The Test and Evaluation Uses of Heterogeneous Computing: GPGPUs and Other Approaches

Dan M. Davis, Gene Wagenbreth & Robert F. Lucas
Information Sciences Institute, USC
Marina del Rey, California

Paul C. Gregory
Lockheed Martin Company.
Suffolk, Virginia

ABSTRACT

The Test and Evaluation Community faces conflicting pressures: provide more computing power and reduce electrical power requirements, both on the range and in the laboratory. The authors present some quantifiable benefits from the implementation of General Purpose Graphics Processing Units (GPGPUs) as heterogeneous processors. This produces power, space, cooling and maintenance benefits that they have documented. Other efforts in the field of power reduction techniques will be outlined, e.g. the Efficient Low-power Microprocessor (ELM) approach of Prof. William Dally and IBM's well publicized Blue Gene project. The utility of all of these techniques for the T&E community will be assessed and the GPGPU approach. The authors will report on several aspects their experience with GPGPUs: programmability, performance of codes implemented in several areas of computational science and the compute power per unit of electrical consumption. An overview of code design and implementation approaches proven useful at JFCOM will be discussed. Actual working code segments will be outlined and explained, along with the design rationale behind them.

AUTHORS

Dan M. Davis is the Director, JESPP Project, Information Sciences Institute (ISI), University of Southern California (USC), focusing on large-scale distributed DoD simulations. As the Assistant Director of the Center for Advanced Computing Research at Caltech, he managed Synthetic Forces Express, bringing HPC to DoD simulations. Prior experience includes work as a Software Engineer at the Jet Propulsion Laboratory and at a classified project at Martin Marietta. He saw duty in Vietnam as a Cryptologist in the USMC and he retired as a Commander, Cryptologic Specialty, U.S.N.R. He received B.A. and J.D. degrees, University of Colorado in Boulder. ddavis@isi.edu

Robert F. Lucas is the Director of the Computational Sciences Division of ISI at the USC. There he manages research in computer architecture, VLSI, compilers and other software tools. He's been the PI on the JESPP project since 2002. Prior to joining ISI, he was the Head of the High Performance Computing Research Department for NERSC at LBNL. Earlier he was the Deputy Director of DARPA's ITO and a member of the research staff of IDA's Center for Computing Sciences. Dr. Lucas received B.S., M.S., and Ph.D. degrees in Electrical Engineering from Stanford University, Palo Alto, California. rflucas@isi.edu

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 parallelization of Fortran programs. He is active in benchmarking, optimization and porting of software and has programmed on CRAY, SGI, Hitachi, Fujitsu, NEC, networked PCs, networked workstations, IBM SP2, as well as conventional machines. He received a B.S. in Math/Computer Science from the University of Illinois, Champagne-Urbana. genew@isi.edu

Paul C. Gregory Jr. is the Lockheed Martin Senior Program Manager, for Modeling & Simulation Support task orders at the Joint Experimentation Directorate (J9) of the U.S. Joint Forces Command in Suffolk, Virginia. Mr. Gregory has supported J9 since August 1998 as an experiment planner, designer, control cell member, and as the contractor lead for the Joint Concept Development Pathway. Earlier, he retired from the Air Force as an electronic warfare officer in the F-4G Wild Weasel aircraft. He received a B.S. from the University of Tennessee, Knoxville and an M.S., at Embry-Riddle Aeronautical University, Long Beach California. paul.c.gregory@lmco.com

The Test and Evaluation Uses of Heterogeneous Computing: GPGPUs and Other Approaches

Introduction and Background

It is commonly held that Test and Evaluation (T&E) is one of the most critical steps in the development of virtually all Defense systems (Fox *et al.*, 2003). It is the central means of making sure that new systems will reliably perform its intended functions in its intended environment, often combat. T&E of current systems is an elaborate and time consuming process that reflects both the intricacies of the object of the test and the range of equipment, personnel and environments required. Many argue that this process consumes far too much of the time that it takes to put new systems into the hands of the warfighters and uses way too many resources, without much obvious benefit for those on in combat.

