Science & Technology
Facilities Council

Technical Report
DL-TR-2009-001

Evaluation of ClearSpeed Accelerators for HPC

IN Kozin

August 2009

ISSN 1362-0207

Evaluation of ClearSpeed Accelerators for HPC

Igor N. Kozin
Computational Science and Engineering Department, STFC Daresbury Laboratory, Daresbury, Warrington, Cheshire, WA4 4AD, UK

Abstract

ClearSpeed Accelerator Tera-scale System (CATS) has been evaluated in an HPC setup by running a number of benchmarks and real applications. Our tests demonstrated a significant advantage of the technology in terms of performance per Watt and density of compute power over the current commodity offerings. However this needs to be set against considerations of "total cost of ownership" and the real benefit of the ClearSpeed acceleration technology currently appears to be the ability to deliver greater performance in HPC data centres which are limited by the power and/or space available rather than purchase and running costs.

23/10/2008 – first release
10/12/2008 – minor revision

Introduction

It is a matter of fact that High Performance Computing (HPC) is nowadays mostly commodity driven. While HPC installations continue to grow in their number and size, the price of HPC components has decreased dramatically in recent years. Not only has this been due to the HPC technologies becoming more ubiquitous and hence dropping in price, but also to a large extent due to a wide spread use of commodity processors. The statistics of the latest Top500 HPC list[1] is highly indicative: the number of x86-based entries (both 32- and 64-bit) increased from non-existence (only 1% in 2000) to more than 80% in 2008.

It has been noted that the aggregate performance of the Top500 list grew quicker than the Moore's law. This indicates that the performance growth was at least partly fuelled by the increase in system size rather than only the power of individual processors. The latest (June 2008) Top500 list featured a petaflop entry for the first time. While it is very likely that the number of petaflop systems will continue to grow, it is also clear that there is a natural limit on the total power which can be supplied to a large computer installation. In a way, this limitation is an upper boundary on the 'power wall' hit by the race to increase the processor clock a few years back. Faced with the technological difficulties of power dissipation and parasitic currents, the clock frequency was frozen at around 3–4 GHz. Instead multi-core processors started to proliferate offering higher performance while keeping the thermal envelope at the same level.

The power limitations put on the processors prompted a rethink of mainstream computing and a turn to, sometimes very aggressive, parallel processing. This departure saw radical innovations such as SiCortex which offers the use of very low power processors as building blocks.[2] At the same time accelerators also became more viable technologies. Perhaps the

[1] http://www.top500.org
[2] http://sicortex.com/

most popular contenders are ClearSpeed, Cell Broadband Engine and GPGPU (General Purpose GPU). They all have their pluses and minuses. GPUs are a commodity and therefore low cost. The down side is they are mostly oriented at 32-bit computing because that satisfies the gaming world. Although 64-bit capabilities have recently been added by both nVidia and ATI/AMD, the performance is much lower than in 32-bit. The Cell is a commodity as well because it is used by Sony in PlayStation3 but again its 64-bit performance is rather low since it is done in emulation. Trying to capitalise on the interest in the Cell processor, IBM started manufacturing PowerXCell 8i which offers 64-bit processing in hardware. However the latter seems to fail in being a commodity and therefore a much higher price tag should be expected. In stark contrast to the two technologies ClearSpeed offered single and double precision from the very beginning, aiming specifically at general purpose computational acceleration but at a cost. One of the key ClearSpeed points is very low power and therefore potential for very high computing density.

It will be only natural if commodity systems continue to dominate the Top500 list due to their low cost and the ever increasing number of cores. However, if the performance boundaries need to be pushed beyond the average then alternative technologies need to be considered. It is not too surprising that the current number one system is Cell based. Thus it is worth keeping an eye on the new and emerging technologies. In this paper we present a report on our ClearSpeed evaluation system. The benchmarking results are put into context by comparing the performance with that of modern commodity multi-core servers. The implications of performance versus power and the total cost of ownership are also discussed. No comparison to GPGPUs or Cell is made at this time as these studies are being planned for the future. However it is worth mentioning here a recent report on the use of GPU in HPC published by Hewlett-Packard.[3]

ClearSpeed ecosystem

The architecture

At the heart of the ClearSpeed offering is the CSX600 processor, which is a system-on-a-chip (SoC) comprising 96 processing cores, an internal network, embedded SRAM, interfaces to external memory and inter-processor I/O ports. ClearSpeed call the aggregation of 96 cores a multi-threaded array processor (MTAP) and each core is referred to as a Processing Element (PE). Each PE contains a 32/64-bit Multiplier and Adder (single and double precision) and 6KB of SRAM. Unlike in some other accelerator technologies, the arithmetic conforms to the IEEE 754 standard. The processor is typically set to work at 210 MHz thus making the theoretical peak performance of a single processor 40.3 GFLOPS in double precision.

The actual product available for sale is the ClearSpeed Advance e620 card. Early versions had a PCI-X interface which was later superseded by PCI Express x8. Each card contains two CSX600 processors and 1 GB 200 MHz DDR2 SDRAM. The memory has Error Correcting Code (ECC) and therefore the cards can be used in large HPC installations.

