Ambisonic audio system optimization using a HPC cluster

Moore, David; Mair, Quentin; Wakefield, Jonathan

Publication date:
2011

Document Version
Author accepted manuscript

AMBISONIC AUDIO SYSTEM OPTIMIZATION USING AN HPC CLUSTER

David Moore
School of Engineering and Computing
Glasgow Caledonian University
Cowcaddens Road
Glasgow, G4 0BA, UK
j.d.moore@gcu.ac.uk

Quentin Mair
School of Engineering and Computing
Glasgow Caledonian University
Cowcaddens Road
Glasgow, G4 0BA, UK
q.mair@gcu.ac.uk

Jonathan Wakefield
University of Huddersfield
School of Computing and Engineering
Queensgate
Huddersfield, HD1 3BU, UK
j.p.wakefield@hud.ac.uk

ABSTRACT

This paper investigates the use of the Glasgow Caledonian University VOTER HPC cluster for optimizing Ambisonic surround sound decoders. The cluster was used to run an exhaustive search coded using MPI C/C++ in order to obtain an optimal set of Ambisonic decoder parameters for a fixed resolution. The execution time of running this problem on the cluster compares favorably to work carried out on other hardware and shows that a significant increase in time-to-solution is achievable.

1 INTRODUCTION

High Performance Computing (HPC) technology is becoming more readily available. HPC products can now be used in conjunction with desktop computers (e.g. NVIDIA graphics processing units). 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. In manufacturing, this increase in available processing power has led to reduced production cycles through increased efficiency (see for example Rolls-Royce 2011), whilst in academia researchers have been able to investigate new problems and reconsider problems previously disregarded as too computationally demanding. In this paper we investigate the use of HPC technology for Ambisonic audio system optimization.

Ambisonics is a surround sound technique built around perceptual models of localisation. The system is designed to take into account the fact that human hearing uses different mechanisms for sound localisation in different frequency ranges. This is one of the key advantages that Ambisonics holds over other techniques as it was designed with human perception in mind (Gerzon 1994). Ambisonics is not a new technology but in recent years it has generated growing interest due to its potential application in a wide number of application areas such as broadcasting and computer games (Baume and Churnside 2010). However, the design of Ambisonic systems for irregular speaker layouts (such as 5.1) is a non trivial task that can involve the use of computationally intensive search algorithms. The aim of this work is to expand upon previous work in this area by using a cluster to run a search algorithm for Ambisonic system optimization (Moore and Wakefield 2009).

The following section provides an overview of HPC and looks at some existing HPC technologies. Section 3 then provides background on Ambisonic systems. Finally, Section 4 details the application of the Glasgow Caledonian University HPC VOTER cluster for running an exhaustive search for the design of an Ambisonic system.

2 HPC TECHNOLOGIES

2.1 HPC Architectures

Flynn (1966) proposed a simple, but broad, classification of high performance computer architectures based on the number of instruction and data streams that can be processed simultaneously. These include Single Instruction Single Data (SISD), Single Instruction Multiple Data (SIMD), Multiple Instruction Single Data (MISD), and Multiple Instruction Multiple Data (MIMD) (illustrated in Figure 1). In the current era of HPC only SIMD and MIMD type 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 - even when there is lower processing power per core. Furthermore, advances in single-core processor speeds are slowing because of the physical limitations of chip manufacturing with present day materials so this performance gap is likely to increase (El-Rewinim and Abd-El-Barr 2005).

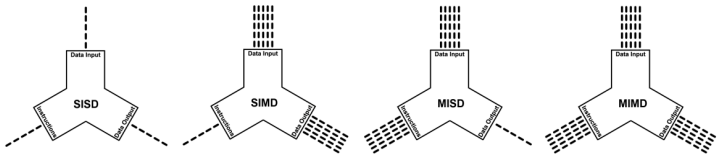

Figure 1: HPC Architectures defined by Flynn

HPC systems can also be classified by the manner 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 (see Figure 2a). Such systems usually employ high-speed, low latency, inter-connection (i.e. buses or switches) and are relatively easy to program. In contrast, a distributed memory system typically consists of multiple processors each with its own local memory space (see Figure 2b). The key issue in this type of system is the potential performance overhead introduced by inter-processor communication. Programming for this architecture is considered harder as the computer programmer has to ensure all communication operations do not introduce significant latency. 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) (El-Rewinim and Abd-El-Barr 2005).

