1 Requirements

The goal of the project is to have the GPUs that perform the cross correlation (X engine) during the night-time data gathering also compress during the day, when the correlator is not in use. The number of GPUs required for the X engine is determined by the number of cross products, and therefore scales with N^2, where N is the number of antenna polarizations in the array (except on small array sizes, which may be bandwidth bound). The amount of data that our correlator produces, and which must then be compressed is given by the equation:

$$\frac{\frac{N_{antpol}(N_{antpol}+1)}{2} \cdot N_{\text{channels}} * 2 * 32\text{b}}{t_{\text{integration}}} \tag{1}$$

which also scales with N^2. Therefore, if the GPUs used as the X engine are also sufficient to compress the incoming data on current arrays, they should also be sufficient on larger arrays.

PAPER

The PAPER array has 128 dipole antennas, and therefore 256 antenna polarizations. With 1024 channels and an integration time of 10 seconds, this gives $\frac{\frac{256(256+1)}{2} \cdot 1024 \cdot 2 \cdot 32\text{b}}{10s} = 216\text{Mbps}$, which would have to be processed on 16 GPUs, which is 13.5Mbps/GPU.

HERA

The proposed HERA array would be similar to the PAPER array, but would have 1024 dipole antennas and 256 GPUs. 8 times as many antennas results in 64 times the amount of data, or $64 * 216\text{Mbps} = 1.4\text{Tbps} = 55\text{Mbps/GPU}$.

2 Current State

The most recent tests were done on 2 megapixel images in batches of 10 on two NVIDIA GPUs, the K20 TITAN, and the GeForce 690. The results were 2.12 seconds per image on the TITAN, and 3.7 seconds on 1 core of the dual core 690. This gives a result of 63Mbps/GPU for the TITAN and 36Mbps for the GeForce 690, meaning the GPUs are currently fast enough to keep up with the data rate of the X engine. However, windowing effects may cause the computation to require up to 3 times as many uses of the data, so either improving the speed further or mitigating these effects is necessary.

Testing with multiple threads used snb1, a machine with 2 GeForce 690 GTX cards, each with 2 cores. The profiler used in previous tests caused segfaults when used with threading, so a new baseline was needed. One thread took 5.3 seconds for one iteration, and 9.0 seconds for 2 iterations, consistent with the previously obtained 3.7 second runtime per iteration. However, this version did not properly parallelize. With a multithreaded version running on snb1 (78e6675701b2e41fbf734dea10f08153e74d21dc on github)on 2013/6/7, with one thread for each of the 4 cores, the CUDA and C portion of the code took 16 seconds for 4 iterations, again with a 2 Mpx image, of 67Mbps/GPU. However, the python portion of the code is less efficient and with that overhead, the code took 25 seconds, which translates to 43Mbps/GPU. There is little difference (1%) between using 2 GPUs on one 690, and 1 GPU on each of 2 690 cards. 8 iterations 55.28 seconds 2 Mpx 16 iterations 50.4 seconds 1 Mpx 6/24 32 iterations 49.5 seconds .5 Mpx 16 iterations 48.6 seconds 1 Mpx

GPU	thd (pl)	img size	t (s)	stop if div	iter	Mbps	git hash
1×CPU	1 (of 1)	1024 × 2048 (1)	68.5 (68.5)	T	864	2.0 (2.0)	e448e3
1×CPU	1 (of 6)	1024 × 2048 (6)	282. (47.)	F	1	0.48 (2.9)	e448e3
1×CPU	1 (of 6)	1024 × 2048 (6)	202. (34.)	T	864	0.65 (3.9)	e448e3
1×K20	1 (of 1)	1024 × 2048 (1)	2.12 (2.12)	T	864	63 (63)	a8306a
1×690	1 (of 2)	1024 × 2048 (1)		T			a8306a
0.5×690	1 (of 1)	1024 × 2048 (1)	3.70 (3.70)	T	864	36 (36)	a8306a
2×690	1 (of 4)	1024 × 2048 (4)	6.92 (1.73)	T	864	19.5 (78)	e448e3
2×690	1 (of 4)	1024 × 1024 (4)	3.16 (0.79)	T	864	21.5 (86)	e448e3
2×690	1 (of 4)	1024 × 512 (4)	1.56 (0.39)	T	864	21.5 (86)	e448e3
2×690	1 (of 4)	1024 × 1024 (4)	3.04 (0.76)	T	864	22.0 (88)	e448e3
2×690	1 (of 5)	1024 × 2048 (5)	6.00 (1.20)	T	864	2.0 (134)	36ebff
2×690	1 (of 4)	1024 × 2048 (4)		T			36ebff