Parallelization and Optimization of 4D Binary Mixture Monte Carlo Simulations using Open MPI and CUDA

Sergey Artemchuk and Paula A. Whitlock

Brooklyn College/CUNY

Many thanks to the High Performance Computing Center, College of Staten Island/CUNY for a generous grant of computer time

and

Marvin Bishop, Manhattan College

Why mixtures of hyperspheres in 4 dimensions?

- Hard sphere interactions govern most fluid and solid systems.
- Mathematically, the number of lattices increases as the dimensionality increases.
- Networks where each node has more than three connections are modeled by multidimensional representations.
- Networks that have different sized, non-interacting objects in them can be modeled as mixtures.

The Metropolis Monte Carlo Method applied to hard hypersphere systems

Must sample the Boltzmann distribution function

$$f(\mathbf{R}) = \frac{exp[-\sum \phi(r_{ij})/k_b T]}{\int exp[-\sum \phi(r_{ij})/k_b T] d\mathbf{R}} \tag{1}$$

R is the d-dimensional vector of coordinates of the centers of mass of the M hyperspheres and $\mathbf{r}_i = (x_{i1}, x_{i2}, \cdots, x_{id})$, $i = 1, \cdots, M$

k_b is Boltzmann's constant and T is the absolute temperature of the system.

The pair potential, $\phi(r_{ij})$ represents the interaction between two hyperspheres.

The sampling algorithm generates random walks

Propose a move from the current position of a hypersphere, $\mathbf{X}$, to a new position, $\mathbf{X}'$ chosen from a probability distribution function, $H(\mathbf{X}' \mid \mathbf{X})$.

The new position is accepted or rejected based on the probability $p(\mathbf{X}' \mid \mathbf{X})$.

A recursive relationship develops between the distribution functions, $f_n(\mathbf{R})$, represented by each step of the random walk.

As long as the system is ergodic and obeys detailed balance, $f_n(\mathbf{R}) \rightarrow f(\mathbf{R})$ is guaranteed to be true as n, the number of passes, becomes large.

The network where the Monte Carlo simulations are executed

- 48 8 core SGI X3481U host machines with Intel Core 7i - Nehalem
- Each host has 2 Nvidia Fermi GPUs.
- Hosts are connected by a 40 Gbit/sec QDR infiniband.

FERMI GPU Schematic

Functional Unit Foot-Print of *Data*-Oriented PTX ISA FERMI PTX ISA and micro-architecture, 512 (448)