One solution to ameliorating these costs and delays is the increased use of computer simulations, ranging from argent-based-models of battlespaces to MCAE analyses of hardware to esoteric simulations using computational fluid dynamics to assess everything from new air-frames to plume dispersals of chemical agents.

But computing costs are significant as well. These costs are not only the computer purchase price, be it a small workstations or time on High Performance Computers (HPC). They must include the costs of training, programming, maintaining, validating and supporting extensive code bases. (Kepner, 2004) These questions are even more urgent because of the increasing emphasis in T&E concerns the expenditures of money and time in the development process. Efficiency is critical when cost overruns and schedule delays are deleterious and costly (Fox, 2004).

One potential approach to reducing costs, time-to-roll-out and physical danger, all the while improving validity, transparency and utility, is to adopt the strategy of heterogeneous computing. Heterogeneous computing is the use of a variety of different types of computational units to aid the Central Processing Unit (CPU), such as accelerators like General Purpose Graphics Processing Units (GPGPUs), Field Programmable Gate Arrays (FPGAs), Digital Signals Processors (DSPs) and the Sony/ Toshiba/IBM (STI) Cell processor.

There is a growing body of evidence on the use of these devices, some of it created by the authors in their work on large-scale battlespace simulations at the US Joint Forces Command (JFCOM) and its Joint Concept Development and Experimentation Directorate (J9).

JSAF

One program in use at JFCOM is the Joint SemiAutomated Forces (JSAF) code. JSAF is loaded onto a network of processors in either workstations or Linux Clusters. They communicate, via a local or wide area network. Communication is implemented with High Level Architecture (HLA) and a custom version of Runtime Infrastructure (RTI) software, called RTI-s. A run is implemented as a federation of simulators or clients and multiple clients, in addition to JSAF, are typically included in a simulation.

As is common in the T&E community, operational imperatives drive experimental designs that require even further expansions of simulation code capabilities. These needs include some of the following:

- More background entities
- More complex behaviors
- Larger geographic area
- Multiple resolution terrain
- More complex environments

The energy efficiency issues addressed here are 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.

In T&E settings, the need for power conservation is still paramount, mainly 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 Maryland, and by installation, such as with the fact that size and temperature constraints that differ from a High Performance Computing Center to a test aircraft cockpit. Nevertheless, all of the previously mentioned parameters are important, critical or vital, as the case may be.

While most equipment suffers from high heat, electronics are especially sensitive. The micro-circuitry now employed in every phase of computing is 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.

This paper will investigate innovative and effective ways to accomplish the same amount of computation, while using significantly less total energy. The technique studied by the authors is using General Purpose Graphics Processing Units, to effectively handle computationally intensive activity "spikes". The authors will report on three specific aspects their use of GPGPUs:

- code drafting and development hurdles and opportunities
- codes modified in several areas of computational science
- a wide range of software results in FLOPS per Watt parameters in various hardware configurations.

An introductory synopsis of algorithmic design and implementation strategies should allow the T&E users 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. A separate paper is being presented at this conference on training by one of the authors, Wagenbreth.

GENERAL PURPOSE GRAPHICS PROCESSING UNITS AS COMPUTER ACCELERATORS

Methodology Employed in Simulation

To better analyze potential T&E use, the method implemented by this team for Forces Modeling and Simulation (FMS) will be set forth. They used existing DOD simulation codes running on advanced Linux clusters operated by JFCOM. The previous J9 clusters were on Maui and at Wright Patterson Air Force Base in Ohio but the new cluster enhanced with 64-bit CPUs and NVIDIA 8800 graphics processing units (GPUs) was in Suffolk at JFCOM (Lucas *et al.*, 2007).

The T&E community at large would find communality in the Forces Modeling and Simulation (FMS) use of accelerator-enhanced nodes, as many of the same issues are germane. Further, envisioning other acceleration targets is not difficult:

- physics-based phenomenology,
- CFD plume dispersion,

- computational atmospheric chemistry,
- data analysis,

GPGPU experiments were first conducted on a more manageable code set, to ease the programming burden and hasten the results. BLAS (Basic Linear Algebra Subprograms) routines (Dongarra *et al.*, 1993) were seen as an appropriate candidate. An MCAE "crash code" arithmetic kernel was used as vehicle for a basic demonstration problem, based on earlier work (Diniz *et al.*, 2004).

