Developing and Deploying Advanced Algorithms to Novel Supercomputing Hardware

Robert J. Brunner[1,2], Volodymyr V. Kindratenko[2], and Adam D. Myers[1]

1) Department of Astronomy
2) National Center for Supercomputing Applications
University of Illinois at Urbana-Champaign

rb@astro.uiuc.edu, kindr@ncsa.uiuc.edu, adm@astro.uiuc.edu

NCSA Production HPC Systems

- **Dell Intel® 64 Linux Cluster [abe]**
  - Dell blade system with 1,200 PowerEdge 1955 dual socket, quad core compute blades, an InfiniBand interconnect and 100 TB of storage in a Lustre filesystem
  - Peak performance: **88 TF**
- **Dell Blade system [t3]**
  - 1,040 dual-core Intel 2.66 GHz processors an InfiniBand interconnect, 4.1 terabytes of total memory, and a 20 terabyte Lustre filesystem
  - Peak performance: **22.1 TF**
- **Dell Xeon Cluster [tungsten]**
  - 2,560 Intel IA-32 Xeon 3.2 GHz processors, 3 GB memory/node
  - Peak performance: **16.38 TF** (9.819 TF sustained)
  - Top 500 list debut: #4 (November 2003)
- **IBM IA-64 Linux Cluster [mercury]**
  - 1,774 Intel Itanium 2 1.3/1.5 GHz processors, 4 GB and 12 GB memory/node
  - Peak performance: **10.23 TF** (7.22 TF sustained)
  - Top 500 list debut: #15 (June 2004)
- **SGI Altix [cobalt]**
  - 1,024 Intel Itanium 2 processors
  - Peak performance: **6.55 TF** (6.1 TF sustained)
  - Top 500 list debut: #48 (June 2005)
- **IBM pSeries 690 [copper]**
  - 384 IBM POWER4 p690 processors, 7 with 64 GB/system, 4 with 256 GB/system
  - Peak performance: **2 TF** (708 GF sustained)
  - Top 500 list debut: #99 (June 2003)

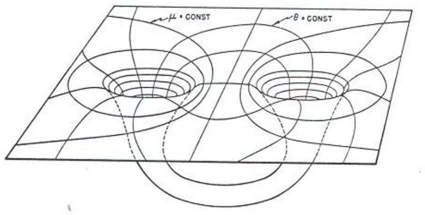
Source: http://www.ncsa.uiuc.edu/UserInfo/Resources/Hardware/

HPC Challenges

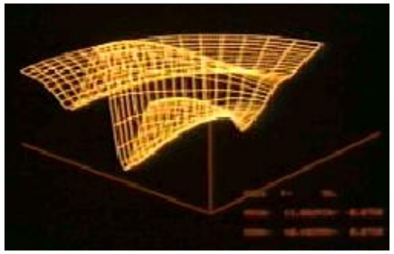
- **The gap between the application performance and the peak system performance increases**
  - Few applications can utilize high percentage of microprocessor peak performance, but even fewer applications can utilize high percentage of the peak performance of a multiprocessor system
- **Computational complexity of scientific applications increases faster than the hardware capabilities used to run the applications**
  - Science and engineering teams are requesting more cycles than HPC centers can provide
- **I/O bandwidth and clock wall put limits on computing speed**
  - Computational speed increasing faster than memory or network latency is decreasing
  - Computational speed is increasing faster than memory bandwidth
  - The processor speed is limited due to leakage current
  - Storage capacities increasing faster than I/O bandwidths
- **Building and using larger machines becomes more and more challenging**
  - Increased space, power, and cooling requirements
    - ~$1M+ per year in cooling and power costs for moderate sized systems
  - Application fault-tolerance becomes a major concern

Black Hole Collision Problem

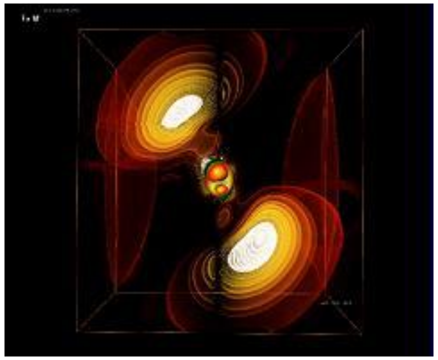
1,800,000,000X



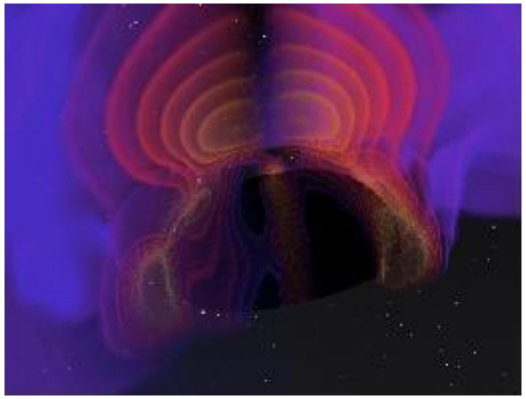
1963
Hahn and Lindquist
IBM 7090
One Processor
Each 0.2 MF
3 Hours



1977
Eppley and Smarr
CDC 7600
One Processor
Each 35 MF
5 Hours



1999
Seidel and Suen, *et al.*
NCSA SGI Origin
256 Processors
Each 500 MF
40 Hours



2001
Seidel *et al*
NCSA Pentium III
256 Processors
Each 1 GF
500,000 Hours total
plus 500,000 hours at NERSC

(~50 KW)

300X

30,000X

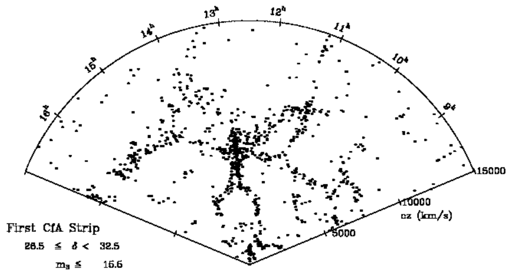
~200X

Processor speedup is *only* 5000x

Source: http://gladiator.ncsa.uiuc.edu/PDFs/iacat/Pennington_PetascaleSystems_Apr06.pdf

