Big data and AI Proxy Benchmarks for Simulation

Chen Zheng

http://prof.ict.ac.cn

ICT, Chinese Academy of Sciences

ASPLOS 2018, Williamsburg, VA, USA

中科院计算所

INSTITUTE OF COMPUTING TECHNOLOGY

Overview

- **Simulation of Big Data and AI Workloads**
  - **Challenges & Motivation**
- **Dwarf-based Simulation Benchmarks**
- **Evaluation on X86 Processor**
- **Case Study on ARM Processor**

Simulation for Big Data and AI

- Challenges
  - Simulators have limited supports on complex software stacks
    - For example: Hadoop modes
      - Standalone mode
      - Pseudo-distributed mode
      - Fully-distributed mode
  - Different modes have large behavior differences
- Long running time is unbearable
  - 1000+ times execution time than physical machine

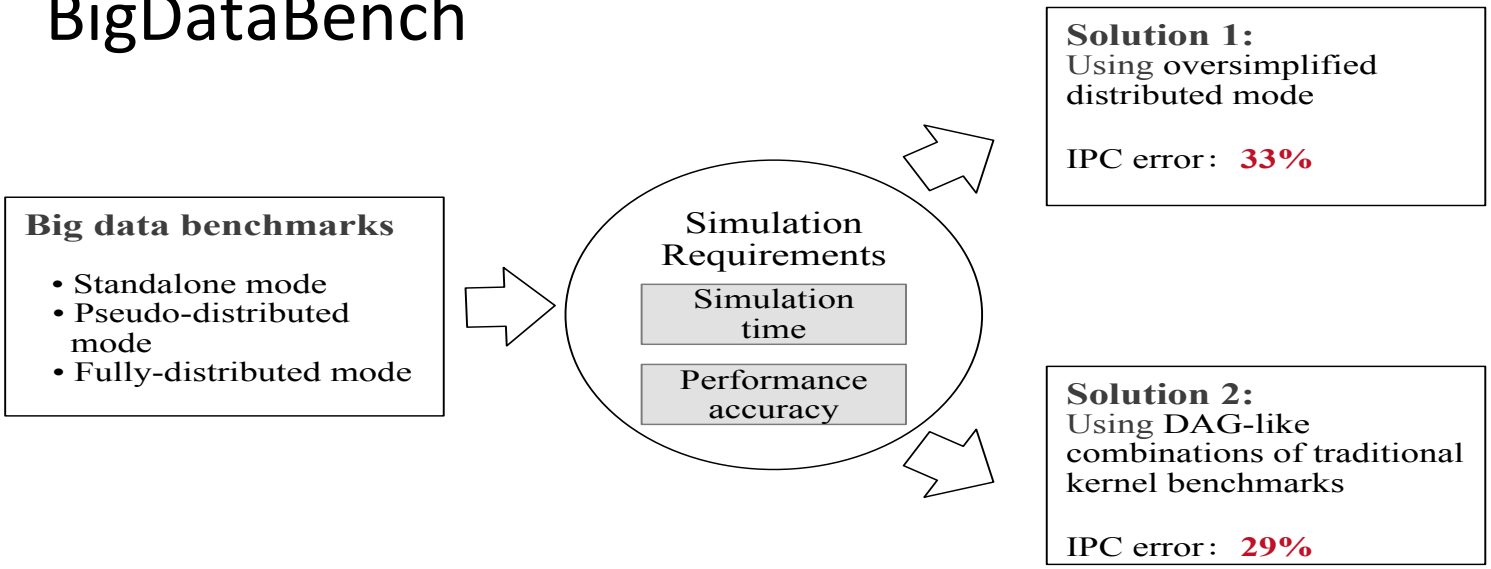
Motivation

- Traditional simulation method
  - A case-by-case simulation
    - Sampled trace or synthetic trace
  - Focus on specific workload and architecture

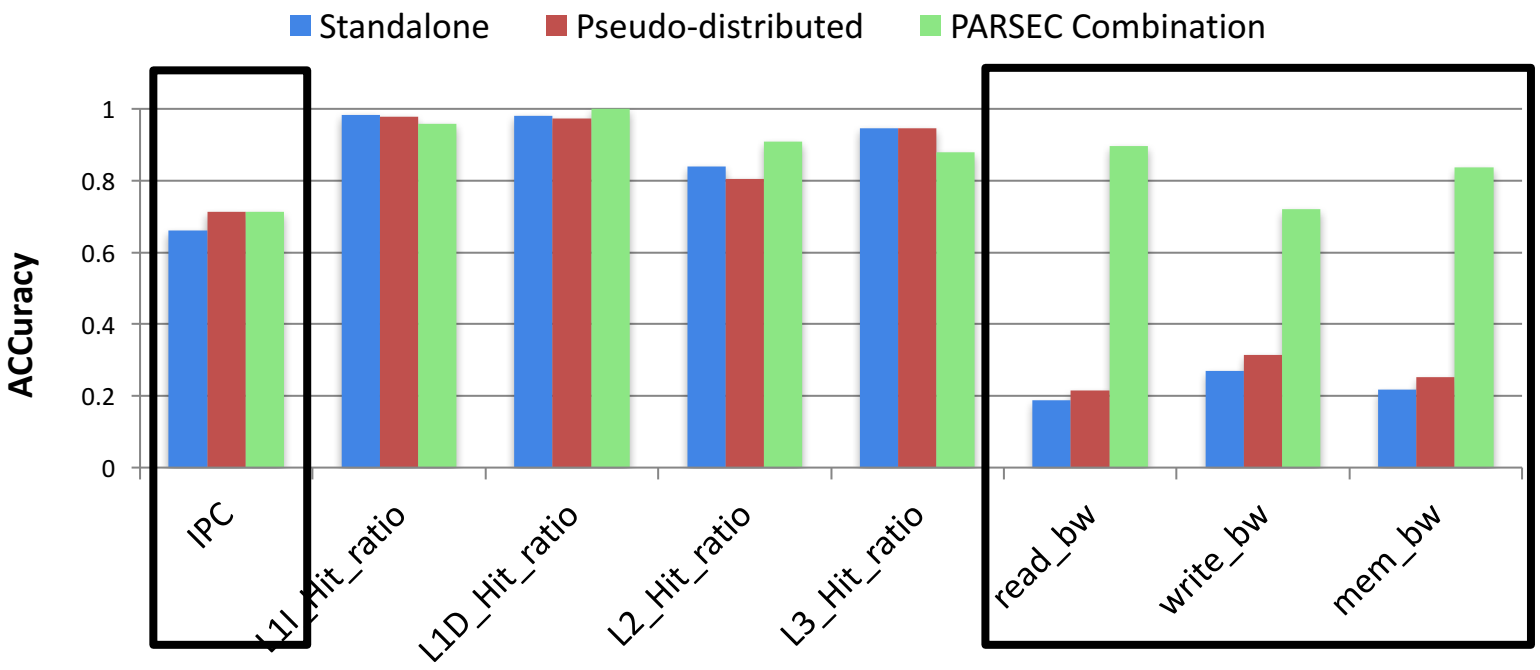
Not suit for big data scenario

Motivation (Cont')