This preliminary characterization of GPU acceleration focused on a subset of the large space of numerical algorithms, in this case: factoring large sparse symmetric indefinite matrices. Such problems often arise in Mechanical Computer Aided Engineering (MCAE) applications. The ISI team made use of the SGEMM (Single precision GEneral Matrix Multiply) algorithm (Whaley, 1998).

The GPU should also be a very attractive T&E computation accelerator to overcome hurdles, *e.g.* sparse matrix factorization. Previous generations of accelerators, such as those designed by Floating Point Systems (Charlesworth *et al.*, 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 in test and evaluation settings, the GPU, data transfer and interaction between the host and the GPU needs to be reduced to an acceptable minimum. The T&E user should be warned that the conduct of this analysis is not trivial and the costs of it must always be born in mind when considering the use of GPGPUs (Kepner, 2004).

Implementation Research Results

Results for recent runs on the C1060 from NVIDIA are shown in Figure 1, which plots the time is takes to factor the matrix, as a function of the number of cores employed, both with and without the GPU. ISI used a dual-socket Nehalem host, sustaining 10.3 GFLOPS when using one core, and 59.7 GFLOPS when using all eight. When the GPU is employed, it performs 6.57E+12 operations, 92% of the total, and sustains 98.1 GFLOPS in doing so. The code's overall performance with the GPU improves to 61.2 GFLOPS when one host core is used, and 79.8 GFLOPS with all eight. For perspective, reordering and symbolic factorization take 7.9 seconds, permuting the input matrix takes 2.64 seconds, and the triangular solvers take 1.51 seconds. (Lucas *et al.*, 2010)

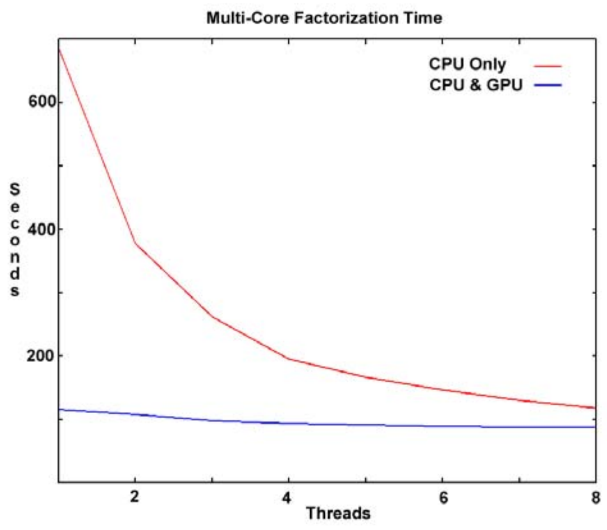

Figure 1. Multicore factorization time, with and without the GPU

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).

Another application that may have T&E uses is a fast and large-scale graph-based construct, *e.g.* route-planning algorithms found in complex urban environment simulations. 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 *et al.*, 2008)

Early Experimentation Results at JFCOM

T&E users may benefit from an awareness of the initial year of research on JFCOM's GPU-enhanced cluster, *Joshua.* It 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. *Joshua* easily met its stability and availability requirements from JFCOM.

Any potential user would be interested in the issues of getting the machine up and running. A typical problem was getting the correct OS installed and coordinating that with the NVIDIA staff's recommendations as to varying versions and incompatibilities. Those types of issues still obtain today.

Joshua provided 24x7x365 enhanced, distributed and scalable computational resources that did enable joint warfighters at JFCOM and International partners to develop, explore, test, and validate 21st century battlespace concepts. 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.

