HiBench: A Representative and Comprehensive Hadoop Benchmark Suite

Shengsheng Huang, Jie Huang, Yan Liu and Jinquan Dai

Intel Asia-Pacific Research and Development Ltd., Shanghai, P.R.China, 200241

{shengsheng.huang, jie.huang, yan.b.liu, jason.dai}@intel.com

I. THE HIBENCH SUITE

MapReduce and its popular open source implementation, Hadoop, are moving toward ubiquitous for Big Data storage and processing. Therefore, it is essential to quantitatively evaluate and characterize the Hadoop deployment through extensive benchmarking. In this paper, we present *HiBench* [1], a representative and comprehensive benchmark suite for Hadoop, which consists of a set of Hadoop programs including both synthetic micro-benchmarks and real-world applications. Currently the benchmark suite contains eleven workloads, classified into four categories, as shown in Table I.

TABLE I
HIBENCH WORKLOADS

Category	Workload
Micro Benchmarks	Sort
	WordCount
	TeraSort
	EnhancedDFSIO
Web Search	Nutch Indexing
	Page Rank
Machine Learning	Bayesian Classification
	K-means Clustering
Analytical Query	Hive Join
	Hive Aggregation

A. Micro Benchmarks

The *Sort*, *WordCount* and *TeraSort* programs contained in the Hadoop distribution are three popular micro-benchmarks widely used in the community, and therefore are included in HiBench. Both the Sort and WordCount programs are representative of a large subset of real-world MapReduce jobs – one transforming data from one representation to another, and another extracting a small amount of interesting data from a large data set.

In HiBench, the input data of Sort and WordCount workloads are generated using the *RandomTextWriter* program contained in the Hadoop distribution. The TeraSort workload sorts 10 billion 100-byte records generated by the *TeraGen* program contained in the Hadoop distribution.

We have also extended the *DFSIO* program contained in the Hadoop distribution to evaluate the aggregated bandwidth delivered by HDFS. The original DFSIO program only computes the average I/O rate and throughput of each map task, and it is not straightforward how to properly sum up the I/O rate or throughout if some map tasks are delayed, re-tried or speculatively executed by the Hadoop framework. The *Enhanced DFSIO* workload included in HiBench computes the aggregated bandwidth by sampling the number of bytes read/written at fixed time intervals in each map task; during the reduce and post-processing stage, the samples of each map task are linear interpolated and re-sampled at a fixed plot rate, so as to compute the aggregated read/write throughput by all the map tasks [1].

B. Web Search

The *Nutch Indexing* and *Page Rank* workloads are included in HiBench, because they are representative of one of the most significant uses of MapReduce (i.e., large-scale search indexing systems).

The Nutch Indexing workload is the indexing sub-system of Nutch [2], a popular open-source (Apache) search engine; we have used the crawler sub-system in Nutch to crawl an in-house Wikipedia mirror and generated about 2.4 million web pages as the input of this workload. The Page Rank workload is taken from a test case in SmartFrog (an open source framework for managing distributed systems) [3]; it is an open source implementation of the page-rank algorithm, a link analysis algorithm used widely in web search engines. We have used the Wikipedia page-to-page link database [4] as the input of the Page Rank workload.

C. Machine Learning

The *Bayesian Classification* and *K-means Clustering* implementations contained in Mahout [4], an open-source (Apache) machine learning library built on top of Hadoop, are included in HiBench, because they are representative of one of another important uses of MapReduce (i.e., large-scale machine learning).

The Bayesian Classification workload implements the trainer part of Naive Bayesian (a popular classification algorithm for knowledge discovery and data mining). The input of this benchmark is extracted from a subset of the Wikipedia dump. The Wikipedia dump file is first split using the built-in *WikipediaXmlSplitter* in Mahout, and then prepared into text samples using the built-in *WikipediaDatasetCreator* in Mahout. The text samples are finally distributed into several files as the input of the benchmark.

The K-means Clustering workload implements K-means (a well-known clustering algorithm for knowledge discovery and data mining). Its input is a set of samples, and each sample is

represented as a numerical d-dimensional vector. We have developed a random data generator using statistic distributions to generate the workload input.

D. Analytic Query

The Join and Aggregation queries in the Hive performance benchmarks [6] are included in HiBench, because they are representative of another one of the most significant uses of MapReduce (i.e., OLAP-style analytical queries).

Both Hive Join and Aggregation queries are adapted from the query examples in Pavlo et. al [7] and their inputs are defined in [7]. They are intended to model complex analytic queries over structured (relational) tables – Hive Aggregation computes the sum of each group over a single read-only table, while Hive Join computes the both the average and sum for each group by joining two different tables.

