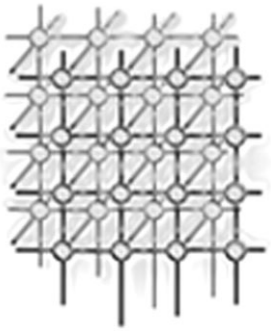
CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2006; (in press)
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe.1040



The Argus prototype: aggregate use of load modules as a high-density supercomputer

Xizhou Feng[1], Rong Ge[2] and Kirk W. Cameron[2,*,†]

[1]*Department of Computer Science and Engineering, University of South Carolina, Columbia, SC 29208, U.S.A.*
[2]*Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, U.S.A.*

SUMMARY

This paper describes the ARGUS prototype, a high-density, low-power supercomputer built from an IXIA network analyzer chassis and load modules. The prototype is configured as a diskless distributed system that is scalable to 128 processors in a single 9U chassis. The entire system has a footprint of 0.25 m^2 (2.5 ft^2), a volume of 0.09 m^3 (3.3 ft^3) and maximum power consumption of less than 2200 W. We compare and contrast the characteristics of ARGUS against various machines including our on-site 32-node Beowulf and LANL's Green Destiny. Our results show that the computing density (Gflops ft^{-3}) of ARGUS is about 30 times higher than that of the Beowulf and about three times higher than that of Green Destiny with a comparable performance. Copyright © 2006 John Wiley & Sons, Ltd.

Received 21 April 2005; Revised 8 September 2005; Accepted 24 September 2005

KEY WORDS: performance evaluation; high-density computing; cluster system; parallel and distributed system; parallel computer architecture

1. INTRODUCTION

Mainstream high-performance computing systems often consist of clusters of symmetric multiprocessors (SMP) communicating across fast interconnects. Computing resources may be for a specific purpose (e.g. Earth simulator) or general purpose (e.g. a 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.

*Correspondence to: Kirk W. Cameron, Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, U.S.A.
†E-mail: cameron@cs.vt.edu

Contract/grant sponsor: National Science Foundation
Contract/grant sponsor: Department of Energy
Contract/grant sponsor: IXIA Corporation

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). These machines offer exceptional speed (and often guaranteed performance) for certain applications. However, design constraints including performance, power and space make 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. Although the original IXIA machine is designed for a single task, we have created a configuration that provides general-purpose, high-end parallel processing in a Linux environment. Our system provides computational power surpassing Green Destiny [1,2] (another low-power supercomputer) while decreasing volume by a factor of three.

2. SYSTEM DESIGN

Figure 1 is a detailed diagram of the prototype architecture we call ARGUS. This architecture consists of four sets of separate components: the IXIA chassis, the IXIA load modules, the multi-port fast ethernet switch and a network file system (NFS) server.

The chassis contains a power supply and distribution unit, a 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 (mainly used for system management being too slow for message transfer). Each load module provides up to eight RISC processors (called port processors) in a dense form factor and each processor has its own operating system, cache (L1 and L2), main memory and network interface. In addition, field programmable gate array (FPGA) elements on each load module aid real-time analysis of network traffic. Although 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 that the 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 the IX server. An external Linux machine running a 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 four LM1000TXS4 load modules (see http://www.ixiacom.com/library/catalog/ for specification) [3] configured as a 16-node distributed memory system, i.e. each port processor is considered as an individual node.

