東海大學

資訊工程研究所

碩士論文

指導教授: 楊朝棟博士

實作結合虛擬化部署的 Hadoop 生態系統入口網

The Implementation of a Hadoop Ecosystem Portal with Virtualization Deployment

研究生: 蔣元斌

中華民國一零六年七月

東海大學碩士學位論文考試審定書

東海大學資訊工程學系 研究所

研究生 蔣 元 斌 所提之論文

實作結合虛擬化部署的 Hadoop 生態系統入口網

經本委員會審查，符合碩士學位論文標準。

學位考試委員會

召 集 人 許慶賢 簽章

委 員 黃國展

姜尚瑜

劉榮春

指 導 教 授 楊朝棟 簽章

中華民國 106 年 6 月 27 日

摘　要

現今巨量資料在許多領域如資訊、金融、醫學等的發展皆具有越來越重要的地位，因此巨量資料的研究、分析、處理等的需求也越來越多。目前用於巨量資料的環境大部份是使用 Hadoop 或 Spark 來進行處理分析，但此類環境的建置需具備有一定的專業知識與對系統的熟悉度，對一般使用者來說有一定的難度。另此類環境的操作皆須以輸入指令的方式進行操作，對習慣使用圖形桌面環境的使用者存在學習使用指令操作的門檻。為了降低使用者操作巨量資料工具進行處理分析的門檻，本論文利用 Liferay Portal 實作了結合 Hadoop 與 Spark 巨量資料平台的網頁使用者介面，並更進一步將巨量資料平台與網頁使用者介面整合於虛擬機映像檔中，讓使用者能夠快速方便地部建巨量資料平台與執行巨量資料的工作。希望提供便利的網頁使用者介面的同時，減少對工作時運算效能的影響，且降低建立巨量資料平台的難度與所需時間。本論文亦進行了透過使用指令與透過網頁使用者介面執行巨量資料處理工作的效能比較，使用的是 HiBench 的測試套件，測試在提供使用者方便性的同時，網頁使用者介面對巨量資料處理工作時的效能所造成影響的程度。另外，由於相關研究的需要，在本環境中亦建置了 OpenCV 的環境，也實際使用粒子圖像測速法程式驗證 OpenCV 函式庫在本系統中的可用性。

關鍵字: 巨量資料平台，Portlet，虛擬化，Hadoop，Spark。

Abstract

The requirements of research, analysis, processing and storing of big data are more and more because big data is increasingly important for development in the fields of information technology, finance, medicine, etc. Most of the big data environments are built on Hadoop or Spark. But the constructions of these kinds of big data platform are not easy for normal users because of the lacks of professional knowledge and familiarity with the system. And users usually have to learn how to use the command line for operations. To make it easier to use the big data platform for data processing and analysis, we implemented the web user interface combining the big data platform including Hadoop and Spark. And we packaged the whole big data platform into the virtual machine image file along with the web user interface so that users can construct the environment and do the job more quickly and easily. We provide the convenient web user interface, reduce the difficulty of building a big data platform and save time but do not reduce too much performance of the system. And we also made the comparison of performance between the web user interface and the command line using the HiBench benchmark suit. In addition, we have also built the OpenCV library in our system for our related research. The functionality of OpenCV in our system is validated by performing the Particle Image Velocimetry (PIV) applications.

Keywords: Big Data Platform, Portlet, Virtualization, Hadoop, Spark.

致謝詞

回顧在東海大學資訊工程研究所的這兩年，我受到許多師長的教導和同儕們的協助，使我在研究的領域持續精進。透過對巨量資料與虛擬化技術的深入研究，將這些技術實際應用於研究與開發中，建立了一套系統並完成本篇論文。

在研究領域能有今日的成果，非常感謝我的指導教授楊朝棟博士。從大學時期的課程就受到楊老師的教導，後來楊老師鼓勵我繼續在資訊領域做更深入的研究，除了提供研究資源，也讓我有培養表達能力以及拓展國際視野的機會。楊老師不僅傳授我做研究的方法與學問，也教導我為人處世的態度，謝謝老師用心的教導。

特別感謝口試委員許慶賢教授、劉榮春教授、姜自強教授以及黃國展教授特地撥空前來參加我的論文口試，在口試時點出我論文的盲點也給予許多的建議，讓我能再將論文修改得更加完整，學生衷心感謝。

也很感謝國家高速網路與計算中心的蔡惠峰博士、張文鎰博士以及吳建衡助理研究員，在研究與開發的過程中提供了許多協助與建議，使我的論文更完整。

另外也要感謝我的學長姐們做為榜樣，傳承寶貴的學習與研究經驗，並在我感到疑惑時適時給予指點。也感謝實驗室的夥伴與學弟妹們互相扶持與指教，在各自擅長的領域提供協助，共同學習與成長。

最後，也是最重要的，感謝始終支持著我的家人。在求學的期間，你們適時的關心與叮嚀給予我持續努力的動力。你們對我的支持也讓我無後顧之憂，能夠專注於學業與研究，非常感謝。

東海大學資訊工程學系 高效能計算實驗室 蔣元斌 一零六年七月

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Big Data is becoming more and more important today. Due to the rapid development of computers, networks and information services, a large amount of data has been generated [1] [2]. A variety of theories, researches, and applications of big data flourish and become a trend. Big data were only accessed by scientists, researchers or large companies before, but it is getting closer to us now and more related to human being and many types of fields. Many industries see it as an important resource. The relevant researches, development, storage, applications and environments are constantly expanding and updating because of the development of big data. It is a good time for people who are willing to get into the field of big data because there are more and more resources available.

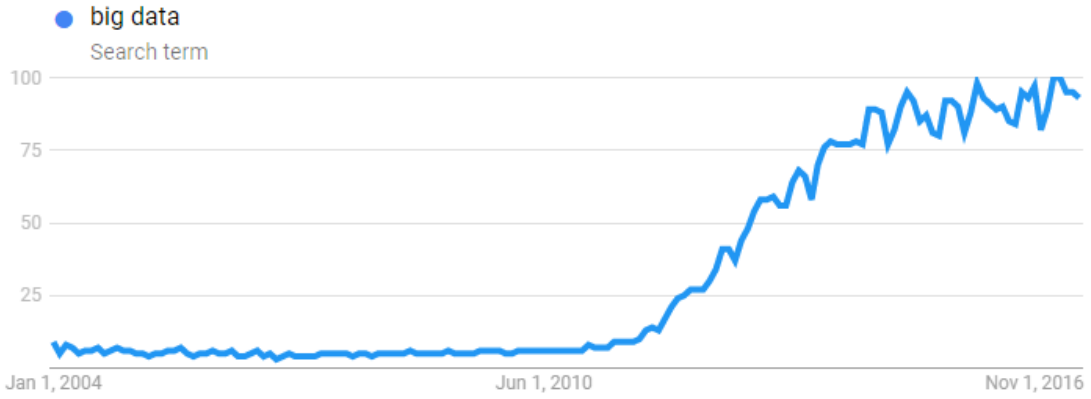

FIGURE 1.1: Search interest of big data [3]

1.1 Motivation

Although there are many resources about big data and many applications available, normal users may have some problems using big data tools at present. The problems may be how to prepare the environment suitable for big data, how to set up the whole environment, there is some possibility during the installation, unfamiliar with the command-line interface of Linux and the big data tools, etc. Because of the need for big data research, we want to address these situations that are present. We want to do the researches and develop a way to simplify the pre-operation and installation process of the big data platform. The way that can deploy the big data platform directly in the existing environment, doesn't require the dedicated devices, let users choose the environment they are familiar to, makes users operating the tool intuitively, reduces the chance of error and makes the different types of jobs executing together. Besides, we want to make the file management, job status monitoring and job scheduling more easier and the capability for advanced users to adding functions by themselves.

1.2 Contributions

In this work, we implemented the web user interface applying to Hadoop ecosystem. Hadoop platform is widely adopted to big data analysis, processing and storage. The web user interface provides the user-friendly way to executing many kinds of jobs, operations and management. Besides, the installation of Hadoop ecosystem is time consuming and possible to make some error to normal users. We package the web user interface along with the whole Hadoop ecosystem into a virtual machine image file. It can be apply to different kinds of environments, make users deploy the whole system in the environment they want, simplify the process of building the system to save time and reduce the chance of error. The web user interface for this system is modular and allows users to modify or add the desired functions based on their needs by introduce Liferay Portal into our

system. We did some experiments to compare the performance between using the portal and the command line, Hadoop and Spark in our system.

1.3 Thesis Organization

In Chapter 2, we will describe the background information and previous studies related to our work, including big data, Hadoop ecosystem, the portal and the portlet, OpenCV, and the virtualization technology. In Chapter 3, the system architecture and the implementation are introduced. Chapter 4 shows the experiments done in our system, including the experimental environment and the experimental results. And we summarized our work and the future work of this thesis in Chapter 5.

Chapter 2

Background Review and Related Works

In this chapter, the background information related to our work, including big data, Hadoop ecosystem, portlet, OpenCV, and the virtualization technology are introduced.

2.1 Big Data

Big data means the data sets which are hard to be processed with traditional methods or tools owing to the large volume or the high complexity. Big data can be the data collected from sensors, log files generated while servers are running, or the user behavior recodes and posted information on the Internet.

The term was defined as the combination of 3Vs: Volume, Velocity and Variety [4]. These are the generic big data properties. Nowadays, the big data system definition is extended to the following 7Vs [5]:

- Volume: the amount of data which is generated and stored; The data quantity can easily reaches tens of TB today.
- Velocity: the speed of data generated and processed

- Variety: the types of data; The data can be unstructured or in various formats like text, picture, video, 3D model, and so on.
- Veracity: the accuracy of data; There may be incorrect parts or noises needing to be filtered in the data.
- Variability: the uncertainty of data; The similar or the same value of data may have different meaning.
- Visualization: the ways of presenting results of processing; The results should be easily understood by using charts, graphs or other visual presentations.
- Value: the usefulness or importance of the results got from the data; The value is actually what we want to get from doing many thing to data.

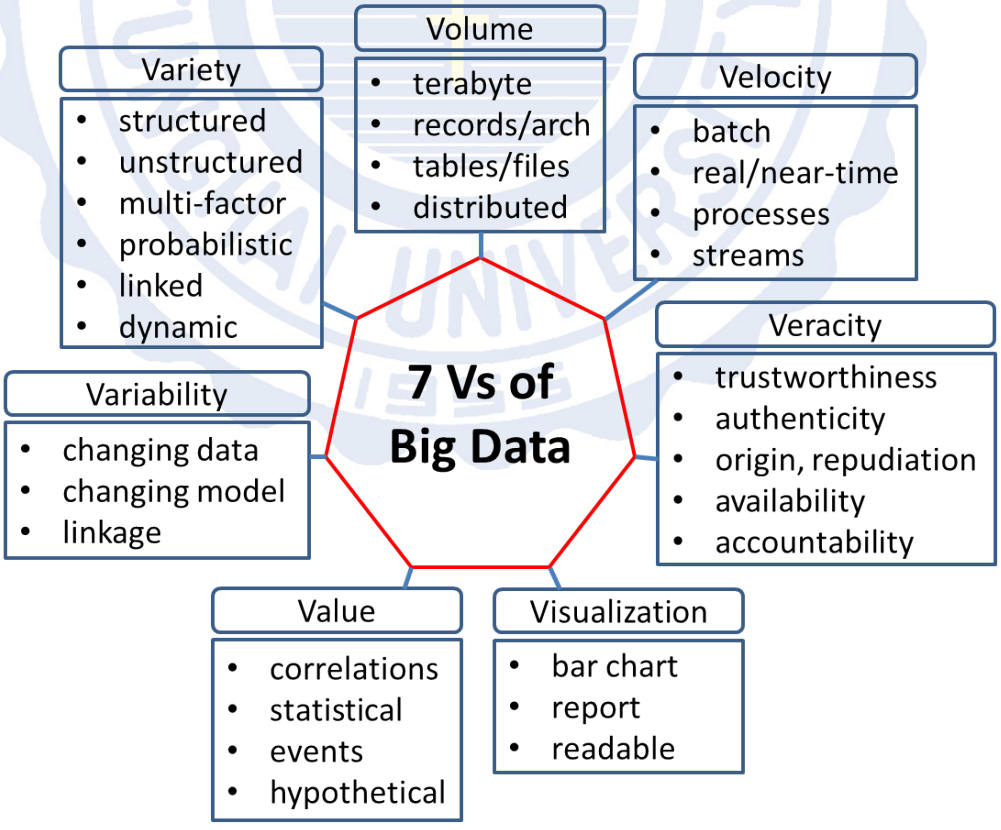

FIGURE 2.1: 7 Vs of big data [5]

2.2 Hadoop Ecosystem

2.2.1 Hadoop

Apache Hadoop [6] [7] is a open-source software framework for big data processing. Owing to its reliability, scalability and distributability, Hadoop can provide processing capacity by integrating the computational resource of the thousands nodes in the cluster. The implementation of Hadoop is based on two research results published by Google, Google Distributed System (DFS) in 2003 and MapReduce programming framework in 2004. The architecture of Hadoop is shown in Figure 2.2. The Hadoop framework is constructed on the Hadoop Distributed File System (HDFS) and it manage jobs and resource by YARN. The MapReduce [8] programming framework is implemented to operate the distributed processing by dividing the files into the same block size and distributing them to nodes [9]. Hadoop includes these modules:

- Hadoop Common: Java libraries and the common utilities;
- Hadoop Distributed File System (HDFS): a distributed file system of Hadoop;
- Hadoop YARN: a framework for job scheduling and cluster resource management;
- Hadoop MapReduce: a YARN-based programming model for big data processing.

2.2.2 HDFS

The Hadoop Distributed File System (HDFS) is a distributed file system that provides data storage with reliability, scalability and fault tolerance [10]. It is designed to be deployed on low-cost hardware. HDFS is suitable for big data applications because it provides high-throughput and streaming data access and can store data of different kinds of format. HDFS has master-slave architecture

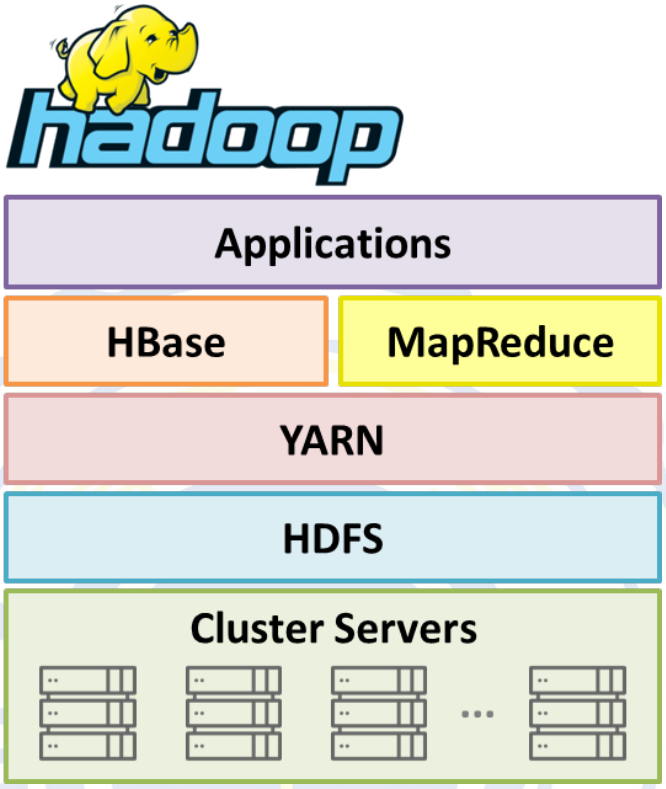

FIGURE 2.2: Hadoop architecture

including a single namenode and multiple datanodes. In the HDFS, data is divided into some blocks and distributed to nodes. If there are some datanodes down, HDFS can recovery the data with the backups on the other working datanodes [11]. Figure 2.3 shows the HDFS master-slave architecture.

