# Early Results from the ACES Interconnection Network Project

Duncan Roweth
Cray Inc, droweth@cray.com

Richard Barrett, Scott Hemmert
Sandia National Laboratories, rbarrett@sandia.com, shemmert@sandia.com

*Abstract*—In the initial phases of ACES-INP we have developed MPI trace file injection methods for Cray simulators that allow detailed analysis of network behavior under realistic application traffic loads. We have gathered application traces for CTH, Sage, and the adaptive mesh refinement step of xNobel, running at scale on the Cielo Cray XE6 system. We are using this data to test and tune our routing algorithms and to access the benefits of functionality proposed for future generations of interconnect. Early results show good performance on realistic application traffic.

As part of the Exascale research program, the DOE lab community is developing mini applications (MiniApps) that are representative of the computational core of major ASC codes. In future work we will integrate execution of these applications with our simulators, enabling us to study the communication patterns of future-looking applications, either by simulating larger scales than we can trace, or by modifying the communication patterns of existing applications to explore new programming models. This approach will allow us to analyze the transition from the bulk synchronous communication paradigm in common use today, to an asynchronous model in which new results are communicated as soon as they have been computed

***Keywords: MiniApps, Co-Design, MPI, Dragonfly Network***

## I. INTRODUCTION

Los Alamos National Laboratory and Sandia National Laboratories have collaborated to create a New Mexico center for high performance computing: the Alliance for Computing at the Extreme Scale (ACES). ACES is funded by the U.S. Department of Energy's Advanced Simulation and Computing (ASC) program and was formed to enable the solution of critical national security problems through the development and deployment of high performance computing technologies. In spring 2010 the ACES consortium and Cray Inc initiated the ACES Interconnection Network Project (ACES-INP), a collaborative research project focused on a potential future interconnection network.. The intent of this project is to analyze potential capabilities that would result in significant performance benefits for a suite of ASC applications. The project encompasses both the NIC and router architectures, with early work focusing on the efficiency of the Dragonfly topology for the communication patterns of selected ASC applications. Assuming our collaborative research and development effort is successful, Cray plans to provide this functionality in future commercially available computer systems.

Due to the historical characteristics of high performance interconnects, many ASC applications have evolved to use a bulk synchronous model of computation. Using the bulk synchronous model, applications go through distinct computation and communication phases, which results in the applications sending larger messages than would be natural for the utilized algorithms. One of the main drivers for this evolution was low MPI messaging rates, leading to the need to bundle messages together in order to saturate network bandwidth.

The bulk synchronous model leads to underutilization of the network during the computation stage. This in turn exaggerates injection bandwidth requirements as the communication occurs at the end of each phase, adding to the time per cycle. On current systems, the computation times are long and these overheads are generally small, but as the computational rate of the nodes increases, or as we increase the number of nodes used to solve a fixed size problem (strong scaling) this ceases to be the case, bandwidths must increase in order to maintain overall efficiency.

This problem is further exacerbated as typical HPC node architectures have not been able to adequately overlap computation and communication, likely due to interference generated in the memory system when both the network and CPU attempt to access the memory simultaneously. The bundling of messages leads not only to higher interconnect bandwidth requirements, but it also wastes valuable memory bandwidth as data is copied into and out of send/receive buffers.

To address these inefficiencies, applications will need to move away from bulk synchronous message aggregation to more asynchronous communication using more natural sized messages distributed throughout computation, eliminating the distinct computation and communication stages. However, applications will need help from the network in order to make this feasible. In particular, the network will need to support high message rates and allow a high percentage of bandwidth utilization for relatively small messages (512 to 2K bytes).

Another important aspect of network utilization is how well the interconnect topology supports the application communication patterns (which nodes communicate with which nodes). The ASC codes are dominated by two- and

three-dimensional physics, where most of the state is transferred only between adjoining cells in the simulation. This leads to the majority of communication being between logical nearest neighbor processes. However, the mapping of application meshes to the physical structure of the machine breaks this locality, increasing average hop counts and hence network load.

Studies of the communication traffic generated by ASC applications show that CTH [2] is the most bandwidth sensitive [2] and as such it is the focus for our work.

As part of the DARPA HPCS program Cray is developing a new network design, known as Dragonfly [3] that provides significantly higher global bandwidth than current systems. This paper outlines the results of trace-based simulations of Sandia's CTH application used to analyze how well the Dragonfly network supports applications that primarily use logical nearest neighbor communication, and will act as the baseline against which we can compare the efficiency of implementing more asynchronous communications.

In the following sections, we introduce the CTH application and it's relevant communication characteristics. Section II describes the simulation infrastructure used for the research and Section III provides an overview of the Dragonfly topology. Simulation results and analysis are detailed in Section IV. Initial conclusions of our work are presented in Section V. Section VI describes future plans for extending our simulation capabilities to allow direct execution of MiniApps.