Another option is to configure each load module as an SMP node. This option requires use of the IxBus between load modules. The IxBus (and the PowerPC 750CXe processor) does not maintain cache coherence and has a limited bandwidth. Thus, this option was eliminated from consideration at an early stage since software-driven cache coherence will drastically limit performance. We opted to exchange data between all processors through the Ethernet connection. Hence, one recommendation

DOI: 10.1002/cpe

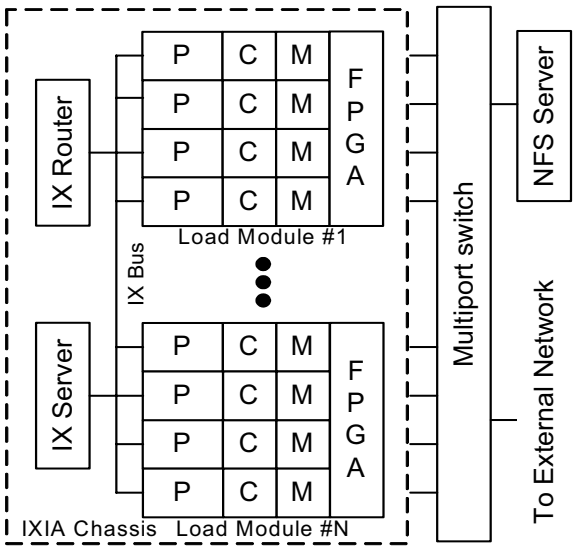


Figure 1. The hardware architecture of ARGUS prototype. Up to 16 load modules are supported in a single IXIA 1600T chassis. A single bus interconnects modules and the chassis PC while external disks and the cluster front-end are connected via an Ethernet switch (P, processor; C, cache; M, memory).

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 [4] and each processor has one 128 MB memory module and one network port with an auto-negotiating 10/100/1000 Mbps copper ethernet interface. The 1392 MIPS PowerPC 750CXe CPU employs 0.18 μm CMOS copper technology, running at 600 MHz with a 6.0 W typical power dissipation. This CPU has independent on-chip 32 KB, eight-way set associative, physically addressed caches for instructions and data. The 256 KB L2 cache is implemented with on-chip, two-way set-associative memories and a 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 six 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 (i.e. Message Passing Interface (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 (the MPI implementation by Argonne National Lab and Mississippi State University) and a series of benchmarks have been built and installed on ARGUS.

DOI: 10.1002/cpe

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:

(1) performance: these are traditional high-performance systems (e.g. ASCI Q) where performance (Gflops) is the absolute priority;
(2) cost: these are systems built to maximize the performance/cost ratio (Gflops $\$^{-1}$) using commercial-off-the-shelf components (e.g. Beowulf);
(3) power: these systems are designed for reduced power (Gflops W^{-1}) to improve reliability (e.g. Green Destiny) using low-power components;
(4) density: these systems have specific space constraints requiring integration of components in a dense form factor with specially designed size and shape (e.g. Green Destiny) for a high performance/volume ratio (Gflops ft^{-3}).

Although high performance systems are still the most common system in the HPC community; low-cost, low-power, low-profile and high-density systems are emerging. BlueGene/L (IBM) [5] and Green Destiny (LANL) are two examples that are designed under cost, power and space constraints.

From a design priority and capability perspective, ARGUS is the most comparable supercomputer to Green Destiny. Both systems rely on components targeted at commodity markets and prioritize reliability (i.e. power consumption) and space constrains. Both systems can be categorized as Beowulf clusters but differ from regular Beowulf clusters in their reliability and density. However, ARGUS and Green Destiny differ in several areas.

(1) Green Destiny is built on RLX ServerBlades^{TM} that result in a relatively small form factor. In contrast, the ARGUS is built on IXIA load modules whose design prioritizes space but provides a general-purpose functionality unusual in space-constrained systems.
(2) Green Destiny uses the Transmeta Crusoe TM5600 CPU for low-power and high-density. ARGUS uses the PowerPC 750CXe embedded microprocessor that consumes less power but matches the sustained performance of the Transmeta Crusoe TM5600.
(3) In contrast with Green Destiny, which combines server hardware, such as CPU, memory, and the network controller into a single expansion card, ARGUS achieves much higher density at the expense of mechanical parts (namely a local disk) and multiple processors on each load module (or blade). 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 in many aspects, we use the total cost of ownership (*TCO*) metrics proposed by Feng *et al.* [1] as the basis of evaluation. For completeness, we also evaluate our system using traditional performance metrics of benchmarks.

