Efficient Computation of Radial Distribution Function on GPUs

Yi-Cheng Tu * and Anand Kumar

Department of Computer Science and Engineering

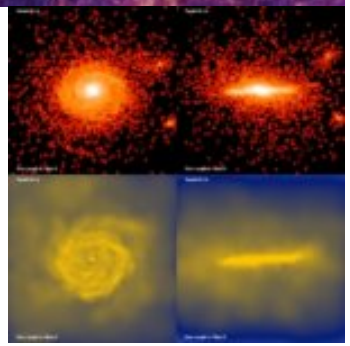
University of South Florida, Tampa, Florida

Overview

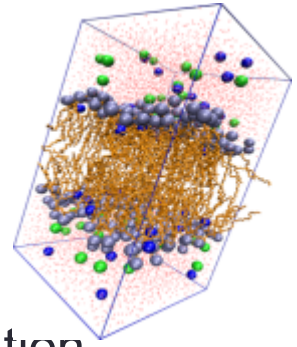
- Introduction and Motivation
- Spatial Distance Histogram (SDH)
- GPU Architecture
- Brute-force SDH Algorithm on GPU
- Advanced SDH Algorithms on GPU
- Conclusions and Future Work

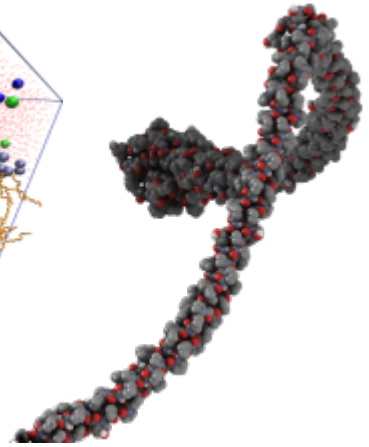
Introduction and Motivation

- Scientific databases: very large amount of data
  - CERN Hadron Collider generates 15 PB every year
  - Next Generation Genome Sequencer generates
    100s of GB in few days



- Molecular / Particle simulations
  - Study physical systems through simulations
  - System state is stored at many time instances
  - Each instance is named a ***frame***



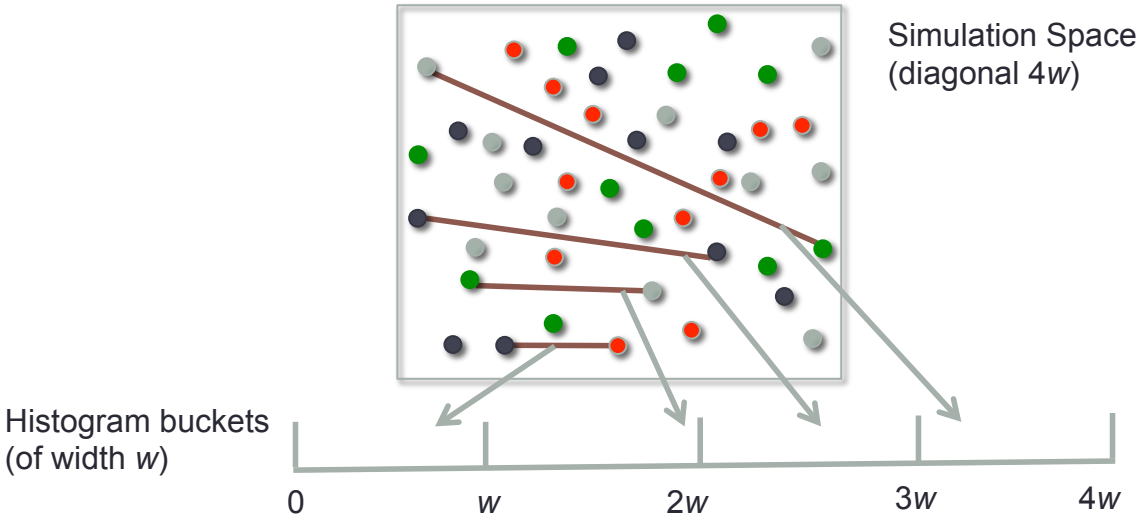


- Frame constitutes – system state
  - Measurement of particle properties
    - Physical location, velocity, charge, mass etc.
  - Thousands of frames are generated in a simulation

Introduction and Motivation

- Millions of particles / atoms are simulated
- Analytical query processing in MS data
  - Center of mass
  - System density, energy
  - Mean square displacement
  - Radial distribution function (RDF)

Spatial Distance Histogram (SDH)
- Key step for computing RDF and other queries
- Interesting and tough as compared to others - $O(N^2)$



Overview

- Introduction and Motivation
- Spatial Distance Histogram (SDH)
- GPU Architecture
- Brute-force SDH Algorithm on GPU
- Advanced SDH Algorithms on GPU
- Conclusions and Future Work

SDH Algorithms on CPUs

- Naïve algorithm in simulation software (e.g., GROMACS)
- Space partitioning trees (*kd*-tree) [NIPS00]
  - Process group of particles as one unit
- Density-Map SDH (DM-SDH) based on Quad/Oct-trees [ICDE09,VLDBJ11,EDBT12,TKDE13,TKDE14]
  - Runs in $O(N^{(2d-1)/d})$ for *d*-dimensional data
  - Approximate variants with running time independent of *N*

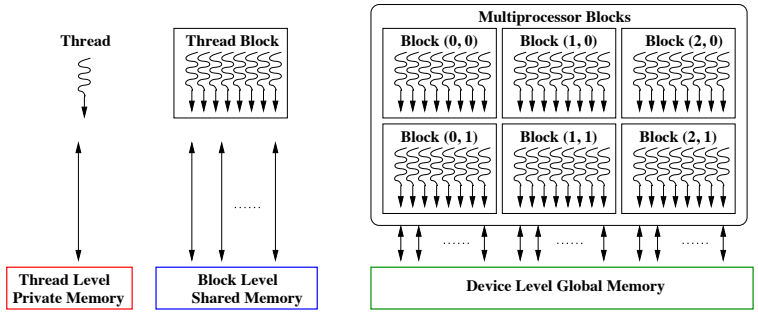
- Modern multicore hardware posses tremendous power
- Utilize the power for On-the-fly SDH computation
- All aforementioned algorithms are easily parallelizable

