sidHadoop; Secure Inter-Domain Hadoop

Author

Muhammad Obaid ur Rehman

NUST201464152MSEECS63114F

Supervisor

DR. SHAHZAD SALEEM

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SECURITY (MS-IS)

DEPARTMENT OF COMPUTING (DoC)

SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE (SEECS)

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY (NUST)

ISLAMABAD, PAKISTAN

(July 2018)

Declaration

I certify that this research work titled "*sidHadoop; Secure Inter-Domain Hadoop*" is my own work. The work has not been presented elsewhere for assessment. The material that has been used from other sources it has been properly acknowledged / referred.

Signature of Student

Muhammad Obaid ur Rehman

NUST201464152MSEECS63114F

THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS thesis written by Mr. Muhammad Obaid ur Rehman, (Registration No NUST201464152MSEECS63114F), of SEECS (School/College/Institute) has been vetted by undersigned, found complete in all respects as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial fulfillment for award of MS/M Phil degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the said thesis.

Signature: ________________________________

Name of Supervisor: _Dr. Shahzad Saleem______

Date: ___________________________________

Signature (HOD): __________________________

Date: ___________________________________

Signature (Dean/Principal): __________________

Date: ___________________________________

Approval

It is certified that contents and form of thesis titled **“sidHadoop; Secure Inter-Domain Hadoop”** submitted by **Mr. Muhammad Obaid ur Rehman**, have been found satisfactory for the requirements of the degree.

Advisor: ___**Dr. Shahzad Saleem**___

Signature: ____________________

Date: ________________________

Committee Member 1: ___**Mr. Fahad Satti**___

Signature: ____________________________

Date: _______________________________

Committee Member 2: ___**Ms. Hirra Anwar**___

Signature: _____________________________

Date: ________________________________

Committee Member 3: __**Dr. Muddassir Malik**_

Signature: _____________________________

Date: ________________________________

Copyright Statement

Acknowledgements

I am very thankful to my Allah Almighty for all his help and blessings in every stage of my life.

I am also thankful to my parents, my sisters and to my brother for supporting and encouraging me throughout my life.

I would like to give special thanks to my respected sir and supervisor Dr. Shahzad Saleem for his help throughout my thesis. Besides my supervisor, I am also very thankful to GEC Committee members: Mr. Fahad Satti, Miss Hirra Anwar and Dr. Muddassir Malik for their efforts, encouragement, and support to overcome numerous obstacles I have been facing throughout my research.

Finally, I would like to express my gratitude to all the individuals who have rendered valuable assistance to my study.

This work is dedicated to my exceptional parents and adored siblings whose tremendous and unconditional support and for my respected teachers whose virtuous guidance led me to this wonderful accomplishment.

Abstract

Hadoop is a big-data processing framework which is widely used for data storage and processing. Now-a-days security is one of the major concerns in the digital world. Any system is only considered reliable when it provides proper measures to secure the valuable data of an organization. Due to the vast popularity and success of Hadoop framework, its use cases started to evolve from an in-house deployment to grid, cloud and other heterogeneous environments. Researchers have provided some solutions to access geographically distant resources for Hadoop computation and storage, utilizing different techniques and frameworks. Due to security and design issues in those frameworks, we proposed to deploy Hadoop in inter-domain environment.

Inter domain communication can help in collaboration without actually sharing the large amounts of data between independent Hadoop clusters. If the need to scale resources is temporary and or the resources are geographically distributed then sidHadoop can help to securely share resources of Hadoop clusters. One Hadoop cluster cannot communicate with another Hadoop cluster in the current out-of-the box setup. The proposed solution is working to achieve secure communication between two independent Hadoop clusters. For abstraction and security purpose the resources are not delegated to foreign cluster instead the master nodes communicate over WAN and post jobs for each other. The jobs are run within a cluster just like a single independent Hadoop setup. This way, the Hadoop core features are not disturbed and the benefits of Hadoop are still achieved.

Our solution helps in the collaboration among different Hadoop clusters. It has use cases in academia and business world. It can ease the collaboration of resources of organization with multiple Hadoop deployments that are geographically distributed. It can help to utilize/control/manage all these deployments from one single location. Similarly, different educational institutions having their Hadoop clusters and collaboration agreement with other institutions will be able make use of data and/or resources of inter-institute Hadoop clusters in a secure manner.

Key Words: *Hadoop, inter-domain, End-point security, Channel Security, WAN, SSL, Mutual Authentication, Web Services, Geo-distributed resources*

Contents

List of Figures

List of Tables

List of Acronyms

sidHadoop	Secure Inter-Domain Hadoop
CD	Cross Domain
GFS	Google File System
HDFS	Hadoop Distributed File Storage
MR	Map Reduce
C2A	Client Authentication-Authorization
R2A	Remote Authentication-Authorization
UM	User Manager
CM	Certificate Manager
LCO	Local Cluster Operations
RCO	Remote Cluster Operations
SSL	Secure Socket Layer
TDE	Transparent Data Encryption

Chapter 1

Introduction and Background

1. Introduction

The term big data is used to describe huge volumes of structured or unstructured data and the technologies and means to process this data [1]. The advancements in technology and because of its easy and low-cost public access, a boom was witnessed in the amount of data generated all over the world. Increase in the data generation desperately required platforms to process these huge amounts of data. In 2003, Google published a paper introducing Google File System (GFS), a fast, efficient and reliable storage platform based on a distributed architecture which offered the use of cheap commodity hardware-based storage clusters [2]. In 2004 Google published another paper introducing MapReduce, a programming model for parallel processing of huge amounts of data in a distributed environment [3]. Development community was set to develop technologies which could cater to the needs of processing big data.

In 2006, Hadoop was created as an open source project. It introduced the implementations of GFS and MapReduce to provide fast, efficient, fault-tolerant storage and processing platform [4], [5]. Hadoop supported the use of commodity hardware in a distributed and cluster oriented environment. Hadoop had two main pillars, Hadoop Distributed File System HDFS [6] based on Google File System concept and MapReduce. It offered very fast processing of large amounts of data due to its distributed storage platform and parallel computing architecture [7]. Initially, Hadoop was introduced with very little or no security mechanisms for securing the data which was stored in it. This was because the data that was supposed to be used for processing was only public and non-sensitive. At the time, there were no existing security technology or mechanism suited for the complex parallel architecture of Hadoop. Kerberos here came to the rescue which required few design modifications for integration in Hadoop [7]. Kerberos was the first add-on for Hadoop in the direction of security [8]. Kerberos was not a complete solution for making Hadoop ready for a production environment but it provided a foundation of security for Hadoop [9]. Apache Ranger [10], Apache Sentry [11], and Apache Knox [12] are only a few names in the list of security solutions for Hadoop that emerged in the open source community with the collaborations of tech giants like Yahoo, Hortonworks, and Cloudera [13]. Hadoop received security add-ons that provided a range of options of security levels depending upon the business requirement. These security add-ons for Hadoop were both open-source as well as enterprise solutions. Hadoop not only received upgrades in the security domain, it was also advancing in the domain of analytics. So Hadoop was ready for a production environment in the form of various mature open-source and enterprise solutions. Hadoop is being used in production by many tech giants in the industry like Amazon, Alibaba, Cloudera, Hortonworks and Yahoo to name a few. [14]

Hadoop was designed to be a private in-house environment based solution but was capable of handling thousands of nodes. For example, Yahoo deployment of Hadoop consists of over 42,000 nodes spread over hundreds of clusters and storing more than 200 petabytes of data [15]. The advancements in other domains of technology, like cloud computing, pushed towards the deployment of Hadoop on a cloud. Cloud deployment is a very feasible solution which can provide comfortably elastic storage and processing scalability to Hadoop cluster size [16]. There are many cloud solutions in the market which are offering ready-made Hadoop deployments on their platform. AWS, Google Cloud etc are among the cloud vendors that provide paid and/or free cloud resources for Hadoop [1]. But cloud technology is still not reliable to store sensitive data due to many security threats [17]. Some researchers may not feel comfortable in keeping their data in a shared storage space with other users worldwide while others may have a large amount of data and computation that would be financially too expensive to move into the cloud [18]. The community of information security is working on reducing the threats in the cloud environment to the bare minimum. Even though cloud computing provides very easy and elastic scalability of computation and storage resources but it is sometimes costly considering Small and Medium level organizations. In-house deployment of Hadoop on commodity hardware suits well for SMEs where low-cost hardware can be used. Hadoop can be deployed in a single cluster or multi-cluster setup, so one whole Hadoop instance is confined within these clusters. Horizontal scalability of resources is possible but sometimes becomes very hectic in updating configuration of the cluster to add the new node/s into the cluster.

Researchers have provided some solutions to access geographically distant resources for Hadoop computation and storage utilizing different techniques and providing frameworks. HOG; Hadoop on Grid provides a solution to access and utilize resources that are distributed over the Grid [19]. But it requires specialized hardware. PigOut; is a system that provides federated data processing over multiple clusters [20]. This comes somewhat close to Inter-Domain Communication between multiple clusters with the use of Pig query language for Hadoop. A single Pig query can be used to access multiple clusters. G-Hadoop; is a solution which provides utilization of resources spread across multiple data centers [21]. It requires a centralized server and specialized broadband connection to efficiently perform its tasks.

Some of these solutions negate the basic feature provided by Hadoop which is the use of commodity hardware for its deployment. Different solutions were studied which provide newly designed frameworks and fulfill the requirements of their own scenarios, but neither of these solutions caters the need of making multiple Hadoop instances, that are geo-distributed, communicate with each other. An organization with multiple Hadoop deployments that are geographically distributed cannot utilize/control/manage all these deployments from one single location. Similarly, educational institutions having their Hadoop implementations for research and development purpose and collaboration agreement with other institutions will not be able to to make use of data and/or resources of Hadoop from other universities. If a job needs to be executed on a distant cluster, a human resource may be required to deliver the job to be executed to the other cluster. Hadoop instances in different domains cannot access the data or resources of each other because Hadoop does not provide support for it.

Given the current situation, there is much that needs to be done in the domain of big data. There is a dire need to overcome the constraints which Hadoop puts on its users and introduce systems and techniques to contribute to the domain of big data. A lot of work is done and is being done in the field of big data. The details of Hadoop, inter-domain Hadoop and security requirements for securing inter-domain Hadoop are discussed later.

1.2 Background (Extended Intro)

Hadoop has become an attractive platform for large-scale data analytics. Its popularity is high because of its many features like parallel processing, fault tolerance, easy scalability, open-source nature, support community, new developments and regular updates [7]. It distributes data over a set of available nodes in a single or multi-cluster environment. Hadoop Distributed File System HDFS handles the storage operations in Hadoop. The processing is handled by the famous divide-and-conquer rule which, in this case, is defined as MapReduce (MR) algorithm. HDFS and MR are two key functional components of Hadoop. The diagram below highlights the architecture.

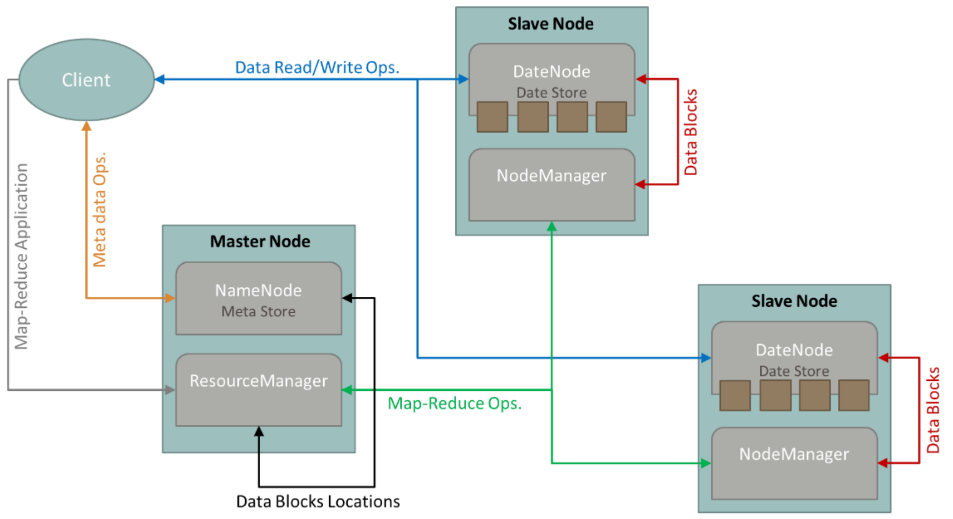


Figure 1: Hadoop Architecture

Hadoop Distributed File System (HDFS) a scalable file system that distributes and stores data across all machines in a Hadoop cluster. The HDFS is a master and slaver framework which contains multiple slaves also called datanodes and a single master also called namenode. HDFS stores data on the compute nodes, providing very high aggregate bandwidth across the cluster.

NameNode runs on a "master node" that tracks and directs the storage of the cluster. It manages the file system namespace, regulates access to files by clients, executes the operations on file system namespace and maps data blocks to data nodes. The name node makes all decisions concerning block replication as well.

DataNode runs on "slave nodes," which make up the majority of the machines within a cluster. The NameNode instructs data files to be split into blocks, each of which is replicated three times and stored on machines across the cluster. These replicas ensure the entire system won't go down if one server fails or is taken offline—known as "fault tolerance."

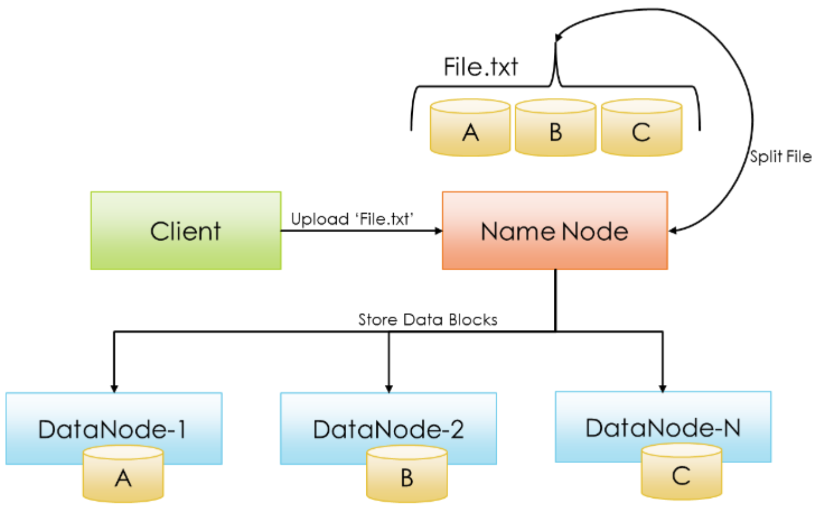


Figure 2: HDFS Architecture

MapReduce was originally created by Google, its strength lies in the ability to divide a single large data processing job into smaller tasks. All MapReduce jobs are written in Java, but other languages can be used via the Hadoop Streaming API, which is a utility that comes with Hadoop. A client writes a Map/Reduce Job and gives it to the cluster while specifying the input and output files. This Job is then split into smaller tasks as per the requirements set in the job. This is the Map step during which data sets are processed and only the required data is extracted, processed into the form of key-value pairs and then forwarded to the Reduce Phase as results. The reduce phase then combines the results, obtained from the map phase, together and the results are written to an output file on the HDFS. The client can then obtain the results of the Map/Reduce from the output file initially specified.

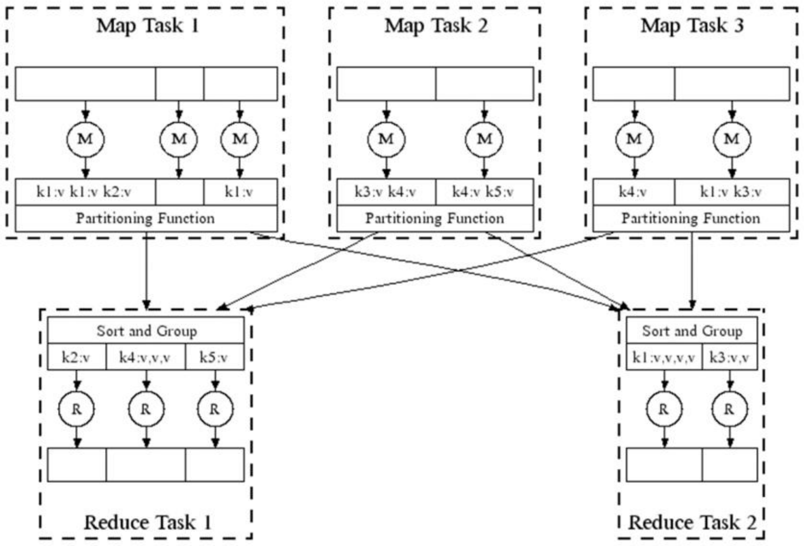


Figure 3: Map Reduce processing model

Hadoop is considered one of the very first few outstanding technologies in the field of big data. Its maturity into security oriented solution made it ready for production environment. In Hadoop, the data, which is required for processing is spread over a vast cluster of nodes, with a very basic 'SSH' level security [22]. If the sensitive data of the organization is needed for processing, then putting it in the cluster will make it accessible to all and thus leave the data vulnerable. Hadoop did not focus enough on security at the start of the project [23]. Its main goal was to efficiently process the huge amounts of data so the security aspects were traded-off. Hadoop's security has continually come under scrutiny, especially given its concurrent processing architecture [24]. To cater to the authentication vulnerabilities, Hadoop introduced central authentication through Kerberos [25]. Through Kerberos, Hadoop achieved authentication using tokens. Different components of Hadoop were able to authenticate each other using modified tokens of Kerberos. The basic level of security was covered by Kerberos and it was later incorporated into core Hadoop distribution.

