Understanding the Behavior of In-Memory Computing Workloads

Rui Hou

Institute of Computing Technology,CAS

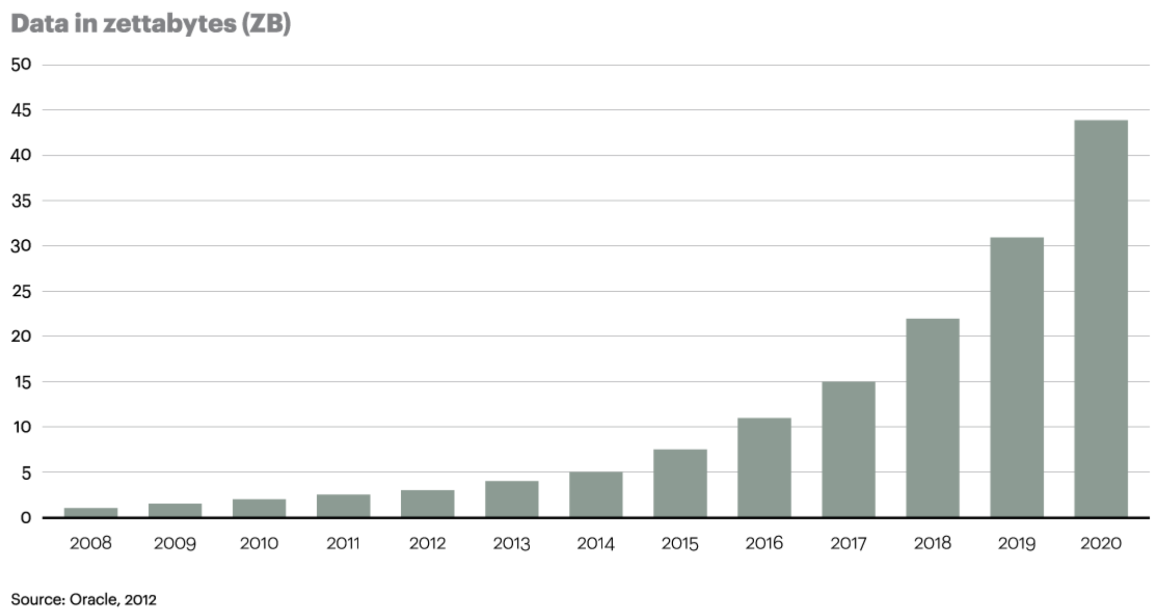
July 10, 2014

中国科学院计算技术研究所
Institute of Computing Technology,Chinese Academy of Sciences

Outline

- Background
- Methodology
- Results and Analysis
- Summary

Background



- **The era of big data is coming**
- **Data is growing at 40% annual rate, reaching nearly 45ZB by 2020**
- **Off-line processing like Hadoop has been the dominant scenarios in the past**

Background

- Online big data processing grows fast
- In-memory computing is becoming the major approach
- Spark is born!!
- However：

Whether existing system can support on-line real-time processing workloads efficiently?

What kind of optimizations or even revolutions are required?

中科院计算所

Institute of Computing Technology, Chinese Academy of Sciences

Outline

- Background
- Methodology
- Results and Analysis
- Summary

Methodology

- Hardware
  - 17-node X86 cluster
  - Two Intel Xeon 2.40GHz E5645 processors, 64GB DDR3

- Measurement tools
  - Microarchitecture：Intel Vtune Amplifier
  - CPU and I/O：Tools of Linux
  - Memory：Hyper Memory Trace Tracker (HMTT)

Workloads

Spark&Hadoop

- Naive Bayes
  - machine learning, e-commerce
- Grep
  - search engine, social network
- Hive&Shark
  - data warehousing
- PageRank
  - search engine
- Connected Components
  - graph analysis

