Reconfigurable Hybrid Interconnection for Static and Dynamic Scientific Applications

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

ABSTRACT

As we enter the era of peta-scale computing, system architects must plan for machines composed of tens or even hundreds of thousands of processors. Although fully connected networks such as fat-tree configurations currently dominate HPC interconnect designs, such approaches are inadequate for ultra-scale concurrencies due to the superlinear growth of component costs. Traditional low-degree interconnect topologies, such as 3D tori, have reemerged as a competitive solution due to the linear scaling of system components relative to the node count; however, such networks are poorly suited for the requirements of many scientific applications at extreme concurrencies. To address these limitations, we propose HFAST, a hybrid switch architecture that uses circuit switches to dynamically reconfigure lower-degree interconnects to suit the topological requirements of a given scientific application. This work presents several new research contributions. We develop an optimization strategy for HFAST mappings and demonstrate that efficiency gains can be attained across a broad range of static numerical computations. Additionally, we conduct an extensive analysis of the communication characteristics of a dynamically adapting mesh calculation and show that the HFAST approach can achieve significant advantages, even when compared with traditional fat-tree configurations. Overall results point to the promising potential of utilizing hybrid reconfigurable networks to interconnect future peta-scale architectures, for both static and dynamically adapting applications.

CF'07, May 7–9, 2007, Ischia, Italy.

1. INTRODUCTION

For over two decades, the performance of commodity microprocessor-based supercomputing systems has been reliant on clock frequency improvements from the scaling of microchip features due to Moore's Law. However, since the introduction of 90nm chip technology, heat density and changes in the dominant physical properties of silicon have moderated the pace of clock frequency improvements. As a result, the industry is increasingly reliant on unprecedented degrees of parallelism to keep pace with the ever-growing demand of high-end computing (HEC) capabilities. Thus, in the imminent era of petaflop systems, computational platforms are expected to be comprised of tens or even hundreds of thousands of processors.

In today's supercomputing landscape, fully-connected networks, such as crossbar and fat-tree configurations, dominate HEC interconnect designs [21]. Unfortunately, the component cost of these network topologies scale superlinearly with the number of nodes in the system — making these designs impractical (if not impossible) at the ultra-scale level. Consequently, HEC system architects are increasingly considering lower degree topological networks, such as 2D and 3D tori, that scale linearly in cost relative to system scale. However, adoption of networks with a lower topological degree of connectivity requires renewed attention to application process placement, as a topology-oblivious process mapping may result in significant performance degradation. This kind of topological mismatch may be mitigated by sophisticated task migration and job-packing by the batch system, but such migration impacts overall system efficiency and is often too complex to implement on modern parallel systems. Additionally, a growing fraction of applications exhibit dynamically adapting computational structures, due in part to the wider adoptance of multi-scale simulation methodologies. These algorithms are characterized by evolving communication requirements, which change dynamically at runtime. Statically mapping this class of codes onto a lower degree interconnect topology may result in hopelessly inefficient performance at large scale.

In response to these growing concerns, we propose HFAST, a Hybrid Flexibly Adaptable Switch Topology (HFAST) [20] that employs optical circuit switches (Layer-1) to dynamically provision packet switch blocks (Layer-2) at runtime. This work makes several important contributions. First, we examine the topological communication characteristics of state-of-the-art scientific computations across a broad range of domains, including a comprehensive exploration of an adaptive mesh refinement (AMR) calculation. Our analysis represents the most detailed study of the evolving communication requirements for a dynamic AMR simulation to date. Next, we present a optimization methodology for mapping application processes onto the HFAST architecture, which minimizes the number of message hops across the interconnect. Quantitative results on realistic, large-scale applications demonstrate that, even for modest levels of concurrency, the HFAST approach can be used to design an interconnect network as effective as a fat-tree, with fewer switching resource requirements. Overall results point to the promising potential of utilizing hybrid reconfigurable networks to interconnect future peta-scale architectures, for both static and dynamically adapting applications.

2. HYBRID SWITCH ARCHITECTURE

As the HEC community moves towards peta-scale architectures comprised of tens or hundreds of thousands of processors, the industry will be hard-pressed to continue building cost-effective fully connected networks. 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 found that *lambda-switching* — hybrid interconnects composed of circuit switches together with packet switches — presents a cost-effective solution to a similar set of problems.

2.1 Circuit Switch Technology

Packet switches, such as Ethernet, Infiniband, and Myrinet, are the most commonly used interconnect solution for large-scale parallel computing platforms; unfortunately, these network technologies all have relatively high cost per-port. 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. Recently, 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 low-voltage differential signaling over copper wire. However, optical links require a transceiver that converts from optical to electrical signals so the silicon circuits can perform their switching decisions. These Optical Electrical Optical (OEO) conversions further add to the cost, latency, and power consumption of switches. Fully-optical packet switches (i.e. 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, such as in the recent OSMOSIS project, which was a DARPA funded collaboration between IBM and Corning to develop a fully optical packet switch [1].

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. They also eliminate the need for OEO conversion through optical transceivers, which are costly and consume considerable power. 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) OEO 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 their prices are dropping rapidly as the market for the technology grows larger and more competitive. Our work examines leveraging MEMS-based circuit-switch technology, in the context of ultra-scale parallel computing platforms.

