any other node and thus, the source needs to be sent along with the data. The flexibility of the EPS network is very useful but comes at a cost. By using the OCS for high-bandwidth traffic and by using the EPS for low-bandwidth and unpredictable traffic, the best of both mechanisms can be utilized.

The demultiplexer between the FIFOs and networks determines which network each FIFO will use. The only requirement for the multi-FIFO block is that it can read two values concurrently. The Control Registers are used to store the index of the FIFO(s) that utilize the OCS network and enable the EPS network to receive all other data. The simplicity of this design enable rapid migration of data between the OCS and EPS networks without placing a burden on the Compute node and without tightly synchronizing the NIC's activities with the Compute node.

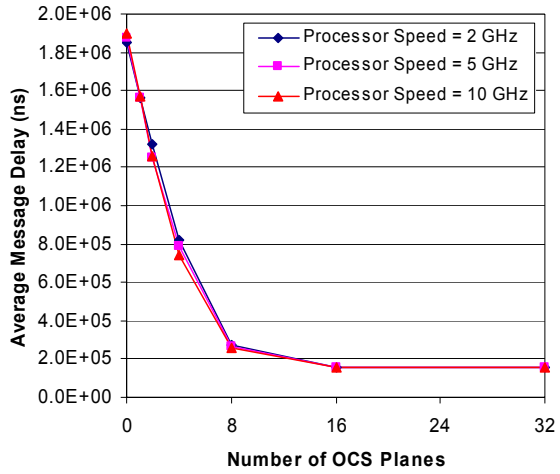
4.3 Performance Evaluation

A simplified simulation was used to examine the performance of communications in a n-processor system connected using the two networks. The EPS was taken to be an $n \times n$ crossbar switch, E, while the OCS network consisted of k optical crossbars, O_0 , …, O_{k-1}. Each node in the system reads communication events from an input trace file and simulates the events (for example *MPI_Send*, *MPI_receive*, *MPI_bcast*, …). Communication events in the trace file can be separated by a *compute(t)* event, which emulates a serial computation of duration t. A trace file can either be generated synthetically, or obtained from an application execution.

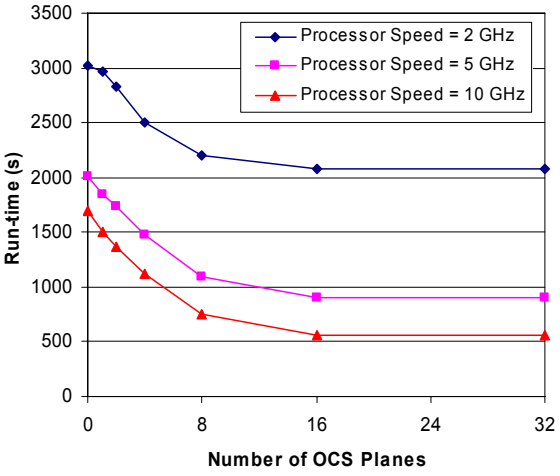
For this work, we use trace files generated from the execution of RF-CTH with AMR. Specifically, we link the application to a special MPI library to generate, during execution, communication events as well as the time between the successive executions of two MPI operations (It should be noted that trace generation is intrusive and the execution time will be slightly increased due to the additional time keeping and the trace generation instructions). The simulation allows the specification of many system parameters including those listed in Table 3. The values used in the simulation are also listed in Table 3.

Table 3. Simulation system parameters.

Parameter	Value
Node count	128
Number of OCS planes	0 - 32
Link speed in the OCS	8Gb/s
Link speed in the electronic network	1-10Gb/s
Electronic/optical switch arbitration time	100ns
Connection establishment delay in the OCS	3ms
Connection delay in the electronic network	100ns
Electronic and optical link propagation delay	200ns
Processor operating frequency	1 – 10GHz
Overhead of an MPI_Send or MPI_Receive	5000 cycles



(a) Average message delay



(b) RF-CTH run-time

Figure 11. Performance using multiple OCS planes.

The migration and replacement policies are specified by writing corresponding modules in the simulator. The results reported here are for the following two policies:

Migration policy: Request a connection from *s* to *d* in the OCS when a message larger than 32 bytes is sent from *s* to *d*.