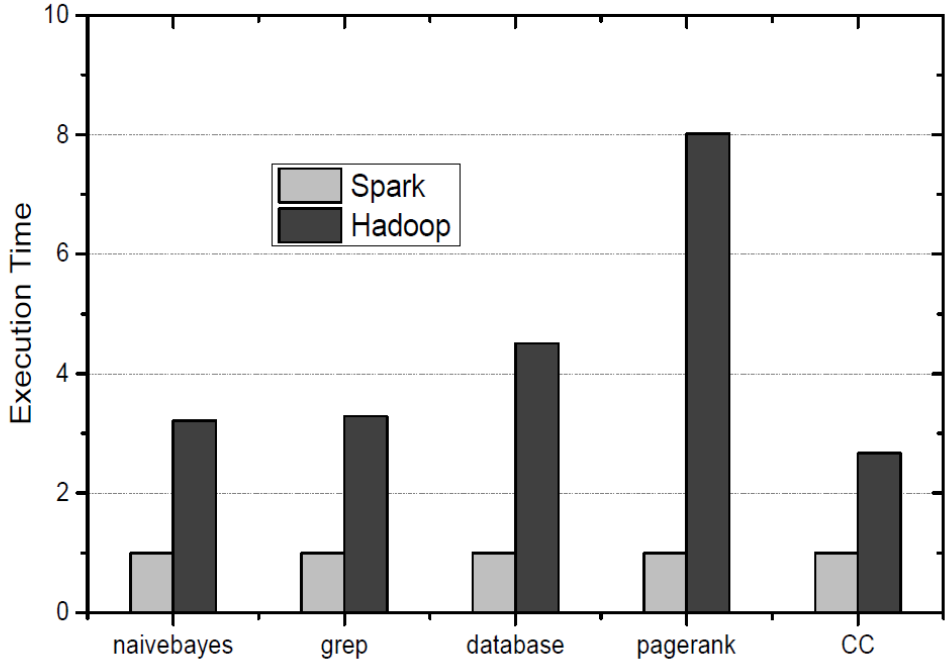
Compared Benchmarks

- CloudSuite
- DesktopCloud
- SPEC CPU2006
- TPC-C

Outline

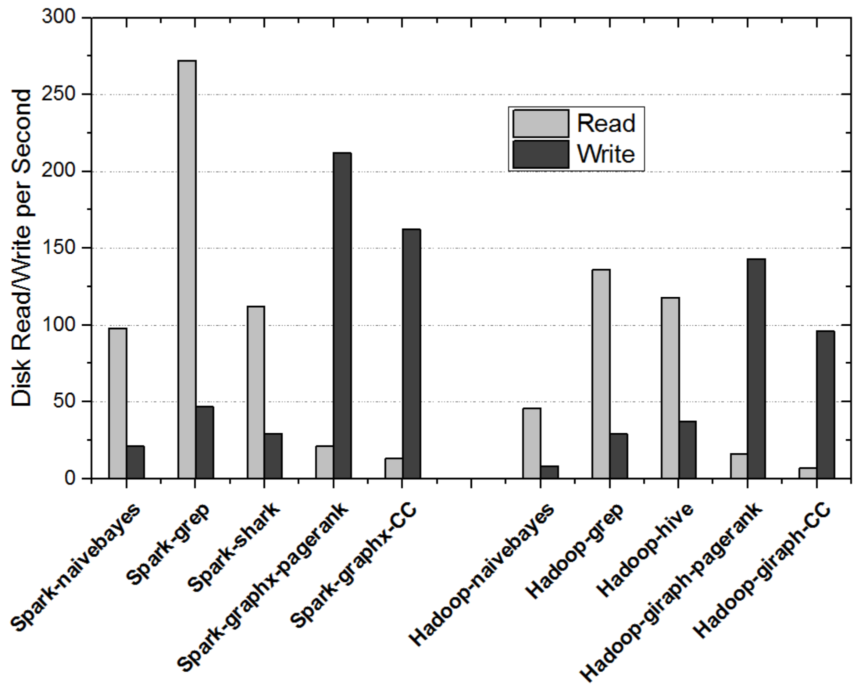
- Background
- Methodology
- Results and Analysis
  - Performance of Execution
  - Memory Access
  - Micro-Architecture
- Summary