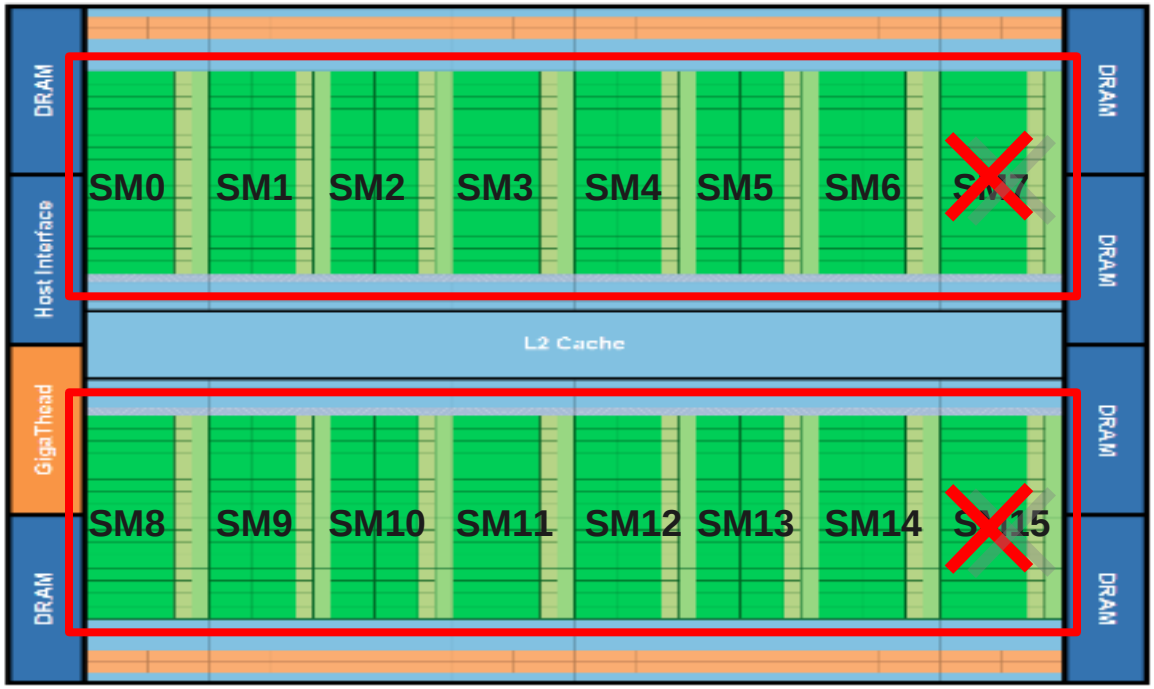
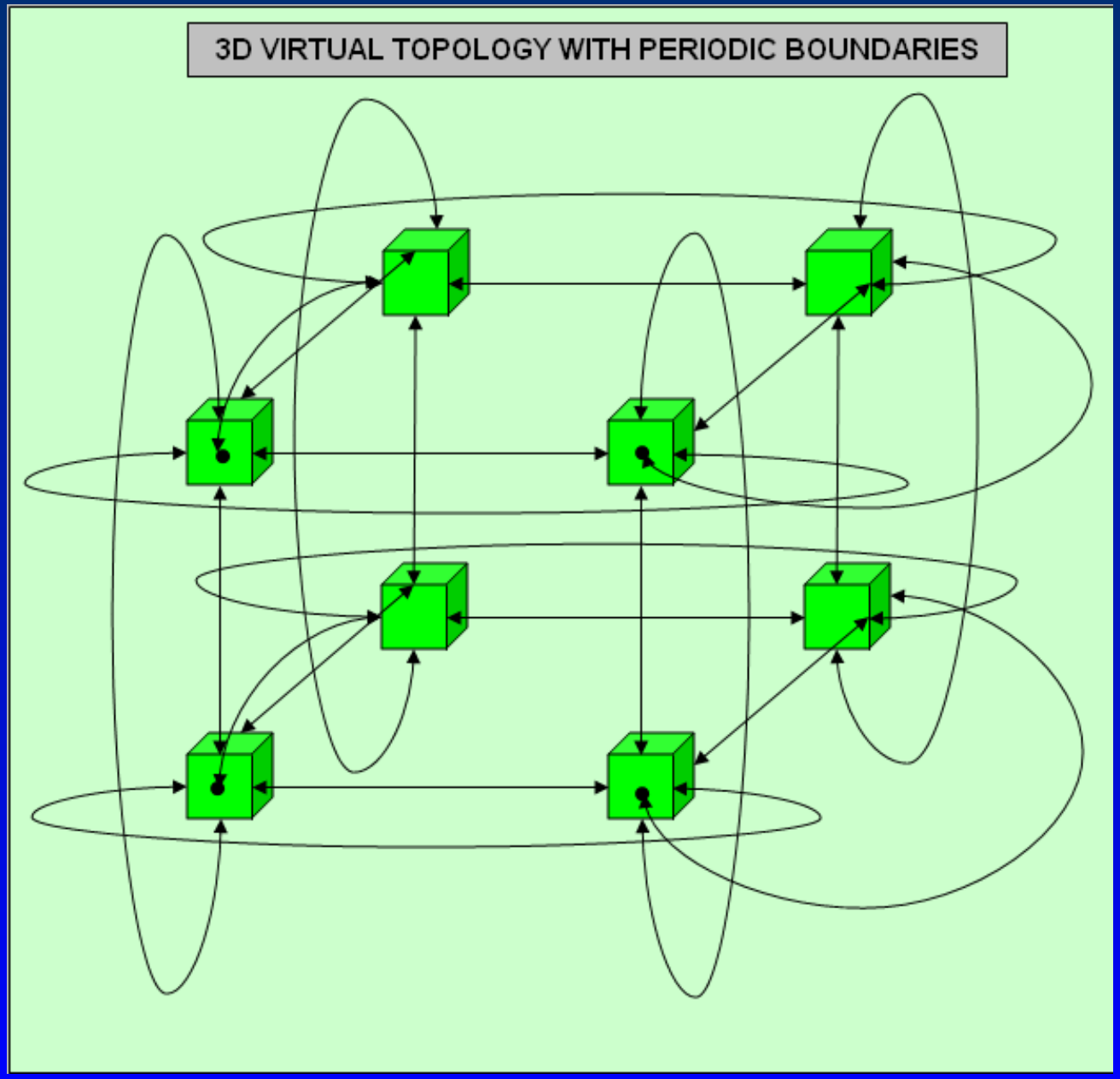
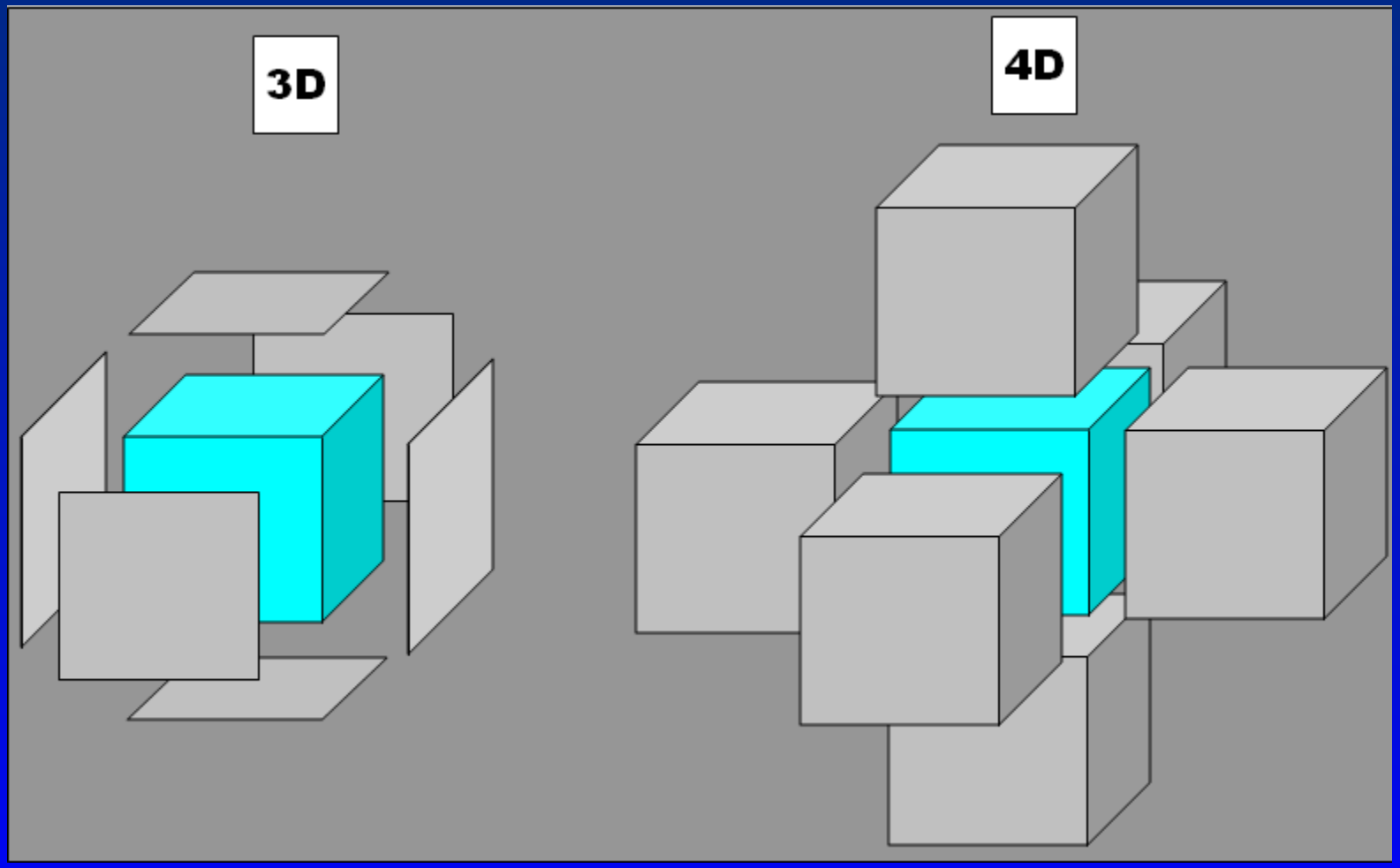


