Analyzing the Scalability of Graph Algorithms on Eldorado

Keith D. Underwood[1], Megan Vance[2], Jonathan Berry[1], and Bruce Hendrickson[1]

[1]Sandia National Laboratories
P.O. Box 5800, MS-1110
Albuquerque, NM 87185-1110

{kdunder, jberry, bahendr }@sandia.gov

[2]University of Notre Dame
Computer Science and Engineering Department
384 Fitzpatrick Hall
Notre Dame, IN 46545
mvance@nd.edu

Abstract

The Cray MTA-2 system provides exceptional performance on a variety of sparse graph algorithms. Unfortunately, it was an extremely expensive platform. Cray is preparing an Eldorado platform that leverages the Cray XT3 network and system infrastructure while integrating a new revision of the MTA-2 processors that is pin compatible with the AMD Opteron socket. Unlike the MTA-2, this platform will have a more constrained network bisection bandwidth and will pay a high penalty for random memory accesses. This work assesses the hardware level scalability of the Eldorado platform on several graph algorithms.

1 Introduction

Algorithms that operate on sparse graph structures are of particular interest to the informatics community. Core algorithms include operations such as connected components, S-T connectivity, sparse matrix vector multiplication, and subgraph isomorphism. These types of algorithms perform extremely poorly on the commodity processors that make up many of today's supercomputers. They tend to make somewhat random references to data that is distributed across the entire system. Thus, their performance is dominated by both the latency of remote accesses and the rate at which those accesses can occur (message rate).

The MTA-2[1, 2] performs extremely well on these algorithms. A 220 MHz MTA-2 processor achieves comparable performance to a 3 GHz Pentium-4. More importantly, because the MTA-2 provides remote load/store operations at a high rate and tolerates remote latency, it scales dramatically better than commodity systems. The downside of the MTA-2 was cost. To address this issue, Cray is introducing the Eldorado[3] system that places MTA-2 processors into AMD Opteron sockets and leverages the much lower cost infrastructure of the Cray XT3 system.

Unfortunately, the changes that reduce the cost also impact performance. The network cannot provide full bisection bandwidth (a feature that MTA-2 depends on). The memory system has changed from sustaining full-rate random accesses to the standard DRAM used by an AMD Opteron. Eldorado will even include a cache (of sorts). This combination of changes calls into question the potential for Eldorado to provide the same orders of magnitude advantage (at scale) that the MTA-2 could.

To evaluate the Eldorado platform, we took a series of measurements from graph kernels on the MTA-2. The kernels included three versions of connected components, three sizes of S-T connectivity, subgraph isomorphism, and sparse matrix vector multiplication. We used these measurements as inputs to a simulation model to evaluate the impacts of the changes to the memory system as well as the network. This paper presents simulation and analytical results that indicate that the Eldorado platform will perform surprisingly well in the face of its limitations. While we do not address *software* scalability issues, the *hardware* appears to scale quite well on a range of applications.

2 Background and Related Work

The Tera supercomputer was introduced in the early 1990s[1, 2]. Numerous efforts have explored the performance of the Tera system and its successor, the MTA-

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

1-4244-0910-1/07/$20.00

2[4, 5]. This paper focuses on an aspect that earlier works have not: scalability to hundreds of nodes.

This work uses SST[6] for its simulation environment. SST provides a hybrid discrete-event/synchronous simulator that enables modeling at both the cycle and discrete event levels. In many ways, the simulations are similar to the message PAssing computeR SIMulator, PARSIM[7]. PARSIM models program execution as a generalized algorithm divided into computation and communication — no attempt is made to model the internals of the processor. In much the same way, we attempt to model what the processor will be doing, but not actually execute an MTA-2 program.

Unlike other network simulators, such as DaSSF[8] and GTNetS[9], our models provide cycle level arbitration details of the router. Packets in the simulation are only two flits long; thus, detailed modeling of features such as queue depth and queue arbitration are critical.

