Analyzing Ultra-Scale Application Communication Requirements for a Reconfigurable Hybrid Interconnect

John Shalf, Shoaib Kamil, Leonid Oliker, David Skinner
CRD/NERSC, Lawrence Berkeley National Laboratory, Berkeley, CA 94720

ABSTRACT

The path towards realizing peta-scale computing is increasingly dependent on scaling up to unprecedented numbers of processors. To prevent the interconnect architecture between processors from dominating the overall cost of such systems, there is a critical need for interconnect solutions that both provide performance to ulta-scale applications and have costs that scale linearly with system size. In this work we propose the Hybrid Flexibly Assignable Switch Topology (HFAST) infrastructure. The HFAST approach uses both passive (circuit switch) and active (packet switch) commodity switch components to deliver all of the flexibility and fault-tolerance of a fully-interconnected network (such as a fat-tree), while preserving the nearly linear cost scaling associated with traditional low-degree interconnect networks. To understand the applicability of this technology, we perform an in-depth study of communication requirements across a broad spectrum of important scientific applications, whose computational methods include: finite-difference, lattice-bolzmann, particle in cell, sparse linear algebra, particle mesh ewald, and FFT-based solvers. We use the IPM (Integrated Performance Monitoring) profiling layer to gather detailed messaging statistics with minimal impact to code performance. This profiling provides us sufficiently detailed communication topology and message volume data to evaluate these applications in the context of the proposed hybrid interconnect. Overall results show that HFAST is a promising approach for practically addressing the interconnect requirements of future peta-scale systems.

1. INTRODUCTION

As the field of scientific computing matures, the demands for computational resources are growing at a rapid rate. It is estimated that by the end of this decade, numerous mission-critical applications will have computational requirements that are at least two orders of magnitude larger than current levels [1, 2, 17]. However, as the pace of processor clock rate improvements continues to slow, the path towards realizing peta-scale computing is increasingly dependent on scaling up the number of processors to unprecedented levels. To prevent the interconnect architecture from dominating the overall cost of such systems, there is a critical need to effectively build and utilize network topology solutions with costs that scale linearly with system size.

HPC systems implementing *fully-connected networks* (FCNs) such as fat-trees and crossbars have proven popular due to their excellent bisection bandwidth and ease of application mapping for arbitrary communication topologies. In fact, as of November 2004, 94 of the 100 systems in the Top500 [4] employ FCNs (92 of which are fat-trees). However, it is becoming increasingly difficult and expensive to maintain these types of interconnects, since the cost of an FCN infrastructure composed of packet switches grows superlinearly with the number of nodes in the system. Thus, as supercomputing systems with tens or even hundreds of thousands of processors begin to emerge, FCNs will quickly become infeasibly expensive. This has caused a renewed interest in networks with a lower topological degree, such as mesh and torus interconnects (like those used in IBM BlueGene/L, Cray RedStorm, and Cray X1), whose costs rise linearly with system scale. However, only a subset of scientific computations have communications patterns that can be effectively embedded onto these types of networks.

One of the principal arguments for moving to lower-degree networks is that many of the phenomena modeled by scientific applications involve localized physical interactions. However, this is a dangerous assumption because not all physical processes have a bounded locality of effect (e.g. n-body gravitation problems), and not all numerical methods used to solve scientific problems exhibit a locality of data dependencies that is a direct reflection of the phenomena they model (e.g. spectral methods and adaptive mesh computations). As a result, lower-degree interconnects are not suitable for all flavors of scientific algorithms. Before moving to a radically different interconnect solution, it is essential to understand scientific application communication requirements across a broad spectrum of numerical methods.

Several studies have observed that many applications have communication topology requirements that are far less than the total connectivity provided by FCN networks. For instance, the most demanding applications investigated by Vetter and Mueller [18, 19] indicate that the applications that scale most efficiently to large numbers of processors tend to depend on point-to-point communication patterns where each processor's average *topological degree of communication* (TDC) is 3–7 distinct destinations, or neighbors.

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

This provides strong evidence that many application communication topologies exercise a small fraction of the resources provided by FCN networks.

However, even if the network offers a topological degree that is greater than or equal to the application's TDC, traditional low-degree interconnect approaches have several significant limitations. First, there is no guarantee the interconnect and application communication topologies are isomorphic—hence preventing the communication graph from being properly embedded into the fixed interconnect topology. Additionally, adoption of networks with a lower degree of topological connectivity leads to considerable problems with runtime job scheduling. Unless the communication topology is known before application processes are assigned to nodes, the mapping of the application process topology to the fixed network topology may result in hopelessly inefficient performance. This kind of topological mismatch can be mitigated by sophisticated task migration and job-packing by the batch system, but such migration impacts overall system efficiency. Furthermore, task migration is a complex software technology, especially for distributed architectures, and is often absent on modern parallel computers. Finally, individual link or node failures in a lower-degree interconnection network are far more disruptive than they are to a fully-interconnected topology. Any failure of a node within a mesh will create a gap in the interconnect topology that further complicates job scheduling, and, in some cases, message routing. Even if the interconnect can route around failure, overall application performance then becomes dependent on the slowest link in the interconnect. In contrast, when a node fails in an FCN, it can be taken offline without compromising the messaging requirements for the remaining nodes in the system.

To address these deficiencies, we propose the Hybrid Flexibly Assignable Switch Topology (HFAST) infrastructure. The HFAST approach uses both passive (layer-1 / circuit switch) and active (layer-2 / packet switch) commodity components to deliver all of the flexibility and fault-tolerance of a fat-tree interconnect, while preserving the nearly linear cost scaling associated with traditional low-degree interconnect networks. In order to understand the applicability of this technology, we perform an in-depth study of the communication requirements across a broad spectrum of important scientific applications, whose computational methods include: finite-difference, lattice-bolzmann, particle in cell, sparse linear algebra, particle mesh ewald, and FFT-based solvers. To efficiently collect this data, we use the IPM profiling layer, which gathers detailed messaging statistics with minimal impact on code performance. The derived messaging statistics enable us to compute the topological requirements of these applications, in the context of our hybrid interconnect network. Overall results show that HFAST is a promising approach for practically addressing the interconnect requirements of future peta-scale systems.

2. HYBRID SWITCH ARCHITECTURE

Given the superlinear cost of constructing FCN architectures, the interconnect will rapidly become the dominant cost of such systems. As we move towards petaflops systems with tens (or hundreds) of thousands of processors, the industry will be hard-pressed to continue to build fat-tree networks into the peta-scale era. For an alternative to fat-trees and traditional packet-switched interconnect architectures, we can look to recent trends in the high-speed wide area networking community, which has arrived at a cost-effective hybrid solution to similar problems.

2.1 Circuit Switch Technology

Packet switches, such as Ethernet, Infiniband, and Myrinet, are the most commonly used interconnect technology for large-scale parallel computing platforms. A packet switch must read the header of each incoming packet in order to determine on which port to send the outgoing message. As bit rates increase, it becomes increasingly difficult and expensive to make switching decisions at line rate. Most modern switches depend on ASICs or some other form of semi-custom logic to keep up with cutting-edge data rates. Fiber optic links have become increasingly popular for cluster interconnects because they can achieve higher data rates and lower bit-error-rates over long cables than is possible using low-voltage differential signaling over copper wire. However, optical links require a transceiver that converts from the optical signal to electrical so the silicon circuits can perform their switching decisions. The Optical Electrical Optical (OEO) conversions further add to the cost and power consumption of switches. Fully-optical switches that do not require an OEO conversion can eliminate the costly transceivers, but per-port costs will likely be higher than an OEO switch due to the need to use exotic optical materials in the implementation.