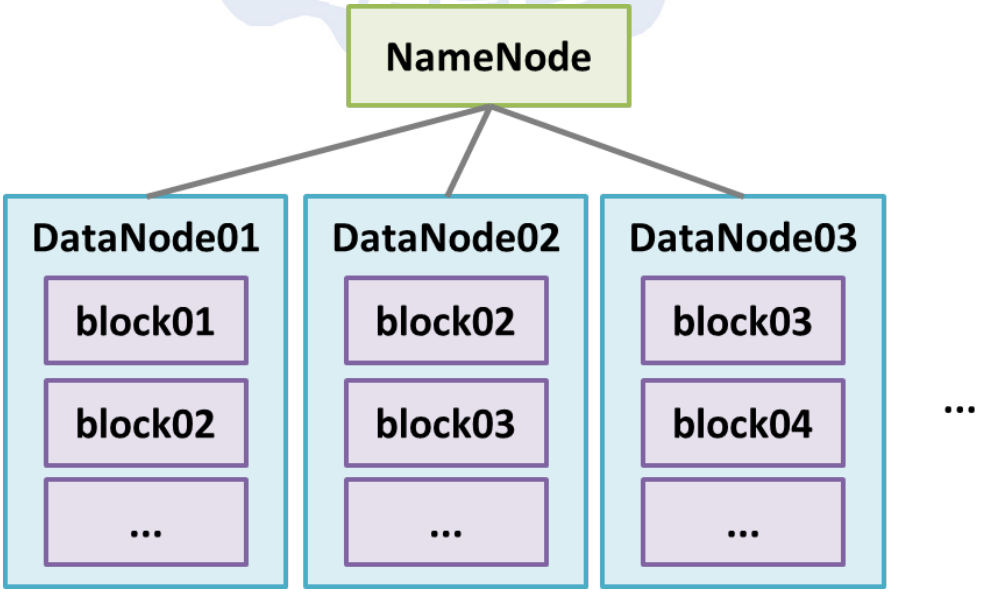

FIGURE 2.3: HDFS master-slave architecture

2.2.3 Spark

Apache Spark is an open-source cluster-computing framework. It was originally developed by AMPLab at University of California, Berkeley as a distributed data processing framework. The Spark project was donated to the Apache Software Foundation in 2013 and became a Top-Level Apache Project in 2014.

Spark provides the application programming interface (API) centered on the data abstraction called the resilient distributed dataset (RDD) distributed to the nodes in the cluster with fault tolerance for programmers. Spark is designed to improve the performance of MapReduce by offering the in-memory processing. The programs run with Spark can be up to 100 times faster than Hadoop MapReduce in memory or 10 times faster on disk. The features of RDD are conducive to the implementation of iterative algorithms, accessing datasets multiple time using loops, and analyzing data interactively. For machine learning systems, the iterative algorithms are the training methods. Thus Spark is the framework suitable for machine learning.

Spark requires a cluster manager and a distributed storage system for its operating environment. Spark supports three modes for cluster management: standalone, Hadoop YARN, and Apache Mesos. For distributed storage, Spark can link to various interfaces including Hadoop Distributed File System (HDFS), MapR File System (MapR-FS), Cassandra, OpenStack Swift, Amazon S3, and so on. The programming languages supported by Spark include Java, Scala, Python and R. Spark also can be run in pseudo-distributed local mode convenient for testing or development. In this mode, Spark runs all the applications on a single machine. The components of Apache Spark are listed below:

- Spark Core: the base engine providing the distributed task scheduling, basic input and output operations, and the RDD abstraction for the Spark platform;
- Spark SQL: a module supporting querying structured and semi-structured data by query languages;

- Spark Streaming: a extension of Spark Core providing the real time processing with scalability, high throughput and fault tolerance;
- MLlib: a scalable library of machine learning algorithms;
- GraphX: a graph computation engine supports the massively parallel algorithms.

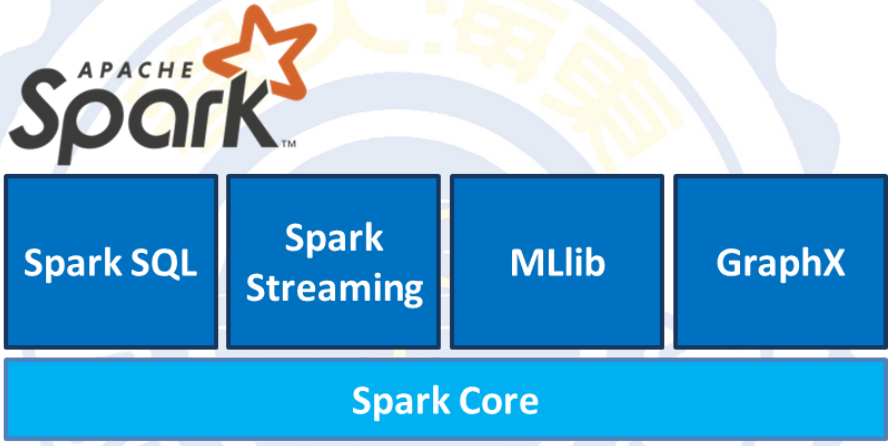

FIGURE 2.4: Spark architecture

2.3 Portal and Portlet

A portal [12] [13] is a specially designed website with specific contents and appearance. It gathers resource and information together from the system or other source and presents them to users on the single entry point with consistent way. For users, they can easily obtain resource and information and use the applications from one location.

A portal is consists of several portlets. A portlet is a pluggable user interface software component. It is based on Java, can be deployed, configured and displayed in the web container. A portlet can be regarded as a miniature web application. It is managed by portlet container, used to process the requirement from the container and generate contents dynamically. While initializing a configurable porlet, the permission of the portlet can be set so that confirming whether the

user can configure the portlet or not. Portlets can be set to transmit information to each other. Some examples of portlet applications are e-mail, weather reports, discussion forums, and news. Portlets are useless without a portal because they need it to be deployed on.

One of the main jobs of a portal is to gather the contents generated by portlets. Multiple portlets can be displayed on a portal. Users can choose which portlets they want to see and customize the layout. A portal provides the identity verification, the authorization and the management tools for the system administrators. Figure 2.5 shows the example of portal and portlet layout.

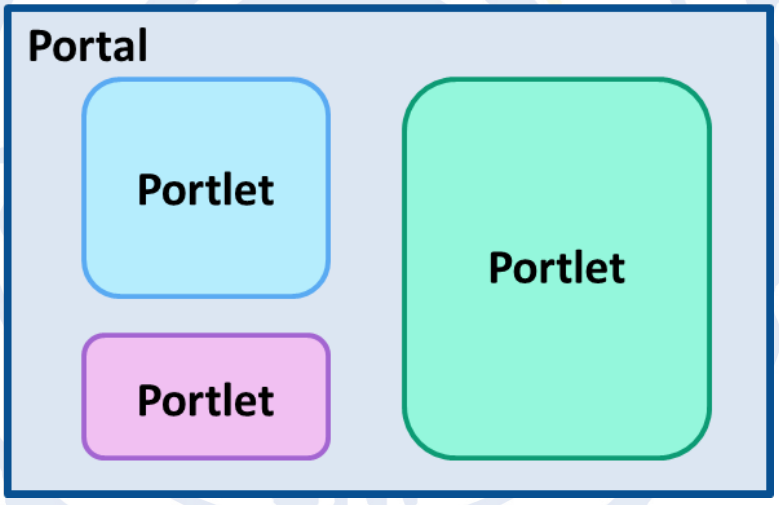

FIGURE 2.5: Example of portal and portlet layout

Liferay Portal is Java-based web. It is convenient for Java programmers because Hadoop and Spark also use Java programming language. The programmer can be concentrate on Java programming and develops the Hadoop applications and portlets. If necessary, HTML, CSS and JavaScript also can be used to develop the portlets. Liferay Portal provides functions not only those mentioned above but some additional components. Figure 2.6 shows the functions of Liferay Portal.

2.4 OpenCV

OpenCV (Open Source Computer Vision Library) [15] [16] is an open-source and cross-platform, computer vision and machine learning software library. It was

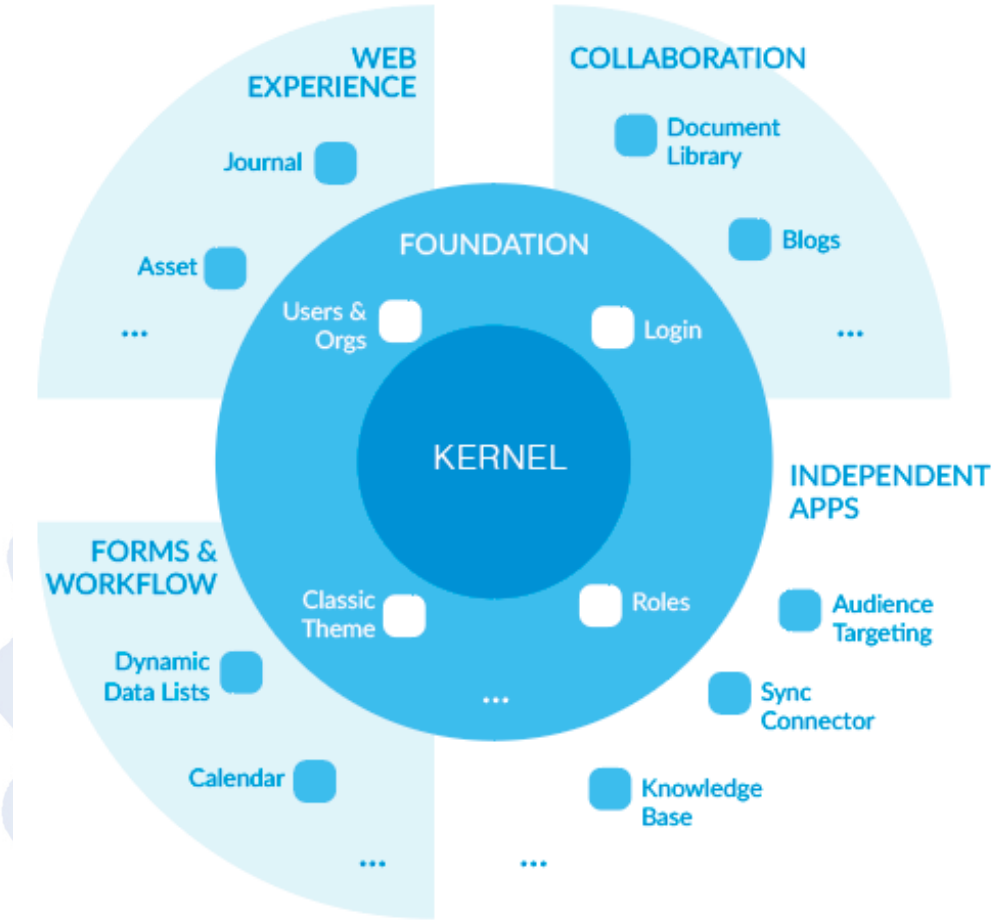

FIGURE 2.6: Funcitons of Liferay Portal [14]

started by Intel and is released under a BSD license so that it is free for commercial and research purposes. OpenCV was designed to provide a common infrastructure for computer vision applications and accelerate the use of machine perception in the commercial products.

OpenCV is the leading open-source library for computer vision, image processing and machine learning. It was designed for computational efficiency and real-time applications. OpenCV was written in optimized C and C++, and it can be linked to C, C++, Python, Java and MATLAB. The operating systems supported by OpenCV include Windows, Linux, Mac OS, Android, iOS, and so on. OpenCV also supports the multi-core operations.

OpenCV is used widely in the field of image processing, image and video

reading and saving, matrix operations, statistics, automated inspection and monitoring, robot and driver-less car navigation and control, medical image analysis, image and video search and retrieval, and so on.

OpenCV has a modular structure. The following are some modules OpenCV provides [17]:

- Core functionality: a compact module defining basic data structures;
- Image processing: an image processing module that includes linear and non-linear image filtering, geometrical image transformations, color space conversion, histograms, and so on;
- video: a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms;
- calib3d: basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction;
- features2d: salient feature detectors, descriptors, and descriptor matchers;
- objdetect: detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on);
- highgui: an easy-to-use interface to simple UI capabilities;
- Video I/O: an easy-to-use interface to video capturing and video codecs;
- gpu: GPU-accelerated algorithms from different OpenCV modules.

OpenCV is adopted by many companies, research organizations, and government organs like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota. OpenCV is maintained by a non-profit foundation OpenCV.org.

2.5 Virtualization

Virtualization [18] [19] is the technology that can abstract, manage, and redistribute the computing resources like CPU, memory, storage, network, applications, and so on. Virtualization can make the computing resources more flexible and scalable, and reduce the costs. In this work, we use the virtual machine (VM) an application of the virtualization technology to build our system in it. We can configure the computing resource of a virtual machine and package it into a image file easy to be move or copy to another environment. A virtual machine contain the whole operating system and applications in it. The resource of a virtual machine can be reconfigured according to the resource in the environment. Figure 2.7 shows the architecture of virtualization.

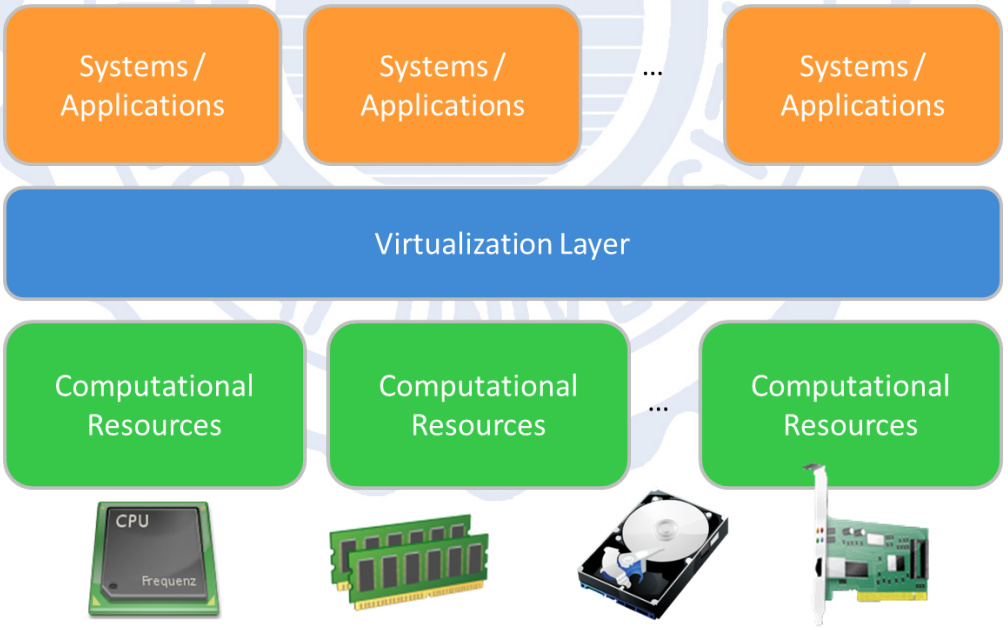

FIGURE 2.7: Virtualization

Another kind of virtualization is desktop virtualization. Users can use the remote virtual desktop environment via network. That can share the computing resources of the server to multiple users at the same time. And the computing resources are centralized and easy to be managed.

