ARGUS: SUPERCOMPUTING IN 1/10 CUBIC METER

Xizhou Feng, Rong Ge and Kirk W. Cameron
Department of Computer Science and Engineering
University of South Carolina, Columbia, SC 29208, USA
{fengx, ge, kcameron}@cse.sc.edu

ABSTRACT

We propose ARGUS, a high density, low power supercomputer built from an IXIA network analyzer chassis and load modules. The prototype is a diskless MPP scalable to 128 processors in a single 9U chassis. The entire system has a footprint of 1/4 meter2 (2.5 ft2), a volume of 0.09 meter3 (3.3 ft3) and maximum power consumption of less than 2200 watts. We compare and contrast the characteristics of ARGUS against various machines including our 32-node Beowulf and LANL's Green Destiny. Our results show that the computing density (GFLOP/ft3) of ARGUS is about 30 times higher than that of the Beowulf, about 3 times higher than that of Green Destiny while performance is comparable.

KEY WORDS

Super computing, high density computing, cluster system, performance evaluation, parallel and distributed system

1. Introduction

Mainstream high performance computing systems often consist of clusters of symmetric multi-processors (SMP) communicating across fast interconnects. Computing resources may be special purpose (e.g. Earth Simulator) or general purpose (e.g. network of workstations). While these high-end systems often provide unmatched computing power, they are extremely expensive, requiring special cooling systems, enormous amounts of power and dedicated building space to ensure reliability. It is common for a supercomputing resource to encompass an entire building and consume tens of megawatts of power.

In contrast low-power, high-throughput, high-density systems are typically designed for a single task (e.g. image processing [1]). These machines offer exceptional speed (and often guaranteed performance) for certain applications. Design constraints include performance, power, and space making them expensive to develop and difficult to migrate to future generation systems.

We propose an alternative approach augmenting a specialized system (i.e. an Ixia network analyzer) that is designed for a commodity marketplace under performance, power, and space constraints. Though the original Ixia machine is designed for a single task, we have created a configuration that provides general-purpose high-end processing in a Linux environment. Our system provides computational power surpassing Green Destiny [2, 3, 4] (another low-power supercomputer) while decreasing volume by a factor of 3.

2. System Design

Figure 1 provides a detailed diagram of the architecture of our prototype called ARGUS. This architecture consists of four sets of separate components: the IXIA chassis, the IXIA Load Modules, the multi port fast Ethernet switch and an NFS server.

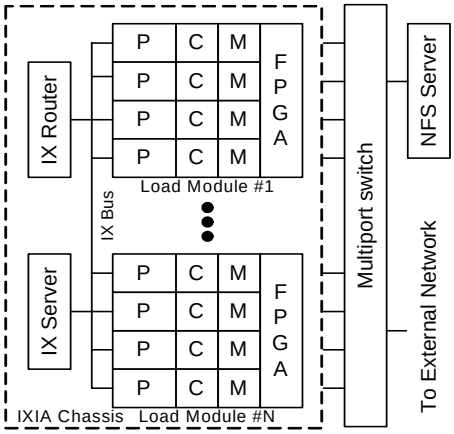

Fig. 1. ARGUS Hardware Architecture. Up to 16 Load Modules are supported in a single IXIA 1600T chassis. A single bus interconnects modules and the chassis PC while external disks and the cluster front-end are connected via an Ethernet switch. P=processor, C=Cache, M=Memory.

The chassis contains a power supply and distribution unit, cooling system, and runs windows system and proprietary software (IX server and IX router). Multiple (up to 16) Load Modules plug into the chassis and communicate with the chassis and each other via an IxBus. Each Load Module provides up to 8 RISC processors in a dense form factor and each processor has its own operating system, cache (L1 and L2), main memory and network interface. Additional FPGA elements on each Load Module aid real-time analysis of network traffic. Though the performance abilities of these FPGAs have merit, we omit them from consideration for two reasons: 1) reprogramming is difficult and time

consuming, and 2) it is likely FPGA elements will not appear in succeeding generation Load Modules to reduce unit cost.

