國 立 成 功 大 學
電 機 工 程 學 系
電 腦 與 通 信 工 程 研 究 所

碩 士 論 文

(初稿)

跨叢集之聯邦 Hadoop 分散式檔案系統

FedHDFS^{+}：Federated HDFS across Multi-Hadoop Clusters

研究生： 劉澤恩 Student: Tse-En Liu

指導教授： 謝錫堃 Advisor: Ce-Kuen Shieh

共同指導教授： 張志標 Co-Advisor: Jyh-Biau Chang

Institute of Computer and Communication Engineering

Department of Electrical Engineering

National Cheng Kung University

Tainan, Taiwan

July, 2015

中華民國 一零四年七月

跨叢集之聯邦 Hadoop 分散式檔案系統

劉澤恩* 謝錫堃** 張志標**

國立成功大學　電機工程學系

電腦與通信工程研究所

摘要

如今，資料持續不斷的增長並且環繞在我們的生活上，資料中心能便利地提供我們對這些大資料達到有效的儲存與收集,並且快速的處理、運算及分析。然而，資料實際上會因為不同地區性的關係，而分散的儲存在不同地區的資料中心。因此許多相關研究以及組織基於 Hadoop 之下，開始針對分散在不同地區的大資料，設計多叢集運算系統並且減少資料的傳輸，而這樣的系統不僅能夠聯合多個運算叢集的資源還能分析更龐大的資料，以利於找出更多價值。

不幸的是，儘管目前舊有的多叢集運系統能夠提供給我們許多便利性及優點，但仍然還是在初期發展的階段，而最主要的缺點是只有設計 Hadoop MapReudce 的運算框架達到跨叢集運算，卻忽略了這些跨叢集的 Hadoop 分散式檔案系統(HDFS)。這些各自為政的 HDFS，在多叢集運算系統之下就無法達到真正的透通性，例如使用者在運算之前必須繁瑣地查看多個 Hadoop 分散式檔案系統的資料；此外也因為多叢集運算系統沒有整合的檔案系統，跨叢集運算之前，需使用者額外提供系統每個資料中心所需要運算的資料，才能讓系統能夠將運算搬移到資料上,另外無法有效地提供多階段的MapReduce運算的應用。

在本篇論文中，我們提出一套系統稱 FedHDFS+, 能夠有效的跨叢集聯合多個 Hadoop 分散式檔案系統(HDFS)，並且提供單一全域的命名空間讓使用者可看到多個 Hadoop 叢集的檔案資料，在啟動多叢集運算時，系統能完全透通性

的自動化將運算搬移到資料所在處。除此之外，我們整合了多叢集運算框架(Fed-MR)，在 FedHDFS+的內部設計下，能夠在多叢集運算時達到效能最佳化。
關鍵字：多叢集運算系統、聯合分散式檔案系統、地區分散性資料、全域命名空間
*作者　**指導教授

FedHDFS+：Federated HDFS across Multi-Hadoop clusters

Tse-En Liu* Ce-Kuen Shieh** Jyh-Biau Chang**

Institute of Computer and Communication Engineering,

Department of Electrical Engineering,

National Cheng Kung University

Tainan, Taiwan

Abstract

Nowadays datacenter allow storing and processing increasing amounts of massive unstructured data which rapidly creeping of our life. However, data is actually stored across multiple geographically distributed datacenters in the real world. For this reason, few of researchers and organizations focus on the multi-cluster computing system aimed at analyzing geographically distributed data among independent organizations while avoiding data movement. Unfortunately, despite the many advantages of the infant multi-cluster computing systems, it still possesses a major drawback for only developing MapReduce of computation layer, but lack for HDFS integration on storage layer. Separated HDFS leads to multi-cluster computing systems have multiple HDFS image within cross-domain system, the system was unware of the geo-data locations, users need to define extra jobs' workflow for each job-submission, and it cannot supports multiple-stage MapReduce applications on geo-data.

In this paper we propose a Federated Hadoop Distributed File System (FedHDFS+), which enable provides global Hadoop HDFS image, and supports location-aware job submission across multiple Hadoop clusters. With our system, it takes multi-cluster

computing systems more elastically, availability and transparently. In addition, FedHDFS+ is compatible with our previous work: Fed-MR framework. We design dynamically Top Cloud selection mechanism for Fed-MR jobs, as a result, it helps us to get optimum performance on Fed-MR framework.

Keywords：Multi-cluster computing system, Federated HDFS, Geo-distributed data, Global Namespace

*Author **Advisor

Contents

Figures

Tables

Chapter 1 : Introduction

Data is rapidly creeping into every element of our life, stream from scientific organization, engineering disciplines and the Internet of Things, these huge of disorderly data which called “Big data [1]”, comes from a great variety of sources and generally has in three types, include structured data, semi-structured data and unstructured data. The primary goal of big data analytics is to help companies or organizations make more informed business decisions and insights. However, effective analytics and generate new insights over big data, has emerged as a key ingredient, Hadoop [2] is powerful of large-scale data-intensive cloud computing platform. This community driven open-source project governed by the Apache Software Foundation [3], it was originally implemented at Yahoo based on papers published by Google in 2003 and 2004.

In short, the core of Hadoop consists of a distributed storage part (Hadoop Distributed File System) [4] and a parallel processing part (MapReduce) [5]. The design goal is to achieve data-intensive computing, MapReduce transfers packaged code for nodes to process in parallel, based on the data each node needs to analysis. In other words, you don’t need to consider how you intend to process your data before you store it; this strategy is moving computation to the data, instead of moving the data to the computation, it allows Hadoop to achieve high data locality which in turn results in high performance. Hadoop also work at several different organizations like Hortonworks [6], Microsoft [7], Facebook [8], Cloudera [9] and many others around the world.

Nowadays, large organizations today have a planetary-scale footprint and operate tens of data centers around the globe; they have external-facing geo-distributed data, where required by an application are generated, stored and maintained by multiple independent

datacenters. One of the examples is the specialized field of High Energy Physics (HEP), in total, ATLAS and the three other main detectors at the Large Hadron Collider (LHC) produced 13 petabytes (13 × 1015 bytes) of data in 2010, as illustrated in Fig. 1. These huge amounts of geo-distributed datasets are stored on the Worldwide LHC Computing Grid that consists of more than 140 computing centers distributed across 34 countries. In addition, Volley [10] also point out as clouds services grow to span more and more globally distributed datacenters, there is an increasingly urgent need for geo-distributed data analytics among these datacenters.

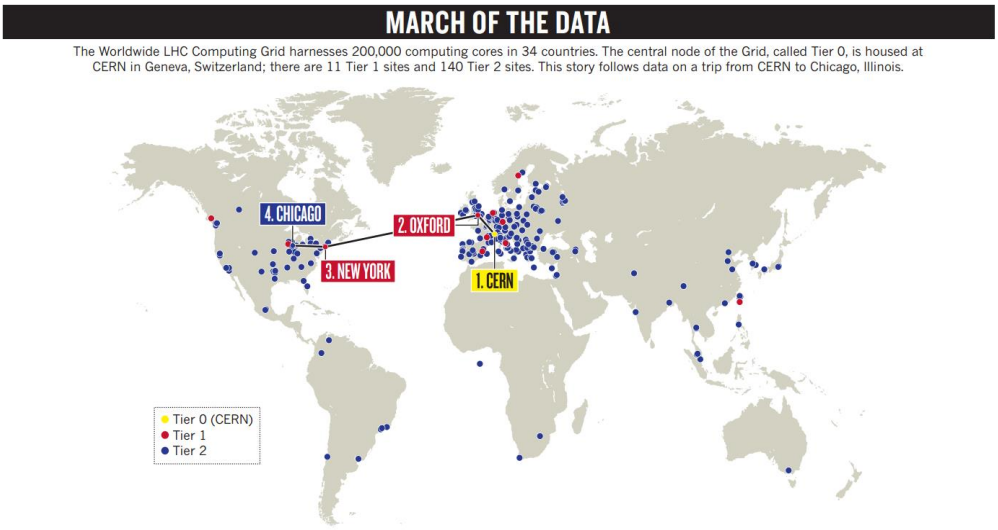

Figure 1: Geographically distributed data

As the mentioned above, the naïve approach to perform cluster computing on large-scale geo-distributed datasets is to migrate all data to a single cluster, and then traditionally perform cluster computation the usual way. However moving all data to a single datacenter before processing the data is expensive, so sharing a Hadoop clusters between organizations is attractive, because it enables statistical multiple (lowering cost) and allows organizations to share a large geo-datasets, in order to achieve geo-data analytics across multiple data centers, a field of multi-cluster system research has proposed to better handle geo-distributed

data, such as SAGE [11], GEO-PACTs [12], HMR [13], Fed-MR [14], which enable reduce the overhead of data transfers significantly and efficient analyzing all such geo-distributed data globally in a consistent.

Unfortunately, despite the many advantages of the infant multi-cluster computing systems, it still possesses a major drawback for only developing MapReduce of computation layer, but lack for HDFS integration on storage layer. Separated HDFS leads to multi-cluster computing systems have not enough transparent to users to run data-intensive applications, hence, there are some limitations for existing multi-cluster processing frameworks: 1) multiple Hadoop HDFS image within cross-domain system, it cause users cannot efficiently manage geo-data, 2) system unware of the geo-data locations, it need to user defined extra jobs' workflow for each job-submission and 3) Poorly multiple-stage MapReduce applications on geo-data, it indicates when cross-domain job finished each time, final output data will aggregate to the specific cluster, for this reason, single-stage is poor in combine multiple clusters' resource to analysis next time.

We proposed Federated Hadoop Distributed File System (FedHDFS^{+}), which unifies multiple HDFS across multi-Hadoop clusters, and achieves following benefits:

- Provides Single (Global) Hadoop HDFS Image
- Transparent to users to run Cross-Domain Applications
- Support Multiple-Stage MapReduce applications for clusters

Firstly, our solution provide single Hadoop HDFS image among multiple clusters, it is effective to achieve data location-aware job submission automatically. For user-level, they don't need to concern about where data physical resided.

New ways of integration multiple HDFS image and linking geo-distribute datasets have played a large role in generating new insights, and creative approaches to visualizing geo-data. We also concentrate on geographical distribution of data for sequential execution of MapReduce jobs to optimize the latency time between jobs. The previous cross-domain execution strategy of MapReduce program is not efficiency for multiple-stage cross-domain MapReduce processes and as it does not know about the geo-data location. Thus, to overcome these issues, we are enhancing our proposed work with unifies multiple Hadoop HDFS and provide Global Namespace. With these issues in mind, FedHDFS^{+} is designed to give users and IT a unified view of their data under a global namespace, independent of where the files are physically stored or from where they are accessed. IT can centrally manage data and analysis content through a central cluster, and users gain this unified view to enhance productivity and efficiency around file management and collaboration.

The remainder of the paper is as followings. Chapter 2 presents some background on HDFS & Hadoop architecture and also shows related works about previous multi-cluster computing frameworks, furthermore, point out some problems. Chapter 3 illustrates the FedHDFS^{+} overview, design issues and system architecture. Chapter 4 shows implementation includes system components in details. Chapter 5 presents the experimental measurements. Conclusion and future work is provided in Chapter 6.

Chapter 2 : Backgrounds and Related Works

2.1 Backgrounds

A. Hadoop & HDFS

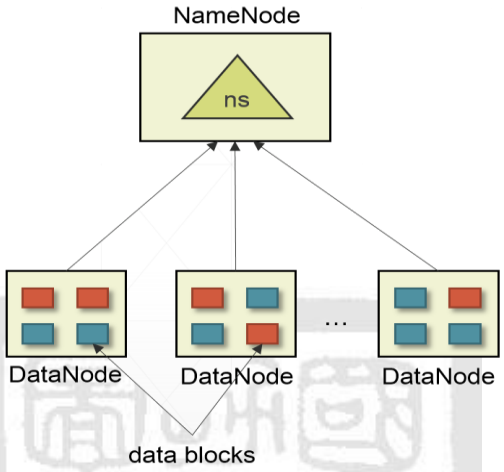

Figure 2: Hadoop distributed file system

The Hadoop Distributed File System (HDFS) is a distributed scalable file system with a master/slave architecture in Fig. 2. It designed to be deployed on low-cost hardware and enable to store large files across multiple machines to achieve highly fault-tolerant and reliability.

