CUDA Performance Study on Hadoop MapReduce Clusters

CSE 930 Advanced Computer Architecture @ Fall 2010

Chen He Peng Du

che@cse.unl.edu pdu@cse.unl.edu

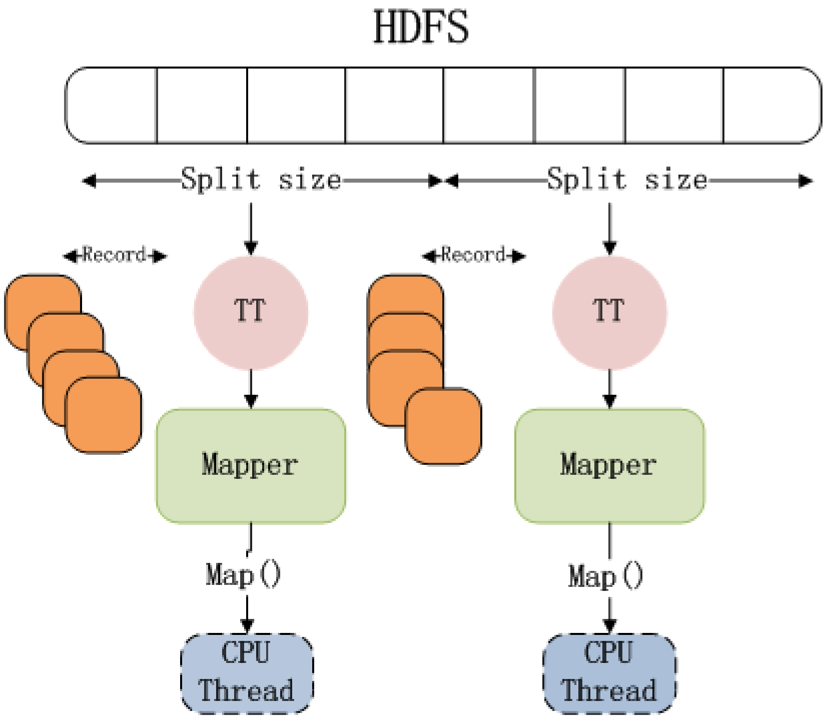
University of Nebraska-Lincoln

Overview

- Introduction
- Methodology
- Evaluation
- Conclusions
- Future work

Introduction

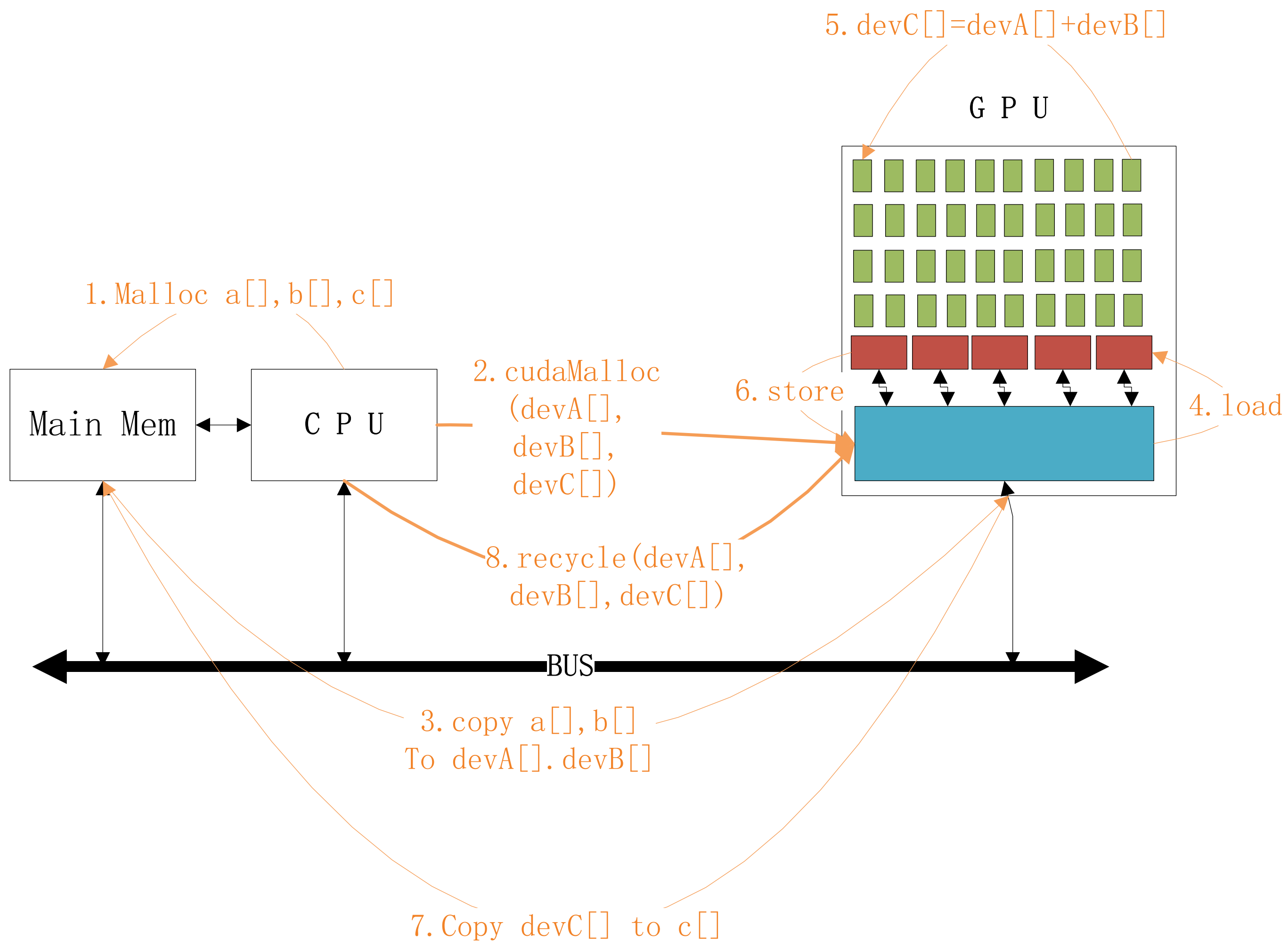
