Big Data Simulator version

User Manual

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

Content

1 Motivation

For system and architecture researches, i. e., architecture, OS, networking and storage, the number of benchmarks will be multiplied by different implementations, and hence become massive. For example, BigDataBench 3.0 provides about 77 workloads (with different implementations). Given the fact that it is expensive to run all the benchmarks, especially for architectural researches that usually evaluate new design using simulators, downsizing the full range of the BigDataBench benchmark suite to a subset of necessary (non-substitutable) workloads is essential to guarantee cost-effective benchmarking and simulations.

2 Methodology

1) Identify a comprehensive set of workload characteristics from a specific perspective, which affect the performance of workloads.

2) Eliminate the correlation data in those metrics and map the high dimension metrics to a low dimension.

3) Use the clustering method to classify the original workloads into several categories and choose representative workloads from each category.

The methodology details of subsetting (downsizing) workloads are summarized in our **IISWC 2014** paper **[PDF]**.

3 Architecture subset

The BigDataBench architecture subset is for the architecture communities. Currently, it downsizes the full BigDataBench 3.0--- 77 workloads---to 17 representative workloads. Each workload represents a workload cluster with a different size. Note that BigDataBench architecture subset is all from a computer architecture point of view. Results may differ if subsetting is performed from a different point of view.

3.1 Microarchitectural Metric Selections

We choose a broad set of metrics of different types that cover all major characteristics. We particularly focus on factors that may affect data movement or calculation. For

example, a cache miss may delay data movement, and a branch misprediction flushes the pipeline. We choose the 45 metrics from micro-architecture aspects as follows.

- Instruction Mix
- Cache Behavior
- Translation Look-aside Buffer (TLB) Behavior
- Branch Execution
- Pipeline Behavior
- Offcore Requests and Snoop Responses
- Parallelism
- Operation Intensity

3.2 Removing Correlated Data

Given a large number of workloads and metrics, it is difficult to analyze all the metrics to draw meaningful conclusions. Note, however, that some metrics may be correlated. For instance, a long latency cache miss may cause pipeline stalls. Correlated data can skew similarity analysis— many correlated metrics will overemphasize a particular property's importance. So we eliminate correlated data before analysis. Principle Component Analysis (PCA) is a common method for removing such correlated data. We first normalize metric values to a Gaussian distribution. Then we use Kaiser's Criterion to choose the number of principle components (PCs). Finally we choose nine PCs, which retain 89.3% variance.

3.3 Workloads Similarity

In order to show the similarity among each workload, we also employ hierarchical clustering, which is one common way to perform such analysis, for it can quantitatively show the similarity among workloads via a dendrogram. Figure I show the dendrogram, which quantitatively measures the similarity of the full BigDataBench workloads (version 3.0). The dendrogram illustrates how each cluster is composed by drawing a U-shaped link between a non-singleton cluster and its children. The length of the top of the U-link is the distance between its children. The shorter the distance, the more similar between the children. We use Euclidean distance. Further, we use the single linkage distance to create the dendrogram.

