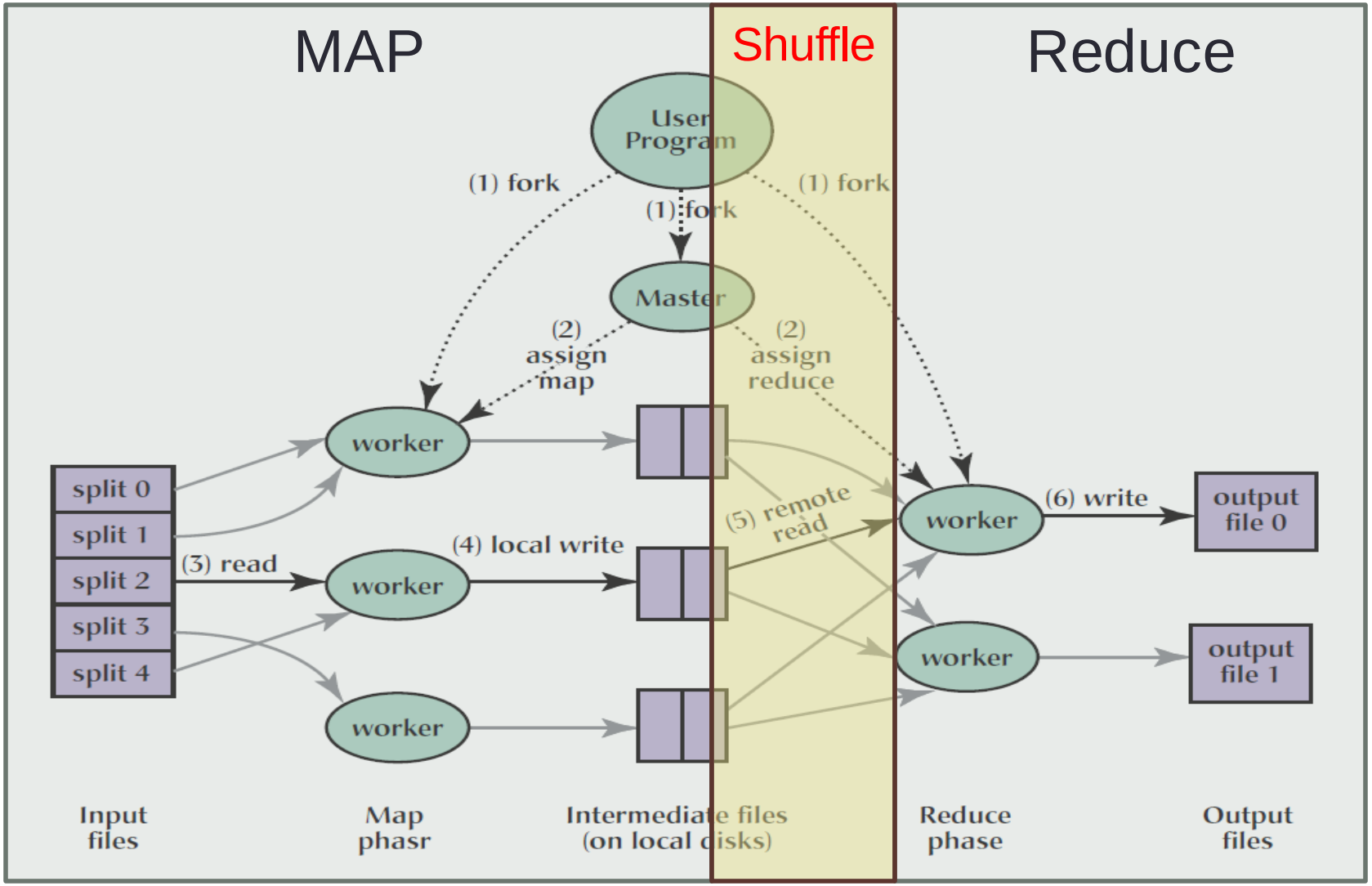
MapReduce on GPUs

Amit Sabne, Ahmad Mujahid Mohammed Razip, Kun Xu

MapReduce


MAP
Shuffle
Reduce
User Program
(1) fork
(1) fork
(1) fork
Master
(2) assign map
(2) assign reduce
worker
worker
worker
worker
worker
split 0
split 1
split 2
split 3
split 4
(3) read
(4) local write
(5) remote read
(6) write
output file 0
output file 1
Input files
Map phasr
Intermediate files (on local disks)
Reduce phase
Output files

Hadoop

- Open-source MapReduce framework from Apache, written in Java
- Used by Yahoo!, Facebook, Ebay, Hulu, IBM, LinkedIn, Spotify, Twitter etc.
- Primary usage – Data mining/machine learning algorithms on large datasets

Motivation

- *Map* phase of MapReduce programming model is extremely parallel
- *Combine*, the local reduce stage, is partially parallel
- On average more than 60% of the execution time is spent in (Map + Combine)
- GPU memory bandwidth > 10 * CPU memory bandwidth

Question :

Can we run the Map and Combine phases of MapReduce on an extremely parallel machine, like a GPU?