2.1 Eldorado Platform

The Eldorado platform[3] is being introduced as a successor to the MTA-2. Table 1 lists the "speeds and feeds" for the MTA-2 and Eldorado for comparison. The architectures are different in numerous ways. The processor clock has gone up by over a factor of two. The architecture has shifted from uniform memory access (UMA) to non-uniform memory access (NUMA) and from full random access bandwidth to limited random access bandwidth. This change brought the introduction of a 128 KB, 4-way set associative cache with 64 byte cache lines. Finally, the network technology changed and that changed the relative bisection bandwidth per processor from 220 Mref/s to 75 Mref/s. In addition, bisection bandwidth per processor is linear with the number of processors on the MTA-2, but *decreases* as the system size of the Eldorado grows beyond 512 nodes.

3 Methodology

While it is currently not possible to simulate all of the factors affecting performance simultaneously, good approximations can be achieved by simulating some of the factors independently and unifying the simulations through analysis. We chose to analyze several graph algorithms, simulate the performance implications of the DRAM subsystem, and simulate the performance of the network at 512 nodes.

3.1 "Application" Data

Our first objective was to obtain baseline data on the behavior of the selected algorithms from an optimized graph library. We began by measuring the rate at which applications make memory references based on the percentage of VLIW instructions that make a memory reference. This data was obtained from the standard MTA-2 performance tools. The percentage (typically 50% or more) is high enough to imply that a statistical representation is sufficient.

The requested memory access rate does not fully characterize the application, because Eldorado is a non-uniform memory architecture with a much greater penalty for remote accesses (in terms of both latency and access rate) than for local accesses. A second important criteria is the breakdown of local and remote accesses. Cray instrumented their simulator to distinguish stack accesses (effectively, local accesses on the Eldorado platform) from heap (remote) accesses. It was also necessary to capture the full address trace to perform some simulations of the cache architecture. While the stack accesses are not the only accesses that can be made local on Eldorado, making other accesses local would require changes to both the system software and programming model. Realistically, there is limited opportunity for exploiting local memory allocation in graph operations on highly unstructured graphs.

3.2 Cache Experiments

One of the many aspects of Eldorado that differs from the MTA-2 is the memory system. Rather than memory distributed throughout the fabric (independent of any node) will full random access bandwidth, Eldorado has a DRAM attached to each node. To better use this DRAM, Eldorado incorporates a buffer in the memory controller that is managed as a cache of the local DRAM only. The performance of this "cache" is critically important to the performance of the machine as a whole.

To measure the performance of the cache, memory access traces were taken from a single processor on the MTA-2 simulator. Each access in the trace was flagged as a local (stack) or remote access. While the trace from a single processor would certainly capture numerous threads executing, it is possible that the relatively small graphs used in the simulation environment would not lead to full utilization of the processor. To explore this possibility, the single processor data was used to generate multiple threads of access. Measurements were taken (in powers of 2) for each possibility between 1 and 128 replicas. Each replica of the addresses was offset by a constant value (4999872) from the previous replicate to prevent the artificial introduction of cache hits.

Typical codes will also have a combination of local (stack) and remote (heap/dataset) accesses. This skews the timing of the threads and pollutes the cache with network traffic. Thus, the simulator imposed a statistical remote latency and blocked local access based on outstanding load limits. Simultaneously, the cache was exposed to high-rate random accesses to provide a network "polluting" factor.

Table 1. MTA-2 and Eldorado characteristics

Property	MTA-2	Eldorado
Clock	220 MHz	500 MHz
Local Memory Rate (Best)	N/A	500 Mref/s
Local Memory Rate (Worst)	N/A	100 Mref/s
Data "Cache"	N/A	128 KB, 64B line
Topology	Modified-Cayley	3D-Torus
Remote Memory Rate (Net, Best)	220 Mref/s	75 Mref/s
Bisection BW	$3.5GB/s \times P$	$15.3GB/s \times P^{2/3}$

3.3 Network Simulation

The second major factor in Eldorado performance that differs dramatically from the MTA-2 is the network. While the MTA-2 used a full bisection bandwidth modified Cayley graph network, Eldorado uses the 3D torus network from the Cray XT3. The experiments here assume an $8 \times 8 \times 8$ topology of 512 nodes. Variations from this topology that reduced bisection bandwidth would have negative consequences for performance.

