Audio Engineering Society

Convention Paper

Presented at the 126th Convention
2009 May 7–10 Munich, Germany

The Potential of High Performance Computing in Audio Engineering

David Moore[1], Jonathan Wakefield[1]

[1] School of Computing and Engineering, University of Huddersfield, West Yorkshire, HD1 3BB, UK
d.j.moore@hud.ac.uk

ABSTRACT

High Performance Computing (HPC) resources are fast becoming more readily available. HPC hardware now exists for use in conjunction with standard desktop computers. This paper looks at what impact this could have on the Audio Engineering industry. Several potential applications of HPC within audio engineering research are discussed. A case study is also presented which highlights the benefits of using the Single Instruction, Multiple Data (SIMD) architecture when employing a search algorithm to produce surround sound decoders for the standard 5-speaker surround sound layout.

1. INTRODUCTION

High Performance Computing (HPC) is the use of computers to solve numerically intensive problems [1]. It includes computing systems from multiple workstations to supercomputers assigned to solve the some of the world's most demanding computational problems. Currently, HPC implementation involves distributing a problem across multiple processors that operate in parallel. Breaking down a problem in this manner can result in significant increases in speed of execution over traditional approaches where processes are run in series.

In the past, mainly due to expense, access to HPC systems has been restricted to large organisations and academic institutions. However, HPC resources are now becoming more readily available and affordable [2]. For example, a number of companies have developed HPC products ready to be used in conjunction with desktop computers (e.g. ClearSpeed Accelerator Cards [3] and NVIDIA Graphics Processing Units [4]). In addition, networking technologies have advanced significantly allowing computers to be easily linked to form computer “clusters” and “grids” capable of sharing the load of a data processing problem [5]. Despite the growing availability of HPC resources, it is the belief of the authors that HPC is not yet widely applied in audio engineering research. The aim of this paper is to highlight the potential that this mode of computing offers to this field.

The following section provides background on HPC and looks at some existing HPC technologies. In section 3 a number of potential applications of HPC within audio engineering are discussed. These will include highly challenging applications which are currently regarded as too computationally expensive either to compute or perform in real-time. In section 4 a case study is presented where the power of HPC is demonstrated for a specific audio engineering application - the optimisation of a 5-speaker Ambisonic decoder using a computer search algorithm.

2. BACKGROUND

HPC is currently used for a wide range of applications from the simulation and modelling of physical phenomena, to data mining and visualisation [6-8]. Before discussing existing HPC technologies and potential applications within audio engineering, HPC architectures will be defined.

2.1. HPC Architectures

Flynn [9] proposed a simple, but broad, classification of high performance computer architectures based on the number of instruction streams and data streams that can be processed simultaneously (figure 1 illustrates these architectures). They are:

SISD (Single Instruction, Single Data) - defines the operation of a sequentially operating single CPU computer. Speed of execution is limited to the internal speed of the computer.

SIMD (Single Instruction, Multiple Data) - involves multiple processors simultaneously executing the same instruction but on different data. A central control unit is employed for issuing instructions.

MISD (Multiple Instruction, Single Data) - involves multiple instructions operating on a single stream of data. This method is rarely used as it makes more sense to use multiple instructions to operate on multiple data.

MIMD (Multiple Instructions, Multiple Data) – this architecture employs multiple processors that execute multiple instructions on different data streams. Each processor operates independently and asynchronously.

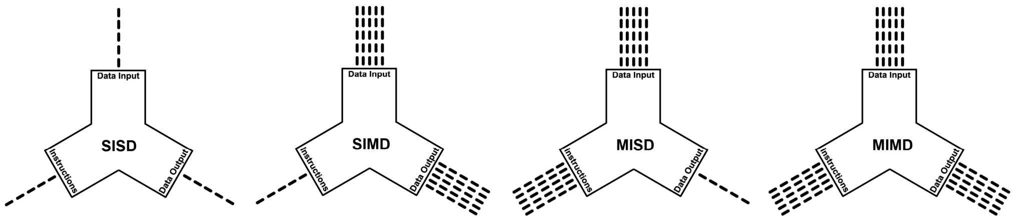

Figure 1: HPC architectures defined by Flynn

In the current era of HPC only SIMD and MIMD architectures tend to be used because of their parallel processing nature. The combined processing power of a multi-core SIMD or MIMD system in general significantly exceeds the speed of the fastest single-core SISD system. Furthermore, the advance in single-core processor speeds is gradually slowing because of the physical limitations of chip manufacturing with present day materials [5].