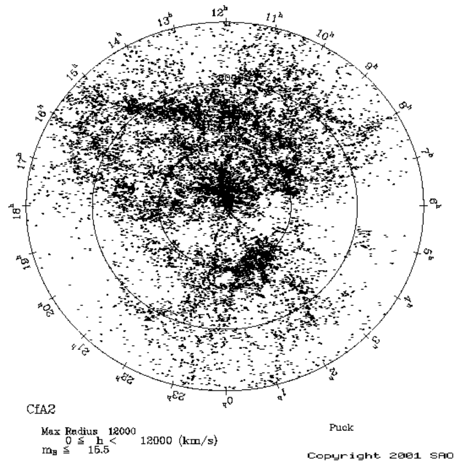
Digit{ized|al} Sky Surveys

From Data Drought to Data Flood



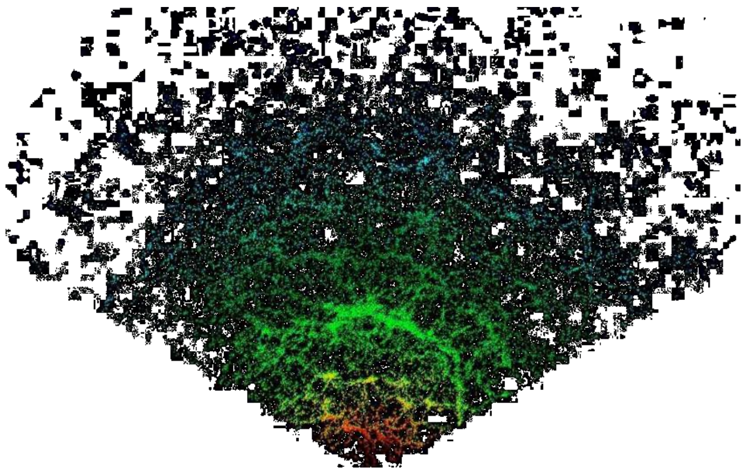
1977-1982
First CfA Redshift Survey

spectroscopic observations of
1,100 galaxies



1985-1995
Second CfA Redshift Survey

spectroscopic observations of
18,000 galaxies



2000-2005
Sloan Digital Sky Survey I

spectroscopic observations of
675,000 galaxies

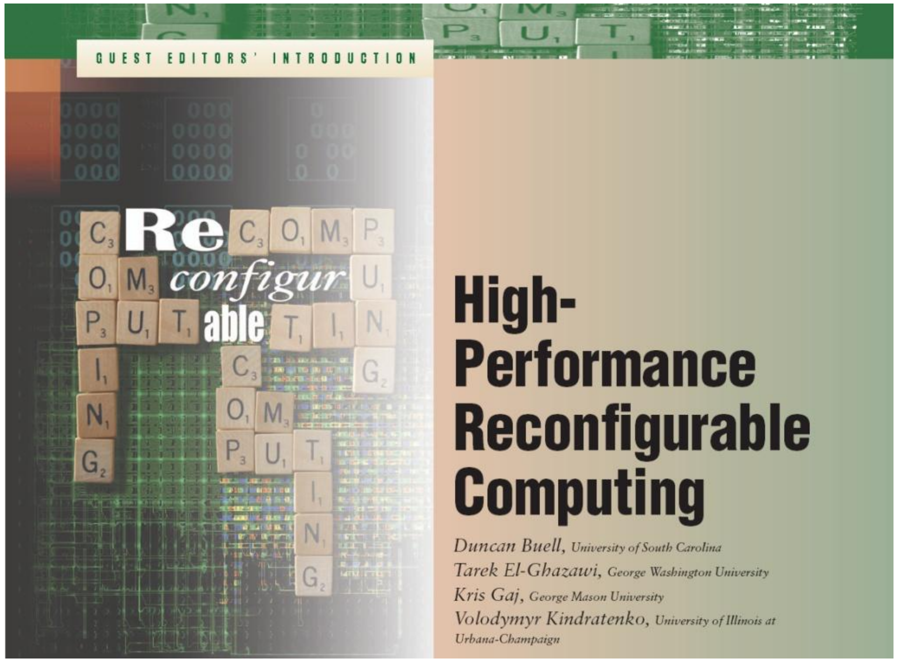
Sources: http://www.cfa.harvard.edu/~huchra/zcat/
http://www.sdss.org/

New Ways of Computing

- **General-purpose processors**
  - Multi-core
- **Special-purpose processors**
  - Field-Programmable Gate Arrays (FPGAs)
    - Digital signal processing, embedded
  - Graphics Processing Units (GPUs)
    - Desktop graphics accelerators
  - Sony/Toshiba/IBM Cell Broadband Engine
    - Game console and digital content delivery systems
  - …

High-Performance Reconfigurable Computing (HPRC)

- **Gerald Estrin's idea of "fixed plus variable structure computer"**
  - reconfigurable hardware is tailored to perform a specific task
    - as quickly as a dedicated piece of hardware
  - once the task is done, the hardware is adjusted to do other tasks
  - the main processor controls the behavior of the reconfigurable hardware
- **Wikipedia's definition**
  - "Reconfigurable computing is computer processing with highly flexible computing fabrics. The principal difference when compared to using ordinary microprocessors is the ability to make substantial changes to the data path itself in addition to the control flow."
- **Field Programmable Gate Array (FPGA) is the enabling technology**


GUEST EDITORS' INTRODUCTION

High-Performance Reconfigurable Computing

Duncan Buell, University of South Carolina
Tarek El-Ghazawi, George Washington University
Kris Gaj, George Mason University
Volodymyr Kindratenko, University of Illinois at Urbana-Champaign

- **IEEE Computer, March 2007**
- **High-Performance Reconfigurable Computers are parallel computing systems that contain multiple microprocessors and multiple FPGAs. In current settings, the design uses FPGAs as coprocessors that are deployed to execute the small portion of the application that takes most of the time—under the 10-90 rule, the 10 percent of code that takes 90 percent of the execution time.**

Reconfigurable Computing

Promises

- Higher sustained performance
  - exploring inherent parallelism in algorithms
    - spatial parallelism, instruction level parallelism
  - matching computation with data flow
- FPGAs are on a faster 'growth' curve than CPUs
  - Can keep up with the increasing complexity of scientific applications
- Reduced power requirements as compared to microprocessor-based systems
  - Larger systems can be built
- Faster execution, better resource utilization, and lower power consumption