The network simulation starts at the edge of the processor with the HyperTransport (HT) link into the Seastar network chip. The HT link was simulated over a range of parameters, from a minimal value to the maximum possible peak value. The performance of the HT link was found not to impact the overall results. The router was modeled in great detail. All of the router queue depths are modeled as are the latencies moving from one queue to the next. The queue arbitration uses round-robin arbitration. In addition, we make the assumption of a dimension order routing algorithm. This does not account for any gains that could be achieved by virtual channel spreading or the losses that might occur if a node were down. Adaptive routing is not implemented in the XT3 router.

The network link has a 3.84 GB/s data rate, but less than 3 GB/s is available for actual data transfer because of the reliability protocol on the link. The link is modeled as a 4 GB/s link with overhead added to account for the link protocol. Serialization and deserialization latency is modeled as part of the latency associated with controlling the link.

3.4 Network Traffic Generation

Network traffic was generated using statistical methods that are designed to match the algorithms. The nominal request rate was taken to be the fraction of instructions including a memory reference multiplied by the clock rate of the Eldorado processor. This yielded a sweep space of 150, 230, 300, and 400 million references per second. The local versus remote (stack versus heap) percentages were swept from 10% local to 80% local, since coding styles can be changed to increase the amount of local data used. The distribution of network accesses was assumed to be purely random. That is, the hash algorithm used to distribute memory on the MTA-2 processor is assumed to work and the code is assumed to have no significant hotspots.

The traffic generated at each node was assumed to originate from some number of threads running on the Eldorado processor (32, 64, or 128). Each thread was assumed to be a stream of random addresses. A load-to-use parameter, known as "lookahead" was imposed on each thread for lookaheads of 2, 4 and 8. Threads were only scheduled if they had sufficient lookahead remaining.

3.5 Memory Impacts on the Network

Network requests must be serviced by the memory subsystem on the Eldorado chip. Integrating 512 full memory system simulations with the simulation of the network would have yielded prohibitive simulation times. Thus, network simulations were run with statistical memory characteristics and swept over several values. Memory characteristics were determined by average cache hit rate giving each access a probability of hitting the cache. Misses consumed part of a simulated memory bandwidth and incurred a higher delay, which was also impacted by contention. Network accesses were assumed to never hit the cache (and only pollute it) while local accesses were studied over several cache hit rates.

3.6 Simulation Durations

When simulating the cache and memory subsystem, entire traces were simulated; however, when simulating a 512 node network, it is necessary to choose a shorter simulation duration. Given the constant, uniform, statistical nature of the network traffic generation, it is possible to reach "steady state" relatively quickly. To reach steady state, we ran the simulation until network characteristics did not change for several checkpoints.

Table 2. Memory access characteristics of several kernels

Kernel Name	% Memory References	% Stack	Access Rate Mref/s		
			Total	Global	Local
Connected Components: Bully	59	46	295	159	136
Connected Components: Kahan	60	53	300	141	159
S-T Connectivity: Small	75	10	375	338	37
S-T Connectivity: Medium	60	28	300	216	84
S-T Connectivity: Large	60	32	300	204	96
Sparse Mat. Vect.	46	53	230	108	122
Subgraph Isomorphism	30	34	150	99	51

4 Results

Data was gathered from each of the graph kernels and used to study the impact of the "cache" introduced on the Eldorado. Information from these studies is correlated to network simulations, which enables us to make predictions about the scalability of the graph kernels studied.

4.1 Graph Kernel Characteristics

Table 2 presents characteristics from several graph kernels. Two versions of connected components are presented: the Bully algorithm (the best performing algorithm on the MTA-2) and the Kahan algorithm. S-T connectivity is presented for the "small" (less than 30 nodes visited), "medium" (1000-2000 nodes visited), and "large" cases (at least 10000 nodes visited). Sparse matrix vector multiply and subgraph isomorphism are also presented.

Two measurements are presented along with three derived metrics that are related to Eldorado. The first column shows the percentage of VLIW instructions that include a memory reference. The second column is the percentage of those references that go to the stack (local on Eldorado). The last three columns convert these characteristics to an access rate which is broken into local and global rates.