Authorization was handled later by different open source solutions like Apache Sentry [11] and Apache Ranger [10]. Apache Sentry provides granular RBAC for Hadoop components whereas Apache Ranger provides fine-grained

RBAC to Hadoop and its components. They both provide a plugin for tools associated with Hadoop like hive. The plugin is attached with the component and the communication to that component is allowed or denied through the plugin. Both these solutions provide support for Kerberos.

Ranger provides additional features like auditing and management console, whereas sentry requires third-party solutions for these features. Later releases of Hadoop also included a new feature called Transparent Data Encryption TDE in HDFS [26]. TDE is a technology that is used to encrypt data at rest. It performs encryption and decryption in a transparent fashion.

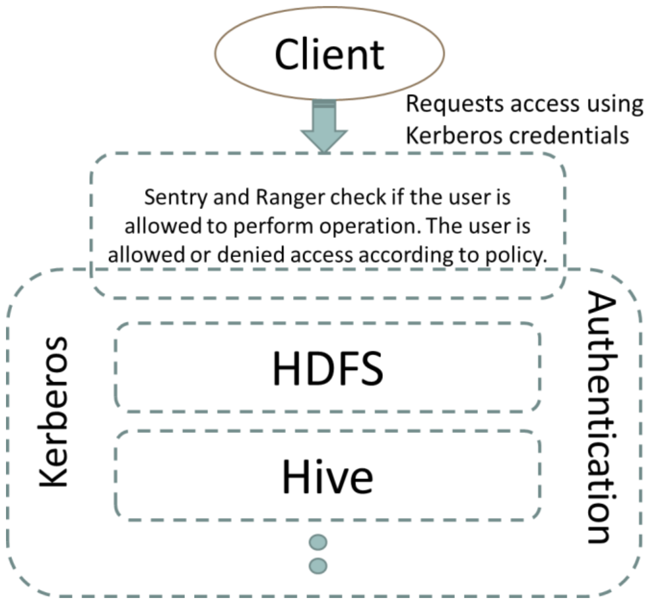

Client
Requests access using Kerberos credentials
Sentry and Ranger check if the user is allowed to perform operation. The user is allowed or denied access according to policy.
Kerberos
Authentication
HDFS
Hive

Figure 4: Hadoop with Security Add-ons

The newly included features greatly supported the basic pillars of security like Authentication, Authorization, Confidentiality and Integrity. This made Hadoop ready for a production environment, with multiple security options suitable for different organizational requirements. While there were security advancements for Hadoop, the technology development for enhancing the capabilities of Hadoop did not stop. Many organizations started working on their own solutions to handle different types of data in Hadoop [27]. Facebook started the project Pig [28], which provides a solution for reading data from Hadoop using structured data approach.

Pig is a platform for manipulating data stored in HDFS. It consists of a compiler for MapReduce programs and a high-level language called Pig Latin [29]. It provides a way to perform data extractions, transformations and loading, and basic analysis without having to write MapReduce programs. Furthermore, other projects like Hive [30] and Ambari [31] are just a few names in a long list of projects that were made for Hadoop. Hive is a data warehousing and SQL-like query language that presents data in the form of tables. Hive programming is similar to database programming. Ambari is a web interface for managing, configuring and testing Hadoop services and components. It provides an easy interface as opposed to the direct shell used for performing most of the tasks in Hadoop.

Similarly, solutions started emerging for using Hadoop in a geo-distributed environment. Researchers felt this need because the resources were spread over geographically different locations and instead of moving all the resources into one place, it was thought to explore Hadoop in a geo-separated design. One of these solutions is HOG; Hadoop on Grid. This solution discusses the use of specialized Open Science Grid OSG [32] clusters with Hadoop. In HOG, the Data nodes are distributed over various data centers that are part of the OSG whereas the Name node is established on a specialized stable node. This node responsible for maintaining the connection between the various datacenters where the slave nodes are situated. Thus a single Hadoop instance is created that has its nodes spread over different locations.

A similar solution to discuss here is G-Hadoop [21], a MapReduce framework that aims to enable large-scale distributed computing across multiple clusters. It uses G-Farm [33] for file storage system as opposed to the original HDFS. G-Farm provides native support for wide-area operations. The data nodes are spread over multiple locations with a single stable master node that's responsible for managing the slaves and distributing data and jobs to these slaves. G-Hadoop makes a lot of design changes to the core Hadoop design to achieve Hadoop functionality over the grid. These solutions show that the need to use Hadoop over a geo-distributed environment was building up. During the same period, providing Hadoop solutions over cloud was another hot domain being explored [34]. Many researchers were working on achieving the features that Hadoop offered onto the cloud infrastructure. Amazon now offers elastic on-demand Hadoop deployment as do other organizations.

The different solutions discussed above over the geographically distributed environment for Hadoop try to make a single Hadoop deployment. They also discuss the use of specialized hardware for processing or specialized network bandwidth requirements which makes it difficult for the implementation of these solutions. The solutions in the discussion also lack a strong security stance required for Hadoop, especially when resources utilized by these solutions are spread over WAN.

In this research we are proposing Hadoop deployment, where two or more independent Hadoop clusters which are situated over different administrative domains can securely communicate with each other over WAN without actually sharing huge amounts of data. The solution will help in the collaboration of Hadoop resources. Resources will not go to waste and temporary scale will be facilitated by using resources from other Hadoop instances. The security part will ensure the reliability of the collaboration between Hadoop instances. With this solution, academic institutes will be able to share their public data and the best part will be that complete data will not be needed to transmit but only the required data after processing will be transmitted that lessens the amount of data to traverse over WAN. Multiple organizations will be able to share their resources and a single organization with multiple offices will be able to fully utilize its resources.

Chapter 2

Literature Review

2 Related Work

When Hadoop was introduced, the security implications were not in the scope of its operations. The data in Hadoop was not supposed to be sensitive, the environment of Hadoop cluster was supposed to very private and only simple 'ssh' security to access different physical slave nodes was sufficient. As many companies started implementing Hadoop in their production environment, the sensitive data of the companies was required to be put in the Hadoop clusters which made it vulnerable. Hadoop is in use at many of the world's largest online media companies like Yahoo, LinkedIn, Twitter, Fox Interactive Media and Facebook. A huge list of all the companies who are using Hadoop is provided on the Hadoop website [14]. For the sake of their sensitive data, many companies started customizing Hadoop according to their own requirements of security. Many companies started supporting the open-source community to develop open-source solutions for securing Hadoop.

The latest release of Hadoop has included new features called Transparent Data Encryption (TDE) in HDFS and Wire Encryption. The newly included features have greatly supported confidentiality and integrity pillars of security [26].

Researchers have put significant efforts to the easy submission and scheduling of MapReduce jobs in clusters, grids and clouds. Researchers all around are working on the storage and processing of data in Hadoop. They are trying to overcome the constraints that Hadoop puts on its users. Following are the few papers that provide a solution in the domain of geo-distributed Hadoop resources which were studied to learn about this domain.

2.1 A Hierarchical Framework for Cross-Domain MapReduce Execution [18]

In this paper, the authors stated that the MapReduce programming model [3] provides an easy way to execute pleasantly parallel applications. Many life science applications fit this data-intensive programming model and are benefited by the scalability it delivered. One such application is AutoDock [35], which consists of a set of tools for predicting the bound conformations of flexible ligands to macromolecular targets. However, researchers required sufficient processing and storage resources to fully enjoy the features of MapReduce. For example, a typical AutoDock based virtual screening experiment usually has a large amount of docking processes from different ligands and it takes a lot time to execute them on a single MapReduce cluster. Although enterprise clouds can provide virtually large number of computation and storage resources on-demand, but due to high-price, unreliability and possibly other concerns, many researchers perform their tests on a number of small clusters with a small number of nodes which could not fully utilize the benefits of MapReduce.

In this paper, the authors are working on Quarry FutureGrid [36] and TeraGrid [37] clusters but these cluster only provided a limited number of nodes which could be used at one time. Also, these clusters are separated by different administrative domains and due to the current Hadoop structure, a single more powerful cluster could not be created by making these clusters work together. One approach could be to combine the underlying physical clusters as a single virtual cluster by adding a special infrastructure layer. MapReduce jobs could be executed on this virtual cluster. But using specialized hardware is not everyone’s cup of tea. The framework discussed in this paper gathers computation resources from multiple clusters and execute MapReduce jobs. This Hierarchical MapReduce Framework consists of two layers, the upper layer is the Global Controller which contain job scheduler, data transferor and user-supplied global reducer, and the bottom layer consists of multiple local clusters for executing distributed

MapReduce jobs. Each local cluster has an MR master node with a workload reporter and a job manager.

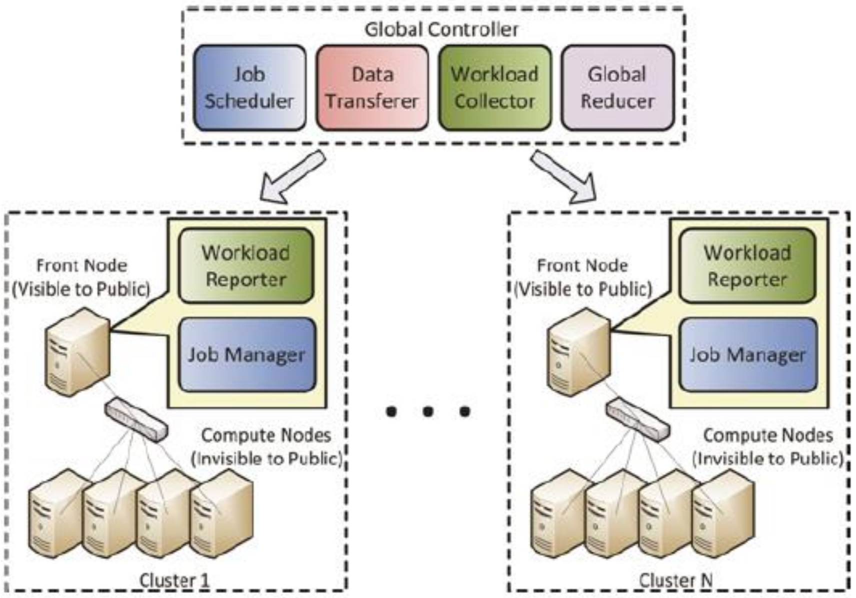


Figure 5: A Hierarchical Framework for Cross-Domain MapReduce Execution

The global controller, works as a centralized server that accepts user-provided MR jobs and divides them for execution on different local clusters domains. It divides the jobs into sub-jobs based on the computation resources of the local clusters. After the jobs are completed, the global controller collects the results from all local clusters and performs a final reduction to consolidate the results. In this framework, a user has to write two reducers, 1st is the conventional reducer and the 2nd is the global reducer for final reduction of results. In this framework, the global controller divides the data blocks and sends them to multiple local clusters. It balances the workload by sending tasks in accordance to the capabilities of each cluster and of each node. The local results are returned back to the global controller for global reduction. Their experimental evaluation using AutoDock over MapReduce shows that their load-balancing algorithm performs efficient distribution of workload across multiple clusters. It also reduces the total execution time span of the entire MapReduce execution.

2.1.2 Limitations of "A Hierarchical Framework for Cross-Domain MapReduce Execution"

In this paper, they have presented a hierarchical MapReduce framework that can gather computation resources from different clusters and run MapReduce jobs across them. It can help with the life science applications. It is suitable for life science applications which are both compute intensive and data intensive. They consume a large number of CPU cycles while processing massive data sets which are either in a large group of small files or could be split naturally. But their framework doesn't include the remote connection to Hadoop clusters.

This framework mainly focuses on multiple clusters which are within a single administrative domain and use 'ssh' and 'scp' to transfer data across different clusters. This could not work in a globally distributed clusters and the security implications will be huge. They have also discussed the lack of security mechanism in their solution for the protection of data and jobs. Their framework is suitable for submission and scheduling of massive parallel jobs to reduce the time and efforts.

2.2 HOG: Distributed Hadoop MapReduce on the Grid [19]

MapReduce programming model provides a powerful data processing platform for enterprise and academic applications. In this paper, a novel Hadoop MapReduce framework runs on the Open Science Grid [26] which is spread across multiple locations across the United States. The solution is titled Hadoop on the Grid (HOG). It is different from previous MapReduce platforms that run on dedicated environments like clusters or clouds. HOG provides an open source, scalable, and dynamic MapReduce environment on the opportunistic resources of the grid. In HOG, fault tolerance of Hadoop is improved for data analysis over WAN by converting resources in different data centers to virtual racks and by developing multi-institution failure domains. In HOG, A single Hadoop instance is created that comprises of physical nodes that are geographically distributed over the grid using OSG. In HOG, the Namenode instance that handles HDFS and the job tracker that handles MapReduce are kept on a stable centralized server. So that the master node is available all the time. The slave/worker nodes are distributed among various data centers. These data centers are transformed into virtual racks so that these can be accessed from the centralized master node.

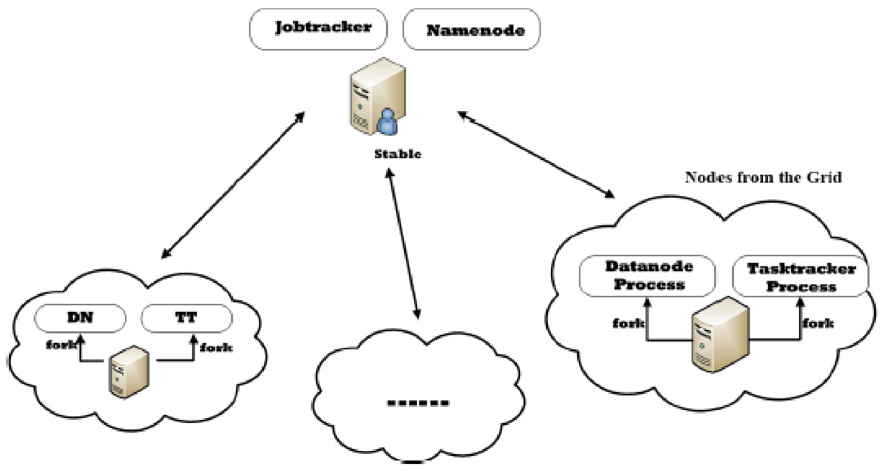


Figure 6: Hadoop on Grid

In HOG, they have provided site awareness added to the rack awareness concept of core Hadoop. It is similar to rack awareness but this implementation has resources spread over multiple sites so if a whole site goes down or a node inside a site goes down, it can be identified by site awareness and Hadoop will start to replicate the data that was inside that site to another active site. The changes to the Hadoop original design are transparent to already deployed Hadoop MapReduce applications. In the evaluation, HOG was extended to 1100 nodes on the grid. Moreover, HOG was evaluated over a simulated Facebook Hadoop MapReduce workload. It concludes that rapid scalability of HOG can provide comparative performance to a local Hadoop cluster.

2.2.2 Limitations of HOG: Distributed Hadoop MapReduce on the Grid

In this paper, Hadoop infrastructure was created on the Open Science Grid. The contribution of this research includes the detection and resolving of zombie datanode problem, site awareness, and a data availability solution for HOG. The limitation of this research includes the security issues, HOG utilized HTTP to perform RPC calls between nodes whereas, in the OSG, users have to use a trusted certificate to access resources.

In this paper, the target is to consolidate geographically distributed resources into consolidated virtual clusters, they have made very few changes to the Hadoop architecture which is a good thing but their framework requires specialized hardware. Transforming data centers into virtual racks can be costly and will require dedicated fast broadband for efficient communication between Hadoop clusters. Also, there is the use of centralized stable server which requires specialized hardware to maintain stability along with high-speed internet. It achieves the goal by transforming independent clusters into one single Hadoop instance as it was discussed in the previous paper an alternate solution to the framework discussed.

2.3 Towards a Cross-Domain MapReduce Framework (2013) [38]

Cross Domain Hadoop (CD-Hadoop) focuses on Multi-Level Secure (MLS) environment for Hadoop. Its idea is to run Hadoop with multiple layers/domains of security. CD-Hadoop prototype is implemented on Security Enhanced Linux (SELinux) [39] which provides the configuration to enforce MLS aware policy for different sensitivity levels based on security labels of subjects and objects. In CD-Hadoop cluster, there is one physical namenode and multiple physical datanodes. Namenodes, for different sensitivity levels, reside on the same physical node.

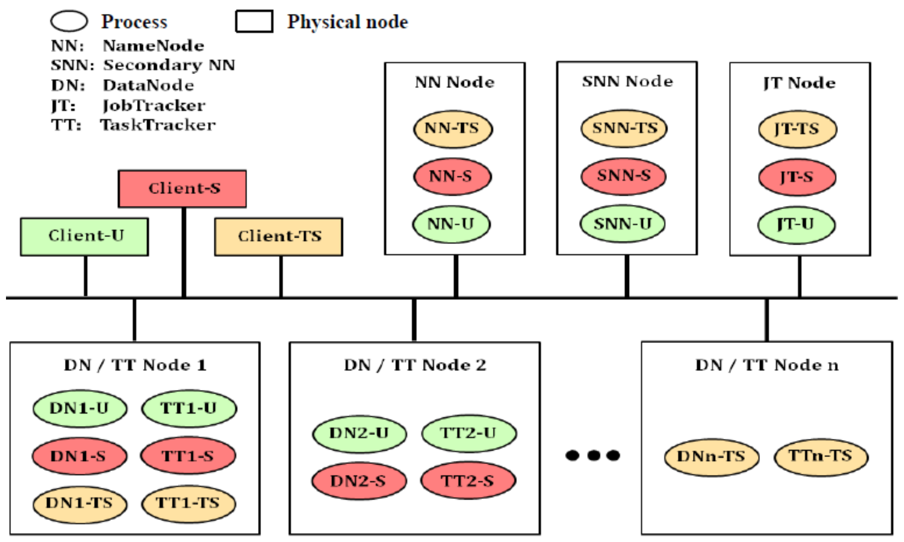


Figure 7: Towards A Cross-Domain MapReduce Framework

Different security levels each have a separate surrogate Hadoop instance with its own namenode and datanode. These surrogate namenode and datanodes reside in the same physical namenode and datanodes but with different authorization and sensitivity levels.