NameNode, the machine running the server process is the HDFS master. At a high level, the NameNode's primary responsibility is storing the HDFS namespace (ns), any change to the file system namespace or its properties is recorded by the NameNode. This means things like the directory tree, file permissions, and the mapping of files to block IDs. In addition, there are a number of DataNodes, which are deployed on each machine and provided data storage services for the shared file system. These machines typically store the data blocks in

its local disk and perform block creation, deletion, and replication upon instruction from the NameNode.

B. Global Namespace

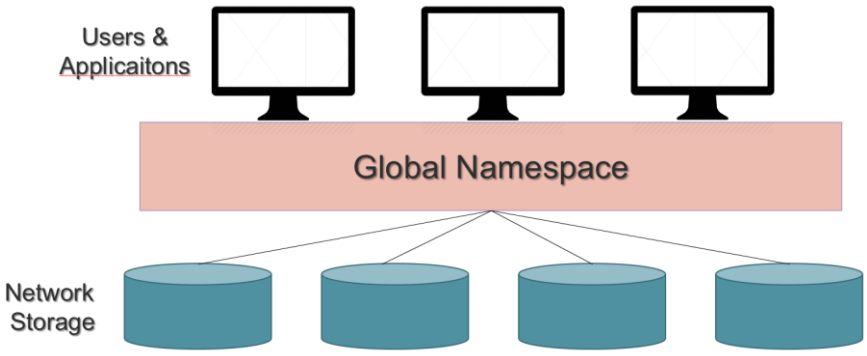

Figure 3: Global Namespace

Global namespace [15], as showed in Fig. 3, is an important mechanism for managing distributed file storage, allowing access to file data regardless of physical location. Files across the enterprise are dispersed among a wide range of devices, servers and cloud storage repositories. Global namespace provides virtual namespace to seamlessly integrate local and cloud storage for users, which means an organizations does not have to change their existing environment in order to deploy a namespace, but can make a seamless transition from their current environment to one using a global namespace. The global namespace approach to file management dramatically simplifies file management, as it frees the user to add, move, change, and reconfigure physical storage without affecting how clients access it.

C. Distributed Data-Intensive Computing

Data-intensive computing [16] is a class of parallel computing applications which use a data parallel approach to processing large volumes of data typically terabytes or petabytes in size. For geo-distributed data-intensive applications, instead of transferring data explicitly from site to site, hence moving computation near the data is a more efficient computing

model. Hadoop MapReduce and several large scale data-intensive computing frameworks [17, 18] exemplify this model. However, above mentioned tools are originally designed for cluster computing on a single set of tightly-coupled nodes and perform poorly or not at all when deploy computation across multi-clusters. In our approach will aims this issues to extend to multi-cluster computing, and achieve effective geo-data analyzation for multiple-stage MapReduce Applications.

2.2 Related Works

In this section, the multi-cluster computing frameworks are introduced. Some of previous cross-domain systems based on the hierarchical MapReduce framework are also introduced, including our own previous work, Federated MapReduce (Fed-MR) [14], which extensively provides user programming-level transparency. Previous works have summed up two main approaches to execute MapReduce over geographically distributed data in multi-cluster settings [13, 14, 17].

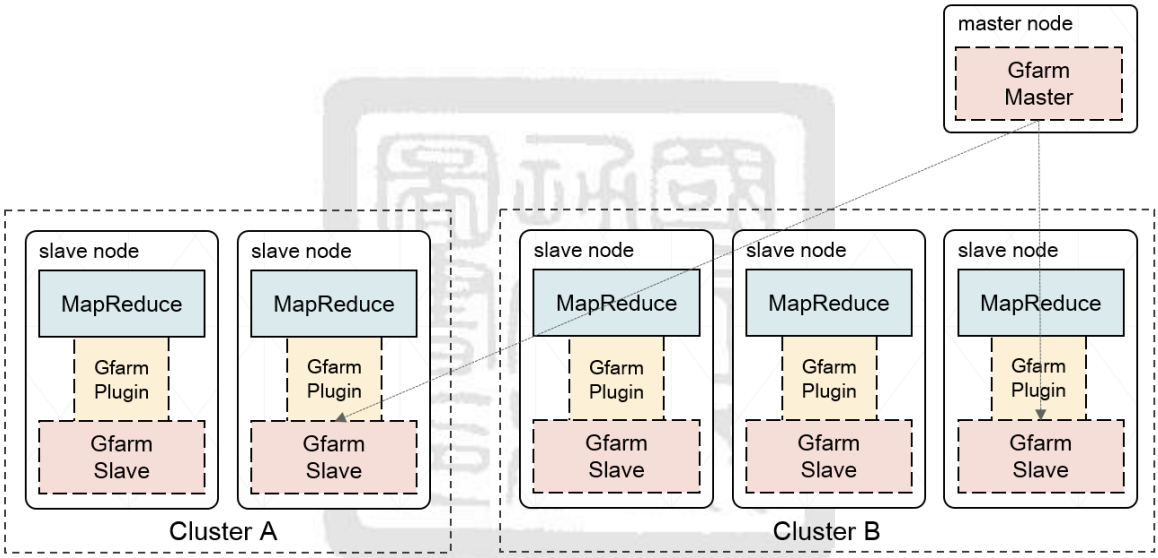

Figure 4: G-Hadoop system architecture

The first is to globally deploy a single MapReduce platform over all of the clusters' resources. G-Hadoop [19] as shown in Fig. 4, is a framework for large-scale distributed computing across data centers with multiple clusters, they replaces Hadoop's native distributed file system with the Gfarm Grid file system as a global distributed file system that supports MapReduce framework. Gfarm was a distributed file system designed to share vast amounts of data between globally distributed clusters connected via a wide-area network. MapReduce applications in G-Hadoop are scheduled across multiple clusters using a hierarchical scheduling approach. The MapReduce tasks are firstly scheduled among the

clusters using Hadoop's data-aware scheduling policy and then among compute nodes using the existing cluster scheduler on the target clusters. G-Hadoop maintains the master/slave model of Hadoop, where the slave nodes are simple workers, while the master node accepts the jobs submitted by the user, splits them into smaller tasks, and finally distributes the tasks to the slave nodes in a wide-area network. Although this avoids movement of the raw data, WAN connected nodes can severely affect overall MapReduce performance. As we mentioned above, it have been reported that G-Hadoop has some limitations for distributed data processing across multiple data centers, because of deploy a single MapReduce platform over Gfarm, which is a cross-WAN distributed file system. Besides, the prior Gfarm architecture allows only a single metadata server for the entire thousand slave nodes, leads to Gfarm become a heavyweight file system.

The second approach of hierarchical MapReduce framework, avoids both inter-cluster raw data movement and deployment-induced deficiencies by deploying independent MapReduce platforms on each cluster.

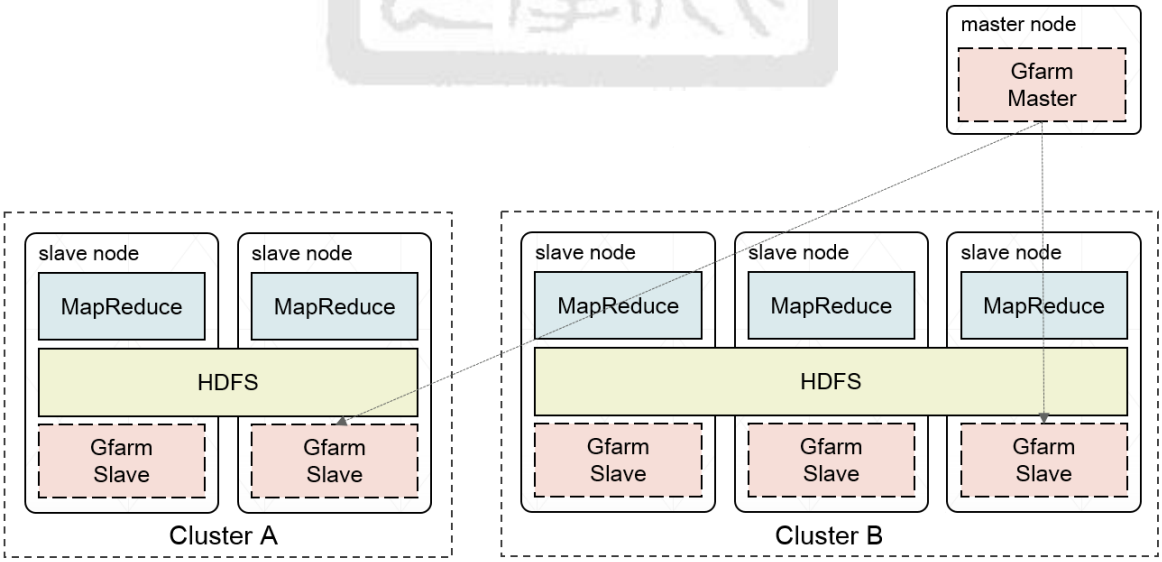

Figure 5: HMR system architecture

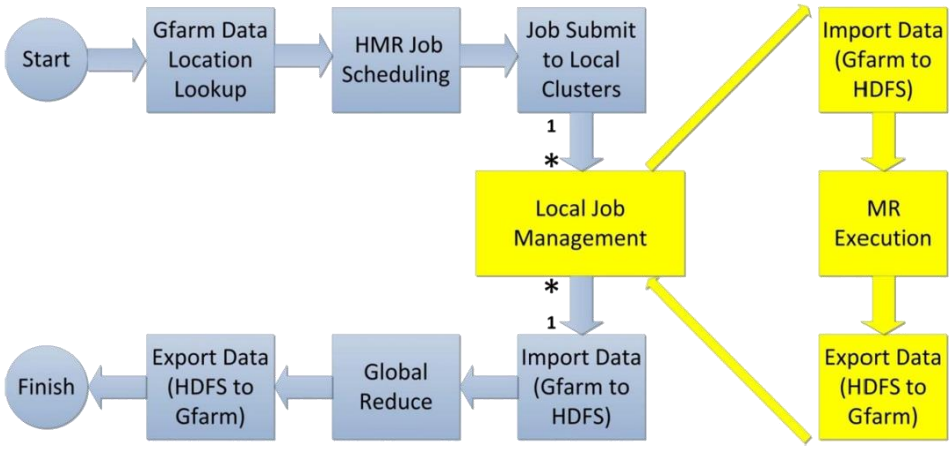

Figure 6: HMR execution workflow

As illustrated in Fig. 5 and Fig. 6, Luo et al. [13] proposed a hierarchical MapReduce framework from multiple Hadoop clusters. In HMR framework consists of two layers, the top layer has a global controller that accepts user submitted MapReduce jobs and distributes them across different local Hadoop cluster domains. In the beginning of cross-domain job, users submit their job to the global controller, upon receiving a user job, the global controller divides the job into sub-jobs according to the capability of each local cluster. The programming model of hierarchical MapReduce framework is the "Map-Reduce-Global Reduce" model where computations are expressed as three functions: Map, Reduce, and Global Reduce, as illustrated in Fig. 6. For data-intensive applications, instead of transferring data explicitly from site to site, they also explore using a shared file system Gfarm [20] to share data sets among the Global Controller and local Hadoop clusters. The steps of a HMR execution with Gfarm shows in the Fig. 6, the blue box steps are executed on the Global Controller and the yellow box steps are executed on local MapReduce clusters. From the yellow stage in HMR execution workflow, need to import data From Gfarm to HDFS each Hadoop cluster to computing, when local Hadoop clusters processing finished, they will import data From HDFS to Gfarm, hence, Disk I/O need to be considered, it will affect overall performance on multi-cluster computing.