[3] G. Lupton, and D. Thulin, “Accelerating HPC using GPU’s” (2008), http://www.hp.com/techservers/hpccn/hpccollaboration/ADCatalyst/downloads/accelerating-HPCUsing-GPUs.pdf

ClearSpeed asserted that the card is capable of a 66 GFLOPS sustained performance doing DGEMM which corresponds to 82% efficiency.

The card's energy consumption is only 33 Watts which translate into an impressive 2 GFLOPS per Watt. However, a card requires a host to be plugged in which dramatically increases the total power. In order to address the needs of the HPC market and improve the overall power efficiency and compute density in a large installation, twelve e620 cards were aggregated together to produce so-called CATS 600 (Clearspeed Accelerated Tera-scale System). Such a system has a theoretical peak performance of 968 GFLOPS while consuming up to 850 Watts (our measurements showed a lower number, see below). Physically a CATS is a noisy 1U rackmount server which is composed of two independent halves each containing six cards and a PCI Express x8 interface. Thus a CATS can be attached either to one host server with two PCI Express x8 slots or two servers with one.

Very recently ClearSpeed introduced a new CSX700 processor which has the computing power of two CSX600 combined. It also features a number of improvements including integrated PCI Express x16 (however there is still no Gen 2 support) and DDR2-533 memory controllers. The successor of e620, the e710 card has only one processor but the processor is clocked at slightly higher frequency (250 MHz). All these technological innovations allowed reducing cost/performance by more than a factor of two as well as further improving power efficiency. The new card consumes only 25 Watts (3.84 GFLOPS per Watt). There is a new CATS as well, CATS 700, with peak performance of 1152 GFLOPS and typical power 500 Watts. However the CATS connection is still dual PCI Express Gen1 x8 and therefore suffers from the same limitation as the CATS600 used in this evaluation.

Programming ClearSpeed

The CSX600 processor can be programmed in a SIMD (Single Instruction Multiple Data) style using C^n which is a slightly restricted C language with extensions.[4] In order to differentiate scalar and parallel operations the new mono and poly data types are introduced. The mono data type has only one instance which is accessible by all the PEs. The poly data type has multiple instances which exist on the PEs and are therefore visible only to the corresponding PE. Pointers based on the two data types are permitted thus resulting in four different types of pointers. Finally the memory access can be managed either using semaphores or the ClearSpeed API (CSAPI).[5] The ClearSpeed device driver has a very low-level, hardware-oriented API which is not intended to be used directly by the programmer. The device-level interface provided by the CSAPI has basic support for loading and executing code on CSX processors. It also allows data to be transferred between the host and the accelerator memory, and supports access to hardware features such as semaphores, DMA operations and memory management.

Since the ClearSpeed cards are off-load accelerators they require a host processor to initiate the device and get the data in and out. Respectively, an accelerated application can be divided into the host program running on the host processor and the ClearSpeed code running on the card. Therefore a ClearSpeed enabled platform constitutes a heterogeneous architecture in that the binary structure of executables for the host and the accelerator are incompatible.[6] The host executable is a regular binary which, in the process of its execution,

[4] ClearSpeed Software Development Kit Reference Manual Version 3.0.
[5] ClearSpeed CSX600 Runtime Software User Guide Version 3.0.
[6] Other examples of heterogeneous accelerator architectures are GPGPU and Cell Broadband Engine.

loads a ClearSpeed binary into the board, passes the data, launches the execution, collects the results and perhaps repeats these steps a few more times.

ClearSpeed API and Libraries

There are three modes of using a ClearSpeed accelerator which can be listed in order of increasing sophistication and required effort as following:

- interception of standard library calls (e.g. BLAS),
- using CSAPI and accelerated libraries (e.g. BLAS, LAPACK and FFT),
- using CSAPI and native ports to ClearSpeed architecture.

In the first mode, the user code does not need to be modified at all as all the trouble of communicating to the board is taken care of by the ClearSpeed provided libraries. The only condition needed for this to work is that the external library should be a shared library (or a dynamic library under Windows). Then the ClearSpeed library, placed earlier in the library path, intercepts the function calls it can accelerate and redirects them to the accelerator board. This is the mode which allows the acceleration of third party applications such as MATLAB without instrumenting them. While this method is the least cumbersome from the user standpoint, it may incur too much communication overhead as each redirection requires the loading of ClearSpeed binary, the communication of the user data and the transfer of the results back. These shortcomings lead naturally to the second method in which the user data can be passed to the board once and then repeatedly operated on by calling ClearSpeed accelerated libraries. The downside is obviously that the relevant part of the user code needs to be ported in order to be able to use CSAPI. Finally the user may resort to writing a custom C^n code if this offers much better performance or the above methods are unacceptable. It should be noted that it is possible to write a user code exclusively in C^n, compile it and execute by running the ClearSpeed code loader – `csrun`. However, using a host program offers more flexibility and the benefits such as management of ClearSpeed resources, status and error checks and the possibility of using multiple hosts and accelerators (e.g. through MPI).

The latest version 3.1 of the SDK saw the addition of CSPX API which provided a higher-level API that can be more easily used by an application developer who does not want to concentrate on communication but rather on quick and efficient implementation of an algorithm. Although there is only a C API and no Fortran API, calls from Fortran are possible through C wrappers. In addition to APIs, ClearSpeed offer the following libraries:

- CSXL – ClearSpeed math library,
- CSDFT – ClearSpeed Discrete Fourier Transform,
- Random number generators.