and Pitfalls

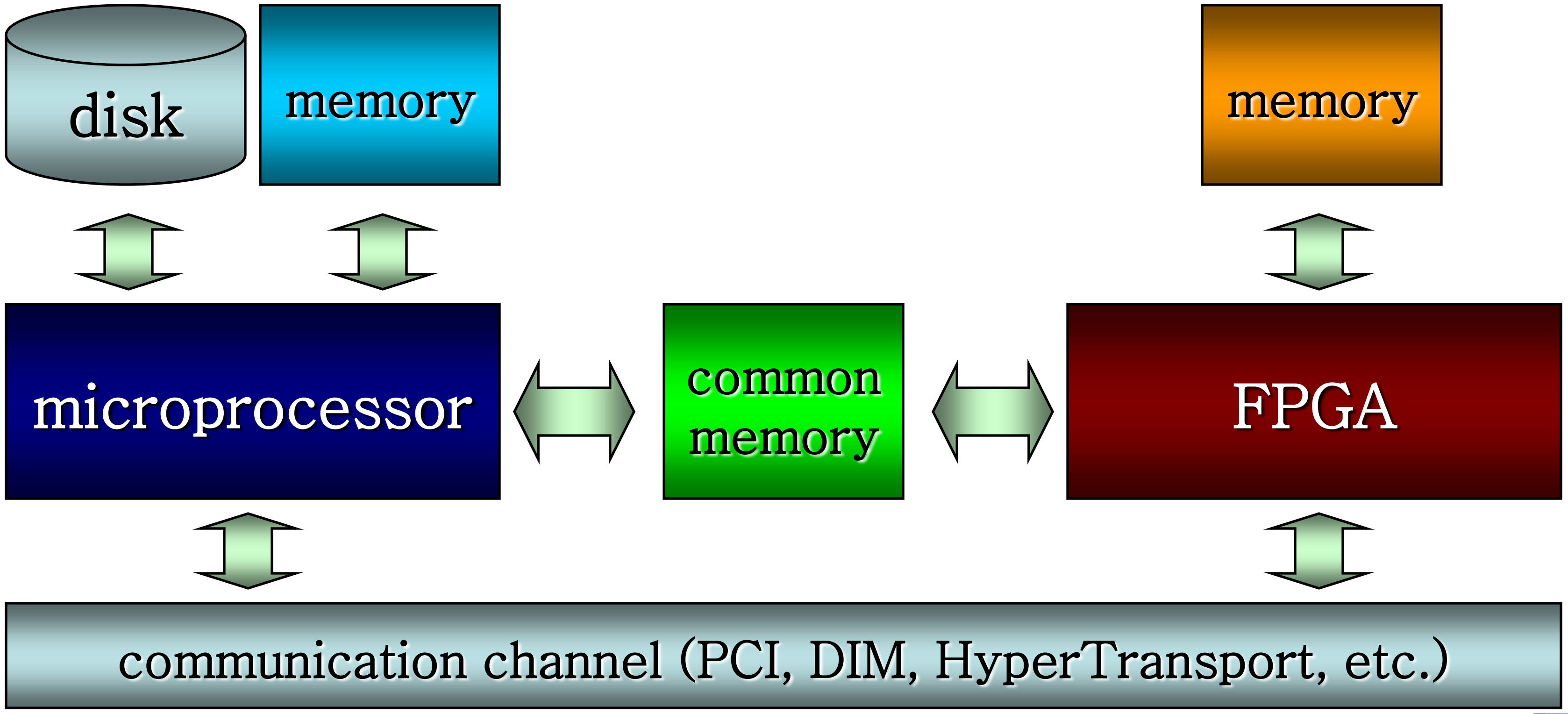
- Current FPGA technology does not address the needs of scientific computing community
  - Gate count on FPGAs only recently became sufficient for practical use in applications with DPFP
  - No dedicated FP hardware support
- Software development for RC systems by computational scientists still remains not easy
  - Software development methodology for RC is different from software development methodology for microprocessor-based systems

Our Motivations

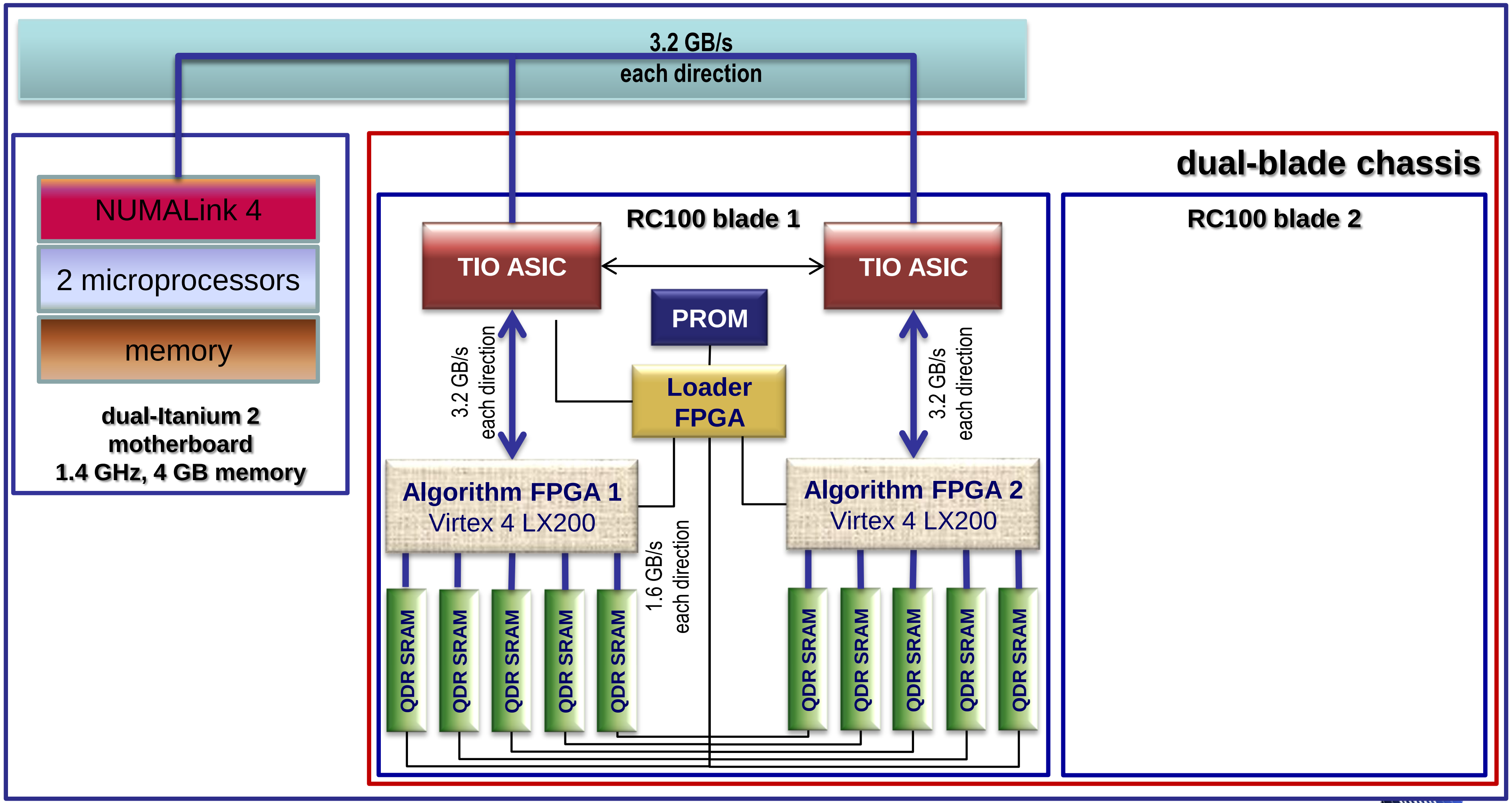
- **Can Reconfigurable Computing be used to accelerate computationally intensive applications in Cosmology?**
  - Speedup of an order of magnitude or more
- **Can computational scientists effectively use Reconfigurable Computing without the need to re-write all their code from scratch?**
  - Reuse of legacy code is important
- **Can computational scientists effectively use Reconfigurable Computing without the need to become hardware experts?**
  - C/Fortran style of code development as opposite to hardware design tools and hardware description languages
- **Is this technology viable today and will it be viable in 5, 10 years from now?**
  - Technology development roadmap
  - FPGA performance trends vs. multi-core CPU performance trends