- Hadoop MapReduce



Original Mapper

Introduction

CPU+GPU Architecture

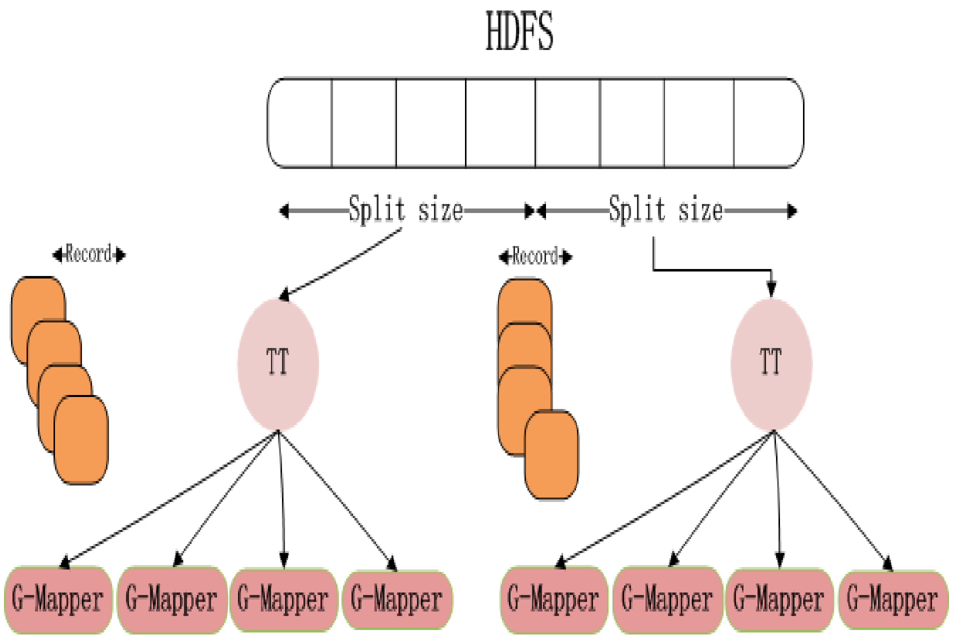

5. devC[]=devA[]+devB[]
G P U
1. Malloc a[], b[], c[]
Main Mem
C P U
2. cudaMalloc (devA[], devB[], devC[])
6. store
4. load
8. recycle(devA[], devB[], devC[])
BUS
3. copy a[], b[] To devA[]. devB[]
7. Copy devC[] to c[]