DOI: 10.1002/cpe

4.1. Cost, power and space metrics

TCO refers to all expenses related to acquisition, maintaining and operating the computing system within an organization:

$$TCO = AC + OC \tag{1}$$

$$AC = HWC + SWC \tag{2}$$

$$OC = SAC + PCC + SCC + DTC \tag{3}$$

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 *TCO* and the performance (Gflops) is designed to quantify the effective cost of distributed system.

According to a formula derived from Arrhenius law, component life expectancy decreases 50% for every 10 °C (18 °F) temperature increase. Since system operating temperature is roughly proportional to its power consumption, a lower power consumption implies a longer component life expectancy and a lower system failure rate. Since both ARGUS and Green Destiny use low-power processors, the performance to power ratio (Gflops W^{-1}) can be used to quantify power efficiency. A high Gflops W^{-1} ratio implies a lower power consumption for the same number of computations and hence a lower system working temperature and much higher system stability and reliability (i.e. a lower component failure rate).

Since 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 computing density. Feng *et al*. propose the footprint as the metric of computing density [1]. 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 a general comparison to a performance/cost design (i.e. Beowulf) and to target future design improvements. Benchmarking is performed at two levels.

- *Micro-benchmarks.* Using several micro-benchmarks such as LMBENCH [6], MPPTEST [7], NSIEVE [8] and Livermore LOOPS [9] we provide detailed performance measurements of the core components of the prototype CPU, memory subsystem and communication subsystem.
- *Kernel application benchmarks.* We use LINPACK [10] and the NAS Parallel Benchmarks (NPBs) [11] 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.

For direct performance comparisons, we use an on-site 32-node Beowulf cluster called DANIEL. Each node on DANIEL is a 933 MHz Pentium III processor with 1 GB memory running Red Hat Linux 8.0. The head node and all slave nodes are connected with two 100 Mbps Ethernet switches. In general, we expect DANIEL to out-perform ARGUS, although normalizing our results for clock rate

DOI: 10.1002/cpe

Table I. 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 the estimated LINPACK performance. All other systems use LINPACK performance.) 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.

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 ($1000)	∼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/$1000)	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

(i.e. using machine clock cycles instead of seconds) shows that performance is comparable given that DANIEL is designed for performance/cost and ARGUS for performance/space.

In 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 in order to avoid outliers due to local and machine-wide perturbations.

5. EXPERIMENTAL RESULTS

5.1. Measured cost, power and space metrics

We make direct comparisons between ARGUS, Green Destiny and DANIEL, based on the aforementioned metrics. The results are given in Table I. 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 (ALM1000T8) currently available [3]. Each ALM1000T8 load module has eight 1856 MIPS PowerPC processors with a gigabit Ethernet interface and 1 GB memory per processor. Space efficiency is calculated by mounting four chassis in a single 36U rack (excluding I/O node and Ethernet switches so as to be comparable to Green Destiny). The LINPACK performance of ARGUS64 is

DOI: 10.1002/cpe

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.

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 both the tree code performance as reported in [2] and the estimated LINPACK performance by Feng [12] for comparison denoted with parentheses in Table I.

We estimated the acquisition cost of ARGUS using prices published by IBM in June 2003 and industry practice. 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 for $5000–10 000. Assuming the chassis costs another $10 000, the 128-node ARGUS may cost $90 000–170 000 in *AC*. Following the same method proposed by Feng *et al*., the *OC* of ARGUS is less than $10 000. Therefore, we estimate the *TCO* of ARGUS128 is below $200 000. 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.

Although *TCO* is suggested as a better metric than acquisition cost, the estimation of *DTC* is subjective and the acquisition cost is the largest component of *TCO*. Although, these three systems have similar *TCO* performances, Green Destiny and ARGUS have larger acquisition cost than DANIEL due to their initial system design cost. System design cost is high in both cases since the design cost has not been amortized over the market size—this would effectively occur as production matures.

