Efficiency, energy efficiency and programming of accelerated HPC servers: Highlights of PRACE studies

Lennart Johnsson

Department of Computer Science
University of Houston
and
School of Computer Science and Communications
KTH

To appear in Springer Verlag "GPU Solutions to Multi-scale Problems in Science and Engineering", 2011

Abstract

During the last few years the convergence in architecture for High-Performance Computing systems that took place for over a decade has been replaced by a divergence. The divergence is driven by the quest for performance, cost-performance and in the last few years also energy consumption that during the life-time of a system have come to exceed the HPC system cost in many cases. Mass market, specialized processors, such as the Cell Broadband Engine (CBE) and Graphics Processors, have received particular attention, the latter especially after hardware support for double-precision floating-point arithmetic was introduced about three years ago. The recent support of Error Correcting Code (ECC) for memory and significantly enhanced performance for double-precision arithmetic in the current generation of Graphic Processing Units (GPUs) have further solidified the interest in GPUs for HPC.

In order to assess the issues involved in potentially deploying clusters with nodes consisting of commodity microprocessors with some type of specialized processor for enhanced performance or enhanced energy efficiency or both for science and engineering workloads, PRACE, the Partnership for Advanced Computing in Europe, undertook a study that included three types of accelerators, the CBE, GPUs and ClearSpeed, and tools for their programming. The study focused on assessing performance, efficiency, power efficiency for double-precision arithmetic and programmer productivity. Four kernels, matrix multiplication, sparse matrix-vector multiplication, FFT, random number generation were used for the assessment together with High-Performance Linpack (HPL) and a few application codes. We report here on the results from the kernels and HPL for GPU and ClearSpeed accelerated systems. The GPU performed surprisingly significantly better than the CPU on the sparse matrix-vector multiplication on which the ClearSpeed performed surprisingly poorly. For matrix-multiplication, HPL and FFT the ClearSpeed accelerator was by far the most energy efficient device.

1. Introduction.

1.1 Architecture and performance evolution

High-Performance Computing (HPC) has traditionally driven high innovation in both computer architecture and algorithms. Like many other areas of computing it has also challenged established approaches to software design and development. Many innovations have been responses to opportunities offered by the exponential improvements of capabilities of silicon based technologies, as predicted by "Moore's Law"[1], and constraints imposed by the technology as well as packaging constraints. Taking full advantage of computer system capabilities require architecture aware algorithm and software design, and, of course, problems for which algorithms can be found that can take advantage of the architecture at hand. Conversely, architectures have historically been targeted for certain workloads. In the early days of electronic computers, even at the time transistor technologies replaced vacuum tubes in computer systems, scientific and engineering applications were predominantly based on highly structured decomposition of physical domains and algorithms based on local approximations of continuous operators. Global solutions were achieved through a mixture of local or global steps depending on algorithm selected (e.g., explicit vs. implicit methods, integral vs. differential methods). In most cases methods allowed computations to be organized into similar operations on large parts of the domains and data accessed in a highly regular fashion. This fact was exploited by vector architectures, such as the very successful Cray-1 [2], and highly parallel designs such as the Illiac IV (1976) [3,4,5,6], the Goodyear MPP (Massively Parallel Processor) (1983) [7] with 16,896 processors, the Connection Machine [8,9,10] CM-1 (1986) with 65,536 processors and the CM-2 (1987) with 2048 floating-point accelerators, These machines all were of the SIMD (Single Instruction Multiple Data) [11], data-parallel, or vector type, thus amortizing instruction fetch and decode over several, preferably large number of operands. The memory systems were designed for high bandwidth, which in the case of the Cray-1 [2] and the Control Data Corp. 6600 [12,13] was achieved by optimizing it for access of streams of data (long vectors), and in the case of MPPs through very wide memory systems. The parallel ma-

chines with large numbers of processors had very simple processors, indeed only 1-bit processors. (It is interesting to note that the data parallel programming model is the basis for Intel's recently developed Ct technology [14,15] and was also the basis for RapidMind [16] acquired by Intel in 2009.)

The emergence of the microprocessor with a complete CPU on a single chip [17,18,19,20] targeted for a broad market and produced in very high volumes offered large cost advantages over high-performance computers designed for the scientific and engineering market and led to a convergence in architectures also for scientific computation. According to the first Top500 [21] list from June 1993, 369 out of 500 systems (73.8%) were either "Vector" or "SIMD", while by November 2010 only one Vector system appears on the list, and no SIMD system. Since vector and SIMD architectures were specifically targeting scientific and engineering applications whereas microprocessors were, and still are, designed for a broad market, it is interesting to understand the efficiencies, measured as fraction of peak performance, achieved for scientific and engineering applications on the two types of platforms. The most readily available data on efficiencies, but not necessarily the most relevant, is the performance measures reported on the Top500 lists based on High-Performance Linpack (HPL) [22] that solves dense linear systems of equations by Gaussian elimination. The computations are highly structured and good algorithms exhibit a high degree of locality of reference. For this benchmark, the average floating-point rate as a fraction of peak for all vector systems was 82% in 1993, Table 1, with the single vector system on the 2010 list having an efficiency of over 93%, Table 2. The average HPL efficiency in 1993 for "Scalar" systems was 47.5%, but improved significantly to 67.5% in 2010. The microprocessors, being targeted for a broad market with applications that do not exhibit much potential for "vectorization", focused on cache based architectures enabling applications with high locality in space and time to achieve good efficiency, despite weak memory systems compared to the traditional vector architectures. Thus, it is not all that surprising that microprocessor based systems compare relatively well in case of the HPL benchmark. The enhanced efficiency over time for microprocessor based systems is in part due to increased on-chip memory in the form of three levels of cache in current microprocessors, and many added features to improve performance, such as, e.g., pipelining, pre-fetching and out-of-order execution that add complexity and power consumption of the CPU, and improved processor interconnection technologies. Compiler technology has also evolved to make more efficient use of cache based architectures for many application codes.

Processor Architecture	Count	Share %	R_{max} Sum (GF)	R_{peak} Sum (GF)	Proces-sor Sum
Vector	334	66.80 %	650	792	1,242
Scalar	131	26.20 %	408	859	15,606
SIMD	35	7.00 %	64	135	54,272
Totals	500	100%	1,122.84	1,786.21	71,120

Table 1. June 1993 Top 500 list by process architecture [21]

Processor Architecture	Count	Share %	R_{max} Sum (GF)	R_{peak} Sum (GF)	Processor Sum
Vector	1	0.20 %	122,400	131,072	1,280
Scalar	497	99.40 %	43,477,293	64,375,959	6,459,463
N/A	2	0.40 %	73,400	148280	11,584
Totals	500	100%	43,673,092.54	64,655,310.70	6,472,327

Table 2. November 2010 Top500 list by processor architecture [21]

The scientific and engineering market also had a need for good visualization of simulated complex physical phenomena, or visualization of large complex data sets as occurring for instance in petroleum exploration. Specialized processor designs, like the LDS-1 [23] from Evans & Sutherland [24, 25] that initially targeted training simulators, evolved to also cover the emerging digital cinema market as well as engineering and scientific applications. As in the case of standard processors, semiconductor technology evolved to a point where much of the performance critical processing could be integrated on a single chip, such as the Geometry Engine [26,27] by Jim Clark who founded Silicon Graphics Inc [28] that came to dominate the graphics market until complete Graphics Processing Units (GPUs) could be integrated onto a single chip (1999) [29,30] at which time the cost had become sufficiently low that the evolution became largely driven by graphics for gaming with 432 million such units shipped in 2010 [31] (compared to about 350 million PCs [32] and 9 million servers [33] according to the Gartner group). Thus, since in the server market two socket servers are most common, but four and even

8-socket servers are available as well, the volumes of discrete GPUs (as opposed to GPUs integrated with CPUs, e.g. for the mobile market) and CPUs for PCs and servers are almost identical. Today, GPUs are as much of a mass market product as microprocessors are, and prices are comparable (from about a hundred dollars to about two thousand dollars depending on features).

With their design target having been efficient processing for computer graphics GPUs lend themselves to vector/stream processing. As in the case of the vector machines for scientific and engineering applications GPUs are optimized for applying the same operation to large amounts of (structured) data and have memory systems that support high execution rates. Over time GPUs have enhanced their floating-point arithmetic performance significantly and since 2008 also incorporated hardware support for double-precision floating-point operations and moved towards support of the IEEE floating-point standard. Double-precision floating-point performance and compliance with the IEEE floating-point standard are critical for many scientific and engineering applications. The evolution of GPU floating-point performance since 2002 is shown in Figure 1 [34].

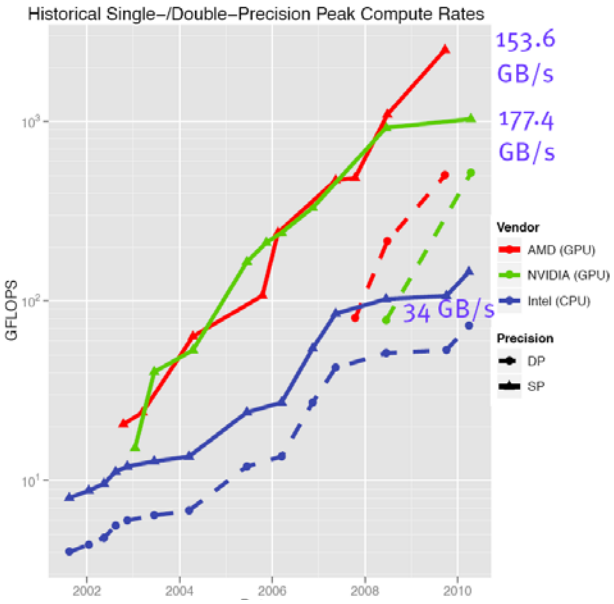

Fig. 1. Performance growth of GPUs and CPUs 2002 – 2010. [34]

As seen in Figure 1, in 2003 the GPU single-precision floating-point performance was only modestly higher than that of common IA-32 [35] microprocessors by, e.g., AMD and Intel, and there was no hardware support for double-precision

floating-point arithmetic, so many application developers in science and engineering did not find the benefits of porting codes to GPUs sufficiently large to warrant the effort to do so. However, as is also apparent from the figure, the performance trajectories for GPUs have been quite different from those of CPUs, so that today a GPU may have 10 – 30 times higher single-precision performance than a CPU, with the AMD/ATI Radeon HD5870 [36,37] having a peak single-precision performance of 2.7 TF (10^{12} flops/s (floating-point operations per second)). Moreover, today GPUs not only support double-precision arithmetic, but the performance advantage compared to a CPU may be a factor of five or more.

Good application performance also requires high memory bandwidth. Today, the memory bandwidth for high-end GPUs is about 150 GB/s [36, 37, 38, 39], which compares very favorably with that of IA-32 microprocessors by AMD and Intel that today has a memory bandwidth of 25 – 30+ GB/s. (The Intel Westmere-EP 6-core CPU has three memory channels each with a peak data rate of 10.8 GB/s (32.4 GB/s total with DDR3 1.333 GHz DIMMs [40], whereas the AMD Magny-Cours 8- and 12-core CPUs have a peak memory data rate of 28.8 GB/s across four channels for DDR3 1.333 GHz DIMMs due to limitations in the North Bridge [41]. Observed Stream [42] benchmark numbers are 20.5 GB/s [43] and 17.9 GB/s [41] for the Intel Westmere-EP CPU and 27.5 GB/s [44], 24.7 GB/s [41] and 19.4 GB/s [45] for the AMD Magny-Cours CPU (on a per CPU basis).

Thus, today GPUs offer about five times the memory bandwidth and about a factor of five higher peak double-precision floating-point performance than IA-32 microprocessors, and the cost is comparable. For instance, nVidia's Tesla C2050 lists for about $2,500, and the ATI FirePro 3D V9800 is priced similarly, compared to a list price of $1,663 for the top-of-the line Intel Westmere-EP CPU (3.46 GHz, 6-cores, 12 MB L3 cache) [40] whereas the top-of-the line AMD Magny-Cours CPU has a list price of $1,514 (2.5 GHz, 12-cores, 12GB L3 cache) [46]. The lowest costs versions of CPUs may cost as little as 20% of the top-of-the line CPUs, comparable to the GPUs targeted for the low end consumer market.

1.2 Energy efficiency

Performance and cost-performance are the traditional measures affecting choice of technology and platforms for high-performance scientific and engineering applications. In recent years energy efficiency in computation has become another important and sometimes deciding factor in the choice of platform. Since a few years

ago the life-time energy cost including cooling of servers has exceeded the cost of the server itself, Figure 2 [47].

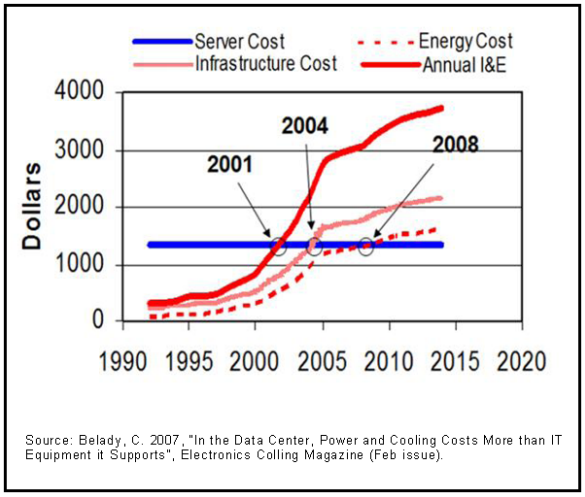

Fig. 2. Evolution of US power and cooling costs for a standard IA-32 server [47]

For microprocessors a large contribution to the performance gain from one generation to the next was increased clock frequency, until about a decade ago. The first microprocessor, the Intel 4004 [17, 18, 19, 20] introduced in 1971 had a clock frequency of 0.74 MHz. By the end of 2002, Intel introduced a Pentium 4 clocked at 3.06 GHz using its Northwood core [48]. The clock frequency was further increased to 3.4 GHz in a version available in early 2004 and further to 3.8 GHz in the Prescott core introduced later that year. (The 3.8 GHz Prescott Pentium 4 is the highest clock frequency ever used in an Intel CPU.) Thus, over a period of about 30 years clock frequencies for Intel microprocessors increased by a factor of about 5,000, followed by a slight decline since its peak in 2004, Figure 3. The evolution is similar for CPUs from AMD, though traditionally AMD CPUs have operated at somewhat lower clock rates, as shown in Figure 4, and lower power consumption.