- Two previous simulation solutions for big data analytics workloads
  - Using a fully-distributed mode Sort workload in BigDataBench



Data Accuracy of Previous Solution



- *read_bw: read memory bandwidth*
- *write_bw: write memory bandwidth*
- *mem_bw: all memory bandwidth*

Overview

- **Simulation of Big Data and AI Workloads**
  - **Challenges & Motivation**
- **Dwarf-based Simulation Benchmarks**
- **Evaluation on X86 Processor**
- **Case Study on ARM Processor**

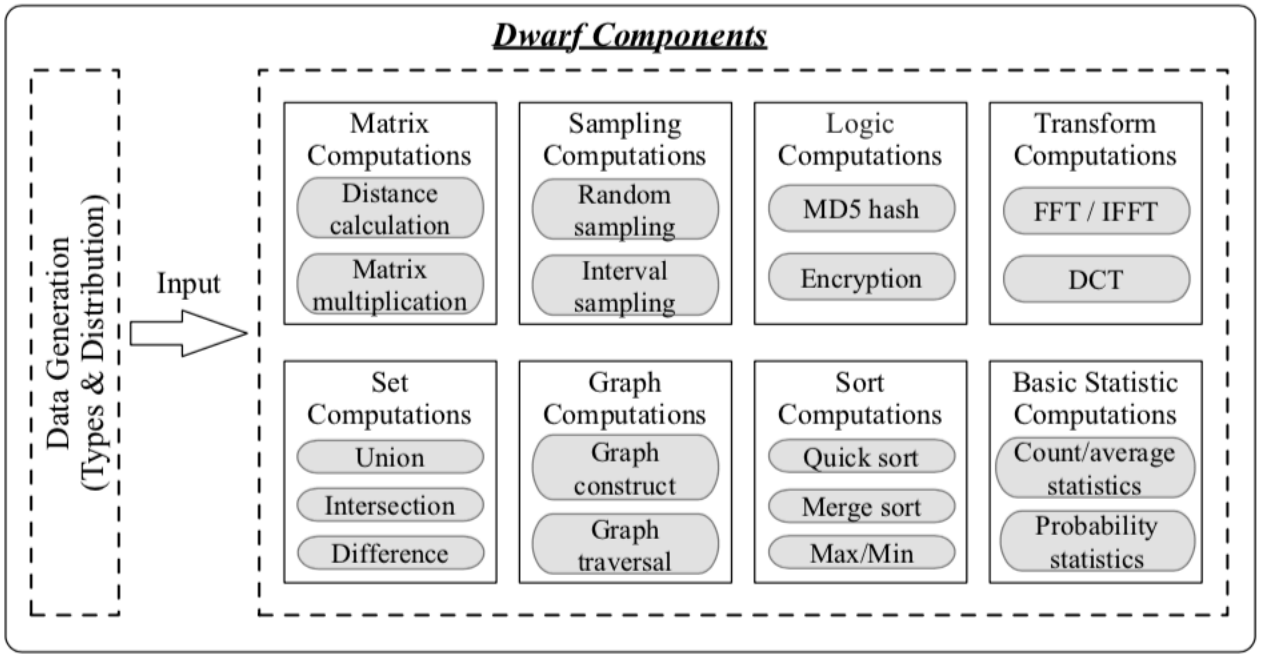
Dwarf-based Simulation

- Big data dwarfs
  - a minimum set to represent maximum patterns of big data analytics
- A light-weigh simulation benchmark on the basis of big data dwarfs
  - Shorten the simulation time by 100s times
  - Average micro-architectural data accuracy is above 90% on X86 and ARMv8 processors

Basic statistics computation	Sort computation	Transform computation	Logic computation
Graph computation	Sampling computation	Matrix computation	Set computation

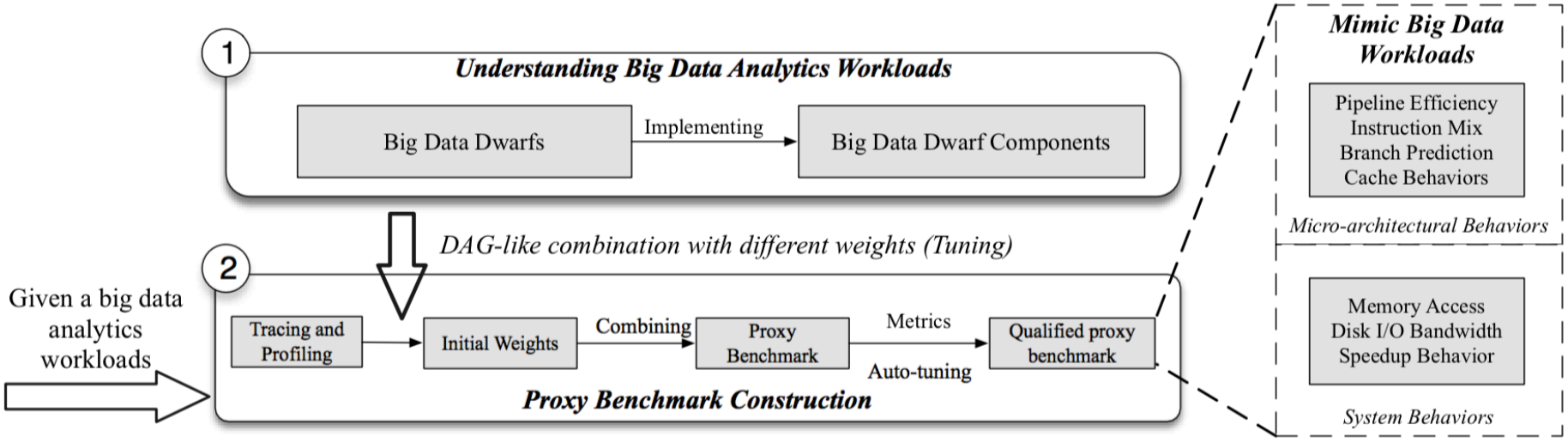
Dwarf Components

- Data generation tools
- Dwarf implementations (OpenMP & Pthreads)