Circuit switches have long been recognized as a cost-effective alternative to packet switches, but it has proven difficult to exploit this technology in cluster interconnects because the hard-wired circuit switches are oblivious to message/packet boundaries. Although it is possible to reconfigure the optical path through the circuit switch, the overhead is on the order of milliseconds and one must be certain that no message traffic is propagating through the light path when the reconfiguration occurs. The overhead of ensuring the interconnect has achieved a quiescent state undercuts the advantages of a pure circuit-switched approach. In comparison, a packet-switched network can trivially multiplex and demultiplex messages destined for multiple hosts without requiring any configuration changes. Our HFAST approach overcomes the limitations of pure circuit-switched approaches by applying packet switching resources judiciously to maintain the cost advantages of circuit switching. Detailed discussions of these technologies as well as their implementations and tradeoffs can be found in [3, 10, 20].

Our proposed hybrid interconnect architecture has the most similarity to the Interconnection Cached Network (ICN) [10] approach. However, whereas the ICN would require task migration to preserve optimal graph embedding, the HFAST approach allows tasks to remain in-situ as the interconnect adapts to the evolving job requirements. This feature is particularly advantageous for the adaptive applications that are the focus of this paper.

2.2 HFAST: Hybrid Flexibly Assignable Switch Topology

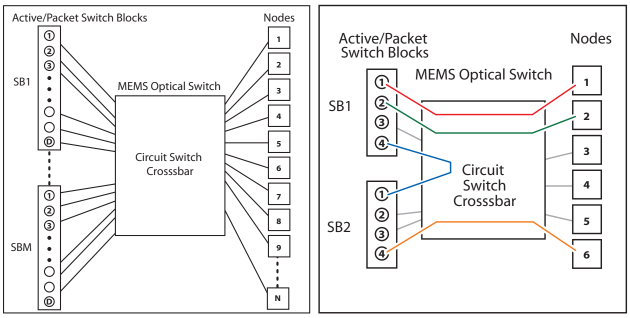


Figure 1: General layout of HFAST (left) and example configuration for 6 nodes and active switch blocks of size 4 (right). In this example, node 1 can communicate with node 2 by sending a message through the circuit switch (red) in switch block 1 (SB1), and back again through the circuit switch (green) to node 2.

To address the limitations of pure packet- or circuit-based switches, we propose the HFAST interconnect architecture, composed of a hybrid mix of (Layer-1) passive/circuit switches that 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. Figure 1 shows how the fully-connected passive circuit switches are placed between the nodes and the active (packet) switches. The communication topology is thus determined by the provisioning of circuits between nodes and active switches, and between blocks of active switches. This allows active packet switch blocks to be used as a flexibly assignable pool of resources that support adaptable formation of communication topologies without any job placement requirements. Using less-expensive circuit switches, the packet switch blocks could emulate many different interconnect topologies such as a 3D torus or densely-packed 3D mesh. Codes that exhibit non-uniform degree of communication can be supported by assigning additional packet switching resources to the processes with greater communication demands. Additionally, topology can be incrementally adjusted at runtime to match the requirements of codes with dynamically adapting communication patterns.

Furthermore, there are a number of research avenues to leverage HFAST resources in a large-scale environment. One approach is to study the 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 may be capable of automatically inserting the necessary synchronization points in communication-free phases of the computation, allowing the circuit switches time to reconfigure since the Layer-1 switches do not otherwise respect packet boundaries for in-flight messages.

Note that 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. Unlike the ICN and OptIPuter [8], 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.

In the HFAST context, the circuit switch technology is most effective when a dedicated circuit is used primarily for the bandwidth-bound messages. Our recent measurements on a number of different existing interconnect architectures saw minimal improvements in performance of applications that were dominated by the smaller latency-bound messages [20] (in spite of the theoretical ability of RDMA to combine and pipeline small messages). Therefore, we focus on messages that are larger than the bandwidth-delay product, defined as the minimum message size that can theoretically saturate the link — conservatively set to 4KB. Our analysis filters out the latency-bound messages with the assumption that they can be carried either as transit traffic that requires several hops through the HFAST interconnect topology to reach its destination, or routed to a secondary low-latency low-bandwidth interconnect that uses much lower cost components for handling collective communications with small payloads. An example of such a low-latency secondary network can be found in the BlueGene/L Tree network [7], which is designed to handle fast synchronization and collectives where the message payload is typically very small.

3. NON-ADAPTIVE SCIENTIFIC APPLICATIONS

To evaluate the potential effectiveness of HFAST, we must first develop an understanding of the communication requirements of scientific codes across a broad spectrum of parallel algorithms. Here we examine the behavior of six codes based on non-adaptive numerical methods, whose communication patterns remain mostly static throughout the course of the computation: Cactus, LBMHD, GTC, MAD-

Name	Lines	Discipline	Problem and Method	Structure
Cactus [6]	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
MADbench [5]	5,000	Cosmology	CMB Analysis via Newton-Raphson	Dense Matrix
ELBM3D [13]	3,000	Fluid Dynamics	Fluid Dynamics vi Lattice Bolzmann	Lattice/Grid
BeamBeam3D [18]	23,000	Particle Physics	Poisson's equation via Particle in Cell and FFT	Particle/Grid

Table 1: Overview of the static scientific applications examined in this work.

