MapReduce on GPUs

Amit Sabne, Ahmad Mujahid Mohammed Razip, Kun Xu
MapReduce

MAP

Shuffle

Reduce

Input files

Map phase

Intermediate files (on local disks)

Reduce phase

Output files

User Program

Master

worker

worker

worker

(1) fork

(2) assign map

(2) assign reduce

(1) fork

(1) fork

(3) read

(4) local write

(5) remote read

(6) write

output file 0

output file 1
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

Hadoop System

Node 1
- Map 1
- Map 2

Node 2
- Map 3
- Map 4

Input 1 & 2
Input 3 & 4

System Mapper

System Reducer (Black Box)

Input 1
Input 2
Input 3
Input 4

User Map - CUDA

User Combine - CUDA

GPU

7
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


Thank You!