Circuit switches, in contrast, create hard-circuits between endpoints in response to an external control plane – just like an old telephone system operator's patch panel, obviating the need to make switching decisions at line speed. As such, they have considerably lower complexity and consequently lower cost per port. For optical interconnects, micro-electromechanical mirror (MEMS) based optical circuit switches offer considerable power and cost savings as they do not require expensive (and power-hungry) optical/electrical transceivers required by the active packet switches. Also, because non-regenerative circuit switches create hard-circuits instead of dynamically routed virtual circuits, they contribute almost no latency to the switching path aside from propagation delay. MEMS based optical switches, such as those produced by Lucent, Calient and Glimmerglass, are common in the telecommunications industry and the prices are dropping rapidly as the market for the technology grows larger and more competitive.

2.2 Related Work

Circuit switches have long been recognized as a cost-effective alternative to packet switches, but it has proven difficult to exploit the technology for use in cluster interconnects because the switches do not understand message or packet boundaries. It takes on the order of milliseconds to reconfigure an optical path through the switch, and one must be certain that no message traffic is propagating through the light path when the reconfiguration occurs. In comparison, a packet-switched network can trivially multiplex and demultiplex messages destined for multiple hosts without requiring any configuration changes.

The most straightforward approach is to completely eliminate the packet switch and rely entirely on a circuit switch. A number of projects, including the OptIPuter [8] transcontinental optically-interconnected cluster, use this approach for at least one of their switch planes. The OptIPuter nodes

use Glimmerglass MEMS-based optical circuit switches to interconnect components of the local cluster, as well as to form transcontinental light paths which connect the University of Illinois half of the cluster to the UC San Diego half. One problem that arises with this approach is how to multiplex messages that arrive simultaneously from different sources. Given that the circuit switch does not respect packet boundaries and that switch reconfiguration latencies are on the order of milliseconds, either the message traffic must be carefully coordinated with the switch state or multiple communication cards must be employed per node so that the node's backplane effectively becomes the message multiplexor; the OptIPuter cluster uses a combination of these two techniques. The single-adapter approach leads to impractical message-coordination requirements in order avoid switch reconfiguration latency penalties, whereas the multiadapter approach suffers from increased component costs due to the increased number of network adapters per host and the larger number of ports required in the circuit switch.

One proposed solution, the ICN (Interconnection Cached Network) [10], recognizes the essential role that packet switches play in multiplexing messages from multiple sources at line rate. The ICN consists of processing elements that are organized into blocks of size k which are interconnected with small crossbars capable of switching individual messages at line rate (much like a packet switch). These k-blocks are then organized into a larger system via a $k * N_{blocks}$ ported circuit switch. The ICN can embed communication graphs that have a consistently bounded topological degree of communication (TDC) less than k. The jobs must be scheduled in such a way that the bounded contraction of the communication topology (that is, the topological degree of every subset of vertices) is less than k. This is an NP-complete problem for general graphs when $k > 2$, although such contractions can be found algorithmically for regular topologies like meshes, hypercubes, and trees. If the communication topology has nodes with degree greater than k, some of the messages will need to take more than one path over the circuit switch and therefore share a path with other message traffic. Consequently the bandwidth along that path is reduced if more than one message must contend for the same link on the network. Job placement also plays a role in finding an optimal graph embedding. Runtime reconfiguration of the communication topology on an ICN may require task migration in order to maintain an optimal embedding for the communication graph. The HFAST approach detailed in this work has no such restriction to regular topologies and needs no task migration.

Finally, there are a number of hybrid approaches that use combination packet/circuit switch blocks. Here each switching unit consists of a low bandwidth dynamically-routed network that is used to carry smaller messages and coordinate the switch states for a high-bandwidth circuit switched network that follows the same physical path. Some examples include Gemini [7], and Sun Microsystems Clint [9]. Each of these uses the low-bandwidth packet-switched network to set up a path for large-payload bulk traffic through the circuit switch hierarchy. While the circuit switch path is unaware of the packet boundaries, the lower-speed packet network is fast enough to mediate potential conflicts along the circuit path. This overcomes the problems with coordinating message traffic for switch reconfiguration exhibited by the purely circuit-switched approach. While promising, this architecture suffers from the need to use custom-designed switch components for a very special-purpose use. In the short term, such a specialized switch architecture will have difficulty reaching a production volume that can amortize the initial development and manufacturing costs. Our target is to make use of readily available commodity components in the design of our interconnect in order to keep costs under control.

2.3 HFAST: Hybrid Flexibly Assignable Switch Topology

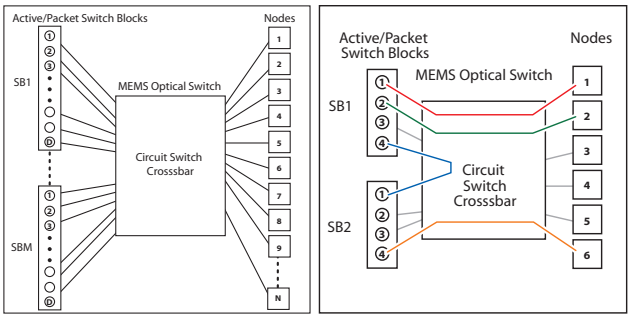

Figure 1: General layout of HFAST (left) and example configuration for 6 nodes and active switch blocks of size 4 (right).

We propose HFAST as a solution for overcoming the obstacles we outlined above, by using (Layer-1) passive/circuit switches to dynamically provision (Layer-2) active/packet switch blocks at runtime. This arrangement leverages the less expensive circuit switches to connect processing elements together into optimal communication topologies using far fewer packet switches than would be required for an equivalent fat-tree network composed of packet switches. For instance, packet switch blocks can be arranged in a single-level hierarchy when provisioned by the circuit switches to implement a simpler topology like a 3D torus, whereas a fat-tree implementation would require traversal of many layers of packet switches for larger systems – contributing latency at each layer of the switching hierarchy. Therefore this hybrid interconnection fabric can reduce fabric latency by reducing the number of packet switch blocks that must be traversed by a worse-case message route.

Using less-expensive circuit switches, one can emulate many different interconnect topologies that would otherwise require fat-tree networks. The topology can be incrementally adjusted to match the communication topology requirements of a code at runtime. Initially, the circuit switches can be used to provision densely-packed 3D mesh communication topologies for processes. However, as data about messaging patterns is accumulated, the topology can be adjusted at discrete synchronization points to better match the measured communication requirements and thereby dynamically optimize code performance. MPI topology directives can be used to speed the runtime topology optimization process. There is also considerable research opportunities available for studying compile-time instrumentation of codes to infer communication topology requirements at compile-time. In particular, languages like UPC offer a high-level approach for exposing communication requirements at compile-time. Similarly, the compiler can automatically insert the neces-

System	Technology	MPI Latency	Peak Bandwidth	Bandwidth Delay Product
SGI Altix	Numalink-4	1.1us	1.9 GB/s	2 KB
Cray X1	Cray Custom	7.3us	6.3 GB/s	46 KB
NEC Earth Simulator	NEC Custom	5.6us	1.5GB/s	8.4 KB
Myrinet Cluster	Myrinet 2000	5.7us	500MB/s	2.8 KB
Cray XD1	RapidArray/IB4x	1.7us	2GB/s	3.4 KB

Table 1: Bandwidth delay products for several high performance interconnect technologies. This is the effective peak unidirectional bandwidth delivered per CPU (not per link).

sary synchronization points that allow the circuit switches time to reconfigure since the Layer-1 switches do not otherwise respect packet boundaries for in-flight messages.

HFAST differs from the bounded-degree ICN approach in that the fully-connected passive circuit switch is placed between the nodes and the active (packet) switches. This supports a more flexible formation of communication topologies without any job placement requirements. Codes that exhibit non-uniform degree of communication (e.g. just one or few process(es) must communicate with a large number of neighbors) can be supported by assigning additional packet switching resources to the processes with greater communication demands. Unlike the ICN and OptIPuter, HFAST is able to treat the packet switches as a flexibly assignable pool of resources. In a sense, our approach is precisely the inverse of the ICN – the processors are connected to the packet switch via the circuit switch, whereas the ICN uses processors that are connected to the circuit switch via an intervening packet switch.