Introduction

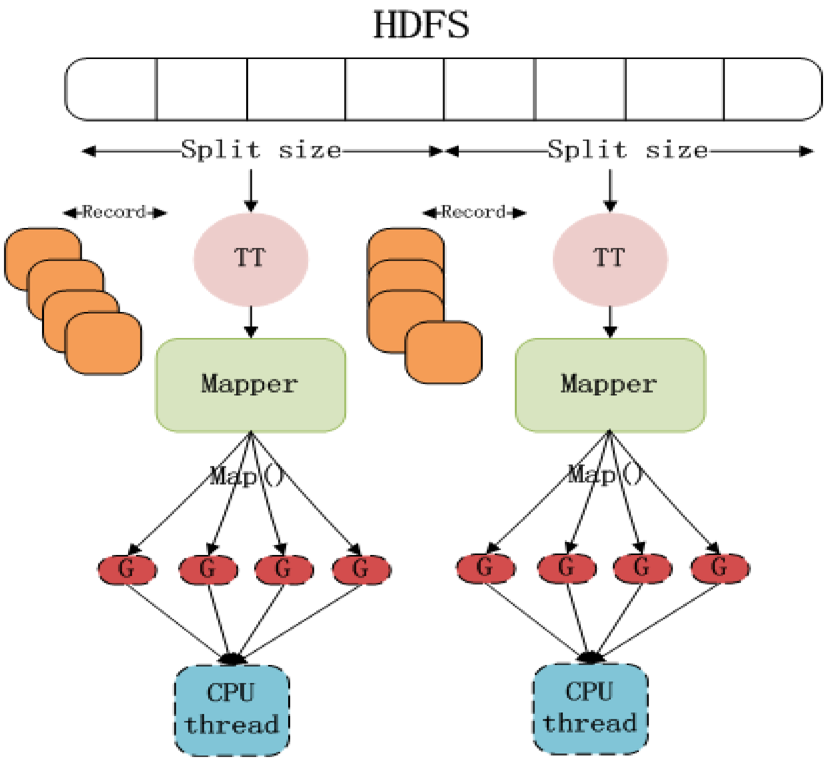
- Questions
  - Can we introduce CUDA into Hadoop MapReduce Clusters?
    - Mechanism and implementation
  - Is this reasonable?
    - Effects and Costs

Methodology

- Question-1:Can we introduce CUDA into Hadoop ?



