BigDataBench

User's Manual

Contents

1. Background......3
2. Introduction of BigDataBench......3
2.1 BigDataBench 1.0......3
2.2 Usage and licences of BigDataBench......3
2.3 Design Methodology......3
2.3.1 Following An Incremental Approach......4
2.3.2 Methodology of Generating Big Data......4
2.3.3 Considering Variety of Workloads......5
2.4 Workloads of BigDataBench......5
2.5 Brief introduction of workloads......5
2.5.1 Basic operation......5
2.5.2 Classifier......6
2.5.3 Service......7
3. Prerequisite Software Packages......7
3.1 Brief introduction of basic softwares......7
3.2 Set up......8
4. How to use BigDataBench......8
4.1 Files structure and explanation......9
4.2 Generate the Big Data......10
4.2.1 Prepare for generate big data......10
4.2.2 Expand to big data......10
4.3 Configuring Workloads......11
4.3.1 Convert the Data Format......11
4.3.2 Run the Big Data Benchmark......11
Reference......13

This document presents information on BigDataBench---a big data benchmark suite from web search engines, including a brief introduction and the usage of it. The information and specifications contained are for researchers who are interested in big data benchmarking.

Publishing information:

Release	**1.0**
Date	**5/30/2013**

Contact information:

Emails: **gaowanling@ict.ac.cn**
wl@ncic.ac.cn

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

1. Background

As information is growing explosively, more and more data are created every day. However, there is no existing benchmark suite for big data applications. For this reason, we propose a new benchmark suite----BigDataBench, a big data benchmark suite from web search engines, to benchmark and rank systems that are running big data applications. Search engine service providers treat data, applications, and web access logs as business confidentiality, so the real big data is unavailable. Even if open data are available, downloading PB-scale data is not acceptable, so we determine to generate almost real big data from small-scale real data. We consider representative application domains, typical workloads, and the way to generate big data into the design of BigDataBench. The big data we generated should preserve the characteristics of the small-scale data.

2. Introduction of BigDataBench

2.1 BigDataBench 1.0

Big Data Benchmark is a benchmark suite for big data applications. The first release consists of 6 applications (Sort, Grep, Wordcount, Naive Bayes, Support Vector Machine(SVM) and Nutch Server) that have been selected based on their popularity in Search Engine which is one of today's represented scenarios. Meanwhile, it provides a data generation tool which can generate big data with user-specified data scales. All the analysis workloads including Sort, Grep, Wordcount, Naive Bayes and SVM use the unified data set.

2.2 Usage and licenses of BigDataBench

BigDataBench is available for researchers interested in pursuing research in the field of big data application. BigDataBench's software components are all available as open-source software. All of the software components are governed by their own licensing terms. Researchers intending to use BigDataBench are required to fully understand and abide by the licensing terms of the various components.

2.3 Design Methodology[1]

Considering the complexity, diversity, workload churns, and rapid evolution of big data systems, we take an incremental approach in big data benchmarking. For the first step, we pay attention to search engines, which are the most important domain in

Internet services in terms of the number of page views and daily visitors.

2.3.1 Following an Incremental Approach

For the first step, we single out the most important application domain. Firstly, we pay attention to Internet services, and rank the main application domains according to a widely acceptable metric —the number of page views and daily visitors. We investigate the top sites listed in Alexa [2], of which the rank of the sites is calculated using a combination of average daily visitors and page views. We find the search engine is the mostly popular application domain, 40% of the top sites are search engines, such as Google, Bing and etc. Data from the Internet study [3] also prove the popularity of search engines, which shows that 92% of online adults use search engines to find information on the web. So we focus on the search engine firstly.

For the second step, we choose typical workloads from web search engines as candidates of our BigDataBench.

Considering the workload churns and emerging workloads domains, we believe that the mature big data benchmark suites will take a long way to go. We will continuously add more representative applications and remove the out-of-data applications.

2.3.2 Methodology of Generating Big Data

Big data benchmarks require big data set as the input to drive their workloads. Our previous work [4] collected three search engine companies' traces and also found that the frequently used distributions cannot capture the key characteristics of real data. Moreover, there is a large performance gap between running search engines with real data and randomly generated data [4]. In this case, only real data can reflect the real system behaviors, and hence the real life data is preferred [5] in big data benchmarks. However, it is a big challenge to obtain real big data as follows: Firstly, most of end user do not own real big data whereas Internet service companies who own the real life big data would not like to share big data for commercial confidentiality and user privacies; secondly, even though big data is openly available, downloading terabyte or even gigabyte scale data is too costly to be acceptable.