### *A. CTH*

CTH is a multi-material, large deformation, strong shock wave, solid mechanics code developed at Sandia National Laboratories. CTH has models for multi-phase, elastic viscoplastic, porous and explosive materials, using second-order accurate numerical methods to reduce dispersion and dissipation and produce accurate, efficient results. For these tests we used the shaped charge problem, in three dimensions on a rectangular mesh. Figure 1 illustrates sample output. The shaped-charge consists of a cylindrical container filled with high explosive capped with a copper liner. When the explosive is detonated from the center of the back of the container, the liner collapses and forms a jet which is shown striking a target.

Computation is characterized by regular memory accesses, and is fairly cache-friendly, with operations focusing on two dimensional planes. Inter-process communication aggregates internal-boundary data for all variables into user managed message buffers, subsequently sent to up to six nearest neighbors. CTH's very large message aggregation scheme in the bulk synchronous programming (BSP) model induces a strong separation of computation and communication, illustrated in Figure 2.

Each time step, CTH makes 90 calls to MPI collective functionality, 19 calls to exchange boundary data (two dimensional "faces"), and three calls to propagate data across faces (in the *x, y* and *z* directions). Collective communication is typically a reduction (MPI_Allreduce) of a single variable, although there are also reductions of larger sizes. Each boundary exchange aggregates data from 40 three dimensional arrays, representing 40 variables. Message buffers are constructed for each face, approximately one third of which are contiguous, one third of which are stride $y$, and one third of which are stride $x \times y$. For the problems being studied the bulk of traffic comprises messages of between 2 and 3 megabytes. Given this runtime profile, CTH performance will be most strongly impacted by the exchange of large messages between nearest neighbors, preceded by the accumulation of that data into message buffers and succeeded by the unpacking of the messages into the appropriate arrays.

As currently configured, the CTH message aggregation scheme would benefit strongly from increased interconnect bandwidth. However, experiments are also being conducted to determine if the message aggregation strategy could be dismantled given Cray XE6 (and future predicted interconnect) capabilities. Additional benefits would be reduced on-node memory costs due to reduced, or even eliminated, message buffering requirements.

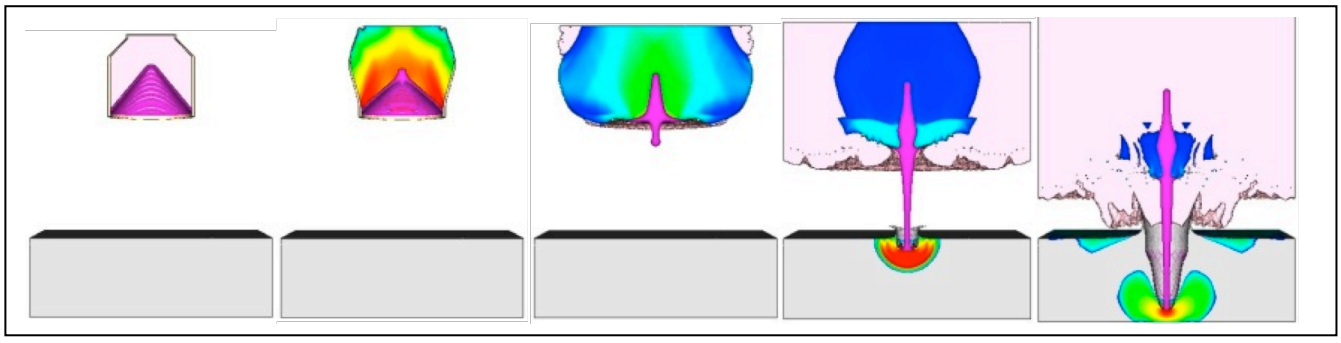

Figure 1: Sample output from CTH shaped-charge problem. The shaped-charge consists of a cylindrical container filled with high explosive capped with a copper liner. When the explosive is detonated from the center of the back of the container, the liner collapses and forms a jet which is shown striking a target.

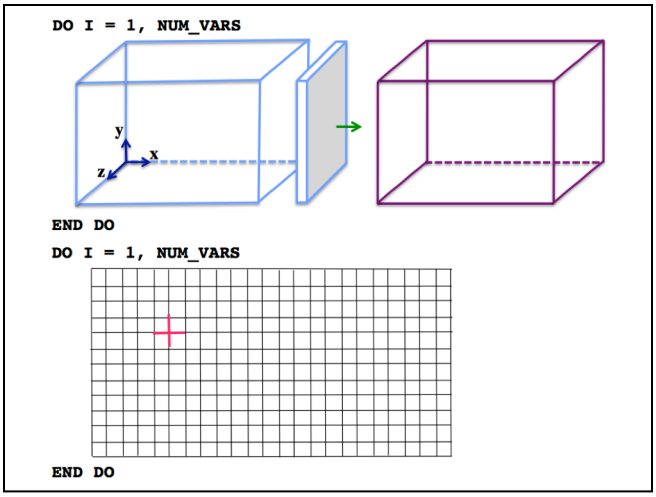


Figure 2: CTH bulk synchronous communication patterns. Each process exchanges face data with its neighbors on every timestep.