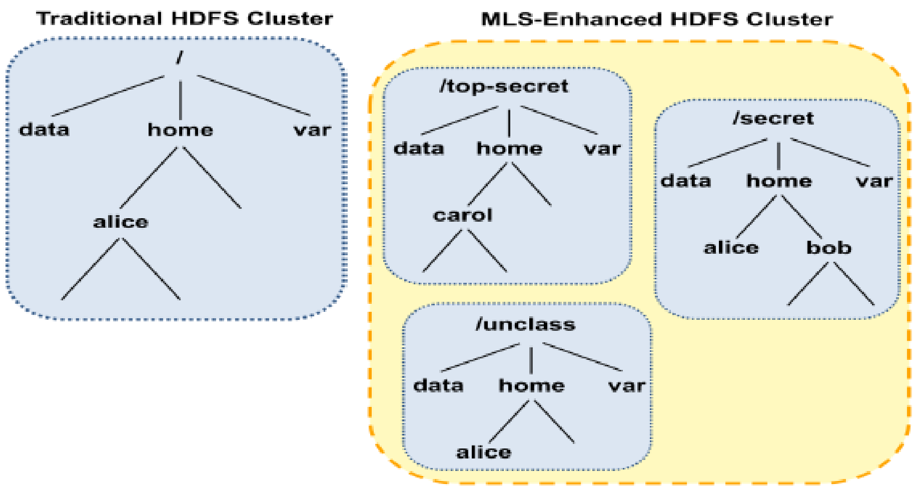


Figure 8: Sensitivity Levels

This paper describes different security levels as independent security domains residing inside the same hardware. This provides an abstraction layer for data authorization. Lower level security users cannot access higher security level data whereas higher security level users can access the lower security level data. This is called read-down approach for cross-domain data read operations. But write operations are only allowed in the same sensitivity domain. This solution does not offer independent Hadoop instances to communicate with each other at the same security level, but instead, it dynamically divides a single cluster into multiple clusters according to the security labels required.

2.3.2 Limitations of "Towards A Cross-Domain MapReduce Framework (2013)"

This paper provides a good solution for high-security organizations that require different authorization levels for its different users. It introduces MLS aware environment [40], multiple instances of Hadoop can run at different sensitivity levels while their access to Hadoop resources is constrained by underlying trusted OS. It provides a highly secure MapReduce platform but it also shows performance degradation. This performance degradation can be acceptable in cases where security is the first priority than efficiency. The solution discussed in this paper dynamically creates multiple Hadoop instances in separate security domains. The physical nodes are responsible for handling the surrogate nodes and communication across different domains is only allowed though the read-down approach.

2.4 G-Hadoop: MapReduce across Distributed Data Centers for Data-Intensive Computing [21]

This paper states that there is significant rise in the computational requirements for comprehensive data-intensive analysis of scientific data. In High Energy Physics (HEP) for example, the Large Hadron Collider (LHC) produced 13 petabytes of data in 2010. This large volume of data is processed on more than 140 computing centers spread across 34 countries. The MapReduce programming model provides highly efficient processing support for large-scale data-intensive computing applications. But, current MapReduce implementations are can only be deployed on single cluster environments. It could not provide support for large-scale distributed data processing across multiple data centers. It uses workflow systems for distributed data processing across geo-distributed resources. The workflow paradigm has some limitations for distributed data processing, such as reliability and efficiency.

In this paper, the design and implementation of G-Hadoop, a MapReduce framework is presented that aims to enable large-scale distributed computing across multiple data centers. G-Hadoop is slightly similar to Hadoop on Grid, it also requires a centralized master node that controls/utilizes the slave nodes spread across different geographical locations. G-Hadoop uses G-Farm [33] file system rather than the HDFS used by core Hadoop. G-Farm provides a global virtual file system across multiple administrative domains and is optimized for wide-area operations to provide site awareness.

The master node in G-Hadoop comprises of two main components, the Metadata server for keeping a record of data in the G-Farm file system and the Job Tracker which is responsible for splitting the jobs into smaller tasks, distributing ad scheduling these tasks among the participating clusters of G-Hadoop. This job tracker is a modified version of the core Hadoop job tracker which is responsible for similar tasks but in the local cluster environment. The slave node in G-Hadoop consists of TaskTracker and I/O Server.

Task Tracker which is similar to core Hadoop task tracker but modified to handle G-Hadoop architecture. Its job is to accept tasks from Job Tracker and report back the status. It also keeps a check on the workers and submits the results back to Job Tracker. I/O Server manages the data stored in the storage of G-Hadoop. It links with the metadata server and is configured to store the data in the high-performance file system of its cluster. In this paper, the proposed system does not generate the metadata by splitting the files into blocks of data like core Hadoop hdfs, but instead, it keeps a complete file as a block, that is why less metadata is generated.

2.4.2 Limitations of "G-Hadoop: MapReduce across Distributed Data Centers for Data-Intensive Computing"

The presented framework in this paper supports distributed data-intensive computation among multiple administrative domains using existing unmodified MapReduce applications. The proposed system requires specialized hardware/software for storage that is why G-Farm file system is used. The files are not distributed into smaller blocks which allows less parallel environment as compared to core Hadoop architecture. It requires high-performance network solutions to maintain efficient communication between the clusters. This paper proposes a solution in which a master node is centralized and slave nodes are distributed which are connected virtually. The goal of this research is to advance the MapReduce framework for large-scale distributed computing across multiple data centers with multiple clusters and does not focus on remote connectivity to Hadoop clusters. This paper also discusses the security shortcomings of the proposed solution, but they have worked on covering these security issues in another paper which is discussed below.

2.5 A security framework in G-Hadoop for big data computing across distributed Cloud data centers [16]

The G-Hadoop system discussed above re-uses the Hadoop mechanism for authentication and job submission which is sufficient for a single administrative domain setup but not for the grid. The framework proposed in this paper has the following properties; Single Sign-On, Privacy of user information, Access Control, Scalability, Immutability, and Protection against attacks. The proposed security framework employs PKI and uses a CA server. The Gfarm file system used by G-Hadoop already applies the Grid Security Infrastructure GSI [41] so it includes CA. The framework utilizes this CA to design a suitable symmetric cryptography to secure G-Hadoop. As G-Hadoop has physical resources spread across multiple data centers so communication between these resources is performed over WAN. The proposed security framework extends G-Hadoop in the phase of submitting jobs from a user and the phase of job termination, where additional steps are performed to authenticate the communication parties and to establish a secure connection before executing jobs/tasks.

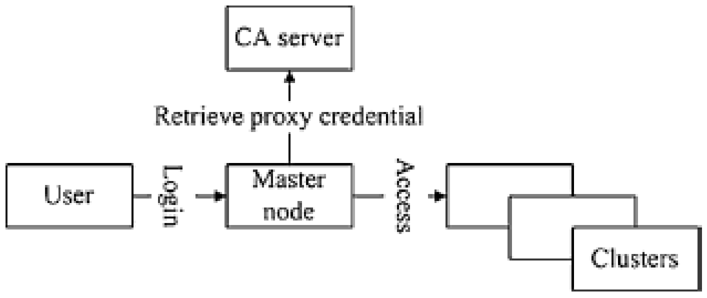


Figure 9: Security Framework for G-Hadoop

The whole workflow consists of the following main phases: user authentication, proxy credential assignment, preparing authentication information on the masternode, authentication of the masternode and slavenodes, as well as job execution, termination, and disconnection.

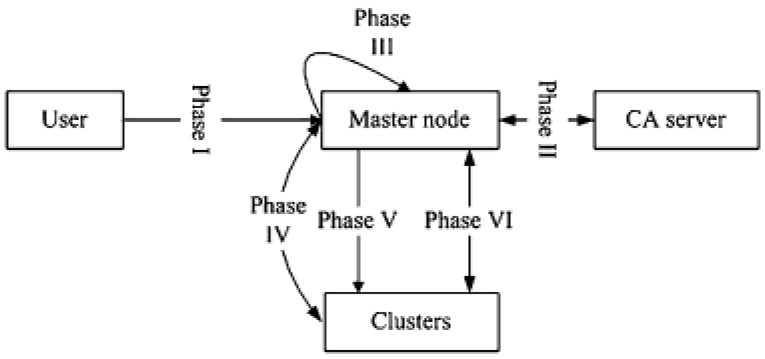


Figure 10: Security Architecture

The structure of G-Hadoop for user does not change significantly with security implementation, the user has to log-in to G-Hadoop on the master node and it takes care of the rest of the steps. To provide proxy credentials, the master node communicates with the CA Server over SSL. It authenticates itself and requests the CA Server to provide session keys for the datanodes to authenticate.

This must be noted that the datanodes are spread across different data centers that is why session security is necessary for the communication between namenode and datanodes. Using the session keys, the namenode authenticates with datanodes and issues the job that is required to run. When executing the jobs, the namenode assigns the user session information with each job for uniqueness and security. When jobs are completed, the datanode returns the result back to the namenode using the same session information for that job and the user gets to see the results it intended.

2.5.2 Limitations in “A security framework in G-Hadoop for big data computing across distributed Cloud data centers (2014)”

This paper lays out a comprehensive security mechanism for G-Hadoop. It takes care of user authentication with g-Hadoop, namenode and datanodes mutual authentication and job execution security. The proposed solution uses SSL security and PKI using CA provided by G-Farm. In the proposed solution, SSL security is also being targeted for accessing the remote namenode as discussed in this paper. G-Hadoop offers the use of geo-distributed physical resources as a single Hadoop cluster, but in core Hadoop, a namenode requires to access the datanode directly to execute the jobs. So the security mechanism required for this solution offered the use of session keys for the period of job execution in which the namenode makes use of these session keys to securely access the datanodes. This solution is sufficient in the scenario of G-Hadoop, but it also inherits the limitations of G-Hadoop solution as they are discussed in the limitations of G-Hadoop section.

2.6 PigOut: Making Multiple Hadoop Clusters Work Together [20]

The presented system in this paper enables federated data processing over multiple Hadoop clusters which is titled PigOut. PigOut provides the interface to write a single script in a high-level language to execute jobs on multiple Hadoop clusters. Manual labor of writing multiple scripts for different clusters and to coordinate the execution for different clusters is removed. PigOut partitions a single, user-provided script into multiple scripts for different clusters accordingly. Moreover, PigOut generates workflow descriptions to keep a check on the execution across clusters. In doing so, PigOut uses existing solutions that are built around Hadoop, reducing extra efforts needed from users or administrators. For example, PigOut uses Pig Latin [23], which is a renowned query language for Hadoop MapReduce. The modification to Pig Latin a merely in the form of extension in Pig Latin with full backward compatibility. PigMix is used for evaluation of the proposed solution, which is the standard benchmark for Pig. It was demonstrated that PigOut's automatically-generated scripts and workflow definitions have comparable performance to manual, hand-tuned ones. They also reported their experience with manually writing multiple scripts for a set of federated clusters, and compared the process with PigOut's automated approach.

2.6.2 Limitations of "PigOut: Making Multiple Hadoop Clusters Work Together"

This paper presented PigOut, a federated data processing system over multiple Hadoop clusters. PigOut takes care of all aspects of automation, which include script and workflow generation, data transfer, and optimization suitable for cross-cluster execution. The proposed solution does not require any extra work from the users or cluster administrators of hadoop because it supports Pig Latin's syntax out of the box. It utilized standard, core Hadoop components without any modification.

PigOut uses 'scp' for the transfer of pig scripts and data to remote clusters which is a good and efficient solution for a local deployment but over WAN this could not suffice. Furthermore, there is no mention of security mechanisms e.g., authentication between clusters, the integrity of source cluster and the job received from that cluster. There is a need for a security mechanism for the implementation of PigOut because the mechanism used for local implementation of Hadoop cannot cope for the distributed structure of multiple Hadoop instances.

While PigOut is a feasible solution to the multiple data sets over multiple clusters problem. Its implementation is only confined to using Pig queries over Hadoop clusters. If a cluster does not use Apache Pig, then this implementation cannot provide a link to that cluster and manual execution of job will be required for that cluster. It does not facilitate the problem of accessing an organizational cluster from outside the organization to access open data. Nor does it solve the problem of using external clusters for data analysis based on data-sets stored on them. What PigOut does is strictly limited to an organizational setup of multiple Hadoop clusters.

2.7 Key Features from Related Work:

This solution may try to achieve, enhance or subtract these features. Table 1, highlights the key features of the above-discussed papers. In this table, the points discussed are not all those which can be considered as features for the scenario of this research. Some of the points, mentioned above, are features as per their respective research papers but they may become either drawbacks or unnecessary in terms of the research scope of this research thesis. For example, some of the above-mentioned papers require specialized hardware and network components.

Similarly, the proposed solutions discussed in the related work section create a single instance of Hadoop with stable centralized Namenode handling all the geo-distributed datanodes. This increases the traffic flow because the namenode has to keep track of all the distributed resources in the form of heartbeat or health status. Site awareness is also required when a central server is handling remote resources situated at different geographical locations or sites

Table 1: Key Features from Related Work

Features	**CD-Map Reduce Execution** [18]	**HOG** [19]	**CD-Map Reduce Frame-work** [38]	**G-Hadoop** [21]	**Security Framework in G-Hadoop** [16]	**PigOut** [20]
Minimal Changes in core Hadoop			✓			✓
Geo-Distributed Hadoop resources	✓	✓		✓		
Minimal Data Traversal			✓			✓
Stable and Centralized Namenode	✓	✓		✓		
Site Awareness		✓		✓		
Use of Specialized Hardware	✓	✓		✓		
Use of Specialized Network		✓		✓		
Multi-Layer Security MLS			✓			
Transmission security					✓	
Single Sign-On					✓	
Protection from attacks			✓		✓	

Minimal changes in core Hadoop architecture is a very important feature because it provides support for backward compatibility. But some of the above-discussed solutions are offering a lot of design changes. For example, G-Hadoop uses the g-farm file system in place of hdfs. To implement g-Hadoop, legacy Hadoop systems would require new configuration and software suite so the backward compatibility is not supported.

The concept of Geo-distributed Hadoop resources suggest the use of resources distributed over different administrative and geographical domains. The papers for g-Hadoop, Hadoop on grid, cross-domain MR execution provide solutions to connect these resources into a single Hadoop cluster. These resources are turned into a single Hadoop cluster by deploying a stable namenode. The datanodes that are distributed over different locations are either virtualized or redeployed with new software components depending on the solution in question. Minimal data traversal is achieved by converting the resources that contain data into slave nodes/data nodes for geo-distributed Hadoop. By doing this, the huge amount of data does not need to be traversed but instead, simple software components providing an upgrade for making geo-distributed Hadoop slave nodes are traversed. The results of the computations on the existing data may be required to traverse over WAN but by this, the data traversal is lessened by a considerable amount.

Only a few papers in the above section discuss the requirements of security in their solution. For example, a security framework for g-Hadoop paper builds a comprehensive security stance for g-Hadoop providing transaction security, single sign-on and other features. The paper 'Towards cross-domain MapReduce framework' proposes Hadoop with Multi-layer security domains. The MLS aware solution creates multiple sensitivity layers and describes them as different security domains encapsulated within the same hardware. These sensitivity layers though add extra security controls to the solution but would also increase the performance overhead.

Chapter 3

Research Methodology

3.1 Research Approach

Considering the complex architecture of Hadoop, there are a lot of interconnected factors to consider. Although increased security is a critical need nowadays, the non-technical factors, that is, the human beings, have to be given equal importance. This is because human beings are the ones who are going to be the actual users of any deployment of Hadoop. Better security practices are really important especially when applying security features to a solution that is more focused towards the efficiency of work, but too much security effects the efficiency if that is the main target. Therefore, the tradeoff between security and usability has to be developed in such a way that the technology does not lose its core features.

The solution, being dealt with, has its roots in the high-speed performance of compute-intensive tasks. So the security complexity has to be kept at such a level that the resources and data are secure as well as it don't lose much of the efficiency of the main algorithm. Thus, behavioral research, for studying the non-technical factors, was also made a part of this research. It was made sure that all factors, related to Hadoop and its security mechanisms are studied, and a secure, yet practical solution is presented.

The research was divided into two phases. In the first phase, the basic concepts of Hadoop and protocols employed were understood along with the idea of secure communication over an insecure channel. The problem of Inter-Domain communication between multiple Hadoop instances was studied. After understanding the concept of inter-domain Hadoop and some solutions presented by the research community which utilize the concept of inter-domain Hadoop in different capacities through the years, sid-Hadoop; Secure Inter-Domain Hadoop was designed and presented. After multiple solutions were studied relating to inter-domain Hadoop, and how these solutions had fewer security concerns, a solution was devised with the idea of providing better security and usability.

The second phase was the testing phase in which the proposed solution is tested against any kind of leakage of data so that this solution can hold against security threats over the public network. Later, these metrics were analyzed to assess the efficiency and effectiveness of the solution. Through this behavioral research, a clear idea was obtained about the practicality of the proposed solution.

