DCBench

A Benchmark suite for Data Center Workloads

May 27th 2013

Revision Sheet

Release No.	**Date**	**Revision Description**
Rev. 1.0	27/05/2013	DCBench v1.0

1.Motivation

In the context of digitalized information explosion, more and more businesses are analyzing massive amount of data– so-called big data – with the goal of converting big data to “big value”. Typical data analysis workloads include business intelligence, machine learning, bio-informatics, and ad hoc analysis. The business potential of the data analysis applications in turn is a driving force behind the design of innovative data center systems, both hardware and software.

Benchmarks, as the foundation of quantitative design approach, are used ubiquitously to evaluate the benefits of new designs and new systems. So we would like to assemble a benchmark suite for emerging data center workloads which we call ***DCBench***.

2. Methodology

A benchmark suite must have a target class of machines and a target class of applications. For this effort, it is data center systems. Such a benchmark suite must meet the following goals.

Representative Workloads

A benchmark suite should include representative workloads in the target class machines. There are many benchmark suites for special fields, such as SPEC CPU for processors, SPEC Web for Web servers. The workloads in those benchmarks are all representative in their own field. In data centers, applications range from simple reporting to deep data mining, and only those workloads can reflect the real performance of the target class of machines.

Diverse Programming Models

In data centers, there are a large amount of programming models, e.g., MapReduce, Dryad, for developers or users to write applications, since there is no one-fit-all solution. Different programming models will have great effects on the performance behaviors. So the benchmark suite should be based on different programming models, which can characterize the target class of machines more comprehensively.

Distributed

The target class of machine is data center. Data center is a large distributed environment, e.g., Google’s data center has about 450000 nodes. In general, most of workloads in data centers are distributed on several nodes. An application running on a single computer cannot represent the applications in real world data centers.

Employ State-of-art Techniques

In data centers, workloads change frequently, which Barroso et al.call workload churns. So the benchmark suite should include recently used and emerging techniques in different domains.

In order to find workloads which meet all the requirements listed in Above, we firstly decide and rank main application domains according to a widely acceptable metric—the number of pageviews and daily visitors, and then single out the main applications from the most important application domains. We investigate the top sites listed in Alexa (http://www.alexa.com/topsites), of which the rank of sites is calculated using a combination of average daily visitors and page views. We classified the top 20 sites into 5 categories including search engine, social network, electronic commerce, media streaming and others. Figure 1 shows the categories and their respective share. To keep concise and simple, we focus on the top three application domains: search engine, social networks and electronic commerce.

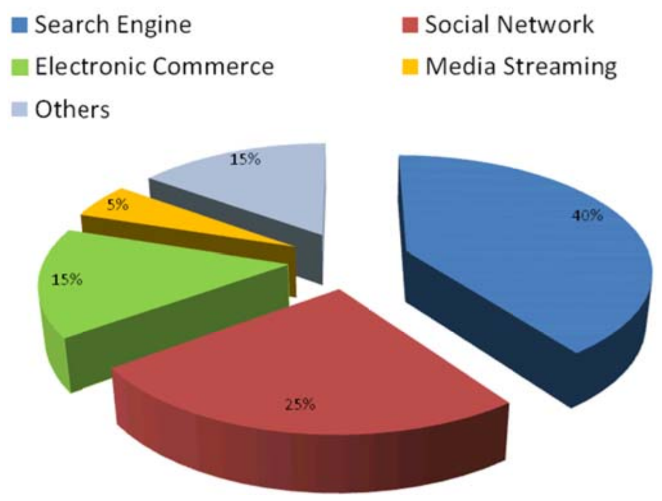

Figure 1 Top Site on the Web

3 Licenses

DCBench is available for researchers interested in pursuing research in the field of data centers. Software components of DCBench are all available as open-source softwares and governed by their own licensing terms. Researchers intending to use DCBench are required to fully understand and abide by the licensing terms of the various components. For now, DCBench is open-source under the Aapche License, Version 2.0. Please use all files in compliance with the License.

