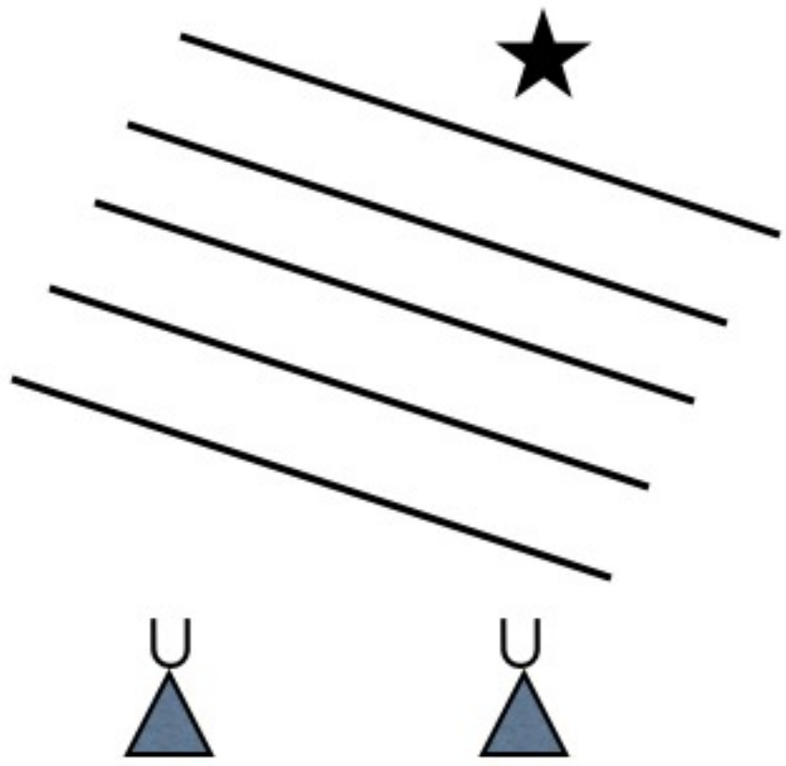
GPU Correlation

Jonathan Sievers

NB - not my day job.

Basics of Interferometers

- Many radio telescopes work together
- From a source, electric field is plane waves at earth.
- Radio telescopes measure electric field, then E from pairs of telescopes multiplied.
- Builds up measurement of Fourier transform of sky.
- Each pair of antennas measures one Fourier mode, called a *baseline*. Often measure two polarizations of E as well.



Correlation

- Have to multiply electric fields together - called correlation.
- In the past, most correlators were analog.
- Digital becoming cheaper, but serious computing work.
- Sensitivity of telescope proportional to bandwidth, so want as wide a band as possible.
- Working on GPU correlator for Giante Metre-wave Radio Telescope, in India.

Correlator room at GMRT

More Detail on Correlation

- Before electric field goes into correlator, gets downshifted to low frequency.
- Ideally, frequency observed is ~1/3 base frequency. So, telescope at 1.4 GHz would have 400 MHz bandwidth, 90 GHz would have 20-30 GHz, etc.
- Want correlator to keep sub-frequencies separate, so want to split up electric field into frequency channels. Do this with FFTs.
- After splitting into bands, then multiply the E fields for each pair of antennas for each frequency, and accumulate. Note n^2 scaling.
- Traditionally, bandwidth is cheap for analog correlators, channels are cheap for digital correlators.
- Correlators are usually *the* limiting factor for large radio arrays.

Some Numbers

- New GMRT correlator - desired 400 MHz bandwidth (=800 megasamples/second) for 30 dual-pol'n antennas, with RFI monitors, total of 64 inputs.
- New correlator would be major upgrade to GMRT, current bandwidth is 32 MHz.
- Need to FFT 800e6*64 = 5e10 numbers/second.
- have $n(n-1)$ baselines for n antennas, so for GMRT, have about 2000 baselines, for 1.6 trillion correlations per second. Large, but not insane. If doing as floats, electric fields are complex, so need 6.4e12 MADs/s.
- Good problem for GPUs - would need ~1000 CPU cores, vs. ~20 Fermi chips running efficiently. Other options include FPGAs.

FFT

- FFTs work very well on GPUs - e.g. got 90 million *64 FFT/s sec for ~1024 channels on GTX295.
- Would need a total of ~9 boards from splitting apart into channels.
- Will probably need to do a shuffle between FFT and correlation, not expected to be expensive.