3.2 Relation of Design Science and the Presented Research

In this research, work has been done to solve a certain problem that is, providing security for communication between multiple Hadoop instances separated by different administrative domains. Thus, an effort has been made to tackle the problem with the theoretical, as well as, a practical aspect in mind. Such research, which aims to solve problems, can be done better by using "a design science approach [42]".

In design science, the focus is on developing and evaluating the performance of a designed artifact with the target of improving the functional performance of the original artifact. The areas, where design science is most applied, are engineering and computer science. This is because when artifacts, like algorithms and human/computer interfaces, are redesigned or reevaluated, the focus is on solving specific problems. According to [43], the difference between natural science and design science is that the former tries to understand reality while the latter attempts to create it.

The products of design science approach serve human purposes, usually. The evaluation of these products is done to prove their performance, improvement, value and utility [43]. The suggested technical solution, in this research, was tested for its utility, improvement and performance.

In design science, innovation is the main part whereby new ideas, practices and products, are created for serving humans more efficiently [44]. In this research, a new solution was developed which provides security for inter-domain Hadoop communication which will allow the users of this technology to utilize and share storage and performance resources that are available on geographically distant Hadoop clusters. According to Aken, design science aims to solve improvement and construction problems, in order to implement an innovation [45].

The explanation, given by Aken, also corresponds with this thesis. One of the basic objectives of this research was the extension of the theoretical basis as an innovation. The extension, here, is the new security architecture for inter-domain Hadoop. This extension tackles some specific attacks on the inter-domain Hadoop while trying to maintain the efficiency of Hadoop. Another innovation was to bring the attention of the research community towards the development of secure inter-domain Hadoop.

In short, design science research methodology is chosen because it helps in testing both the theoretical and practical features of the designed artifacts. This work is focused on the "improvement problem", as discussed by Aken [45].

3.3 Process of Research

The research process, carried out, will be mapped onto the design science research process in this section. According to [46], design science research process comprises of five sub-processes, which are listed below.

1. Awareness of the problem
2. Suggestion
3. Development
4. Evaluation
5. Conclusion

The same concepts about design science research are put forward in[47]. The five different phases of design science research method have been shown in the form of Figures 3.1 and 3.2 [46]. These figures have helped in understanding this research method properly.

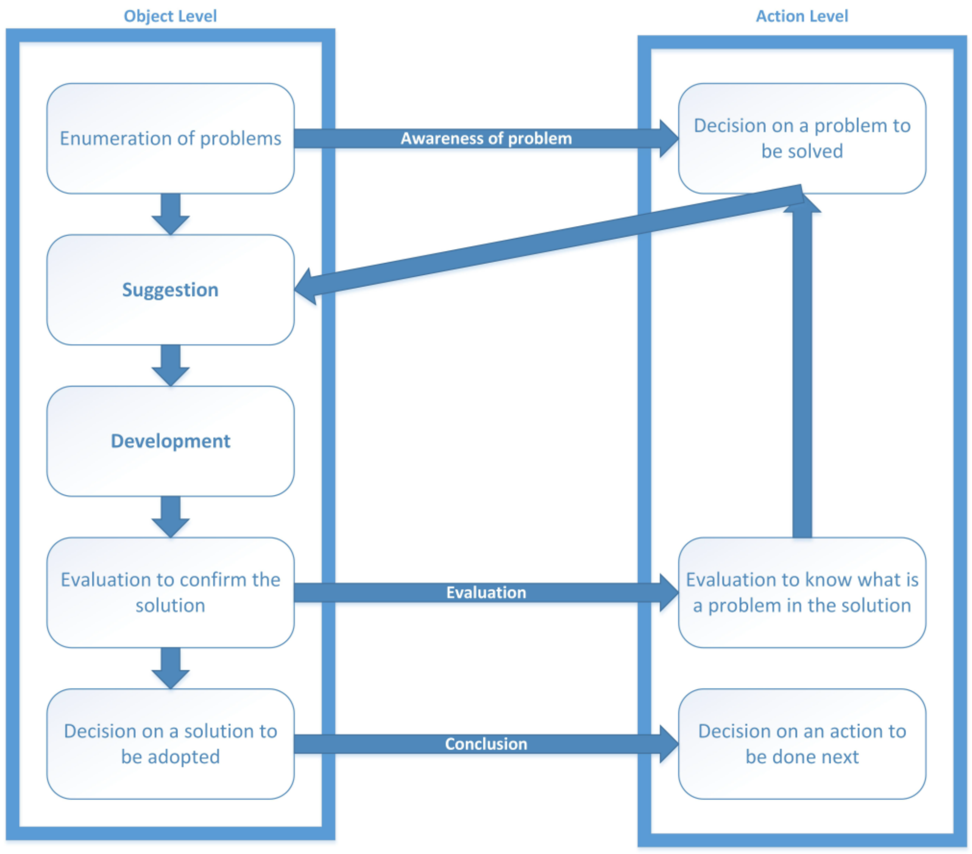

Object Level
Action Level
Enumeration of problems
Awareness of problem
Decision on a problem to be solved
Suggestion
Development
Evaluation to confirm the solution
Evaluation
Evaluation to know what is a problem in the solution
Decision on a solution to be adopted
Conclusion
Decision on an action to be done next

Figure 11: Design Cycle [46]

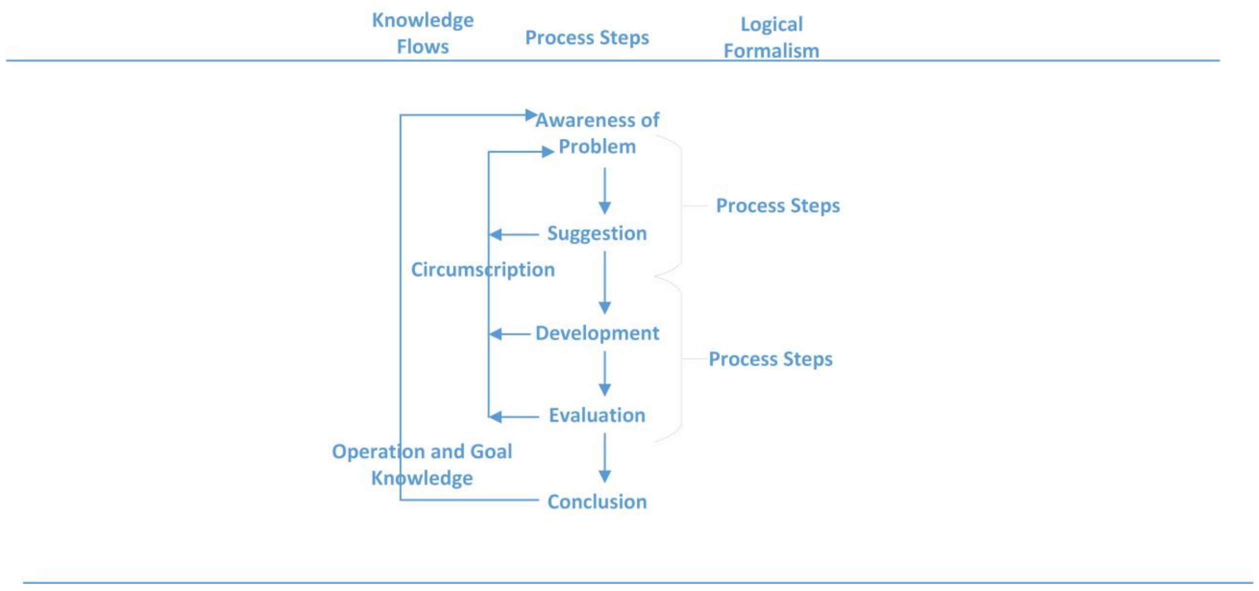


Figure 12: Reasoning in the Design Cycle [47]

3.3.1 Awareness of the Problem

During the first phase of design science research methodology, a researcher makes himself or herself familiar with the problem and its related domain. Also called "improvement research", design science research demands awareness of a problem so a suitable solution can be suggested to [47]. By comparing the object, under consideration, with its specifications, a problem is identified in this phase [46].

Studying and understanding the related domain was the first phase of this research. The basic concepts of Hadoop security architecture of Hadoop were understood [4], [7]. The concept of inter-domain Hadoop was understood and lacking solutions for secure inter-domain problems was realized. The concepts of secure communication over WAN was also discovered. In order to understand the research done on inter-domain Hadoop or its relating solutions were studied [19]–[21], [34], [38]. Through this study, it came to be known that there is no such comprehensive solution exists that can provide secure communication over WAN for two independent Hadoop clusters.

3.3.2 Suggestion

During the first phase, the knowledge base for the domain and the awareness of the problem was built. Based on these, a solution was suggested. In this phase, mainly, the following two steps were taken.

- For solving the problem, the required key concepts were suggested[46].
- A solution, to the current problem, was inferred from the knowledge base of the domain (built during 'Awareness of the problem' phase) by using abduction [47].

The gathered knowledge helped in understanding Hadoop and inter-domain Hadoop thoroughly along with their limitations and security architecture. In addition, the tradeoff between security and usability was understood. Thus, a solution which provides security for communication over WAN between Hadoop clusters is presented. Through this solution, dormant resources of Hadoop clusters can be shared and utilized up to its full potential.

3.3.3 Development

In this phase, the implementation of an artifact is done in the light of the suggestion phase [47]. If anything unsolved comes up, the whole design cycle should be repeated [46]. The suggested sidHadoop; Secure Inter-Domain Hadoop was developed in this phase and prepared for testing.

3.3.4 Evaluation

During the evaluation of the solution to find issues in performance and suggest further improvement, a new iteration of the design cycle is needed if any problem is found [46]. It should be remembered that, in a typical design science approach, the development, evaluation and suggestion phases are performed iteratively [47].

Effectiveness told about how security mechanisms are protecting against certain attacks that the traffic over WAN is prone to. The parameter of efficiency told about how the provided solution effects the efficiency of Hadoop functionality because if the originality of Hadoop is lost by losing its efficiency than the solution may not be worth making. Through this usability study, it was found out that the newly developed system did offer an adequate degree of security. The usability of one scheme was found to be better than the other.

3.3.5 Conclusion

This phase helps in offering a probable solution and/or changing the description of the objects [46] and signals the end of a design research project [47]. The knowledge, obtained as an outcome of this research, was shared and disseminated in this phase.

Particular solution i.e., Secure inter-domain Hadoop, was suggested along with a supporting usability study. This solution was built after proper consideration of the balance between security and usability. The solution was tested against security from multiple attacks possible over WAN. Further research work can be carried out by extending the scope of this solution and providing a solution for securely and efficiently data sharing for Hadoop over WAN.

Chapter 4

Proposed Solution

4. Proposed Solution:

Hadoop is a highly scalable open source framework installed over commodity hardware for distributed storage and processing of very large data sets, generally known as Big Data. Due to the security concerns, Hadoop does not allow remote connectivity which in return demands the need for Hadoop administrators, who manually do the storage and processing operations. Solutions like HOG and G-Hadoop as discussed before are either making a single deployment of Hadoop over geo-distributed physical resources and, sometimes, are making use of specialized hardware. These solutions require a centralized authority to manage the distributed resources. The design changes made by these solutions are very high and some solutions almost strip away the core concept/features provided by Hadoop. Our proposed solution does no such things.

sidHadoop; Secure Inter-Domain Hadoop is proposed in which different independent Hadoop deployments, that are geographically distant can securely communicate with each other. sidHadoop has use cases in academia as well as the business world. An organization with multiple offices each with its own deployment of Hadoop will be able to securely share resources with the implementation of sidHadoop. Similarly, educational institutes can share their resources with other institutes using sidHadoop.

With the help of sidHadoop, resources can be utilized to their full extent and it reduces the bandwidth usage because it only transmits the jobs to other clusters and other clusters only reply with the results. This solution offers no design changes in the core-Hadoop design that makes sure of the backward compatibility for already existing clusters. Our solution can be termed as an add-on for Hadoop.

It enhances the capability of Hadoop to be able to communicate with other Hadoop instances to share resources and if required, the data as well. First, the design of Inter-Domain Hadoop is discussed in which the two Hadoop instances will communicate with each other over WAN.

4.1 Inter-Domain Hadoop:

Inter-Domain Hadoop is a concept in which multiple independent Hadoop instances communicate with each other, separated by different administrative domains, in order to share their resident resources and/or data. One Hadoop instance requests the other Hadoop instances to perform some computation in the form of MR Job. The MR Job and the data if it's not already existing on the other clusters is then transmitted over WAN. The clusters perform the computation on their local cluster and sent the results back to the requesting clusters. These Hadoop instances could be geographically distant and the communication between them can only be performed over WAN. The below diagram shows the workflow of inter-domain Hadoop.

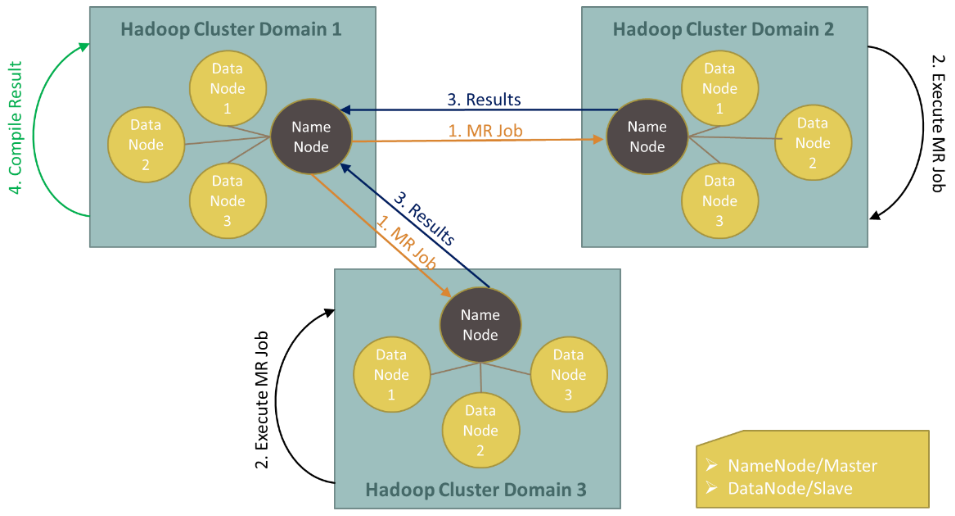


Figure 13: Inter-Domain Hadoop Design

Communicating over WAN is a huge security risk for private communications because the channels are not secure and any adversary can try and look into the conversation between multiple Hadoop instances. That is why securing the communication between inter-domain Hadoops is very important.

Hadoop itself provides sufficient security measures for a single domain environment. There are a number of security add-ons that can be used with Hadoop to enforce authentication and authorization mechanisms. These add-ons include Kerberos, Apache Ranger, Apache Sentry and Apache Knox. HDFS itself now provides Transparent Data Encryption for stored data. But when the switch towards inter-domain environment is made, these security mechanisms are not sufficient. In the inter-domain environment, the communication flows out of the private network which changes the security variables altogether. When communication flows over Wide Area Network, the threat matrix for Hadoop needs to be redefined.

The inter-domain Hadoop model is less emphasized upon the security and the solutions that discuss inter-domain Hadoop negate one of the core feature of Hadoop which states that Hadoop can be used on any commodity hardware and does not require any specialized equipment. The focus has now shifted towards cloud deployments of Hadoop, but the target of this research and development is that if multiple instances of Hadoop exist and there is a need of temporary scale then Hadoop instances should collaborate with each other to share available resources.

4.2 sidHadoop; Secure Inter-Domain Hadoop:

This project titled sidHadoop targets the security aspects of Hadoop when multiple Hadoop instances collaborate with each other. sidHadoop will provide a complete solution to use Hadoop in inter-domain environment. Hadoop collaboration will result in the sharing of resources between independent Hadoop instances across different administrative domains. The communication between multiple Hadoop instances needs to be secure to protect individual setups of Hadoop and avoid any misuse of physical resources. The below diagram shows the high-level design for adding security to the inter-domain Hadoop model.

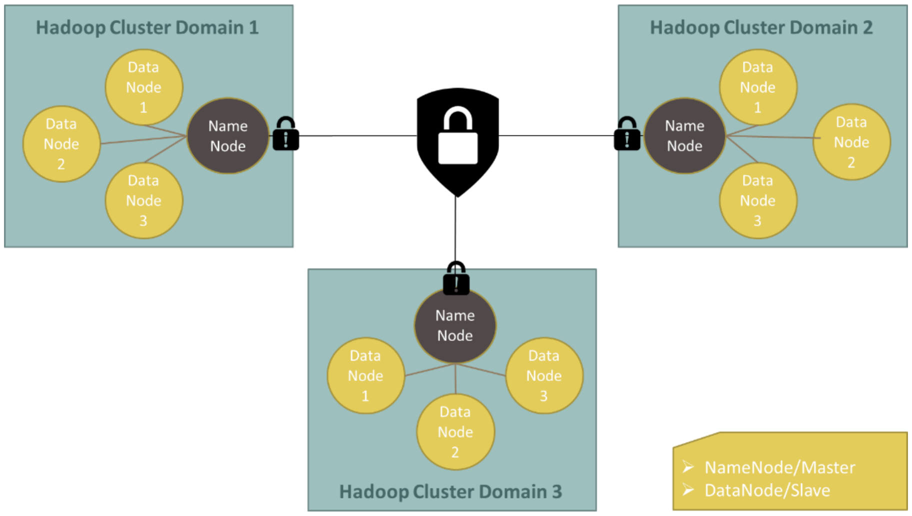


Figure 14: Adding security to Inter-Domain Hadoop

If Hadoop instances are at different geographical locations, then the communication between them will be required to happen over Wide Area Network. In general, a Hadoop setup is kept inside a private network to avoid any kind of exploitation from the outside world.