4. Representative workloads

We choose the most popular applications in those three application domains. Table 1 shows application scenarios of each workload, indicating most of our chosen workloads are intersections among three domains.

Table 1 Workloads in DCBench

Category	Workloads	Programming model	language	source
Basic operation	Sort	MapReduce	Java	Hadoop
	Wordcount	MapReduce	Java	Hadoop
	Grep	MapReduce	Java	Hadoop
Classification	Naïve Bayes	MapReduce	Java	Mahout
	Support Vector Machine	MapReduce	Java	Implemented by ourselves
Cluster	K-means	MapReduce	Java	Mahout
		MPI	C++	IBM PML
	Fuzzy k-means	MapReduce	Java	Mahout
		MPI	C++	IBM PML
Recommendation	Item based Collaborative Filtering	MapReduce	Java	Mahout
Association rule mining	Frequent pattern growth	MapReduce	Java	Mahout
Segmentation	Hidden Markov model	MapReduce	Java	Implemented by ourselves
Warehouse operation	Database operations	MapReduce	Java	Hive-bench
Feature reduction	Principal Component Analysis	MPI	C++	IBM PML
	Kernel Principal Component Analysis	MPI	C++	IBM PML
Vector calculate	Paper similarity analysis	All-Pairs	C&C++	Implemented by ourselves
Graph mining	Breadth-first search	MPI	C++	Graph500
	Pagerank	MapReduce	Java	Mahout

Service	**Search engine**[1]	**C/S**	**Java**	**Implemented by ourselves**
	Auction	**C/S**	**Java**	**Rubis**
Interactive real-time application	**Media streaming**[2]	**C/S**	**Java**	**Cloudsuite**

The following example gives how the above applications are used in search engine.

Figure 2 is a very simple prototype of Google search engine. Only the Core functionality is show in the figure. In the crawling phase, crawlers will traverse URL list (this is graph mining algorithm) in the URL server. When the web pages are fetched, they should be parsed by indexers. The indexers needs grep text from HTML files and then segment text file into words. After the PangRank step, the inverted index generation and lexicon building phase will count words frequency for web pages and sorter do sort. When the query comes to searcher, it will do word segmentation and send terms to searcher's back end. Back end will find the matched document by vector calculation and return top N items to user after sorting the matched documents. Table 2 lists the representative algorithms used in search engine prototype.

