BigDataBench: a Big Data Benchmark Suite from Web Search Engines

Wanling Gao, Yuqing Zhu, Zhen Jia, Chunjie Luo, Lei Wang, Jianfeng Zhan, Yongqiang He, Shiming Gong, Xiaona Li, Shujie Zhang, and Bizhu Qiu

http://prof.ict.ac.cn/BigDataBench

Third Workshop on Architectures and Systems for Big Data (ASBD 2013) held in conjunction with ISCA-40
Acknowledgements

• This work is supported by the Chinese 973 project (Grant No.2011CB302502), the Hi-Tech Research and Development (863) Program of China (Grant No.2011AA01A203, No.2013AA01A213), the NSFC project (Grant No.60933003, No.61202075), the BNSF project (Grant No.4133081), and Huawei funding.
Requirements for Big Data Benchmarking

To truly reflect
- Use cases
- Requirements

To rapidly evolve
- New workloads
- New use cases

To widely cover
- Application domains
- Data types

<<Proposal for a Big Data Benchmark Repository>>--Andries Engelbrecht . WBDB2012
Challenges

- **Current State—Immature**
  - “We Don't Know Enough to Make a Big Data Benchmark Suite”

*An Academia-Industry View, Yanpei Chen, UC Berkeley/Cloudera WBDB2012*
State-of-Practice and State-of-Art

- **Sort**: MinuteSort, JouleSort, TeraByte Sort
  - Only one application: one-fits-all solution?
- **GridMix, Hadoop microbenchmark**
  - Generating data set randomly
  - Ignoring the characteristics of real-world data
- **BigBench**
  - From electronic commerce
  - Only describe a specification without implementation
BigDataBench: an open source project

- Available from [http://prof.ict.ac.cn/BigDataBench](http://prof.ict.ac.cn/BigDataBench)

- Data generation tools
  - Generate GB, TB, or PB-scale data from seed data

- BigDataBench 1.0 Beta version
  - Six benchmarks from web search engines
Outline

- Motivation
- Benchmarking Methodology
- Case Studies
- Future Work
Benchmarking Methodology

Following An Incremental Approach

Considering Variety of Workloads

Methodology of Generating Big Data
Investigate Application Domain

So many application domains

Why we choose Search Engine for the first step?
Popularity of Search Engines

92% of online adults use search engines to find information on the web.

(pew internet study)
Page Views & Daily Visitors

- Search Engine is the most important Internet service workload.
- 40% of top 20 websites

http://www.alex.com/topsites/global;0
Benchmarking Methodology

Following An Incremental Approach

Considering Variety of Workloads

Methodology of Generating Big Data
Our two practices

- Surveying Open-Source Common Search Engines
  - Backed by practitioners of several industry partners
    - From Facebook, Yahoo!, Huawei, Baidu, and Sogou

- Building a semantic search engine (Chinese)
  - ProfSearch
    - Search scientists or professionals
    - 251,564 researchers across 260 universities and institutes
    - http://prof.ict.ac.cn/
Overview of a Typical Search Engine
Algorithms in a Typical Search Engine

- **Store Server**
- **Repository**
- **Crawler: Parallel**
- **URL Server**
- **Anchors**
- **URL Resolver**
- **Links**
- **Doc Index**
- **Lexicon**
- **Pagerank**
- **Searcher**
- **User Requests**

**Internet**
- **graph mining**
- **grep & segmentation**

**Barrels: Hits Information**
- **word count**

**Sorter**
- **sort**

**vector calculation**

**ASBD 2013 ISCA-40**
ProfSearch

Crawler Workloads
- Scrapy

Analysis Workloads
- SVM, Naïve Bayes, K-means, HMM, CRFs, LSA, LDA

Store and Management Workloads
- HDFS – Storing unstructured web pages
- HIVE – Storing semi-structured intermediate data
- MySQL – Storing structured data extracted from the web

Web Service Workloads
- Sphinx
Three “V” of Big Data

Diversity of typical workloads must be considered!
Workloads in BigDataBench 1.0 Beta

- Analysis Workloads
  - Simple but representative operations
    - Sort, Grep, Wordcount
  - Highly recognized algorithms
    - Naïve Bayes, SVM

- Search Engine Service Workloads
  - Widely deployed services
    - Nutch Server
## Features of Workloads

<table>
<thead>
<tr>
<th>Workloads</th>
<th>Resource Characteristic</th>
<th>Computing Complexity</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sort</td>
<td>I/O bound</td>
<td>$O(n \times \log n)$</td>
<td>Integer comparison domination</td>
</tr>
<tr>
<td>Wordcount</td>
<td>CPU bound</td>
<td>$O(n)$</td>
<td>Integer comparison and calculation domination</td>
</tr>
<tr>
<td>Grep</td>
<td>Hybrid</td>
<td>$O(n)$</td>
<td>Integer comparison domination</td>
</tr>
<tr>
<td>Naïve Bayes</td>
<td>/</td>
<td>$O(m \times n)$</td>
<td>Floating-point computation domination</td>
</tr>
<tr>
<td>SVM</td>
<td>/</td>
<td>$O(M \times n)$</td>
<td>Floating-point computation domination</td>
</tr>
<tr>
<td>Nutch Server</td>
<td>I/O &amp; CPU bound</td>
<td></td>
<td>Integer comparison domination</td>
</tr>
</tbody>
</table>
Variety of Workloads are included

