Communication Requirements and Interconnect Optimization for High-End Scientific Applications

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

***Abstract*— The path towards realizing peta-scale computing is increasingly dependent on building supercomputers with unprecedented numbers of processors. To prevent the interconnect from dominating the overall cost of these ultra-scale systems, there is a critical need for high-performance network solutions whose costs scale linearly with system size. This work makes several unique contributions towards attaining that goal. First, we conduct one of the broadest studies to date of high-end application communication requirements, 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 (Integrated Performance Monitoring) profiling layer to gather detailed messaging statistics with minimal impact to code performance. Using the derived communication characterizations, we next present fit-trees interconnects, a novel approach for designing network infrastructure at a fraction of the component cost of traditional fat-tree solutions. Finally, we propose the Hybrid Flexibly Assignable Switch Topology (HFAST) infrastructure, which uses both passive (circuit) and active (packet) commodity switch components to dynamically reconfigure interconnects to suit the topological requirements of scientific applications. Overall our exploration leads to a promising directions for practically addressing the interconnect requirements of future peta-scale systems.**

I. Introduction

As 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 grand-challenge applications will have computational requirements that are at least two orders of magnitude larger than current levels [1], [2], [20]. However, as the pace of processor clock rate improvements continues to slow [3], 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.

High performance computing (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 the November 2004, 94 of the 100 systems in the Top500 [21] employed 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 and Cray XT series), whose costs rise linearly with system scale. Indeed, the number of systems using lower degree interconnects such as the BG/L and Cray Torus interconnects has increased from 6 systems in the November 2004 list to 28 systems in the most recent Top500 list of June 2007. 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.

This work presents several unique contributions. First in Section II, we conduct one of the broadest studies to date of high-end application communication requirements, 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 (Integrated Performance Monitoring) profiling layer to gather detailed messaging statistics with minimal impact to code performance. Using the derived messaging statistics, Section III next explores a novel *fit-tree* approach for designing network topologies that require only a fraction the resources utilized by conventional fat-tree interconnects. Finally, in Section IV we propose one methodology for implementing dynamically reconfigurable fit-tree solutions using 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.

Overall results lead to a promising approach for practically addressing the interconnect requirements of future peta-scale systems. Although our three research thrusts — HPC communication characterization, fit-tree design, and HFAST networks — work closely in concert together; each of these components could also be considered as independent contributions, which advance the state-of-the art in their respective areas.

TABLE I
BANDWIDTH DELAY PRODUCTS FOR SEVERAL HIGH PERFORMANCE INTERCONNECT TECHNOLOGIES. THIS IS THE EFFECTIVE PEAK UNIDIRECTIONAL BANDWIDTH DELIVERED PER CPU (NOT PER LINK).

System	Technology	MPI Latency	Peak Bandwidth	Bandwidth Delay Product
SGI Altix	NUMAlink-4	1.1us	1.9 GB/s	2 KB
Cray XT4	Seastar 2	7.3us	1.2 GB/s	8.8 KB
NEC SX-9	IXS Super-Switch	3us	16GB/s	48 KB
AMD Infiniband 4x (Sun/TACC)	IB4x DDR	2.3us	950MB/s	2.2 KB

II. SCIENTIFIC APPLICATION COMMUNICATION REQUIREMENTS

In order to quantify HPC interconnect requirements, we must first develop an understanding of the communication characteristics for realistic scientific applications. Several studies have observed that many applications display communication topology requirements that are far less than the total connectivity provided by fully-connected networks. For instance, the application study by Vetter and Mueller [22], [23] indicates 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. This provides strong evidence that many application communication topologies exercise a small fraction of the resources provided by fully-connected networks.

In this section, we expand on previous studies by exploring detailed communication profiles across a broad set of representative parallel algorithms. 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. Finally, we study the communication topology of each application, determining the average and maximum TDC for bandwidth-limited messaging.

A. IPM: Low-overhead MPI profiling

To profile the communication characteristics of the scientific applications in our study, we employ the Integrated Performance Monitoring (IPM) [13] tool — an application profiling layer that allows us to non-invasively gather the communication characteristics of these codes as they 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 that 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.

IPM collects a wide variety of communication information and stores it in a fixed-size hash table. In this work, we are principally using information that 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. Experiments were run on the NERSC IBM SP and the IBM TJ Watson's BlueGene/L.

B. Message Size Thresholding

Before we explore the TDC for our application suite, we must first quantify the thresholding size for messages that are bandwidth limited. Otherwise, we may mistakenly presume that a given application has a high TDC even if trivially small (latency-bound) messages are sent to majority of its neighbors.

To derive an appropriate threshold, we examine the product of the message bandwidth and the delay (latency) for a given point-to-point connection. The *bandwidth-delay product* 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, we focus on the minimum message size that can theoretically saturate the link, i.e. those messages that are larger than the bandwidth-delay product.

Table I shows the bandwidth-delay products for a number of leading-edge interconnect implementations, where the best performance hovers close to 2 KB. We therefore choose 2 KB as our target bandwidth-limiting messaging threshold. This reflects the state-of-the-art in current switch technology and an aggressive goal for future leading-edge switch technologies. We assume that below this threshold, the latency-bound messages would not benefit from a dedicated point-to-point circuit. Such messages are only affected by topology when it comes to the number of links traversed, and cannot be sped up by increasing available bandwidth.

C. Evaluated Scientific Applications

We now highlight the salient features of the eight applications studied in this work. The high level overview of the codes and methods is presented in Table II. Each of these applications is actively run at multiple supercomputing centers, consuming a sizable amount of computational resources. Descriptions of the algorithms and scientific impacts of these codes have been extensively detailed elsewhere [4]–[6], [12], [16]–[19]; here we present a brief overview of each application.

BBeam3D [19] models the colliding process of two counter-rotating charged particle beams moving at close to the speed of

TABLE II
OVERVIEW OF SCIENTIFIC APPLICATIONS EVALUATED.

Name	Lines	Discipline	Problem and Method	Structure
BBeam3D [19]	28,000	High Energy Physics	Vlasov-Poisson via Particle in Cell and FFT	Particle/Grid
Cactus [5]	84,000	Astrophysics	Einstein's Theory of GR via Finite Differencing	Grid
GTC [17]	5,000	Magnetic Fusion	Vlasov-Poisson via Particle in Cell	Particle/Grid
LBCFD [18]	3,000	Fluid Dynamics	Navier-Stokes via Lattice Boltzmann Method	Grid/Lattice
MADbench [4]	5,000	Cosmology	CMB Analysis via Newton-Raphson	Dense Matrix
PARATEC [6]	50,000	Material Science	Density Functional Theory via FFT	Fourier/Grid
PMEMD [12]	37,000	Life Sciences	Molecular Dynamics via Particle Mesh Ewald	Particle
SuperLU [16]	42,000	Linear Algebra	Sparse Solve via LU Decomposition	Sparse Matrix

light. The application is a 3D particle-in-cell computation that contains multiple models (weak-strong, strong-strong) and multiple collision geometries (head-on, long-range, crossing angle), with collisions calculated self-consistently by solving the Vlasov-Poisson equation using Hockney's FFT method. Thus the code exhibits communication characteristics due to the PIC method as well as the FFT.

Cactus [5] is an astrophysics computational toolkit designed to solve the challenging coupled nonlinear hyperbolic and elliptic equations that arise from Einstein's Theory of General Relativity. Consisting of thousands of terms when fully expanded, these partial differential equations (PDEs) are solved on a regular grid using finite differences on a block domain-decomposed grid distributed over the processors.