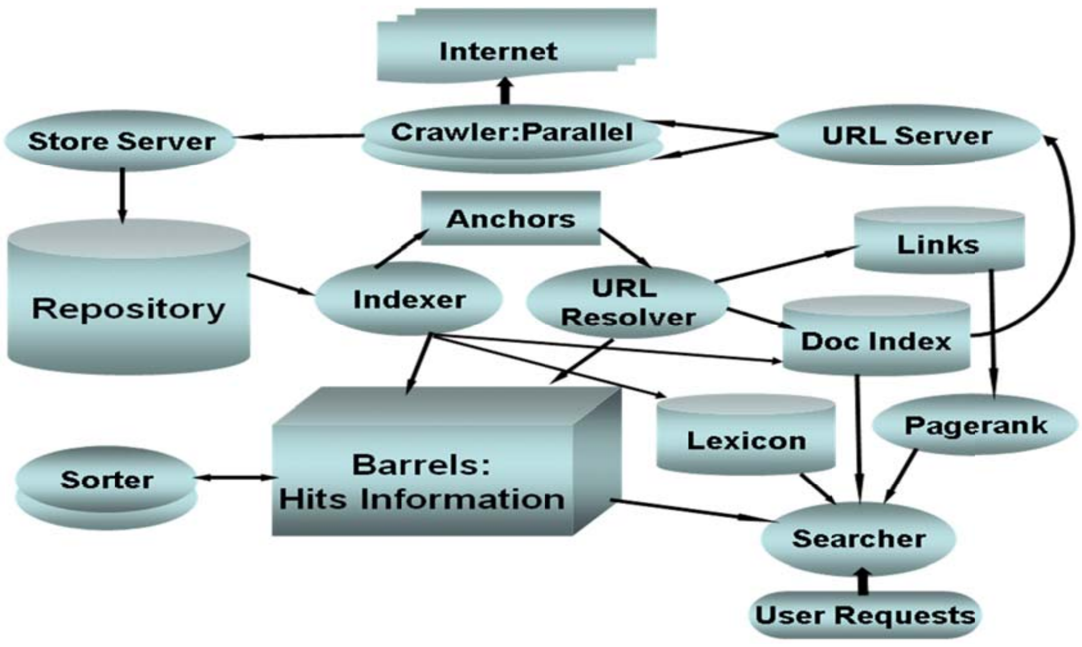

Figure 2 Google search engine prototype

[1] For search We also have a XCP (Xen Cloud Platform) version image. More details in section 5.7

[2] For media streaming we also have a XCP (Xen Cloud Platform) version image. More details in section 5.7

Table 2 Representative Algorithms in Search Engine

Algorithms	**Role in the search engine**
graph mining	crawl web page
Grep	abstracting content from HTML
segmentation	word segmentation
pagerank	compute the page rank value
Word counting	word frequency count
vector calculation	document matching
sort	document sorting

5. Deploy and Run the Benchmark

For our benchmark suite consist different programming model based workloads and some services. We would like to introduce the deployment and running separately according to their programming model.

5.1 Hadoop based benchmark

5.1.1 Prerequisite Software Packages

We adopt some machine learning algorithms from Mahout and some database operations based on Hive, so we also need to install Mahout and Hive in advance.

Hadoop

We recommend version 1.0.2, which was used and tested in our environment. Version 1.0.2 download link:

mirrors.tuna.tsinghua.edu.cn/apache/hadoop/common/hadoop-1.0.2

Mahout

We recommend mahout-distribution-0.6, which was used and tested in our environment.

Mahout-distribution-0.6 download link:

http://mirrors.tuna.tsinghua.edu.cn/apache/mahout/

Hive

We recommend version 0.6.0, which was used and tested in our environment.

Version 0.6.0 download link:

http://hive.apache.org/releases.html#Download

For how to deploy them, this document will not show the details. Users can find the details from corresponding web site or readme.

5.1.2 Deploy the Benchmarks

Users need to decompress the package on the specified directory.

tar -zxvf DCBench-hadoop.tar.gz

Then we will get a directory DCBench-hadoop which we will refer to it as **$DCBench-hadoop_HOME**

In our **$DCBench-hadoop_HOME**, there are three main folders ***Workloads, basedata***, and ***datagen*** which store scripts, original dataset and data generation tools respectively.

The configuration file is *config.include* under ***$DCBench-hadoop_HOME /workloads/configuration/***, which is used to configure the path. Notice that, users must modify this configuration file based on their environment.

config.include: record related configurations
Parameters:
basedata_dir: data storage path in local
tempdata_dir: temporary data storage directory
Temporary data is generated during the data preparatory phase, and will be deleted after final data are uploaded to HDFS.
hdfsdata_dir: corresponding HDFS directory
basejars_dir: the path of basic jars
hadoop_example_jar: the name of hadoop-examples-jar, users need to change this name in terms of the Hadoop jars release they use.
Notice: please modify these paths according to machine configuration before running applications.

file_all.include: record the datasets now existing on HDFS. Users can refer to this file to write batch scripts. This file only reflects the operations on HDFS through our scripts, that means, if user delete the datasets directly by Hadoop commands, these operations will not be refresh in this file.

