FLOPS per Watt: Heterogeneous-Computing's Approach to DoD Imperatives

Dan M. Davis, Robert F. Lucas, Gene Wagenbreth, John J. Tran & Joel Agalsoff
Information Sciences Institute/Univ. of So. Calif.
Marina del Rey, California
{ ddavis, rflucas, genew, jtran, agalsoff }@isi.edu

Thomas D. Gottschalk
Center for Advanced Computing Research
California Institute of Technology
Pasadena, California
tdg@cacr.caltech.edu

ABSTRACT

Electrical power used in computing is increasingly a vital factor in all computing, from laptops to PetaFLOPS. Cost, portability, ecological concerns and hardware life are all negatively impacted by burgeoning power requirements. More than the rest of the world, the U.S. DoD has special requirements to restrain the use of electrical power, ranging from battery life for devices in the field to environmental responsibility for major DoD Supercomputing Centers. The authors will discuss the special insights they have gained into the implementation of one technique, the use of General Purpose Graphics Processing Units as heterogeneous processors and they will further outline the state of the art in the field of power reduction techniques, ranging from IBM's Blue Gene series to Prof. William Dally's Efficient Low-power Microprocessor (ELM) approach and compare and contrast them with the experience of the authors on JFCOM's Joshua, a 256 node, GPGPU enhanced cluster. Using GPGPUs to effectively handle computationally intensive activity "spikes" is manifestly germane to defense computational needs. Quantitatively, the authors will report on three specific aspects their use of GPGPUs: programming environment constraints and opportunities, performance of codes modified in several areas of computational science and the FLOPS per Watt parameter in a wide range of software and hardware configurations. An overview of algorithmic design and implementation strategies will be laid out. Actual working code segments will be discussed and explained, along with the design rationale behind them. The authors' experience in training other DoD users in this technique will assist program managers in scoping training requirements. This data should allow other DoD researchers and users to effectively anticipate the benefits of this approach as far as their own code is concerned and further, it should enable them to effectively evaluate the varying benefits of all of the approaches currently extant.

ABOUT THE AUTHORS

Dan M. Davis is the Director, JESPP Project, Information Sciences Institute (ISI), University of Southern California, and has been active in large-scale distributed simulations for the DoD through two decades. As Assistant Director of the Center for Advanced Computing Research at Caltech, he managed Synthetic Forces Express, a major simulation project. Earlier, he was a Software Engineer on the All Source Analysis System project at JPL and did classified research at Martin Marietta. He was the principal author on the design and proposal of the 256 node, GPGPU-Enhanced cluster at JFCOM. An active duty Marine Cryptologist, he retired as a Commander, USNR. He served as the Chairman of the Coalition of Academic Supercomputing Centers. He received a B.A. and a J.D., both from the University of Colorado in Boulder.

Thomas D. Gottschalk is a Member of the Professional Staff, a Senior Research Scientist at the Center for Advanced Computing Research (CACR), and Lecturer in Physics all at the California Institute of Technology. He has worked at CACR for more than a decade advancing the use of massive parallel computers for simulation. His instructional duties include Statistics and Experimental Design for Caltech Physics Graduate students. Dr. Gottschalk has been active in parallel programming for nearly twenty years, with efforts spanning integrated circuit design, intelligent agent simulations, theater missile defense, and physics modeling. He consults for a number of other organizations, including his work on space-based systems for the Aerospace Corporation. He received a B.S. in Physics from Michigan State University and a Ph.D. in Theoretical Physics from the University of Wisconsin.

Robert F. Lucas is the Director of the Computational Sciences Division of the University of Southern California's Information Sciences Institute (ISI) and a Research Associate Professor in the Department of Computer Science at the same University. There he manages research in computer architecture, VLSI, compilers and other software tools. He has been the principal investigator on the JESPP project since its inception in 2002. Prior to joining ISI, he was the Head of the High Performance Computing Research Department for the National Energy Research Scientific Computing Center (NERSC) at Lawrence Berkeley National Laboratory, the Deputy Director of DARPA's Information Technology Office, and a member of the research staff of the Institute for Defense Analysis's Center for Computing Sciences. From 1979 to 1984 he was a member of the Technical Staff of the Hughes Aircraft Company. Dr. Lucas received his BS, MS, and PhD degrees in Electrical Engineering from Stanford University in 1980, 1983, and 1988 respectively.

Gene Wagenbreth is a Systems Analyst for Parallel Processing at the Information Sciences Institute at the University of Southern California, doing research in the Computational Sciences Division. Prior positions have included Vice President and Chief Architect of Applied Parallel Research and Lead Programmer of Pacific Sierra Research, where he specialized in tools for distributed and shared memory parallelization of Fortran programs. He has also been active in benchmarking, optimization and porting of software for private industry and government labs. He has programmed on CRAY, SGI, Hitachi, Fujitsu, NEC, networked PCs, networked workstations, IBM SP2, as well as conventional machines. He received a BS in Math/Computer Science from the University of Illinois in 1971.