The Gyrokinetic Toroidal Code (GTC) is a 3D particle-in-cell (PIC) application developed to study turbulent transport in magnetic confinement fusion [17]. GTC solves the non-linear gyrophase-averaged Vlasov-Poisson equations [15] in a geometry characteristic of toroidal fusion devices. By using the particle-in-cell method, the non-linear PDE describing particle motion becomes a simple set of ordinary differential equations (ODEs) that can be easily solved in the Lagrangian coordinates.

LBCFD [18] utilizes an explicit Lattice-Boltzmann method to simulate fluid flows and to model fluid dynamics. The basic idea is to develop a simplified kinetic model that incorporates the essential physics, and reproduces correct macroscopic averaged properties. LBCFD models 3D simulations under periodic boundary conditions, with the spatial grid and phase space velocity lattice overlaying each other, distributed with a 3D domain decomposition.

Based on the MADspec cosmology code that calculates the maximum likelihood angular power spectrum of the cosmic microwave background (CMB), MADbench [4] is a simplified benchmark that inherits the characteristics of the application without requiring massive input data files. MADbench tests the overall performance of the subsystems of real massively-parallel architectures by retaining the communication and computational complexity of MADspec and integrating a dataset generator that ensures realistic input data. Much of the computational load of this application is due to its use of dense linear algebra.

PARATEC (PARAllel Total Energy Code [6]) performs ab-initio quantum-mechanical total energy calculations using pseudopotentials and a plane wave basis set. In solving the Kohn-Sham equations using a plane wave basis, part of the calculation is carried out in real space and the remainder in Fourier space using specialized parallel 3D FFTs to transform the wavefunctions. The communication involved in these FFTs is the most demanding portion of PARATEC's communication characteristics.

PMEMD (Particle Mesh Ewald Molecular Dynamics [12]) is an application that performs molecular dynamics simulations and minimizations. The force evaluation is performed in an efficiently parallel manner using state of the art numerical and communication methodologies. PMEMD uses a highly asynchronous approach to communication for the purposes of achieving a high degree of parallelism.

SuperLU [16] is a general purpose library for the direct solution of large, sparse, nonsymmetric systems of linear equations on high performance machines. The library routines perform an LU decomposition with partial pivoting as well as a triangular system solve through forward and back substitution. This application relies on sparse linear algebra of various kinds for its main computational kernels, ranging from a simple vector scale to a large triangular solve.

Together, this collection of numerical methods spans the characteristics of a great many more applications, especially with respect to communication patterns. For example the core algorithm of the PARATEC code studied here, has the communication characteristics of many other important plane-wave density functional theory (DFT) calculations. Likewise a large number of finite difference and particle-mesh codes to exhibit similar communication patterns to Cactus and PMEMD. Note that certain quantities relevant to the present study, such as communication degree, are largely dictated by the scientific 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, however the eight applications detailed here broadly represent a wide range of scientific disciplines and modern parallel algorithms under realistic computational demands.

D. Communication Characteristics

We now explore the communication characteristics of our studied applications, by quantifying the MPI call count distributions, collective and point-to-point buffer sizes, and topological connectivity.

1) Call Counts: The breakdown of MPI communication call types is shown in Table III, for each of our studied applications. Here, we only consider calls dealing with communication and synchronization, and do not analyze other types of MPI functions which do not initiate or complete message traffic. Notice that overall, these applications utilize only a small subset of the entire MPI library. Most codes use a small variety of MPI calls,

TABLE III

BREAKDOWN OF MPI COMMUNICATION CALLS, PERCENTAGE OF POINT-TO-POINT (PTP) MESSAGING, MAXIMUM AND AVERAGE TDC THRESHOLDED BY 2 KB, AND FCN UTILIZATION (THRESHOLDED BY 2 KB) FOR EVALUATED APPLICATION ON 256 PROCESSORS.

Function	BB3D	Cactus	GTC	LBCFD	MADbench	PARATEC	PMEMD	SuperLU
Isend	0%	26.8%	0%	40.0%	5.3%	25.1%	32.7%	16.4%
Irecv	33.1%	26.8%	0%	40.0%	0%	24.8%	29.3%	15.7%
Wait	33.1%	39.3%	0%	0%	0%	49.6%	0%	30.6%
Waitall	0%	6.5%	0%	20.0%	0%	0.1%	0.6%	0%
Waitany	0%	0%	0%	0%	0%	0%	36.6%	0%
Sendrecv	0%	0%	40.8%	0%	30.1%	0%	0%	0%
Send	33.1%	0%	0%	0%	32.2%	0%	0%	14.7%
Gather	0%	0%	47.4%	0%	0%	0.02%	0%	0%
(All)Reduce	0.5%	0.5%	11.7%	0.02%	13.6%	0%	0.7%	1.9%
Bcast	0.02%	0%	0.04%	0.08%	6.8%	0.03%	0%	5.3%
% PTP Calls	99.2%	98.0%	40.8%	99.8%	66.5%	99.8%	97.7%	81.0%
TDC (max,avg)	66,66	6,5	10,4	6,6	44,39	255,255	255,55	30,30
FCN Utilization	25.8%	2.0%	1.6%	2.3%	15.3%	99.6%	21.4%	11.7%

and utilize mostly point-to-point communication functions (over 90% of all MPI calls), except GTC, which relies heavily on `MPI_Gather`. Observe also that non-blocking communication is the predominant point-to-point communication model for these codes.

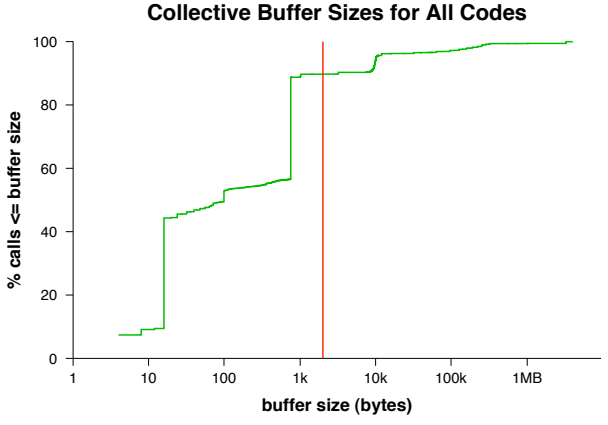


Fig. 1. Buffer sizes distribution for collective communication for all codes. The pink line demarcates the 2 KB bandwidth-delay product.

2) Buffer Sizes for Collectives: Figure 1 presents a cumulative histogram of buffer sizes for collective communication (that is, communication that involves all of the processors), across all eight 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. These results are consistent with previous studies [22], [23] and validate IBM's architectural decision to dedicate a separate lower-bandwidth network on their BlueGene/L machine for collective operations. For this broad class of applications, collective messages are mostly constrained by the latency of the interconnect, regardless of the topological interconnectivity.

3) Point-to-Point Buffer Sizes: A cumulative histogram of buffer sizes for point-to-point communication is shown in Figure 2 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, LBCFD, and BBeam3D use a relatively small *number* of distinct buffer 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, MADbench, and PARATEC use many different buffer sizes, ranging from a few bytes to over a megabyte in some cases. Overall, Figure 2 demonstrates that unlike collectives (Figure 1), point-to-point messaging in these applications uses a wide range of buffers, as well as large message sizes. In fact, for all but three of the codes, buffer sizes larger than the 2 KB bandwidth-delay product account for $> 80\%$ of the overall point-to-point message sizes.

