A Characterization of Big Data Benchmarks

Wen Xiong, Zhibin Yu*, Zhendong Bei, Juanjuan Zhao, Fan Zhang, Yubin Zou, Xue Bai, Ye Li, Chengzhong Xu
Center for Cloud Computing, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences,
518055 ShenZhen, China
*corresponding author
{wen.xiong, zb.yu, zd.bei, jj.zhao, zhangfan, yb.zou, bai.xue, li.ye, cz.xu}@siat.ac.cn

***Abstract*—recently, big data has been evolved into a buzzword from academia to industry all over the world. Benchmarks are important tools for evaluating an IT system. However, benchmarking big data systems is much more challenging than ever before. First, big data systems are still in their infant stage and consequently they are not well understood. Second, big data systems are more complicated compared to previous systems such as a single node computing platform. While some researchers started to design benchmarks for big data systems, they do not consider the redundancy between their benchmarks. Moreover, they use artificial input data sets rather than real world data for their benchmarks. It is therefore unclear whether these benchmarks can be used to precisely evaluate the performance of big data systems.**

In this paper, we first analyze the redundancy among benchmarks from ICTBench, HiBench and typical workloads from real world applications: spatio-temporal data analysis for Shenzhen transportation system. Subsequently, we present an initial idea of a big data benchmark suite for spatio-temporal data. There are three findings in this work: (1) redundancy exists in these pioneering benchmark suites and some of them can be removed safely. (2) The workload behavior of trajectory data analysis applications is dramatically affected by their input data sets. (3) The benchmarks created for academic research cannot represent the cases of real world applications.

Keywords: workloads; similarity; micro-architecture metrics; mapreduce; trajectory data;

I. INTRODUCTION

In recent years, big data has been evolved into a buzzword from academia to industry all over the world. Companies such as Google, Facebook, and Baidu already built many big data systems. Governments including USA, China, and Japan etc. allocate a lot of funds to support the big data research. While big data is super hot, it also comes with challenges. One of which is how to evaluate the performance of big data systems. Benchmarks are the most important part of evaluating the performance of an IT system. Hence, the above challenge is converted into how to design a benchmark suite for big data systems.

As a benchmark suite, two requirements must be satisfied. (1) The number of benchmarks in the suite should be as small as possible. (2) The benchmarks in the suite should be as diverse as possible. The first requirement aims to reduce the time needed to evaluate an IT system. The second one attempts to represent a wide range of applications. Big data as an emerging area, satisfying the two requirements is extremely challenging. Moreover, the two requirements of a benchmark suite are contradictory, further complicating the big data benchmarking problem.

Among big data platforms, the MapReduce/Hadoop is one of the most widely used frameworks. Therefore, we focus on benchmark suites for Hadoop systems in this paper. Since benchmarking big data systems is utter important, a few of studies have been done to design big data benchmark suites [1][2][3][4][5]Various communities and organizations related big data have released benchmark suites such as Cloudsuite [1], GridMix [2], Hive performance benchmark [5], ICTBench [3], and HiBench [4].

However existing Hadoop benchmark suites can't properly evaluate Hadoop framework because of their limitations of representativeness and diversity. Generally, researchers and end users partially use these benchmark suites due to simulation constraints, system calls or hardware circumstances issues. But a randomly selected subset would not be proper for evaluating their systems [9]. For a specific application domain, it would be beneficial to cost saving by finding out a valid minimal subset without missing diversity. Moreover, this subset includes the same characteristics as these benchmark suites do, and it obviously decreases simulation or evaluation time.

Generally, in a benchmark suite design, it needs to meet a number of considerations. These considerations are as follows: (1) a benchmark suite should have workloads that are representative of a wide range of application domains; (2) workloads in a benchmark suite should have diversity of data characteristics; and (3) a benchmark suite should not have redundant workloads in itself. However, existing hadoop benchmark suites providers present suites without analyzing redundancy between their workloads and these already popular typical workloads. It is necessary to conduct similarity analysis in these typical workloads due to the aforementioned reasons.

MapReduce based benchmarks require big data sets as the inputs to drive their workloads [7]. H Xi et al. [15] found that frequently used distributions cannot capture the key characteristics of real world data. Chen et al. [17] found that application traces from larger-scale MapReduce Clusters deployed in Facebook and Cloudera did not fit well-know statistical distributions. In this case, only real data can reflect the real system behaviors and workload characteristics, and hence the real world data is preferred in big data benchmarks. However, to obtain real world data is a big challenge because of two primary reasons, these reasons are as follows: (1) the owner of real world data would not like to share their big data for business confidentiality and user privacies; and (2) even though the real world data is available on the internet, it is unacceptable for researchers to download terabyte scale data under the condition of current internet traffic. Therefore, we provide real world data with two workloads under the permission of our collaborator. These two workloads are *sztod* and *hotregion*, both of which are typical programs in our internal project related to trajectory data processing in real world.