The ARGUS128 is built with a single IXIA 1600T chassis with 16 blades where each blade contains eight CPUs. The chassis occupies $44.5 \times 39.9 \times 52$ cm^3 (about 0.09 m^3 or 3.3 ft^3). Green Destiny consists of 10 chassis with each chassis contains 10 blades and each blade has only one CPU. DANIEL includes 32 rack-dense server nodes and each node has one CPU.

Due to the large difference in system footprints (50 ft^3 for DANIEL, 30 ft^3 for Green Destiny and 3.3 ft^3 for ARGUS) and relatively small differences in single processor performance (711 Mflops for DANIEL, 600 Mflops for Green Destiny and 300 Mflops for ARGUS), ARGUS has the highest computing density, 30 times higher than DANIEL and three times higher than Green Destiny. As space efficiency is almost equivalent to computing density, we observe the same facts as above.

Table I shows ARGUS128 is twice as efficient as DANIEL and has a similar efficiency to Green Destiny. This observation contradicts our expectations that ARGUS should fair better against Green Destiny in power efficiency. However, on further investigation, we suspect that (1) the ARGUS cooling system is less efficient (or works harder given the processor density), (2) our use of peak power consumption on ARGUS compared with average consumption on Green Destiny is unfair, (3) the Green Destiny LINPACK projections (not measured directly) provided in the literature are overly optimistic or (4) some combination thereof. In any case, our results indicate power efficiency should be revisited in succeeding designs although the results are respectable, particularly given the processor density.

5.2. Results of traditional benchmarks

A single RLX System 324 chassis with 24 blades from Green Destiny delivers 3.6 Gflops computing capability for the tree code benchmark. A single IXIA 1600T with 16 load modules achieves 34 Gflops for the LINPACK benchmark. Due to the varying number of processors in each system, performance

DOI: 10.1002/cpe

Table II. Memory subsystem performance.

Parameters	ARGUS	DANIEL
CPU clock rate	600 MHz	922 MHz
Clock cycle time	1.667 ns	1.085 ns
L1 Data cache size	32 KB	16 KB
L1 Data cache latency	3.37 ns $\approx$ 2 cycles	3.26 ns $\approx$ 3 cycles
L2 Data cache size	256 KB	256 KB
L2 Data cache latency	19.3 ns $\approx$ 12 cycles	7.6 ns $\approx$ 7 cycles
Memory size	128 MB	1 GB
Memory latency	220 ns $\approx$ 132 cycles	153 ns $\approx$ 141 cycles
Memory read bandwidth	146–2340 MB s^{-1}	514–3580 MB s^{-1}
Memory write bandwidth	98–2375 MB s^{-1}	162–3366 MB s^{-1}

per chassis and performance per processor are used for performance comparisons. Table I shows DANIEL achieves the best performance per processor and ARGUS achieves the worst. ARGUS has poor performance on double MUL operation (discussed in the next section) that dominates operations in LINPACK. ARGUS performs better for integer and single precision float operations. Green Destiny outperforms ARGUS on multiply operations since designers were able to work with Transmeta engineers to optimize the floating point translation of the Transmeta processor.

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 II. DANIEL, using its high-power, high-profile off-the-shelf Intel technology, outperforms ARGUS at each level in the memory hierarchy in raw performance (time). However, normalizing with respect to cycles shows how the clock rate partially explains the disparity. The resulting 'relative performance' between DANIEL and ARGUS is more promising. Argus performs 50% better than Daniel at the L1 level, 6% better at the main memory level but much worse at the L2 level. Increasing the clock rate of the PowerPC processor and the L2 implementation in ARGUS would improve raw performance considerably.

IPC is the number of *instructions* executed per *clock cycle*. Throughput is the number of instructions executed per second (or *IPC* $\times$ *clock cycle*). Peak throughput is the maximum throughput possible on a given architecture. Peak throughput is only attained when the ideal *IPC* (optimal instruction-level parallelism) is sustained on the processor. Memory accesses, data dependencies, branching and other code characteristics limit the achieved throughput on the processor. Using microbenchmarks, we measured the peak throughput for various instruction types on the machines under study.

