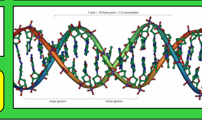
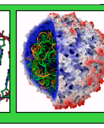
Accelerating Applications with Inter-Block GPU Communication via Fast Barrier Synchronization

SyNeRGy
http://synergy.cs.vt.edu/

Wu Feng and Shucai Xiao {wfeng, shucai}@vt.edu Virginia Tech





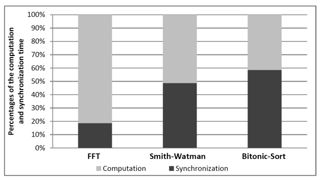
1. Motivation and Background

GPGPU: General-Purpose Computation for GPU

- The need to move from *data (or task) parallel computation* to *general-purpose computation?*

Current State of the Art in GPU Computing

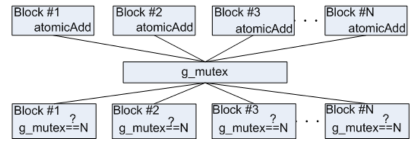
1. Focus on accelerating computation rather than inter-block GPU communication (via barrier synchronization).
2. Inter-block GPU communication for more general-purpose computation consumes a substantial portion of total execution time.
   - Current Approach
     Barrier synchronization across multiprocessors via kernel launch.



2. Lock-Based Approaches

Communication via GPU Simple Synchronization (GSS)

- *Implementation:* A global mutually-exclusive (mutex) variable that controls the execution of all blocks.
- *Properties*
  - ✓ Atomic add required
  - ✓ The more blocks in a kernel, the slower the inter-block synchronization time

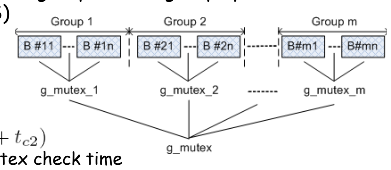


Communication via GPU Tree-Based Synchronization (GTS)

- *Implementation:* Blocks divided into groups. Each group synchronized with GPU simple synchronization (GSS)
- *Properties*
  - ✓ Atomic adds in different groups execute in parallel
  - ✓ Synchronization time

$$t_{GTS} = (\hat{n} \cdot t_a + t_{c1}) + (m \cdot t_a + t_{c2})$$

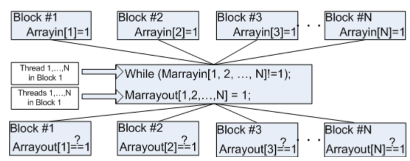
where t_a: atomic add time, t_c: mutex check time and m: group number, $\hat{n}$: block number per group



3. Lock-Free Approach

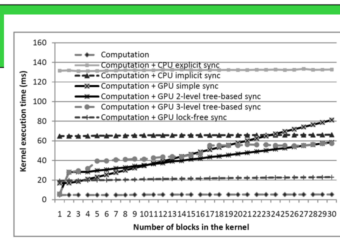
Communication via GPU Lock-Free Synchronization

- *Implementation:* Two arrays control inter-block GPU communication. Controlling threads in "Block #1" synchronize via __syncthreads().
- *Properties*
  - ✓ No need for atomic add
  - ✓ All operations executed in parallel
  - ✓ Constant synchronization time

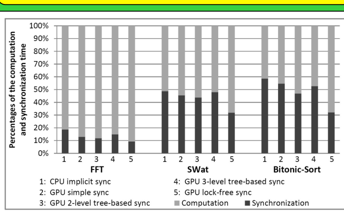


4. Performance Verification of GPU Communication via Barrier Synchronization

1. CPU implicit synchronization takes *12 times longer* than the computation time.
2. GPU-based barrier synchronization takes less time than both CPU implicit and explicit synchronization.
   - The execution time of GPU simple synchronization (GSS) varies linearly with the number of blocks.
   - The execution time of GPU lock-free synchronization is constant.
3. The empirical results match our performance models.



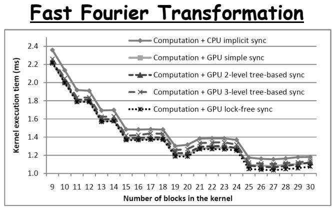
5. Performance Evaluation of GPU Communication: FFT, Smith-Waterman, Bitonic Sort



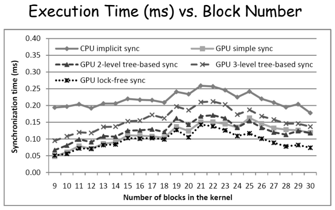
Percentage of Computation Time vs. Synchronization Time

1. Bitonic sort experiences a 38% improvement, Smith-Waterman 24%, and FFT 8%.
2. The percentage of time spent in synchronization is smallest for FFT and largest for bitonic sort.
3. The variances in synchronization time are consistent with our performance models.

Fast Fourier Transformation

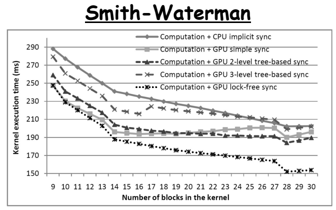


Execution Time (ms) vs. Block Number

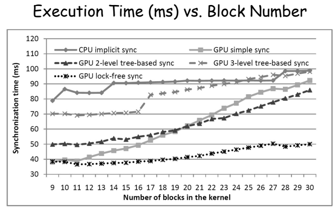


Synchronization Time (ms) vs. Block Number

Smith-Waterman

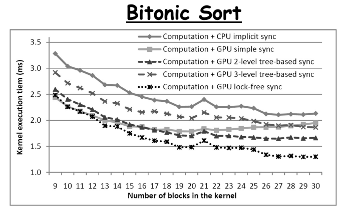


Execution Time (ms) vs. Block Number

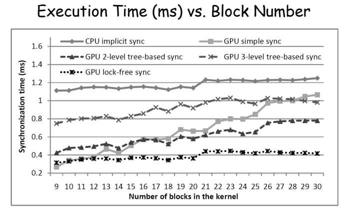


Synchronization Time (ms) vs. Block Number

Bitonic Sort



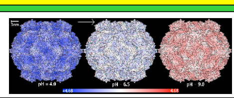
Execution Time (ms) vs. Block Number



Synchronization Time (ms) vs. Block Number

6. Related Work: Data-Parallel Apps on the GPU

N-Body Molecular Dynamics: 5000x speed-up
CPU Version: 1,334 minutes.
GPU Version: 0.26 minutes = 16 *seconds*.



7. Acknowledgements

NVIDIA
NSF

VirginiaTech
Invent the Future