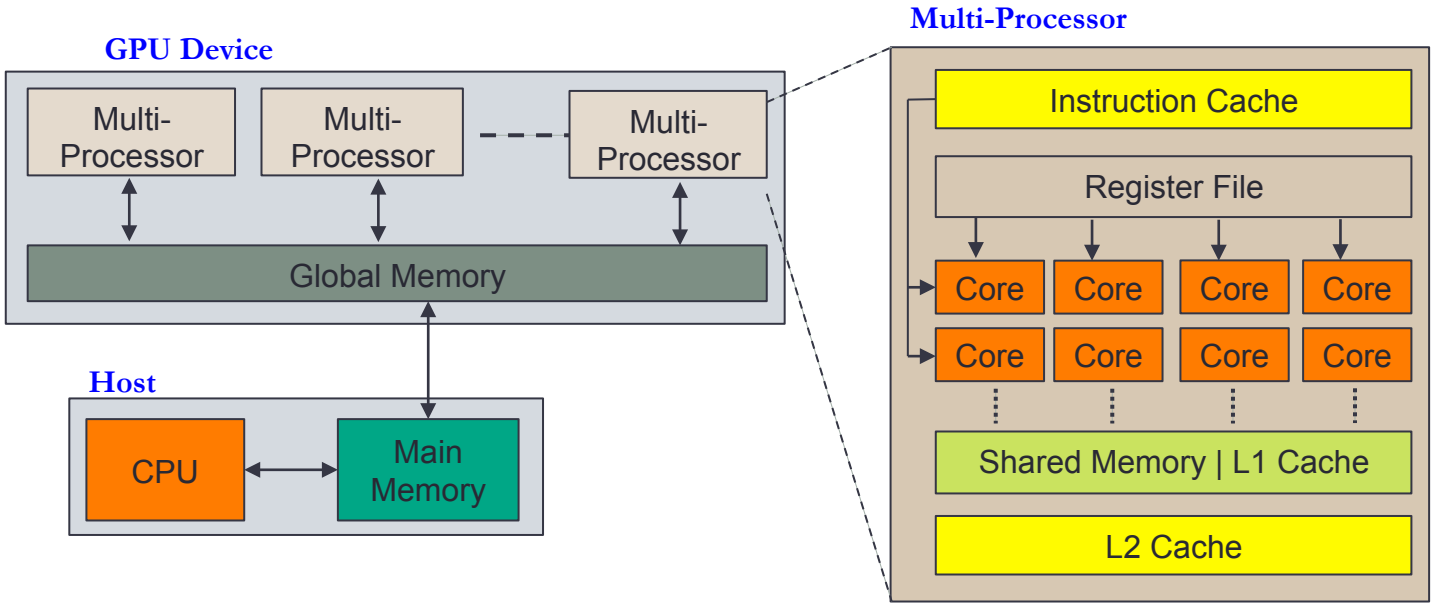
Overview

- Introduction and Motivation
- Spatial Distance Histogram (SDH)
- GPU Architecture
- Brute-force SDH Algorithm on GPU
- Advanced SDH Algorithms on GPU
- Conclusions and Future Work

GPU Architecture

- GPU host thousands of cores
- Hierarchy of memory with different access latency
- Process data in SIMD fashion
- Multiple threads access memory in parallel