HPC systems can also be classified by how their processors and memory are connected. They generally fall into two different categories: *shared memory* systems or *distributed memory* systems. In a shared memory system all processors have equal access to a global memory space so changes made to data by one processor, are seen by all other processors (illustrated in figure 2). Such systems usually employ high-speed, low latency, inter-connection (i.e. buses or switches) and are relatively easy to program.

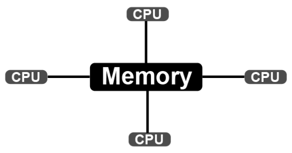

Figure 2: Shared memory

In contrast, a distributed memory system typically consists of multiple processors each with its own local memory space (see figure 3).

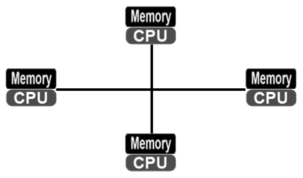

Figure 3: Distributed memory

The key issue in this type of system is the potential performance overhead introduced by inter-processor communication. Programming for this architecture is also considered harder as the computer programmer has to handle all communication operations. However, one of the main advantages of distributed memory systems is that they are more scalable than shared memory systems (i.e. the number of processors can be increased without significant decrease in overall efficiency) [5].

Hybrid systems (*distributed-shared memory*) are also sometimes employed (see figure 4). A distributed-shared memory system combines the best of both architectures in that inter-processor communication is reduced and systems are easier to program [5].

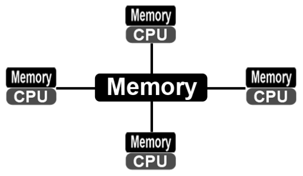

Figure 4: Distributed-shared memory

2.2. Existing HPC Technologies

One of the clear trends in HPC is expensive specially designed hardware is being substituted by more cost-effective off the shelf solutions [5]. This section will look at some of these options.

2.2.1. Graphics Processing Units (GPUs)

In recent years GPUs have shown a marked increase in their performance and capabilities. Moreover, general purpose computations can now more easily be made on graphics hardware allowing them to be used for high performance computation. Their architectures, which are normally SIMD, are highly parallel consisting of many processor cores. For example, one current era shared memory GPU card (NVIDIA Tesla C1060) consists of 240 processor cores which when combined can give a peak single precision floating processing performance of 933 GFLOPS/s. To put this in perspective, this kind of computational power was not generated by expensive specially designed supercomputers until about 1997 [10].

Figure 5: The NVIDIA Tesla C1060

In addition to hardware advances, GPU companies are encouraging software development for their architectures by providing higher level programming models (e.g. CUDA by NVIDIA). These models incorporate libraries that are commonly used for audio processing (e.g. Fast Fourier Transforms).

In the literature there are a growing number of examples of HPC using GPUs (see for example [11-14]). In one relevant example, Röber et al report a performance increase of at least a factor of 25 when using a GPU for computing sound wave propagation with waveguide meshes [15].

2.2.2. Dedicated Hardware Accelerators

Another method of augmenting a desktop computer's processing power is to employ an application accelerator card specially designed for HPC such as those developed by ClearSpeed.

Like GPUs, ClearSpeed accelerators come equipped with multiple parallel processors. However, ClearSpeed cards actually employ an enhanced SIMD architecture where each processor has its own dedicated memory as well as access to a shared global memory space. These cards are also very efficient in terms of power consumption (i.e. average of 25W per card).

There are several examples in the literature that demonstrate the performance gains which can be achieved with accelerator hardware (see for example [16, 17]). One relevant example by Bradford et al focused on implementing the computationally expensive Sliding Phase Vocoder [18].

2.2.3. Clusters

A cluster is a group of stand-alone computers connected via a network (MIMD architecture). The combined group of computers typically functions as a single centrally managed system with distributed or distributed-shared memory [5].

