How to Use BigDataBench 4.0

Jianfeng Zhan, Chen Zheng, and Wanling Gao
http://prof.ict.ac.cn

ICT, Chinese Academy of Sciences

ASPLOS 2018, Williamsburg, VA, USA
General Steps to Use BigDataBench

- Current release
  - Version 4.0 on [http://prof.ict.ac.cn](http://prof.ict.ac.cn)

- General steps to run the benchmarks
  - Prepare the package of BigDataBench
  - Prepare the environments of the selected software stack
  - Generate data sets as you need
    - *You can find a genDate* or a *prepare* shell script in each directory of the benchmarks
  - Run the scripts or commands (User Manual!)
Directory Structure

- Root directory
  - Micro Benchmark
    - AI (TensorFlow, Caffe2)
    - Offline analytics (Hadoop, Spark, Flink, MPI)
    - Graph analytics (Hadoop, Spark, Flink, GraphLab, MPI)
  - Component Benchmark
    - NoSQL (Hbase, MongoDB)
    - Online service (Xapian)
    - Data warehouse (Hive, SparkSQL, Impala)
    - Streaming (Spark streaming, JStorm)
  - Data Generator (BDGS)
BDGS - Text

- **Text_datagen**
  - Wikipedia generator - 3 trained models
    - lda_wiki1w, wiki_1w5, wiki_noSW_90_Sampling
  - Amazon movie review generator – 2 models
    - amazonMR1, AMR1_noSW_95_Sampling
  - Use “gen_text_data.sh”

```
Text_datagen/gen_text_data.sh $MODEL_NAME $FILE_NUM $LINES_PER_FILE $WORDS_PER_LINE $Out_Dir
```

- **Wiki example:**
  - e.g. lda_wiki1w
  - e.g. 10
  - e.g. 100
  - e.g. 10000

- **Amazon example:**
  - e.g. amazonMR1
  - e.g. 10
  - e.g. 100
  - e.g. 10000
BDGS - Graph

Graph_datagen

- Kronecker Model
  - Weighted graph
  - Un-weighted graph

Amazon un-weighted graph:
  To Run: Graph_datagen/gen_weighted_graph.sh

Amazon weighted graph:
  To Run: Graph_datagen/gen_kronecker_graph -o:amazon_gen_16.txt -m:"0.9532 0.5502; 0.4439 0.2511" -i:16

Facebook graph:
  To Run: Graph_datagen/gen_kronecker_graph -o:facebook_g_16.txt -m:"0.9999 0.5887; 0.6254 0.3676" -i:16

Google graph:
  To Run: Graph_datagen/gen_kronecker_graph -o:google_g_16.txt -m:"0.8305 0.5573; 0.4638 0.3021" -i:16

- e.g. kronecker model parameter
- Vertex: $2^{16}$
BDGS - Table

- Table_datagen
  - E-commerce data generation
    - PDGF: uses XML configuration files for data description and distribution
      `Table_datagen/e-com/generate_table.sh`
  - Personal Resume generation

```
Table_datagen/personal_generator/gen_resume.sh "NUM_RESUME" "NUMBER_OF_FILES" "OUT_DATA_DIR"
```
# Micro Benchmark

