Technical Report Of PPA & BPA in Batched Streaming System

We implement PPA & BPA in Spark Streaming-1.5.0.The detail of implementation are as follows.It includes 4 parts:

1.How to build a Spark cluster.

This part introduces steps to build a Spark cluster.In this part ,we will deploy Hadoop cluster for using HDFS.

2.How to run PPA & BPA in Batched Streaming System.

This part introduces steps to run PPA & BPA in Batched Streaming System.

3.How to build PPA & BPA in Batched Streaming System.

This part introduces steps to build PPA & BPA in Batched Streaming System from source code .

4.How to add PPA & BPA on a newest Spark cluster.

This part introduces steps to add PPA & BPA on a newest Spark cluster.For task scheduling algorithm,unlike Hadoop , Spark does not provide a programming interface.That means we have to modify it's source code and rebuild Spark if we want to add a new task scheduling algorithm for Spark.

Basic Environment Description	
OS:	Ubuntu 14.04
JAVA version:	Jdk 1.7
Hadoop version :	2.6.0
Scala version:	2.10
Spark version:	1.5.0

Part 1 : Build a Spark cluster

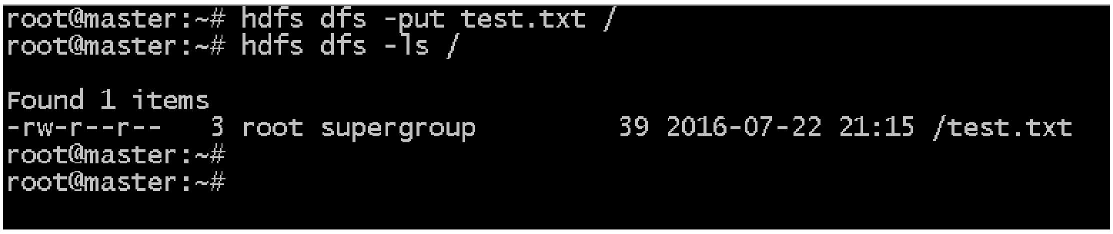
1) Above all, we need at least three machines:one master and two slaves.

2) Configure SSH to login each slave without password on master.

- ssh-keygen -t dsa -P " -f ~/.ssh/id_dsa
- cat ~/.ssh/id_dsa.pub >>~/.ssh/authorized_keys
- scp ~/.ssh/authorized_keys each slave:~/.ssh/

3)Download Hadoop 2.6.0 (http://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-2.6.0/hadoop-2.6.0.tar.gz).And install Hadoop.

- Change file core-site.xml .Add properties:fs.default.name and hadoop.tmp.dir .
- Change file hdfs-site.xml .Add properties:dfs.namenode.secondary.http-address , dfs.namenode.name.dir , dfs.datanode.data.dir, dfs.replication and dfs.webhdfs.enabled.
- Add slaves' hostname to file slaves.
- Execute hadoop namenode -format and start-dfs.sh .
- Execute hdfs -put somefile / and hdfs dfs -ls / to make sure it's in there.As the follows in Figure 1.



```
root@master:~# hdfs dfs -put test.txt /
root@master:~# hdfs dfs -ls /

Found 1 items
-rw-r--r--   3 root supergroup         39 2016-07-22 21:15 /test.txt
root@master:~#
root@master:~#
```

Figure 1:Test for HDFS

4)Download Spark 1.5.0 (http://spark.apache.org/downloads.html).And install Spark.

- Add slaves' hostname to file slaves.
- Change file spark-env.sh .Add the follows contents:

```
export SCALC_HOME=XXX
export JAVA_HOME=XXX
export SPARK_LOCAL_DIRS=XXX
export SPARK_MASTER_IP= XXX
export SPARK_MASTER_PORT=XXX
export SPARK_MASTER_WEBUI_PORT=XXX
export SPARK_WORKER_PORT=XXX
export SPARK_WORKER_MEMORY=XXX Such as 4G
export SPARK_WORKER_CORES=The slave's cpu core
```

- Execute $SPARK_HOME/sbin/start-all.sh .

Part 2 : Run PPA & BPA in Batched Streaming System

Above all, we need replace $SPARK_HOME/lib/spark-assembly-1.5.0-hadoop2.6.0.jar with attachment/spark-assembly-1.5.0-hadoop2.6.0.jar. The system contains two parts : Prediction Module and Scheduler Module.

- The Prediction Module is the foundation of Scheduler Module.It is used to analyse the running log of a Spark Streaming application,and it will build a file named "ApplicationName.obj".
- The Scheduler Module is used for task scheduling according to the file created by Prediction Module.

The specific steps are as follows:

1) Configure and start Spark HistoryServer.