Name	P	% Communicating Partners	Average Messages/P
Cactus	1024	0.5%	12,018
LBMHD	256	4.6%	23,907
GTC	256	1.6%	5,595
MADbench	256	15.3%	14,122
ELBM3D	1024	0.6%	12,024
BeamBeam3D	1024	25.2%	8,952,000

Table 2: Communication intensity of static scientific applications examined in this work, showing percent of communicating partners and the average (per processor) number of exchanged messages.

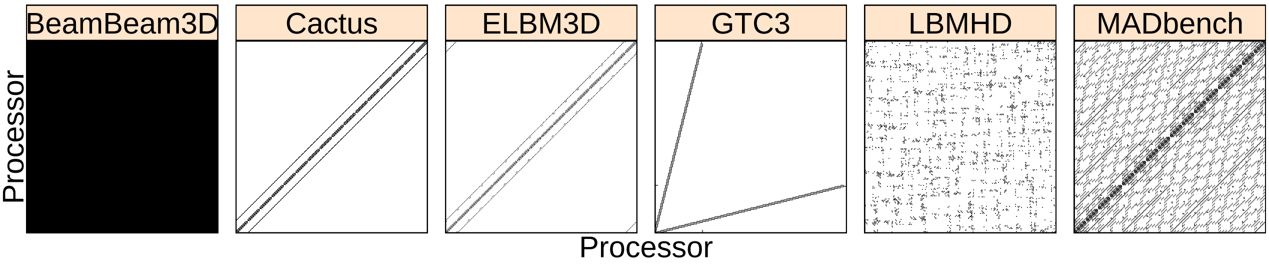


Figure 2: Communication topology of the static applications in our study (from left to right): BeamBeam3D (p=1024), Cactus (p=1024), GTC3 (p=256), ELBM3D (p=1024), LBMHD (p=256), and MADbench (p=256).

bench, ELB3D, and BeamBeam3D. These codes represent candidate ultra-scale applications that have the potential to fully utilize leadership-class computing systems. A brief overview is presented in Table 1; detailed descriptions of the algorithms and scientific impact of these codes can be found in [5, 6, 13–15, 18].

Figure 2 shows the communication topologies for the static applications in this study. The graphs provide the volume and pattern of message exchanges between all tasks as recorded by the IPM [12] profiling layer. The X and Y axes each represent the discrete range of 1 to P. If communication occurs between two given processors x and y, a point (x, y) is plotted on the plane; a darker color intensity represents larger data volume exchange between the two processors. Because we assume that switch links are bidirectional, the values at (x, y) and (y, x) are always identical. The percentage of communicating partners for each static application is quantified in Table 2, which shows the relative sparseness of almost all the static topology graphs: four of the six codes communicate with less then 5% of the processors involved in the computation. These codes thus reflect the large class of applications that vastly underutilize a fully connected network.

4. ADAPTIVE MESH REFINEMENT CALCULATION

In order to understand the potential of utilizing our hybrid reconfigurable interconnect in the context of dynamically adapting applications, we explore the communication details of an adaptive mesh refinement (AMR) calculation. Although the AMR methodology has been actively researched for over two decades [2], our work represents the first study to quantify the evolving communication requirements of this complex application.

AMR is a powerful technique that reduces the computational and memory resources required to solve otherwise intractable problems in computational science. The AMR strategy is to start with a problem on a relatively coarse grid and dynamically refine it in regions of scientific interest or where the coarse grid error is too high for proper

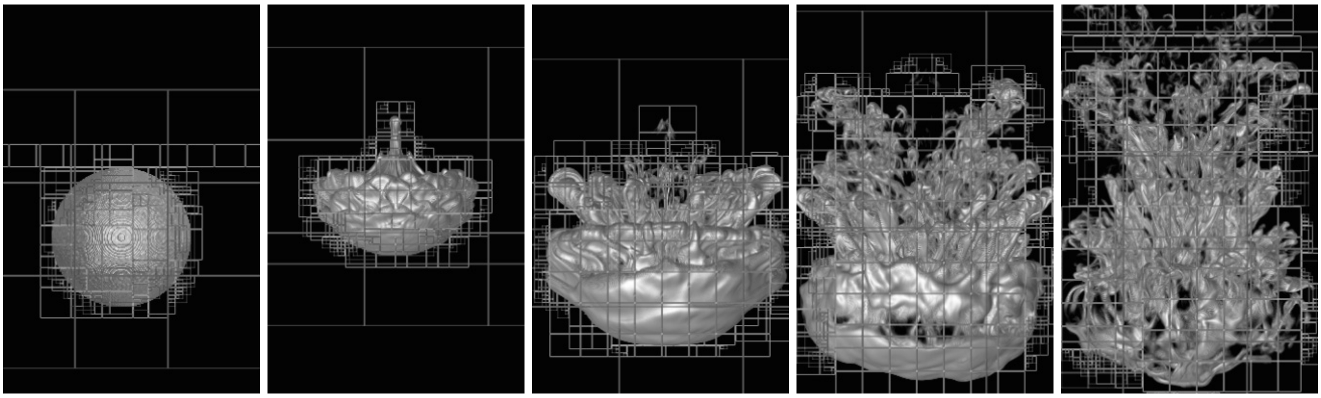


Figure 3: Deformation of Helium bubble as it passes through shock front in the AMR simulation.

numerical resolution. Not surprisingly, the software infrastructure necessary to dynamically manage the hierarchical grid framework tends to make AMR codes far more complicated than their uniform grid counterparts. Despite this complexity, it is generally believed that future multi-scale applications will increasingly rely on adaptive methods to study problems at unprecedented scale and resolution.