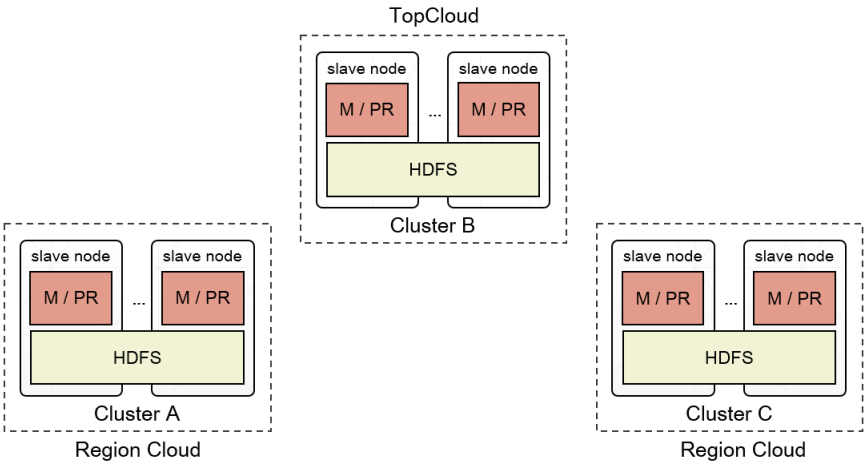

Figure 7: Fed-MR architecture (Map / Proxy-Reduce)

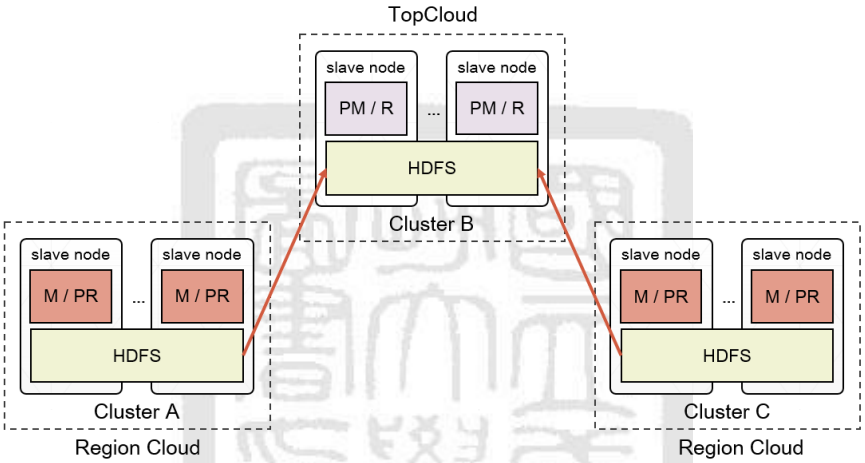

Figure 8: Fed-MR architecture (Proxy-Map / Reduce)

Our previous work, Federated MapReduce (Fed-MR) provided a transparent way to run original MapReduce jobs across multiple clusters without any extra programming burden. Fig. 7 and Fig. 8 illustrates how Fed-MR uses the proxy phases to decompose an original MapReduce job into sub-jobs for the region clusters. Fed-MR, solved the above issue by introducing a proxy-based method, users don't need to implement an extra Global Reduce job to achieve global MapReduce execution. The original map and reduce phases was complemented with a proxy-reduce and a proxy-map phase for MapReduce over multiple clusters. At its essence, the automated proxy phases are responsible for the passing of intermediate data across clusters for aggregated computation. Users supply a job consisting only of a map and reduce function, just as usual.

```
<jobconf>
  <input>
    <datacenter>DC1</datacenter>
    <location>/user/ec2-user1/webinput</location>
  </input>
  <mapper>gmr.WordCountMapper</mapper>
  ...
  <optimizeForTime>false</optimizeForTime>
  <mrFunction>gmr.WordCountMRFunction</mrFunction>
  ...
</jobconf>
```

Figure 9: Example of cross-domain job configuration

Actually, separated Hadoop distributed file system of multi-cluster computing, such as Fed-MR [14], HMR [13] and G-MR [21], will bring about non-transparently for users and system to run cross-domain MapRedcue jobs before job submitted. For the file-system-level, multiple HDFS are without federating each other, so cross-domain system was unware of the geo-data locations, it leads to poorly data-location jobs submission for system, and cannot atomically moving computation to the data. As a solution for the above, developer need to define job's configuration, example as show in Fig. 9, which contains workflow, data location, datacenters, etc. Besides, these hierarchical MapReduce frameworks only perform single-stage MapReduce for multi-cluster computing, and cannot supports multiple-stage, real-world data is dirty, we need to do several preprocessing stage for cleaning unstructured data, and until data already need to aggregated.

Chapter 3 : FedHDFS+ System Design

In order to achieve transparently multi-clusters computing, FedHDFS+ collaborates with our previous work, hierarchical MapReduce framework Fed-MR (as mentioned in chapter 2. Related works), which enable run original MapReduce across geographically distributed clusters without any introducing additional global functions. The goal of FedHDFS+ is unified multiple HDFS across different heterogeneous Hadoop clusters and achieves following benefits:

- Provides Single (Global) Hadoop HDFS Image
- Transparent to users to run Cross-Domain Applications
- Support Multiple-Phases MapReduce applications for clusters

FedHDFS+ is potentially to play the important role of file-system-level in multi-cluster computing area, it makes users don't need to traverse multiple Hadoop distributed file system, and provides data-location aware jobs submission for data-intensive applications across several clusters, it takes multi-cluster computing systems more elastically, availability and transparently. So next, we will going to illustrate FedHDFS+ system overview, describe our primarily design issues on high level and show FedHDFS+ system architecture in detail.

3.1 System Overview

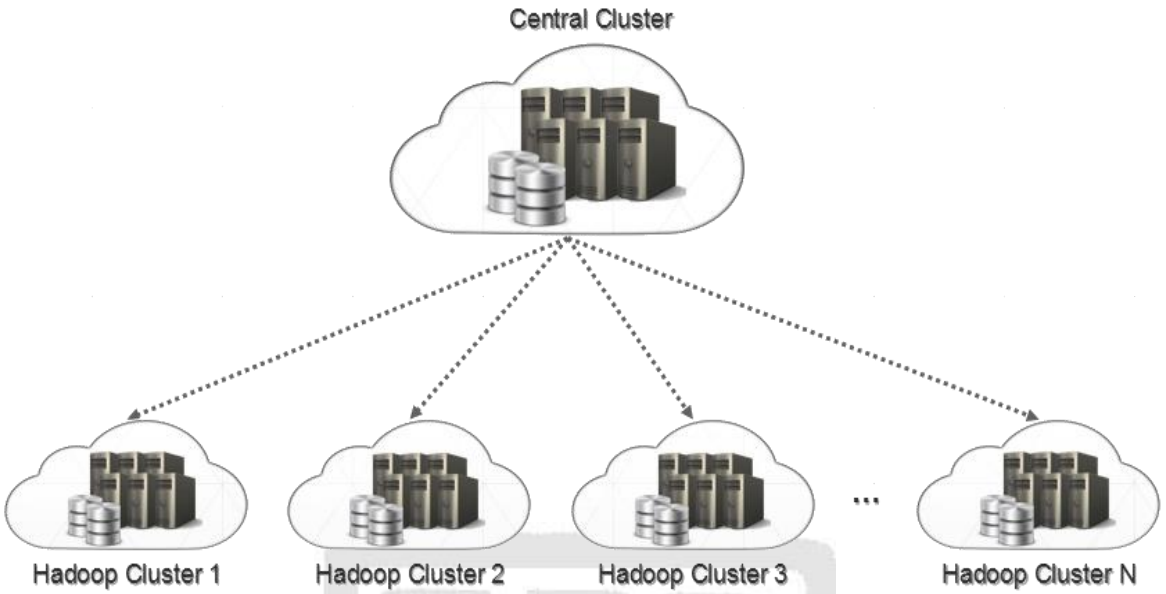

Figure 10: High-level view of multi-cluster computing

Fig. 10 is a high-level diagram of multiple distributing Hadoop clusters, which connected to a Wide-Area Network or Local Area Network among datacenters. We adopt a hierarchical master/slave architecture to manage separated HDFS among Hadoop clusters. The FedHDFS^{+} federates loosely-coupled HDFS, which provides users be able to achieve transparently global-view and enable to access geo-distributed data sets where different physical locations are. In other words, we proposed FedHDFS^{+} to make users no longer need to know where the geo-distributed datasets physically resided; with the central cluster management, it present a unique ability to aggregate separated HDFS image into one; therefore, it would be more beneficial to achieve data-intensive application analysis across multiple Hadoop clusters.

3.2 Design Issues

Our target environment for FedHDFS$^+$ is a system of multiple distributed heterogeneous Hadoop clusters. The storage layer of the single Hadoop cluster typically consist of common hardware interconnected with Local Area Network, and the file-system-layer of these multiple Hadoop HDFS typically connected with Wide-Area Network, hence we will detailed information about our proposed FedHDFS$^+$ for multi-cluster computing system. The main issues regarding how the FedHDFS$^+$ are designed to federate separated HDFS, and supports user transparency and system transparency. We concluded with following three main points when developing FedHDFS$^+$:

- How to light-weightly unify multiple HDFS across clusters? Compare to Gfarm gird file system, FedHDFS$^+$ is a lightweight middleware file system among clusters, was adopt primordial HDFS without any modification, and managed by our approach based on Master/Slave Hierarchical architecture. A master process will be running in the central cluster, and periodic communicate with each NameNode of Hadoop cluster.

- How to efficiently aggregate multiple unique namespaces into one? Global namespace is a concept that clearly delivers significant benefits. We implement a global namespace to provide a way to manage multiple Hadoop file system namespaces. It is particularly useful for distributed Hadoop clusters having multiple NameNodes, and hence multiple namespaces, in FedHDFS$^+$. In our solution is distinguish from client side mount tables of some Unix/Linux systems, collection of geo-distributed data is not we wanted, but correlation instead.

- How to perform sequential MapReduce jobs on multiple clusters? In some previous works of hierarchical MapReduce frameworks, was only perform single-stage MapReduce for multi-cluster computing. FedHDFS^{+} elastically support user-defined output data locations where physically resides on different clusters, hence, enable supports multiple-stage MapReduce processing on geo-distributed data.

3.3 System Architecture

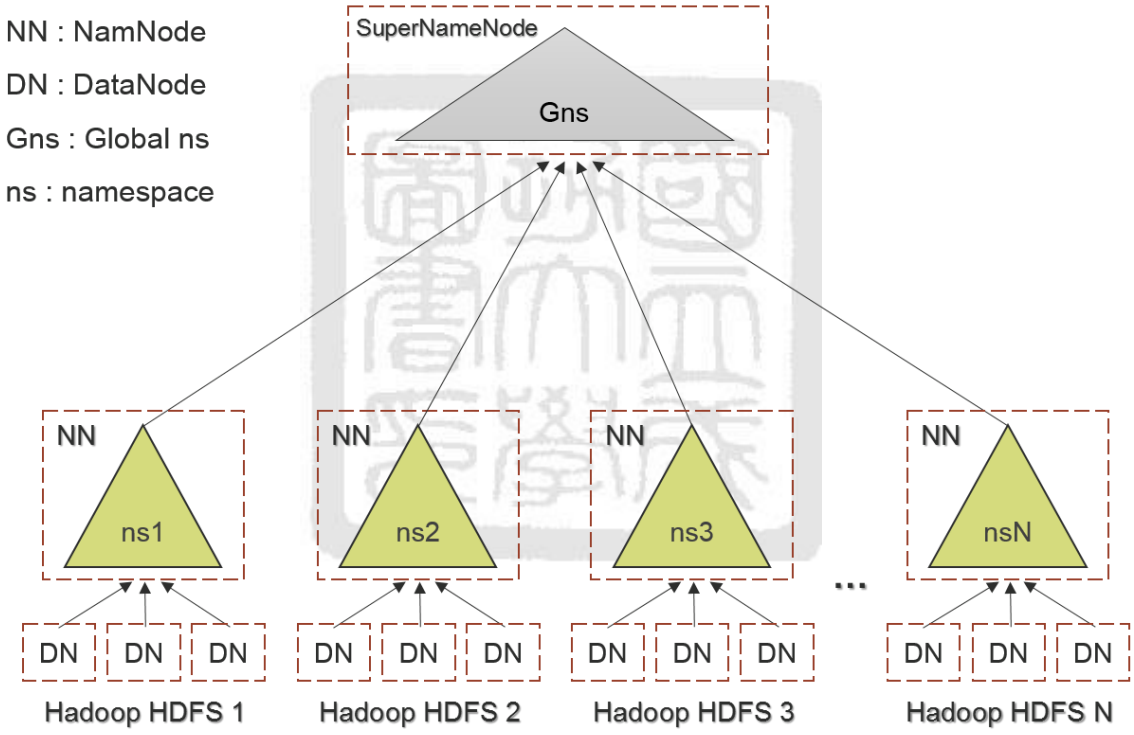

Figure 11: FedHDFS^{+} system architecture

Fig. 11 illustrates the overall system architecture and components of FedHDFS^{+}, was adopt master/slave architecture of original Hadoop distributed file system be used. A FedHDFS^{+} system consists of a single Master Node (SuperNameNode) be similar to NameNode, SuperNameNode is a lightweight process that running in the central cluster, and non-stop communicate with real NameNodes of each Hadoop cluster's HDFS. It also enable

to manage the global file system namespace, metadata and regulates access to distributed HDFS which is comprised of interconnected clusters of nodes where files and directories reside.

