User Guide

Overview

CHCF-1.0 is a redistributable package implementing several multimedia mining algorithms such as feature extraction, K-Means clustering, BoF (or BoW) vector generation, HE training and inverted file index generation which are discussed in our paper [1]. It is provided aiming at illustrating the tremendous power of heterogeneous computer cluster by utilizing the dedicated implementation of the algorithms running on multi-GPUs. As a demonstration, several functions and features are limited or hidden from the end-users. For instance, only one option of feature extraction is provided, which is Hessian Affine algorithm. The available algorithms are listed as follows.

Table 1 Available algorithms provided by CHCF-1.0

Algorithm name	**Description**
ImgProc	For feature extraction, only Hessian Affine method running on CPU is implemented.
Clustering	K-Means clustering algorithm running on CPU and GPU.
BoW	BoW vector generation running on CPU and GPU.
HETrain	HE training algorithm running on CPU and GPU.
IFGen	Inverted file index generation running on CPU and GPU.

The package contains scripts, JAR (Java Archive) and shared object files, which can be executed under Linux with the presence of third-party libraries and frameworks. The detailed description of the necessary libraries and frameworks can be found in the section of installation.

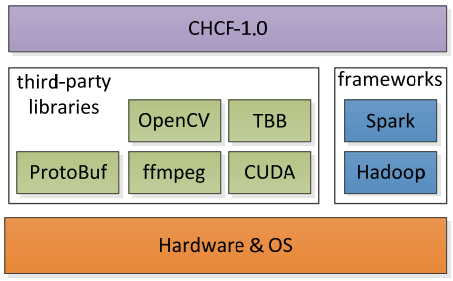

Fig. 1 Dependency of CHCF-1.0

The dependency of CHCF-1.0 is shown in Fig. 1. The libraries of ProtocolBuffer and TBB are contained in the CHCF-1.0 package so as to avoid unnecessary mistakes since they are stable and small-size.

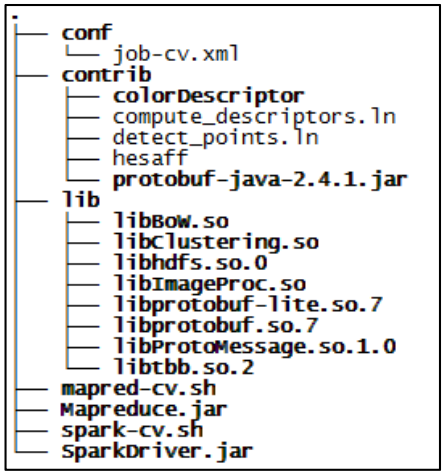

Fig. 2 Structure of the directory

The structure of the package is depicted in Fig. 2. Three sub-directories, conf, contrib and lib, are placed under the root, which hold configuration file template of the computation job, the third-party executable files and shared object files, respectively. Under the root directory, there are two JAR files, which implement the driver code on Hadoop and Spark, and two corresponding Linux shell scripts that are responsible for launching the computation jobs. The detailed description of the files is listed below.

Table 2 Descriptions of the files

File	Description
Mapreduce.jar mapred-cv.sh	The driver code of the algorithms against Hadoop and the corresponding scripts launching the computation job
SparkDriver.jar spark-cv.sh	The driver code of the algorithms against Spark and the corresponding scripts launching the computation job
job-cv.xml	The template of the configuration file of the computation job. Every property defined in this file is followed by a default value and simple description
libBoW.so	The module implemented algorithms of BoW vector generation, HE training and inverted file index generation running on GPU
libClustering.so	The module implemented K-Means clustering running on GPU
libImageProc.so	The module implemented feature extraction
libProtoMessage.so.1.0	The module implemented all data structures shared by the algorithms, such as feature vector, BoW vector and so on
colorDescriptor compute_descriptors.ln detect_points.ln hesaff	Detectors and descriptors for feature extraction retrieved from third-party libraries or open source code
protobuf-java-2.4.1.jar	The JAR file of library ProtocolBuffer downloaded from Google website
libprotobuf.so.7 libprotobuf-lite.so.7	The runtime of library ProtocolBuffer, downloaded from Google website.

libtbb.so.2	The runtime of library Intel TBB, downloaded from Intel website
libhdfs.so.0	The C/C++ API library of HDFS, downloaded from Apache Hadoop

Configuration file

All path related properties listed in the cfg file are indicated as HDFS path.

