BigDataBench-SparkStreaming User Manual

1. Introduction

BigDataBench-SparkStreaming based on the stream computing system SparkStreaming, it contains three benchmark programs:

(1) BigDataBench-SparkStreaming-TopN

(2) BigDataBench-SparkStreaming-Grep

(3) BigDataBench-SparkStreaming-Kmeans

2. Deploy Spark

Please refer to the section "3.3 Setting up Spark" in 《BigDataBench User Manual》.

3. Running the Workloads

3.1 Application Domain- E-commerce

3.2 Application Domain- Social Networks

3.2.1 Workload - SparkStreaming-Kmeans

1. Required Software Stacks

SparkStreaming: 1.6

Hadoop: 2.6

2. Download workloads

Download the Benchmark form this link:

http://prof.ict.ac.cn/bdb_uploads/bdb_streaming/BigDataBench_V3.2_SparkStreaming.tar.gz

Unpack the downloaded tar file:

```
$tar xzvf BigDataBench_V3.2_SparkStreaming.tar
```

3. Prepare the input

Generate training files and test files.

Open one terminal, and generate training files.

```
$cd BigDataBench/SparkStreaming-KMeans/data/
$vim KMeansTrainDataGenerator.sh
```

Modify 'benchmark_home' and 'destinationDir', for example:

benchmark_home=$SparkStreaming-KMeans

destinationDir=$benchmark_home/data/kmeans_train

jar_home=$benchmark_home/data

Run KMeansTrainDataGenerator.sh:

```
$ ./KMeansTrainDataGenerator.sh
```

Open another terminal, and generate test files.

```
$cd SparkStreaming-KMeans/data/
$vim KMeansTestDataGenerator.sh
```

Modify 'benchmark_home' and 'destinationDir', for example:

benchmark_home=$SparkStreaming-KMeans

destinationDir=$benchmark_home/data/kmeans_test

jar_home=$benchmark_home/data

Run KMeansTestDataGenerator.sh:

```
$ ./KMeansTestDataGenerator.sh
```

4. Run the workload

Make sure Spark have been successfully started, and then commit the SparkStreaming jar:

```
$cd SparkStreaming-KMeans
$vim run-sparkstreaming-kmeans.sh
```

Modify run-sparkstreaming-topn.sh, for example：

benchmark_home=$SparkStreaming-TopN

spark_home=/root/spark/spark-1.6.0-bin-hadoop2.6

jar_home=$benchmark_home/out/artifacts/bigdatabench_sparkstreaming_kmeans_jar

trainingDir=file:///$benchmark_home/data/kmeans_train

testDir=file:///$benchmark_home/data/kmeans_test

And modify the following parameters on demand：

<trainingDir> <testDir> <batchDuration> <numClusters> <numDimensions>

Then, run run-sparkstreaming-kmeans.sh:

```
$ ./run-sparkstreaming-kmeans. sh
```

The information of selecting workload will be printed on the screen:

…….

16/03/02 12:42:30 INFO DAGScheduler: Job 8 finished: main at NativeMethodAccessorImpl.java:-2, took 0.054434 s

---

Time: 1456893750000 ms

(0.0,2)

(1.0,2)

(2.0,0)

(3.0,1)

(4.0,2)

(5.0,1)

(6.0,2)

(7.0,0)

(8.0,0)

(9.0,0)

…

16/03/02 12:42:30 INFO JobScheduler: Finished job streaming job 1456893750000 ms.1 from job set of time 1456893750000 ms

3.2.1 Workload – SparkStreaming-TopN

1. Required Software Stacks

Spark: 1.6

Hadoop: 2.6

2. Download workloads

Download the Benchmark form this link:

http://prof.ict.ac.cn/bdb_uploads/bdb_streaming/BigDataBench_V3.2_SparkStreaming.tar.gz

Unpack the downloaded tar file:

```
$tar xzvf BigDataBench_V3.2_SparkStreaming.tar
```

3. Prepare the input

Open one terminal, and run the "wordGenerator.sh" script.

```
$cd BigDataBench/SparkStreaming-TopN/data/
$vim wordGenerator.sh
```

Modify 'sourcefile' and 'destinationDir', for example:

sourcefile=$SparkStreaming-TopN/data/word.txt

destinationDir=$SparkStreaming-TopN/data/input_topN

Run wordGenerator.sh:

```
$ ./wordGenerator.sh
```

It will generate txt files in the directory 'destinationDir'.