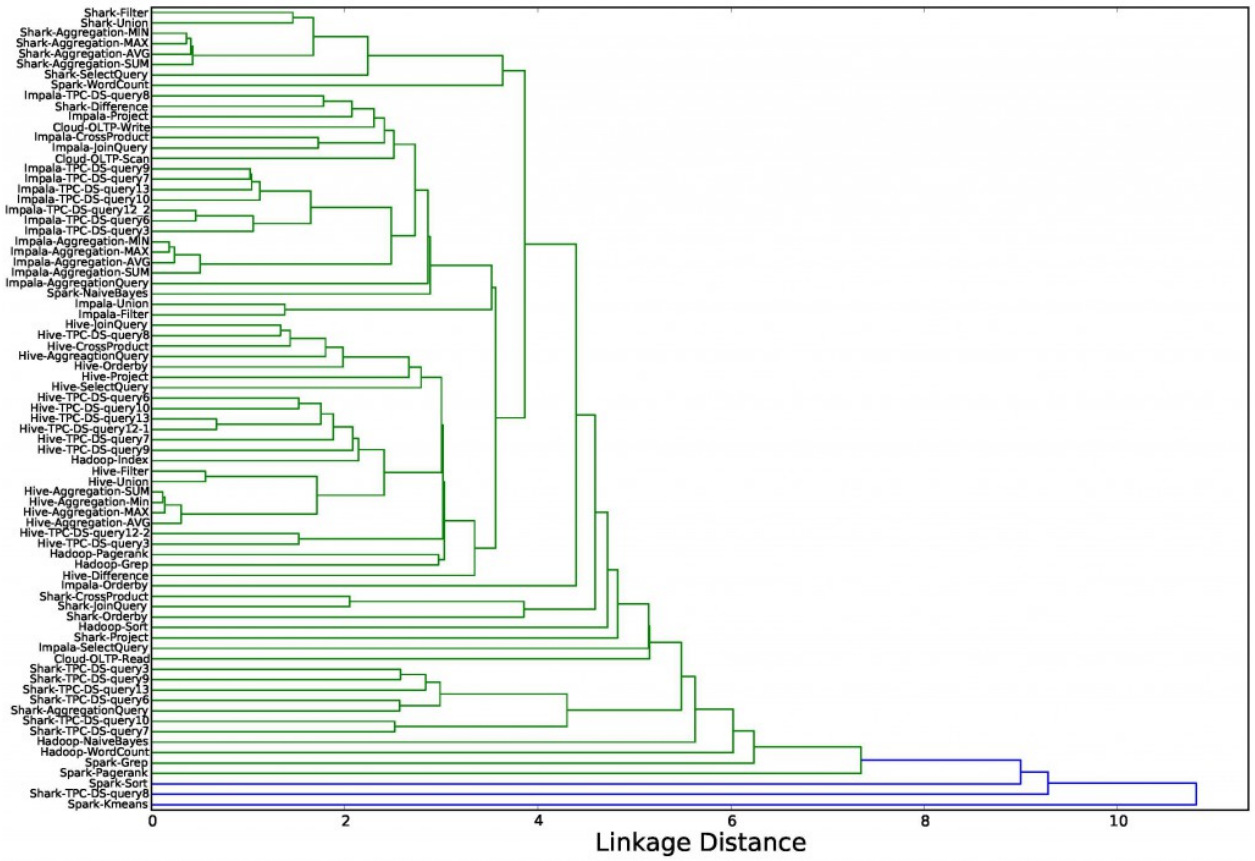

Figure 1 Similarity of the full BigDataBench 3.0 workloads.

4.Clustering

We use K-Means clustering on the nine principle components obtained from the PCA algorithm to group workloads into similarly behaving application clusters and then we choose a representative workload from each cluster. In order to cluster all the workloads into reasonable classes, we use the Bayesian Information Criterion (BIC) to choose the proper K value. The BIC is a measure of the “goodness of fit” of a clustering for a data set. The larger the BIC scores, the higher the probability that the clustering is a good fit to the data. Here we determine the K value that yields the highest BIC score.

We ultimately cluster the 77 workloads into 17 groups, which are listed in Table I.

Table I Clustering results

Cluster	Workloads
1	Cloud-OLTP-Read, Impala-JoinQuery, Shark-Difference, Hadoop-Sort, Cloud-OLTP-San, Ipala-TPC-DS-query8, Impala-Crossproduct, Impala-Project, Impala-AggregationQuery,