- Offline analytics & Graph analytics
- Streaming

<table>
<thead>
<tr>
<th>Micro Benchmark</th>
<th>Involved Dwarf</th>
<th>Application Domain</th>
<th>Workload Type</th>
<th>Data Set</th>
<th>Software Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sort</td>
<td>Sort</td>
<td>SE, SN, EC, MP, BI</td>
<td>Offline analytics</td>
<td>Wikipedia entries</td>
<td>Hadoop, Spark, Flink, MPI</td>
</tr>
<tr>
<td>Grep</td>
<td>Set</td>
<td></td>
<td>Offline analytics</td>
<td>Wikipedia entries</td>
<td>Hadoop, Spark, Flink, MPI</td>
</tr>
<tr>
<td>WordCount</td>
<td>Basic statistics</td>
<td></td>
<td>Streaming</td>
<td>Random Generate</td>
<td>Spark streaming</td>
</tr>
<tr>
<td>MD5</td>
<td>Logic</td>
<td></td>
<td>Offline analytics</td>
<td>Wikipedia entries</td>
<td>Hadoop, Spark, Flink, MPI</td>
</tr>
<tr>
<td>Connected Component</td>
<td>Graph</td>
<td>SN</td>
<td>Graph analytics</td>
<td>Facebook social network</td>
<td>Hadoop, Spark, Flink, GraphLab, MPI</td>
</tr>
<tr>
<td>RandSample</td>
<td>Sampling</td>
<td>SE, MP, BI</td>
<td>Offline analytics</td>
<td>Wikipedia entries</td>
<td>Hadoop, Spark, MPI</td>
</tr>
<tr>
<td>FFT</td>
<td>Transform</td>
<td>MP</td>
<td>Offline analytics</td>
<td>Two-dimensional matrix</td>
<td>Hadoop, Spark, MPI</td>
</tr>
<tr>
<td>Matrix Multiply</td>
<td>Matrix</td>
<td>SE, SN, EC, MP, BI</td>
<td>Offline analytics</td>
<td>Two-dimensional matrix</td>
<td>Hadoop, Spark, MPI</td>
</tr>
</tbody>
</table>
Offline Analytics - RandSample

■ Target: run RandSample microbenchmark

■ General steps:

```
[root@hw125 tpds2017]# $HADOOP_HOME/bin/hadoop jar dwarf-hadoop/sample-operations/RandSample/out/artifacts/ RandSampleFileSystem\n1.2
18/03/22 16:30:16 INFO client.RMProxy: Connecting to ResourceManager at hw125:172.18.11.125:0032
18/03/22 16:30:16 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. In
ted this.
18/03/22 16:30:17 INFO input.FileInputFormat: Total input paths to process : 200
18/03/22 16:30:17 INFO mapreduce.JobSubmitter: number of splits:800
18/03/22 16:30:17 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1521170223299_0040
18/03/22 16:30:17 INFO impl.YarnClientImpl: Submitted application application_1521170223299_0040
18/03/22 16:30:17 INFO mapreduce.Job: The url to track the job: http://hw125:8088/proxy/application_1521170223299_0040
18/03/22 16:30:17 INFO mapreduce.Job: Running job: job_1521170223299_0040
18/03/22 16:30:21 INFO mapreduce.Job: Job job_1521170223299_0040 running in uber mode : false
18/03/22 16:30:21 INFO mapreduce.Job: map 0% reduce 0%
```

- hadoop jar RandSample.jar RandSample <input> <output> <sample_ratio>
Offline Analytics – FFT example

- **Target:** run “FFT” micro benchmark using hadoop
- **General steps:**
  - Prepare Hadoop environment
  - Prepare matrix data
    - cd /BigDataBench_V4.0_Hadoop/MicroBenchmark/OfflineAnalytics/FFT
    - sh genData_FFT.sh
      - sh generate -matrix <mat_row> <mat_col> <sparsity>
  - Run FFT:
    - sh run_FFT.sh
      - hadoop jar fft.jar org.fft.fft <input file> <outputfile1> <outputfile2>
        - <log2_col>, <log2_co> (auto-generated by run_FFT.sh)
Streaming – Grep example

- **Target:** run grep benchmark using Spark streaming

- **General steps:**
  - Prepare Spark streaming environment
  - `cd /BigDataBench_V4.0_Streaming/MicroBenchmark/Streaming/Grep`
  - `./run-sparkstreaming-grep.sh`

```bash
# Usage: Grep <numStreams> <host> <port> <batchMillis>
# * <numStream> is the number rawNetworkStreams, which should be same as number
# * of work nodes in the cluster
# * <host> is "localhost".
# * <port> is the port on which RawTextSender is running in the worker nodes.
# * <batchMillis> is the Spark Streaming batch duration in milliseconds.
```
## Micro Benchmark

