

Accelerators For High Performance Computing Investigation

Jim Bovay, Brent Henderson
Hsin-Ying Lin, Kevin Wadleigh
High Performance Computing Division
Hewlett-Packard Company

January 24, 2007

Table of Contents

1	Accelerators	3
1.1	Introduction.....	3
1.2	Accelerator background.....	3
1.3	Accelerator interconnects	4
1.4	AMD and Intel accelerator programs	4
1.5	Other technical issues	4
1.6	Conclusion	5
2	Types of accelerators	7
2.1	GPUs	7
2.2	FPGAs	8
2.3	ClearSpeed.....	10
2.4	IBM Cell.....	11
3	Accelerator Interconnects	12
3.1	PCI-X, PCI-e.....	12
3.2	Link connections: HTX (AMD), CSI (Intel)	12
3.3	CPU replacement in socket: AMD, Intel CPUs	12
3.4	CPUs and accelerators on the same chip	13
3.5	The importance of the interconnect.....	13
4	High Performance Computing Considerations.....	14
4.1	Metrics	14
4.2	High Performance Computing applications and kernels	14
4.3	Iterative refinement	15
5	Investigation Findings.....	16
5.1	GPUs	16
5.2	FPGA.....	19
5.3	ClearSpeed.....	20
5.4	Customer Codes	22
5.5	Current Status.....	22
5.6	Conclusions and predictions	23
A	Appendix – Accelerators and ECC.....	25
A.1	GPUs	25
A.2	FPGAs	25
A.3	ClearSpeed.....	25
A.4	Cell	25

1 Accelerators

1.1 Introduction

For many years microprocessor single thread performance has increased at rates consistent with Moore's Law for transistors. In the 1970s-1990s the improvement was mostly obtained by increasing clock frequencies. Clock speeds are now improving slowly and microprocessor vendors are increasing the number of cores per chip to obtain improved performance. This approach is not allowing microprocessors to increase single thread performance at the rates customers have come to expect. After numerous customer requests for more information, a group was formed within HP's High Performance Computing Division to investigate some of these approaches. Alternative technologies include:

- General Purpose Graphical Processing Units (GPGPUs)
- Field Programmable Gate Arrays (FPGAs) boards
- ClearSpeed's floating-point boards
- IBM's Cell processors

These have the potential to provide single thread performance orders of magnitude faster than current "industry standard" microprocessors from Intel and AMD. Unfortunately performance expectations cited by vendors and in the press are frequently unrealistic due to very high theoretical peak rates, but very low sustainable ones.

Many customers are also constrained by the energy required to power and cool today's computers. Some accelerator technologies require little power per Gflop/s of performance and are attractive from this reason alone. Others accelerators require much more power than can be provided by systems such as blades. Finally, the software development environment for many of the technologies is cumbersome at best to nearly non-existent at worst. An ideal accelerator would have the following characteristics:

- Much faster than standard microprocessors for typical HPC workloads
- Improves price/performance
- Improves performance/watt
- Is easy to program

The HPC space is challenging since it is dominated by applications that use 64-bit floating-point calculations and these frequently have little data reuse. This report gives the results of a hands-on effort to benchmark the various technologies so that it can be determined which ones are most beneficial for HPC customers. HPCD personnel are also doing joint work with software tool vendors to help ensure their products work well for the HPC environment.

This report gives an overview of accelerator technologies, the HPC applications space, hardware accelerators and software tools applicable to HPC, benchmark results, recommendations on which technologies hold the most promise, and speculations on the future of these technologies.

1.2 Accelerator background

Accelerators are computing components containing functional units, together with memory and control systems that can be easily added to computers to speed up portions of applications. They can also be aggregated into groups for supporting acceleration of larger problem sizes. For

example, gamers use attached graphical processors to improve graphics performance. Each accelerator being investigated has many (but not necessarily all) of the following features.

- A slow clock period compared to CPUs
- Aggregate high performance is achieved through parallelism
- Transferring data between the accelerators and CPUs is slow compared to the memory bandwidth available for the primary processors
- Needs lots of data reuse for good performance
- The fewer the bits, the better the performance
- Integer is faster than 32-bit floating-point which is faster than 64-bit floating-point
- Learning the theoretical peak is difficult
- Software tools lacking
- Requires programming in languages designed for the particular technology

1.3 Accelerator interconnects

In typical computer systems, memory bandwidth rates between the CPU and main memory is high. Similarly, accelerators may have very high bandwidth to their local memory. However, issues arise when having to communicate between the CPU memory and accelerators. This section explores options available today, these include:

- PCI-X, PCI-e
- Link connections: HTX (AMD), CSI (Intel)
- CPU replacement in socket: AMD, Intel CPUs
- CPUs and accelerators on the same chip

See Chapter 3 for more details.

1.4 AMD and Intel accelerator programs

In 2006 both AMD and Intel announced long term programs that make accelerators easier to incorporate in their systems.

AMD's umbrella accelerator program is named Torrenza and includes making the definition of the HyperTransport HTX connector available to partners so that they can design HTX slots and socket replacement chips for AMD systems. AMD also acquired high performance GPU vendor ATI in 2006 and has recently announced their Fusion program which will make a single chip with some mix of CPU and GPU cores in 2008.

After the Torrenza announcement, Intel discussed their Geneseo program. The main component of Geneseo is to add extensions to PCI-e to improve performance for accelerators. Another part of Geneseo will make the Intel front side bus definition available to partners and this is similar to AMD's Torrenza effort. The main downside is that Geneseo products won't be available until 2009, while Torrenza is available today.

1.5 Other technical issues

HPC customers want acceleration technologies to behave like conventional computer systems. However, there are some technical limitations to current accelerators that present some

difficulties in achieving this goal.

1.5.1 Coherency and non-coherency

Customers are used to servers that maintain cache coherency across processors. However, one should assume accelerator solutions do not maintain cache coherency today. There are several efforts in progress to address this. The HyperTransport Consortium has proposed a cache coherent HyperTransport which would allow processors and accelerators to maintain cache coherency. Vendors are also discussing socket replacement chips for bus based systems that would also have cache coherency.

1.5.2 Conformance to IEEE 754 Standard

Microprocessors today conform to the IEEE Standard for Binary Floating-Point Arithmetic (IEEE 754-1985). Besides specifying number formats for what can be thought of as typical floating-point numbers, the definition also defines how to handle naturally occurring edge conditions. IEEE 754 defines how to represent special values such as infinity and NaN, how to interpret denormalized input values, how to round numbers, and how to handle exception conditions. It is these edge conditions that are problematic for accelerators. The ClearSpeed accelerator claims to have IEEE 754 support, but most other accelerators do not. FPGAs are usually programmed to work with standard IEEE data representations, but do not handle the edge conditions in accordance with the standard. In theory FPGAs can implement IEEE 754 robustly, but this would take a great deal of space and would likely cancel any performance advantage inherent in the FPGA. GPUs only implement 32-bit floating-point today and the IEEE 754 standard is not adhered to. The last bit is sometimes incorrectly rounded, there is no mechanism for detecting floating-point exceptions, division and square root are implemented in non-standard ways, and many other issues occur. However many accelerator vendors still claim that they conform to IEEE 754 when they do not.

