A GPU based solution for distributed FX correlation

Richard Dodson[*]
ICRAR, UWA
E-mail: richard.dodson@uwa.edu.au

Chris Phillips
ATNF, CSIRO
E-mail: chris.phillips@csiro.au

Dick Ferris
ATNF, CSIRO
E-mail: Dick.Ferris@csiro.au

The steps which form part of the VLBI correlation chain (digitisation, delay correction, channelisation and integration) can be reordered to reduce the associated losses. This involves processing as much as possible at the antenna site before passing through the rate limiting step of recording to disk, or streaming over network connections. We present our plans for such an implementation, and the gains that this approach will produce. We present the results from our first steps towards this goal, for which we are experimenting with a Graphics Processing Unit (GPU) to perform the antenna based corrections. This step involves using the GPUs to provide an autocorrelator to run in parallel to the VLBI data recorder.

Science and Technology of Long Baseline Real-Time Interferometry:
The 8th International e-VLBI Workshop, EXPReS09
June 22 - 26 2009
Madrid, Spain

[*]Speaker.

1. Introduction

We present our ideas for a Graphics Processing Unit (GPU)-based real time pre-processor (channeliser) which will form part of a distributed FX correlator. We show our results from the early investigations. This project is *not* an attempt to utilise GPUs in a software correlator, which we could expect to be an important but incremental advance in the performance of such VLBI correlator solutions. Instead we aim to reorder the data processing chain in such a way as to deliver higher performance overall. The technology which will allow this is the combination of disk-based recorders and GPUs. Disk based recorders are now common among VLBI networks, for example the Australian data recording system as described in these proceedings [1].

The traditional data processing chain is correlator dependent but generally has the following processes:
• Wide bandwidth baseband analogue output from the first stage amplifiers and down-conversion.
• Filtering to narrow bandwidth and digitization to 2 bits per sample.
• Recording on or transferring via data-rate limited media: disk or internet connection.

And at the correlator:
• Channelisation
• Delay compensation
• Fringe rotation
• Cross correlation
• Integration

Potential losses [2] associated with these steps are: 12% loss for perfect Gaussian noise inputs 2-bit sampled at the correct transition voltages and 8% fringe rotation loss if a 3-level fringe rotation is used. There is a discrete delay step loss if the samples are not aligned on the timing boundary. In the best case, when the delays are calculated for the centre of the band rather than the edge, this is 3%. These losses give a net 'efficiency' of 0.78 or a 'correlation loss' of 22%; a net equivilent increase of Tsys of 27%.

An alternative data processing chain is;
• Wide bandwidth baseband analogue output from the first stage amplifiers and down-conversion.
• Filtering to narrow bandwidth, digitization to 8 bits per sample and reading into the computer recording system.

And on the GPU
• Channelisation
• Delay compensation
• Fringe rotation
• RFI mitigation
• Reduction to 2 bits per sample, or better, encoding.
• Recording on or transferring via data-rate limited media: disk or internet connection.

And at the correlator
• Cross correlation
• Integration

By maximising the processing done before the quantisation we can effectively eliminate all of the losses except the 2-bit quantisation loss. Also, perhaps more importantly, we can remove the broad spectrum contamination from any Radio Frequency Interference (RFI) before the quantisation. Receiver noise containing RFI does not have a Gaussian distribution of voltages, therefore is poorly quantised by the simplistic 2-bit 4-level encoding, producing contamination in nearby frequency channels. Strong Maser lines have similar issues. At a lesser level the bandpass edges, where the gains roll off, are similarly poorly handed by this encoding approach because the transition voltages are no longer optimal. These problems can be overcome by preprocessing the data before quantisation as having a high number of bits per sample avoids these issues. The CABB and WIDAR correlators also form their channels with a high number of bits per sample, but do not reduce this for transfer to a distant correlator, as they are not designed for VLBI usage.

To expand on the steps and their gains:

For any FX correlator the first step is to convert from time based samples, sampled at Δt, to a frequency based representation. This is conventionally done with an FFT (but it could be done with a polyphase filter bank as for the CABB correlator). The FFT can take complex or real samples to produce a channelised output, where the number of channels (N) equals the number of complex inputs or half the number of real inputs and the channel width is the reciprocal of N$\times\Delta$t.

Once the data is channelised it is trivial to apply phase and delay corrections as provided by some a-priori model, normally via the defacto standard, CALC [3]. In the Fourier domain we can apply fractional delay corrections as we are not constrained to the sample time boundaries. Therefore we no longer suffer from the discrete delay losses. Furthermore as all inputs have high numbers of bits per sample the fringe rotation can also be done at full resolution, removing the associated losses from that operation.

Questions have been raised as to whether the fringe rotation correction, which introduces a correlated correction across the bandpass, implies that the 'random noise' assumption required for this encoding is broken. We counter this argument as follows. The input is white noise, each FFT accepts N complex (or 2N real) samples of Δt and the FFT forms N channels of frequency width 1/(N Δt). The samples in these channels are independent, as the input is white noise. The fringe rotation applies a phase shift per channel, which is coherent across the bandpass, but as the input numbers are random the output is also random. Finally as the samples are random, then the spectrum is white noise, and 2-bit/4-level encoding will work correctly. The thresholding level is set independently for each channel, so bandpass gain rolloff and interference peaks are correctly handled.