4) Topological Connectivity: We now explore 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 form an undirected graph that describes the topological connectivity required by each application. Note that this graph is undirected as we assume that most modern switch links are bi-directional; as a result, the topologies shown are always symmetric about the diagonal. From this topology graph we then calculate the quantities that describe communication patterns at a coarse level. Such reduced metrics are important in allowing us 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 lower-degree and non-traditional interconnects. In addition to showing the max and average, we explore a thresholding heuristic based on the bandwidth-delay product (see Section II-B) that disregards smaller latency-bound messages. In many cases, this thresholding lowers the average and maximum TDC substantially. An analysis of these results in the context of topological network designs are presented in Section II-E.

Figure 3(a) shows the topological connectivity of BBeam3D for $P = 256$ as well as the effect of eliminating smaller (latency-bound) messages on the number of partners. Observe the high TDC for this charge density calculation due to its reliance on data transposes during the 3D FFTs. For this code, the average TDC is 66 neighbors, while the maximum is 255; both of these are insensitive to cutoffs lower than 64KB messages. BBeam3D thus represents an application class which exhibits a TDC smaller than the full connectivity of a fat tree, with little sensitivity to bandwidth-limited message thresholding.

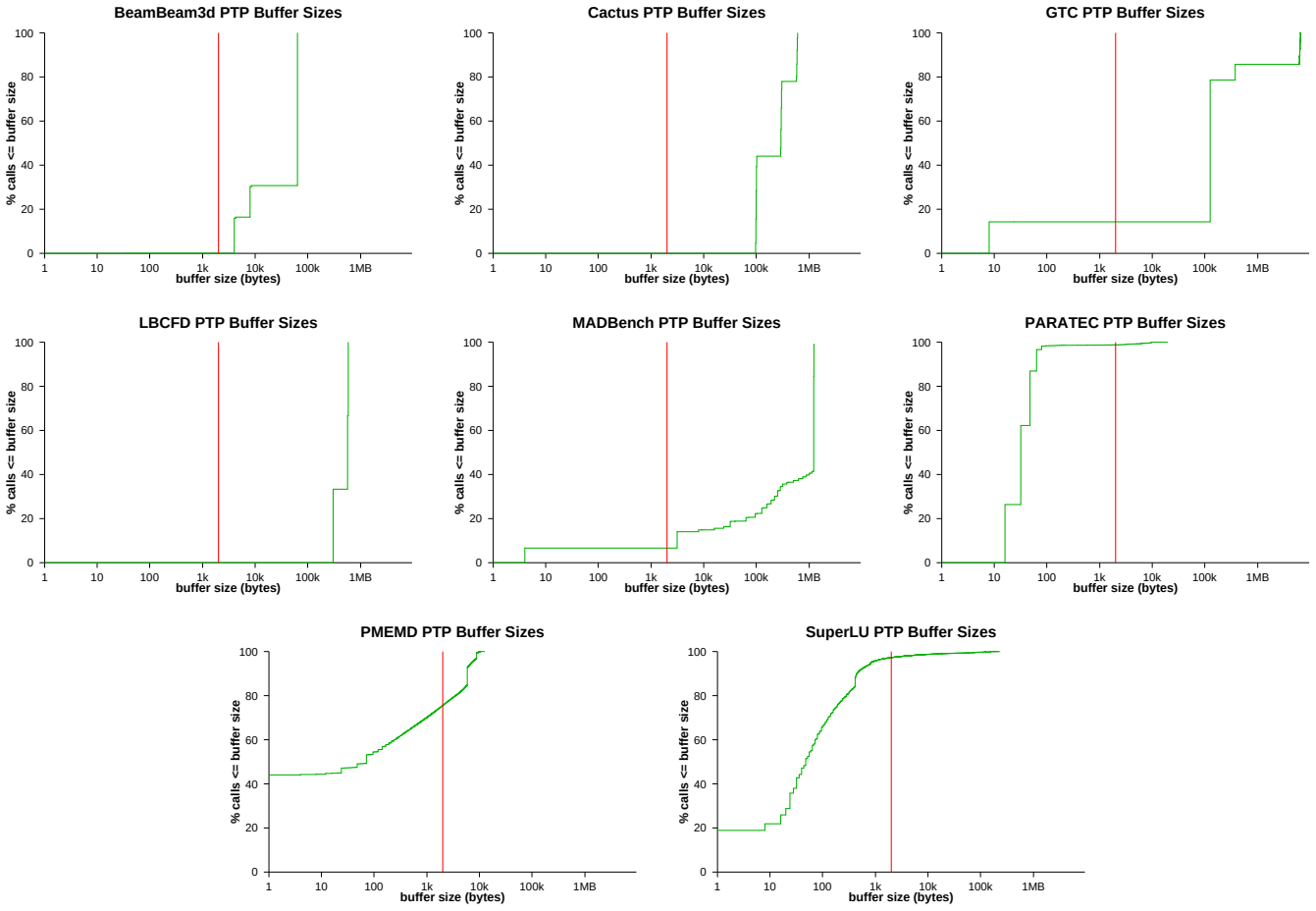


Fig. 2. Buffer sizes distribution for point-to-point communication. The pink lines demarcate the 2 KB bandwidth-delay product.

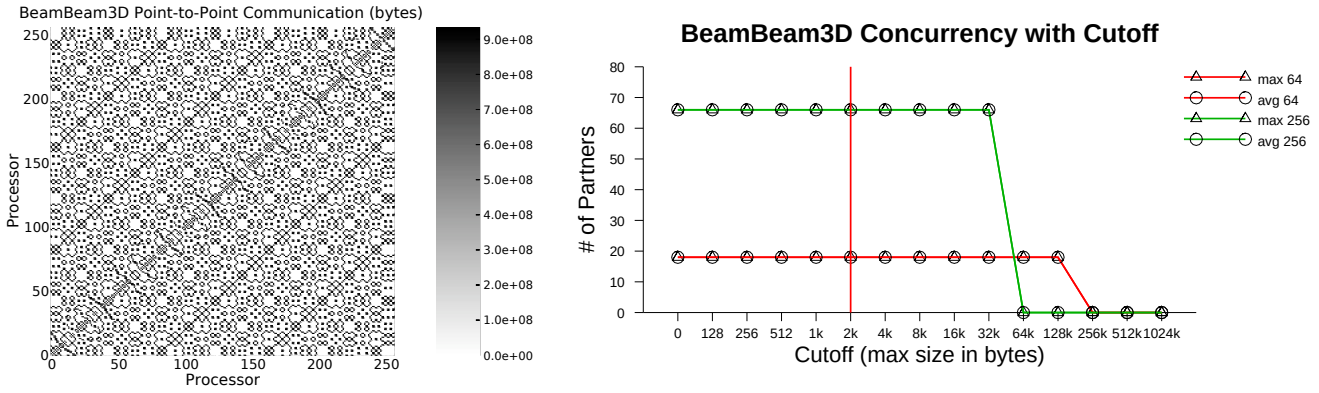
In Figure 3(b), we see that the ghost-zone exchanges of Cactus result in communications with "neighboring" nodes, represented by diagonal bands. In fact, each node communicates with at most 6 neighbors due to the regular computational structure of this 3D stencil code. On average, the TDC is 5, because 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.