Referring to Table 1, we can compare the application's demands the platform's capabilities. When most accesses hit the cache, an Eldorado processor can service 500 million memory references per second. When no accesses hit cache, the node can service 100 million memory references per second from the DRAM directly. Network requests never hit the cache and the network could be requesting up to 75 million references per second. The remaining 25 million references per second plus any data serviced by the cache must be sufficient for the local accesses. Clearly, the network will not be sufficient to sustain most of the codes at maximum rate. What is slightly more subtle, however, is that it is also possible for the DRAM to become the limiting factor if the cache hit rate is insufficient for local accesses.

Table 3. Cache hit rate of several kernels for several replication degrees

	Replications			
Kernel Name	$64\times$	$16\times$	$4\times$	$1\times$
CC: Bully	20%	63%	85%	99%
CC: Kahan	13%	52%	79%	92%
S-T Connectivity	85%	95%	99%	99%
Sparse Mat. Vect.	70%	85%	93%	99.9%
Subgraph Iso.	63%	69%	85%	87%

4.2 Caching Simulation

Table 3 shows the cache hit rates of each graph kernel under varying degrees of trace replication. Small, medium, and large cases for the S-T connectivity kernel could not be separated due to the nature of the address trace. Interpretation of this data requires considering two factors: how busy was the single node simulation and how busy will the Eldorado node be. These factors are difficult to determine, so the caching simulation swept over a range of possibilities. For each kernel, the columns represent the cache hit rate when we assume that Eldorado will have to be $64\times$ as busy (or $16\times$, $4\times$, or $1\times$) as the MTA-2 simulation. For the local DRAM to be fast enough that it is not the bottleneck, it must be able to service the local node's local request rate. If we assume that the network is consuming 75% of the local DRAM bandwidth, then the local request rate that misses the cache must be below 25 Mref/s. Table 4 shows the number of local DRAM accesses each kernel would need assuming the cache hit rates in Table 3.

If Eldorado has fewer than $4\times$ as many threads per node as the MTA-2 simulation, the cache should perform adequately. Higher rates may lead to limitations from the DRAM; however, even at $16\times$ replication, the network is likely to be more limiting that the DRAM system.

Table 4. DRAM access rate (Mref/s) needed

	Replications			
Kernel Name	64×	16×	4×	1×
CC: Bully	108	50	20	1.3
CC: Kahan	138	76	33	12.7
S-T Connectivity	14	5	1	1
Sparse Mat. Vect.	37	18	9	0.1
Subgraph Iso.	19	16	8	7

4.3 Network Simulation

Over 1300 application configurations were simulated for a 512 node ($8 \times 8 \times 8$) system. This section presents those data points in the space immediately around the current characteristics measured for the algorithms analyzed. For each application, we assume the "typical" lookahead is 4. The typical number of threads per processor is assumed to be 64. The access rate, local reference percentage, and cache hit rate are taken from the application measurements. The programming model of Eldorado and optimizations made for Eldorado may shift any number of these properties. Thus, for each kernel, the impact of moving along each of two planes of interdependent variables is graphed. The first plane presented is the access rate/local reference percentage plane to explore the impacts of adding or removing traffic from the network. The second plane presented is the lookahead/threads per processor plane to explore the impacts of available concurrency.

4.3.1 General Trends

Cache hit rate can be critically important. Where the stack percentage is high, the DRAM reference rate can easily become the bottleneck. As an example, a 50% hit rate is insufficient when the stack percentage is 50% and the access rate is 230 Mref/s. Stack percentage determines when the network bisection bandwidth saturates; thus, it tends to be the primary bottleneck. The 230 Mref/s to 300 Mref/s memory instruction rate means that the typical 50% stack percentage seen in the graph kernels will lead, in and of itself, to a $2\times$ performance penalty at a system scale of 512 nodes. On a positive note, this is comparable to the performance that might be expected from a current MTA-2 system. With the right development environment, many of the algorithms should achieve a higher "local" memory percentage. Lookahead and the number of threads per processor are tightly coupled parameters. Fewer threads leads to the need for more lookahead. Similarly, lower lookahead requires more threads. Generally speaking, a lookahead of 4 looks achievable. With a lookahead of 4, 64 threads generally look sufficient.