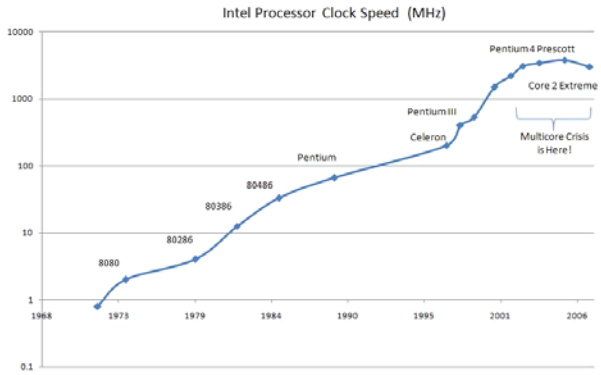

Fig. 3. Intel CPU clock rates 1971 – 2007. [49]

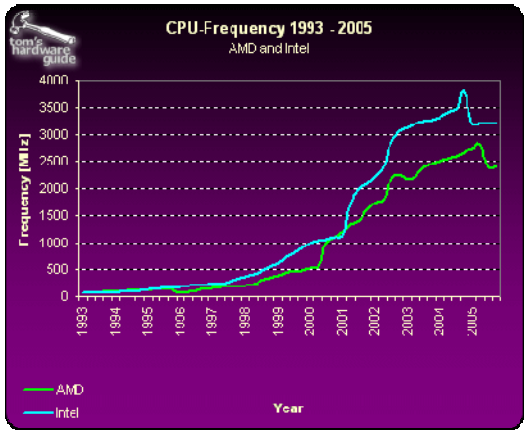

Fig. 4. AMD and Intel CPU clock rates, 1993.- 2005. [50]

The reason for the apparent limit on clock frequency is that, for CMOS technology, the dominating technology for microprocessors, the dynamic switching power P depends on voltage and clock frequency as $P \propto cV^2f$. This relationship is due to the fact that CMOS is a charge transfer technology in which charges on gates of transistors effectively acting as capacitors are drained and restored in switching transistors on or off. The energy stored on a capacitor (gate) is $\propto cV^2$. Furthermore, for CMOS the clock frequency $f \propto V$. Hence, the power dissipation increases very rapidly with the clock frequency. In fact, even though V typically has been reduced form one chip generation to the next, the power density for Intel

CPUs doubled for each generation as shown in Figure 5. The evolution of the power consumption for AMD CPUs [50] has been similar, Figure 6. In 1999 Fred Pollack of Intel stated in his keynote at Micro 32 that "We are on the Wrong side of a Square Law" [51] and concluded with a new goal for CPU design: "Double *Valued Performance* every 18 months, at the same power level", something that the industry has largely adhered to since almost a decade ago.

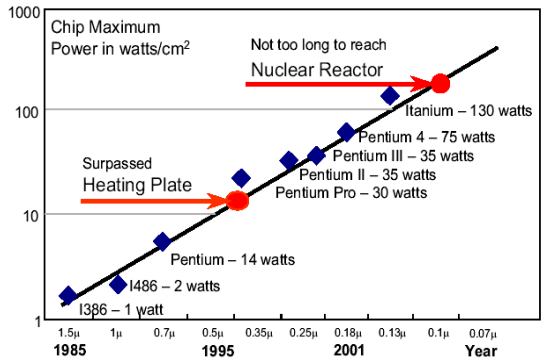

Fig. 5. Heat density of Intel CPUs, Source Shekhar Borkar, Intel.

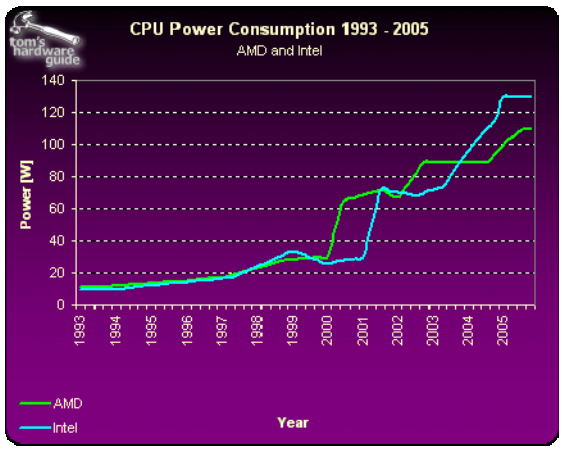

Fig. 6. Comparison of the power consumption of AMD and Intel IA-32 CPUs [50].

The energy per instruction for a range of Intel CPUs [52] is shown in Table 3. The approach taken to achieve "Double valued performance every 18 months, at the same power level" has been to introduce multi-core CPUs exploiting reduced feature sizes in CPU manufacturing, and slightly reducing the maximum clock frequencies. This approach has enabled "double valued performance" to continue for applications that can take advantage of parallelism, but at a cost in application porting and development, and a challenge for compiler developers. High parallelism is becoming main stream, not only by increased core count per chip, but also by increased number of operations a core can perform in a single clock cycle, from one floating-point operation per cycle about a decade ago for IA-32 designs to currently four and in the next generation eight, resulting in a capability to currently carry out 48 floating-point operations per cycle in the case of the AMD 12-core chip.

Product	Normalized Performance	Normalized Power	EPI on 65 nm at 1.33 volts (nJ)
i486	1.0	1.0	10
Pentium	2.0	2.7	14
Pentium Pro	3.6	9	24
Pentium 4 (Willamette)	6.0	23	38
Pentium 4 (Cedarmill)	7.9	38	48
Pentium M (Dothan)	5.4	7	15
Core Duo (Yonah)	7.7	8	11

Table 3. Energy per instruction for Intel CPUs [52].

The power consumption of CMOS processors, as mentioned above, raises steeply with the clock frequency, and of course the number of transistors. The most recent IA-32 CPU by Intel, the 6-core Westmere-EP CPU, (3.46 GHz, 1.17 billion transistors, 240 mm^2 in 32 nm technology) [53] and by AMD, the 8- and 12-core Magny-Cours CPU (2.5 GHz, 2 billion transistors, 692mm^2 in 45 nm technology) [41] both dissipates up to 130 -140W in their highest clock rate versions, while the current generation GPUs from AMD/ATI (0.825 GHz, 2.15 billion transistors, 334 mm^2 in 40nm technology) [36,54] and nVidia (0.575 GHz, 3 billion transistors, 550mm^2 also in 40 nm technology) [38,55] both have a maxi-

mum power rating of 225W. But, since the GPUs have a peak double-precision performance about five times higher than that of the IA-32 CPUs, the GPUs still may deliver higher energy efficiency for applications. We summarize this information in Table 4.

	nm	Trans. (Billions)	Die mm^2	Cores	Memory BW GB/s	I/O BW GB/s	GF DP	W
Nehalem-EP	45	0.731	263	4	3x10.8	2x25.6	53.3	130
Westmere-EP	32	1.17	240	6	3x10.8	2x25.6	83.0	130
AMD Magny-Cour	45	2	692	12	4x10.8*	4x25.6	120.0	137
Tesla C1060	65	1.4	576	240	102.4	8	77.8	188
Tesla C2050	40	3	550	448	144	8	515.2	225
ATI HD5870	40	2.15	334	1600	153.6	8	544	225

Table 4. Some chip characteristics for CPU and GPU processors. (* limited to 28.8 GB/s by the Northbridge)

Estimates of the peak double-precision floating-point rate per W at the chip level is shown in Table 5 [56] for a few processors. The table shows an advantage by a factor of 2.5 to about 4 of GPUs over CPUs. Thus, GPUs in addition to offering potentially higher performance and lower cost-performance in regards to hardware cost, GPUs also have the potential to offer a further cost advantage by being more energy efficient and more environmentally friendly despite their higher power rating.

ARM Coretx-9			ATOM			AMD 12-core			Intel 6-core			ATI 9370		
Cores	W	GF/W	Cores	W	GF/W	Cores	W	GF/W	Cores	W	GF/W	Cores	W	GF/W
4	~2	~0.5	2	2+	~0.5	12	115	~0.9	6	130	~0.6	1600	225	~2.3

nVidia Fermi			TMS320C6678			IBM BQC			ClearSpeed CX700		
Cores	W	GF/W	Cores	W	GF/W	Cores	W	GF/W	Cores	W	GF/W
512	225	~2.3	8	10	~4	16	~50	~4	192	10	~10

Table 5. Estimates of theoretical performance/W for some processor alternatives [56].

Green500 Rank	MFLOPS/W	Site*	Computer*	Total Power (kW)
1	1684.20	IBM Thomas J. Watson Research Center	NNSA/SC Blue Gene/Q Prototype	38.80
2	958.35	GSIC Center, Tokyo Institute of Technology	HP ProLiant SL390s G7 Xeon 6C X5670, Nvidia GPU, Linux/Windows	1243.80
3	933.06	NCSA	Hybrid Cluster Core i3 2.93Ghz Dual Core, NVIDIA C2050, Infiniband	36.00
4	828.67	RIKEN Advanced Institute for Computational Science	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect	57.96
5	773.38	Forschungszentrum Juelich (FZJ)	QPACE SFB TR Cluster, PowerXCell 8i, 3.2 GHz, 3D-Torus	57.54
5	773.38	Universitaet Regensburg	QPACE SFB TR Cluster, PowerXCell 8i, 3.2 GHz, 3D-Torus	57.54
5	773.38	Universitaet Wuppertal	QPACE SFB TR Cluster, PowerXCell 8i, 3.2 GHz, 3D-Torus	57.54
8	740.78	Universitaet Frankfurt	Supermicro Cluster, QC Opteron 2.1 GHz, ATI Radeon GPU, Infiniband	385.00
9	677.12	Georgia Institute of Technology	HP ProLiant SL390s G7 Xeon 6C X5660 2.8Ghz, nVidia Fermi, Infiniband QDR	94.40
10	636.36	National Institute for Environmental Studies	GOSAT Research Computation Facility, nvidia	117.15

Fig. 7. The 10 most energy efficient systems on the November 2010 Top500 list [18]

The potential for higher energy efficiency than that of IA-32 CPUs is indeed real as demonstrated by measurements for HPL. The Green500 list ranks systems on the Top500 list based on their HPL energy efficiency. On the November 2010 list eight of the ten most energy efficient systems use some form of accelerator, with five using GPUs and three using the Cell Broadband Engine (CBE)[57, 58, 59]. Systems using GPUs ranked 2nd, 3rd, 8th, 9th, and 10th. The IBM Blue Gene/Q to be delivered late in 2011 or 2012 ranked 1st with an energy efficiency of 1,684 MF/W. Compared to the Blue Gene/P, its predecessor, the BG/Q has double the execution width of each core, and twice the number of cores per node. Few details are available at this time. The most energy efficient GPU accelerated system achieved an efficiency of 958 MF/W, while the most energy efficient system using the CBE for acceleration achieved 773 MF/W [60]. This system used an experimental interconnection network connecting nodes via the CBE internal high-speed bus. Non-accelerated systems using the latest generation IA-32 CPUs achieved an energy efficiency of about 350 – 400 MF/W for HPL.

1.3 GPU integration and Programming

Programming and code generation for both CPUs and GPUs today requires effective exploitation of parallelism for high efficiency. IA-32 CPUs support common programming languages, such as C, C++, Fortran, etc with a choice of mature compilers that generate efficient code. GPUs on the other hand with a quite different memory architecture and different instruction set have traditionally required specialized and sometimes proprietary languages and compilers. This fact, and the lack of architectural support for many operations commonly used in science and engineering applications have been a limiting factor on their wide-spread adoption. However, the hardware support for general purpose use of GPUs is improving rapidly, thus lowering the barrier towards wide adoption. The good double-precision arithmetic performance and support for IEEE arithmetic are also important factors in today's strong interest in GPUs. However, GPUs are not standalone processors and requires a host, which typically for HPC applications is a common microprocessor. GPUs are "add-on" units typically integrated into the system using the I/O bus of the CPU. This bus can be a performance bottleneck in many cases since data needs to move between the CPU memory and the smaller but faster GPU memory for many applications. As GPUs become integrated onto CPU chips this bottleneck will disappear, but at least initially the GPUs integrated with CPUs on the same chip will not have their own high bandwidth memory system one of the key advantages of today's GPUs. In future generation CPUs the role of GPUs or stream processors may very well change for the scientific and engineering market and stream or vector architectures taking on the primary role, as in the case of Intel's Many Integrated Core (MIC) CPUs [61].

To alleviate some of the programming issues associated with having to produce code for both CPUs and GPUs in a heterogeneous node the Open Computing Language [62], OpenCL, was conceived with version 1.0 published in December 2008 and version 1.1 in September 2010 [63]. OpenCL has been developed by the Khronos Group that also developed OpenGL. Because of the potential benefits of being an Open Standard, OpenCL was included in the assessment despite the fact that only prerelease compilers were available.

1.4 Concurrency comparison between CPUs and GPUs

On-chip parallelism is increasing rapidly for both CPUs and GPUs. The current generation CPUs can carry out up to about 50 double-precision floating-point operations concurrently (48 for the AMD 12 core Magny-Cours CPUs) whereas

GPUs can carry out in the order of 500 – 600 double-precision floating-point operations concurrently (640 for the AMD/ATI HD5870 and FirePro 3D V9800GPUs). Though the concurrency for GPUs is about 10 times higher than for CPUs, the peak performance difference is smaller because the GPUs operate at lower clock frequency (e.g. max 2.5 GHz for the AMD 12-core CPU versus max 0.825 GHz for the AMD/ATI GPU)). As silicon technologies evolve to allow for smaller feature sizes enabling more transistors to be put on the same die, chip designers so far has used the increased capability for additional cores, increased on-chip memory, and less often for execution units of increased width. However, for CPUs the next generation from both AMD and Intel will double the width of the execution units as well as increase the number of cores, thus significantly increasing the peak capabilities, and bringing the parallelism required for peak performance of a IA-32 chip to a level of 100 operations or more. Over about a decade the number of floating-point operations per cycle per core will have increased from one to eight. Hence, though there will be a difference in the degree of parallelism to be expressed and managed, both CPUs and GPUs will have comparable challenges in regards to concurrency. In regards to the viability of GPUs for "general purpose" scientific and engineering computations Shalf et al at LBNL [64] made the interesting observation that only 80 instructions out of the close to 300 instructions on IA-32 platforms were used across a broad range of codes.