Replacement policy: Least Recently Used – when a request to establish an OCS circuit from *s* to *d* occurs and all optical ports out of *s* or all optical ports into *d* are occupied by other connections, then at least one and at most two OCS connections should be torn-down to allow for the establishment of the new connection. The connections that have been least recently used in the OCS are torn-down to make room for the new connection.

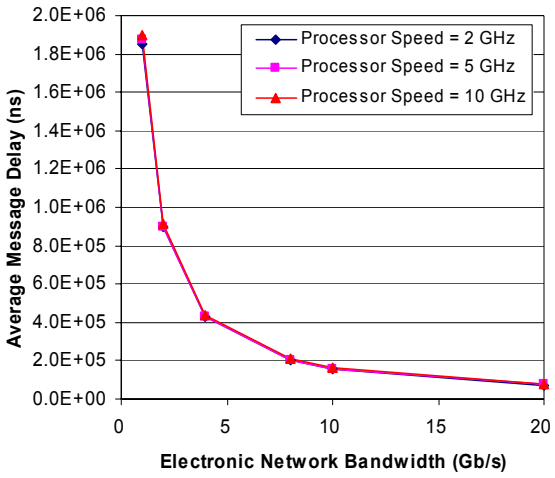
Other potential circuit allocation policies (e.g., least actively used, or setting up circuits in a run on the basis of communication traffic in a prior run, among others) have not yet been evaluated relative to these policies. Effectiveness of any particular policy may depend strongly on specific application traffic patterns and specific system parameters.

Both the application run-time and the average message delay are reported. In Figure 11, we show the effect of the number of OCS planes on performance. The performance improves when the number of planes, *k*, increases up to 16. The improvement beyond $k=8$ is very small because, as was shown in Section 3, the number of heavily used connections in the CTH program rarely exceeds 10.

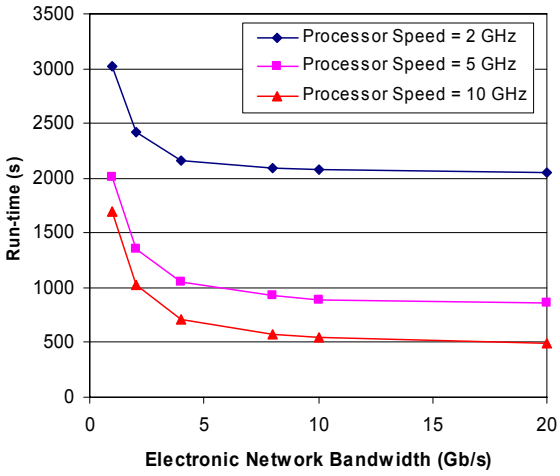
In Figure 12, the effect of using only a fast electronic network instead of augmenting a slow electronic network with an OCS is shown. In order to match the performance of the dual EPS + OCS, the bandwidth of the EPS network has to be increased. For example, if 8 OCS planes are assumed the average message delay is 0.25ms irrespective of the processor speed (Figure 11a), and the bandwidth of the electronic network required to match this performance is approximately 7Gb/s (Figure 12a).

Further simulation results using UMT2K are shown in Figure 13 where two classes of systems are compared. The first type uses fast EPS with no OCS. In this configuration six fast EPS fat-trees are used. The second type uses a slow EPS augmented with a number K_o of optical crossbars, with K_o being varied between zero and 24. As expected, the performance improves when the number of optical planes increases up to 12 planes. Using 24 OCS planes does not improve the performance much because 12 planes are enough to support the communication patterns of the applications. The results also show that using 6 fast EPS planes is equivalent to using one slow EPS plane with 12 OCS planes. This simulation was done with different parameters than those in the previous example.

Work is in progress on simulating the performance of other applications.