CSXL contains DGEMM, ZGEMM, DTRSM, which are Level 3 BLAS functions, and DGETRF, DGETRS, DGESV, DPOTRF, DPOTRS, DPOSV, DGEQRF, DORGQR, DORMQR, which are part of LAPACK. This is clearly a very small subset with an emphasis on double precision. There is also a card-side version of DGEMM. Overall it is fair to say that ClearSpeed did a good job providing all the necessary tools and their SDK is very mature.

The setup

We deployed two CATS 600 in our Intel Woodcrest cluster. The nodes are Dell PowerEdge 1950 servers with two PCI Express x8 extension slots one of which is occupied by an InfiniPath card. Therefore we had to connect each of our two CATS to two servers. Here is a summary of technical details of our setup:

- Processor: Xeon 5160 3.0 GHz (dual-core Woodcrest)
- Chipset: Intel 5000X (Greencreek)
- System bus: 1333 MHz
- Memory: 8 GB 667 MHz FB-DIMMs
- Software stack: SuSE 10.1, InifiniPath 2.1, Intel 10.1 compilers, Intel MKL 9.1, ACML 4.0.1, ClearSpeed SDK 3.0

The installation was remarkably quick and straightforward. It basically required putting the two CATS into the rack, plugging the adapter cards into the host nodes and cabling things together with the provided cables. The CATS needs to be powered up first and only then can the host be started up. The switching off happens in reverse order. However, if a remotely controlled power distribution unit (PDU) is available then it should be possible to integrate the CATS with appropriate cluster management tools for remote cluster operation and management.

At the time of the CATS delivery the latest SDK release was 2.5 which does not support SuSE 10.1. Fortunately SDK 3.0 beta was available which did support SuSE 10.1 and the official release of SDK 3.0 followed very shortly afterwards. The SDK was installed on the head node with the tools directory exported via NFS to the compute nodes. The ClearSpeed compiler requires a license server which was also installed on the head node. No other package installations were necessary. The SDK comes as an rpm package containing files and a post-installation script. The script was extracted and executed on the host nodes. This loaded the ClearSpeed drivers and the CATS were ready for operation. The only problem was that one of the adapter cards was not firmly seated in the host. This was quickly troubleshooted with the help of ClearSpeed engineers. Although the cluster can also be booted to Windows CCS 2003 and Windows platform is supported by ClearSpeed, an evaluation and assessment under Windows was not attempted.

As stated above, each host node is powered by two Intel Xeon 5160 processors each of which is capable of two fused add-multiply operations. Thus each core yields 12 GFLOPS which translate to 48 GFLOPS per server. Since this level of performance is a bit out of date we also used our HP ProLiant DL160 G5 servers with Intel Harpertown E5472 processors clocked at 3.0 GHz. These are quad-core processors and capable of 48 GFLOPS each or 96 GFLOPS per server.

Benchmarks

In order to assess the performance and usability of ClearSpeed cards we used a mix of benchmarks and real applications. There are not too many applications yet which can be readily used with ClearSpeed boards. One of the early applications ported to ClearSpeed is the molecular dynamics package AMBER. Unfortunately the port of AMBER to SDK 3.0 was not available at the time of testing. Another interesting application is BUDE (Bristol University Docking Engine). It is still in development and aimed at performing Monte Carlo-based drug docking. There are a number of other applications such as MATLAB or Computational ElectroMagnetics which can be accelerated by redirecting calls from standard libraries. Finally there are a number of ported code kernels relevant to financial services. We selected the following set of benchmarks which is described in detail below:

- matrix-matrix multiplication (DGEMM from BLAS),
- High Performance Linpack (linear solver),
- MOLPRO (computational chemistry package),
- European option pricing (Monte Carlo simulation kernel).

Before we move on to the benchmarks it is worth investigating the I/O subsystem of our setup because one PCI Express link shared by six cards may provide reason for concern. To this end we performed a number of write and read bandwidth tests between the host memory and the cards. CSAPI offers synchronous and asynchronous ways of transferring the data to and from the cards. In the synchronous mode the host was able to write at about 440-500 MB/s and read at about 600-630 MB/s. In the asynchronous mode overlapped transfers to several cards was possible. While an increase in the bandwidth was observed while doing asynchronous transfer, and this transfer mode is certainly recommended over the synchronous one, the bandwidth was quickly saturated by two cards. Thus three or more cards started competing with each other over the bandwidth. The maxima that we could achieve were about 1200 MB/s when writing and slightly over 1300 MB/s when reading. However in practice this I/O bottleneck can often be effectively hidden by doing I/O asynchronously and overlapping it with some useful work.

Transfer	Write	Read
Synchronous	500 MB/s	630 MB/s
Asynchronous	1200 MB/s	1300 MB/s

Table 1: Write and read bandwidth from the host to the card (see text for details).