Table III shows the results of our throughput experiments. Integer performance typically outperforms floating point performance on ARGUS. For DANIEL (the Intel architecture) floating point (F-xxx in Table III) and double (D-xxx in Table III) performances are comparable for ADD, MUL and DIV respectively. This is not true for ARGUS where F-MUL and D-MUL are significantly different as observed in our LINPACK measurements. We expect the modified version of the PowerPC architecture

DOI: 10.1002/cpe

Table III. Instruction performance with LMBENCH (*IPC*, instructions per clock; MIPS, millions of instructions per second; I, integer; F, single precision floating point; D, double precision floating point).

	ARGUS			DANIEL		
Instruction	Cycles	*IPC*	MIPS	Cycles	*IPC*	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

(with an additional floating point unit) present in IBM BlueGene/L to equalize the performance difference with the Intel architecture in future systems. CPU throughput measurements normalized for clock rates (MIPS) show that 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.

The performance of message communication is critical to overall parallel system performance. We measured network communication latency and bandwidth with the MPPTEST tools available in the MPICH distribution. Results show that ARGUS performance is slightly worse yet comparable to DANIEL. MPI point-to-point latency is 104 μs (about 62 387 CPU cycles) on ARGUS and 87 μs (about 80 184 CPU cycles) on DANIEL. Both systems use 10/100 Mbps Ethernet switches so this is somewhat expected. However, we observed a larger message injection overhead on ARGUS as message size approaches typical packet size. This is most likely due to the memory hierarchy disparity already described.

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. Livermore loops is a set of 24 DO-loops extracted from operational code used at the Lawrence Livermore National Laboratory.

The NSIEVE benchmark results show that for a small array size ARGUS has a higher MIPS rating (980 MIPS) than DANIEL (945 MIPS). However, as array sizes increase, the relative performance of ARGUS against DANIEL decreases. This reflects the differences in L2 cache performance between ARGUS and DANIEL.

The performance results from Livermore loops are summarized in Table IV. We observe DANIEL achieves 1.5–2 times higher Mflops rating than ARGUS for most statistical values and ARGUS achieves the best worst-case execution time for this benchmark. For instance, in real time codes where worst-case performance must be assumed, ARGUS may be a better choice. However, examining

DOI: 10.1002/cpe

Table IV. Livermore loops performance (NORM, normalized performance, obtained by dividing the Mflops by CPU clock rate).

	ARGUS		DANIEL	
	Mflops	NORM	Mflops	NORM
Maximum rate	731.5	1.22	1281.9	1.37
Quartile Q3	225.0	0.38	377.6	0.40
Average rate	174.5	0.29	278.9	0.30
Geometric mean	135.5	0.23	207.2	0.22
Median Q2	141.6	0.24	222.2	0.24
Harmonic mean	106.6	0.18	133.6	0.14
Quartile Q1	66.4	0.11	132.6	0.14
Minimum rate	46.2	0.08	20.0	0.02
Standard deviation	133.8	0.22	208.5	0.22
Average efficiency	18.52%		16.16%	
Mean precision (digits)	6.24		6.35	

Table V. Linpack benchmark results on ARGUS.

NP	Problem size	Gflops	Gflops/processor	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	12 000	3.393	0.212	11.42

performance normalized for clock rates (NORM) on this benchmark, the two systems perform similarly.

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 two benchmark suites: LINPACK [10] and the NPBs [11]. Since we have already established the performance difference between ARGUS and DANIEL for single node (see the previous section), we will only discuss the parallel performance of ARGUS.