1.5.3 Error Correction Codes

HPC customers generally require that their computations provide correct results, and the same results from the same computation each time. Like standard processors, accelerators are subject to transient errors (“soft errors”) in their electronic components, caused primarily by energetic neutrons generated by high energy cosmic rays. Industry standard processors protect operational and user data from such soft errors using redundancy in its error paths and control systems to detect and correct words that have become corrupted in this way. These redundancy mechanisms are usually called Error Correction Code (ECC), and are typically capable of Single bit error Correction and double bit error Detection (SECDED). ECC is especially important for clusters of systems, since the likelihood of such an error occurring during a computation goes up linearly with the number of components participating in any given computation. Since most accelerator vendors have evolved in workload environments where such errors are insignificant, they have not provided adequate ECC mechanisms in their products. In some cases, this is a straightforward process. In others, there are significant difficulties in providing such data protection. Appendix A is an analysis of accelerators and ECC and was written by Kevin Harris.

1.6 Conclusion

As the rate of increase of clock speeds on CPUs has slowed down, customers are looking for new and innovative ways to speed up their applications. Many of these customers have heard about acceleration technology and the claims surrounding it. Several hardware vendors have products that are currently available or will be available soon in this space. There are advantages and disadvantages of each type of product. Different application workloads appear to benefit more from accelerator technologies than others. Moreover, certain combinations of workloads and

accelerator types appear well suited to each other. This phenomenon has already begun the process of fragmenting the HPC marketplace. So both HPCD and customers need to be aware of these differences when thinking about using accelerators. These issues are described in more detail in the rest of this paper.

2 Types of accelerators

As noted above the following accelerators were studied:

- General Purpose Graphical Processing Units (GPGPUs)
- Field Programmable Gate Arrays (FPGAs)
- ClearSpeed's floating-point accelerators
- IBM Cell processors

2.1 GPUs

If you're into gaming, you have a GPU on your home computer to perform 3-d graphics. Because of the huge volume of graphics boards, these chips are cheap. Since graphic algorithms have a lot of data reuse it doesn't matter than the connection to the CPU has low bandwidth. In the last couple of years GPU board designers have started including the ability to perform 32-bit floating-point calculations in their boards. The theoretical 32-bit floating-point performance is impressive and exceeds a third of a Tflop/s although achievable performance is a small percentage of this. So some application workloads that require only 32-bit floating point are beginning to use GPGPUs. Today's GPUs cannot perform 64-bit floating-point calculations and this limits GPU usefulness in HPC. However, it's been widely reported in the media that GPUs will be able to perform 64-bit floating-point calculations in 2007-2008, although users should not expect these to run the same rate as 32-bit floating-point calculations. Another characteristic of high performance GPUs is that they require a lot of power (and hence a lot of cooling). So they're fine for a workstation, but not for systems such as blades that are heavily constrained by cooling. However, floating-point calculations require much less power than graphics calculations. So a GPU performing floating-point code might use only half the power of one doing pure graphics code.

Most GPUs achieve their best performance by operating on four-tuples each of which is a 32-bit floating-point number. These four components are packed together into a 128-bit word which is operated on as a group. So it's like a vector of length four and similar to the SSE2 extensions on x86 processors. The ATI R580 has 48 functional units each of which can perform a 4-tuple per cycle and each of those can perform a MADD instruction. At a frequency of 650 MHz, this results in a rate of $0.65 \text{ GHz} \times 48 \text{ functional units} \times 4 \text{ per tuple} \times 2 \text{ flops per MADD} = 250 \text{ Gflop/s}$. The recent NVIDIA G80 GPU takes a different approach since it includes 32-bit functional units instead of 128-bit ones. Each of the 128 scalar units runs at 1.35 GHz and can perform a single 32-bit floating-point MADD operation so its theoretical peak is $1.35 \text{ GHz} \times 128 \text{ functional units} \times 2 \text{ flops per MADD} = 345 \text{ Gflop/s}$. Unfortunately GPUs tend to have a small number of registers so measured rates are frequently less than 10% of peak. GPUs do have very robust memory systems that are faster (but smaller) than that of CPUs. Maximum memory per GPU is about 1 GB and this memory bandwidth may exceed 40 GB/s. As discussed above GPUs do not adhere to the complete IEEE 754 standard or have robust ECC.

2.1.1 Hardware

There are two dominant producers of high performance GPU chips: NVIDIA and ATI. ATI was purchased by AMD in November 2006. Until recently both GPU companies were very secretive about the internals of their processors. However, now both are opening up their architecture to encourage third party vendors to produce better performing product. ATI's has their Close To Metal (CTM) API. This is claimed to be an Instruction Set Architecture (ISA) for ATI GPUs so that software vendors can develop code using the CTM instructions instead of writing everything in graphics languages. This will make software development easier and will lead to improved performance. NVIDIA is taking a different approach in that they've announced their CUDA

program for their latest generation GPUs. CUDA started with the C language, added some new extensions and produced a compiler for the language. Software vendors will write code for CUDA instead of graphics code to achieve improved performance. It remains to be seen which approach is best.

AMD has also announced the Fusion program which will place CPU and GPU cores on a single chip by 2009. An open question is whether the GPU component on the Fusion chips will be performance competitive with ATI's high power GPUs.

2.1.2 Software

Most GPU programs are written in a shader language such as OpenGL (Linux, Windows) or HLSL (Windows). These languages are very different from C or Fortran or other common high level languages usually used by HPC scientists. Therefore our team began exploring other languages that would be more acceptable to HPC users.

The most popular alternative to shader languages are streams languages – so named because they operate on streams (vectors of arbitrary length) of data. These are well suited for parallelism and hence GPUs since element in a stream can be operated on by a different functional unit. The first two streams languages for GPUs were BrookGPU and Sh (now named RapidMind). BrookGPU is a language that originated in the Stanford University Graphics Lab to provide a general purpose language for GPUs. This language contains extensions to C that can be used to operate on the four-tuples with single instructions. This effort is currently in maintenance mode because its creator has left Stanford so our team is not pursuing it. However in October 2006, PeakStream announced their successor to BrookGPU. Although they claim their language is really C++ with new classes it looks like a new language. They have created some 32-bit floating-point mathematical routines and we're in the process of evaluating them. PeakStream also is working closely with AMD/ATI, but not with NVIDIA.