Related Work

- Phoenix – MapReduce for multi-cores

- Mars – MapReduce on a single node GPU
  - Pros – Performs GPU specific optimizations
  - Cons – Restricted to a single node system

- GPMR – MapReduce for GPU cluster with CUDA + MPI
  - Pros – Displays MapReduce Scalability in a GPU cluster
  - Cons – CUDA + MPI impose a productivity challenge

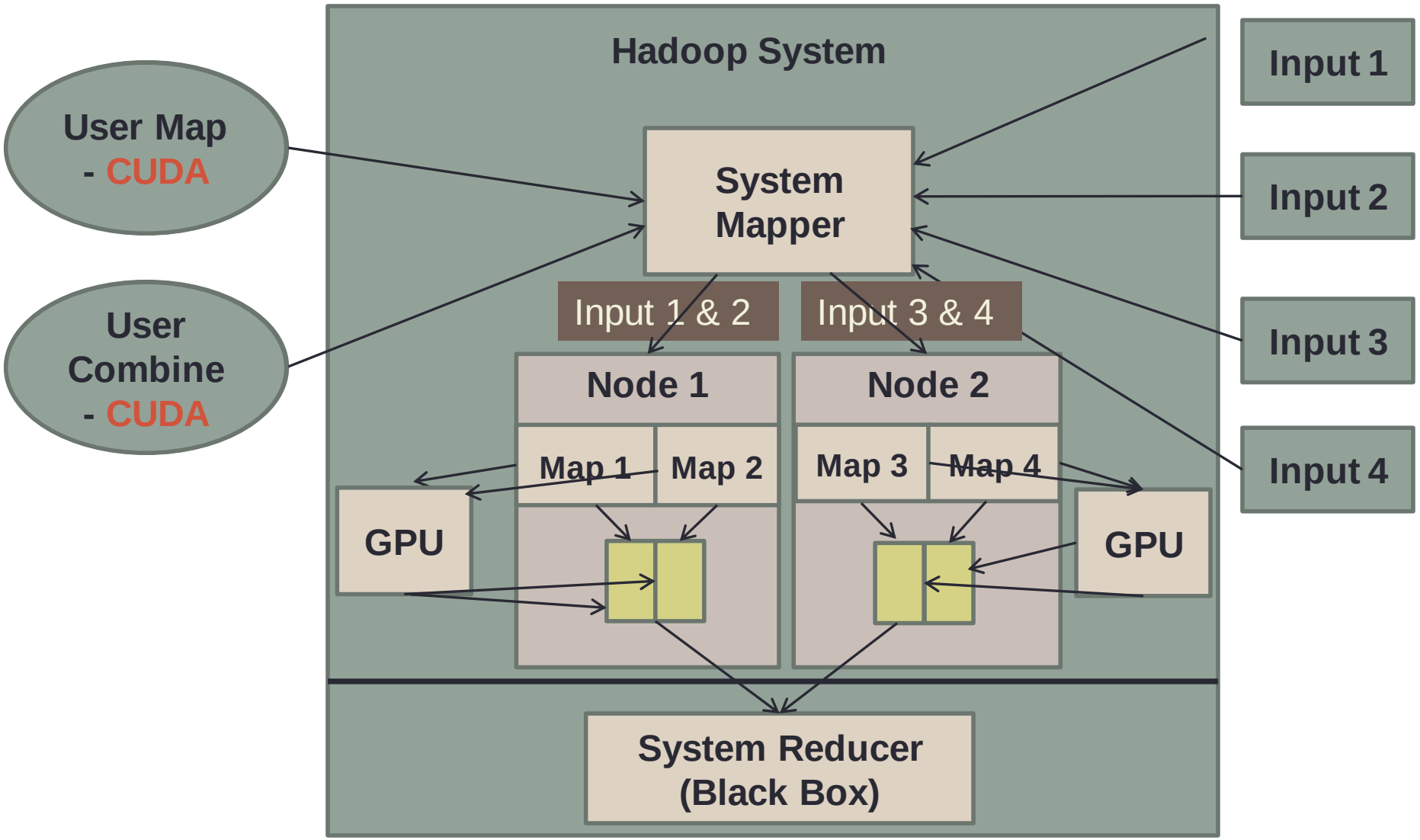
Challenges & Scope of the Project

- Programming Language
- Avoiding divergence on GPUs
- Combiner Implementation – exploiting partial parallelism

Scope :

- Run on single machine → Evaluate effectiveness of GPUs
- Use hand-written *CUDA* Map + Combine code
- Compare CPU vs. GPU Hadoop performance on different data sizes.