A key component of an AMR calculation is dynamic mesh regridding, which dynamically changes the grid hierarchy to accurately capture the physical phenonema of interest. Cells requiring enhanced resolution are identified and tagged using a specified error indicator, and then grouped into rectangular patches that sometimes contain a few cells that were not tagged for refinement. These rectangular patches are then subdivided to form the grids at the next level. This process is repeated until either the error tolerance criteria is satisfied or a specified maximum level of refinement is reached.

4.1 HyperCLaw Overview

Our work examines HyperCLaw, a hybrid C++/`Fortran` AMR code developed and maintained by CCSE at LBNL [9, 19] where it is frequently used to solve systems of hyperbolic conservation laws using a higher-order Godunov method. In HyperCLaw most of the communication overhead occurs in the FillPatch operation, which exihibits complicated and irregular communication patterns. FillPatch presents a very complex nonuniform but sparse communication pattern. Once it completes, a higher-order Godunov solver is applied to each resulting grid. This solver is compute-intensive, requiring upwards of a full second for the problems we ran in this study, during which time no interprocessor communication occurs. HFAST circuit-switch reconfiguration could therefore occur during this compute-only phase, to dynamically incorporate the evolving communication requirements of the AMR calculation.

4.2 Evolution of Communication Topology

The HyperCLaw problem examined in this work profiles a hyperbolic shock-tube calculation, where we model the interaction of a Mach 1.25 shock in air hitting a spherical bubble of helium. This case is analogous to one of the experiments described by Haas and Sturtevant [11]. The difference between the density of the helium and the surrounding air causes the shock to accelerate into and then dramatically deform the bubble. An example of the kind of calculation HyperCLaw performs is seen in Figure 3, along with an overlaid representation of the grids used.

For this paper, we examine a refinement of three levels (0, 1, 2), with 0 being the lowest (or base) level and 2 being the highest (or finest) level, where most of the computation time is spent. Three levels of refinement are typical for calculations of this kind. Note that the refinement is a factor of two in each of the three coordinate directions.

In Figure 4, we see the topology of message exchanges for communication at levels 1 and 2. Note that these two levels exhibit very different communication patterns. Because level 1 represents a coarser problem than that of level 2, there are far fewer grid cells (communicating processors) at level 1 than at level 2. At level 2, the increase in the number of 3D grid cells causes each processor to exchange messages with many more cells. This results in a much denser communication topology since each 3D grid cell communicates with up to six others.

The structure of the AMR calculation provides an opportunity for HFAST to adapt its configuration to the evolving topology. The entire calculation is divided into timesteps, each containing distinct communication and computation phases separated by a regridding operation that takes on the order of hundreds of milliseconds. During this period, the optical switches can be reconfigured to take advantage of changes in the most significant communicating partners.

In order to justify the use of a lower degree interconnect topology, and thus to benefit from the HFAST approach, the

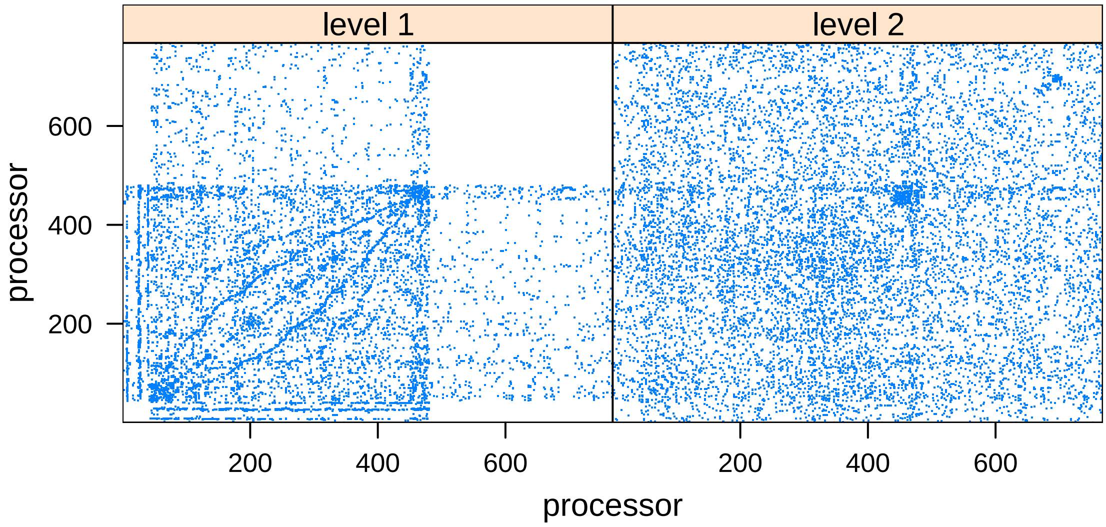


Figure 4: Topology of message exchange for HyperCLaw AMR levels 1 and 2.