The other language we investigated for programming GPUs is RapidMind. This is effort started at the University of Waterloo and led to founding the company RapidMind to productize the language and compilers. This is a language that is embedded in C++ programs and allows GPUs to be abstracted without directly programming in a shader language. While this language is based in graphics programming it is also a general purpose language that can be used for other technical applications. Also, the user does not have to directly define the data passing between the CPU and GPU as the RapidMind compiler takes care of setting up and handling this communication. Since this language was the only viable GPU language suitable for our market, the authors began a series of technical exchanges with RapidMind personnel.

Our team began by implementing a Black-Scholes algorithm, a BLAS-1 SAXPY routine and a BLAS-3 SGEMM matrix-matrix routine using RapidMind to compare the ease of implementation and performance of known OpenGL routines. The RapidMind approach was relatively easy to implement and the performance was acceptable. RapidMind was also interested in optimizing the SGEMM routine and in creating an efficient 1-d FFT routine. After their optimization these routines outperformed all other implementations measured. The result of this collaboration was shown in a poster at SC06 and the results are shown in the "Investigations Findings" section. RapidMind has also simplified the syntax of their language to make it easier to use.

2.2 FPGAs

FPGAs have a long history in embedded processing and specialized computing. These areas include DSP, ASIC prototyping, medical imaging, and other specialized compute intensive areas. An important differentiator between FPGAs and other accelerators is that they are programmable. You can program them for one algorithm and then reprogram them to do a different one. This

reprogramming step may take several milliseconds, so it needs to be done in anticipation of the next algorithm needed to be most effective.

FPGA chips seem primitive compared to standard CPUs since some of the things that are basic on standard processors require a lot of effort on FPGAs. For example, CPUs have functional units that perform 64-bit floating-point multiplication as opposed to FPGAs that have primitive low-bit multiplier units that must be pieced together to perform a 64-bit floating-point multiplication. Also, FPGAs aren't designed to hold a large number of data items and instructions, so users have to consider exactly how many lines of code will be sent to the FPGA. Thousands of lines, for example, would exceed the capacity of most FPGAs.

Compared to modern CPUs, FPGAs run at very modest speeds – on the order of 200-600 MHz. This speed is dependent on the overall capability of the device and the complexity of the design being targeted for it. The key to gaining performance from an FPGA lies in the ability to highly pipeline the solution and having multiple pipelines active concurrently.

Running code on FPGAs is cumbersome as it involves some steps that are not necessary for CPUs. Assume an application is written in C/C++. Steps include:

- Profile to identify code to run on FPGA
- Modify code to use FPGA C language (such as Handel-C, Mitronics, etc.)
- Compile this into a hardware description language (VHDL or Verilog)
- Perform FPGA place-and-route and produce FPGA “bitfile”
- Download bitfile to FPGA
- Compile complete application and run on host processor and FPGA

For example, the latest generation and largest Xilinx Virtex-5 chip has 192 25x18 bit primitive multipliers. It takes 5 of these to perform a 64-bit floating-point multiply and these can run at speeds up to 500 MHz. So an upper limit on double precision multiplication is $[192/5] * 0.5 = 19$ Gflop/s. A matrix-matrix multiplication includes multiplications and additions and the highest claim seen for a complete DGEMM is about 4 Gflop/s, although numbers as high as 8 Gflop/s have been reported for data local to the FPGA. Cray XD-1 results using an FPGA that is about half the size of current FPGAs show DGEMM and double precision 1-d FFTs performing at less than 2 Gflop/s. Single precision routines should run several times faster. FPGAs are very good at small integer and floating-point calculations with a small number of bits. The manager of one university reconfiguration computing site noted: "If FPGAs represent Superman, then double precision calculations are kryptonite."

One HPC discipline that is enamored with FPGAs is astronomy. The current largest very long baseline Interferometry system, LOFAR, has at its heart an IBM Blue Gene system with a peak of 34 Tflop/s. Most of the processing on the Blue Gene systems uses 32-bit floating-point calculations. The next generation system, SKA, to be delivered in the late 2010s will need processing power in the 10-100 Petaflop/s range. The most time consuming algorithms don't need 32-bit computations - 4-bit complex data and calculations are sufficient. Therefore many of these astronomers are experimenting with FPGAs since three FPGA chips can produce performance that exceeds the equivalent of a Tflop/s.

FPGAs belong to a class of products known as Field Programmable Logic devices (FPLD). The traditional and dominant type of FPLD is FPGAs. Recently other types of FPLD have emerged including FPOA (Field Programmable Object Arrays) and FPMC (Field Programmable MultiCores).

2.2.1 Hardware

The dominant FPGA chip vendors are Xilinx and Altera. Both companies produce many different types of FPGAs. Some FPGAs are designed to perform integer calculations while others are designed for floating-point calculations. Each type comes in many different sizes, so most HPC users would be interested in the largest (but most expensive) FPGA that is optimized for floating-point calculations. Other chip companies are the startup Velogix (FPGAs) and MathStar (FPOAs).

2.2.2 Software

Once again the software environment is not what the HPC community is used to using. There's a spectrum of FPGA software development tools. At one end is the popular hardware design language Verilog used by hardware designers. This has very good performance, but the language is very different from what HPC researchers expect. Some vendors have solutions that are much closer to conventional C++. The conventional wisdom is that the closer to standard C that the solution is, the worse the resulting application performs. The reason for this is that to make the best use of FPGAs, users should define exactly how many bits they would like to use for each variable and calculation. The smaller the number of bits, the less space is required on the die, so more can be contained on a chip, and hence the better the performance.

One company used by multiple HPC vendors is Celoxica. Its Handel C language allows users to exactly define the datasize for all variable and calculations. The HPC accelerator team has begun implementing HPC algorithms in Handel-C to gauge its easy of use and performance. See the "Investigation Findings" section for details.

Another language that has potential is Mitrionics' Mitrion-C programming language for FPGAs. The team is also evaluating this to see how applicable it is to the HPC space.

There are also other FPGA C language variants such as Impulse C and Dime-C.

2.2.3 Consortiums

There are two main consortiums related to FPGAs and Reconfigurable Computing (RC). The names FPGA and RC are frequently used interchangeably, but RC is a larger blanket term since it includes other types of reconfigurable technologies such as FPOAs. HP is a charter member of the NSF Center for High-Performance Reconfigurable Computing (CHReC) which is discussed in more detail later. The OpenFPGA organization also seeks to accelerate the adoption of reconfigurable technologies.

2.3 ClearSpeed

ClearSpeed Technology produces a board that is designed to accelerate floating-point calculations. This board plugs into a PCI-X slot, has a clock cycle of 500 MHz, and contains 96 floating-point functional units that can each perform a double precision multiply-add in one cycle. Therefore their board has a theoretical peak of 96 Gflop/s. In late 2006 ClearSpeed previewed boards that are connected to systems by a PCI-e slot. This will help performance get closer to their peak rates.

Clearspeed has a beta release of a software development kit that includes a compiler. There are two ways to use the ClearSpeed boards. One is to make a call to a routine from their math library. This library contains an optimized version of the matrix-matrix multiply subprogram DGEMM. The other way to access the boards is to write routines using the ClearSpeed accelerator language Cⁿ. See the "Investigations Finding" for more performance information.