Accelerated Mapper-1



Accelerated Mapper-2

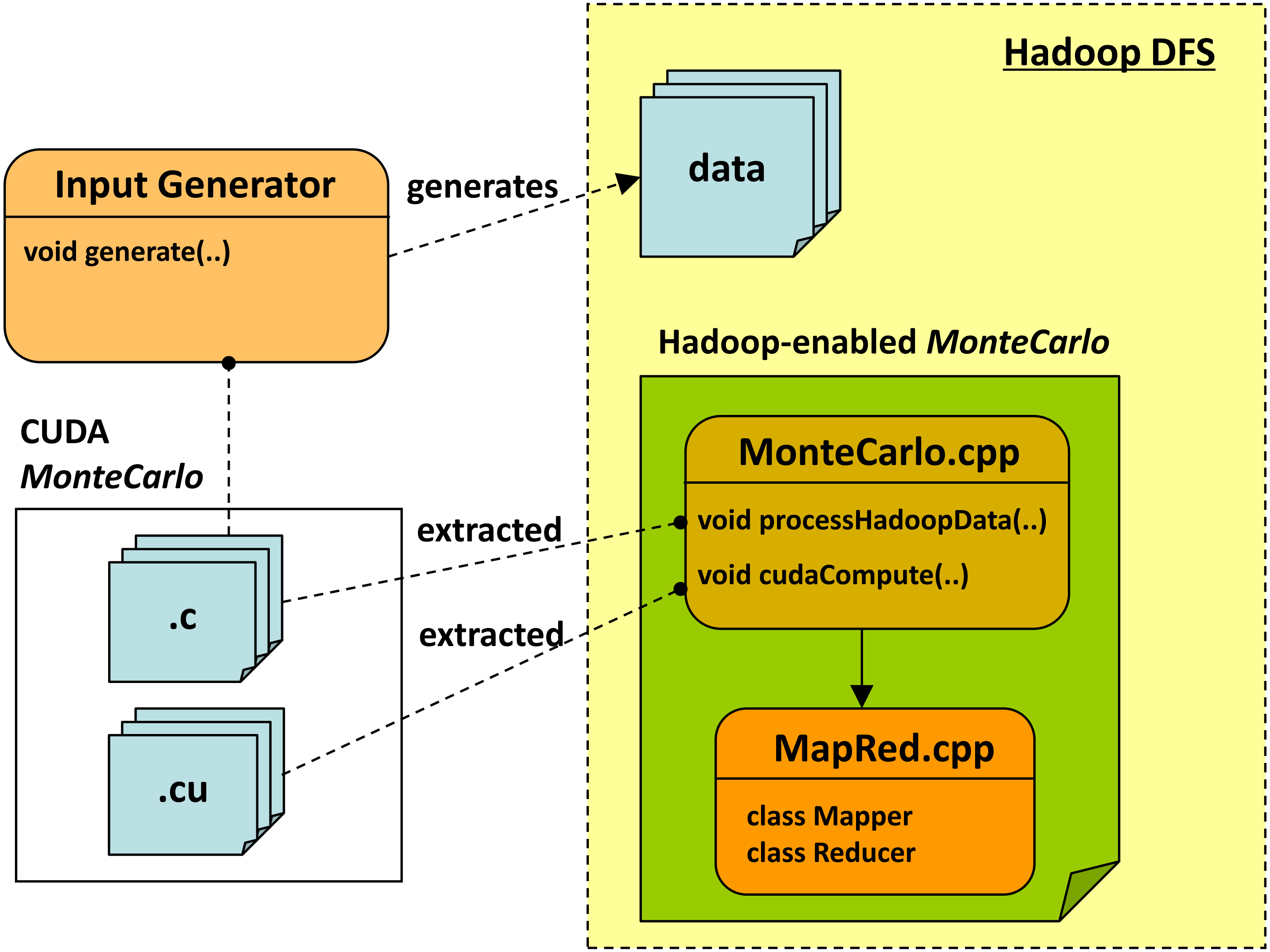
Methodology

- Test cases
  - SDK programs
    - Data intensive: Matrix Multiplication
    - Computation intensive: Monte Carlo
  - MDMR (Molecular Dynamics simulation based on MapReduce)
    - Pure Java program
    - Introduce JCUDA

Methodology

- Port CUDA programs onto Hadoop
  - GPU (CUDA-C) vs CPU (C)
  - Approach
    - MapRed (*processHadoopData* & *cudaCompute*)