As usual, FedHDFS^{+} is similar to single distributed file system for users, we don't need to concern about where data located, or which Hadoop distribute file system that we want to use in multi-cluster computed process. In our main approach, SuperNameNode is play an important role to monitor multiple HDFS, and combine various namespace into one for data visualization, the most important goal is achieve data-location aware job submission for multi-cluster computing. Therefore, it provides scalability to reliably store and process large amounts of geo-distributed data, and Efficiency by distributing data and logic to process it in parallel on nodes where data is geographically located.

A. Master Server – SuperNameNode

So the SuperNameNode corresponds to the master machine and the NameNode to slave machines. Master server was collected HDFS status and fetch metadata information from each NameNode periodically, files and directories are also represented, like filename, path, number of replicas, owner, group, etc. The SuperNameNode maintains and executes related global file system namespace operations. If there are any modifications in the file system namespaces or in its properties, this is tracked by the SuperNameNode, like opening, closing, and renaming files and directories. It also determines the mapping of files to NameNode. The NameNode are responsible for receiving requests and sending to DataNodes, which serving read and write requests from the global file system's clients. Each of DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.

B. *FedHDFS+ Client*

User applications, such as MapReduce, access the global file system using the FedHDFS+ client, a library that exports the HDFS file system interface. Like most conventional file systems, FedHDFS+ supports operations to read, write and delete files, and operations to create and delete directories. The user references files and directories by paths in the global namespace. All HDFS communication protocols build on the TCP/IP protocol. FedHDFS+ clients also connect to a Transmission Control Protocol (TCP) port opened on the SuperNameNode, and then communicate with the NameNode using a proprietary Remote Procedure Call (RPC)-based protocol.

C. *Global namespace of FedHDFS+*

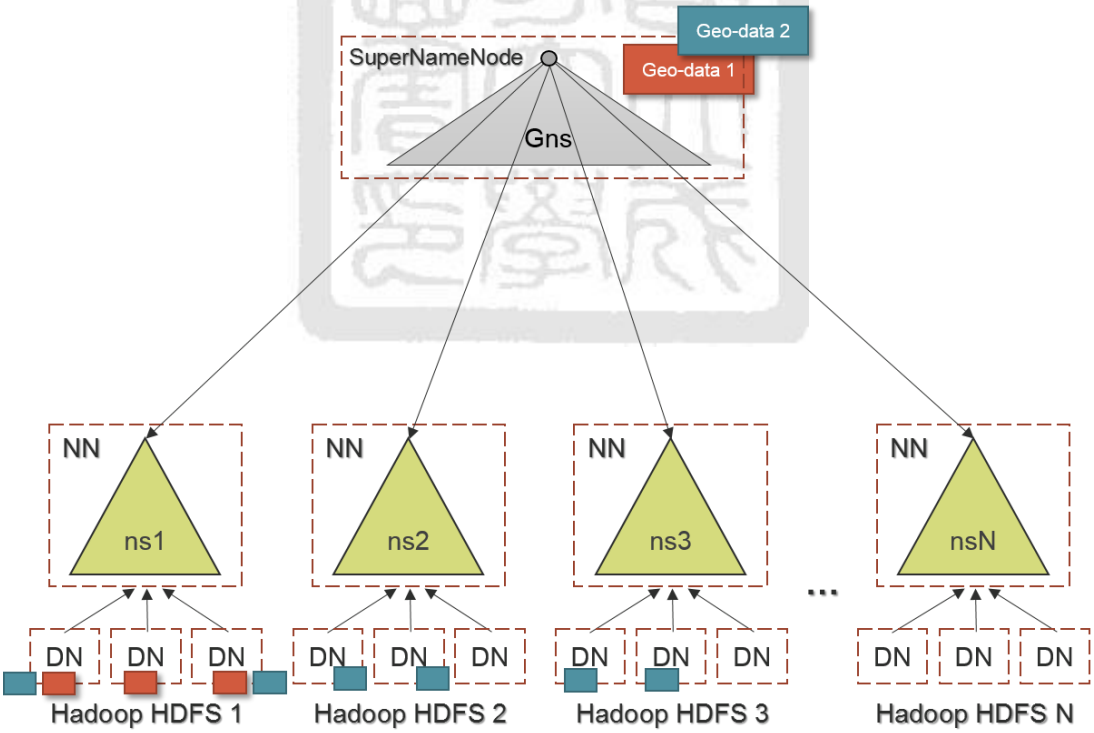

Figure 12: Global namespace of FedHDFS+

FedHDFS+ keeps the multiple file systems' metadata in memory, and supports a traditional hierarchical file organization. A user or an application can create directories and

store files inside these directories of global namespace. The global file system namespace hierarchy is similar to most other existing file systems; one can create and remove files, move a file from one directory to another, or rename a file. As illustrated in Fig. 12, Geo-data1 of the global namespace in SuperNameNode, it clearly to indicate geographically datasets where physically resides on Hadoop HDFS.

Bedsides, FedHDFS^{+} intelligently provides an extension to symbolic links (soft links) to able be union distributed datasets across Hadoop clusters. However, the HDFS architecture still not implementing these features. In computing, a symbolic link is a special type of file that contains a reference to another file or directory in the form of an absolute or relative path and that affects pathname resolution. On the other hand, it allows to associate multiple filenames with a single file. Symbolic links operate transparently for many operations: programs that read or write to files named by a symbolic link will behave as if operating directly on the target file. However, they have the effect of changing an otherwise hierarchical file system from a tree into a directed graph, so symbolic links are designed in FedHDFS^{+}, and provides transparent to users to operate on geographically distributed datasets from various HDFS. As showed in Fig. 12, Geo-data2 of the global namespace in SuperNameNode, it clearly to indicate geographically datasets where physically resides across Hadoop clusters.

The SuperNameNode maintains the global file system namespace. In detail, it typically integrated with the two main directory tree structure. The one is physical volume that mapping multiple various namespaces of each HDFS NameNode maintained. In addition, we don't adopt client side mount tables of some Unix/Linux systems, it is only collecting namespace without any correlation on geo-distributed datasets. The second one is logical volume that expand from FedHDFS^{+} client, users can create virtual files to linking different geographically datasets, and it makes users elastic to do some operations on geo-data across multiple datacenters. For user interface of FedHDFS^{+}, as showed in Fig. 13, users can visualize the global file system tree structure that contain physical drive and logical drive, in depth, each files or folders of these two drive will mapping to physically locations.

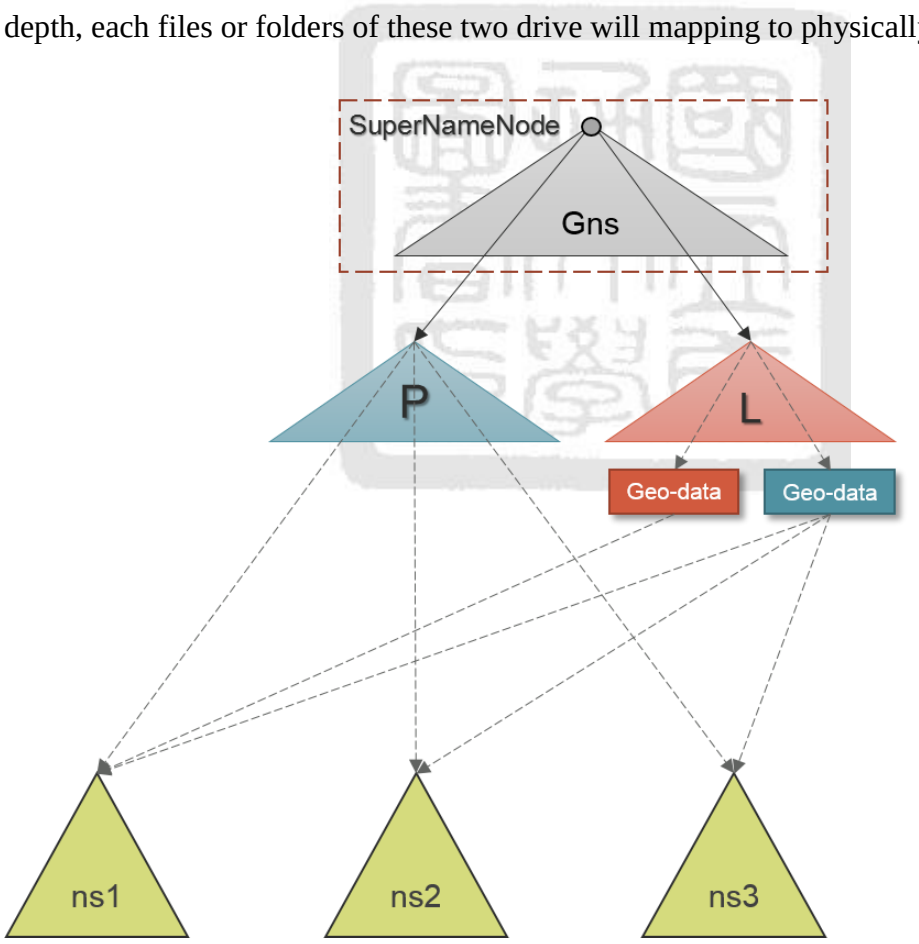

Figure 13: Global namespace tree structure

D. Location-aware job submission

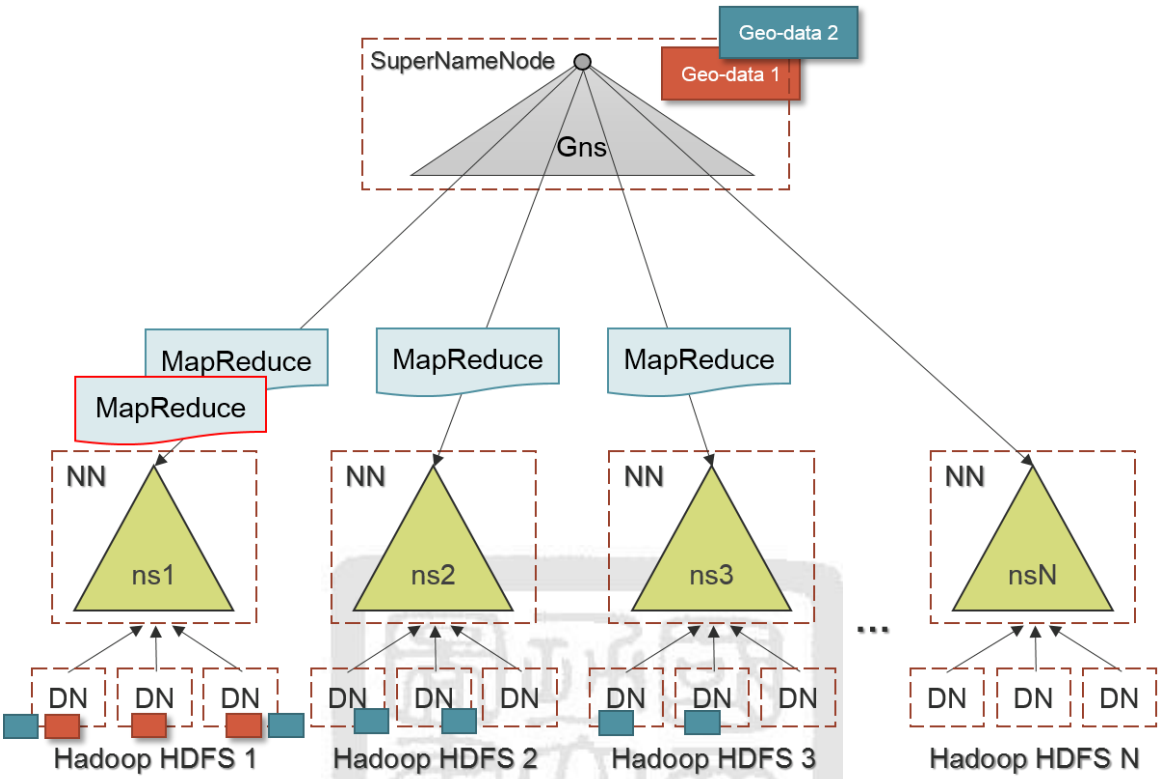

Figure 14: Location-aware job-submission diagram

In our approach as mentioned above, the SuperNameNode besides control slave NameNodes and maintain the global namespace, it also support data location-aware job submission, as showed in Fig. 14. Developer uses Hadoop command to submit MapReduce jobs to the SuperNameNode without any extra global functions, the next step, for the global namespace supported, when SuperNameNode received jobs which also contain the input data, the master server will looking up where the data actually resided, and deliver MapReduce jobs to each NameNode, to continue processing of each task on DataNodes.