2.6 Related Works

Xiuqin Lin et al. [20] in 2013, Ilias Mavridis and Helen Karatza [21] in 2017 all successfully applied Hadoop and Spark to analyze Log. For big data analysis, Hadoop is a good solution; the Apache Hadoop is built on top of HDFS. Hadoop provides an off-line batch computing framework. Spark is compatible with Hadoop HDFS; Spark uses in-memory and distributed memory technology, and it allows repeatedly operations of cache data in memory since Spark in-memory primitives provide performance up to 100 times faster for certain applications. Spark has a higher speed than Hadoop in processing capability, and access to data on YARN. According to the experimental results in the works, we compare the performance between Hadoop and Spark.

Chien-Heng Wu et .al [22] in 2016 studied and used the virtualization technology to develop their small size personal big data platform for developers and made the platform easy to be deployed. They performed the comparison between the virtual machine and their large-scale Hadoop cluster. The result of the work showed the virtual machine is an ideal platform for development and the large-scale Hadoop cluster is great for production runs. According to the result in that work, we build not only the Hadoop ecosystem but also the OpenCV library and Liferay Portal on it. We want to make the platform more easy to be used. And we performed some test on the platform.

Shengsheng Huang et .al [23] in 2010 proposed the realistic and comprehensive benchmark suite for Hadoop called HiBench. And they used that to evaluate the Hadoop framework including the speed, throughput, HDFS bandwidth, CPU, memory, disk I/O usage. And the benchmark suite has the ability to export the graphical charts. HiBench supports used by evaluate the Hadoop and Spark now. It includes different kinds of workloads, like sort, wordcount, TeraSort, machine learning, SQL, web search, graphic benchmark, streaming benchmark, and so on. In our work, we use HiBench to evaluate the performance of our Hadoop ecosystem and compare the performance between Hadoop and Spark.

Timofei Epanchintsev and Andrey Sozykin [24] in 2015 presented the approach to use OpenCV library for distributed image processing on a Hadoop cluster. That is based on the MapReduce Image Processing (MIPr) framework they developed before. The performance and the scalability of that framework were evaluated. We refer to the concept of that work, build the OpenCV library and execute the application to test if OpenCV works in our system.

T. Lakshmi Siva Rama Krishna et .al [25] in 2016 proposed the customized web user interface (CWBUI) for Hadoop Distributed File System that users can perform file system operations to and from HDFS easily by clicking the buttons displayed on the screen rather than using shell commands. They built a Hadoop cluster with ten nodes and modified the original web user interface of Hadoop Distributed File System. They used servlets and Java Server Pages (JSP) to develop the customized web user interface. In our system, we use Liferay Portal server and develop portlets for our web user interface. And the functions of our web user interface include not only performing file system operations but also running Hadoop and other applications.

Chapter 3

System Design and Implementation

The architecture and the implementation of our system are introduced in this chapter. Our system is based on Hadoop ecosystem and packaged into virtual machine images along with the web user interface.

3.1 System Design

The bottom layer of the software part of our system is the hypervisor of the virtaul machine, Oracle Virtualbox and VMware Workstation are tested in this work. And the Ubuntu desktop operating system is installed on the hypervisor. Then the Hadoop ecosystem that includes HDFS, Yarn, ZooKeeper and Spark is built in Ubuntu. The Hadoop applications and the Liferay Portal are based on Hadoop and Liferay server. Users can execute big data jobs through the portal easily. Figure 3.1 shows the architecture of our system. The following are the portlets on the Liferay Portal web user interface in our system:

- job submission: executing Hadoop applications with given jar file and required arguments;

- file upload: uploading the given file to the destination path on Hadoop Distributed File System;
- sequential file packaging: packaging the given files like images into a sequential file and uploading it to the destination path in Hadoop Distributed File System;
- file management: presenting the files and directories on Hadoop Distributed File System.

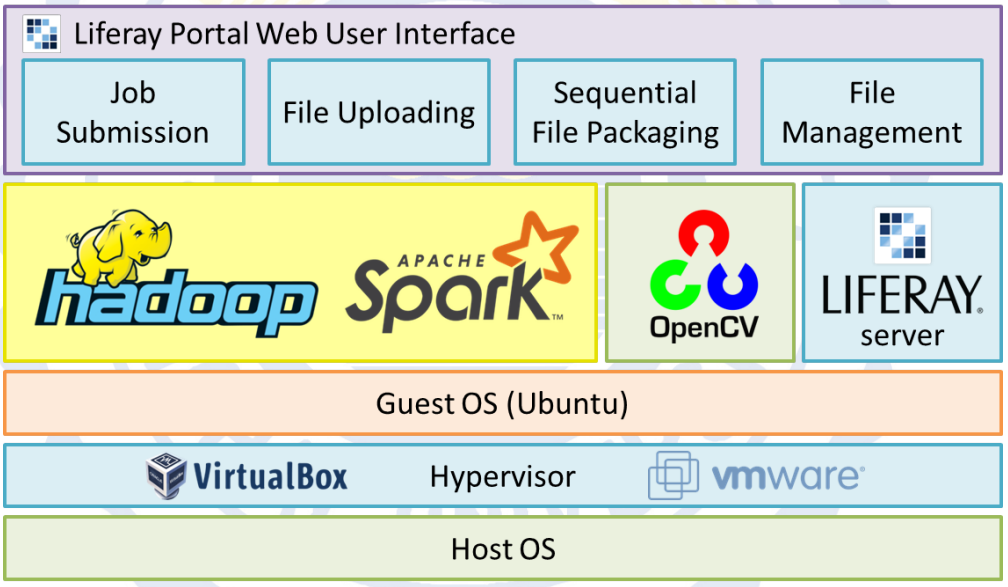

FIGURE 3.1: System architecture

While a user executes jobs through the portlets on the Liferay Portal web user interface, the portlets can not only communicate with each other but also pass the commands to the Hadoop ecosystem, the OpenCV library and the operating system and receive the returned information to run the applications. The interactions of the components in the system are shown in Figure 3.2.

3.2 System Implementation

At the first, we must to check if the virtualization technology, like Intel VT-x or AMD-V, is supported by the CPU of the computer. That can be confirmed by

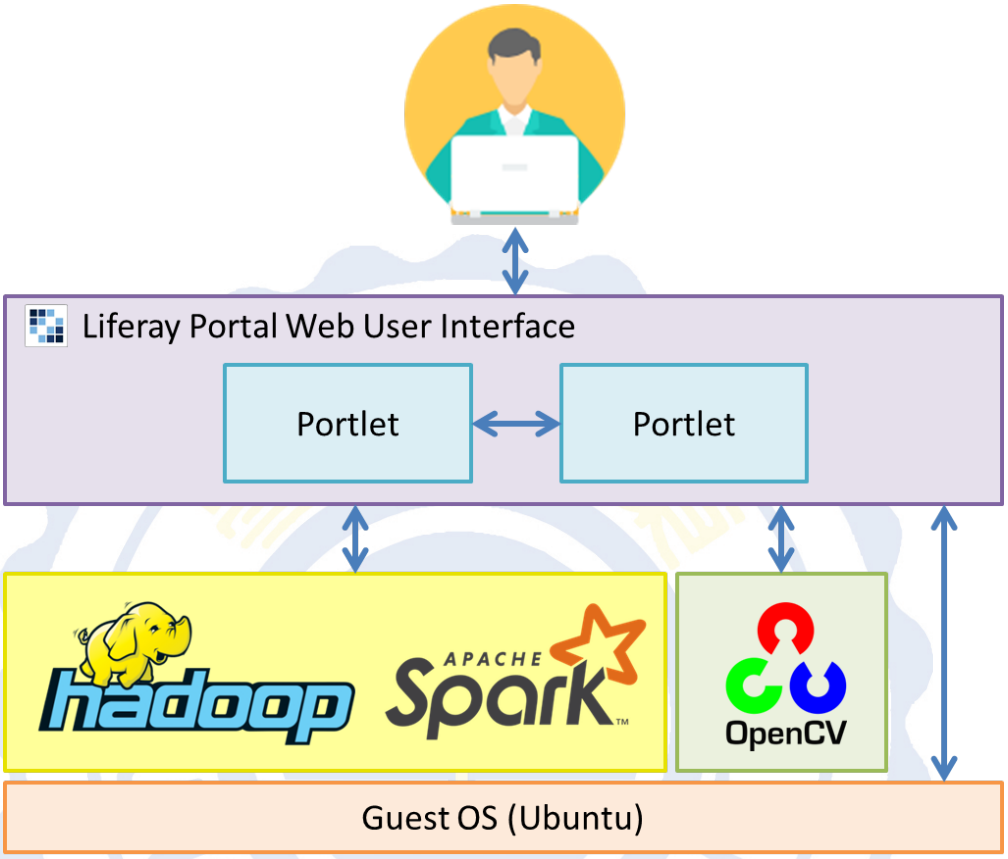

FIGURE 3.2: System interactions

configuring the BIOS settings or using the specific software like Intel Processor Identification Utility [26] shown in Figure 3.3.

3.2.1 Virtualization Platform Installation and Virtual Machine Setup

After the virtualization technology support was confirmed, we install the virtualization platform and its extension component: Oracle Virtualbox with Extension Pack and VMware Workstation with VMtools. The extension components are used to enhance the supportability for the resource of the physical machine. The timings of installing the extension components are different depending on the vendors. Oracle Virtualbox Extension Pack can be installed before adding a virtual machine into the repertory but VMware Workstation VMtools only can be configured after the operating system of a virtual machine is installed.

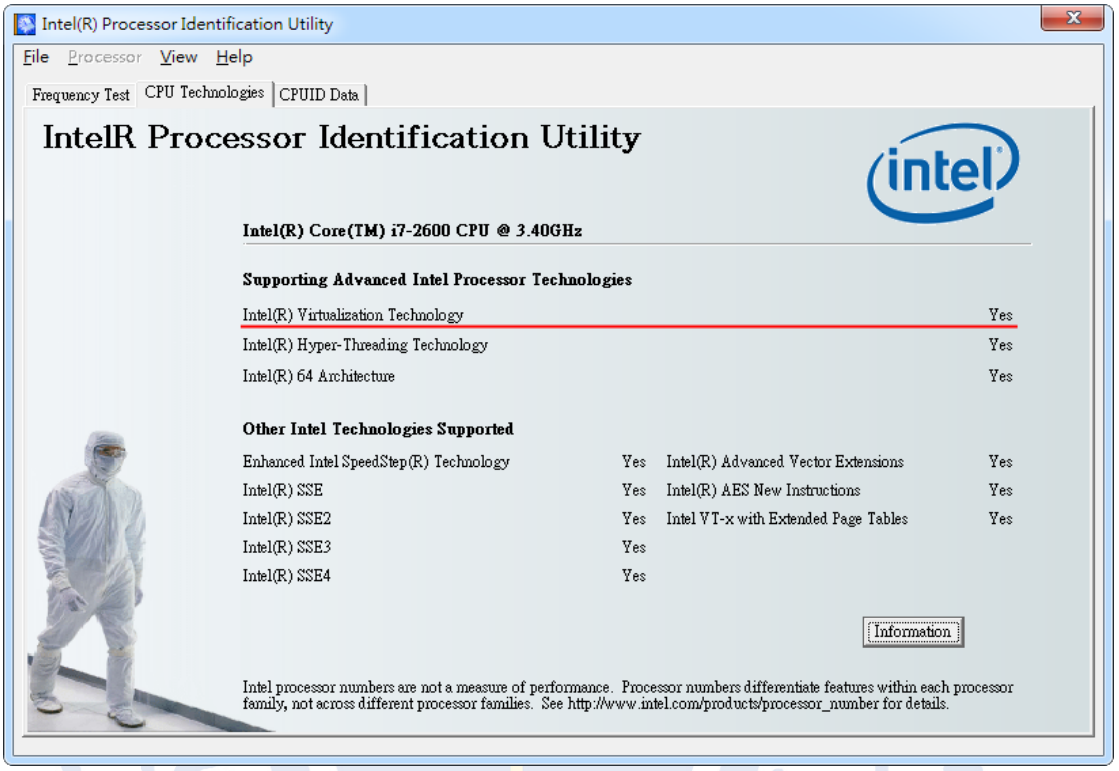

FIGURE 3.3: Intel Processor Identification Utility [26]

While the virtualization platform is done, we can add a new virtual machine and configure the resource for it, such as the number of the cores of the virtual CPU, the size of the memory, the capacity of the virtual disk, the guest operating system, the network settings, the display and so on. Figure 3.4 shows the process of creating a new virtual machine on Oracle Virtualbox.

3.2.2 Hadoop Ecosystem Installation

After booting up the virtual machine and the installation is finished, we install the Java Development Kit first and set the SSH key for Hadoop. Then we build Hadoop, Yarn, HDFS, HBase, Spark, Zookeeper and the environment parameters must be set. Figure 3.5, 3.6 and 3.7 show the status of Hadoop and Spark after the services has been started successfully. In this work, we build the Hadoop ecosystem with the Cloudera CDH 5.5.1 package [27].

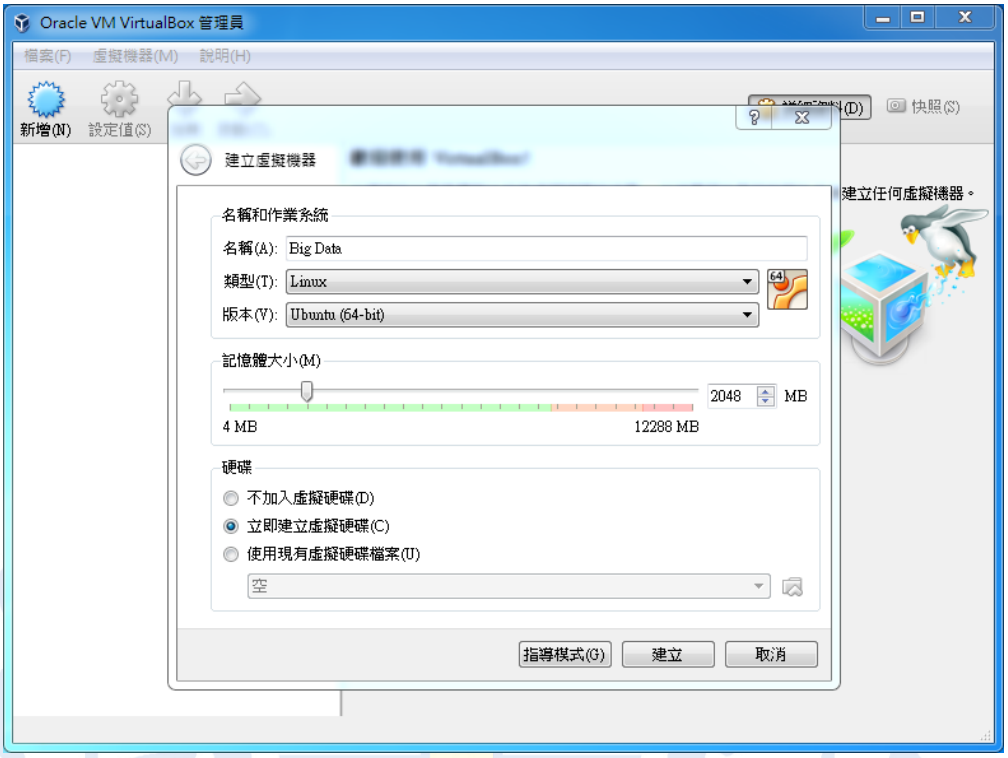

FIGURE 3.4: New virtual machine creation

3.2.3 Liferay Portal server and Integrated Development Environment Installation

Next, the Liferay Portal bundle with Tomcat server is required to be downloaded. After unzipping the package and running the installation batch file, the Liferay server will be installed and started up in several minutes. The following are Eclipse, Liferay IDE plugin for Eclipse and Liferay server. The Liferay development environment is combined with Eclipse, the server can be easily controlled and portlets can be developed through Eclipse. Figure 3.8 shows the Liferay IDE integrated with Eclipse.