Dwarf-based Simulation Methodology

- DAG-like combinations of dwarfs
  - Different weights
  - Computation logic



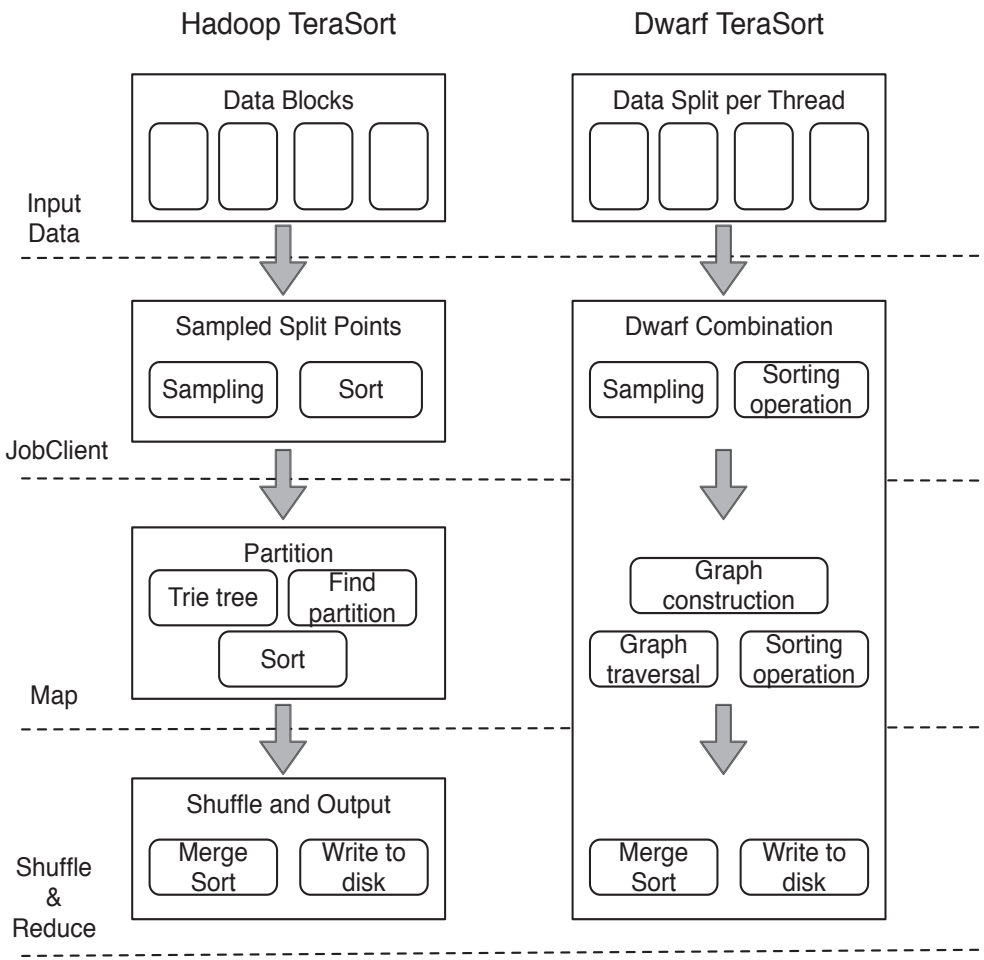
Methodology Comparison

- Traditional simulation methodology
  - Kernel benchmark
  - Synthetic trace
  - Synthetic benchmark

Methodology	Typical Benchmark	Input Data	Different Micro-architecture	Multi-core Scalability	System Evaluation	Accuracy
Kernel Benchmark	NPB [17]	Fixed	Recompile	Yes	Yes	Low
Synthetic Trace	SimPoint [19]	Fixed	Regenerate	No	No	High
Synthetic Benchmark	PerfProx [20]	Fixed	Regenerate	No	No	High
Dwarf-Based Proxy Benchmark	Dwarf Benchmark	On-demand	Auto-tuning	Yes	Yes	High

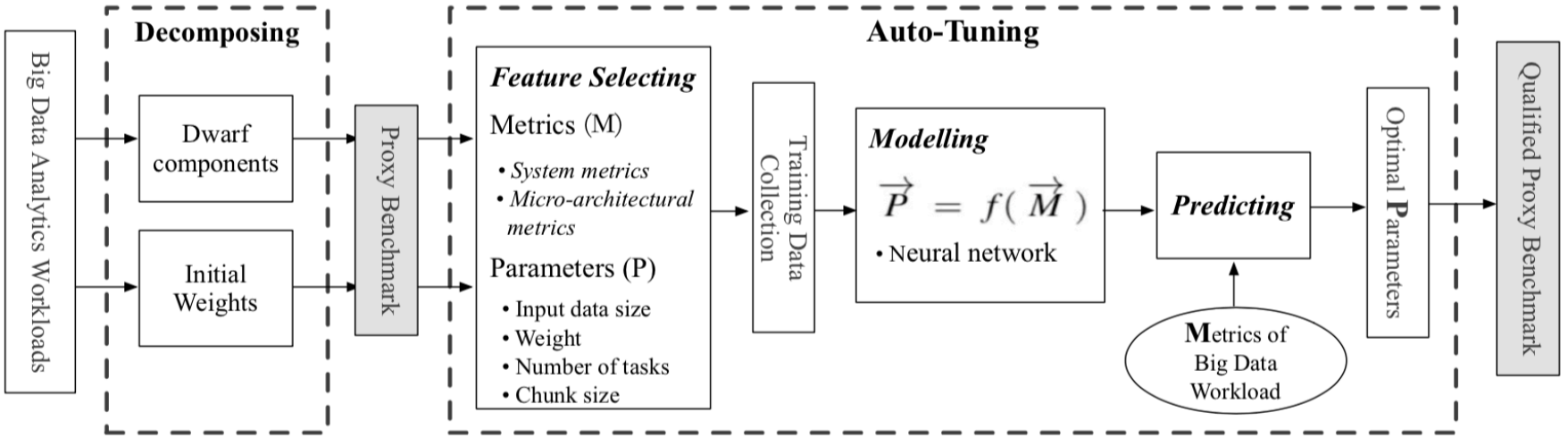
How to Map a Benchmark

- Hadoop TeraSort
  - Sampling
  - Trie tree
- Dwarf TeraSort
  - Dwarf component
    - Operations combination



Dwarf Combination & Tuning

- Modelling and Tuning
  - Metric: $\vec{M} = (\ runtime,\ IPC,\ MIPS,\ L1D\ hitR,\ L2\ hitR,\ .....)$
  - Parameter: $\vec{P} = (\ dataSize,\ chunkSize,\ numTasks,\ weight\ )$