In order to apply for multiple-stage MapReduce applications in multi-cluster computing area, we have classified different job's types, as Table 1 showing:

	Single Domain	**Cross Domain**
Original MapReduce (MR)	Single – MR Job	Multiple – MR Job
Federated MapReduce (Fed-MR)	Single – MR Job	Fed – MR Job

Table 1: Job classification

Real-world data is disorderly, we need to do several preprocessing stage for cleaning unstructured data on each cluster. For the preprocessing phase, geographically datasets maybe heterogeneous, hence we need to submit single – MR Job to clean data each Hadoop cluster; by the way, submit jobs will performed by SuperNameNode; until data already be homogeneous, we apply multiple – MR Job, which was triggering multiple cluster to process in the pipeline. Finally, we need to integrate data to do Fed-MR with hierarchical MapReduce framework, and provides cross-domain computing. FedHDFS^{+} elastically support data location-aware job submission across different clusters, and achieve data-intensive application on geo-data. Furthermore, it also supports different job's type for multiple-stage MapReduce processing on geo-distributed data.

Chapter 4 : Implementation

4.1 System Architecture Components

In this chapter, we will cover detailed system components of FedHDFS^{+}, the following Fig. 15 depicts system components. The bottom blocks describes the multiple Hadoop clusters which developments original HDFS. The upper block describes the FedHDFS^{+} Client and central Hadoop cluster which run SuperNameNode, there are three components or modules located in SuperNameNode: SNN, Global Namespace, and Job-Scheduler. In general, a daemon SuperNameNode will communicate with NameNodes of each Hadoop clusters, moreover, received some related global file system operations.

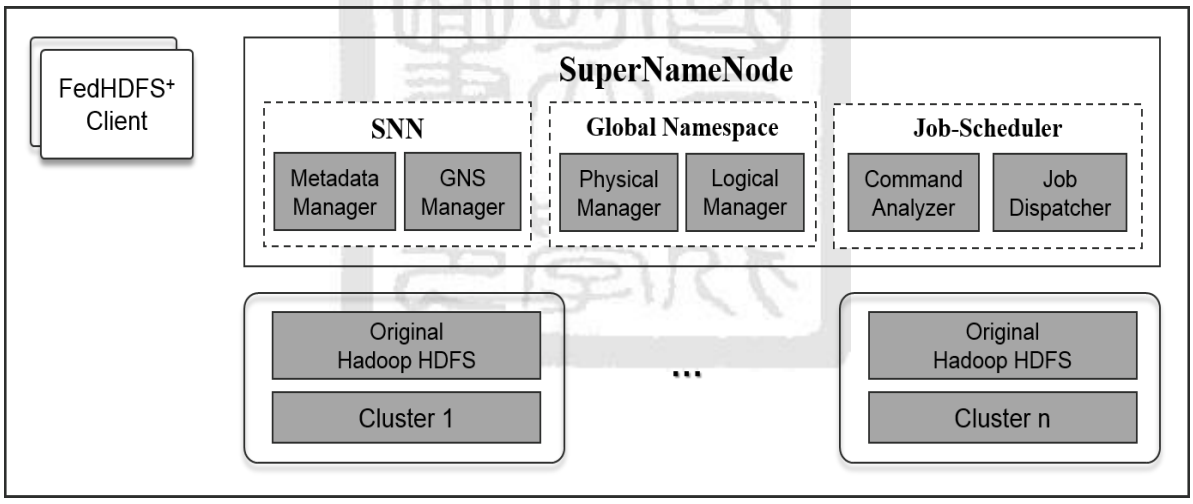

Figure 15: FedHDFS^{+} system components

4.2 SNN

A. Metadata Manager

In beginning, we need to setup system configuration for SuperNameNode. We use the Extensible Markup Language (XML) file to configure the federated Hadoop HDFS. Fig. 16 shows the FedHDFS$^+$ xml configuration. There are two types of block in XML file: SuperNamenode and cluster. SuperNamenode is setting for central Hadoop cluster, also setting a Transmission Control Protocol (TCP) port opened on the SuperNameNode, and then communicate with the inter process. Cluster is used for setting region clouds, which managed by central cloud. Based on setting of XML configuration, FedHDFS$^+$ will service and control multi-cluster Hadoop distributed file systems.

```
<FedHadoopClusters>

    <SuperNamenode>
        <Address> 10.3.1.34 </Address>
        <GNPort> 8765 </GNPort>
        <serializePort> 8764 </serializePort>
        <QueryPort> 8763 </QueryrPort>
    </SuperNamenode>

    <Cluster>
        <HostName> c02 </HostName>
        <fs.default.name> 10.3.1.2:39100 </fs.default.name>
        <hadoop-home.dir> /home/hpds/hadoop-2.4.1 </hadoop-home.dir>
    </Cluster>

    </Cluster>
        …
    </Cluster>

</FedHadoopClusters>
```

Figure 16: SuperNameNode configuration

There is one metadata manager per SuperNameNode that actually collect namespace metadata of each HDFS NameNode, such as filenames, directories, access permissions, and file layout. The entire single file system namespace, including the mapping of blocks to files and file system properties, is stored in a file called the FsImage, as showed in Fig. 17. Metadata manager collected some values of FsImage, and stored in memory. However, unlike block-based distributed file systems, such as GPFS and HDFS, where the master server controls all of the block allocation with a heavy burden on memory. The metadata manager is only collects in related files/folders information periodically, we defined default value is one minute. The metadata manager also periodic fetches HDFS statistics, such as HDFS remaining, HDFS used, HDFS capacity.

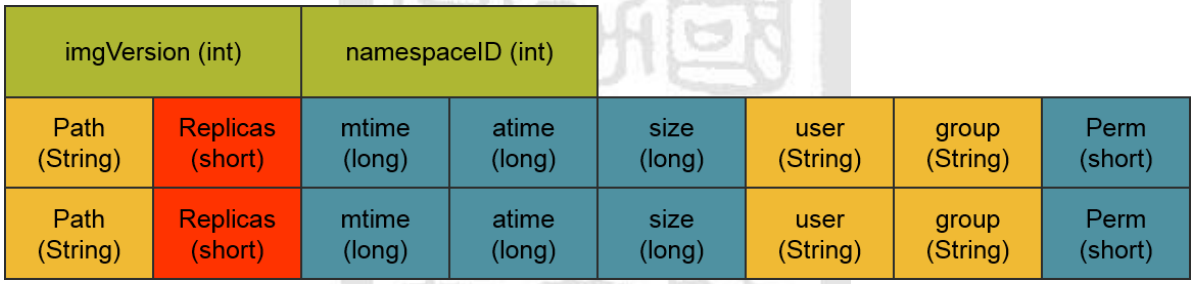

Figure 17: FsImage record

B. GNS manager

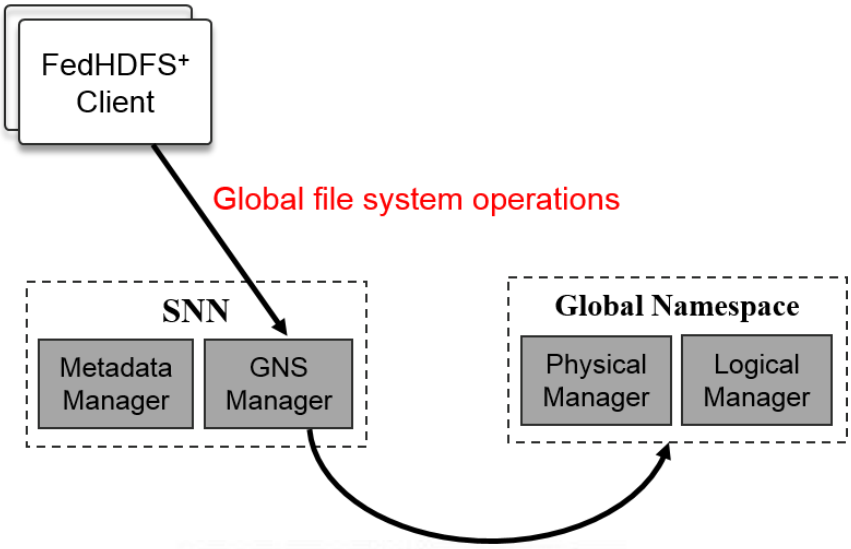

Figure 18: Global namespace manager

As showed in Fig. 18, the global namespace manager receives requests from FedHDFS+ clients in the form of Hadoop RPC requests over a TCP connection. FedHDFS+ clients perform global file system metadata operations through the global namespace manager, typical client requests include creating, opening, closing, and renaming files and directories, etc. We also implementing soft link, which allows to associate multiple files with a single file among distributed Hadoop clusters. Symbolic links operate transparently for multiple file systems.

Besides, the FedHDFS+ global namespace is stored by the SuperNameNode. The SuperNameNode keeps an image of the entire global file system namespace and in memory. It's important that this metadata are safely persisted to stable storage for fault tolerance. At a high level, the global namespace manager's secondary responsibility is storing the HDFS namespace. This process is called a checkpoint, we use that Java provides a mechanism, called object serialization where an object can be represented as a sequence of bytes that includes the object's data as well as information about the object's type and the types of data

stored in the object. In the current implementation, a checkpoint only occurs when the SuperNameNode starts up. Work is in progress to support periodic checkpointing in the near future, we defined default values is one minutes. When the SuperNameNode starts up, it reads the metadata from disk, applies all the information to the in-memory representation of the FsImage, and flushes out this new version into a new FsImage on disk.