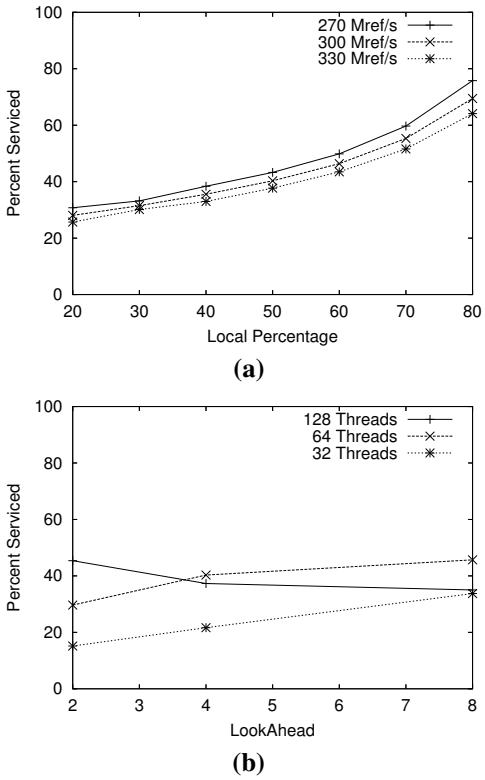

Figure 1. Impact of (a) global request rate and (b) concurrency on Connected Components

4.3.2 Connected Components

The properties of the two connected components algorithms are very similar. About 60% of the instructions reference memory and about 50% of those are local. That leads to a request rate for the network of 150 Mref/s and a local reference rate of about 150 Mref/s. Given the data from Table 3, we will assume a cache hit rate of 70%. Simulations have indicated that the network bisection bandwidth is on the order of 67 Mref/s, and so we would expect 44% of the request rate of these applications to be serviced. That translates into a $2.3\times$ performance penalty from what the peak would be.

Given that the shift to Eldorado may change some of the salient features of the graph kernels, we have explored increasing and decreasing the access rate by 10%. We have also explored the entire spectrum of local percentages for each of these access rates. The results in Figure 1(a) indicate that performance approximately tracks changes in memory demands. To the right of the curve, however, we can observe that there may be value in increasing the number of total references if that makes more references local.

The impact of concurrency points out the need to find high levels of concurrency in the application. That said, 64 threads per processor with a lookahead of 4 seems to be

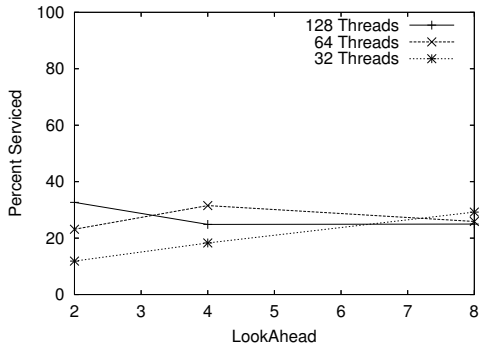

Figure 2. Impact of concurrency on S-T Connectivity (medium)

sufficient to saturate the network. Indeed, providing significantly more concurrency (128 threads, lookahead of 8) seems to reduce performance. While this seems counter-intuitive, it appears multiple times in the simulations. Our best current understanding indicates that this is a real result that is caused by the extremely non-linear relationship between network delay and offered load.

4.3.3 S-T Connectivity

The medium and large S-T connectivity scenarios are similar to connected components. The memory reference rate is the same and the percentage of local accesses is similar; however, since the concurrency impacts can be differ when the local percentage changes, those are graphed in Figure 2. As with Figure 1(b), if too much concurrency is exposed, network performance suffers. The break points differ because the percentage of the memory request traffic that goes to remote nodes is higher. At this level of load, we expect a $3\times$ performance hit against the theoretical maximum.