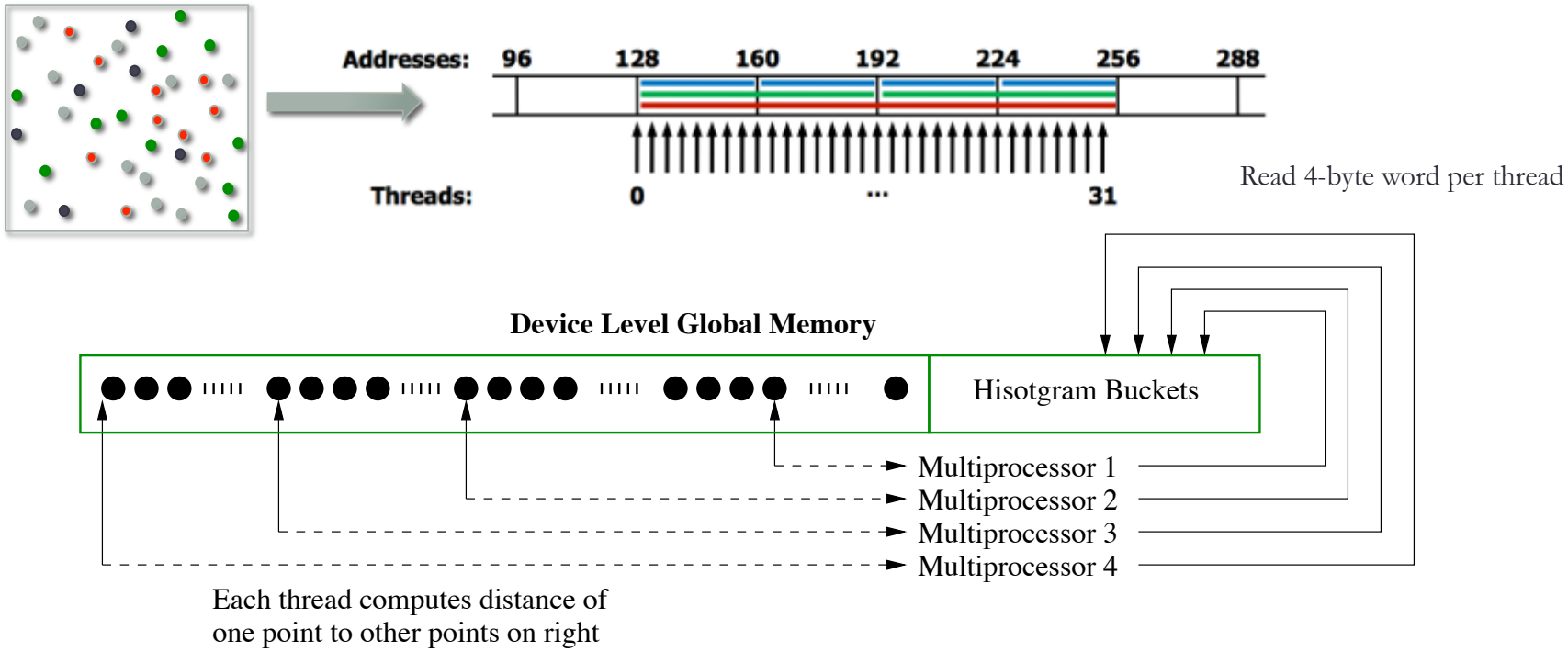
Overview

- Introduction and Motivation
- Spatial Distance Histogram (SDH)
- GPU Architecture
- Brute-force SDH Algorithm on GPU
- Advanced SDH Algorithms on GPU
- Conclusions and Future Work

Brute-force SDH on GPU – Naïve Method

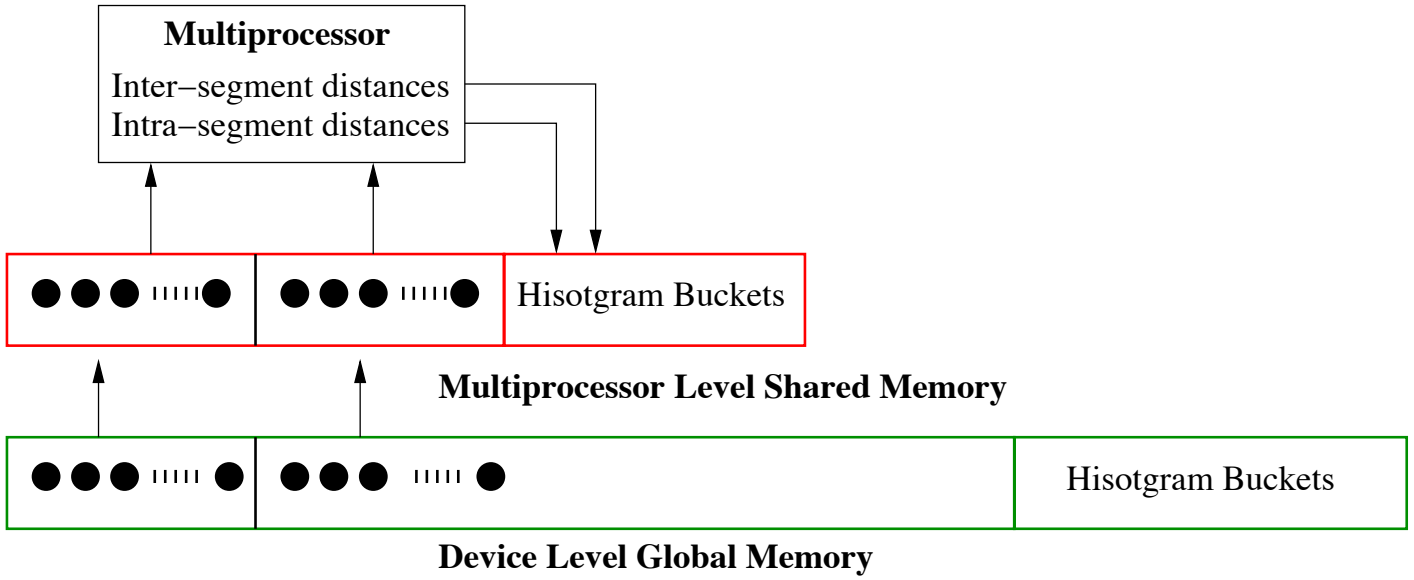
- Load all *N* points into GPU global memory
- Compute all distances and generate histogram

```
for each point p
        Compute distance with all other N-1 points
```



SDH on GPU – Shared Memory Method

- To deal with:
  - High access latency of global memory
  - Histogram buckets are updated by all threads -> serialized writes into buckets



Experimental Results

- GeForce GTX TITAN
  - CUDA Driver Version / Runtime Version: 6.0 / 5.0
  - CUDA Capability Major/Minor version number: 3.5
  - (14) Multiprocessors x (192) CUDA Cores/MP: 2688 CUDA Cores
  - Total amount of global memory: 6144 Mbytes
  - Total amount of shared memory per block: 49152 bytes

- SDH of various number of data points
  - Histogram is assumed to fit in portion of shared memory
  - All data is assumed to fit in global memory

Experimental Results – Time

GPU running time Comparison

Data Points	CPU Main Memory	Shared Memory	Global Memory
100,000	424.7 (7m)	1.54	3.40
300,000	3812 (1h)	11.39	26.42
800,000	27142 (7h)	77.20	177.92
1,600,000	1.5 Day	316.01	699.16
3,200,000	Days	1317.21	2771.21
6,400,000	Days	5335.71	11036.78