2. Highlights of a PRACE study of accelerated IA-32 servers.

2.1 Background

The potential performance, cost/performance and energy efficiency advantages of GPUs are significant, but the programming, and in particular the code porting challenges, are also quite significant. In order to assess the benefits and the code porting challenges PRACE, the Partnership for Advanced Computing in Europe [65], undertook an evaluation of GPU accelerated servers during the second half of 2008 and 2009. The evaluation was made from a data center perspective, i.e., the perspective that codes to be run on a GPU accelerated system could largely only be ported with modest effort using tools targeting heterogeneous node architectures, and not be completely rewritten or hand optimized. Furthermore, the focus was on double-precision arithmetic performance since the intent was to evaluate the merits of GPU accelerated nodes across "all" codes used at partner centers. The tools evaluated were HMPP (Hybrid Multi-core Parallel Programming) [66, 67] from CAPS [68], RapidMind [69, 70] and to a lesser degree the Portland Group Inc's (PGIs) Accelerator Compilers [71, 72] because the PGI products were not available at the time this evaluation started, and OpenCL, as already mentioned. For the GPU test systems the results were compared with nVidia's CUDA [73, 74] whenever possible. In addition to nVidia C1060 accelerated servers, ClearSpeed [75] CSX700 [76] accelerated systems were also assessed, as were systems with CBEs. However, since IBM has decided not to continue with the CBE we do not include results related to it.

The reference platform for the evaluations was a dual socket server equipped with Intel Nehalem 2.53 GHz quad-core CPUs and 3GB DDR3 memory per core. The theoretical peak performance per core of this reference platform thus was 10.12 GF/s. The choice of the Nehalem CPU for the reference platform was motivated by the dominance of Intel EM64T on the November 2009 Top500 [21] list on which this processor family accounted for 79% of the CPUs, see Figure 8, and the

Nehalem CPU being the most recent EM64T CPU from Intel at the time of this evaluation.

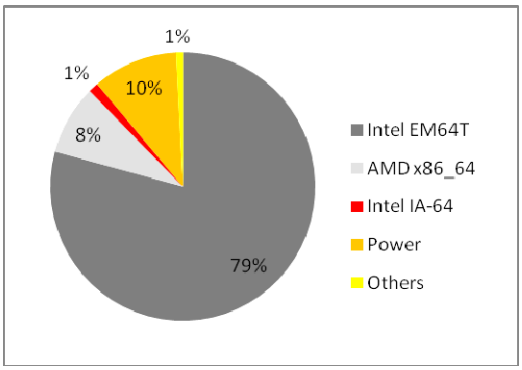

Fig. 8. November 2009 Top500 [21] processor family statistics.

GPU evaluations were made on dual socket, quad-core 2.8 GHz Intel Harpertown servers with two nVidia Tesla servers for each node and two C1060 cards for each Tesla server. The Tesla servers were connected to the hosts over PCI Express Gen2 16x (8GB/s) for each node. The C1060 has 30 stream processors each with eight single-precision (SP) Floating-Point Units (FPUs) and one double-precision (DP) FPU. The peak SP performance is 624 GF and the peak DP performance is 78 GF.

ClearSpeed results were obtained from two platforms; 1) dual socket 2.53 GHz Intel Nehalem servers with 4GB/core with a ClearSpeed-Petapath e710 unit for each server connected via PCI express Gen2 16x [77,78]., 2) dual socket 2.67 GHz Nehalem servers with 3 GB/core and ClearSpeed-Petapath e740 and e780 units, one per CPU socket, connected via PCI express Gen 2 16x [77,78]. The ClearSpeed-Petapath units use 1, 4 or 8 ClearSpeed CSX700 units, each with a peak double-precision arithmetic performance of 96 GF. A ClearSpeed CSX700 is in turn made up of two Multi-Threaded Array Processors (MTAPs) [79], each with a peak performance of 48 GF, double-precision.

The benchmarks used for the evaluations were a few kernels common in scientific and engineering applications: dense matrix multiplication, solution of dense systems of linear equations (HPL), sparse matrix-vector multiplication, FFT and random number generation. This selection was based on a study of application codes used at PRACE partner sites [80]. These kernels also represent a subset of Phil Colella's well known "Seven Dwarf's" [81] described in [82]. The bench-

mark software used for these functions was EuroBen [83], except for the linear system solution for which High-Performance Linpack (HPL) [22] was used. The EuroBen routines used were

- mod2am for dense matrix-matrix multiplication C=AxB
- mod2as for sparse matrix-vector multiplication c=Axb with the matrix in Compressed Sparse Row (CSR) format
- mo2f for 1-D complex-to-complex Fast Fourier Transform using a radix-4 algorithm
- mod2h for random number generation.
-
- All benchmarks were based on C codes.

2.2 Results for the Reference Platform.

For the reference platform we report both single core and eight core results. The memory system supports a single core well, but not fully all four cores on a CPU for memory intensive applications. Furthermore, a node has NUMA (Non-Uniform Memory Access) [84] characteristics in that in a node each CPU with four cores has its own memory not directly accessible by the cores on the other CPU in a two socket system.

2.2.1 Single core results

Matrix multiplication

The single core dense matrix multiplication using mod2am calling Intel's Math Kernel Library (MKL) [85] is shown in Figure 9. The peak achieved performance is 9.387 GF, 92.8% of peak [78].

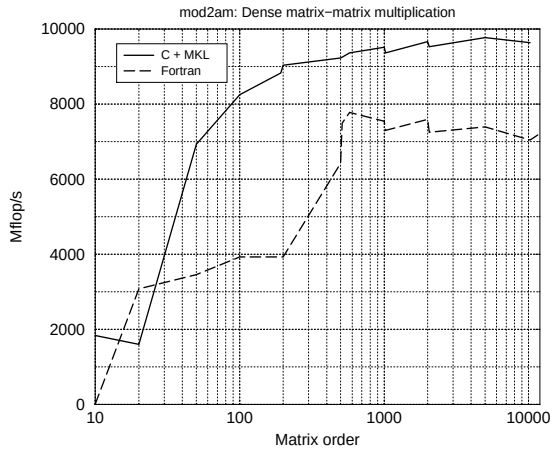

Fig. 9. Mod2am results on a single Nehalem 2.53 GHz core [78].

Sparse matrix-vector multiplication

The single core sparse matrix-vector results [78] are shown on Figure 10. As expected the performance is much lower. Sparse matrix-vector multiplication using compressed formats has a relatively low number of floating-point operations compared to integer operations for address calculations and, for randomly generated sparse matrices, a random memory access pattern that tend to result in poor cache behavior. The peak observed performance is about 13.6% of theoretical peak (10.12.GF). Due to the randomness of the matrix sparsity the performance as a function of matrix size does not follow a smooth progression unlike the case for dense matrix multiplication. The sparse matrix was filled to 15% in all cases.

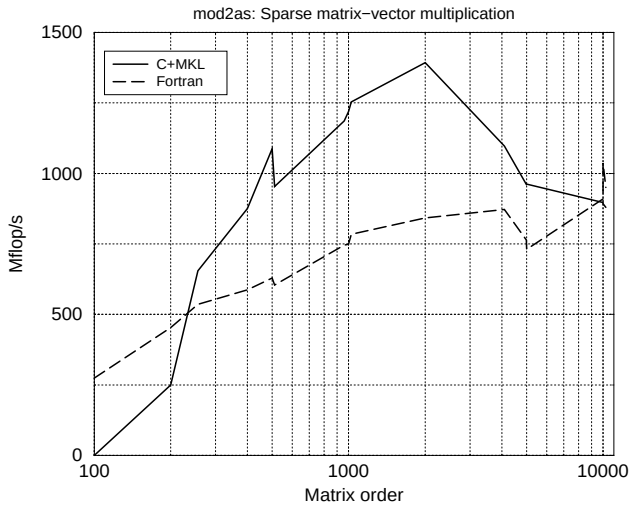

Fig. 10. Mod2as results on a single Nehalem 2.53 GHz core [78].

FFT

The single core FFT results [78] are shown in Figure 11. The peak achieved performance was 2.778 GF, 27.5% of peak. Unlike matrix multiplication and matrix-vector multiplication complex-to-complex FFT computations do not have a balanced number of additions and multiplications. Thus, for this type of FFT the peak core performance of 10.12 GF is never attainable. Complex multiplication requires 4 real multiplications and 2 real additions. A radix-4 computation requires 3 complex multiplications and 4 complex additions/subtractions. In a straightforward organization of the complex operations the complex multiplication results at best in 6 arithmetic operations out of 8 potential hardware arithmetic operations, i.e. 75% utilization, and a complex addition results in 2 out of four potential operations, or 50% utilization. FFTs also have a somewhat complex memory reference patterns using strided access with different strides for different phases of the algorithm. The strided access can result in poor cache behavior. In [86, 87] a performance difference by more than a factor of 10 was observed for different strides for a few different processors.

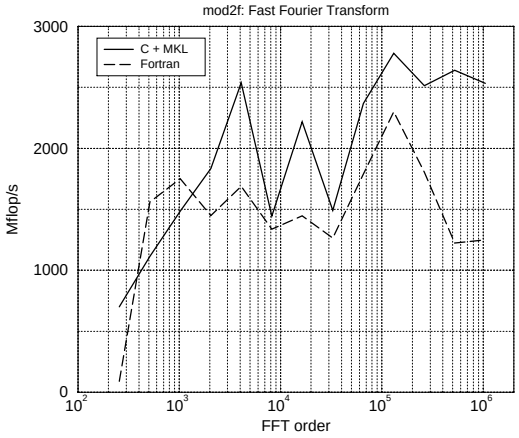

Fig. 11. Mod2f radix-4 complex-to-complex 1-D FFT on a single Nehalem 2.53 GHz core [78].

Random number generation

The single core random number results [78] are shown in Figure 12. Since the random number generator use very few floating-point operations the performance is measured in operations/s. The MKL library does not include a random number generator so results are reported for a C code.

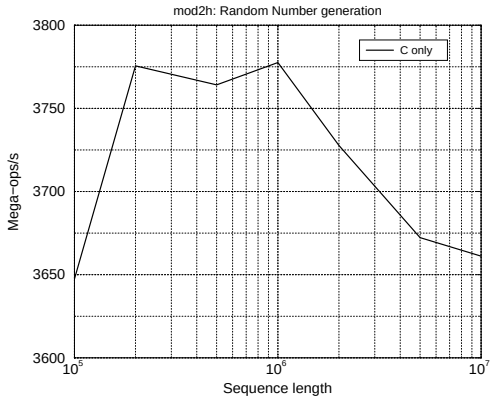

Fig. 12. Mod2h random number generation results on a single Nehalem 2.53 GHz core [78].

2.2.2 Node results

The reference node has two sockets each with a quad-core 2.53 GHz Intel Nehalem CPU. Thus, eight threads can be run concurrently on the reference platform, 16.with hyper-threading [88] with two threads per core. In our tests we did not enable hyoer-threading since it is known to reduce performance in compute intensive cases. Results for 1, 2, 4 and 8 threads are shown in Figures 13 - 16. The MKL version used for the benchmarks supported multi-threading for dense matrix-matrix and sparse matrix-vector multiplication, but not for the FFT. Thus, for the FFT MPI was used to in effect create multiple threads on a reference node. However, at this time MKL does have multi-threaded FFT support [89]. For the random number generator multiple instances were run since neither an MPI nor an OpenMP version did exist, and was not developed

Matrix multiplication

The peak matrix multiplication performance achieved on eight cores using the MKL was 76 GF, which is 93.9% of theoretical peak.

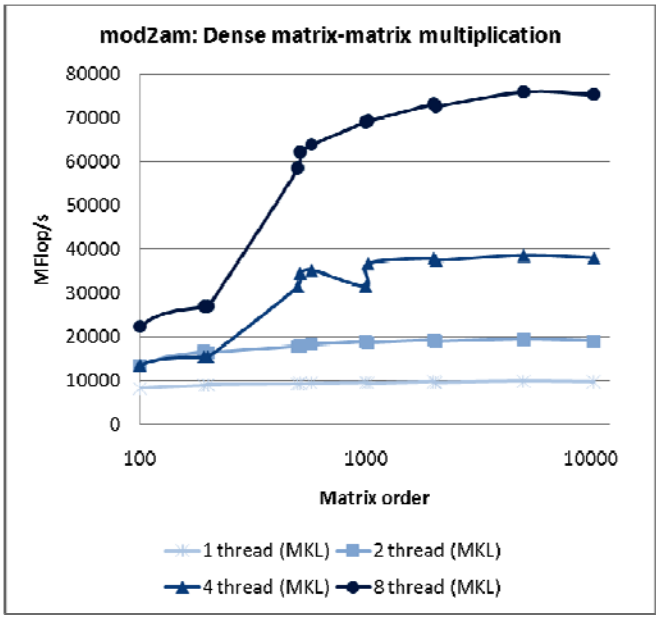

Fig. 13. Mod2am results on a dual socket, 8-core Intel Nehalem 2.53 GHz node with 24 GB memory [78]

Sparse matrix-vector multiplication

For sparse matrix-vector multiplication the performance is highly variable as can be expected due to the randomness of the problem, with a performance peak for four threads of close to 5% of theoretical peak performance. For eight threads the performance is less variable and increases fairly monotonically with matrix size to a peak efficiency of about 3%, Figure 14.