At the beginning of our work, we choose 14 typical workloads form ICTBench and HiBench. Both of which are popular MapReduce related benchmark suites in various communities. Meanwhile, we choose 2 typical workloads from our MapReduce system for processing trajectory data.

HiBench was proposed by Intel Company. It is a new, realistic and comprehensive benchmark suite for Hadoop. And it consists of a set of Hadoop programs including both read-world applications and synthetic mircobenchmarks [4]. ICTBench is a presented by Institute of Computing Technology (ICT), which contains three different benchmark suites, and these benchmark suites are DCBench BigDataBench and CloudRank [3]. DCBench consists of typical data center workloads; these workloads are different from scientific computing applications; they cover applications in important domains such as search engine and electronic commerce; and each workload equal to a single application [6]. BigDataBench was purposed for large-scale systems and architecture researches and for characterizing big data applications; each benchmark in BigDataBench is equal to a single big application [7]. Each benchmark of CloudRank equal to a group of consolidated data center workloads; and it was purposed for capacity planning, system evaluation and researches [8].

It is unclear that there is redundancy between workloads from the abovementioned benchmark suites and we can find out a valid subset of these typical workloads, which minimizes the number of typical workloads but includes the same characteristics as these original benchmark suites do. We address the problem by using micro-architecture level metrics and use *principal component analysis* (PCA) and clustering techniques to analyze similarity in these typical workloads. In particular, the main contributions of this paper are as follows:

- We characterize 16 various typical workloads from HiBench and ICTBench by micro-architecture level metrics.
- We analyze similarity in these various workloads by statistical techniques such as PCA and clustering, the results show that a few workloads have inherent properties.
- We release two typical workloads related to trajectory data process in real-world application domain. These two workloads have dramatically different behavior compared to HiBench and ICTBench.

The rest of the paper is organized as follows. Section 2 introduces the related work. Section 3 depicts the Evaluation methodology. Section 4 characterizes various typical workloads. Section 5 describes the similarity analysis of our experimented workloads. Section 6 describes trajectory data processing. Section 7 concludes the paper.

II. RELATED WORK

Huang et al. conduct quantitative characterization of different workloads in HiBench from three aspects including data access, system resource utilizations and HDFS bandwidth [4]. Dai et al. analyze a series typical MapReduce based workloads by HiTune, which is a distributed profile tool based on dataflow model for Hadoop framework. They characterize these workloads not only from the timeline based CPU, disk and Network utilization, but also from execution process of workloads for analyzing hotpots [10]. Meanwhile we characterize these typical workloads from micro-architecture level metrics, such as instruction per cycle, cache miss ratio, and number of braches per instruction.

Researchers in ICT (Institute of Computing Technology, Chinese Academy of Sciences) did much work in MapReduce-related benchmarking [11][12][13][14]. They release three different benchmark suites, BigDataBench is a big data benchmark suite [11], DCBench is a data center benchmark suite and CloudRank is a cloud computing benchmark suite [6] [12]. Zheng et al. characterize OS behavior of scale-out data center workloads and investigate how the pipeline front end is affected by OS activities [13]. Zhen et al. focus characterization on top three application domains: search engine, social networks and electronic commerce [14]. On the one hand, their work shows that only one application is not enough to represent various categories of data analysis workloads. On the other hand, their study reveals that a series of workloads share many similarities in terms of micro-architectural characteristics [15]. However, they do not quantitatively analyze similarity in these typical workloads by the micro-architectural metrics they obtained.

Aashish et al. analyze redundancy in the SPEC CPU2006 benchmark suite using micro-architecture metrics. They draw inference on similarity of the benchmarks and arrived at meaningful subsets; and these subsets are representative of a wide range of applications areas without having many benchmarks with similar characteristics. The research result could obviously decrease simulation time for system architecture researches [9]. On the one hand, we apply the same methods to conduct similarity analysis in typical workloads in research domains respectively by micro-architecture level metrics. On the other hand, for a given system, each workload in SPEC CPU2006 was executed as single process in a single physical machine. Meanwhile, in our experiments, each MapReduce based workload was running as a multiple process program in a distributed computing environment, which consists of nine physical machines.