Based on the above reasons, we would like to generate synthetic data preserving key characteristics of real data. There are two key characteristics we must consider – semantic and locality. Semantic characteristic reflects the insightful meaning of real life data. Locality reflects the data access patterns. We investigate the real life data with the purpose of getting a semantic model and a locality model. We find the real search engine query terms follow zipf's law [4]. We then generate the synthetic query trace on the basis of the real search query terms we have gotten and let the query terms follow zipf's law. The temporal locality can be reflected by using reuse distance. We calculate each real term's reuse distance and generate the synthetic data according to the real terms' reuse distance.

2.3.3 Considering Variety of Workloads

In addition to three ”V” of big data: volumes, velocity, and variety, our previous work [6] showed that diversity of workloads must be considered in big data benchmarking since they have different characteristics in term of computation behaviors, memory and I/O access patterns. A search engine involves in many important workloads, and we must choose typical workloads for assembling BigDataBench.

2.4 Workloads of BigDataBench

All the workloads in Big Data Benchmark are as follows:

	Applications	Source
Basic Operations	Sort	Hadoop example
	Wordcount	
	Grep	
Classification	Naive Bayes	Mahout
	SVM	Implemented based on libsvm
Service	Nutch Server	Search Benchmark[7]

Table 1. Workloads in BigDataBench

2.5 Brief introduction of workloads

2.5.1 Basic operation

Sort

The Sort----simply uses the MapReduce framework to sort the input directory into the output directory. The inputs and outputs must be Sequence files where the keys and values are BytesWritable. The mapper is the predefined IdentityMapper and the reducer is the predefined IdentityReducer, both of which just pass their inputs directly to the output.

WordCount

WordCount----reads text files and counts how often words occur. The input is text files and the output is text files, each line of which contains a word and the count of how often it occurred, separated by a tab. Each mapper takes a line as input and breaks it into words. It then emits a key/value pair of the word and constant 1. Each reducer sums the counts for each word and emits a single key/value with the word and sum. As an optimization, the reducer is also used as a combiner on the map outputs. This reduces the amount of data sent across the network by combining each word into a single record.

Grep

Grep----extracts matching strings from text files and counts how many times they occurred. The program runs two MapReduce jobs in sequence. The first job counts how many times a matching string occurred and the second job sorts matching strings by their frequency and stores the output in a single output file. Each mapper of the first job takes a line as input and matches the user-provided regular expression against the line. It extracts all matching strings and emits key/value pairs of (matching string, 1). Each reducer sums the frequencies of each matching string. The output is sequence files containing the matching string and count. The reduce phase is optimized by running a combiner that sums the frequency of strings from local map output.

2.5.2 Classifier

Naive Bayes

Naive Bayes----classifier is a simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions. Here we use the input data set below to drive the Mahout Naïve Bayes classifier.
Input data set: 20 Newsgroups
The 20 Newsgroups data set is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. The 20 newsgroups collection has become a popular data set for experiments in text applications of machine learning techniques, such as text classification and text clustering. We will use Mahout Bayes Classifier to create a model that would classify a new document into one of the 20 newsgroups.
Users can download the data set from here: Wikipedia dump
Mahout Bayes Classifier will split Wikipedia dump up into chunks. These chunks are then further split by country. From these splits, a classifier is trained to predict what country an unseen article should be categorized into.

Support Vector Machine

SVM----supervised learning models with associated learning algorithms that analyze data and recognize patterns. It is always used for classification and regression analysis. The basic SVM is a non-probabilistic binary linear classifier taking a set of input data and predicts, and forming the output with two possible classes. In addition to performing linear classification, SVM can efficiently perform non-linear classification by mapping their inputs into high-dimensional feature spaces[8].
We implement SVM on Hadoop using Hadoop-streaming framework and we use the 20 Newsgroups data set mentioned above as the training set.

