Understanding Big Data Workloads on Modern Processors using BigDataBench

Jianfeng Zhan

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

Professor, ICT, Chinese Academy of Sciences
and University of Chinese Academy of Sciences

HPBDC 2015
Ohio, USA

中科院计算所
INSTITUTE OF COMPUTING TECHNOLOGY

Outline

- **BigDataBench Overview**
- **Workload characterization**
- **Multi-tenancy version**
- **Processors evaluation**

What is *BigDataBench*?

- An open source big data benchmarking project
  - http://prof.ict.ac.cn/BigDataBench
  - Search Google using "**BigDataBench**"

BigDataBench Detail

Methodology

- Five application domains
- Propose benchmark specifications for each domain

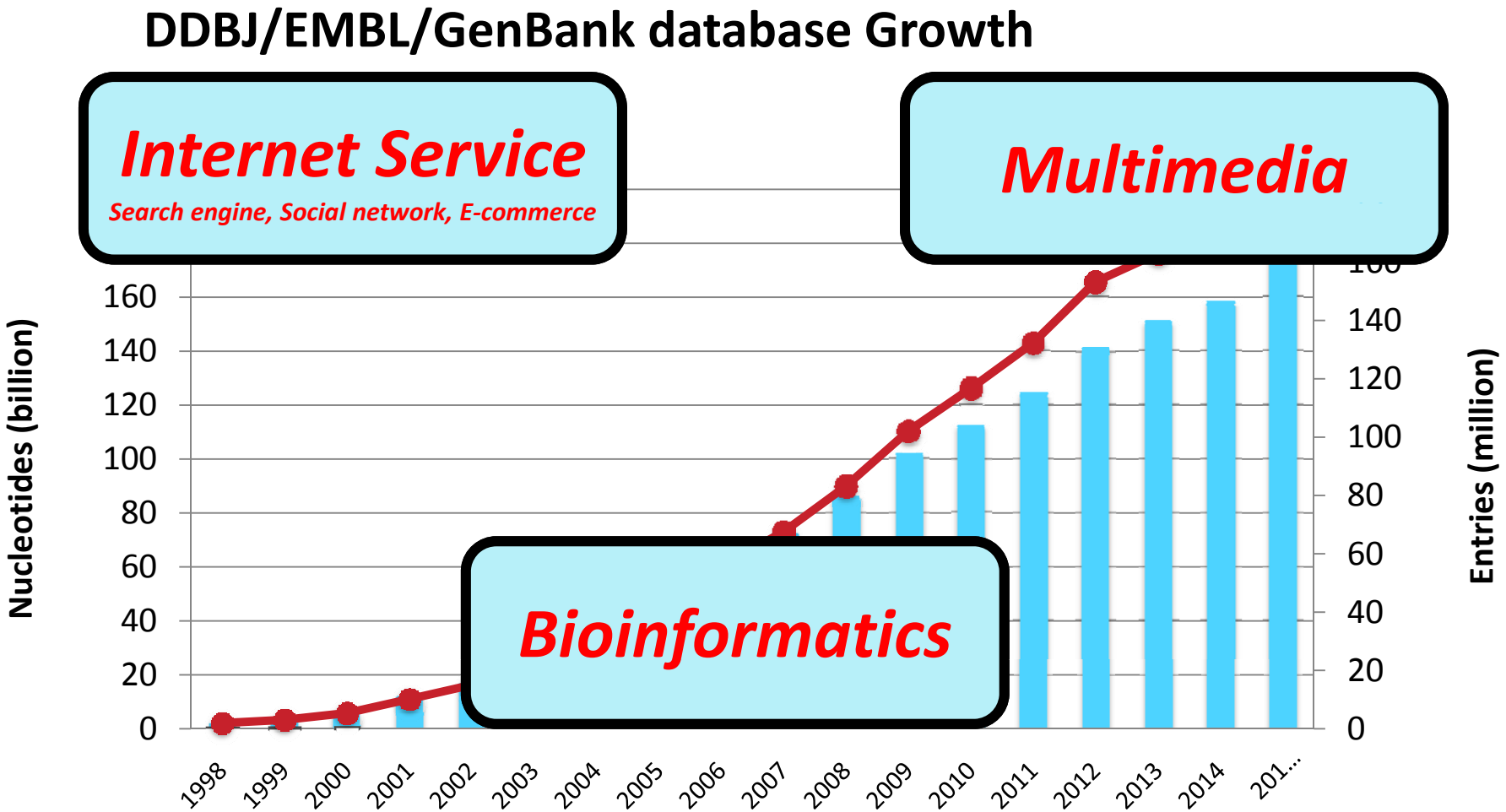
Implementation

- 14 Real world data sets & 3 kinds of big data generators
- 33 Big data workloads with diverse implementation

Specific-purpose Version

- BigDataBench subset version

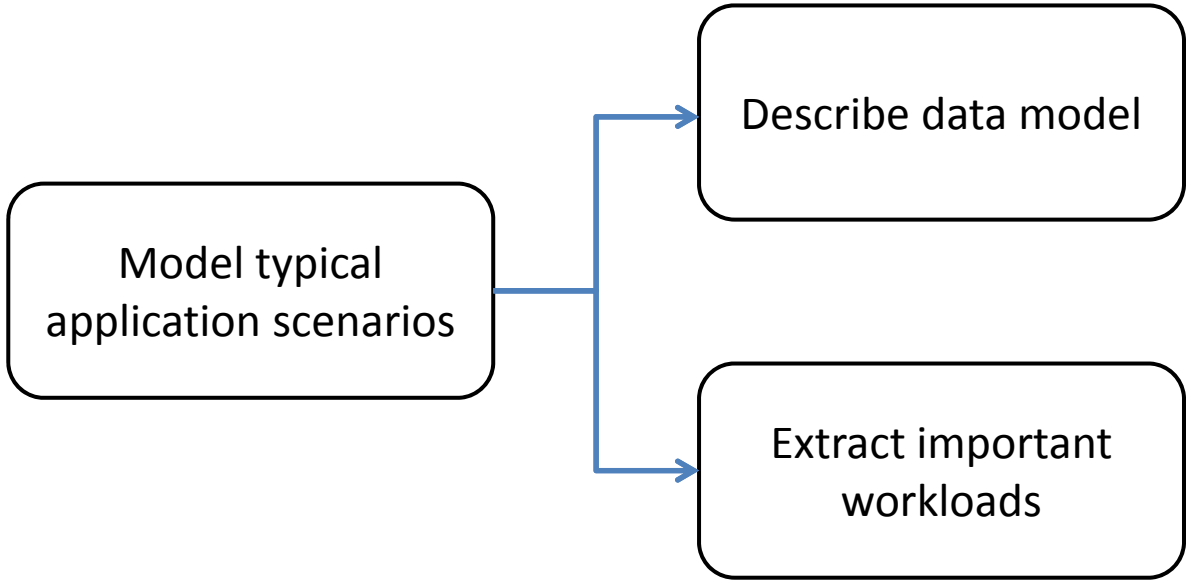
Five Application Domains



http://www.ddbj.nig.ac.jp/breakdown_stats/dbgrowth-e.html#dbgrowth-graph

Benchmark specification

- Guidelines for BigDataBench implementation
  - Data model
  - workloads



BigDataBench Details

Methodology

- Five application domains
- Benchmark specification for each domain

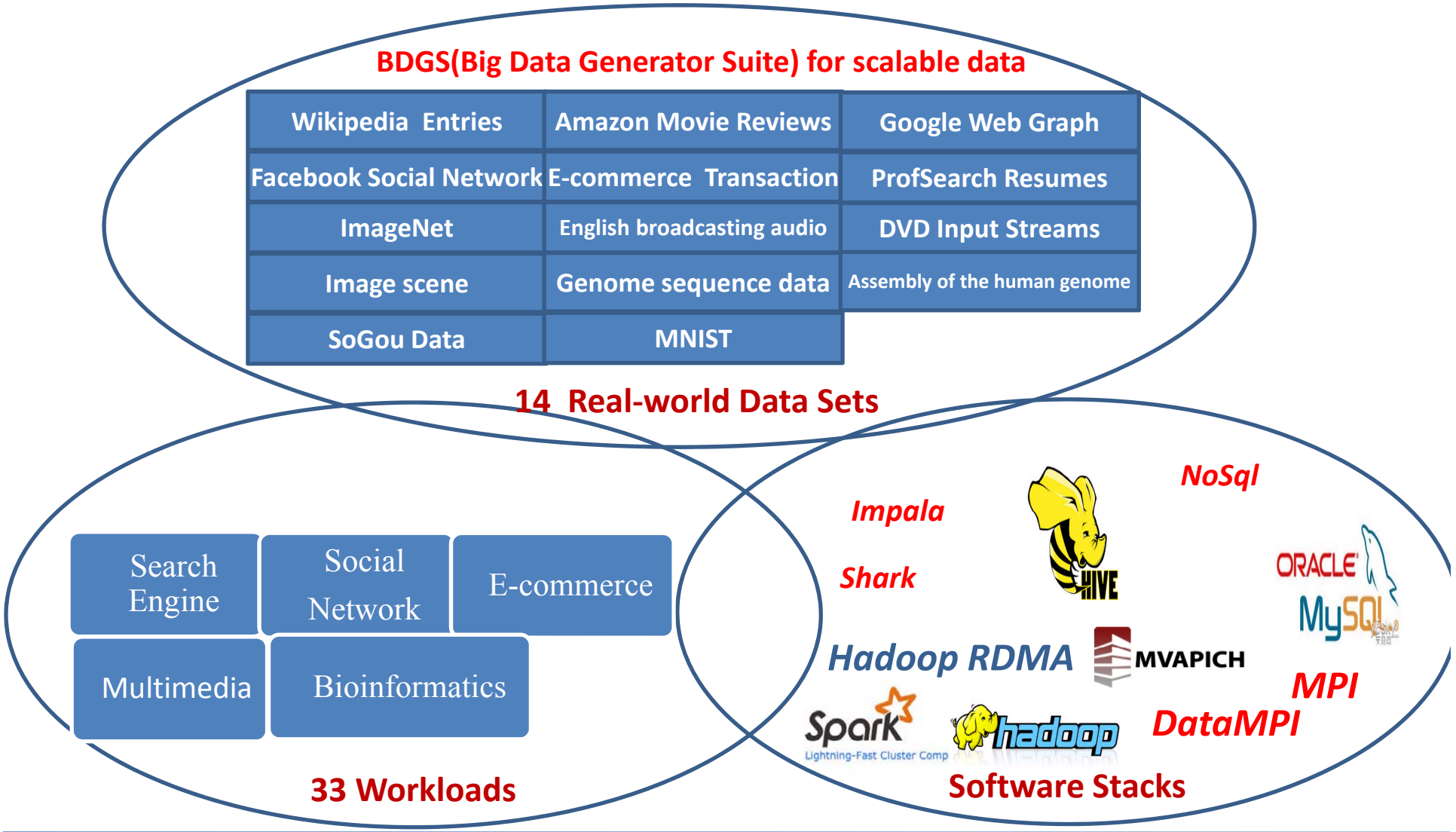
Implementation

- 14 Real world data sets & 3 kinds of big data generators
- 33 Big data workloads with diverse implementation

