VLBI DAS – What is needed

• Convert analog signal to digital
• Format data including time code

In the age of disk recording and software correlation anything else is “optional”
VLBI DAS – What is desirable

- Channelization
  - Polyphase filterbank
- Re-quantize data
  - Reduced data rate
- Tsys extraction
- Digital linear to circular conversion
- Plus lots more – phasecal extraction etc
“Traditional DAS”

• High speed (multi-bit) sampler and FPGA for processing
  • RDBE, DBBC etc
• FPGA high compute capability, low power usage and compact
• Ideal for VLBI usage
• But…
GPU Processing

• GPUs are extremely fast and cheap
• Nvidia GTX 690
  • $1000
  • 6 Tflops (theoretical)
GPU Arithmetic Performance
GPU Memory Bandwidth

Processor Memory Bandwidth Trends

Bandwidth (bytes/sec)

8800 GTX, 9800 GTX, GTX 280, GTX 480, GTX 580, GTX 680, Core i7-3960X, Core i7-975, Core i7-990X
CUDA

• A parallel computing architecture developed by NVIDIA.
• Extensions to C(++) to allow massive parallelization running on a GPU
  • Run thousands of threads in parallel
• Very simple to program
__global__ void multiply_kernel(float *a, 
   const float *b, 
   const float *c) {

   const size_t i = blockDim.x * blockIdx.x 
                     + threadIdx.x;

   a[i] = b[i] * c[i];
}
GPU DAS Experiment

• Goal 1 GHz dual pol
  • 8 Gbps @ 2bits
• Assume dual pol sampled at 8 bit real
• Channelize to 16 MHz
• Linear to Circular conversion
• (Tsys)
• Convert to 2 bit
• Pack (interleave)
Steps Required

• Copy data to GPU
• Convert 8 bit to float
• De-interleave data
• FFT (eventually polyphase filterbank)
• Complex gain correction
• Linear to circular
• Calculate RMS
• Quantization and interleave
• Testing on GTX 590 ($500)
I/O Bandwidth

- 1 GHz real samples @8 bits, dual pol $\Rightarrow$ 32 Gbps
- bandwidthTest
  - Host $\Rightarrow$ GPU 47 Gbps
  - GPU $\Rightarrow$ Host 50 Gbps

- CUDA allows DMA transfer while processing
Convert to Float

• 142ms

```c
__global__ void unpack8bit_2chan_kernel(float *dest,
const int8_t *src, int N) {

const size_t i = blockDim.x * blockIdx.x + threadIdx.x;
const size_t j = i*2;

dest[i] = static_cast<float>(src[j]);
dest[i+N] = static_cast<float>(src[j+1]);
}
```
FFT

• 465 ms

cufftPlan1d(&plan, 32, CUFFT_R2C, batch);
cufftSetCompatibilityMode(plan,
    CUFFT_COMPATIBILITY_NATIVE);

cufftExecR2C(plan, in, out);
Linear to Circular

• 254 ms

```c
__global__ void linear2circular_kernel(cuFloatComplex *data,
                                        int nchan, int N, cuFloatComplex *gain) {

    int i = blockDim.x * blockIdx.x + threadIdx.x;
    int c = i % nchan;
    cuFloatComplex temp;

    data[i] = cuCmulf(data[i], gain[c]);
    data[i+N] = cuCmulf(data[i+N], gain[c+nchan]);

    temp = cuCsubf(data[i], data[i+N]);
    data[i+N] = cuCaddf(data[i], data[i+N]);
    data[i] = temp;
}
```
RMS

• “Reduction” so potential problematic
• Took SDK sample for mean calculation
• Assumed real sampled data

• 115 msec
Convert to 2bit and interleave

• Just use if/then/else block with 3 thresholds
• Specially coded 32 channel dual pol case

• 5.8 seconds!!!
Convert to 2bit and interleaves, try 2

- Just use if/then/else block with 3 thresholds
- Used simple dual pol case

- 247 msec
Summary

<table>
<thead>
<tr>
<th>Step</th>
<th>Time (msec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transfer to GPU</td>
<td>OK</td>
</tr>
<tr>
<td>Unpack and convert to float</td>
<td>142</td>
</tr>
<tr>
<td>FFT</td>
<td>465</td>
</tr>
<tr>
<td>RMS</td>
<td>115</td>
</tr>
<tr>
<td>Circular to Linear</td>
<td>253</td>
</tr>
<tr>
<td>Quantize and pack</td>
<td>247</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1222</strong></td>
</tr>
</tbody>
</table>

• But 2 GPU so OK
Polyphase filterbank?

• Tests using FFT channelization
• Prefer PFB
• Wisdom says PFB 1.5-2x the compute of FFT
• Assuming 2x time for FFT and dual GPU gives total computer time as 0.84 seconds

• 320msec to implement Tsys extraction if simple RMS calculation is not enough
# CPU Comparison?

<table>
<thead>
<tr>
<th></th>
<th>GTX 590</th>
<th>2x Intel E5-2609 @ 2.40GHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unpack and convert to float</td>
<td>142</td>
<td>315</td>
</tr>
<tr>
<td>FFT (PFB)</td>
<td>465 (930)</td>
<td>1469 (2939)</td>
</tr>
<tr>
<td>RMS</td>
<td>115</td>
<td>243</td>
</tr>
<tr>
<td>Circular to Linear</td>
<td>253</td>
<td>1196</td>
</tr>
<tr>
<td>Quantize and pack</td>
<td>247</td>
<td>479</td>
</tr>
<tr>
<td>Total (sec)</td>
<td>0.61 (0.84)</td>
<td>3.7 (5.2)</td>
</tr>
</tbody>
</table>
## GPU Comparison

<table>
<thead>
<tr>
<th></th>
<th>GTX 590</th>
<th>GTX690</th>
<th>GTX670*</th>
<th>GTX480</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unpack</td>
<td>142</td>
<td>156</td>
<td>148</td>
<td>136</td>
</tr>
<tr>
<td>FFT</td>
<td>465</td>
<td>420</td>
<td>407</td>
<td>450</td>
</tr>
<tr>
<td>RMS</td>
<td>115</td>
<td>111</td>
<td>174</td>
<td>110</td>
</tr>
<tr>
<td>Circular to Linear</td>
<td>253</td>
<td>220</td>
<td>213</td>
<td>241</td>
</tr>
<tr>
<td>Pack</td>
<td>247</td>
<td>123</td>
<td>353</td>
<td>260</td>
</tr>
<tr>
<td>Total (sec)</td>
<td>1.22</td>
<td>1.14</td>
<td>1.30</td>
<td>1.20</td>
</tr>
</tbody>
</table>

*Single GPU, rest are dual GPU
Conclusion

• Modern GPU should have enough compute power to cover most VLBA DAS requirements
• A lot of scope for speed improvements
  • Combine some stages together

• A working system would allow for a very generic backend
  • VLBI, spectral line, pulsar, fast transient detector...
Thank you

CSIRO Astronomy and Space Science

t  +61 2 9372 4608
E  Chris.Phillips@csiro.au
w  www.atnf.csiro.au