2.5.3 Service

Nutch Server

Nutch is an open source web-search software project. Nutch provides a transparent alternative to commercial web search engines. It has a highly modular architecture, allowing developers to create plug-ins for media-type parsing, data retrieval, querying and clustering.
For the latest information about Nutch, please visit the website at:
    http://lucene.apache.org/nutch/
and the wiki, at:
    http://wiki.apache.org/nutch/
To get started using Nutch read Tutorial:
    http://lucene.apache.org/nutch/tutorial.html

3. Prerequisite Software Packages

3.1 Brief introduction of basic software

3.1.1 Hadoop

We recommend version 1.0.2, which was used and tested in our environment. Download link:
mirrors.tuna.tsinghua.edu.cn/apache/hadoop/common/hadoop-1.0.2

3.1.2 Mahout

We recommend mahout-distribution-0.6, which was used and tested in our environment.
Download link:
http://mirrors.tuna.tsinghua.edu.cn/apache/mahout/

3.2 Set up

3.2.1 Setting up Hadoop

Our benchmark now runs on Hadoop framework, so we need to deploy Hadoop environment at first. To install Hadoop on cluster, users have to unpack the Hadoop package on all the cluster nodes with same path.

3.2.2 Setting up Mahout

Unpack the package,

```
tar -zxvf mahout-0.6.tar.gz
```

It's easy to install Mahout. Users just need to add Mahout path to environment variable of computer, that is, modify the file ~/.bashrc. The commands are:

(1) Open the file ~/.bashrc

```
vim ~/.bashrc
```

(2) Add the following two sentences in the file

```
export MAHOUT_HOME=/path/to/mahout-distribution-0.6
export PATH=${MAHOUT_HOME}/bin:${PATH}
```

(3) Make the changes work

```
source ~/.bashrc
```

(4) Test the set up

```
mahout －help
```

If it lists much information about usage that means it has installed the Mahout successfully.

Link:
https://cwiki.apache.org/MAHOUT/mahout-wiki.html#MahoutWiki-Installation%252FSetup

3.2.3 Setting up BigDataBench

Users just need to decompress the package on the specified directory.

```
tar -zxvf BigDataBench-1.0.tar.gz
```

In our BigDataBench-1.0 package, there are four main folders:

TextProduce----store data generation tools;

ToSeqFile----store script for data format conversion;

RunWorkload----store scripts running Sort, Grep, Wordcount and Naive Bayes;

SVM----store relative material running SVM;

Nutch----store search server

4. How to use BigDataBench

In this part, we will introduction how to run the Big Data Benchmark for each workload. It mainly has two steps. The first step is to generate the big data and the second step is to run the applications using the data we generated.