5.1.3 Basic data set

In our benchmark suite, some benchmark have the corresponding data generator to generate the wanted size data. But there are also some benchmarks, their data is hard to generate, so we provide some original datasets as the basic data set. If users want to drive a benchmark with larger data set, we provide the scripts to duplicate the basic data set to multiple copies. Table 3 lists the corresponding relationship between applications and original datasets. The datasets can be seen under ***$DCBench-hadoop_HOME/basedata/.***

Table 3 The corresponding relationship between application and original datasets

Category	Application	Original Datasets	
Base-operations	Sort	Generated	
	WordCount		
	Grep		
Classification	Naive Bayes	wikipediainput (1.95G)	
	Support Vector Machine	svm-data (1.96G)	
Clustering	K-means	low(default)	sougou-low-tfidf-vec (121K)
		mid	sougou-mid-tfidf-vec (4M)
	Fuzzy-K-means	high	sougou-high-tfidf-vec (16M)
Recommendation	Item based collaboration filtering	ibcf-data (1.95G)	
Association rule mining	Frequent pattern-growth	low(default)	fpg-accidents.dat (3.97M)
		mid	fpg-retail.dat (33.86M)
		high	fpg-webdocs.dat (1.38G)
Sequence-leaning	Hidden markov model	hmm-data (1.95G)	
Hive-bench	Grep select	Generated	
	Ranking select		
	Uservisits aggregation		
	Uservisits-ranking join		

5.1.4 Run the Benchmark

Basic operation

Sort

Scripts path: $DCBench-hadoop_HOME/workloads/base-operations/sort

Command: ./prepare-sort.sh 10g

Parameter： the size of dataset users want to specify, it can be m/M, g/G, t/T.

Explanation: it will produce 10GB data for sort. Script will adjust parameter *byte_per_map* and *maps_per_host* in *config-sort.xml* according to current deployment. Then RandonWriter generates data using this new configuration file.

Command: ./run-sort.sh 10g

Explanation: run sort application with 10 GB dataset

WordCount

Scritps path: $DCBench-hadoop_HOME/workloads /base-operations/wordcount

Command: ./prepare-wordcount.sh 10g

Parameter: the size of dataset users want to generate, it can be m/M, g/G, t /T.

Explanation: it will produce 10g dataset for wordcount. Script will adjust parameter *byte_per_map* and *maps_per_host* in *config-sort.xml* according to current deployment. Then RandontextWriter generates dataset using this new configuration file.

Command： ./run-wordcount.sh 10g

Explanation: run with 10 GB dataset

Grep

Scritps path: $DCBench-hadoop_HOME/workloads/base-operations/grep

Command: ./prepare-grep.sh 10g

Parameter: the size of dataset users want to generate, it can be m/M, g/G, t/T.

Explanation: it will produce 10g dataset for grep. Script will adjust parameter *byte_per_map* and *maps_per_host* in *config-sort.xml* according to current deployment. Then RandontextWriter generates dataset using this new configuration file.

Command： ./run-grep.sh 10g

Explanation: run grep with 10GB dataset.

Classifier

Naive Bayes

Scripts path: $DCBench-hadoop_HOME/workloads/classification/NaiveBayes/

Original Input Dataset path: $DCBench-hadoop_HOME/basedata/wikipediainput

Command: ./prepare-bayes.sh 2

Parameter： the multiple of original dataset users want to expand.

Explanation: execute the data preparation, it will copy the original dataset according to parameter, then upload the new dataset onto the HDFS. For example, if the basic dataset is 2.1G, and the parameter is 2, then the new dataset will be 4.2 GB.

Command： ./run-bayes.sh 2

parameter： Run the Naive Bayes with 2 copies original data set. The parameter, 2, here is corresponding to above.

Explanation： This command will choose dataset ***bayes-4.2G*** to run bayes application. The dataset must be generated in advance. Users can refer to *file.include* to check the dataset now existing on HDFS.