Figure 1 shows the general HFAST interconnection between the nodes, circuit switch and active switch blocks. The diagram on the right shows an example with 6 nodes and active switch blocks of size 4. In this example, node 1 can communicate with node 2 by sending a message through the circuit switch (red) in switch block 1 (SB1), and back again through the circuit switch (green) to node 2. This shows that the minimum message overhead will require crossing the circuit switch two times. If the TDC of node 1 is greater than the available degree of the active SB, multiple SBs can be connected together (via a myriad of interconnection options). For the example in Figure 1, if node 1 was to communicate with node 6, the message would first arrive at SB1 (red), then be transferred to SB2 (blue), and finally sent to node6 (orange) — thus requiring 3 traversals of the circuit switch crossbar and two active SB hops.

2.4 Small Messages and the Bandwidth Delay Product

The product of the bandwidth and the delay for a given point-to-point connection describes precisely how many bytes must be "in-flight" to fully utilize available link bandwidth. This can also be thought of as the minimum size required for a non-pipelined message to fully utilize available link bandwidth. Vendors commonly refer to an $N_{1/2}$ metric, which describes the message size below which you will get only 1/2 of the peak link performance. The $N_{1/2}$ metric is typically half the bandwidth-delay product. In this paper, however, we choose to focus on messages that are larger than the bandwidth-delay product, which is the minimum message size that can theoretically saturate the link.

Table 1 shows the bandwidth-delay products for a number of leading-edge interconnect implementations. The best bandwidth-delay products hover close to 2 KB. Therefore, we will choose 2 KB as our target bandwidth-delay product threshold. It reflects the state of the art in current switch technology and an aggressive goal for future leading-edge switch technologies. Below this threshold, we presume that the messages will not benefit from a dedicated point-to-point circuit because such messages will not be able to fully utilize the available bandwidth. Such messages would be routed over multiple links or a lower-bandwidth interconnect that is used for collectives.

Therefore, in addition to a high-bandwidth hybrid interconnect, we see the need for a second low-latency low-bandwidth interconnect for handling collective communications with small payloads. A tree network, similar to the one used in the IBM BlueGene/L, does not incur a large additional cost because it is designed to handle low-bandwidth messages and can therefore employ considerably less expensive hardware components. This network could also carry small point-to-point messages that do not benefit from the high-bandwidth hybrid interconnect. However, such messages could also be routed over the high-bandwidth links without provisioning a dedicated path.

2.5 Hypothesis

We summarize the applicability of HFAST as follows: Applications with communication patterns that are *isotropic* (that is, a topologically regular communication pattern) and bounded by a low TDC (*case i*) can be mapped onto regular limited-connectivity topological networks, such as n-dimensional torii or hypercubes. Effective packing of multiple jobs and link/node fault tolerance may still pose a difficult challenge for these types of configurations. Applications with communication patterns which are *anisotropic* (an irregular topology) and bounded by a low TDC (*case ii*) cannot be embedded perfectly in a fixed mesh network, and therefore benefit from adaptive interconnects. Of these codes, if the *maximum* TDC is bounded by a low degree, then bounded-degree approaches such as ICN will be sufficient. For applications where the *average* TDC is bounded by a small number, while the *maximum* TDC is arbitrarily large (*case iii*), the more flexible HFAST approach to allocating packet-switch resources is warranted. The final case is *case iv*, where the TDC of an application consistently equals the number of processors used by the application. The full-bisection capability of an FCN is still required for such applications.

We hypothesize that *case i* covers a few of the studied applications, indicating mesh/torus interconnects may be sufficient for a diverse workload (fault-tolerance and job-packing problems with the fixed-topology interconnects notwithstanding). Likewise, we surmise that few codes will be described

Name	Lines	Discipline	Problem and Method	Structure
Cactus [3]	84,000	Astrophysics	Einstein's Theory of GR via Finite Differencing	Grid
LBMHD [15]	1,500	Plasma Physics	Magneto-Hydrodynamics via Lattice Boltzmann	Lattice/Grid
GTC [14]	5,000	Magnetic Fusion	Vlasov-Poisson Equation via Particle in Cell	Particle/Grid
SuperLU [13]	42,000	Linear Algebra	Sparse Solve via LU Decomposition	Sparse Matrix
PMEMD	37,000	Life Sciences	Molecular Dynamics via Particle Mesh Ewald	Particle
PARATEC [6]	50,000	Material Science	Density Functional Theory via FFT	Fourier/Grid

Table 2: Overview of scientific applications examined in this work.

by *case iv*: thereby undercutting the motivation for using FCNs. Thus, for the wide class of applications with low average TDC covered by cases *i-iii*, the HFAST approach offers a much lower cost solution than FCNs, with the flexibility to effectively handle anisotropic communication patterns and nodes with high TDC. Additionally, the HFAST architecture gracefully handles application mapping and link/node failures. Nevertheless, it is important to keep in mind that the class of applications characterized by high average TDC is best served by FCNs.

One important missing piece in HFAST is direct support for the kinds of high-radix communication patterns required by MPI collective operations like broadcasts and reductions. We hypothesize that the typical message size for collective communication is much smaller than the bandwidth-delay product of the high-speed network and therefore will benefit little from a dedicated circuit provisioned by HFAST, and that such messages are better suited for a dedicated low-bandwidth tree network.

3. METHODOLOGY

3.1 IPM: Low-overhead MPI profiling

In order to profile the communication characteristics of scientific applications for our study, we employ the Integrated Performance Monitoring (IPM) tool – an application profiling layer that allows us to non-invasively gather the communication characteristics of these codes as they are run in a production environment. IPM brings together multiple sources of performance metrics into a single profile that characterizes the overall performance and resource usage of the application. It maintains low overhead by using a unique hashing approach which allows a fixed memory footprint and minimal CPU usage. IPM is open source, relies on portable software technologies and is scalable to thousands of tasks.

Since most of the workload we are interested in uses MPI for parallelism, we have focused on implementing IPM through the name-shifted profiling interface to MPI. The use of the profiling interface to MPI is of widely recognized value in profiling MPI codes [18, 16]. The name-shifted or PMPI interface allows each MPI call to be wrapped by profiling code that collects communication performance information.

IPM collects a wide variety of communication information through this interface, storing it in a fixed size hash table. In this work, we are principally using the information which encodes the number and timing of each MPI call. We gather communication information on each task about each MPI call with a unique set of arguments. Arguments to MPI calls contain message buffer size, as well as source and destination information. In some cases we also track information from the `MPI_Status` structure. For instance, in the case of `MPI_Send`, IPM keeps track of each unique buffer size and destination, the number of such calls, as well as the total, minimum and maximum runtimes to complete the call. IPM also allows code regions to be defined, enabling us to separate application initialization from steady state computation and communication patterns, as we are interested, primarily, in the communication topology for the application in its post-initialization steady state. We ran using IPM on Seaborg, the NERSC IBM SP. Overall, the IPM analysis captures the topology and nature of communication, which is essential for understanding application applicability to the HFAST methodology.

3.2 Evaluated Applications

In order to evaluate the potential effectiveness of utilizing hybrid interconnect networks, we must first develop an understanding of the communication requirements of scientific applications across a broad spectrum of parallel algorithms.

In this section we highlight the salient features of the applications studied in this work. The high level overview of the codes and methods is presented in Table 2. Each of these applications is actively run at multiple supercomputing centers, consuming a sizable amount of computational resources. Detailed descriptions of the algorithms and scientific impact of these codes has been detailed elsewhere [5, 13, 6, 14, 15].