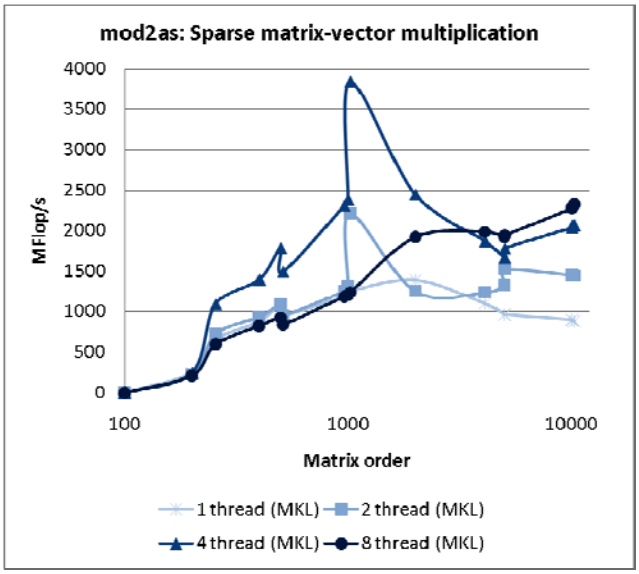

Fig. 14. Mod2as results on a dual socket, 8-core Intel Nehalem 2.53 GHz node with 24 GB memory [78]

FFT

From Figure 15 it is apparent that the single node MPI code for the FFT is performing poorly. Indeed the performance is much worse than the single thread code regardless of the number of MPI processes on a node. Since these benchmarks were carried out Intel has released a multi-threaded MKL FFT code [89] with much improved performance also for a single thread..The results reported for a 2.8

GHz dual socket Nehalem are shown in Figure 16. The single thread performance is about twice what we observed for the MKL version we used, and the multi-threaded version using one thread per core has a peak performance about six times higher than the single thread performance we measured. Using hyper-threading with two threads per core results in a performance boost that for some sizes may exceed 30% and result in an efficiency of up to about 25% for the node, similar to our observed single core performance without hyper-threading..

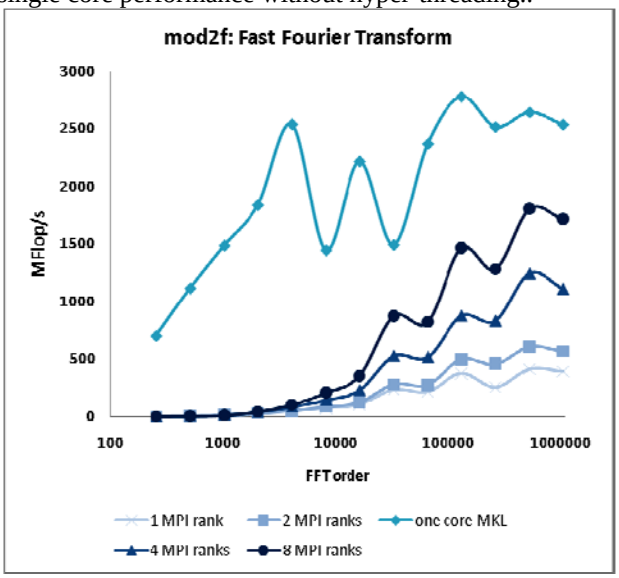

Fig. 15. Mod2f results for on a dual socket, 8-core Intel Nehalem 2.53 GHz node with 24 GB memory [78]

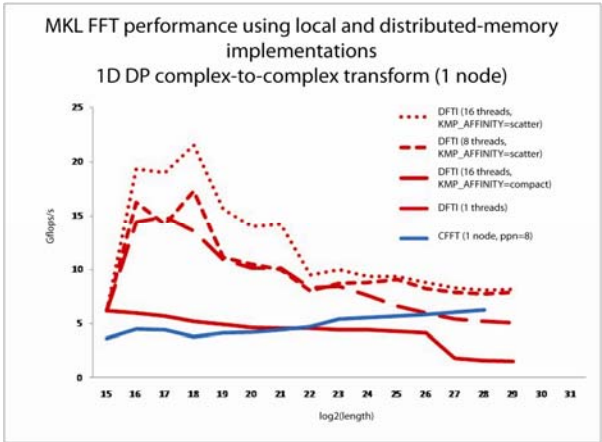

Fig. 16. Performance for Intel's recently released multi-threaded MKL FFT on a 2.8 GHz dual socket Nehalem platform [89].

Random number generation

For the random number generator the aggregate performance increases almost in proportion to the number of instances run, as seen in Figure 17.

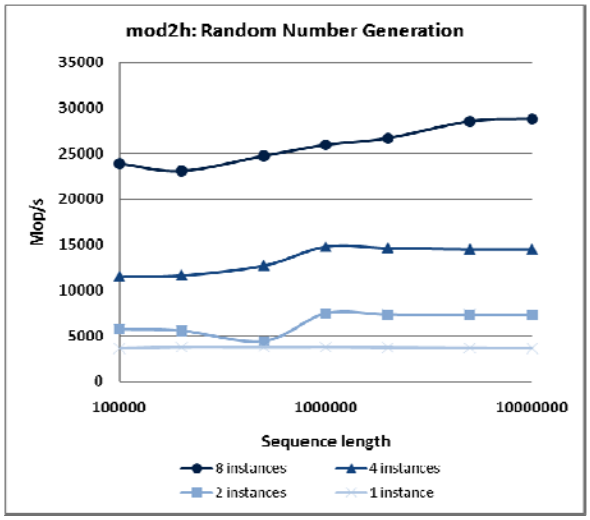

Fig. 17. Mod2h results on a dual socket, 8-core Intel Nehalem 2.53 GHz node with 24 GB memory [78]

HPL

For HPL a best single node efficiency of close to 87% has been reported for the Intel Nehalem, see e.g. [90, 91]. The measurements performed on the reference platform are in line with these results.

2.2.3 Energy efficiency

In regards to energy efficiency matrix multiplication is known to exercise the CPU heavily and hence result in high power consumption. The HPL benchmark that is used for the Green500 [92] list depends heavily on matrix multiplication. For the reference platform we measured a maximum power consumption of 303 W for matrix multiplication [78], resulting in 251MF/W at the achieved 76GF. For HPL a power efficiency of 230 MF/W was observed [78], which is in line with the expected power efficiency given the difference in efficiencies of matrix multiplication and HPL using the MKL. No power measurements were carried out for the sparse matrix-vector multiplication, the FFT and the random number generation. The FFT is fairly floating-point intensive, but not as intensive as matrix multiplication, but relatively more memory reference intensive. On this basis we estimate the maximum power consumption to about 250W for the FFT resulting in an estimated power efficiency of 50 – 80 MF/W for the performance reported in Figure 16.

2.3 nVidia C1060 GPUs

Matrix multiplication

For matrix multiplication on the C1060 nVidia's CUBLAS was used in analogy with using MKL on the reference platform. Since in many applications the data set on which the computations are performed is allocated to the memory of the host processor, subsets of data on which computations are to be performed need to be transferred to the GPU memory and results transferred back. Thus, performance

was measured both for the computations on the GPU itself with data fetched and stored in its local memory and for the situation when data needs to be fetched from the CPU memory and results stored in it. Figure 18 shows the results, with the lower performance curve including the pre and post computation data transfers between CPU memory and the GPU. Since matrix multiplication requires $2N^3$ operations but only $3N^2$ data elements need to be transferred, the data transfer time decreases in significance as N increases. The peak of the on GPU performance with CUBLAS is about 82%, which drops to a peak of about 76% if data transfers are included. These results are in agreement with the results reported in [93].

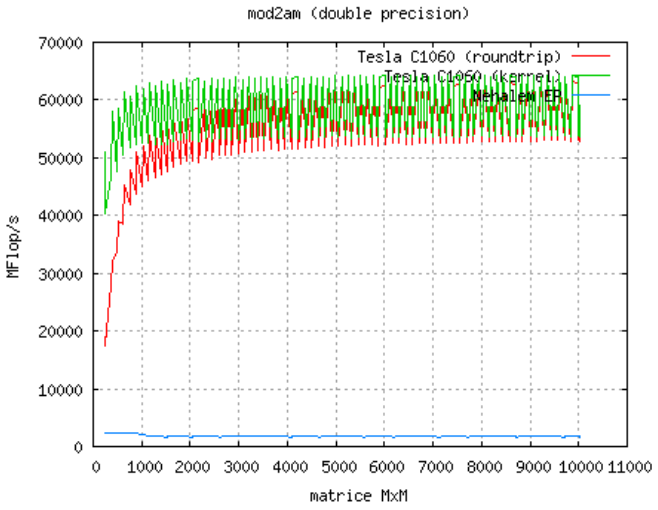

Fig. 18. Mod2am results on nVidia C1060 GPU with 78GF peak performance [77].

Sparse matrix-vector multiplication

For sparse matrix-vector multiplication the results are shown in Figure 19. It is interesting to note that with the data on the GPU the peak observed performance is about 9 GF, or about 11.5% of peak, a higher fraction of peak than on the CPU. This result is in line with the results in [94]. However, if data needs to be fetched from CPU memory and results transferred back, then the data transfer time dominates and the efficiency drops to about 1%. For sparse matrix-vector multiplication both operation count and data transfer is of order O(N).

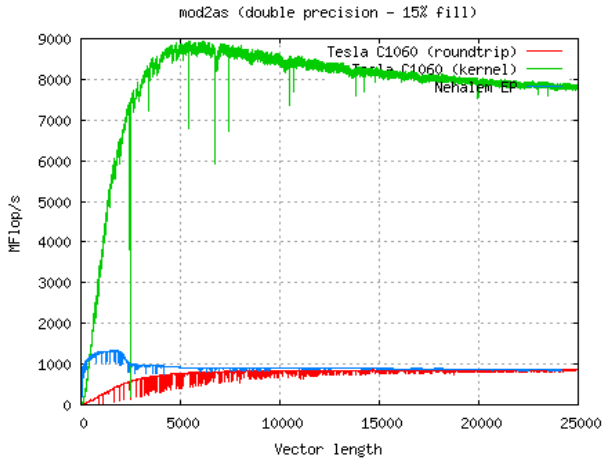

Fig. 19. Mod2as results on nVidia C1060 GPU with 78GF peak performance [77].

FFT

The FFT performance on the C1060 is shown in Figure 20. At the time of the benchmark there was no double-precision CUDA FFT available so a complete port of the mod2f FFT to CUDA was necessary resulting in a CUDA code with about 3,000 lines. The peak performance achieved including data transfers to the CPU memory was about 4 GF, about 5% of peak. At this time the nVidia CUFFT is available and is reported to achieve close to 30 GF on a C1060 [95] excluding data transfer. For FFT the operations count is O(NlogN), and thus the impact of the data transfer expected to be less significant than for sparse matrix-vector multiplication but more significant than for matrix multiplication. The peak efficiency of the single core Nehalem FFT is about 25%. The recently released multi-threaded MKL FFT [89] has an improved single thread performance that is estimated to about 5.4 GF for a single core of the reference platform and about 20 GF for 16 threads on the reference platform, scaling the results in [89] with the ratios of the clock frequencies of the reference platform and the platform in [89] (the MKL hyper-threaded version performs better than the single thread per core version) . Thus, the recent MKL release achieves about 54% efficiency on a single core and a peak of about 25% on the node, while CUFFT achieves a peak efficiency of about 38% on the C1060

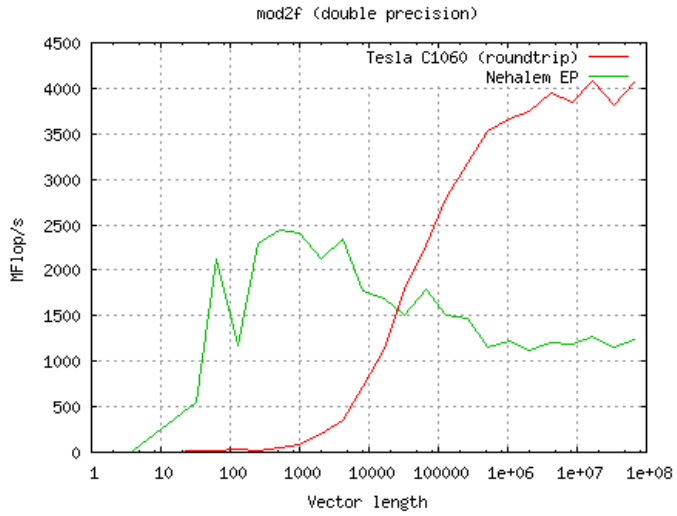

Fig. 20. Mod2f results in hand coded CUDA on nVidia C1060 GPU with 78GF peak performance [77].

HPL

For HPL a peak efficiency for one Nehalem core and one C1060 GPU was measured to be 59.5 GF, 68%, whereas the efficiency dropped to 52.5% using all 8 cores of the host and four C1060 GPUs [78]. The peak power efficiency was 270 MF/W. The single C1060 results are in line with what is reported in [93].

Energy efficiency

GPUs draw significant power with the C1060 having a specified max power of 188W [96] and an estimated typical power consumption of 160W. The Intel Nehalem CPU used for the reference platform has a maximum power dissipation of 80 W [97].

For the reference platform during maximum load for matrix multiplication the CPUs account for about 50% of the power consumption of the reference platform. With the C1060 reaching close to 60 GF for matrix multiplication, Figure 18, and assuming the maximum specified power consumption for this case, the GPU power efficiency is estimated at 300 MF/W. Similarly, for the CPUs alone, the achieved performance using MKL was 76GF and assuming the maximum CPU power consumption the CPU power efficiency is estimated to be 475 MF/W. The fact that the GPU in case of HPL improves the combined energy efficiency is due

to the fact that the power consumption by the memory, fans, power supplies, motherboard etc is already accounted for in the reference platform power efficiency (that is about half of the CPU power efficiency).

2.4 ClearSpeed CSX700

Matrix multiplication

Matrix multiplication carried out on a single Multi-Threaded Array Processor (MTAP) [79] of which there are two on a CSX700 is shown in Figure 21. For the CSX700 the peak observed performance was 85 GF [77], or 88.5% of peak. For the e780 with 8 CSX700 units the peak observed performance was 520 GF [77], 68% of peak.