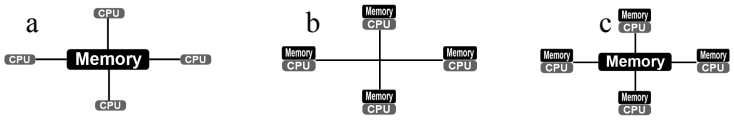

Figure 2: Typical HPC memory/processor interconnection

The above overview of HPC architectures is by no means exhaustive. Other hybrid systems exist such as *distributed-shared memory* which aim to combine the best of both architectures in that inter-processor communication is reduced and systems are easier to program (see Figure 2c).

2.2 Existing HPC Technologies

One of the current clear trends in computing is that expensive specially designed HPC hardware is being substituted by more cost-effective off the shelf solutions for use in conjunction with desktop computers. For example, Graphic Processing Units (GPUs) have shown a marked increase in their performance and capabilities and GPU companies are encouraging software development for their architectures by providing higher level programming models (e.g. CUDA by NVIDIA). In the literature there are a growing number of examples of HPC using GPUs (e.g. Benjamin et al 2010 and Southern et al 2010).

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). Several examples in the literature demonstrate the performance gains which can be achieved with accelerator hardware. For example, see Bradford et al (2007).

HPC applications are also undertaken using computer 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 (El-Rewinim and Abd-El-Barr 2005). 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. Clusters can be augmented with multiple GPUs or accelerators for increased capability (Fan et al 2004).

3 AMBISONICS

Ambisonic audio systems employ a flexible and efficient encoding/decoding model where it is possible to mix 2D or 3D sound without *a priori* knowledge of the geometry of the loudspeaker array. Once sound is encoded in Ambisonic form, it can later be decoded and played back over potentially any loudspeaker arrangement without change to the producers original mix intentions. This is an attractive feature especially when there is a growing demand for surround sound media to be shared between different playback devices (e.g. TV, games consoles) and different venues. An overview of the encoding stage of the system follows, then in section 3.2 the decoding stage will be discussed with reference to the optimization problem to be undertaken.

3.1 Encoding

Sound can be encoded to Ambisonic format using the commercially available SoundField microphone. Alternatively, sound can be encoded into 2D Ambisonic format using the following equations:

$$\begin{aligned} W &= S.\frac{\sqrt{2}}{2} \\ X &= S.\cos\theta \\ Y &= S.\sin\theta \end{aligned} \tag{1}$$

with W, X and Y representing the encoded Ambisonic signals for the horizontal plane, S the audio signal and the angle θ denoting the azimuth of the sound source. It should be noted that the additional component for encoding height (Z) has not been included here and the above equation is for encoding 'first-order' Ambisonics only. Higher order Ambisonics includes additional components resulting in greater spatial sampling accuracy (Daniel and Moreau 2004).

3.2 Decoding

To playback Ambisonic audio, a re-composition of encoded audio components is made that takes into account the location of each loudspeaker. For example, for first-order horizontal Ambisonics the output of each loudspeaker is a weighted sum of the encoded audio:

$$S_i = \alpha_i W + \beta_i X + \gamma_i Y \tag{2}$$

where S_i is the gain of the ith loudspeaker, W, X, and Y are the encoded audio signals, and α_i, β_i, and γ_i are the decoder parameters for the ith loudspeaker. Decoder parameters for regular arrangements of loudspeakers (e.g. square, hexagon etc.) are straightforward to derive analytically by matrix inversion (Heller et al 2010). However, for irregular arrangements, such as those often found in the domestic environment, it becomes a more complicated matter. A non-linear system of equations needs to be solved in order to produce a set of decoder parameters with good localisation performance around the listener. An alternative method of deriving decoder parameters for irregular layouts is to use a search algorithm to find a suitable set of decoder parameters which best fit design objectives specified in a fitness function. In previous work a heuristic search algorithm known as the Tabu Search has been employed for finding a "good" set of decoder parameters for the standard ITU 5-speaker layout (Moore and Wakefield 2007). This method was employed because searching exhaustively for the best set of decoder parameters using modern desktop computer processing power is not feasible. In this work, however, we run an exhaustive search on an HPC cluster.