HPRC System Concept Overview

- Microprocessor
- Reconfigurable processor

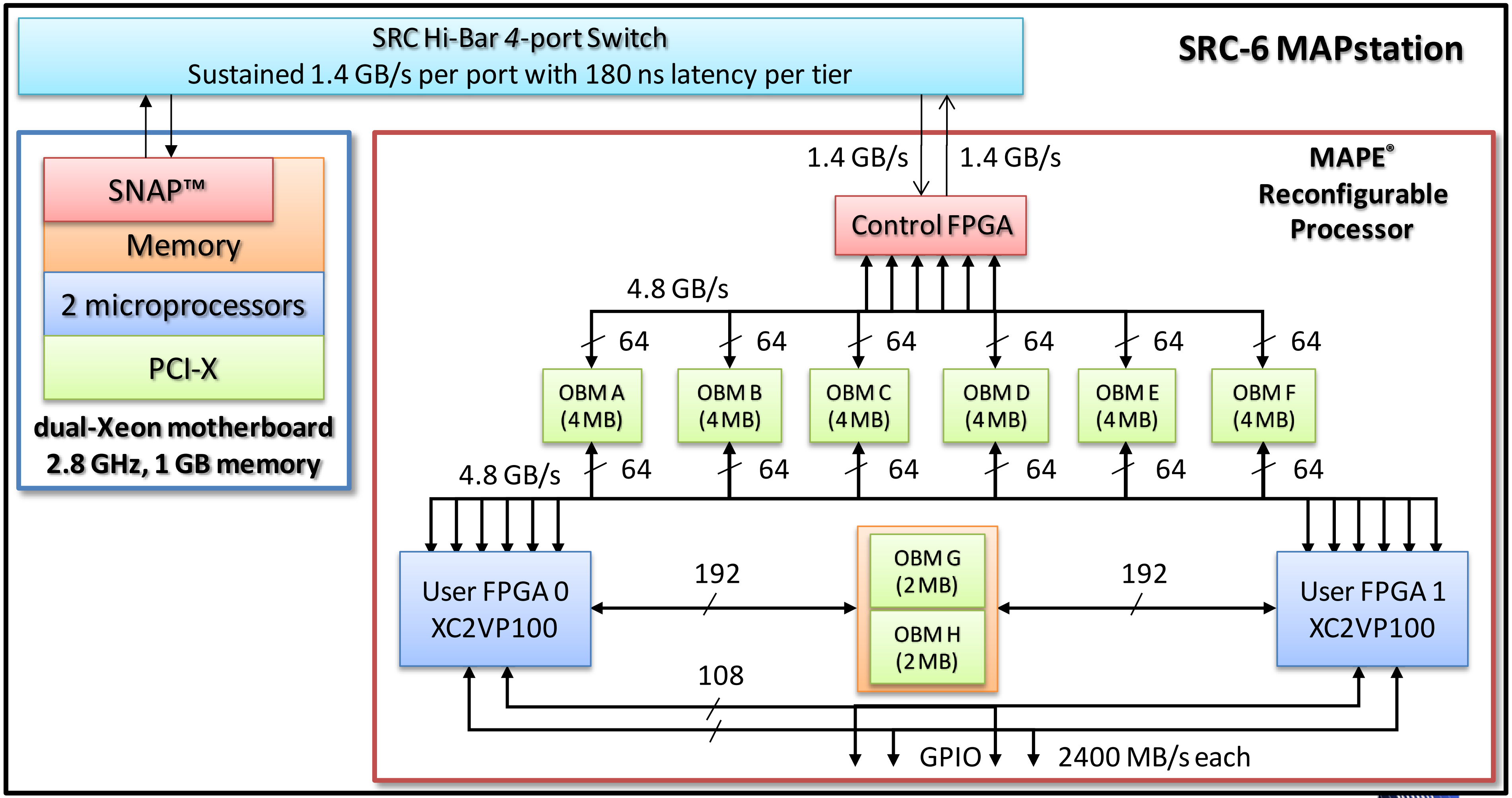


SGI Altix 350 with RC100 Blade

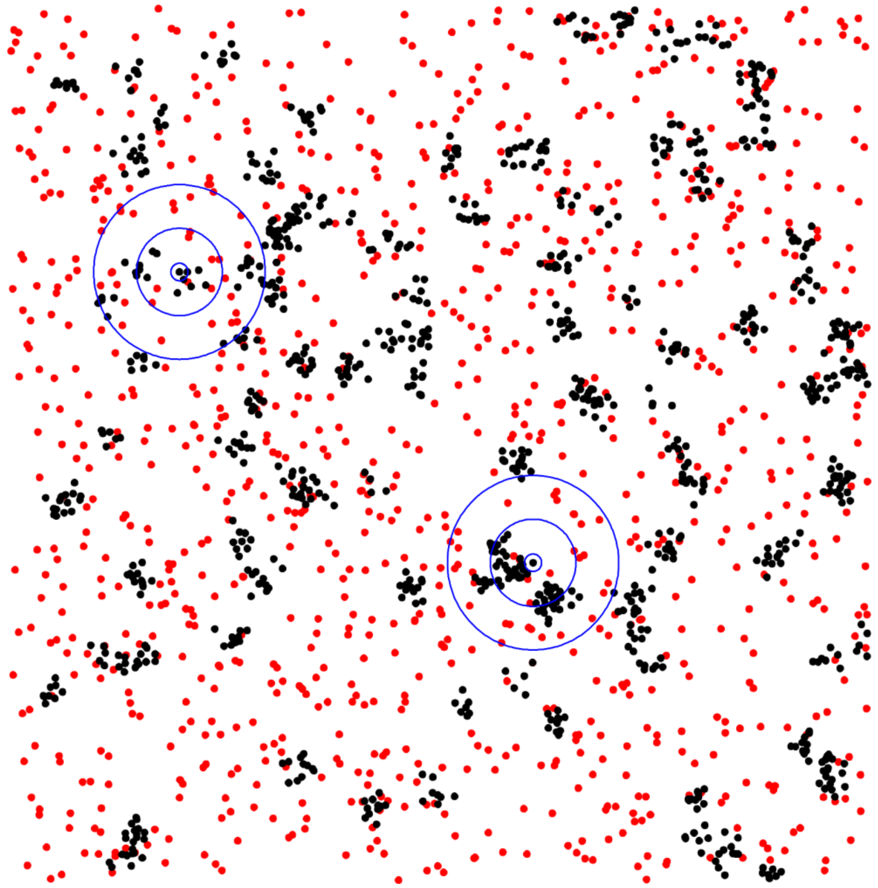


NCSA

SCR-6 Reconfigurable Computer



Two-point Angular Correlation



- **TPACF, denoted as $\omega(\theta)$, is the frequency distribution of angular separations θ between celestial objects in the interval $(\theta, \theta + \delta\theta)$**
  - θ is the angular distance between two points
- **Red Points are, on average, randomly distributed, black points are clustered**
  - Red points: $\omega(\theta)=0$
  - Black points: $\omega(\theta)>0$
- **Can vary as a function of angular distance, θ (blue circles)**
  - Red: $\omega(\theta)=0$ on all scales
  - Black: $\omega(\theta)$ is larger on smaller scales