Specific-purpose Version

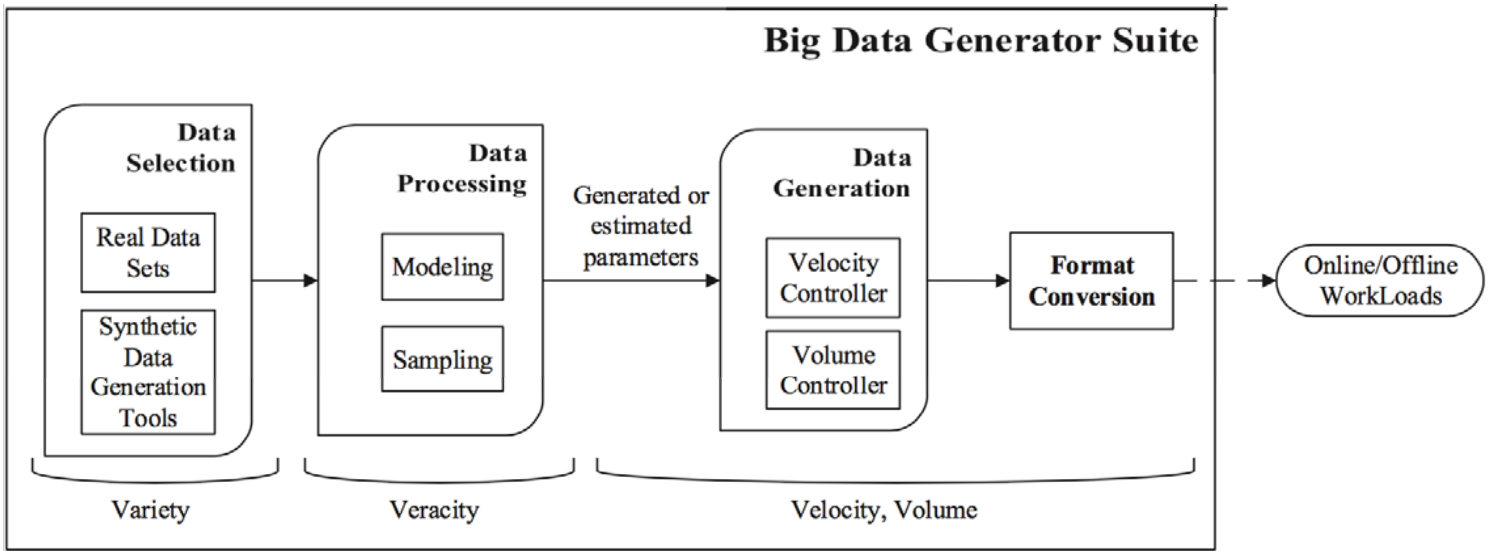
- BigDataBench subset version

BigDataBench Summary



Big Data Generator Tool

- 3 kinds of big data generators
  - Preserving original characteristics of real data
  - Text/Graph/Table generator



BigDataBench Details

Methodology

- Five application domains
- Benchmark specification for each domain

Implementation

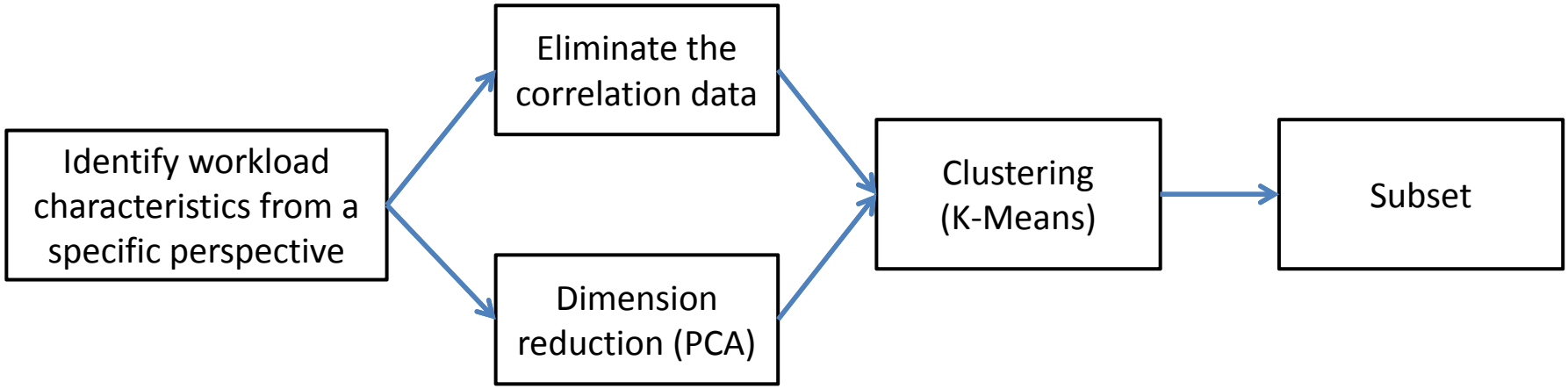
- 14 Real world data sets & 3 kinds of big data generators
- 33 Big data workloads with diverse implementations

Specific-purpose Version

- BigDataBench subset version

BigDataBench Subset

- Motivation
  - Expensive to run all the benchmarks for system and architecture researches
    - multiplied by different implementations
    - BigDataBench 3.0 provides about 77 workloads



Why BigDataBench?

	Specification	Application domains	Workload Types	Work loads	Scalable data sets (from real data)	Multiple implementations	Multitenancy	Subsets	Simulator version
BigDataBench	Y	Five	Four[1]	33	8	Y	Y	Y	Y
BigBench	Y	One	Three	10	3	N	N	N	N
Cloud-Suite	N	N/A	Two	8	3	N	N	N	Y
HiBench	N	N/A	Two	10	3	N	N	N	N
CALDA	Y	N/A	One	5	N/A	Y	N	N	N
YCSB	Y	N/A	One	6	N/A	Y	N	N	N
LinkBench	Y	N/A	One	10	N/A	Y	N	N	N
AMP Benchmarks	Y	N/A	One	4	N/A	Y	N	N	N

[1] *The four workloads types include Offline Analytics, Cloud OLTP, Interactive Analytics and Online Service*

BigDataBench Users

- http://prof.ict.ac.cn/BigDataBench/users/
- Industry users
  - Accenture, BROADCOM, SAMSUMG, Huawei, IBM
- China's first industry-standard big data benchmark suite
  - http://prof.ict.ac.cn/BigDataBench/industry-standard-benchmarks/
- About 20 academia groups published papers using BigDataBench

BigDataBench Publications

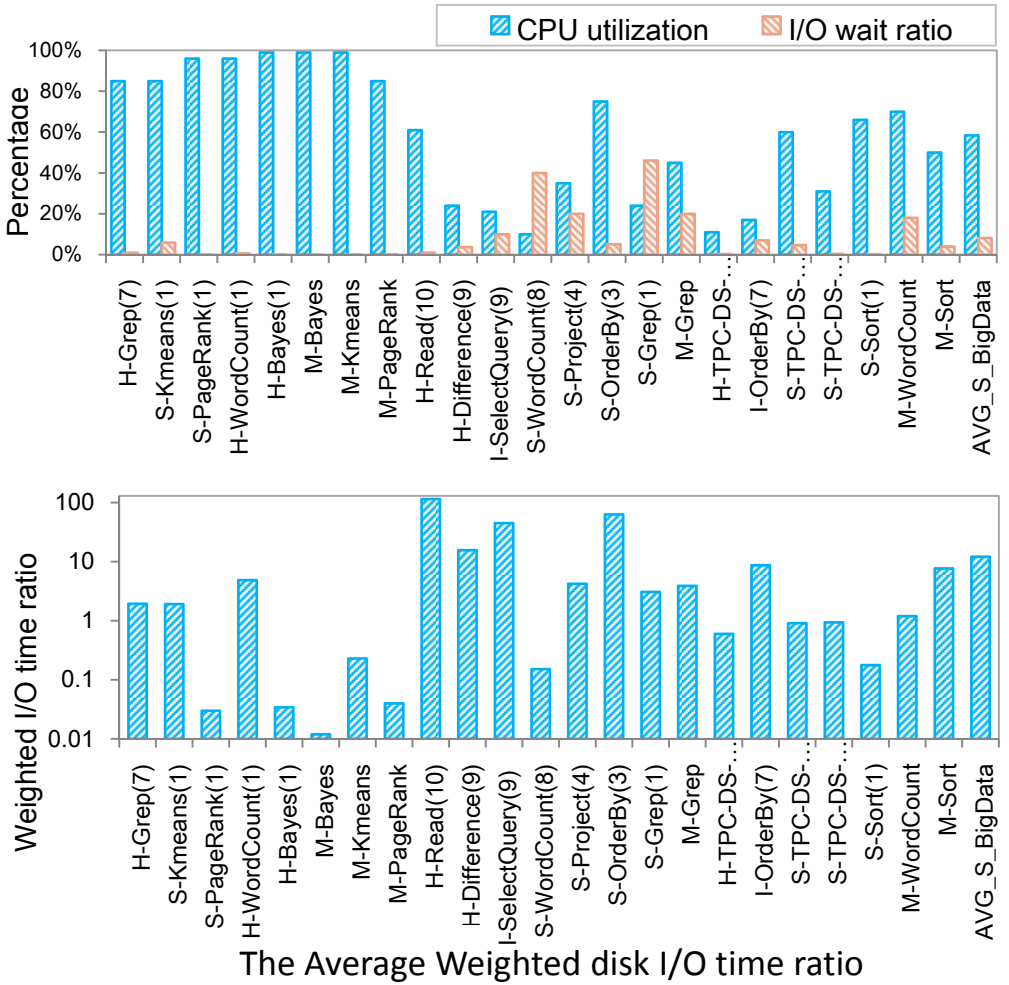
- BigDataBench: a Big Data Benchmark Suite from Internet Services. 20th IEEE International Symposium On High Performance Computer Architecture (HPCA-2014).
- Characterizing data analysis workloads in data centers. 2013 IEEE International Symposium on Workload Characterization (IISWC 2013)（**Best paper award**）
- BigOP: generating comprehensive big data workloads as a benchmarking framework. 19th International Conference on Database Systems for Advanced Applications (DASFAA 2014)
- BDGS: A Scalable Big Data Generator Suite in Big Data Benchmarking. The Fourth workshop on big data benchmarking (WBDB 2014)
- Identifying Dwarfs Workloads in Big Data Analytics arXiv preprint arXiv:1505.06872
- BigDataBench-MT: A Benchmark Tool for Generating Realistic Mixed Data Center Workloads arXiv preprint arXiv:1504.02205

Outline

- **BigDataBench Overview**
- **Workload characterization**
- **Multi-tenancy version**
- **Processors evaluation**

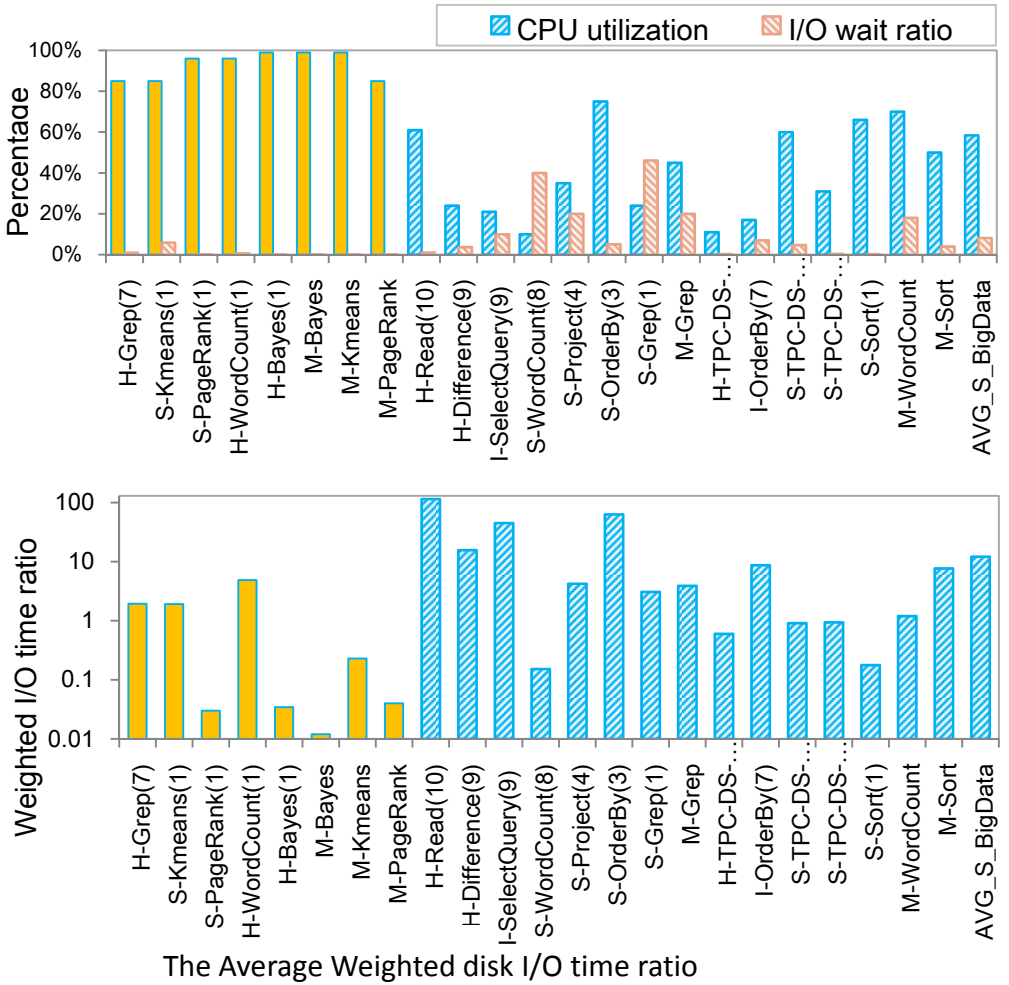
System Behaviors

- Diversified system level behaviors:



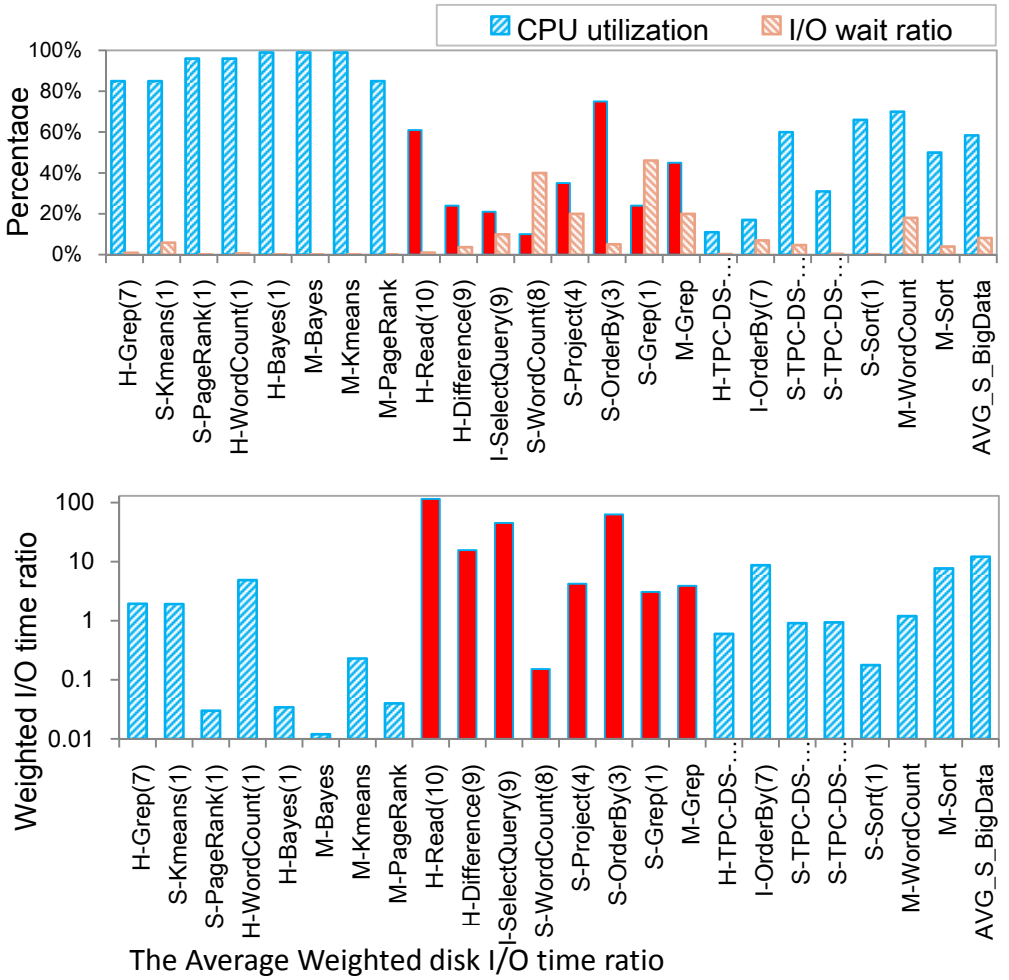
System Behaviors

- Diversified system level behaviors:
  - High CPU utilization & less I/O time



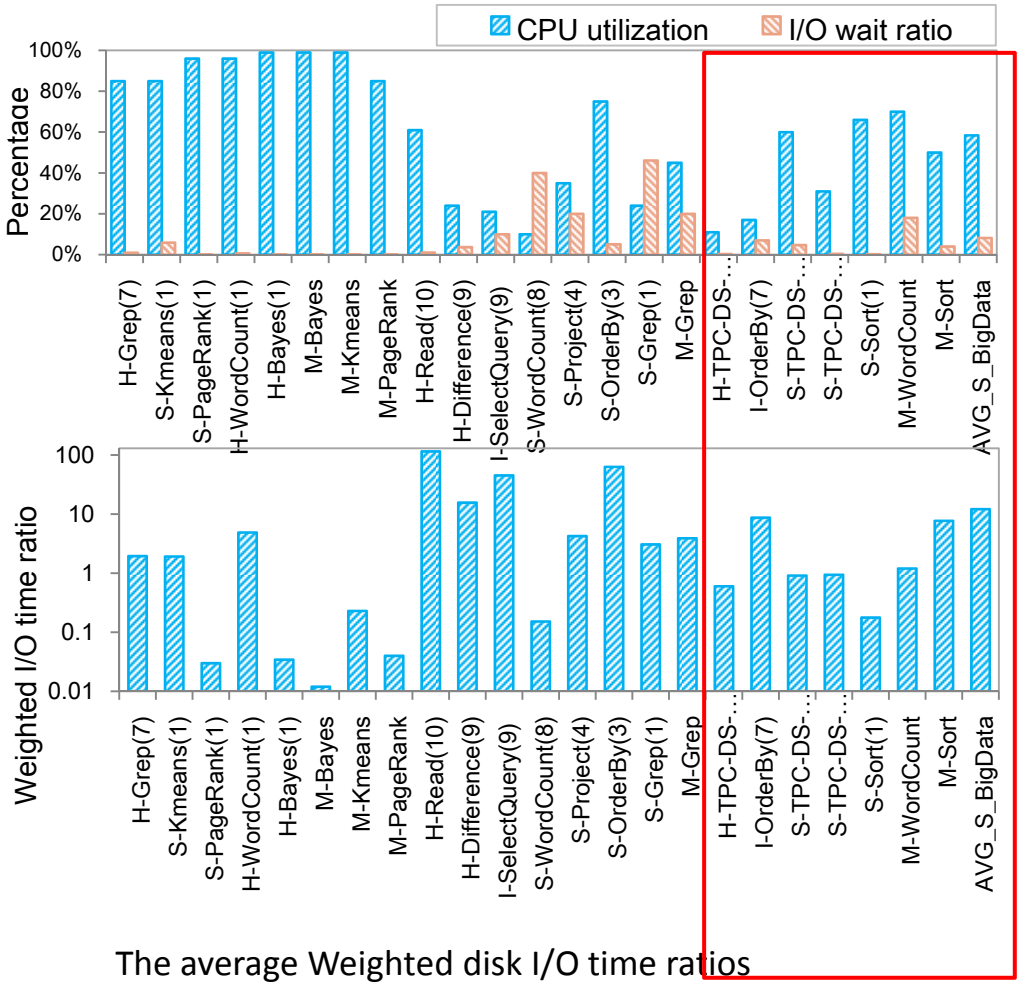
System Behaviors

- Diversified system level behaviors:
  - High CPU utilization & less I/O time
  - Low CPU utilization relatively and lots of I/O time



System Behaviors

- Diversified system level behaviors:
  - High CPU utilization & less I/O time
  - Relatively low CPU utilization & lots of I/O time
  - Medium CPU utilization & I/O time



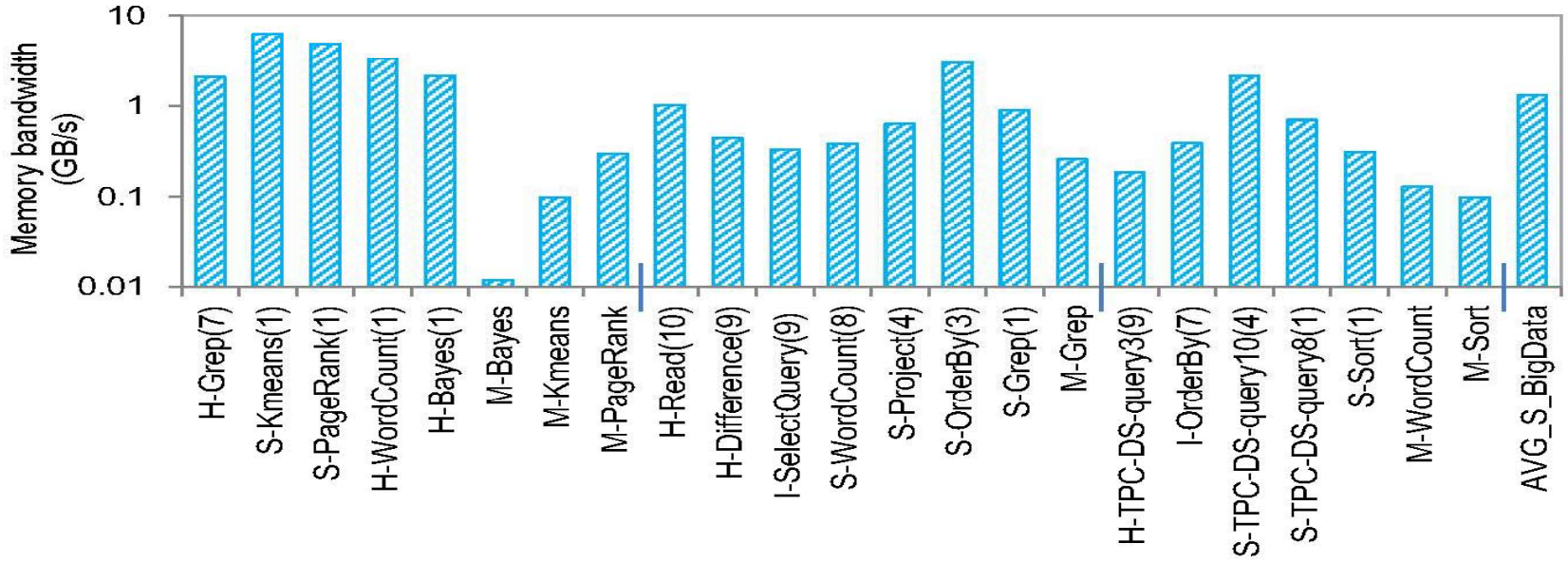
The average Weighted disk I/O time ratios

Workloads Classification

- From perspective of system behaviors:
  - System behaviors vary across different workloads
  - Workloads are divided into 3 categories:

Type	Workloads
CPU Intensive	H-Grep, S-Kmeans, S-PageRank, H-WordCount, H-Bayes, M-Bayes, M-Kmeans and M-PageRank
I/O Intensive	H-Read, H-Difference, I-SelectQuery, S-WordCount, S-Project, S-OrderBy, M-Grep and S-Grep
Hybrid	H-TPC-DS-query3, I-OrderBy, S-TPC-DS-query10, S-TPC-DS-query8, S-Sort, M-WordCount and M-Sort