As shown in Figure 3(c), we see that GTC exhibits regular communication structure typical of a particle-in-cell calculation that uses a one-dimensional domain decomposition. Each processor exchanges 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 by the GTC simulation.

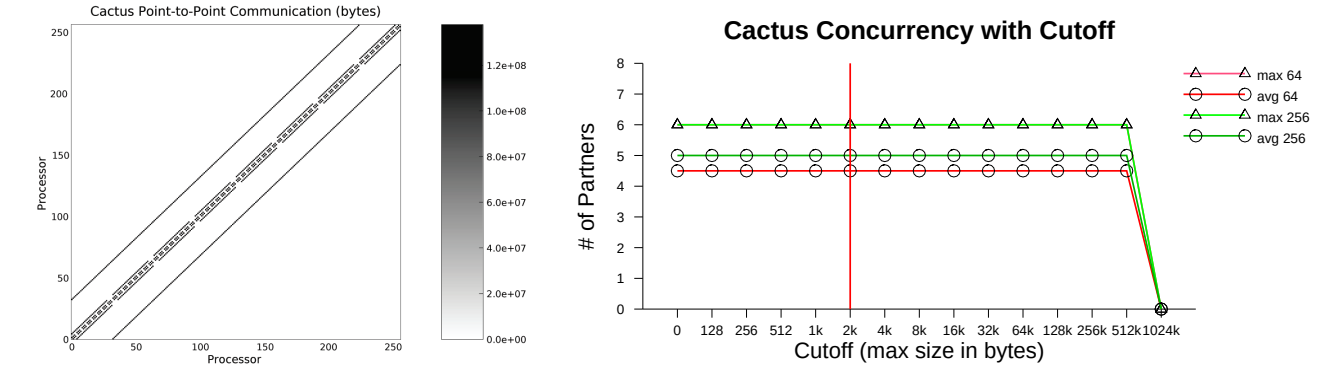
The connectivity of LBCFD is shown in Figure 3(d). Structurally, we see that the communication, unlike Cactus, is scattered (not occurring on the diagonal). This is due to the interpolation between the diagonal streaming lattice and underlying structure grid. Note that although LBCFD streams the data in 27 directions (due to the 3D decomposition), the code is optimized to reduce the number of communicating neighbors to 6, as seen in Figure 3(d). 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 insensitive to thresholding, showing that the majority of the messages are bandwidth-bound.

MADbench's communication topology characteristics are shown in Figure 3(e). Each processor communicates with 38 neighbors on average, dropping to 36 if we eliminate messages smaller than 2 KB. The communication is relatively regular due to the underlying dense linear algebra calculation, with an average and maximum TDC that are almost identical. MADbench is another example of a code whose overall TDC is greater than the connectivity of a mesh/torus interconnect, but still significantly less than the number of links provided by a fat-tree.

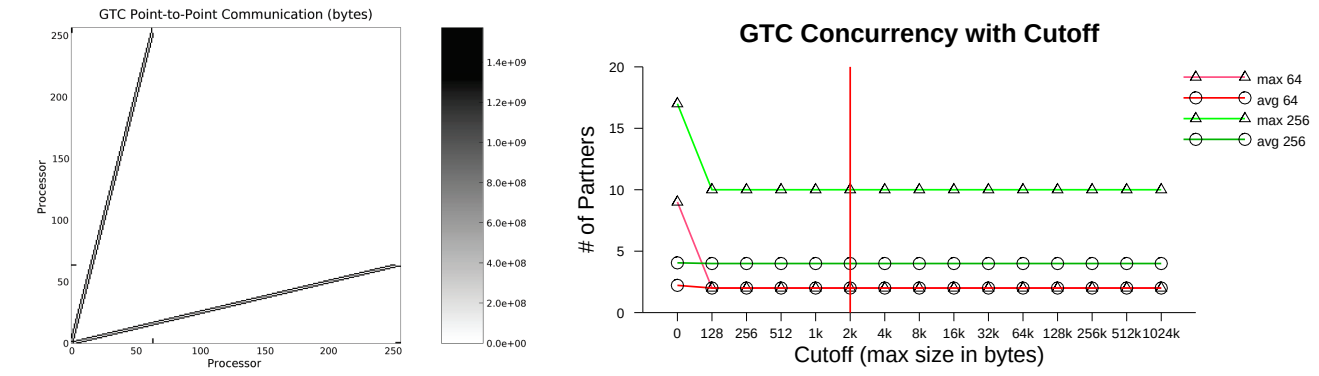
Figure 3(f) shows the complex structure of communication of the PMEMD particle mesh ewald calculation. Here the maximum and average TDC is equal to P and the degree of connectivity is a function of concurrency. For the spatial decomposition used in this algorithm, the communication intensity between two tasks drops 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*



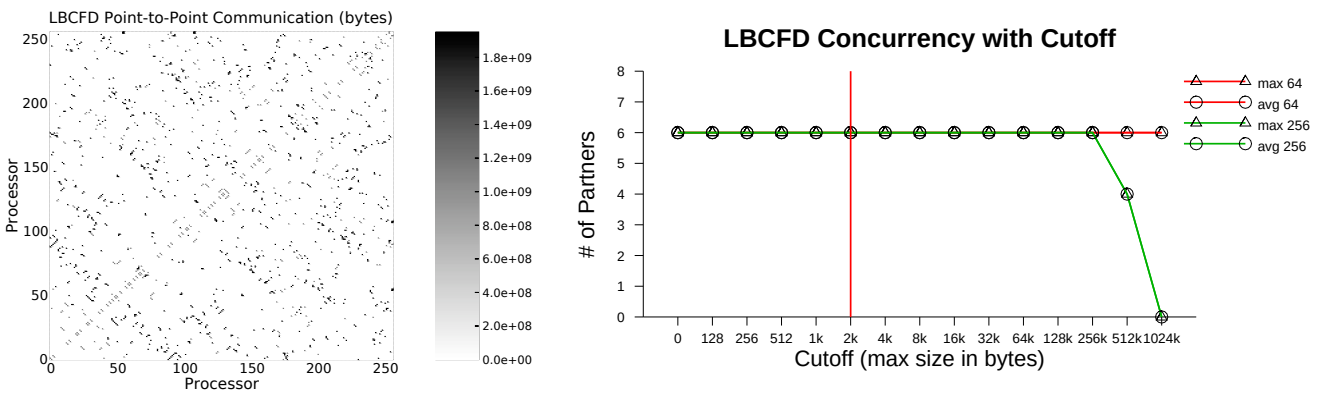
(a)



(b)

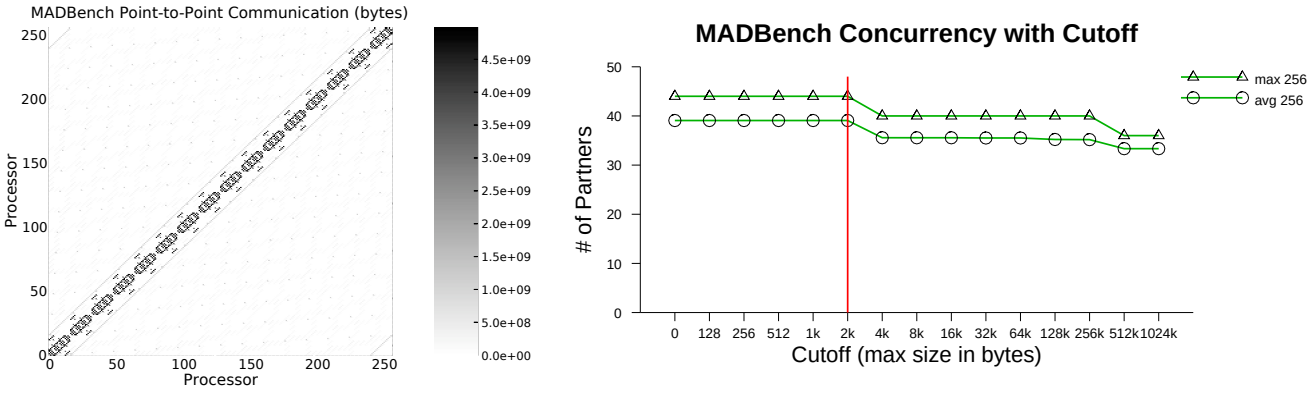


(c)