System Design



CUDA Implementation

Mapper:

- Input is split, every GPU thread works on same sized input
- Input is in text format → Inherent intra-thread diversion
- Map output sorting – Strings represented as a hash function

Combiner:

- One thread per Streaming Multiprocessor

Evaluation

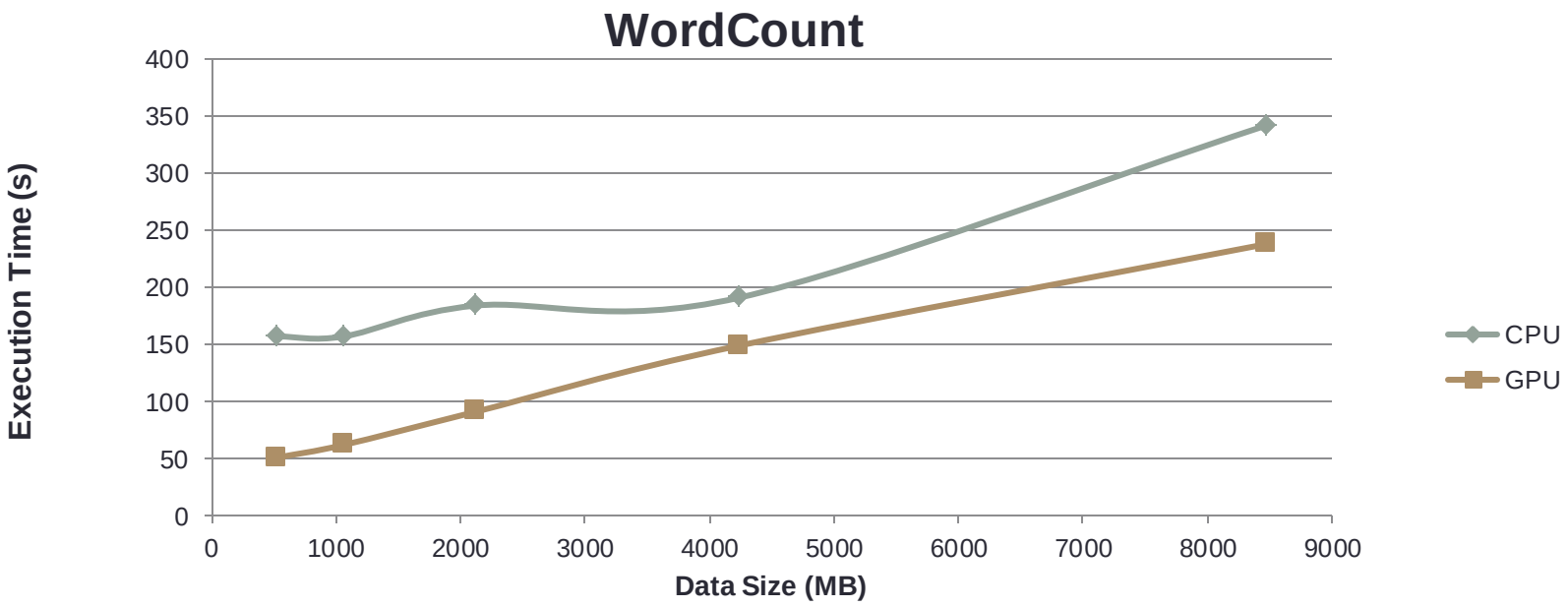
Setup

- CPU – AMD Opteron 6282 – 16 cores, 2.6 GHz
- GPU – Tesla M2090 – 16 SMs

Benchmarks

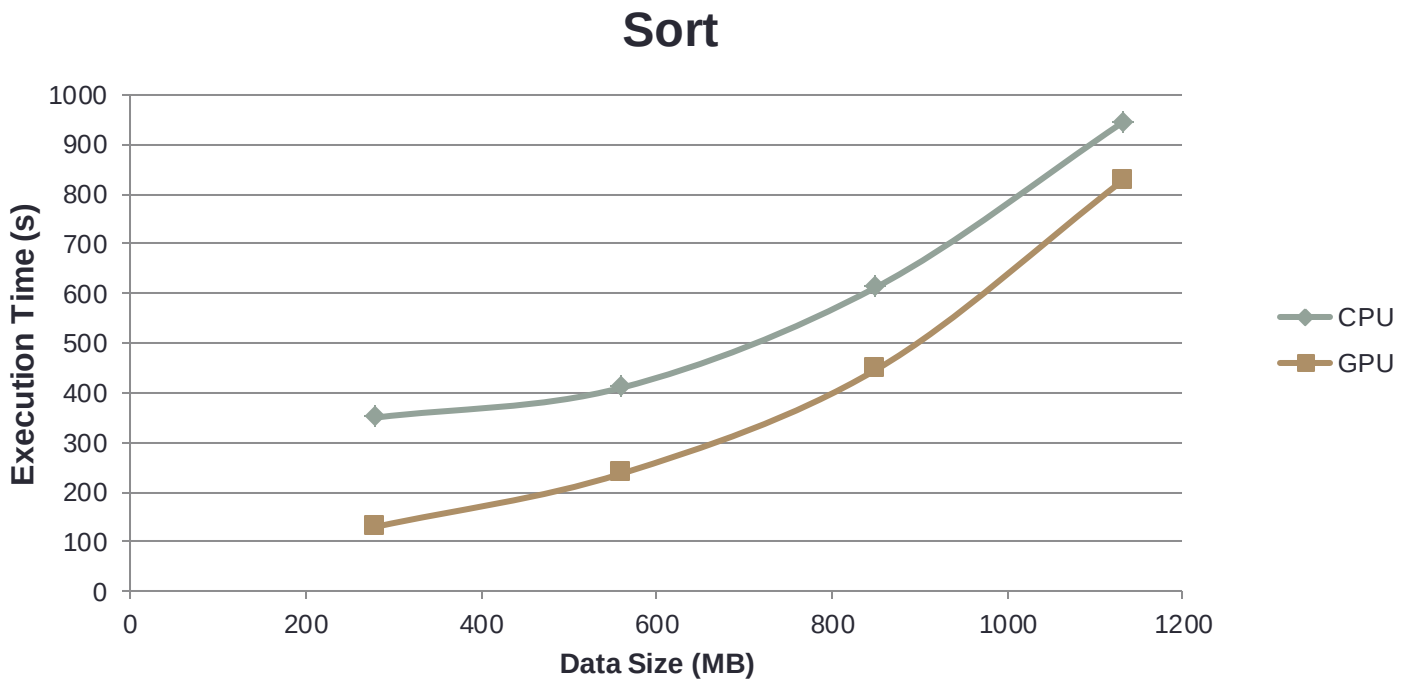
- WordCount – Data size up to 8GB
- Sort – Data size up to 1GB

Evaluation - WordCount



- File Size – 250 MB
- At 4 GB data, all CPU cores are engaged
- Combiner reduces the intermediate data

Evaluation – Sort



- Sort does not have a combiner → Huge intermediate data
- CPU parallelism is restricted by I/O
- Most of the time is consumed by reducer

Future Work

- Multiple Map tasks run in a serialized manner on the GPU
  → Combine them into a single, bigger Map task

- Hadoop scheduling in CPU + GPU environment

References

[1] Jeffrey Dean and Sanjay Ghemawat. “MapReduce: simplified data processing on large clusters”. Commun. ACM, 51(1):107–113, January 2008.

[2] Hadoop. http://hadoop.apache.org/.