## II. METHODOLOGY

Cray has developed two network simulators in support of the Dragonfly design. Dflysim is a high-level simulator that models an abstract network and router. It is used to investigate high-level routing performance and tradeoffs, especially for adaptive routing. Rtrsim is a cycle-accurate micro-architecture simulator modeling the router in detail. It is used to investigate system-level correctness and performance. Rtrsim provides more detail and accuracy, but is much slower. It has been parallelized using MPI and scales well to moderate node counts.

For the Cascade program, both simulators were driven using synthetic traffic: uniform random, permutation or worst case. For ACES we wanted the ability to study network behavior with a variety of ASC application specific traffic patterns. The initial method chosen to achieve this was the addition of support for MPI event traces as a traffic source for rtrsim. This approach allows traces to be captured from the full application executing a realistic problem.

The Sandia DUMPI event trace mechanism [4] was selected for use in the ACES project. DUMPI was developed to facilitate detailed tracing of MPI calls. It provides a low-overhead API for reading large trace files into a simulator. DUMPI event traces can be converted to Open Trace Format (OTF) format for use with standard trace visualization tools.

The Cray network simulators provide per-packet injection and ejection functions. The injection function is called on each cycle for each port on which the network is able to accept new data. The callback fills out a packet description and returns its size in flits (flits are the basic unit of data transfer in Cray networks). Flits are delivered to the output ports on each cycle. The ejection function is called each time a complete packet is delivered. The existing synthetic traffic pattern generators provide an injection function that delivers new packets on some proportion of cycles specified by the input load. The ejection functions log packet statistics. New injection and ejection functions are provided as part of this work, together with a new initialization function that specifies the trace file.

The purpose of Cray's router simulators is to study network behavior in the large and individual router behavior in detail. The NIC model is simple, serving to determine the mapping of traces to NICs and hence to router ports. Detailed NIC models are used for other purposes, but were not used as part of this study. The trace metafile determines the size of the job (number of traces). The NIC model determines the number of traces per NIC and the number of jobs to run simultaneously. Where the job size is small relative to the number of NICs the same set of traces will be used repeatedly. Alternatively the number of jobs used can be specified. Some NICs will be idle (inject no traffic) if size of system being simulated is not divisible by the job size.

The router model requests packets from each NIC in turn. The NIC model requests packets from each MPI task (trace) in turn. Support is provided for multiple traces (MPI tasks) per NIC. If all traces are blocked then no data is supplied on this cycle. A round-robin scheme tracks which NIC/trace last supplied a packet.

### A. *MPI Model*

At this stage in the ACES work we are using MPI event traces to study the behavior of the network under application specific traffic patterns. We are not looking at the impact of MPI implementation or how it provides progression. As such we use an MPI model in which each message consists of a sequence of write request packets from source to destination, with overlapped response packets returning in the opposite direction. Where possible, packets are issued eagerly at a rate matching the injection bandwidth of the hardware. The destination NICs consume packets at a rate equal to the ejection bandwidth of the hardware.

The MPI model maintains a per-trace structure representing the state of an MPI task. Each time a network port can accept a new packet the MPI injection function is called with the relevant trace. The trace can be in one of three states:

- Ready to start a new message
- Part way through sending an existing message
- Blocked waiting for messages to complete

If the trace is ready to start a new message it takes the next one from the transmit queue. If the transmit queue is empty new events are read from the trace file until a blocking event is encountered. Corresponding entries (send or wait) are added to the transmit queue. Receives are checked against the unexpected message queue. The message is marked as complete if a match is found. If there is no match an entry is added to the receive queue, together with a wait for blocking receives.

If the trace is part way through sending a message the next packet is returned. If the trace is blocked waiting then no data is supplied. The NIC model checks the next trace. If all traces are blocked then no data is supplied on this cycle. The status of each blocked message is evaluated on each cycle.

The ejection function is called as each packet is delivered. On delivery of the last request packet the message is passed to the MPI model which searches for a matching receive. If a match is found the corresponding receive is marked as complete, otherwise unexpected. Sends are marked as complete on return of the last response.

Collectives are implemented as a sequence of point-to-point operations with each process communicating to its parent/children on a tree on neighbours on a ring as appropriate. Collectives operate on MPI_COMM_WORLD or communicators created by the application.

## III. DRAGONFLY NETWORK

The 3D torus design used in the Cray XE6 provides excellent scalability, but global bandwidth per node declines with system size[1]. The Dragonfly network provides global bandwidth that scales with system size. It does this at linear cost, with one Aries router per blade. There is no requirement for expensive external switches. Dragonfly networks were developed as a result of work on high radix routers undertaken by Cray and Stanford University [5] as part of the DARPA HPCS program.