**AI**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Transform</th>
<th>Data Format</th>
<th>AI</th>
<th>Models</th>
<th>Frameworks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convolution</td>
<td>Transform</td>
<td>SN, EC, MP, BI</td>
<td>AI</td>
<td>Cifar, ImageNet</td>
<td>TensorFlow, Caffe</td>
</tr>
<tr>
<td>Fully Connected</td>
<td>Matrix</td>
<td>SN, EC, MP, BI</td>
<td>AI</td>
<td>Cifar, ImageNet</td>
<td>TensorFlow, Caffe</td>
</tr>
<tr>
<td>Relu</td>
<td>Logic</td>
<td>SN, EC, MP, BI</td>
<td>AI</td>
<td>Cifar, ImageNet</td>
<td>TensorFlow, Caffe</td>
</tr>
<tr>
<td>Sigmoid</td>
<td>Matrix</td>
<td>SN, EC, MP, BI</td>
<td>AI</td>
<td>Cifar, ImageNet</td>
<td>TensorFlow, Caffe</td>
</tr>
<tr>
<td>Tanh</td>
<td>Matrix</td>
<td>SN, EC, MP, BI</td>
<td>AI</td>
<td>Cifar, ImageNet</td>
<td>TensorFlow, Caffe</td>
</tr>
<tr>
<td>MaxPooling</td>
<td>Sampling</td>
<td>SN, EC, MP, BI</td>
<td>AI</td>
<td>Cifar, ImageNet</td>
<td>TensorFlow, Caffe</td>
</tr>
<tr>
<td>AvgPooling</td>
<td>Sampling</td>
<td>SN, EC, MP, BI</td>
<td>AI</td>
<td>Cifar, ImageNet</td>
<td>TensorFlow, Caffe</td>
</tr>
<tr>
<td>BatchNorm [37]</td>
<td>Basic Statistics</td>
<td>SN, EC, MP, BI</td>
<td>AI</td>
<td>Cifar, ImageNet</td>
<td>TensorFlow, Caffe</td>
</tr>
<tr>
<td>Dropout [38]</td>
<td>Sampling</td>
<td>SN, EC, MP, BI</td>
<td>AI</td>
<td>Cifar, ImageNet</td>
<td>TensorFlow, Caffe</td>
</tr>
</tbody>
</table>
AI – Conv2d example

- Target: run conv2d micro benchmark using TensorFlow

- General steps:
  - Prepare TensorFlow environment
  - Prepare image data
  - Config image directory in conv2d.py
  - python conv2d.py
## Micro Benchmark

- **NoSQL**

<table>
<thead>
<tr>
<th></th>
<th>Read</th>
<th>Write</th>
<th>Scan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Set</td>
<td>Set</td>
<td>Set</td>
</tr>
<tr>
<td>Type</td>
<td>SE, SN, EC</td>
<td>SE, SN, EC</td>
<td>SE, SN, EC</td>
</tr>
<tr>
<td>System</td>
<td>NoSQL</td>
<td>NoSQL</td>
<td>NoSQL</td>
</tr>
<tr>
<td>Feature</td>
<td>ProfSearch resumes</td>
<td>ProfSearch resumes</td>
<td>ProfSearch resumes</td>
</tr>
<tr>
<td>Database</td>
<td>HBase, MongoDB</td>
<td>HBase, MongoDB</td>
<td>HBase, MongoDB</td>
</tr>
</tbody>
</table>
NoSQL – Write example

- Target: run “write” operations using HBase

- General steps:
  - Prepare HBase according to the office guide
    - sh /hbase-0.94.5/bin/hbase shell
    - create 'usertable','f1','f2','f3'
  - Prepare YCSB as the workload generator
    - YCSB is in the directory of BasicDatastoreOperactions/ycsb-0.1.4
  - Run YCSB commands like this:
    - sh bin/ycsb load hbase -P workloads/workloadc -p threads=<thread-numbers> -p columnfamily=<family> -p recordcount=<recordcount-value> -p hosts=<hosOp> -s>load.dat
Component Benchmark