4 AMBISONIC DECODER OPTIMIZATION

4.1 Cluster Configuration

The Glasgow Caledonian University VOTER HPC cluster is a large 64-bit Intel based computer system purchased from Viglen. It is partitioned into two equal but logically separate sub-clusters, one Windows based, the other based on Linux Rocks. In this paper we run our case study on the Windows sub-cluster which is based on half the computing resource available. Each sub-cluster provides a "MIMD with distributed memory" style architecture. The Windows sub-cluster consists of one head node and 6 identical compute nodes i.e. it is a tightly-coupled cluster. The functions of the head node are software development and job control (although software development could be carried out anywhere and the binaries copied across). Jobs are submitted to the compute nodes using a job scheduler, along with any parameters e.g. number of cores required and which compute node(s) to use.

Each of the compute nodes contains two Intel Xeon E5410 quad core processors each on a separate motherboard i.e. 8 cores per node, giving a total of 48 cores maximum available to a job. Each Intel processor runs at 2.33GHz with 12MB cache and 1333MHz FSB. Each motherboard contains 8GB RAM. Internally the nodes form their own private gigabit Ethernet based subnet. The head node is also connected to the university's internal research network to allow access via Windows Remote Desktop etc. Microsoft Windows 2003 Server HPC is currently installed together with Visual Studio 2008. A Windows shared folder allows binaries and data to be accessed by all nodes.

In recent years the implementation of scientific parallel processing applications on clusters and supercomputers has been largely achieved using the Message Passing Interface (MPI) (Walker and Dongarra 1996). MPI is a de facto standard with implementations in many mainstream programming languages and operating systems. Typically an application is split into separate parallel processes; each process is assigned to a single core. MPI offers an extensive library to support synchronous and asynchronous point-to-point message passing, collective communication e.g. broadcast, rendezvous and on the wire data formats.

A large body in the literature exists showing how to implement standard numerical analysis algorithms with MPI. It is necessary to define an algorithm which allows the problem to split into parallel parts each running on a separate processor. As the memory is distributed it is also necessary to consider using point-to-point message passing or collective communication mechanisms to transfer interim results or send processing requests to worker processes. It is also possible to identify processor id numbers; this facilitates the equal subdivision of a parameter space between multiple cores which is useful, for example, to facilitate an exhaustive search. MPI language bindings are available for C/C++ with third party bind-

ings including .NET and Java. On the VOTER HPC cluster runtime and development libraries for Microsoft MPI C/C++ are installed.

4.2 Exhaustive Search Implementation

Running an exhaustive search is guaranteed to locate the best solution in the search domain for all solutions evaluated. However, it can be time consuming. The number of potential solutions to evaluate in an exhaustive search is dependent on the number of parameters and also the parameter resolution. In this work the aim was to derive a first order Ambisonic decoder for the standard ITU 5-speaker layout which requires 8 decoder parameters; each parameter is constrained to the range [0, 1]. The resolution that solutions can be evaluated at is ultimately determined by the time of execution for the fitness function (this is the performance bottleneck). The time of execution of the fitness function combined with the processing speed of the system and number of solutions evaluated determines the total time to complete the search.

The exhaustive search implementation is coded in MPI C/C++ and is an adaptation of code taken from a parallel application solving the logic circuit satisfy problem by exhaustive search (Burkardt 2008). This approach allows equal parallelization of the algorithm across all available cores (in our case 48 cores) i.e. for a given *c* cores, *d* dimensions (parameters) and *r* resolution we divide the processing space equally amongst *c* processes. The solution is "embarrassingly parallelized" i.e. each of the parallel processes runs to completion without any need for interim communication with any other process. The fitness function is the same as that used in earlier work which compared performance on a single core Intel Xeon PC with performance on a ClearSpeed accelerator (Moore and Wakefield 2009). We collect the minimum fitness value identified by each process in a file on the Windows share and then identify the overall minimum fitness value and its corresponding parameter values.

4.3 Results

As shown in Table 2, the application was run with various resolution sizes in order to assess any improvements in execution times over earlier work using different architectures (Moore and Wakefield 2009). We have also interpolated times for lower resolution values not currently feasible. Note that the lower the solution fitness minimum, the better the quality of solution. All implementations gave the same solution giving us confidence that the implementations on different hardware are consistent.

