Astronomical Data Analysis Software and Systems XVII **7B.3**
ASP Conference Series, Vol. XXX, 2008
J. Lewis, R. Argyle, P. Bunclarck, D. Evans, and E. Gonzales-Solares, eds.

GPUs for data processing in the MWA

S. Ord, L. Greenhill, R. Wayth, D. Mitchell

Harvard-Smithsonian Center for Astrophysics, Cambridge, MA, USA

K. Dale, H, Pfister, R. G. Edgar

Harvard University, Cambridge, MA, USA

Abstract. The MWA is a next-generation radio interferometer under construction in remote Western Australia. The data rate from the correlator makes storing the raw data infeasible, so the data must be processed in real-time. The processing task is of order 10 TFLOPs^{-1}. The remote location of the MWA limits the power that can be allocated to computing. We describe the design and implementation of elements of the MWA real-time data processing system which leverage the computing abilities of modern graphics processing units (GPUs). The matrix algebra and texture mapping capabilities of GPUs are well suited to the majority of tasks involved in real-time calibration and imaging. Considerable performance advantages over a conventional CPU-based reference implementation are obtained.

1. Introduction

The Murchison Wide-Field Array (Lonsdale et al. 2007) is a 512 element, low frequency, radio interferometer, currently under construction in Western Australia. The instrument has a number of ambitious science goals, grouped under four main science packages, the detection of the Epoch of Re-Ionization; solar, heliospheric and ionospheric science; a systematic survey for radio transients; and the Galactic and extra-galactic science package, which is a large umbrella organization of science interests including the interstellar medium, large area surveys, pulsars and the Galactic magnetic field. The instrument is also novel in that the intention is to process and calibrate all observations in real–time, as the raw data rate is too large to capture and process offline.

2. The Real–Time System

An FPGA[1] correlator will cross–correlate the signals received by the 512 antennas, its output being the correlations of 130,000 baseline pairs, each consisting of 768 frequency channels with 4 polarizations. This data stream is then integrated, calibrated and imaged by the real–time system (RTS).

[1] Field Programmable Gate Array

Calibration in this sense refers to calculating the complex gain of each antenna element and the time dependent vector field that describes refractive source position shifts due to the ionosphere. This process involves the observation of catalogue radio sources and the performance of a large linear least squares minimization in order to obtain the best estimation of the antenna gains and ionospheric offsets (Mitchell et al. 2008). Imaging refers to the construction, from the input data and calibration information, via a Fourier transform, of four images in the Stokes parameters, I, Q, U and V in a form that can be readily aggregated in frequency and time.

2.1. RTS Tasks

The calibration tasks are applied to a data-set which can be thought of as a Fourier transform of the sky brightness distribution. These data are manipulated to measure and remove the contribution from catalogue radio sources in decreasing order of predicted flux. Each measurement being used to constrain the complex gains of the antenna elements, this is an iterative process. The data-set then undergoes a convolution that interpolates the samples onto a regular grid to permit the operation of the Fast Fourier Transform (FFT).

In the case of the MWA the subsequent FFT operation results in images that are measurements of instrumental polarization in a coordinate system that is a slant-orthographic projection of the celestial sphere. Such images cannot be directly combined (e.g. co-added) as both the projection and the instrumental polarization change as a function of time (Cornwell and Perley 1992). The RTS takes the novel approach of transforming these wide-field images into a integrable representations of the polarization state of the sky by resampling the images onto an all-sky pixelisation. Firstly the measured instrumental polarization is converted into a set of Stokes parameters by the application of a 4x4 matrix transformation. This matrix is a function of time and position on the sky relative to the instrument and must be calculated by the RTS for each pixel. The slant-orthographic projection is converted into HEALPIX (Górski et al 2005) via a flux redistribution algorithm requiring the calculation of input and output polygonal overlaps. The pixelisation can then be imaged via the HPX projection (Calabretta and Roukema 2007) with no further interpolation.

Simulations and code development has indicated that the compute budget for this pipeline is approximately 10 TFLOPs^{-1}. The imaging tasks are by far the most computationally intensive operations. This compute capability will be provided, on site, by the real–time computer (RTC)

3. The RTC

A 3.2 GHz Harpertown CPU from Intel can provide approximately 100 GFLOPs^{-1}. Which indicates that 100 are required to meet the RTC processing requirements. With 100 compute-nodes and assuming a conservative 300W per compute-node of power consumption results in an RTC power requirement of 30kW, not including active cooling. The RTC has to operate in the desert under very strict power limitations, the current power budget being 20kW, which indicates that the FLOP/Watt ratio of even the most recent CPU is not sufficient to perform the task. In contrast a single 2008 NVIDIA GT200 series GPU can provide

800 GFLOPs^{-1}. Even if the power consumption per node is raised to 600 W, 32 GPU nodes can be powered by this budget providing 20 TFLOPs^{-1} of theoretical performance.