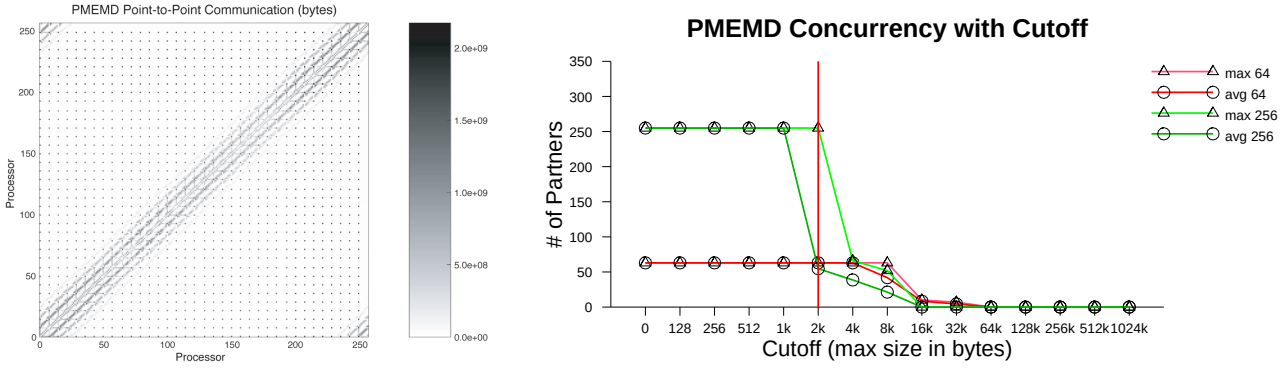


(d)

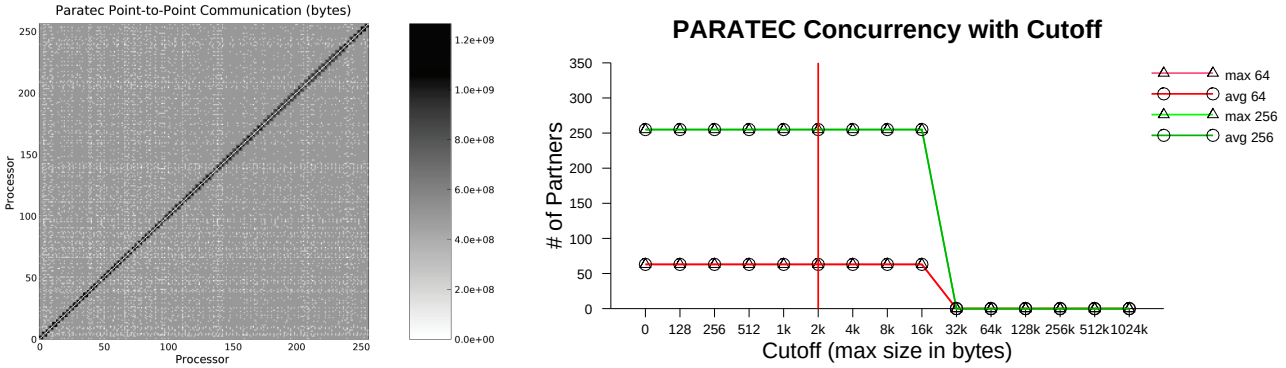
Fig. 3. Topological connectivity of (a) BBeam3D, (b) Cactus, (c) GTC and (d) LBCFD, showing (left) volume of communication at P=256 and (right) effect of 2KB thresholding on TDC for P=64,256.



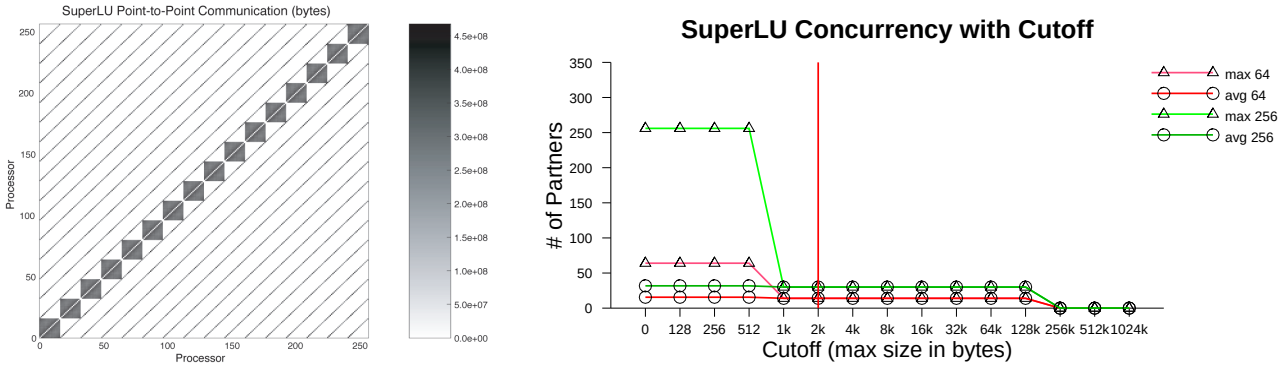
(e)



(f)



(g)



(h)

Fig. 3. (Continued) Topological connectivity of (e) MADbench, (f) PMEMD, (g) PARATEC, and (h) SuperLU, showing (left) volume of communication at P=256 and (right) effect of 2KB thresholding on TDC for P=64,256.

connectivity to 55, even though the maximum TDC remains at 256. This application class thus exhibits a large disparity between the maximum and average TDC.

Figure 3(g) 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 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 an FCN configuration provides.

Finally, Figure 3(h) shows the connectivity and TDC for SuperLU. 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 3(h) (see [16]).

We now analyze our measured topological connectivity in the context of interconnect requirements.

E. Communication Connectivity Analysis

Based on the topological connectivities of our applications, we now categorize our codes 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 tori or hypercubes. 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, but require more connectivity than provided by tori or mesh networks. Of these codes, if the *maximum* TDC is bounded by a low degree, then bounded-degree approaches will be sufficient, as long as the degree is high enough to allow an embedding of the application into the interconnect. For applications where the *average* TDC is bounded by a small number, while the *maximum* TDC is arbitrarily large (*case iii*), using a simple regular bounded-degree interconnect is over-provisioning the network, since many processors will have many more links than they need. 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 required for such applications.

F. 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 four codes exhibiting the most regularity in their communication exchanges: Cactus, GTC, LBCFD, and MADbench. 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 (i.e. consistent with *case i*). LBCFD and MADbench also display a low degree of connectivity; however, while their communication pattern is isotropic, their respective structure is not isomorphic to a regular mesh, thereby corresponding to *case ii* classification. 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 a fixed mesh/torus topology would be not well suited for this class of computation.