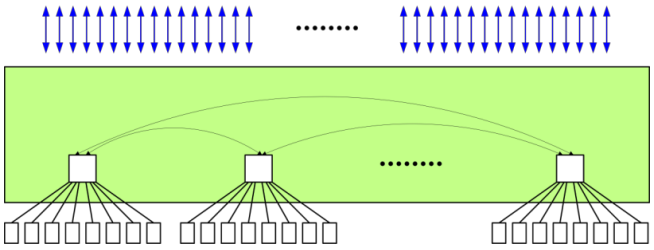

Figure 3: Electrical network connecting the nodes in a Dragonfly group. The optical links for each router (shown in blue) are pooled to form a single high-radix router.

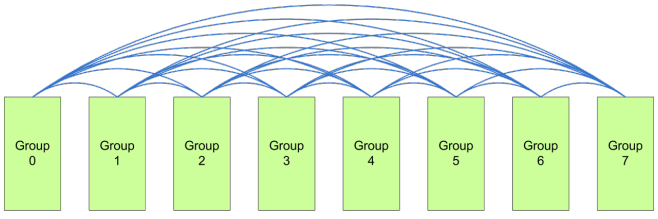


Figure 4: All-to-all network connecting together the groups of a Dragonfly network.

Increasing signal speeds limit the length of electrical links to just a few meters. This is sufficient to connect a few hundred nodes, but not a large system. Optical links are required as well, adding substantially to the cost. In a Dragonfly network, each router provides a mix of short range electrical links and long range optical links. Large groups of nodes are connected using the electrical links (see figure 3). The optical links for all nodes in a group are then pooled as if to form a single very high radix router. These links are used to connect each group to all of the other groups (see figure 4).

One of the strengths of the Dragonfly design is that the electrical group size can be tailored to the system packaging. Electrical groups map to mechanical structures, chassis or cabinets. As each new group is added the system acquires more global optical links. The all-to-all connection structure between groups ensures that bisection bandwidth grows linearly with the number of groups. For large systems global bandwidth is twice the bisection bandwidth.

The Cascade network routes packets either deterministically or adaptively along either a minimal or non-minimal path. A local minimal route (within a group) will always take at most two hops. A global minimal route (between groups) will route minimally in both the source and destination groups, and will take exactly one global optical link. Non-minimal routing in Cascade is an implementation of Valiant's routing algorithm [6]. It is used to avoid congestion and to spread non-uniform traffic evenly over the set of available links in the system. Non-minimal routes can be thought of as routing "up" from the source to a randomly selected intermediary and then minimally "down" to the target. Non-minimal routing doubles the network load. Dragonfly networks over-provision electrical bandwidth within each group so as to compensate for this.

Cascade provides packet-by-packet adaptive routing. Each router along the path will attempt to route minimally but may select non-minimal paths so as to avoid congestion. Routing decisions are made locally using information collected from each router and its neighbours. This data is distributed out-of-band so as to ensure that each device has up-to-date congestion information.

Details of the Dragonfly network used in the Cray Cascade system remain under NDA pending launch of the product later this year. An initial objective of the ACES Interconnect project was to study the behavior of ASC applications such as CTH on Dragonfly networks.

## IV. SIMULATION RESULTS

Simulations were performed using a detailed model of the Cascade hardware (group structure, cycle times, link speeds, etc.) and routing algorithms. Traces were distributed over NICs using a block allocation policy. In our study time spent in computation is set to zero, increasing the network load. This is a conservative approach, but one that is representative of bulk synchronous applications that compute in one phase and communicate in another. Where

[1] Performance of a global communication pattern such as all-to-all; scales inversely with the size of the longest dimension on a 3D Torus.

good overlap is possible communication will in general be spread over longer periods of time reducing network load.

Initial CTH simulations were run with 1024 traces, unused NICs were left idle. Initial results were good, but as the simulation continued the network load dropped off. This was found to be the result of imbalance in the communication traffic. Comparing the sequence of calls made by different processes we found variation of up to 30% in the amount of traffic being sent on each cycle. The network injection load drops off as a result of this imbalance.

In order to study the bandwidth achieved, we truncated message sizes to 64K bytes. In figures 5 and 6 we show initial results for 20 million cycles (approximately 20 milliseconds) with minimal and non-minimal routing. With minimal routing the simulation completed 4 application timesteps, with non-minimal routing the simulation completed 8 timesteps.

The average hop count plot (see figure 5) illustrates aliasing between the problem size and the group size. Some processes perform local communication (within a single group) with all 6 peers while others communicate with 4 local peers and 2 in another group. Non-minimal paths are longer as traffic is routed via an intermediary selected using the non-minimal routing tables and a deterministic hash computed on the packet. In figure 6 we show the total number of packets transmitted by each NIC. The peak counts reflect both the application and the performance of the network with a given routing mode. The lower numbers of packets transferred by some NICs reflect the communication structure of the application. Non-minimal routing shows significantly better performance.