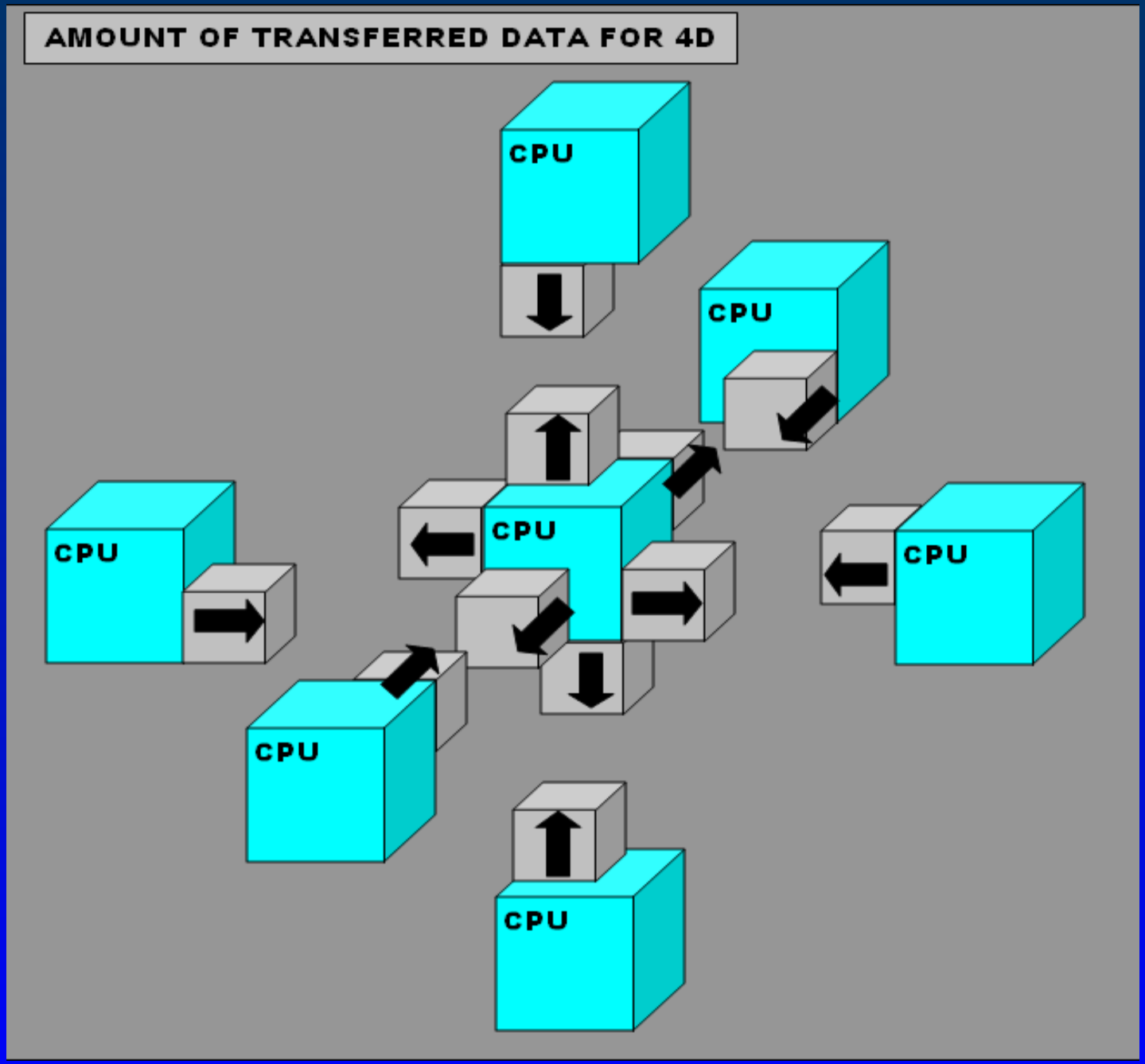
Figure 5. NVIDIA's Fermi GPU architecture consists of multiple streaming multiprocessors (SMs), each consisting of 32 cores, each of which can execute one floating-point or integer instruction per clock. The SMs are supported by a second-level cache, host interface, GigaThread scheduler, and multiple DRAM interfaces. (Source: NVIDIA)