3.2.4 OpenCV Environment Setup

The last step is building and compiling the OpenCV library [28]. After downloading and unzipping the OpenCV library package, installing the required software

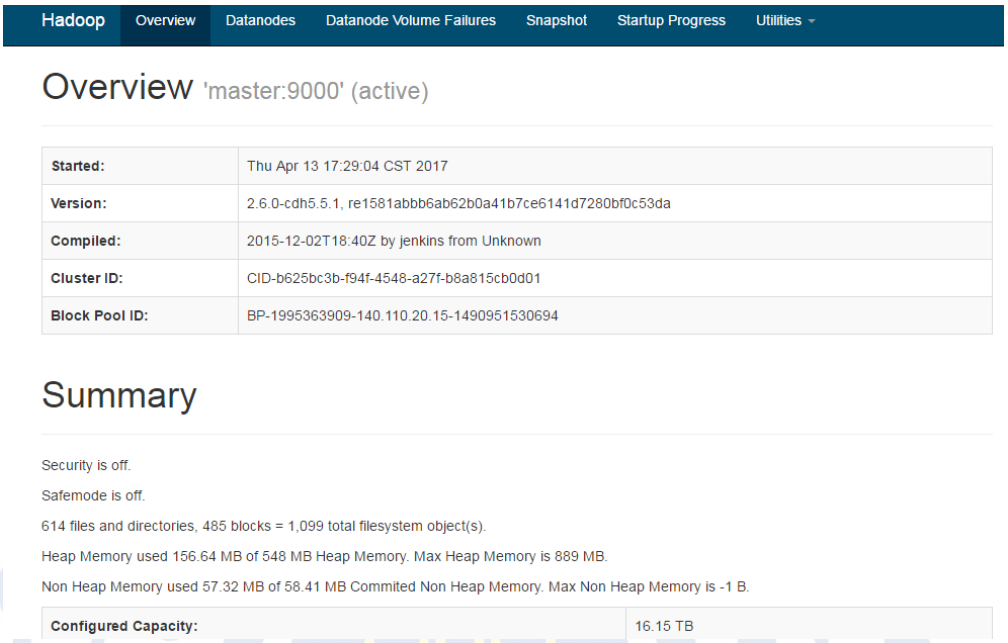

FIGURE 3.5: Hadoop status

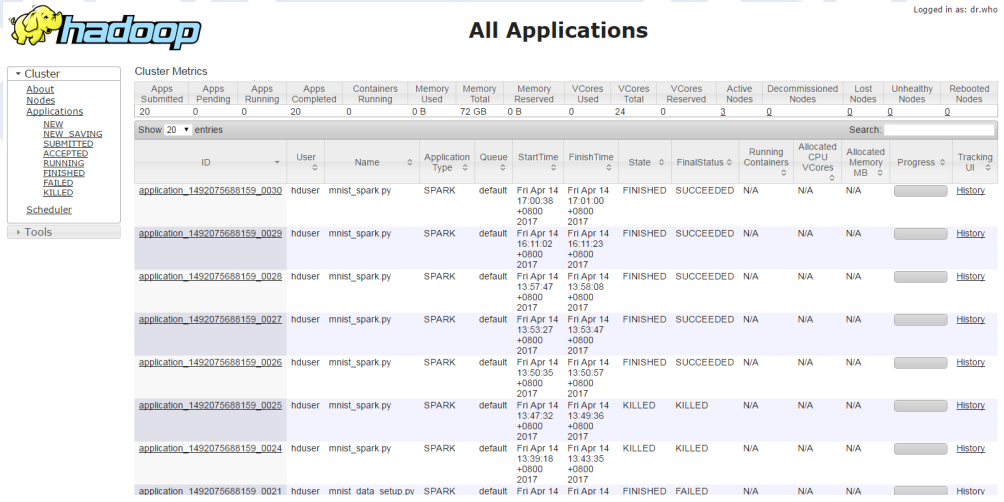

FIGURE 3.6: Hadoop cluster status

packages and building the OpenCV library, the OpenCV library files must be generated if OpenCV is installed successfully. The installation result of OpenCV is shown in Figure 3.9. So far, the related environment is built and can be used to develop the Hadoop, Spark, portlet, and OpenCV applications.

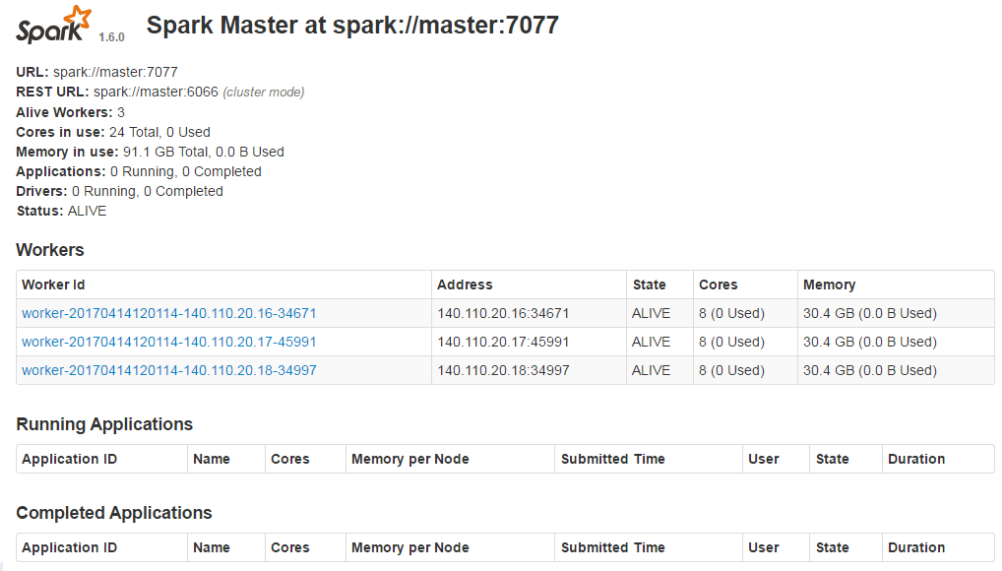

FIGURE 3.7: Spark status

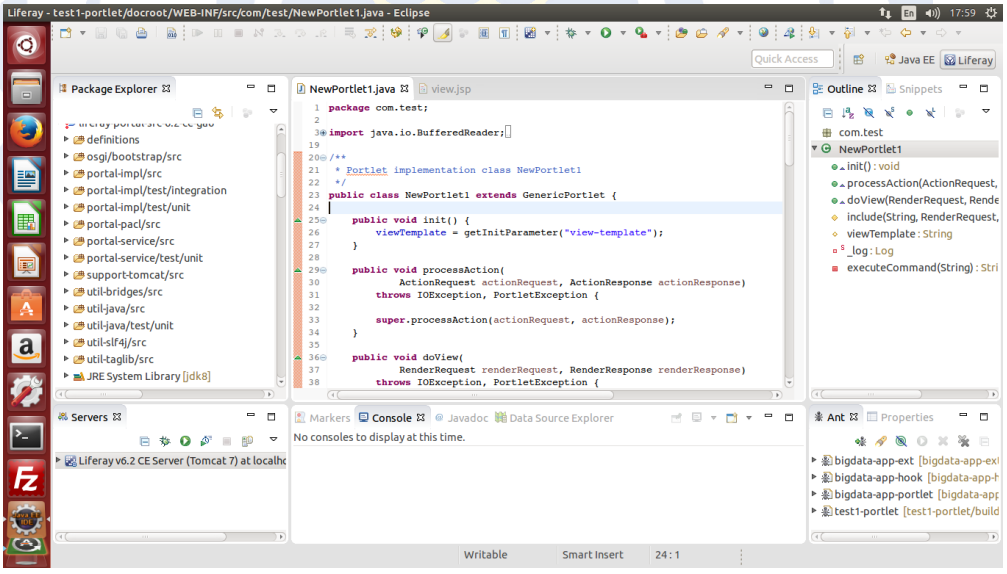

FIGURE 3.8: Liferay IDE

3.2.5 Liferay Portal, Portlet Development and Virtual Machine Image File Export

Liferay Portal is Java-based web. The portlets are developed in Java programming language. If necessary, HTML, CSS and JavaScript also can be used to develop the portlets. We implemented portlets that can perform Hadoop and Spark operations including jar file execution, file uploading, file management and sequential file packaging. The portlets are modular so they can be add into or remove from the

```
perf/perf_matop.cpp.o
[100%] Linking CXX executable ../../bin/opencv_perf_stitching
[100%] Built target opencv_perf_stitching
[100%] Building CXX object modules/gpu/CMakeFiles/opencv_perf_gpu.dir/
perf/perf_video.cpp.o
[100%] Building CXX object modules/gpu/CMakeFiles/opencv_perf_gpu.dir/
perf/perf_precomp.cpp.o
[100%] Linking CXX executable ../../bin/opencv_perf_gpu
[100%] Built target opencv_perf_gpu
root@master:/opt/opencv-2.4.4#
```

FIGURE 3.9: Installation result of OpenCV

Liferay Portal web page by the user. The Liferay Portal web page we implemented is shown in Figure 3.10.

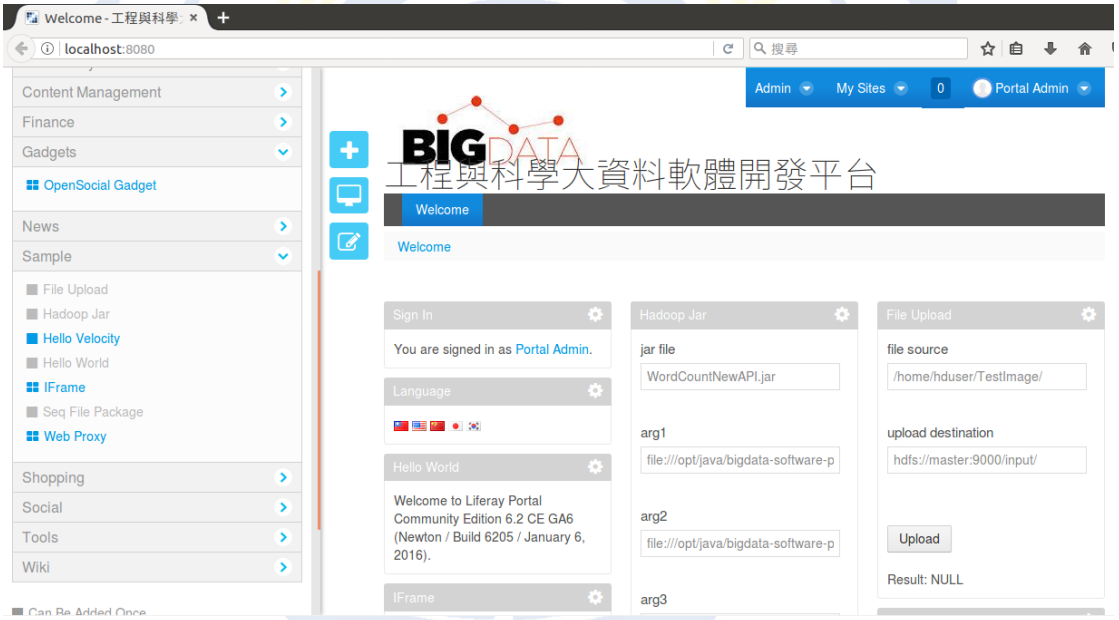

FIGURE 3.10: Liferay Porlet web UI

We package the whole environment including the Ubuntu desktop operating system, the Hadoop ecosystem and the Liferay Portal web user interface into a virtual machine image file. Figure 3.11 shows the process of virtual machine export. The virtual machine image file can be deployed on Oracle Virtualbox and VMware Workstation, and the computing resource of the virtual machine can be configured according to the environment which the virtual machine be deployed in. Figure 3.12 and 3.13 show the results of the deployment.