Execution Time



- **Hadoop benchmarks are 2.7 times to 8.4 times slower than Spark benchmarks**

Disk I/O

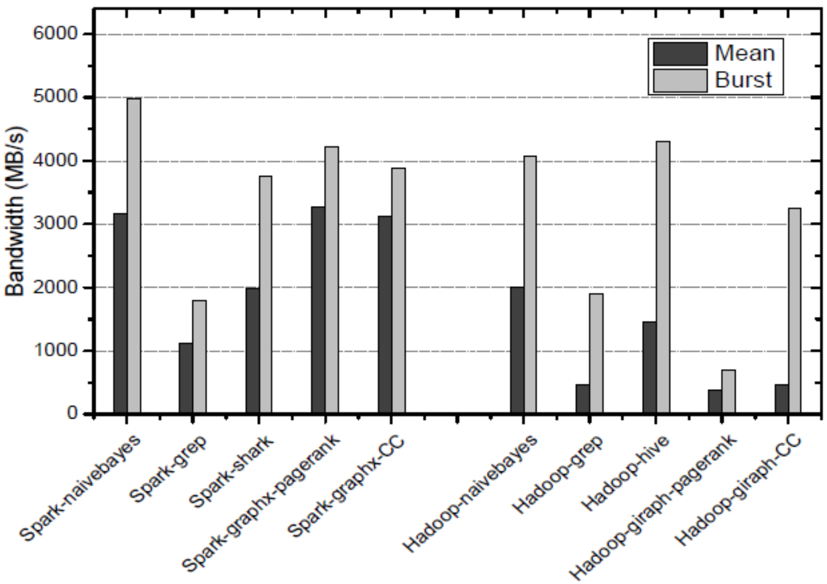


- **Spark benchmarks are larger than Hadoop benchmarks on average**
  - **Spark faster than Hadoop when accessing same input and output datasets from disk**

Outline

- Background
- Methodology
- Results and Analysis
  - Performance of Execution
  - Memory Access
  - Micro-Architecture
- Summary

Burst and Average Bandwidth of Memory

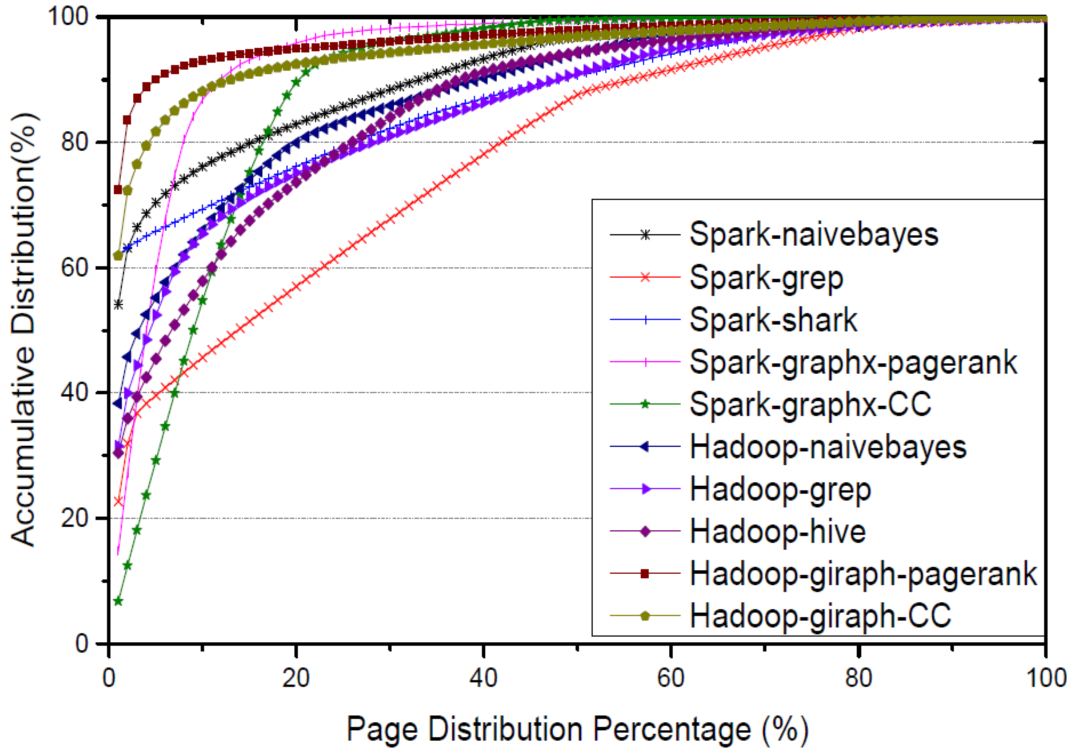


- Sample memory bandwidths every 1ms
- **Burst bandwidth**: the average value of the top 10% of the bandwidth samples