BBeam3d, SuperLU and PMEMD all 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*). Thus a regular mesh or torus would be inappropriate for this class of computation, while an FCN remains underutilized.

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. PARATEC's large global communication requirements can only be effectively provisioned with an FCN network.

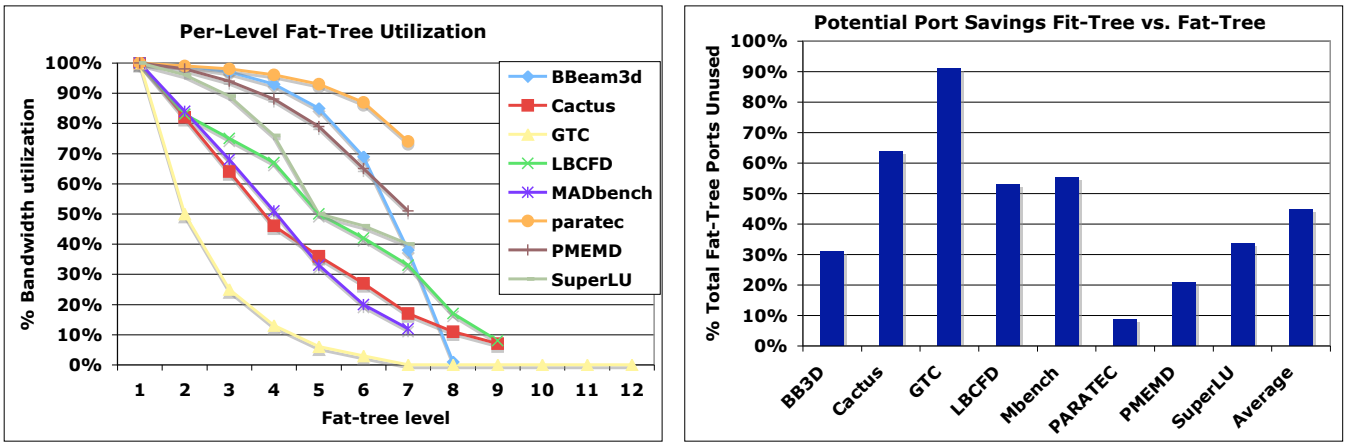
In summary, only one of the eight codes studied (Cactus) offered a communication pattern that maps isomorphically to a 3D mesh network topology (*case i*). This indicates that mesh/torus interconnects may be insufficient for a diverse scientific workloads. Additionally, only PARATEC fully utilizes the FCN at large scales (*case iv*); thereby undercutting the motivation for using FCNs across a broad range of computational domains. The under-utilization of FCN for our codes can be clearly seen in the last column of Table III. Thus, for a wide range of applications (cases *ii* and *iii*), we believe there is space to explore alternative interconnect architectures that contain fewer switch ports than a fat-tree but greater connectivity than mesh/tori networks; such interconnects are explored further in Section III.

III. Fit-tree Interconnect Designs

Given the topological connectivities of our applications in Section II, we now explore optimizing the interconnection network design. We start with an analysis of fat-tree interconnects and develop the concept of a *fit-tree*, which allows comparable performance at a fraction of the interconnect resources.

A. Fat-Tree Resource Requirements

Conceptually, a fat-tree is a k-ary tree with processors on the bottom-most level, where the thicknesses (capacities) of the edges increase at higher levels of the tree. Here, k is defined by the $k \times k$ switch block size used to implement the network. That is, 2×2



(a) (b)

Fig. 4. (a) Underutilization of fat-tree bandwidth for the examined application suite. Level 1 refers to the bottom of the tree closest to the processors. The vertical axis represents percentage of the bandwidth utilized at each level. (b) The potential savings in the number of required ports (cost) for an ideal fit-tree compared with the fat-tree approach.

switches will yield a binary tree, 4×4 switches yield a 4-ary tree, etc. In a conventional fat-tree, the total bandwidth is constant for each level of the tree; thus the thickness of an edge at level $i+1$ is k times the thickness at level i. Messages can travel up the tree and back down to traverse from a processor to any other processor without being constrained by bandwidth limitations; this structure can be thought of as a "folded" Benes network [8].

We now quantify the relation between the number of levels, number of processors and the number of switch boxes. A fat-tree with L levels built with $k \times k$ switches can have up to $2k^L$ processors, since the number of nodes is multiplied by k at each level from the root down to the tree's bottom. Conversely, the depth of a fat-tree for P processors built with $k \times k$ switches is $\log_k P - \log_k 2$. The corrective term of 2 is due to the root level of the fat-tree, where all switch ports are available for the lower level, unlike intermediate levels, where half of the ports are used for connections with the higher level. Since the total bandwidth at each level is constant, so are the number of switch ports per level. As a result, the bottom level of the fat-tree, which connects the processors to the network, requires $\lceil \frac{P}{k} \rceil$ switches; thus a fat-tree with L levels built with $k \times k$ switches requires $\frac{LP}{k} = 2Lk^{L-1}$ switches. Conversely, building a fat tree for P processors requires $(\log_k P - \log_k 2)\lceil \frac{P}{k} \rceil$ of $k \times k$ switches.

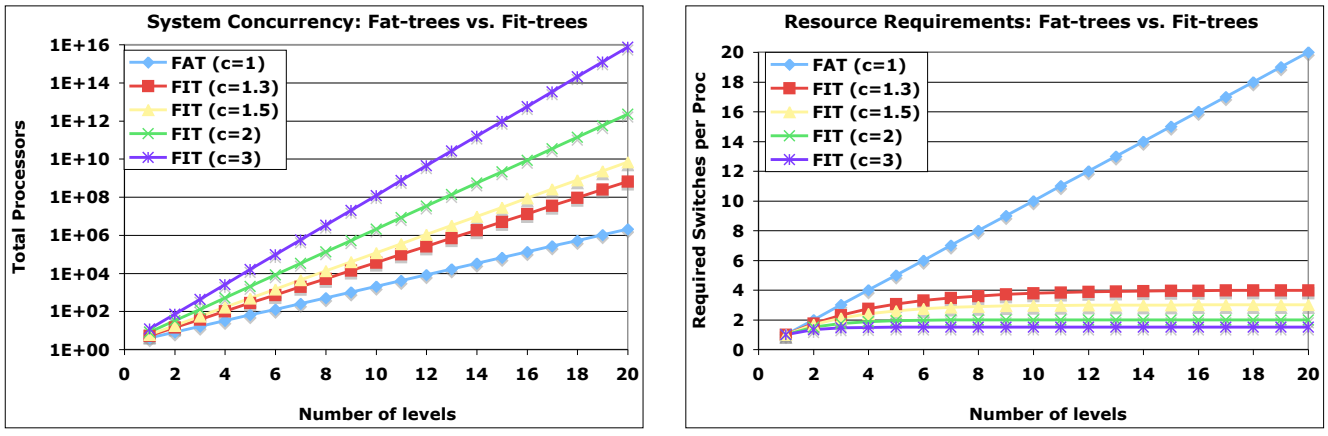
Constructing fat-trees where the network bandwidth is preserved at all levels is extremely challenging for thousands of processors, and simply infeasible for petascale systems with tens or hundreds of thousands of processors. Besides the construction complexity, the performance of a fat-tree network degrades while the cost increases sharply with increasing processor count. From the performance perspective, as the depth of the tree increases with larger concurrencies, the number of hops per message increases, corresponding to larger message latencies. While latency due to the interconnection network may not be significant for small to medium number of processors, it can dominate the message transmission cost at very high concurrencies. Additionally, the cost of a fat-tree grows superlinearly with larger parallel systems, since fat-tree construction depends on the number of switching blocks as well as the number of cables employed. These factors eliminate fat-tree topologies as a practical interconnection paradigm for next generation supercomputers.