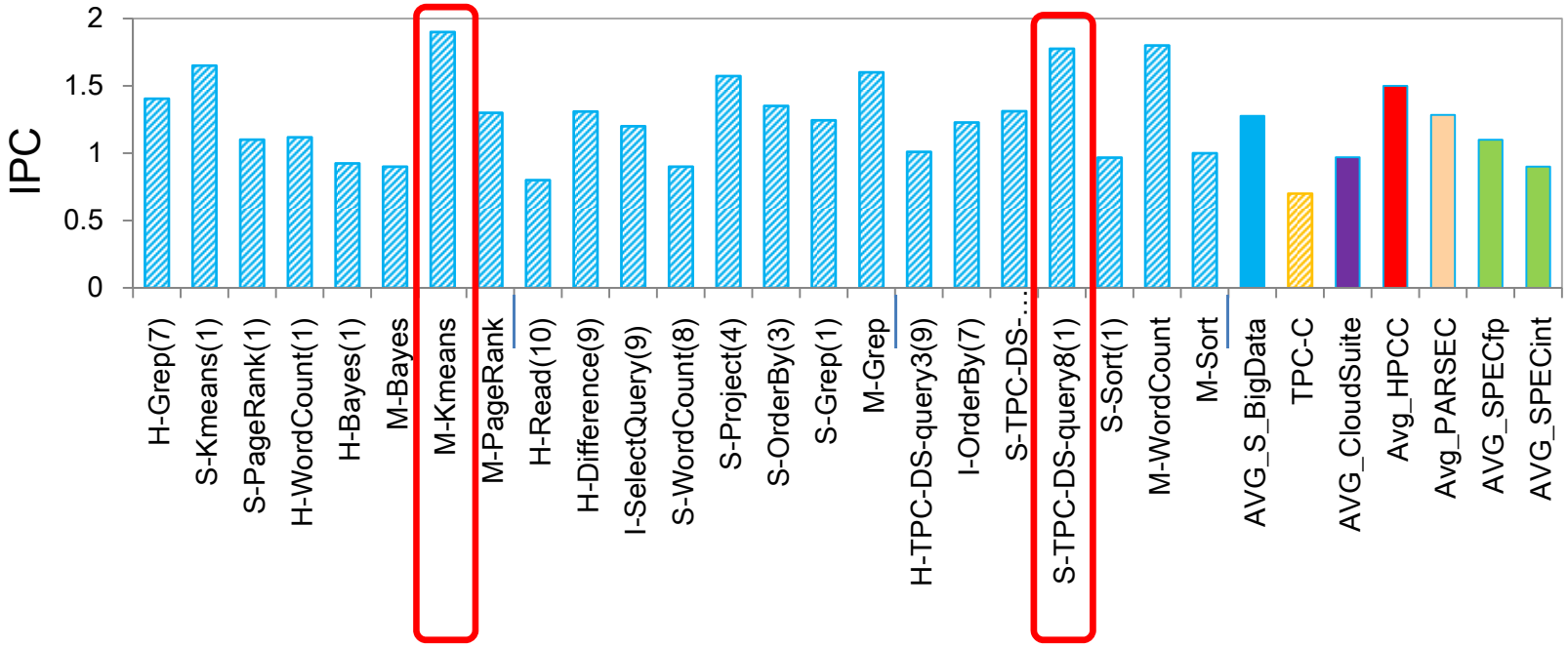
Off-Chip Bandwidth



Most of CPU-intensive workloads have higher off-chip bandwidth (3GB/s), Maximum is 6.2GB/s; Other workloads have lower off-chip bandwidth (0.6GB/s).

MPI based workloads need low memory bandwidth.

IPC of BigDataBench vs. other benchmarks

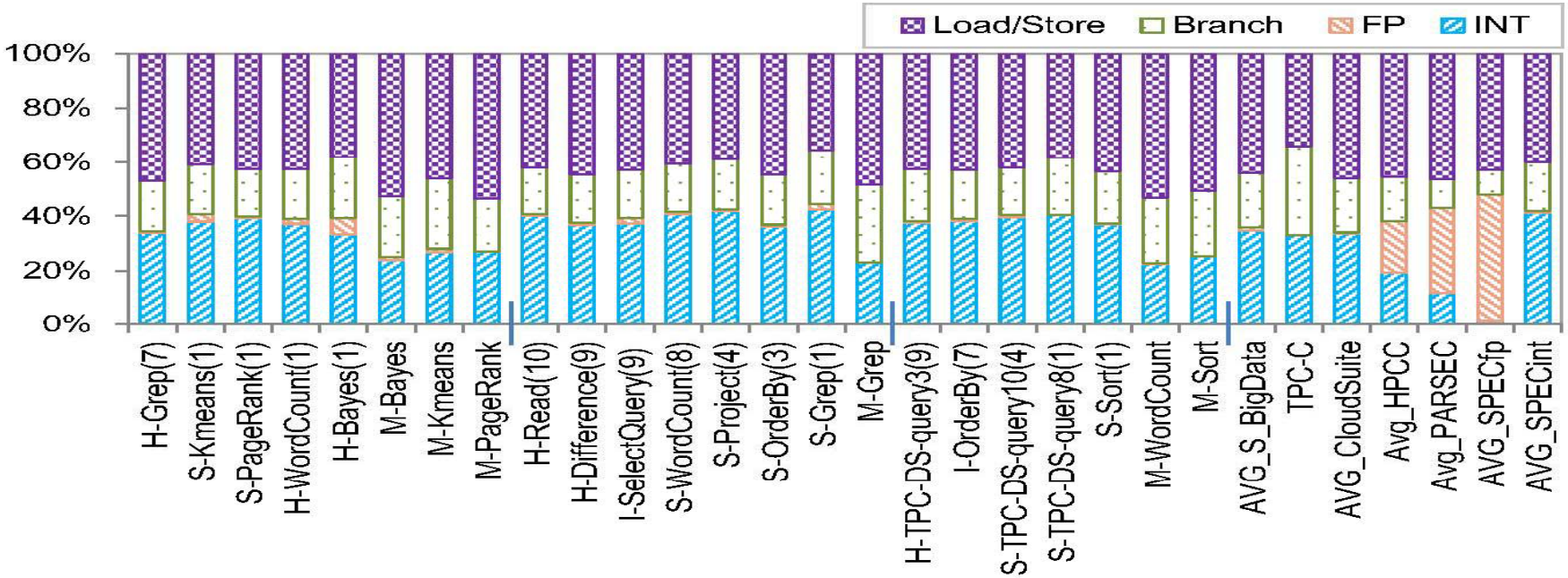


The average IPC of the big data workloads is larger than that of CloudSuite, SPECFP and SPECINT, similar with PARSEC and slightly lower than HPCC

The avrerage IPC of BigDataBench is 1.3 times of that of CloudSuite

Some workloads have high IPC (M_Kmeans, S-TPC-DS-Query8)

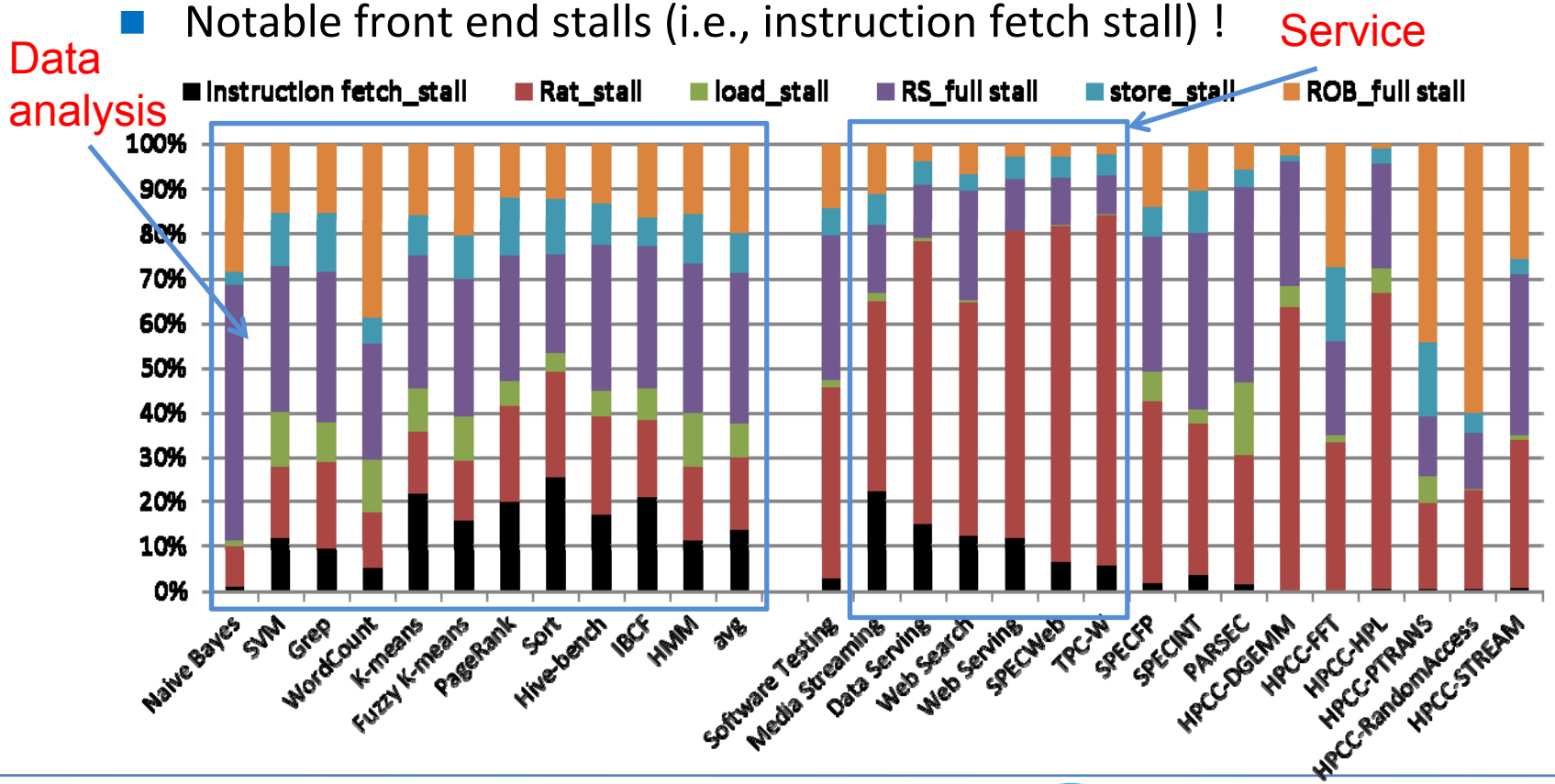
Instructions Mix of BigDataBench vs. other benchmarks



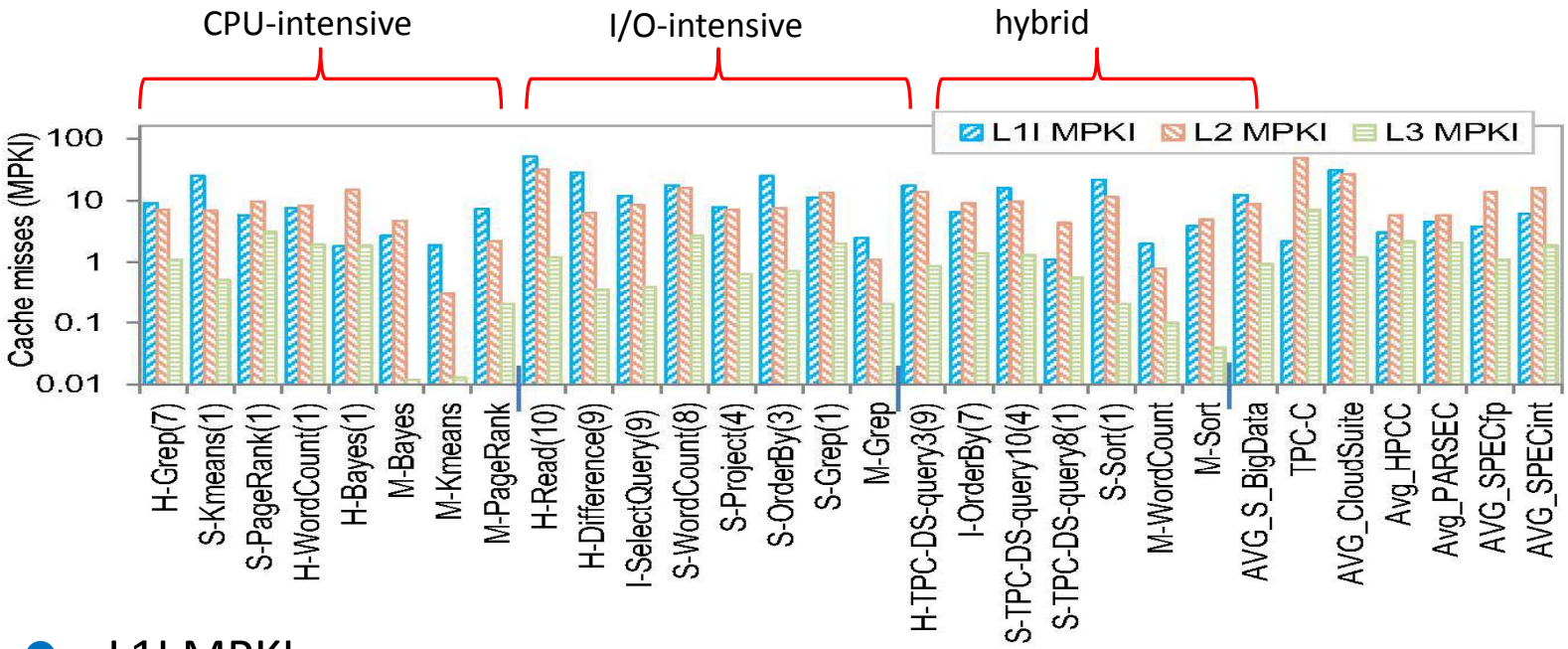
- **Big data workloads are data movement dominated computing with more branch operations**
  - 92% percentage in terms of instruction mix (Load + Store + Branch + data movements of INT)

Pipeline Stalls

- The service workloads have more RAT (Register Allocation Table) stalls
- The data analysis workloads have more RS (Reservation Station) and ROB (ReOrder Buffer) full stalls
- Notable front end stalls (i.e., instruction fetch stall) !