Command: ./delete-bayes.sh 2

Parameter: to delete 2 copies of basic dataset on HDFS

Explanation: delete dataset on HDFS, for this command, it will delete the dataset ***bayes-4.2G*** *(same as above)* on HDFS.

Support Vector Machine

Scripts path： $DCBench-hadoop_HOME/workloads/classification/SVM/

Original Input Dataset path：$DCBench-hadoop_HOME/basedata/svm-data

Command: ./prepare-svm.sh 2

Parameter： the number of copies of original dataset

Explanation: execute the data preparation. It will copy the original dataset according to parameter, and then upload the new dataset onto the HDFS. For example, if the basic dataset is 2.1G, and the parameter is 2, then the new dataset will be 4.2 GB.

Command： ./run-svm.sh 2

Parameter: the number of copies of original dataset, used to specify dataset

Explanation: execute the SVM program with the specified dataset. For this command, it will run application with dataset *SVM-4.2G*. The dataset must be generated in advance. Users can refer to *file.include* to check the dataset now existing on HDFS.

Command: ./delete-svm.sh 2

Parameter: to delete extended data with 2 copies of basic dataset on HDFS

Explanation: delete dataset on HDFS, for this command, it will delete the dataset ***SVM-4.2G*** on HDFS.

Cluster

K-means

Scripts path: $DCBench-hadoop_HOME/workloads/cluster/kmeans/

Original Input Dataset path： $DCBench-hadoop_HOME/basedata/sougou-low-tfidf-vec

$DCBench-hadoop_HOME /basedata/sougou-mid-tfidf-vec

$DCBench-hadoop_HOME /basedata/sougou-high-tfidf-vec

Command: ./prepare-kmeans.sh low|mid|high

Parameter: low|mid|high (represent the dataset with different size)

Explanation: upload the basic dataset to the HDFS. the relationship between the parameter and the dataset is showed in Table 3.

Command： ./run-kmeans.sh low|mid|high

Parameter： low|mid|high (represent the dataset with different size)

Explanation： the relationship between the parameter and the dataset is showed in Table 3.

Command: ./delete-kmeans.sh low|mid|high

Parameter： low|mid|high (represent the dataset with different size)

Explanation: delete the corresponding dataset on HDFS.

Fuzzy-kmeans

Scripts path: $DCBench-hadoop_HOME/workloads/cluster/fkmeans/

Original Input Dataset path： $DCBench-hadoop_HOME /basedata/sougou-low-tfidf-vec

$DCBench-hadoop_HOME /basedata/sougou-mid-tfidf-vec

$DCBench-hadoop_HOME /basedata/sougou-high-tfidf-vec

Command: ./prepare-fkmeans.sh low|mid|high

Parameter: low|mid|high (represent the dataset with different size)

Explanation: upload the basic dataset to the HDFS

Command： ./run-fkmeans.sh low|mid|high

Parameter： low|mid|high (represent the dataset with different size)

Explanation： the relationship between the parameter and the dataset is showed in Table 3.

Command: ./delete-fkmeans.sh low|mid|high

Parameter： low|mid|high (represent the dataset with different size)

Explanation: delete the corresponding dataset on HDFS.

Recommendation

Item based collaboration filtering

Scripts path: $DCBench-hadoop_HOME/workloads/recommendation/ibcf/

Original Input Dataset path: $DCBench-hadoop_HOME/basedata/ibcf-data

Command： ./prepare-ibcf.sh 2

Parameter： *2* is the number of copies of original dataset users want to expand

Explanation： execute the data preparation. It will copy the original dataset, then upload the new dataset to HDFS.

Command： ./run-ibcf.sh 2

Parameter: in accordance with *prepare-ibcf.sh*, used to specify dataset

Explanation: execute the IBCF program with the specified dataset. For this command, it will run application with dataset ***ibcf-24M*** *(same as above)*. The dataset must be generated in advance.