- **Hadoop** only use **15%** of the peak bandwidth of 6.4GB/s (800MHz)
- **Spark** can reach about **40%** of the peak bandwidth
- **Burst bandwidth** of **Hadoop** exceed **198%** of average bandwidth, while **Spark** is only **47%** higher than average

Memory access of Spark is much more stable

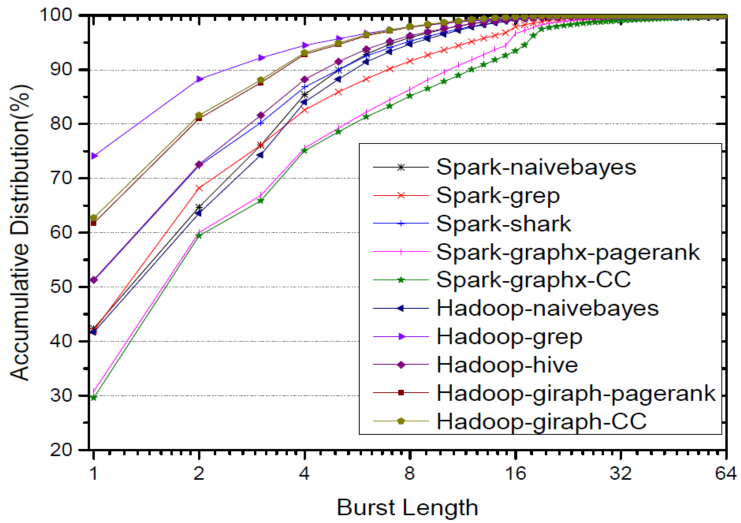
Page Access Frequency



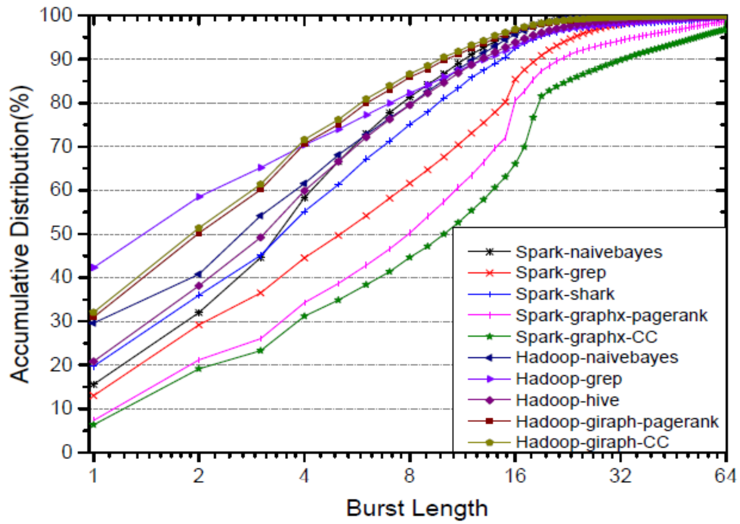
- **About 80% of the memory requests access only 20% of the pages**
- **For some Spark workloads, 90% memory requests just access 10% pages**