The approach is only really feasible in the age of eVLBI as previous suggestions for antenna based preprocessing, in particular LO Doppler shift corrections, had to be abandoned for reliability reasons. Now VLBI systems have end-to-end continuous reliability checking we can transfer significant portions of the correlation to the antennas. This has the advantage of making much better use of the distributed processing power on-site, and has the potential advantage of reducing the correlator power requirements. This can be significant in remote array sites (such as for the Murchison Radio Observatory, where ASKAP is being built).

2. Progress towards the distributed FX correlator

2.1 Target hardware: GPUs

GPUs are currently co-processor units, which sit on the PCI-bus, and are designed to perform the highly parallel calculations required for the high definition graphics called for in modern computer applications; mostly games. They can now be programmed reasonably easily, with languages such as CUDA or OpenCL, to tackle generalised high performance computing problems. They have a high number of independent threads, so are suitable for parallel processing problems, but the PCI bus limits the IO on and off the cards. Furthermore there are more intricate issues with memory on the card as there are several layers of resources split between the different GPU threads. It is possible to make what is nominally a highly parallel problem run much slower than it would on a CPU by poor implementation of memory usage. We are developing our system on a GPU as it seems to offer a very cheap solution to the processing requirements for this project, and gives us the opportunity to learn about these problems.

2.2 Progress so far

We decided to use CUDA, as the language is very 'C-like' which will minimise the development time. OpenCL was not yet released when we started the project. This limits us to NVIDIA cards. Our demonstration is going to be with the LBA data recording system, referred to above. This system has limited space for additional cards, and so we are limited to a single slot. The most powerful NVIDIA GPU we could obtain which has a single slot profile is the 9800GT, a GeForce series 9, costing ∼\$300. Our model had 0.5 GB of memory.

The PCI IO rate is nominally 4GB/s. Assuming that we work at 8 bits per (real) sample we should therefore be able to handle 2 GHz of bandwidth. However we also need to be able to FFT this data, process it and stream it back off the card. Fortunately CUDA can now support asynchronous reads and writes, so we could stream (float; i.e. 32 bit) data onto (and off) the card whilst processing the current data block. With this program structure we tested three NVIDIA cards: our 9800GT, a GTX 285 with 1 GB, and a FX-1700 Quadro.

We performed the simple experiment of transferring a block of float-sized data, transforming it, and transferring the output back. These operations were in different streams, and the FFT was found to be the rate limiting step. We experimented with different FFT sizes to find the most efficient mode. Figure 1 reproduces these results. We have demonstrated that the 9800GT card can sustain datarates the equivalent of 120 MHz (i.e. in this case 240 M (real) samples) when loading and channelising our target datasets. We also note that the turn over in performance is when the FFT requirements are greater than on-card memory, and for the GTX 285 with double the capacity the performance continues to rise.

Following the demonstration that the card was capable of achieving the goals we had set ourselves we set to providing a real-life, if not real-time, demonstration. We took some of the data recorded at 256 Mb/s from the previous VLBI session and adapted the existing auto-correlator LBA program (fauto) to run on the GPU (gauto). This was a far from optimum approach as the program structure was not suited to running in a streamed mode, nor performing the 2 bit to float decoding on the card, both of which can be expected to provide significant speed ups. Nevertheless we were able to process the data and form autocorrelations using the GPU. In the process we discovered how