III. EVALUATION METHOD

A. Workloads choosing

To keep the objective workloads set minimal and without missing diversity. The selected workloads must meet two primary requirements.

Firstly, the workloads we choose must be representative. There are representative workloads in MapReduce based workloads in various application domains such as search

TABLE I. WORKLOADS AND BENCHMARK SUITES

Index	*Workload*	*Benchmark suite*
1	Sort	HiBench
2	Wordcount	HiBench
3	Grep	ICTBench
4	Terasort	HiBench
5	Bayes	HiBench
6	K-means	HiBench
7	Nutch indexing	HiBench
8	Pagerank	HiBench
9	Hive-aggregate	HiBench
10	Hive-join	HiBench
11	Hotregion	Internal workload
12	Sztod	Internal workload
13	SVM (support vector machine)	ICTBench
14	Ibcf (Item based collabrative,recom mandation)	ICTBench
15	Fpg (Frequent pattern growth)	ICTBench
16	Hmm (Hidden Markov model)	ICTBench

engine, social network and electronic commerce. These workloads are typical in their own fields; and they are quietly different from each other in the terms of characteristics such as high level data graph and different input/output ratio.

Secondly, the selected workloads should be popular in various communities and be widely validated by academia and industry.

Based on these considerations, we choose fifteen typical workloads from HiBench, ICTBench, and our internal project, which are related to trajectory data processing in real-world data.

Detail introduction of workloads from ICTBench or HiBench can be found in [10] or [12], and the workloads and benchmark suites are listed in Table 1.

Hotregion and *sztod* are typical workloads in our big data system for real world spatio-temporal data processing. And these data are produced from the subway transportation system and taxicab transportation in Shenzhen. In particular, h*otregion* is a workload for analyzing GPS data of taxicab. *Sztod* is a workload for processing transaction data of subway transportation system.

B. Experiment platform and Hadoop cluster setup

In our experiment, we deploy a Hadoop cluster consisting five nodes, which are one master and four slaves. All nodes in the cluster have the same configurations and are connected through 1000MB Ethernet network. Each node has two Intel Xeon E5620 processors, and it is equipped with 8TB disk, and 16GB memory. Our Linux is ubuntu 12.04,

TABLE II. DETAILS OF HARDWARE CONFIGURATIONS

CPU type	Intel Xeon E5620
# cores	8 cores @ 2.4G
# threads	16 threads
# Sockets	2
ITLB	4-way set assocuative, 64 entries
DTLB	4-way set assocuative, 64 entries
L2 TLB	4-way set assocuative, 512 entries
L1 DCache	32KB, 8-way set assocuative, 64byte/line
L1 ICache	32KB, 4-way set assocuative, 64byte/line
L2 Cache	256KB, 8-way set assocuative, 64byte/line
L3 Cache	12.8MB, 16-way set assocuative, 64byte/line

and the kernel version is 3.2.0. The detail configuration parameters of each node are listed in Table 1.

The version of Hadoop and JDK is 1.0.3 and 1.7.0, respectively. The source Benchmark suites are HiBench-2.2 and CloudRank1.0. The profile tools are HiTune-1.0 and oprofile-0.98. Memory size for child process of task tracker is 400MB, and each slave node has 8 map slots and 8 reduce slots.

C. Experiment methodology

In our experiments, we get OS-level performance data and micro-architectural data by using hardware performance counters. We use operf—a profiling tool for Linux 2.90+ based systems by specifying the event number and corresponding mask. In our work, we observe 12 CPU events, and specify single process mode for the operf. L2 cache is a unified cache, but operf can distinguish L2 instruction cache miss event from L2 data cache miss event. It provides non-root user the capability to collect data from any child process of task tracker, which is the daemon of MapReduce framework in slave nodes. In addition, we use HiTune—a distributed profiling tool for Hadoop-framework to monitor the progress of high level dataflow graph of each workload and collecting data of timeline-based utilization of system resources such as disk, CPU, memory and network.

We start collecting performance data after a warm up period for each workload, and profiling tools is working in the whole lifetime of each workload. Based on oprofile toolkit, we implemented a series of scripts, which can automatically execute for collecting data from all slave nodes and processing data. And we conduct experiments for each workload three times and report the mean value.