B. Fat-Tree Utilization

In this section, we analyze what fraction of the available fat-tree bandwidth is utilized by our studied applications. In previous work [14], we employed two methods to assign tasks to processors: one that assigns processors based on the natural ordering of the tasks, and a second method that aims to minimize the average number of hops for each message using a heuristic based on graph partitioning. For the analysis here, we assign tasks to processors using the heuristic methodology.

To create an instance of communication, we use the application communication patterns presented in Section II. For a given instance, a processor sends a message to one of its communicating partners chosen at random. As an approximation of the communication overhead, we create $10P$ instances of communication for each application, and route the messages on the interconnect; recording how many messages reach each given level of the fat-tree. Using this estimation strategy, we simulate the behavior of each application to determine the communication load on the network.

Figure 4(a) displays the results for bandwidth utilization of a fat-tree built with 2×2 switches. In this figure, the horizontal axis corresponds to the the fit-tree level starting with the leaf nodes (i.e. the processors). The vertical axis correspond to bandwidth utilization, which we compute by counting the number of messages that reach a given level, and comparing this number with the level's total available bandwidth (P for a fat-tree). The results show that the bandwidth utilization drops sharply as the tree level increases. For GTC, this number drops to 0 at level seven, indicating that the highest six levels of the fat-tree are not used at all. A similar trend is seen in all examined applications. Even for PARATEC, which uses all-to-all-communication in its FFT, bandwidth utilization goes down to 74% at the top level even though P is only 256 processors. These results clearly show that fat-tree bandwidth is underutilized for most applications, especially for those that can scale up to thousands of processors. In the next section, we will use this observation to propose an alternative interconnection topology that is dynamically reconfigurable.



(a) (b)

Fig. 5. Comparison of fat-tree and fit-tree scalabilities in terms of (a) potential system concurrency for a fixed number of tree levels and (b) the required number of switches per processor.

C. Fit-Tree Approach

The motivation for the *fit-tree* topology comes from our observation that the available bandwidth of a fat-tree is not utilized at all levels — especially the higher ones — of a fat-tree network for many scientific computing applications. Exploiting this observation, we propose the fit-tree topology, which is an improvement on fat-trees, providing better scalability in terms of both performance and cost.

Consider an intermediate node in a fat-tree that is the root of a sub-tree of P' processors. In a conventional fat-tree, this node corresponds to a $P' \times P'$ switch, whose ports are assigned so that P' of them are connected to the lower level and P' are connected to the higher level. This provides P' different communication channels for P' processors. Since some of this bandwidth is redundant (Section III-A), eliminating a portion of the connections within the higher level will not degrade performance. Note that although this network design optimization decreases cabling requirements, it does not improve switch costs or overall performance.

In the proposed fit-tree design, the number of ports used for connections to the higher levels of the tree is less than the number of ports used for connections to the lower levels. This approach leverages otherwise unutilized switch ports to increase the number of connected nodes at lower tree levels, allowing an increase in the number of processors rooted at a node (at the same level). Thus our fit-tree design has a $c:1$ ratio between the number of ports that go down and up (respectively) for each intermediate level, where we call $c > 1$ the *fitness ratio*. Conversely, a conventional fat-tree has a $1:1$ ($c = 1$) ratio between bandwidth up and down at each intermediate level.

The fit-tree methodology enables building larger systems for a fixed number of levels in the interconnect tree. A direct comparison with fat-trees can be made in two ways. If we assume that total bandwidth is preserved at each level of the tree, a fat-tree is built using k children per node. However, a fit-tree's node has ck children, where c is the fitness ratio. This translates to an exponential advantage in the number of processors the interconnect can support, as a fit-tree of L levels built with $k \times k$ switches and a $c:1$ ratio will contain $2(ck)^L$ processors as opposed to $2k^L$ for a fat-tree. Conversely, the depth of a fit-tree for a fixed number of processors P built with $k \times k$ switches and a $c:1$ ratio is $\log_{ck} P - \log_{ck} 2$. This reduced number of levels for a fixed number of processors translates to a reduction in switch count. Figure 5(a) shows the advantage of the fit-tree approach for potential system concurrency given a fixed number of levels.

Alternatively, we can consider fixing the number of tree levels while decreasing the total fit-tree bandwidth at the higher levels. For a fat-tree the total bandwidth provisioned is computed as:

$$\sum_{i=1}^{L-1} \frac{P}{kc^{i-1}} = \frac{P(\frac{1}{c^L} - 1)}{k(\frac{1}{c} - 1)} < \frac{Pc}{k(c-1)}$$

In a fit-tree, however, the bandwidth can be reduced by c at each level of the tree. It is worth noting that the total bandwidth, and thus the number of switches, scales linearly with P, which provides perfect scalability for fit-trees. For example, given $c = 2$, the total number of required switches will be no more than two times the number of switches at the first level. Figure 5(b) highlights the fit-tree advantage, by showing a comparison of the required number of switch components when between fat- and fit-trees using varying fitness ratios.

In practice it is possible to build hybrid topologies, where each node has more than k children, thus allowing the bandwidth to reduce gradually. This allows fit-tree designers to trade off between cost savings and performance. In Section IV we propose a hybrid optical/electrical interconnect solution that would allow fit-tree designs that could be dynamically reconfigured to the requirements of the underlying application. We now examine the potential advantage of a fit-tree architecture for our evaluated set of scientific codes.

D. Fit-Tree Evaluation

In the previous section, we showed how fit-trees can significantly improve the cost and scalability of fat-trees, while preserving performance. The critical question is therefore determining the appropriate fitness ratio for a given computation. In this section, we investigate the fitness ratio requirements of our studied applications. For these experiments, we use the same experimental setup as in Section III-B, and compute the fitness ratio of level $i+1$ as the ratio between the bandwidth utilization at level $i+1$ and i respectively. To reduce the effect of outliers, we consider 4 to be the highest allowable fitness ratio.

TABLE IV
FITNESS RATIOS FOR (TOP) EACH APPLICATIONS ACROSS ALL LEVELS AND (BOTTOM) EACH LEVEL ACROSS ALL APPLICATIONS

Code	BB3D	Cactus	GTC	LBCFD	MADbench	PARATEC	PMEMD	SuperLU
Minimum	1.01	1.22	1.92	1.11	1.19	1.01	1.02	1.04
Average	1.60	1.40	3.01	1.41	1.44	1.05	1.12	1.17
Maximum	4.00	1.59	4.00	1.94	1.67	1.18	1.27	1.52
Median	1.09	1.36	3.00	1.24	1.44	1.03	1.09	1.12

Level	1	2	3	4	5	6	7	8
Minimum	1.01	1.01	1.02	1.03	1.07	1.15	1.54	1.57
Average	1.21	1.22	1.26	1.39	1.35	1.62	2.87	2.57
Maximum	2.00	2.00	1.92	2.17	2.00	3.00	4.00	4.00
Median	1.12	1.09	1.15	1.31	1.22	1.43	2.97	2.13

Table IV(top) presents the fitness ratios of each examined application. For clarity of presentation, we only show the minimum, average, maximum, and median across all fit-tree levels. Results show that (as expected) fitness ratios are higher for applications with sparse communication: GTC, BB3D, LBCFD, MADbench, Cactus, and SuperLU. Note that these applications are known to exhibit better scalability compared to communication-intensive computations such as PARATEC and PMEMD. However, it is remarkable that even PARATEC, which require global 3D FFTs, has a fitness ratio of 1.18 at its top level.

