BigDataBench Subset I

User's Manual

1 Cloud OLTP

We use YCSB to run database basic operations. And, we provide the HBase to run operations for each operation.

HBase_Write

To Prepare

1. tar zxvf Marginal_ones_as_representatives.tar.gz
2. cd $hbase
3. bin/hbase shell
   create 'usertable','f1','f2','f3'
4. cd /Marginal_ones_as_representatives/Cloud_OLTP_Read/ycsb-0.1.4
5. bin/ycsb load hbase -P workloads/workloadc -p threads=<thread-numbers> -p columnfamily=<family> -p recordcount=<recordcount-value> -p hosts=<hostip> -s>load.dat

A few notes about this command:

- <thread-number> : the number of client threads, this is often done to increase the amount of load offered against the database.
- <family> : In Hbase case, we used it to set database column. You should have database usertable with column family before running this command. Then all data will be loaded into database usertable with column family.
- <recorcount-value>: the total records for this benchmark. For example, when you want to load 10GB data you shout set it to 10000000.
- <hostip> : the IP of the hbase's master node.

To run

1. cd Marginal_ones_as_representatives/Cloud_OLTP_Read/ycsb-0.1.4
2. sh bin/ycsb run hbase -P workloads/workloadc -p threads=<thread-numbers> -p columnfamily=<family> -p operationcount=<operationcount-value> -p hosts=<hostip> -s>tran.dat

A few notes about this command:

- <thread-number> : the number of client threads, this is often done to increase the amount of load offered against the database.
- <family> : In Hbase case, we used it to set database column. You should have database usertable with column family before running this command. Then all data will be loaded into database usertable with column family.
- < operationcount-value >: the total operations for this benchmark. For example, when you want to load 10GB data you shout set it to 10000000.
- <hostip> : the IP of the hbase's master node.

2 Hadoop-version

Wordcount,Grep

To prepare

1. tar zxvf Marginal_ones_as_representatives.tar.gz
2. cd Marginal_ones_as_representatives/Hadoop_wordcount_grep
3. sh genData_MicroBenchmarks.sh

To run

sh run_MicroBenchmarks.sh

Naive Bayes

To prepare

1. tar zxvf Marginal_ones_as_representatives.tar.gz
2. cd Marginal_ones_as_representatives/Hadoop_Bayes
3. tar zxvf mahout-distribution-0.6.tar.gz
4. export Marginal_ones_as_representatives/Hadoop_Bayes/mahout-distribution-0.6
   and then you can run it

Basic command-line usage

1. cd Marginal_ones_as_representatives/Hadoop_Bayes
2. sh genData_naivebayes.sh

To run

sh run_naivebayes.sh

3 Spark-version

Sort, Grep, Wordcount

(If you use not one machine you must download the spark on each machines,and must download in the right way**)**

To prepare

1. tar zxvf Marginal_ones_as_representatives.tar.gz
2. cd Marginal_ones_as_representatives/Spark_wordcount_grep_sort
3. sh genData_MicroBenchmarks.sh
   sh sort-transfer.sh

(before to run the sort you should put the sort-transfer file in your hadoop and then to run sort-transfer.sh,

the sort-transfer you can find in Marginal_ones_as_representatives)

To run
when you chose sort like this
./run-bigdatabench cn.ac.ict.bigdatabench.Sort <master> <data_file> <save_file> [<slices>]
parameters:
<master>: URL of Spark server, for example: spark://172.16.1.39:7077
<data_file>: the HDFS path of input data, for example: /test/data.txt
<save_file>: the HDFS path to save the result
[<slices>]: optional, times of number of workers

when you chose grep like this
./run-bigdatabench cn.ac.ict.bigdatabench.Grep <master> <data_file> <keyword> <save_file> [<slices>]
parameters:
<master>: URL of Spark server, for example: spark://172.16.1.39:7077
<data_file>: the HDFS path of input data, for example: /test/data.txt
<keyword>: the keyword to filter the text
<save_file>: the HDFS path to save the result
[<slices>]: optional, times of number of workers