Matrix-matrix multiplication

As mentioned above ClearSpeed provide an accelerated linear algebra library CSXL as part of their SDK. CSXL is relatively simple and straightforward to use. There are two methods. One method is to rename the standard library the user application is linked against and name the ClearSpeed provided library as the standard library. Another method is to directly link (or re-link) the user application against the ClearSpeed library. Then depending on the function, the ClearSpeed manager library will either pass the relevant call to the ClearSpeed board or to the standard library. The pointer to an actual BLAS and a LAPACK is set via the environmental variable CS_HOST_BLAS. As discussed above this is the easiest way of using ClearSpeed cards and as such not the most efficient one since it requires data transfers to and from the card with each call. Nevertheless when the matrix size is sufficiently large the overhead is comparatively small and it is possible to get performance reasonably close to the peak.

We used DGEMM as the most natural function to assess the real performance and ran it for various square matrices up to 8000 in size. The performance of a single card (two CSX600 processors) is presented in Figure 1. Only the best results of several repeated runs are presented.

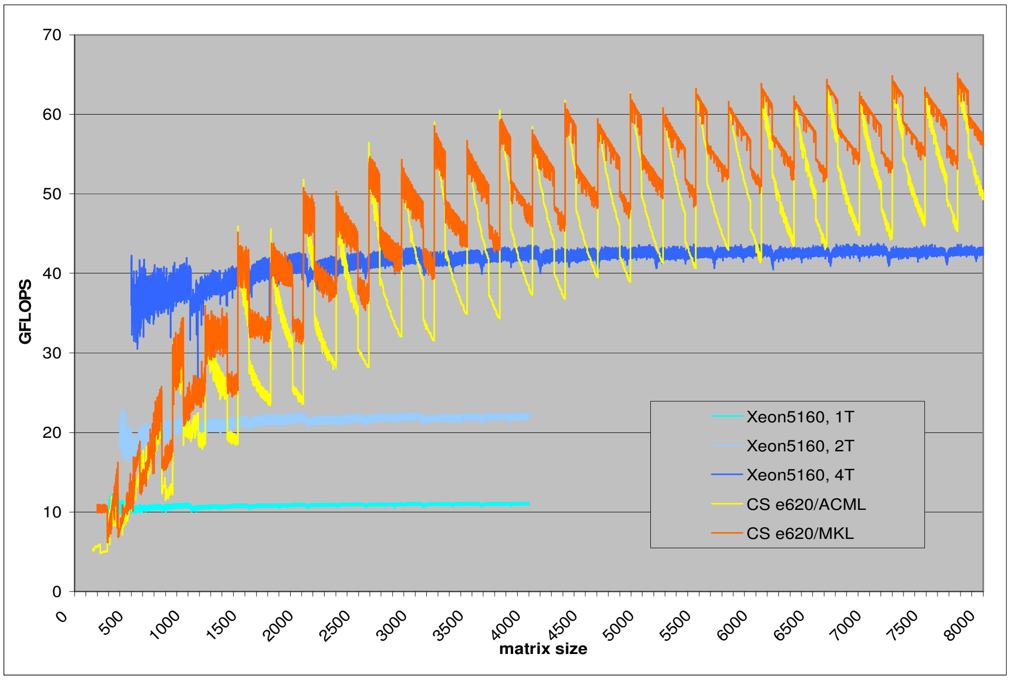

Figure 1: DGEMM performance on ClearSpeed e620 and Xeon 5160 as a function of matrix size.

It is easy to see that the ClearSpeed performance is the best for sizes *3M 96* and the worst for *3M 96–1* where *M* is an integer. It is obvious that the size should be proportional to 96 because of the 96 PEs in CSX600. However, it is less clear why all the sizes which are multiples of 96 are not equally good. It is also obvious that the performance of the host processor matters as well even though by default only the accelerator is used (if available). The default is the same as if the environmental variable CS_BLAS_HOST_ASSIST_PERCENTAGE were set to zero. The Intel MKL was clearly superior to the ACML but using different host libraries did not affect the peak performance. At the peak we saw about 63 GFLOPS which is 78% efficiency and already reasonably close to the 66 GFLOPS of sustained performance claimed by ClearSpeed. The performance can be further improved by using yet larger matrix sizes.

To put the ClearSpeed performance into context we need to contrast it with the host performance which is also shown in Figure 1 for one, two and four threads. The latter represent a fully loaded node with two dual-core processors. The performance of the host server scales linearly with the number of threads and tops at about 43 GFLOPS which is 90% of the theoretical peak. This level of performance is superseded by modern dual-socket quad-core processors which can double that number. Our HP DL160 G5 server with two Xeon E5472 processors clocked at 3.0 GHz achieved 85 GFLOPS. Thus in terms of DGEMM performance we can position a single ClearSpeed card between a Woodcrest and a Harpertown server.

Therefore a host may have a substantial computational power which should and actually can be used. All that is required is to set CS_BLAS_HOST_ASSIST_PERCENTAGE to a necessary value. Table 2 below summarises the performance results for a range of assist percentages for one of the most optimal matrix sizes (N=4320) and one of the least optimal (N=4319).

Assist %	0	5	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
N=4319	46	46	46	46	46	47	48	48	48	50	50	50	50	50	53	53	53	51
N=4320	62	64	64	67	67	67	67	67	63	63	63	61	50	50	50			

Table 2: The performance in GFLOPS achieved running DGEMM with matrix sizes 4319 and 4320 on a ClearSpeed e610 board with CS_BLAS_HOST_ASSIST_PERCENTAGE set to a range of values from 0 to 23.