- Execute hdfs dfs -mkdir dirname to make a directory in HDFS. The Spark HistoryServer will save the running log of all Spark Streaming applications in the directory.
- Change config file $SPARK_HOME/conf/spark-defaults.conf . Add the following contents:

```
spark.eventLog.enabled   true
spark.eventLog.dir       HDFS directory
spark.eventLog.compress true
```

- Change config file $SPARK_HOME/conf/spark-env.conf . Add the following contents:

```
export SPARK_HISTORY_OPTS="-Dspark.history.ui.port=PORT
                          -Dspark.history.fs.logDirectory=HDFS directory"
```

- Execute ./$SPARK_HOME/sbin/start-history-server.sh

2) Submit a streaming application . We provide three benchmarks as illustrated in Table 1 and Table 2 if readers have no streaming application.

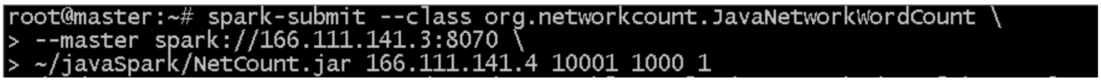
Table 1: Three benchmarks

Application	ClassName	Parameters	Description
Grep	Org.networkcount. JavaGrep	<hostname> <port> <interval> <RegExp> [Socket_Connection_num]	Finds the number of input strings matching a pattern
JavaTopK	org.networkcount JavaTopK	<hostname> <port> <interval> <topnum> [Socket_Connection_num]	Finds the k most frequent words
WordCount	org.networkcount. JavaNetworkWordCount	<hostname> <port> <interval> [Socket_Connection_num]	Counts the number of word

Table 2: Parameters Setting For Benchmarks

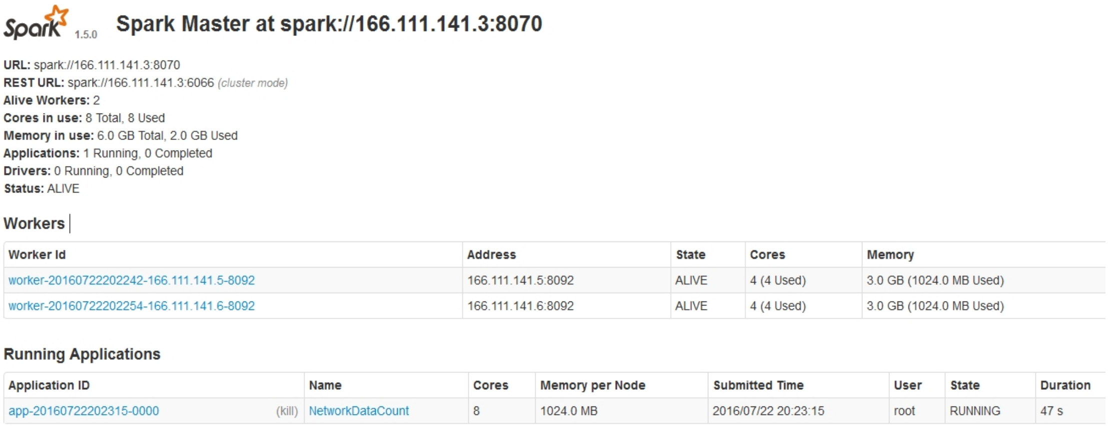
Parameters	Meaning	value
hostname	Socket Server's ip	ip
topnum	The value of k	positive number,default 1
port	The socket port of Socket server	positive number
Socket_Connection_num	The number of connection	positive number
RegExp	The pattern uesd for filtering words	pattern

As illustrated in figure 2 to figure 4 ,we show the process of running WordCount .



```
root@master:~# spark-submit --class org.networkcount.JavaNetworkWordCount \
> --master spark://166.111.141.3:8070 \
> ~/javaSpark/NetCount.jar 166.111.141.4 10001 1000 1
```

Figure 2:Submit a Spark Streaming application



Spark Master at spark://166.111.141.3:8070

URL: spark://166.111.141.3:8070
REST URL: spark://166.111.141.3:6066 *(cluster mode)*
Alive Workers: 2
Cores in use: 8 Total, 8 Used
Memory in use: 6.0 GB Total, 2.0 GB Used
Applications: 1 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