Correlation w/CUBLAS

- Want to multiply E field for all antennas for lots of channels for a length of time.
- FFT'd E field is natively 3D - dimensions are antenna, frequency, and time.
- If data stored correctly, then can do loop over frequency of n_{ant} by n_t matrix multiplications into n_{ant} by n_{ant} matrix.
- Not working well - on current Tesla, get ~7e6 samples/s correlated for array, works out to <6e10 MADs/s (using cublasSsyrk), only 10% efficient. Even for larger problems, seems to saturate at 20% efficient.
- NVidia guys - any change this gets better? Difference between 13 boards to to correlations and 130 is large.
- Have tried writing my own float multiplier, where chunks of data brought into shared memory and multiplied, get similar speeds.

Precision, Schmision

- S/N per sample *very* low ~ 10^{-6} or so. Full float multiplication wasteful. Turns out 1-bit multiplication is ~65% efficient, 2-bit is ~85%, 3-bit is 96%, 4-bit 99%.
- Can get 4 operations for price of one at 3 bits, at expense of packing/unpacking operations:

```
  0000000000000000RRR00000000III
x 0000000000000000RRR00000000III
  ------------------------------------------
=0000RRRRRR0000XXXXXXX00000IIIIII
```

Can do 16 multiplications before parts overlap.

Balancing Act

- Now to do correlations, thread blocks bring in chunks of data, multiply, and accumulate. Chunk of data is a set of time samples and antennas for one frequency.
- Implementation works out to be a direct trade-off between register/shared usage and memory bandwidth.
- If work on all 64 antennas, data only leaves main memory once. But, now have 64^2*4=16kb arrays on chip.
- Say do 8x8 chunk, then each sample gets read 8 times. Doable, but getting close to memory bandwidth limits. But can run multiple blocks simultaneously.
- For 3-bit correlator, pack data, so 1 unsigned int has 10 time samples in it.

Performance

- NB - code very raw, but think time #'s are correct.
- Get something like 20-30 x10^6 correlations/s (vs. 7floating) for 64 inputs, right general ballpark given CUBLAS.
- But, if switch to packing/unpacking into registers but otherwise do same work, goes to 90x10^6 corr/s. Only problem is code can't produce correct answers that ways.
- Am I stuck with this if I used shared memory?

Current Implementation

- Each thread in 8x8 block owns 1 baseline.
- Loop until done:
  - thread *i,j* reads antenna *i* data for time chunk *j*
  - loop over *t* (1->10)
  - thread *i,j* unpacks t^{th} sample into shared memory. (5-6 ops)
  - thread i,j conjugates sample, makes redundant copy for loops
  - loop over *k* (1->8)
    - Thread *i,j* tmp data += shared(*i,k*)*shared(*j,k*) (8 ops)
  - Thread data (float) += tmp data (uint) -> (7 ops)

Last step only needs to be done every 16 mults, so a final version would have 7 -> 3.5 (effective)

Core Computational Routine

```
unsigned int myrealdata=real_data_global[my_load_off+my_ant1_off+threadIdx.y+i*BLOCKSIZE];
unsigned int myimdata=im_data_global[my_load_off+my_ant1_off+threadIdx.y+i*BLOCKSIZE];


for (int samp=0;samp<=32-NBIT;samp+=NBIT) {
  unsigned int accum=0;
  //this definitely needs checking
  data[mylocalind] =((myrealdata>>samp*NBIT)&MASK)+((myimdata>>samp*NBIT)&MASK)<<11;
  data_conj[mylocalind]=data[mylocalind]^XOR;  //this is the conjugate
  data[mylocalind+BLOCKSIZE*BLOCKSIZE]=data[mylocalind];
  data_conj[mylocalind+BLOCKSIZE*BLOCKSIZE]=data_conj[mylocalind];
  //must absolutely be fixed.
  for (int jj=threadIdx.x;jj<threadIdx.x+BLOCKSIZE;jj++) {
    //accum+=data[mylocalind]*data_conj[mylocalind];
    accum += data[threadIdx.x*BLOCKSIZE+jj]*data_conj[threadIdx.y*BLOCKSIZE+jj];
  }
  real_corr+= accum-((accum>>22)&BIG_MASK);
  im_corr+=(accum>>11)&BIG_MASK;
 }
}
```

Wrap-Up

- Correlating signals from antennas is at the heart of vast majority of radio astronomy today.
- Back-of-envelope should be able to make upgraded GMRT correlator from O(20-30) current GPUs.
- Current performance on floating point precision correlation is ~10% of theoretical.
- Doing comparably, perhaps a bit better, for 3-bit correlation (but less work, so faster).
- GMRT correlator relatively small. Example - ALMA (currently being built in Chile) will have ~100 inputs, 5 times bandwidth, 20 million solder points. Square Kilometer Array will have 10 times bandwidth, 2500 dishes.