There is no disk on each Load Module. We allocate a small portion of memory at each port to store an embedded version of the Linux OS kernel and application downloaded from IX Server. An external Linux machine running NFS file server is used to provide external storage for each node. A possible improvement is to use networked memory as secondary storage but we did not attempt this in the initial prototype. Due to cost considerations, although the Load Modules support 1000 Mbps Ethernet on copper, we used a readily available switch operating at 100 Mbps.

The first version of the ARGUS prototype is implemented with one IXIA 1600T chassis and 4 LM1000TXS4 Load Modules [5] configured as a 16-node distributed memory system, i.e., each port processor is considered an individual node. Another option is to configure each Load Module as an SMP node. The second option requires use of the IxBus between Load Modules. The IxBus bus (and the PowerPC 750 processor) does not maintain cache coherence and suffers from limited bandwidth. Thus, early on we eliminated this option from consideration since software-driven cache coherence will limit performance drastically. We opted to communicate data between all processors through the Ethernet connection. Hence one recommendation for future implementations is to significantly increase the performance and capabilities of the IxBus. This could result in a cluster of SMPs architecture allowing hybrid communications for improved performance.

Each LM1000TXS4 Load Module provides four 1392 MIPS PowerPC 750CXe RISC processors [6] with 128M memory and four network ports with auto-negotiating 10/100/1000 Mbps Copper Ethernet interface. The 1392 MIPS PowerPC 750CXe CPU employs 0.18 micrometer CMOS copper technology, running at 600 MHz with 6.0W typical power dissipation. This CPU has independent on-chip 32K bytes, eight-way set associative, physically addressed caches for instructions and data. The 256KB L2 cache is implemented with on-chip, two-way set associative memories and synchronous SRAM for data storage. The external SRAM are accessed through a dedicated L2 cache port. The PowerPC 750CXe processor can complete two instructions per CPU cycle. It incorporates 6 execution units including one floating-point unit, one branch processing unit, one system register unit, one load/store unit and two integer units. Therefore, the theoretical peak performance of the PowerPC 750CXe is 1200 MIPS for integer operations and 600 MFLOPS for floating-point operations.

In ARGUS, message passing (e.g. MPI) is chosen as the model of parallel computation. We ported gcc3.2.2 and glib for PowerPC 750 CXe to provide a useful development environment. MPICH 1.2.5 and a series of benchmarks have been built and installed on ARGUS. Following our augmentation, ARGUS resembles a standard Linux-based cluster running existing software packages and compiling new applications.

3. Related Work

According to design priorities, general-purpose supercomputers can be classified into four categories:

- **Performance**: These are traditional high-performance systems (e.g. ASCI Q) where performance (GFLOPS) is the absolute priority.
- **Cost**: These are systems built to maximize the performance/cost ratio (GFLOPS/$) using commercial-off-the-shelf components (e.g. Beowulf).
- **Power**: These systems are designed for reduced power (GFLOPS/Watt) to improve reliability (e.g. Green Destiny) using low-power components.
- **Density**: These systems have specific space constraints requiring integration of components in a dense form factor (e.g. Green Destiny) for a high performance/volume ratio (GFLOPS/ft^3).

Though high performance systems are still a majority in the HPC community low cost, low power, low profile and high density systems are emerging. Blue Gene/L (IBM) [15] and Green Destiny (LANL) are two examples designed under cost, power and space constraints.

ARGUS is most comparable to Green Destiny. Green Destiny prioritizes reliability (i.e. power consumption) though this results in a relatively small form factor. In contrast, the ARGUS design prioritizes space providing general-purpose functionality not typical in space-constrained systems. Both Green Destiny and ARGUS rely on system components targeted at commodity markets.

Green destiny uses the Transmeta Crusoe TM5600 CPU for low power and high density. Each blade of Green Destiny combines server hardware, such as CPU, memory, and the network controller into a single expansion card. ARGUS uses the PowerPC 750CXe embedded microprocessor which consumes less power but matches the sustained performance of the Transmeta Crusoe TM5600. ARGUS' density comes at the expense of mechanical parts (namely local disk). For perspective, 240 nodes in Green Destiny fill a single rack (about 25 ft^3); ARGUS can fit 128 nodes in 3.3 ft^3. This diskless design makes ARGUS more dense and mobile yet less suitable for applications requiring significant storage.