When you chose wordcount lie this
./run-bigdatabench cn.ac.ict.bigdatabench.WordCount <master> <data_file> <save_file> [<slices>]
parameters:
<master>: URL of Spark server, for example: spark://172.16.1.39:7077
<data_file>: the HDFS path of input data, for example: /test/data.txt
<save_file>: the HDFS path to save the result
[<slices>]: optional, times of number of workers

PageRank

To prepare
1. tar zxvf Marginal_ones_as_representatives.tar.gz
2. cd /Marginal_ones_as_representatives/Spark_Pagerank/Pagerank
3. sh genData_PageRank.sh

To run
./run-bigdatabenchorg.apache.spark.examples.PageRank
<master> <file> <number_of_iterations> <save_path> [<slices>]
parameters:
#<master>: URL of Spark server, for example: spark://172.16.1.39:7077
#<file>: the HDFS path of input data, for example: /test/data.txt

#<number_of_iterations>: number of iterations to run the algorithm
#<save_path>: path to save the result
#[<slices>]: optional, times of number of workers

Kmeans

The Kmeans program we used is obtained from Mahout-0.8.
(If you use not one machine you must download the spark on each machines,and must download in the right way**)**

To prepare

1. tar zxvf Marginal_ones_as_representatives.tar.gz
2. cd Marginal_ones_as_representatives/Spark_Kmeans /Kmeans
3. sh genData_ Kmeans.sh

To run

./run-bigdatabench org.apache.spark.mllib.clustering.KMeans <master> <input_file> <k> <max_iterations> [<runs>]

parameters:
#<master>: URL of Spark server, for example: spark://172.16.1.39:7077
#<input_file>: the HDFS path of input data, for example: /test/data.txt
#[<k>]: number of centers
#<max_iterations>: number of iterations to run the algorithm
#[<runs>]: optional, level of parallelism to split computation into

4 Hive-version

Hive_Difference

To prepare

Generat Table_data
1. tar zxvf Marginal_ones_as_representatives.tar.gz
2.cd /Marginal_ones_as_representatives/BigDataGeneratorSuite/Table_datagen/e-com
3. java -XX:NewRatio=1 -jar pdgf.jar -l demo-schema.xml -l demo-generation.xml -c -s ‐sf <parameter>
4. mkdir BigDataBench/BigDataGeneratorSuite/Table_datagen/
5.mv Marginal_ones_as_representatives/BigDataGeneratorSuite/Table_datagen/e-com/output /BigDataBench/BigDataGeneratorSuite/Table_datagen

Greate table
1. cd $HIVE_HOME/bin
2. ./hive

create database bigdatabench;
use bigdatabench;
create table bigdatabench_dw_order(order_id int,buyer_id int,create_date string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' STORED AS TEXTFILE;

create table bigdatabench_dw_item(item_id int,order_id int,goods_id int,goods_number double,goods_price double,goods_amount double)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' STORED AS TEXTFILE;
load data local inpath
'BigDataBench/BigDataGeneratorSuite/Table_datagen/output/OS_ORDER.txt'
overwrite into table bigdatabench_dw_order;
load data local inpath 'BigDataBench
/BigDataGeneratorSuite/Table_datagen/output/OS_ORDER_ITEM.txt' overwrite into table bigdatabench_dw_item;
create table item_temp as select ORDER_ID from bigdatabench_dw_item;

To run

cd Marginal_ones_as_representatives/Hive_Difference
sh BigOP-e-commerce-difference.sh

(For ease of use, we recommend that you use a local mysql server to store metadata)

Hive_TPC_DS_query3

To prepare

Generate data

1. cd Marginal_ones_as_representatives/Hive_TPC_DS_query3
2. ./dsdgen -scale <data-size> -dir <data-directory>

A few notes about this command:

- <data-size> : the number of the data size, the unite is GB.
- <data-directory> :the path where your data want to put.