John J. Tran is a former researcher at the Information Sciences Institute/USC who is currently pursuing a doctorate in Computer Engineering from the Viterbi School of Engineering at the University of Southern California. He received both his BS and MS Degrees in Computer Science and Engineering from the University of Notre Dame, where he focused on Object-oriented software engineering, large-scale software system design and implementation, and high performance parallel and scientific computing. He has worked at the Stanford Linear Accelerator Center, Safetopia, and Intel. His current research centers on Linux cluster engineering, effective control of parallel programs, and communications fabrics for large-scale computation.

Joel Agalsoff is a Research Assistant at the Information Sciences Institute, University of Southern California. He is an undergraduate student in Computer Sciences in the Viterbi School of Engineering, University of Southern California, and anticipates receiving a degree in Computer Science and a minor in Psychology in 2010. He has led projects in Multi-Threaded Autonomous Simulation and security. Joel comes from a well-known family in movie production and is an accomplished brass musician.

FLOPS per Watt: Heterogeneous-Computing's Approach to DoD Imperatives

Dan M. Davis, Robert F. Lucas, Gene Wagenbreth, John J. Tran & Joel Agalsoff
Information Sciences Institute/Univ. of So. Calif.
Marina del Rey, California
{ ddavis, rflucas, genew, jtran, agalsoff }@isi.edu

Thomas D. Gottschalk
Center for Advanced Computing Research
California Institute of Technology
Pasadena, California
tdg@cacr.caltech.edu

INTRODUCTION

Wars can be said to be won by power, and in this instance, the power in question is in the form of electricity. This commodity translates into many other factors as well: space, electrically radiated signatures, heat signatures, inhabitability, supply issues and maintenance. U.S. technological ascendency mandates ready and reliable use of electrical devices. While this alone will not win wars, the lack of it may well loose one. Simulations for the DoD make extensive use of computers, so they are likewise bound by the same constraints. This paper will present one technique, the use of General Purpose Graphics Processing Units as heterogeneous processors, and survey other approaches. It will compare and contrast those techniques with the experience of the authors on JFCOM's *Joshua*, a 256 node GPGPU-enhanced cluster used for urban battlespace simulations.

BACKGROUND

The issue addressed here is not a new one. The lack of energy resources and the inability to adequately conserve existing power reserves can arguably be advanced as one of the reasons for the loss of World War II by both major axis powers. On the other hand, a captured German General was secretly recorded as he said, "You mustn't forget that a war has never yet been really decided by a new weapon." General Heinrich Eberbach had just been discussing the German "V Weapons" with his son, Oberleutnant zur See Heinz Eberbach, who was also in the POW camp in England in September of 1944. (Neitzel, 2005). Yet, this nation owes it to its warfighters to send them into battle with every reasonable advantage our technology can confer.

In a more recent conflict, one of the authors, Davis, was a linguist in the Marine Corps in Vietnam, where his bunker was at Con Thien, only one kilometer south of the DMZ. It was the only one that was supplied with a generator that ran day and night to supply the several racks of electronic gear in the operations area. The author can personally attest to the unwelcomed attention that brought as well as the habitability issues caused by that much gear in an enclosed area in a tropical combat area. (Davis, 2005).

In less exotic settings, the need for power conservation is still paramount for cost, maintenance and habitability reasons. These may vary by region, *e.g*. power is on the order of three times as expensive on Maui as it is in Ohio, and by installation, such as with the size and temperature constraints that differ from a High Performance Computing Center to a combat aircraft cockpit. Nevertheless, all of the previously mentioned parameters are important, critical or vital, as the case may be.

Computing itself is amongst the chief of the power sensitive applications. The micro-circuitry now employed in every phase of computing is especially prone to energy constraints, the principal culprit being the need to conduct heat away from the sensitive circuits that are generating their own heat. While calling attention to this concern, it is not the intent of this paper to focus on heat dissipation mitigation techniques.

The techniques reported and surveyed here look to innovative and effective ways to accomplish the same amount of computational power, while using significantly less total energy. The technique studied by the authors is using General Purpose Graphics Processing Units (GPGPUs), to effectively handle computationally intensive activity "spikes". The authors will report on three specific aspects their use of GPGPUs:

- programming environment constraints and opportunities
- performance of codes modified in several areas of computational science
- FLOPS per Watt parameters in a wide range of software and hardware configurations.