4. Methodology

As ARGUS and Green Destiny are similar, we use the total cost of ownership (TCO) metrics proposed by Feng et al [3] as the basis of evaluation. For

completeness, we also evaluate our system using traditional performance metrics of benchmarks.

4.1 Cost, Power, and Space Metrics

$$TCO = AC + OC \quad (1)$$
$$AC = HWC + SWC \quad (2)$$
$$OC = SAC + PCC + SCC + DTC \quad (3)$$

TCO refers to all expenses related to acquisition, maintaining and operating the computing system within an organization. Equations (1-3) provide TCO components including acquisition cost (AC), operations cost (OC), hardware cost (HWC), software cost (SWC), system-administration cost (SAC), power-consumption cost (PCC), space-consumption cost (SCC) and downtime cost (DTC). The ratio of total cost of ownership (TCO) and the performance (GFLOPS or giga-floating-point-operations per second) is designed to quantify the effective cost of distributed system reliability.

Since both ARGUS and Green Destiny use low-power processors, the performance to power ratio (GFLOPS/watt) can be used to quantify power efficiency. The rate of heat dissipation is proportional to power consumption and temperature difference between a system and its ambient environment. Therefore, lower power consumption means lower rate of heat dissipation, lower system working temperature and much higher system stability and reliability.

As both ARGUS and Green Destiny provide small form factors relative to traditional high-end systems, the performance to space ratio (GFLOPS/ft^2 for footprint and GFLOPS/ft^3 for volume) can be used to quantify density. Feng et al propose footprint as the metric of computing density [3]. While ARGUS performs well in this regard for a very large system, we argue it is more precise to compare volume. We provide both measurements in our results.

4.2 Performance Metrics

We use an iterative benchmarking process to determine the system performance characteristics of the ARGUS prototype for general comparison to a performance/cost design (i.e. Beowulf) and to identify future design improvements. Benchmarking is performed at two levels:

1) **Micro-benchmarks:** Using several micro benchmarks such as LMBENCH [7], MPPTEST [8], NSIEVE [9] and Livermore LOOPS [10], we provide detailed performance measurements of the core components of the prototype: CPU, memory subsystem and communication subsystem.
2) **Kernel application benchmarks**: We use LINPACK [11] and the NAS Parallel Benchmarks [12] to quantify performance of key application kernels in high performance scientific computing. Performance bottlenecks in these applications may be explained by measurements at the micro-benchmark level.

Since we do not have access to the Green Destiny system for direct benchmark comparisons, we use an on-site 32-node Beowulf cluster called DANIEL. Each node on DANIEL is a 933MHZ Pentium III processor with 1 Gigabyte memory running Red Hat Linux 8.0. The head node and all slave nodes are connected with two 100M Ethernet switches. We expect DANIEL to out-perform ARGUS generally, though our results normalized for clock rate (i.e. using machine clock cycles instead of seconds) show performance is comparable given DANIEL is designed for performance/cost and ARGUS for performance/space. We provide results in seconds and cycles though the Green Destiny literature uses cycle-to-cycle comparisons only.

For direct measurements, we use standard UNIX system calls and timers when applicable as well as hardware counters if available. Whenever possible, we use existing, widely-used tools (e.g. LMBENCH) to obtain measurements. All measurements are the average or minimum results over multiple runs at various times of day to avoid outliers due to local and machine-wide perturbations.