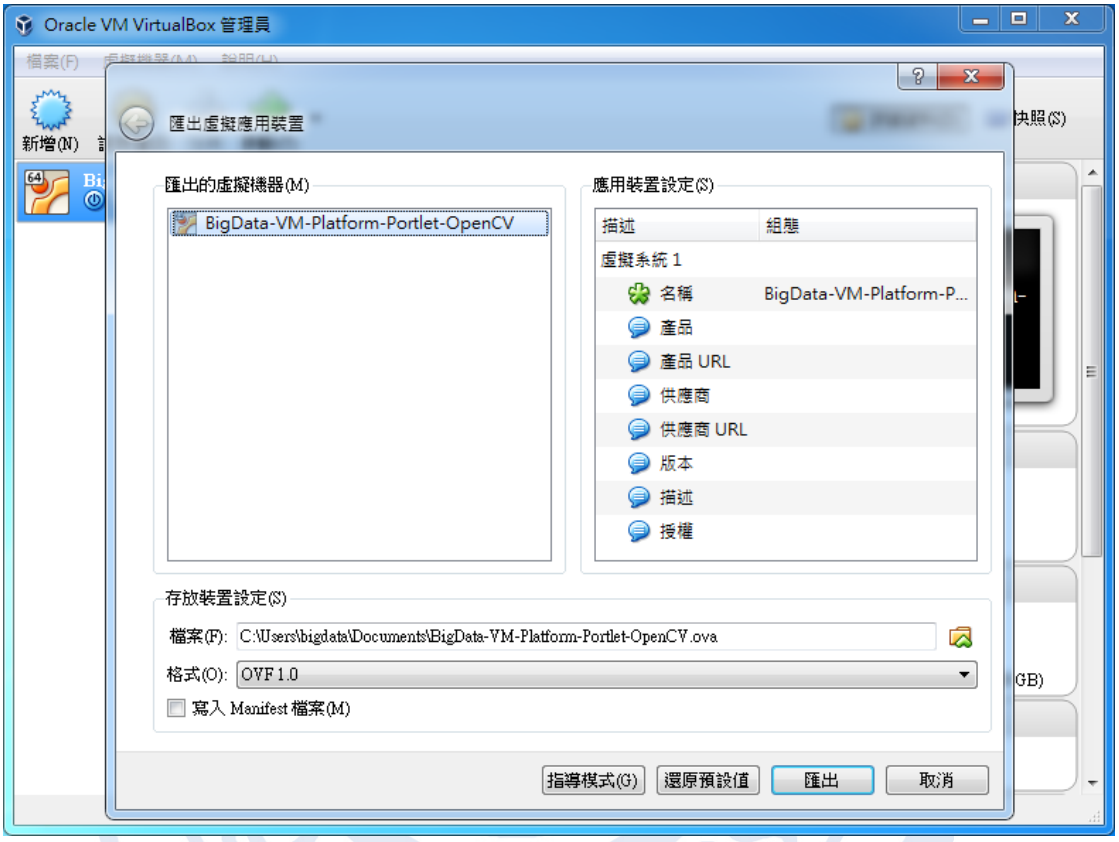

FIGURE 3.11: Virtual machine export

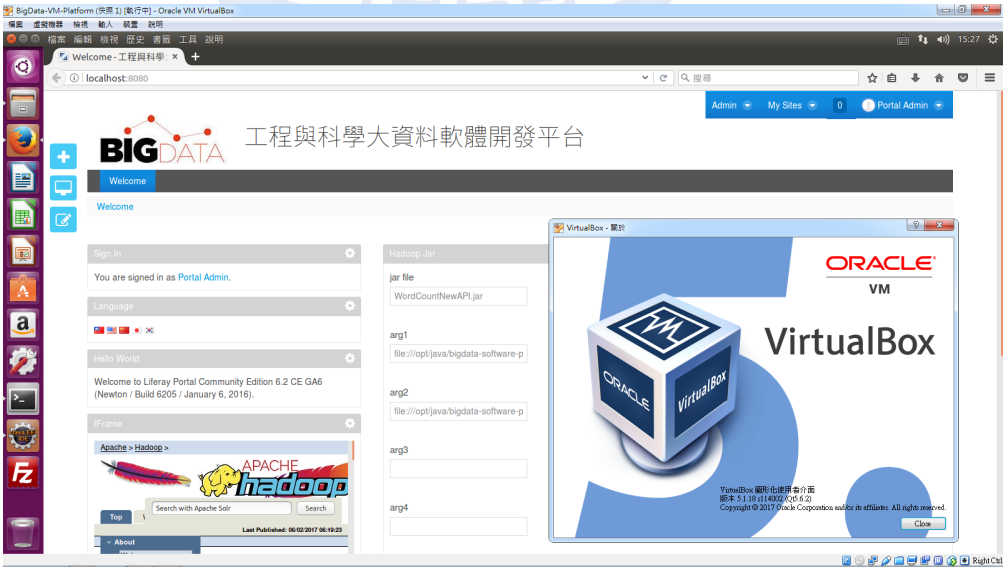

FIGURE 3.12: Virtual machine in Oracle Virtualbox

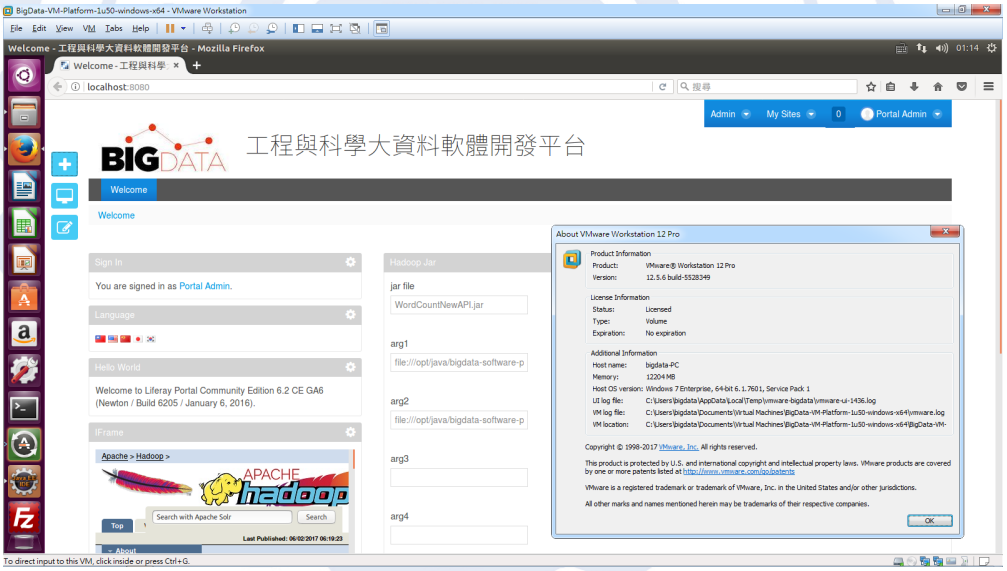

FIGURE 3.13: Virtual machine in VMware Workstation

Chapter 4

Experimental Results

In this chapter, we show the system and experimental results. In Section 4.1, we introduce the experimental environment, including the hardware specification and the software information. Our experimental results are presented in Section 4.2 in detail including the virtual machine deployment in different virtualization environments, functionality validation of the portlets, performance comparison between using the portlets on the web user interface and command line, performance comparison between Hadoop and Spark, and the last one OpenCV environment validation.

4.1 Experimental Environment

In this work, we perform the experiments on a desktop personal computer to simulate the user operating environment. The hardware specification is shown in Table 4.1. The operating system on the computer is Microsoft Windows 7 SP1, and the one in our virtaul machine is Ubuntu Desktop 16.04 LTS. The virtualization platforms are Oracle Virtualbox and VMware Workstation. The software includes Hadoop ecosystem, Liferay Portal, OpenCV and the benchmark suite HiBench. The detail software versions are shown in Table 4.4.

TABLE 4.1: Personal computer hardware specification

CPU	Intel Core i7-2600 (3.4GHz)
Memory	DDR3 RAM 12 GB
Graphics	Intel HD Graphics 2000
Hard Disk	1TB SATA III Hard Disk

TABLE 4.2: Laptop hardware specification

CPU	Intel Core i7-3632QM (2.2GHz)
Memory	DDR3 RAM 8 GB
Graphics	AMD Radeon HD 7500M/7600M Series
Hard Disk	128GB SATA III SSD
External Hard Drive	500GB HDD (SATA to USB 3.0)

TABLE 4.3: Configurations of virtual machine

vCPU	4 cores
vRAM	8 GB
vHDD	48 GB

TABLE 4.4: Software information

Software Name	**Version**
Oracle Virtualbox	5.1.18
VMware Workstation	12.5.6
Microsoft Windows	7 SP1
Ubuntu	16.04 LTS
Hadoop	2.6.0 (CDH 5.5.1)
Spark	1.6.0
ZooKeeper	3.4.5 (CDH 5.5.1)
HBase	1.0.0 (CDH 5.5.1)
Liferay IDE	2.2.4 GA5
Liferay Portal	6.2.5 GA6
OpenCV	2.4.4
HiBench	6.0

4.2 Experimental Results

4.2.1 Virtual Machine Deployment in Different Virtualization Environments

We deploy the virtual machines on Windows using Oracle Virtualbox and VMware Workstation, then record each the time they took. The process of Virtual Machine image file import on VMware Workstation is shown in Figure 4.1 and the one on Oracle Virtualbox is shown in Figure 4.2.

The size of our virtual machine image file is about 7.73 GB. The each average time of virtual machine image import is shown in Table 4.5 and 4.6. There are two part of this experiment. One is using the desktop personal computer and the other is using a laptop with an external hard drive. To simulate the scenario of making the virtual machine portable, we put the virtual machine image file in the external hard drive and connect it to the laptop. And the configurations of the two virtualization platform of this part are set to store the vHDD on the external hard drive.

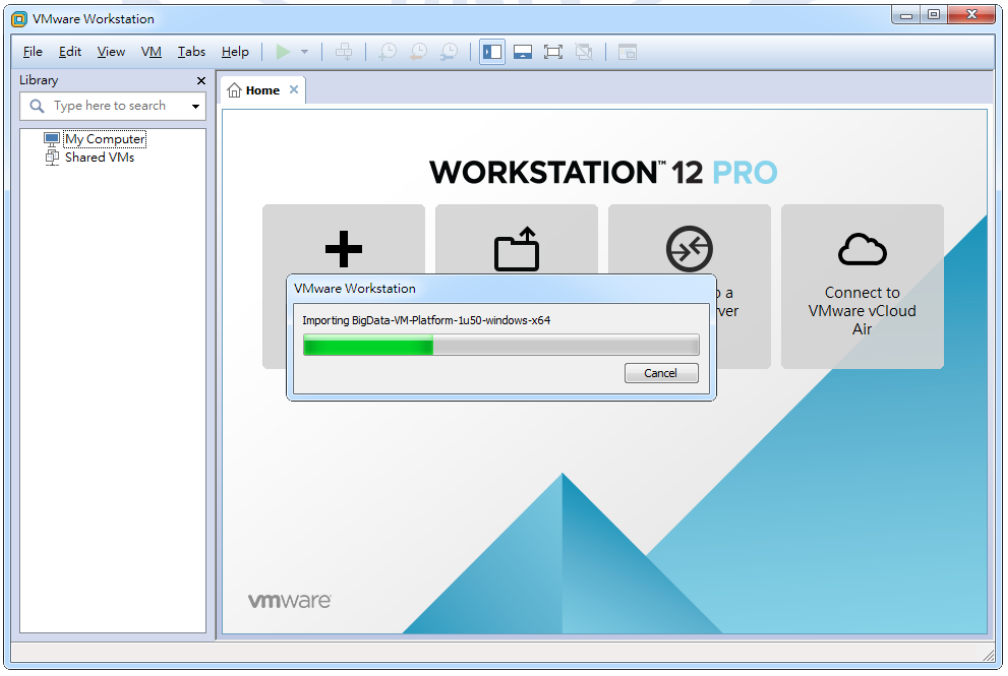

FIGURE 4.1: Process of VM image import on VMware Workstation

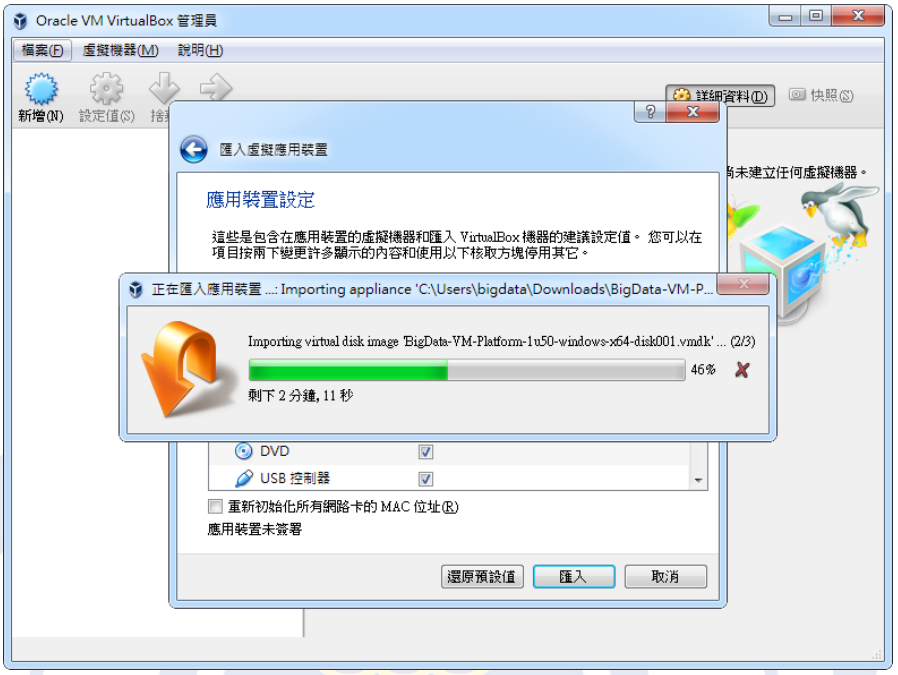

FIGURE 4.2: Process of VM image import on Oracle Virtualbox

TABLE 4.5: Average time of VM image import on PC

Hypervisor	Time of VM image import
Oracle Virtualbox (5.1.18)	~173 s
VMware Workstation (12.5.6)	~338 s

TABLE 4.6: Average time of VM image import on laptop

Hypervisor	Time of VM image import
Oracle Virtualbox (5.1.18)	~755 s
VMware Workstation (12.5.6)	~1363 s

The time importing the virtual machine image took is less than 10 minutes. It is faster than installing the operating system and all the Hadoop software by users themselves. The process of image file import is much easier than building the whole system. And there is no need to keep concentrating on the process of import. That means the virtual machine can help simplifying the process of building the system and saving time.

4.2.2 Functionality Validation of Portlets

In this experiment, we use the portlets on the web user interface to execute a wordcount job to demonstrate our function of the portlets we developed.

First, we upload our sample text file to the HDFS by filling in the source file full path and the upload destination field, clicking the upload button and waiting for the result. Figure 4.3 shows the HDFS file upload portlet. Then we can check the directory we uploaded a file to by the HDFS browser shown in Figure 4.4.

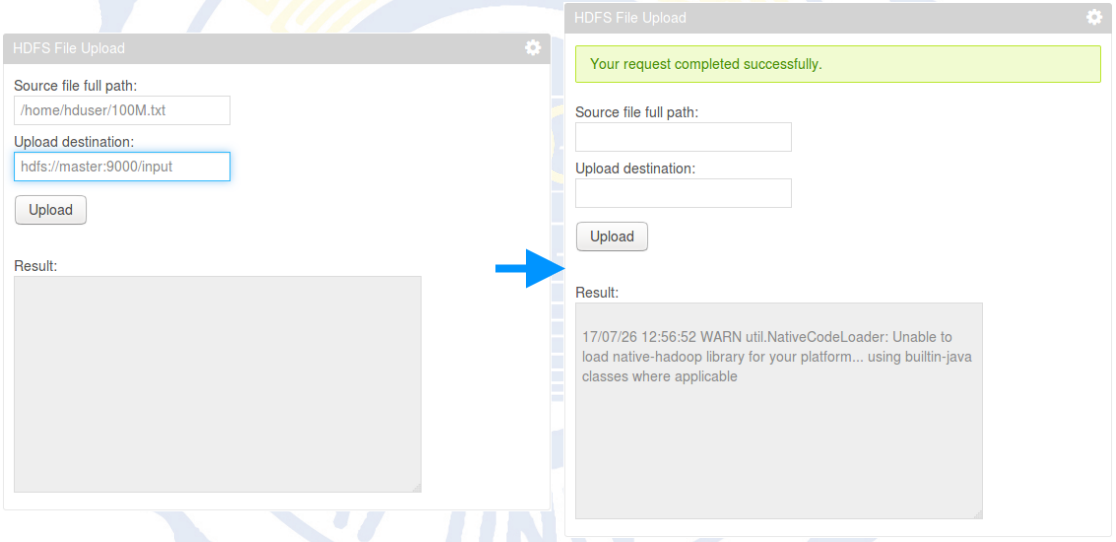

FIGURE 4.3: Portlet: HDFS file upload

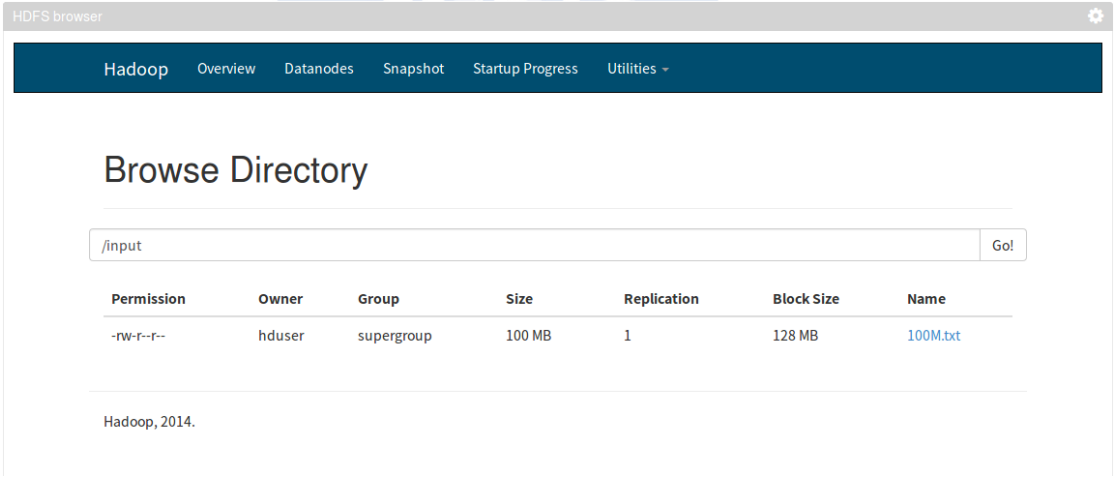

FIGURE 4.4: HDFS browser (upload destination)

Second, the full path of the wordcount jar file, input text file and output destination must to be set. The execution result is shown in the text field below

after clicking the submission button and waiting for execution finished. Figure 4.5 is the portlet of Hadoop job submission. We also can check the output directory we set through the HDFS browser shown in Figure 4.6. to upload our sample text file to the HDFS, browse the directory on the HDFS, execute the wordcount jar application on Hadoop, and check the output result.