A rudimentary overview of algorithmic design and implementation strategies should allow the readers to conceptualize the applicability of this technique to their own situations. To assist in this analysis, an actual working code segment will be discussed and displayed, along with the design rationale behind it. Further, as such new techniques cannot be implemented willy-nilly, the authors feel that their experience in training other DoD users to implement their approach will assist program managers in scoping and justifying training requirements.

Without belaboring the issue further, the authors propose to proceed on the premise that more computational power from less electrical power is a good thing. It is a good that should always be open to further analysis and reconsideration, and there is no thought here of forestalling either of those considerations in other contexts and at other times. As General Eberbach would no doubt have had it, the victor must still win the field by the projection of military power, the quality of its soldiers and the manifest will to win.

GENERAL PURPOSE GRAPHICS PROCESSING UNITS AS COMPUTER ACCELERATORS

Methodology

The method experienced by this team was the use of existing DOD simulation codes on advanced Linux clusters operated by JFCOM. The effort reported herein supplants the previous JFCOM J9 DC clusters with a new cluster enhanced with 64-bit CPUs and nVidia 8800 graphics processing units (GPUs) (Lucas, 2007a).

The initial driver for the Forces Modeling and Simulation (FMS) use of accelerator-enhanced nodes was principally the faster processing of line-of-sight calculations. Envisioning other acceleration targets was easy:

- physics-based phenomenology,
- CFD plume dispersion,
- computational atmospheric chemistry,
- data analysis,

The first experiments were conducted on a smaller code set, to facilitate the programming and accelerate the experimentation. An arithmetic kernel from an MCAE "crash code" (Diniz, 2004) was used as vehicle for a basic "toy" problem. This early assessment of GPU acceleration focused on a subset of the large space of numerical algorithms, factoring large sparse symmetric indefinite matrices. Such problems often arise in Mechanical Computer Aided Engineering (MCAE) applications. It made use of the SGEMM (Single precision GEneral Matrix Multiply) algorithm (Whaley, 1998) from the BLAS (Basic Linear Algebra Subprograms) routines (Dongarra, 1993).

The GPU is a very attractive candidate as an accelerator for computational hurdles such as sparse matrix factorization. Previous generations of accelerators, such as those designed by Floating Point Systems (Charlesworth 1986) were for the relatively small market of scientific and engineering applications. Contrast this with GPUs that are designed to improve the end-user experience in mass-market arenas such as gaming.

In order to get meaningful speed-up using the GPU, it was determined that the data transfer and interaction between the host and the GPU had to be reduced to an acceptable minimum. The reader should be warned that this analysis is not trivial and must always be born in mind when considering the use of GPGPUs.

Implementation Research Results

The sum of the time spent at each level of an elimination tree is shown in Figure 1. The sum from all of the super-nodes at each level is plotted as the red curve. The time spent assembling frontal matrices and stacking their Schur complements is represented by the yellow curve. These are the overheads associated with using the multi-frontal method. The total time spent at each level of the tree when running on the host appears below in blue. The time spent factoring the frontal matrices is the difference between the blue and yellow curves. The time spent at each level of the elimination tree when using the GPU to factor the frontal matrices is brown in the graph below. The difference between the brown curve and the yellow one is the time spent on the GPU. (Lucas, 2007b)

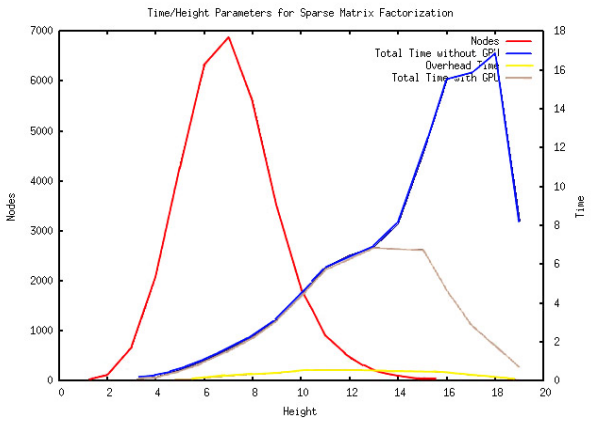

Figure 1. Number of Super-nodes and time spent factoring each level of the elimination tree.

Looking at Figure 1 above, it seems apparent that the GPU is very effective at reducing the time spent factoring the large frontal matrices near the root of the elimination tree. The difference between the brown and blue curves is the cumulative time of 52.5 seconds by which the GPU accelerated the overall factorization. Similar results could be expected from similar codes.

The SGEMM function used in this work was supplied by nVidia. In testing, it was found that it could achieve close to 100 GFLOP/s, over 50% of the peak performance of the nVidia GTS GPU. Thus the efforts were focused on optimizing the functions for eliminating off-diagonal panels (GPUl) and factoring diagonal blocks (GPUd). A more detailed description of this technique can be found in an unpublished paper (Lucas, 2007b).