4. Run the workload

Make sure Spark have been successfully started, and then commit the JStorm topology:

Open another terminal:

```
$cd SparkStreaming-TopN/
$vim run-sparkstreaming-topn.sh
```

Modify run-sparkstreaming-topn.sh, for example：

benchmark_home=$SparkStreaming-TopN

spark_home=/root/spark/spark-1.6.0-bin-hadoop2.6

jar_home=$benchmark_home/out/artifacts/bigdatabench_sparkstreaming_topn_jar

data_home=file:///$benchmark_home/data/input_topN

And modify the following parameters on demand：

<directory> <top>

Then, run run-sparkstreaming-topn.sh:

```
$ ./run-sparkstreaming-topn. sh
```

The information of selecting workload will be printed on the screen:

…….

16/03/01 19:09:10 INFO DAGScheduler: Job 9 finished: count at TopKMain.scala:37, took 0.048571 s

Top hashtags in last 30 seconds (452 total):

the (53 words)

and (46 words)

data (37 words)

to (35 words)

of (32 words)

performance (24 words)

a (23 words)

(19 words)

workload (18 words)

can (17 words)

16/03/01 19:09:10 INFO JobScheduler: Finished job streaming job 1456830550000 ms.0 from job set of time 1456830550000 ms

……

3.3 Application Domain- SearchEngine

3.3.1 Workload - SparkStreaming-Grep

1. Required Software Stacks

Spark: 1.6

Hadoop: 2.6

2. Download workloads

Download the Benchmark form this link:

http://prof.ict.ac.cn/bdb_uploads/bdb_streaming/BigDataBench_V3.2_SparkStreaming.tar.gz

Unpack the downloaded tar file:

```
$tar xzvf BigDataBench_V3.2_SparkStreaming.tar
```

3. Prepare the input

Open one terminal, and run the "run-datagenerator.sh" script.

```
$cd BigDataBench/SparkStreaming-Grep/
$vim run-datagenerator.sh
```

Modify 'benchmark_home' and 'spark_home', for example:

benchmark_home=$SparkStreaming-Grep

spark_home=/root/spark/spark-1.6.0-bin-hadoop2.6

Run run-datagenerator.sh:

```
$ ./run-datagenerator.sh
```

Screen output is as follows:

Listening on port 9999

4. Run the workload

Make sure Spark have been successfully started, and then commit the SparkStreaming jar:

Open another terminal:

```
$cd SparkStreaming-Grep/
$vim run-sparkstreaming-grep.sh
```

Modify run-sparkstreaming-grep.sh, for example：

benchmark_home=$SparkStreaming-Grep

spark_home=/root/spark/spark-1.6.0-bin-hadoop2.6

And modify the following parameters on demand：

<numStreams> <host> <port> <batchMillis>

Then, run run-sparkstreaming-grep.sh:

```
$ ./run-sparkstreaming-grep.sh
```

The information of selecting workload will be printed on the screen:

Received 9291 records

16/03/01 18:34:59 INFO JobScheduler: Total delay: 0.893 s for time 1456828499000 ms (execution: 0.807 s)

Received 92910 records

16/03/01 18:35:00 INFO JobScheduler: Total delay: 0.818 s for time 1456828500000 ms (execution: 0.792 s)

Received 92910 records16/03/01 18:35:01 INFO RawNetworkReceiver: Read a block with 1003520 bytes

1. Appendix

Format specification

Bold for emphasis.

Italic for fold and file names

`$command` for command lines

Contents for contents in configuration files

Some exception explanations should be put in footnote