One can see that the improvement is not very sensitive to the assist value. Secondly the peak performance is achieved at different values of the parameter and the relative increase is higher for N=4319. Therefore a suitable compromise needs to be found depending on the task at hand. For instance, it should be possible to smooth the sawtooth character of the picture above by setting the assist percentage to 17–18.

Finally it is worth commenting on the simultaneous execution. For the matrix size of N=4320 used in the table above the overall execution time of a single DGEMM call is about 2.9 seconds whereas the transfer time to and from the board should take about 0.7 seconds. Obviously CSXL does not transfer the whole matrix to the card at once and only then starts operating on it. If it were so, the card would not be able to handle very large matrices which do not fit into local on-board memory whereas we were able to operate on matrices larger than 1 GB. Instead it implements blocking as a normal BLAS would do. Thus the transfer time is effectively hidden. Nevertheless the simultaneous execution of four DGEMM processes does incur a noticeable contention and therefore some loss of efficiency for the reasons explained above.

HPL

The High Performance Computing Linpack benchmark[7] (HPL) is arguably the most popular HPC benchmark and the basis of the Top500 list. CSXL provides just enough functionality to accelerate HPL. The software package solves a linear system $A\ x = b$ of order N which is a rather common task in engineering. HPL is a parallel application requiring Message Passing Interface (MPI) and therefore allows the use of several ClearSpeed cards as well as several nodes simultaneously. Building HPL is relatively easy and ClearSpeed provided an HPL port among their examples. Furthermore there is a whitepaper[8] published by ClearSpeed where an increase of 34 GFLOPS per Advance X620 card has been achieved. The host servers used by ClearSpeed in that exercise are very similar to the ones used in the present evaluation; the main difference is the amount of memory – 14 GB vs 8 GB in our setup.

We built HPL with both Intel MKL and ACML support and the results of our test are presented in Table 3. The performance of a single card is indeed very high but the efficiency dropped from around 80% for DGEMM to around 50%. Expectedly the performance with MKL is better than ACML and this stresses again the importance of the base library.

[7] http://www.netlib.org/benchmark/hpl/

[8] "Industry leading LINPACK performance per watt for accelerated industry-standard systems", ClearSpeed (2007). http://www.clearspeed.com/docs/resources/Performance_LINPACK_06_19_07.pdf

number of processes	matrix size	CS performance with MKL / GFLOPS	CS performance with ACML / GFLOPS
	one node		
1	27648	43	45
2	27648	65	60
4	27648	89	83
	four nodes		
4	27648	106	
4	55296	156	
8	55296	247	
16	55296	355	

Table 3: HPL performance on one and four nodes with attached ClearSpeed. In the latter case the processes are equally distributed between all four nodes.

While the single card performance is in reasonable agreement with the ClearSpeed number in the whitepaper (please notice that ClearSpeed used host assist at 25% in their benchmarking whereas we did not use it at all) it is easy to see that the scaling is not very good. The increase in performance from one to two processes on a single node is 51% whereas from two to four is only 37%. In contrast if each card has its own node then the increase is much higher. This seems to point again to the communication bottleneck. Nevertheless, the aggregate performance of both CATS servers running HPL seems to be impressive – 355 GFLOPS.

The above results need to be contrasted with the pure host performance. The same four Intel Woodcrest servers that drive the CATS sustained 155 GFLOPS running HPL on their own or 81% of the 192 GLOPS theoretical peak. Since a Harpertown processor has twice as many cores, two Harpertown servers could have driven the same number of ClearSpeed boards as four Woodcrest servers. However a Harpertown processor can be as efficient as Woodcrest if not slightly better. Our single Intel Harpertown Xeon E5272 server 3.0 GHz achieved 78 GFLOPS and two would beat the Woodcrest number.[9] Thus either way the host performance is not too far from the two CATS. Clearly the HPL benchmark is not the one where ClearSpeed shines primarily because the HPL code was not changed. If the bandwidth was not a problem and/or the code could be modified so that all the cards are used then the performance of ClearSpeed would be much better and possibly reach or even exceed 6x43 GFLOPS on a half CATS.

[9] Intel's Harpertown Xeon E5262 2.8 GHz demonstrated 153 GFLOPS on two nodes (16 processes) and 306 GFLOPS on four (source: HPCC web site http://icl.cs.utk.edu/hpcc/).

MOLPRO

MOLPRO is a program for molecular electronic structure calculations.[11] It is distinguished from other commonly used packages in its emphasis on high accuracy, the feature which makes it very popular in academic research. Frederick Manby, one of the leading MOLPRO developers, and his group realised that by devising a suitable algorithm they can tap into the huge computational power provided by ClearSpeed technology.[12] They started with the development of a massively parallel method for density-functional theory in the Kohn-Sham framework and implemented it using CATS600. Subsequently the new code was merged into the latest release of MOLPRO 2008.1 which was used in the present evaluation. The results presented below were obtained using the code downloaded from the main repository. In the earlier tests we were clearly limited by the need to have one CPU process per card. Porting an application natively to ClearSpeed eliminates this problem. The new code added new functionality to MOLPRO and is serial at the host processor level. However it can use multiple ClearSpeed cards which can be controlled by a parameter in the input file.