4.3 Global Namespace

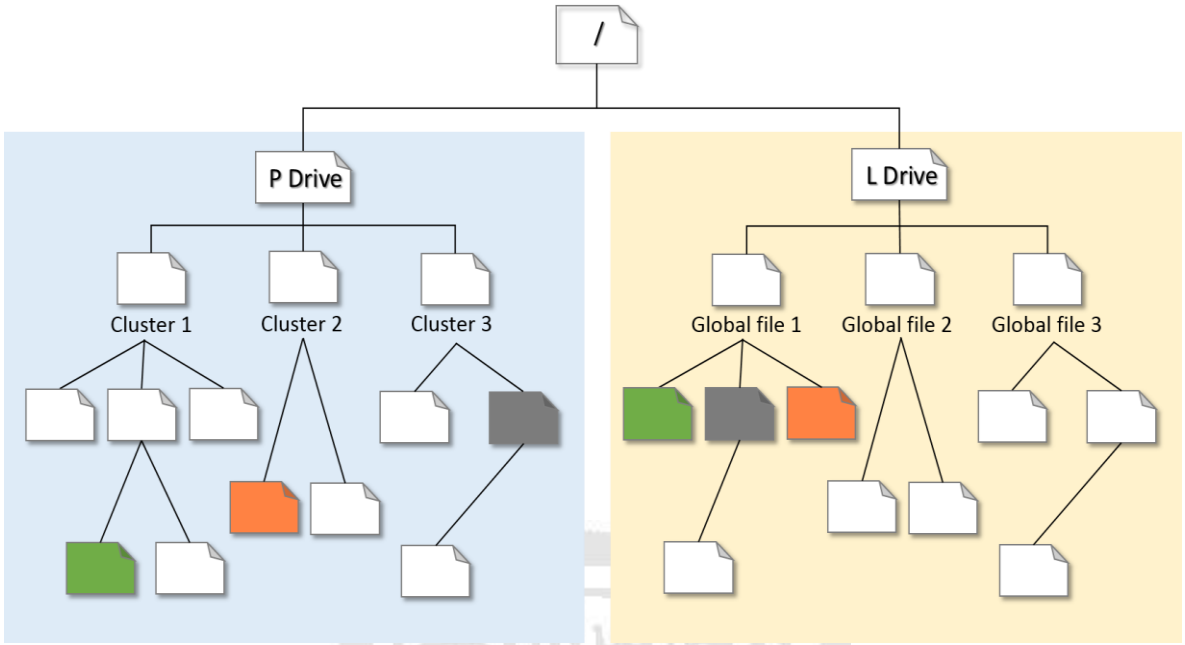

Figure 19: Global file system tree structure

This level of the global file system provides the view of multiple physical Hadoop distributed file systems. For the user-level of FedHDFS^{+}, as showed in Fig. 19, global file system tree structure will decompose into two subtree, the left side of tree, is a physical drive which contain file system hierarchy of each Hadoop HDFS, the list of files or folders are mapping to physical locations; the right side of tree, is a logical drive which contain global files (also logical files), each logical file is mapping to physical files across clusters of the left side, Besides, in the global namespace module is consist two main manager, physical drive is managed by physical manager, and logical drive is managed by logical manager, we will detailed information in each manager.

A. *Physical Manager*

Physical File / Folder	Host, Len, Time, Replica, Owner, Group, Permission
File B	Cluster 02, 241567, 20150712, 3, hpds, supergroup, r-x--x---
File C	Cluster 13, 125489, 20150701, 2, hpds, supergroup, rwx-w-r-x
File D	Cluster 07, 36254, 20150715, 1, hpds, supergroup, rwx--x-w-
Folder A	Cluster 19, 0, 20150729, 0, hpds, supergroup, rwx--x--x
Folder B	Cluster 13, 0, 20150712, 0, hpds, supergroup, rwx--x---
Folder C	Cluster 09, 0, 20150718, 0, hpds, supergroup, rwx--x--x

Physical Table

Figure 20: Content of physical table

HDFS metadata represents the structure of HDFS directories and files in a tree. It also includes the various attributes of directories and files, such as ownership, permissions, quotas, and replication factor. All examples shown are from testing a build of the soon-to-be-released Apache Hadoop 2.4.1.

The physical manager primary responsibility is receiving the each HDFS metadata from the SNN's metadata manager collected, and actual booking work for the SuperNameNode. It tracks important table which is physical file or folder mapping to file's metadata, as illustrated in Fig. 20, include filename, foldername, length, access time, modification time, number of replication, which owner, group and permission.

B. Logical Manager

Global File	Physical File / Folder	Len, Time, Replica, Owner, Group, Permission
Global File 2	Folder A, Folder B	841567, 20150715, hpds, supergroup, rw--xr-x
Global File 3	File B, File D	625142, 20150701, hpds, supergroup, rwx--x-w-
Global File 4	File C, Folder B	3651521, 20150712, hpds, supergroup, rwx--x--x
Global File 5	File D, Folder C	6548541, 20150718, hpds, supergroup, rwxr-x---

Figure 21: Content of logical table

Logical implies a higher view than the physical. Users relate to data logically by data element name; however, the actual fields of data are physically located in different clusters' HDFS. The logical manager primary responsibility is receiving the global file system operations from the FedHDFS^{+} clients, and immediately construct logical table as showed in Fig. 21. The logical manager tracks important table which is logical file mapping to physical file or folder or file & folder across HDFS.

In FedHDFS^{+} file system operations, a special command is symbolic link called "Union", which is a type of file that contains a reference to another file or directory in the form of an absolute or relative path and that affects pathname resolution. On the other hand, it allows to associate multiple filenames with a single file by users. Union operate transparently for global files' operations across multiple clusters: programs that access to files named by a symbolic link will behave as if operating directly on the target files among clusters.

4.4 Job-Scheduler

A. Command Analyzer

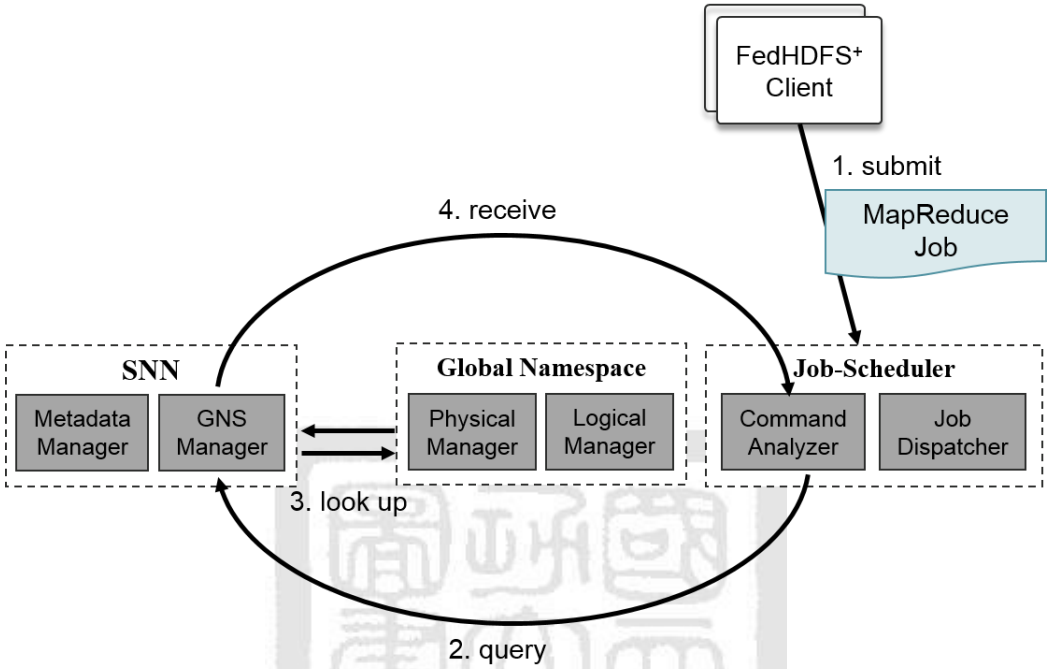

Figure 22: Workflow of job-submission

submit jar [jarName] [calssName] [globalInputFile] [globalOutputFile]

In order to achieve multi-cluster computing for transparently, it means that we don't need to define tedious configuration about dataset locations, we proposed command analyzer to look up global input file and global output file, and submit MapReduce jobs to each cluster which maintain data actually. As illustrated in Fig. 22, firstly, user submit their jobs to the central cluster. When the command analyzer of SuperNameNode receive MapReduce jobs that include Global file, it sends global file name to Global namespace manager to look up the global namespace and query the physically data locations. Finally, the command analyzer will received physical data location from the global manager, these information also include that data is resides on which clusters to need to be submitted. Besides, there is several type of jobs need to define when submit jobs to clusters before, so we will illustrate in next section.

B. Job-Dispatcher

	Single Domain	**Cross Domain**
Original MapReduce (MR)	Single – MR Job	Multiple – MR Job
Federated MapReduce (Fed-MR)	Single – MR Job	Fed – MR Job

Table 2: Types of Job

For multiple-stage applications, we need to define three main job's type: single-MapReduce, Multiple-MapReduce and Federated-MapReduce. Single – MR Job and Multiple – MR Job is depend on that need to submit to single cluster or multiple clusters, besides Fed – MR Job is adopt the hierarchical MapReduce framework in our previous work. Therefore, when geographically datasets need to do different processing phase, we can submit Single – MR Job to the cluster, however data already become homogeneous datasets, and we may do another prepossessing phase, until data need to aggregate to cooperate. We will submit Fed – MR Job to execute hierarchical MapReduce framework and trigger all Hadoop clusters which maintain data. In previous multi-cluster computing system, these different type of jobs need to supply command flag by users. As previously mentioned, with our FedHDFS^{+} support, it can automatically decide which type from input file and output file and complete submission. As Showed below:

*submit jar [jarName] [calssName] [**physicalPath**] [**physicalPath**]*

Single – MapReduce Job

*submit jar [jarName] [calssName] [**logicalPath**] [**logicalPath**]*

Multiple – MapReduce Job

*submit jar [jarName] [calssName] [**logicalPath**] [**physicalPath**]*

Federated – MapReduce Job

Chapter 5 : Experimental Evaluation

In this chapter, we will evaluate some applications on our proposed system: FedHDFS$^+$. In order to present our system is compatible with our previous work: Fed-MR framework, we evaluate FedHDFS$^+$ based on two applications. We also compare the multiple-stage application performances when running on Fed-MR framework without FedHDFS$^+$, and with FedHDFS$^+$. Afterwards, we present one focused investigations on how Fed-MR framework can benefit from the built-with FedHDFS$^+$ among Hadoop clusters and automatically select Top Cloud mechanisms in Region Clouds.

5.1 Environments

Hardware	Content
CPU model	Xeon E3-1230 v3
Number of CPU	1
Total Number of Cores	4 (8 Hyper-thread)
CPU Speed	3.3 GHz
Memory	16 G
Network	Gigabit Ethernet
Hard Disk	1 T

Table 3: Hardware specification

Software	Version
Operation System	Ubuntu Server 14.04 LTS
Linux Kernel	Linux 3.13.0-29-generic
Java Version	Java (TM) SE 1.8.0_20
Hadoop Version	Hadoop 2.4.1 (stable release)

Table 4: Software specification

5.2 Deployment of multiple Hadoop Clusters

In our target Hadoop cluster environment, we deployed 4 Hadoop clusters, each cluster contain 1 NameNode and 5 DataNodes. Besides, we only have single datacenter, it means that inter cluster connected in the Local Area Network, those data workers connected via Gigabit network. Hence, we also simulate those clusters distributed in the Wide Area Network. Network traffic control is the one way to limited network bandwidth in Linux. For the intra-cluster, we can limit each worker's container outgoing and incoming bandwidth by setting the tc. For the inner-cluster, we adopt local area network. As bellow showing:

Type	# cluster	Network	# worker	# cores
1	4	LAN (10MB/s)	5 x 4	5 x 4 x 4
2	4	WAN (100MB/s)	5 x 4	5 x 4 x 4

Table 5: Hadoop cluster specification

5.3 Applications and Dataset

A. Applications

"WordCount" application is a simple program which reads text files and counts how often words occur. Each mapper task takes a line as input and splits it into words, and form the initial key/value pair as <"word", 1>. In the reduce phase the keys are grouped together and the values for similar keys are summed. "*SearchWord"* application is designed to search input for appearances of strings. This tool is designed to be used to search for one or more block in files, but can be used for general text search, assuming the search strings don't contain tokens. It assumes only one search string will appear per line.

For multi-stage application, we exploit a *"WordCount"* application be a preprocessing phase. It simulate that amount of massive data need to clean firstly. When data already do aggregation phase, we adopt "*SearchWord"* to filter words that we don't need it in the result.

B. Datasets

We use Common Crawl dataset, which contains approximately 6 billion web documents stored on a publicly accessible, scalable computer cluster. Common Crawl is a nonprofit organization that crawls the web and freely provides its archives and datasets to the public. It contains raw web page data, extracted metadata and text extractions. Common Crawl currently stores the crawl data using the Web ARChive (WARC) format. The WARC format allows for more efficient storage and processing of CommonCrawl's free multi-billion page web archives, which can be hundreds of terabytes in size.