Our focus is on the security as well as the performance of Hadoop. The main services of security are authorization, authentication, confidentiality, integrity, accountability and availability [13] on which the Hadoop performance and security in separate environments will be judged. The two different lock symbols in the diagram above represent different security issues that this system will cater. These security issues are mentioned in Table 2.

Table 2: Security Objectives

Code	**End Point Security (EPS)**	**Code**	**Channel Security (CS)**
EPS-1	Mutual Authentication between servers	CS-1	Confidentiality of data/job
EPS-2	Authorization of servers	CS-2	Integrity of data/job
EPS-3	Source Integrity	CS-3	Avoid replay attacks
EPS-4	Single Sign-On	CS-4	Avoid MITM attacks

The security objectives are divided into two sections, End Point Security (EPS) and Channel Security (CS). These security measures are not existing in the inter-domain hadoop model and we propose to add these in our proposed secure inter-domain model. The security objectives are discussed below

4.2.1 End Point Security (EPS):

To ensure the security between two independent clusters, the public facing end points need to be protected from attacks. We set the following objectives to protect the end-point servers.

1. EPS-1 Mutual Authentication between servers
2. EPS-2 Authorization of servers
3. EPS-3 Source Integrity
4. EPS-4 Single Sign-On

4.2.1.1 EPS-1: Mutual Authentication between servers:

As discussed earlier, inter-domain hadoop model does not provide any security mechanism for the protection of our end-point servers. When two servers needed to communicate with each other, there was no check available to verify the authenticity of these clusters. That is why we are proposing to add Mutual Authentication between servers. The two servers will first authenticate each other at the start of the communication session so that the authenticity of each cluster is verified before resource collaboration could take place.

4.2.1.2 EPS-2: Authorization of servers:

The two security pillars Authentication and Authorization always go hand in hand. The authorization feature comes after the authentication is completed between two parties. The inter-domain hadoop lacks this feature as well. Any server could try and access the resources of our cluster without the proper check of authority. We are proposing to add authorization feature in our proposed solution, so that only authorized servers could use the resources of our cluster. After authentication, the authority given to the servers needs to be checked so that the servers which are not allowed to access certain data or resources are barred from accessing them.

4.2.1.3 EPS-3: Source Integrity:

Verifying that the source which is communicating is actually what it claims to be defines the source integrity. This provides protection from high jacking of session. Inter-domain hadoop provides no check for verifying the integrity of the source. In our proposed solution, if any undeclared change occurs to the source which is communicating, it will be considered that the source has been tampered with and the session will be terminated. The integrity of source will also help protect from man-in-the-middle attacks.

4.2.1.4 EPS-4: Single Sign-On:

Single Sign-On feature provides the user to access the different resources of the cluster by performing a single sign-in to the application. The client does not have to authenticate itself to each different services. In our proposed solution, we will add single sign-on feature so that the client only has to authenticate only once in our application, and will not have to authenticate itself to the remote clusters separately. The proposed solution will authenticate automatically to the remote cluster on behalf of the client.

4.2.2 Channel Security (CS):

When communication flows over public wide area network, the channel is the not prone to various attacks. The security of channel is as important as the security of the end points. We set the following objectives to protect the channel

1. CS-1 Confidentiality of data/job
2. CS-2 Integrity of data/job
3. CS-3 Avoid replay attacks
4. CS-4 Avoid MITM attacks

4.2.2.1 CS-1 Confidentiality of data/job

Inter-domain hadoop solution uses plain text channel for communication between servers. This means that the data between two servers could be read by any adversary from the communication channel. Our solution proposes to provide confidentiality of data so that our data is protected when it is traversing over public channel. With the confidentiality feature, our data will not be leaked to unauthorized sources that could be tapping the public channel.

4.2.2.2 CS-2 Integrity of data/job

It is very important to verify the integrity of data that is traversing over public channel. Just like other security features, inter-domain hadoop lacks in this aspect as well. The data integrity is not the concern in the inter-domain hadoop model, but instead, it only provides solution to connect multiple independent hadoop instances with each other. The proposed solution will also include the feature of verifying the integrity of data. Integrity of data ensures the originality of data that the data is not tampered with when it travels on the public channel.

4.2.2.3 CS-3 Avoid replay attacks

Capturing the communication session and using the same session on a later time is called relay attack. In the replay attack, the traffic is intercepted and is captured for later use. By doing this, an attacker high jacks the user session to a secure service by replaying the authentication traffic. The security checks can be bypassed and unauthorized access can be granted. In our solution, protection from replay attacks will be ensured. Mechanisms will be applied to avoid the replay of traffic from a previously captured session. By applying this feature unauthorized access to our secure servers will be avoided.

4.2.2.4 CS-4 Avoid MITM attacks

Man-In-the-Middle attack is a form of security threat in which an attacker comes between two parties, and acts as the authorized entity to which each party is communicating. The traffic from one side is intercepted and replayed to other side to make the other side believe that it is contacting the right entity. Protection from MITM attacks is very important for a secure system and our solution proposes to provide just that. By providing the right security mechanism, our system will not allow attacks from channel bullies and will provide security from any form of unauthorized access.

4.3 sidHadoop Architecture

sidHadoop maintains the core architecture of Hadoop with its master/slave storage and processing model. The proposed design deals with the remote communication over WAN. It allows one cluster to be able to communicate with another cluster only if it has the sidHadoop plugin deployed. The sidHadoop plugin makes sure that the security of the communication channel is achieved. Certain tests will be performed to show that the required security is achieved. A high-level diagram of sidHadoop is shown in Figure 15.

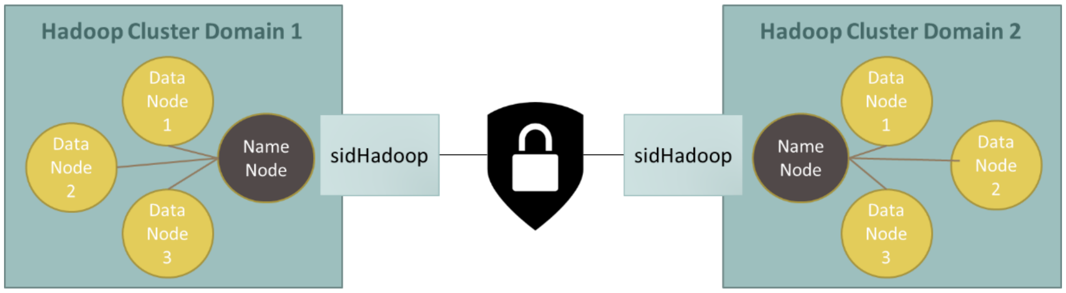


Figure 15: sidHadoop Plugin with Hadoop Clusters

sidHadoop is a web-based solution which consists of the following core modules; Authentication-Authorization module which is divided into sub-modules Client Authentication-Authorization module (C2A) and Remote Cluster Authentication-Authorization module (RC2A), Cluster Operations module which is divided into sub-modules Local Cluster Operations module (LCO) and Remote Cluster Operations module (RCO) and Trust management module which is divided into sub-modules Certificate Manager CM and User Manager UM.

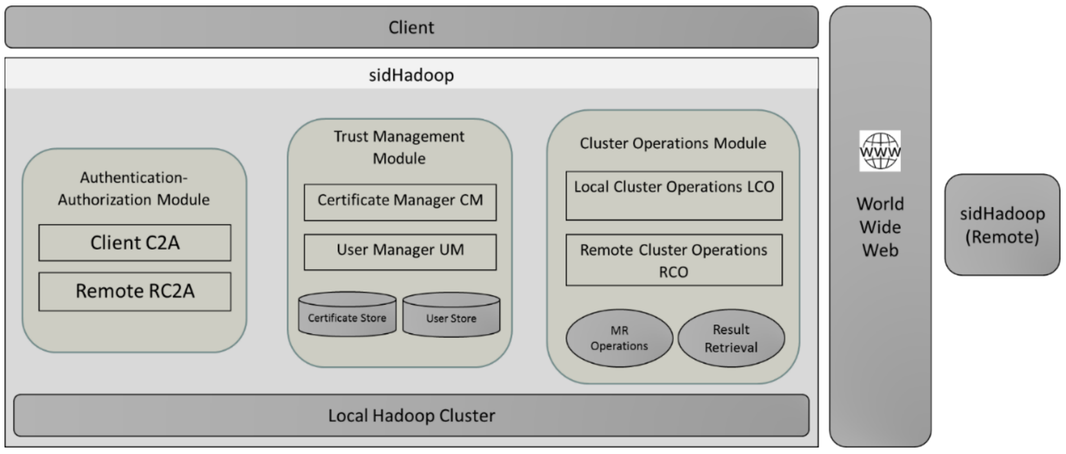


Figure 16: sidHadoop Components

The client first has to login to the application which decides the level of access allowed to the client. The access is reflected in the user interface that is displayed to the client. A user having access to only local cluster operations will be able to perform operations on the local cluster only. If the user is allowed access to remote cluster operations, then the user will be able to perform operations on the remote cluster.

Our application provides a certificate management module which deals with the remote Hadoop clusters resident inside other administrative domains. In this module, the administrator uploads a certificate of the trusted remote Hadoop cluster. These certificates are the first step in establishing a connection between remote clusters.

This is to be noted that the connection between two clusters will only be possible when both clusters have each other's certificate uploaded into the sidHadoop certificate management module. Remote Cluster Authentication module verifies if the requests coming from remote clusters are coming from the authorized source. Requests from remote source carry a certificate of their local domain. That certificate is verified by the certificate management module and only the verified requests are allowed to perform further operations on a local cluster by the remote cluster.

4.3.1 Authentication-Authorization Module:

When a client first logs on to the sidHadoop plugin, the authentication request is intercepted by Client Authentication-Authorization C2A sub-module. C2A then sends the request to Trust Management module. In Trust Management Module, User Manager UM validates the user credentials and access is granted to the user according to the role it has been assigned. The access is granted by C2A module to the client.

Similarly, when a request is received from a remote cluster, Remote Authentication-Authorization module intercepts the request. R2A then sends the request to Certificate Manager CM to validate the certificate credentials provided by the remote request. The R2A also replies with the certificate of its local cluster for mutual authentication. The remote cluster follows the same step of verification. This method ensures the 2-way SSL authentication between the local and remoter cluster. After the request is validated, only then the operations requested are performed. The R2A module takes care of the access that is granted to the remote request and sends the request to Cluster Operation Module.

4.3.2 Trust Management Module:

Trust management module is divided into two sub-modules, User Manager UM and Certificate Manager CM. User Manager has its own database of users by the name of User Store. When UM receives a request from C2A module, it checks the User store for the credentials that are provided. If the credentials are validated, the UM module replies with the role defined for this particular user. Finally, the C2A performs its action based on the reply from the UM.

Certificate Manager Module has its own database of trusted certificates by the name of Certificate Store. CM receives the remote request from R2A module for validation. CM then checks the certificate store for the trusted certificates that are available. If the provided certificate is already available in the cert store, then the CM replies with the authorized actions that are allowed to this particular remote cluster.

4.3.3 Cluster Operations Module:

Cluster Operations module directly deals with the Hadoop cluster. It is divided into two sub-modules that are Local Cluster Operations LCO module and Remote Cluster Operations RCO module. The LCO module directly interacts with the name node of the local Hadoop cluster to perform basic operations. The two basic operations that are allowed here are MR Job submission and retrieving the results back. When the user asks to execute some operation on its local cluster, the LCO module intercepts the request and sends the required instructions to name node. The name node executes the submitted MR job and notifies about the job completion. The LCO module then retrieves the result from the cluster to the local computer for the client to access.

If the client wishes to run a job on a remote cluster, then the Remote Cluster Operations module receives the request. RCO module is responsible for sending requests to remote clusters and also, it receives the requests from remote clusters to perform operations on the local cluster. The client submits an MR job to the application and requests to execute this job on the remote cluster. The RCO module receives the request along with the MR job, it then encrypts the job with two levels of PKI encryption. First encryption is performed using the private key of the local cluster. This will ensure the source integrity. Second, the encrypted bytes are re-encrypted using the public key of the remote cluster. The public key of the remote and private key of the local cluster are retrieved from the Cert Store. The encrypted MR job is then transmitted over secure SSL channel to the remote cluster.

The remote cluster first verifies the credentials as discussed in module 1. Then the received MR job is decrypted. The decryption process is the reverse of the process which was used for decryption. First, the job is decrypted using the private key of the local cluster and then it is re-decrypted using the public key of the remote cluster. The keys for decryption are also facilitated by the Certificate

Manager. After decryption, the RCO submits the job to LCO so that LCO can execute it on the local cluster. The LCO notifies of the job completion to RCO. LCO also retrieves the results from the cluster and submits them to RCO. The RCO then returns the results back to the remote cluster.

4.4 sidHadoop Workflow Scenario

The functionality of this solution will be discussed using three scenarios. sidHadoop allows users to execute jobs on remote clusters as well as on local cluster. The workflow of the solution is discussed based on the following scenarios.

1. Submit & Execute Job on local Cluster only
2. Submit & Execute Job on Remote Cluster
3. Receive & Execute job from Remote Cluster

4.4.1 Submit & Execute Job on Local Cluster:

Submitting and executing a job on local cluster is pretty straightforward. There is no remote communication required for this interaction. The client will first have to authenticate itself to the sidHadoop plugin. The authentication request is received by C2A module. In the second step, C2A module validates the credentials from UM. After validation, the third step is that the C2A module allows client access to Cluster Operations Module. The client then submits an MR job to LCO module in the fourth step. The LCO is responsible for interacting with the local cluster. The LCO then uploads the MR job to the local cluster and performs the execution in step five. After the job is completed, the LCO then retrieves the results from the local cluster and to the application in step six. Finally, the client is provided with the results received from the local cluster.

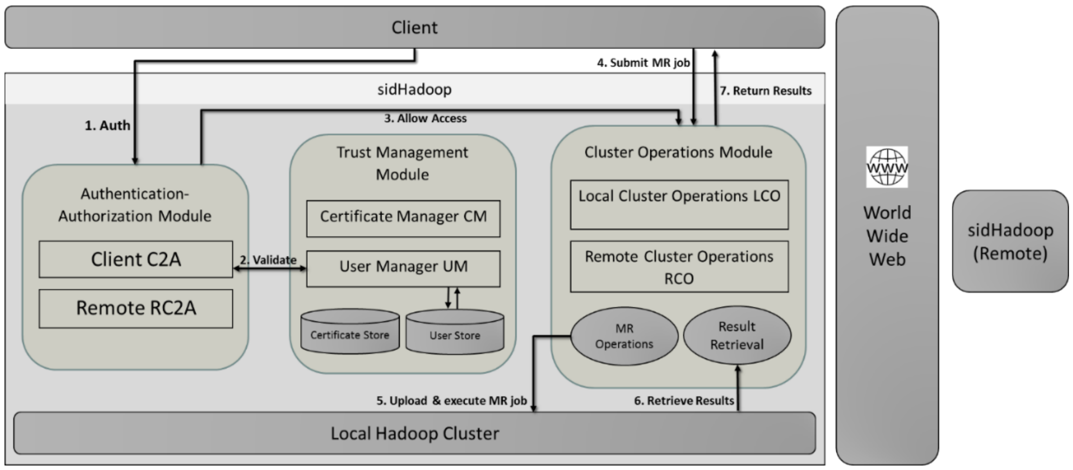


Figure 17: Execute Job on Local Cluster using sidHadoop

4.4.2 Submit & Execute Job on Remote Cluster:

Submitting and executing a job on the remote cluster is different and the core functionality of sidHadoop is explored in this scenario. Remote communication between the local and remote cluster is required for this interaction. The client will first have to authenticate itself to the sidHadoop plugin. The authentication request is received by C2A module. In the second step, C2A module validates the credentials from UM. After validation, in the third step, the C2A module allows client access to Cluster Operations Module. The client then submits the MR job to RCO module in the fourth step. The RCO is responsible for interacting with the remote clusters. The RCO then uploads the MR job to the Remote cluster and requests to execute this job in the step fine. The RCO then retrieves the results back from the remote cluster to the application in step six. Finally, the client is provided with the results received from the remote cluster.

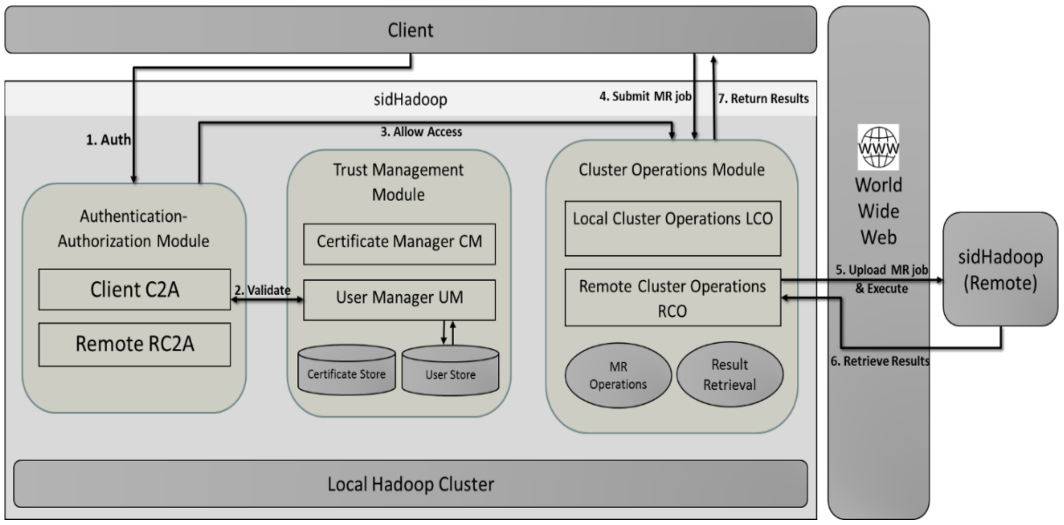


Figure 18: Execute Job on Remote Cluster with sidHadoop

4.4.3 Recieve & Execute Job on Remote Cluster:

The last scenario describes the process flow when a request is received from a remote cluster. In the second scenario, the client submits the job for execution on a remote cluster. In this scenario, the workflow is shown after the step five in the previous scenario and before the step 6. When the job is submitted to a remote cluster by RCO, it is accepted at the remote cluster but first, the remote cluster request is authenticated. The 2-way SSL authentication takes place transparently in this step. The RC2A module receives and validates the request from CM. After validation, in the third step, the RC2A module allows the request to access the Cluster Operations Module. In step four, the remote cluster uploads the MR job and requests to execute this job to RCO. The RCO is responsible for interacting with the remote clusters only so RCO sends this job and requests the LCO to execute the job in step five. The LCO performs the operations on local cluster as discussed in the first scenario and returns the results back to the RCO in the same step five. RCO then returns the results back to the Remote cluster in the final step.