Cache Behaviors of BigDataBench

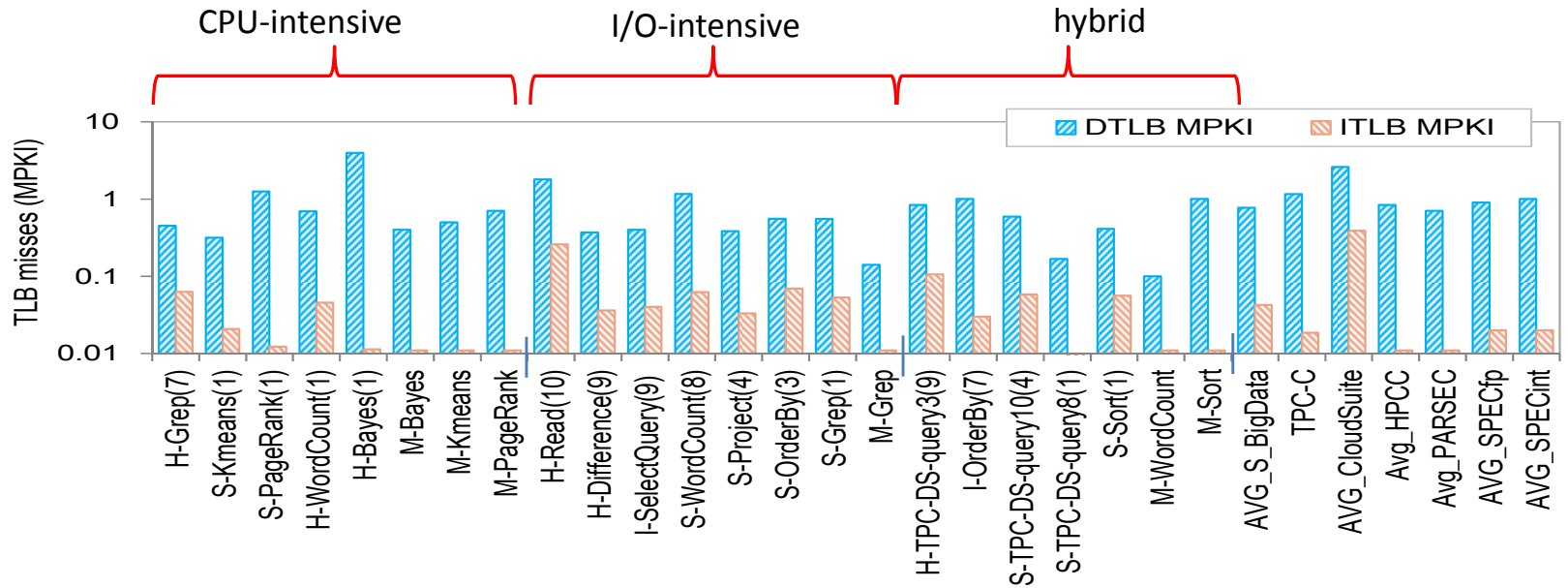


- L1I MPKI
  - ***Larger than traditional benchmarks, but lower than that of CloudSuite (12 vs. 31)***
  - ***Different among big data workloads***
    - *CPU-intensive(8), I/O intensive(22), and hybrid workloads(9)*
  - ***One order of magnitude differences among diverse implementations***
    - *M_WordCount is 2, while H_WordCount is 17*

Cache Behaviors

- L2 Cache：
  - *The IO-intensive workloads undergo more L2 MPKI*
- L3 Cache:
  - *The average L3 MPKI of the big data workloads is lower than all of the other workloads*
- The underlying software stacks impact data locality
  - MPI workloads have better data locality and less cache misses

TLB Behaviors



- ITLB
  - *IO-intensive workloads undergo more ITLB MPKI.*
- DTLB
  - *CPU-intensive workloads have more DTLB MPKI.*

Our observations from BigDataBench

- Unique characteristics
  - Data movement dominated computing with more branch operations
    - 92% percentage in terms of instruction mix
  - Notable pipeline frontend stalls
- Different behaviors among Big Data workloads
  - Disparity of IPCs and memory access behaviors
    - CloudSuite is a subclass of Big Data
- Software stacks impacts
  - The L1I cache miss rates have one order of magnitude differences among diverse implementations with different software stacks.

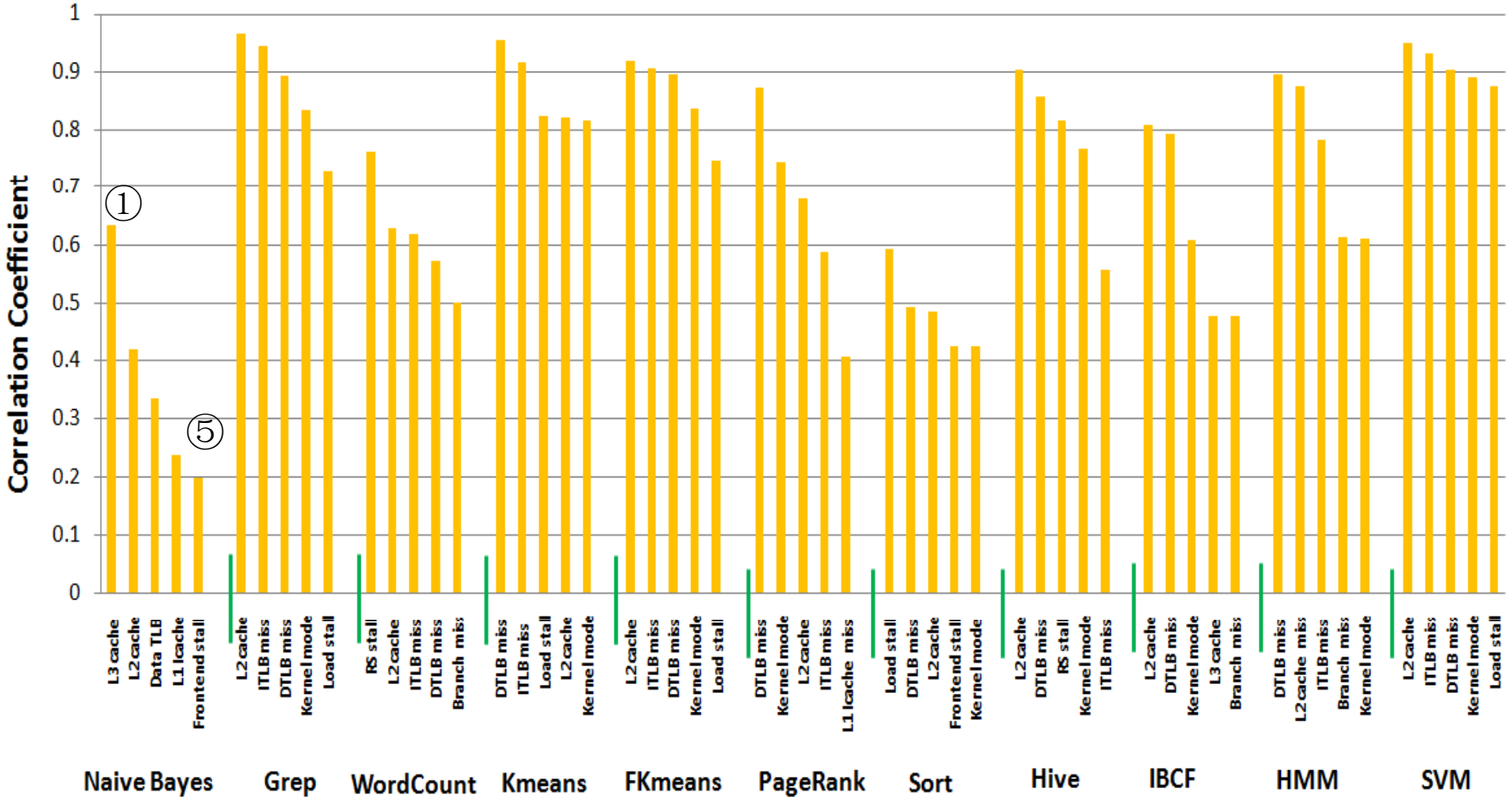
Correlation Analysis

- Compute the correlation coefficients of CPI with other micro-architecture level metrics.
  - Pearson's correlation coefficient:

$$\rho(X,Y) = corr(X,Y) = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}$$

  - Absolute value (from 0 to 1) shows the dependency:
    - The bigger the absolute value, the stronger the correlation.

Top five coefficients



Insights

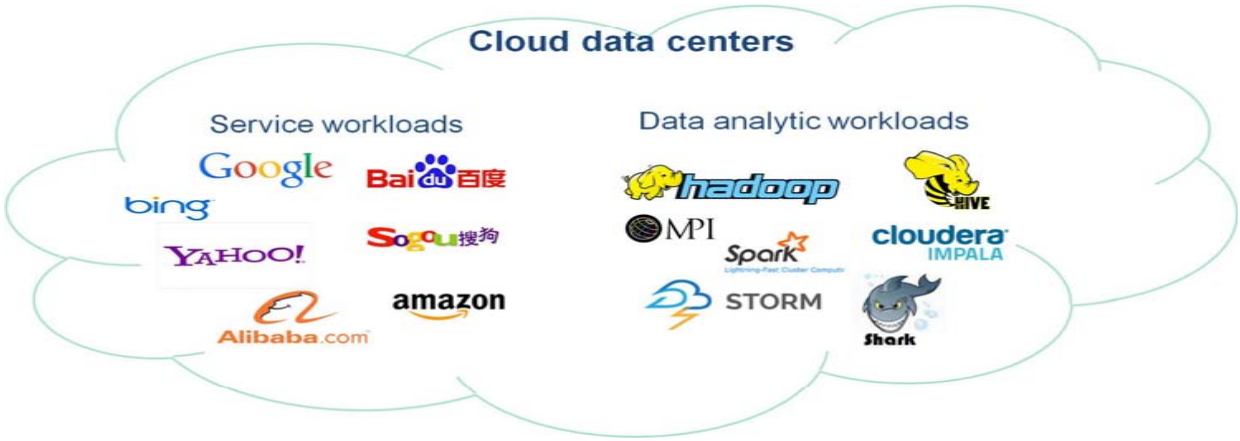
- Frontend stall does not have high correlation coefficient value for most of big data analytics workloads
  - Frontend stall is not the factor that affects the CPI performance most.
- L2 cache misses and TLB misses have high correlation coefficient values.
  - The long latency memory accesses (access L3 cache or memory) affect the CPI performance most and should be the optimization point with highest priority.