IV. WORKLOADS CHARACTERIZATION

In this section, we focus our experiments on seven micro-architecture level characteristics, which calculated by fourteen hardware performance counters. These characteristics are *instruction per cycle (IPC)*, L1 instruction cache miss ratio, L2 instruction cache miss ratio, last level

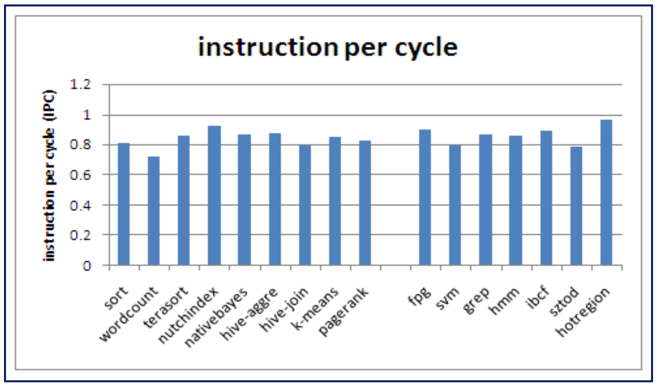

Figure 1. Instruction per cycle of each workloads.

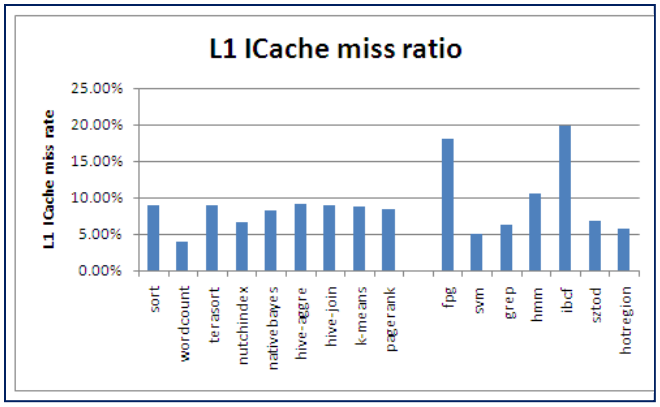

Figure 2. L1 Instruction cache miss ratio of each workloads.

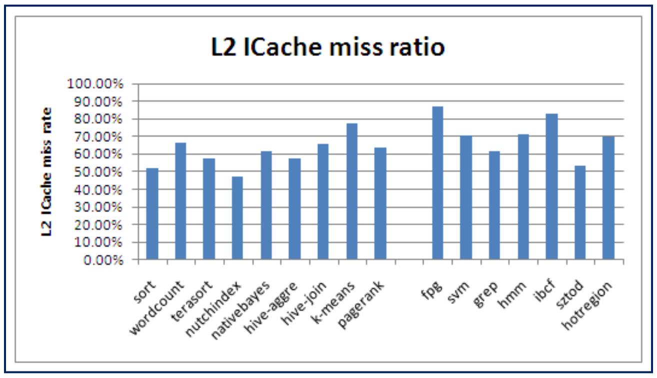

Figure 3. L2 Instruction cache miss ratio cycle of each workloads.

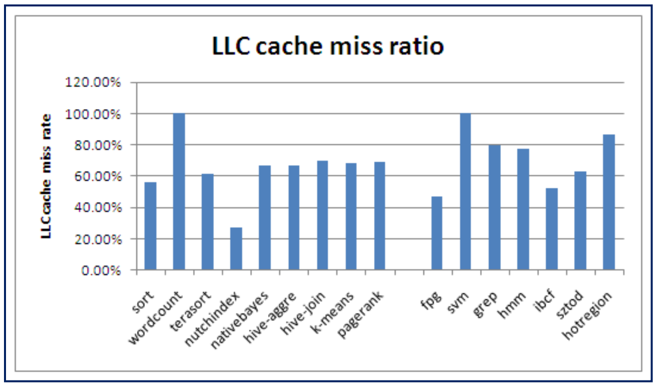

Figure 4. Last level cache miss ratio of each workloads.

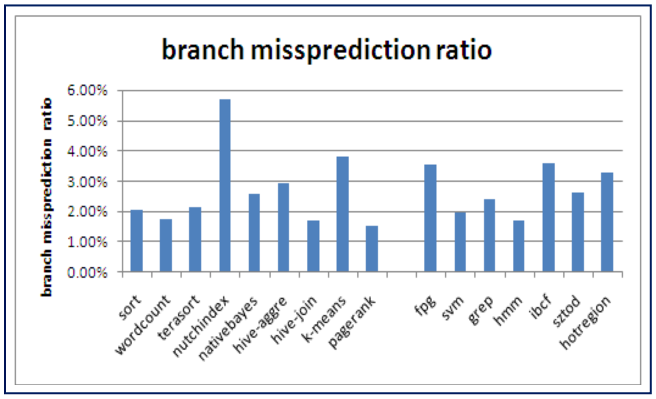

Figure 5. Branch miss prediction ratio of each worklaods.