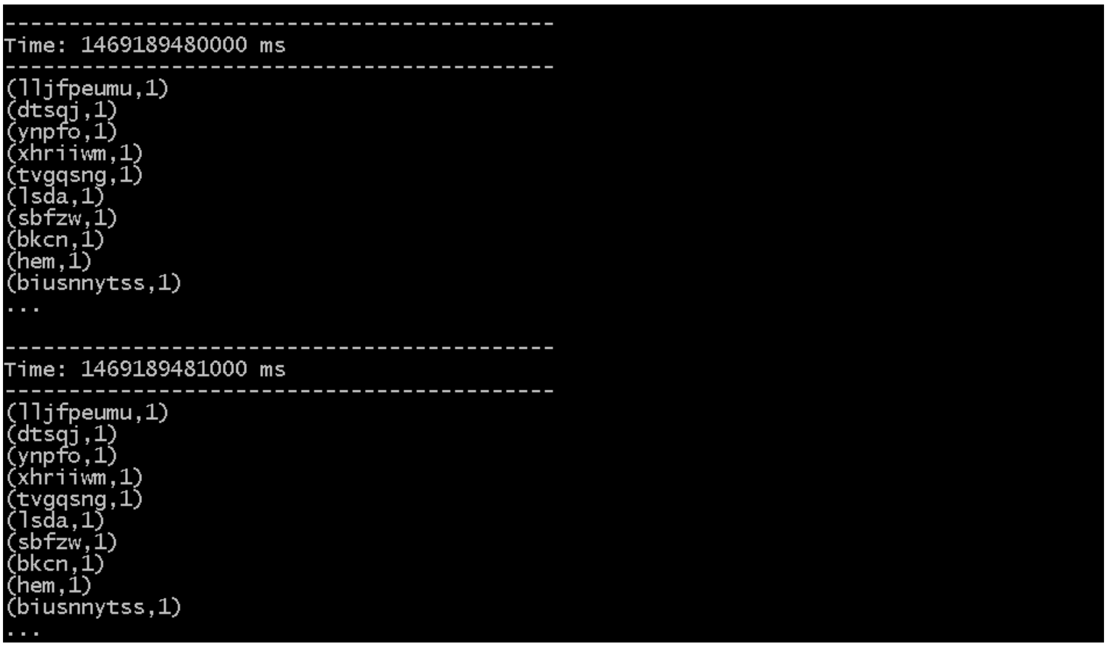
Workers

Worker Id	Address	State	Cores	Memory
worker-20160722202242-166.111.141.5-8092	166.111.141.5:8092	ALIVE	4 (4 Used)	3.0 GB (1024.0 MB Used)
worker-20160722202254-166.111.141.6-8092	166.111.141.6:8092	ALIVE	4 (4 Used)	3.0 GB (1024.0 MB Used)

Running Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20160722202315-0000 (kill)	NetworkDataCount	8	1024.0 MB	2016/07/22 20:23:15	root	RUNNING	47 s

Figure 3: Spark WEBUI



```
-------------------------------------------
Time: 1469189480000 ms
-------------------------------------------
(lljfpeumu,1)
(dtsqj,1)
(ynpfo,1)
(xhriiwm,1)
(tvgqsng,1)
(lsda,1)
(sbfzw,1)
(bkcn,1)
(hem,1)
(biusnnytss,1)
...

-------------------------------------------
Time: 1469189481000 ms
-------------------------------------------
(lljfpeumu,1)
(dtsqj,1)
(ynpfo,1)
(xhriiwm,1)
(tvgqsng,1)
(lsda,1)
(sbfzw,1)
(bkcn,1)
(hem,1)
(biusnnytss,1)
...
```

Figure 4:Running results of WordCount

3) Change config file $SPARK_HOME/conf/spark-defaults.conf . Add the following contents:

   spark.customize.scheduler.filedirpath   The directory of obj file

   spark.customize.setcustomize   True means enabled PPA &BPA   in Batched Streaming

   spark.customize.scheduler.mode We provide two modes: BPA and PPA

4) Restart Spark cluster and resubmit the streaming application. Run command:

   $SPARK_HOME/sbin/stop-all.sh

   $SPARK_HOME/sbin/start-all.sh

Part 3 : Build PPA & BPA in Batched Streaming System

1) Download the Spark 1.5.0 Source Code (http://spark.apache.org/downloads.html).

2) Unzip attachment/SystemSource.zip .

3) Unzip the core.tar.gz.

4) Replace folder named core in spark source with the folder unzipped in step 3 .

5) Run command:

   $SPARK_SOURCE_HOME/build/mvn clean

6) Run command:

```
$SPARK_SOURCE_HOME/make-distribution.sh --name NewSparkName --tgz -Phadoop-2.6 -Pyarn
```

Part 4:Add PPA & BPA on a newest Spark cluster

1) Add core/src/main/java/org/apache/spark/prediction to the new Spark source folder. This directory is a bridge between Prediction Module and Scheduler Module.

2) Add core/src/main/scala/org/apache/spark\prediction to the new Spark source folder.

3) We need alter core/src/main/scala/org/apache/spark/scheduler/DAGScheduler.scala . Execute Prediction.stagePrediction(stage) in the function named submitMissingTasks to combine specified stage and prediction results in OBJ file.

4)Add BPA and PPA algorithm to core/.../scala/.../scheduler/TaskSchedulerImpl.scala .After that, execute Prediction.addtaskId(taskId,stageId,index) in TaskSetManager.scala . This one code will forecast the requirement of CPU Resource for one task in a stage.

5) Modify core/.../scala/.../scheduler/cluster/CoarseGrainedSchedulerBackend.scala to fit centesimal CPU resource requirement of tasks .

6)Complete Part 3.

Note:In this part we only tell reader some files need to be altered and why those file must be modified.