When the number of nodes visited by the S-T connectivity algorithm is small, we see an extremely poor match to Eldorado. 75% of the instructions reference memory and 10% of those are local. Reducing the extreme to an 80% reference rate with 20% of those being local leads to a request rate for the network of 320 Mref/s and a local reference rate of 80 Mref/s. Based on Table 3, we will assume a cache hit rate of 90%. Given the network bisection bandwidth, we would expect 21% of the request rate of these applications to be serviced. That translates into a $5\times$ performance penalty from the peak.

Changes in access rate yield similar trends to what was seen with connected components; however, the right side of the curve is steeper. The impact of concurrency on this example (Figure 3) are virtually non-existent. The network is so overwhelmed that reducing concurrency only has an impact at the lowest level (32 threads, lookahead of 2).

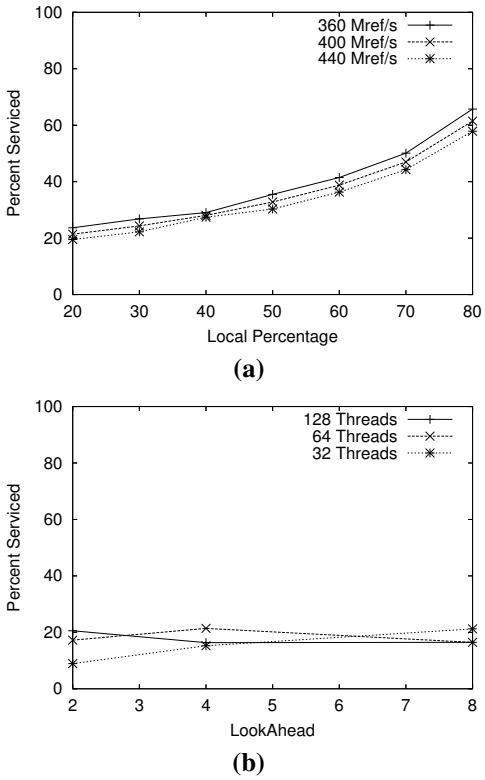

Figure 3. Impact of (a) global request rate and (b) concurrency on S-T Connectivity (small)

4.3.4 Sparse Matrix Vector Multiply

Sparse matrix vector multiply is a good match to the Eldorado architecture. Only 46% of the instructions reference memory and over 50% of those are local. That leads to a request rate for the network of 108 Mref/s and a local reference rate of 122 Mref/s. We will use a cache hit rate of 90% from Table 3. Based on network bisection bandwidth, we would expect over 62% of the request rate to be serviced and yield a mere $1.6\times$ performance penalty.

Figure 4(a) paints a promising picture. When the application and architecture match, small improvements in the application behavior lead to dramatically better performance. This is promising given that sparse matrix vector multiply performs well on distributed memory platforms. With a slightly improved programming model that includes distributed memory, it should be possible to achieve all of the potential performance on this application.

The concurrency data in Figure 4(b) is strikingly different from that of the network bound algorithms. When the network is not constricting remote memory requests, the amount of concurrency exposed becomes critically important. It is possible to see a factor of 1.5 to 2 difference in application performance based on the exposed concurrency.

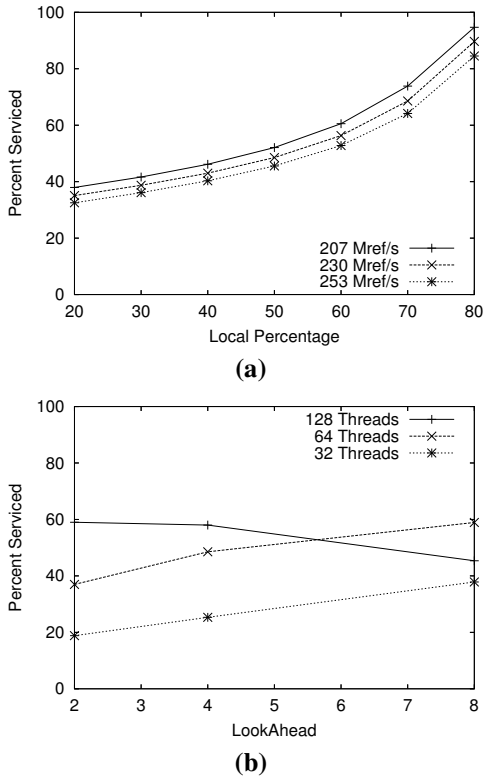

Figure 4. (a) Request rate and (b) concurrency impact on Sparse Matrix Vector Multiply