Experimental Results – Speedup

GPU Speedup Comparison with respect to CPU

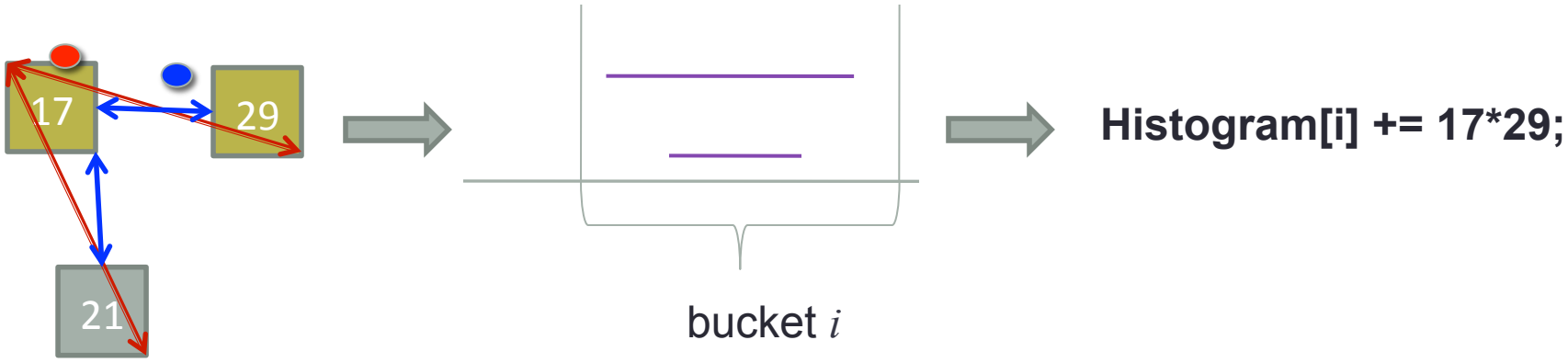
Data Points	Shared Memory	Global Memory
100,000	275.7	124.9
300,000	334.6	144.2
800,000	351.5	152.5
1,600,000	410.1	185.3
3,200,000		
6,400,000		

Overview

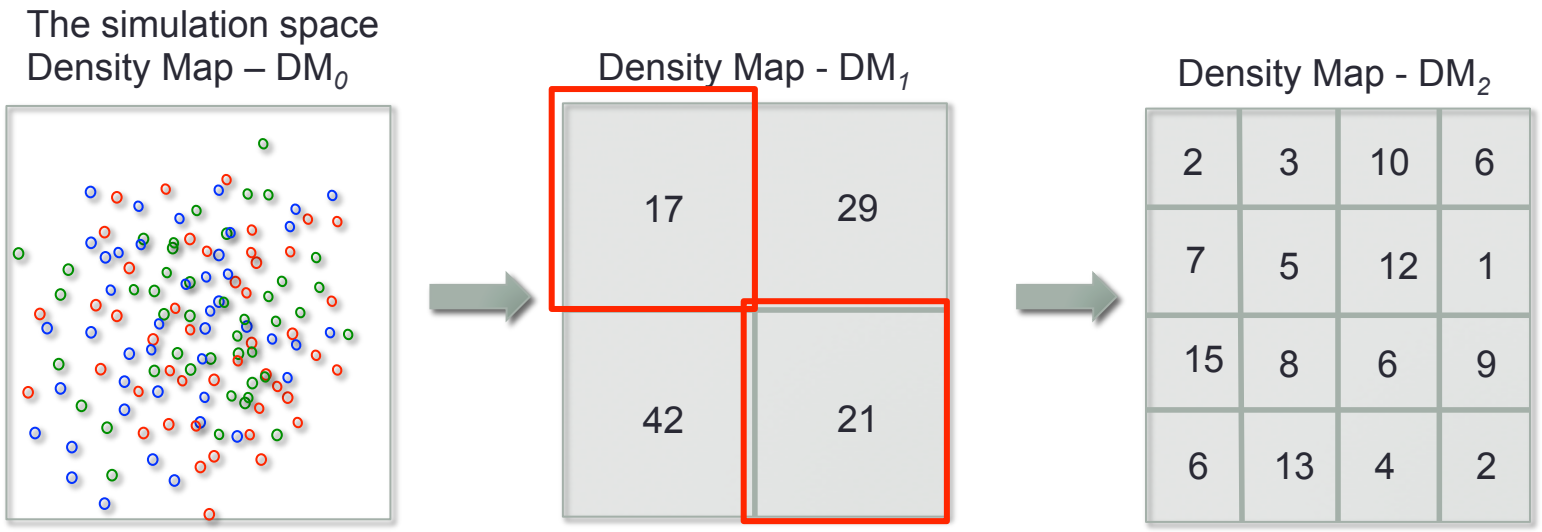
- Introduction and Motivation
- Spatial Distance Histogram (SDH)
- GPU Architecture
- Brute-force SDH Algorithm on GPU
- Advanced SDH Algorithms on GPU
- Conclusions and Future Work