5. Results

5.1 Cost, Power and Space Metrics

Based on the aforementioned metrics we make direct comparisons between ARGUS, Green Destiny and DANIEL. The results are given in Table 1. Two ARGUS systems are considered: ARGUS64 and ARGUS128. ARGUS64 is the 64-node update of our current prototype with the same Load Module. ARGUS128 is the 128-node update with the more advanced IXIA Application Load Module (ALM) currently available. Space efficiency is calculated by mounting 4 chassis in a single 36U rack (excluding I/O node and Ethernet switches to be comparable to Green Destiny). The LINPACK performance of ARGUS64 is extrapolated from direct measurements on 16-nodes and the performance of ARGUS128 is predicted using techniques similar to Feng et al. as 2×1.3 times the performance of ARGUS64 (each ALM is equipped with 8 1856 MIPS PowerPC with 512MB memory).

All data on the 32-node Beowulf, DANIEL is obtained from direct measurements. There is no direct measurement of LINPACK performance for Green Destiny in the literature. We use its Tree Code performance as reported [4] and the estimated LINPACK performance by Feng [13] (denoted with parenthesis in Table 1) for comparison.

Density. The difference in system footprints comes with their physical configurations. The ARGUS128 is built with a single the IXIA 1600T chassis with 16 blades and each blade contains 8 CPUs. The chassis occupies 44.5×39.9×52 cm^3 (about 0.09 m^3 or 3.3 ft^3). Green Destiny consists of 10 chassis; each chassis contains 24 blades; and each blade has only one CPU. DANIEL includes 32 rack-dense sever nodes and each node has only one CPU. Due to the large difference in system footprints and small difference in single processors performance, ARGUS achieves the highest computing density, 30 times higher than DANIEL, and 3 times higher than Green Destiny.

Cost. We estimated the acquisition cost of ARGUS using prices published by IBM in June 2003. Each PowerPC 750CXe costs less than $50. Considering memory and other components, each ALM Load Module will cost less than $1000. Including software and system design cost, each Load Module could sell $5000~$10000. Assuming the chassis costs another $10,000, the 128-node ARGUS may costs $90K~170K in acquisition cost (AC). Following the same method proposed by Feng et al., the operating cost (OC) of ARGUS is less than $10K. Therefore, we estimate the TCO of ARGUS128 is within the range of $100K~200K. The downtime cost of DANIEL is not included when computing its TCO since it is a research system and often purposely rebooted before and after experiments. The TCO of the 240-node Green Destiny is estimated based on the data of its 24-node system [2].

Table 1: Performance Comparisons under cost, power and space efficiency metrics (For Green Destiny, the first value corresponds to its Tree Code performance; the second value in parenthesis is its estimated LINPACK performance. All other systems use LINPACK performance)

Machine	DANIEL	Green Destiny	ARGUS64	ARGUS128
CPUs	32	240	64	128
Performance (GFLOPS)	17	39 (101)	13	34
Area (ft^2)	12	6	2.5	2.5
TCO ($K)	~100	~350	100~150	100~200
Volume(ft^3)	50	30	3.3	3.3
Power(kW)	2	5.2	1	2
GFLOPS/proc	0.53	0.16 (0.42)	0.20	0.27
GFLOPS Per Chassis	0.53	3.9	13	34
TCO Efficiency (GFLOPS/K$)	0.17	0.11 (0.29)	0.08~0.13	0.17~0.34
Computing Density (GFLOPS/ft^3)	0.34	1.3 (3.3)	3.9	10.3
Space Efficiency (GFLOPS/ft^2)	1.4	6.5 (16.8)	20.8	54.4
Power Efficiency (GFLOPS/ft^3)	8.5	7.5 (19.4)	13	17

Though TCO is suggested as a better metrics than acquisition cost, the estimation of downtime cost (DTC) is subjective. Though, these three systems have similar TCO's, Green Destiny and ARGUS have larger acquisition costs than DANIEL. System design cost is high in both cases since the design cost has not been amortized over the market size – which would effectively occur as production matures.