The first accelerator enhanced system to make the Top500 list is the TSUBAME grid cluster in

Tokyo. It is entry 9 on the November 2006 list and derives about ¼ of its performance from ClearSpeed boards and the rest from Opteron processors.

2.4 IBM Cell

For the PlayStation 3 IBM, Toshiba and Sony have designed an accelerator named Cell. Although the Cell was not on the original list of technologies to evaluate it has become the mind share leader of acceleration technologies. The processor consists of an IBM PowerPC processing element which is responsible for scheduling eight synergistic processing elements (SPEs). IBM has measured these SPEs executing SGEMM at over 200 Gflop/s. Theoretical peak performance on double precision calculations is modest and is less than 12 Gflop/s. IBM claims that a complete SDK is available for the Cell, but notes that assembly code is required to get good performance. Mercury Computer Systems also produces Cell blades and PCI-e cards and we're pursuing getting access to these system.

3 Accelerator Interconnects

In typical computer systems, memory bandwidth rates between the CPU and main memory is high. Similarly, accelerators may have very high bandwidth to their local memory. However, issues arise when having to communicate between the CPU memory and accelerators. This section explores options available today, these include:

- PCI-X, PCI-e
- Link connections: HTX (AMD), CSI (Intel)
- CPU replacement in socket: AMD, Intel CPUs
- CPUs and accelerators on the same chip

3.1 PCI-X, PCI-e

This easiest and cheapest is to have the accelerator on a board that plugs into PCI-X or PCI-Express slots. Since PCI-e has a higher bandwidth than PCI-X, PCI-e is preferred. HP produces servers with various PCI-e implementations. For example, the highest bandwidth occurs on servers with PCI-e x16 since this has a theoretical peak of 4 GB/s in either direction. There are GPU boards and FPGA boards that support PCI-e. The ClearSpeed boards plugs into a PCI-X slot today however. Even though PCI-e x16 bandwidth is very high, measurable bandwidth on accelerators boards up to October 2006 peaked at about 1 GB/s. These measurable rates are expected to at least double in 2007 as GPU vendor improve their solutions to obtain a higher percentage of PCI-e peak.

3.2 Link connections: HTX (AMD), CSI (Intel)

Another way to connect accelerators to servers is by using links. In 2007, HP's Proliant DL145G3 Opteron based system will debut and this has an unused but functional HyperTransport connection that is exported as an HTX slot. Vendors such as Celoxica are producing FPGA boards that will plug directly into this. Thus the DL145G3 can be enhanced with an FPGA. Since HTX connections can run at 8 GB/s one would assume the bandwidth would be much better than a PCI-e connection. However FPGAs have relatively slow clock periods (around 500 MHz) and these rates reduce the HTX rates to around 1 GB/s which are about the same as PCI-e rates. The measured latency using HT should be better than that using PCI-e though.

There are no vendors today that produce GPU boards connected by an HTX slot. Since AMD purchased ATI in 2006, it makes this more likely to happen.

Intel will has discussed their future links based interconnect, Coherent Scalable Interconnect (CSI), and it is expected that accelerators will be able to connect to CSI after it is available.

3.3 CPU replacement in socket: AMD, Intel CPUs

Companies such as XtremeData and DRC produce FPGA boards that plug directly into the Opteron processor slot. Thus a CPU removed and replaced by a board containing a FPGA. This has the disadvantage of decreasing the number of processors available for the computer, but lets the accelerator access the on-node memory with a low latency. This solution is appealing for general Opteron systems including blade solutions.

Some companies are exploring ways to connect FPGA boards to Intel sockets on bus based systems. These solutions may have a short lifespan depending on how soon CSI is available.

3.4 CPUs and accelerators on the same chip

Eventually vendors will produce true heterogeneous systems that have some number of CPU cores and some number of accelerator cores on a single chip. AMD recently acquired GPU vendor ATI and has already announced their Fusion program whose goal is to produce a single chip containing both CPU and GPU cores in 2008.

The closest system to this today is the Cell board designed for the PlayStation3. It consists of a PowerPC core that is responsible for coordinating eight accelerator cores. In a true heterogeneous system the CPU would also be responsible for executing portions of application code in tandem with accelerators. Future versions of Cell are expected to have more SPE cores, 64-bit computations, and internal ECC support, which will establish a good target for competition as an HPC accelerated computing platform.

3.5 The importance of the interconnect

The performance of Fast Fourier Transforms (FFTs) gives a good demonstration of how important interconnect bandwidth is for accelerators. FFTs are widely regarded as having good data reuse. Some government applications require one million point FFTs where the real and imaginary components are represented with 32-bit floating-point numbers. This uses 8 MB of data and requires 100 million floating-point operations. Suppose the interconnect bandwidth is 800 MB/s (a typical value) and that the accelerator is infinitely fast so that no time is spent performing the calculations. The resulting rate is only 5 Glop/s. Clearly the bandwidth is the bottleneck in this case.

4 High Performance Computing Considerations

4.1 Metrics

As discussed earlier, there are many metrics that can be used to measure the benefit of accelerators. Some important ones to consider are:

- Price/performance (want to increase Gflop/s / \$) – the more costly the accelerator, the faster it must be to succeed
- Computational density for system (want to increase Gflop/s / cubic meter) – accelerators can improve this significantly
- Power considerations (want to increase Gflop/s / watt) – some technologies require very little power while other require so much they can't be used in low power systems
- Cluster system Mean Time Between Failure (want to increase Gflop/s * MTBF) – if accelerators allow a reduction in node count, the MTBF may improve significantly.

4.2 High Performance Computing applications and kernels

HPC is challenging for accelerators compared to many disciplines. Most applications require 64-bit floating-point calculations. While some applications have appreciable data reuse, many have very little reuse. Some important applications in HPC are

- Solve a system of equations (64-bit floating-point) using either a direct method based on matrix-matrix multiply or an iterative method leveraging matrix-vector multiplication. Systems of equations are used in Mechanical Computer Aided Engineering (MCAE) and by the Top500 benchmark.
- Vector-vector and matrix-vector operations (64-bit floating-point) are used just about everywhere including MCAE and Life and Material Sciences (L&MS).
- Signal processing (32-bit floating-point) uses Fast Fourier Transforms (FFTs) and convolutions and are used in the petroleum and government/aerospace applications.
- Integer/bit matrix operations used by government/aerospace.
- Floating-point calculations with a small number of bits are used by government/aerospace and astronomy applications.
- Intrinsic (log, sine, etc.) calculations (64-bit floating-point) are used by Black-Scholes algorithms in the financial community.

Since data reuse rates are so important for accelerators, common mathematical kernels related to the above areas were analyzed to determine which ones are good candidates for accelerators. Table 1 shows important kernels used in HPC and the amount of data reuse for each.