Virtual Topology

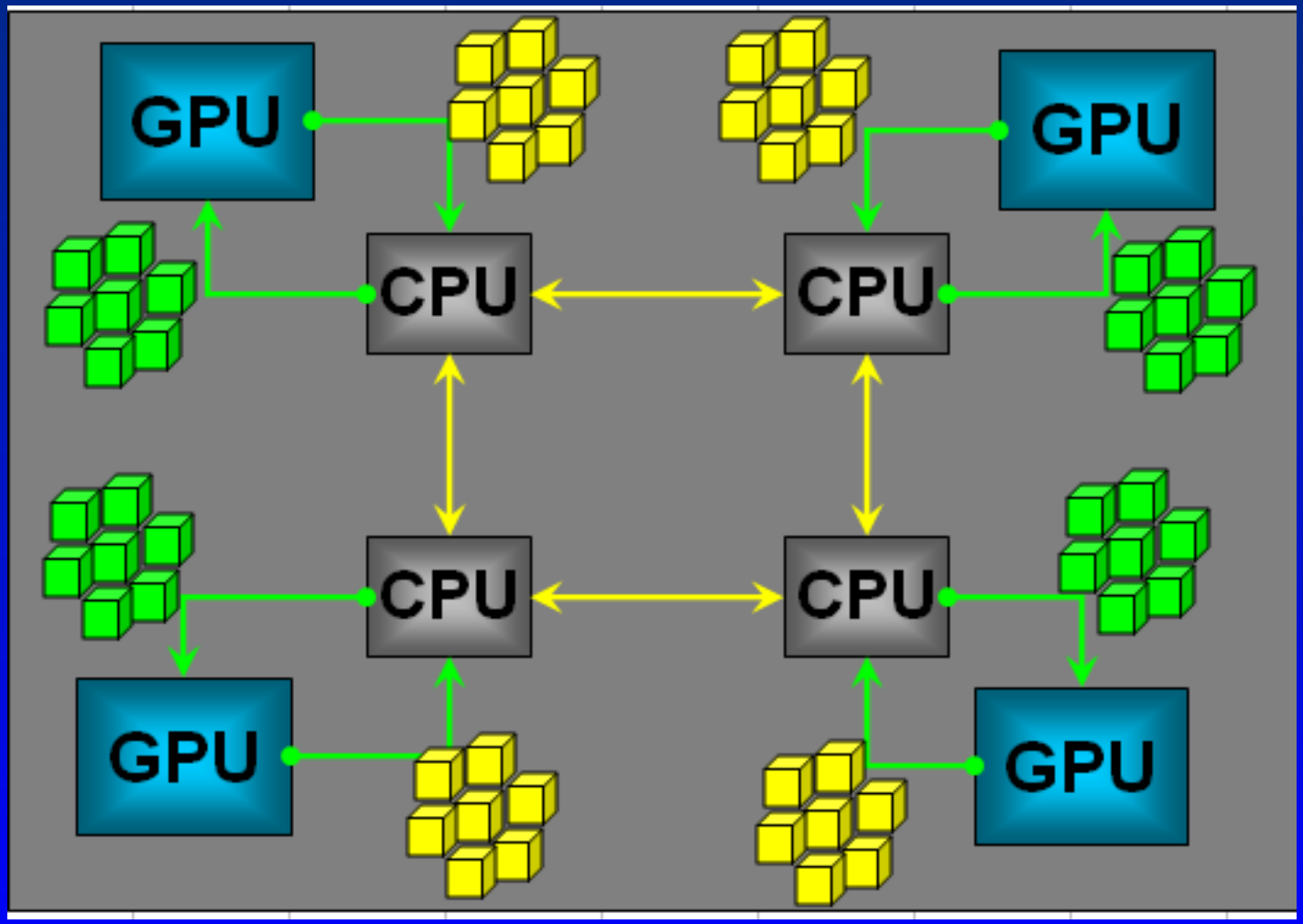


Amount of data that needs to be transferred in 4D vs 3D

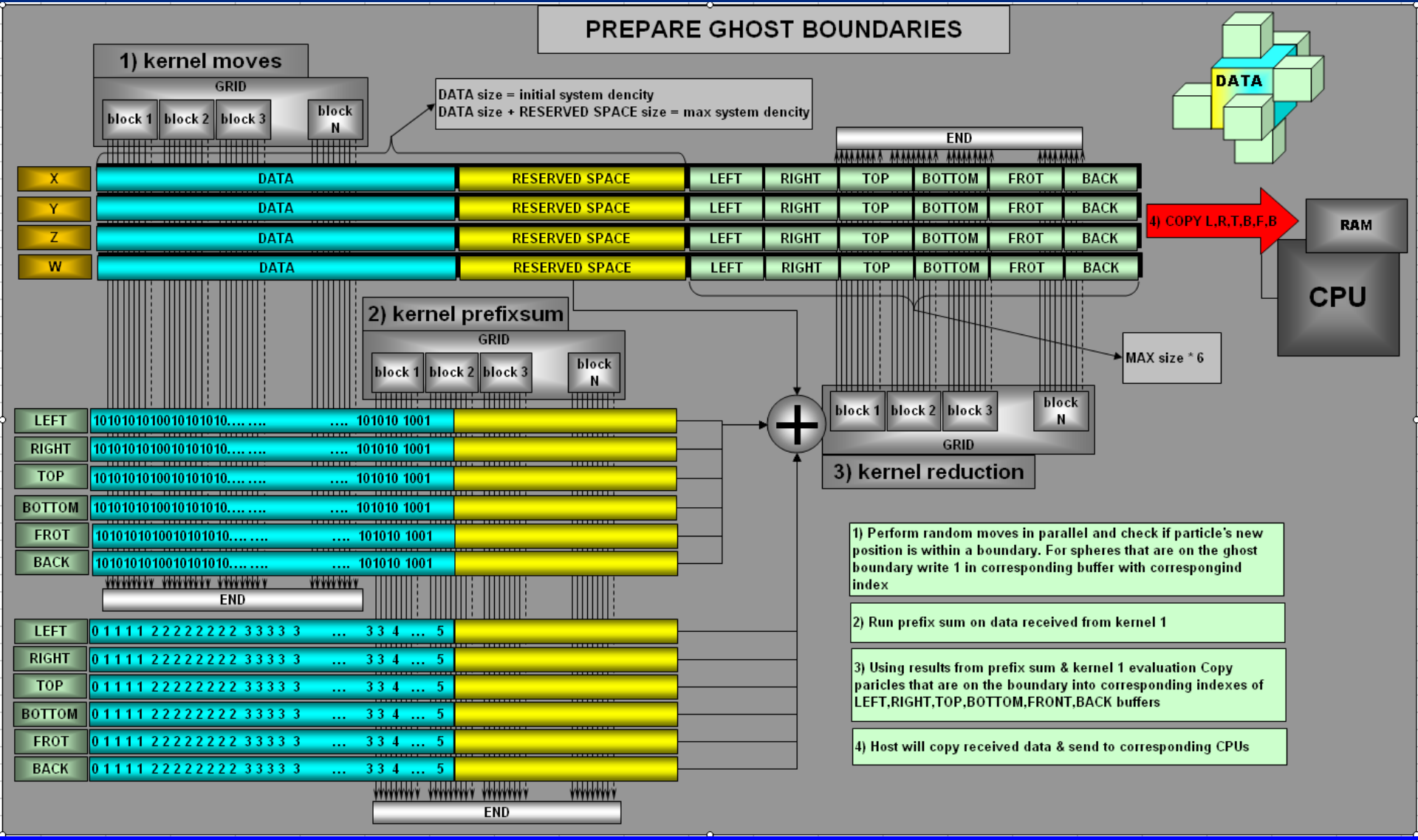



AMOUNT OF TRANSFERRED DATA FOR 4D
CPU
CPU
CPU
CPU
CPU
CPU
CPU

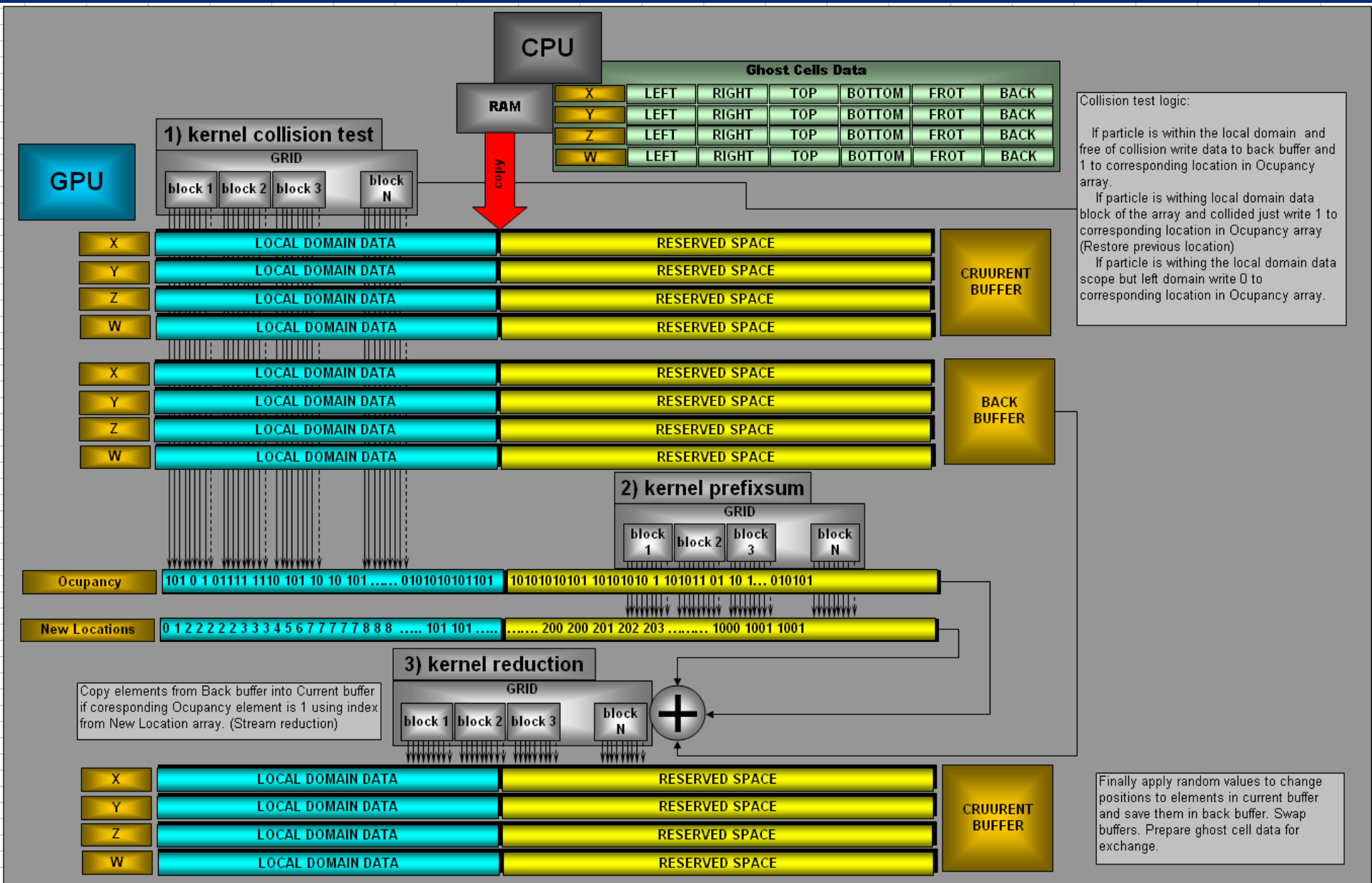
Data must be communicated to the GPUs as well as the CPUS on the network