Power. All these systems have their own internal cooling systems that contribute a large percentage (about 30%) of the power consumption. The power efficiencies (including cooling costs) for all systems are comparable. Table 1 shows ARGUS128 is twice as efficient as DANIEL and about the same as Green Destiny. This observation is surprised us. We expected ARGUS to fair better against Green Destiny in power efficiency. However upon further investigation we suspect either 1) the ARGUS cooling system is less efficient (or works harder given the processor density) or 2) our use of peak power consumption on ARGUS compared to average consumption on Green Destiny is unfair or 3) the Green Destiny LINPACK estimates provided in the literature are overly optimistic. In any case, our results indicate power efficiency should be revisited in succeeding designs, though the results are respectable, particularly given the processor density.

5.2 Performance Results

Due to varying numbers of processors across systems, performance/chassis and performance/processor are used as the basis of performance comparison. A single RLX System 324 chassis with 24 blades from Green Destiny delivers 3.9 GFLOPS computing capability for Tree Code benchmark. While a single IXIA 1600T with 16 Load Modules gives 34 GFLOPS for LINPACK benchmark. But when performance/processor is compared, DANIEL is the best and ARGUS performs worst. The reason is that ARGUS has poor performance on double MUL operation (as we seen in next section) which happens to the dominating operations in LINPACK. ARGUS will perform better with integer and single precision float operations dominated benchmark and applications. Green Destiny out performs ARGUS additionally since designers were able to optimize the floating point translation of the Transmeta processor which is not possible on the IBM Power PC architecture used in ARGUS.

5.2.1 Microbenchmark Results

Memory hierarchy performance (latency and bandwidth) is measured using the lat_mem_rd and bw_mem_xx tools in the LMBENCH suite. The results are summarized in Table 2. DANIEL uses its high-power, high-profile off-the-shelf Intel technology to outperform ARGUS at each level in the memory hierarchy in raw performance (time). Normalizing with respect to cycles however, shows how clock rate partially explains the disparity. The resulting "relative performance" between

DANIEL and ARGUS is more promising. ARGUS performs 50% better than Daniel at L1 level, 6% better at main memory level, but much worse at L2 level. Increasing the clock rate of the PowerPC processor and the L2 implementation in ARGUS would improve raw performance considerably.

Table 2: Memory Subsystem Performance

Parameters	ARGUS	DANIEL
CPU Clock Rate	600MHz	922MHz
Clock Cycle Time	1.667ns	1.085ns
L1 Data Cache Size	32KB	16KB
L1 Data Cache Latency	3.37ns≈2 cycles	3.26ns≈3 cycles
L2 Data Cache Size	256KB	256KB
L2 Data Cache Latency	19.3ns≈12cycles	7.6ns ≈ 7 cycles
Memory Size	128MB	1GB
Memory Latency	220ns≈132 cycles	153ns≈141 cycles
Memory Read Bandwidth	146~2340MB/s	514~3580MB/s
Memory Write Bandwidth	98~2375MB/s	162~3366MB/s

Table 3 shows the instruction throughput for ARGUS and DANIEL profiled with the LMBENCH suite. Throughput is computed using

$$Throughput = \frac{parallelism}{cycles} \cdot clock\_rate \qquad (4)$$

"Parallelism" captures the pipelined functional unit capacity of the processor and reflects the achievable throughput of a given instruction type under ideal conditions. The resulting "throughput" will only be realized in a real code when a single type of instruction dominates the instruction stream executed by the processor and dependencies allow for near-optimal instruction-level parallelism. The results show that: 1) the integer performance of ARGUS typically outperforms its floating point performance; 2) While Intel architectures perform similar for float and double operations, this is not the case for the embedded PowerPC on ARGUS. Once "normalized" CPU performance is compared, ARGUS performs better than DANIEL for integer ADD/DIV/MOD, float ADD/MUL and double ADD instructions, but worse for integer MUL and double DIV instructions.