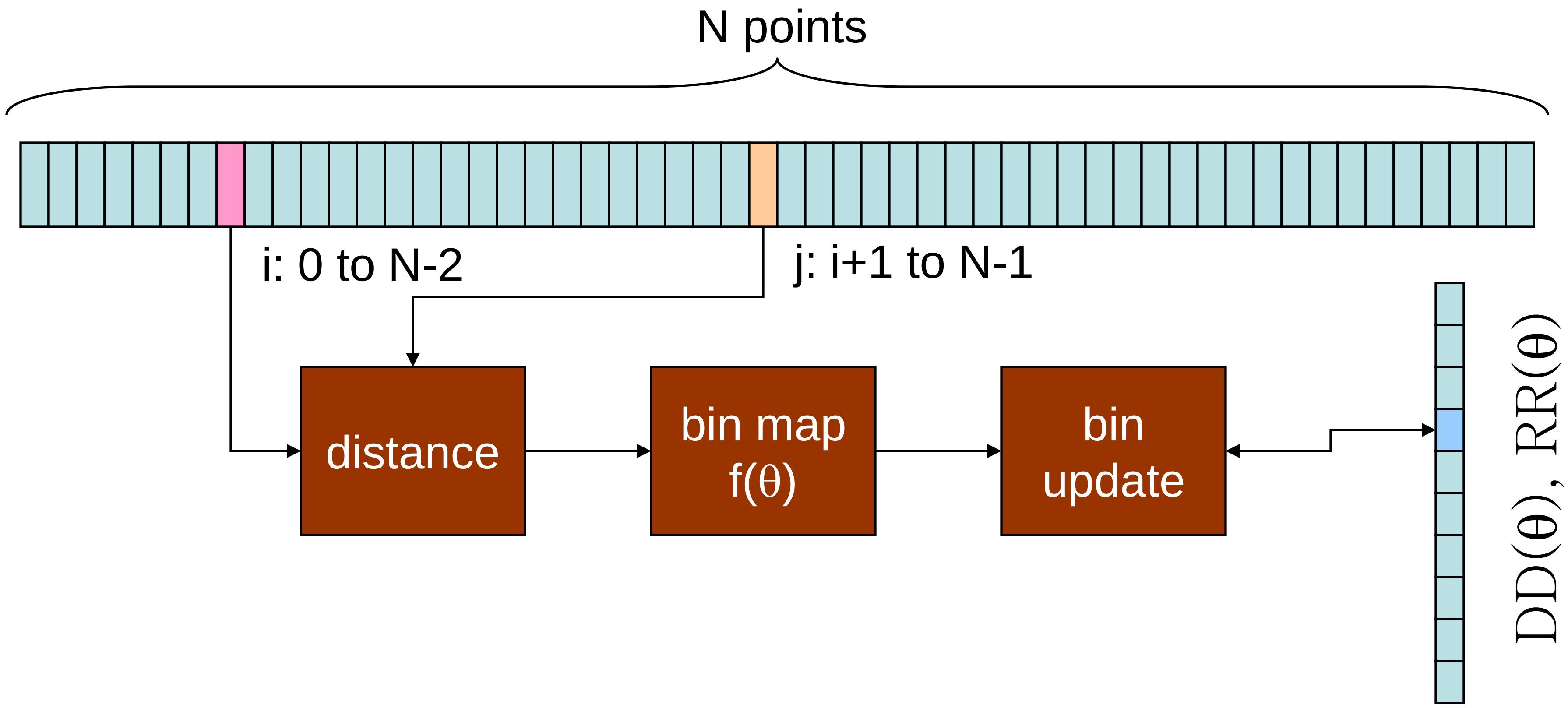
The Method

- **The angular correlation function is calculated using the estimator derived by Landy & Szalay (1993):**

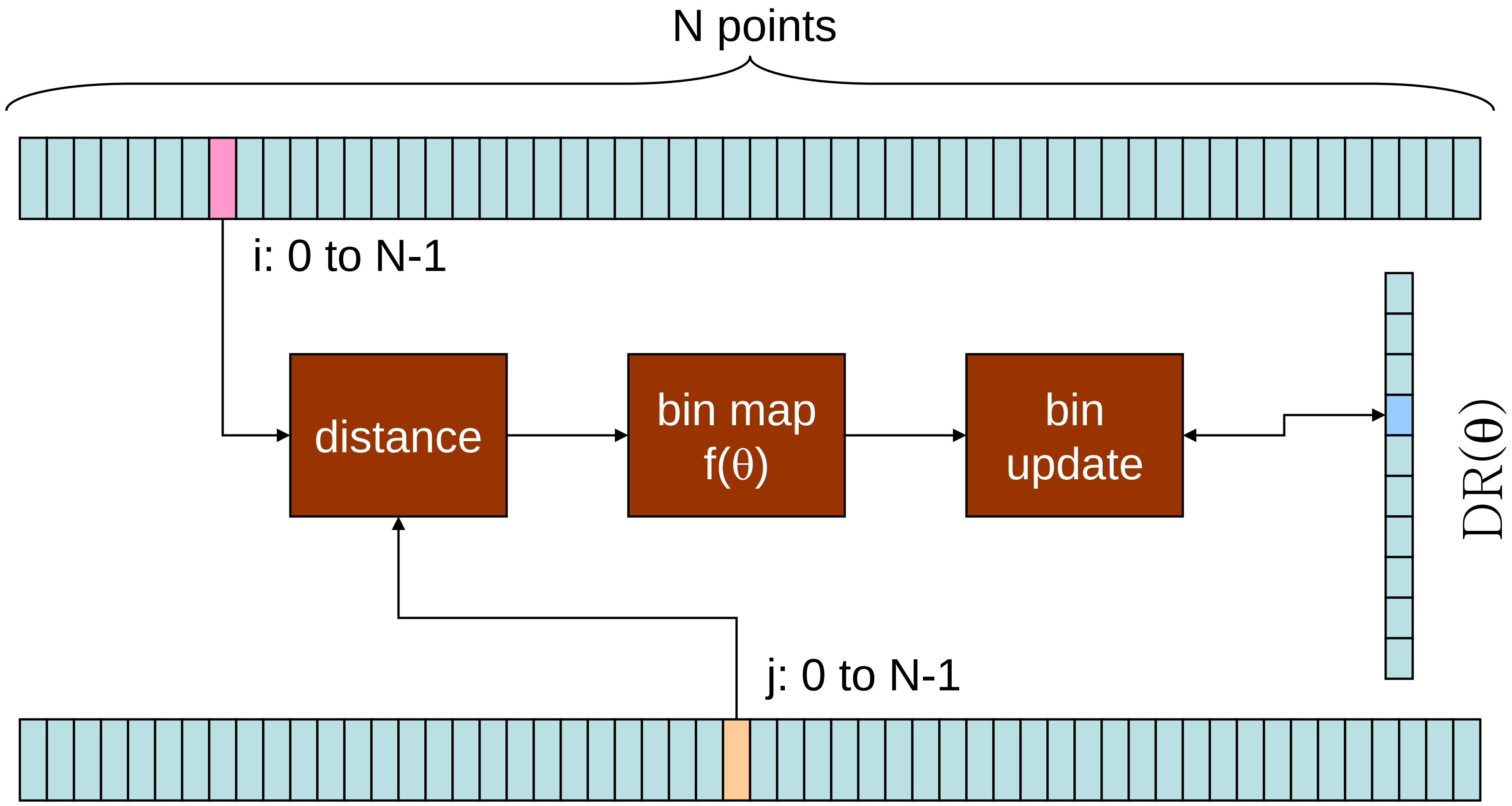
$$\omega(\theta) = \frac{\frac{1}{n_D^2}\cdot DD(\theta) - \frac{2}{n_D n_R}\sum DR_i(\theta)}{\frac{1}{n_R^2}\sum RR_i(\theta)} + 1$$