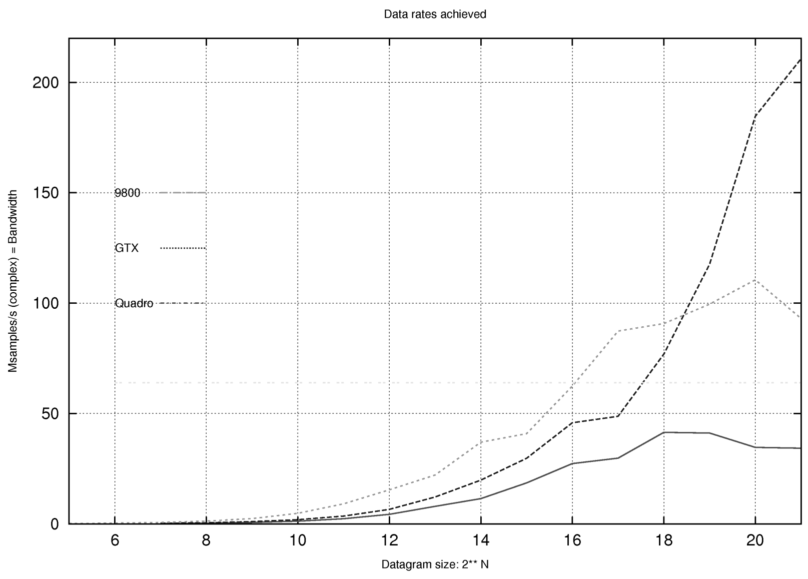
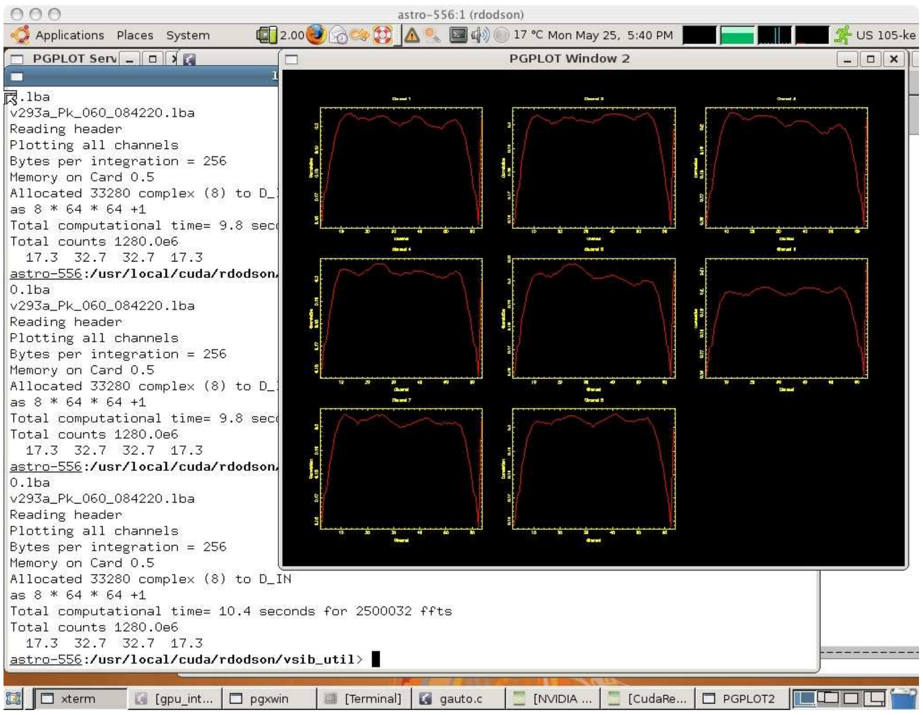


Figure 1: Data Rates, in equivalent bandwidth, with increasing buffersize. The most efficient processing to for the largest transfers.



```
.lba
v293a_Pk_060_084220.lba
Reading header
Plotting all channels
Bytes per integration = 256
Memory on Card 0.5
Allocated 33280 complex (8) to D_
as 8 * 64 * 64 +1
Total computational time= 9.8 sec
Total counts 1280.0e6
  17.3  32.7  32.7  17.3
astro-556:/usr/local/cuda/rdodson
0.lba
v293a_Pk_060_084220.lba
Reading header
Plotting all channels
Bytes per integration = 256
Memory on Card 0.5
Allocated 33280 complex (8) to D_
as 8 * 64 * 64 +1
Total computational time= 9.8 sec
Total counts 1280.0e6
  17.3  32.7  32.7  17.3
astro-556:/usr/local/cuda/rdodson
0.lba
v293a_Pk_060_084220.lba
Reading header
Plotting all channels
Bytes per integration = 256
Memory on Card 0.5
Allocated 33280 complex (8) to D_IN
as 8 * 64 * 64 +1
Total computational time= 10.4 seconds for 2500032 ffts
Total counts 1280.0e6
  17.3  32.7  32.7  17.3
astro-556:/usr/local/cuda/rdodson/vsib_util>
```

Figure 2: A screen-shot showing of the autocorrelations formed on the GPU. The datafile was recorded for 10 seconds at 256 Mb/s. The autocorrelations are processed, on average, within 10 seconds.

sensitive the performance was to efficient memory management. Figure 2 shows the successful autocorrelation of 10 second data files recorded from Tidbinbilla for experiment V293A.

2.3 Steps to be done

We have successfully applied for NVIDIA support to extend this project, and are advertising an internship to work on the delay compensation, RFI mitigation, quantisation and off-card streaming steps.

3. Conclusions

We propose a reordering of the VLBI correlating processing chain to provide significant improvements in efficiency, and transfer the bulk of the processing to the antenna site. We have demonstrated that low cost GPUs can provide the computing power required. We have written a simple autocorrelator to run in parallel with the data recording on the LBA to provide us with experience towards our goal and a realtime auto-correlator for these antennas. We plan, with the help of NVIDIA, to develop this alternative correlator approach. We look forward to reporting on the outcomes of these investigations.

References

[1] Phillips, C. et al., *LBADR: The LBA Data Recorder*, in proceedings of*Science and Technology of Long Baseline Real-Time Interferometry*, (2009), PoS(EXPReS09) 099

[2] Thompson, A.R., Moran, J., Swenson, G., *Interferometery and Synthesis in Radio Astronomy*, Wiley, (2001).

[3] GSFC, NASA, *Mk-5 VLBI Analysis Software Calc/Solve*,