E. Data Compression

Data compression is aggressively used in real-world Hadoop deployments, so as to minimize the space used to storing the data, and to reduce the disk and network I/O in running MapReduce jobs. Therefore, each workload in HiBench (except for Enhance DFSIO) can be configured to run with compression turned on or off – when compression is enabled, both the input and output of the workload will be compressed (using a user specified codec), so as to evaluate the Hadoop performance with intensive data compressions.

II. Discussion and Future Work

Benchmarking is the quantitative foundation of any computer system research. In this section, we discuss the tradeoffs of existing approaches to Hadoop benchmarking, as well as possible improvements for HiBench in the future.

Existing Hadoop benchmark programs can be roughly categorized into two classes – micro-benchmarks (such as sorting programs) and synthetic workloads (such as Gridmix3 [8] and SWIM [9]). Micro-Benchmarks are important elements for evaluating Hadoop performance. In particular, the sorting program has been pervasively accepted as an important performance indicator of MapReduce, because sorting is an intrinsic behavior of the MapReduce framework. For instance, both Yahoo and Google have used TeraSort to evaluate their MapReduce cluster [10][11]. Therefore, they are included in HiBench, even though they are just micro level benchmarks and do not exhibit some important characteristics of real world Hadoop applications.

Synthetic workloads (such as GridMix3 and SWIM) intend to model the characteristics of a Hadoop cluster by collecting traces of all the jobs in the cluster over an extended period of time, synthesizing the workload based on the traces, and executing the workloads via replaying the synthesized traces. In this manner, they can potentially evaluate the effects of the interactions of concurrent Hadoop jobs, which is impossible if just a single Hadoop job is executed from start to end. On the other hand, synthetic workloads are as good as their models, and it is unclear how representative their models are of the behaviors of Hadoop clusters. For instance, GridMix3 and SWIM only models the data read/written by each map or reduce task, while completely ignores the computing characteristics of these tasks. Even though there are some efforts on adding compute models to GridMix3 [12], only simple instructions such as ADD or SQRT are used to emulate the CPU utilizations; consequently, they cannot properly evaluate the impacts of, for instance, better compression or vector instructions on Hadoop performance. In addition, real world Hadoop applications may have data access patterns outside the original MapReduce model that is strictly followed by these synthetic workloads; for instance, Nutch Indexing needs to read/write a lot of temporary files on local disks in its reduce tasks, which cannot be replayed by these workloads.

A lot of users (both inside and outside of Intel) have used HiBench extensively for their Hadoop evaluation and tuning. We have focused on providing real-world applications for Hadoop benchmarking in HiBench; consequently, these users can clearly articulate the behaviors and characteristics of specific Hadoop applications in their work. In future, we would like to explore several improvements to HiBench based on the user feedbacks, including broader coverage of Hadoop applications, better support of scale-up and scale-down of cluster sizes, and evaluations of the interactions of concurrent Hadoop jobs.

References

[1] S. Huang, J. Huang, J. Dai, T. Xie, and B. Huang, "The HiBench Benchmark Suite: Characterization of the MapReduce-Based Data Analysis", ICDEW, March, 2010.

[2] Nutch homepage. http://lucene.apache.org/nutch/

[3] P. Castagna, "Having fun with PageRank and MapReduce," Hadoop User Group UK talk. Available: http://static.last.fm/johan/huguk-20090414/paolo_castagna-pagerank.pdf

[4] H. Haselgrove, "Using the Wikipedia page-to-page link database," Available: http://users.on.net/~henry/home/wikipedia.htm

[5] Mahout homepage. http://lucene.apache.org/mahout/

[6] "A Benchmark for Hive, PIG and Hadoop", http://issues.apache.org/jira/browse/HIVE-396

[7] A. Pavlo, A. Rasin, S. Madden, M. Stonebraker, D. DeWitt, E. Paulson, L. Shrinivas, and D. J. Abadi. "A Comparison of Approaches to Large-Scale Data Analysis", *SIGMOD*, June, 2009

[8] GridMix3. http://hadoop.apache.org/mapreduce/docs/current/gridmix.html

[9] Y. Chen, A. Ganapathi, R. Griffith, R. Katz. "The Case for Evaluating MapReduce Performance Using Workload Suites", MASCOTS, 2011.

[10] O. O'Malley and A. C. Murthy, "Winning a 60 Second Dash with a Yellow Elephant", http://sortbenchmark.org/Yahoo2009.pdf

[11] "Sorting 1PB with MapReduce", http://googleblog.blogspot.com/2008/11/sorting-1pb-with-mapreduce.html

[12] "Emulate CPU Usage of Tasks in GridMix3". https://issues.apache.org/jira/browse/MAPREDUCE-2106