An exemplar application that has immediate use for a fast and large-scale graph-based construct is a route-planning algorithm found in complex urban conflict simulation, *e.g.* the Joint Semi-Automated Forces (JSAF) simulation. JSAF currently employs a heuristic A* search algorithm to do route planning for its millions of entities –- the algorithm is sequential and thus very computationally expensive. Using the GPU, the JSAF simulation can off-load the route-planning component to the GPU and remove one of its major bottlenecks. (Tran, 2008)

Early Experimental Results

The initial year of research on the JFCOM cluster *Joshua* was marked with typical issues of stability, O/S modifications, optimization and experience. All of the major stated goals of the cluster proposal were met or exceeded. Research use by JFCOM was at a low level of operation due to issues outside the prevue of this report, but Joshua easily met its stability and availability requirements from JFCOM.

Early work centered on the issues of getting the machine up and running. One problematic issue was getting the correct OS installed and coordinating that with the nVidia staff's recommendations as to varying version incompatibilities. Joshua provided 24x7x365 enhanced, distributed and scalable computational resources that did enable joint warfighters at JFCOM as well as its U.S. Military Service and International partners to develop, explore, test, and validate 21st century battlespace concepts in JFCOM J9's Joint Futures Laboratory (JFL). The specific goal was to enhance global-scale, computer-generated military experimentation by sustaining more than 2,000,000 entities on appropriate terrain with valid phenomenology.

The JFCOM team deployed existing DOD simulation codes which were previously run on advanced Linux clusters located at an appropriate site or sites, *e.g.* the Maui High Performance Computing Center (MHPCC) and the Aeronautical Sciences Center HPC group (ASC-MSRC). This team then proposed supplementing the previous JFCOM J9 cluster assets with a new cluster enhanced with 64-bit CPUs and graphics processing units (GPUs), in the form that became *Joshua*. They began the process of modifying the legacy code to enable efficacious use of the new capabilities. As an important step in this procedure, the ISI team taught three GPGPU programming courses for DoD personnel.

The major quantifiable goal for Joshua was the simulation of at least two million JSAF entities, including culture and forces. This was more than achieved in a major breakthrough in which ten million entities were simulated in a Middle Eastern urban environment, complete with demographically correct civilians. (Figure 2)

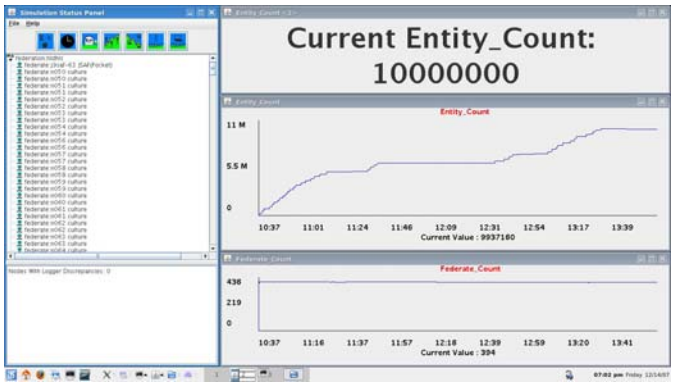

Figure 2. Screen Capture of Ten Million Entity Run

The technical and computational challenges were interesting, but not daunting. FMS battlespace portrayals of this magnitude and flexibility was previously impossible due to limitations of computational power. An earlier pair of clusters had enabled the development and implementation of a proven scalable code base capable of using thousands of nodes interactively. The JFCOM team continued to address community-wide issues such as: enhanced security for distributed autonomous processes, interactive HPC paradigms, use of advanced architectures, self-aware models, global terrain with high-resolution insets and physics-based phenomenology, requisite for Joint Experimentation by JFCOM.

The GPGPU-enhanced cluster was capable of providing warfighters with the new dedicated cluster assets that enabled the simulation that was required to provide the large number of entities and the global terrain nec-

essary to adequately rep-resent the complex urban battlefield of the 21st Century. It enabled experimenters to simulate the full range of forces and civilians, all interacting in future complex battlespaces.

There is a general consensus that there are two possible ways to improve simulation fidelity: (a) by increasing entity counts (quantitatively) and (b) by increasing realism (qualitatively) of entity behaviors and resolution of the environment. Numerous efforts have been made to increase the former, *e.g*. SF Express (Brunnet, *et al.* 1998) and Noble Resolve (USJFCOM 2006). These included the use of the Scalable Parallel Processors (SPP) or clusters of compute nodes (Wagenbreth, et al. 2005). As for the latter, JFCOM M&S teams have made great strides to improve entity behavior models (Ceranowicz, *et al*. 2002 and 2006) by adding intelligence to the simulation entity behaviors, and with these improvements entities behave in more realistic fashions. Because JSAF has been required to participate in more urban operations, the density of the road and trail networks has dramatically increased. This dictates an increase in computational costs (in terms of how entities relate to the environment), which is the heart of this research effort.