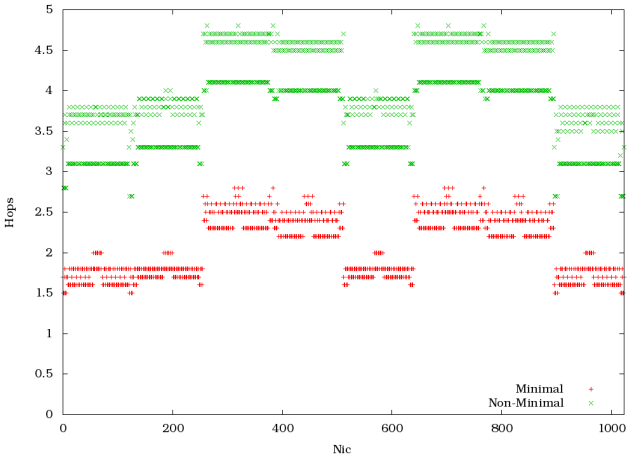


Figure 5: Initial CTH simulation results. Average hop count per NIC for minimal (red) and non-minimal (green) routing.

With a single trace per NIC we see an average bandwidth of 0.58 flits per cycle (see figure 6). The per-trace logs show substantial amounts of time waiting for blocking transfers to complete.

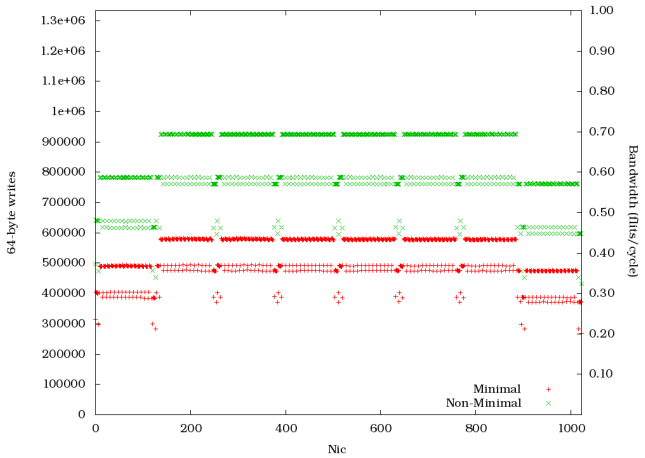


Figure 6: Initial CTH simulation results. Number of 64-byte writes completed per NIC for minimal and non-minimal routing.

The NIC does not inject traffic during this time. In practice there would be multiple processes per node (equating to multiple traces per NIC in our simulation) and traffic from one process would fill the periods when another was idle. In order to study these effects we ran a number of simulations with varying numbers of traces per NIC. In figure 7 we show the number of 64-byte writes per NIC and the equivalent bandwidth in flits per cycle with four traces per NIC. The average transfer rate was 1.22 flits/cycle; approximately 1.9 times higher than with a single trace per NIC. In this study the simulation sustained an average of 68% of peak Cascade MPI bandwidth using CTH trace traffic.

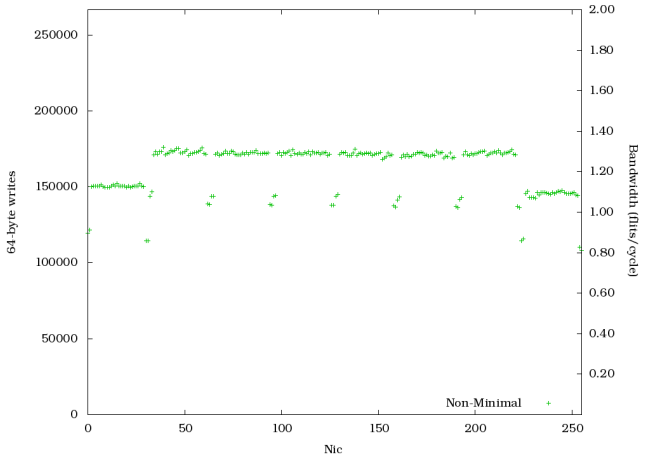


Figure 7: Number of 64-byte request packets transferred per NIC. Results are for 4 CTH traces per NIC with non-minimal routing

The simulator includes a full model of the Cascade load measurement logic and routing pipeline. These simulations allowed us to develop our routing algorithms and check their properties on a wide range of application specific traffic patterns. In figure 8 we show the distribution of load over the optical links connecting groups. One of the objectives of the routing algorithms is to balance this load

across the available links. Our results show good load balance. We were also able to demonstrate good uniformity in the number of times each router was selected as the intermediary for non-minimal routing.