	Cloud-OLTP-Write
2	Hive-TPC-DS-query10, Hive-TPC-DS-query12-1, Hive-Difference, Hadoop-Index, Hive-TPC-DS-query6, Hive-TPC-DS-query7, Hive-TPC-DS-query9, Hive-TPC-DS-query13, Hive-TPC-DS-query12-2
3	Hive-Orderby, Hive-SelectQuery, Hive-TPC-DS-query8, Impala-SelectQuery, Hive-Crossproduct, Hive-Project, Hive-JoinQuery, Hive-AggregationQuery
4	Impala-TPC-DS-query6, Impala-TPC-DS-query12_2, Hive-TPC-DS-query3,Spark-NaiveBayes, Impala-TPC-DS-query7, Impala-TPC-DS-query13, Impala-TPC-DS-query9, Impala-TPC-DS-query10, Impala-TPC-DS-query3
5	Shark-Union, Spark-WordCount, Shark-Aggregation-AVG, Shark-Filter, Shark-Aggregation-MAX, Shark-SelectQuery, Shark-Aggregation-MIN, Shark-Aggregation-SUM
6	Impala-Filter, Impala-Aggregation-AVG, Impala-Union, Impala-Orderby, Impala-Aggregation-MAX, Impala-Aggregation-MIN, Impala-Aggregation-SUM,
7	Hive-Aggregation-AVG, Hive-Aggregation-MIM, Hive-AggregationSUM, Hadoop-Grep, Hive-Union, Hive-AggregationMAX, Hive-Filter, Hadoop-Pagerank
8	Shark-TPC-DS-query9, Shark-TPC-DS-query7, Shark-TPC-DS-query10, Shark-TPC-DS-query3
9	Shark-AggregationQuery, Shark-TPC-DS-query6, Shark-Project Shark-TPC-DS-query13
10	Shark-JoinQuery, Shark-Orderby, Shark-Crossproduct
11	Spark-Kmeans
12	Shark-TPCDS-query8
13	Spark-Pagerank
14	Spark-Grep
15	Hadoop-WordCount
16	Hadoop-NaiveBayes
17	Spark-Sort

5. Representative Workloads Selection

There are two methods to choose the representative workload from each cluster. The first is to choose the workload that is as close as possible to the center of the cluster it belongs to. The other is to select an extreme workload situated at the “boundary” of each cluster.

Combined with hierarchical clustering result, we select the workload situated at the “boundary” of each cluster as the representative workload. The rationale behind the

approach would be that the behavior of the workloads in the middle of a cluster can be extracted from the behavior of the boundary, for example through interpolation. So the representative workloads are listed in Table II. And the number of workloads that each selected workload represents is given in the third column.

Table II Treat the marginal ones as representative workloads

	Workload name	Number of workloads in its cluster
1	Cloud-OLTP-Read	10
2	Hive-Difference	9
3	Impala-SelectQuery	9
4	Hive-TPC-DS-query3	9
5	Spark-WordCount	8
6	Impala-Orderby	7
7	Hadoop-Grep	7
8	Shark-TPC-DS-query10	5
9	Shark-Project	3
10	Shark-Orderby	3
11	Spark-Kmeans	1
12	Shark-TPC-DS-query8	1
13	Spark-Pagerank	1
14	Spark-Grep	1
15	Hadoop-WordCount	1
16	Hadoop-NaiveBayes	1
17	Spark-Sort	1

6 Simulator Images

To facilitate micro-architectural simulation, we deploy the 17 representative applications listed on Simics, a full-system simulator. We then provide the simulator images for researchers to download.

The workloads in BigDataBench are all distributed workloads using big data software stacks such as Hadoop, Spark and etc. Those workloads are running on a cluster, which

consists of a master and several slaves. The master node distributes tasks and slaves execute the tasks. We simulate a two nodes cluster (one master and one slave), and we provide both images. Users should boot up both the images and submit the job on master node. For the slave node is the one that process the whole job, if users want to get some performance data, slave node should be focused on.

6.1 Deployment

Simics installation (recommended to install in the /opt/virtutech directory)

1 Download the appropriate Simics installation package from the download site, such as simics-pkg-00-3.0.0-linux.tar

2 Extract the installation package, the command is as follows:

tar xf simics-pkg-00-3.0.0-linux.tar

Will add a temporary installation directory, called simics-3.0-install

3 Enter the temporary installation directory, run the install script, the command is as follows

cd simics-3.0-install

sh install-simics.sh

4 The Simics requires a decryption key, which has been unpacked before. decode key has been cached in $HOME/.simics-tfkeys.

$ HOME / .simics-tfkeys

5 When the installation script is finished, Simics has been installed in the / opt / virtutech / simics- <version> /, if the previous step to specify the installation path, this path will be different

6 When the Simics is successfully installed, temporary installation directory can be deleted

6.2 Workloads running

Hadoop-based workloads

Experimental environment

Cluster: one master one slaver

Software : We have already provide the following software in our images.

Hadoop version：Hadoop-1.0.2

ZooKeeper version：ZooKeeper-3.4.5

Hbase version：HBase-0.94.5

Java version：Java-1.7.0

Users can use the following commands to drive the Simics images.