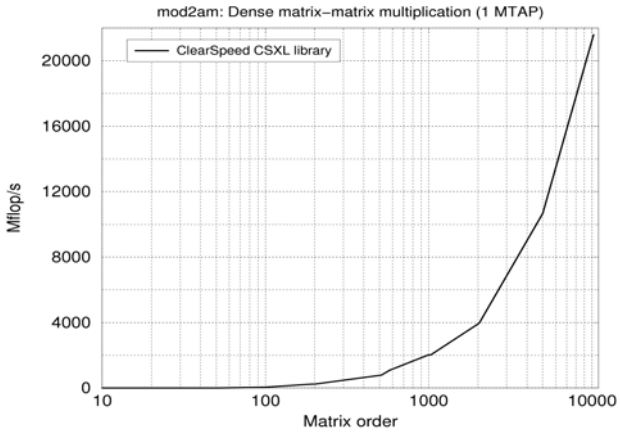

Fig. 21. Mod2am results on one MTAP with a peak performance of 48 GF [78].

As is clear from Figure 21 the ClearSpeed performance is not significant in comparison with the host CPU until the matrix dimensions are in the order of a few thousands. The library [98] that comes with the ClearSpeed hardware recognize this and leaves the multiplication of the matrices to be performed on the host for small matrices. In fact, the software allows for load sharing between the host and the ClearSpeed board. Figure 22 shows the aggregate performance for matrix multiplication as a function of the host assist. The choice of matrix dimensions for

the benchmark was compliant with the CSX700 unit working with tiles that for M and N are multiples of 192 and for K a multiple of 288, for multiplication of an MxK matrix by a KxM matrix. For other matrix shapes the CSX700 library partitions the matrices into tiles compliant with these restrictions and has the host execute the remaining matrix parts for a correct result. For the matrix shapes studied in this benchmark the maximum performance exceeds 130 GF at 42% host assist for the largest M=N. The combined peak performance represents 71% of the combined theoretical peak performance. This is lower than the peak efficiency for the CSX700 card (88.5%) and the host (93.9%), but the matrices chosen for this experiment did not maximize performance for either.

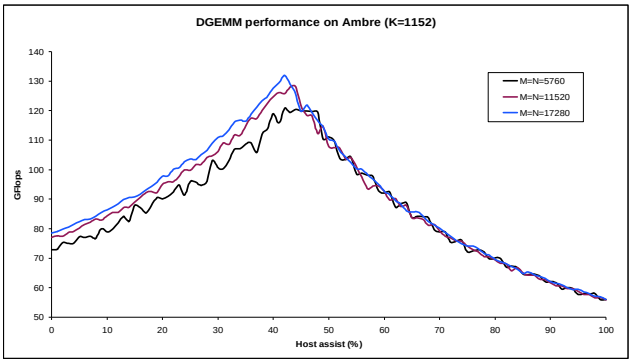

Fig. 22. Mod2am results on the reference platform equipped with a ClearSpeed CSX700 accelerator as a function of the host assist percentage. Peak host performance 80.96 GF, peak CSX700 performance 96 GF [78].

Sparse matrix-vector multiplication

The sparse matrix-vector performance is shown in Figure 23. The performance is exceedingly poor with a peak performance of only close to 30 MF, or less than 0.1% of the peak performance. The MTAP has an architecture that favors streams, like GPUs, but clearly its performance for random memory accesses is very poor.

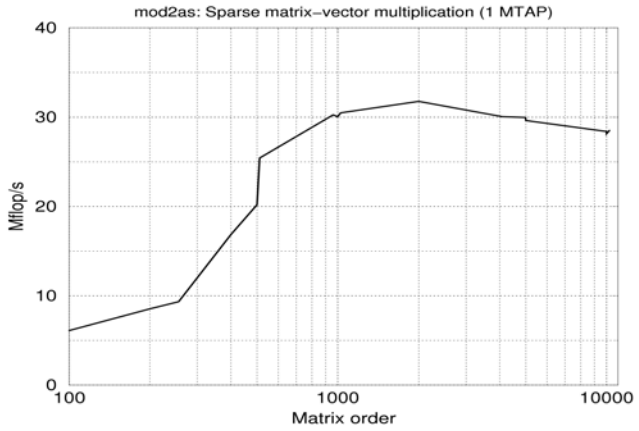

Fig. 23. Mod2as results on one MTAP with a peak performance of 48 GF [78].

FFT

For complex-to-complex 1-D FFTs the results are shown in Table 6. The best observed performance was 9.9 GF, 10.3% of peak. Comparing to the MKL performance reported in [89] the reference node performs better than the CSX700, but a CSX700 delivers a peak performance about twice that of a single core of the reference platform. .

Size	1 MTAP	2MTAP
256	2.8	5.7
512	3.4	6.7
1024	3.8	7.4
2048	4.2	9.4
4096	5.0	9.9
8192	3.7	7.9

Table 6. Mordf results on the CSX700 with peak performance of 96 GF (48 GF per MTAP) [78].

Random number generation

The performance for random number generation is shown in Figure 24 for a single MTAP. The MTAP performance is about 10% lower than the performance of a single core of the reference platform.

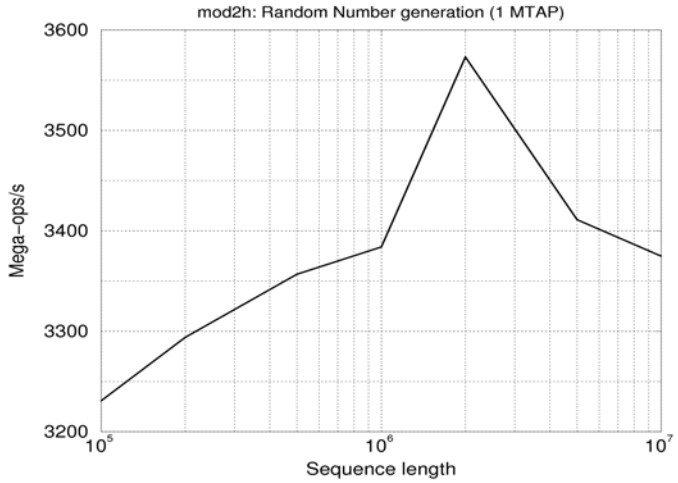

Fig. 24. Mod2h results on 1 MTAP.[78]

HPL

For HPL that depends heavily on matrix multiplication the CSX700 contributed 43.75 GF at 42% host assist, yielding an overall efficiency of 63%.[78]. The results on the manufacturer web site indicates a peak HPL performance of 56.1 GF [99] corresponding to an efficiency of 58.4%.

Energy efficiency

In regards to energy efficiency the CSX700 was observed to consume about 10W in idle state (9.5 – 10.5 W observed) [78] and about 16 W performing matrix multiplication [78]. Thus, with a peak matrix multiplication performance of 85 GF the power efficiency is about 5300 MF/W for the CSX700, while for HPL our results yield in excess of 2700 MF/W for the CSX700 alone at a delivered rate of 43.75 GF and a combined power efficiency of 350 MF/W for the reference platform with one CSX700.

For FFT the peak measured performance was 9.9 GF. The power consumption for the FFT was not measured, but it clearly must be in the 10 – 16 W range [78] resulting in a power efficiency in the 600 – 1000 MF/W range. For the reference platform the idle power was measured to be about 140 W and the peak power 303W [78] resulting in a power efficiency range of 70 – 150 MF/W. Thus, though the absolute performance for the CSX700 is inferior to the MKL multi-threaded reference platform performance, the energy efficiency is a factor 6 - 8 times better.

For random number generation the aggregate performance for the reference platform is about 4 times higher than the CSX700 performance, but the power consumption is estimated to be 10 – 20 times higher and hence the CSX700 considerably more power efficient.

2.5 Performance comparison

Figure 25 summarizes the performance results for matrix multiplication normalized to the reference platform. The C1060 has slightly lower theoretical peak double-precision performance (78GF) and the CSX700 has slightly higher theoretical peak performance (96 GF) than the reference platform (81GF). The combined peak performance of the reference platform and a CSX700 is close to 2.2 times that of the reference platform itself, while adding a C1060 results in a node with 1.96 times the performance of the reference platform.

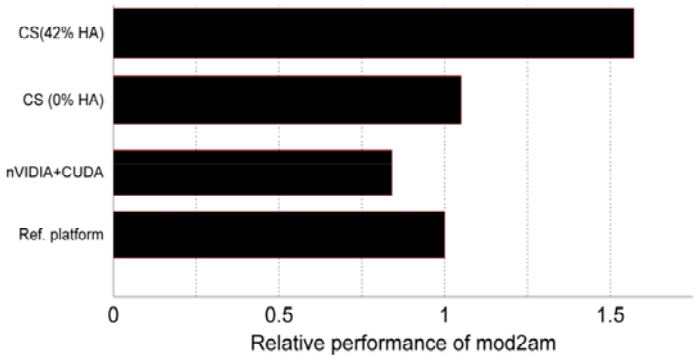

Fig. 25. Mod2am performance on the nVidia C1060 GPU and ClearSpeed CSX700 relative to the reference platform [78].

For sparse matrix-vector multiplication both the C1060 and CSX70 do not offer any performance advantage, Figure 26.

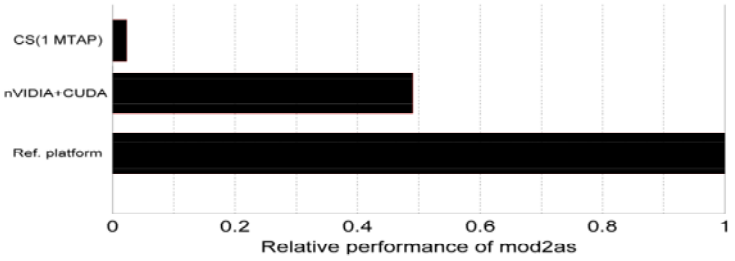

Fig. 26. Mod2as performance on the nVidia C1060 GPU and ClearSpeed CSX700 relative to the reference platform [78].

For the complex-to-complex 1-D radix-4 FFT the relative results we observed are shown in Figure 27. However, since our measurements were made, a new version of the MKL library has been released that improved the reference platform performance with up to more than 7 times thus making the reference platform performance superior to the CSX700. nVidia has also released a CUFFT version that supports double-precision arithmetic and that achieves about 50% better performance than that of MKL on the reference platform.

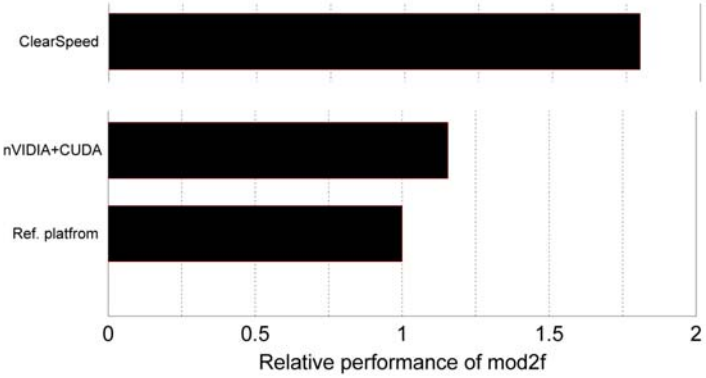

Fig. 27. Mod2f relative performance using MKL version 10.1 on the reference platform alone and with nVidia C1060 GPU or ClearSpeed CSX700 acceleration [78]. MKL release 10.2 having a multi-threaded version of the FFT and improved single core performance has resulted in the reference platform achieving about twice the performance of the CSX700, and a new release of the CUFFT has resulted in the C1060 achieving a peak performance about 50% higher than the reference platform.

For random number generation a single CSX700 MTAP has a performance comparable to a single core of the reference platform. No random number generator was available for the C1060 at the time of the benchmark.

For HPL, a single core of the reference platform in combination with one C1060 GPU was measured to yield 59.5 GF [78] corresponding to 68% efficiency while all eight cores together with four C1060 resulted in a peak node performance of 206 GF out of a possible 393 GF corresponding to 52.5% efficiency.

We summarize our own measurements and some from the literature in Table 7 in order to compare efficiencies of the selected benchmarks on the different architectures, and the energy efficiencies of the devices in isolation and together as an integrated system.

	Host (81GF)		C1060 (78GF)		C1060 incl transf		CSX700 (96GF)		Host+CSX700	
	GF	Eff.%	GF	Eff %	GF	Eff %	GF	Eff %	GF	Eff %
Mod2am	76	93.9	64	82.1	61	78.2	85	88.5	130	73.4
Mod2as	3.8	4.7	9	11.9	1	1.3	0.03	0	-	-
Mod2f	20*[89]	24.7	30[95]	38.5	4	5.1	9.9	10.3	-	-
HPL		87[90]	50[100]	64.1		52.5	56[99]	58.3	75*	42.4*

Table 7. Summary of peak performance and efficiency. (* denotes estimates.)

For the CSX700 the HPL performance is derived from [99]. This estimate compares fairly well with estimating the performance from the CSX600 perform-

ance reported in [101] by scaling the performance with the ratio of the peak performances of the CSX700 and CSX600 units, thus assuming the same efficiency for the units. For the host plus CSX700 HPL performance the number is estimated from the measured performance of an eight node system with four CSX700 per node [78]. The performance of one such node was measured at 206.25GF with 43.75GF contributed by each CSX700. Thus, in this in this configuration the four CSX units in a node contributed 175GF to the node performance and the host 31.25 GF.

In regards to efficiency we notice that for matrix multiplication all three architectures do well, as expected, with the host having a slight advantage. For sparse matrix-vector multiplication none does well, with the CSX700 performing by far the worst. Surprisingly the C1060 perfromed better than the host, but in combination with the host the C1060 is not efficient due to the low computational intensity of sparse matrix-vector multiplication (computations and data transfer are both of order O(N)).

For the FFT the C1060 offers the best efficiency using the optimized CUFFT from nVidia which has about 50% higher efficiency than the optimized MKL for the reference platform (38.5% vs. 24.7%). The CSX700 efficiency is less than half of that of the reference platform and about 25% of the efficiency of the C1060.

The HPL performance as expected is somewhat lower than that of matrix multiplication on which it depends heavily, and the relative merits of the host, the C1060 and the CSX700 are about the same with the CSX700 however ending up with an efficiency about the same as that of the C1060.