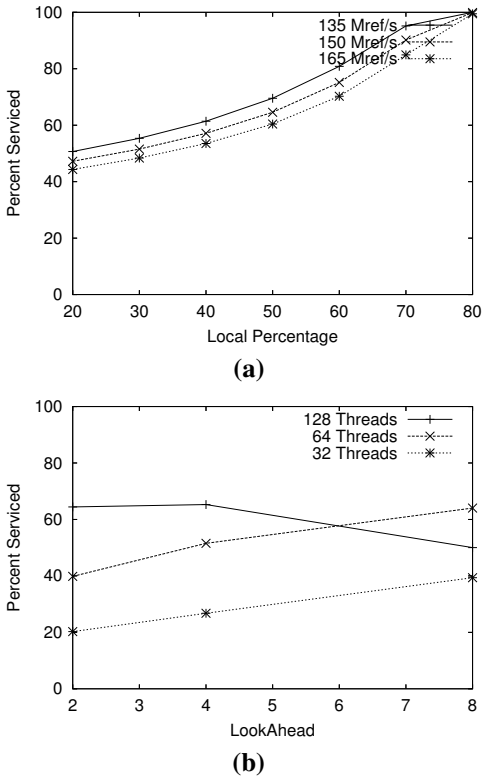

Figure 5. Impact of (a) global request rate and (b) concurrency on Subgraph Isomorphism

4.3.5 Subgraph Isomorphism

Subgraph Isomorphism is a great match to the Eldorado architecture, thanks to a filtering step that dominates its performance. Only 30% of the instructions reference memory, but only 34% of those are local. This yields the lowest network request rate (99 Mref/s) and one of the lowest local request rates (51 Mref/s). All of this is accompanied by a cache hit rate of over 70%. This translates into an algorithm that should run within $1.5\times$ of peak. The most promising part of subgraph isomorphism is that the percentage of memory references that are local could be increased to 60 or 70%, at which point the network could satisfy most of the requests. This could be achieved by moving a single copy of the subgraph to a "local" memory location.

4.4 Load Imbalance Issues

A major concern with the Eldorado network is the potential for a network interface become overloaded. This network hot spot could lead to traffic congestion in the network that affects system level performance. Figure 6 graphs the impact on the overall average, the source nodes, the target nodes, and the "other" nodes. The Y-axis is the percent serviced and the X-axis is an overload metric. Percent overload is the percent of the traffic that is sent to the target set of nodes *beyond their fair share*. As an example, in a 512 node system, each node should send 0.2% of its traffic to every other node. If a node has a *percent overload* of 2 with a target node set of 2 nodes, it will send 1.2% of its traffic to each of those two nodes and will reduce the traffic to the other 510 nodes equally.

Figure 6(a) shows the scenario where a group of 8 source nodes overload 2 target nodes[1]. This is a "small, bad application" scenario. Neither the average nor the non-participating nodes are particularly affected. Even the source nodes are not dramatically affected until the overload is extreme. Clearly, a single, small job is unlikely to have a particularly detrimental effect on the system.

In contrast, Figure 6(b) shows 510 nodes overloading 2 targets. Even the 0.25% overload scenario shows a significant degradation. At 0.5% overload, the average service rate has dropped to only 57% of the peak service rate. That is a $1.5\times$ performance hit for an increase of 1 memory reference in 200 going to a pair of nodes.

[1]Request rate is fixed at 300Mref/s, local percentage at 50%, cache hit percentage at 90%, and lookahead at 4.