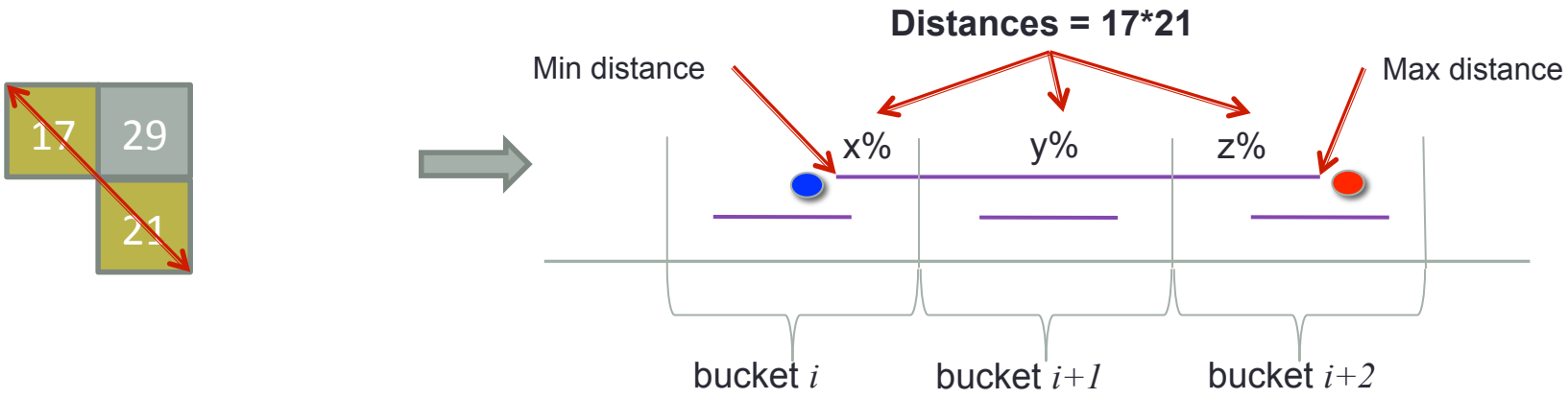
Background of DM-SDH

- Main idea: avoid the calculation of pairwise distances
- Observation:
  - two groups of points can be processed in one shot if the range of all inter-group distances falls into a histogram bucket
  - We say these two groups are *resolved* into that bucket



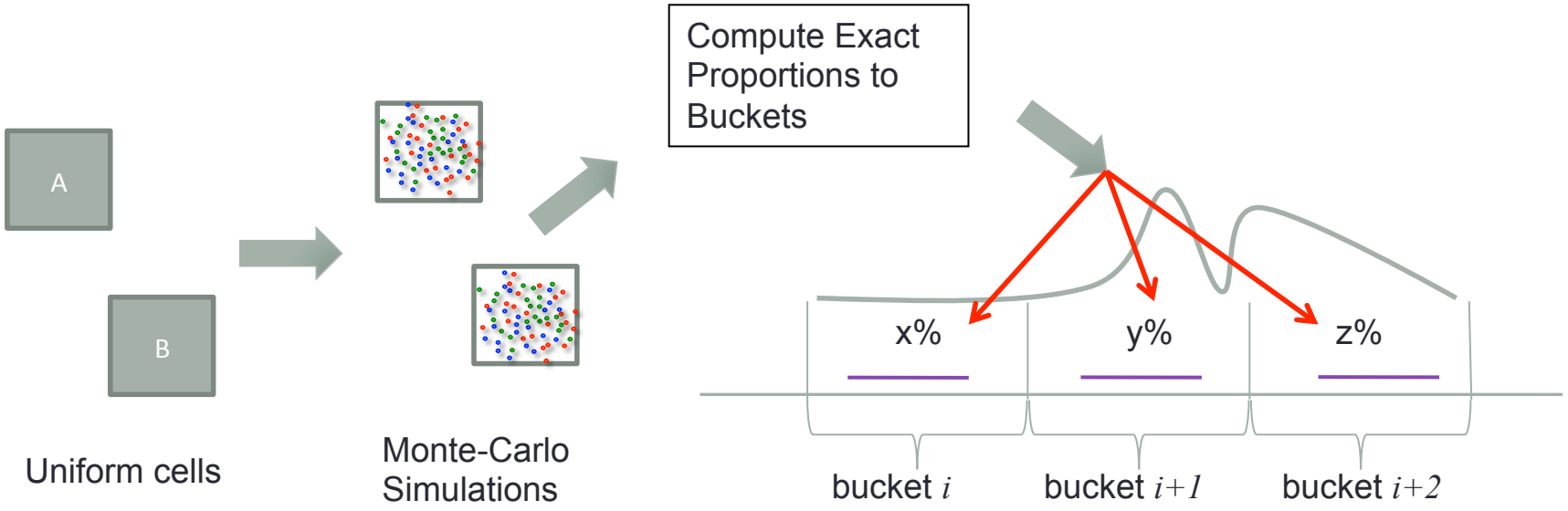
Background of DM-SDH





Spatial Uniformity

- Closed from PDF of distances – approximate PDF is derived
- Monte-Carlo simulation is another approximation to PDF

