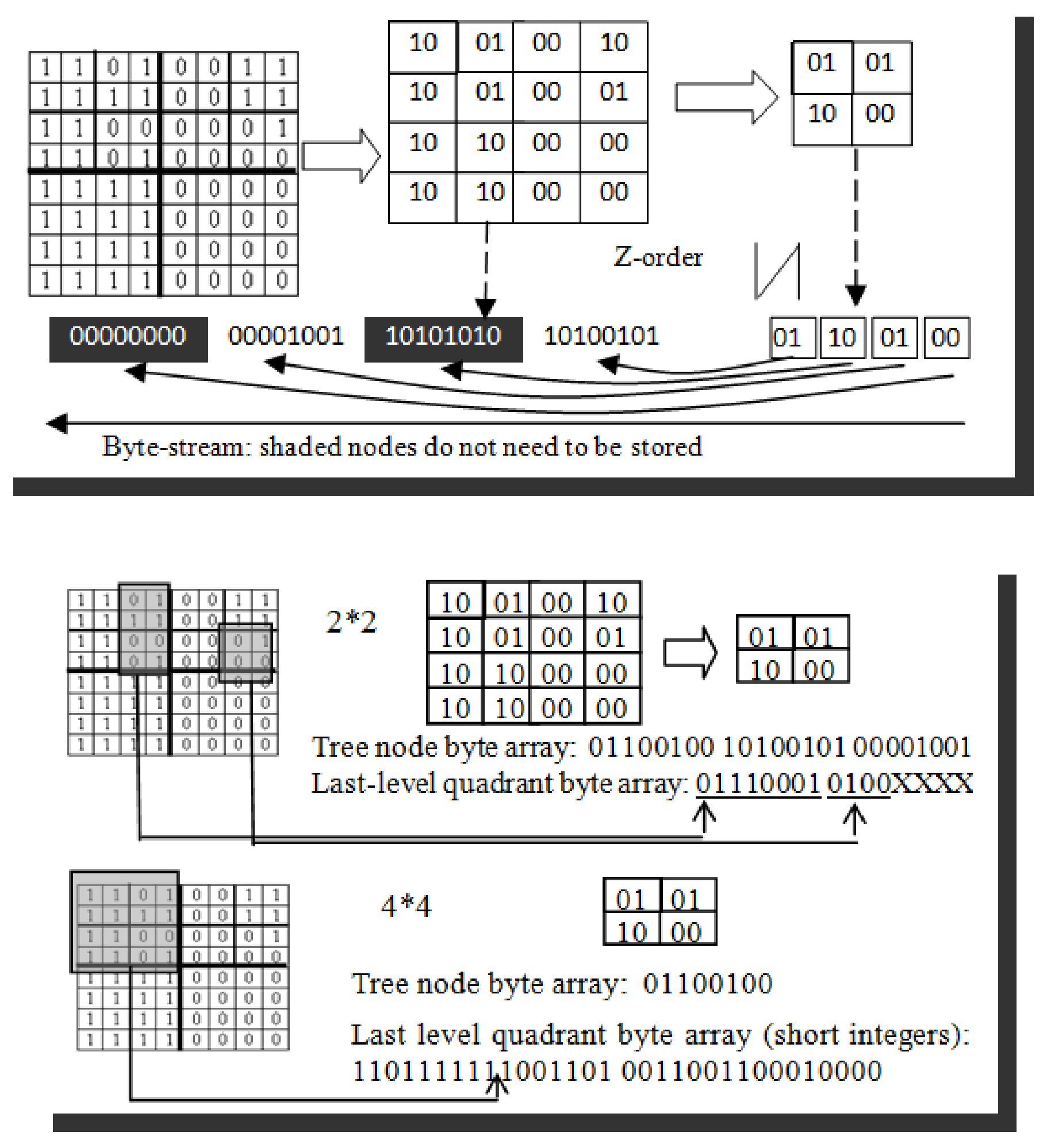
Parallel Quadtree Coding of Large-Scale Raster Geospatial Data on GPGPUs