[3] Bingsheng He, Wenbin Fang, Qiong Luo, Naga K. Govindaraju, and Tuyong Wang. “Mars: a MapReduce framework on graphics processors”. In Proceedings of the 17th international conference on Parallel architectures and compilation techniques, PACT ’08, pages 260–269, New York, NY, USA, 2008. ACM.

[4] J.A. Stuart and J.D. Owens. “Multi-GPU MapReduce on GPU Clusters”. In Parallel Distributed Processing Symposium (IPDPS), 2011 IEEE International, pages 1068 –1079, may 2011.

[5] Linchuan Chen, Xin Huo, and Gagan Agrawal. “Accelerating MapReduce on a coupled CPU-GPU architecture”. In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC ’12, pages 25:1–25:11, Los Alamitos, CA, USA, 2012. IEEE Computer Society Press.

[6] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrakis. Evaluating mapreduce for multi-core and multiprocessor systems. In HPCA ’07: Proceedings of the 2007 IEEE 13th International Symposium on High Performance Computer Architecture, pages 13–24, Washington, DC, USA, 2007. IEEE Computer Society.

[7] F. Ahmad, S. T. Chakradhar, A. Raghunathan, and T. N. Vijaykumar. Tarazu: optimizing mapreduce on heterogeneous clusters. In Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS ’12, pages 61–74, New York, NY, USA, 2012. ACM

Thank You!