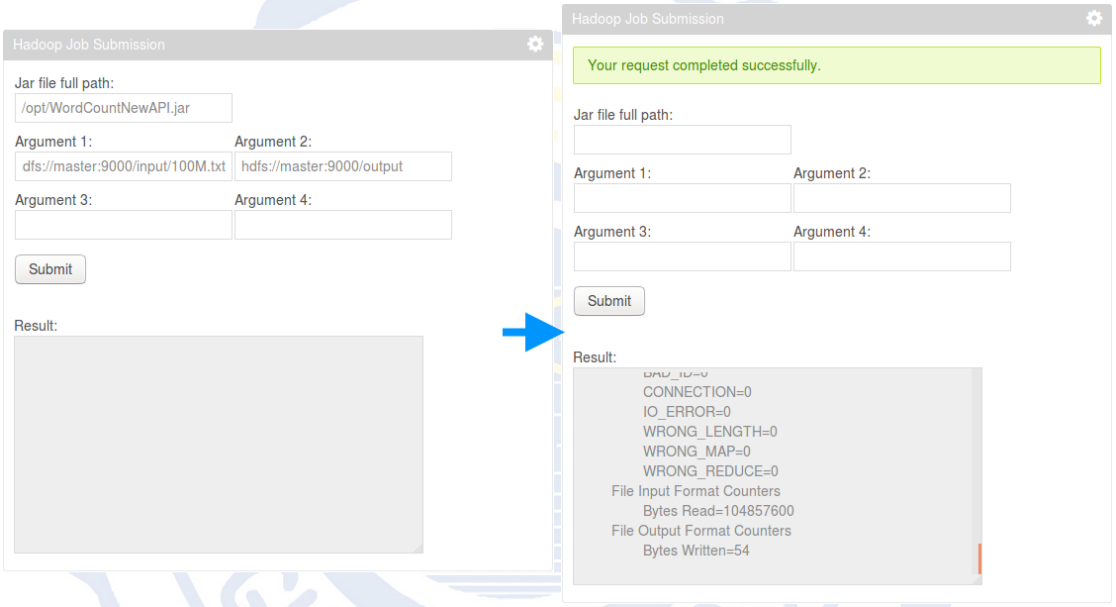

FIGURE 4.5: Portlet: Hadoop job submission

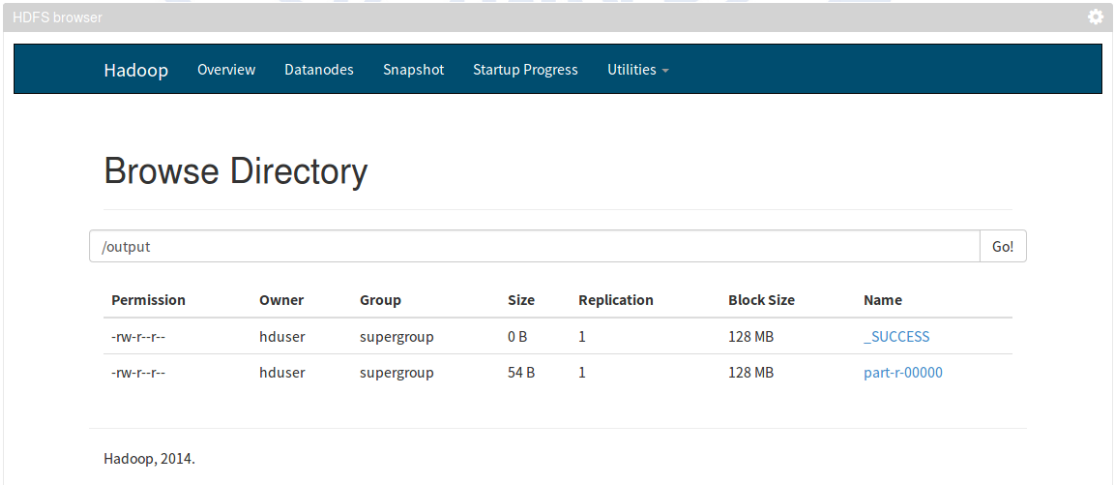

FIGURE 4.6: HDFS browser (output directory)

Last, we can check the output file on the HDFS by downloading it through the HDFS browser or using the portlet of command execution shown in Figure 4.7 to show the content of the output file.

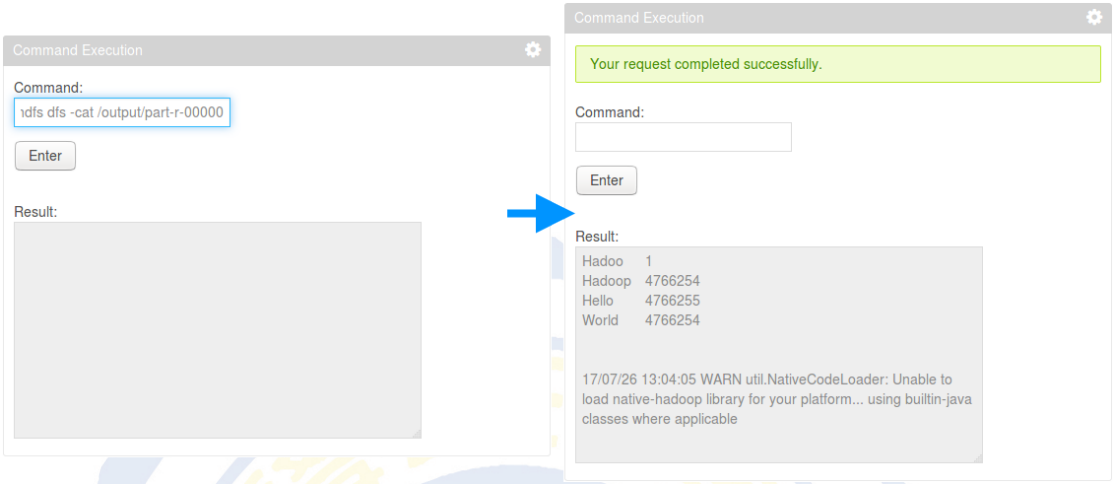

FIGURE 4.7: Portlet: Command excution

4.2.3 Performance Comparison between the Portal and the Command Line

The following are experiments of performance comparison between using the command line and the portal web user interface. We use the Hadoop and Spark benchmark suite — HiBench to evaluate the performance of our platform. We run the three kinds of workloads, sorting, TeraSort and wordcount. Figure 4.8 shows the result of building HiBench and Figure 4.9 is the process of running HiBench.

```
hduser@master: ~/HiBench-master
[INFO] ------------------------------------------------------------------------
[INFO] Building hadoopbench 6.1-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ hadoopbench ---
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO]
[INFO] hibench ............................................ SUCCESS [ 12.179 s]
[INFO] hibench-common ..................................... SUCCESS [05:40 min]
[INFO] HiBench data generation tools ...................... SUCCESS [ 33.423 s]
[INFO] sparkbench ......................................... SUCCESS [  0.007 s]
[INFO] sparkbench-common .................................. SUCCESS [01:09 min]
[INFO] sparkbench micro benchmark ......................... SUCCESS [  6.097 s]
[INFO] sparkbench project assembly ........................ SUCCESS [  4.937 s]
[INFO] hadoopbench ........................................ SUCCESS [  0.002 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 07:46 min
[INFO] Finished at: 2017-06-21T23:42:53+08:00
[INFO] Final Memory: 57M/789M
[INFO] ------------------------------------------------------------------------
hduser@master:~/HiBench-master$
```

FIGURE 4.8: Result of building HiBench

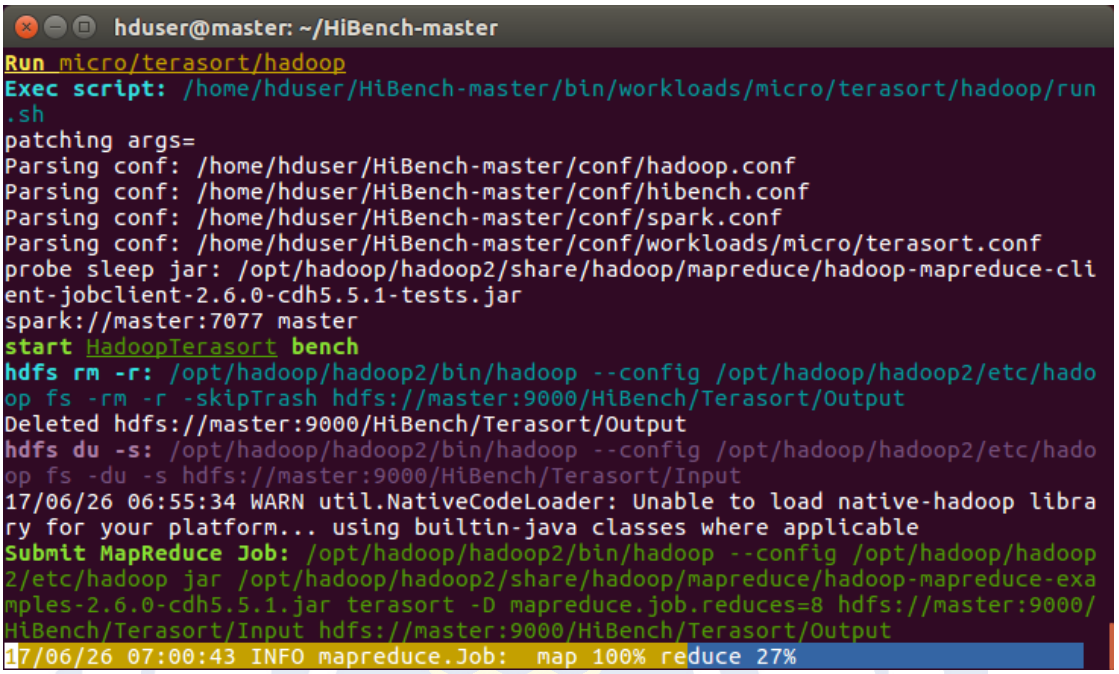

FIGURE 4.9: Process of running HiBench

In theory, the execution time of using command line should be better than or equivalent to the ones of using the portal interface. We run the benchmark with three kinds of dataset scale. The each experimental data is the average value of the five times result. The experimental data show that most of them meet the cognition. In this work, most of the differences of performance are related to the execution time.

Figure 4.10 shows that it takes more time to sort data through the portal web user interface. And we find that the differences of the sorting performance between using command line and the portal web user interface with Hadoop are higher than the one with Spark.

We can get the result that it also takes more time to perform the wordcount application through the portal web user interface from Figure 4.10. But the difference of the wordcount performance between using command line and the portal web user interface with Spark and the large scale dataset is quite higher than others.

Figure 4.12 shows that it takes more time to perform the TeraSort application through the portal web user interface but Hadoop with the small scale dataset.