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)

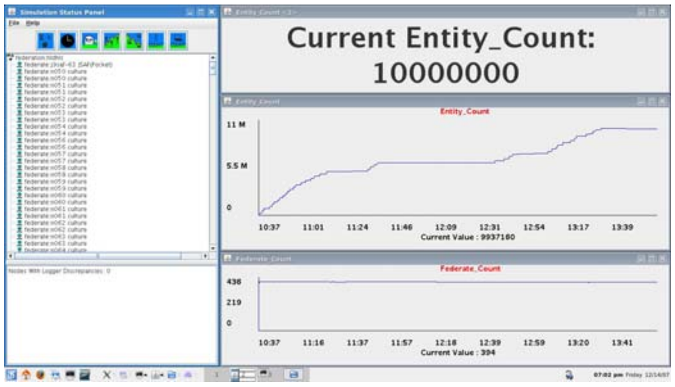

Figure 2. Screen Capture of Ten Million Entity Run

The tasks of overcoming implementation hurdles and stabilizing the compute environment were interesting, but not daunting. Agent-Based Models (ABM) combat simulations of this size and sophistication were previously impossible due to limitations of computational power. The earlier pair of clusters had enabled the development and implementation of a proven scalable code base capable of using thousands of nodes interactively. The ISI team continues to address issues of interest to the T&E community 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, many of which have their counterparts in test and evaluation..

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 JFCOM has been required to simulate 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 was the heart of that research effort.

Power Consumption Analyses

Finding a great deal of interest in GPGPU acceleration, the following work, while necessarily preliminary due to the design dynamics of the devices being offered, may prove useful to those facing power issues today. In any case, these analyses do support the proposition that the use of GPGPUs is probably indicated as a viable method for reducing power consumption per unit of computation (usually quantified here as FLOPS, *i.e.* FLoating Point Operations Per Second). Let us examine the extra power requirement for a system, first at the maximum power drain specified, then the drain at high computational loads, the drain at idle, and finally the drain with the GPGPU card removed from the node.

ISI had access to three versions of the NVIDIA GPUs that were tested, the 8800, 9400 and 9800. The NVIDIA C1060s and C2050s were not available for this early test. Data on them will be presented when it is available. 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 may 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.

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.

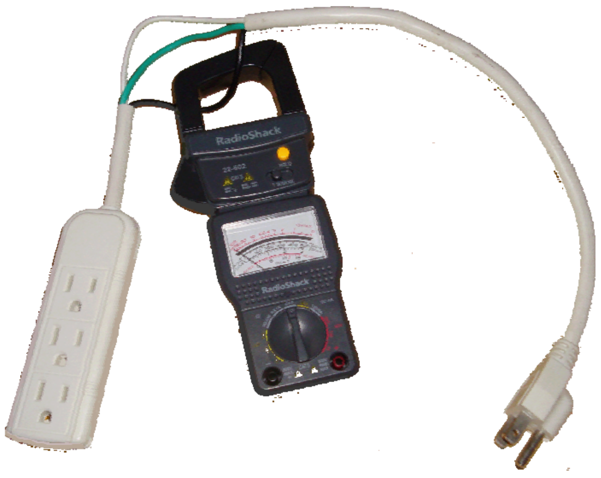

Figure 3. Ammeter and harness used for current quantification

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 1) to the maximum extent feasible, 2) at idle while running, 3) at a sleep or hibernate mode and 4) 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 environments?". The details of the hosts are omitted here for space considerations but are available from the author's, upon request.

Table 1. Power Readings using Different GPGPUs

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. For T&E purposes, the authors would recommend something more on the order of 1 GPGPU per four to eight cores of a CPU.