Power Consumption Analyses

As the plethora of GPGPU accelerators continues to burgeon, the findings cited in the following analyses are notional at best, of interest only as a model approach at a more cynical level. In any case, these analyses do support the proposition that the use of GPGPUs is probably indicated as an acceptable method to reduce power consumption per unit of computations (usually quantified here as FLOPS, *i.e*. FLoating Point Operations Per Second). Let us first examine the extra power drain load, at the maximum power drain specified, the drain at high computational loads, the drain at idle and the drain with the GPGPU card removed from the node.

Three versions of the nVidia GPUs were tested, the 8800, 9400 and 9800. The Tesla chip, of incredible interest to this team, was not available for test in our academic setting. In each case, the host for the GPGPU was chosen to best compliment the GPU itself, so different platforms were used in every instance. While this many seem as comparing apples and oranges, that is a necessary result of the choice of the target GPUs and would be more convoluted if they were all tried on one platform, with the concomitant compromises.

In each case, a Model 22-602 Radio Shack AC Ammeter Probe, as seen below in Figure 3 was used to test current flow to the entire node.

Figure 3. Image of the Ammeter used for Current Testing

Wattage parameters from the vendor are typically maximum current allowed, not typical current usage under various conditions. That is why the authors measured each value themselves. All values in this paper were either measured or calculated.

In each case, the amperage was measured, within the accuracy of the meter, of the current to the node under test while exercising the GPU to the maximum extent feasible, at idle while running, at a sleep or hibernate mode and then finally, with the subject card removed. Cost, time and instrumentation constraints precluded measuring the entire power consumption of the cluster *Joshua*, so figures for that power consumption were derived from findings and from data available from the vendors.

The authors wish to issue a caveat about the amperages cited. They can reliably be used for comparative purposes, but care should be exercised if trying to calculate actual amperages to be experienced in different computational environments and using different analytic tools. The accuracy of the meter used could be reliably certain to return comparative figures, but the absolute numbers might be off by some significant fraction. Test/retest numbers were very stable, giving some assurance that the comparative values were meaningful. The question that was being posed was: "How much power does the GPGPU card consume in each of several different states and with different host environ-

ments. The details of the hosts are omitted here for space considerations but are available from the author's, upon request.

Table 1. Some Power Readings using Different GPGPUs

GPU Status →	Whole Node Watts (± 4%)			
	Max	**Idle**	**Sleep**	**Removed**
8800	264	228	228	156
9400	444	360	324	275
9800	730	586	540	460

These data indicate that the entire node takes on the order of 50% more power at full load and that the GPGPU adds on the order of 15-20% power consumption, even at rest, assuming one GPGPU card per processor.

Experience in Training CUDA Programmers

Utility of the GPGPUs is only manifest if programmers can easily adopt the implementation practices required. Offering several courses has proven to be an aid to the DoD users as they worked to take full advantage of heterogeneous computing clusters. The GPGPUs can be programmed using the new CUDA (Compute Unified Device Architecture) programming language. The DoD users needed to make the most of the new GPGPU-enhanced Linux clusters, and learning to use it also acted as an introduction to those clusters with STI (Sony, Toshiba, IBM) Cell processors, Field Programmable Gate Arrays (FPGAs) and other heterogeneous assets. The tutorials focused on general approaches, as well as the specific requirements of the CUDA programming paradigm. Programming models, code examples and practice problems in CUDA were presented and implemented in class, some on nVidia 8800 (or higher)-enabled lap tops. This accessibility and course successes support the GPGPU vision.

These tutorials were presented in two and three day formats under the auspices of a High Performance Computing Modernization Programs (HPCMP) organization to "rave reviews." More than 90 DoD users have taken the course. The main instructor was Gene Wagenbreth. He is an experienced HPC programmer, with over 30 years experience starting with the ILLIAC IV. He has been effectively using the nVidia GPGPUs for three years and has taught this course three times. The students taking the course reported it effective and further reported they could then program their own applications with ease. This programming accessibility may well mitigate in favor of the GPGPU approach, even though fine tuning or the other approaches may have greater power savings to tout.

Sample of GPGPU Programming in CUDA

Without delving too deeply into CUDA programming, the authors think it prudent to show the reader some indication of what CUDA programming entails.