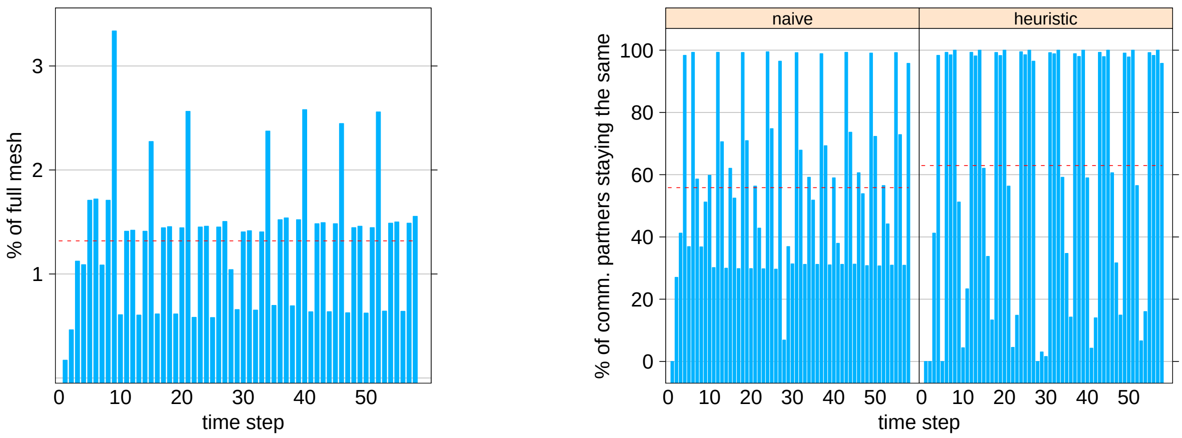
proportion of communicating partners (for messages over the bandwidth-delay product) of HyperCLaw must be significantly less than the number of nodes, P. Recall that the HFAST approach utilizes circuit switch technology for only bandwidth-bound messages, while latency-bound communications are routed to a secondary, low-bandwidth interconnect. We thus remove all messages whose sizes are less than 4KB from our analysis, as these message sizes are small enough that the messages are not bandwidth-bound on most modern interconnect technologies.

Another condition necessary to make HFAST a viable option for this dynamic calculation is the requirement that the set of communicating partners not change unpredictably and sharply at each time step. Otherwise, it would be impossible to appropriately reconfigure the optical switches *a priori* for a given time step.

Figure 5(a) shows the fraction of communicating partners — represented as the percentage of the possible partners each processor communicates with — for given AMR timesteps of our HyperCLaw simulation (run at $P = 768$). We also present Figure 5(b) which shows the percentage of communicating partners that did not change, using both the naïve approach based on the previous timestep and a simple heuristic strategy based on the previous timestep at the same AMR grid level. Observe that the heuristic approach significantly improves the percentage of communicating partners that do not change across remapping phases. This optimization will be further explored in the processor alocation algorithm of Section 5.

Based on Figure 5(a) we observe that no timestep exhibits a communication pattern that requires more than 4% of the available pathways through a fully connected network. While such a pattern is clearly not isomorphic to a mesh or torus topology, it is well matched to the bisection bandwidth offered by a lower-degree interconnect, provided the interconnect topology can be adapted to the application's communication pattern. We thus argue that the adaptation offered by HFAST enables a lower degree network to conform to the application's complex communication topology, whereas a lower-degree interconnect with a fixed topology, such as a torus or mesh, would not be well suited to the complex communication pattern. But this observation holds for a snapshot of the communication topology requirements in time. The true situation is far more complex as this topology evolves over time in a data-dependent fashion as the simulation progresses.

In order to understand the time-evolution of the communication topology, we focus on the incremental changes in the set of communicating partners as the simulation progresses. To understand this analysis, it is necessary to know that the pattern of computation for the AMR grid levels is a level 0 timestep, within which are two level 1 timesteps, within which are two level 2 timesteps. We exclude data from the level 0 timesteps. As a result, our analysis of Figure 5(b) shows a six-timestep pattern: $[0], 1, 2, 2, 1, 2, 2$. Careful reading of the data shows that the changes in communicating partners also follows a six-step pattern, which is most evident in the patterns of taller bars (high-percent of the same partners). A slight variation occurs at steps 25-27 that is due to the boundary between a checkpoint/restart of the code at that point. Overall, the heuristic of using the same AMR level for the previous time step is an improve-



(a)

(b)

Figure 5: HyperCLaw communication for refinement levels 1 and 2, showing (a) the percentage of communicating partners for messages greater than 4KB, and (b) the percent of these communicating partners that stay the same between AMR timesteps. The dashed line indicates the average value.

ment to the consistency of communicating partners, and the data strongly suggests that an algorithm that takes advantage of the six-step pattern could do even better. Detailed verification of this conjecture will be the subject of future work. From these results we conclude that it is feasible to accurately predict how to remap the underlying optical switches based on a snapshot of the current communication topology.

It is important to note that there are a wide variety of AMR algorithms and implementations, and our observations do not necessarily apply to all AMR simulations. However, this analysis does show the applicability of the HFAST approach to similar AMR calculations as well as dynamic applications in general.