    - Main (*Hadoop Pipes*)
    - Scripts (*runbase.sh*, *run-<prog>-CPU/GPU.sh*)
    - Input data generators

Methodology



Methodology

- MDMR (Molecular Dynamics simulation based on MapReduce)
  - Time Complexity by using CPU

$$T(n) = c_1 n^2 + c_2 n + c_3$$

  - We can simply employ GPU to parallel the n-squre portion and reduce the time complexity to linear (within the limit of GPU threads)

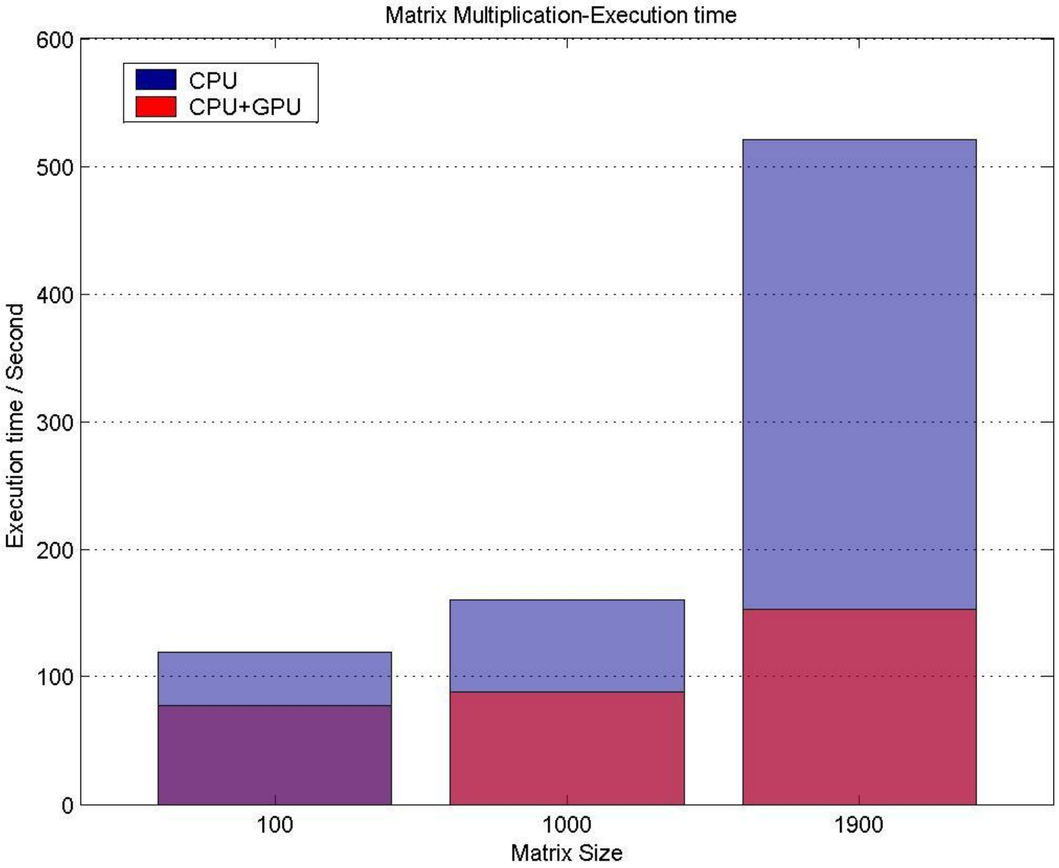
$$T^{'}(n) = c_1(dn) + c_2 n + c_3 + c_4$$