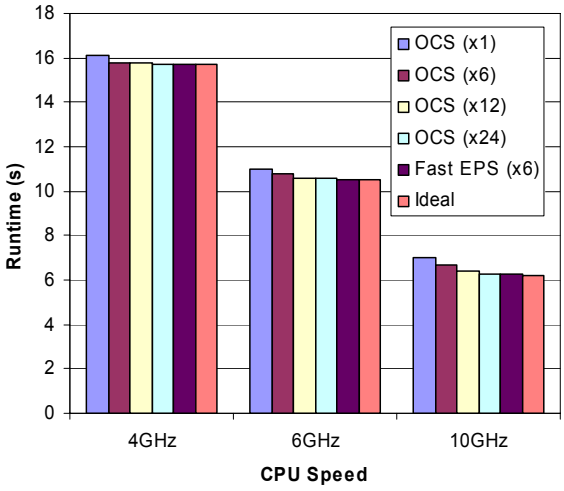


(a) Average message delay

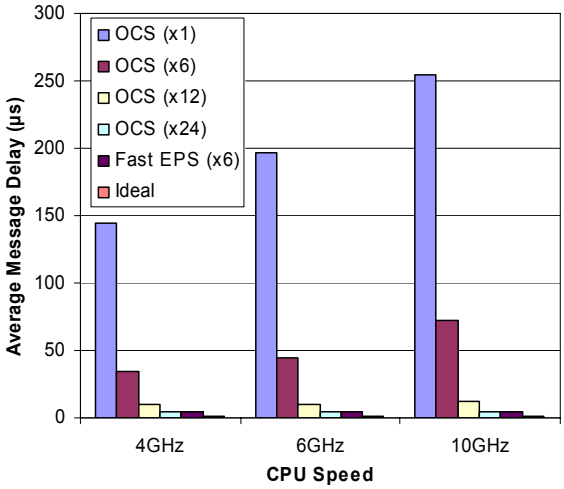


(b) RF-CTH run-time

Figure 12. Performance using only a fast electronic network.



(a) run-time (512 nodes)



(b) Average message delay (512 nodes)

Figure 13. Simulation results for UMT2K

4.4 Handling poor temporal locality

The traffic monitor can be equipped with the capability of detecting when the communication in a running application does not exhibit temporal locality and leads to repeated allocation and de-allocation of connections in the OCS (thrashing). Upon making this determination, the migration policy can embed a given predetermined logical topology in the OCS and then use multi-hop routing as necessary for the communications. Possible logical topologies embeddings onto OCS for multi-hop routing depend on the number of OCS planes available. This number is the maximum node degree of the logical topology, which determines the maximum number of hops. For example, an *n*-node system with 6 OCS planes allows the embedding of a 3-D torus topology resulting in a maximum of $1.5\sqrt[3]{n}$ hops. Or a system having *log n* OCS planes allows the embedding of a hypercube topology and results in a maximum of *log n* hops. Finally, it can be shown that the number of hops is limited to 2 if $\sqrt{n}$ OCS planes are available, or generally k for $\sqrt[k]{n}$ planes. The proof and other details of this graph expander embedding scheme, is currently being prepared as a separate paper. Since we expect to have many OCS planes, relative to the total number of nodes, we will, in most cases, use only 2 hops to satisfy any communication

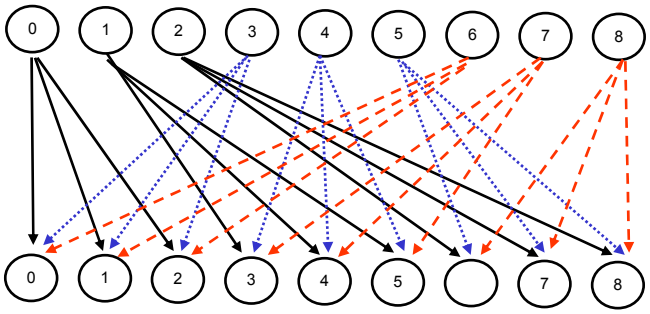


Figure 14. Established circuits for the case of 9 nodes.

We demonstrate the principle of 2-hop connections using an example of a small system containing 9-nodes. Figure 14 shows the logical connectivity between the output NICs and the input NICs of the nodes that lead to the 2-hop communication. The node degree of this graph is 3 and thus three OCS planes are needed to embed this logical topology. It can be easily shown that a communication can take place between any output NIC and any input NIC using at most two hops. For example, although node 1 is not directly connected to node 6, it is connected to node 5, which is connected to node 6. Hence communication between node 1 and 6 will take 2 hops, passing through node 5.