Table 1: HPC kernels and data reuse

Operation	Number of flt.-pt. instructions / Number of loads and stores	Appropriate for accelerators?	Used by
floating-point operations to memory ratio: O(1)			

DAXPY ($y(i) = y(i) + a * x(i)$)	2/3	no	all
DDOT ($s = s + x(i) * y*(i)$)	1	no	all
DGEMV (matrix * vector)	2	no	all
intrinsics (such as log, sine)	~50	yes	finance
floating-point operations to memory ratio: $O(\log_2(n))$			
1-d FFT	$2.5 \log_2(n)$	maybe – depends on interconnect	O&G, gov't/aerospace
2-d FFT	$5.0 \log_2(n)$	maybe – depends on interconnect	O&G, gov't/aerospace
3-d FFT	$7.5 \log_2(n)$	maybe – depends on interconnect	O&G, gov't/aerospace
floating-point operations to memory ratio: $O(n)$			
convolution	$0.5 n$	yes	O&G, gov't/aerospace
DGEMM (matrix * matrix)	$0.5 n$	yes	TOP500, MCAE, some LMS

4.3 Iterative refinement

Some accelerator technologies have much higher performance for 32-bit floating-point calculations than for 64-bit calculations (for example, the Cell processor). One idea for harnessing accelerator performance is to solve systems of equations with direct methods using 32-bit precision and then refine the solutions using a small number of 64-bit precision calculations.

One technique to accomplish this is iterative refinement. The math library LAPACK already contains iterative refinement techniques, but they are used to improve the accuracy of a solution within the same precision (routine DGESVX for example). Jack Dongarra and others have proposed extending the LAPACK functionality for mixed precisions. So an accelerator might solve a system of equations with 32-bit precision and then the accelerator or CPU could finish the solution by using 64-bit precision. The downside is that not all systems are equations are amenable to this technique. One estimate is that 20% of systems fail to produce a solution when using the iterative refinement technique.

Due to this lack of reliability many are dubious about using iterative refinement in production codes. Also the single precision/double precision gap in accelerator performance will likely decrease over time and the importance of this technique may decrease. Note that this technique still uses order (n^3) operations and should not be confused with general iterative solvers which have order (n^2) operations.

5 Investigation Findings

The investigation has resulted in number of presentations to many internal groups and customers. Members of the accelerator team represented HP in a panel discussion on accelerators at the IDC HPC User Forum Meeting in September, organized a panel session at HP-CAST in November, had a poster with RapidMind on display at SC06, and represented HP at the CHReC kickoff meeting in December. An abstract was also submitted to HP's TechCon07 conference based on the accelerator work. Other progress the team has made is split into the different accelerator families.

5.1 GPUs

5.1.1 RapidMind

RapidMind's metaprogramming language is easier to use than typical shader languages and represents an alternative to OpenGL for HPC customers. To perform a SAXPY operation, users can write code to move vectors X and Y to the GPU, perform $Y = Y + a \times X$, and then move the result back to the CPU. The code to implement this appears as follows:

```
// Code for SAXPY (y(i) = y(i) + a * x(i)) in RapidMind
n4 = n/4;
Array<1,Value4f> y_gpu(n4), x_gpu(n4);
memcpy(y_gpu.write_data(), y, 4 * n4 * sizeof(float));
memcpy(x_gpu.write_data(), x, 4 * n4 * sizeof(float));
y_gpu = saxpy(y_gpu,x_gpu);
memcpy(y, y_gpu.read_data(), 4 * n4 * sizeof(float));
...
Program saxpy = RM_BEGIN_PROGRAM("gpu:stream") {
    InOut<Value4f> y_input;
    In<Value4f> x_input;
    y_input = y_input + x_input * *alpha;
} RM_END;
```

The following charts are taken from the RapidMind/HP poster shown at SC06 (http://sc06.supercomputing.org/schedule/event_detail.php?evid=5220). Benchmarks were run on the latest generation systems: NVIDIA 7800 GTX, ATI x1900XT, Intel Woodcrest (2.6 GHz), and AMD Opteron (2.6 GHz). Note that the rates shown include the time spent to pass the necessary data to and from the accelerator. Many vendors ignore this very important data when showing benchmark results. The first Figure shows an ideal situation for accelerators; the European Option Pricing using Black-Scholes code. There's very little data to pass to the accelerator and a large number computations to perform.

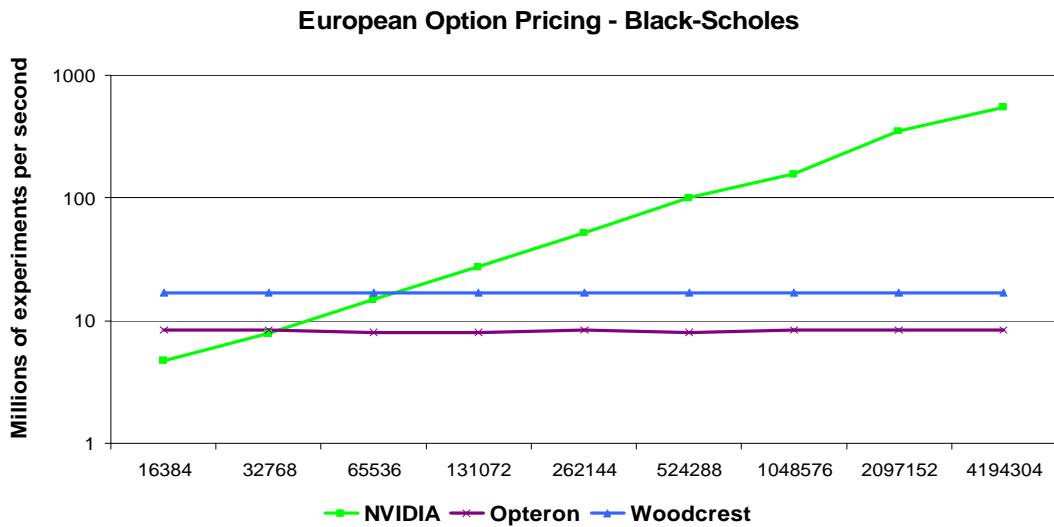


Figure 1: Measured performance on 32-bit Black-Scholes

Matrix-matrix multiply has also has good data reuse is the most important math kernel used in HPC. Most matrix-matrix multiplications in HPC use double precision data, but since GPUs only perform 32-bit floating-point the single precision version (SGEMM) was benchmarked.