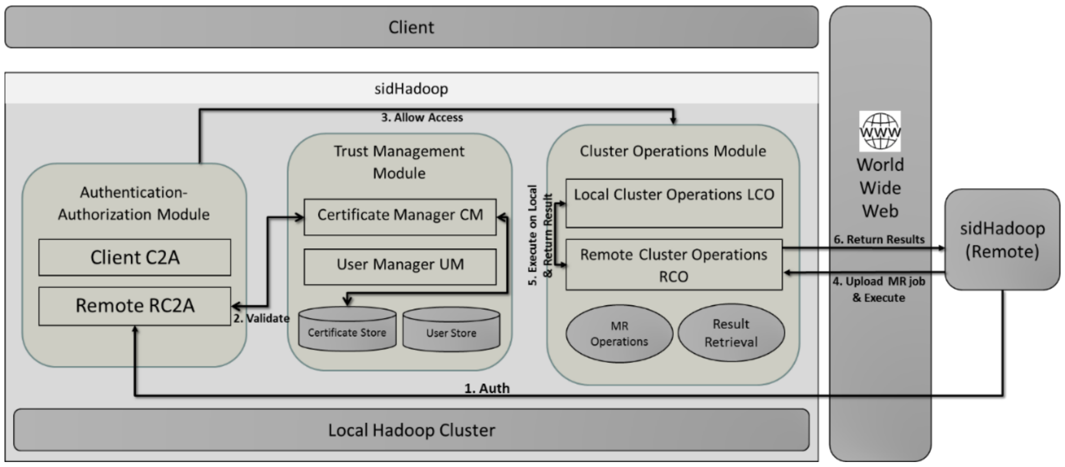


Figure 19: Recieve and Execute Job from Remote Cluster with sidHadoop

In the above-discussed scenarios, sidHadoop provides a management UI for submitting jobs to local and/or remote clusters. The next section discusses how this solution is achieving the security features discussed earlier in the security objectives table.

- The Authentication-Authorization module takes care of the very first security requirement. It authenticates the local clients with the application as well as it authenticates the remote clients from remote clusters with the help of the certificates.
- The 2-way SSL authentication between two remote clusters takes care of the remote authentication and source integrity.
- The verification of certificates and authentication based on certificates provides protection from MITM attacks.
- Protection from replay attacks is achieved by using ‘csrf tokens’ provided by spring framework.
- The encrypted SSL channel used between remote clusters encrypts the traffic to achieve confidentiality and the messages traversed over the channel are concatenated with message authentication code that ensures the integrity of job and data.
- Only authorized traffic is catered by this application so this achieves protection from unauthorized sources.

- The encrypted channel and 2-way SSL authentication also provide protection from job tempering.

sidHadoop application solves the issue of utilizing geographically distributed Hadoop resources by the concept introduced by inter-domain Hadoop. Instead of making one single Hadoop cluster consisting of resources spread across the globe, this solution makes multiple already existing Hadoop instances interact with each other. By using a single Hadoop cluster, the maintenance and management of resources becomes a bottleneck and requires a lot of design modifications in Hadoop. sidHadoop resolves this issue by making the local clusters manage their own resources. Our application provides a solution for secure collaboration of independent Hadoop instances so that the data and resources are shared between these clusters.

Chapter 5

Proof of Concept

5. Implementation:

This solution was developed with the aim of making it easier and secure for Hadoop administrators to collaborate their resources with other instances of independent Hadoop. It provides a management interface for posting jobs on local cluster and securely communicating the same over WAN to remote clusters.

5.1 Choice of Implementation Language and Platform:

In this solution spring framework is used for the implementation of the research modules. It provided support for designing and deploying web services with greater ease. The Spring Framework [48] is an application framework and inversion of control container for the Java platform. The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform. Although the framework does not impose any specific programming model, it has become popular in the Java community as an alternative to, or a replacement for, or even an addition to the Enterprise JavaBeans (EJB) [49] model. The Spring Framework is open source and provides support for security from the very core. The following diagram shows the architecture of the spring framework.

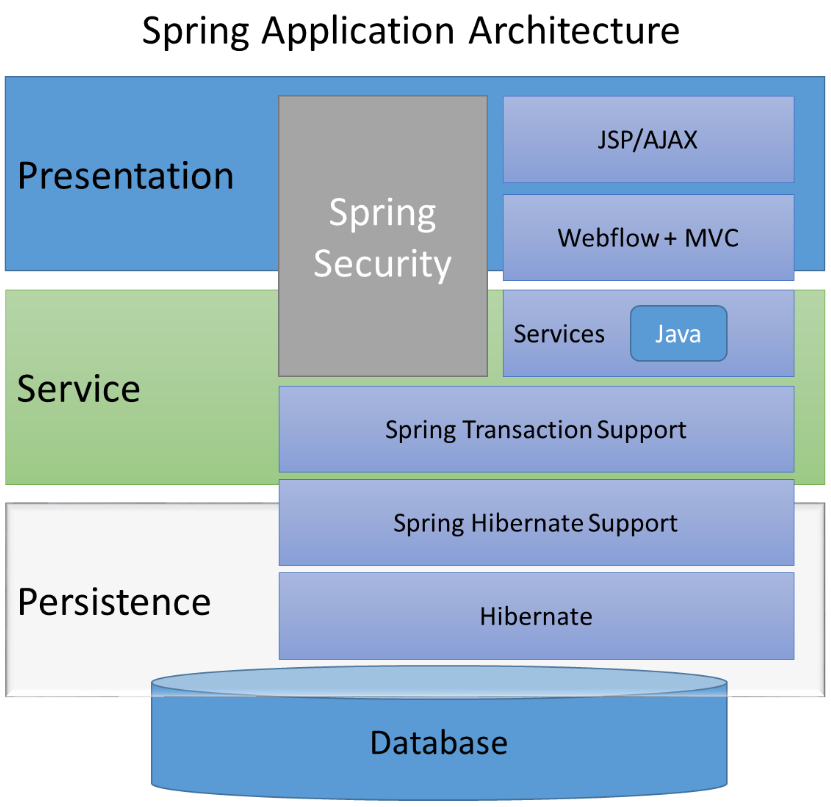


Figure 20: Spring Framework Architecture

Spring's web framework is a well-designed web MVC framework, which provides a great alternative to web frameworks such as Struts or other over-engineered or less popular web frameworks [50]. In addition to that, it offers a modular approach where the different modules can interact with ease. Reusability, troubleshooting and easy alteration of the code were also the added benefits. The main objective for using spring framework was its support for the implementation of security mechanisms. The spring security framework offers a variety of inbuilt security features to implement on your application [51]. It provides a wide range of authentication authorization modules which support integration with the following technologies and more,

- HTTP BASIC authentication headers [52]
- HTTP Digest authentication headers [53]
- HTTP X.509 client certificate exchange [54]
- LDAP [55]

- Form-based authentication (for simple user interface needs)
- OpenID authentication [56]

5.2 Test Environment

For the test environment, two clusters of Hadoop were used that were deployed in two different administrative domains. The very basic deployment of Hadoop was used in which each cluster had one master node and one slave node and both these clusters were kept on separate networks. The specification for both the clusters was kept same. The specifications of nodes are as follows

Table 3: Namenode / Master Node Specifications

Processors	**RAM**	**Storage**	**OS**	**Hadoop**
4 Cores	8 GB	500 GB	Ubuntu 16.4	Apache Hadoop v2.7.3

Table 4: Datanode / Slave Node Specifications

Processors	**RAM**	**Storage**	**OS**	**Hadoop**
2 Cores	4 GB	500 GB	Ubuntu 16.4	Apache Hadoop v2.7.3

5.3 Proof of Concept:

In the proof of concept PoC, the HTTP X.509 [54] client certificate exchange and form-based authentication mechanisms were used. The client first authenticates itself with the application using form based authentication. The form-based authentication is protected from replay attacks with the use of csrf token [57] provided by spring framework. The tokens are used to ensure that each request requires, in addition to the session cookie, a randomly generated token as an HTTP parameter. When a request is submitted, the server must look up the expected value for the parameter and compare it against the actual value in the request. If the values do not match, the request should fail. The form-based authentication gives the access to the sidHadoop application.

5.2.1 Login and Registration View:

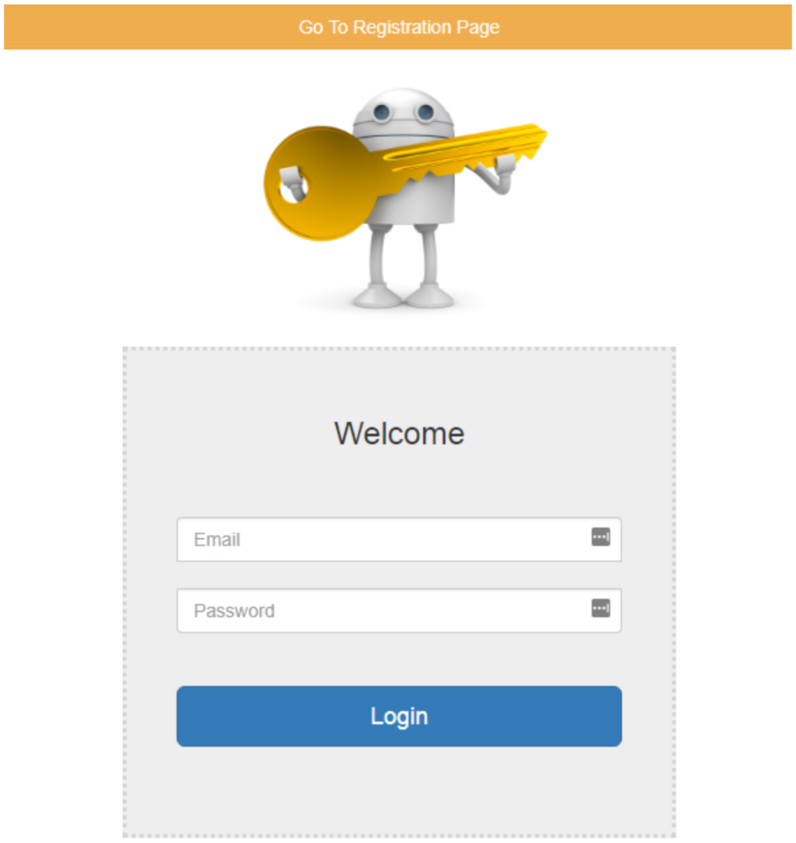
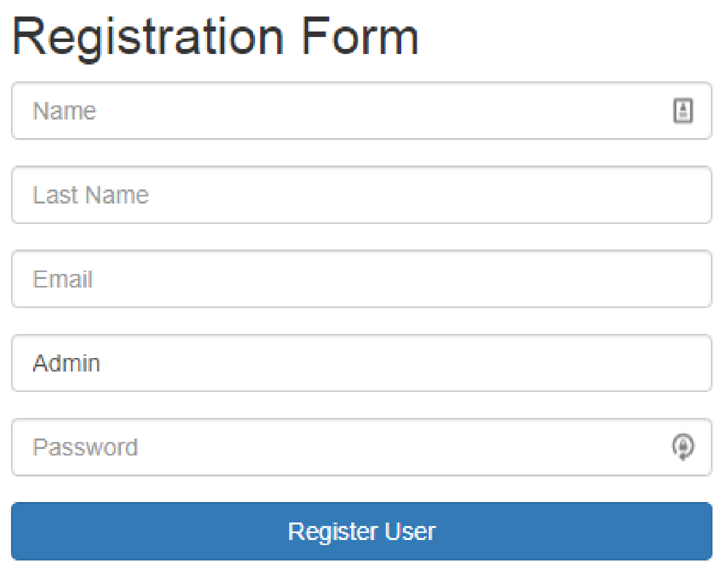


Figure 21: Login page

The login view also contains a link to the registration page. Adding of new users is performed from the Registration page.


Registration Form

Name

Last Name

Email

Admin

Password

Register User

Figure 22: Registration Page

For remote servers to authenticate each other, 2-way SSL mutual authentication is used. The HTTP X.509 client certificate exchange solution is supported by spring security framework and it provides the feature of mutual authentication between remote servers. When the client makes a request for the remote server, its digital certificate is also attached with the request. The server first checks the client certificate and shares its own certificate with the remote client. The remote client verifies the server certificate and replies with an acknowledgment. The server receives the acknowledgment and confirms the request for further processing. The validated request is handed over to the concerned module by the server.

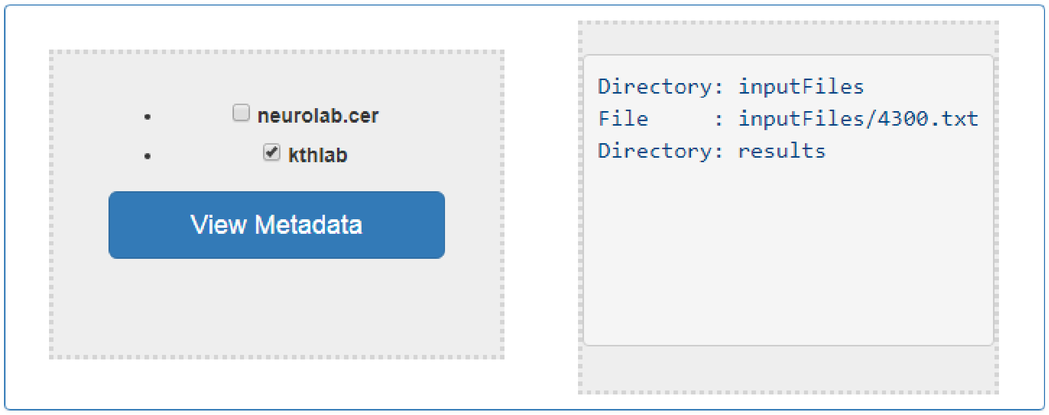
After the client is authenticated to the application, it is served with the main view which performs the core cluster operations for Hadoop. There are three main operations that are performed using the sidHadoop application,

1. View meta-data of the cluster
2. Upload and Execute jar file to the cluster
3. Fetch results from the cluster.

The three operations have a separate section in the view. These core operations define the basic functionality of Hadoop.

5.2.2 Meta Data View:

The meta-data of the cluster is the list of files that are stored in the hdfs. Viewing the meta-data of the cluster shows the directory structure of the files on which operations can be executed. The operations are in the form of an MR job. MR, as discussed in the former sections, is the programming model based on multiple map and reduce operations. The user designs an MR job and creates a jar file. The jar file is then transmitted to the Hadoop cluster for execution. After the MR job has completed execution, the result file can be retrieved from the cluster in the last operation mentioned above.

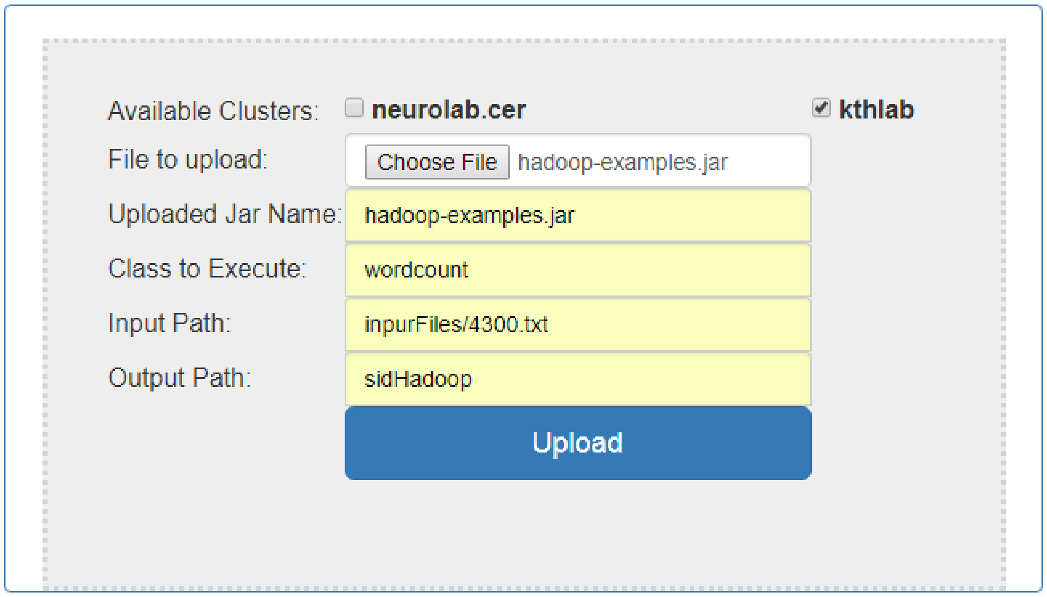


```
Directory: inputFiles
File      : inputFiles/4300.txt
Directory: results
```

Figure 23: MetaData from Hadoop Cluster

In the view meta section, the list of authorized sidHadoop clusters is displayed in the form of checkboxes. The client has to select the cluster and press the view meta button to view its metadata. When the client selects a cluster, the backend business logic checks if the selected cluster is local or remote. If the cluster is local, the module, named Local Cluster Operations (LCO), is given the request for further execution. The LCO sends the meta-data request to the local cluster and returns the result for the view to display. On the other hand, if the request is for the remote cluster, the RCO remote cluster operations module is given the request for further execution. The RCO compiles the request according to the specifications of the remote cluster and sends the new request for execution.