Dataset	Data Size
bin1	100 GB
bin2	200 GB
bin3	300 GB
bin4	400 GB

Table 6: Common Crawl Datasets of evaluation

5.4 Evaluation 1: Multiple-Stage MapReduce Applications

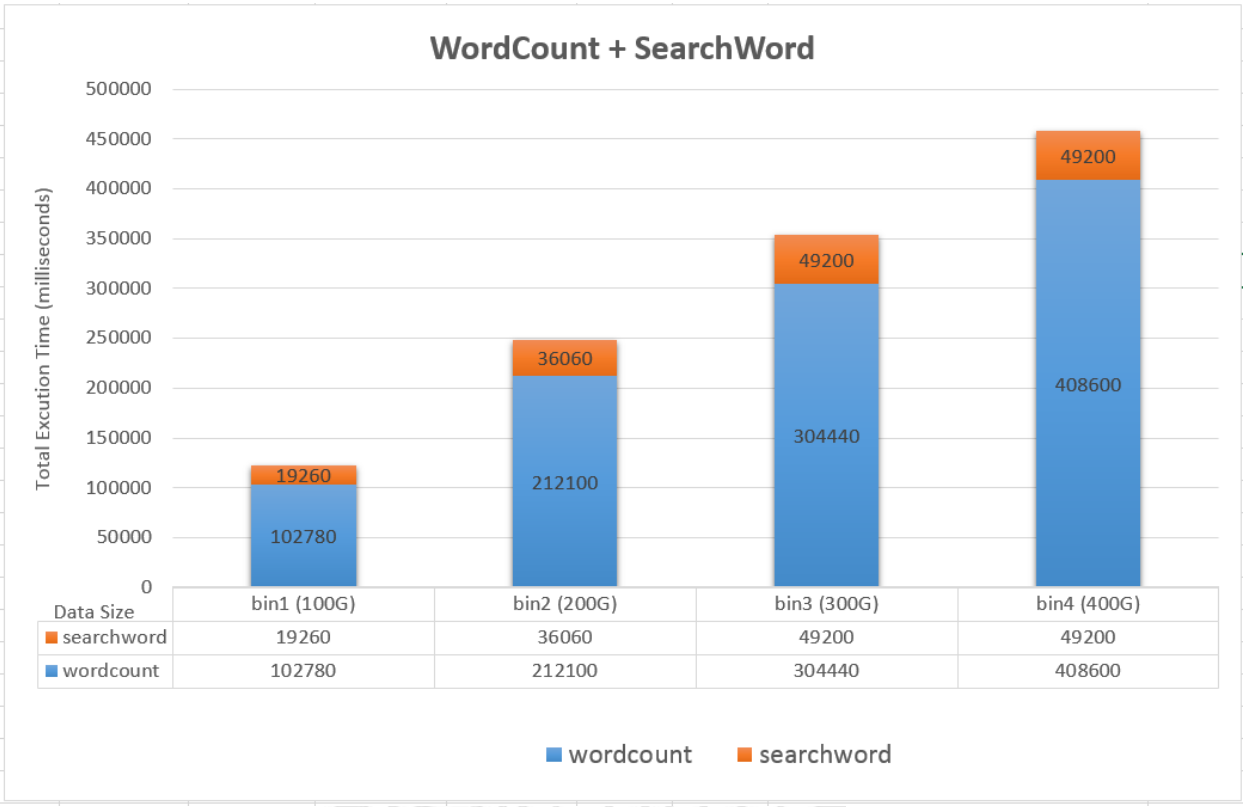

Figure 23: Multiple-stage MapReduce applications

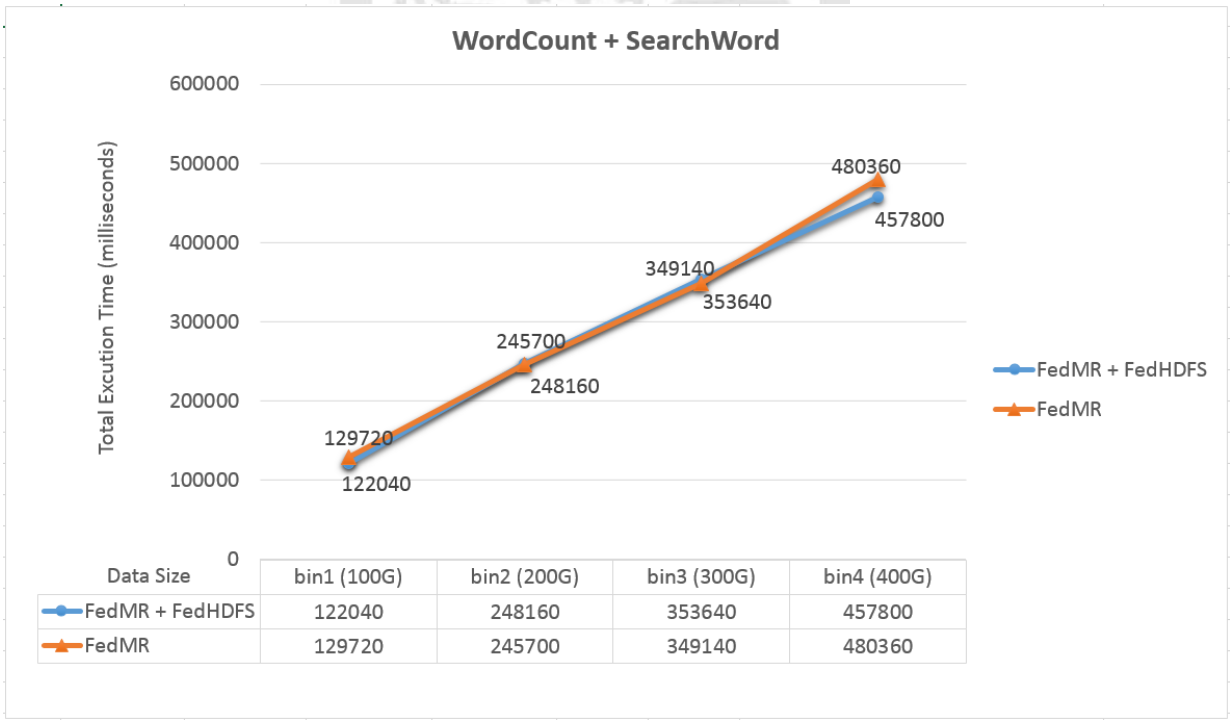

Figure 24: Compare Fed-MR and Fed-MR with FedHDFS^{+}

FedHDFS$^+$ in our proposed, it transparently supports multiple-stage applications. The main goal is for real-word data, was need to do several preprocessing previously, and we supply transparent global file system to submit next-stage multi-cluster Federated MapReduce job without any extra configuration or setting. In other words, FedHDFS$^+$ is compatible with Fed-MR completely. As Fig. 23 shown, our system be enable successfully complete the multiple-stage MapReduce jobs on different datasets. In the first stage, data need to processing much time, because data sill huge. When data-clean finished, data is mostly reduce, and also need to cooperate across cluster, we do Fed-MR Jobs as the final operation.

In previous work, Fed-MR have two way for multi-stage Applications without FedHDFS$^+$, first, it needs to run Fed-MR job at first stage, that cause amount of time consuming to transfer data, besides, the data will aggregate to the certain cluster. When these data need to do next stage for analyze, it will not perform well on performance, because only processing on a single cluster, instead of multi-cluster computing. Second, Fed-MR still can work multiple-MapRedcue, but it have to tedious ssh to each cluster and submit job. Besides, when they run Fed-MR jobs before, still defined some configurations for system. As showed in Fig. 24, although the two lines are close, but with our FedHDFS$^+$ proposed, it makes more transparently, elastically and atomically for users to run multiple-stage applications.

5.5 Evaluation 2: Fed-MR – Top Cloud Selection (WAN / LAN)

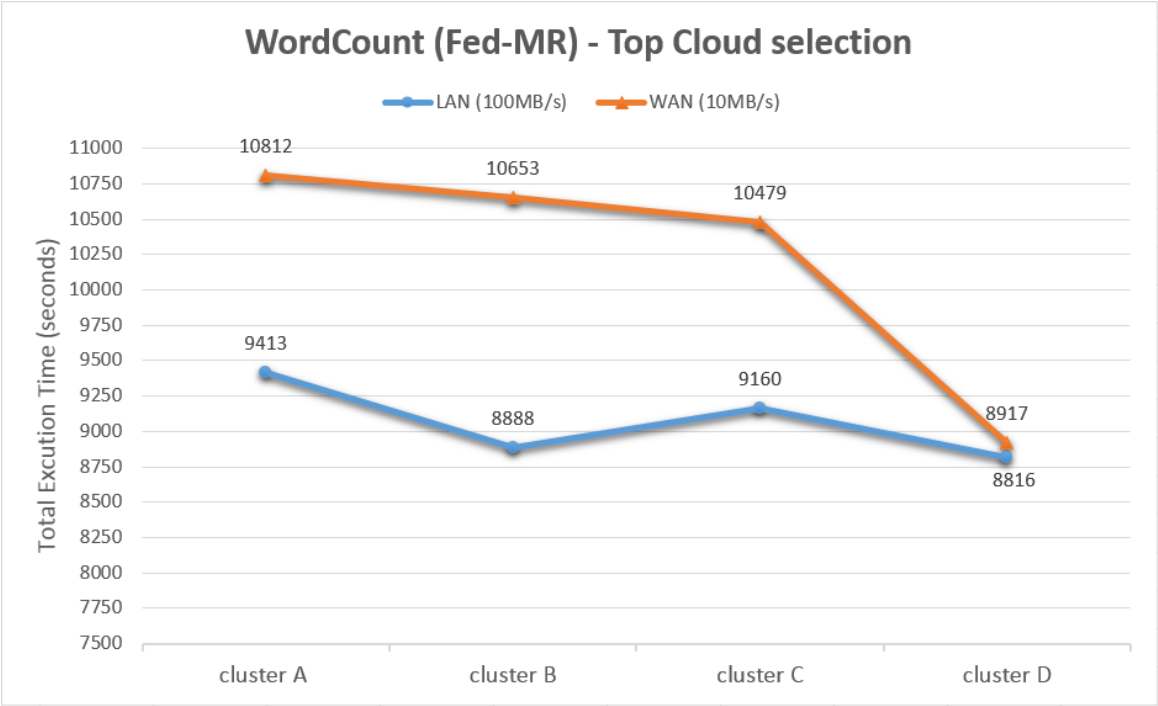

Figure 25: Top Cloud Selection (LAN and WAN)

Cluster	Data Size
cluster A	20 GB
cluster B	40 GB
cluster C	60 GB
cluster D	80 GB

Table 7: Data size of each cluster 1

Fed-MR based on the hierarchical MapReduce framework, which was indicate that final output data will aggregate to a central cluster, we called Top Cloud (as mentioned in chapter 2. Related work). In original Fed-MR, it needs to set Top Cloud and Regions Clouds by users. However, we discover that different data size contained by Top Clouds, will improve different performance. As showed in Fig. 25, below the line is perform Fed-MR in the LAN environment, above the line is simulating real situation in a WAN environment. The final result indicates that Top Cloud maintain bigger datasets will perform good performance among distributed clusters which in the WAN environment.

As a result, we breakdown the Fed-MR into three main phase: Region Cloud execution time, Distcp time (transfer time), Top Cloud execution time, as illustrated in Fig. 26. We clearly find that time consuming will occurs in Region time + Distcp time. For the bigger size (400G) of the right side in the Fig. 26, when select maximum data size of the Region Clouds, it will be able to reduce almost 1 hour to complete WordCount MapReduce job, which is compare minimum data size of the Region Clouds. As a result, in our FedHDFS^{+} supported, when user submit Fed-MR jobs to integrates data across Hadoop clusters, FedHDFS^{+} also enable to atomically select the Top Cloud which is maintains maximum data size, and achieve well-performance.