Evaluation

- **Environment**
  - Head: 2xAMD 2.2GHz, 4GB DDR400 RAM, 800GB HD
  - Slaves: 3 PCs (AMD 2.3G CPU, 2G DDR2-667 RAM, 400GB HD, 1Gbps Ethernet)
  - GPU: XFX 9400GT 64bit 512MB DDR3
  - CUDA 3.2 Toolkit
  - Hadoop 0.20.3
  - ServerTech CWG-CDU power distribution unit (for the power consumption monitoring)
- **Factors**
  - Speedup
  - Power consumption
  - Cost

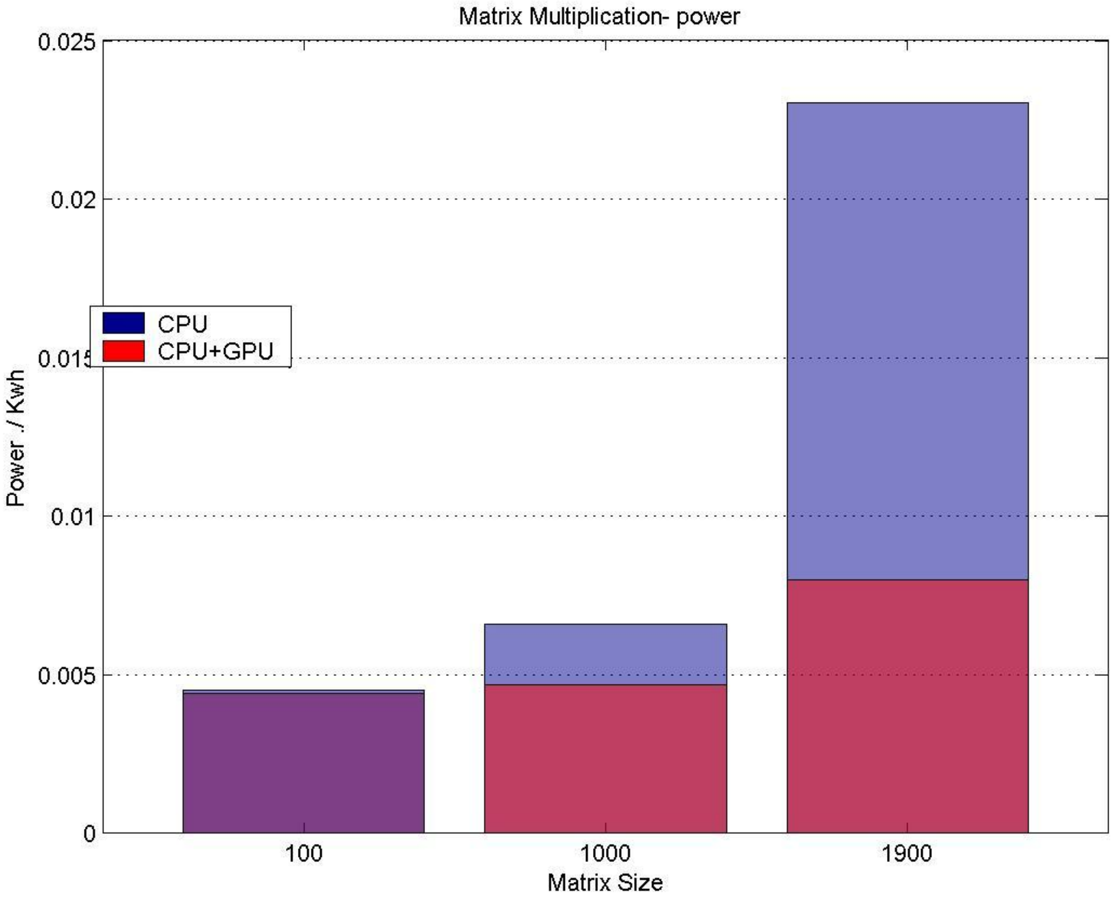
Evaluation

- Matrix Multiplication (Execution time)