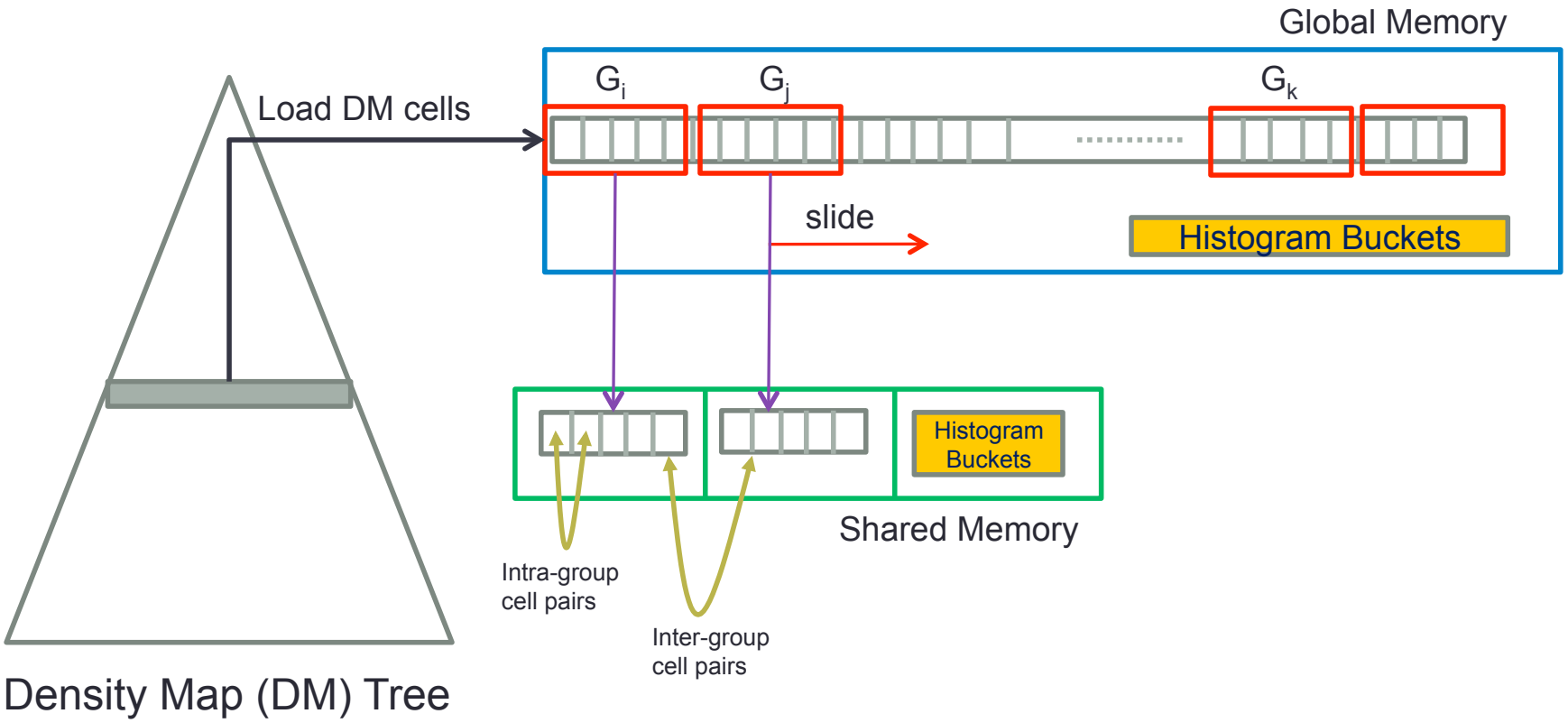


Theorem: Number of distinct Monte-Carlo simulations performed for all pair of cells in density map of *M* cells is *O(M)*

DM-SDH on GPUs

- Load all density maps on global memory.
- Shared memory is loaded by threads in a CUDA block
  - Each thread loads information about one cell
- Each thread processes a pair of cells
  - Replacing only one cell in loop until all cell are processed
  - Next a new cell is loaded and paired with distinct other cells