Ghost Cells

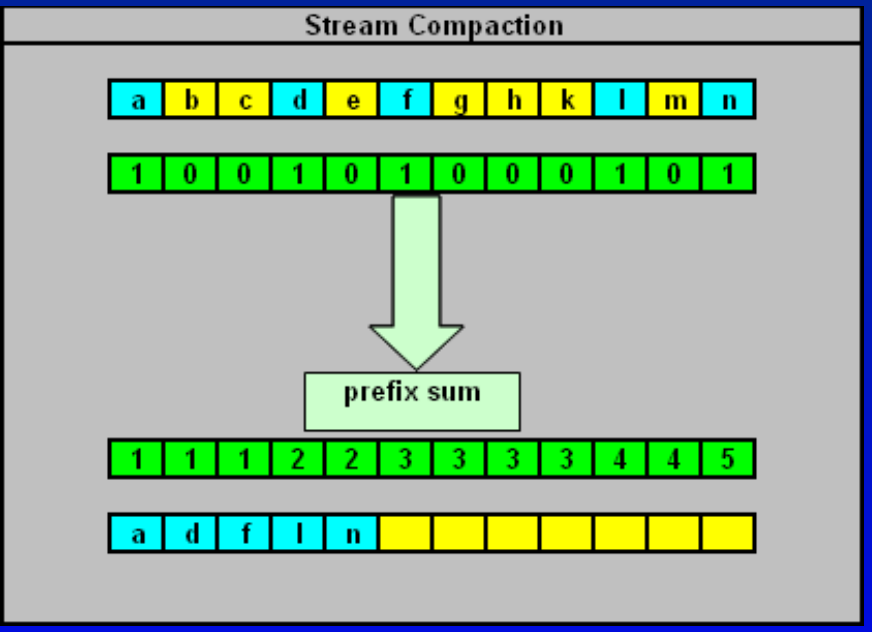

PREPARE GHOST BOUNDARIES
1) kernel moves
GRID
block 1
block 2
block 3
block N
DATA size = initial system dencity
DATA size + RESERVED SPACE size = max system dencity
DATA
END
X
Y
Z
W
DATA
RESERVED SPACE
LEFT
RIGHT
TOP
BOTTOM
FROT
BACK
4) COPY L,R,T,B,F,B
RAM
CPU
2) kernel prefixsum
MAX size * 6
LEFT
RIGHT
TOP
BOTTOM
FROT
BACK
101010101001010101010 101010 1001
END
0 1 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 5
3) kernel reduction
1) Perform random moves in parallel and check if particle's new position is within a boundary. For spheres that are on the ghost boundary write 1 in corresponding buffer with corresponginding index
2) Run prefix sum on data received from kernel 1
3) Using results from prefix sum & kernel 1 evaluation Copy paricles that are on the boundary into corresponding indexes of LEFT,RIGHT,TOP,BOTTOM,FRONT,BACK buffers
4) Host will copy received data & send to corresponding CPUs