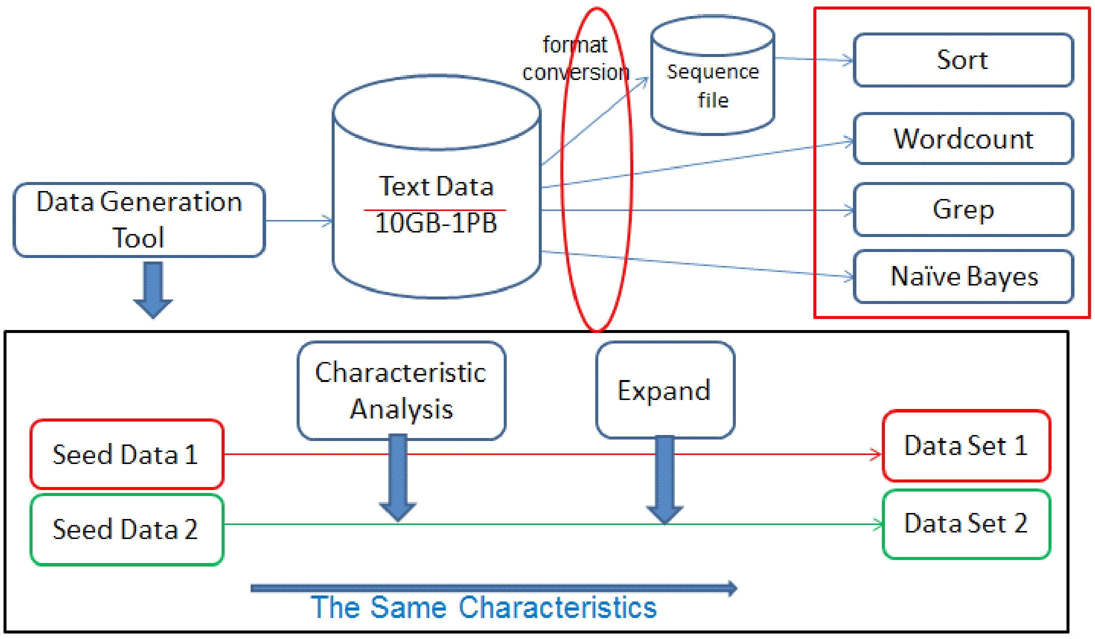

Figure 1. Use Flows of BigDataBench

We provide a data generation tool to complete the first step, and this process contains two substeps. The first substep is to prepare for generate big data by analyzing the characteristics of the seed data which is real owned by users, and the second substep is to expand the seed data to big data maintaining the features of seed data.

After generating the big data, we integrate a series of workloads to process the data.

4.1 File structure and explanation

After unpacking the package, users will see four main folders TextProduce, ToSeqFile, RunWorkload and Nutch.

TextProduce/

count.jar: prepare to generate big data by analyzing the characteristics of small-scale data

TextProduce.jar: expand to a larger data set according to the analysis results of count.jar

bayes-test-input/: store 20newsgroup of wiki data

ToSeqFile/

ToSeqFile$Map.class, ToSeqFile.class, ToSeqFile.jar: these three are used to convert text files to sequence files, you should copy these three to the folder $HADOOP_HOME.

sort-transfer.sh: call ToSeqFile.jar to convert data format.

RunWorkload/

run-sort.sh: run sort application

run-grep.sh: run grep application
run-wordcount.sh: run wordcount application
run-train.sh: run naïve bayes application for training phase
run-bayes.sh: run naïve bayes application for classification phase

SVM/

ForSvm$Map.class, ForSvm.class, ForSvm.jar: these three are used to process text files adapting to the requirements of SVM program, you should copy these three to the folder $HADOOP_HOME.

RunForSvm.sh: call ForSvm.jar to process text files.
en_svm/: store the predicts and running programs of SVM.

Nutch/

see README.txt

4.2 Generate the Big Data

We provide a data generation tool which can generate data with user specified data scale. It has two steps, firstly, it analyze the seed data which is the user-owned small-scale data; secondly, it generate the big data based on the output of the first step.

4.2.1 Prepare to generate big data

To run the program:
$HADOOP_HOME/bin/hadoop jar count.jar <in-dir> <out-dir>
It has two parameters. The first one is the directory of the input file which saves the seed data. The second one is the directory of the output file which saves the characteristics of the seed data.

4.2.2 Expand to big data

To run the program:
$HADOOP_HOME/bin/hadoop jar TextProduce.jar <in-dir> <out-dir> <type-num> <data-size> <group-divide>
It has five parameters:
The first one is the directory of the characteristics of the seed data which is the output file of the first step.
The second one is the directory of the output file which saves the big data.
The third one is the numbers of types in the seed data.
The forth one is the numbers of news need to be generated, because our seed data has approximately 7500 news and its data size is 10M, so if we want to generate the data of 100G, the fourth parameter will be 75000000.

The fifth one is the groups of each type of news will be divided. This parameter is adjustable to the reduce slots in the cluster. Generally, the more the number of reduce slots, the bigger parameter should be set. The number of types in our seed data is 20, and the number of reduce slots for each slave node is 8. Considering that we have 14 slave nodes, so we have 112(14*8) reduce slots in all. In order to execute the job without waiting, we'd better set this parameter no more than 112/20, so we choose 5 in this situation.

4.3 Configuring Workloads

We use the big data we generated as the input of our benchmarks, and it has two steps. The first step is to convert the data format, and here it mainly converts the text file to sequence file which is needed only for the sort application. The second step is to run applications using the big data.

4.3.1 Convert the Data Format

For the sort application, its inputs must be sequence files, while the data generated by the data generation tool for now is text files, so we have to convert the data format so that it can be processed by sort application.
To run the program:
./sort-transfer.sh <in-dir> <out-dir>
It has two parameters, the first one is the directory of the input file which contains the original data, and the second one is the directory of the output file which saves the conversion data.