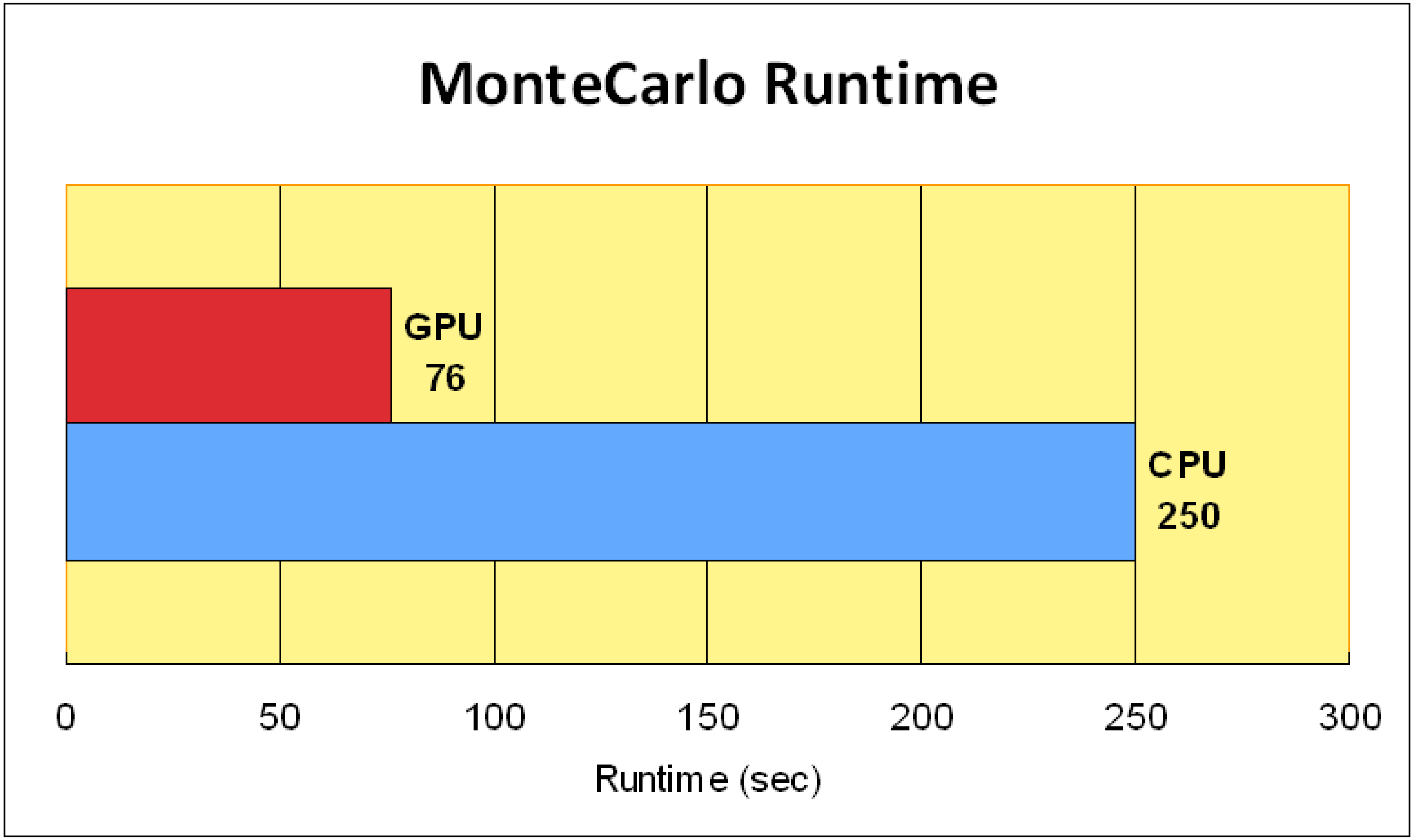


Evaluation

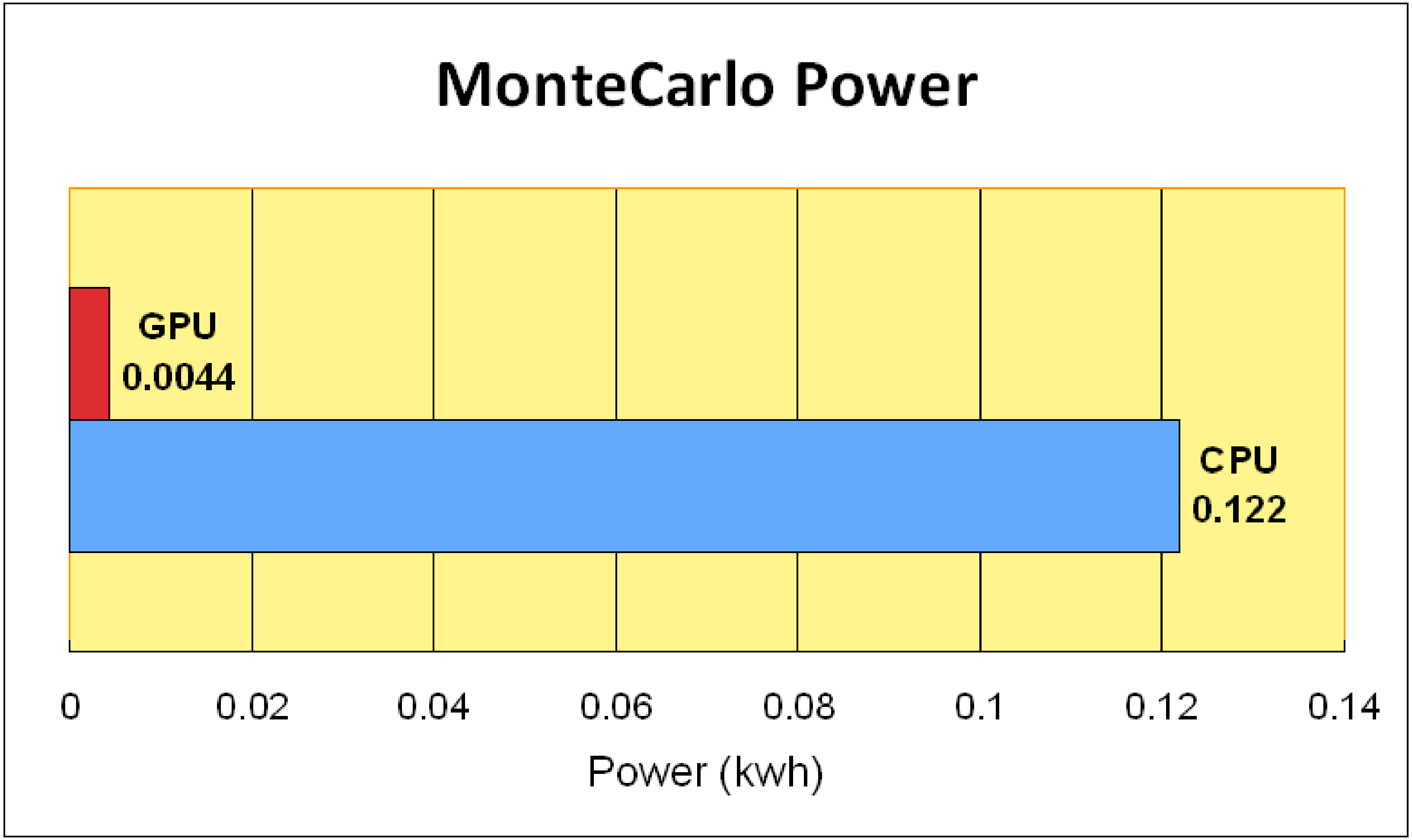
- Matrix Multiplication (Power consumption)