Clusters can be loosely coupled or tightly coupled. In a loosely coupled cluster each contributing computer typically connects through the internet or local area network and can operate independently, potentially undertaking different individual tasks alongside the allocated work. Loosely coupled clusters can consist of computers with different architectures. In contrast, each node in a tightly coupled cluster is dedicated to the system at all times. Nodes in a tightly coupled cluster often have the same architecture and run the same operating system. Furthermore, nodes are usually set up in the same room employing a fast low-latency and high bandwidth local area network for inter-connection.

One of the major advantages of clusters is they can be constructed using existing commodity computers. For example, office employees could use the combined power of their computers for working on a problem during out of office hours. This approach offers high performance at relatively low cost. Also of note is the fact that, each node in a cluster could be equipped with a modern GPU or Application Accelerators [19].

2.2.4. Grids

Grids are very similar in principle to loosely coupled clusters in that multiple computers work together over the internet to solve a problem. However, grids can include a much larger range of resources distributed around the globe (including clusters and supercomputers). Grids are also usually constructed with general purpose Grid software (e.g. BOINC) and problems are typically formulated to involve very little or no inter-node communication (communication tends to be much more expensive than computation in parallel architectures).

One of the major advantages of employing a grid for HPC is the potential amount of computational power available. Several international Grid projects created in recent years have many volunteers who donate the idle time of their computers. For example, folding@home (one of the most popular projects) currently has around 1 million users which when combined give computational processing power rivalling today's supercomputers [20, 21]. Another advantage of Grid computing is it can also facilitate collaboration between research groups.

2.3. Summary

We have discussed several technologies which can be used for HPC. Modern GPUs and Application Accelerators offer a compact solution to HPC, whereas clusters and grids have the potential to offer vast amounts of processing power over networks.

Over the next few years the price of HPC technology is expected to reduce and become even more accessible (i.e. development of higher level programming languages and sharing of resources through the internet because of increasing network speeds). This offers huge potential to audio engineers. Audio applications previously disregarded as too processor intensive to compute or perform in real time can be reconsidered.

3. POTENTIAL HPC AUDIO APPLICATIONS

The use of technologies for HPC is application specific. In general, SIMD architectures are well suited to problems with a high degree of regularity. In audio this could include frame-based audio analysis algorithms such as the Fast Fourier Transform, Wavelet Transform or Short Time Fourier Transform. Other ideal candidates for SIMD would be physical modelling of acoustic spaces involving Finite Element Analysis (FEA) or Boundary Element Analysis (BEM). Indeed, FEA applications are known to map naturally to SIMD architectures [22].

Additionally, there are numerous audio processing algorithms which would benefit from the large amount of computational power HPC has to offer. Some of the more obvious candidates are:

- Reverb by the direct modelling of a physical space
- Sophisticated physical modelling of instruments
- Complex sound synthesis algorithms (e.g. further exploration of real-time voice synthesis [23])
- High resolution beamforming
- Real-time digital room correction

In audio reproduction, Wave Field Synthesis (WFS) would be an ideal candidate for exploiting the computational power of HPC hardware. Rendering complex soundfields consisting of many virtual sources requires a significant amount of processing power. Consequently, real-time playback using a standard PC is not currently feasible. HPC hardware would enable the demanding computational tasks and calculus associated with the WFS rendering process to be computed more quickly by sharing the computational load across multiple parallel processors.

Clearly great potential exists for HPC technology in audio engineering. Before realising the full potential of a system, however, careful investigation is required. It is important to realise that all architectural features of a HPC system (i.e. processors, memory, inter-connection) can strongly influence the performance of an algorithm [5].

4. CASE STUDY

4.1. Introduction

The design of an Ambisonic decoder can be formulated as a search problem. The basic principle is to use a computer search algorithm to find a set of decoder coefficients which best fit the design objectives specified in a fitness function.

In previous work a heuristic search algorithm has been employed for finding "good" decoder coefficients for the standard ITU 5-speaker layout [24-26]. This method was employed because searching exhaustively for the best set of decoder coefficients using modern desktop computer processing power is not feasible. In this work, however, we run an exhaustive search albeit with reduced coefficient resolution for a first order Ambisonic decoder on a computer equipped with ClearSpeed HPC hardware. In addition, we also implement a simple local search to investigate whether running multiple local searches at a higher resolution can yield a better solution than when running a lower resolution exhaustive search. In both cases, reference versions are implemented on a standard desktop computer with an Intel Xeon 2.40 GHz processor.