Command: ./delete-ibcf.sh 2

Parameter: to delete extended data HDFS

Explanation: delete dataset on HDFS, for this command, it will delete the dataset *ibcf-24M (same as above)* on HDFS.

Association rule mining

Frequent pattern-growth

Script path: $DCBench-hadoop_HOME/workloads/associationrulemining/fpg/

Input data set: $DCBench-hadoop_HOME/basedata/fpg-accidents.dat

$DCBench-hadoop_HOME/basedata/fpg-retail.dat

$DCBench-hadoop_HOME/basedata/fpg-webdocs.dat

Command： ./prepare-fpg.sh low|mid|high

Parameter： low|mid|high respects the different data sets

Explanation: upload the basic dataset to the HDFS. the relationship between the parameter and the dataset is showed in Table 3..

Command： ./run-fpg.sh low|mid|high

Parameter： low|mid|high respects the different data set

Explanation： the relationship between the parameter and the dataset is showed in Table 3. .

Sequence learning

Hidden markov model

Scripts path: $DCBench-hadoop_HOME/workloads/sequence-leaning/

Input data set: $DCBench-hadoop_HOME/basedata/hmm-data

Command: ./prepare-hmm.sh 2

Parameter: parameter *2* is the copies of basic dataset users want to expand

Explanation: expand the basic dataset according to parameter, and then upload new dataset to HDFS. For example, if basic dataset is *510M*, then the new dataset is hmm-1020M.

Command： ./run-hmm.sh 2

Parameter: to drive hmm application with 2 copies of original data set

Explanation: execute the HMM program with the specified dataset. For this command, it will run application with dataset *hmm-1020M (same as above)*. The dataset must be generated in advance. Users can refer to *file.include* to check the dataset now existing on HDFS.

Command: ./delete-hmm.sh 2

Parameter: to delete extended data with 2 multiple of basic dataset on HDFS

Explanation: delete dataset on HDFS, for this command, it will delete the dataset

hmm-1020M *(same as above)* on HDFS.

Data warehouse operations

To run this application, firstly, users need to use data generation tools to generate dataset. These tools can specify size of dataset, the things users need to do is just modifying the configurations.

Folders ***htmlgen*** and ***teragen*** under ***$DCBench-hadoop_HOME/datagen*** store all scripts, and ***htmlgen*** will produce two tables called *Ranking01.data* and *UserVisits01.data* with specified size, while, ***teragen*** will produce *Table Grep*.

htmlgen:

Scripts path: ***$DCBench-hadoop_HOME***/datagen/htmlgen/

Command: ./generateData.py

Explanation: The configuration file is **config.txt,** users can change the ***Rowcount for UserVisits*** to decide the size of dataset. And, users need to specify the path that stores the generated dataset, because this tool will delete the files under the specified directory, our suggestion is to create a new directory to store the dataset, then move them to ***$DCBench-hadoop_HOME /basedata/***.

teragen:

Scripts path: ***$DCBench-hadoop_HOME***/datagen/teragen

Command: sh teragen.pl

Explanation: The configuration file is **teragen.pl** itself. Users can modify the parameter ***NUM_OF_RECORDS*** to specify the size of data.

Scripts path: ***$DCBench-hadoop_HOME***/workloads/hive-bench/

HDFS path:	/data/grep/	(input)
	/data/ranking/	(input)
	/data/uservisits/	(input)

/output/hadoop_bench/ (output)

Notice: Users need to create these folders on HDFS in advance.

Command: ./prepare.sh

Explanation: upload datasets to HDFS with path showed above.