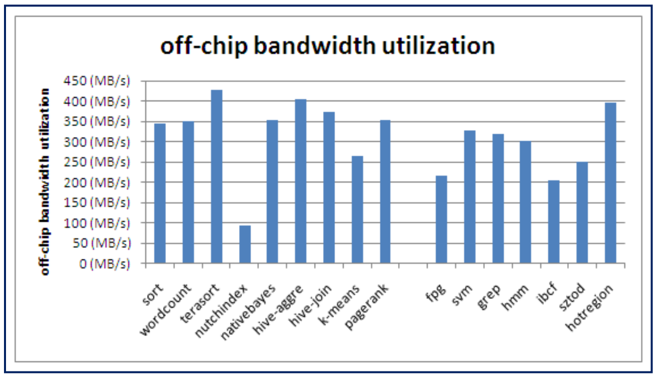

Figure 6. Off-chip bandwidth utilization of each worklaods.

In this section, we focus our experiments on seven micro-architecture level characteristics, which calculated by fourteen hardware performance counters. These characteristics are *instruction per cycle (IPC)*, L1 instruction cache miss ratio, L2 instruction cache miss ratio, last level cache miss ratio, branch miss prediction per instruction, execute time breakdown and off-chip bandwidth. We use these characteristics to conduct similarity analysis of these workloads in next section.

A. *Instruction per cycle (IPC)*

Instruction per cycle (IPC) indicates how many instructions can simultaneously execute in one cycle. Figure 1 shows IPC of these typical workloads, group in the right side includes seven workloads, *fpg, svm*, *grep*, *hmm* and *ibcf* are workloads from cloudrank; *sztod* and *hotregion* are developed by ourselves. Group in the left consists of nine workloads, and all of them are from HiBench. The IPC of these sixteen workloads are range from 0.72 to 0.96, with an average value of 0.85. Wordcount has the lowest IPC value and hotregion has highest value among these workloads. While the workloads are task parallel, the IPC is less than 1. This indicated the parallelism of these workloads is not fully exploited.

B. *Cache miss ratio*

Instruction cache and Translation Look-side Buff (TLB) are two important components, and they provide fetch unit the capability to accelerate instruction fetch.

Figure 2 shows the L1 instruction cache miss ratios of sixteen workloads. The cache miss ratios of these typical workloads are range from 3.9% to 19.8%, with an average value of 8.9%. *Wordcount* has the lowest L1 instruction cache miss ratio and *ibcf* has the highest L1 instruction cache miss ratio.

Figure 3 presents the L2 instruction cache misses of each workload, the cache misses value of these are range from 47.4% to 87.0%. On average, workloads from cloudrank in right side have larger L2 instruction miss rate then workloads from HiBench in the left side. Overall, the L2 cache is ineffective in our experiment platform.

The LLC is the largest on-chip component; its capacity has increased across each processor generation. Our results in Figure 4 show that LLC are one of the key limiters of these MapReduce based applications in our experiment. Both wordcount and svm have almost 100% LLC cache miss ratio.

C. *Branch miss Prediction per instruction*

Branch miss prediction ratio affects the performance directly. Modern out-of-order processors use a functional unit to predict the next branch to avoid pipeline stalls. The pipeline continues whether or not depends on the result of branch predict. If the branch miss prediction occurs, it will cause a dozens of cycles waste due to the pipeline must flush the wrong instructions and fetch the correct ones.

Figure 5 presents the branch miss prediction ratio of each workload, these ratios are range from 1.5% to 5.6%, with an average value of 2.7%. *Pagerank* has the lowest branch miss prediction ratio while *nutch indexing* has the highest branch miss prediction ratio. The results show that the branch predictor of our processor matches these typical MapReduce based applications.

D. *Off-chip bandwidth utilization*

Over the past decade; while the off-chip memory latency has slowly improved, off-chip bandwidth has sharply improved. The speed of the memory bus has increased from dozens of MHz to 1GHz, raising the peak theoretical bandwidth form 544MB/s to 17GB/s. Figure 6 plot the per-core off-chip bandwidth utilization of these typical workloads we have chosen. As the figure 6 depicted, among these workloads we evaluated, terasort is the only one that has the highest utilization ratio with a value of 14%. Overall, in our experiment platform, processors significantly over-provision off-chip bandwidth for these typical workloads.