5. CONCLUSIONS

A substantial part of the cost of a HPC system is due to the interconnect. The bulk of interconnects today use electronic packet switching elements and optical transceivers that dominate this cost. We propose using a combination of two networks to provide high performance at manageable cost.

Our approach is motivated by an empirical analysis of HPC applications, eleven of which we describe in this paper. These applications are drawn from a number of sources and solve a variety of problems. Our analysis shows that the bulk of the inter-processor communication (barring collectives) in these applications is bounded in degree and changes either very slowly or never. Stated differently, our analysis indicates that these applications severely underutilize the switching capability in an electronic packet-switched interconnect. While the interconnect provides the capability for every processor to communicate with every other processor on a per-packet basis, the applications tend to overwhelmingly favor a small number of routes. Based on this observation, we propose the use of two separate communication mechanisms in HPC systems: a cheap and power-efficient mechanism for long-lived transfers of large amounts of data, and another to accommodate collectives and short-lived data exchanges. Separating the communication classes in this manner enables us to target each class with the most appropriate network structure and technology.

For long-lived bulk data transfers, we use an OCS network. OCS switches use optics at all elements of the data path with switching accomplished through the MEMS-based mirrors. For comparative bandwidth, an OCS is substantially less expensive but switches significantly slower than an all-electronic packet switching network. However, since the circuits established in the OCS will be changed quite infrequently, the slower switching speed is not a performance impediment. We route collectives and transient communication over a secondary lower-bandwidth EPS network. With its much higher switching speed, the secondary interconnect is able to handle this kind of traffic with low latency. At the same time, we are able to keep the secondary interconnect from being bandwidth overwhelmed by using it for a portion of the traffic.

We have developed a combination of static (compile-time) techniques and dynamic run-time policies to enable practical use of the combination of the two networks. Our experimental compiler is able to determine long-lived bulk transfers in a number of applications. Where this works, the compiler inserts instructions to setup the OCS at application startup, enabling the use of the OCS without any switching latency during application execution. In cases where the static analysis is not enough, we use a run-time policy that dynamically moves traffic between the OCS and packet-switched networks. With a simulator we have evaluated our design and shown that the combined two networks maintain performance while being substantially cheaper than an all-electronic packet switched network.

There are still several open questions which we will attempt to answer in future work. Issues of cost and cost scalability are obviously important – as we have stated, we have not attempted to do a serious analysis of cost comparison in this paper, since the available data on component cost trends are still very rough. The detailed NIC design is a complex enough subject to warrant a full paper on it's own, but the basic question – "is such an approach feasible?" may be answered by pointing the reader to much more

complex chips designed and operating currently. More analysis of policies for circuit allocation may prove useful, as we only show results for one of many possible implementations in Section 4.3. Finally, the mechanisms for circuit set-up and/or tear down will require another level of detailed design, depending on the exact implementation of the OCS switch.

ACKNOWLEDGEMENTS

This paper is based upon work done in the context of the PERCS project at IBM, which is supported in part by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCH3039004. Los Alamos National Laboratory is operated by the University of California for the U.S. Department of Energy under contract W-7405-ENG-36. We are grateful to Mootaz Elnozahy for his leadership, support and encouragement, as well as his many good suggestions throughout this work.

REFERENCES

[1] ADIGA N.R. ET. AL. 2002. “An Overview of the BlueGene/L Supercomputer”. In Proc. IEEE/ACM SC02. Baltimore.

[2] ARIDOR, Y., DOMANY, T., GOLDSHMIDT, O, SHMUELI, E., MOREIRA, J., AND STOCKMEYER, L. 2004. “Multi-Toroidal Interconnects: Using Additional Communication Links To Improve Utilization of Parallel Computers”. In Proc. 10th Workshop on Job Scheduling Strategies for Parallel Processing, New York.