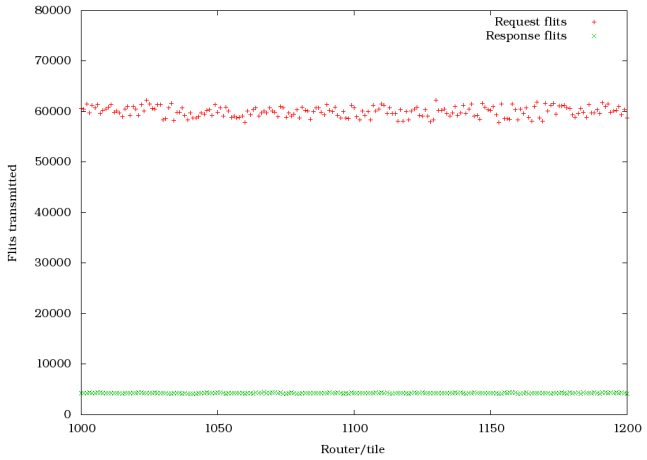


Figure 8: Number of flits transmitted by each optical link. Results are for CTH traffic with non-minimal routing.

The CTH study raised a number of additional questions that could not easily be answered using our trace based simulation: To what extent does communications load imbalance and synchronization in CTH reduce the bandwidth achieved? How will CTH perform on a larger network?

To address these questions we constructed a synthetic traffic generator that follows the CTH structure. On each cycle on which traffic can be injected we generate a packet to each neighbor with equal probability, modeling the case where each process has transfers pending to multiple destinations. In the current version of CTH each process communicates with one neighbor at a time. In general, most, if not all processes on a node will communicate with processes on a neighboring node, limiting path diversity.

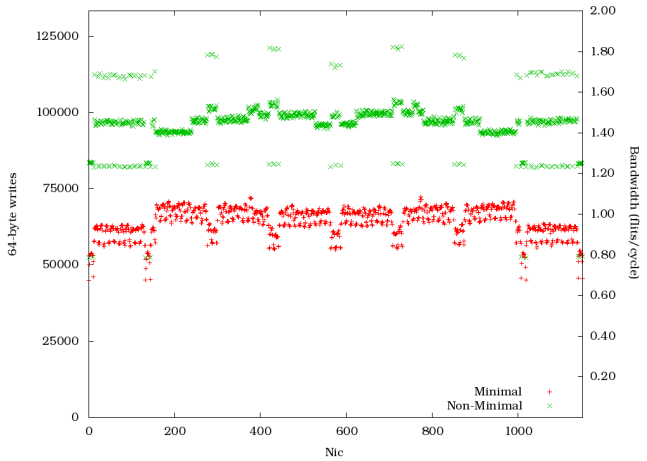


Figure 9: Number of 64-byte writes completed and equivalent bandwidth for synthetic CTH traffic. Results are for minimal routing (red) and non-minimal routing (green).

Average bandwidths obtained using the synthetic traffic generator were 1.45 flits per cycle, approximately 18% higher than that seen with CTH itself, giving us an indication of the application specific overheads. Bandwidths were maintained on larger systems, we tested up to 8000 nodes.

Results so far show significant benefits for non-minimal routing. However, there are likely to be occasions on which the shorter minimal paths will be more lightly loaded. Cascade includes sophisticated adaptive routing hardware designed to select these lightly loaded paths. In order to determine its effectiveness we repeated our tests for a range of adaptive routing settings. Results of this test (see Table 1) show that bandwidths obtained with adaptive routing are up to 25% better than that obtained with non-minimal routing. Best performance is obtained when selecting a non-minimal path if its load is approximately one third higher than the equivalent minimal path. Selecting between minimal paths based on load is shown to be approximately 15% better than selecting them with a deterministic hash.

| Routing mode | 64-byte writes | Minimal |
|---|---|---|
| Non-minimal | 89359 | 0% |
| Adaptive (best settings) | 112377 | 28% |
| Minimal | 66129 | 100% |

Table 1 Number of packets transferred in one million cycles for a range of routing options. Also shown is the percentage of traffic routed minimally.

Repeating our trace based CTH runs with these adaptive routing settings improved performance by between 10 and 15%. In Figure 10 we show average hop counts for trace based CTH traffic with non-minimal and adaptive routing. All NICs show reduced loading with adaptive routing. In this simulation average hop count was reduced from 3.44 to 3.21. The adaptive routing algorithm selected a minimal path on 34.8% of packets.

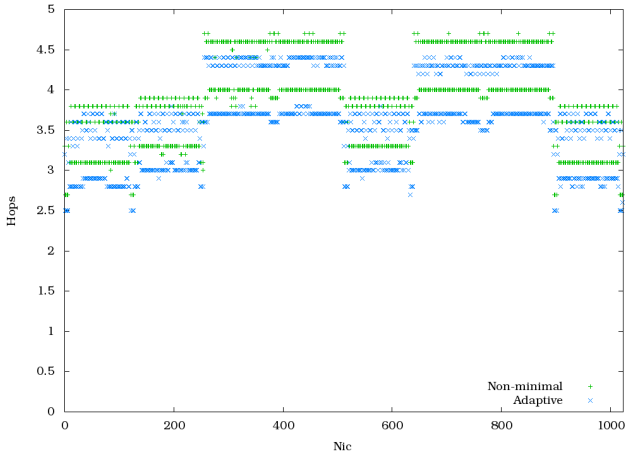


Figure 10: Hop count per NIC. Results are for trace based CTH traffic with non-minimal and adaptive routing.