V. WORKLOADS SIMILARITY

In this section, we choose seven micro-architecture metrics to analyze similarity in typical workloads. These metrics are calculated by their performance counters respectively, detail information of these metrics for each workload on four different machines are as follows: (1) instruction per cycle (IPC); (2) L1 instruction cache miss per instruction; (3) L2 instruction cache miss per instruction; (4) L3 cache miss per instruction; (5) Off-chip bandwidth utilization; (6) number of branches per instruction; and (7) number of mispredicted branches per instruction. We measure fundamental workloads characteristics related to their data locality, branch predictability, and instruction locality by hardware performance counters. Since our experiments are carried out on four different nodes, we take each metric-node pair as a variable, in our experiment; we have 28 variables due to there are four nodes and seven metrics for each workload.

Firstly, we use PCA remove the correlations between this metrics. And then apply hierarchical clustering algorithm to metrics of these workloads. Finally, we analyze the results according to two dendrograms.

A. *Pricinpal component analysis*

Considering twenty-eight variables for each workload from different metric-machine pair, it is impossible to simultaneously look at all experiment data and draw meaningful conclusions from them. Therefore, PCA is used for analyzing the data. Firstly, the data is normalized to a unit normal distribution for isolating the effect of varying ranges of each parameter.

PCA helps to reduce the dimensionality of a data set without too much original information loss. PCA computes a new variable set; these new variables are uncorrelated to each other in this set, new variable in this set are linear combinations of original variables, generally, we take each new variable as a principal component [9].

The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it be orthogonal to (i.e., uncorrelated with) the preceding components. Principal components are guaranteed to be independent if the data set is jointly normally distributed. PCA is sensitive to the relative scaling of the original variables [19].

B. *Clustering and Kmeans*

There are two popular used clustering techniques–hierarchical clustering and K-means clustering. Both of them can be used to group workloads with similar features.

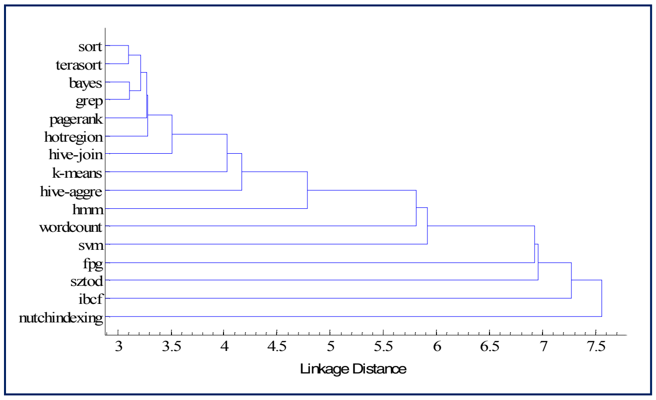

Figure 7. Dendrogram showing similarity between typical MapReduce workloads.

K-means clustering is a method of vector quantization originally from signal processing. K-means clustering aims to partition *n* workloads into *k* clusters in which each workload belongs to the cluster with the nearest mean, where K is a value specified by user. Therefore, we need to cluster workloads for different values of *K* and then select the best fit due to different grouping possibilities [20].

Hierarchical clustering is a "bottom up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy. It is useful in simultaneously looking at multiple clustering possibilities, and use can use a dendrogram for selecting desired number of clusters. It start with a matrix of distance between N cases or workloads, this distance is the Euclidean distance between the workloads metrics [20].

In our experiments, we use hierarchical clustering algorithm to find out a representative subset of these typical workloads aforementioned.

C. *Workloads similarity*

Generally, researchers and customers partially use workloads of a specific benchmark suite due to simulation constraints, system call or hardware circumstance issues.

It would be beneficial to cut down the simulation or evaluation time by finding out a valid minimal subset of a representative, which includes the same characteristics as that benchmark suite dose and without missing diversity.

This section demonstrates the results of analyzing similarity in typical MapReduce based workloads. Figure 7 shows a dendrogram for typical workloads listed in Table I after applying PCA and Hierarchical Clustering on the seven metrics. We measure dissimilarity between workloads by Euclidean distance and create a dendrogram by single-linkage distance. In Figure 7 the horizontal axis shows the linkage distance indicating the similarity and dissimilarity between workloads. Workloads that are outliers have larger linkage distances with the rest of the clusters formed in a hierarchical way. Workloads are positioned close to each other when the distance is smaller, and the ordering on the Y-axis does not have particular significance.