4.3.2 Run the Big Data Benchmark

Sort
To run the program:
./run-sort.sh <in-dir> <out-dir>
To run the benchmark, generate the big data using the data generation tool we have introduced, and convert the data format from text file to sequence file, then we can sort the data.
It has two parameters. The first one is the directory of the big data, such as /user/root/seq-out-100G, which is the directory of seq-out-100G. The second one is the directory of the output file.

Wordcount
To run the this benchmark, the command syntax is
./run-wordcount.sh <in-dir> <out-dir>

To run the benchmark, generate the big data using the data generation tool we have introduced, then it can read the input file and count how often the words occurred.
It has two parameters. The first one is the directory of the big data, such as /user/root/seed-out-100G, which is the directory of seed-out-100G. The second one is the directory of the output file.

Grep

To run the this benchmark, the command syntax is
./run-grep.sh <in-dir> <out-dir>
To run the benchmark, generate the big data using the data generation tool we have introduced, then we can abstract content from the text file.
It has two parameters. The first one is the directory of the big data, such as /user/root/seed-out-100G, which is the directory of seed-out-100G. The second one is the directory of the output file.

Naive Bayes

Training Phase

Command: ./run-train.sh <in-dir> <out-dir>
It has two parameters. The first one is the directory of the small-scale data which is used to train. The second one is the directory of the output which is the training model. Once the model is built, then we can classify big data based on the model.

Classification Phase

Command：./run-bayes.sh <in-dir> <out-dir>
It has two parameters. The first one is the directory of the training model which is the output of the training phase. The second one is the directory of the output which stores the classification results.

SVM

Processing text files

Command: ./RunForSvm.sh <in-dir> <out-dir>
It has two parameters. The first one is the directory of text files which generated by the data generation tool. The second one is the directory of the output which adapts to the requirements of SVM.

Classification Phase

Before running following command, you need to change the parameters in svm2-hadoop.sh. The first one is the directory of hadoop-streaming jar which is /liuwb/hadoop-1.0.2/contrib/streaming/hadoop-streaming-1.0.2.jar in our cluster; The second one is the directory of simple_svm which is /gwl/wiki-cmp/svm/en_svm/simple_svm in our cluster. The other parameters depend on your requirements.
Command: ./svm2-hadoop.sh <in-dir> <out-dir>

It has two parameters. The first one is the directory of processed text files which is the output of the processing step. The second one is the directory of the output which stores the classification results.

Nutch Server

To run the program:
Please visit our website at:
    http://prof.ict.ac.cn/DCBenchmarks/
and downloading:

- Search[6]:

Search_manual_v1.0.pdf   Search-v1.0.tar.gz

Due to the lack of permission to probe real-world web search engines, we set up a search server in our lab using Nutch as the search engine, and SoGou web corpus as the indices and snapshot data. However, we have obtained permission to use a real workload trace from SoGou. We have released the search system as a benchmark for datacenter computing, which is named Search.

Reference

[1] W. Gao, Y. Zhu, Z. Jia, C. Luo, L. Wang, J. Zhan, Y. He, S. Gong, X. Li, S. Zhang, and B. Qiu. 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 The 40th International Symposium on Computer Architecture, 2013.
[2] Alexa website. http://www.alexa.com/topsites/global.
[3] http://searchenginewatch.com/article/2101282/Who-Uses-Search-Engines-92-of-Adult-U.S.-Internet-Users-Study.
[4] H. Xi, J. Zhan, Z. Jia, X. Hong, L. Wang, L. Zhang, N. Sun, and G. Lu. Characterization of real workloads of web search engines. In Workload Characterization (IISWC), 2011 IEEE International Symposium on, volume 11, pages 15–25. IEEE, 2011.
[5] Y. Chen. We Don't Know Enough to make a Big Data Benchmark Suite. Workshop on Big Data Benchmarking. 2012.
[6] Z. Jia, L. Wang, W. Gao, J. Zhan, and L. Zhang. The Implications of Diverse and Scalable Data Sets in Benchmarking Big Data Systems. Second Workshop on Big Data Benchmarking. 2012.
[7] H. Xi, J. Zhan, et al. Characterization of Real Workloads of Web Search Engines. 2011 IEEE International Symposium on Workload Characterization (IISWC-2011).

2011.
[8] https://en.wikipedia.org/wiki/Support_vector_machine