Outline

- **BigDataBench Overview**
- **Workload characterization**
- **Multi-tenancy version**
- **Processors evaluation**

Cloud Data Centers

- Two class of popular workloads
  - Long-running services
    - Search engines, E-commerce sites
  - Short-term data analytic jobs
    - Hadoop MapReduce, Spark jobs



Problem

- Existing benchmarks focus on specific types of workload
- Scenarios are not realistic
  - Does not match the typical data center scenario that mixes different percentages of tenants and workloads shareing the same computing infrastructure

Purpose of BigDataBench-MT

- Developing **realistic** benchmarks to reflect such practical scenarios of mixed workloads.
  - Both service and data analytic workloads
  - Dynamic scaling up and down

- The tool is publicly available from http://prof.ict.ac.cn/BigDataBench/multi-tenancyversion

What can you do with it?

- We consider two dimensions of the benchmarking scenarios
  - From **tenants'** perspectives
  - From **workloads'** perspectives

You can specify the tenants

- The number of tenants
  - **Scalability Benchmark**: How many tenants are able run in parallel ?
- The priorities of tenants
  - **Fairness Benchmark**: How fair is the system, i.e., are the available resources equally available to all tenants? If tenants have different priorities ?
- Time line
  - How the number and priorities of tenants change over time?

You can specify the workloads

- Data characteristics
  - Data type, source
  - Input/output data volumes, distributions
- Computation semantics
  - Source code
  - Big data software stacks
- Job arrival patterns
  - Arrival rate
  - Arrival sequence

Two major challenges

- Heterogeneity of real workloads
  - Different workload types
    - ✓ e.g. CPU or I/O intensive workloads
  - Different software stacks
    - ✓ e.g. Hadoop, Spark, MPI
- Workload dynamicity hidden in real-world traces
  - Arrival patterns
    - ✓ Request/Job submitting time and sequences
  - Job input sizes
    - ✓ e.g. ranging from KB to ZB

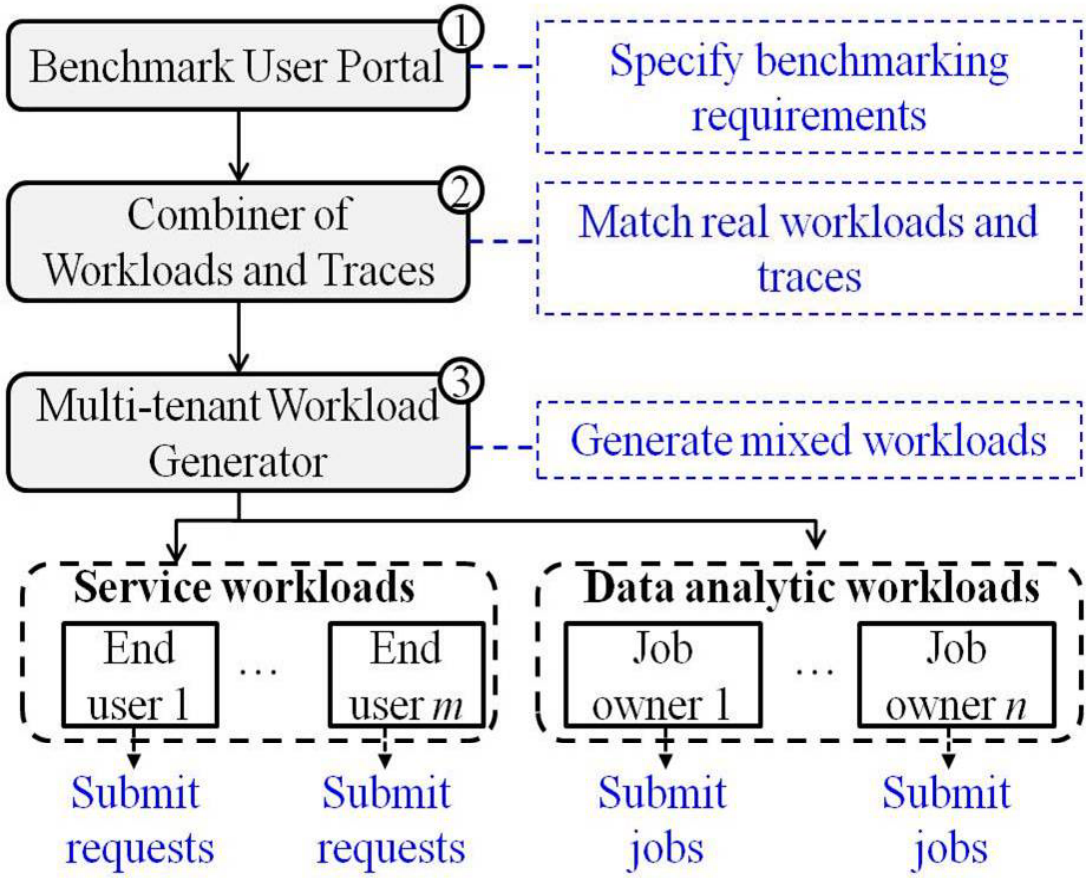
Existing big data benchmarks

Benchmarks	Actual workloads	Real workload traces	Mixed workloads
AMPLab benchmark, Linkbench , Bigbench , YCSB, CloudSuite	Yes	No	No
GridMix , SWIM	No	Yes	No

- How to generate real workloads on the basis of real workload traces, is still an open question.

System Overview

- Three modules
  - Benchmark User Portal
    - A visual interface
  - Combiner of Workloads and Traces
    - A matcher of real workloads and traces
  - Multi-tenant Workload Generator
    - A multi-tenant workload generator



Key technique: Combination of real and synthetic data analytic jobs

- **Goal**: Combining the arrival patterns extracted from real traces with real workloads.
- **Problem**: Workload traces only contain anonymous jobs whose workload types and/or input data are unknown.

Solution: the first step

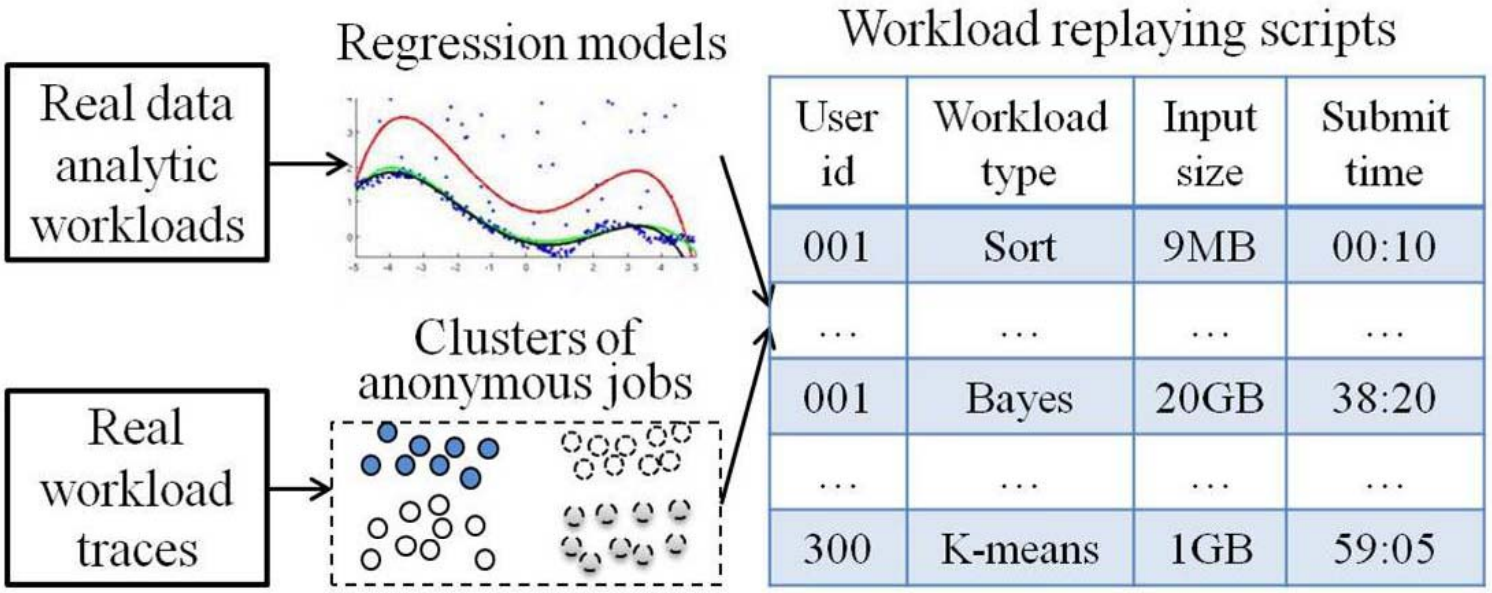
- Deriving the workload characteristics of both real and anonymous jobs

TABLE. Metrics to represent workload characteristics

Metric	Description
Execution time	Measured in seconds
CPU usage	Total CPU time per second
Memory usage	Measured in GB
CPI	Cycles per instruction
MAI	The number of memory accesses per instruction

Solution: the second step

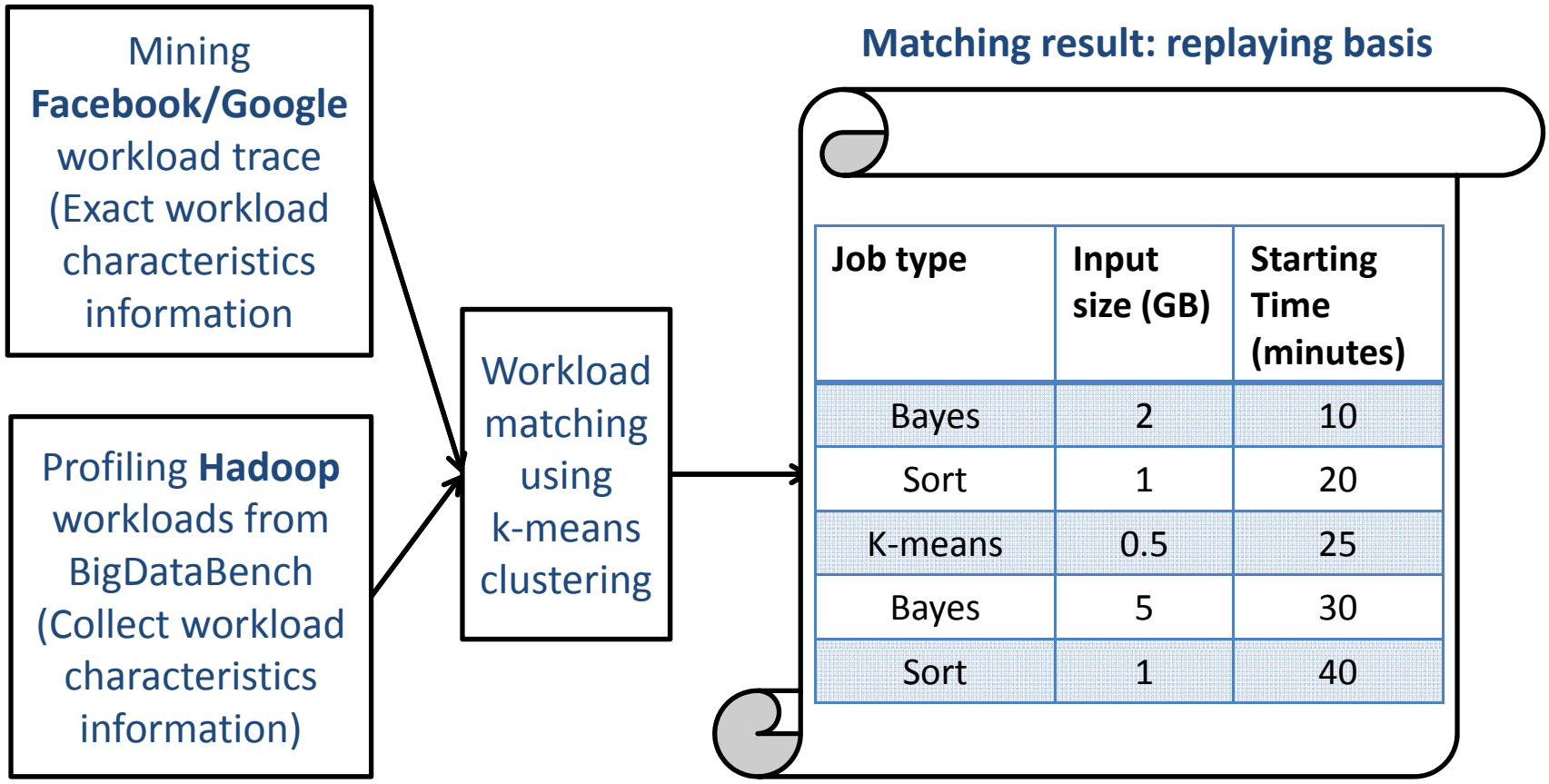
- Matching both types of jobs whose workload characteristics are sufficiently similar