- **where *DD*(θ) and *RR*(θ) are the autocorrelation function of the data and random points, respectively, and *DR*(θ) is the cross-correlation between the data and random points.**

DD & RR Algorithm: Autocorrelation



DR Algorithm: Cross-correlation

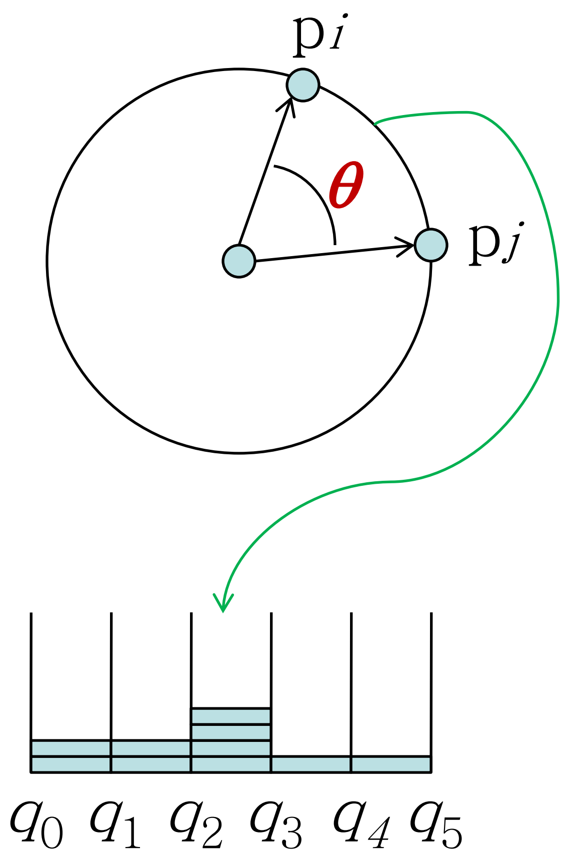


Microprocessor Code Organization

```
// compute DD
doCompute{CPU|FPGA}(data, npd, data, npd, 1, DD, binb, nbins);

// loop through random data files
for (i = 0; i < random_count; i++)
{
        // compute RR
        doCompute{CPU|FPGA}(random[i], npr[i], random[i], npr[i], 1, RRS, binb, nbins);

        // compute DR
        doCompute{CPU|FPGA}(data, npd, random[i], npr[i], 0, DRS, binb, nbins);
}

// compute w
for (k = 0; k < nbins; k++)
{
        w[k] = (random_count * 2*DD[k] - DRS[k]) / RRS[k] + 1.0;
}
```

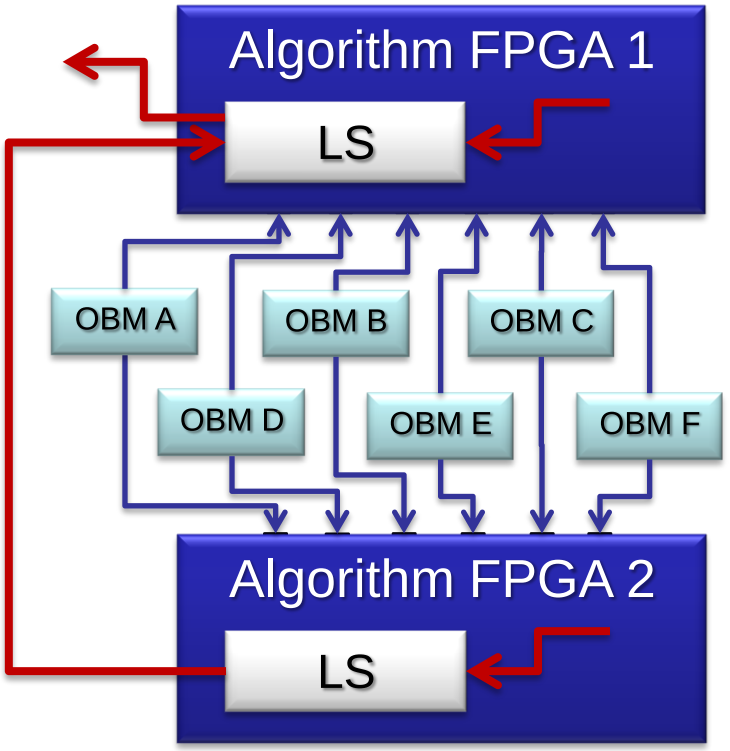
Reference C Kernel Implementation

```
for (i = 0; i < ((autoCorrelation) ? n1-1 : n1); i++)
{
   double xi = data1[i].x;
   double yi = data1[i].y;
   double zi = data1[i].z;

   for (j = ((autoCorrelation) ? i+1 : 0); j < n2; j++)
   {
      double dot = xi * data2[j].x + yi * data2[j].y + * data2[j].z;

      // binary search
      min = 0;  max = nbins;
      while (max > min+1)
      {
         k = (min + max) / 2;
         if (dot >= binb[k])  max = k;
         else min = k;
      };

      if (dot >= binb[min])  data_bins[min] += 1;
      else if (dot < binb[max]) data_bins[max+1] += 1;
      else data_bins[max] += 1;
   }
}
```



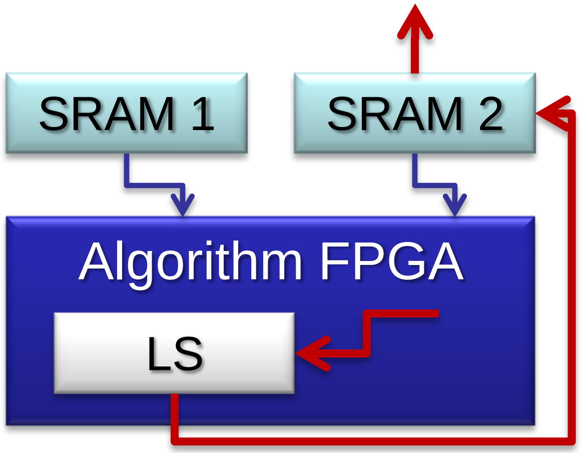
Kernel Written in MAP C (SRC-6)

```
// main compute loop
for (i = 0; i < n1; i++) {
    pi_x = AL[i];  pi_y = BL[i];  pi_z = CL[i];   // point i

  #pragma loop noloop_dep
  for (j = 0; j < n2; j++) {
      // what bin memory bank to use in this loop iteration
      cg_count_ceil_32 (1, 0, j == 0, 3, &bank);

      pj_x = DL[j];  pj_y = EL[j];  pj_z = FL[j];    // point j
      dot = pi_x * pj_x + pi_y * pj_y + pi_z * pj_z;  // dot product

     // find what bin it belongs to
      select_pri_64bit_32val( (dot < bv31), 31, (dot < bv30), 30,
        …
        (dot < bv02), 2,  (dot < bv01), 1,  0, &indx);

      // update the corresponding bin count
              if (bank == 0) bin1a[indx] += 1;
      else if (bank == 1) bin2a[indx] += 1;
      else if (bank == 2) bin3a[indx] += 1;
                          else bin4a[indx] += 1;
  }
}
```