[3] ARIDOR, Y., DOMANY, T., GOLDSHMIDT, O., MOREIRA, J. AND SHMUELI, E. 2005. “Resource Allocation and Utilization in the BlueGene/L Supercomputer”, IBM Journal of Research and Development, 49(2/3), pp. 425-436.

[4] BELL, R., MALONY, A., AND S. SHENDE. 2003. "A Portable, Extensible, and Scalable Tool for Parallel Performance Profile Analysis". In Proc. Europar, LNCS 2790, Springer, Berlin, pp. 17-26.

[5] BLITZ, C.M., AND LIPSCOMB, W.H. 1999. “An Energy-conservative Thermodynamic Model of Sea Ice”. J. Geophysics Research, 104(15), pp. 699-677. http://climate.lanl.gov/

[6] CAIN, H.W., MILLER, B.P., WYLIE, B.J.N. 2002. “A Callgraph-based Search Strategy for Automated Performance Diagnosis”. Concurrency and Computation: Practice and Experience, 14, pp. 203-217.

[7] CAPPELLLO, F., AND GERMAIN, C. 1995. “Toward High Communication Performance Through Compiled Communications on a Circuit Switched Interconnection Network”. In Proc. Int. Symp. On High Performance Computer Architecture, pp. 44-53.

[8] CHAMBERLAIN, R., FRANKLIN, M., AND BAW, C. S. . 2002. “Gemini: An optical interconnection network for parallel processing”. In IEEE Transactions on Parallel and Distributed Processing, 13(10), pp. 1038-1055.

[9] COLLINS, W. D., RASCH, P.J., BOVILLE, B.A., HACK, J.J., MCCAA, J.R., WILLIAMSON, D.L., KIEHL, J.T., BRIEGLEB, B., BITZ, C., LIN, S.J., ZHANG, M. AND DAI, Y. 2004. “Description of the NCAR Community Atmosphere Model (CAM 3.0)”. NCAR Tech. Note NCAR/TN-464+STR

[10] CRAWFORD, D.A., TAYLOR, P.A., AND HERTEL, E.S. 2001. “Adaptive Mesh Refinement in the CTH Shock Physics Hydrocode”. In Proc. New Models and Hydrocodes for Shock Wave Processes in Condensed Matter, Paris, France.

[11] DEFANTI, T., BROWN, M., LEIGH, J., YU, O., HE, E., MAMBRETTI, J., LILLETHUN, D., AND WEINBERGER, J. 2003. “Optical switching middleware for the OptIPuter”. IEICE Transact. Commun. E86-B, 8, pp. 2263-2272.

[12] DIETZ, H.G., AND MATTOX, T.I. 2000. "Compiler Techniques For Flat Neighborhood Networks". In Proc. 13th Int. Workshop on Languages and Compilers for Parallel Computing (LCPC00), Yorktown Heights, New York.

[13] DING, Z., HOARE, R., JONES, A., LI, D., SHAO, S., TUNG, S., ZHENG, J., AND MELHEM, R. 2005. “Switch Design to Enable Predictive Multiplexed Switching in Multiprocessor Networks”. In Proc. 19th IEEE Int. Parallel & Distributed Processing Symp., Denver, CO..

[14] EBERLE, H. AND NILSM GURA N., 2002. "Separated High-bandwidth and Low-latency Communication in the Cluster Interconnect Clint”. In Proceedings of the IEEE/ACM Supercomputing Conference, Baltimore.

[15] GRIPP, J., DUELK, M., SIMSARIAN, J., BHARDWAJ, A., BERNASCONI, P., LAZNICKA, O., ZIRNGIBL, M., AND STILIADIS, D., 2003. "Optical switch fabrics for terabit-class routers and packet switches". J. Optical Networking 2(7), pp. 243-254.

[16] GUPTA, V. AND SCHENFELD, E. 1994. “Combining Message Switching with Circuit Switching in the Interconnection Cached Multiprocessor Network”. In Proc. IEEE Int. Symposium on Parallel Architectures, Algorithms and Networks - ISPAN HORIGUCHI, S.(ed.), pp.143-150.

[17] GUPTA V. AND SCHENFELD, E. 1995. “Task Graph Partitioning and Mapping in a Reconfigurable

Parallel Architecture". Parallel Processing Letters 5(4), pp. 563-574.