5. OPTIMIZATION OF THE INTERCONNECT TOPOLOGY

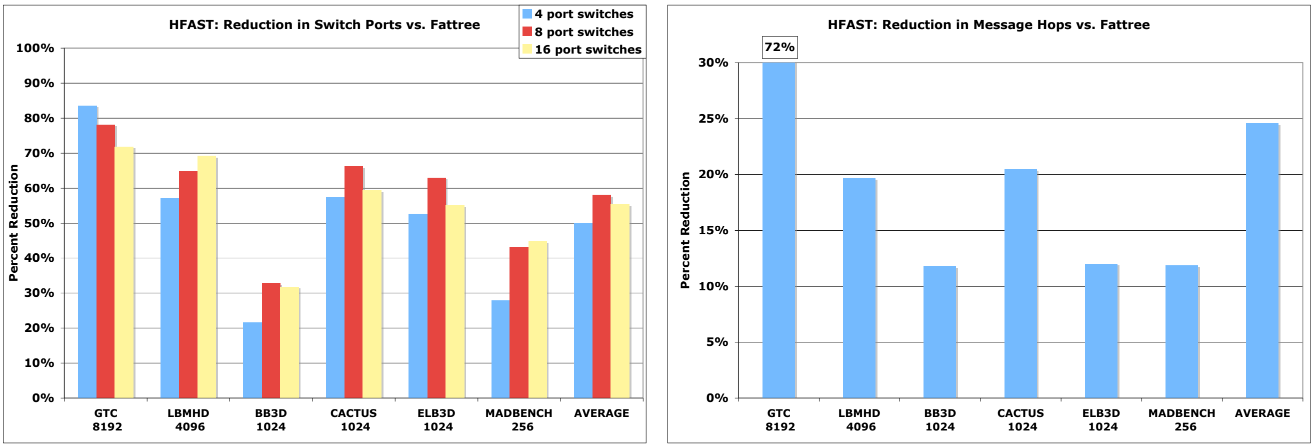
Based on our analysis of communication characteristics for both static and dynamic applications, we now describe an optimization strategy for mapping these characteristics onto the HFAST architecture. First, we define the algorithm for choosing an interconnection topology (also known as “processor allocation”) that minimizes the number of hops per message. Next, we show that for our specific applications, the required bandwidth is much smaller than the full bandwidth available in a traditional fat-tree configuration, especially for large numbers of processors. Note that minimizing the number of message hops reduces the number of required internal ports; thus our motivation here is not to improve interconnect performance, but to achieve equivalent network rates at a lower cost.

5.1 Processor Allocation Algorithm

The effect of processor allocation has been well-observed in the literature and has attracted renewed attention due to increasing numbers of processors in state of the art supercomputers [4]. Processor allocation aims at relocating frequently communicating processes such that they are closer to each other in the communication topology. This has two primary effects: smaller latency and reduced congestion (due to messages consuming bandwidth on fewer links). Obtaining a provably optimal processor allocation is very hard: NP-Complete [17] for the general case. Here, we restrict ourselves to a special case using constraints from the HFAST architecture, and apply a heuristic based on graph partitioning.

Given a graph $G = (V, E)$, a partitioner decomposes the vertex set V into two or more partitions such that there is a non-overlapping subset of the vertices among partitions. We say an edge is *cut* if it connects two vertices from different partitions, and a vertex is a *boundary* vertex if it is connected to a cut edge. In our model, each processor is represented by a vertex in the graph, and two vertices are connected if the associated processors communicate. The goal of our algorithm is to minimize the number of edges that connect vertices from different partitions, while preserving the property that equal numbers of vertices are in each partition.

We further constrain our algorithm to reflect the HFAST networking model, that is, a set of commodity (layer-2) packet and optical switches. The important detail here is that most commodity packet switches have few ports (small radix), so we examine commonly used configurations of 4-, 8-, and 16-port switches. Since we arrange these packet-switches in a tree, for the 4-port case the root of the tree



(a) (b)

Figure 6: Static application comparison of HFAST processor allocation algorithm to fully-connected fat-tree approach showing (a) percentage decrease in number of required switch ports for 4, 8, and 16 port switch configurations and (b) percentage decrease in number of hops (N_{hops}) for an 8 port switch configuration. The concurrency level is shown below each application name.

can have four branches, and all other intermediate nodes can have at most two branches. This constrains the number of cuts for each partitioning: The initial partitioning can perform three cuts, and then each resulting partition undergoes recursive bisection (one cut).

It is important to understand how the results of this algorithm relate to the HFAST architecture, as opposed to classical approaches to processor allocation. Based on the partitions output by the algorithm, the HFAST approach uses optical switches to provision switch blocks so that the number of hops between any two nodes (processors) is the same as the number of cut edges between those two processors. This is, in the abstract, the same as the approach of mapping the partitions onto a fat-tree by moving the processes to the appropriate location. However, the HFAST architecture aims to obtain a similar performance without moving any processes, and without requiring a fully connected network.

To derive the HFAST processor allocation, we implement the partioning algorithm using the MeTis graph partitioning package [16]. We then apply our optimization scheme to detailed IPM communication profiles from runs of both the static and dynamic applications described in Sections 3 and 4. Since this algorithm minimizes the number of hops (N_{hops}) messages need to travel across the interconnect fabric, we use this as the metric for comparing the average predicted N_{hops} using HFAST and the measured N_{hops} on a fat-tree (without processor allocation).