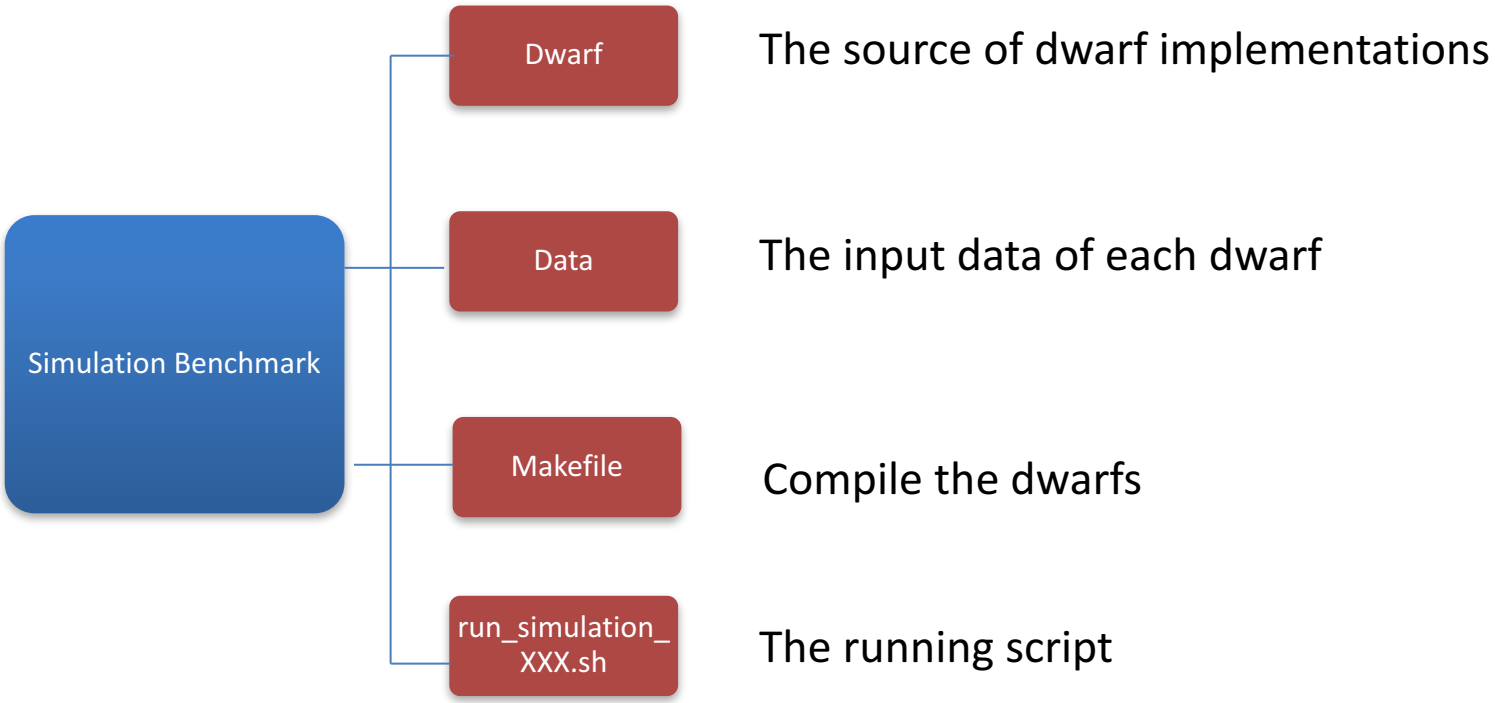
Simulation Benchmarks

- Four representative big data analytics workloads

Big Data Benchmark	Workload Patterns	Data Set	Involved Dwarfs	Involved Dwarf Components
Hadoop TeraSort	I/O Intensive	Text	Sort computations Sampling computations Graph computations	Quick sort; Merge sort Random sampling; Interval sampling Graph construction; Graph traversal
Hadoop Kmeans	CPU Intensive	Vectors	Matrix computations Sort computations Basic Statistic	Vector euclidean distance; Cosine distance Quick sort; Merge sort Cluster count; Average computation
Hadoop PageRank	Hybrid	Graph	Matrix computations Sort computations Basic Statistic	Matrix construction; Matrix multiplication Quick sort; Min/max calculation Out degree and in degree count of nodes
Hadoop SIFT	CPU Intensive Memory Intensive	Image	Matrix computations Sort computations Sampling computations Transform computations Basic Statistic	Matrix construction; Matrix multiplication Quick sort; Min/max calculation Interval sampling FFT/IFFT Transformation Count Statistics

Directory Structure

- Simulation Benchmarks



How to Use

- General steps:
  - Compile dwarf implementations
  - Using Makefile with “make”
  - Run the simulation benchmarks
  - ./run_simulation_XXX.sh

Simulation Benchmarks on GEM5

- Full Systems Mode
  - Mount the image
    - mount -o,loop,offset=32256 /disk_path/disks/ aarch32-ubunt-natty-headless.img /mount_path
  - Copy the dwarf codes to the image
  - Start the gem5
    - ./build/ARM/gem5.opt ./configs/example/fs.py --disk-image=/disk_path/disks/aarch32-ubuntu-natty-headless.img
    - ./m5term localhost 3456

Simulation Benchmarks on GEM5

- Syscall Emulation Mode
  - Install cross-compile tools
  - Run the command
    - ./build/ARM/gem5.opt configs/example/se.py -c sort/src/omp/sort/./multi_thread --options="../dataset/test.data.01 ./output/test.data.01"

Overview

- **Simulation of Big Data and AI Workloads**
  - **Challenges & Motivation**
- **Dwarf-based Simulation Benchmarks**
- **Evaluation on X86 Processor**
- **Case Study on ARM Processor**

Evaluation on X86 Processor

- Configurations

Hardware Configurations			
CPU Type		Intel CPU Core	
Intel ®Xeon E5645		6 cores@2.40G	
L1 DCache	L1 ICache	L2 Cache	L3 Cache
6 × 32 KB	6 × 32 KB	6 × 256 KB	12MB
Memory	Ethernet	Hyper-Threading	
32GB,DDR3	1Gb	Disabled	
Software Configurations			
Operating System	Linux Kernel	JDK Version	Hadoop Version
CentOS 6.4	3.11.10	1.7.0	2.7.1

Experiment Setup

- One-master-four-slave cluster
- Hadoop Workloads
  - Fully-distributed workloads from BigDataBench
    - TeraSort: 100GB records
    - Kmeans: 100GB dense vector data
    - PageRank: 2^26-vertex graph data
    - SIFT: one hundred thousand images from ImageNet