Jianting Zhang[1], Simin You[2] and Le Gruenwald[3]
1 City College of New York 2 CUNY Graduate Center 3 University of Oklahoma

Abstract

Global remote sensing and large-scale environmental modeling have generated huge amounts of raster geospatial data. While the inherent data parallelism of large-scale raster geospatial data allows straightforward coarse-grained parallelization at the chunk level on CPUs, it is largely unclear how to effectively exploit such data parallelism on massively parallel General Purpose Graphics Processing Units (GPGPUs) that require fine-grained parallelization. In this study, we have developed an efficient spatial data structure called BQ-Tree to code raster geospatial data by exploiting the uniform distributions of quadrants of bitmaps at the bitplanes of a raster. A fine-grained parallelization scheme has been implemented using Nvidia CUDA. Experiments show that the GPGPU implementation is capable of decoding a BQ-Tree encoded 16-bits NASA MODIS geospatial raster with 22,658*15,586 cells in 190 milliseconds, i.e., 1.86 billion cells per second, on an Nvidia C2050 GPU card. The performance achieves a 5.9X speedup when compared with the best dual quadcore CPU implementation and a 36.9X speedup compared with a highly optimized single core CPU implementation.

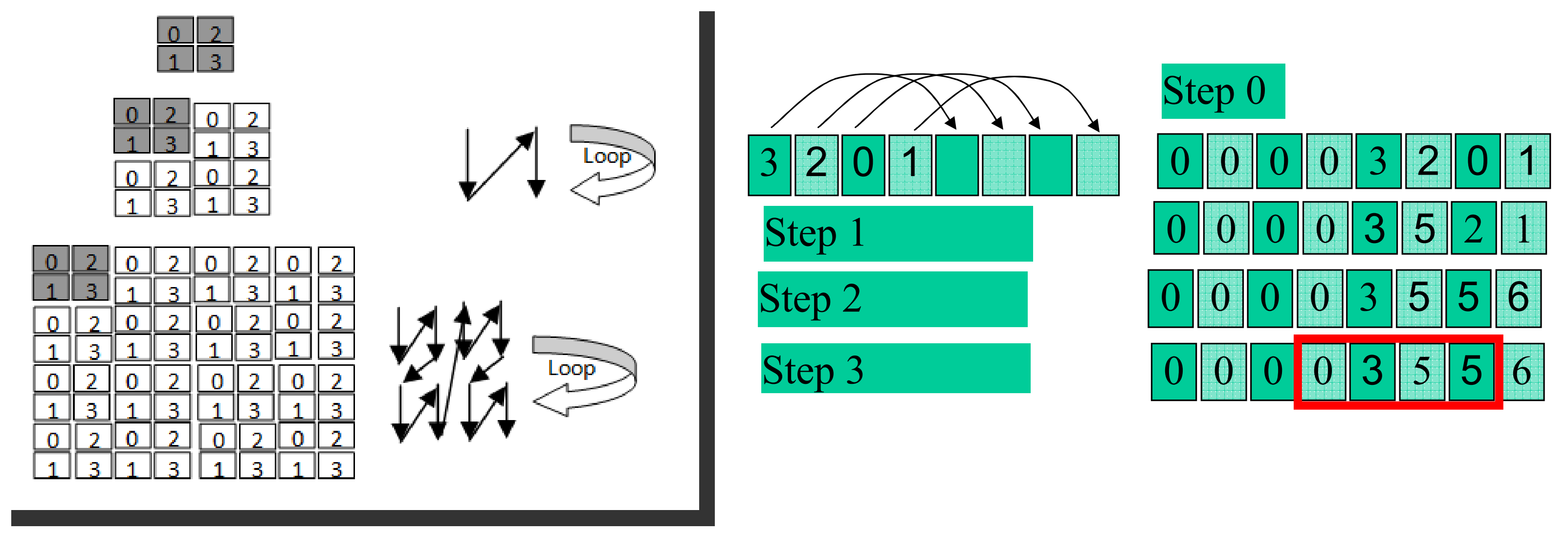
Structure of BQ-Tree


Z-order
Byte-stream: shaded nodes do not need to be stored
2*2
Tree node byte array: 01100100 10100101 00001001
Last-level quadrant byte array: 01110001 0100XXXX
4*4
Tree node byte array: 01100100
Last level quadrant byte array (short integers): 1101111111001101 0011001100010000