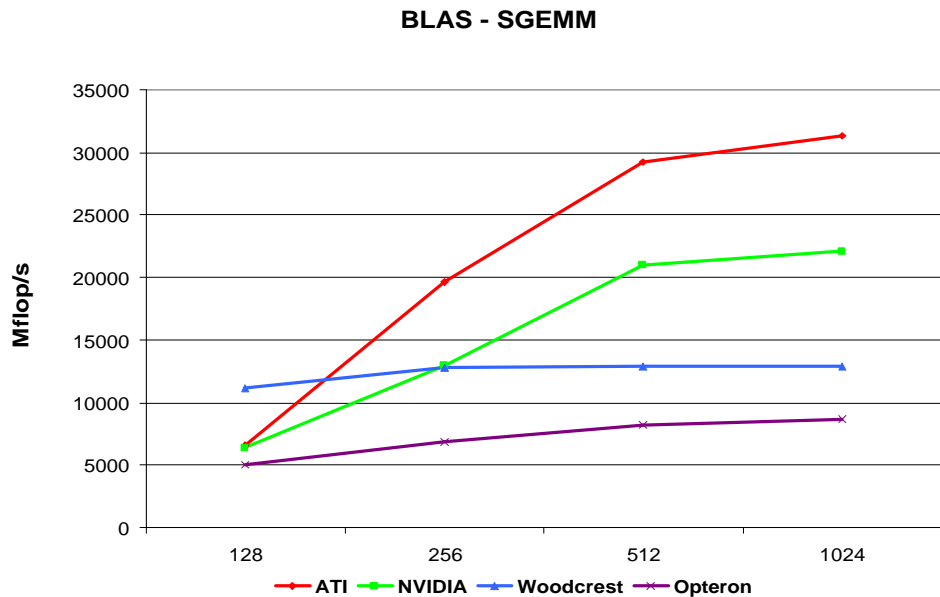


Figure 2: Measured performance on 32-bit matrix-matrix multiplication

FFTs are used extensively by oil & gas and government customers. While FFTs are widely regarded as having good data reuse compared to most codes, the reuse is not nearly as good as in matrix-matrix multiplication. Therefore, the performance of this kernel is highly dependent on the interconnect rates. The 32-bit calculations shown below are common.

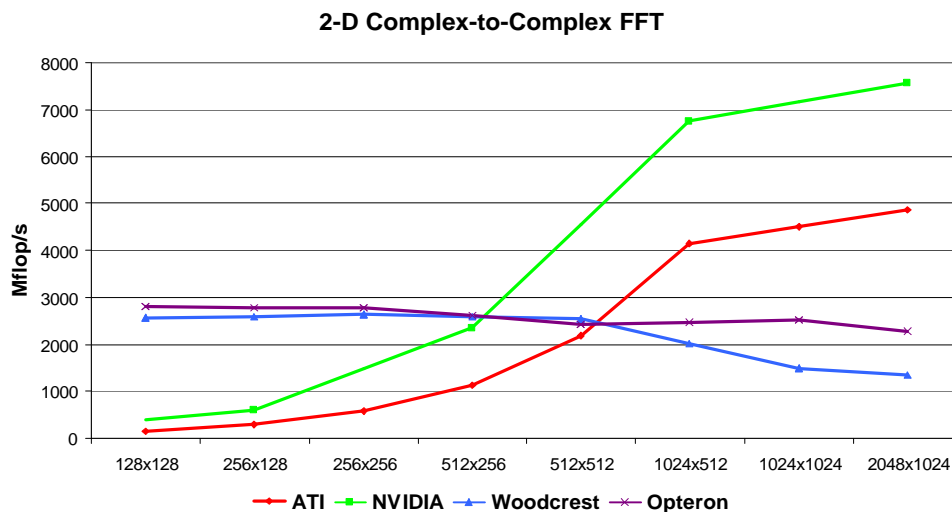


Figure 3: Measured performance on complex 2-d FFTs with real and imaginary components of size 32-bits

5.1.2 PeakStream

PeakStreams' Streams programming language is also easier to use than OpenGL. They have implemented a subset of the BLAS routines for execution on the GPU. So to perform a SAXPY operation, users can write code to move vectors X and Y to the GPU, perform $Y = Y + a \times X$ or call `cblas_saxpy()` to perform the SAXPY on the GPU, and then move the result back to the CPU. The code to implement this appears as follows:

```

{
  Arrayf32 X_GPU, Y_GPU;
  X_GPU.writel(n, X, SP_MEMORY_NORMAL);
  Y_GPU.writel(n, Y, SP_MEMORY_NORMAL);
  Y_GPU = Y_GPU + ( a * X_GPU);
  //  cblas_saxpy( n, a, X_GPU, incX, Y_GPU, incY );
  Y_GPU.readl(Y, n * sizeof(float));
}

```

AMD loaned HP the latest high performance ATI FireStream GPU to benchmark the PeakStream code. This board is connected to the GPU by PCI-e x16 which has a theoretical peak of 4.0 GB/s. Writing code using PeakStream's API gave measured rates of 0.8 GB/s to download (move the data from the CPU to the GPU) and 0.4 GB/s for readback (to move the data from GPU to CPU). PeakStream provides a sample matrix-matrix multiplication code which was modified and used to obtain the results shown in Figure 4. The total performance line shows the result of passing matrices A and B to the GPU, calculating a matrix multiply $C = A \times B$, and passing the resulting matrix C back to the CPU; this rate is approximately 60 Gflop/s. The GPU performance rate is a lower bound for the GPU performance: the data was passed once, but used in ten matrix multiply calculations, and the result passed back; this rate is approximately 100 Gflop/s.

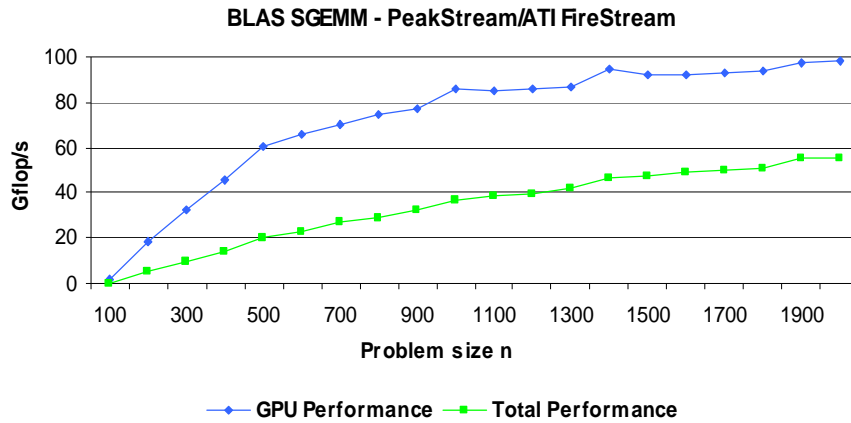


Figure 4: Measured performance on 32-bit matrix-matrix multiplication using PeakStream software on ATI FireStream GPU

5.1.3 RapidMind and PeakStream Comparison

Both RapidMind and PeakStream are Streams languages that can be used for technical applications and as expected they have several things in common. They are both embedded in C or C++ routines and can be used with common C/C++ compilers. Streams languages are more cumbersome than C/C++ since users operate on streams of data and things like array indices may not even be available. Of the two languages PeakStream is easier to use, because it obscures the fact that most GPUs operate on 4, 32-bit tuples of data. RapidMind makes the 4-tuple accessible to programmers so users can manipulate data at a finer level. PeakStream provides a library of BLAS routines written for the GPU that users can access. Note that these routines assume that the necessary data has already been moved to the GPU. RapidMind generates OpenGL code and is available for both ATI and NVIDIA processors whereas PeakStream uses the ATI CTM interface and is only available on ATI GPUs. The following table compares the two products.