User id	Workload type	Input size	Submit time
001	Sort	9MB	00:10
…	…	…	…
001	Bayes	20GB	38:20
…	…	…	…
300	K-means	1GB	59:05

An Example

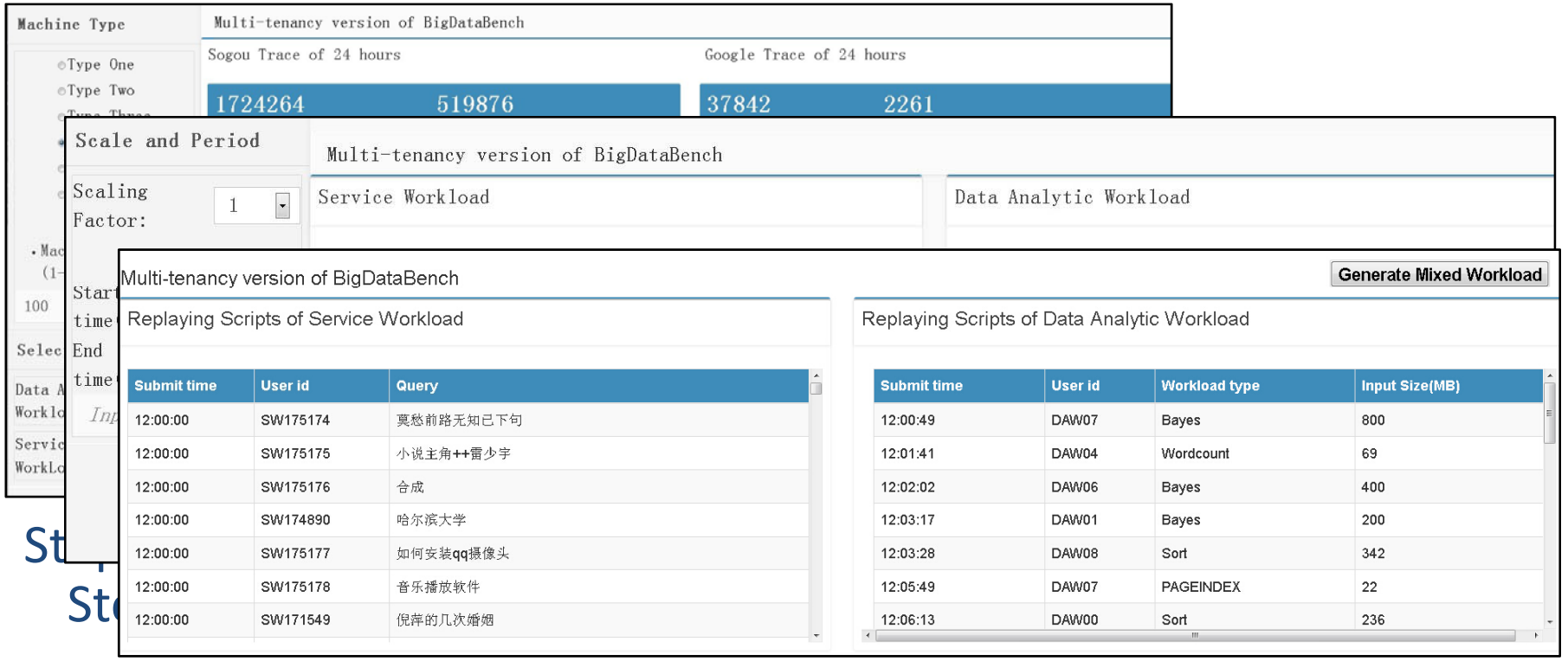
- An example of matching Hadoop workloads



Job type	Input size (GB)	Starting Time (minutes)
Bayes	2	10
Sort	1	20
K-means	0.5	25
Bayes	5	30
Sort	1	40

System demostration

- Three steps to generate a mix of search service and Hadoop MapReduce jobs
- Traces : 24-hour Sogou user query logs and Google cluster trace.



Step 3-Generation of mixed workloads

Workloads and traces in BigDataBench-MT

- Multi-tenancy V1.0 releases:

Workloads	Software stack	Workload trace
Nutch Web Search	Apache Tomcat 6.0.26, Search Server（Nutch）	Sogou (http://www.sogou.com/labs/dl/q-e.html)
Hadoop	Hadoop 1.0.2	Facebook (https://github.com/SWIMProjectUCB/SWIM/wiki)
Shark	Shark 0.8.0	Google data center (https://code.google.com/p/googleclusterdata/)

Outline

- **BigDataBench Overview**
- **Workload characterization**
- **Multi-tenancy version**
- **Processors evaluation**

Core Architecture

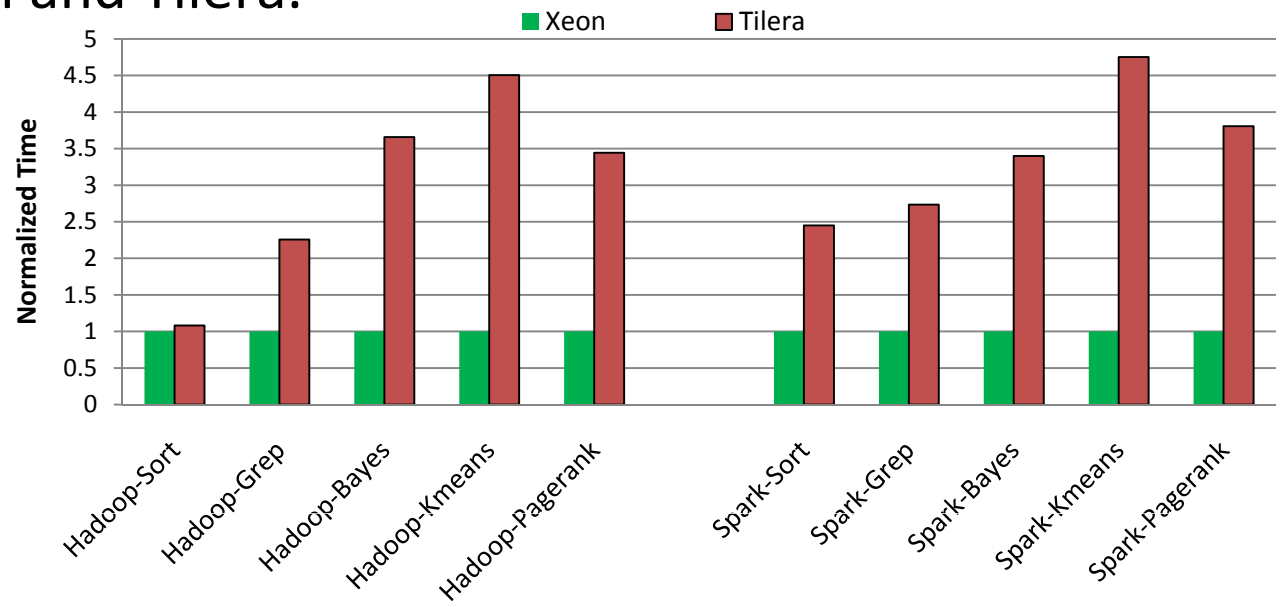
- Multi brawny core (Xeon E5645, 2.4 GHz)
  - 6 Out-of-Order cores
  - Dynamic Multiple Issue (supper scalar)
  - Dynamic Overclocking
  - Simultaneous multithreading
- Many wimpy core architecture (Tile-Gx36, 1.2 GHz):
  - 36 In-Order cores
  - Static Multiple Issue (VLIW)

Experiment methodology

- User real hardware instead of simulation
- Real power consumption measurement instead of modeling
- Saturate CPU performance by:
  - Isolate the processor behavior
    - Over-provisions the disk I/O subsystem by using RAM disk
  - Optimize benchmarks
    - Tune the software stack parameters
    - JVM flags to performance

Execution time

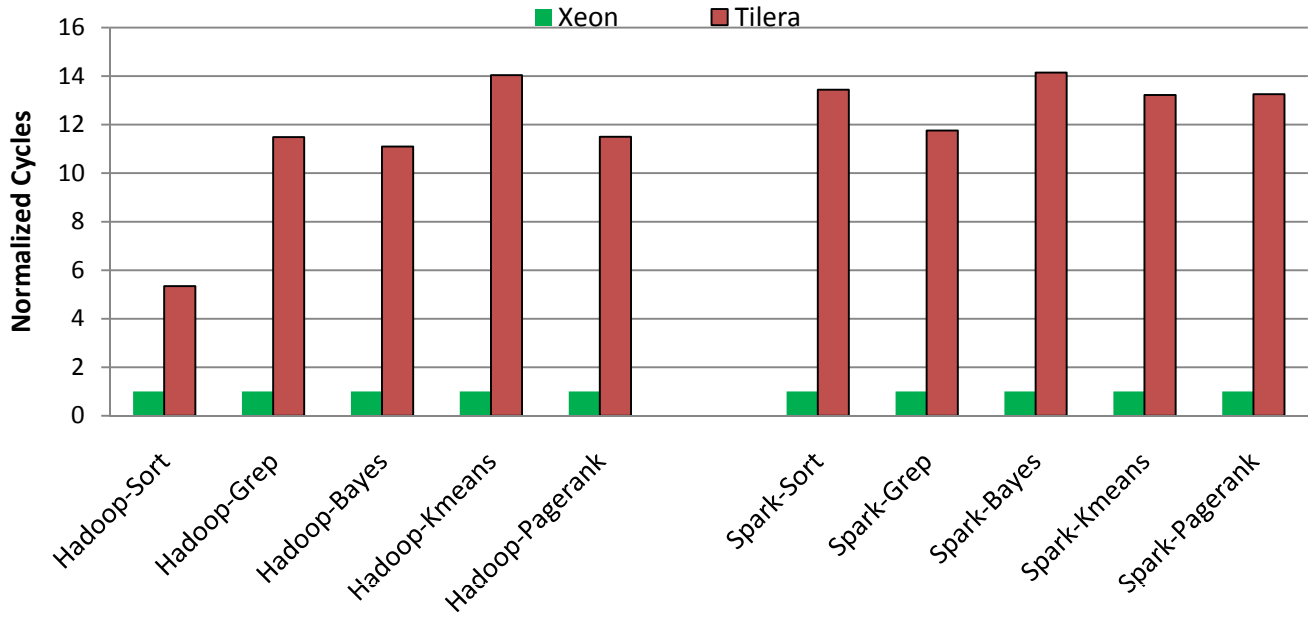
- For Hadoop based sort, the performance gap is about 1.08×.
- For the other workloads, more than 2× gaps exist between Xeon and Tilera.



- From the perspective of execution time, the Xeon processor is better than Tilera processor all the time.