Table 3: Profile of the Execution Unit with LMBENCH (Cycles: the execution time (CPU clock cycles) of an instruction type; P: parallelism; MIPS: throughput of an instruction type; I: Integer; F: Single precision floating point; D: Double precision floating point)

Instruction	ARGUS			DANIEL		
	Cycles	P	MIPS	Cycles	P	MIPS
I-BIT	1	1.5	900	1	1.93	1771
I-ADD	1	2.0	1200	1	1.56	1393
I-MUL	2	1.0	300	4	3.81	880
I-DIV	20	1.0	30	39	1.08	36
I-MOD	24	1.0	25	42	1.08	24
F-ADD	3	3.0	600	3	2.50	764
F-MUL	3	3.0	600	5	2.50	460
F-DIV	18	1.0	33	23.6	1.08	42
D-ADD	3	3.0	600	3	2.50	764
D-MUL	4	2.0	300	5	2.50	460
D-DIV	32	1.0	19	23.6	1.08	42

The performance of message communication is critical to overall parallel system performance. We measured message communication latency and bandwidth with the MPPTEST tool available in the MPICH distribution. Result shows that ARGUS performance is slightly worse (yet comparable) to DANIEL in absolute value. The MPI point-to-point latency on ARGUS is 104 microseconds (about 62,000 CPU cycles); on DANIEL it is 87 microseconds (about 80,000 CPU cycles). Both use 10/100 Mbps Ethernet so this is somewhat unexpected. However, further analyses shows ARGUS has a larger overhead for message injection (as message size approaches typical packet size) than DANIEL – most likely due to the memory hierarchy disparity already mentioned.

For further comparison, we measured the performance of two additional sequential benchmarks: NSIEVE and Livermore Loops. NSIEVE is a sieve of Eratosthenes program that varies array sizes to quantify the performance of integer operations. The NSIEVE benchmark results show that for small array sizes, ARGUS with high MIPS 980 beats DANIEL with high MIPS 945. However, as array sizes increase, the relative performance of ARGUS decreases more than DANIEL. This again confirms the disparity between these machines in L2 cache performance. The performance results from Livermore loops are summarized in Table 4. Although DANIEL achieves 1.5~2 times higher MFLOPS than ARGUS for in most cases, ARGUS achieves the best, worst-case execution time for this benchmark. For "relative performance" on this benchmark, these two systems are very similar.

Table 4: Livermore Loops Performance (REL. P.: Relative performance, obtained by divide the MFLOPS with CPU clock rate)

	ARGUS		DANIEL	
	MFLOPS	REL. P.	MFLOPS	REL. P.
Maximum Rate	731.5	1.22	1281.9	1.37
Minimum Rate	46.2	0.08	20.0	0.02
Standard Dev	133.8	0.22	208.5	0.22

5.2.2 Parallel Performance

The ARGUS prototype architecture can execute both commercial and scientific applications. In this paper, we focus on scientific applications and provide results for LINPACK [11] benchmark.

LINPACK is arguably the most widely used benchmark for scientific applications and its measurements form the basis for the Top500 list [14] of most powerful computers in the world. HPL, a parallel version of the linear algebra subroutines in LINPACK that that solves a (random) dense linear system in double precision (64 bits) arithmetic on distributed-memory computers is used for measurements. HPL provides the ability to scale workloads for better performance by adjusting array sizes. To ensure good performance, we compiled and installed the BLAS libraries with the aid of ATLAS (Automatically Tuned Linear Algebra Software). Table 5 shows the LINPACK benchmark results on the

16-node ARGUS prototype. The prototype achieves 3.4 GFLOPS, about 210 MFLOPS each node or 70% peak throughput of "double MUL" operations.

Table 5: LINPACK Benchmark Results on ARGUS

NP	Problem Size	GFLOPS	GFLOPS/proc	Speedup
1	3000	0.297	0.297	1.00
2	3000	0.496	0.248	1.67
4	5000	0.876	0.219	2.95
8	8000	1.757	0.221	5.91
16	12000	3.393	0.212	11.42

To better identify the parallel performance trends, we also studied the scalability of ARGUS under the strong scaling rules (fixed problem size) using NAS parallel benchmark. The scaling curves show that ARGUS resembles most Ethernet-based Beowulf-cluster and its parallel scalability is mostly limited by its quality and interconnections.

6. Summary and Conclusions

In our work, we implemented the ARGUS prototype as a new approach to cluster computing that uses the aggregate processing elements on network analysis Load Modules for parallel computing. Our work shows that this architecture has advantages such as high scalability, small footprint, reduced power, high availability, ultra-high density and limited mobility.