Grep select

Command: ./benchmark_gs.sh

Explanation： main operation in grep_select

```
SELECT *
FROM grep
WHERE field LIKE '%XYZ%'
```

Ranking select

Command： ./benchmark_rs.sh

Explanation： main operation in rankings_select

```
SELECT pageRank, pageURL
FROM rankings
WHERE pageRank > 10
```

Uservisits aggregation

Command： ./benchmark_ua.sh

Explanation： main operation in rankings, uservist

```
SELECT sourceIP, SUM(adRevenue)
FROM uservisits
GROUP BY sourceIP
```

Uservisits-rankings Join

Command： ./benchmark_ruj.sh

Explanation： main operation in uservisits_aggre

```
SELECT sourceIP, avg(pageRank), sum(adRevenue) as totalRevenue

FROM rankings R

JOIN

( SELECT sourceIP, destURL, adRevenue

FROM uservisits UV

WHERE UV.visitDate > '1999-01-01' AND UV.visitDate < '2000-01-01' )

NUV ON (R.pageURL = NUV.destURL)

group by sourceIP

order by totalRevenue DESC limit 1
```

5.2 MPI Based Benchmark

There are also some benchmarks in our benchmark suite based on MPI. For those benchmarks, we get the applications from IBM Parallel Machine Learning (PML) Toolkit and Graph 500.

5.2.1 Prerequisite Software Packages

MPI

We recommend mpich2-1.3.1, which was used and tested in our environment. Other MPI version should also be fine.

For how to deploy them, this document will not show the details. Users can find the details from corresponding web site or readme.

5.2.2 Deploy the Benchmarks

Users need to decompress the package on the same directory in all the nodes in cluster, which users want to run them on.

tar -zxvf DCBench-MPI.tar.gz

Then we will get a directory DCBench-MPI which we will refer to it as

$DCBench-MPI_HOME

In our $DCBench-MPI_HOME, there are two main folders ***ML***, and ***graph500*** which store. The ***ML*** directory stores K-means, Fuzzy K-measn, PCA, and KPCA benchmarks and corresponding scripts, configuration files. The ***graph500*** directory stores the Breadth-first search benchmark (BFS).

Deploy PML

The applications in ML directory are all from IBM PML Toolkit, and the IBM only provides the binary file, not the source, so we just provide the Linux version without source code. Those applications do can use directly after uncompressing.

Deploy BFS

Modify configuration file

Modify the **$DCBench-MPI_HOME/graph500/make.inc** file under the directory of graph500, modify the execution command of MPICC:

MPICC = /path/to/MIP_installation_directory/bin/mpicc -I /path/to/MIP_install_directory /include -L /path/to/MIP_install_directory/lib

3. Set the environment variable:

Modify ~/.bashrc file and add the following line

export PATH=$PATH:/path/to/MIP_installation_directory/bin/

and then source the file by using

source ~/.bashrc

Enter the directory **$DCBench-MPI_HOME/graph500/mpi** and execute the command below:

Make clean

Make

4. Test

Execute *ldd graph500_mpi_one_sided* or *ldd graph500_mpi_simple* in directory **$DCBench-MPI_HOME/graph500/mpi** to check whether exists library files, if yes, there is no problem.

5.2.3 Run the Benchmark

Generate data for K-means, Fuzzy K-means, KPCA, and PCA

The data generator will generate a matrix for all the four applications in ARFF formant. Each line in the matrix represents a element waiting for clustering.

Command:

$DCBench-MPI_HOME/ML/data_gen/datagenerate.py M N >$file.arff

Parameter:

M : number of lines

N : number of columns

$file.arff : the file users specified to store the data

After generating the ARFF formant users need to convert the ARFF format data using the following command:

$DCBench-MPI_HOME/ML/Linux/dataconverter ARFF $file.arff $fileout

The *$fileout* is the converted file.