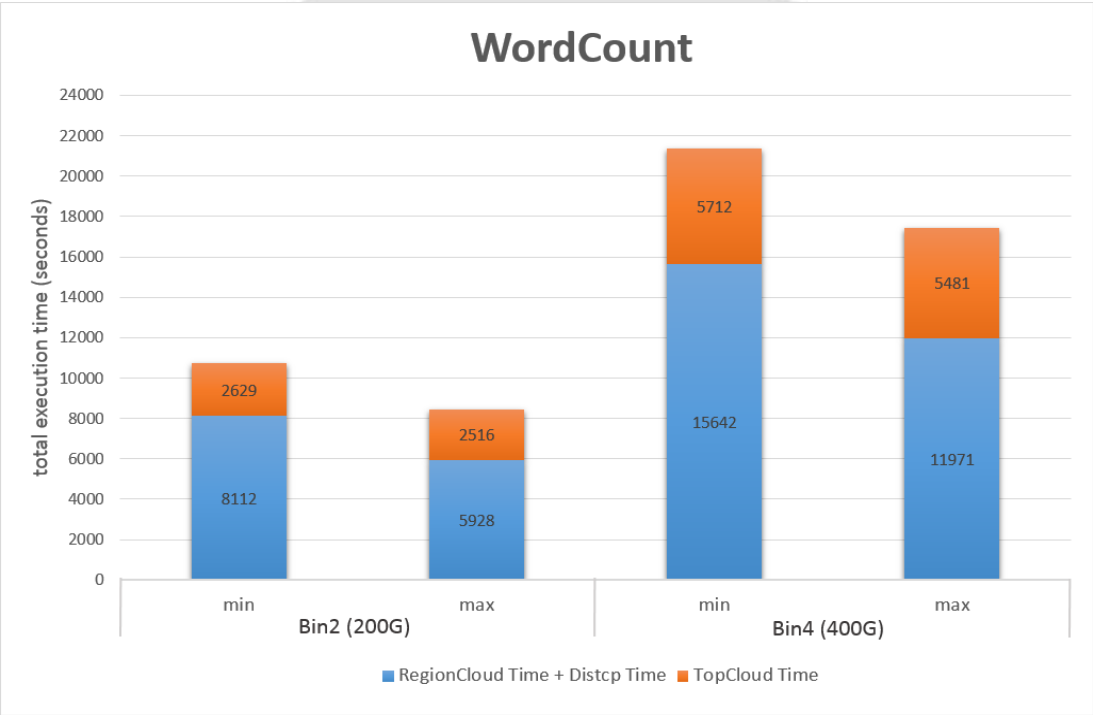

Figure 26: Compare different data size

	Sub data size of Bin2	Sub data size of Bin4
cluster A	20 GB	40 GB
cluster B	40 GB	80 GB
cluster C	60 GB	120 GB
cluster D	80 GB	160 GB

Table 8: Data size of each cluster 2

5.6 Discussion on Top Cloud Selection

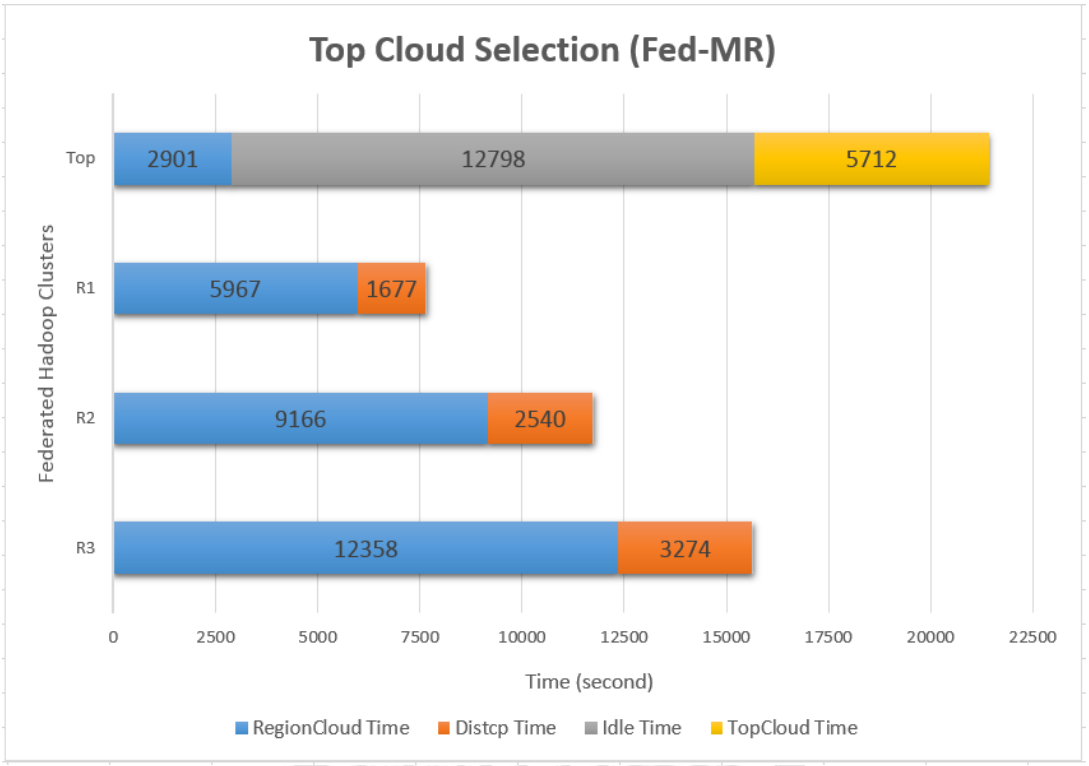

Figure 27: Fed-MR job without optimization

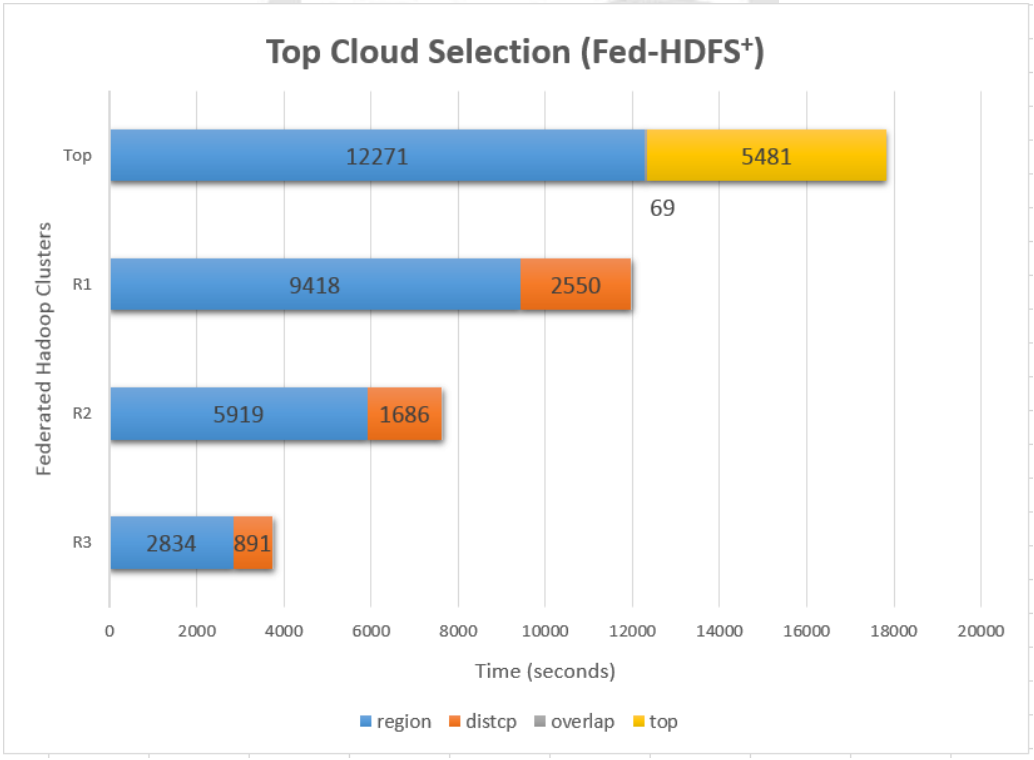

Figure 28: Fed-MR job with optimization

In this section, we will discussing the performance which is related to “how to select Top Cloud of Region Clouds”. In addition, for FedHDFS^{+} supported, it can dynamically select Top Cloud before Fed-MR job submit, and help us to get optimum performance for Fed-MR framework, as showed in Fig. 30.

As mentioned above, the reason that effect Fed-MR overall performance is data transfer overhead. In detail, it also occurs another issue about select wrong Hadoop cluster, which will idle whole cluster at a period of times. As showed in Fig. 29, unfortunately, Top Cloud will finish in region-execution-phase: map/proxy reduce, and then wait another clusters still operating or transferring to Top Cloud and causing idle of Top Cloud.

Chapter 6 : Conclusion and Future Work

With the increasing volume of geographically distributed data which stores across multiple datacenters. Multi-cluster computing is more and more popular and requirements for organizations. In this paper we proposed Federated Hadoop Distributed File System (FedHDFS$^+$) in multi-cluster comping system, the main core was unified multiple HDFS across Hadoop clusters, and provides single (global) Hadoop HDFS image, which efficiently moving computation near the data. It is transparent to users to run Cross-Domain Applications. We also support multiple-stage MapReduce applications for multi-cluster computing frameworks.

FedHDFS$^+$ is potentially to play the important role of file-system-level in multi-cluster computing area, it makes users don't need to traverse multiple Hadoop distributed file system, and provides data-location aware jobs submission for cross-domain data-intensive applications, it takes multi-cluster computing systems more elastically, availability and transparently. In addition, FedHDFS$^+$ is compatible with our previous work: Fed-MR framework, we provides dynamically Top Cloud selection mechanism before processing Fed-MR jobs. As a result, in our FedHDFS$^+$ supported, it helps us to get optimum performance for Fed-MR framework.

In the future work, there some inadequacies that we will implement to further enhance our system. We will also integrated with Fed-RDD, which is another multi-cluster computing framework base on spark. We will enhance FedHDFS$^+$ to be more robust with more convenient user interface.

References

[1] Manyika, James, et al. "Big data: The next frontier for innovation, competition, and productivity." (2011).

[2] Hadoop. Avaliable from: https://hadoop.apache.org/

[3] Apache Software Foundation. Avaliable from: http://www.apache.org/

[4] Shvachko, Konstantin, et al. "The hadoop distributed file system." Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on. IEEE, 2010.

[5] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." Communications of the ACM 51.1 (2008): 107-113.

[6] Hortonworks. Avaliable from: http://hortonworks.com/

[7] Microsoft. Avaliable from:

http://azure.microsoft.com/en-us/solutions/hadoop/

[8] Facebook. Avaliable from: http://hortonworks.com/big-data-insights/how-facebook-uses-hadoop-and-hive/

[9] Cloudera. Avaliable from: http://www.cloudera.com/content/cloudera/en/about/hadoop-and-big-data.html

[10] Agarwal, Sharad, et al. "Volley: Automated Data Placement for Geo-Distributed Cloud Services." NSDI. 2010. Wang, Ping, et al. "Honeypot detection in advanced botnet attacks." International Journal of Information and Computer Security 4.1 (2010): 30-51.

[11] Tudoran, Radu, Gabriel Antoniu, and Luc Bouge. "SAGE: Geo-Distributed Streaming Data Analysis in Clouds." Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International. IEEE, 2013.

[12] Kakade, Kirtimalini N., and T. A. Chavan. "Improving Efficiency of GEO-Distributed Data Sets using Pact." (2014).

[13] Luo, Yuan, and Beth Plale. "Hierarchical mapreduce programming model and scheduling algorithms." Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012). IEEE Computer Society, 2012.

[14] Wang, Chun-Yu, et al. "Federated MapReduce to Transparently Run Applications on Multicluster Environment." Big Data (BigData Congress), 2014 IEEE International Congress on. IEEE, 2014.

[15] Anderson, O. Ted, et al. "Global namespace for files." IBM systems Journal 43.4 (2004): 702-722.

[16] Moore, Reagan, et al. "Data-intensive computing." The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann (1999): 105-129.

[17] Zaharia, Matei, et al. "Spark: cluster computing with working sets." Proceedings of the 2nd USENIX conference on Hot topics in cloud computing. Vol. 10. 2010.

[18] Buyya, Rajkumar. "High performance cluster computing." New Jersey: F'rentice (1999).

[19] Wang, Lizhe, et al. "G-Hadoop: MapReduce across distributed data centers for data-intensive computing." Future Generation Computer Systems 29.3 (2013): 739-750.

[20] Tatebe, Osamu, Kohei Hiraga, and Noriyuki Soda. "Gfarm grid file system." New Generation Computing 28.3 (2010): 257-275.

[21] Jayalath, Chamikara, Jose Stephen, and Patrick Eugster. "From the cloud to the atmosphere: running MapReduce across data centers." Computers, IEEE Transactions on 63.1 (2014): 74-87.