ARGUS achieves much higher computing efficiency than another high density computer, Green Destiny but has similar power efficiency. The differences of ARGUS and Green Destiny lie in packing more processors on one blade and multiple nodes share one I/O module.

The benchmarking measurements and comparisons with DANIEL indicate that the current ARGUS prototype has two major performance limitations due to the architectural characteristics of embedded PowerPC processor: L2 cache latency and hardware support for double precision.

We intend to extend our system prototype for a more comprehensive exploration of the results herein taking advantage of advancements (e.g. newer cards, gigabyte Ethernet, etc.). For example, the communication overhead on the processing node should and could be improved through hardware and software tuning for MPI. Also, results from a larger prototype with high-end interconnect would allow more comprehensive scalability analysis.

Acknowledgements

This work was supported in part by Ixia, the NSF CCF-#0347683 and DOE DE-FG02-04ER25608 respectively. We would like to thank Eran Karoly, Dan Kegel, Jan Olderdissen, and Tom Miller of Ixia for all of their support in securing and understanding the hardware used in this project. We would also like to thank Duncan Buell for his help with the FPGA component to this work and access to Daniel.

References:

[1] R. A. Games, "Trends in HPC and HPEC Convergence," *proceedings of 2002 Workshop on High Performance Embedded Computing*, 2002.

[2] W. Feng, M. Warren, and E. Weigle, "The Bladed Beowulf: A Cost-Effective Alternative to Traditional Beowulfs," proceedings of *IEEE International Conference on Cluster Computing (CLUSTER'02)*, Chicago, Illinois, 2002.

[3] W. Feng, M. Warren, and E. Weigle, "Honey, I Shrunk the Beowulf!," proceedings of 2002 *International Conference on Parallel Processing (ICPP'02),* Vancouver, B.C., Canada, 2002.

[4] M. S. Warren, E. H. Weigle, and W.-C. Feng, "High-Density Computing: A 240-Processor Beowulf in One Cubic Meter," proceedings of *IEEE/ACM SC2002 Conference*, Baltimore, Maryland, 2002.

[5] Ixia company, "IXIA Product Catalog", available at: http://www.ixiacom.com/library/catalog/ixia_catalog.pdf

[6] IBM, "IBM PowerPC 750CX/750CXe RISC Microprocessor User's Manual," 2002.

[7] L. McVoy and C. Staelin, "lmbench: Portable tools for performance analysis," proceedings of *USENIX 1996 Annual Technical Conference*, San Diego, CA, 1996.

[8] W. Gropp and E. Lusk, "Reproducible Measurements of MPI Performance," proceedings of *PVM/MPI '99 User's Group Meeting,* 1999.

[9] A. Aburt, "PDS: NSIEVE Version 1.2.", [online] http://www.netlib.org/performance/html/nsieve.intro.html

[10] F. H. McMahon, "The Livermore Fortran Kernels: A computer Test of Numerical Performance Range," Performance Evaluation of Supercomputers, vol. 4, pp. 143-186, 1988.

[11] J. J. Dongarra, J. R. Bunch, C. B. Moller, and G. W. Stewart, LINPACK User's Guide. Philadelphia, PA: SIAM, 1979.

[12] D. Bailey, T. Harris, W. Saphir, R. van der Wijngaart, A. Woo, and M. Yarrow, "The NAS Parallel Benchmarks 2.0," NASA Ames Research Center Technical Report #NAS-95-020 December 1995.

[13] W. Feng, "Making a Case for Efficient Supercomputing," ACM Queue, 1(7), pp. 54-64, 2003.

[14] U. Tennessee, U. Manheim, and NERSC, "Top 500 Supercomputer list," in 18th International Supercomputer Conference, 2003.

[15] N.R. Adiga, and et al, "An overview of the BlueGene/L Supercomputer," proceedings of *IEEE/ACM SC2002 Conference*, Baltimore, Maryland, 2002.