5.2.3 Upload and Execute jar view


Available Clusters: ☐ neurolab.cer ☑ kthlab

File to upload: Choose File hadoop-examples.jar

Uploaded Jar Name: hadoop-examples.jar

Class to Execute: wordcount

Input Path: inpurFiles/4300.txt

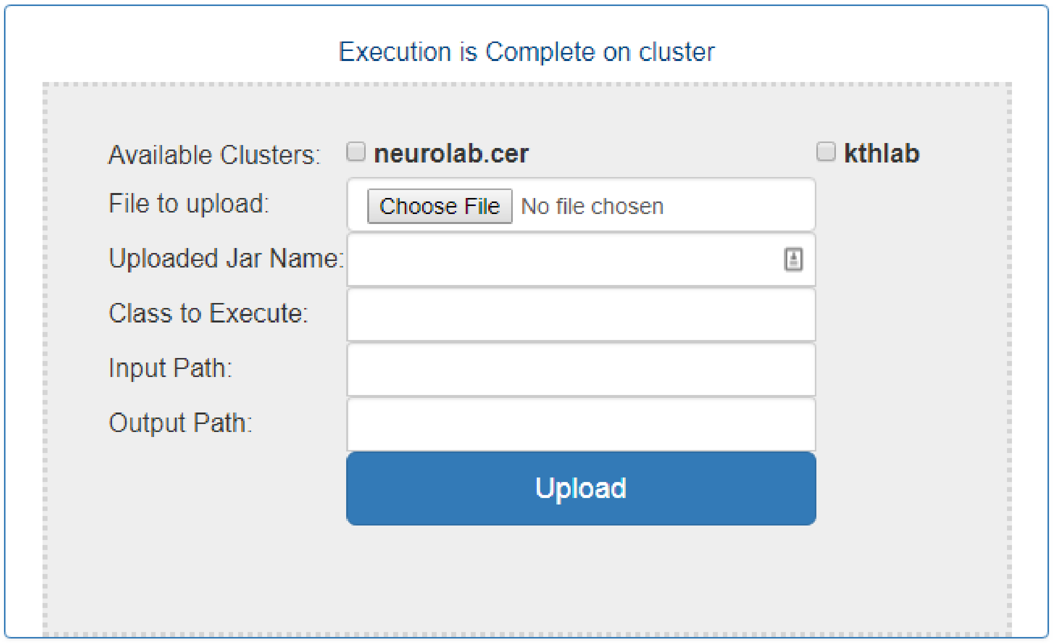
Output Path: sidHadoop

Figure 24: Upload and Execute MR job

The view for Upload and execution of an MR job is shown above. This view also lists down the clusters that are available for execution of MR job. The client has to upload a jar file and provide certain parameters according to the uploaded jar. These parameters are used in the Hadoop command for execution of MR job. Hadoop command is provided below,

>> **$HADOOP_HOME$/bin/Hadoop**<space>**jar**<space>"**jar name + path**"<space>"**class name**"<space>"**input file path**"<space>"**output file path**"

The parameters in Figure 24 are included in the above command and this command is executed on the Hadoop cluster. The selection of cluster and flow of request for the local or remote cluster is performed in the same way as discussed above in the metadata section. After the MR job is executed, the application is notified by a simple message of job completion.



Execution is Complete on cluster

Available Clusters:	☐ **neurolab.cer**	☐ **kthlab**
File to upload:	Choose File No file chosen	
Uploaded Jar Name:		
Class to Execute:		
Input Path:		
Output Path:		

Figure 25: Execution Complete Notification

5.2.4 Results of MR Job View

Finally, the results generated by the executed job are to be retrieved from the executing cluster. The view for fetching the results only requires the name of the output file from client given at the time of execution. The required cluster is selected from the list of available clusters and the name of the output path is provided. The successful retrieval of result file gives a notification similar to one shown in the above figure.

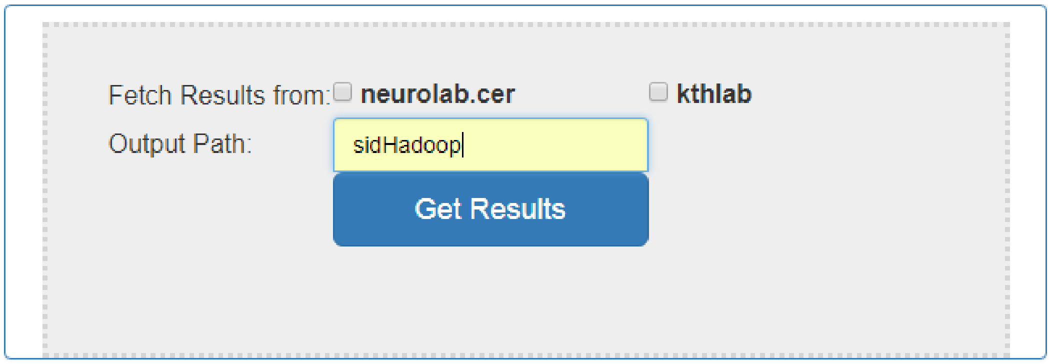


Figure 26: Get Result from Cluster

5.2.5 Certificate Management View

Apart from the cluster operations, there is a view which handles the certificate management. In this view, certificates of the remote clusters are added and removed. When a new certificate is added, the certificate management module adds the new certificate in the trusted chain of the local keystore. At the time of the removal, the certificate is selected from the list of available certificates and is removed from the trusted chain of local keystore.

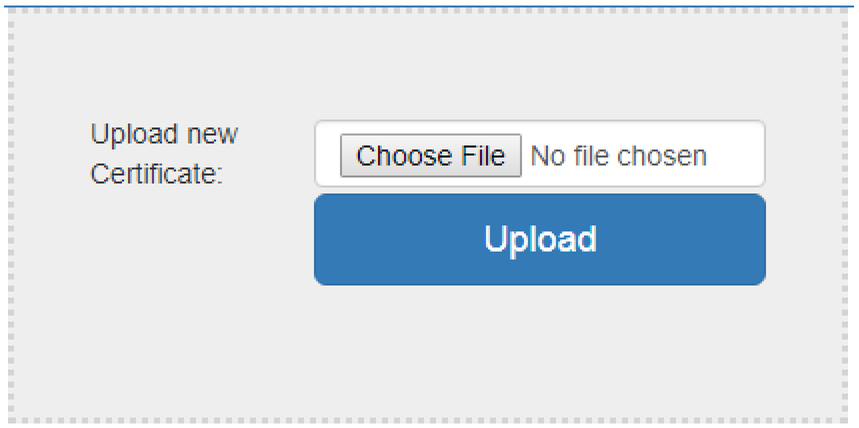


Figure 27: Upload New Certificate

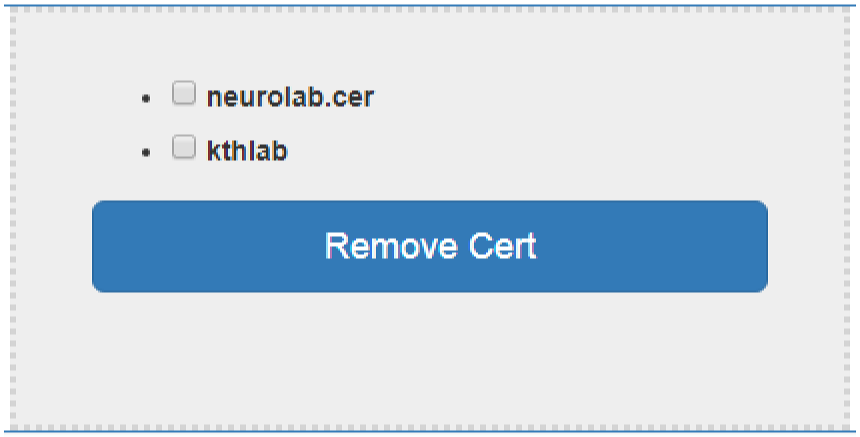


Figure 28: Remove Certificate

Chapter 6

Results and Discussion

6. Results and Discussion

To validate this application, certain tests were performed on the web-service using different security tools. The tools used are Wireshark to analyze the traffic on the wire and Web Vulnerability scanner to scan for any vulnerabilities that might exist in the application.

First, the communication between two Hadoop instances in an inter-domain environment without security implementation is shown and comparison of the traffic with the secure inter-domain solution for Hadoop is done. To analyze the security provided by sidHadoop, this system was studies on the parameters of security mentioned in the proposed solution section.

- End Point Security
- Channel Security

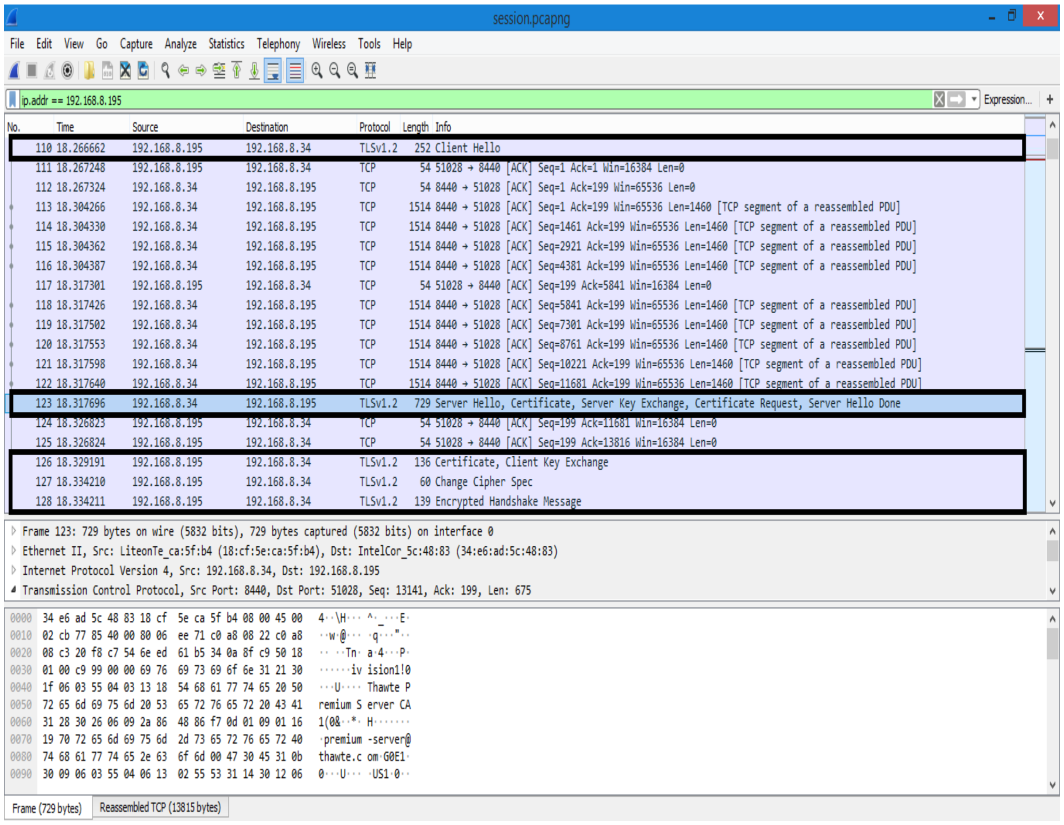
6.1 End Point Security:

In the End-point security section, this system achieves the following,

1. EPS-1 Mutual Authentication between servers
2. EPS-2 Authorization of servers
3. EPS-3 Source Integrity
4. EPS-4 Single Sign-On

6.1.1 EPS-1: Mutual Authentication between servers:

In *sidHadoop*, digital certificates of the two parties are used, so these two parties are authenticated by using 2-way SSL authentication. Mutual authentication is achieved by the two parties before the start of each session. To achieve ***EPS-1*** objective, both parties shared their certificates with each other and stored them in the list of trusted certificates. With each request, the requesting party provides its certificate to the remote party for verification and vice versa. After the two parties had authenticated each other, only then the rest of the request is initiated. Authentication between the two independent servers is key to control the access to our secure hadoop clusters. Security of the hadoop clusters from unauthenticated sources is provided by achieving ***EPS-1***. The below figure shows the requests that were carried out between two servers for the purpose of mutual authentication.


session.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...

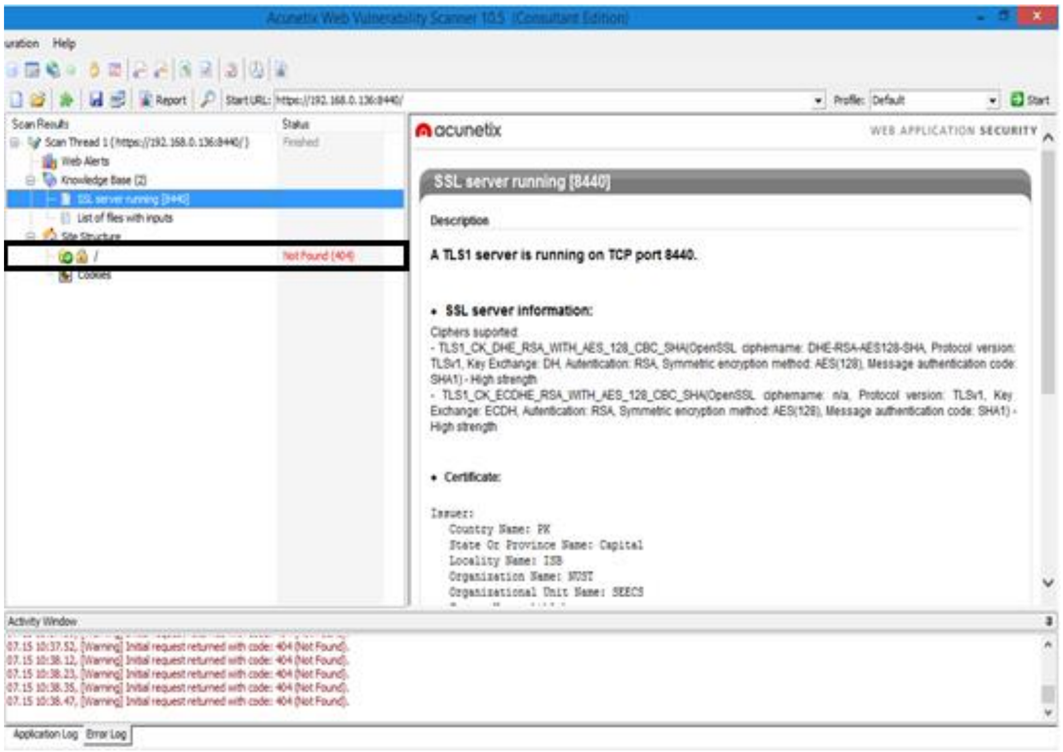
```
ip.addr == 192.168.8.195

No.  Time       Source         Destination    Protocol Length Info
110 18.266662  192.168.8.195  192.168.8.34   TLSv1.2   252 Client Hello
111 18.267248  192.168.8.195  192.168.8.34   TCP        54 51028 → 8440 [ACK] Seq=1 Ack=1 Win=16384 Len=0
112 18.267324  192.168.8.34   192.168.8.195  TCP        54 8440 → 51028 [ACK] Seq=1 Ack=199 Win=65536 Len=0
113 18.304266  192.168.8.34   192.168.8.195  TCP      1514 8440 → 51028 [ACK] Seq=1 Ack=199 Win=65536 Len=1460 [TCP segment of a reassembled PDU]
114 18.304330  192.168.8.34   192.168.8.195  TCP      1514 8440 → 51028 [ACK] Seq=1461 Ack=199 Win=65536 Len=1460 [TCP segment of a reassembled PDU]
115 18.304362  192.168.8.34   192.168.8.195  TCP      1514 8440 → 51028 [ACK] Seq=2921 Ack=199 Win=65536 Len=1460 [TCP segment of a reassembled PDU]
116 18.304387  192.168.8.34   192.168.8.195  TCP      1514 8440 → 51028 [ACK] Seq=4381 Ack=199 Win=65536 Len=1460 [TCP segment of a reassembled PDU]
117 18.317301  192.168.8.195  192.168.8.34   TCP        54 51028 → 8440 [ACK] Seq=199 Ack=5841 Win=16384 Len=0
118 18.317426  192.168.8.34   192.168.8.195  TCP      1514 8440 → 51028 [ACK] Seq=5841 Ack=199 Win=65536 Len=1460 [TCP segment of a reassembled PDU]
119 18.317502  192.168.8.34   192.168.8.195  TCP      1514 8440 → 51028 [ACK] Seq=7301 Ack=199 Win=65536 Len=1460 [TCP segment of a reassembled PDU]
120 18.317553  192.168.8.34   192.168.8.195  TCP      1514 8440 → 51028 [ACK] Seq=8761 Ack=199 Win=65536 Len=1460 [TCP segment of a reassembled PDU]
121 18.317598  192.168.8.34   192.168.8.195  TCP      1514 8440 → 51028 [ACK] Seq=10221 Ack=199 Win=65536 Len=1460 [TCP segment of a reassembled PDU]
122 18.317640  192.168.8.34   192.168.8.195  TCP      1514 8440 → 51028 [ACK] Seq=11681 Ack=199 Win=65536 Len=1460 [TCP segment of a reassembled PDU]
123 18.317696  192.168.8.34   192.168.8.195  TLSv1.2   729 Server Hello, Certificate, Server Key Exchange, Certificate Request, Server Hello Done
124 18.326823  192.168.8.195  192.168.8.34   TCP        54 51028 → 8440 [ACK] Seq=199 Ack=11681 Win=16384 Len=0
125 18.326824  192.168.8.195  192.168.8.34   TCP        54 51028 → 8440 [ACK] Seq=199 Ack=13816 Win=16384 Len=0
126 18.329191  192.168.8.195  192.168.8.34   TLSv1.2   136 Certificate, Client Key Exchange
127 18.334210  192.168.8.195  192.168.8.34   TLSv1.2    60 Change Cipher Spec
128 18.334211  192.168.8.195  192.168.8.34   TLSv1.2   139 Encrypted Handshake Message

▷ Frame 123: 729 bytes on wire (5832 bits), 729 bytes captured (5832 bits) on interface 0
▷ Ethernet II, Src: LiteonTe_ca:5f:b4 (18:cf:5e:ca:5f:b4), Dst: IntelCor_5c:48:83 (34:e6:ad:5c:48:83)
▷ Internet Protocol Version 4, Src: 192.168.8.34, Dst: 192.168.8.195
◢ Transmission Control Protocol, Src Port: 8440, Dst Port: 51028, Seq: 13141, Ack: 199, Len: 675

0000  34 e6 ad 5c 48 83 18 cf  5e ca 5f b4 08 00 45 00   4··\H··· ^·_···E·
0010  02 cb 77 85 40 00 80 06  ee 71 c0 a8 08 22 c0 a8   ··w·@··· ·q···"··
0020  08 c3 20 f8 c7 54 6e ed  61 b5 34 0a 8f c9 50 18   ·· ··Tn· a·4···P·
0030  01 00 c9 99 00 00 69 76  69 73 69 6f 6e 31 21 30   ······iv ision1!0
0040  1f 06 03 55 04 03 13 18  54 68 61 77 74 65 20 50   ···U···· Thawte P
0050  72 65 6d 69 75 6d 20 53  65 72 76 65 72 20 43 41   remium S erver CA
0060  31 28 30 26 06 09 2a 86  48 86 f7 0d 01 09 01 16   1(0&··*· H·······
0070  19 70 72 65 6d 69 75 6d  2d 73 65 72 76 65 72 40   ·premium -server@
0080  74 68 61 77 74 65 2e 63  6f 6d 00 47 30 45 31 0b   thawte.c om·G0E1·
0090  30 09 06 03 55 04 06 13  02 55 53 31 14 30 12 06   0···U··· ·US1·0··

Frame (729 bytes)   Reassembled TCP (13815 bytes)
```

