Accelerating Cosmology Codes

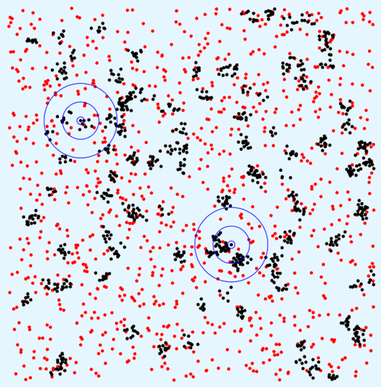
Volodymyr Kindratenko, Dylan Roeh, Guochun Shi (NCSA), *Robert Brunner* (Department of Astronomy), University of Illinois

Two-point Angular Correlation

Two-point angular correlation function (TPACF), ω(θ), is the frequency distribution of angular separations θ between celestial objects in the interval (θ, θ + Δθ)

- angular distance between two points is

$$\theta = \arccos(\ p \cdot q) = \arccos(\ x_p x_q + y_p y_q + z_p z_q)$$



Example: **Red (random data) are, on average, randomly distributed, black (observed data) are clustered**

- random points: **ω(θ)**=0
- observed points: **ω(θ)**>0

TPACF can vary as a function of angular distance, θ (blue circles)

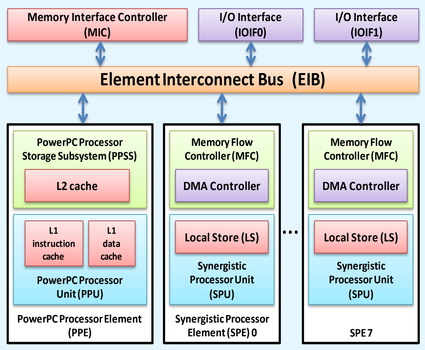
- random: **ω(θ)**=0 on all scales
- observed: **ω(θ)** is larger on smaller scales

TPACF is computed using modified Landy & Szalay estimator

$$\omega(\theta) = \frac{n_R \cdot DD(\theta) - 2\sum_{i=0}^{n_R-1} DR_i(\theta)}{\sum_{i=0}^{n_R-1} RR_i(\theta)} + 1$$

Systems

Cell processor

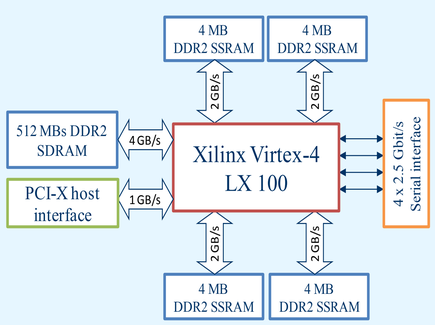


GFLOPS	200
Bandwidth between host memory and PE memory (GB/s)	25.6
Local memory to PE bandwidth (GB/s)	
Frequency (GHz)	3.2
# of processing elements	8

NVIDIA GeForce GTX 280 GPU

GFLOPS	933
Bandwidth between host memory and PE memory (GB/s)	4.0
Local memory to PE bandwidth (GB/s)	141.7
Frequency (GHz)	1.29
# of processing elements	240

Nallatech H101



GFLOPS	20
Bandwidth between host memory and PE memory (GB/s)	1.0
Local memory to PE bandwidth (GB/s)	8
Frequency (GHz)	0.1
# of processing elements	

Algorithm

Algorithm 1: autocorrelation (DD and RR bin counts)

Input: 1) set of Cartesian coordinates of N points $x_1, .., x_N$ on the celestial sphere; 2) for each point x_i, jackknife sample number, $1 \le s_i \le K$, from which x_i is removed; 3) set of M bins given by their boundaries: $[\theta_0, \theta_1), [\theta_1, \theta_2), .., [\theta_{M-1}, \theta_M)$.

Output: for each bin in every jackknife sample, the number of unique pairs of points (x_i, x_j) for which the angular separation $\theta(x_i,x_j)$ is in the respective bin.

```
1  for i=1,..., N-1 do
2      for j=i+1,..., N do
3          l←binmap(θ(x_i,x_j), θ)
4          for k=1,..., K do
5              if s_i ≠ k do B_{k,l}← B_{k,l} +1
```

Algorithm 2: cross-correlation (DR bin counts)

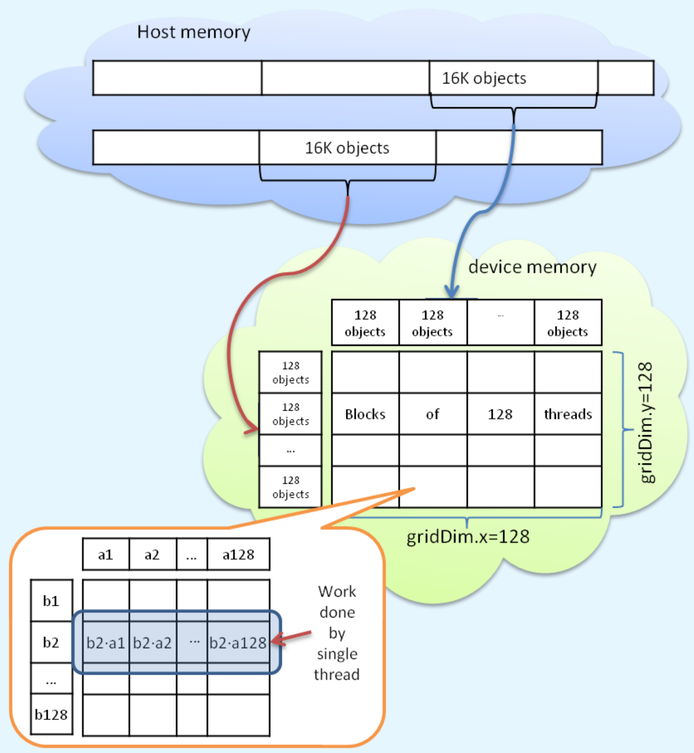
Input: 1) 2 sets of Cartesian coordinates of N points $x_1, .., x_N$ and $y_1, .., y_N$ on the celestial sphere; 2) for each point x_i, jackknife sample number, $0 \le s_i \le K-1$, from which x_i is removed; 3) set of M bins given by their boundaries: $[\theta_0, \theta_1), [\theta_1, \theta_2), .., [\theta_{M-1}, \theta_M)$.