Collision Test

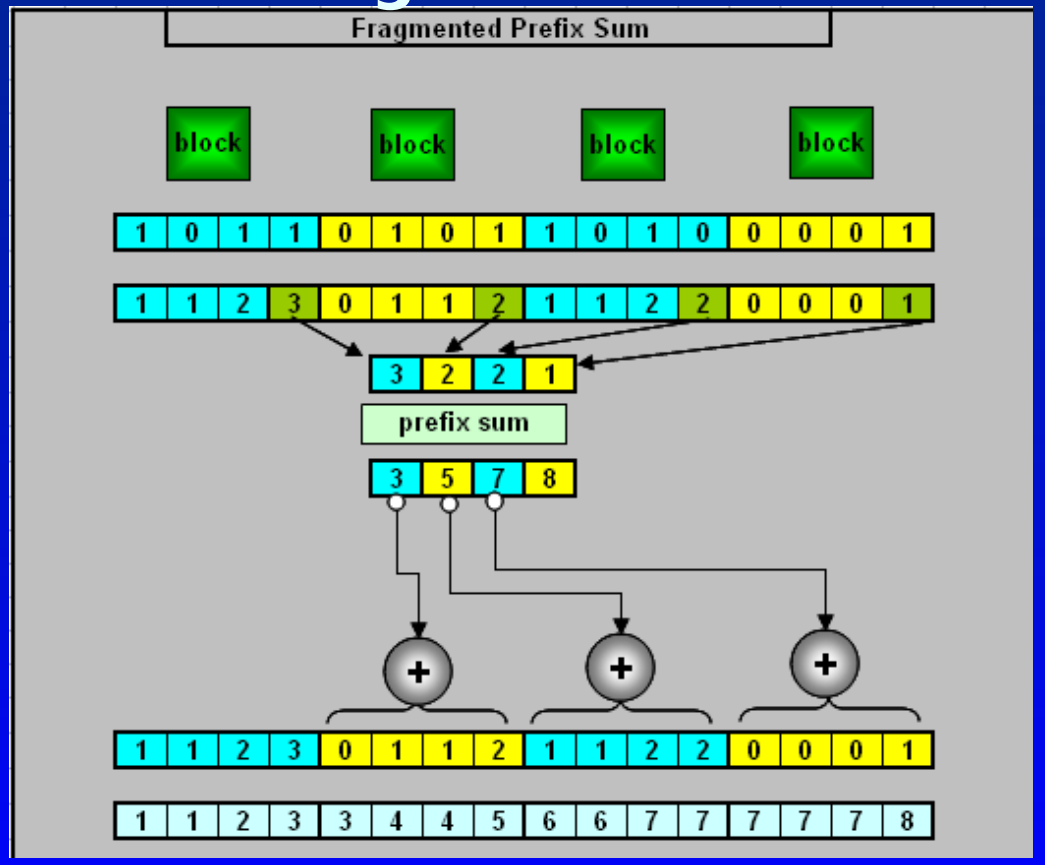


Prefix Sum (stream compaction) implementation with CUDA

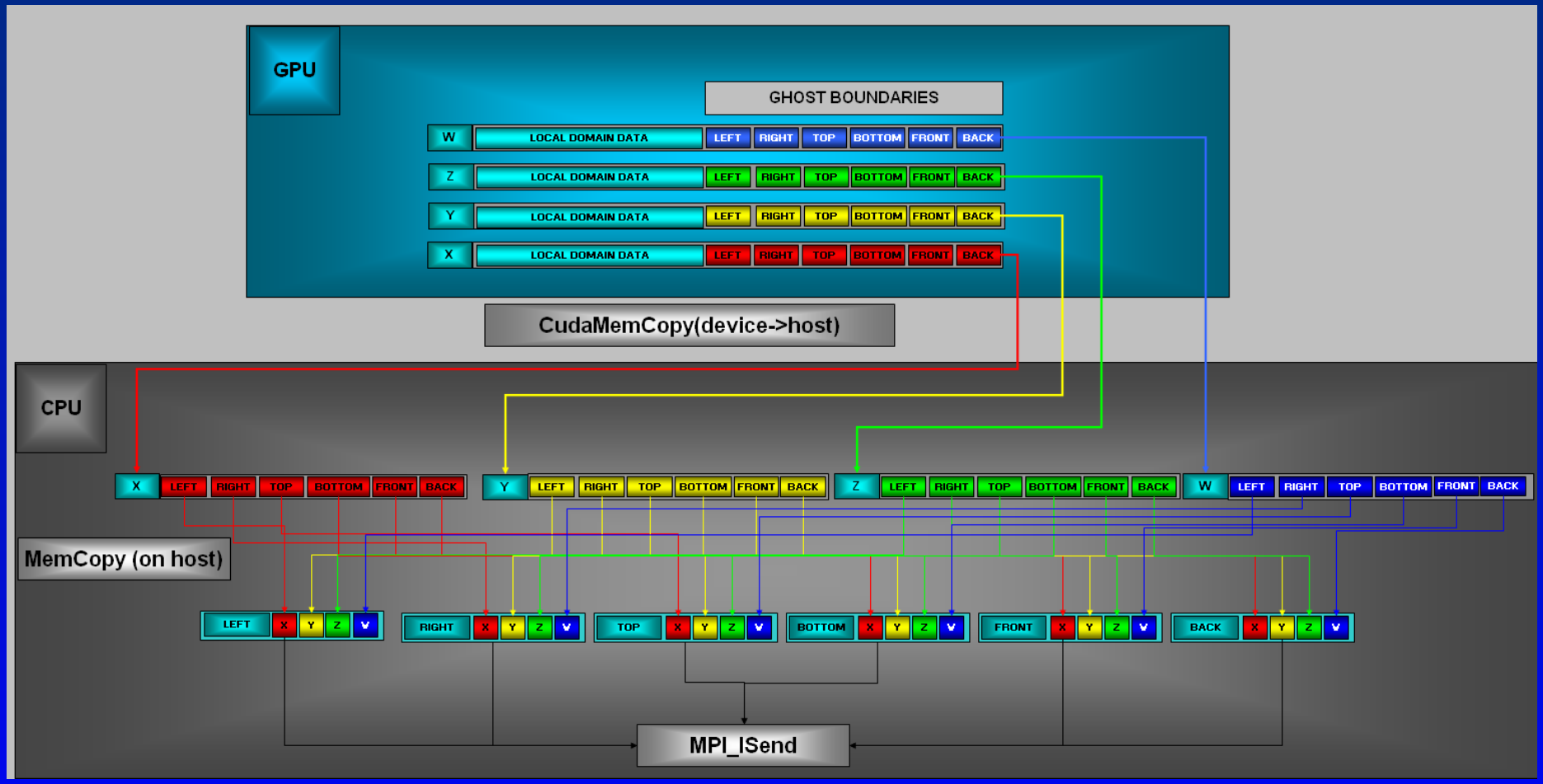
Example



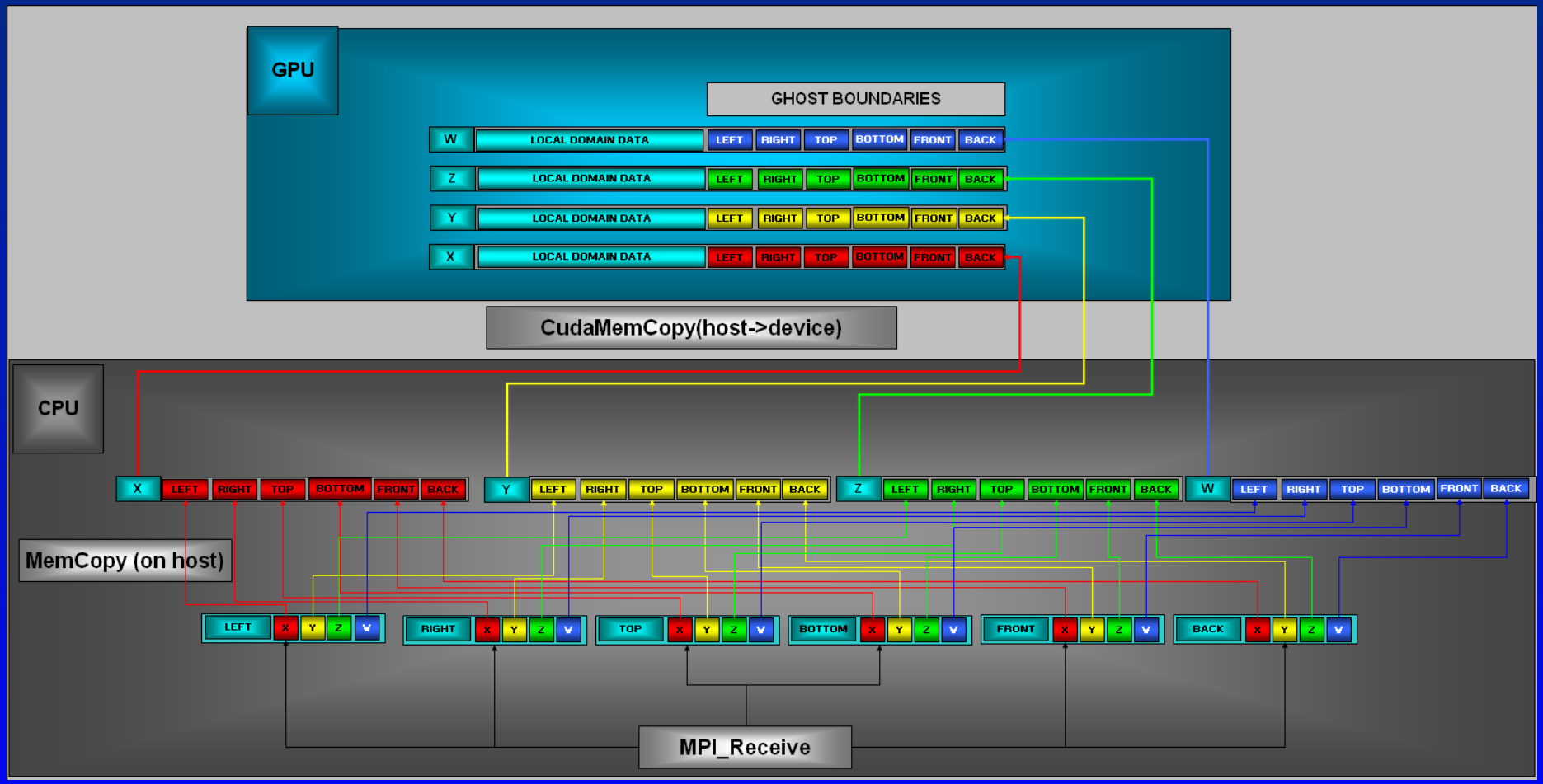
GPU block size limitation requires array fragmentation