Together, this sampling of applications spans the characteristics of a great many more applications, especially with respect to communication pattern. For instance, though we examine PARATEC in the present work, its core algorithm has the communication characteristics of many other important plane wave DFT codes (CPMD, VASP, etc.). Likewise we expect a large number of finite difference and particle-mesh codes to exhibit similar communication patterns, based on our study of Cactus and PMEMD. Certain reduced quantities important to the present study, such as communication degree, should be largely dictated by the problem solved and algorithmic methodology. For instance, in the case of Cactus where finite differencing is performed using a regular grid, the number of neighbors is determined by the dimensionality of the problem and the stencil size. Profiling a greater number of applications would of course improve the coverage of this study, but the six applications studied here broadly represent a wide range of scientific disciplines and modern parallel algorithms.

We also note that in order to study steady-state communication characteristics, we use IPM's regioning feature, which allows us to examine only the profiling data from one section of the code. In particular, we eliminate the large amounts of communication caused by SuperLU during initialization, primarily consisting of large transfers of the input matrix from one node to all of the others.

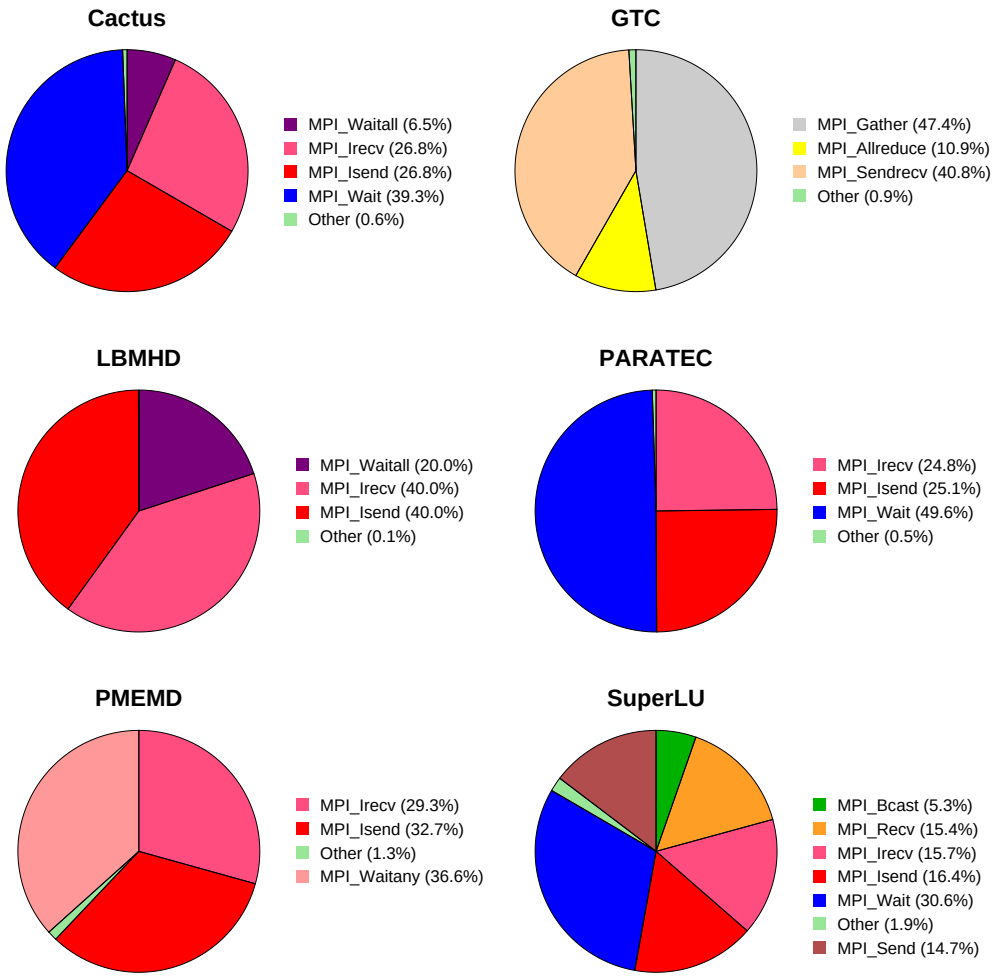

Figure 2: Relative number of MPI communication calls for each of the codes.

4. APPLICATION COMMUNICATION CHARACTERISTICS

As a first step in understanding the applicability of HFAST, we analyze the communication characteristics of the six scientific codes in our study. We use the IPM profiling layer to quantify the type and frequency of application-issued MPI calls, as well as identify the buffer sizes utilized for both point-to-point and collective communications. Lastly, we study the communication topology of each application, determining the average and maximum TDC of each.

4.1 Call counts

The breakdown of MPI communication call types is shown in Figure 2, for each of our studied applications. Notice that overall, there is only a small subset of calls used by these applications relative to the entire MPI library. Here, we only consider calls dealing with communication and synchronization, and do not analyze other types of MPI functions which do not intiate or complete message traffic. Most codes use a small variety of MPI calls, and utilize mostly (over 90% of all MPI calls) point-to-point communication functions, except in the case of GTC, which relies heavily on `MPI_Gather`. Observe also that non-blocking communication is the predominant point-to-point communication model for these codes.

4.2 Collectives Buffer Size

Figure 3 shows the cumulatively histogramed buffer sizes for collective communication, across all six applications. Observe that relatively small buffer sizes are predominantly used; in fact, about 90% of the collective messages are 2 KB or less (shown as the bandwidth-delay product by the pink line), while almost half of all collective calls use buffers less than 100 bytes. This confirms our earlier hypothesis that collective message sizes are generally small, and could be accommodated by a low-bandwidth tree interconnect.

4.3 Point-to-Point Buffer Size

The cumulatively histogramed buffer sizes for point-to-point communication are shown in Figure 4 for each of the applications; once again the 2 KB bandwidth-delay product is shown by the pink vertical lines. Here we see a wide range of communication characteristics across the applications. Cactus and LBMHD use a relatively small *number* of

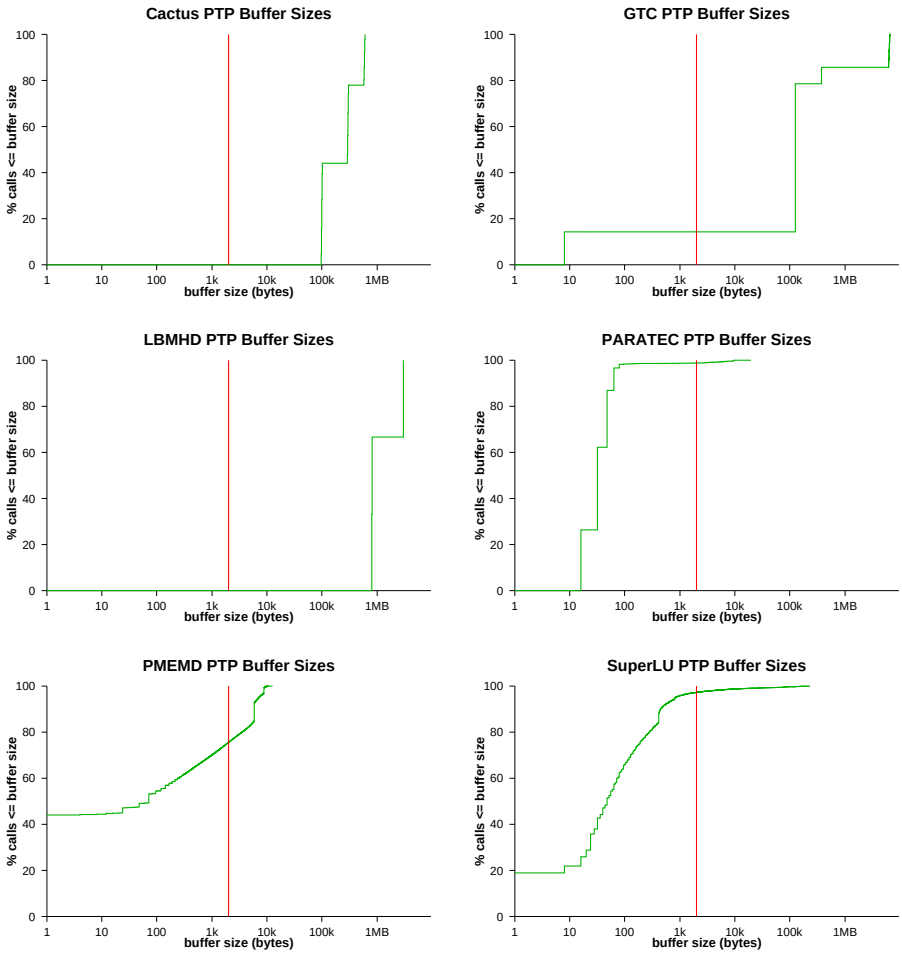

Figure 4: Buffer sizes distribution for point-to-point communication. The pink line demarcates the bandwidth-delay product.