5.2 Static Application Results

Based on the outlined optimization scheme, we now compare the potential benefits of HFAST compared with the traditional fat-tree approach. Figure 6(a) shows the savings in the number of required port switches when using the HFAST methodology for our studied static applications. This gives some indication of the possible cost savings, since there is a fixed cost associated with every switch element. Results show that, on average, the number of switches could be reduced by over 50% compared with fully-balanced fat-trees, even for these relatively small concurrency experiments (in the context of ultra-scale systems). We also explore three possible switch configurations — using 4, 8, and 16 ports — to reflect different interconnect building-block options. Intuition may indicate that as the number of ports per switch increases, the benefit of HFAST diminishes. However, Figure 6(a) empirically demonstrates that this is not the case: for five of our six studied applications, the benefit of HFAST actually increases when the number of ports increases from 4 to 16. This is an encouraging sign for utilizing HFAST in the context of future switch technologies, which will undoubtedly have larger numbers of ports per switch block.

Figure 6(b) shows the benefit of processor allocation with respect to reducing average N_{hops}, compared with the fat-tree approach (using an 8-port switch configuration). This metric reflects the latency overhead of messaging, as each additional hop increases the communication latency component. Observe that the HFAST approach reduces the N_{hops} by almost 25% on average, and by over 72% for the 8192-way GTC experiment — while requiring significantly fewer hardware components than fat-tree implementations. Note that varying numbers of processors (shown below each application) were used for each of the static applications, based on the largest obtainable IPM data sets. It is important to highlight that the experiments conducted to date utilize relatively small numbers of processors compared with the hundreds of thousands (or even millions) that will comprise

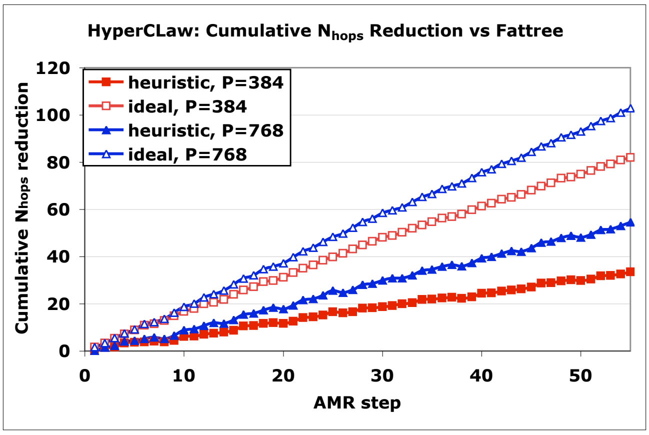


Figure 7: Dynamic AMR application comparison of HFAST processor allocation algorithm to fully-connected fat-tree approach showing cumulative reduction of N_{hops}, using theoretically ideal and heuristic remapping schemes on 384 and 768 processors (with an 8-port switch configuration).

future peta-scale systems. We expect the HFAST advantage to be even more significant on these ultra-scale systems, as the increasing diameter of their networks will dramatically increase the gap between naïve (random) and optimized processor allocation.

5.3 Dynamic Application Results

We now examine HFAST performance in the context of the dynamically adapting communication requirements of the HyperCLaw AMR code. In these experiments, we examine both 384-way and 768-way concurrencies, mapped as 3×128 and 3×256 processors onto a fat-tree, respectively. The results presented in Figure 7 shows how the HFAST approach reduces the average N_{hops} when compared to a traditional fat-tree interconnect. Because the AMR code consists of numerous remapping phases, the savings of N_{hops} is shown as the aggregate total across the multiple time iterations. Two mapping strategies are studied for each concurrency. The first represents a theoretically *ideal* approach where the updated communication pattern could be predicted before the actual remapping phase. The second *heuristic* strategy takes into account that an actual application execution would need to choose an interconnection topology mapping based solely on information from previous timesteps. Our simple heuristic permutes the processors for HFAST timestep T using measurements from step $T - k$, where the offset k is chosen to obtain the most recent timestep at the same AMR grid level — thus avoiding the sharp changes between grid levels seen in Figure 5(b).

Several interesting observations can be made from the HyperCLaw performance data of Figure 7. First, by utilizing an interconnect that can dynamically adapt to an underlying application's evolving communication requirements, the aggregate N_{hops} can be reduced substantially as the calculation progresses through multiple timesteps and data remapping phases — thereby reducing communication overhead. Next, our simple heuristic mapping scheme attains reasonable performance relative to the theoretical ideal (41% and 53% of the ideal case for P=384 and P=768 respectively). Finally, as expected, it can be seen that the higher concurrency experiment (P=768) achieves more significant savings in aggregate N_{hops} compared with the smaller (P=384) test case. These performance gains will continue improving with processor count, especially for the ultra-scale systems targeted by HFAST, where concurrencies are expected to be two or more orders of magnitude greater than our existing experimental platform.

6. 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. To address these challenges, we propose HFAST, a hybrid switch architecture that uses circuit switches to dynamically reconfigure lower-degree interconnects based on the topological requirements of the underlying scientific application.

This work extends our previous HFAST proposal in several important directions. First, we presented an optimization methodology for efficiently mapping application communication requirements onto a processor topology. We then utilized the communication characteristics of six large-scale scientific applications to explore the possible benefits of the HFAST methodology. Results show that, on average, the HFAST approach can save more than 50% of the required switch component hardware compared with a traditional fat-tree approach. Furthermore, HFAST reduces the number of required message hops by an average of 25% — thus demonstrating a potentially significant savings in both infrastructure cost and communication latency overhead.