2.6 Power efficiency comparison

As previously mentioned the peak performances of the reference platform, the C1060 and the CSX700 are fairly comparable, but the efficiencies achieved on the platforms are quite different and the maximum power consumption is also quite different. We did not have the opportunity to carry out power measurements for all benchmarks. Estimated values are marked with *. The results are summarized in Table 8.

	Host			Host+C1060			Host+CSX700		
	GF	W	GF/W	GF	W	GF/W	GF	W	GF/W
Mod2am	76	303	0.251	130*	490*	0.265*	130	315*	0.410*

Mod2f	20*[89]	250*	0.080*	40*	420*	0.095*	25*	260*	0.096*
HPL	69*	303*	0.230			0.270	75*	315*	0.238*

Table 8. Power efficiency of the configurations evaluated.

Adding a CSX700 to a node increases its maximum power consumption by abut 5%, while the C1060 increases it with more than 60%. For matrix multiplication the CSX700 resulted in a total node performance of 130 GF in our tests and hence the power efficiency increased from about 250 MF/W to about 410 MF/W. The power efficiency for the CSX700 itself is about 5.3 GF/W (85GF, 16W) whereas for the Nehalem itself is about 0.475GF/W (76 GF, 160W).

The power estimates for the FFT assumes about 80% (250W) of maximum power for the reference platform. The C1060 in itself has a power efficiency of about 0.175 GF/W (30GF, 170*W) whereas the Nehalem itself has a power efficiency of about 0.155 GF/W (10*GF, 65*W). The CSX700 itself has significantly higher power efficiency; about 0.700 GF/W (10GF, 14*W)

For HPL the power efficiency improves for a host with accelerator compared to the host itself, as expected from the results for matrix multiplication. The marginal improvement for a host with CSX700 is surprising. Considering the CPU itself it has a power efficiency of about 0.440 GF/W (35GF, 80*W), whereas the C1060 itself is estimated to 0.265 GF/W (50GF, 190*W) and the CSX700 is estimated to 3GF/W (45*GF,15*W).

The power efficiency of the CSX700 is a factor of 4 –10 higher than that of the CPU itself for matrix multiplication, FFT and HPL, but unfortunately for FFT and HPL the relatively low fraction of peak realized cause the total platform power efficiency to increase only marginally for a host combined with a single CSX700. The C1060 power efficiency for matrix multiplication, 0.34GF/W (64GF, 190*W) is less than that of the Nehalem, which is also the case for HPL, but the power efficiency is slightly higher for the FFT.

3. Programming Tools Assessment

3.1 HMPP (Hybrid Multi-core Parallel Programming)

The Hybrid Multi-core Parallel Programming (HMPP) preprocessor by CAPS [66, 67, 102] use directives inserted into the source code to control code generation. The directives have the form of special comments in Fortran and pragmas in C. Using the directives the HMPP preprocessor directs the code generation to be made for the desired device by a compiler for that device. The HMPP preprocessor generates the code necessary to manage the data transfers between the host and accelerators and seeks to optimize it. By using directives an annotated code can be compiled by any compiler for any desired platform and hence the annotated code is as portable as the original code. The HMPP preprocessor has a fallback mechanism should an executable code fail to be generated for a particular target accelerator. Should that be the case code is generated for the host by the compiler used for it. The HMPP directives are designed to target functions (codelets) that can be executed on accelerators and for optimizing the data transfers between the host and accelerators.

The architecture of the preprocessor is shown in Figure 28 in which two back-ends of current interest are shown. The HMPP memory model is illustrated in Figure 29. Our focus was on the CUDA back-end because the OpenCL specification was just released at the time of this study. Our target was the nVidia C1060 GPU as accelerator for IA-32 servers. The test platform had dual socket quad-core 2.8 GHz Intel Harpertown CPUs. Initially the HMPP Workbench 1.5.3 was used, later release 2.1.0sp1 when it became available. For the host the Intel compiler version 11.1 was used and for the C1060 the CUDA 2.3 environment.

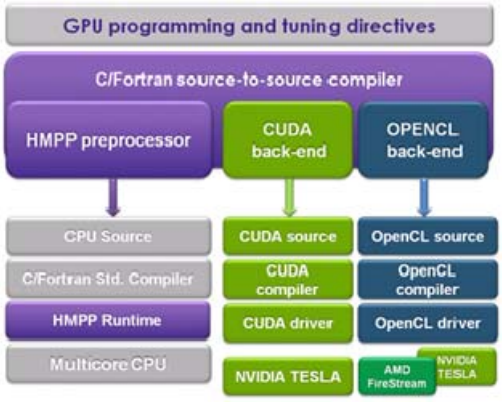

Fig. 28. The architecture of the HMPP preprocessor [66].

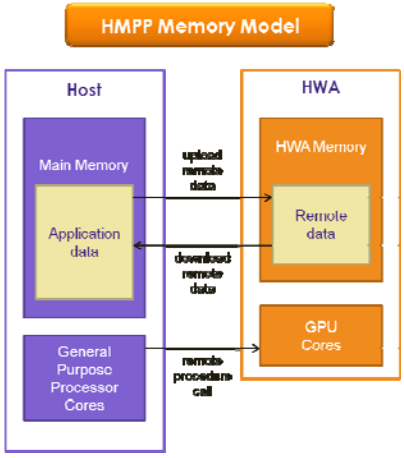

Fig. 29. The HMPP memory model (HWA = HardWare Accelerator) [67]

An example of the use of the HMPP directives is shown in Figure 30.

```
  // simple codelet declara-
tion
  #pragma hmpp Hmxm codelet,
args[a;b].io=in,
args[c].io=out,
args[a].size={m,l},
args[b].size={l,n},
args[c].size={m,n}, TARGET=CUDA
  void mxm(int m, int l, int
n, const double a[m][l], const
double                 b[l][n],
double c[m][n])
  {    int i, j, k;
      for (i = 0; i < m; i++)
{
          for (j = 0; j < n;
j++) {
              c[i][j] = 0.0;}}
      for (i = 0; i < m; i++)
{
          for (k = 0; k < n;
k++) {
              for (j = 0; j <
l; j++) {
                  c[i][k]     =
c[i][k] + a[i][j] * b[j][k];}}}
```

```
  // usage of the codelet
  #pragma hmpp Hmxm advancedload,
args[a;b],       args[a].size={m,l},
args[b].size={l,n}
  for (i = 0; i < nrep; i++) {
  #pragma   hmpp   Hmxm   callsite,
args[a;b].advancedload=true
  #pragma hmpp Hmxm callsite
            mxm(m,  l,  n,  (double
(*)[m]) a, (double (*)[n]) b, (dou-
ble (*)[n]) c);
  }
  #pragma hmpp Hmxm delegatedstore,
args[c]
```

Fig. 30. Illustration of use of HMPP pragma's for definition and use of codelets [78].

The result of using HMPP for matrix multiplication for the C1060 is shown in Figure 31 and for sparse matrix-vector multiplication in Figure 33. These two routines were the only two ported during the course of this study. For matrix multiplication the CUDA code generated by HMPP for a "simple" port has a performance of 60 – 75% of the CUBLAS performance as seen by comparing Figures 31 and 18, which is a very good result for a small effort. However, after code optimization using good knowledge of the target architecture and HMPP performance comparable to, or even better than, that of CUBLAS was obtained, as seen in Figure 32.

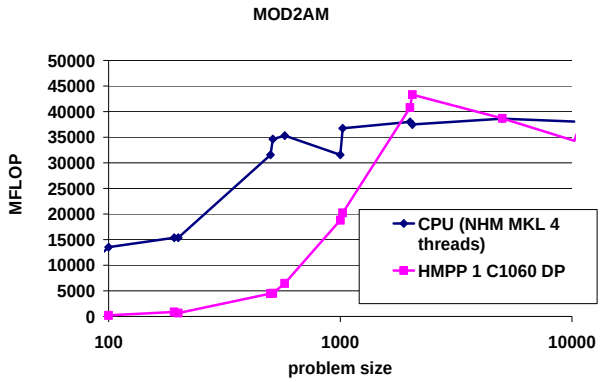

Fig. 31. Mod2am performance on the C1060 using HMPP [78].

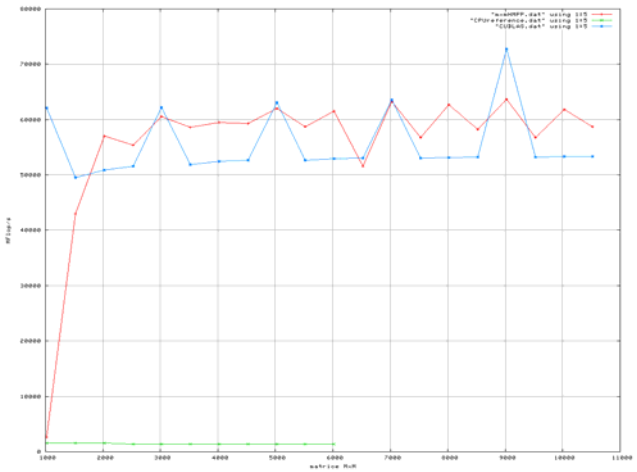

Fig. 32. Optimized performance of matrix multiplication using HMPP compared to CUBLAS for the C1060 [77].

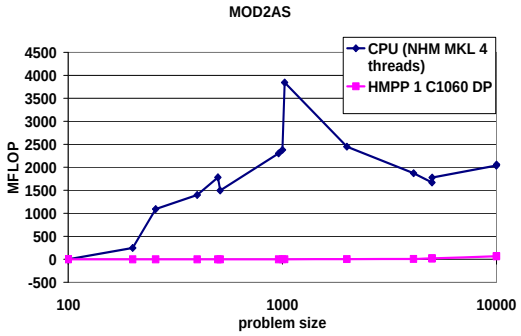

Fig. 33. Mod2as results on the C1060 using HMPP [78].

The lessons learned from the limited use of HMPP are [78]:

Modifying a code to use HMPP to generate a functional code for a GPU is simple. The resulting performance may be quite good for a modest effort, or fairly poor depending on the nature of the computations. For "optimal" performance on a GPU the original code is likely to require modification, unless designed to work well on a streaming architecture.

- Some constructions (such as reductions) are difficult to parallelize and do not perform well on GPUs (or many other highly parallel architectures, some of which have special hardware for reduction operations).
- Producing optimized code for heterogeneous node architectures requires in-depth knowledge of the hardware (not specific to HMPP or GPUs)
- Astute directives for code generation (such as loop reordering, loop fusion, etc.) are a great help to boost performance.
- The performance of codes generated by using HMPP can be equal to or better than that offered by vendor libraries, which is very encouraging.

3.2 RapidMind

The RapidMind Multi-Core Development Platform [103, 104] was designed for application code portability across platforms, including multi-core CPUs, GPUs and the CBE [57]. About a year after this study was initiated RapidMind was ac-

quired by Intel and the RapidMind technology integrated with Intel's Ct technology [14, 15, 104,105] and some of it recently released as part of Intel's Array Building Blocks (ArBB) [107, 108, 109]. RapidMind targeted a data parallel programming model (as did Ct) but did support task-parallel operations. RapidMind added special types and functions to C++ enabling a programmer to define operations (functions) on streams (special arrays). By the freedom to define array operations RapidMind supported more powerful array operations than, e.g., those available in Fortran. Data dependencies and data workflows could be easily described and information necessary for an efficient parallelization included. The compiler and the runtime environment had sufficient information to decide how to auto-parallelize code.

We report results using RapidMind to generate code for the C1060 for matrix multiplication, Figure 34, sparse matrix-vector multiplication, Figure 35, and the radix-4 complex-to-complex 1-D FFT, Figure 36. As can be seen from Figure 34 RapidMind only achieves about 25% of the performance of CUBLAS. The "simple" version was created by adding 20 lines of RapidMind code to the mod2am code from EuroBen. The GPU-optimized code made use of code downloaded from the RapidMnd developer web site. For sparse matrix-vector multiplication RapidMind again achieved about a quarter of the performance of CUBLAS, and for the FFT it achieved about 20% of the performance of our CUDA code. Using RapidMind a first executable was fairly easy to generate, but to achieve good performance significant work and insight into RapidMind and the target architectures was necessary. A more in-depth discussion of the RapidMind porting effort can be found in [110]

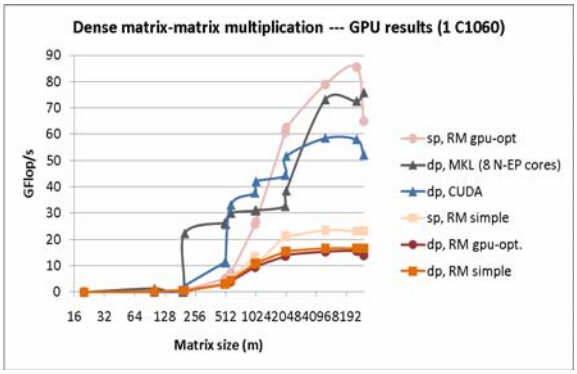

Fig. 34. . Mod2am results using RapidMind compared to using CUDA on the C1060 and MKL on the reference platform [78].

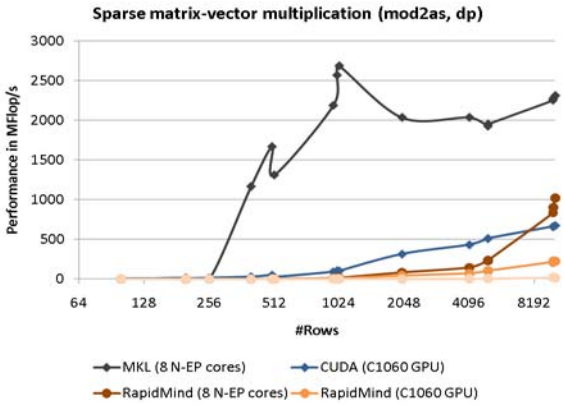

Fig. 35. Mod2as results using RapidMind compared to CUDA on the C1060 and MKL on the reference platform [78].

