Comparative Study of GPU Sorting Algorithms

Proposal Presentation
CSE 5304 Class Project
Paul Wortman

Background/Basics of GPU

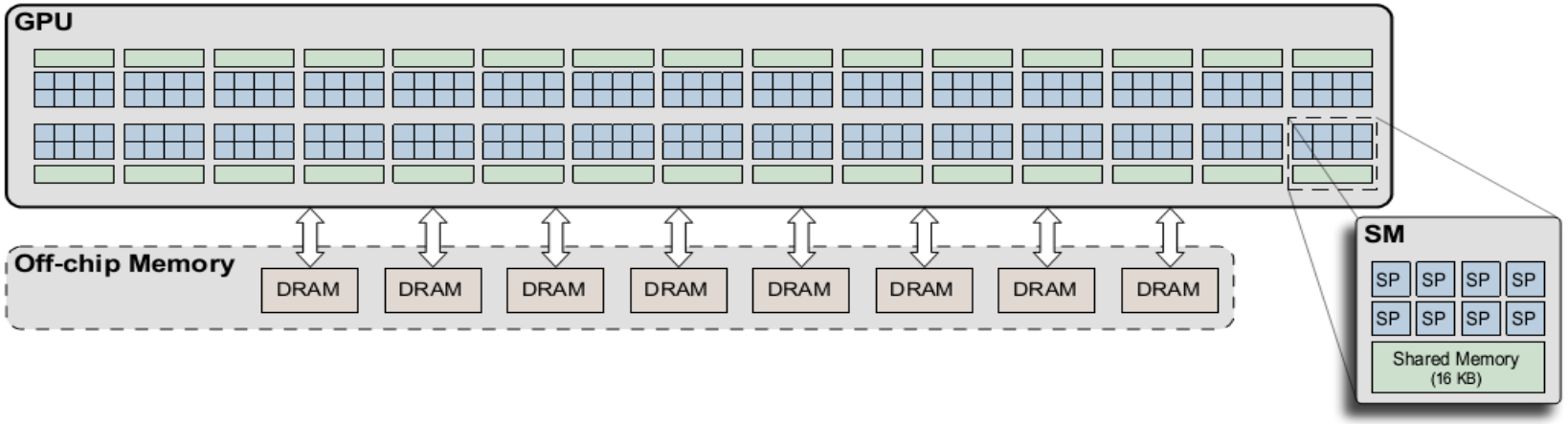

Fig. 1. GeForce GTX 280 GPU with 240 scalar processor cores (SPs), organized in 30 multiprocessors (SMs).

- GPU relies on multithreading (like a cache relies on latency hiding)
  - Necessary to design algorithms to create enough parallel work to maintain full machine utilization
- Threads executed in groups of 32 - names warps
- Warps free to load/store from/to any valid address
  - Supports general gather & scatter access to mem
- Consec. word access->hardware coalesce-> + mem thr.

Motivation/Aims

- Explore parallel algorithm design
- Further understanding of:
  - GPU architecture
  - GPU sorting algorithms
  - Other factors that might play into improvement/degradation of implementations
- Investigate the ways in which sorts have been implemented & the reasoning behind it

Existing Work/Implementations

- Hybrid GPU sort implementation
  - Sintorn & Assarsson
  - Mix of histogram, bucket sort, & merge sort
- GPU Radix Sort
  - Bandyopadhyay & Sahni
  - New implementation for GPU radix sort
    - Specifically aimed for multifield records
- GPU Radix & Merge Sort Algorithms
  - Satish, Harris, Garland
  - Design of high-performance parallel radix & merge sort
    - Exploited many facets to achieve goal

Project Design

- Load multiple GPU sort implementations onto HORNET
  - Implementations obtained from CUDA Toolkit Documentation site
    - Merge sort, simple/advanced quick sort, radix sort
- Run implementations with varying input size/data type (e.g. integer vs key-field)
  - Record & Compare specific aspects of each run
    - CPU usage, run time, memory usage, data throughput, etc.
- Find what factors affect each implementation

Further/Future Work

- Analyse source code to find nuisances that might be degrading performance
- Look into better possible means of implementation
  - To optimize parallelization of code
- Compare GPU architecture to sort implementation architecture
  - Determine which would possibly be more beneficial to improve upon (more so than the other)

References

[1] E. Sintorn and U. Assarsson, “Fast Parallel GPU-Sorting Using a Hybrid Algorithm”

[2] S. Bandyopadhyay and S. Sahni, “GRS - GPU Radix Sort Multifield Records*”

[3] N. Satish, M. Harris, and M. Garland, “Designing Efficient Sorting Algorithms for Manycore GPUs”

[4] CUDA Toolkit Documentation, http://docs.nvidia.com/cuda/index.html