Resolution	Solution Fitness	Time to complete
0.2	199.2100	137 secs
		4 secs
		2.3 secs
0.1	171.6900	290 mins
		8 mins
		4 mins 45s
0.05	153.3589	35 days (estimate)
		25 hours
		14 hours
0.033	-	**13 days (estimate)**
0.02	-	**705 days (estimate)**
0.01	-	**400 years (estimate)**

Table 1: Times and fitness values for the reference, ClearSpeed, and VOTER Cluster searches

The dark shaded cells indicate the times for a reference version which was a search run on a PC with an Intel Xeon 2.4GHz processor. The light shaded cells show the times for searches undertaken using two ClearSpeed x620 accelerators. Finally, the white cells with bold text show times for the VOTER Cluster. The results show an almost 2 fold increase in performance over the previous ClearSpeed approach and an average 65 fold increase over the single core PC execution times. As the processes are CPU-bound i.e. use 100% core utilization with no I/O (until calculations are completed), and also run independently with-

out resource contention, it is unlikely that performance can be increased further on the cluster. We anticipate that although minor code optimization is still possible this will not substantially improve times.

CONCLUSIONS

HPC opens the door to new applications. It makes problems feasible that are currently infeasible to run on desktop computers and can allow computationally expensive algorithms to run within acceptable time constraints and sometimes in real-time. In this work we employed an HPC cluster for running an exhaustive search algorithm for Ambisonic decoder optimization. Running the search on the cluster enabled an optimal solution to be found within a shorter amount of time than in previous work. The authors anticipate a linear increase in time-to-solution with more cores as there is no contention. The results are encouraging bringing lower resolutions within in the realms of possibility. At the present time though a heuristic search still presents the best option in terms of producing decoder using this method. In the future we plan on implementing a version of the above search algorithm in Java MPI for performance comparison as well consider parallel implementations of heuristic search algorithms. In addition, we plan to investigate the use of GPUs for audio processing applications coded using CUDA.

REFERENCES

Baume, C. and A. Churnside 2010, Upping the Auntie: A Broadcaster's Take on Ambisonics, In *Proceedings of the 128th Audio Engineering Society Convention*, London.

Benjamin B., P. Virnau, T. Preis, 2010, Multi-GPU Accelerated Monte Carlo Simulations of the 2D Ising Model, *Computer Physics Communications* 181:1549-1556

Burkardt, J. 2008, Circuit Satisfiability using Message Passing Interface, <http://people.sc.fsu.edu/~jburkardt/> [accessed May, 2011].

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

Daniel, J. and S. Moreau, 2004, Further Study of Sound Field Coding with Higher Order Ambisonics. *In Proceedings of the 116th Audio Engineering Society Convention*, Berlin.

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

Fan, Z., Q. Feng., A. Kaufman and S. Yoakum-Stover, 2004, In *Proceedings of the 2004 ACM/IEEE Conference on Supercomputing*, Washington.

Flynn, M.J. 1966 "Very High Speed Computing Systems", In *Proceeding of the IEEE,* 54(12):1901-1909.

Gerzon, M. A. 1994. General Metatheory of Auditory Localization. In *Proceedings of the 92nd Audio Engineering Society Convention*, Vienna.

Heller, A. J., E. Benjamin, and R. Lee, 2010 Design of Ambisonic Decoders for Irregular Arrays of Loudspeakers by Non Linear Optimization, *In Proceedings of the 129th Audio Engineering Society Convention*, San Francisco.

Moore, J. D. and J. P. Wakefield 2007, The Design and Detailed Analysis of First Order Ambisonic Decoders for the ITU Layout, *In Proceedings of the 122nd Audio Engineering Society Convention*.

Moore, J. D. and J. P. Wakefield 2009, The Potential of High Performance Computing in Audio Engineering, *In Proceedings of the 126th Audio Engineering Society Convention*, Munich.

Rolls-Royce 2011, High Performance Computing, Available via <http://www.rolls-royce.com/technology_innovation/systems_tech/> [accessed May, 2011].

Southern, A., D. Murphy, G. Campos and P. Dias, 2010 Finite Difference Room Acoustic Modelling on a General Purpose Graphics Processing Unit, In *128th Audio Engineering Society Convention*.

Walker, D. W. and J. J. Dongarra 1996, MPI: a Standard Message Passing Interface, Supercomputer, 12(1): 56-68. ASFRA BV.