Kernel Written in Mitrion-C (RC100)

```
// loop in one data set
(bins, afinal, bfinal) = for (i in <0 .. NPOINTS_1>)
{
   (xi, yi, zi, a1, b1) = readpoint(a0, b0, i); // read next point

   uint:64[NBINS] binsB = binsA;
   ExtRAM a2 = a0;
   ExtRAM b2 = b0;

   (binsA, a3, b3) = for(j in <0 .. NPOINTS_1>)
   {
      (xj, yj, zj, a2, b2) = readpoint(a1, b1, j+NPOINTS); // read next point

      float:53.11 dot = xi * xj + yi * yj + zi * zj; // compute dot product

      int:8 indx = findbin(dot, binb); // find what bin it belongs to

      // update bin
      binsB = foreach (bin in binsB by ind) if (ind == indx) bin + 1 else bin;
   } (binsB, a2, b2);
} (binsA, a3, b3);
```



Performance on Different Platforms

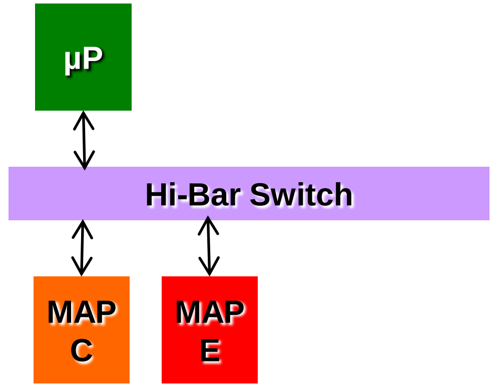
- ~100,000 data points, 100 random files

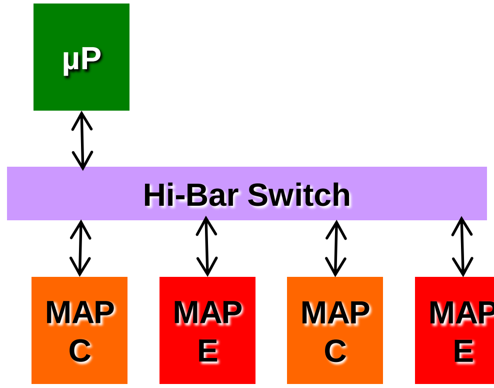
Measured features/ parameters	SRC-6 host *2.8 GHz Xeon*	SRC-6 dual-MAP	SGI Altix host *1.4 GHz Itanium 2*	RC100 blade
# CPUs	2		2	
# FPGAs		4		2
# of compute engines	1	17	2	4
DD time (s)	219.5	3	226.6	49.7
DR+RR time (s)	84,354.3	880.3	47,598.6	4,975.3
Load/convert (s)	20.3	20.7	28.4	27.5
Total (s)	84,594.1	904	47,853.6	5,052.5
Overall Speedup	1.0	93.5x(1) 52.9x	1.0	9.5x(2)

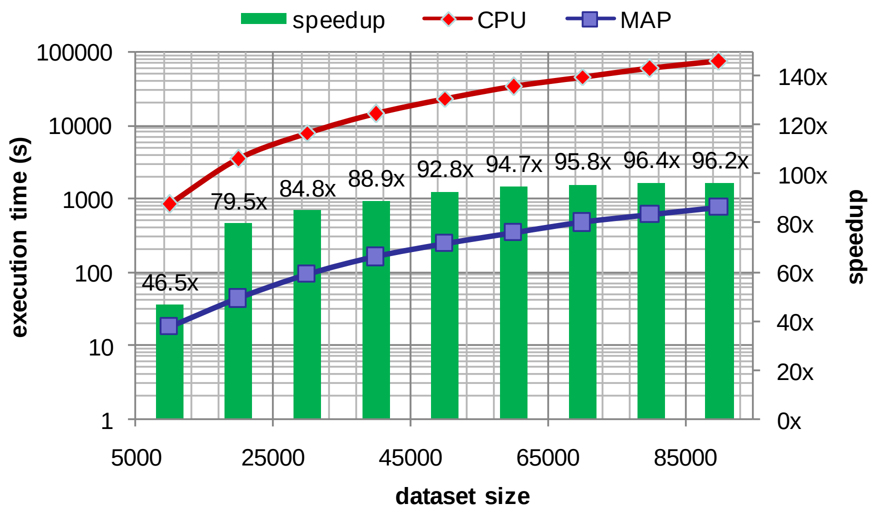
(1) V. Kindratenko, R. Brunner, A. Myers, *Dynamic load-balancing on multi-FPGA systems: a case study*, In Proc. 3rd Annual Reconfigurable Systems Summer Institute - RSSI'07, 2007.
(2) V. Kindratenko, R. Brunner, A. Myers, *Mitrion-C Application Development on SGI Altix 350/RC100*, In Proc. IEEE Symposium on Field-Programmable Custom Computing Machines - FCCM'07, 2007.

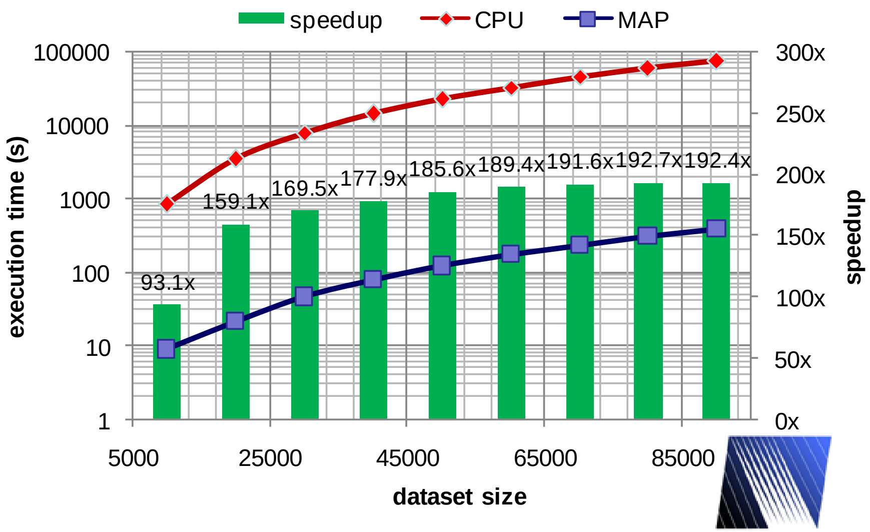
Scalability Study

- Actual (dual-MAP)
- Projected (quad-MAP)