GPGPU Programming in CUDA

Again, looking at the overall productivity issue, programming ease may easily outweigh power consumption and new hardware costs (Kepner, 2004). While we do not want to analyze CUDA programming too stringently, the authors think it advisable to show the potential user 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);
        }
    }
```

A critical factor, if not the most critical one, in heterogeneous programming is the need to understand which algorithms will do enough better on the GPGPU as to warrant the overhead costs of taking them off of the CPU and transferring them to the GPGPU.

For a more disciplined treatment of the programming environment and approach that will be useful, 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. Like all tasks, there seems to be a critical experience level required for reliable programming in this mode.

OTHER APPROACHES TO BETTER COMPUTATION/WATT RATIOS

ELM Moves Data More Efficiently

Many in the T&E community may be familiar with computing pioneer Bill Dally. He has been advancing a different approach to saving power during computation. Analyzing the power utilized on micro-circuits, his team observed that most of the power was being used moving data around the chip. As many of these movements were the non-optimal artifacts of earlier VLSI designs, he and his Stanford team set out to make the data flows more power-efficient. (Dally *et al.*, 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 inter-connects 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.

ELM implementations are designed so that software replaces the fixed function hardware. This removes the inefficiencies associated with this programmability conundrum. 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.

Ensembles, which are simple tiles, 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.

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

The Stanford team reports 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 *et al.*, 2008)

Table 2. Power Savings Using ELM

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

This approach shows much promise, but may not be immediately applicable to the T&E community 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 and Cell Efforts

IBM is also contributing to power reduction technologies. One is the "big-iron" Blue Gene series of high performance computers. The other, like GPGPUs, springs from the mushrooming game industry, *i.e.* the STI (Sony, Toshiba, IBM) Cell processor.

For its Blue Gene initiative, IBM integrated all of the putatively essential sub-systems on a single chip, with 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 m^2 of floor space and cost: FLOPS per dollar have allowed IBM to scale up to very high performance. (Chiu *et al.*, 2005) The issue may be, "Was this done at the expense of general purpose accessibility?".

This is not a classical "general purpose" computer, as it requires significant esoteric skills to make optimal use of its power. The compute 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.

As to the Cell chip, it has many of the same attributes as the GPGPUs do, but suffers from a less rich development environment. While some are doggedly pursuing this approach to heterogeneous computing, some of our colleagues have reported that they are envious of the ease with which the ISI group can make effective use of the GPGPUs.

ANALYSIS

Out of scientific restraint, the authors have assiduously resisted the temptation to claim huge increases in computational power or efficiencies in power consumption per unit computation. They note that, while the NVIDIA processors in the 8800 through the C2050 series may have potential compute power that is nominally in the several hundred GigaFLOPS range, the issue of real interest is, "What will they do to accelerate the programs the T&E user needs?" In the authors' case, early experiences on the simulations run by JFCOM speak to the evaluation segment of T&E, as that is a major thrust at JFCOM.

The GPGPUs can attack some issues, most notably the spikes of activity occasioned by a data surge by the sensor being simulated or a new direction of travel for a large group. 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 *et al.*, 2007) and route-finding spikes (Tran *et al.*, 2008), the GPGPUs do actually provide am effective overall 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.

Critically, the computing power that the T&E professionals need would be made available to them where they need it, on the range or in the field. This is not to say that the authors find thatother 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

T&E will face increasing demands for ever-growing computer systems. Many new technologies offer various paths to increasing computational power, while restraining the numerous 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 Test and Evaluation community.

Acknowledgements

Thanks are due to the excellent staffs at JFCOM, ASC-MSRC and MHPCC and our colleagues at NVIDIA. Some of this material is based on research sponsored by the Air Force Research Laboratory under agreement number FA8750-05-2-0204. Other work is based on research sponsored by the U.S. Joint Forces Command via a contract with the Lockheed Martin Corporation and SimIS, Inc. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. 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 these organizations.

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

Brunett, S., Davis, D., Gottschalk, T., Messina, P., & Kesselman, C., (1998) "Implementing Distributed Synthetic Forces Simulations in Metacomputing Environments," in the Proceedings of the *Seventh Heterogeneous Computing Workshop*, Orlando, Florida, March 1998

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

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 Re-search and Development, Armonk, NY

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

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. 1993, Volume: 1, Issue: 1, pp: 17 – 24

Fox, B. Boital, M., Graser, J.C., and Younossi, O., (2004)*Test and Evaluation Trends and Costs for Aircraft and Guided Weapons*, Rand Corporation, Santa Monica, California.

Kepner, J., (2004), HPC Productivity: An Overarching View, *International Journal of High Performance Computing Applications*, Volume 18, Issue 4, Pages: 393 - 397

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

Lucas, R.F., Wagenbreth, G., & Davis, D. M., (2010), Multifrontal Computations on GPUs and Their Multicore Hosts, in the *Proceedings of the VecPar Conference*, Berkeley, California.

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, *Interservice / Industry Training, Simulation, and Education Conference Proceedings*, Orlando, FL