Next, we conducted an extensive communication-requirement analyis of an adaptive mesh refinement simulation, to study the potential of utilizing our reconfigurable interconnect solution in the context of dynamically adapting multi-scale computations. Results show that by employing our heuristic remapping algorithm, the HFAST approach allows for significant savings in communication overhead, while using fewer network components. Moreover, we expect these savings to grow considerably for future peta-scale architectures, which will undoubtedly require unprecedented levels of concurrencies.

It is important to highlight that the topological permutations discussed in our study are equally applicable to low-degree fixed-topology networks or even IBM's hybrid ICN interconnects [10]. However — unlike HFAST which can dynamically reconfigure the underlying circuit switches — these approaches would require a considerable amount of task migration overhead in order to maintain an optimal embedding of the evolving communication topology.

We also note that the heuristic remapping algorithm introduced in this paper could be applicable to optimizing process placement on systems employing a thin-tree topology, such as the Tokyo Institute of Technology's TSUBAME supercomputing system [22]. In a thin-tree, the upper-tiers of the fat-tree topology have fewer links than are necessary to achieve full-bisection bandwidth. This topology can exploit the sparseness of communication patterns that we observed in Section 3, but only by sorting the process mapping to processes. However, we suspect that the high cost of process migration would isolate the benefits of this approach to static applications. Further examination of the application of our mapping algorithm to thin-trees will be the subject of future work.

In summary, our preliminary findings indicate that an adaptive interconnection architecture such as HFAST offers a compelling combination of flexibility and cost-scaling for next-generation ultra-scale systems. HFAST offers the opportunity for topological permutations without the considerable overhead that would be required for task migration. Future work will expand the scope our application suite and refine our topology optimization schemes, while examining the particular technological components that would bring the HFAST network into existence.

Acknowledgments

All authors from Lawrence Berkeley National Laboratory were supported by the Office of Advanced Scientific Computing Research in the Department of Energy Office of Science under contract number DE-AC02-05CH11231. This research used the resources of the National Energy Research Scientific Computing Center under Contract No. DE-AC02-05CH11231.

7. REFERENCES

[1] F. Abel, C. Minkenberg, R. Luijten, M. Gusat, I. Iliadis, R. Hemenway, R. Grzybowskiand C. Minkenberg, and R. Luijten. Optical-packet-switched interconnect for supercomputer applications. *OSA J. Opt. Network*, 3(12):900–913, Dec 2004.

[2] Adaptive Mesh Refinement for structured grids. `http://flash.uchicago.edu/~tomek/AMR/index.html`.

[3] Kevin J. Barker, Alan Benner, Ray Hoare, Adolfy Hoisie, Alex K. Jones, Darren J. Kerbyson, Dan Li, Rami Melhem, Ram Rajamony, Eugen Schenfeld, Shuyi Shao, Craig Stunkel, and Peter A. Walker. On the feasibility of optical circuit switching for high performance computing systems. In *Proc. SC2005: High performance computing, networking, and storage conference*, 2005.

[4] G. Bhanot, A. Gara, P. Heidelberger, E. Lawless, J.C. Sexton, and R. Walkup. Optimizing task layout on the blue gene/l supercomputer. *IBM Journal on Research and Development*, 49, March/May 2005.

[5] 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.

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

[7] K. Davis, A. Hoisie, G. Johnson, D. Kerbyson, M. Lang, S. Pakin, and F.Petrini. A performance and scalability analysis of the BlueGene/L architecture. In *Proc. SC2004: High performance computing, networking, and storage conference*, Pittsburgh, PA, Nov6-12, 2004.

[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] Center for Computational Sciences and Engineering, Lawrence Berkeley National Laboratory. `http://seesar.lbl.gov/CCSE`.

[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] J.-F. Haas and B. Sturtevant. Interaction of weak shock waves with cylindrical and spherical gas inhomogeneities. *Journal of Fluid Mechanics*, 181:41–76, 1987.

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

[13] J. Shalf J. Carter, L. Oliker. Performance evaluation of scientific applications on modern parallel vector systems. In *VECPAR: 7th International Meeting on High Performance Computing for Computational Science*, Rio de Janeiro, Brazil, July 10-13, 2006.

[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] Metis Homepage. `http://glaros.dtc.umn.edu/gkhome/views/metis`.

[17] Burkhard Monien. The complexity of embedding graphs into binary trees. In L. Budach, editor, *Fundamentals of Computation Theory. Proceedings,*, pages 300–309, 1985.

[18] 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.

[19] C. A. Rendleman, V. E. Beckner, M. L., W. Y. Crutchfield, and John B. Bell. Parallelization of structured, hierarchical adaptive mesh refinement algorithms. *Computing and Visualization in Science*, 3(3):147–157, 2000.

[20] J. Shalf, S. Kamil, L. Oliker, and D. Skinner. Analyzing ultra-scale application communication requirements for a reconfigurable hybrid interconnect. In *Proc. SC2005: High performance computing, networking, and storage conference*, 2005.

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

[22] TSUBAME Grid Cluster. `http://www.gsic.titech.ac.jp`, 2006.