Table 2: Comparison of RapidMind and PeakStream products

	Ease of Use	Generates	Supports
RapidMind	Easy. Exposes 4-tuple to programmer.	OpenGL	AMD/ATI, NVIDIA
PeakStream	Very easy. Obscures 4-tuple. Single precision BLAS on GPU.	CTM	AMT/ATI

5.2 FPGA

At the beginning of this investigation we wanted to determine the performance of FPGAs using C like languages. The two that were most often discussed in HPC are Celoxica's Handel-C and Mitronics Mitrion-C.

5.2.1 Celoxica

Our team took two days of training at Celoxica's office in Austin, Texas and purchased a PCI board from Celoxica that contained an FPGA and connected to our system by PCI. Upon returning to our site we spent months trying to get the SAXPY routine to work. This code

consists of a loop around the statement $y(i) = y(i) + a \times x(i)$. After working through several major issues, Celoxica sent their Dallas employee to the HP Richardson site for a day of work. After this joint work we're now able to execute the kernel on our in-house system. The most complicated part of the code is the data passing component, but the following shows the code that is run on the FPGA to accomplish a SAXPY.

```
{
  DsmWord yDelay;
  unsigned 32 temp;
  par {
    mult32m24p04(a,0,x,0,&temp,__clock,__reset);
    add32m24p05(temp,0,yDelay,0,&result,__clock,__reset);
    Shifter( y, yDelay, 4 );
    Shifter( xyValid, resultValid, 10 );
  }
}
```

Since the HP system has a very slow interconnect, the performance rates are very low as expected. This code will be rehosted to HT connected systems when these are available. Our experience is that the FPGA development environment and methodology is complex and that the software vendors need to be directly involved to get very simple HPC running on FPGAs.

5.2.2 Mitronics

Discussions with Mitronics have continued over several months. One of our team attended training on Mitrion-C, but we do not have a system to test it on at this point. Mitronics use Nallatech's boards to host their software. We are supposed to receive a Nallatech board to experiment with in February 2007.

5.2.3 CHReC

HP is a charter member of the NSF Center for High-Performance Reconfigurable Computing (CHReC). As part of the membership HP attended the kick-off meeting in December and voted on which FPGA programs at George Washington University (GWU) should be funded in 2007. Each program will have on average three graduate students and GWU had five proposals, of which three could be funded. The two programs that HP was most interested, one relating to compiler efficiency and the other relating to profilers for FPGAs were both approved.

5.3 ClearSpeed

ClearSpeed has a math library that can be utilized by linking and setting an environment variable. The ClearSpeed boards are only used if it is determined that the problem size is large enough so that there is a benefit from using the boards. HP has several ClearSpeed loaner boards and we've run many tests using their library. While there are many objects in this library, nearly all are stub routines which do not do any work. The one routine that ClearSpeed has spent a lot of time optimizing is the double precision matrix-matrix multiply code DGEMM which is use in the HPL benchmark sited Top500.

The performance obtained on DGEMM is close to 50 Gflop/s per board for very large matrices. ClearSpeed also issued a press release showing the speedup of an HP cluster using their system (<http://www.hpcwire.com/hpc/860356.html>) and performing the HPL benchmark. As discussed above the first accelerator enhanced system to make the Top500 list uses ClearSpeed boards.

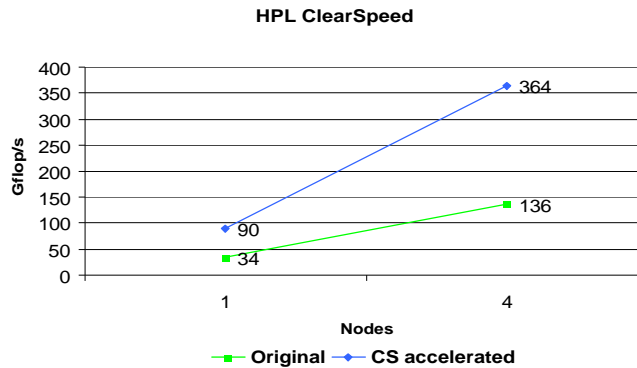


Figure 5: HPL benchmark speedup of HP DL380G5 (2 Woodcrest processors per node) using ClearSpeed boards

It is more difficult to show the advantage of accelerating FFTs on the ClearSpeed board since it is connected by PCI-X and this bandwidth negates any computational advantage. They have a beta version of their SDK which should allow us to get some additional results in the near future.

HP has also measured Monte Carlo simulation code provided by ClearSpeed. This is an ideal code for accelerators since there is a great deal of data reuse. The ClearSpeed code runs ten times faster than the non-tuned CPU code but only twice as fast as tuned CPU code which used Intel's math library.

HP has begun experimenting with the ClearSpeed **Cⁿ** programming language. ClearSpeed started with the C language and added new datatypes "mono" and "poly". The default datatype is mono, that is there is only one instance of this data. Poly data allows all functional units to be working simultaneously on portions of the data. A SAXPY was implemented using **Cⁿ** where the original mono data was created on the ClearSpeed board, converted to poly data, operated on, and converted back to mono data. The original SAXPY code consists of a loop around the statement $y(i) = y(i) + a \times x(i)$. Of course in a real application the data might begin as poly data or be converted as soon as possible, but the example below gives a feel for how the language works. Note also that the data used below was initialized on the ClearSpeed board and does not include the code necessary to copy the data from the CPU to the ClearSpeed board and back to the CPU.

```

poly float apoly, xpoly, ypoly;
mono int * poly xsrc_addr, ysrc_addr;
mono numberofpes;
poly short penum;
numberofpes = get_num_pes(); // returns 96
penum = get_penum();
memcpym2p(&apoly, a, sizeof(float));
for (i = 0; i < *n; i+= numberofpes ) {
    if (penum <= (*n - 1)) {
        xsrc_addr = x + i + penum;
        ysrc_addr = y + i + penum;
        memcpym2p(&xpoly, xsrc_addr, sizeof(float));
        memcpym2p(&ypoly, ysrc_addr, sizeof(float));
        ypoly = apoly * xpoly + ypoly;
        memcpyp2m(ysrc_addr, &ypoly, sizeof(float));
    }
}

```

5.4 Customer Codes

We had hoped to be working more closely with customers by this point. This part of the investigation is progressing though and the Next Steps section discusses future collaborations.