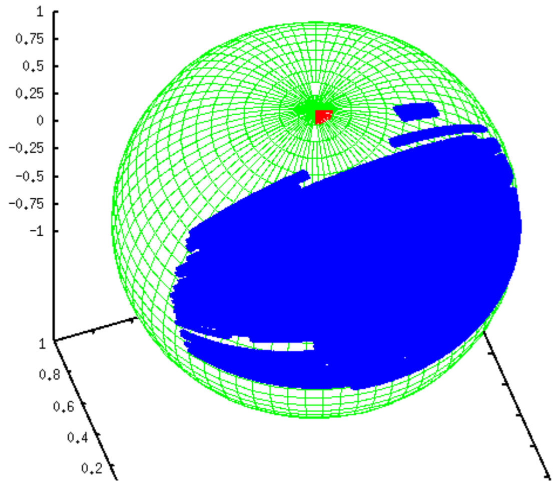


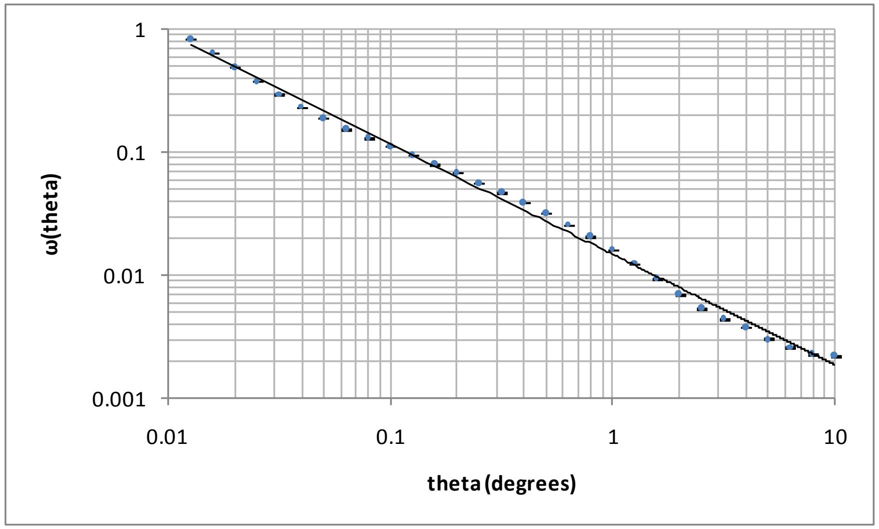




First Results Obtained on SRC-6

- **SDSS DR5 photometric-selected Luminous Red Galaxy sample**
  - Observed dataset consisting of 1,641,323 points
  - 100 random datasets, 1,000,000 points each
- **Model**
  - Error estimation using 10 subsets
- **Compute time**
  - 10.2 days (vs. 980 days on a single 2.8 GHz Intel Xeon chip)





Lessons Learned

- **Porting an existing code to an RC platform is considerably more difficult than developing a new code**
  - Requires an in-depth understanding of the code structure and data flow
  - Code optimization techniques used in the microprocessor-based implementation are not applicable for RC implementation
  - Data flow schemes used in the microprocessor-based implementation in most cases are not suitable for RC implementation
- **Only few scientific codes can be ported to an RC platform with relatively minor modifications**
  - 90% of time is spent while executing 10% of the code
- **Vast majority of the codes require significant restructuring in order to be 'portable'**
  - No well-defined compute kernel
  - Compute kernel is too large to fit on an FPGA
  - Compute kernel operates on a small dataset or is called too many times
    - function call overhead becomes an issue

Lessons Learned

- **Effective use of high-level programming languages/tools, such as MAP C/Carte (SRC-6) and Mitrion-SDK/Mitrion-C (RC100), to develop code for RC platform requires some limited hardware knowledge**
  - Memory organization and limitations
    - Explicit data transfer and efficient data access
  - On-chip resources and limitations
  - RC architecture-specific programming techniques
    - Pipelining, streams, …
- **Most significant code acceleration can be achieved when developing the code from scratch; code developer then has the freedom to**
  - structure the algorithm to take advantage of the RC platform organization and resources,
  - select most effective SW/HW code partitioning scheme, and
  - setup data formats and data flow graph that maps well into RC platform resources

Future Work

- **Integrate optimizations into the two-point angular correlation algorithm**
  - Algorithmic enhancements
  - Tree-data structures
- **Extend work to other cosmology algorithms**
  - Power spectrum
  - Higher order correlations
- **Extend work to other algorithm classes**
  - Machine-learning algorithms
- **Expand to other special-purpose computing platforms**
  - Cell/B.E. processor
  - NVIDIA G80 GPU

Conclusions

- **Reconfigurable Computing holds some potential for accelerating Cosmology applications**
  - Dual-MAP implementation of the two-point angular correlation algorithm outperforms a 2.8 GHz CPU by a factor of over 90
- **Reuse of legacy code is not easy and is not always possible**
  - Experience with porting existing codes to SRC-6 shows that the code has to be significantly restructured/simplified before it becomes feasible to port it to SRC-6
- **C/Fortran style of code development is possible and is quite effective with tools such as Carte and Mitrion-C**
  - Even though it still requires some hardware knowledge of the RC platform

Acknowledgements

- **Work funded by NASA grant NNG06GH15G**
- **SRC Computers, Inc. collaborators**
  - David Caliga, Dan Poznanovic, Dr. Jeff Hammes, Jon Huppenthal
- **An SGI Altix 350 system with an RC100 blade was kindly made available by SGI**
  - Special thanks to Matthias Fouquet-Lapar, Tony Galieti, and Dick Riegner, all from SGI, for their help and support with the SGI system
- **Mitrion SDK for RC100 RASC system was kindly provided by Mitrionics AB**
  - Special thanks for Stefan Möhl and Jace Mogill, both from Mitrionics AB, for their help with Mitrion SDK

RSSI

RECONFIGURABLE SYSTEMS SUMMER INSTITUTE

- **When: July 17-20, 2007**
- **Where: NCSA, Urbana, IL**
- **What:**
  - July 17
    - Nallatech Training and Users Group Workshop
    - SGI/Mitrionics workshop
    - SRC Users Meeting
  - July 18
    - A keynote by **Alan D. George**, director of the National Science Foundation Center for High-Performance Reconfigurable Computing (CHREC)
    - Poster session
  - July 19
    - OpenFPGA meeting
  - July 18-20
    - 22 vendor and academic presentations
    - 15 exhibitors
- **http://rssi.ncsa.uiuc.edu**

Sponsored by:

NCSA

MANCHESTER 1824 The University of Manchester

Open fpga

These sponsors are members of OpenFPGA

CRAY

SRC COMPUTERS IMPLICIT+EXPLICIT™ ARCHITECTURE

NALLATECH The High Performance FPGA Solutions Company

mitrionics™

AMD

intel

ALTERA

XILINX

sgi