Output: for each bin in every jackknife sample, the number of unique pairs of points (x_i, y_j) for which the angular separation $\theta(x_i,y_j)$ is in the respective bin.

```
1  for i=1,..., N do
2      for j= 1,..., N do
3          l←binmap(θ(x_i,y_j), θ)
4          for k=1,..., K do
5              if s_i ≠ k do B_{k,l}← B_{k,l} +1
```

GPU (CUDA) Implementation

Bin assignment kernel



Histogram kernel

Adaptation of Podlozhnyuk, V. "64-bin Histogram." 2007. http://developer.download.nvidia.com/compute/cuda/sdk/website/projects/histogram64/doc/histogram64.pdf.

FPGA (Dime-C) Implementation

```
53  for (i = 0; i < i_end; i++) {       // loop thru one dataset
54      x1 = X1[i];  y1 = Y1[i]; z1 = Z1[i]; jk = (int)JK[i];
55      if (doSelf) j_start = i+1; else j_start = 0;
56      for (j = j_start; j < n2; j++) {       // loop thru second dataset
57          if (doSelf) offset = j; else offset = n1 + j;
58          x2 = X2[offset]; y2 = Y2[offset]; z2 = Z2[offset];
59          dot = x1 * x2 + y1 * y2 + z1 * z2;          // dot product
60          // unrolled binary search
61          if (dot < binb15) { if (dot < binb23) { if (dot < binb27) { if (dot < binb29) { if (dot < binb30) indx = 31;
62          else indx = 30; } else { if (dot < binb28) indx = 29; else indx = 28; } } else { if (dot < binb25)  {
63          if (dot < binb26) indx = 27;  else indx = 26; } else { if (dot < binb24) indx = 25; else indx = 24; } } }
64          else { if (dot < binb19) { if (dot < binb21) { if (dot < binb22) indx = 23; else indx = 22; } else {
65          if (dot < binb20) indx = 21; else indx = 20; } } else { if (dot < binb17) { if (dot < binb18) indx = 19;
66          else indx = 18; } else { if (dot < binb16) indx = 17; else indx = 16; } } } }
67          else { if (dot < binb07) { if (dot < binb11) { if (dot < binb13) { if (dot < binb14) indx = 15; else indx = 14; }
68          else { if (dot < binb12) indx = 13; else indx = 12; } } else { if (dot < binb09) { if (dot < binb10) indx = 11;
69          else indx = 10; } else { if (dot < binb08) indx = 9; else indx = 8; } } } else { if (dot < binb03) { if (dot < binb05){
70          if (dot < binb06) indx = 7; else indx = 6; } else { if (dot < binb04) indx = 5; else indx = 4; } } else {
71          if (dot < binb01) { if (dot < binb02) indx = 3; else indx = 2; } else { if (dot < binb00) indx = 1; else indx =0; }}}}
72          bin_bank = j % 4;  // update bin values
73          if (jk != 0) val0 = 1; else val0 = 0;             // update corresponding bin for jk=0
74          if (bin_bank == 0) binv1_00[indx] = binv1_00[indx] + val0;
75          else if (bin_bank == 1) binv2_00[indx] = binv2_00[indx] + val0;
76          else if (bin_bank == 2) binv3_00[indx] = binv3_00[indx] + val0;
77          else binv4_00[indx] = binv4_00[indx] + val0;
...
126     } }
```

Cell B./E. Implementation

PPE
Convert data layout from array of structures to vectors of coordinates
Divide work between 8 SPEs

SPE0 ... **SPE7**
DMA datasets from the main memory
for (i=0; i < n1; i++) {
  for (j=0; j < n2; j+=4) {
    compute four dot products at a time
    attempt to compute bin indices in parallel, four at a time
    if failed, compute bin indices one at a time
    compute bin assignment via a lookup table
    update running sum for bin counts }
  update global sum for bin counts }
DMA results to the main memory

PPE
Aggregate results from multiple SPEs

Results

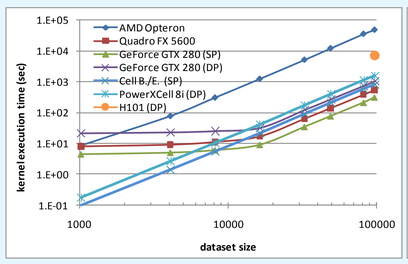


Fig 1. Execution time vs. dataset size

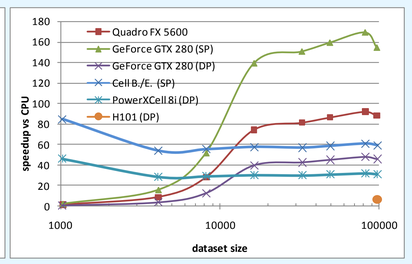


Fig 2. Speedup vs. dataset size

Cell B/E (SP) speedup: 59.1×
Quadro FX 5600 (SP) speedup: 88.2×
GeForce GTX 280 (SP) speedup: 154.1×

H101 (DP) speedup: 6.8×
PowerXCell 8i (DP) speedup: 30.9×
GeForce GTX 280 (DP) speedup: 54.5×

ASTRONOMY
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

NCSA