3.1. The GPU

One of the fundamental problems in High Performance Computing (HPC) is that RAM access speeds have not kept pace with improvements in CPU performance. A modern CPU can operate on two floating point numbers per clock cycle, but fetching those two numbers from memory takes hundreds of clock cycles. This leads to what has been termed "data starvation". CPU development has attempted to overcome this problem using innovative technologies including; large caches, superscalar architecture and branch prediction.

GPU developers have taken another approach to resolving data starvation, they have devoted GPU transistors to extra execution units, each working on a single thread of execution. These threads perform the same operation on different data elements as per the SIMD (single instruction multiple data) paradigm. There are also many more threads queued for execution than are running at any one time. The goal is that whenever a thread is waiting on memory access it gets swapped out for another which has data ready for use. GPU hardware handles all the thread scheduling transparently and a modern GPU can have 100s more execution units than a modern CPU. The difficulty in utilizing this compute capability lies in providing the GPU with enough threads to hide the memory latency.

3.2. Application of GPUs to the RTS

Applications that benefit most from a GPU implementation are ideally massively parallel, high arithmetic intensity operations. As moving data from the host memory to device memory is a time-consuming operation, considerable effort was directed at ensuring the entire RTS pipeline could be implemented on the GPU. Even though some operations are not optimally suited, the benefit in avoiding unnecessary host to device memory transfers was significant.

The GPU code differs from the CPU code due to the particular requirements of GPU programming. GPUs are sensitive to memory access patterns, as a result the porting involved a careful determination of optimal memory structures and even some algorithmic changes. In addition some operations are simply inadvisable on the GPU, for example the NVIDIA CUDA API lacks an atomic floating point accumulate operation, as a result "scatter" type algorithms, where multiple threads may write to the same memory location are not advisable and those that exist in the CPU code have been replaced by "gather" type algorithms, where a thread collects the values for a particular memory location. Redundant on-device memory operations were minimized by ensuring that thread memory requests were coalesced by the GPU; if threads access consecutive memory locations, the GPU performs all the memory access operations simultaneously. GPU threads are also very "light-weight", and fine grained parallelism can therefore produce remarkable performance benefits, the hardware thread manager is so efficient that it was worth considering threads that simply add two numbers together.

The majority of the RTS pipeline has been ported to the GPU and the comparative timings of those elements are presented in Table 1. The image resampling step is yet to be implemented, but this multi-step pipeline already successfully demonstrates the feasibility of a GPU based calibration and imaging system. Experiments indicate a factor of ten improvement over the CPU application.

Table 1. Relative Performance of CPU and GPU Tasks.

Task	CPU[a] (ms)	GPU[b](ms)
CML	250	22
Gridding	480	36
FFT	326	40
Convolution Correction	22.6	1.4
Stokes Conversion	42.4	4
Total	1121	103.4

Note. — (a) INTEL Core2 Quad 2.66GHz (Q9450), ASUS P5E3 Deluxe Motherboard, 4 GiB DDR3 RAM. (b) NVIDIA C1060

4. Summary

GPUs can enable science that otherwise would be impossible due to monetary or power constraints. We have presented a general overview of a complex GPU application to calibrate and image data from the MWA, which has been ported from a CPU-based application. Although not all elements of the RTS have been implemented on the GPU a calibration and imaging pipeline has been demonstrated to work efficiently, and provides considerable performance improvements over a CPU implementation.

References

Calabretta M., & Roukema B. F., 2007, MNRAS, 381, 865

Cornwell T. J., & Perley R. A. 1992, Proc. SPIE, 1351, 706

Górski K. M, Hivon E., Banday A. J., Wandelt B. D., Hansen F. K., Reinecke M., Bartelmann M.., 2005, ApJ, 622, 759

Mitchell D., Greenhill L.J., Wayth R.B., Sault R.J., Lonsdale C.J., Cappallo R.J., Morales M.F., Ord S.M. 2008, IEEE Journal of STSP, 2, 5