Metrics

- Accuracy
  - Micro-architectural metrics
    - Processor performance
    - Cache behavior
    - Instruction mix
    - Branch prediction
    - Memory bandwidth

Category	Metric Name	Description
Processor Performance	IPC	Instructions per cycle
	MIPS	Million instructions per second
Cache Behavior	L1I Hit Ratio	L1 instruction cache hit ratio
	L1D Hit Ratio	L1 data cache hit ratio
	L2 Hit Ratio	L2 cache hit ratio
	L3 Hit Ratio	L3 cache hit ratio
Instruction Mix	Instruction ratios	Instruction ratios of floating-point, load, store, branch and integer instructions
Branch Prediction	Branch Miss	Branch miss prediction ratio
Memory Bandwidth	Memory Read Bandwidth	Data load rate from memory
	Memory Write Bandwidth	Data store rate into memory
	Memory Bandwidth	Data load/store rate from/into memory

Runtime Speedup on Xeon E5645

- Speedup comparing to Hadoop benchmarks
  - 136X for TeraSort
  - 355X for Kmeans
  - 160X for PageRank
  - 90X for SIFT

Workloads	Execution Time (Second)	
	Hadoop version	Proxy version
TeraSort	1500	11.02
Kmeans	5971	8.03
PageRank	1444	9.03
SIFT	721	8.02

Performance Accuracy

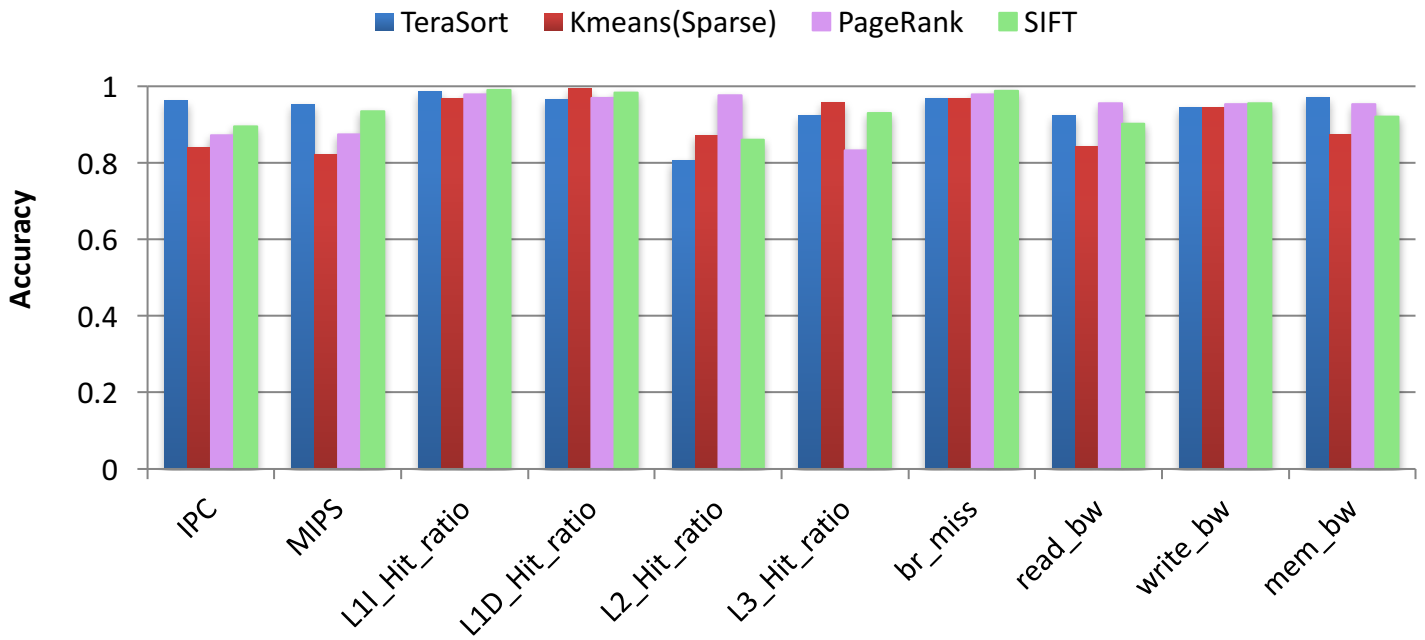
- The accuracy of all selected metrics

$$Accuracy(Val_H, Val_D) = 1 - \left| \frac{Val_D - Val_H}{Val_H} \right|$$

- Val_H – average value of Hadoop benchmark on all slaves
- Val_D – average value of corresponding dwarf benchmark

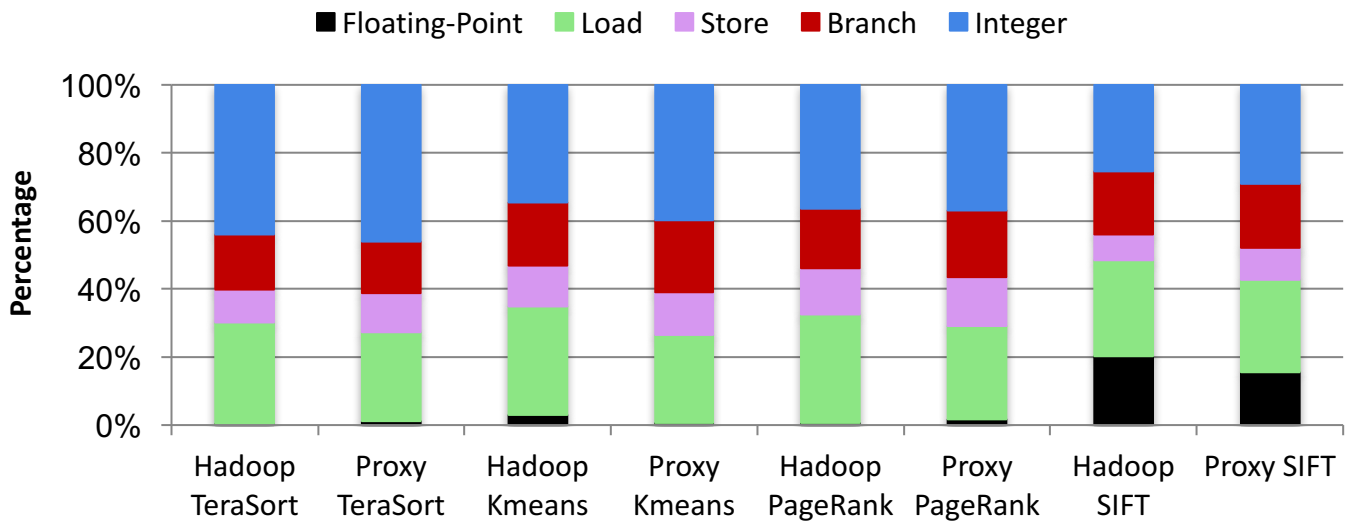
Micro-architectural Data Accuracy on Xeon E5645