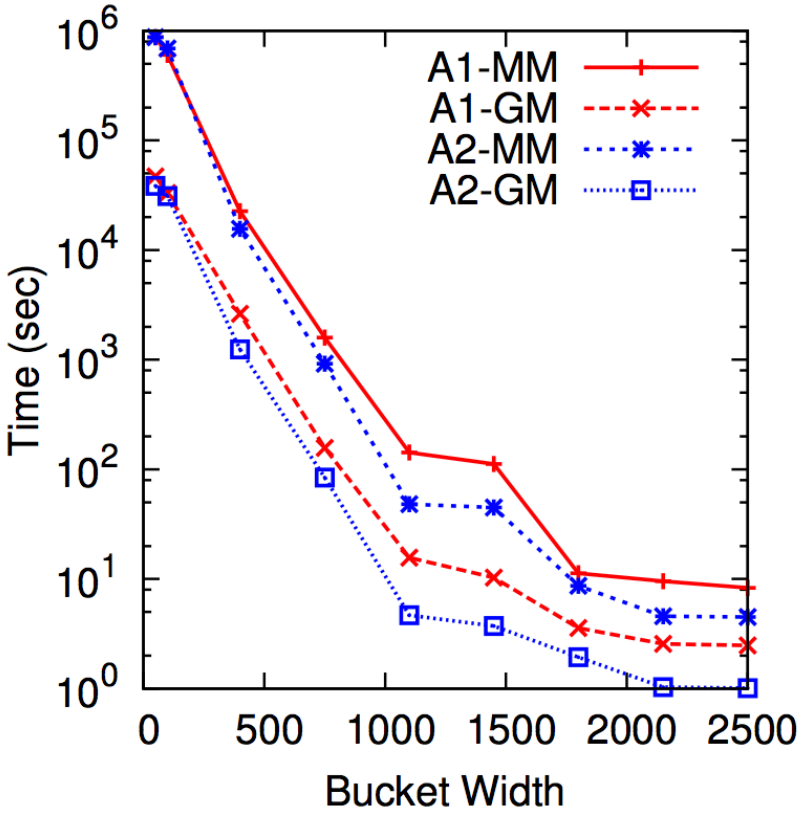
DM-SDH Algorithm on GPUs

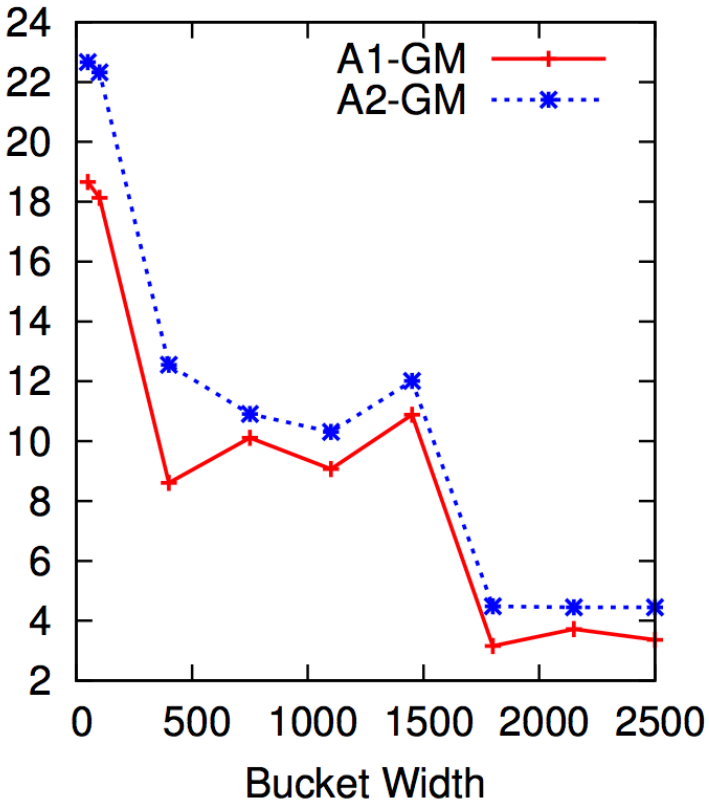


DM-SDH Algorithm on GPUs

- Each CUDA thread
  - Processes a pair of cells from DM
- If pair of cells contain uniformly distributed points
  - Threads in each CUDA block perform Monte-Carlo simulations
  - The probability distribution function for histogram buckets is hashed
- Now for each pair of cells
  - If they have uniformly distributed points
    - Retrieve information from hash and update histogram buckets
  - Else
    - Either compute pair-wise distance of all pairs of points in both cells by naïve way
    - Or use some heuristics