- Workloads
  - Off-line
    - Base Operations
      - I/O bound
        - Sort
      - CPU bound
        - Wordcount
      - Hybrid
        - Grep
  - On-line
    - Machine Learning
      - Naïve Bayes
    - Nutch Server
      - SVM
Benchmarking Methodology

Following An Incremental Approach

Considering Variety of Workloads

Methodology of Generating Big Data
Big Data Puzzle

- Confidential for company
- Difficult to download
- Easily get small-scale data
Methodology of Generating Big Data

To preserve the characteristics of real-world data

- Characteristic Analysis
- Expand

Small-scale Data → Big Data

Semantic
- Word frequency

Locality
- Temporally
- Spatially

Word reuse distance
Word distribution in document
Scalable Data Generation Tool

**Request Generation**
- Real query trace
- Timing Model
- Semantic Model
- Locality Model
- Synthetic query rate

**Input Data Generation**
- Small-scale data
- Semantic Model
- Locality Model
- Synthetic big data
BigDataBench-Data Analysis Workloads

Data Generation Tool
- Generate from small-scale data

Big Data
- Use HDFS as the storage

Representative workloads in Search Engine
- Sort
- Wordcount
- Grep
- Naïve Bayes
- SVM
BigDataBench-Service Workload

Data Generation Tool

Massive Requests

Search service

Search Server

Web Server

Generate from small-scale query sequences and timing sequences

Use local file system as the storage

Search engine
Outline

- Motivation
- Benchmarking Methodology
- Case Studies
- Future Work
Use Case 1: System Evaluation

- Using BigDataBench 1.0 Beta
- Data Scale
  - 10 GB – 2 TB
- Hadoop Configuration
  - 1 master 14 slave node

<table>
<thead>
<tr>
<th>CPU Type</th>
<th>Intel CPU Core</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel® Xeon E5645</td>
<td>6 <a href="mailto:cores@2.40G">cores@2.40G</a></td>
</tr>
<tr>
<td>L1 DCache</td>
<td>L1 ICache</td>
</tr>
<tr>
<td>6 x 32 KB</td>
<td>6 x 32 KB</td>
</tr>
<tr>
<td>L2 Cache</td>
<td>L3 Cache</td>
</tr>
<tr>
<td>6 x 256 KB</td>
<td>12MB</td>
</tr>
</tbody>
</table>

Table 1. Details of node configuration
System Evaluation

- a threshold for each workload
  - 100M ~ 1TB
  - System is fully loaded when the data volume exceeds the threshold

- Sort is an exception
  - An inflexion point (10GB ~ 1TB)
  - Data processing rate decreases after this point

- Global data access requirements
  - I/O and network bottleneck

- System performance is dependent on applications and data volumes.

Figure 3. Data Processing Rates for Different Data Volumes to Process.
Use case 2: Architecture Research

- Using BigDataBench 1.0 Beta
- Data Scale
  - 10 GB – 2 TB
- Hadoop Configuration
  - 1 master 14 slave node

<table>
<thead>
<tr>
<th>CPU Type</th>
<th>Intel CPU Core</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel ® Xeon E5645</td>
<td>6 <a href="mailto:cores@2.40G">cores@2.40G</a></td>
</tr>
<tr>
<td>L1 DCache</td>
<td>L1 ICache</td>
</tr>
<tr>
<td>6 × 32 KB</td>
<td>6 × 32 KB</td>
</tr>
<tr>
<td>L2 Cache</td>
<td>L3 Cache</td>
</tr>
<tr>
<td>6 × 256 KB</td>
<td>12MB</td>
</tr>
</tbody>
</table>

Table 1. Details of node configuration
Use case 2: Architecture Research

Some micro-architectural events are tending towards stability when the data volume increases to a certain extent.

Cache and TLB behaviors have different trends with increasing data volumes for different workloads:
- L1I_miss/1000ins: increase for Sort, decrease for Grep.

Figure 4. Cache and TLB Behaviors of Data Analysis Applications.
Search engine service experiments

- Same phenomena is observed
  - Micro-architectural events are tending towards stability when the index size increases to a certain extent

- Big data impose challenges to architecture researches since large-scale simulation is time-consuming

![Graph showing cache and TLB behaviors of the Nutch Server.](image)

**Figure 5.** Cache and TLB Behaviors of the Nutch Server.

Index size: 2GB ~ 8GB
Segment size: 4.4GB ~ 17.6GB
Outline

- Motivation
- Benchmarking Methodology
- Case Studies
- Conclusion and Future Work
Conclusion (1)

- We create a big data benchmark suite from web search engines
  - Data generation tools and six workloads

- First open-source project on big data benchmarking
  - [http://prof.ict.ac.cn/BigDataBench](http://prof.ict.ac.cn/BigDataBench)

- Welcome downloading
Conclusions (2)

The peak performance of big data systems is dependent on applications and data volumes.

The peak data processing rates of big data systems.
Conclusions (3)

Big data impose challenges to experiment methodologies in architecture researches.
Future Work (1)

- Release BigDataBench 2.0
  - Consider data variety!
    - Structured, semi-structure, and unstructured data
  - Include multimedia data and applications
  - Include online data analysis applications
Future Work (2)

- Include workloads in other important internet service domains
  - Electronic commence
  - Social network

- Internet services take up only a small part of big data applications
  - A long way to go!!!
Thank you!
Any questions?