<table>
<thead>
<tr>
<th>AI</th>
<th>Component</th>
<th>Benchmark</th>
<th>System</th>
<th>Data</th>
<th>Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alexnet</td>
<td>SN, MP, BI</td>
<td>AI</td>
<td>Cifar, ImageNet</td>
<td>TensorFlow, pyTorch</td>
<td>Caffe,</td>
</tr>
<tr>
<td>Googlenet</td>
<td>Matrix, Transform, Sampling, Logic, Basic statistics</td>
<td>SN, MP, BI</td>
<td>AI</td>
<td>Cifar, ImageNet</td>
<td>TensorFlow, pyTorch</td>
</tr>
<tr>
<td>Resnet</td>
<td>SN, MP, BI</td>
<td>AI</td>
<td>Cifar, ImageNet</td>
<td>TensorFlow, pyTorch</td>
<td>Caffe,</td>
</tr>
<tr>
<td>Inception Resnet V2</td>
<td>SN, MP, BI</td>
<td>AI</td>
<td>Cifar, ImageNet</td>
<td>TensorFlow, pyTorch</td>
<td>Caffe,</td>
</tr>
<tr>
<td>VGG16</td>
<td>SN, MP, BI</td>
<td>AI</td>
<td>Cifar, ImageNet</td>
<td>TensorFlow, pyTorch</td>
<td>Caffe,</td>
</tr>
<tr>
<td>DCGAN</td>
<td>SN, MP, BI</td>
<td>AI</td>
<td>LSUN</td>
<td>TensorFlow, pyTorch</td>
<td>Caffe,</td>
</tr>
<tr>
<td>WGAN</td>
<td>SN, MP, BI</td>
<td>AI</td>
<td>LSUN</td>
<td>TensorFlow, pyTorch</td>
<td>Caffe,</td>
</tr>
<tr>
<td>GAN</td>
<td>Matrix, Sampling, Logic, Basic statistics</td>
<td>SN, MP, BI</td>
<td>AI</td>
<td>LSUN</td>
<td>TensorFlow, pyTorch</td>
</tr>
<tr>
<td>Seq2Seq</td>
<td>SE, EC, BI</td>
<td>AI</td>
<td>TED Talks</td>
<td>TensorFlow, pyTorch</td>
<td>Caffe,</td>
</tr>
<tr>
<td>Word2vec</td>
<td>Matrix, Basic statistics, Logic</td>
<td>SE, SN, EC</td>
<td>AI</td>
<td>Wikipedia entries, Sogou data</td>
<td>TensorFlow, pyTorch</td>
</tr>
</tbody>
</table>
AI – Alexnet Example

- Target: run “Alexnet” micro benchmark using Tensorflow
- General steps:
  - Prepare Tensorflow environment
  - Run Alexnet:
    - cd /BigDataBench_V4.0_Tensorflow/ComponentBenchmark/AI/Alexnet
    - python alexnet_cifar10.py
    - Choosing CPU or GPU environment
Component Benchmark

- Offline analytics & Graph analytics
- Streaming

<table>
<thead>
<tr>
<th>Component</th>
<th>Input Data</th>
<th>Output Data</th>
<th>Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>PageRank</td>
<td>Matrix, Sort, Basic statistics, Graph</td>
<td>SE</td>
<td>Graph analytics</td>
</tr>
<tr>
<td>Index</td>
<td>Logic, Sort, Basic statistics, Set</td>
<td>SE</td>
<td>Offline analytics</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Google web graph</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Hadoop, Spark, Flink, GraphLab, MPI</td>
</tr>
<tr>
<td>Rolling top words</td>
<td>Sort, Basic statistics</td>
<td>SN</td>
<td>Streaming</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Random generate</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Spark streaming, JStorm</td>
</tr>
<tr>
<td>Kmeans</td>
<td>Matrix, Sort, Basic statistics</td>
<td>SE, SN, EC, MP, BI</td>
<td>Offline analytics</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Facebook social network</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Hadoop, Spark, Flink, MPI</td>
</tr>
<tr>
<td>Collaborative Filtering</td>
<td>Graph, Matrix</td>
<td>EC</td>
<td>Offline analytics</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Amazon movie review</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Hadoop, Spark</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>Basic statistics, Sort</td>
<td>SE, SN, EC</td>
<td>Offline analytics</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Amazon movie review</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Hadoop, Spark, Flink, MPI</td>
</tr>
<tr>
<td>SIFT</td>
<td>Matrix, Sampling, Transform, Sort</td>
<td>MP</td>
<td>Offline analytics</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ImageNet</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Hadoop, Spark, MPI</td>
</tr>
<tr>
<td>LDA</td>
<td>Matrix, Graph, Sampling</td>
<td>SE</td>
<td>Offline analytics</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Wikipedia entries</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Hadoop, Spark, MPI</td>
</tr>
</tbody>
</table>
Offline Analytics – SIFT example