Table IV(bottom) presents fitness ratios for each level of the fit-tree across applications. Again for clarity of presentation, we only display the minimum, average, maximum, and median values. Results show that, while the fitness ratios are low at the lowest levels, they increase with increasing fit-tree levels. This is expected, as the number of nodes rooted at a node is doubled at each level of the fat-tree, creating room for locality where the percentage of local communication increases.

Based on Table IV it is difficult to decide on a single "ideal" fitness ratio, but the data show strong quantitative support for the fit-tree concept. After all, even the minimum fitness ratio at level six is 1.15. It is worth repeating that our main motivation is interconnect designs for petascale and exascale systems, which are expected to have hundreds of thousands of processors. Therefore, even a fitness ratio of 1.15 will translate to enormous savings in costs and improvements in performance as displayed in Figure 5. The potential savings in switch ports versus a fat-tree for our examined applications is shown in Figure 4(b). Even for the moderate concurrency levels explored here, the hybrid fit-tree approach can reduce the port count requirements by up to 44% (on average).

In the next section, we propose a hardware solution for dynamically constructing fit-trees appropriate for each application, thus building the best-performing interconnect at the lowest cost in terms of switch ports.

IV. HYBRID SWITCH ARCHITECTURE

As we move towards petaflops systems with tens (or hundreds) of thousands of processors, building fully-connected interconnects quickly becomes infeasible due to the superlinear cost of this network design approach. Moreover, our analysis of large-scale scientific applications in Section II quantified the under-utilization of FCNs for a wide array of numerical methods. In Section III, we presented our fit-tree methodology and demonstrated that it has the potential to improve network performance while reducing component costs.

In this section we present a methodology for dynamically creating fit-trees with variable fitness ratios as well as arbitrary network configurations. Our proposed approach, called HFAST (Hybrid Flexibly Assignable Switch Topology), uses passive/circuit switches to dynamically provision active/packet switch blocks — allowing the customization of interconnect resources for application-specific requirements. We begin by looking at recent trends in the high-speed wide area networking community, which has developed cost-effective solutions to similar challenges.

A. 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 end-points 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-electro-mechanical 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.

B. 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 [9] 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 multiplexing 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 multi-adapter 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) [11], 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 [10]. 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 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.

C. HFAST: Hybrid Flexibly Assignable Switch Topology

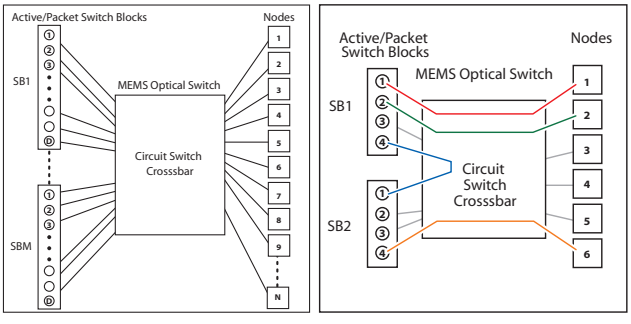


Fig. 6. General layout of HFAST (left) and example configuration for 6 nodes and active switch blocks of size 4 (right).

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 necessary 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 6 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 6, 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.

The HFAST approach holds a clear advantage to statically built interconnects, since additional packet switch resources can dynamically be assigned to the subset of nodes with higher communication requirements. HFAST allows the effective utilization of interconnect resources for the specific requirements of the underlying scientific applications. This methodology can therefore satisfy the topological connectivity of applications categorized in *cases i-iii* (defined in Section II-E). Additionally, HFAST could be used to dynamically create fit-trees with static or variable fitness rations. Finally, the HFAST strategy could even iteratively reconfigure the interconnect between communication phases of a dynamically adapting application [14]. Future work will continue to explore the potential of the HFAST in the context of demanding scientific applications.

V. Summary and Conclusions

There is a crisis looming in parallel computing driven by rapidly increasing concurrency and the non-linear scaling of switch costs. It is therefore imperative to investigate interconnect alternatives, to ensure that future HPC systems can cost-effectively support the communication requirements of ultra-scale applications across a broad range of scientific disciplines. In this work we present several research thrusts to address these concerns: scientific application communication analysis, fit-tree networks, and HFAST reconfigurable interconnects. Although these three components closely complement one another, each investigation could also be considered as an independent research contribution, which advances the state-of-the art in its respective area.

First, we presented one of the broadest studies to date of high-end communication requirements, across a broad spectrum of important scientific disciplines, whose computational methods include: finite-difference, lattice-bolzmann, particle in cell, sparse linear algebra, particle mesh ewald, and FFT-based solvers. Analysis of these data show that most applications do not utilize the full connectivity of traditional fat-tree interconnect solutions. Based on these observations, we proposed the a novel network called the fit-tree architecture. Our work reveals that fit-trees can significantly improve the cost and scalability of fat-trees, while preserving performance. Finally, we propose the HFAST infrastructure, which combines passive and active switch technology to create dynamically reconfigurable network topologies, and could be used to custom tailor fit-tree configurations for specific application requirements.

In future work we plan to expand the scope of both the applications profiled and the data collected through the IPM profiling. 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, including the analysis of full chronological communication traces. By studying the time dependence of communication topology one could expose opportunities to reconfigure an HFAST interconnect within a dynamically evolving computation. Finally, we will continue our exploration of fit-tree solutions in the context of ultra-scale scientific computations.

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] K. Asanovic, R. Bodik, B. Catanzaro, et al. The landscape of parallel computing research: A view from Berkeley. Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, December 2006.

[4] Julian Borrill, Jonathan Carter, Leonid Oliker, David Skinner, and R. Biswas. Integrated performance monitoring of a cosmology application on leading hec platforms. In *Proceedings of the International Conference on Parallel Processing (ICPP), to appear*, 2005.

[5] Cactus Homepage. `http://www.cactuscode.org`, 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] W. J. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 2004.

[9] 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.

[10] 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.

[11] 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.

[12] PMEMD Homepage. `http://amber.scripps.edu/pmemd-get.html`.

[13] IPM Homepage. `http://www.nersc.gov/projects/ipm`, 2005.

[14] S. Kamil, A. Pinar, D. Gunter, M. Lijewski, L. Oliker, and J. Shalf. Reconfigurable hybrid interconnection for static and dynamic scientific applications. In *ACM International Conference on Computing Frontiers*, 2007.

[15] W. W. Lee. Gyrokinetic particle simulation model. *J. Comp. Phys.*, 72, 1987.

[16] 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.

[17] 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.

[18] L. Oliker, A. Canning, J. Carter, et al. Scientific application performance on candidate petascale platforms. In *Proc. IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, Long Beach, CA, Mar 26-30, 2007.

[19] J. Qiang, M. Furman, and R. Ryne. A parallel particle-in-cell model for beam-beam interactions in high energy ring collide rs. *J. Comp. Phys.*, 198, 2004.

[20] 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.

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

[22] 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.

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