Locality of page accesses of Spark is very convergent

Burst Access



Burst Memory Access Distribution

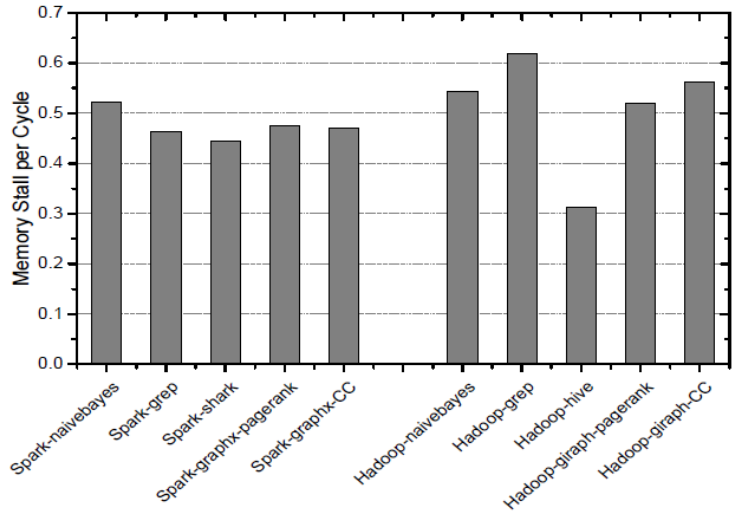


Memory Bus Traffic Distribution

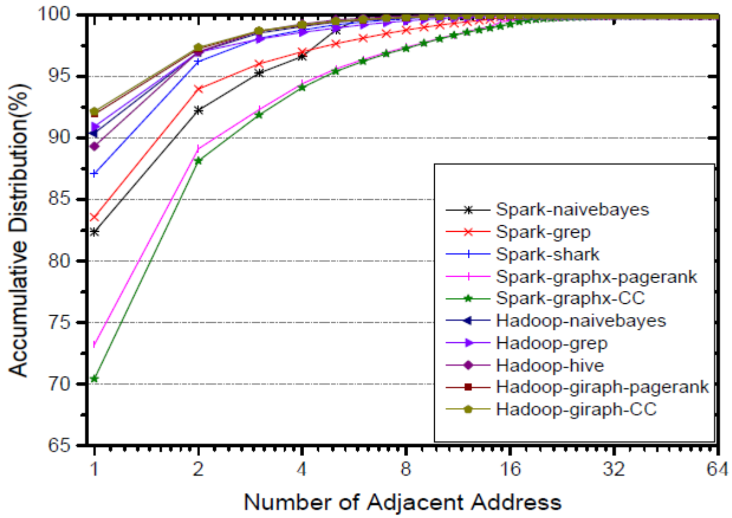
- **Burst with size of one means one cacheline**
- **Burst flows of Spark is 50% higher than Hadoop**
- **Almost all the burst size of Hadoop is less than 16**
- **For Spark, burst contributes 90% of the bus traffic, Hadoop only contributes 70%**

Spark has shown high performance in the memory bandwidth utilization

Adjacent Address Access



Memory Stall Cycles



Adjacent Memory Access

- **Little difference between Spark and Hadoop in the ratio of memory stall**
- **Iterative algorithm of Spark not put much stress on the memory access module of the back-end of pipeline**

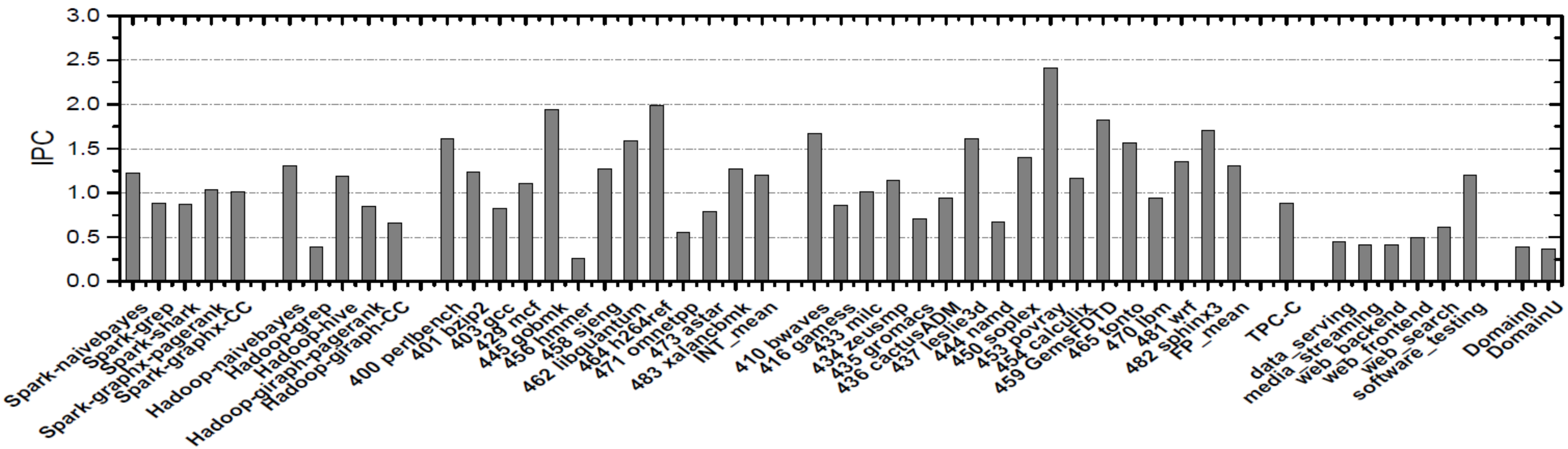
- **Spark have more memory requests with adjacent address than Hadoop**