sizes, but each of these buffers is relatively large. GTC employs small communication buffers, but over 80% of the messaging occurs with 1 MB or larger data transfers. In addition, it can be seen that SuperLU, PMEMD, and PARATEC use many different buffer sizes, ranging from a few bytes to over a megabyte in some cases. Overall, Figure 4 demonstrates that unlike collectives (Figure 3), point-to-point messaging uses a wide range of buffers, as well as large message sizes – sometimes on the order of megabytes.

4.4 Connectivity

In this section, we present the topological connectivity for each application by representing the volume and pattern of message exchanges between all tasks. By recording statistics on these message exchanges we can form an undirected graph which describes the topological connectivity required by the application. This graph is undirected because we assume that switch links are bi-directional. As a result, the topologies shown are always symmetric about the diagonal. From this graph we can calculate certain reduced quantities which describe the communication pattern at a coarse level. Such reduced metrics are important in being able to make direct comparisons between applications. In particular, we examine the maximum and average TDC (connectivity) of each code, a key metric for evaluating the potential of the HFAST approach. In addition to showing the max and average, we explore a thresholding heuristic based on the bandwidth-delay product (see Section 2.4) that disregards smaller latency-bound messages. In many cases, this thresholding lowers the average and maximum TDC substantially.

As shown in Figure 5, we see that GTC has a regular communication structure. This particle-in-cell calculation uses a one-dimensional domain decomposition across the toroidal computational grid, causing each processor to exchange data with its two neighbors as particles cross the left and right boundaries. Additionally, there is a particle decomposition within each toroidal partition, resulting in an average TDC of 4 with a maximum of 17 for the $P = 256$ test case. This maximum TDC is further reduced to 10 when using our 2 KB bandwidth-delay product message size cutoff. These small TDC requirements clearly indicate that most links on an FCN are not being utilized for the GTC simulation.

In Figure 6, we see that the ghost-zone exchanges of Cac-

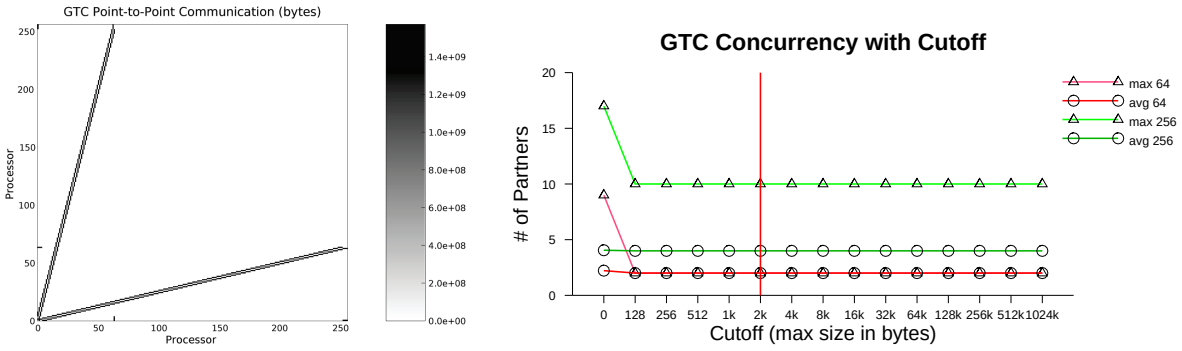

Figure 5: GTC (a)volume of communication at P=256 and (b) effect of thresholding on TDC for P=64,256

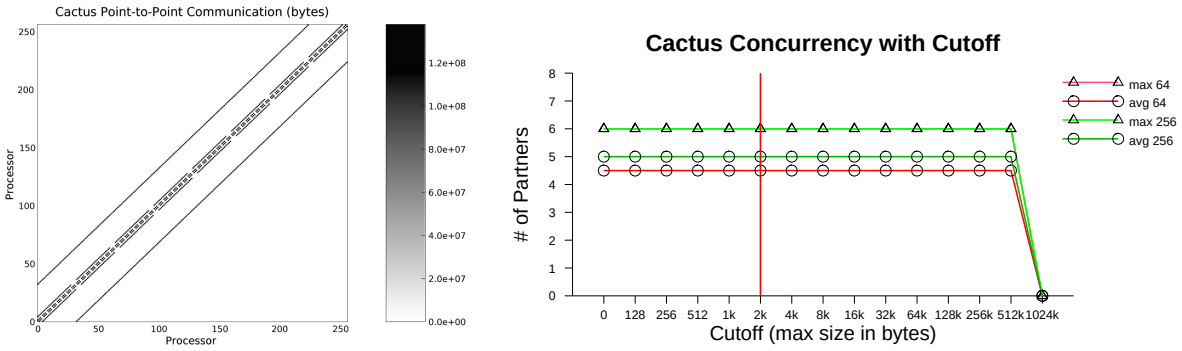

Figure 6: Cactus (a)volume of communication at P=256 and (b) effect of thresholding on TDC for P=64,256

tus result in communications with "neighboring" nodes, represented by diagonal bands. In fact, each node communicates with at most 6 neighbors due to the regular computational structure of this 3D stencil code. On average, the TDC is 5, because some nodes are on the boundary and therefore have fewer communication partners. The maximum TDC is independent of run size (as can be seen by the similarity of the $P = 64$ and $P = 256$ lines) and is insensitive to thresholding, which suggests that no pattern of latency-bound messages can be excluded. Note however that the low TDC indicates limited utilization of an FCN architecture.

The connectivity of LBMHD is shown in Figure 7. Structurally, we see that the communication, unlike Cactus, is scattered (not occuring on the diagonal). This is due to the interpolation between the diagonal streaming lattice and underlying structure grid. Note that although the 3D LBMHD streams the data in 27 directions, the code is optimized to reduce the number of communicating neighbors to 12, as seen in Figure 7. This degree of connectivity is insensitive to the concurrency level, as can be seen by the overlap of the $P = 64$ and $P = 256$ graphs. The maximum TDC is insenstivive to thresholding, showing that there is no pattern of latency-bound messages to be excluded within the HFAST model.

Figure 8 shows the connectivity and TDC for our SuperLU runs. The complex communication structure of this computation results in many point-to-point message transmissions: in fact, without thresholding the connectivity is equal to P. However, by removing the latency-bound messages by thresholding at 2 KB, the average and maximum TDC is reduced to 30 for the 256 processor test case. Also, note that the connectivity of SuperLU is a function of concurrency, scaling proportionally to $\sqrt{P}$, as can be seen by the different TDC requirements of $P = 64$ and $P = 256$ in Figure 8. SuperLU–DIST communication is described more completely in a 2003 paper by Li and Demmel [13].

Figure 9 shows the complex structure of communication of the PMEMD particle mesh ewald calculation. Here the maximum and average TDC is equal to P and the degree of connectivity is a function of concurrency. For the spatial decomposition used in this algorithm, each task's data transfer with another task drops off as their spatial regions become more distant. The rate of this drop off depends strongly on the molecule(s) in the simulation. Observe that for $P = 256$, thresholding at 2 KB reduces the *average* connectivity to 55, even though the maxium TDC remains at 256. This disparity between the maximum and average TDC can be effectively addressed using our proposed HFAST methodology.