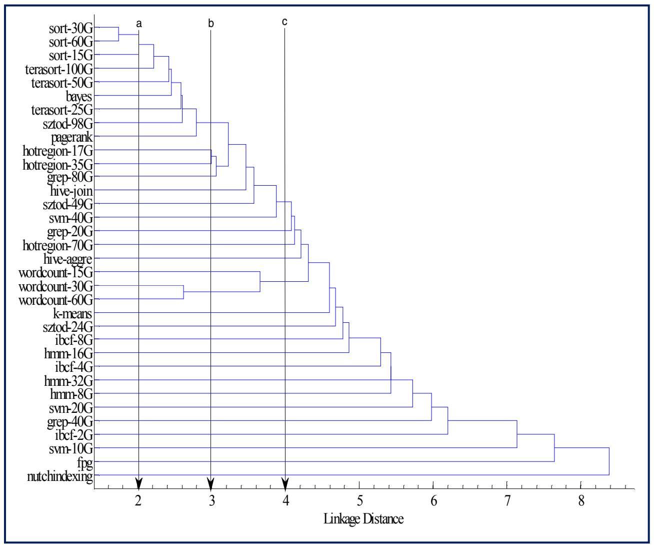

Figure 8. Dendrogram showing similarity between input size for each workload.

We can clearly see from Figure 7, a) *sort* and *terasort* as a workload pair has the smallest distance to each other in these workloads, as well as *bayes* and *grep*. We can infer that both of them share many characteristics. b) *sztod* as typical program in our internal project based on trajectory data processing, which has almost largest linkage distances with the rest workloads in the clusters. We can conclude that this program have inherent properties, which make it dissimilarity to the rest workloads.

Researchers and users can select a representative subset of these workloads by using this dendrogram. For example, if researchers want to reduce a subset includes just ten members; they can draw a vertical line at linkage distance of 4, it will yield a subset of ten workloads.

Figure 8 shows a dendrogram for typical workloads with different data volume. We apply three different input data sets to nine workloads due to constraint of experiment time. This workloads are a) *sort*, *terasort* and *wordcount*, which included in HiBench; b) *hotregion* and *sztod*, which included in our internal project; c) *grep*, *hmm*, *ibcf* and *svm*, which provided by cloudrank. We perform on this data to find similarity between input data for sets these workloads. Workloads with three data sets are represented by the name combine with input size.

As this figure 8 depicted, on the one hand, in some workloads, different input data sets appear clustered together. For example, the largest linkage distance in three input sets of *sort* is less than 2(as the vertical line a marked); *terasort* with different input sets in a same cluster when linkage distance is less than 3(as the vertical line b marked);

On the other hand, some input sets are very different from the other input sets of the same workloads. For example, three input sets of *hmm* is dissimilarity to each other according to their linkage distance, as well as *svm* and *ibcf*.

We can conclude that a few workloads have inherent property and they are tending towards stability when the data volume increases to a certain extent.

VI. Workloads based on spatio-temporal data

In this section, we present an initial idea of a big data benchmark suite for spatio-temporal data.

Shenzhen is a famous international city in south china, which covers an area of almost 2000 square kilometers. And there are more than eighteen millions of people living in it. Therefore, on each day, an amount of spatio-temporal data is produced from subway transportation system, taxicab transportation system and public bus transportation system. In particular, our data center collects and stores more than 90 million of GPS records on a daily basis. These GPS data and smart transaction data offers us the opportunity to investigate and understand the demand pattern of passengers and service level from transit agencies.

There are five typical workloads in our big data system for analyzing spatio-temporal data. All of them are used for traffic flow analysis and spatio-temporal data mining. Currently, we just release two typical workloads due to business confidentiality. These two workloads and their input data are released on the web page (http://cloud.siat.ac.cn/trajactory-data). And we will attempt to release the rest typical workloads with the permission of our collaborator.

A. Transaction Data of smart card

Sztod, there are five different subway lines, 118 subway stations and more than 20 million active smart cards in Shenzhen. On average, we collect 15 million transaction records in one day from subway transportation system. The smart card readers register passengers when they enter or leave a subway station. The information from the readers includes card id, flag of entrance or exit, timestamp, id of smart card reader and subway station name. The algorithm used for data processing is as follows:

1. Assign a subway station pair and a specific direction; count the number of people who enter the origin station and leave the destination station.
2. Repeat step 1 until all pairs of subway station are computed.
3. Find the top *N* station pairs with largest counter.