Workload	Master	Slaver
Wordcount	cd /master	cd /slaver
	./simics -c Hadoopwordcount_L	./simics -c Hadoopwordcount_L
	bin/hadoop jar ${HADOOP_HOME}/hadoop-examples-*.jar wordcount /in /out/wordcount	
Grep	cd /master	cd /slaver
	./simics -c Hadoopgrep_L	./simics -c Hadoopgrep_LL
	bin/hadoop jar ${HADOOP_HOME}/hadoop-examples-*.jar grep /in /out/grep a*xyz	
NaiveBayes	cd /master	cd /slaver
	./simics -c HadoopBayes_L	./simics -c HadoopBayes_LL
	bin/mahout testclassifier -m /model –d /testdata	
Cloud OLTP-Read	cd /master	cd /slaver
	./simics -c YCSBRead_L	./simics -c YCSBRead_LL
	./bin/ycsb run hbase -P workloads/workloadc -p operationcount=1000 -p hosts=10.10.0.13 -p columnfamily=f1 -threads 2 -s>hbase_tranunlimited C1G.dat	

Hive-base workloads

Experimental environment:

Cluster: one master one slaver

Software : We have already provide the following software in our images.

Hadoop version：Hadoop-1.0.2

Hive version：Hive-0.9.0

Java version：Java-1.7.0

Workload	Master	Slaver
Hive-Differ	cd /master	cd /slaver

	./simics HiveDiffer_L	./simics -c HiveDiffer_LL
	./BigOP-e-commerce-difference.sh	
Hive-TPC-DS-query3	cd /master	cd /slaver
	./simics -c Hadoopgrep_L	./simics -c Hadoopgrep_LL
	./query3.sh	

Spark-based workloads

Experimental environment

Cluster: one master one slaver

Software : We have already provide the following software in our images.

Hadoop version：Hadoop-1.0.2

Spark version：Spark-0.8.0

Scala version：Scala-2.9.3

Java version：Java-1.7.0

Workload	Master	Slaver
Spark-WordCount	cd /master	cd /slaver
	./simics -c SparkWordcount_L	./simics -c SparkWordcount_LL
	./run-bigdatabench cn.ac.ict.bigdatabench.WordCount spark://10.10.0.13:7077 /in /tmp/wordcount	
Spark-Grep	cd /master	cd /slaver
	./simics -c Sparkgrep_L	./simics -c Sparkgrep_LL
	./run-bigdatabench cn.ac.ict.bigdatabench.Grep spark://10.10.0.13:7077 /in lda_wiki1w /tmp/grep	
Spark-Sort	cd /master	cd /slaver
	./simics -c SparkSort_L	./simics -c SparkSort_LL
	./run-bigdatabench cn.ac.ict.bigdatabench.Sort spark://10.10.0.13:7077 /in /tmp/sort	
Spark-Pagerank	cd /master	cd /slaver

	./simics -c SparkPagerank_L	./simics -c SparkPagerank_LL
	./run-bigdatabench cn.ac.ict.bigdatabench.PageRank spark://10.10.0.13:7077 /Google_genGraph_5.txt 5 /tmp/PageRank	
Spark-Kmeans	cd /master	cd /slaver
	./simics -c SparkKmeans_L	./simics -c SparkKmeans_LL
	./run-bigdatabench org.apache.spark.mllib.clustering.KMeans spark://10.10.0.13:7077 /data 8 4	

Shark-based workloads

Experimental environment

Cluster : one master one slaver

Software : We have already provide the following software in our images.

Hadoop version：Hadoop-1.0.2

Spark version：Spark-0.8.0

Scala version：Scala-2.9.3

Shark version：Shark-0.8.0

Hive version：hive-0.9.0-shark-0.8.0-bin

Java version：Java-1.7.0

Workload	Master	Slaver
Shark-Project Shark-Orderby	cd /master	cd /slaver
	./simics -c Sharkprojectorder_L	./simics -c Sharkprojectorder_LL
	./runMicroBenchmark.sh	
Shark-TPC-DS-query8	cd /master	cd /slaver
	./simics -c Sharkproquery8_L	./simics -c Sharkquery8_LL
	shark -f query8.sql	
Shark-TPC-DS-query10	cd /master	cd /slaver
	./simics -c Sharkproquery10_L	./simics -c Sharkquery10_LL
	shark -f query10.sql	