Finally, Figure 10 shows the communication requirements of PARATEC. This communication-intensive code relies on global data transposes during its 3D FFT calculations, resulting in large, global message traffic [6]. Here the maximum and average TDC is equal to P, and the connectivity is

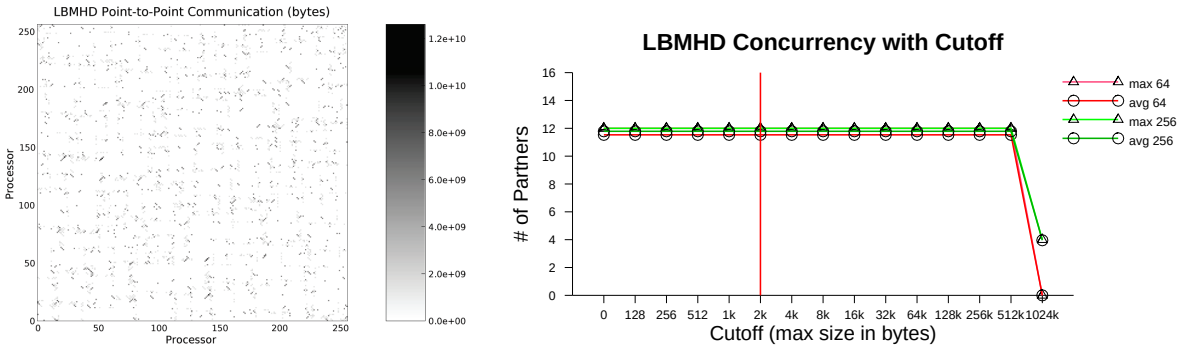

Figure 7: LBMHD (a)volume of communication P=256 and (b) effect of thresholding on TDC for P=64,256

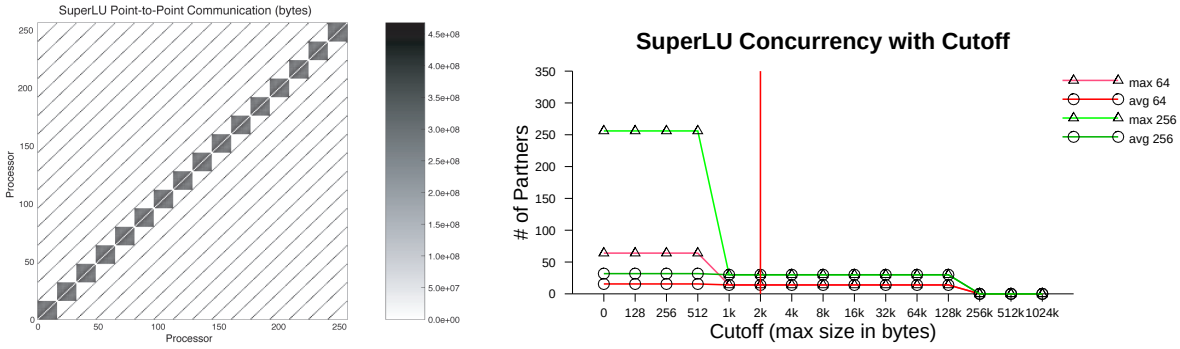

Figure 8: SuperLU (a)volume of communication at P=256 and (b) effect of thresholding on TDC for P=64,256

insensitive to thresholding. Only with a relatively large message size cutoff of 32 KB do we see any reduction in the number of communicating partners required. Thus, PARATEC represents the class of codes that make use of the bisection bandwidth that a fully-connected network configuration provides. This type of communication presents formidable challenges to the HFAST approach or any low-degree connectivity solution.

Table 3 presents a summary of the application communication characteristics derived in this section. (SuperLU and PMEMD exhibit misleadingly low median point-to-point buffer sizes, but this is due to the fact they sometimes send with buffer sizes of 0 bytes, in cases where a communicating partner expects a message that is not necessary for the computation.)

In the next section, we utilize these results to determine the applicability of HFAST to each of the applications we study.

5. ANALYSIS

Based on the analysis of Section 4 we now investigate the potential advantages of the HFAST approach. For each application, we examine the TDC of each node, thresholded by the 2 KB bandwidth-delay product (defined Section 2.4), and determine the number of switches required to build an HFAST network compared with FCN configurations.

One important simplifying assumption we apply to our analysis is that each node contains only a single processor. While most practical systems will likely use SMP nodes, the analysis would need to consider bandwidth localization algorithms for assigning processes to nodes in addition to the analysis of the interconnection network requirements. However, bandwidth localization is a separable issue — one that unnecessarily complicates our analysis of the interconnect behavior and requirements. Therefore, we focus exclusively on single-processor nodes in this paper, and leave the analysis of SMP nodes for future work.

5.1 Collectives

Consistent with our earlier hypothesis in Section 2.5, Figure 3 shows that nearly all of the collective communication payload sizes fall below 2 KB. This result is consistent with previous research [18] and validates IBM's architectural decision to dedicate a separate lower-bandwidth network on BG/L for collective operations. One could imagine computing a minimum-latency routing pattern that is overlaid on the high-bandwidth interconnect topology, but the complexity of such an algorithm is out of the scope of this paper. Therefore, we will presume a lower-bandwidth, low-cost dedicated-tree network, similar to the one in BG/L, will carry the collective messages and possibly small-payload point-to-point messages, and focus the remaining analysis on using HFAST to accelerate large payload point-to-point messages.

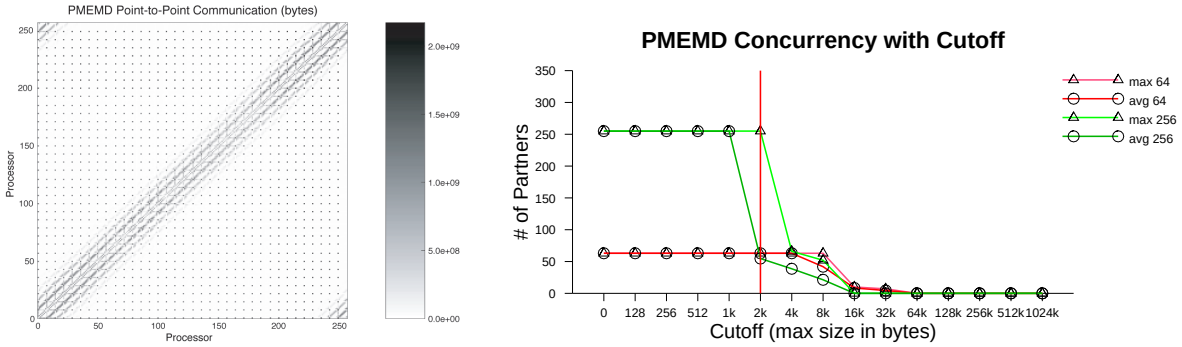

Figure 9: PMEMD (a)volume of communication at P=256 and (b) effect of thresholding on TDC for P=64,256

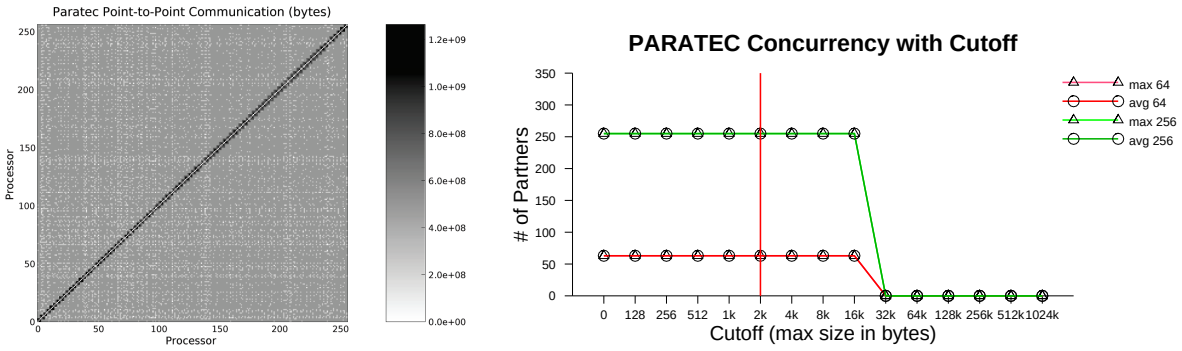

Figure 10: PARATEC(a)volume of communication at P=256 and (b) effect of thresholding on TDC for P=64,256