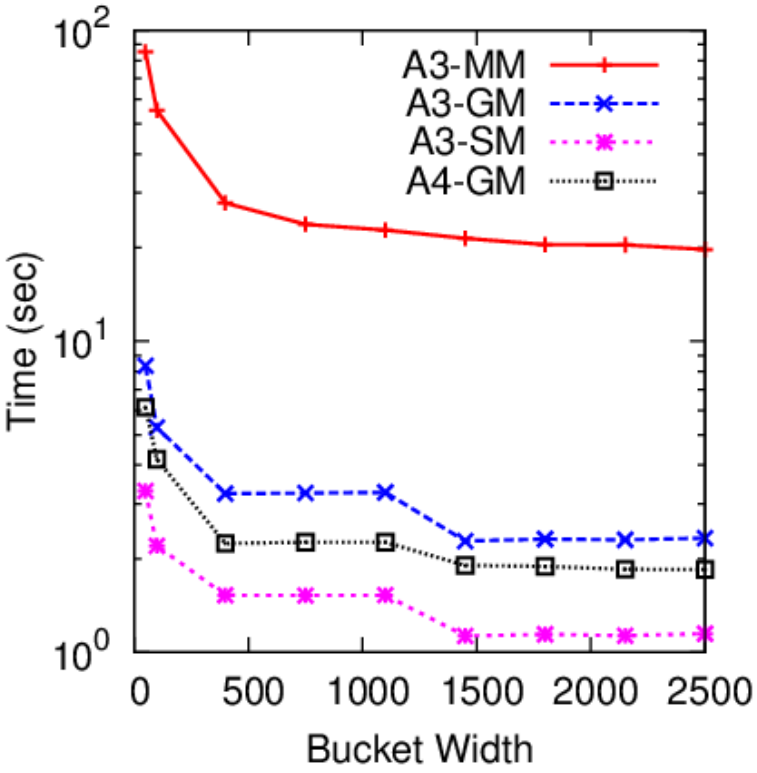
Experimental Results – Performance

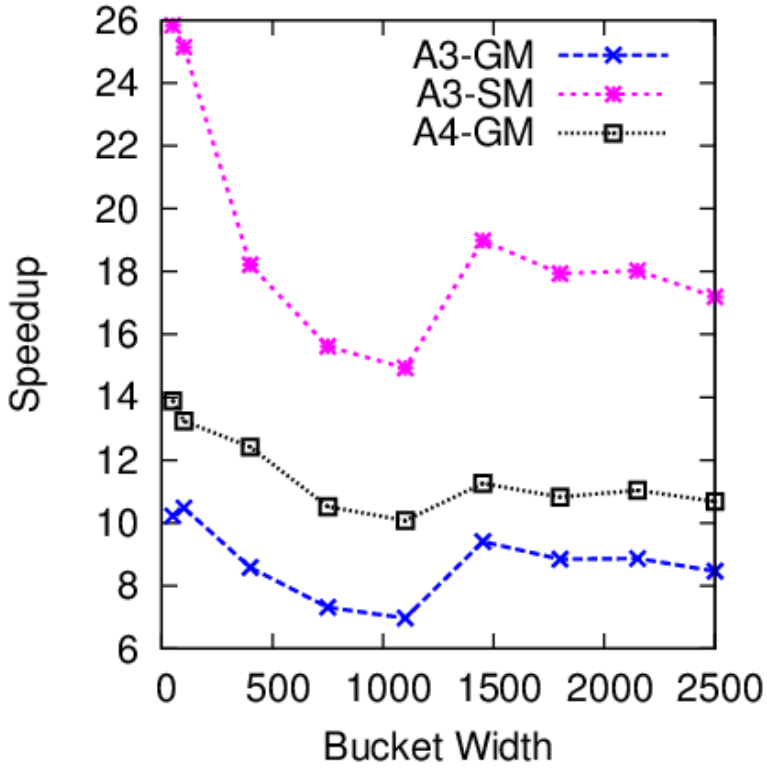




MM: CPU version, GM: GPU version

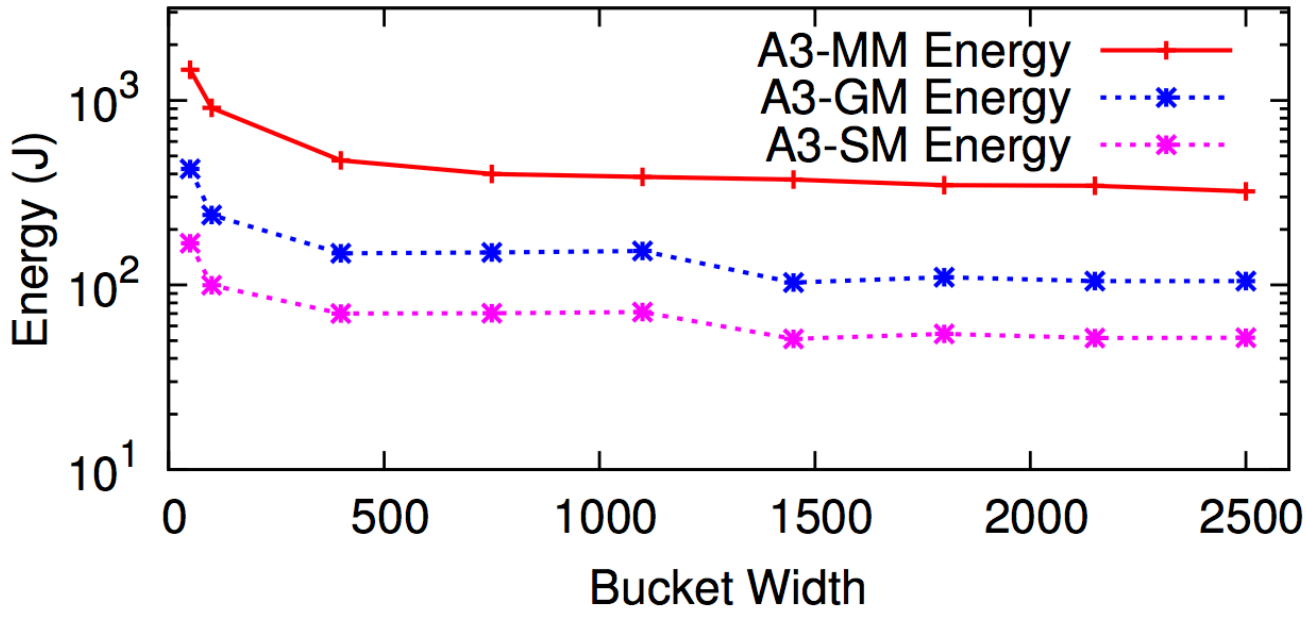
Experimental Results – Performance





MM: CPU version, GM: GPU Global Memory, SM: GPU Shared Memory

Experimental Results - Energy Consumption



DM-SDH vs. Brute-force

- Brute-force algorithm more suitable for GPU deployment
- Main problems for DM-SDH on GPUs:
  - Code divergence (i.e., tree traverse)
  - Size of a cell is much larger than size of a point

Overview

- Introduction and Motivation
- Spatial Distance Histogram (SDH)
- GPU Architecture
- Brute-force SDH Algorithm on GPU
- Advanced SDH Algorithms on GPU
- Conclusions and Future Work

Conclusions and Future Work

- Computing power of GPUs can be harnessed for SDH processing
- Speedup varies (3X – 400X) with the algorithm implemented and problem parameter

- Take advantage of new GPU/CUDA features
  - Shuffle instruction
  - Large register pool
  - Read-only cache
- Extension to other correlation functions

Relevant Publications

- [TKDE14] A. Kumar, V. Grupcev, Y. Yuan, **Y. Tu**, Jin Huang, and G. Shen. Computing Spatial Distance Histograms for Large Scientific Datasets On-the-fly. To appear in *IEEE Transactions on Knowledge and Data Engineering* (*TKDE*).
- [TKDE13] V. Grupcev, Y. Yuan, **Y. Tu**, Jin Huang, S. Chen, S. Pandit, and M. Weng. Approximate Algorithms for Computing Distance Histograms with Accuracy Guarantees. *IEEE Transactions on Knowledge and Data Engineering* (*TKDE*) 25(9):1982-1996, September 2013.
- [VLDBJ11] S. Chen, **Y. Tu**, and Y. Xia. Performance Analysis of A Dual-Tree Algorithm for Computing Spatial Distance Histograms. *The VLDB Journal.* 20(4): 471-494, August 2011.
- [EDBT12] A. Kumar, V. Grupcev, **Y. Tu**, Y. Yuan, and G. Shen. Distance Histogram Computation Based on Spatiotemporal Uniformity in Scientific Data. In Procs. of *15th IEEE International Conference on Extending Database Technology* (*EDBT*). pp.288-299, Berlin, Germany, March 26-30, 2012.
- [ICDE09] **Y. Tu**, S. Chen, and S. Pandit. Computing Distance Histograms Efficiently in Scientific Databases. In *Procs. of 25th International Conference on Data Engineering* (*ICDE*), pp. 796-807, Shanghai, China, March 2009.
- [NIPS00] Gray, A.G., Moore, A.W. N-body problems in statistical learning. In *Advances in Neural Information Processing Systems* (*NIPS*), 1408 pp. 521–527, MIT Press (2000)

THANK YOU

Questions ?

Background of DM-SDH

- The Approximate Algorithm (ADM-SDH)
  - Organize all data into a Quad-tree (2D data) or Oct-tree (3D data).
  - Cache the atoms counts of each tree node (cell)
    - Density map: all counts in one tree level

```
start from one proper density map M0
FOR every pair of cells A and B in M0
        resolve A and B
        IF A and B are not resolvable
        THEN IF at desired density map level
                THEN distribute distances proportionally
                        into overlapping buckets
                ELSE FOR each child cell A' of A
                        FOR each child cell B' of B
                                resolve A' and B'
```

SDH on GPU – Using Registers

- GPU with compute capability 3.0 and up support the _shift__ instruction.
- Load large number of registers with distinct point
- Compute the SDH by shifting one set of register
  - While keeping another set constant
  - All pairs of distances between the two sets will be computed

SDH on GPU – Using Registers