Run K-means, Fuzzy K-means, KPCA, or PCA

Users can run those four applications by using the following commands:

In *$DCBench-MPI_HOME/ML/Linux*/ directory

mpiexec –np num ./pmlexec   ../parameter_files/parameter_file_fuzzy_kmeans.xml *$fileout* $result

Parameters:

num: number of processes

parameter_file_fuzzy_kmeans.xml: the configuration file for certain benchmark, users can change the configuration file according to the benchmark. All the configuration file can be found in *$DCBench-MPI_HOME/ML/parameter_file.* Users can use the default ones or modify the configuration file according to their own needs. The detail of how to modify the configuration file can be found in *$DCBench-MPI_HOME/ML/PML User Guide.pdf*

Run the BFS

Program path: $DCBench-MPI_HOME/graph500/mpi/

Execution command: mpiexec -np proc graph500_mpi_simple scale

Parameter:

proc: the number of processes to execute graph program, it must be an exact power of 2.

scale: The problem scale is the logarithm, base 2, of the number of vertices in the graph; only graphs with power-of-2 vertex counts are supported without source code modification.

5.3 All-Pairs based Benchmark

For the Vector calculation, we include a paper similarity analysis application, which is developed using All-Pairs.

5.3.1 Deploy the Benchmark

Users need to decompress the package on the same directory in all the nodes in cluster, which users want to run them on.

tar -zxvf DCBench-All-pairs.tar.gz

Then we will get a directory DCBench-All-pairs which we will refer to it as **$DCBench-All-pairs_HOME**

In our **$DCBench-All-pairs_HOME**, there are two main folders ***cctool*** and ***paper_similar***. The ***cctool*** directory stores All-pairs framework we need to deploy. The ***paper_similar*** directory stores the benchmark source codes.

Deploy All-Pairs

For most the cases, the All-pairs can be directly used without compiling. So users just need add set the environment variable by using following command:

setenv PATH $HOME/cctools/bin:$PATH

Or modifying ~/.bashrc file and add the following line

export PATH=$PATH:/path/to/MIP_installation_directory/bin/

and then source the file by using

source ~/.bashrc

If users can not use it directly, it is also very easy to build the CCTools from source. Just download a source package: Download

And follow this recipe while logged in as any ordinary user:

```
% gunzip cctools-xxx-src.tar.gz

% tar xvf cctools-xxx-src.tar

% cd cctools-xxx-src

% ./configure

% make

% make install

% setenv PATH ${HOME}/cctools/bin:$PATH
```

Deploy Application

Under **$DCBench-All-pairs_HOME/paper_similar** directory just enter the *make* command, then the application will be compiled.

5.3.2 Run the Benchmark

This benchmark will compare tow papers' similarity. Before running the benchmark we should generate the two paper lists, the benchmark will compute Cartesian product of two lists. The Cartesian product is the similarity of the two papers.

Generate data

Under *$DCBench-All-pairs_HOME/paper_similar* directory

Command:

```
./data_gen.py N >$a.listf
```

Parameter:

N : number of papers in a.list

$a.list : the file users specified to store the paper list

Run

The All-pairs framework runs in the mast-slave model. Users need to specify the master node and slave nodes.

In the all slave nodes, users need to execute the following command:

```
work_queue_worker $master 9123
```

Parameter:

$master: master node's hostname or IP address

9132: the port slave used to communicate with master node. We recommend the users do not change other ports.

In the master node, user needs to execute following command under the directory *$DCBench-All-pairs_HOME/paper_similar*

```
allpairs_master a.list b.list ./main
```

Parameter:

a.list and *b.list* are the paper list generated in last step.

./main is the application's binary.

5.4 Search

The Search's corresponding user manual can be found **here**

5.5 Media Streaming

The Media Streaming corresponding user manual can be found **here**

5.6 Rubis

The Rubis corresponding user manual can be found in **here**

5.7 XCP workloads

Cloud computing provide a new scenarios. It is an infrastructure, which consolidates several virtual machines. Those virtual machines are providing service to users with a “pay as you go” pattern. In order to simulate the virtual environments, we also provide some virtualization workloads. Currently, we provide the Media Streaming and Search virtualization workloads. Those workloads are deployed in the XCP images. Users can deploy those images on their own XCP (Xeon Cloud Platform) and benchmark their system by consult the corresponding user manual, which can be found **here**