We speculate that frequently correct prefetching would relieve the pipeline stalls caused by load store unit

Outline

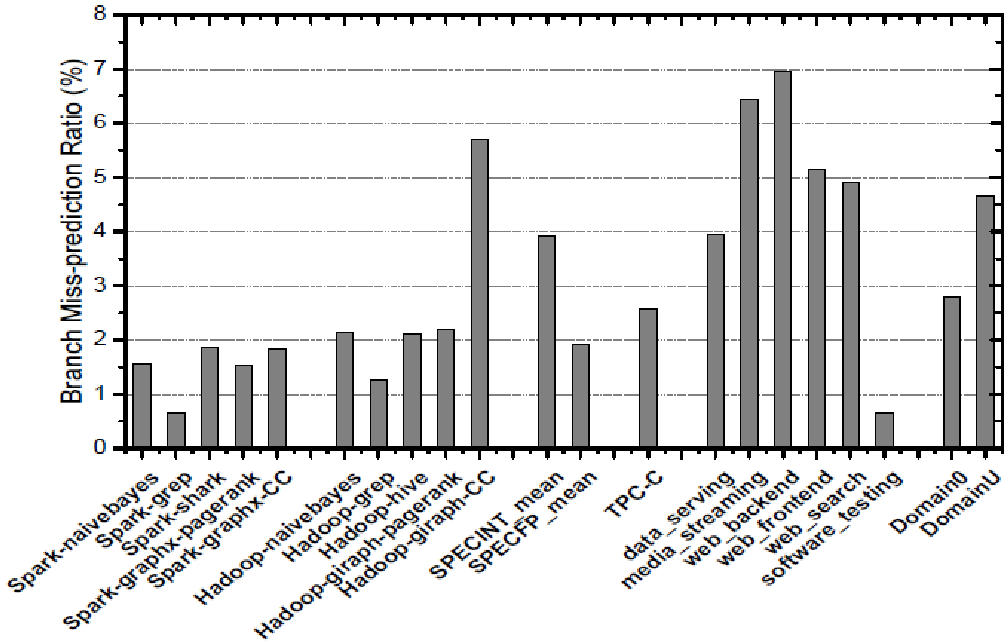
- Background
- Methodology
- Results and Analysis
  - Performance of Execution
  - Memory Access
  - Micro-Architecture
- Summary

IPC



- Spark shows higher IPC than Hadoop, CloudSuite and DesktopCloud
- Compared with SPECCPU, the IPC of Spark is lower

Branch Prediction



- Branch prediction miss rate of Spark is lower than other benchmarks
- Branch instructions of Spark and Hadoop tested have simple patterns
  - The possible reason is Spark and Hadoop prefer simple algorithms

Branch predictor of Intel processors works well for Spark and Hadoop

Outline

- Background
- Methodology
- Results and Analysis
  - Performance of Execution
  - Memory Access
  - Micro-Architecture
- Summary

中科院计算所
Institute of Computing Technology, Chinese Academy of Sciences

Summary

- On general-purpose X86 server processors, Spark work better than Hadoop and scale-out applications
- Characteristics of memory access are different between Spark and Hadoop, in spite of having the same algorithms and same input datasets
  - The average bandwidth of Spark is about 40% of the peak bandwidth, while Hadoop only uses 15%
  - Burst bandwidth of some Spark applications is up to 80% of the peak bandwidth
  - Memory bandwidth optimizations may be preferred by Spark workloads
    - Improving the frequency of memory
    - Using Hybrid Memory Cube (HMC)
    - …

Q&A

Thank You!