LINPACK is arguably the most widely used benchmark for scientific applications and its measurements form the basis for the Top 500 list [13] of fastest supercomputers in the world. Our measurements use HPL, a parallel version of the linear algebra subroutines in LINPACK that solve a (random) dense linear system in double precision (64-bit) arithmetic on distributed-memory computers. 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

DOI: 10.1002/cpe

Table VI. NAS parallel benchmark results on ARGUS.

	Performance (MOP s^{-1})		
CODE	NP = 1	NP = 4	NP = 16
CG	19.61	46.04	88.12
EP	1.69	6.75	24.08
IS	4.06	3.62	18.02
LU	48.66	188.24	674.62
MG	45.50	84.51	233.36
BT	40.04	131.76	436.29
SP	28.72	90.99	299.71

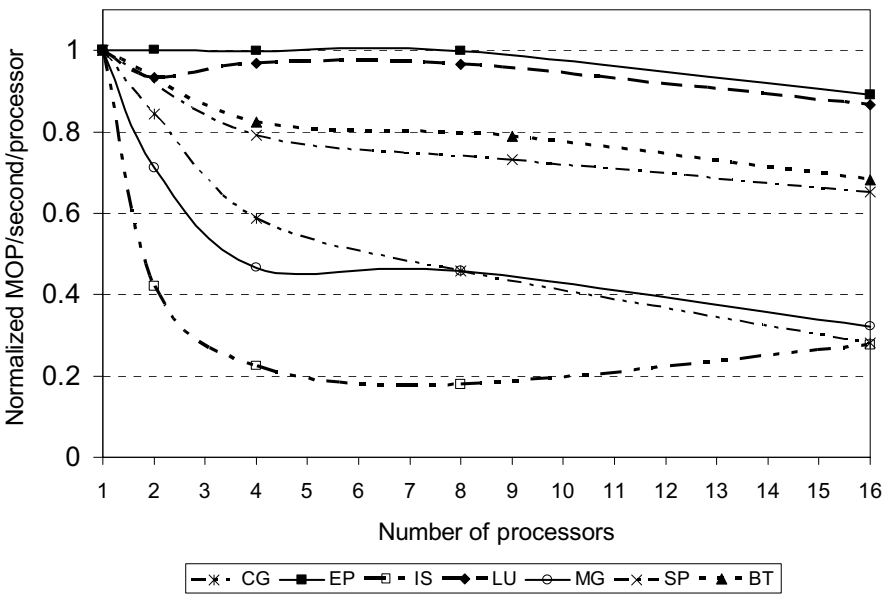


Figure 2. The scalability of NPBs on ARGUS. The curves show the strong scaling of NPB 2.4.1 Class *W* at fixed working set size.

automatically tuned linear algebra software (ATLAS). Table V shows the LINPACK benchmark results on the 16-node ARGUS prototype. The prototype achieves 3.4 Gflops, about 210 Mflops at each node or 70% peak throughput of 'double MUL' operations.

The NPB are a set of eight programs designed to help evaluate the performance of parallel supercomputers. This benchmark suite consists of five application kernels and three pseudo-applications derived from computational fluid dynamics applications. These benchmarks are characterized with different computation/communication ratios described in [11]. The raw performance of NPB 2.4.1 with a problem size of *W* on ARGUS is shown in Table VI. To better identify the

DOI: 10.1002/cpe

performance trends, Figure 2 provides the scalability of ARGUS under strong scaling (i.e. fixed problem size and increasing number of processors).

For 16 nodes, EP and LU show the best scalability. EP should achieve linear speedup since little communication is present. LU achieves super-linear speedup that appears to be levelling off. As working set size remains fixed with an increase in the number of processors, communication is minimal (i.e. strong or fixed-size scaling). Super-linear performance is achieved as the working set gets smaller and smaller on a per node basis.

The curve of IS initially drops but then grows with the number of nodes. These codes stress communication performance. The levelling off of performance indicates the communication costs are not saturating the Ethernet interconnect up to 16 nodes.