Our CTH results show relatively low loading on the global links. In figures 11 and 12 we show the number of flits transferred by every port in the simulated system. Results are for CTH trace based traffic using non-minimal routing (figure 11) and adaptive routing (figure 12). The x-axis is ordered by group, router and port. Loading on the electrical ports is shown in green and black. Loading on the optical ports is show in blue.

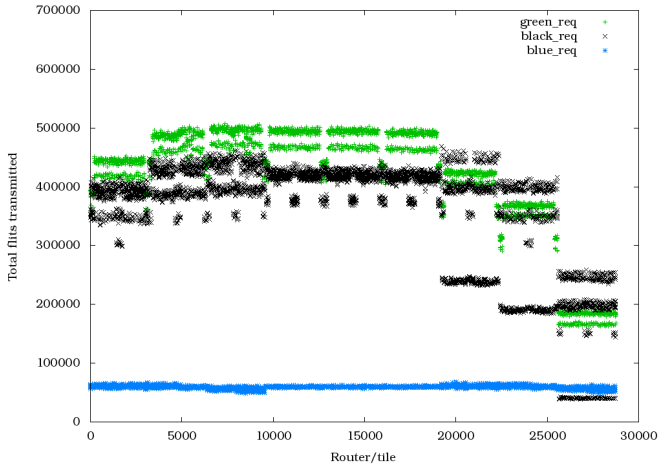


Figure 11 Number of flits transferred per port. Results are for CTH traffic with non-minimal routing.

Loading is low on the right-hand-side of these charts as one third of the NICs in the last group were unused. For adaptively routed traffic (see figure 12) we see higher flit counts on many links as traffic moves away from busy non-minimal paths.

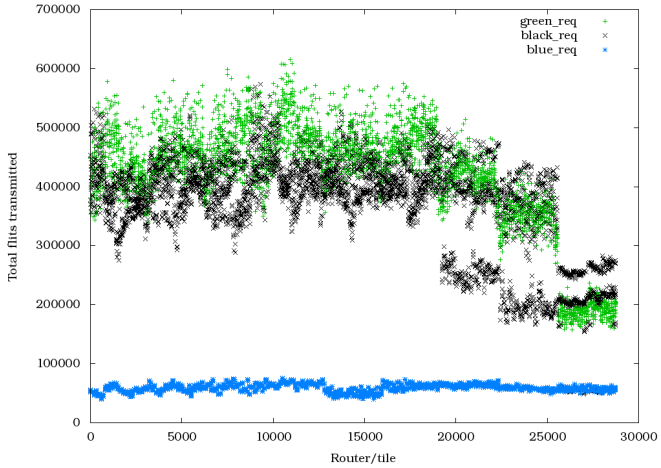


Figure 12: Number of flits transferred per port. Results are for CTH traffic with adaptive routing.

The simulator allows us to reduce the number of global links per group. Route tables are adjusted to exclude unconnected ports just as they will be in the Cascade system. This allows us to test application performance as the number of global links is reduced. Simulation results generated using the CTH traces show no degradation in performance with 50% population of the global links.

## V. Conclusions

CTH processes exchange large amounts of data with neighbors in each of three dimensions. This structure fits naturally onto a torus, although average hop counts increase as we map the logical mesh structure of the application to the physical structure of the machine. On the Dragonfly network used in Cascade, small problems will be contained within a single group they benefit from the excess local bandwidth. Larger problems will be distributed over multiple groups. We have shown that loading on the global links is low and applications with nearest-neighbor communications scale well. Adaptive routing is shown to improve performance on both CTH and our synthetic test cases. The simulation methodology used in this work has allowed us to tune our routing algorithms and to demonstrate key features of these algorithms on realistic traffic.

## VI. Future Work

Work to-date has focused on CTH. The iteration mechanisms used in xNobel and xRage [10] also use local communication patterns. Early results on these codes show that conclusions on network performance drawn from CTH work apply. However, all these codes also perform Adaptive Mesh Refinement (AMR) steps with markedly different communication patterns. We plan to study these codes using traces that cover both iteration cycles and AMR steps.

Working with the full ASC codes and MPI traces gives a detailed picture of the behavior of an application, but it is often a labor intensive effort. To alleviate this, the Mantevo project[7] has produced a set of proxies, called MiniApps, that enable rapid exploration of key performance issues that impact a broad set of scientific application programs of interest to the ASC program and broader computational science and engineering communities. The Mantevo MiniApps are developed and maintained by ASC application code teams, they are available as open source software under an LGPL license.