5.2 Point To Point Traffic

We now discuss each of the applications and consider the class of network best suited for its communication requirements. First, we examine the three codes exhibiting the most regularity in their communication exchanges: Cactus, LBMHD, and GTC. Cactus displays a bounded TDC independent of run size, with a communication topology that isomorphically maps to a regular mesh; thus a fixed 3D mesh/torus would be sufficient to accommodate these types of stencil codes, although an adaptive approach would also fulfill Cactus's requirements (consistent with *case i* described in Section 2.5). LBMHD also displays a low degree of connectivity, but while its communication pattern is isotropic, the structure is not isomorphic to a regular mesh, thereby requiring an adaptive approach such as ICN or HFAST network (*case ii*). Although GTC's primary communication pattern is isomorphic to a regular mesh, it has a maximum TDC that is quite higher than the average due to important connections that are *not* isomorphic to a mesh(*case iii*). Thus, neither a fixed mesh/torus topology, nor the bounded-degree adaptive network of ICN, would be well suited for this class of computation. Here, the HFAST approach holds a clear advantage, since additional packet switch resources can dynamically be assigned to the subset of nodes with higher TDC requirements.

SuperLU and PMEMD exhibit anisotropic communication patterns with a TDC that scales with the number of processors. Additionally, PMEMD has widely differing maximum and average TDC. However, with thresholding, the proportion of processors that have messages that would benefit from the dedicated links is large but stays bounded to far less than the number of processors involved in the calculation (consistent with *case iii*). A regular mesh or torus would be inappropriate for this class of computation, but an FCN remains underutilized. However, the HFAST network can be dynamically reconfigured to satisfy the requirements of these complex applications.

Finally, PARATEC represents the communications requirements for a large class of important chemistry and fluids problems where part of the problem is solved in fourier space. It requires large global communications involving large messages that fully utilize the FCN and are therefore consistent with *case iv*. PARATEC's large global communications are a result of the 3D FFTs used in the calculation, which require two stages of global 3D transposes. The first transpose is non-local and involves communications of messages of similar sizes between all the processors, resulting in the uniform background of 32 KB messages. In the second transpose, processors only communicate with neighboring processors, resulting in additional message traffic along the diagonal of the graph. A more detailed description of the communication requirements can be found in [6].

Code	Procs	% PTP Calls	median PTP buffer	% Col. calls	median Col. buffer	TDC @ 2KB cutoff(max,avg)	FCN Circuit Utilization (avg.)
GTC	64	42.0	128k	58.0	100	2, 2	3%
	256	40.2	128k	59.8	100	10, 4	2%
Cactus	64	99.4	299k	0.6	8	6, 5	9%
	256	99.5	300k	0.5	8	6, 5	2%
LBMHD	64	99.8	811k	0.2	8	12, 11.5	19%
	256	99.9	848k	0.1	8	12, 11.8	5%
SuperLU	64	89.8	64	10.2	24	14, 14	22%
	256	92.8	48	7.2	24	30, 30	25%
PMEMD	64	99.1	6k	0.9	768	63, 63	100%
	256	98.6	72	1.4	768	255, 55	22%
PARATEC	64	99.5	64b	0.5	8	63, 63	100%
	256	99.9	64	0.1	4	255, 255	100%

Table 3: Summary of code characteristics for point-to-point (PTP) and collective (Col.) communications.

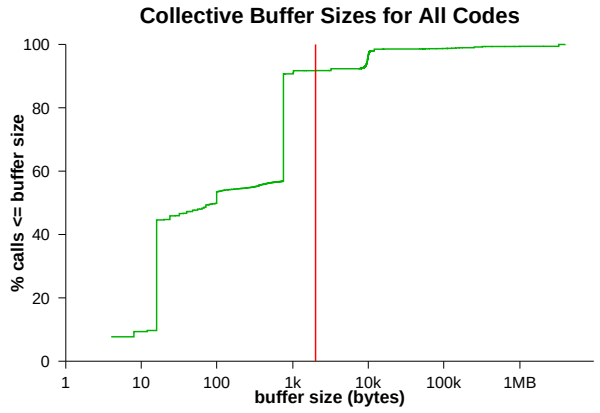

Figure 3: Buffer sizes distribution for collective communication for all codes. The pink line demarcates the bandwidth-delay product.

PARATEC is an example where the HFAST solution is inappropriate. The large global communication requirements can only be effectively provisioned with an FCN network.

In summary, only one of the six codes studied offered a communication pattern that maps isomorphically to a 3D mesh network topology (*case i*). Only one of the codes fully utilizes the FCN at large scales (*case iv*). The preponderance of codes can benefit from an adaptive communication network that uses a lower radix active switching solution. This result is consistent with the hypothesis stated in Section 2.5.

5.3 HFAST Cost Model

Fat Tree networks are built in layers of N-port switches such that L layers can be used to create a fully connected network for P processors where $P = 2 * (N/2)^L$. However, the number of switch ports in the interconnection network per processor grows at a rate of $(1 + 2(L - 1))$. So, for instance, a 6 layer fat-tree composed of 8-port switches requires 11 switch ports for each processor for a network of 2048 processors! Messages must traverse up to 21 layers of packet switches to reach their destination. While state-of-the-art packet switches typically contribute less than 50ns to the message latency, traversing 21 layers of them can become a significant component of the end-to-end latency.

With the HFAST solution, the number of ports required for the passive circuit switch grows by the same proportion as a full FCN. However, the cost per port for the circuit switch is far less than the cost per port for a packet switch using a leading-edge technology. Packet switches, the most expensive component per-port, can be scaled linearly with the number of processors used in a given system design. So unlike a fixed topology mesh, hypercube, or torus interconnect, the cost of HFAST is not entirely linearly-proportional to the number of processors because of the cost of the fully connected circuit switch. However, the cost of the most *expensive* component, the packet switches and network interface cards for the hosts, scales proportionally with the number of processors.

We introduce a simple cost function that represents the applicability of HFAST given the TDC of each node in the computation. To simplify our analysis, we present an upper-bound that does not use any sophisticated graph-theoretic methods to optimize mappings. In addition, we assume a homogenous active switch block size of 16 ports.

Generally, the cost $Cost_{HFAST}$ is given by

$$N_{active} * Cost_{active} + Cost_{passive} + Cost_{collective},$$

where N_{active} is the number of active switch blocks required, and $Cost_{active}$, $Cost_{passive}$, and $Cost_{collective}$ are the respective costs of a single active switch block, the passive switch, and the collective network. HFAST is effective if $Cost_{HFAST} < Cost_{fat-tree}$.

For a given code, we examine each node in turn. For each node, if the TDC is less than the active switch block size (in our case 15), we assign it one active switch block. However, if the TDC is greater than 15, we assign it the number of switch blocks needed to build a tree network large enough to communicate with all of the node's partners. This algorithm uses potentially twice as many switch ports as an optimal embedding, but it has the advantage that it will complete in linear time.

As an example, we determine the cost for Cactus, a code that exhibits an average and maximum TDC of 6 per node. For each node, then, we assign a single active switch block, giving us $N_{active} = P$. That is, the number of active switch blocks required is equal to the number of processors in a run. For codes like PMEMD that exhibit a maximum TDC that is higher than the average, additional packet switch blocks

can be provisioned (if available) to construct a higher-radix tree network to support the higher-degree communication pattern required by that subset of processors.