First, here is some FORTRAN code:

```
do j = jl, jr
    do i = jr + 1, ld
        x = 0.0
        do k = jl, j - 1
            x = x + s(i, k) * s(k, j)
        end do
        s(i, j) = s(i, j) - x
    end do
end do
```

Now, here is the same algorithm, implemented into CUDA:

```
ip=0;
for (j = jl; j <= jr; j++) {
    if(ltid <= (j-1)-jl){
        gpulskj(ip+ltid) = s[IDXS(jl+ltid,j)];
        }
    ip = ip + (j - 1) – jl + 1;
    }

__syncthreads();

for (i= jr + 1 + tid; i <= ld;
          i += GPUL_THREAD_COUNT) {
    for (j = jl; j <= jr; j++) {
        gpuls(j-jl,ltid) = s[IDXS(i,j)];
        }
    ip=0;
    for (j = jl; j <= jr; j++) {
        x = 0.0f;
        for (k = jl; k <= (j-1); k++) {
            x  = x      + gpuls(k-jl,ltid) * gpulskj(ip);
            ip = ip + 1;
            }
            gpuls(j-jl,ltid) -= x;
        }
    for (j = jl; j <= jr; j++) {
        s[IDXS(i,j)] = gpuls(j-jl,ltid);
        }
    }
```

The real art in the programming is understanding which algorithms will do so much better on the GPGPU as to warrant the overhead costs of taking them off of the CPU and transferring them to the GPGPU. For more detail, the reader is referred to the authors' web sites on GPGPU processing. (Davis, 2009) nVidia also offers course materials on line and the authors willingly acknowledge the assistance that nVidia has given to them.

OTHER APPROACHES TO INCREASING COMPUTATION/WATT RATIOS

Bill Dally's ELM Moves Data More Efficiently

Computing pioneer Bill Dally has been advancing a different approach to saving power during computation. Analyzing the power utilized at the micro-circuit level, he observed that most of the power was being used moving data around the chip. As many of these movements were non-optimal artifacts of earlier VLSI design constraints, he and his team at Stanford set out to make the data flows more power efficient. (Dally, 2008)

Professor Dally's ELM project has sought high-performance in the creation of a low-power and programmable embedded system. He has sought to reduce the very inefficient memory transfers by designing a chip comprised of many efficient tiles and providing a full software stack. It is his intention that ELM will be able to reduce or eliminate the need of fixed function logic blocks in passively cooled systems.

The ELM team maintains that energy consumption in modern processors is dominated by the supplying of instructions and data to functional units. If interconnects benefit less than logic from advances in semiconductor technologies, driving the interconnects has accounted for an increasing fraction of the energy consumed. This may account for more than 70% of the energy consumed by the computing unit.

Providing a platform that can execute real-time computationally intensive tasks and still reduce the power utilized is the goal of the ELM architecture. This is being done in reaction to the fact that embedded systems, *e.g*. cell phones, are comprised of microprocessors and fixed-function circuitry. Programmability for the system is provided by the microprocessor, but it is too inefficient to meet the computation, timing, and power constraints of many communication and multimedia protocols. This, in turn, requires fixed function logic to be added to embedded systems to provide the necessary performance. Unfortunately, this cannot be changed once the system has been fabricated. To remove the inefficiencies associated with this programmability conundrum, the ELM is designed so that software replaces the fixed function hardware. Clearly, this is a good thing since software applications are more cost-effective to create and update than silicon and the concomitant power savings are still realized.

Simple tiles, called Ensembles, are made up of software managed memory (EM) and several Ensemble Processors (EPs). Professor Dally maintains that these small tiles are much more energy efficient than large cores and offer more computation contexts for each die area. The team is developing the tiled architecture using software to take advantage of the available computation resources. The rationale here is that a larger software up-front cost will be amortized over a programs' lifetimes.

Each EP can issue both an arithmetic and memory operation using a two wide instruction. Load latencies are managed easily. Pre-fetching into the instruction registers prior to execution eliminates stalling on jumps. Some old parallelization techniques are used, *e.g.* the ELM architecture supports single-instruction multiple data (SIMD) execution within an Ensemble. All EPs execute in lock-step with instructions coming from a single IRF. This has effectively quadrupled the amount of instructions that can be stored.

A 64-entry, software managed instruction register file is available to the EPs. The small size of the register files is adequate to hold the inner loops of programs with little performance degradation. Reduced instruction-supply energy is realized by having only one instruction fetch per cycle per PE.