Table 3 Properties defined in the configuration file

Property	Description
AlgorithmsName	The name of algorithms to be evaluated, available values include: ImgProc, Clustering, BoW, HETrain and IFGen
ImgRootDir	The path of root directory holding the image set which is the input data of ImgProc algorithm
FeatureFileDir	The path of directory holding the feature vectors which are generated by ImgProc
ImageProcTimeout	The timeout of each individual sub-task for the ImgProc algorithm. The computation job of ImgProc is divided into several sub-tasks according to the total size of the images to be processed and the value of **ImageProcInputSplitSizeMB**. A suitable timeout is helpful for detecting the potential failures of individual nodes and restarting the sub-task on other feasible nodes timely, which in turn improve the throughput capacity of the cluster
FeatureFileSizeInMB	The max size of feature vectors split created by ImgProc algorithm. Obviously, the feature file is the input data for algorithms of Clustering, BoW, HETrain and IFGen. A value little bit smaller than that of block size of HDFS can prevent sub-tasks from reading one copy of a single input file remotely according to the policy of HDFS
ImageProcInputSplitSizeMB	The size of one input split for ImgProc. The smaller the value is set, the more sub-tasks will be created, which yield higher parallelization as well as more overhead
ClusterPath	The path of the cluster file produced by Clustering
NumOfClusters	The number of centroids for Clustering
ClusteringLoopNum	The max number of iteration for Clustering
ThresholdClustering	The threshold for Clustering
ClusteringInputSplitSizeInMB	The size of one input split for Clustering. It is similar to **ImageProcInputSplitSizeMB**
ClusteringTimeout	The timeout of each individual sub-task for the Clustering. It is similar to **ImageProcTimeout**
ClusteringUsingGpu	Enable GPU for Clustering
InitClusterGlobalCached	Reduce data retrieval from HDFS and improve the performance if this option is enabled

BowInputSplitSizeInMB	The size of one input split for BoW. It is similar to **ImageProcInputSplitSizeMB**
BowResultFilePath	The path of BoW vectors generated by BoW
BowTimeout	The timeout of each individual sub-task for the BoW. It is similar to **ImageProcTimeout**
BoWUsingGpu	Enable GPU for BoW
BowClusterGlobalCached	It is similar to **InitClusterGlobalCached**
IFGenRes	The path of the inverted file index generated by IFGen
HETrainRes	The path of the training result generated by HETrain
HeTrainingCltsCached	It is similar to **InitClusterGlobalCached**
HeProcTimeout	The timeout of each individual sub-task for the HETrain and IFGen. It is similar to **ImageProcTimeout**
HeGenCltsCached	It is similar to **InitClusterGlobalCached**
HeGenHeCached	It is similar to **InitClusterGlobalCached**

How to run

Before launching a computation job, all third-party libraries and frameworks are supposed to be installed in the computer cluster. Please refer to the section of installation for detailed information.

TIPS: how to check the availability of the necessary libraries in your system?
Enter the root directory of CHCF-1.0, and issue the commands below:

1. export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:lib
2. ldd lib/libImageProc.so | grep 'not found'

If no results show up, all necessary libraries are available.

After preparing the data of the algorithm (images for feature extraction, for instance) and putting them onto HDFS, follow the commands below to launch the computation job.

1) Enter the root directory of CHCF-1.0.
2) Configure the cfg file of the computation. Use the template file job-cv.xml as a start and refer to the detailed descriptions in the template for each property.
3) Launch the computation executed by Hadoop by issuing the command "./mapred-cv.sh <job-cfg-file>".
4) Command "./spark-cv.sh <job-cfg-file>" is used to execute the computation on Spark.

TIPS: All data including input and output are placed on HDFS and the output data such features, clusters are encoded by ProtocolBuffer.

Example – Offline Training of Image Retrieval

The offline training of the image retrieval consists of three stages: feature extraction, clustering and BoW vector generation, which can be processed by ImgProc, Clustering and BoW, respectively. Therefore, we set the value of AlgorithmsName to "ImgProc, Clustering, BoW" in the cfg file.