- Average data accuracy
  - 94% for TeraSort, 91% for Kmeans, 93% for PageRank, 94% for SIFT



Instruction Mix Breakdown

- Preserve the instruction mix characteristics
  - less floating-point instructions, more integer operations
  - higher demand for data movement than instruction executions
- SIFT workload: preserve the floating-point instruction characteristic

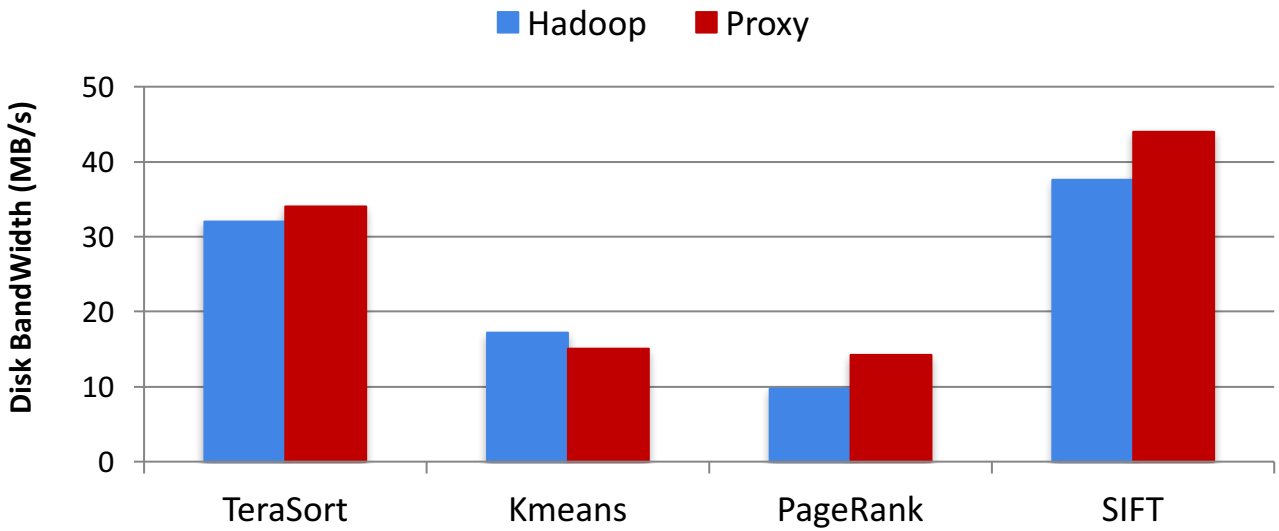


Disk I/O Behavior

- Disk I/O Bandwidth

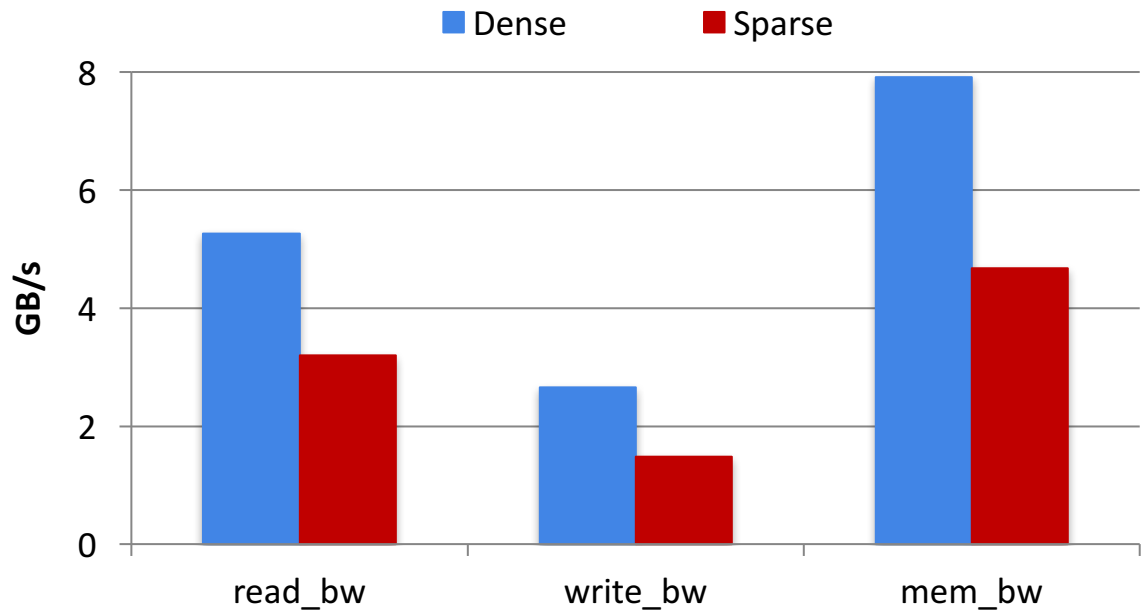
$$BW_{DiskI/O} = \frac{(Sector_{Read+Write}) * Size_{Sector}}{RunTime}$$