4.2. ClearSpeed HPC Hardware

Two ClearSpeed x620 boards were used for accelerating the searches (see figure 6). The x620 boards have dual CSX600 chips and 1 GB SRAM. Each chip has an array of 96 processor elements (PE) that each operate at 210 MHz and have 6KB of local memory.

Figure 6: ClearSpeed x620

The boards can be programmed in a SIMD style using C^n (an extension of the C programming language). C^n has special data types to differentiate between non-parallel data instances (mono) and parallel data instances (poly) [27]. ClearSpeed provide optimised standard math functions which process poly-scalars (i.e. one calculation per PE) or poly-vectors (i.e. 4 calculations per PE). Poly-vectors more efficiently exploit the parallel architecture of the boards by allowing 384 calculations to be made simultaneously on each chip (i.e. 96 PEs x 4). An example program is provided in figure 7.

```
#include <lib_ext.h>
#include <vmathp.h>

// __NUM_PES__ is the number of processor element
(96)
#define SAMPLES (__NUM_PES__ * 4)
#define PI 3.14159265358979

int main(void)
{
  // __FVECTOR is a poly-vector
  __FVECTOR sine, angle = {0,0,0,0};

  // get_penum() returns the ID of each PE (0 - 95)
  poly int pnum = get_penum();

  // Set up the angles for each element of the
vector
  angle[0] = (__NUM_PES__*0 + pnum) * PI / SAMPLES;
  angle[1] = (__NUM_PES__*1 + pnum) * PI / SAMPLES;
  angle[2] = (__NUM_PES__*2 + pnum) * PI / SAMPLES;
  angle[3] = (__NUM_PES__*3 + pnum) * PI / SAMPLES;

  // calculate sine of angle
  sine = cs_sinp(angle);

  return 0;
}
```

Figure 7: An example C^n program showing the poly and poly-vector data types.

4.3. Implementation

In both search implementations a multiple objective fitness function was used for evaluating decoder coefficient sets generated by the search. The reader is referred to a recent research paper by the authors where the fitness function algorithm is described in detail [24].

4.3.1. Exhaustive Search

Running an exhaustive search is guaranteed to locate the best solution in the search domain. However, it can be time consuming. The number of potential solutions to evaluate in an exhaustive search is dependent on the number of coefficients and also the coefficient resolution. A first order Ambisonic decoder for the ITU 5-speaker layout requires 8 decoder coefficients when the decoder type is frequency-independent (i.e. one set of coefficients for all frequencies). Each coefficient is constrained to the range [0, 1]. Table 1 details the maximum number of potential solutions which need to be evaluated for this type of decoder given different coefficient resolutions.

Resolution	Number of potential solutions
0.2	$6^8 \approx 10^{6.2}$
0.1	$11^8 \approx 10^{8.3}$
0.05	$21^8 \approx 10^{10.6}$
0.01	$101^8 \approx 10^{16.0}$
0.001	$1001^8 \approx 10^{24.0}$
0.0001	$10001^8 \approx 10^{32.0}$

Table 1: Number of potential solutions for a first order frequency-independent Ambisonic decoder for different decoder coefficient resolutions

In the reference version of the exhaustive search, each potential solution is evaluated in series using a nested for-loop structure (each loop variable corresponds to a decoder coefficient). In the accelerated version, however, the nested loops were unrolled with 1536 potential solutions simultaneously evaluated at each iteration of the search algorithm (i.e. each chip on each board evaluates 384 potential solutions in parallel by using the poly-vector data types).

4.3.2. Simple Local Search

A local search was also coded to take advantage of the ClearSpeed architecture. 1536 instances of the search were simultaneously executed in parallel by using the poly-vector data types. Using this method allowed many different areas of the search domain to be explored at the same time.

The starting point for each search was chosen at random. Candidate moves were then made at a fixed resolution in positive and negative directions along each coordinate axis in the search domain (each axis corresponds to a decoder coefficient). This process was repeated until 1000 moves had been made. It should be noted that stopping the search after a fixed number of moves is not normally ideal. It is generally considered more appropriate to stop a search after a fixed number of *bad* moves to allow the search to reach a local minimum (this implementation would be better suited to a MIMD architecture). However on a SIMD architecture this will ensure that all PEs are fully employed because they will all start and end each search at the same time.