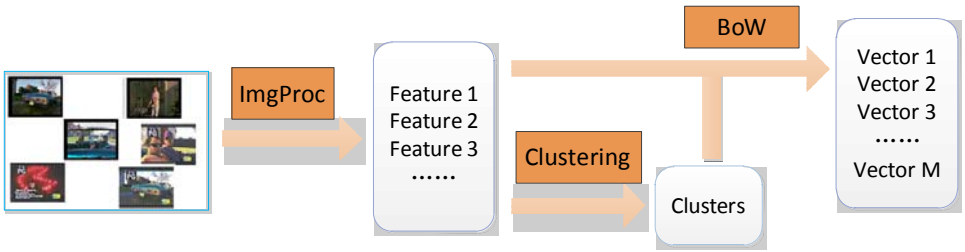

Fig. 3 The flow chart of offline training

To launch ImgProc, the imageset has to be uploaded onto HDFS at first. If we put images in the directory of "/images/scene/train", for instance, the value of ImgRootDir is supposed to set to "/images/scene/train" in the cfg file. The feature vectors will be placed in the directory specified by the property of FeatureFileDir. Similarly, the cluster file generated by clustering will place in the directory specified by property of ClusterPath.

Contents of directory /images/scene

Goto : /images/scene go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission
train	dir				2016-04-19 11:49	rwxr-xr-x

Fig. 4 The directory of images on HDFS

Contents of directory /images/features

Goto : /images/features go

Go to parent directory

Name	Type	Size
attempt_201603162316_0112_m_000000_0_node02_1461205998091_c_feature_0.res	file	59.76 MB
attempt_201603162316_0112_m_000000_0_node02_1461206000853_c_feature_1.res	file	59.89 MB
attempt_201603162316_0112_m_000000_0_node02_1461206214902_c_feature_0.res	file	59.67 MB
attempt_201603162316_0112_m_000001_0_node06_1461205959773_c_feature_0.res	file	59.91 MB
attempt_201603162316_0112_m_000001_0_node06_1461205962219_c_feature_1.res	file	40.88 MB
attempt_201603162316_0112_r_000000_0_node06_1461206212181_r_feature_0.res.com.res	file	48.26 MB

Fig. 5 The feature vectors on HDFS

Fig. 6 The cluster generated by Clustering

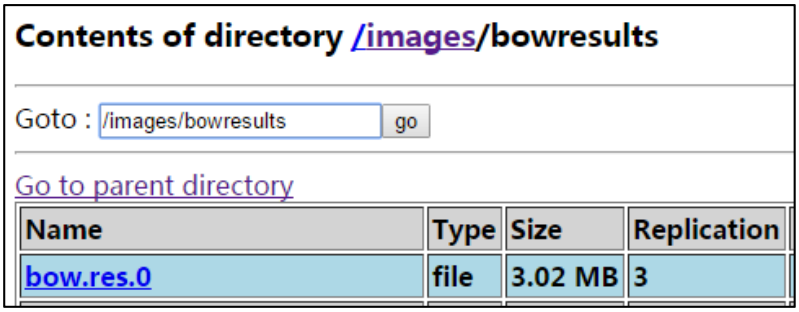

Fig. 7 The BoW vectors on HDFS

Installation

Hardware
Each node in the cluster is equipped with one NVidia GPU at least.

OS
Ubuntu-12.04 Desktop 64 bit

JDK
Install JDK by using apt-get or downloading the zip package from internet.

Hadoop
Please refer to the manual of Hadoop for its installation on the cluster. Basically, you just need to put the directory of Hadoop that is unzipped from the package file hadoop-1.2.1.tar.gz under the directory of /usr/local. And then configure the Hadoop in the cluster by modifying the files of core-site.xml, hdfs-site.xml, mapred-site.xml and hadoop-env.sh in the directory of conf.

Spark
Please refer to the manual of Spark for its installation. The version we used is 1.4.1.

The CHCF package released is dependent on the third-party libraries with the **exact version** indicated below. There can be no guarantee that the package will work well when other versions of the third-party libraries are installed in the system. Please install the third-party libraries orderly.

(1) cuda-5.5 (including the NVidia display driver)
(2) ffmpeg-0.10.6
(3) opencv-2.4.7 (cmake with version >= 2.8.10 is prerequisite for compiling and enable CUDA)

Environment variables configuration. Make sure the paths are the exact ones in your system.

```
export HADOOP_HOME=/usr/local/hadoop
export CUDA_PATH=/usr/local/cuda
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$CUDA_PATH/lib64
```

Reference

1. Hanli Wang, Bo Xiao, Lei Wang, Fengkuangtian Zhu, Yu-Gang Jiang, and Jun Wu, CHCF: A Cloud-based Heterogeneous Computing Framework for Large-Scale Image Retrieval, ***IEEE Transactions on Circuits and Systems for Video Technology***, vol. 25, no. 12, pp. 1900-1913, Dec. 2015.