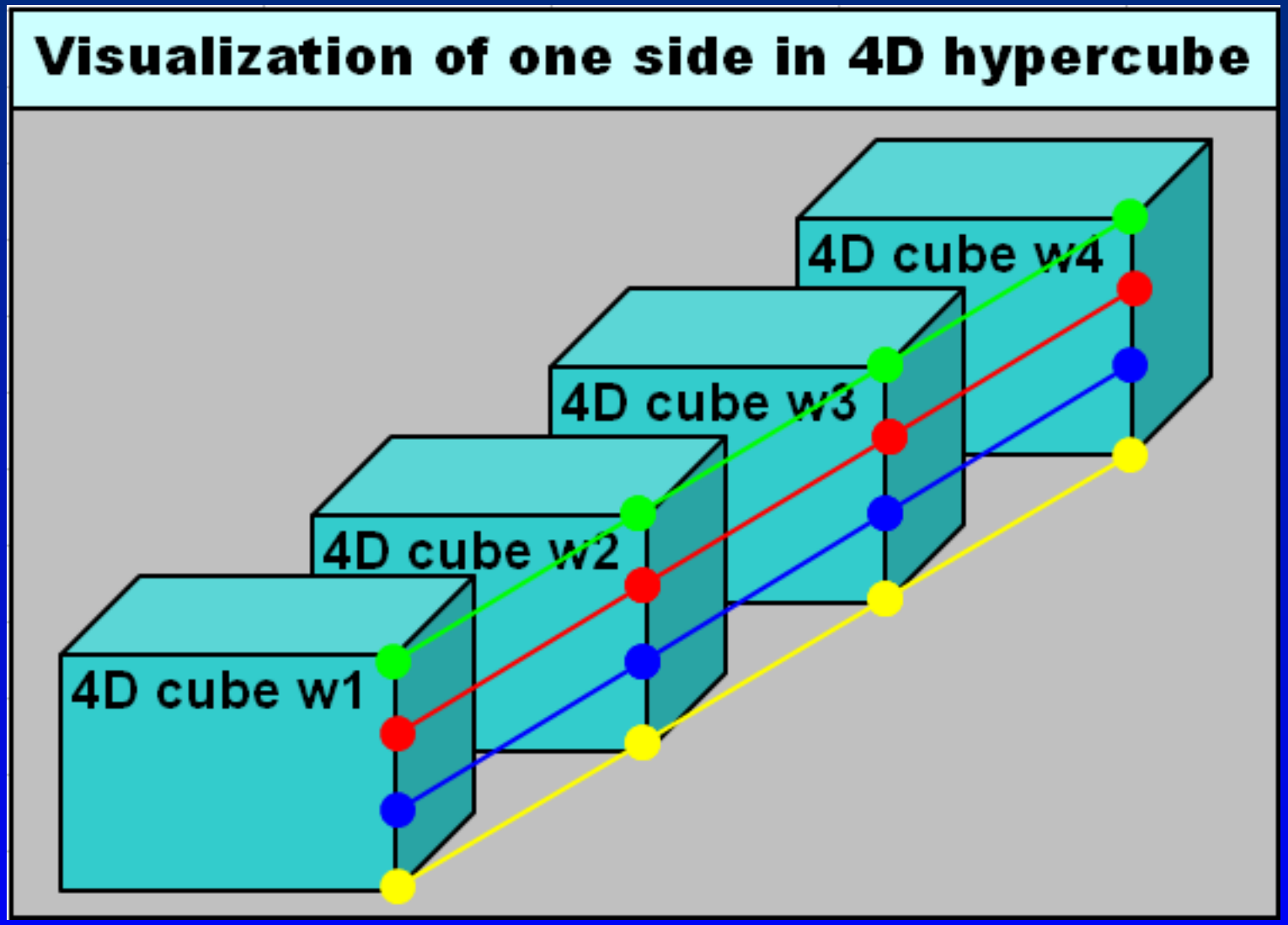
Copy ghost cells data from GPU to CPU



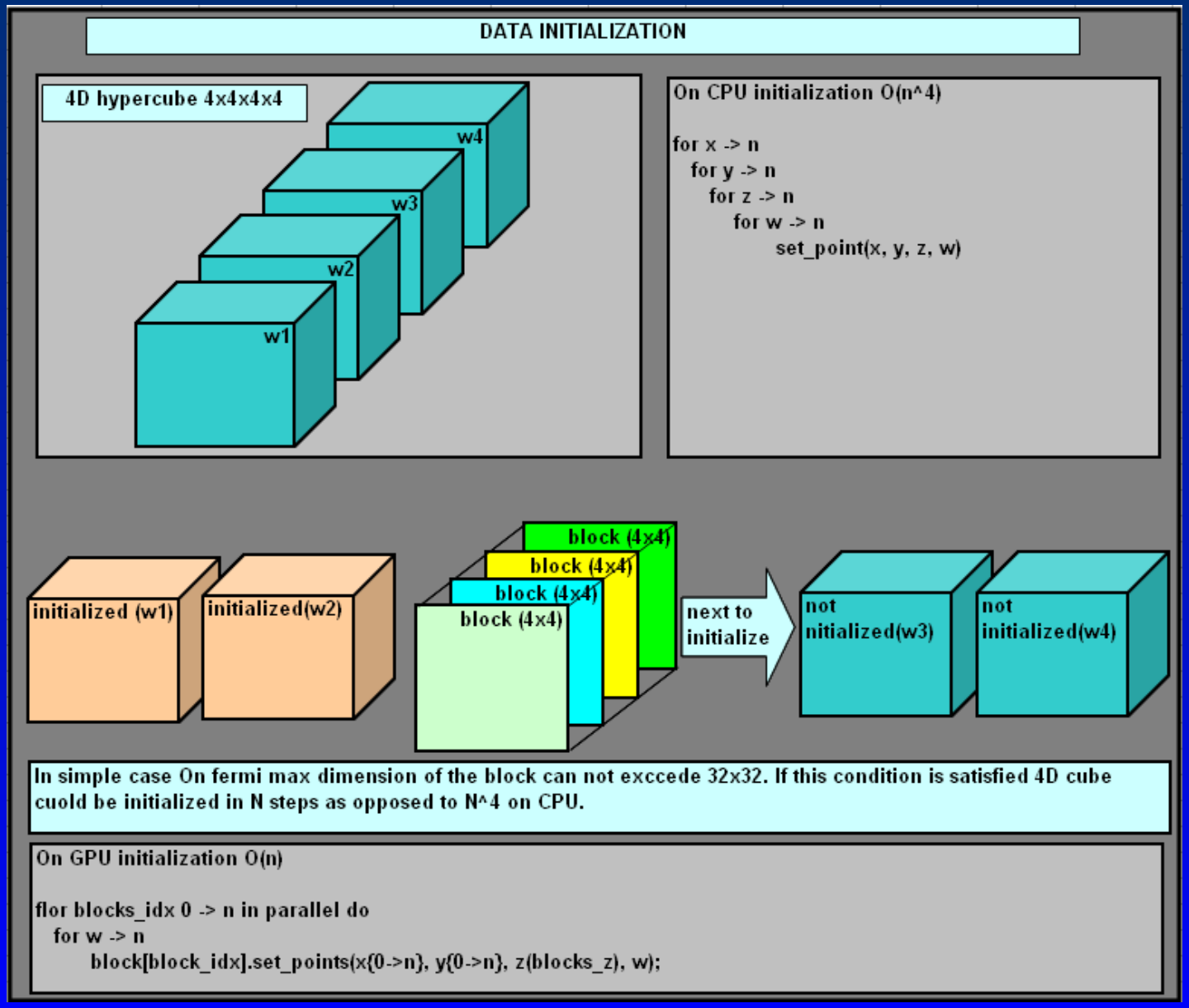
Copy ghost cells data form CPU to GPU



4D space visualization (parallel mirrors)

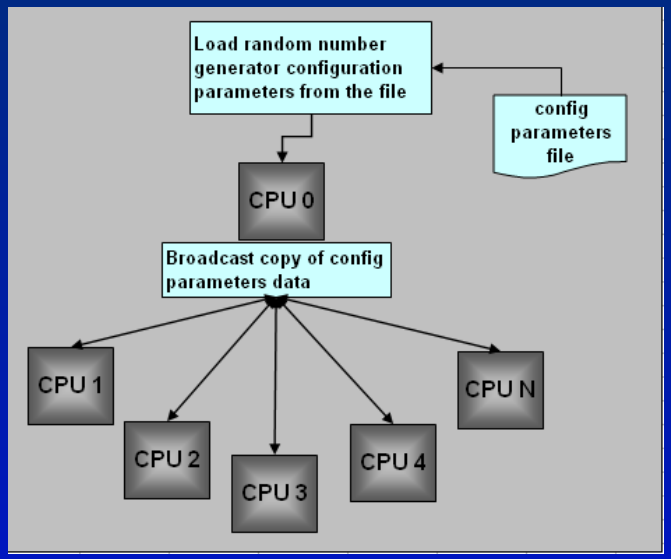


Initialization

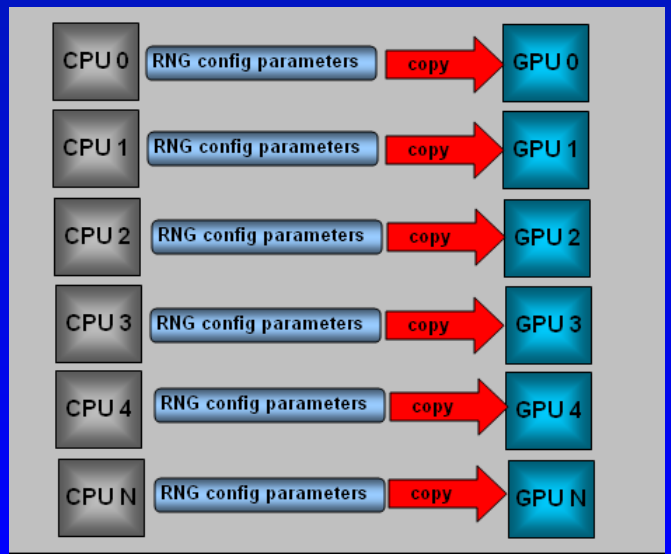


Distributing the Mersenne Twister pseudorandom number generator over the network.

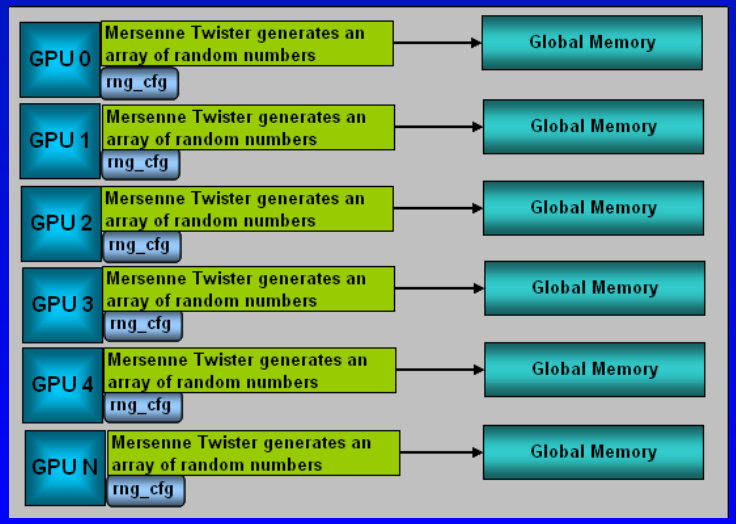
Step 1



Step 2



Step 3



Results from running multiple threads on a single CPU/GPU pair

Number of hyperspheres		256	512	1024
Computer code	# of threads	time(s)	time(s)	time(s)
C program on host	1	6.5	26.73	109.82
Cuda code on GPU	1	75.56	296.81	1196.32
Cuda code on GPU	32	5.37	17.38	65.88
" " " "	64	3.09	9.54	34.11
" " " "	128	1.96	5.78	18.74
" " " "	256	1.32	3.77	12.42
" " " "	512	--	2.8	8.91

Simulation distributed over a network of 8 cpu/gpu pairs. Each subdomain is 2X2X2.

System Size	# of subdomains per side	IO Time (seconds)	Total Time (seconds)
4X4X4X4	2	5.174	8.588
6X6X6X6	3	14.825	20.413
8X8X8X8	4	33.870	45.209
10X10X10X10	5	65.410	86.934
12X12X12X12	6	112.126	166.576
14X14X14X14	7	179.147	274.769
16X16X16X16	8	268.812	440.752
18X18X18X18	9	384.145	644.352

Conclusions

- Significant speedups in runtime were achieved by converting the code to run on the GPU
- Dividing configuration space into subdomains allows much larger systems, up to 104,976 hyperspheres, to be studied

Future Improvements

- Use streams to copy data between the CPU and GPU
- Use an optimized stream compaction algorithm:
  - http://gpgpu.org/developer/cudpp
- Using the warp vote functions would increase the rate of hypersphere overlap detection.