4.4. Results

Table 2 details the total search times and best solution fitness values for the reference and accelerated versions of the exhaustive search. The darker cells indicate the times for the reference version.

Resolution	Solution Fitness	Time to complete
0.2	199.2100	137 secs
		4 secs
0.1	171.6900	290 mins
		8 mins
0.05	153.3589	35 days (estimate)
		25 hours
0.01	-	27,710 years (estimate)
		815 years (estimate)

Table 2: Times and fitness values for the reference and accelerated versions of the exhaustive search algorithm. Lower solution fitness scores are better.

For each coefficient resolution the accelerated version of the exhaustive search was able to achieve a speed increase of about a factor of 34 when compared to the reference version. The significance of this can clearly be seen when examining the completion times for a coefficient resolution of 0.05 and the estimated completion times for a coefficient resolution of 0.01.

Table 3 details the fitness values for the best solutions found using the reference and accelerated versions of the local search. Timings are also displayed in seconds (the darker cells indicate the times for the reference version).

Resolution	Solution Fitness	Time to complete
0.2	281.7471	22 mins 28 secs
	180.6340	56 secs
0.1	158.6650	22 mins 27 secs
	159.8490	56 seconds
0.05	156.5300	22 mins 30 secs
	155.0872	56 secs
0.01	152.3650	22 mins 36 secs
	146.8572	56 secs
0.001	149.5521	22 mins 33 secs
	146.4251	56 secs

Table 3: Times and fitness values for the reference and accelerated versions of the local search algorithm. Lower solution fitness scores are better.

In this case, the results show that when running the searches in parallel, there was a 24 fold increase in time-to-solution when compared to the reference version.

Another interesting factor from these results is that running multiple local searches yields a better solution than running an exhaustive search at a low resolution. This is also the case when using the same resolution as the exhaustive search because the local search is started from a random point (as opposed to a rounded number).

4.5. Summary

In summary, this case study has looked at using ClearSpeed HPC hardware to improve the performance of a local search and a global search for Ambisonic decoder coefficients. The results show that for this application a significant increase in speed of execution is possible. For the exhaustive search this makes it possible locate a better global solution by searching at a higher resolution. For the local search more solutions can be evaluated within a space of time increasing the chances of finding a better solution.

It should be noted that newer ClearSpeed hardware is currently available (e710). We expect that by using this hardware it would be possible to improve the search speeds further.

5. CONCLUSIONS

HPC opens the door to new applications in audio engineering. It makes problems feasible that are currently infeasible to run on desktop computers and can allow computationally expensive algorithms to run in real-time.

Currently, SIMD appears to be the most appropriate architecture for HPC audio applications. It is well suited to several audio processing algorithms. In a case study it was shown that by employing SIMD hardware, faster calculations could be made when implementing a search for decoder coefficients. This resulted in better solutions being found in a shorter time.

Finally, at this point we could ask the question - what technical advances or new avenues of research could be made in audio engineering if significantly more computational power was available?

6. ACKNOWLEDGEMENTS

The authors would like to thank ClearSpeed and Russell Bradford of Bath University for providing access to the Accelerator Boards used in this research.

7. REFERENCES

[1] Openshaw, S., and Turton, I., "High Performance Computing and the Art of Parallel Computing", Routledge, 2000.

[2] Strohmaier, E., Dongarra, J.J., Meuer, H.W., and Simon, H.D, "Recent Trends in the Market Place of High Performance Computing", Parallel Computing, vol. 34, no. 3-4, pp. 261-273, 2005.

[3] ClearSpeed Technology plc, http://www.clearspeed.com/ , (last accessed 2009 March).

[4] NVIDIA Corporation, http://www.nvidia.com/object/tesla_computing_solutions.html/, (last accessed 2009 March).

[5] El-Rewinim H., and Abd-El-Barr, M., Advanced Computer Architecture and Parallel Processing, Wiley-Interscience, First Edition, 2005.

[6] Szpakowski, A., Barczak, K., Tyszkiewicz, C., and Pusetelny, T., "Physical Calculations aided with HPC Cluster", J. de Physique IV, vol. 129, pp.155, 2005 October.

[7] Goil, S., and Choudhar, A., "High Performance OLAP and Data Mining on Parallel Computers", Data Mining and Knowledge Discovery, vol. 1, no. 4, pp.391-417, 1997 December.