B. GPS position of Taxicab

Hotregion, at a time interval range from ten seconds to thirty seconds, more than 28000 taxicabs equipped with GPS device report their position information to our system in real time. Each record report from these taxies has four primary fields; these fields are *timestamp*, taxicab *id*, *longitude* and *latitude*. The algorithm used for data processing is as follows:

1. Divide the area into 10000*10000 grids according to latitude and longitude.
2. Assign a time range; count the number of taxicab for each grid.
3. Repeat the step 2 until all grids are processed
4. Find the top *N* grids with largest counter of taxicab.

VII. Conclusions and future work

The results show that there are characteristic redundancies between these existing MapReduce based workloads under the condition of a specific data volume range. For example, *sort* and *terasort* are clustered into one cluster due to the workload-pair sharing many same characteristics in micro-architecture level, as well as *bayes* and *grep*.

Sztod, a workload for processing trajectory data from a subway transportation system, which has inherent proprieties, and these properties make it obvious dissimiliar to the rest typical workloads we have chosen.

A few workloads such as *terasort* and *horegion*, are tending towards stability in micro-architecture level metrics when the data volume increases to a certain extent.

In future work, our research steps are as follows:

First, we will further select typical MapReduce based workloads from various application domains to conduct similarity analysis.

Second, we would like to present a big data benchmark suite for spatio-temporal data.

References

[1] http://parsa.epfl.ch/cloudsuite/cloudsuite.html

[2] http://hadoop.apache.org/mapreduce/docs/current/gridmix.html

[3] ICTBench home page. http://prof.ict.ac.cn/ICTBench/

[4] S. S. Huang, J. Huang, J. Q. Dai, T. Xie, and B. Huang. "The HiBench benchmark suite: Characterization of the MapReduce-based data analysis". Huang, Shengsheng, Jie Huang, Jinquan Dai, Tao Xie, and Bo Huang. "" In Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on, pp. 41-51. IEEE, 2010.

[5] Hive home page. http://hive.apache.org

[6] DCBench home page. DChttp://prof.ict.ac.cn/DCBench/

[7] BigDataBench home page. http://prof.ict.ac.cn/BigDataBench/

[8] CloudRank home page. http://prof.ict.ac.cn/BigDataBench/

[9] A. Phansalkar, A. Joshi, and L. K. John, "Analysis of redundancy and application balance in the SPEC CPU2006 benchmark suite". 2007. In Proceedings of the 34th annual international symposium on Computer architecture (ISCA '07). ACM, New York, NY, USA, 412-423.

[10] J. Q. Dai et al. "Hitune: dataflow-based performance analysis for big data cloud". Proc. of the 2011 USENIX ATC (2011): 87-100.

[11] W. Gao, et al. "BigDataBench: a Big Data Benchmark Suite from Web Search Engines". The Third Workshop on Architectures and Systems for Big Data (ASBD 2013) in conjunction with ISCA 2013.

[12] C. Luo, et al. "CloudRank-D: Benchmarking and Ranking Cloud Computing Systems for Data Processing Applications. Fronters of Computer Science, 2012, 6(4): 347–362

[13] Z. Chen et al. "Characterizing OS behavior of Scale-out Data Center Workloads. Seventh Annual Workshop on the Interaction amongst Virtualization, Operating Systems and Computer Architecture (WIVOSCA 2013). In Conjunction with ISCA 2013.

[14] Z, Jia, et al. "Characterizing Data Analysis Workloads in Data Centers". 2013 IEEE International Symposium on Workload Characterization （IISWC-2013)

[15] H. Xi et al. "Characterization of Real Workloads of Web Search Engines". 2011 IEEE International Symposium on Workload Characterization （IISWC-2011).

[16] Z. Jia et al. “The Implications of Diverse Applications and Scalable Data Sets in Benchmarking Big Data Systems”. Second workshop of big data benchmarking (WBDB 2012 India) & Lecture Note in Computer Science (LNCS)

[17] Y. Chen et al, “We Don’t Know Enough to make a Big Data Benchmark suite”. Workshop on Big Data Benchmarking. 2012

[18] J. Zhan et al, “High volume computing: Identify and characterizing throught oriented workloads in data centers”. In Parallel and Distributed processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International pages 1712-1721. IEEE, 2012.

[19] http://en.wikipedia.org/wiki/Principal_component_analysis

[20] http://en.wikipedia.org/wiki/K-means_clustering