./mvdata.sh <data-directory>

Create the TPC-ds tables

1. Firstly, you should put TPC-DS data into the hdfs.
2. cd Marginal_ones_as_representatives/Hive_TPC_DS_query3
3. sh create-tpcds-tables.sh

To run

cd Marginal_ones_as_representatives/Hive_TPC_DS_query3
sh run-TPC-DS.sh

5 Shark-version

Project,Orderby

To prepare
Generate data
1. tar zxvf Marginal_ones_as_representatives.tar.gz
2.cd /Marginal_ones_as_representatives/BigDataGeneratorSuite/Table_datagen/e-com
3. java -XX:NewRatio=1 -jar pdgf.jar -l demo-schema.xml -l demo-generation.xml -c -s ‑ sf <parameter>
Upload the text files in Marginal_ones_as_representatives/BigDataGeneratorSuite/Table_datagen/e-com /output/ to HDFS and make sure these files in different pathes.

Create tables
1. cd Marginal_ones_as_representatives/Shark_project_orderby
2. shark
create external table bigdatabench_dw_item(item_id int,order_id int,goods_id int,goods_number double,goods_price double,goods_amount double) ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' STORED AS TEXTFILE LOCATION ‘path to OS_ODER_ITEM.txt’;
create external table bigdatabench_dw_order(order_id int,buyer_id int,create_date string) ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' STORED AS TEXTFILE LOCATION ‘path to OS_ORDER.txt’;
create table item_temp as select ORDER_ID from bigdatabench_dw_item;

To run
cd Marginal_ones_as_representatives/Shark_project_orderby
edit free_m.sh to make sure it runs correctly.
shark –f projection.sql
shark –f orderby.sql

6 Impala-version

Orderby

To prepare
Generate data
1. tar zxvf Marginal_ones_as_representatives.tar.gz

2.cd /Marginal_ones_as_representatives/BigDataGeneratorSuite/Table_datagen/e-com
3. java -XX:NewRatio=1 -jar pdgf.jar -l demo-schema.xml -l demo-generation.xml -c -s – sf <parameter>
Upload the text files in Marginal_ones_as_representatives/BigDataGeneratorSuite/Table_datagen/e-com /output/ to HDFS and make sure these files in different pathes

Create tables
1. cd $HIVE_HOME/bin
2. ./hive
create external table bigdatabench_dw_item(item_id int,order_id int,goods_id int,goods_number double,goods_price double,goods_amount double) ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' STORED AS TEXTFILE LOCATION ‘path to OS_ODER_ITEM.txt’;
create external table bigdatabench_dw_order(order_id int,buyer_id int,create_date string) ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' STORED AS TEXTFILE LOCATION ‘path to OS_ORDER.txt’;
create table item_temp as select ORDER_ID from bigdatabench_dw_item;

To run
cd Marginal_ones_as_representatives/ Impala_Orderby
edit free_m.sh and impala-restart.sh to make sure them run correctly.
sh runMicroBenchmark.sh

SelectQuery

AMPLab workloads

To prepare
Generate data
1. tar zxvf Marginal_ones_as_representatives.tar.gz
2.cd /Marginal_ones_as_representatives/BigDataGeneratorSuite/Table_datagen/e-com
3. java -XX:NewRatio=1 -jar pdgf.jar -l demo-schema.xml -l demo-generation.xml -c -s – sf <parameter>
Upload the text files in Marginal_ones_as_representatives/BigDataGeneratorSuite/Table_datagen/e-com /output/ to HDFS and make sure these files in different pathes

Create tables
1. cd $HIVE_HOME
2. ./hive
create external table bigdatabench_dw_item(item_id int,order_id int,goods_id int,goods_number double,goods_price double,goods_amount double) ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' STORED AS TEXTFILE LOCATION ‘path to OS_ODER_ITEM.txt’;

create external table bigdatabench_dw_order(order_id int,buyer_id int,create_date string) ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' STORED AS TEXTFILE LOCATION ‘path to OS_ORDER.txt’;
create table item_temp as select ORDER_ID from bigdatabench_dw_item;

To run

cd Marginal_ones_as_representatives/Impala_SelectQuery
edit free_m.sh and impala-restart.sh to make sure them run correctly.
sh runQuery.sh