[8] O'Donovan, A., Duraiswami, R., and Gumerov, N.A., "Real Time Capture of Audio Images and their use with Video", IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2007 October.

[9] Flynn, M.J., "Very High Speed Computing Systems", Proc. IEEE, vol. 54, no. 12, pp. 1901-1909, 1966 December.

[10] Top 500. http://www.top500.org/ (last accessed 2009 March).

[11] Gallo, E., and Tsingos, N., "Efficient 3D Audio Processing with the GPU", ACM Workshop on General Purpose Computing on Graphics Processor, 2004 August.

[12] Alerstam, E., Svensson, T., and Anderson-Engels S., "Parallel Computing with Graphics Processing Units for High-Speed Monte Carlo Simulation of Photon Migration", J. of Biomedical Optics, vol. 13, no. 6, pp. 06504-1–06504-3. 2008 November/December.

[13] Shimobaba, T., Sato, Y., Miura, J., Takenouchi, M., and Ito, T., "Real-Time Digital Holographic Microscopy using the Graphics Processing Unit", Optics Express, vol. 16, no. 16, pp. 11776–11781, 2008 July.

[14] S. Yang., X. Zhije., J. Xiangqian and J. Pickering, "Discrete Wavelet Transform on Consumer-Level Graphics Processing Unit", in the proceeding of the Computing and Engineering Researchers' Conference, University of Huddersfield, pp.40-47, November 2008.

[15] Röber, N., Spindler, M., and Masuch, M., "Waveguide-Based Room Acoustics through Graphics Hardware", Proc. of the ICMC, 2006 November.

[16] Bradley, C., and Gaster, B.R., "Exploiting Loop-Level Parallelism for SIMD Arrays using OpenMP", in the Proceedings of the 3rd International Workshop on OpenMP: A Practical Programming Model for the Multi-Core Era, pp.89-100, 2007.

[17] Yamamoto, Y., Fukaya, T., Uneyama, T., Takata, M., Kimura, K., Iwasaki, M., and Nakamura Y., "Accelerating the Singular Value Decomposition of Rectangular Matrices with the CSX600 and the Intergrable SVD", Parallel Computing Technologies, 2007.

[18] Bradford, R., Dobson, R., and Ffitch, J., "The Sliding Phase Vocoder", In the Proceedings of the 2007 International Computer Music Conference, S. O. Ltd, vol. 2, ICMA, pp. 449-452, 2007 August.

[19] Zhe, F., Qui, F., Kaufman, A., and Yoakum-Stover, S., "GPU Cluster for High Performance Computing", in the proceedings of the ACM/IEEE SC2004 Conference, pp. 47, 2004 November.

[20] Shirts, M., and Pande, V.S., "Screen Savers of the World Unite", Science, vol. 290, no. 5498, pp. 1903-1904, 2000 December.

[21] folding@home current user statistics http://folding.stanford.edu/English/Stats, (last accessed 2009 March)

[22] Göddeke, D., Strzodka, R., and Turek, S., "Accelerating Double Precision FEM Simulations using GPUs", in the Proceedings of ASIM 2005 – 18th Symposium on Simulation Techniques", 2005 September.

[23] Gibson, I.S., Howard, D.M., and Tyrrell, A.M., "Real-time Singing Synthesis using a Parallel Processing System", in the proceeding of the IEE Colloquium on Audio and Music Technology: The Challenge of Creative DSP. 1998.

[24] Moore, J.D., and Wakefield, J.P., "The Design of Improved First Order Ambisonic Decoders by the Application of Range-Removal and Importance in a Heuristic Search Algorithm", in the 31st AES International Conference, London, UK, 2007 June.

[25] Wiggins, B., Paterson-Stephens, I., Lowndes, V., and Berry, S., "The Design and Optimisation of Surround Sound Decoders using Heuristic Methods", presented at the Conference of the UK Simulation Society, Emmanuel College, Cambridge, UK, 2003.

[26] Craven, P.G., "Continuous Surround Panning for 5-Speaker Reproduction", in the 24th AES International Conference, Banff, Canada, 2003 June.

[27] The ClearSpeed Software Development Kit Introductory Programming Manual, http://support.clearspeed.com/resources/documentation/sdk_preliminary_programming.pdf, 2007 July.