5.4.1 Lucas solver code

Robert Lucas, Division Director of Computational Science, Information Sciences Institute, University of Southern California, produces a linear equation solver that is used in the ANSYS and LSTC ISV packages. He has also created a linear equation solver benchmark for accelerators. It measures the performance of a dense kernel of a symmetric indefinite multifrontal solver by generating a dense triangular matrix and partially factoring it. The code can be run using single or double precision calculations and includes two techniques for the factorization: a left-looking factorization without pivoting and factorization using Duff-Reid pivoting.

Our team attempted to run Lucas' benchmark on ClearSpeed boards using double precision calculations, but there was no advantage since the problem sizes were not large enough to engage the ClearSpeed boards. ClearSpeed claims they are working on making Lucas' code perform better on their boards.

The single precision version was successfully run on ATI GPUs using the PeakStream software. A version of the matrix-matrix multiply routine SGEMM was written using PeakStream and this routine called instead of the original SGEMM code. For eliminating 2000 out of 9000 equations overall performance of 12 Gflop/s was measured with many of the SGEMM calls running at over 24 Gflop/s.

5.5 Current Status

The following table shows the current of performance on accelerators and CPUs.

Table 3: Theoretical vs. measured Gflop/s rates on 32 and 64-bit matrix-matrix multiplication

	Theoretical SGEMM	Measured SGEMM	Theoretical DGEMM	Measured DGEMM
GPU ^a ATI FireStream	250	60 ^f	not possible	not possible
FPGA ^b Virtex-4 V4LX200 ^d	40 ^c		4 ^d	
ClearSpeed	96		96	45 ^f
Cell ^e	205	201	15	12
Xeon 3.8 GHz (1 core)	15	11 ^c	8	6 ^d
Woodcrest 2.67 GHz (1 core)	21	14 ^f	11	8 ^f

Notes:

a – 1 M Black-Scholes iterations run 10x typical CPU rates

b – On some Virtex-4 chips four-bit complex dot products have a peak that corresponds to 480 Gflop/s

c – Results from Xilinx presentation

d – DGEMM without transfer rates has been measured at 8 Gflop/s by Scott Campbell at University of Colorado

e – All results are from IBM paper “Cell Broadband Engine Architecture and its first implementation”

f – Measured by HP accelerator team

The following table shows a relative summary using various metrics.

Table 4: Summary of accelerator technologies applicable to HPC

	GPU	FPGA	ClearSpeed	Cell
Price	\$	Chip \$\$, Package \$\$\$	\$\$\$	Chip \$, Package \$\$\$
Volume	High	Medium for chip	Low	High for chip
Power	High	Low	Medium	Medium
Good at	Graphics, 32-bit	Integer, small number of bits	64-bit	Graphics, 32-bit
64-bit floating-point available, performance	No, medium in 2007	Yes, low	Yes, high	Yes, low in 2007, high in 2008
IEEE-754	No	No	Yes	No
ECC	No	No	Yes	No
HPC Performance	Low - Medium	Medium	High	Medium - High
Does it need HPC to succeed?	No	No	Yes	No

5.6 Conclusions and predictions

There are multiple families of accelerators suitable for executing applications from portions of HPC space. These include GPGPUs, FPGAs, ClearSpeed and the Cell processor. Each type is good for specific types of applications, but they all need applications with a high calculation to memory reference ratio. They are best at the following:

- GPGPUs: graphics, 32-bit floating-point
- FPGAs: embedded applications, applications that require a small number of bits
- Clearspeed: matrix-matrix multiplication, 64-bit floating-point
- Cell: graphics, 32-bit floating-point

Common traits for accelerators today include slow clock frequencies, performance is through parallelism, low bandwidth connections to CPU, and the lack of standard software tools. Today accelerators are connected to the CPU by PCI-X, PCI-e and HyperTransport. Future trends include

- faster clock periods, better nm technology, more parallelism
- improved interconnect bandwidth (PCI-e Gen2, later HT) and latency and the ability to

achieve a higher percentage of the interconnect performance

- multicore/multichip accelerators
- heterogeneous processors, that is, some number of CPU cores + some number of accelerators cores of various types
- a better software development environment – users need a standard C compiler which generates code for whichever accelerator is most appropriate and a complete software tool chain that includes debuggers and profilers

A Appendix – Accelerators and ECC

Contributed by Kevin Harris

Robust error correction code is very important for large scale clusters. Regarding soft errors on the accelerators:

- No accelerator is even in the same ballpark as ordinary x86, x86-64, or IA-64 processors when it comes to detecting and correcting soft errors.
- The risk of a soft error leading to a wrong answer in a large production cluster is high enough that customers need to be aware of the situation.
- As technology improves and feature sizes decrease, soft error rates go up because ever lower energy particles can cause soft errors. So systematic detection and suppression of soft errors is essential in production compute environments.

A.1 GPUs

All existing GPUs are unprotected. Any soft error will lead to a wrong answer somewhere. The only discussion is how likely it is, which can't be answered easily without testing in atmospheric or in a neutron lab. We assume that Fusion processors from AMD (ATI) will have adequate protection, but it is unclear how other GPU processors will deal with the issue.

A.2 FPGAs

Ordinary CPUs and GPUs (and Cell, etc.) can suffer soft errors in the data cells, flipping a bit at random in a data location (e.g. cache or register) or in a data path. ECC methods can provide Single Error Correction with Double Error Detection (SECDDED) and work well in these environments. FPGAs can suffer the same kinds of errors, but no design for providing ECC in the data paths has been provided by any of the FPGA vendors. Some provide ECC for the data portions of the FPGA, and paths to main memory. But the FPGA can suffer a different kind of failure as well: a neutron strike to the "configuration" memory that determines the function of the FPGA. Both major FPGA vendors (Xilinx and Altera) as well as Velogix have recognized this problem and addressed it, but there is no evidence that any of the software tools vendors we've been discussing the "C to gates" capability with have attempted to exploit the chip-level configuration protection mechanisms yet. The only obvious alternative solution is redundancy: duplicate the function in logic and compare the results - either 2x or 3x (Triple Modular Redundancy) depending on how quickly you want to recover from the problem. This effectively reduces the amount of usable area by up to 2/3, effectively setting back FPGA sizes by a full generation. FPGA chip vendors now recognize the problem and are pursuing solution approaches with the design software providers. Software for designing applications using FPGAs have been slow in implementing a solution to this problem.

A.3 ClearSpeed

The current generation of the Clearspeed processors has some ECC functionality however it is not as robust as that used by CPUs which may lead to issues in production environments. Based on customer requirements, they are expected to address this issue in future products.

A.4 Cell

IBM went to the trouble to provide full ECC memory protection (but only supports XDR Rambus memory, which limits the actual board level implementations severely) as they would normally

for their other Power architecture processors. But they provide no ECC protection for the internal data structures: the Element Interconnect Bus (EIB) or the local memory associated with the Synergistic Processing Elements (SPEs). This is viewed as a limitation in large configurations.