The procedure outlined above creates an efficient mapping when average TDC is less than the switch block size. However, the method yields a far less efficient mapping, relative to a Fat-Tree, for codes with higher TDC. The mapping procedure uses the packet switches exclusively for fan-in and fan-out of connections between nodes, and therefore does not exercise the full internal bisection connectivity of these switch blocks.

The general problem of switch block assignment can be reduced to the clique-mapping problem where tightly interconnected cliques are mapped to switch blocks in order to maximize the utilization of internal switch connectivity. The optimal solution to the fully generalized clique-mapping problem is NP-complete [12]. However, the fact that the switch blocks are of finite size bounds the complexity of the problem to less than NP-complete, but it still involves a large search space. We are investigating various heuristics that provide sub-optimal solutions in polynomial time.

6. CONCLUSIONS + FUTURE WORK

Most high-end computing platforms currently employ fully interconnected networks whose cost grows superlinearly relative to system size. Thus these interconnect technologies will become impractical for next-generation peta-flop platforms employing tens (or hundreds) of thousands of processors. Fixed networks (such as 3D meshes/torii), on the other hand, are inflexibly bounded in their topological degree and suffer the overheads of job fragmentation. In this work, we propose the HFAST interconnect which combines passive and active switch technology to create dynamically reconfigurable network topologies. Our solution maintains a linear cost function between the expensive (active switch) components and system scale, allows anisotropic communication patterns within the applications, and obviates the need for job-packing by the batch-system. We have analyzed 6 parallel codes that represent a wide array of scientific algorithms and associated communication topologies. We have shown that these codes have communication requirements that typically underutilize the capabilities of an FCN, but also do not usually map isomorphically to a fixed n-dimensional network like a mesh or torus. This suggests constructing a hybrid interconnect that supports a lower average topological degree (like mesh networks), but uses circuit switches so the topology can adapt to code requirements that otherwise cannot map to the fixed mesh topology. We show that the costs of the circuit switch components of HFAST scale similarly to a fat-tree, but the cost per port of the circuit switch is far less than the cost of components used to implement an FCN. The cost of the most expensive component, the packet switches, scales linearly with system size just as they would for a 3D mesh interconnect. HFAST also offers benefits for fault-tolerance and job scheduling that are not available from a mesh interconnect.

The cost estimates presented in this work are based on a rather simple topology mapping algorithm that is not optimal and in many cases is an overestimate of the optimal mapping and therefore cost. The simple approach we have taken allows embedding arbitrary communication topologies in linear time, but may miss certain more complex highly optimal mappings between switch and application. An optimal process mapping may consume as little as half as many ports as the general approach detailed above, but is an NP-complete problem for embedding arbitrary communication graphs. For problems that are consistent with *case i* and *case ii*, there exist algorithms with bounded complexity for communication graph mapping. In future work, we will investigate clique-mapping techniques to improve the quality of our mapping algorithm. We may also adapt the genetic programming approaches used for optimizing the fixed switch topology of the Flat Neighborhood Networks [11] to optimize the embedding. An even more promising approach is to apply runtime iterative or adaptive approaches that incrementally arrive on an optimal embedding by monitoring runtime communication and gradually optimizing the switch topology to minimize communication time as described in Section 2.3.

In future work we hope to expand the scope of both the applications profiled and the data collected through IPM. The low overhead of IPM profiling opens up the possibility of the characterization of large and diverse application workloads. We will also pursue more detailed performance data collection. For instance producing a full chronological communication trace of most applications would incur significant performance penalties; however, computing a time-windowed TDC as the application progresses would not. By studying the time dependence of communication topology one could expose opportunities to reconfigure an HFAST switch as the application is running. It is expected that different program phases will have different communication needs and might benefit from such runtime reconfiguration.

7. REFERENCES

[1] Science case for large-scale simulation. In D. Keyes, editor, *DOE Office of Science Workshop*, June 2003.

[2] Workshop on the roadmap for the revitalization of high-end computing. In D.A. Reed, editor, *Computing Research Association*, June 2003.

[3] Cactus Homepage. `http://www.cactuscode.org`, 2004.

[4] Top 500 supercomputer sites. `http://www.top500.org`, 2005.

[5] A. Canning, J. Carter, J. Shalf, and S. Ethier. Scientific computations on modern parallel vector systems. In *Proceedings of the IEEE Conference on Supercomputing*, 2004.

[6] A. Canning, L.W. Wang, A. Williamson, and A. Zunger. Parallel empirical pseudopotential electronic structure calculations for million atom systems. *J. Comput. Phys.*, 160:29, 2000.

[7] Roger D. Chamberlain, Ch'ng Shi Baw, and Mark A. Franklin. Gemini: An optical interconnection network for parallel processing. *IEEE Trans. on Parallel and Distributed Systems*, 13, October 2002.

[8] T. DeFanti, M. Brown, J. Leigh, O. Yu, E. He, J. Mambretti, D. Lillethun, and J. Weinberger. Optical switching middleware for the optiputer. *IEICE Trans. FUNDAMENTALS/COMMUN./ELECTRON./INF. & SYST*, February 2003.

[9] Hans Eberle and Nils Gura. Separated high-bandwidth and low-latency communication in the cluster interconnect clint. In *Proceedings of the IEEE*

Conference on Supercomputing, 2002.

[10] Vipul Gupta and Eugen Schenfeld. Performance analysis of a synchronous, circuit-switched interconnection cached network. In *ICS '94: Proceedings of the 8th international conference on Supercomputing*, pages 246–255, New York, NY, USA, 1994. ACM Press.

[11] Thomas Hauser, Timothy I. Mattox, Raymond P. LeBeau, Henry G. Dietz, and P. George Huang. High-cost cfd on a low-cost cluster. In *Proceedings of the IEEE Conference on Supercomputing, Dallas, Texas*, November 4-10 2000.

[12] L.T. Kou, L.J. Stockmeyer, and C.K. Wong. Covering edges by cliques with regard to keyword conflicts and intersection graphs. 21:135–138, 1978.

[13] Xiaoye S. Li and James W. Demmel. Superlu–dist: A scalable distributed-memory sparse direct solver for unsymmetric linear systems. *ACM Trans. Mathematical Software*, 29(2):110–140, June 2003.

[14] Z. Lin, S. Ethier, T.S. Hahm, and W.M. Tang. Size scaling of turbulent transport in magnetically confined plasmas. *Phys. Rev. Lett.*, 88, 2002.

[15] A. Macnab, G. Vahala, P. Pavlo, , L. Vahala, and M. Soe. Lattice boltzmann model for dissipative incompressible MHD. In *Proc. 28th EPS Conference on Controlled Fusion and Plasma Physics*, volume 25A, 2001.

[16] Rolf Rabenseifner. Automatic profiling of MPI applications with hardware performance counters. In *Proceedings of the 6th European PVM/MPI User's Group Meeting (EuroPVM/MPI)*, pages 35–42, September 1999.

[17] H.D. Simon, W.T. Kramer, W. Saphir, J. Shalf, D.H. Bailey, L. Oliker, M. Banda, C. W. McCurdy, J. Hules, A. Canning, M. Day, P. Colella, D. Serafini, M.F. Wehner, and P. Nugent. Science-driven system architecture: A new process for leadership class computing. *Journal of the Earth Simulator*, 2, January 2005.

[18] Jeffery S. Vetter and Frank Mueller. Communication characteristics of large-scale scientific applications for contemporary cluster architectures. In *Proceedings of the 16th International Parallel and Distributed Processing Symposium (IPDPS)*, 2002.

[19] Jeffrey S. Vetter and Andy Yoo. An empirical performance evaluation of scalable scientific applications. In *Proceedings of the IEEE Conference on Supercomputing*, 2002.