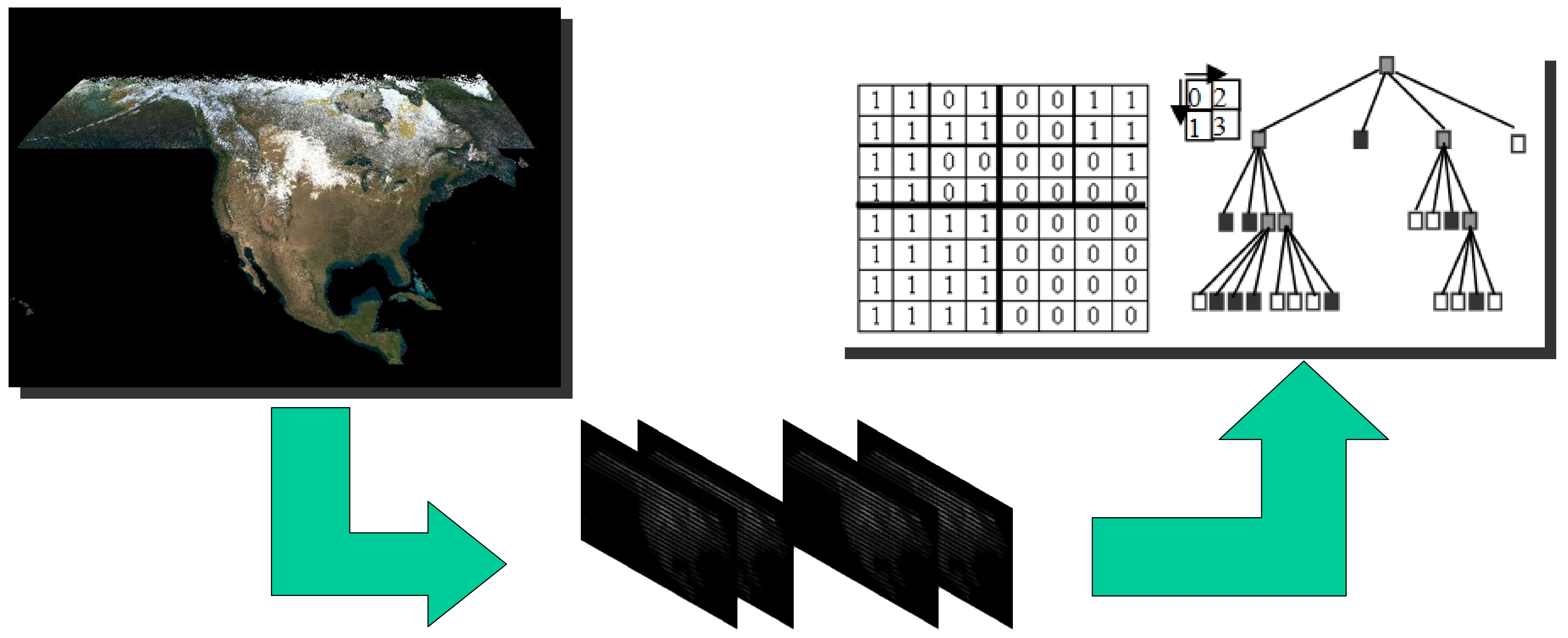
A BQ-tree is represented by the combination of two data streams, i.e., a quadtree index byte array and a last level quadrant signature byte array.

BQ-Tree Decoding on GPGPUs

1. All the threads assigned to a computing block are bundled together to process a quadrant of matrices in a BQ-Tree pyramid during decoding.
2. The collective process is looped over all the quadrants and all levels of the pyramid, i.e., Process Collectively and Loop (PCL).
3. The starting positions of the threads in the byte streams are calculated efficiently on the fly in GPGPU shared memories – only the starting position of the computing block needs to be pre-generated.