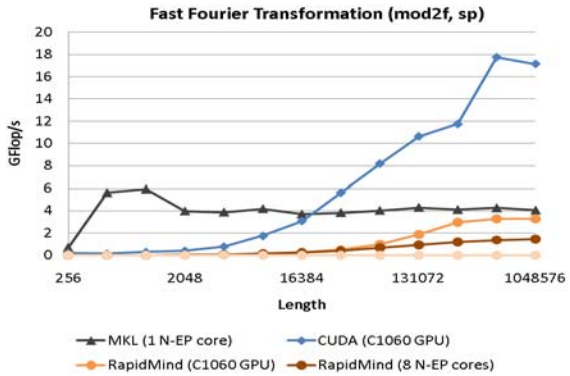

Fig. 36. Mod2f results using RapidMind compared to CUDA on the C1060 and MKL with one thread on the reference platform [78].

3.3 PGI Accelerator Compilers

PGI in 2009 released enhanced versions of their C and Fortran compilers that use directives to control code generation for specialized hardware [71, 72] like GPUs using a similar approach to the one used for the HMPP preprocessor. We investigated the PGI accelerator compiler capabilities on the matrix multiplication and sparse matrix-vector multiplication EuroBen codes. The results are shown in Figures 37 and 38. For matrix multiplication the generated code achieved a peak performance slightly in excess of 8GF, or 11% of the peak C1060 performance. On the host platform the PGI compiler generated code achieved at best 17% of theoretical peak. For the sparse matrix-vector multiplication it is interesting to note that he PGI compiler generated code achieves a performance comparable to that of MKL. However, on the GPU the performance of the PGI generated code is significantly lower than that of the CUBLAS, see Figure 19.

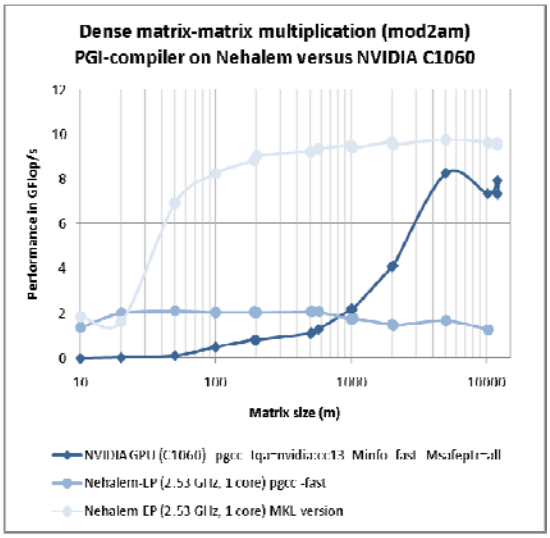

Fig. 37. Mod2am results using the PGI Accelerator C compiler for the C1060 [78].

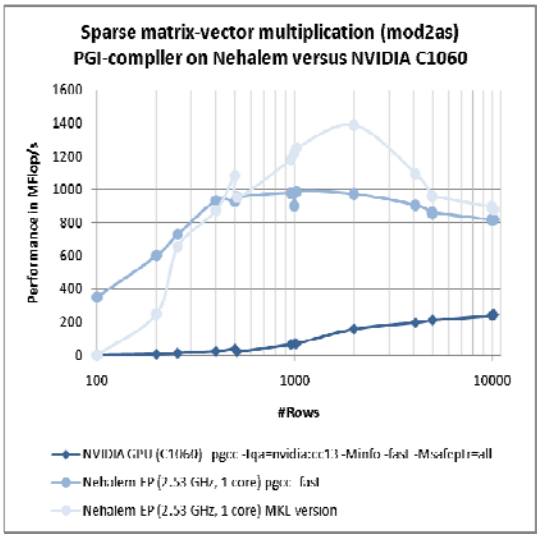

Fig. 38. Mod2as results using the PGI Accelerator C compiler for the C1060 [78].

3.4 Programming Tools Comparison

The CAPS HMPP preprocessor as well as the PGI Accelerator Compiler and RapidMind were immature tools at the time of this study. OpenCL was also included but the beta compilers available during this effort were buggy and the porting efforts had significant problems. For matrix multiplication, which is ideal for many architectures, including GPUs, and a function for which many compilers perform well, the GPU codes generated by RapidMind and the PGI Accelerator Compiler were not good. The performance achieved was at best about 20% and 11% of peak, respectively. The CAPS HMPP preprocessor did better and after an optimization effort generated code that performed comparable to the CUBLAS, a very good result. For sparse matrix-vector multiplication all tools generated poor code for the C1060 with a peak performance of much less than 1% of peak and a factor 5 – 10 worse than CUBLAS. The PGI Accelerator Compiler generated good code though for the host (comparable in performance to MKL).

Clearly, producing code that performs well is a very important aspect of a programming tool for HPC. However, ease of use including debugging is also important for productivity [111] as determined in the DARPA High Productivity Com-

puting Systems program [112]. Measuring productivity is difficult since many hard to measure factors may influence the outcome, such as the programmers experience with similar programming tasks, familiarity with the tools, platform etc, and the extent to which code needs to be modified or entirely redesigned, rewritten and debugged. Though the number of lines of code is a debated measure it is generally agreed that error rates and debugging time are likely to increases with increased code size, and that fewer lines of code is an indication of the expressiveness and quality of a language. Thus, for the porting efforts undertaken the number of lines of code was measured, and so was the time to produce and debug the code, and in some cases optimize it. Though there is much uncertainty in the data it nevertheless appears true that languages and programming models targeting expressiveness do result in shorter codes [113].

Table 9 shows a sample of the measurements. The lines of codes reported are true but unfortunately misleading in that the HMPP code includes several versions that were produced in attempting to get good performance [78]. But the code size for one version is nevertheless larger than for RapidMind. The reported time for RapidMind, which include learning the tool, does show that the learning curve can be significant for new tools that address a complex programming situation.

	Lines of code		Development time		Performance (% of peak)	
	mod2am	mod2as	mod2am	mod2as	mod2am	mod2as
CAPS HMPP	976	979	5	0.5	78.99	0.09
RapidMind	30	27	18.5	12	19.85	0.29

Table 9. Programmer productivity measurements for mod2am and mod2as using CAPS HMPP and RapidMind.

4 Conclusions

Though double-precision arithmetic performance was not a strong point for GPU at the time of the evaluation, the expected evolution of GPU performance and programmability and potential advantage in energy efficiency made it interesting for PRACE to evaluate GPUs as accelerators for future HPC systems, in particular from a programming and energy efficiency point of view. The programming issues associated with heterogeneous node architectures and the streaming architecture of GPUs are likely to remain as support for double-precision arithmetic and programming flexibility in future generations of GPUs improve. This expectation has already been realized to some degree since our study was undertaken.

In regards to the fraction of peak performance achieved for the Nehalem CPU, and the C1060 and the CSX700 by themselves, the Nehalem performed the best for matrix multiplication with an efficiency of 93.9%, with the CSX700 being second at 88.5% and the C1060 being third at 82.1%. From an energy perspective the CSX700 outperformed the CPU by a factor of more than 11 and the C1060 by an estimated factor of more than 15. For HPL the differences were somewhat less dramatic with the CSX700 having a power efficiency about 7 times higher than the Nehalem CPU and a power efficiency more than 10 times that of the C1060. For FFT, the C1060 with a good library implementation achieved a peak efficiency of close to 40%, about 50% better than the CPU. The CSX700 did not perform well and only achieved 25% efficiency. However, because it only consumes less than 10% of the power of the C1060, it still had the best power efficiency, estimated at more than three times that of the C1060. The estimated power efficiency of the Nehalem for FFT was about half of that of the C1060, but the Nehalem CPU itself has comparable power efficiency. The C1060 performed surprisingly well on the sparse matrix-vector multiplication benchmark and indeed performed significantly better than the CPU, and should also, despite its high power consumption have a power efficiency much better than the CPU. The CSX700 performed very poorly on the sparse matrix-vector multiplication.

In regards to an integrated system as expected the data transfers between the CPU memory and the accelerators have a significant impact on the benefit for less compute intensive tasks, such as FFT and in particular sparse matrix-vector multiplication. For compute intensive tasks such as matrix multiplication and HPL the accelerators offers both a performance boost and improved power efficiency,

while for computations such as FFT the performance improvement may be less but a power efficiency improvement may still be possible because “shared infrastructure”, such as memory, is included in the host measures.

For all the power efficiency measurements and estimate we caution that results can easily be misleading depending on the intended objective. Measuring total power consumption and performance is fairly straightforward, but in attempting to assess what to expect for future generation systems it is necessary to have observations for the various components themselves, since the components are likely to change in different ways. Component energy measurements are difficult and may require hardware modification, in particular for current measurements.

The programming tools that were evaluated were all immature, with OpenCL being so immature that reliable results were not obtained. For HMPP and RapidMind creating a usable code was quite simple and only required a modest learning effort, but creating an efficient code required a measurable effort requiring good knowledge of both the tool and the architecture of the target devices.

Acknowledgements

The results reported here are due to efforts by many members of the PRACE Preparatory Phase Work Package 8 and documented in a deliverable to the European Commission under grant agreement RI-211528 within the EU Commission’s infrastructure initiative INFRA-2007-2.2.2.1. Support for this effort has also been received from SNIC, the Swedish National Infrastructure for Computing a metacenter for HPC under the Swedish Research Council which is gratefully acknowledged.

References

4004 Single Chip 4-Bit P-Channel Microprocessor (1987): Intel Corporation. [17]

AMD Opteron™ Processor Pricing (2011) *Advanced Micro Devices, Inc.* Retrieved May 2, 2011, from http://www.amd.com/us/products/pricing/Pages/server-opteron.aspxAdvanced Micro Devices, Inc. [46]

ATI Radeo HD 5870 Graphics *Advanced Micro Devices, Inc.* Retrieved May 2, 2011, from http://www.amd.com/us/products/desktop/graphics/ati-radeon-hd-5000/hd-5870/Pages/ati-radeon-hd-5870-overview.aspx#2Advanced Micro Devices, Inc. [37]

CAPS) *CAPS enterprise*, from http://www.caps-entreprise.com/index.phpCAPS enterprise [68]

Cell (2011, April 28) *Wikipedia*, from http://en.wikipedia.org/w/index.php?title=Cell&oldid=426379510Wikipedia [58]

ClearSpeed *ClearSpeed Technology*, from http://www.clearspeed.com/ClearSpeed Technology [75]

Comparison of AMD graphics processing units (2011, May 2) *Wikipedia*, from http://en.wikipedia.org/w/index.php?title=Comparison_of_AMD_graphics_processing_units&oldid=427053994Wikipedia [36]

Comparison of Nvidia graphics processing units (2011, May 3) *Wikipedia*, from http://en.wikipedia.org/wiki/Comparison_of_Nvidia_graphics_processing_unitsWikipedia [38]

Connection Machine (2011, April 25) *Wikipedia*, from http://en.wikipedia.org/wiki/Connection_MachineWikipedia [8]

CSX700 Datasheet (2011). (06-PD-1425 Rev 1E). [79]

CSX700 Processor (2011). [76]

CUDA (2011, May 2) *Wikipedia*, from http://en.wikipedia.org/w/index.php?title=Special:Cite&page=CUDA&id=427059959 Wikipedia [73]

CUDA Case Studies (2009). [95]

CXSL User Guide (2010). (06-RM-1305), 54. [98]

EuroBen Benchmark *EuroBen*, from http://www.euroben.nl/index.phpEuroBen [83]

Evans & Sutherland (2011, March 18) *Wikipedia*, from http://en.wikipedia.org/wiki/Evans_%26_SutherlandWikipedia [26]

Evans & Sutherland) *Evans & Sutherland*, from http://www.es.com/Evans & Sutherland [24]

Flynn's Taxonomy (2011, May 4) *Wikipedia*, from http://en.wikipedia.org/wiki/Flynn's_taxonomyWikipedia [11]

GeForce 256 *NVIDIA Corporation* Retrieved May 2, 2011, from http://www.nvidia.com/page/geforce256.htmlNVIDIA Corporation [30]

Goodyear MPP (2011, April 25) *Wikipedia*, 2011, from http://en.wikipedia.org/wiki/Goodyear_MPPWikipedia [7]

GPU shipments report by Jon Peddie Research *Jon Peddie Research*, from http://jonpeddie.com/publications/market_watch/Jon Peddie Research [31]

Graphics processing unit (2011, May 4) *Wikipedia*, from http://en.wikipedia.org/w/index.php?title=Graphics_processing_unit&oldid=427152592Wikipedia [29]

HMPP Open Standard (2011, February 23) *Wikipedia*, from http://en.wikipedia.org/w/index.php?title=HMPP_Open_Standard&oldid=415481893 Wikipedia [67]

HP Challenge Benchmark Record (2011) *University of Tennessee* Retrieved May 2, 2011, from http://icl.cs.utk.edu/hpcc/hpcc_record.cgi?id=403University of Tennessee [43]

HPC Challenge Benchmark Record (2011) *University of Tennessee*, from http://icl.cs.utk.edu/hpcc/hpcc_record.cgi?id=434University of Tennessee [45]

Hybrid Multi-core Parallel Programming Workbench,) *CAPS Enterprise*, from http://www.caps-entreprise.com/fr/page/index.php?id=49&p_p=36CAPS Enterprise [66]

IA-32 (Intel Architecture 32-bit) (2011, April 29) *Wikipedia*, from http://en.wikipedia.org/wiki/IA-32Wikipedia [35]

ILLIAC IV (2011, April 25) *Wikipedia* Retrieved May 2, 2011, from http://en.wikipedia.org/wiki/ILLIAC_IVWikipedia [5]

ILLIAC IV System Characteristics and Programming Manual (1972). Burroughs Corporation. [6]

ILLIAC IV *Burroughs Corporation*, from http://archive.computerhistory.org/resources/text/Burroughs/Burroughs.ILLIAC%20IV.1974.102624911.pdfBurroughs Corporation [3]

Intel 4004 (2011, April 2) *Wikipedia*, from http://en.wikipedia.org/wiki/Intel_4004Wikipedia [20]

Intel 56XX Series Products (Formerly Westemere-EP_) *Intel Corporation* Retrieved May 2, 2011, from http://ark.intel.com/ProductCollection.aspx?codeName=33174Intel Corporation [40]