The other four curves SP, BT, CG and MG have similar trends but different slopes. The performance of these codes reflect the communication to computation ratio. EP and LU are dominated by computation whereas IS and FT are dominated by communication. The SP, BT, CG and MG codes sit somewhere in between. Trends here are similar (though less pronounced) than the communication-bound codes. SP, BT, CG and MG are more sensitive to the number of nodes as it affects the number of communications. Performance is then likely to move downward with the number of nodes until a plateau is reached prior to network saturation (i.e. similar to the plateau in IS and FT performance). At some later point all of these codes will reach the limits of either the input data set size (Amdahl's law) or the interconnect technology (saturation) where performance will drop drastically again. Our system is too small to observe these types of problems, so this is the subject of future work.

6. SUMMARY AND CONCLUSIONS

ARGUS exemplifies an architectural design with trade-offs between performance, cost, space and power. 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 volumetric footprint, reduced power, high availability and ultra-high processor density.

ARGUS achieves higher computing efficiency than Green Destiny, a comparable system with similar power efficiency. ARGUS is presently capable of packing more processors per blade than Green Destiny, although future versions of both machines will undoubtedly address this issue.

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. Also, the communication overhead on the processing node should and could be improved through system-specific hardware and software tuning of MPI. Furthermore, results from a larger prototype with a faster interconnect would allow more comprehensive scalability analyses.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their insightful comments and Duncan Buell for access to the DANIEL Beowulf. We also thank the National Science Foundation, the Department of Energy and the IXIA Corporation for supporting this work.

DOI: 10.1002/cpe

REFERENCES

1. Feng W, Warren M, Weigle E. The bladed Beowulf: A cost-effective alternative to traditional Beowulfs. *Proceedings of the IEEE International Conference on Cluster Computing (CLUSTER'02)*, Chicago, IL, 2002. IEEE Computer Society Press: Los Alamitos, CA, 2002.
2. Warren MS, Weigle EH, Feng W. High-density computing: A 240-processor Beowulf in one cubic meter. *Proceedings of the 15th IEEE/ACM SC2002 Conference*, Baltimore, MD, 2002. IEEE Computer Society Press: Los Alamitos, CA, 2002.
3. IXIA. IXIA product catalog, 2003. http://www.ixiacom.com/products [2 December 2005].
4. IBM. PowerPC 604e User's Manual, 1998. http://www-3.ibm.com/chips/techlib/techlib.nsf/products/PowerPC_604e_Microprocessor/ [2 December 2005].
5. Adiga N *et al.* An overview of the BlueGene/L supercomputer. *Proceedings of the Supercomputing Conference 2002*, Baltimore, MD, 2003. IEEE Computer Society Press: Los Alamitos, CA, 2003.
6. Mcvoy L, Staelin C. Lmbench: Portable tools for performance analysis. *Proceedings of the USENIX 1996 Annual Technical Conference*, San Diego, CA, 1996. USENIX Association: Berkeley, CA, 1996.
7. Gropp W, Lusk E. Reproducible measurements of MPI performance. *Proceedings of the 6th European PVM/MPI'99 User's Group Meeting*, Barcelona, 1999.
8. Gilbreath J. A high-level language benchmark. *BYTE* 1981; **6**:180–198.
9. Mcmahon FH. The Livermore Fortran kernels: A computer test of numerical performance range. *Technical Report UCRL-53745*, Lawrence Livermore National Laboratory, December 1986.
10. Dongarra J *et al. Linpack User's Guide*. SIAM: Philadelphia, PA, 1979.
11. Bailey D *et al.* The NAS Parallel Benchmarks 2.0. *Technical Report NAS-95-020*, NASA Ames Research Center, December 1995.
12. Feng W. Making a case for efficient supercomputing. *ACM Queue* 2003; **1**:54–64.
13. University of Tennessee, University of Manheim and NERSC. Top 500 supercomputer list. *Proceedings of the 18th International Supercomputer Conference*, Phoenix, AZ, 2003. ACM Press: New York, 2003.

DOI: 10.1002/cpe