- Preserve the disk I/O characteristics



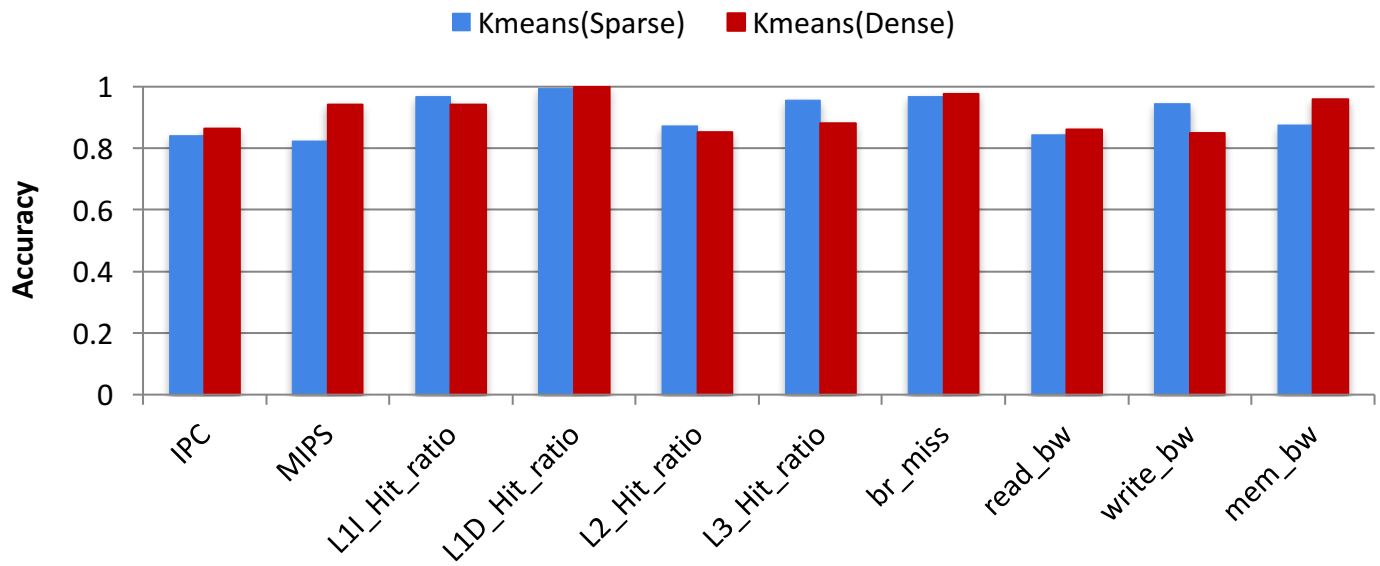
Impact of Input Data

- Data impact on memory bandwidth for Hadoop Kmeans
  - Dense vector: all elements are none-zero
  - Sparse vector: 90% elements are zero



Data Accuracy using Different Data

- Hadoop Kmeans and Dwarf Kmeans
  - Only change the input data sparsity
  - Average data accuracy above 92%



Overview

- **Simulation of Big Data and AI Workloads**
  - **Challenges & Motivation**
- **Dwarf-based Simulation Benchmarks**
- **Evaluation on X86 Processor**
- **Case Study on ARM Processor**

Case Study on ARMv8 Processor

- Runtime speedup comparing to Hadoop benchmarks
- Performance accuracy
- Multi-core scalability
- Runtime speedup across different processors
  - ARMv8 and Xeon E5-2690 V3 (Haswell)

Configurations

- ARMv8 and Haswell processors

Hardware Configurations		
Model	ARMv8	Xeon E5-2690 V3
Number of Processors	1	1
Number of Cores	32	12
Frequency	2.1GHz	2.6GHz
L1 Cache(I/D)	48KB/32KB	32KB/32KB
L2 Cache	8 x 1024KB	12 x 256KB
L3 Cache	32MB	30MB
Architecture	ARM	X86_64
Memory	64GB, DDR4	64GB, DDR4
Ethernet	1Gb	1Gb
Hyper-Threading	None	Enabled
Software Configurations		
Operating System	EulerOS V2.0	Red-hat Enterprise Linux Server release 7.0
Linux Kernel	4.1.23-vhulk3.6.3.aarch64	3.10.0-123.e17.x86-64
GCC Version	4.9.3	4.8.2
JDK Version	jdk1.8.0_101	jdk1.7.0_79
Hadoop Version	2.5.2	2.5.2

Experiment Setup

- One-master-one-slave cluster
- Hadoop Workloads
  - Fully-distributed workloads from BigDataBench
    - TeraSort: 50GB records
    - Kmeans: 50GB dense vector data
    - PageRank: 2^24-vertex graph data

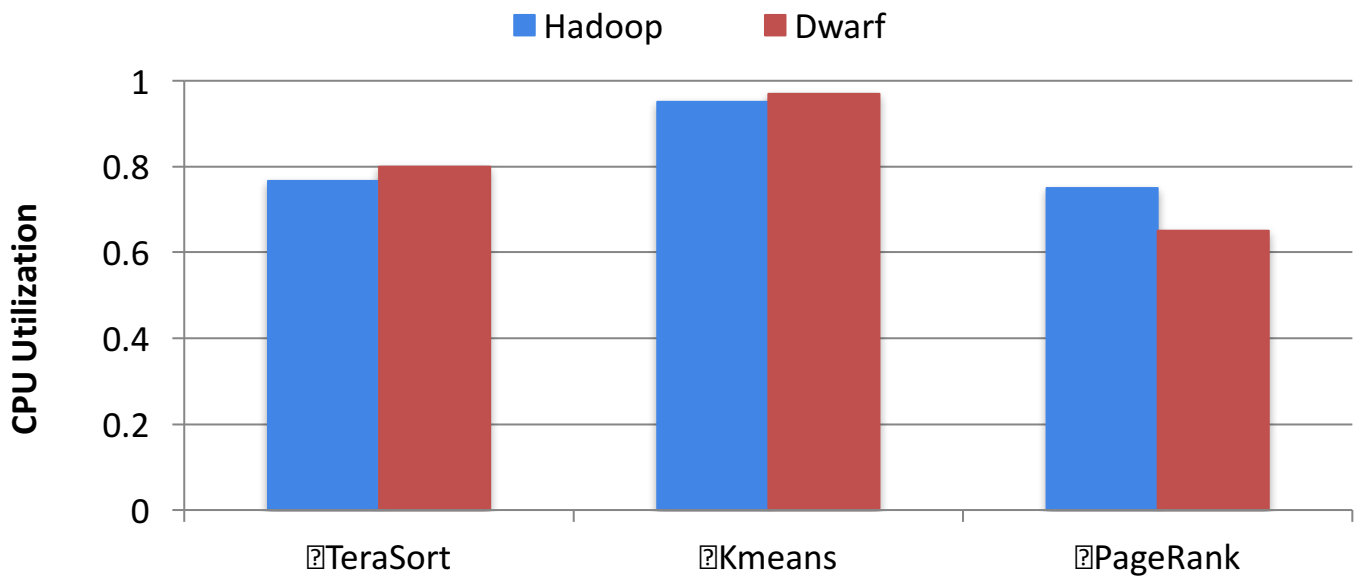
Runtime Speedup on ARMv8

- Speedup comparing to Hadoop benchmarks
  - 336X for TeraSort
  - 386X for Kmeans
  - 690X for PageRank

Workloads	Execution Time (Second)	
	Hadoop version	Dwarf version
TeraSort	1378	4.10
Kmeans	3347	8.68
PageRank	4291	6.22