Intel Hyper-Threading Technology (Intel HT Technology)) *Intel Corporation*, from http://www.intel.com/technology/platform-technology/hyper-threading/index.htmIntel Corporation [88]

Intel Math Kernel Library *Intel Corporation*, from http://software.intel.com/en-us/articles/intel-mkl/Intel Corporation [85]

Intel Processor Clock Speed (MHz)), from http://smoothspan.files.wordpress.com/2007/09/clockspeeds.jpg [49]

Intel Xeon Processor E5540) *Intel Corporation*, from http://ark.intel.com/Product.aspx?id=37104&processor=E5540&spec-codes=SLBF6Intel Corporation [97]

Intel(R) Array Building Blocks for Linux OS, User's Guide (2011). (324171-006US), 74. [109]

Intel(R) Array Building Blocks Virtual Machine, Specification (2011). (324820-002US), 118. [108]

Intel's Ct Technology Code Samples (2010) *Intel*, from http://software.intel.com/en-us/articles/intels-ct-technology-code-samples/Intel [106]

Introducing Intel many Integrated Core Architecture) *Intel Corporation*, from http://www.intel.com/technology/architecture-silicon/mic/index.htmIntel Corporation [61]

LDS-1/PDP-10 Display System. In ESC Corporation (Ed.). Salt Lake City: Evans & Sutherland Computer Corporation. [23]

Linpack, ClearSpeed (2010) *CleerSpeed Technology Limited*, from http://www.clearspeed.com/applications/highperformancecomputing/hpclinpack.phpCleerSpeed Technology Limited [99]

Memory Bandwidth (STREAM) – Two-Socket Servers (including AMD Opteron™ 6100 Series Processors) (2011) *Advanced Micro Devices, Inc.* Retrieved May 2, 2011, from http://www.amd.com/us/products/server/benchmarks/Pages/memory-bandwidth-stream-two-socket-servers.aspxAdvanced Micro Devices, Inc. [44]

Non-Uniform Memory Access (2011, May 1) *Wikipedia*, from http://en.wikipedia.org/wiki/Non-Uniform_Memory_AccessWikipedia [84]

OpenCL - The open standard for parallel programming of heterogeneous systems (2011) *Khronos Group*, from http://www.khronos.org/opencl/Khronos Group [62]

Pentium 4 (2011, May 3) *Wikipedia* Retrieved May 2, 2011, from http://en.wikipedia.org/wiki/Pentium_4Wikipedia [48]

PGI Accelerator Programming Model for Fortran & C (2010). 36. [72]

Portland Group Inc. Accelerated Compilers) *STMicroelectronics* from http://www.pgroup.com/resources/accel.htmSTMicroelectronics [71]

PRACE *PRACE*, from http://www.prace-ri.eu/PRACE [65]

PRACE Preparatory Phase Project, Deliverable 8.3.1, Technical Component Assess-ment and Development Report (2009). [77]

Productivity benefits of Intel Ct Technology (2010) *Intel Corporation*, from http://software.intel.com/en-us/articles/productivity-benefits-of-intel-ct-technology/Intel Corporation [105]

RapidMind (2011, April 29) *Wikipedia*, from http://en.wikipedia.org/wiki/RapidMindWikipedia [16]

RapidMind (2011, April 29) *Wikipedia*, from http://en.wikipedia.org/wiki/RapidMindWikipedia [69]

Silicon Graphics (2011, March 29) *Wikipedia*, from http://en.wikipedia.org/wiki/Silicon_GraphicsWikipedia [28]

Sophisticated library for vector parallelism, Intel Array Building Blocks: A Flexible Parallel Programming Model for Multicore and Many-Core Architectures) *Intel Corporation*, from http://software.intel.com/en-us/articles/intel-array-building-blocks/Intel Corporation [107]

Tesla C1060 Computing Processor Board Specification (2010). (BD-04111-001_v06). [96]

Tesla C2050/C2070 GPU Computing Processor (2010): NVIDIA Corporation. [39]

The Cell project at IBM Research *IBM*, from http://www.research.ibm.com/cell/IBM [59]

The Cray-1 Computer System (1976). Minnesota: Cray Research, Inc. [2]

The Green 500: ranking the world's most energy-efficient supercomputers (2010) *The Green500* Retrieved May 2, 2011, from www.green500.orgThe Green500 [92]

The Intel 4004, A Big Deal Then, A Big Deal Now) *Intel Corporation* Retrieved May 2, 2011, from http://www.intel.com/about/companyinfo/museum/exhibits/4004/facts.htmIntel Corporation [18]

The OpenCL Specification Version: 1.1 (2010). 379. [63]

Top 500) *Top500.org*, from http://www.top500.org/Top500.org [21]

What is CUDA? (2011) *NVIDIA Corporation*, from http://www.nvidia.com/object/what_is_cuda_new.htmlNVIDIA Corporation [74]

Writing Applications for the GPU Using the RapidMind™ Development Platform (2006). 7. Retrieved from http://www.cs.ucla.edu/~palsberg/course/cs239/papers/rapidmind.pdf [70]

A. Ali, Johnsson, L, & Mirkovic, D (2007). *Empirical Auto-tuning Code Generator for FFT and Trigonometric Transforms.* Paper presented at the 5th Workshop on Optimizations for DSP and Embedded Systems, 2007 International Symposium on Code Generation and Optimization, San Jose, CA. [87]

A. Ghuloum, TS, G. Wu, X. Zhou, J. Fang, P. Guo, B. So, M. Rajagopalan, Y.Chen, B. Chen. (2007). Future-Proof Data Parallel Algorithms and Software on Intel® Multi-Core Architecture. *Intel Technology Journal, 11*(4). [15]

A. Petitet, RCW, J. Dongarra, A. Cleary (2008). HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers *University of Tennessee Computer Science Department* Retrieved May 2, 2011, from http://www.netlib.org/benchmark/hpl/University of Tennessee Computer Science Department [22]

Alan D. Simpson, MB, Jon Hill (2008). [80]

Anwar Ghuloum, ES, Jesse Fang, Gansha Wu, Zin Zhou (2007). Ct: A Flexible Parallel Programming Model for Tera-scale Architectures. [14]

Asanovic, K, Bodik, R, Catanzaro, BC, Gebis, JJ, Husbands, P, Keutzer, K, . . . Yelick, KA (2006). The Landscape of Parallel Computing Research: A View from Berkeley. (UCB/EECS-2006-183). [82]

Belady, CL. (2007). In the Data Center, Power and Cooling Costs More Than the IT Equipment It Supports. *Electronics Cooling.* [47]

Bell, BS (2009). RV870 Architecture *FS Media, Inc.*, from http://www.firingsquad.com/hardware/ati_radeon_hd_5870_performance_preview/page3.aspFS Media, Inc. [54]

Christadler, I, & Weinberg, V (2010). *RapidMind: Portability across Architectures and its Limitations.* Paper presented at the Facing the Multi-core Challenge (Conference Proceedings), , Heidelberg. [110]

Clark, J. (1980). A VLSI Geometry Processor for Graphics. *Computer Magazine, 13*(7), 59-68. [26]

Clark, J. (1982). The Geometry Engine: A VLSI Geometry Systems for Graphics. *Computer Graphics, 16*(3), 127-133. [27]

Colella, P (2004). *Defining Software Requirements for Scientific Computing.* [81]

Dolbeau, R, Bihan, S, & Bodin, F (2007). *HMPP: A Hybrid Multi-core Parallel Programming Environment*. Paper presented at the roceedings of the Workshop on General Purpose Processing on Graphics Processing Units (GPGPU 2007), Boston. http://www.caps-entreprise.com/upload/ckfinder/userfiles/files/caps-hmpp-gpgpu-Boston-Workshop-Oct-2007.pdf [100]

Dolbeau, R, Bihan, S, & Bodin, F (2007). *HMPP: A Hybrid Multi-core Parallel Programming Environment.* Paper presented at the Proceedings of the Workshop on General Purpose Processing on Graphics Processing Units (GPGPU 2007), Boston. [102]

Dongarra, J, Graybill, R, Harrod, W, Lucas, R, Lusk, E, Luszczek, P, . . . Tikir, M. (2008). DARPA's HPCS Program: History, Models, Tools, Languages. *Advances in Computers, 72*, 1-100. [112]

E. Phillips, MF (2010). CUDA Accelerated Linpack on Clusters, E. Phillips. [93]

Ed Grochowski, MA Energy per Instruction Trends in Intel® Microprocessors. [52]

Erbacci, G, Cavazzoni, C, Spiga, F, & Christadler, I (2009). Report on Petascale Software Libraries and Programming Models. *Deliverable 6.6*(RI-211528), 163. [113]

Feldman, M (2009). Benchmark Challenge: Nehalem versus Istanbul, HPC Wire. *HCP Wire*. Retrieved from http://www.hpcwire.com/hpcwire/2009-06-18/benchmark_challenge_nehalem_versus_istanbul.html [90]

Gelas, JD (2008). Linpack: Intel's Nehalem versus AMD Shanghai. *Anandtech*. Retrieved from http://www.anandtech.com/show/3470 [91]

Gelas, JD (2010). AMD's 12-core "Magny-Cours" Opteron 6174 vs. Intel's 6-core Xeon *Anandtech*, from http://www.anandtech.com/show/2978Anandtech [41]

Hills, WD (1989). *The Connection Machine*. Cambridge: MIT Press. [9]

Homberg, W (2009). Network Specification and Software Data Structures for the eQPACE Architecture *Jülich Supercomputing Center (JSC)*, from http://www2.fz-juelich.de/jsc/juice/eQPACE_Meeting/Jülich Supercomputing Center (JSC) [60]

John Shalf, DD, Leonid Oliker, Michael Wehner (2006). *Green Flash: Application Driven System Design for Power Efficient HPC.* Paper presented at the Salishan Conference on High-Speed Computing. [64]

Johnsson, L (2011). Overview of Data Centers Energy Efficiency Evolution. In SR I Ahmad (Ed.), *Handbook of Green Computing*: CRC Press. [56]

Kanellos, M (2001). Intel's Accidental Revolution. *CNET News*. Retrieved from CNET News website: [19]

Kennedy, K, Koelbel, C, & Schreiber, R. (2004). Defining and Measuring the Productivity of Programming Languages. *International Journal of High Performance Computing Applications, 18*(4), 441-448. [111]

Kozin, IN (2008). Evaluation of ClearSpeed Accelerators for HPC. 15. [101]

Matsuoka, S, & Dongarra, J TESLA GPU Computint. [94]

McCalpin, JD). STREAM: Sustainable Memory Bandwidth in High-Performance Computers *University of Virginia* Retrieved May 2, 2011, from http://www.cs.virginia.edu/stream/University of Virginia [42]

McCool, MD. (2007). RapidMind Multi-core Development Platform. *CASCON Cell Workshop*. [104]

McCool, MD. (2008). Developing for GPUs, Cell, and Multi-core CPUs Using a Unified Programming Model. *The Linux Journal*. [103]

Mirkovic, D, Mahasoom, R, & Johnsson, L (2000). *An Adaptive Software Library for Fast Fourier Transforms.* Paper presented at the 2000 International Conference on Supercomputing, Santa Fe, NM. [86]

Moore, GE. (1965). Craming more components onto integrated circuits. *Electronics, 38*(8), 114-117. [1]

Introduction to Parallel GPU Computing, Center for Scalable Application Development Software (2010 July 26-29). [34]

Petrov, V, & Fedorov, G (2010). MKL FFT performance – comparison of local and distributed-memory implementations. *Intel Software Network.* Retrieved from http://software.intel.com/en-us/articles/mkl-fft-performance-using-local-and-distributed-implementation/ [89]

Pettey, C. (2011). Gartner Says Worldwide PC Shipments in Fourth Quarter of 2010 Grew 3.1 Percent; Year-End Shipments Increased 13.8 Percent: Gartner, Inc. Retrieved from http://www.gartner.com/it/page.jsp?id=1519417. [32]

Pettey, C, & Stevens, H (2011). Gartner Says 2010 Worldwide Server Market Returned to Growth with Shipments Up 17 Percent and Revenue 13 Percent *Gartner, Inc.* Retrieved May 2, 2011, from http://www.gartner.com/it/page.jsp?id=1561014Gartner, Inc. [33]

Pollack, F (1999). *New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies.* Paper presented at the Proceedings of the 32nd Annual IEEE/ACM International Symposium on Microarchitecture, Haifa, Israel. [51]

Ramnath Sai Sagar, JL, Aad van der Steen, Iris, & Christadler, HH (2010). PRACE Preparatory Phase Project Deliverable 8.3.2, Final technical report and archi-tecture proposal. [78]

Shimpi, AL (2010). New Westmere Details Emerge: Power Efficiency and 4/6 Core Plans *AnandTech, Inc.*, from http://www.anandtech.com/show/2930AnandTech, Inc. [53]

Team, TsG (2005). The Mother of All CPU Charts 2005/2006 *Bestofmedia Network*, from http://www.tomshardware.com/reviews/mother-cpu-charts-2005,1175.htmlBestofmedia Network [50]

Thelen, E (2003). The Connection Machine -1-2-5 *Ed-Thelen.org* Retrieved May 2, 2011, from http://ed-thelen.org/comp-hist/vs-cm-1-2-5.htmlEd-Thelen.org [10]

Thelen, E (2005). ILLIAC IV *Ed-Thelen.org* Retrieved May 2, 2011, from http://ed-thelen.org/comp-hist/vs-illiac-iv.htmlEd-Thelen.org [4]

Thomas Chen, RR, Jason Dale, Eiji Iwata (2005). Cell Broadband Engine Architecture and its first implementation. Retrieved from https://www.ibm.com/developerworks/power/library/pa-cellperf/ [57]

Thornton, JE (1963). Considerations in Computer Design - Leading Up to the Control Data 6600. [13]

Thornton, JE (1970). *The Design of a Computer: The Control Data 6600*. Glenview: Scott, Foresman and Company. [12]

Valich, T (2010). nVidia GF100 Architecture: Alea iacta est. Retrieved from http://www.brightsideofnews.com/print/2010/1/18/nvidia-gf100-architecture-alea-iacta-est.aspx [55]