Proposed Solution

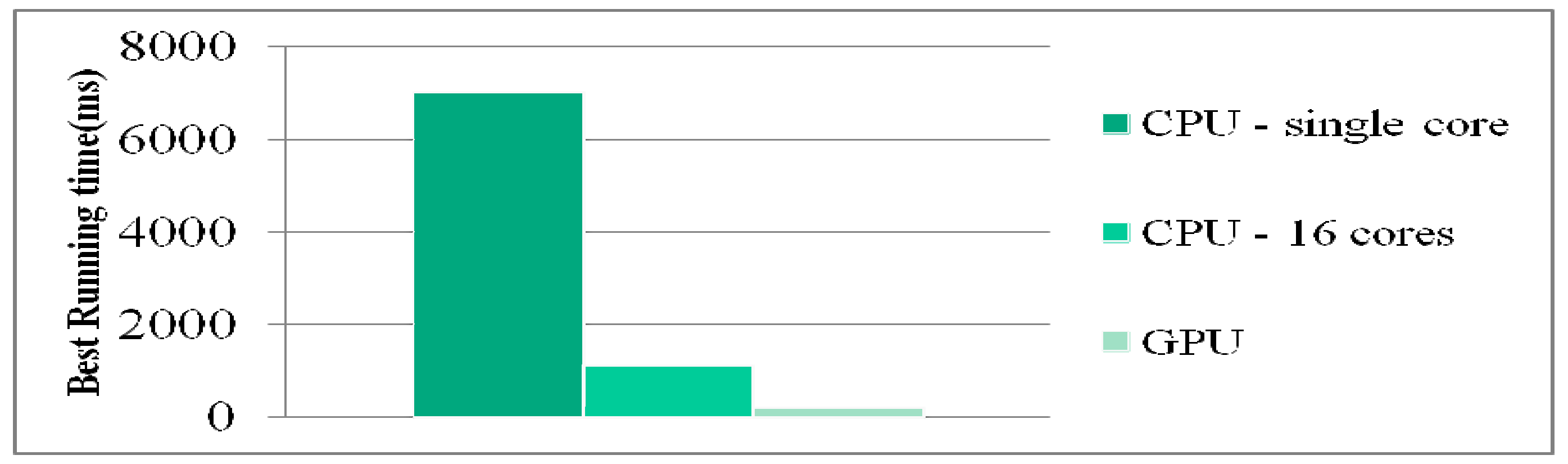


Coding geospatial rasters as sets of Bitplane Quadtrees (BQ-Tree)

- BQ-Tree is an efficient and cache-friendly data structure that is suitable for both CPUs and GPUs
- BQ-Trees can be used for data compression and query processing simultaneously
- Decoding can be accelerated using massively parallel GPGPU technologies

Experiments

Compressed Size				
Chunk Size	1024*1024		4096*4096	
Data Volume	771,751,936		805,306,368	
Last Level Quadrant Size	2*2	4*4	2*2	4*4
Node Array Size	131,277,689	35,700,341	131,276,652	35,699,304
LLQS Array Size	136,290,830	191,154,696	136,290,199	191,154,696
Total Encoded Size	267,568,519	226,855,037	267,566,851	226,854,000
Compression Ratio	34.67%	29.39%	33.23%	28.17%



Data: NASA MODIS band1 of the North America 2003097 imagery, 22,658*15,586 cells and 706,295,176 bytes

Hardware: SGI Octane III equipped with two Intel Xeon E5520 CPUs, 48 GB memory and two Nvidia C2050 GPUs (only one node is used)

Software: gcc 4.6.1 (with O3 for speed) on CPU and CUDA 4.0 toolkit on GPU

Results: (1) 3X compression ratio (2) **36.9X** speedup of GPGPU decoding when compared with single CPU thread (7005 ms/190 ms)

Future Work

- Integrate the BQ-Tree encoding engine with query processing frontends in databases (e.g., SciDB and FastBit)
- Develop an end-to-end visual analytical prototype system accelerated by GPGPUs for large-scale geospatial rasters