Cycle Counts

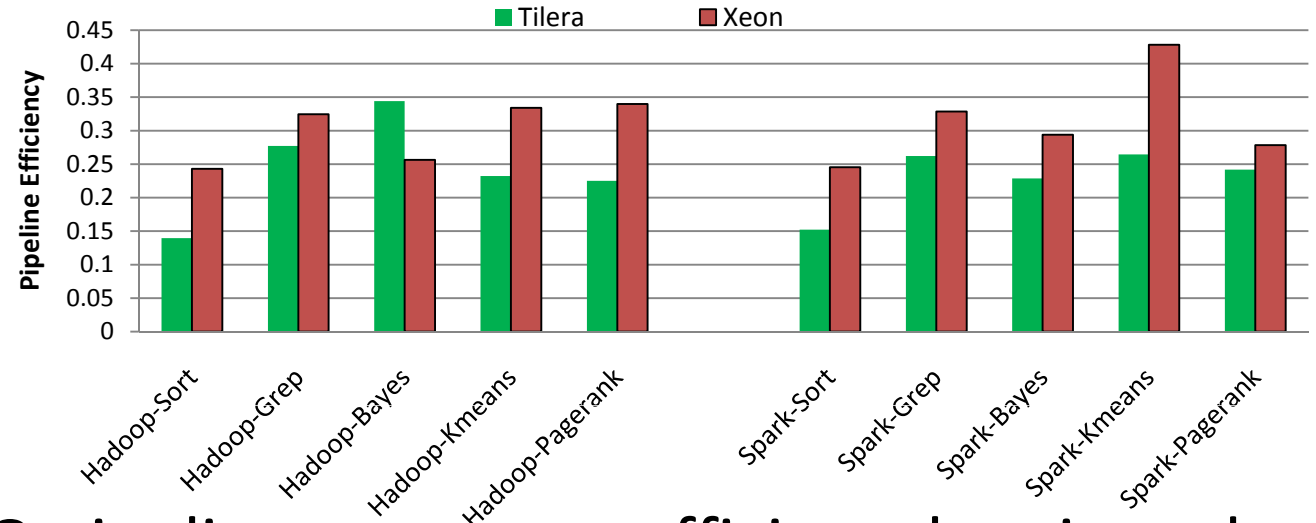
- There are huge cycle count gaps between Xeon and Tilera ranging from 5.3 to 14.
- Tilera need more cycles to complete the same amount of work.



Pipeline Efficiency

- The theoretical IPC:
  - ➢ Xeon: 4 instructions per cycle
  - ➢ Tilera: 1 instruction bundle per cycle
- Pipeline efficiency:

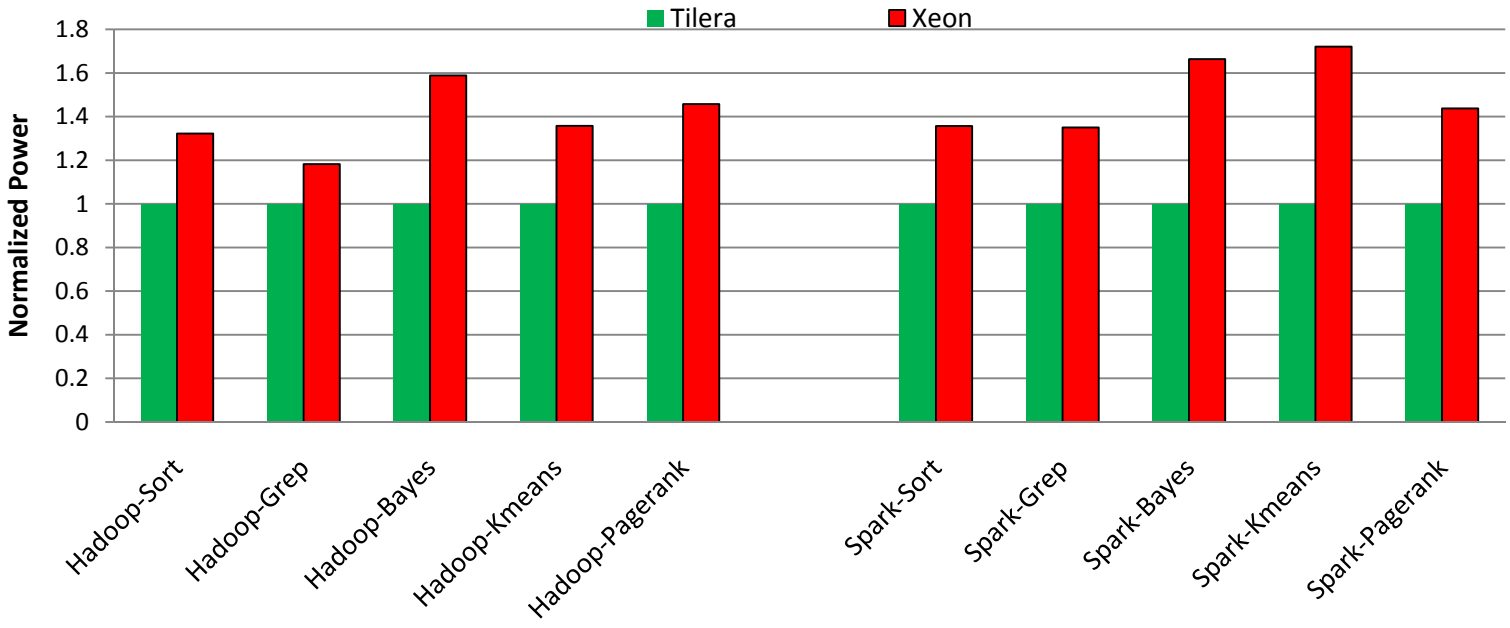
$$Pipeline\ Efficiency = \frac{Application\ IPC}{The\ Theoretical\ IPC}$$



- OoO pipelines are more efficient than in-order ones

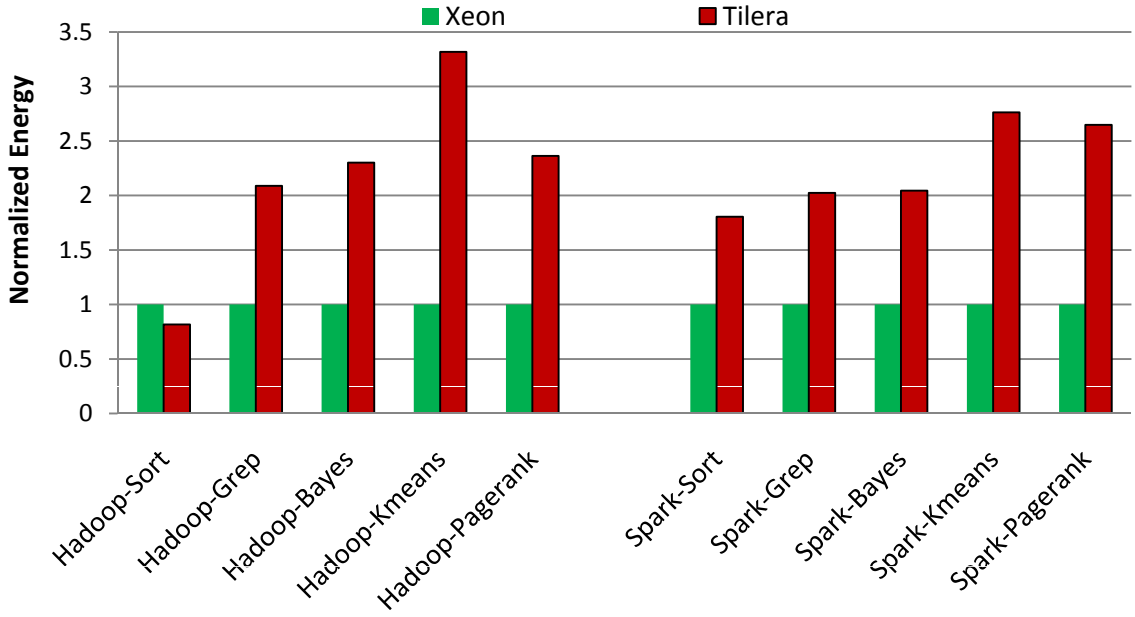
Power Consumption

- Tilera is power optimized.
- Xeon consumes more power.



Energy Consumption

- Hadoop based sort consumes less energy on Tilera than on Xeon
  - Hadoop sort is an extremely I/O intensive workloads.
- Tilera consumes more energy than Xeon to complete the same amount of work for most big data workloads
  - The longer execution time offsets the lower power design



Total Cost of Ownership (TCO) Model[*]

- Three-year depreciation cycle
- Hardware costs associated with individual components
  - CPU
  - Memory
  - Disk
  - Board
  - Power
  - Cooling

[*] K. Lim et al. **Understanding and designing new server architectures for emerging warehouse-computing environments**. *ISCA 2008*

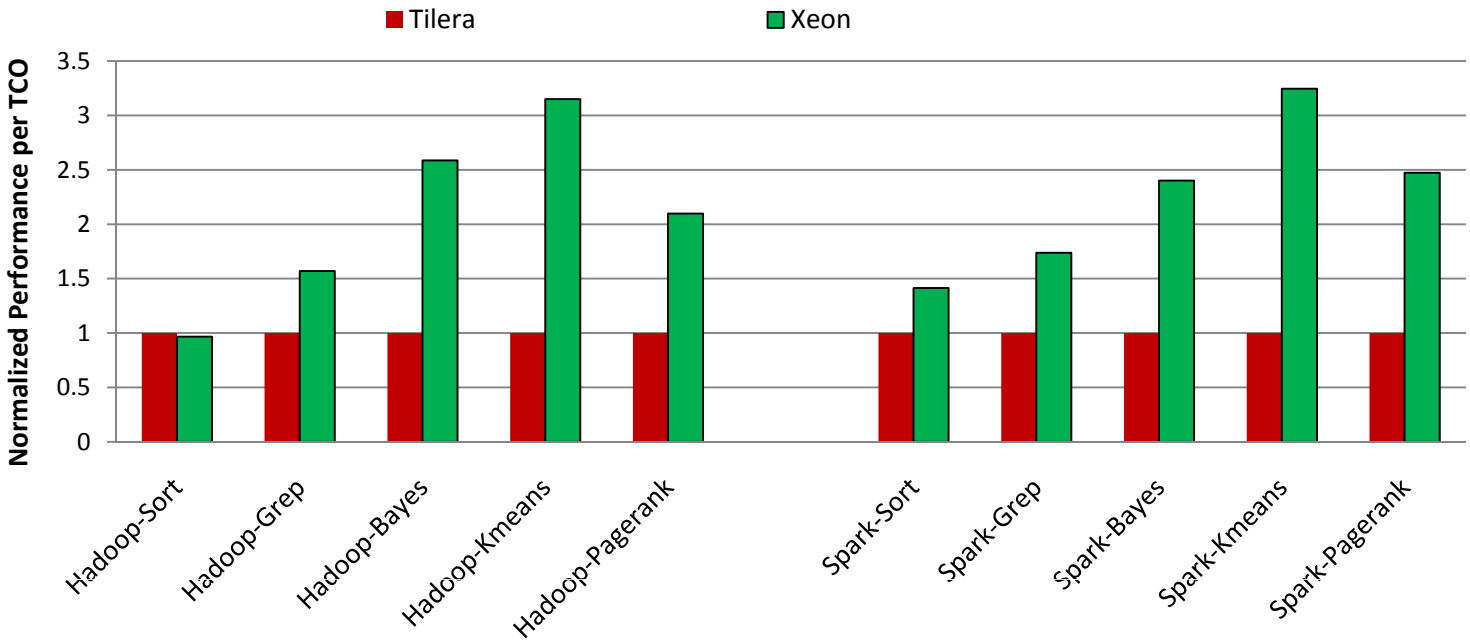
Cost model

Details	Xeon server	Tilera server
Total hardware cost ($)	1490	710
CPU	593	171
Memory	252	238
Disk	120	120
Board + mngmnt	275	80
Power + fans	250	101
Server Power (Watt)	117+CPU	78+CPU
CPU	Measured	Measured
Memory	24	22
Disk	16	16
Board + mngmnt	42	20
Power + fans	35	20

- The cost data originate from diverse sources:
  - Different vendors
  - Corresponding official websites
- Power and cooling:
  - An activity factor of 0.75

Performance per TCO

- Haoop-based Sort has higher performance per TCO on the Tilera.
- For other workloads, Xeon outperforms Tilera.



Key Takeaways

- Try using an open-source big data benchmark suite from http://prof.ict.ac.cn/BigDataBench
- Big Data: data movement dominated computing with more branch operations
  - 92% percentage in terms of instruction mix
- Multi-tenancy version: replaying mixed workloads according to publicly available workloads traces.
- Wimpy-core processors only suit a part of big data workloads.


QUESTIONS
And
Answers