- **Target:** run “SIFT” component benchmark using hadoop
- **General steps:**
  - Prepare Hadoop environment
  - Prepare SIFT data
    - cd /BigDataBench_V4.0_Hadoop/ComponentBenchmark/OfflineAnalytics/SIFT
  - Put the image data under SIFT directory

```
[root@hul25 hpc2018]# hadoop jar /home/gwu/hipi/tools/sift/build/libs/sift.jar /img-data/image09.bib /image-out/siftout
10/03/22 15:52:49 INFO client.RMProxy: Connecting to ResourceManager at hul25/172.18.11.125:8032
10/03/22 15:52:50 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
10/03/22 15:52:50 INFO input.FileInputFormat: Total input paths to process : 1
  Spawned 60 map tasks
10/03/22 15:52:50 INFO mapreduce.JobSubmitter: number of splits:60
10/03/22 15:52:51 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1521170223290_0039
10/03/22 15:52:51 INFO impl.YarnClientImpl: Submitted application application_1521170223290_0039
10/03/22 15:52:51 INFO mapreduce.Job: Running job: job_1521170223290_0039
10/03/22 15:52:56 INFO mapreduce.Job: Job job_1521170223290_0039 running in user mode : false
10/03/22 15:52:56 INFO mapreduce.Job: map 0% reduce 0%
```

```
hadoop jar sift.jar <out.hib> <outsif>
<out.hib>: genData_SIFT.sh generate data
<outsif>: the result to save path
```
Streaming – Kmeans example

- Target: run kmeans benchmark using Spark streaming

- General steps:
  - Prepare Spark streaming environment
  - cd /BigDataBench_V4.0_Streaming/ComponentBenchmark/Streaming/Kmeans
  - ./run-sparkstreaming-kmeans.sh

******************************************************************************
#StreamingKMeans <trainingDir> <testDir> <batchDuration> <numClusters> <numDimensions>
#: The directory used to save the training file
#: The directory used to save the test file
#: Batch's time interval
#: The number of clusters
#: The dimension of vector (which is number of columns), it needs to be same as "columnNum" in
#: KMeansTestDataGenerator.sh and KMeansTrainDataGenerator.sh
******************************************************************************
Graph Analytics – PageRank

- **Target:** run “PageRank” component benchmark using hadoop
- **General steps:**
  
  ```
  # ComponentBenchmark/PageRank
  sh run_PageRank.sh
  ```

  ```java
  /home/s002/BigDataBench_V4.0_Hadoop/ComponentBenchmark/GraphAnalytics/PageRank/run_PageRank.sh
  ```

  ```
  - [Graph]
  - [Analytics]
  - [PageRank]
  ```
Online Service – Xapian (cont’)

- Target: run searching using Xapian
- General steps:
  - 3) Online searching
    - Run xapian/run_networked.sh

```
[root@hw114 xapian]# ./run_networked.sh
TBENCH_SERVER = TBENCH_CLIENT_THREADS = Client left, removing All clients exited. Server finishing
```
Online Service – Xapian

- Target: run searching using Xapian
- General steps:
  1) Install Xapian according to user manual
     - ./build.sh to install harness (gcc version > 4.8)
     - xapian/build.sh to install xapian

```
[root@hw114 xapian]# ./build.sh
```

```
g++ -o xapian_integrated main.o server.o client.o ..../harness/client.o ..../harness/tbench_server_integrated.o `./xapian-core-1.2.13/install/bin/xapian-config --libs` -lpthread -lrt
g++ -o xapian_networked_server main.o server.o ..../harness/tbench_server_networked.o `./xapian-core-1.2.13/install/bin/xapian-config --libs` -lpthread -lrt
g++ -o xapian_networked_client client.o ..../harness/client.o ..../harness/tbench_client_networked.o `./xapian-core-1.2.13/install/bin/xapian-config --libs` -lpthread -lrt
```
Online Service – Xapian (cont’)