The assertion made is that there can be power reductions of on the order of two orders of magnitude for individual operations on the silicon. In Table 2 below, Dally's team presents their data on power reductions. (Balfour, 2008)

Table 2. Power Savings Using ELM

Ensemble Processor			
Technology	TSMC CL013G (V_{DD}=1.2V)		
Clock Frequency	200 MHz		
Average Power	28 mW		
Multipliers	16-bit + 40-bit acc.	16.5 pJ/op	
IRFs	64 128-bit registers	16 pJ/read	18 pJ/write
XRFs	32 32-bit registers	14 pJ/read	8.7 pJ/write
ORFs	8 32-bit registers	1.3 pJ/read	1.8 pJ/write
ARF	8 16-bit registers	1.1 pJ/read	1.6 pJ/write
Ensemble Memory	8KB	33 pJ/read	29 pJ/write
RISC Processor			
Technology	TSMC CL013G (V_{DD}=1.2V)		
Clock Frequency	200 MHz		
Average Power	72 mW		
Multiplier	16-bit + 40-bit acc.	16.5 pJ/op	
Register File	40 32-bit registers	17 pJ/read	22 pJ/write
Instruction Cache	8KB (2-way)	107 pJ/read	121 pJ/write
Data Cache	8KB (2-way)	131 pJ/read	121 pJ/write

This approach shows much promise, but is not immediately applicable and may be encumbered by the, as yet demonstrable capability of journeymen programmers to master the analytical techniques required for optimization. Further, the authors were not able to find any data that supported an analysis of overall power savings. In

an analogous way, there is a temptation for GPGPU advocates to claim huge processing speed-ups for some restricted sub-routine, but they less inclined to say what the impact was on the total functioning code base that is actually needed by the user.

IBM's Blue Gene Experiments

IBM has also been the conceptual midwife for power reduction technologies, one, like GPGPUs, springing from the mushrooming game industry, *i.e*. the STI (Sony, Toshiba, IBM) Cell processor and, another, the more elitist Blue Gene series of high performance computers. The STI Cell chip is a matter for a different paper, but the Blue Gene vision merits a little attention here.

IBM integrated all of the putatively essential subsystems on a single chip, each of the computational or communications nodes dissipating low power (about 17 watts, including DRAMs). Low power dissipation enables the installation of as many as 1024 compute nodes and the necessary communications nodes in the standard computer rack. This can be done in accordance with standard limits on electrical power supply and air cooling. As discussed earlier, the important performance metrics in terms of power: FLOPS per watt, space: FLOPS per m2 of floor space and cost: FLOPS per dollar have allowed IBM to scale up to very high performance. (Chiu, 2005) The issue may be, “Was this done at the expense of general purpose accessibility?”.

Many experienced programmers do note that this is not a classical "general purpose" computer, as it requires significant esoteric skills to make optimal use of its power. The nodes are attached to three parallel communications networks: peer-to-peer communications use a 3D toroidal network, collective communications use a collective network and fast barriers use a global interrupt network and external communications are provided by an Ethernet network. File system operations are handled by the I/O nodes on behalf of the compute nodes Finally, there is a management net to provide access to the nodes for configuration, booting and diagnostics.

The compute nodes in Blue Gene/L support a single user program using a minimal operating system. A limited number of POSIX calls are supported, and only one process may be run at a time. Green threads must be implemented in order to simulate local concurrency. C, C++, or Fortran are the supported languages and as is common with clusters, MPI is used for communication.

The Blue Gene/L system can be partitioned into electronically isolated sets of nodes to allow multiple programs to run concurrently. The major drawbacks seem to be that the hardware is not based on a commercially supported product, as are the Cell processor implementations and the GPGPU accelerations, and on the potentially problematic programming environment.

ANALYSIS

The authors have resisted the temptation to claim huge savings in power consumption per unit computation. They note that, while the nVidia processors in the 8800 series may have power which is in the several hundred GigaFLOPS range, the issue of real interest is, "What will it do on the programs the user needs?" In the authors' case, that is the simulations run by JFCOM. The GPGPUs can attack some issue, most notably the spikes of activity occasioned by a sensor being simulated or a new direction of travel for a major unit. These spikes are tailor-made for resolution by GPU processing, bearing close resemblance to the visualization algorithms for which the GPU was designed. By easily handling the visualization (Wagenbreth, 2007) and route-finding spikes (Tran, 2008). the GPGPUs do actually provide am effective doubling of effective computing for the cost of approximately a 30% increase in power. Clearly this is desirable at this level, and considering the newness of the approach, more impressive gains might be anticipated for later.

In the case of *Joshua*, one GPGPU for every 8 cores was considered prudent and experience has shown that the GPGPUs have not been insufficient to meet the needs imposed upon them. In this case, the power increase is more on the order of 5%, with the anticipated doubling of computational power. Should this ratio turn out to be valid in other, more constrained implementations, as described above, the benefits will be significant Increased habitability, reduced heat signatures, increased battery life, reduced environmental stress on electronic components, and other benefits would accrue with almost trivial energy costs. More importantly, the computing power the warfighter needs would be made available to him where he needs it. This is not to say that other approaches to heterogeneous high performance computing may not also hold promise. As with all new technologies, the costs in terms of availability, adoptability and training must be kept in mind.