Building MOLPRO is easy and this time was no exception. There were two test cases provided which get a performance boost when running on ClearSpeed – chorismate ($C_{10}H_8O_6$) and neuraminidase ligand, DPC ($C_{20}H_{27}O_5N_3$). Both are quite large molecules but the latter test is substantially more expensive. In computing the electronic energies, we used 6-31G* basis as in the original paper but our host processor was Intel Xeon instead of AMD Opteron.

	Chorismate	DPC
	T/sec	T/sec
CS CSX600	316	1405
CS e620 x1 (CSX600 x2)	161	708
CS e620 x3 (CSX600 x6)	57	260
CS e620 x6 (half CATS)	22	152
Xeon 5160 core x1	155	639
Xeon E5472 core x1	143	576

Table 4: MOLPRO performance (elapsed time in seconds) on ClearSpeed e620 and Intel servers.

Table 4 shows the total time elapsed running the two test cases. The number of ClearSpeed processor or cards is indicated in the first column; the number of Intel cores is always one. Remember that there are two ClearSpeed chips on the card. The table demonstrates very good ClearSpeed performance and excellent scaling. Furthermore the scaling for the Chorismate test is superlinear. The table also shows that the performance of an Intel core is slightly better than that of the ClearSpeed card (two CSX600 chips). It seems natural to assume that if the MOLPRO code were threaded the way the ClearSpeed code was parallelised then the pure x86 code would also scale quite well given the same underlying algorithm. So we might expect (assuming linear scaling) that the Woodcrest server would

[11] MOLPRO, version 2008.1, a package of *ab initio* programs, H.-J. Werner, P. J. Knowles, R. Lindh, F. R. Manby, M. Schutz, and others, see http://www.molpro.net.
[12] P. Brown, C. Woods, S. McIntosh-Smith and F. R. Manby, "A massively multicore parallelization of Kohn-Sham theory" J. Chem. Theory Comput., **4**, 1620 (2008).

achieve 39 and 160 seconds respectively for Chorismate and DPC and the Harpertown server 18 and 72 seconds. Thus a Harpertown server should perform reasonably close to a half CATS server.

European option pricing

Where ClearSpeed really shines is in financial applications. A number of financial kernels had been ported to ClearSpeed in order to demonstrate the full efficiency and benefits of the platform. These include the following

- Black-Scholes analytic pricing formula,
- European option pricing via binomial method,
- European option pricing via Monte Carlo,
- Asian option pricing via Monte Carlo,
- Pricing American options via explicit finite differences, and
- Broadle-Glasseman random tree method.

We selected the European options calculation using Monte Carlo because financial Monte Carlo calculations (and Monte Carlo calculations in general) represent a very popular class of calculations which can be frequently found in benchmark reports. Furthermore the price of a European option can be evaluated analytically. On the one hand there is a Black-Scholes formula for European options; on the other it is possible to compute the same value using a Monte Carlo simulation.[13]

As part of a large portfolio, each option calculation requires generation of millions of random numbers and subsequent operations with them. In our particular case, the European call option calculation proceeds as follows:

> *Generate a normally distributed random number y,*
> *Compute option value =* $MAX\left(\{S \exp((r - 0.5\,\sigma^2)\,T + \sigma\,\mathrm{sqrt}(T)\,y) - E\},\ 0\right)$

where S is the underlying value, E is the exercise price, r is the interest rate, σ is stock volatility and T is the time period. Then the average over all realisations recomputed back to the starting time is the option value

> *Option value = exp(-r T) Average.*

Since pseudo-random generators usually produce uniformly distributed numbers a conversion step to the normal distribution is required. The Box-Muller method is a fairly popular one:

> *take two real numbers* $0 < (x_1, x_2) < 1$,
> $y_1 = \mathrm{sqrt}(-2\log(x_1))\cos(2\pi x_2)$
> $y_2 = \mathrm{sqrt}(-2\log(x_1))\sin(2\pi x_2)$.

Thus each simulation step involves quite a few computationally expensive calls to mathematical functions. The selection of a random number generator is very important both in terms of quality of random numbers and speed. The Mersenne Twister algorithm provides

[13] For example, see P. Wilmott "Paul Wilmott Introduces Quantitative Finance" 2nd Edition, John Wiley & Sons, Ltd (2007).

a good compromise. It can have a period of up to $2^{19937}-1$ and it is reasonably quick.[14] Parallelisation can be achieved through dynamic creation of generators.[15] In order to boost performance even further, ClearSpeed employed Vector Math Library (VML) which allows aggregation of four operations into a single instruction issue. This has been achieved by using `__DVECTOR` data types instead of `poly double`. Furthermore the user does not need to worry about porting the Mersenne Twister algorithm because it is readily available in the random number library along with a few other random number generators.