Evaluation

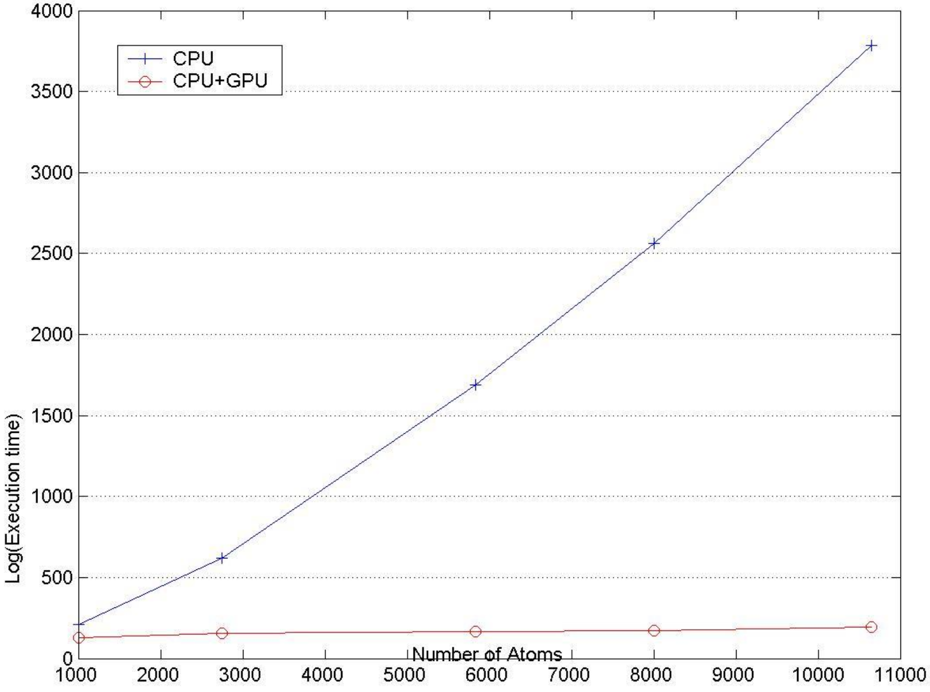

MonteCarlo Runtime
GPU
76
CPU
250
0
50
100
150
200
250
300
Runtime (sec)

Evaluation


MonteCarlo Power
GPU
0.0044
CPU
0.122
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
Power (kwh)

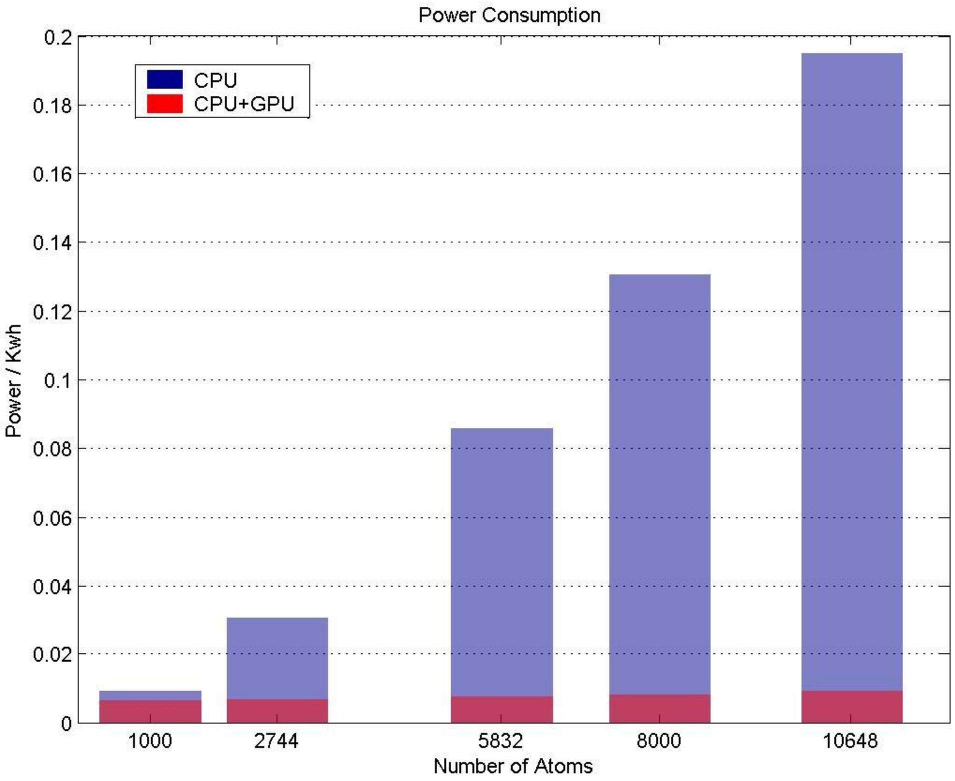
Evaluation

- MDMR
  - Execution time



Evaluation

- MDMR
  - Power consumption



Conclusions

- Introduced GPU into MapReduce cluster and obtained up to **20 times** speedup.
- Reduced up to **19/20** power consumption with the current preliminary solution and work load.
- Compared with upgrading CPUs and adding more nodes, deploying GPU on Hadoop has **high cost-to-benefit ratio**.
- Provided **practical implementations** for people wanting to construct MapReduce clusters with GPUs.

Future Work

- Port more CUDA programs onto Hadoop.
- Incorporate reducers into the experiments
- Support heterogeneous clusters which mixed GPU-nodes and non-GPU nodes.

Reference

- nVIDIA CUDA

  http://developer.nvidia.com/object/cuda-3.2/
- Hadoop, *http://www.hadoop.com.*
- J. Polo, D. Carrera, Y. Becerra, V. Beltran, J. Torres and E. Ayguadé Performance *Management of Accelerated MapReduce Workloads in Heterogeneous Clusters*, ICPP2010, (2010), 654-662.
- C. He, D. Swanson. *Molecular Dynamics simulation based on MapReduce*, poster section, *LCI 2010*, (2010).