- **Target:** run searching using Xapian

- **General steps:**
  - 2) Configuration
    - `vim xapian/run_networked.sh`

```bash
# The server number configuration
NSERVERS=12
# Queries per second
QPS=5000
# Warmup configuration: the query numbers before testing
WARMUPREQS=10
# The total queries
REQUESTS=1000000
```
## Component Benchmark

### Data warehouse

<table>
<thead>
<tr>
<th>Operation</th>
<th>Data Flow</th>
<th>EC</th>
<th>Data Source</th>
<th>Transaction Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrderBy</td>
<td>Set, Sort</td>
<td>EC</td>
<td>Data warehouse</td>
<td>E-commerce</td>
</tr>
<tr>
<td>Aggregation</td>
<td>Set, Basic statistics</td>
<td>EC</td>
<td>Data warehouse</td>
<td>E-commerce</td>
</tr>
<tr>
<td>Project</td>
<td>Set</td>
<td>EC</td>
<td>Data warehouse</td>
<td>E-commerce</td>
</tr>
<tr>
<td>Filter</td>
<td>Set</td>
<td>EC</td>
<td>Data warehouse</td>
<td>E-commerce</td>
</tr>
<tr>
<td>Select</td>
<td>Set</td>
<td>EC</td>
<td>Data warehouse</td>
<td>E-commerce</td>
</tr>
<tr>
<td>Union</td>
<td>Set</td>
<td>EC</td>
<td>Data warehouse</td>
<td>E-commerce</td>
</tr>
</tbody>
</table>
Data Warehouse – Select example

- Target: run “Select” benchmark using hadoop hive
- General steps:
  - Prepare Hadoop and hive environment
  - Run the data generation script

```
root@g02:/home/BigDataBench_V4.0_Hadoop/ComponentBenchmark/Datawarehouse/Select# ./run_Select.sh
Java HotSpot(TM) Server VM warning: ignoring option MaxPermSize=512m; support was removed in 6.0
Java HotSpot(TM) Server VM warning: ignoring option MaxPermSize=512m; support was removed in 6.0
Java HotSpot(TM) Server VM warning: ignoring option MaxPermSize=512m; support was removed in 6.0
Java HotSpot(TM) Server VM warning: ignoring option MaxPermSize=512m; support was removed in 6.0
Java HotSpot(TM) Server VM warning: ignoring option MaxPermSize=512m; support was removed in 6.0
Java HotSpot(TM) Server VM warning: ignoring option MaxPermSize=512m; support was removed in 6.0
Java HotSpot(TM) Server VM warning: ignoring option MaxPermSize=512m; support was removed in 6.0
Java HotSpot(TM) Server VM warning: ignoring option MaxPermSize=512m; support was removed in 6.0
Query ID = root_20180322153226_bbbdb38f6_d227-4cbb-b0a9-edc5e2f5f1
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1521686472463_0007, Tracking URL = http://s002:8088/proxy/application_1521686472463_0007/
Kill Command = /home/w/hadoop-2.7.1/bin/hadoop job -kill job_1521686472463_0007
Hadoop job information for Stage-1: number of mappers: 3; number of reducers: 0
2018-03-22 15:32:35,610 Stage-1 map = 0%, reduce = 0%
2018-03-22 15:32:40,160 Stage-1 map = 17%, reduce = 0%, Cumulative CPU 13.98 sec
2018-03-22 15:32:49,169 Stage-1 map = 33%, reduce = 0%, Cumulative CPU 19.12 sec
2018-03-22 15:33:06,754 Stage-1 map = 50%, reduce = 0%, Cumulative CPU 33.22 sec
2018-03-22 15:33:10,863 Stage-1 map = 67%, reduce = 0%, Cumulative CPU 36.9 sec
2018-03-22 15:33:23,253 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 50.27 sec
```
Conclusion

- Website: http://prof.ict.ac.cn

- Please refer to user manual for more details!