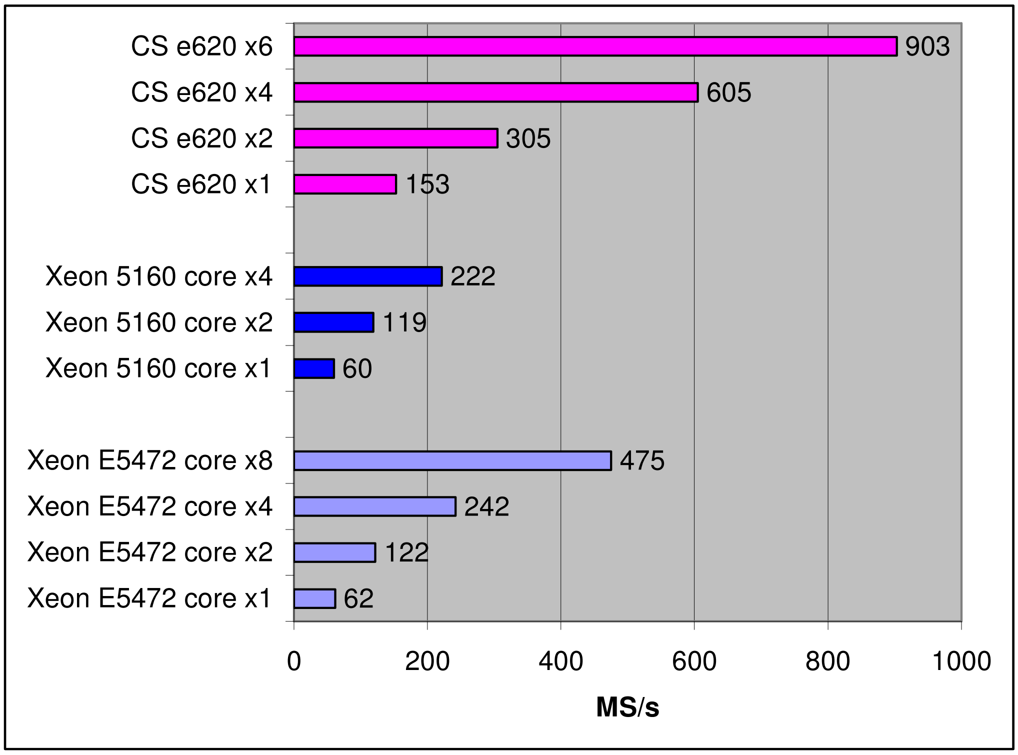

Figure 2: Performance of European option pricing on ClearSpeed e620 and Intel servers.

The performance of option calculations can be conveniently represented in millions of simulations per second or MS/s. The ClearSpeed e620 card can do impressive 153 MS/s which scale linearly up to 903 MS/s on six cards. The Monte Carlo calculations are inherently ‘embarrassingly parallel’ and therefore 1800 MS/s on the whole CATS should be achievable. ClearSpeed also provided a reference C implementation of the same calculation but that could perform only 8 MS/s on a single core of the host. Since that was clearly not adequate we wrote our own benchmark using the original Mersenne Twister code provided by Makoto Matsumoto on his web site. Although the new code yielded 22 MS/s which was a substantial improvement the author claimed that their SIMD-oriented Fast Mersenne Twister doubles the performance. However we opted instead for the Mersenne Twister generator

[14] M. Matsumoto and T. Nishimura, "Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator", ACM Trans. on Modeling and Computer Simulation Vol. 8, No. 1, January pp.3-30 (1998). http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html
[15] Makoto Matsumoto and Takuji Nishimura, "Dynamic Creation of Pseudorandom Number Generators", Monte Carlo and Quasi-Monte Carlo Methods 1998, Springer, 2000, pp 56--69. http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/DC/dgene.pdf

provided by Intel MKL. Not only does it contain Vector Statistical Library (VSL) and Vector Math Library (VML) but also an efficient method of parallelisation using the concept of streams. The result was a C code parallelised using OpenMP, VSL and VML. The single thread performance improved to 60 MS/s and the total server performance achieved 222 MS/s on Woodcrest and 475 MS/s on Harpertown. The results presented graphically in Figure 2 indicate that CATS 600 is approximately four times quicker than our Harpertown server.

Performance, power and cost

So far we have evaded confronting the question of what is actually a fair comparison of a ClearSpeed solution and a commodity server. At first it seems normal to compare one card and one processor or core but in reality we deal with servers, not cards. Thus it is more natural to compare one Intel server with the CATS server or half of the CATS. Table 5 below reports a short summary of our benchmarking results extrapolated where necessary to the full Intel Harpertown server and half of the CATS. The last line in Table 5 presents the performance ratio of half of the CATS and the Harpertown server. We observe that the ratio drops significantly from the theoretical peak value of 5 to around 1–2 on real codes but a factor of four should be achievable on DGEMM kind of load.

Power is becoming more and more frequently mentioned in the context of performance analysis. If the downside of a spectacular performance is disproportionate power consumption then the utility of such a device may be questionable. Perhaps one of the most important features of the ClearSpeed architecture is its low power utilisation and high performance per Watt value. Therefore we felt that it is impossible not to address the question of power. In order to give a quantitative answer we used a consumer grade power meter which, although straightforward to use, was not able to provide very accurate power measurements. However we estimated that the accuracy of our power measurements should not be worse than 10%.[16] The idle and heavy load power measurements are presented in the second and third columns of Table 5. The CATS 600 under heavy load consumes about 590W which was extrapolated from running DGEMM on eight cards. The idle power is slightly lower but not by much. In contrast to that not only the heavy load power of the Intel servers is lower but also the idle power is nearly half of the heavy load (HPL) numbers. The difference in power between the Woodcrest and the Harpertown is likely to be due to the doubled size of memory (16GB vs 8 GB). The full amount of power consumed by a ClearSpeed setup will depend on whether the CATS is attached to one or two servers. Thus its power budget is expected to be around 1 kW or more. We should note however that the new CATS 700 consumes less power and is probably on the same level as the Intel servers.