[18] HABATA, S., UMEZAWA, K., YOKOKAWA, M., AND KITAWAKI, S. 2004. "Hardware system of the Earth Simulator". Parallel Computing, 30(12), pp. 1287-1313.

[19] HALLIWELL, G.R. 2004. "Evaluation of Vertical Coordinate and Vertical Mixing Algorithms in the Hybrid Coordinate Ocean Model (HYCOM)", Ocean Modeling, Vol. 7, pp. 285-322.

[20] HEMENWAY, R., GRZYBOWSKI, R., MINKENBERG, C., AND LUIJTEN, R., 2004. "Optical-packet-switched interconnect for supercomputer applications". J. Optical Networking 3(12), pp. 900-913.

[21] HOISIE, A., LUBECK, O., AND WASSERMAN, H.J. 2000 "Performance and Scalability Analysis of Teraflop-Scale Parallel Architectures using Multidimensional Wavefront Applications". Int. J. of High Performance Computing Applications, 14(4), pp. 330-346.

[22] JONES, P.W. 1997. "The Los Alamos Parallel Ocean Program (POP) and Coupled Model on MPP and Clustered SMP Computers". In Making its Mark – The Use of Parallel Processors in Meteorology, HOFFMAN, G.R. (Ed), World Scientific Publishing. http://climate.lanl.gov/

[23] KERBYSON, D.J., ALME, H.J., HOISIE, A., PETRINI, F., WASSERMAN, H.J., AND GITTINGS, M.L. 2001. "Predictive Performance and Scalability Modeling of a Large-scale Application". In Proc. IEEE/ACM Supercomputing, Denver, CO.

[24] KERBYSON, D.J., AND, BARKER, K.J. 2005. "Automatic Identification of Communication Patterns via Templates". In Proc. ISCA Int. Conf. on Parallel and Distributed Computing Systems.

[25] KERBYSON, D.J., AND JONES, P.W. 2005 "A Performance Model of the Parallel Ocean Program". Int. J. of High Performance Computing Applications, 19(13).

[26] LEISERSON, C.E., et al. 1992. "The network architecture of the Connection Machine CM-5", In Proc. ACM Symp. Parallel Algorithms and Architectures, pp. 272–285.

[27] MATHIS, M., AND KERBYSON, D.J. 2004. "Performance Modeling of Unstructured Mesh Particle Transport Computations", In Proc. PMEO-IPDPS, Santa Fe, NM.

[28] OLIKER, L., CANNING, A., CARTER, J., SHALF, J., SIMON, H. ETHIER, S., PARKS, D., KITAWAKI, S., TSUDA, Y., AND SATO, T. 2005. "Performance of UltraScale Applications on Leading Vector and Scalar HPC platforms". to appear in the Journal of Earth Simulator.

[29] PARDO, F., ET. AL. 2003. "Optical MEMS devices for telecom systems", Proc. of SPIE , v 5116 II, pp. 435-444.

[30] SCOTT, S.L. 1996. "Synchronization and communication in the T3E multiprocessor", In Proc. ASPLOS-VII, Sept. 1996.

[31] SIVARAM, R., STUNKEL, C.B., AND D. K. PANDA. 1997. "A reliable hardware barrier synchronization scheme". in Proc. of the 11th Int. Parallel Processing Symp., Geneva, Switzerland, pp. 274-280.

[32] SPOONER, D.P., AND KERBYSON, D.J. 2005. "Performance Feature Identification by Comparative Trace Analysis", to appear in Future Generation Computer Systems, Elsvier.

[33] VEERARAGHAVAN, M., ZHENGA, X., LEEB, H., GARDNERC, M., AND FENGC, M. 2003. "CHEETAH: circuit-switched high-speed end-to-end transport architecture". Proceedings of the SPIE, Volume 5285, pp. 214-225.

[34] YUAN, X., MELHEM, R., AND R. GUPTA. 1996. "Compiled communication for All-optical TDM Networks". In Proc. IEEE/ACM Supercomputing, Pittsburgh, PA.