The Mantevo MiniApps have been demonstrated to accurately reflect key performance issues of a set of full physics and engineering application codes [8]. The miniGhost application [9] is a bulk-synchronous message passing code whose structure is modeled on the computational core of CTH. Two new variants have been developed that will allow us to explore alternative implementations. The options are:

- *Bulk synchronous parallel with message aggregation (BSPMA):* This version accumulates face data, from all variables as they are computed, into message buffers, one per neighbor. Thus six messages are transmitted, one to each neighbor, each time step.

- *Single variable, aggregated face data (SVAF):* This version transmits data as soon as computation on a variable is completed, face data is aggregated. Thus six messages are transmitted for each variable (up to 40), one to each neighbor, and each time step.
- *Single variable, contiguous pieces (SVCP):* A modification of SVAF, with four of the six faces eliminating intermediate buffering in exchange for a significant increasing in the number of messages. That is, the *x-y* faces are transmitted as they are found contiguously in memory, the *x-z* faces are transmitted one message per contiguous column, and the rows of the *y-z* faces are aggregated into individual messages.

A number of possible techniques are being considered for integration of MiniApps and the network simulators. We might, for example, provide an MPI library that interfaces directly with the simulator's injection and ejection functions. The process is complicated by both the simulator and miniGhost being MPI applications that want to call MPI_Init, control execution and generate communications traffic. There is also a problem of relative scale. We typically run a small number of miniGhost timesteps on a large number of MPI ranks. The network simulator typically runs for a large number of cycles on fewer MPI ranks, hundreds rather than thousands or tens of thousands. A simple expedient is to run the MiniApp first to generate trace files in local temporary storage and then use them with the current version of the network simulator.

In subsequent phases of the ACES-INP project we plan to use the miniGhost application to study the impact of moving away from the bulk synchronous model. This work will be undertaken using a combination of prototype Cascade hardware and simulations of future networks.

## ACKNOWLEDGMENT

This work is undertaken as part of the ACES Interconnect Project Subcontract number B580786.

This material is based upon work supported by the Defense Advanced Research Projects Agency under its Agreement No. HR0011-07-9-0001. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency.

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

## REFERENCES

[1] E.S.Hertel, R.L.Bell, M.G.Elrick, A.V.Farnsworth and G.I. Kerley and J.M.McGlaun and S.V.Petney and S.A.Silling and P.A.Taylor, and L.Yarrington, CTH: A Software Family for Multi-Dimensional Shock Physics Analysis, Proceedings, 19th International Symposium on Shock Waves, 1993.

[2] J.H.Laros III, K.T.Pedretti, S.M.Kelly, W.Shu, and C.T.Vaughan Energy Based Performance Tuning for Large Scale High Performance Computing Systems. 20th High Performance Computing Symposium, Orlando, Florida, 2012.

[3] J.Kim, W.J.Dally, S.Scott, D.Abts. Technology-Driven, Highly-Scalable Dragonfly Topology. Proceedings of the 35th Annual International Symposium on Computer Architecture, 2008

[4] J.C.L.Adalsteinsson, H.Cranford, S.Kenny, J.P.Pinar, A. Evensky, D.A.Mayo J. A Simulator for Large-Scale Parallel Computer Architectures.*International Journal of Distributed Systems and Technologies (IJDST), 1*(2), pages 57-73. doi:10.4018/jdst.2010040104, 2010

[5] S. Scott, D. Abts, J. Kim, and W. J. Dally. The BlackWidow high-radix clos network. In Proc. of the International Symposium on Computer Architecture (ISCA), pages 16–28, Boston, MA, 2006.

[6] L.G.Valiant. A scheme for fast parallel communication. SIAM Journal on Computing, 11(2), pages 350–361, 1982.

[7] M.A.Heroux D.W.Doerfler, P.S.Crozier, J.W.Willenbring, H.C.Edwards, A.Williams, M.Rajan, E.R.Keiter, H.K. Thornquist, and R.W.Numrich, Improving Performance via Mini-applications, Sandia National Laboratories, SAND2009-5574, 2009, https://software.sandia.gov/mantevo

[8] R.F.Barrett, P.S.Crozier, S.D.Hammond, M.A.Heroux, P.T.Lin, T.G.Trucano, C.T.Vaughan and A.B.Williams, Assessing the Validity of the Role of Mini-Applications in Predicting Key Performance Characteristics of Scientific and Engineering Applications, Sandia National Laboratories, SAND2012-TBD, 2012.

[9] R.F.Barrett, C.T.Vaughan, and A.Heroux, MiniGhost: A Miniapp for Exploring Boundary Exchange Strategies Using Stencil Computations Scientific Parallel Computing, Sandia National Laboratories, SAND2011-5294832, 2011

[10] M.Gittings, R.Weaver, M.Clover, T.Betlach, N.Byrne, R.Coker, E.Dendy, R.Hueckstaedt, K.New, W.R.Oakes, D.Ranta, and R.Stefan, The RAGE Radiation-Hydrodynamic Code, Journal of Computational Science & Discovery, vol. 1, no. 1, p. 015005, 2008