[16] In our measurements, the voltage and current readings were stable but the power factor was at times erratic. Therefore the measurements below assume the power factor of one although in reality it might have been slightly less than that. However it seems reasonable to believe that the modern power supply units used by ClearSpeed, Dell and HP are of good quality and their power factors are not lower than 0.9; hence the accuracy estimate of 10%.

	Power (idle) /Watts	Power (load) /Watts	Peak perf /GFLOPS	DGEMM perf /GFLOPS	Linpack perf /GFLOPS	European options / (MS/s)	Molpro (Chorismate) perf relative to 1x Xeon 5160 core /%
CS e620			81	60	43	153	161
CS CATS600	470	590	968				
half CATS	235	295	486	360	89	900	705
Xeon 5160 core			12	11		23	100
Xeon 5160 server	270	420	48	43		222	400
Xeon E5472 server	240	460	96	85	78	475	867
half CATS/E5472 srv			5.1	4.2	1.1	1.9	0.8

Table 5: Power estimates for the CATS and the Intel servers together with the summary of performance figures. The black numbers are based on measurements, the red ones are estimates.

Another important factor which was not mentioned so far is the cost of both equipment and power. As the price of the latter continues to rise it becomes an increasingly important factor. Table 6 presents an estimate of the running cost of a Harpertown server assuming the three year exploitation period, extra power consumption due to air conditioning at 50% and the electricity cost at the current market price 12 pence per kWh. Note that we do not make any assumptions towards further price increase. Already we can readily see that the running cost of a server is close to the cost of the server itself. In order to evaluate the potential cost saving of using the CATS let us assume that a single CATS performance is equal to four servers which is the best case scenario according to Table 5. Thus we have one server (the host) and CATS on the one hand and four or five servers on the other depending on whether we are using the host for computation or not. The cost of four servers is approximately £18500 (three year energy plus four servers at £2500) which gives us the maximum cost of the CATS for it to have an economic sense. This limit would have been higher if we were able to identify an application which has a performance advantage similar to DGEMM. Then the saving is the difference between £18500 and the actual cost of CATS 600.

Number of Harpertown servers	Power incl aircon @ 50% /Watts	Energy /year /kWh	Energy cost @ 12p per kWh /year /£	3 year energy cost /£
1	675	5913	710	2129

Table 6: Power and running cost estimates for HP DL160 G5 server based on Intel Harpertown processors. The estimate assumes: air conditioning at 50%, electricity cost at the current market price 12 pence per kWh, exploitation period of three years.

Conclusions

We have evaluated ClearSpeed accelerators by running a number of different benchmarks, computational kernels and real codes. The ClearSpeed architecture certainly has a lot of computational power as demonstrated above. Furthermore we saw that the Clearspeed Accelerated Tera-scale System (CATS) is indeed able to provide very high computational density thanks to having twelve accelerator cards packed into it. However, when it was compared head-to-head with the modern Intel-based commodity servers it appears that, although slower, the Intel servers provide close if not better price/performance value. The performance and low power advantage is further diminished by the fact that the CATS requires one or two host servers. Therefore for data centres not constrained by space and power, the total cost of ownership must be considered and it is currently not in favour of ClearSpeed. Further technological developments and continuing rising cost of electricity may change that.

Recently ClearSpeed introduced the new Advance e710 card and respectively CATS 700 server based on it. ClearSpeed informed us that the new card is at least 20% faster than the card used in the present evaluation. Furthermore it has been claimed that the new Advance e710 card achieves more than twice the price/performance of its predecessor, the e610 card. Therefore the estimates presented here are somewhat outdated. However it should be clear that the most important factor in the ClearSpeed value proposition is the price of CATS 700 itself.

In this paper, we tried to explore many aspects of the ClearSpeed architecture such as power, programmability and cost, although performance was clearly our primary objective. Different application areas have different priorities and perhaps there are some which will find ClearSpeed advantageous. However for HPC at large the cost is usually the number one provided that the programming effort is reasonable. Therefore we think that ClearSpeed use in HPC will be limited unless its value becomes more attractive. Furthermore, six-core commodity processors are already available and the server roadmaps promise six- and eight-core processors as well as further extensions to SSE. All these advances are going to reduce the performance gap between commodity processors and ClearSpeed. GPGPUs are another threat to ClearSpeed. On the one hand they are much more power hungry but on the other they cost much less as well. It is still an early day for widespread use of GPGPUs in HPC but either way it seems very likely that the Top500 list will continue to be dominated by the commodity hardware at least in the immediate future.

Acknowledgements

We are very grateful to Tim Lanfear and Simon McIntosh-Smith (ClearSpeed) for very useful discussions and helping with the CATS, Frederick Manby and Philip Brown (University of Bristol) for helping with the MOLPRO port to ClearSpeed, Peter Knowles (University of Cardiff) for providing a MOLPRO licence and Jonathan Follows (STFC) for a number of comments on the manuscript.