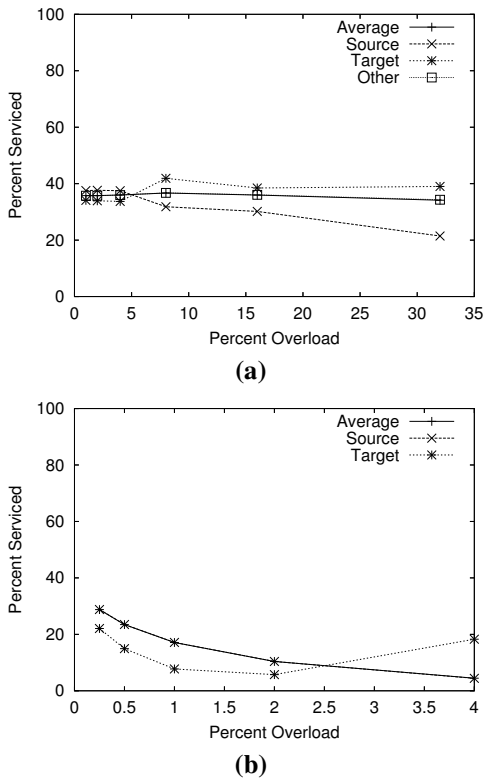

Figure 6. Impacts of (a) constrained (8-to-2) overload and (b) widespread (510-to-2)overload

5 Conclusions

While the Eldorado platform shares many of the characteristics of the MTA-2 platform, it also has dramatic differences that could inhibit performance. The performance critical network and memory systems have changed for the worse, while the processor rate has increased. This paper presents simulations of the various components of the system and ties those results together through analysis. The results are promising in that the limitations of the network and DRAM do not seem to impose dramatic constraints on application performance. Subgraph isomorphism and sparse matrix vector multiply should achieve performance that is only 35% below the peak performance. The connected components algorithms and the S-T connectivity algorithm do not perform as well, but they are still expected to be within 60-75% of the peak processor performance. Thus, we do not see a severe *architectural* limitation to the scaling of the Eldorado platform to 512 nodes. Scalability, however, extends beyond the machine architecture. Issues of thread creation and thread management along with a variety of run-time system issues and compiler issues can put scalability at risk. These issues cannot yet be directly evaluated for the Eldorado platform.

References

[1] R. Alverson, D. Callahan, D. Cummings, B. Koblenz, A. Porterfield, and B. Smith, "The tera computer system," in *Proceedings of 1990 International Conference on Supercomputing (ICS90)*, Amsterdam, The Netherlands, June 1990, pp. 1–6.

[2] G. Alverson, R. Alverson, D. Callahan, B. Koblenz, A. Porterfield, and B. Smith, "Exploiting heterogeneous parallelism on a multithreaded multiprocessor," in *Proceedings of 1992 International Conference on Supercomputing (ICS92)*, Washington, DC, USA, July 1992, pp. 188–197.

[3] J. Feo, D. Harper, S. Kahan, and P. Konecny, "Eldorado," in *2nd Conference on Computing Frontiers*, Ischia, Italy, May 2005, pp. 28–34.

[4] S. Brunett, J. Thornley, and M. Ellenbecker, "An initial evaluation of the tera multithreaded architecture and programming system using the c3i parallel benchmark suite," in *Proceedings of the 1998 ACM/IEEE conference on Supercomputing*, Nov. 1998.

[5] A. Snavely, L. Carter, J. Boisseau, A. Majumdar, K. S. Gatlin, N. Mitchell, J. Feo, and B. Koblenz, "Multiprocessor performance on the tera mta," in *Proceedings of the 1998 ACM/IEEE conference on Supercomputing*, Nov. 1998.

[6] A. Rodrigues, *Programming Future Architectures: Dusty Decks, Memory Walls, and the Speed of Light*. University of Notre Dame, 2006, ch. 3, pp. 56–81.

[7] A. Symons and V. L. Narasimhan, "The design and application of PARSIM - a message passing computer simulator," 1997. [Online]. Available: citeseer.nj.nec.com/288506.html

[8] J. Liu and D. M. Nicol, *DaSSF 3.1 User's Manual*, Dartmouth, April 2001.

[9] G. F. Riley, "The georgia tech network simulator," in *Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research*. ACM Press, 2003, pp. 5–12.