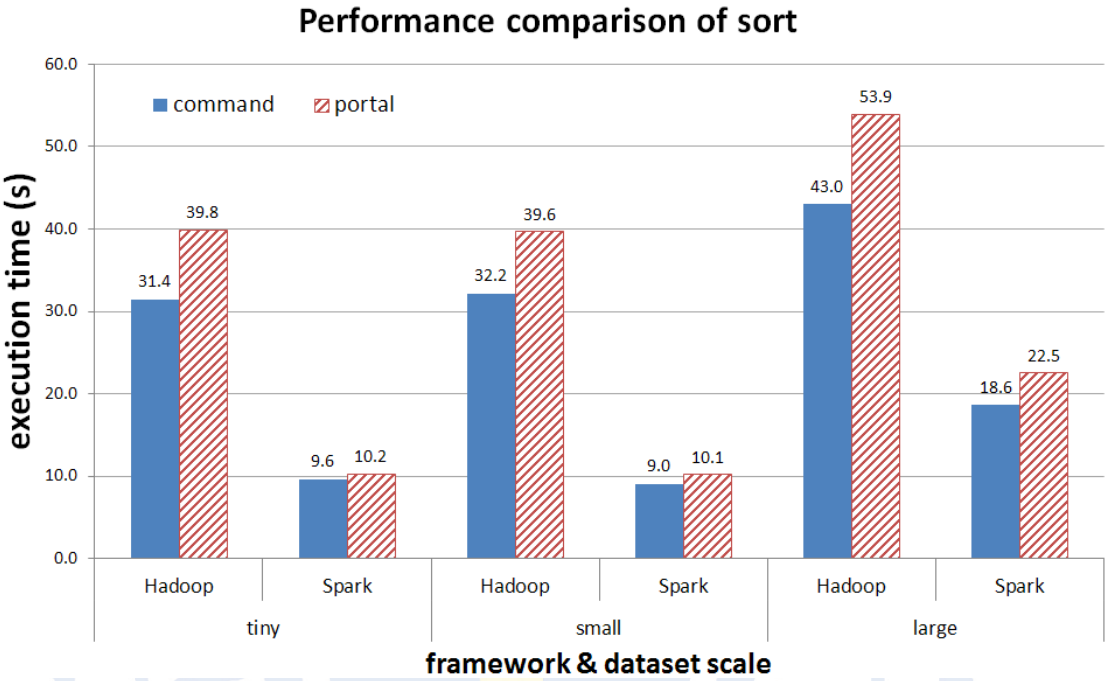

FIGURE 4.10: Sorting performance comparison

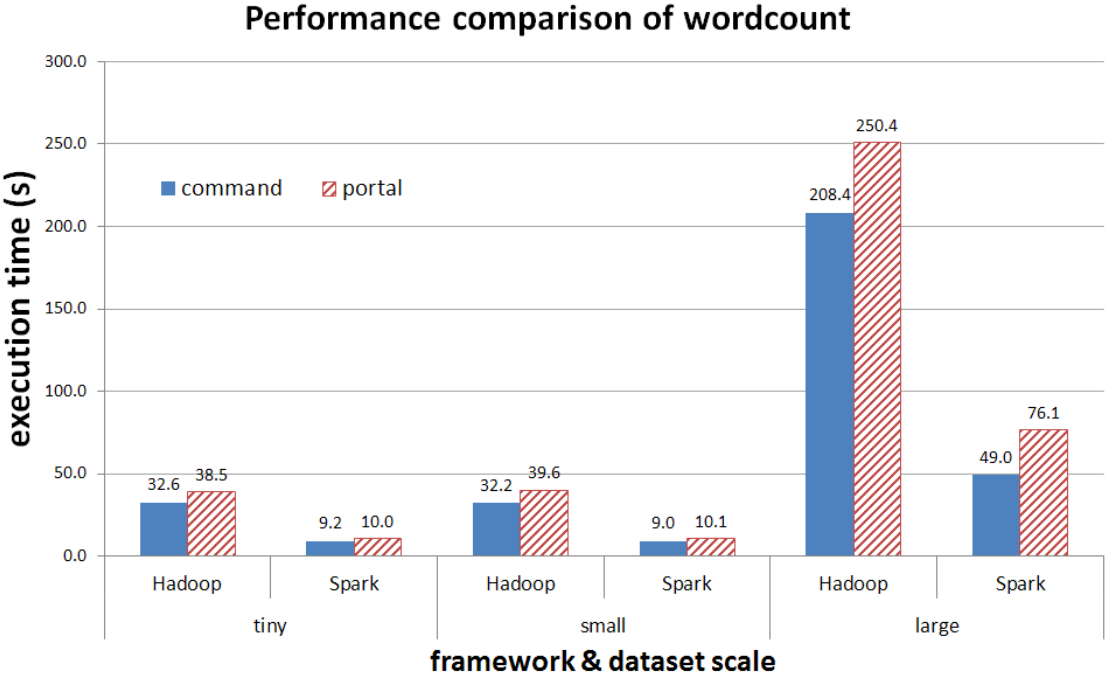

FIGURE 4.11: Word count performance comparison

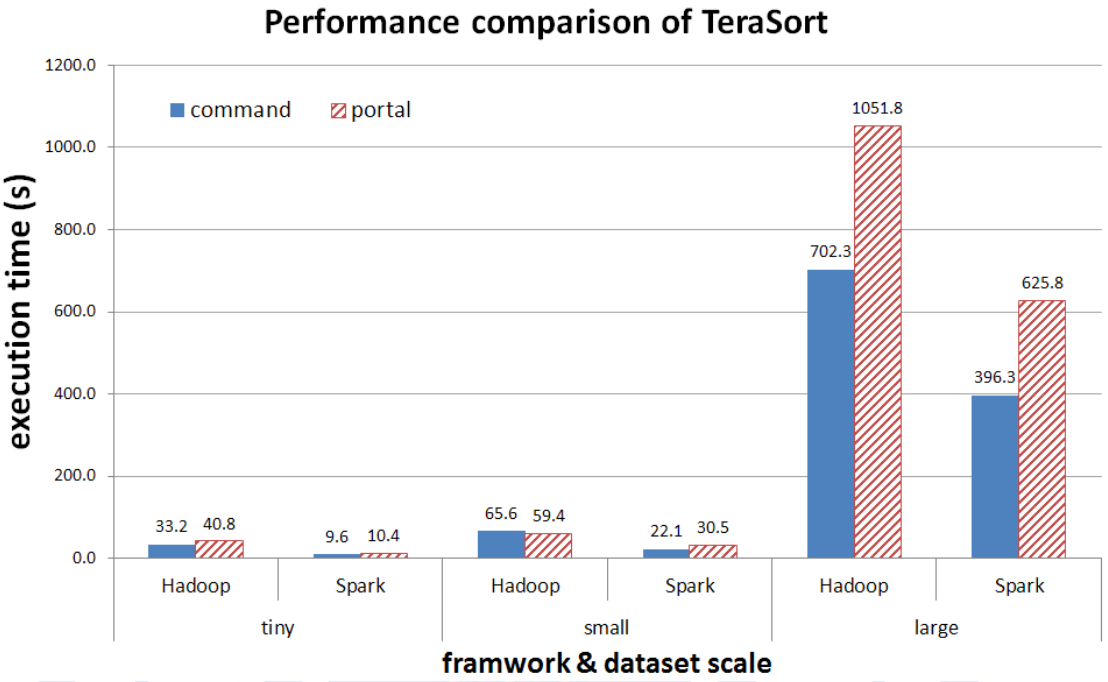

FIGURE 4.12: Terasort performance comparison

4.2.4 Performance Comparison between Hadoop and Spark

In this section, we perform the Hadoop and Spark performance comparison in our system. We test whether the performance comparison between Hadoop and Spark in our system is like the results of previous studies or not.

In Figure 4.13 and 4.14, we get that all the execution time of Spark are shorter than the one of Hadoop. The results of the experiments are similar to those presented in other works before. The performance of Spark is better than Hadoop even on our single-node virtual machine. And we find while the dataset scale is large in three kinds of jobs, the differences of performance are much greater.

4.2.5 OpenCV Environment Validation

For our related research, we need the environment with Hadoop and OpenCV together. So we have also built the OpenCV support environment in our virtual machine and make some test about Particle Image Velocimetry (PIV) applications

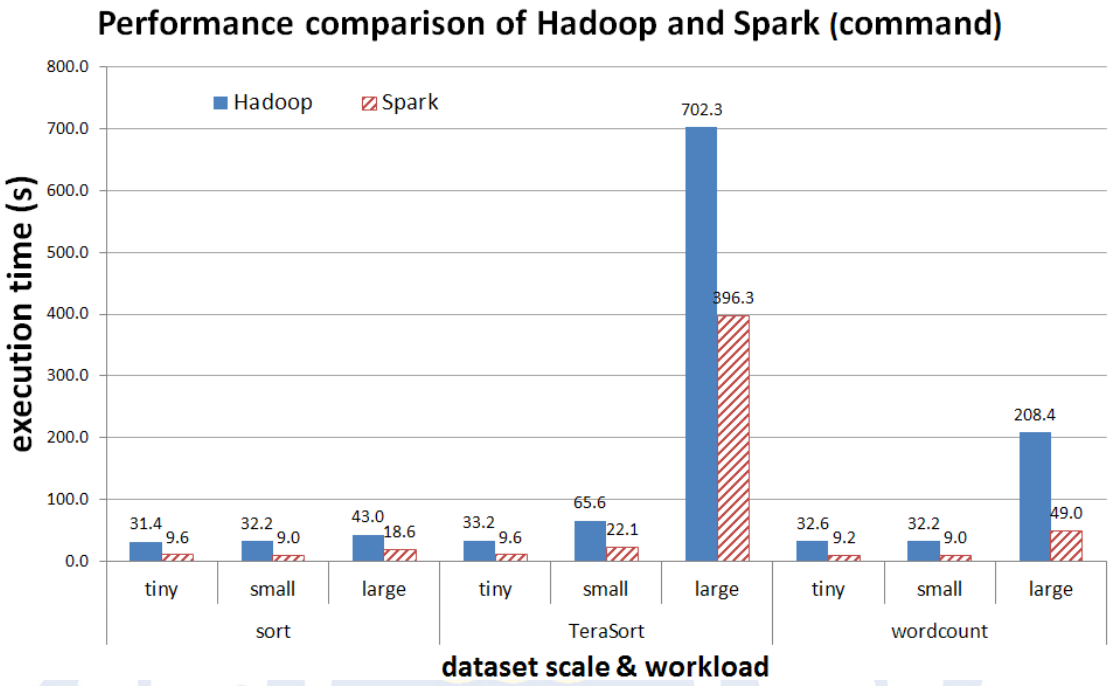

FIGURE 4.13: Performance comparison between Hadoop and Spark (command)

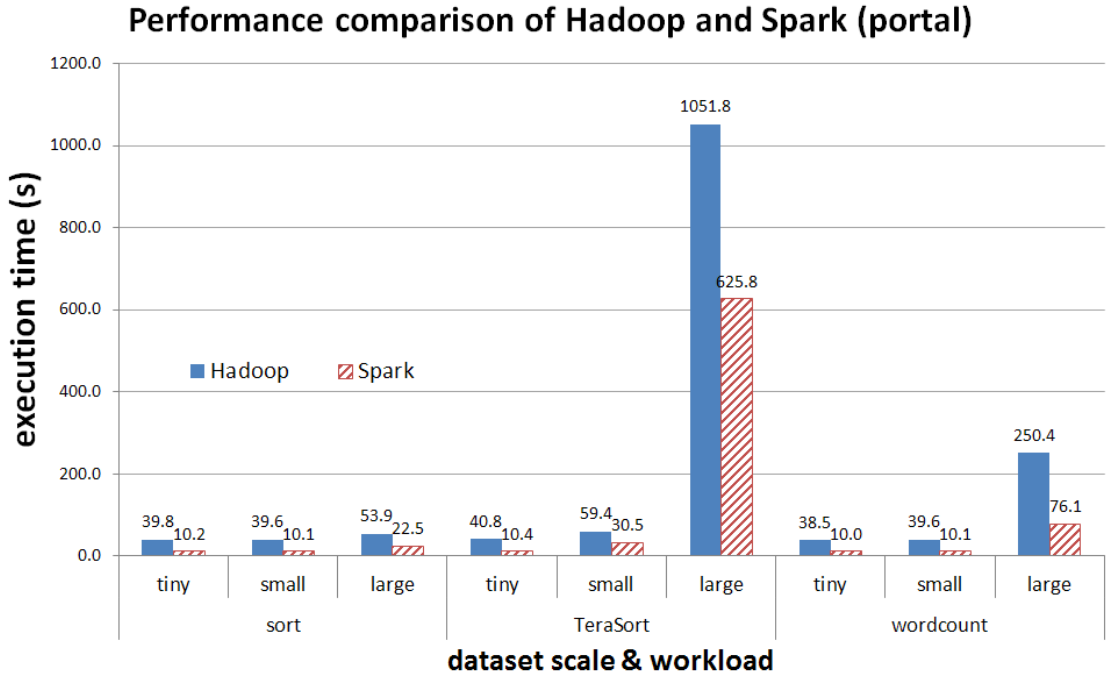

FIGURE 4.14: Performance comparison between Hadoop and Spark (portal)

to validate the OpenCV environment in our system. The Figure 4.15 and 4.16 below show the input images and output results of a application. The example application can read two pictures and calculate the shift direction and the amount of displacement by drawing the arrows on the output picture. This application uses

the OpenCV library and can be developed to determine the water flow direction using computers.

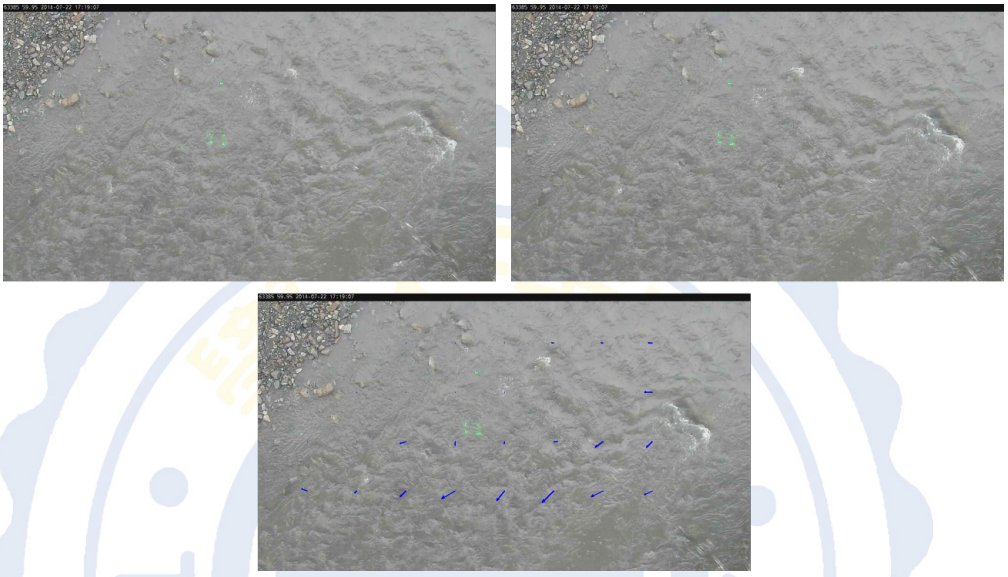

FIGURE 4.15: PIV result - river

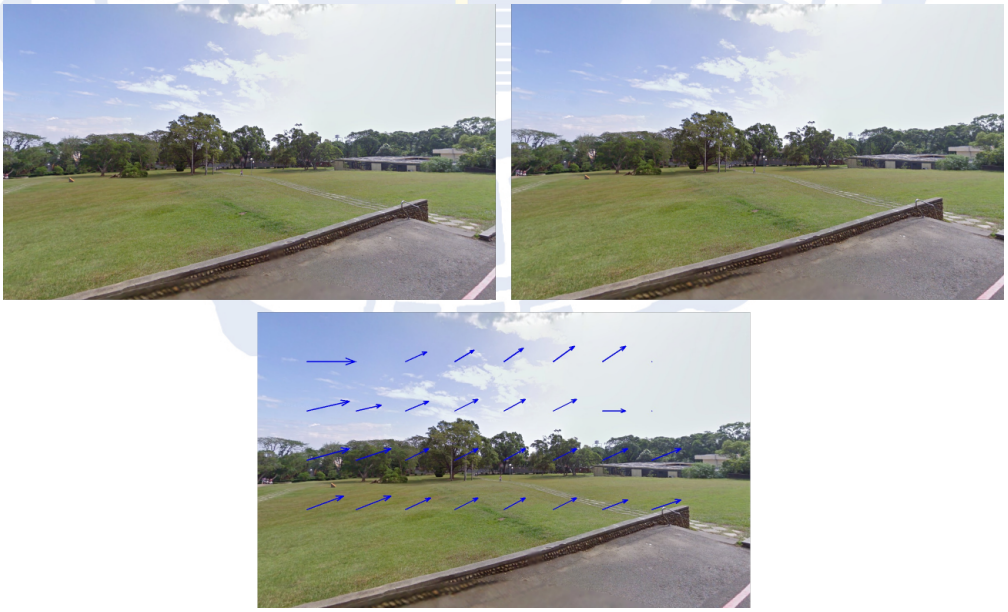

FIGURE 4.16: PIV result - campus [29]

Chapter 5

Conclusions and Future Work

5.1 Concluding Remarks

In this work, we developed and implemented a portal web user interface and portlets that facilitated the use of the Hadoop ecosystem and integrated the interface with the Hadoop ecosystem into a virtual machine image file. It provides a fast and convenient way to set up the platform for users. The processes of building the whole system were described and the implementation of our system was presented. We have actually tested how quick the process of using the virtual machine image file to deploy our system on several desktop environments. And we had executed the Hadoop job through the portlets we developed on the web user interface to validate the functionality. The differences of performance between using the portal web user interface and the command line to perform several works with Hadoop and Spark in our system are also tested. Executing the jobs with the large scale dataset by using the portlets takes more time than using the command line. The differences of performance between Hadoop and Spark are also presented. The result of the performance comparisons are similar to those presented in other studies before. In theory, the execution time of using command-line should be better than or equivalent to the execution time of

using the portal web user interface. Our experimental data show the actual test comparison results. That also shows that there is still room for improvement.