In more mundane settings, say a domestic computing center, the cost savings in power alone are significant. As the numbers on power usage for large clusters such

as Joshua are merely daunting in Virginia, in more remote areas such as the Maui High Performance Computing Center where they face electric rates that are literally multiples of what is common on the mainland, it is reasonable to look at the doubling of computational power as vital. It means that one's FLOPS per Watt improvements may generate on the order of savings from $2,500 per hour to $5,000 per hour, at $.09 and $.20 per Kilowatt Hour respectively for the two centers.

CONCLUSIONS

Many new technologies offer various paths to increasing computational power and restraining the multiple and varied costs of power consumption. The authors maintain that even their conservative approach and carefully substantiated claims support the tenet that heterogeneous computing displays many attractive features of interest to the DoD.

ACKNOWLEDGEMENTS

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government. This material is based on research sponsored by the Air Force Research Laboratory under agreement number FA8750-05-2-0204. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

REFERENCES

Balfour, J., William J. Dally, W. J., Black-Schaffer, D., Parikh, V. and Park, JS., (2008), An Energy-Efficient Processor Architecture for Embedded Systems, in *IEEE Computer Architecture Letters*, Vol 7, No. 1, Irvine CA

Ceranowicz, A., M. Torpey, W. Hellfinstine, J. Evans and . Hines. (2002). Reflections on building the joint experimental federation. In Proceedings of the *Interservice/I industry Training, Simulation and Education Conference*, Orlando, FL

Ceranowicz, A. & Torpey, M., (2005), Adapting to Urban Warfare, *Journal of Defense Modeling and Simulation*, 2:1, January 2005, San Diego, Ca

Ceranowicz, Andy, Torpey, M., & Hines, J. (2006) "Sides, Force, and ROE for Asymmetric Environments," *Interservice/Industry Training, Simulation, and Education Conference Proceedings*, Orlando, FL

Charlesworth, A., & Gustafson, J., (1986), Introducing Replicated VLSI to Supercomputing: the FPS-164/MAX Scientific Computer, in *IEEE Computer*, 19:3, pp 10-23, March 1986

Chiu, G. L.-T., Gupta M., and Royyuru, A. K., Guest Editors (2005), Blue Gene, in *IBM Journal of Research and Development*, Armonk, NY

Dally, W. J., James Balfour, J., Black-Shaffer, D., Chen, J., Harting, R.C., Parikh, V., Park, JS., Sheffield, D. (2008). Efficient Embedded Computing, in *IEEE Computer*, Los Alamitos, CA.

Davis, D. M. , (2005), Un-published Military Biography of Dan M. Davis, retrieved on 06 Jul 2009 from: http://www.isi.edu/~ddavis/DanzFiles/bios/DDavis-MilitaryBio.pdf

Davis, D. M., (2009). *General Purpose Computing Using GPUs on a Linux Cluster*: *An Introduction and Programming Practicum*, retrieved 06 Jul 2009 from: http://www.isi.edu/~ddavis/GPU/Course3/

Diniz, P., Lee, Y.-J., Hall, M. & Lucas, R., (2004), A Case Study Using Empirical Optimization for a large, Engineering Application, in the Proceedings of the *18th International Parallel and Distributed Processing Symposium*, April 2004, pp: 200-208

Dongarra, J., (1993), *Linear algebra libraries for high-performance computers: a personal perspective, Parallel & Distributed Technology: Systems & Applications*, IEEE, Feb. '93, Volume: 1, Issue: 1, pp: 17 - 24

Lucas, R.F., Wagenbreth, G., Tran, J.J., & Davis, D. M., (2007a), Implementing a GPU-Enhanced Cluster for Large-Scale Simulations; *Interservice/Industry Training, Simulation, and Education Conference Proceedings*, Orlando, FL

Lucas, R.F., Wagenbreth, G., Tran, J.J., & Davis, D. M., (2007b), Multifrontal Computations on GPUs, unpublished ISI White Paper, retrieved 06Jul09 from: www.isi.edu/~ddavis/JESPP/2007_Papers/SC07/mf2_gpu_v0.19a-nms.doc

Neitzel, S., (2007), *Taping Hitler's Generals, Transcripts of Secret Conversations, 1941-1945*, Frontline Books,

Tran, J.J., Lucas, R.R., Yao, K-T., Davis, D.M. and Wagenbreth, G., Yao, K-T., & Bakeman, D.J., (2008), A High Performance Route-Planning Technique for Dense Urban Simulations, *I/ITSEC Conference*, Orlando, FL