CPU Utilization on ARMv8

- The similar utilization of CPU resources

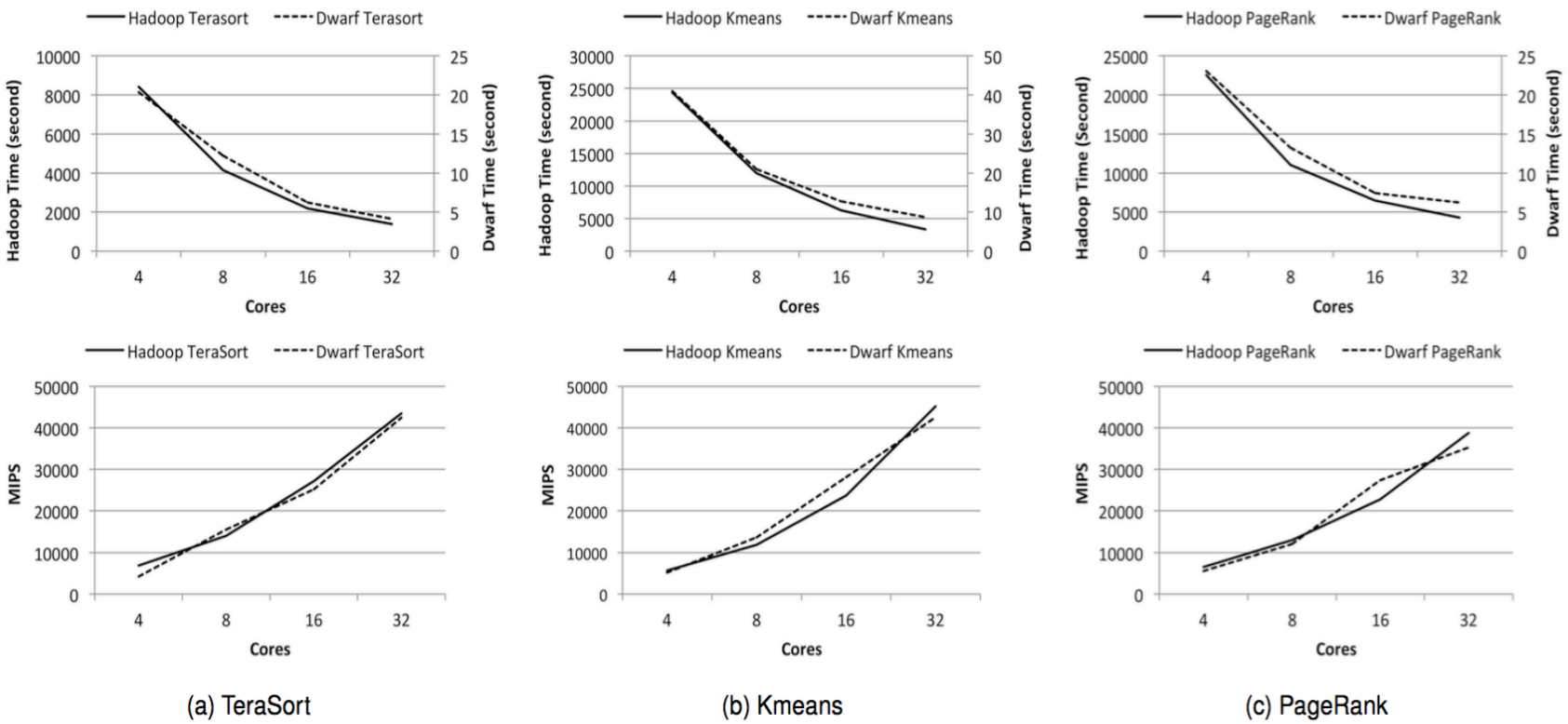


Data Accuracy on ARMv8

- Average data accuracy
  - 93% for TeraSort, 95% for Kmeans, 92% for PageRank

Multi-core Scalability on ARMv8

- Similar multi-core scalability trends



Runtime Speedup Equation

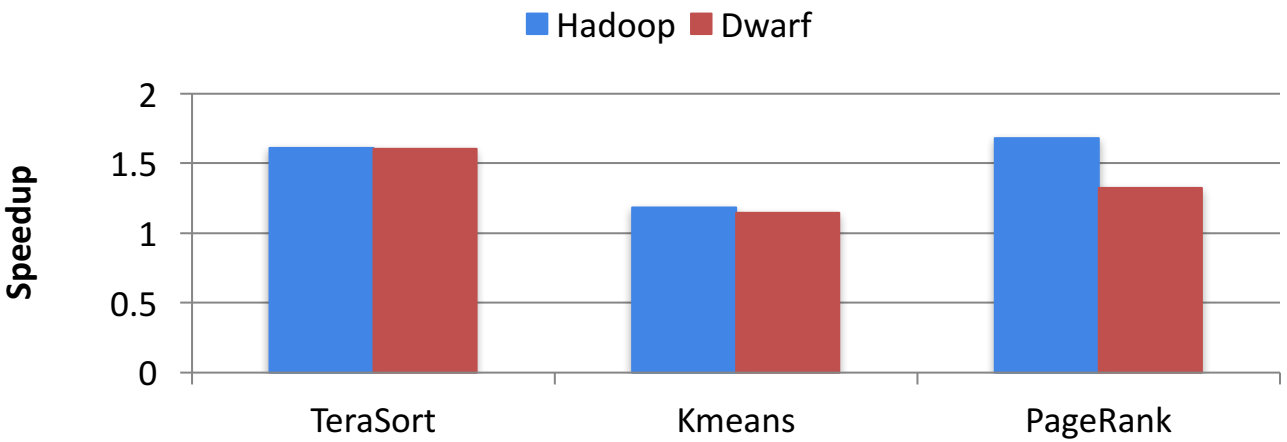
- The runtime speedup across different processors

$$Speedup(Time_{Haswell}, Time_{ARMv8}) = \frac{Time_{ARMv8}}{Time_{Haswell}}$$

- $Time_{Haswell}$ – runtime on Haswell processor
- $Time_{ARMv8}$ – runtime on ARMv8 processor

Runtime Speedup across Different Processors

- ARMv8 V.S. Xeon E5-2690 V3 (Haswell)
  - Consistent speedup trends



Future Work

- Expanding the dwarf set
  - We acknowledge our current eight dwarfs may be not enough for other applications we fail to investigate
- More implementations for the dwarf library
- Expanding dwarf-based simulation benchmarks

Conclusion

- Big data dwarfs & Dwarf library
  - A comprehensive methodology
  - Eight kinds of frequently-appearing operations
- Dwarf-based simulation benchmarks
  - 100s runtime speedup with respect to the benchmarks from BigDataBench
  - average micro-architectural data accuracy is above 90% on both X86_64 and ARM

Thank You !