5.2 Future Work

This study is mainly the implementation of the functional part of the system. The performance of our system needs to be enhanced. The results of the works can be produced into a chart or other ways to show, in order to achieve the effect of visualization of data. In the course of the study, the possibility of using the portal interface of the machine to connect to other Hadoop servers had been thought about. We will keep studying on the feasibility of that function in the future so that the use environment of this portal web user interface can be more extensive. In addition, the current study of the Spark machine learning library to join the platform. We hope this platform not only has the ability to handle huge amounts of data, but also extends to the field of machine learning. With the increment of capabilities, the volume of our system may be more and more larger. It may be possible that we will provide the customized system that users can choose which features they want and exclude the unnecessary parts to reduce the volume of the system. The multi-node version of the system is also worth to be developed.

References

[1] Min Chen, Shiwen Mao, and Yunhao Liu. Big data: A survey. *Mobile Networks and Applications*, 19(2):171–209, 2014.

[2] Chao-Tung Yang ; Yin-Zhen Yan ; Ren-Hao Liu ; Shuo-Tsung Chen. Cloud City Traffic State Assessment System Using a Novel Architecture of Big Data. *2015 International Conference on Cloud Computing and Big Data (CCBD)*, 2015.

[3] Search interest of big data - Google Trends, 2017. `https://trends.google.com/trends/explore?date=all&q=big%20data&hl=en`.

[4] Douglas Laney. 3D data management: Controlling data volume, velocity, and variety. Technical report, META Group, February 2001.

[5] Apoorva Gupta. Big data analysis using computational intelligence and hadoop: A study. In *2015 International Conference on Computing for Sustainable Global Development, INDIACom 2015*, pages 1397–1401, 2015.

[6] Apache hadoop, 2014. `http://hadoop.apache.org/`.

[7] Hadoop, 2017. `http://en.wikipedia.org/wiki/Apache_Hadoop`.

[8] Mapreduce, 2017. `https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html`.

[9] Jens Dittrich and Jorge-arnulfo Quian. Efficient Big Data Processing in Hadoop MapReduce. *Proceedings of the VLDB Endowment*, 5(12):2014–2015, 2012.

[10] Dhruba Borthakur. The hadoop distributed file system: Architecture and design, 2007. http://hadoop.apache.org/docs/r0.18.0/hdfs_design.pdf.

[11] Farag Azzedin. Towards a scalable HDFS architecture. In *Proceedings of the 2013 International Conference on Collaboration Technologies and Systems, CTS 2013*, pages 155–161, 2013.

[12] What Is a Portlet - O'Reilly Media, 2017. http://archive.oreilly.com/pub/a/java/archive/what-is-a-portlet.html.

[13] Portals and Portlets: The Basics, 2017. http://editorial.mcpressonline.com/web/mcpdf.nsf/wdocs/5232/$file/5232_exp.pdf.

[14] Introduction to Liferay development, 2017. https://dev.liferay.com/zh/develop/tutorials/-/knowledge_base/7-0/introduction-to-liferay-development.

[15] About - OpenCV, 2017. http://opencv.org/about.html.

[16] OpenCV | NVIDIA Developer, 2017. https://developer.nvidia.com/opencv.

[17] Introduction - OpenCV 2.4.4.0 documentation, 2017. http://docs.opencv.org/2.4.4/modules/core/doc/intro.html.

[18] Virtualization Technology & Virtual Machine Software - VMware, 2017. https://www.vmware.com/il/solutions/virtualization.html.

[19] Yukio Tsuruoka. Cloud computing - current status and future directions. *Journal of Information Processing*, 24(2):183–194, 2016.

[20] Xiuqin Lin, Peng Wang, and Bin Wu. Log analysis in cloud computing environment with Hadoop and Spark. *2013 5th IEEE International Conference on Broadband Network & Multimedia Technology*, pages 273–276, 2013.

[21] Ilias Mavridis and Helen Karatza. Performance evaluation of cloud-based log file analysis with apache hadoop and apache spark. *Journal of Systems and Software*, 125:133–151, 2017.

[22] Chien-Heng Wu, Franco Lin, Wen-Yi Chang, Whey-Fone Tsai, Hsi-Ching Lin, and Chao-Tung Yang. Big data development platform for engineering applications. In *Proceedings - 2016 IEEE International Conference on Big Data, Big Data 2016*, pages 2699–2702, 2017.

[23] Shengsheng Huang, Jie Huang, Jinquan Dai, Tao Xie, and Bo Huang. The hibench benchmark suite: Characterization of the mapreduce-based data analysis. In *Proceedings - International Conference on Data Engineering*, pages 41–51, 2010.

[24] Timofei Epanchintsev and Andrey Sozykin. Processing large amounts of images on Hadoop with OpenCV. In *CEUR Workshop Proceedings*, volume 1513, pages 137–143, 2015.

[25] T.L.S.R. Krishna, T. Ragunathan, and S.K. Battula. Customized web user interface for Hadoop Distributed File System. *Advances in Intelligent Systems and Computing*, 380:567–576, 2016.

[26] Determine if your processor supports Intel Virtualization Technology - Intel, 2017. `https://www.intel.com/content/www/us/en/support/processors/000005486.html`.

[27] Cloudera enterprise 5.5.x documentation, 2017. `https://www.cloudera.com/documentation/enterprise/5-5-x.html`.

[28] Installation in linux —OpenCV 2.4.13.3 documentation, 2017. `http://docs.opencv.org/2.4/doc/tutorials/introduction/linux_install/linux_install.html`.

[29] Tunghai University - Google Maps, 2017. `https://www.google.com.tw/maps/@24.1786972,120.6000693,3a,75y,139.52h,91.5t/data=!3m5!1e1!3m3!2e0!7i13312!8i6656`.

[30] M. Mazhar, Anand Paul, Awais Ahmad, and Suengmin Rho. Urban planning and building smart cities based on the Internet of Things using Big Data analytics. *Computer Networks*, 2016.

[31] Lars George. *HBase: The Definitive Guide*. O'REILLY, 2012.

[32] Yang Jin, Tang Deyu, and Zhou Yi. A distributed storage model for ehr based on hbase. In *Information Management, Innovation Management and Industrial Engineering (ICIII), 2011 International Conference on*, volume 2, pages 369–372, Nov 2011.

[33] Haijie Ding, Yuehui Jin, Yidong Cui, and Tan Yang. Distributed storage of network measurement data on hbase. In *Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference on*, volume 02, pages 716–720, Oct 2012.

[34] Jun Bai. Feasibility analysis of big log data real time search based on hbase and elasticsearch. In *Natural Computation (ICNC), 2013 Ninth International Conference on*, pages 1166–1170, July 2013.

[35] Chen Zhang and Xue Liu. Hbasemq: A distributed message queuing system on clouds with hbase. In *INFOCOM, 2013 Proceedings IEEE*, pages 40–44, April 2013.

Appendix A

Hadoop Installation

I. Modify hosts

```
# sudo vim /etc/hosts
```

II. Modify hostname

```
# sudo vim /etc/hostname
# sudo service hostname start
```

III. Install Java JDK

```
# sudo apt-get -y install openjdk-7-jdk
# sudo ln -s /usr/lib/jvm/java-7-openjdk-amd64 /usr/lib/jvm/jdk
```

IV. Add hadoop user

```
 # sudo addgroup hadoop
 # sudo adduser --ingroup hadoop hduser
 # sudo adduser hduser sudo
```

V. Creat SSH authentication login

```
# ssh-keygen -t rsa -f \~{}/.ssh/id\_{}rsa -P ""
# cp \~{}/.ssh/id\_{}rsa.pub ~/.ssh/authorized\_{}keys
# scp  -r ~/.ssh hduser:~/
```

VI. Download hadoop

```
# cd ~
# wget http://ftp.twaren.net/Unix/Web/apache/hadoop/common \\
   /hadoop-2.6.0/hadoop-2.6.0.tar.gz
# tar zxf hadoop-2.6.0.tar.gz
# mv hadoop-2.6.0.tar.gz hadoop
```

VII. Add the environment variable

```
# vim .bashrc

export JAVA_HOME=/usr/lib/jvm/jdk/
export HADOOP_INSTALL=/home/hduser/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
```

VIII. Set hadoop config

```
# cd hadoop/etc/hadoop
# vim hadoop-env.sh

export JAVA_HOME=/usr/lib/jvm/jdk/

# vim core-site.xml

<property>
   <name>fs.default.name</name>
   <value>hdfs://hadoop-master:9000</value>
</property>

# vim yarn-site.xml

<property>
```

```
   <name>yarn.nodemanager.aux-services</name>
   <value>mapreduce_shuffle</value>
</property>
<property>
   <name>yarn.resourcemanager.hostname</name>
   <value>hduser</value>
</property>

# cp mapred-site.xml.template mapred-site.xml
# vim mapred-site.xml

<property>
   <name>mapreduce.framework.name</name>
   <value>yarn</value>
</property>

# mkdir -p ~/mydata/hdfs/namenode
# mkdir -p ~/mydata/hdfs/datanode
# vim hdfs-site.xml

<property>
   <name>dfs.replication</name>
   <value>2</value>
 </property>
 <property>
   <name>dfs.namenode.name.dir</name>
   <value>/home/hduser/mydata/hdfs/namenode</value>
 </property>
 <property>
   <name>dfs.datanode.data.dir</name>
   <value>/home/hduser/mydata/hdfs/datanode</value>
 </property>

# vim slaves
hadoop-master
node01
node02
node03
node04
node05
node06
node07
node08
node09
node10
node11
node12
```

IX. Copy hadoop to all nodes

```
# scp -r /home/hduser/hadoop node01:/home/hduser
# scp -r /home/hduser/hadoop node02:/home/hduser
# scp -r /home/hduser/hadoop node03:/home/hduser
# scp -r /home/hduser/hadoop node04:/home/hduser
# scp -r /home/hduser/hadoop node05:/home/hduser
# scp -r /home/hduser/hadoop node06:/home/hduser
# scp -r /home/hduser/hadoop node07:/home/hduser
# scp -r /home/hduser/hadoop node08:/home/hduser
# scp -r /home/hduser/hadoop node09:/home/hduser
# scp -r /home/hduser/hadoop node010:/home/hduser
# scp -r /home/hduser/hadoop node011:/home/hduser
# scp -r /home/hduser/hadoop node012:/home/hduser
```

X. Format HDFS

```
# hdfs namenode -format
```

XI. Start hadoop

```
# start-all.sh
```

XII. Use jps to see java running program

```
# jps
```

XIII. MapReduce JobTracker monitoring website

```
# hadoop-master:50030
```

Appendix B

Spark Installation

I. Download and Unzip Scala

```
#wget  \\
http://ftp.twaren.net/Unix/Web/apache/spark/spark-1.4.1/spark-1.4.1-bin-hadoop2.6.tgz
#tar zxf spark-1.4.1-bin-hadoop2.6.tgz
#mv spark-1.4.1-bin-hadoop2.6 spark
#cd spark/conf
```

IV. Set Spark config

```
#vim spark-env.sh
export SCALA_HOME=/usr/lib/scala
export JAVA_HOME=/usr/lib/jvm/jdk
export SPARK_MASTER=master
export HADOOP_HOME=/home/hduser/hadoop
export SPARK_HOME=/home/hduser/spark
export SPARK_LIBARY_PATH=.:$JAVA_HOME/lib:$JAVA_HOME/jre/lib:$HADOOP_HOME/lib/native
export YARN_CONF_DIR=$HADOOP_HOME/etc/hadoop

#vim slaves
hadoop-master
node01
node02
node03
node04
node05
node06
```

```
node07
node08
node09
node10
node11
node12
```

III. Copy spark to all nodes

```
# scp -r /home/hduser/spark node01:/home/hduser
# scp -r /home/hduser/spark node02:/home/hduser
# scp -r /home/hduser/spark node03:/home/hduser
# scp -r /home/hduser/spark node04:/home/hduser
# scp -r /home/hduser/spark node05:/home/hduser
# scp -r /home/hduser/spark node06:/home/hduser
# scp -r /home/hduser/spark node07:/home/hduser
# scp -r /home/hduser/spark node08:/home/hduser
# scp -r /home/hduser/spark node09:/home/hduser
# scp -r /home/hduser/spark node010:/home/hduser
# scp -r /home/hduser/spark node011:/home/hduser
# scp -r /home/hduser/spark node012:/home/hduser
# bin/start-hbase.sh
```

Appendix C

HBase Installation

I. Download HBase

```
# cd ~
# wget http://ftp.twaren.net/Unix/Web/apache/hbase\\
   /hbase-1.0.0/hbase-1.0.0-hadoop2-bin.tar.gz
```

II. Unzip hbase-1.0.0-hadoop2-bin.tar.gz

```
# tar zxf hbase-1.0.0-hadoop2-bin.tar.gz
```

III. Move the File of HBase

```
# mv hbase-1.0.0-hadoop2 hbase
```

IV. Set HBase config

```
# cd hbase
# vim conf/hbase-env.sh

export JAVA_HOME=/usr/lib/jvm/jdk
export HBASE_HOME=/home/hduser/hbase

# hadoop fs -mkdir /hbase
# vim conf/hbase-site.xml
```

```
<property>
        <name>hbase.rootdir</name>
        <value>hdfs://hadoop-master:9000/hbase</value>
</property>
<property>
        <name>hbase.cluster.distributed</name>
        <value>true</value>
</property>
<property>
        <name>hbase.zookeeper.quorum</name>
        <value>Test-master</value>
</property>

# vim conf/regionservers
hadoop-master
node01
node02
node03
node04
node05
node06
node07
node08
node09
node10
node11
node12
```

III. Copy jar to hbase/lib

```
# rm lib/hadoop-*
# cd /home/hduser/hadoop/share/hadoop
# cp *.jar /home/hduser/hbase/lib/
```

IV. Copy hbase to all nodes

```
# scp -r /home/hduser/hbase node01:/home/hduser
# scp -r /home/hduser/hbase node02:/home/hduser
# scp -r /home/hduser/hbase node03:/home/hduser
# scp -r /home/hduser/hbase node04:/home/hduser
# scp -r /home/hduser/hbase node05:/home/hduser
# scp -r /home/hduser/hbase node06:/home/hduser
# scp -r /home/hduser/hbase node07:/home/hduser
```

```
# scp -r /home/hduser/hbase node08:/home/hduser
# scp -r /home/hduser/hbase node09:/home/hduser
# scp -r /home/hduser/hbase node010:/home/hduser
# scp -r /home/hduser/hbase node011:/home/hduser
# scp -r /home/hduser/hbase node012:/home/hduser
# bin/start-hbase.sh
```

V. HBase monitoring website

```
# hduser:60010
```