Figure 29: Encrypted SSL Handshake between multiple sidHadoop

6.1.2 EPS-2: Authorization of servers:

In sidHadoop, the access to remote clusters is only provided to authorized sources. The authority/roles based authorization is used in our proposed solution. The authority to run jobs on our local cluster is not given to every remote clusters.

The access of every remote cluster to our local one is limited and after authentication, verification of authority given to remote cluster is also checked. This provides protection from servers that have been barred from using our remote clusters. The user may have only read authority to our server and may not have execution rights. The ***EPS-2*** objective ensures that the access is granted to remote cluster according to the authority a server has. Figure 29, shows the vulnerability assessment test performed using Acunetix Web Vulnerability Scanner. It clearly shows that the tests returned '404 Server not Found' error which is only because the scanner does not have an authorized certificate to access the server URL.



SSL server running [8440]

Description

A TLS1 server is running on TCP port 8440.

- SSL server information:

Ciphers suported:
- TLS1_CK_DHE_RSA_WITH_AES_128_CBC_SHA(OpenSSL ciphername: DHE-RSA-AES128-SHA, Protocol version: TLSv1, Key Exchange: DH, Autentication: RSA, Symmetric encryption method: AES(128), Message authentication code: SHA1) - High strength
- TLS1_CK_ECDHE_RSA_WITH_AES_128_CBC_SHA(OpenSSL ciphername: n/a, Protocol version: TLSv1, Key Exchange: ECDH, Autentication: RSA, Symmetric encryption method: AES(128), Message authentication code: SHA1) - High strength

- Certificate:

```
Issuer:
   Country Name: PK
   State Or Province Name: Capital
   Locality Name: ISB
   Organization Name: NUST
   Organizational Unit Name: SEECS
```

```
07.15 10:37.52, [Warning] Initial request returned with code: 404 (Not Found).
07.15 10:38.12, [Warning] Initial request returned with code: 404 (Not Found).
07.15 10:38.23, [Warning] Initial request returned with code: 404 (Not Found).
07.15 10:38.35, [Warning] Initial request returned with code: 404 (Not Found).
07.15 10:38.47, [Warning] Initial request returned with code: 404 (Not Found).
```

Figure 30: Vulnerability Scan result

6.1.3 EPS-3: Source Integrity:

Verification of Source Integrity is an additional check to provide access to authorized servers only. It ensures the originality of the source, and our solution provides verification of source integrity by using the above mentioned certificates. In Figure 28, the mutual authentication is shown between the two servers. The authentication steps include the sharing of digital certificates and verification of these certificates provide source integrity. The verification of the certificate before each request ensures that the request is coming from an authenticated and valid source hence validating ***EPS-3*** objective. Any unauthorized request simply gets the response of server error because all service requests are only visible to sources after authentication. The test also shows that the web server is equipped with TLSv1 security and is using an encrypted channel for communication. Any authorized server that has been modified by an attacker is not provided access to the cluster, because the integrity of the source is compromised.

6.1.4 EPS-4: Single Sign-On:

In sidHadoop, the client only performs a single sign-in on the application. The client does not require to provide their credentials to remote cluster every time they access one. The application handles this automatically and completes the ***EPS-4*** objective. When the request is sent by client to remote cluster, the sidHadoop authenticates to the remote cluster on behalf of the client. This way, the client credentials are not sent to multiple sources for verification, instead the local application verifies them and authenticate with remote server using server certificate on behalf of the client.

6.2 Channel Security:

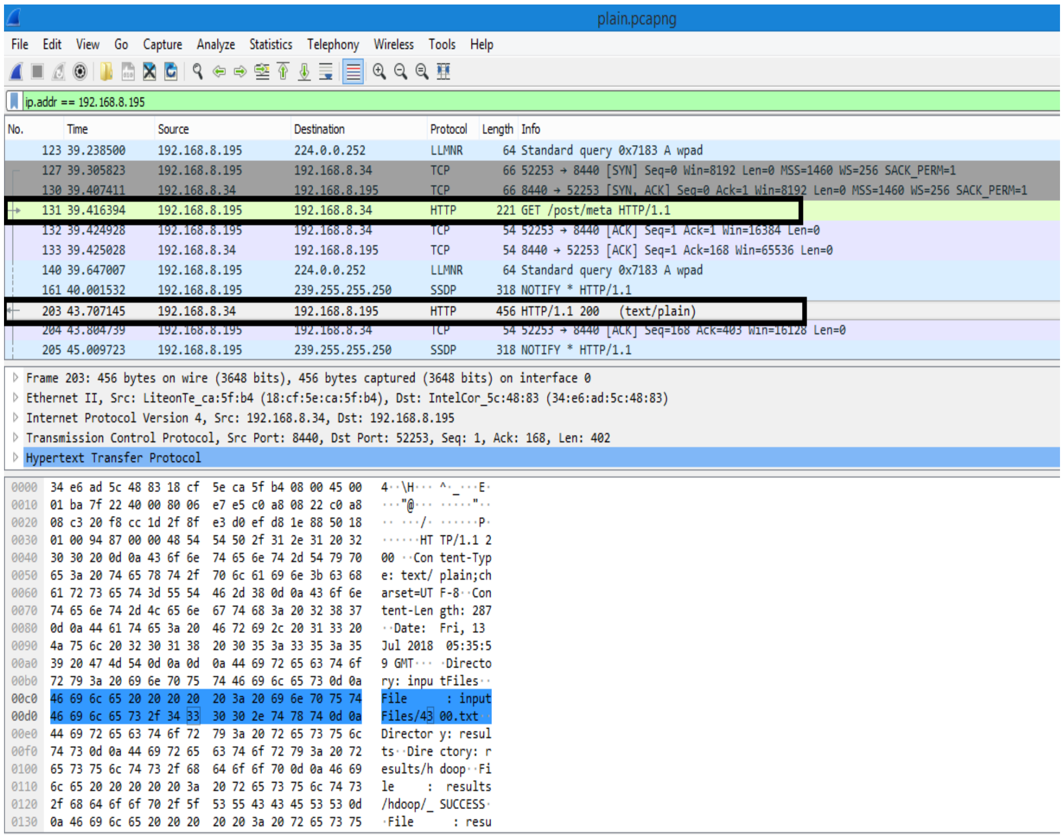
In the Channel security section, this system achieves the following,

1. CS-1 Confidentiality of data/job
2. CS-2 Integrity of data/job
3. CS-3 Avoid replay attacks
4. CS-4 Avoid MITM attacks

6.2.1 CS-1 Confidentiality of data/job

Inter-domain Hadoop solution uses plaintext channel to share requests and data over WAN. This loses the confidentiality for the data and requests. Securing the channel in communication between multiple sidHadoop instances is the main task and is being achieved in this solution. sidHadoop shares the requests and data over encrypted SSL channel. The encrypted channel ensures the confidentiality of data by sending cipher text instead of plaintext over WAN.

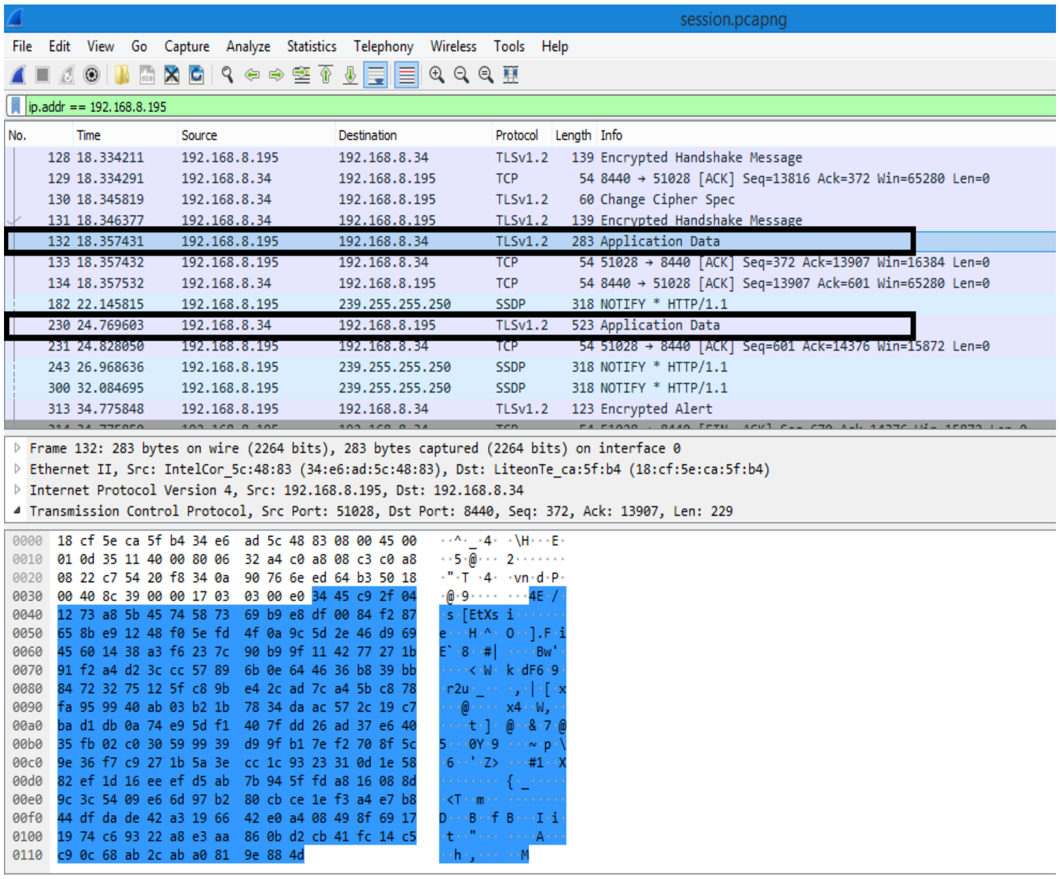
The Figure 31, shows the traffic when inter-domain Hadoop communication is performed. The request and response to and from the server are all served in plaintext. The request in the above figure is asking the remote cluster to return the meta-data from the Hadoop cluster. The meta-data from Hadoop cluster shows the list of files and directory structure in the Hadoop distributed file system. This information maybe sensitive for some organizations and needs to be protected.


plain.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
ip.addr == 192.168.8.195

No.  Time       Source         Destination      Protocol Length Info
 123 39.238500  192.168.8.195  224.0.0.252      LLMNR      64 Standard query 0x7183 A wpad
 127 39.305823  192.168.8.195  192.168.8.34     TCP        66 52253 → 8440 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
 130 39.407411  192.168.8.34   192.168.8.195    TCP        66 8440 → 52253 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
 131 39.416394  192.168.8.195  192.168.8.34     HTTP      221 GET /post/meta HTTP/1.1
 132 39.424928  192.168.8.195  192.168.8.34     TCP        54 52253 → 8440 [ACK] Seq=1 Ack=1 Win=16384 Len=0
 133 39.425028  192.168.8.34   192.168.8.195    TCP        54 8440 → 52253 [ACK] Seq=1 Ack=168 Win=65536 Len=0
 140 39.647007  192.168.8.195  224.0.0.252      LLMNR      64 Standard query 0x7183 A wpad
 161 40.001532  192.168.8.195  239.255.255.250  SSDP      318 NOTIFY * HTTP/1.1
 203 43.707145  192.168.8.34   192.168.8.195    HTTP      456 HTTP/1.1 200   (text/plain)
 204 43.804739  192.168.8.195  192.168.8.34     TCP        54 52253 → 8440 [ACK] Seq=168 Ack=403 Win=16128 Len=0
 205 45.009723  192.168.8.195  239.255.255.250  SSDP      318 NOTIFY * HTTP/1.1

Frame 203: 456 bytes on wire (3648 bits), 456 bytes captured (3648 bits) on interface 0
Ethernet II, Src: LiteonTe_ca:5f:b4 (18:cf:5e:ca:5f:b4), Dst: IntelCor_5c:48:83 (34:e6:ad:5c:48:83)
Internet Protocol Version 4, Src: 192.168.8.34, Dst: 192.168.8.195
Transmission Control Protocol, Src Port: 8440, Dst Port: 52253, Seq: 1, Ack: 168, Len: 402
Hypertext Transfer Protocol

0000  34 e6 ad 5c 48 83 18 cf  5e ca 5f b4 08 00 45 00   4··\H··· ^·_···E·
0010  01 ba 7f 22 40 00 80 06  e7 e5 c0 a8 08 22 c0 a8   ···"@··· ·····"··
0020  08 c3 20 f8 cc 1d 2f 8f  e3 d0 ef d8 1e 88 50 18   ·· ···/· ······P·
0030  01 00 94 87 00 00 48 54  54 50 2f 31 2e 31 20 32   ······HT TP/1.1 2
0040  30 30 20 0d 0a 43 6f 6e  74 65 6e 74 2d 54 79 70   00 ··Con tent-Typ
0050  65 3a 20 74 65 78 74 2f  70 6c 61 69 6e 3b 63 68   e: text/ plain;ch
0060  61 72 73 65 74 3d 55 54  46 2d 38 0d 0a 43 6f 6e   arset=UT F-8··Con
0070  74 65 6e 74 2d 4c 65 6e  67 74 68 3a 20 32 38 37   tent-Len gth: 287
0080  0d 0a 44 61 74 65 3a 20  46 72 69 2c 20 31 33 20   ··Date:  Fri, 13 
0090  4a 75 6c 20 32 30 31 38  20 30 35 3a 33 35 3a 35   Jul 2018  05:35:5
00a0  39 20 47 4d 54 0d 0a 0d  0a 44 69 72 65 63 74 6f   9 GMT··· ·Directo
00b0  72 79 3a 20 69 6e 70 75  74 46 69 6c 65 73 0d 0a   ry: inpu tFiles··
00c0  46 69 6c 65 20 20 20 20  20 3a 20 69 6e 70 75 74   File     : input
00d0  46 69 6c 65 73 2f 34 33  30 30 2e 74 78 74 0d 0a   Files/43 00.txt··
00e0  44 69 72 65 63 74 6f 72  79 3a 20 72 65 73 75 6c   Director y: resul
00f0  74 73 0d 0a 44 69 72 65  63 74 6f 72 79 3a 20 72   ts··Dire ctory: r
0100  65 73 75 6c 74 73 2f 68  64 6f 6f 70 0d 0a 46 69   esults/h doop··Fi
0110  6c 65 20 20 20 20 20 3a  20 72 65 73 75 6c 74 73   le     :  results
0120  2f 68 64 6f 6f 70 2f 5f  53 55 43 43 45 53 53 0d   /hdoop/_ SUCCESS·
0130  0a 46 69 6c 65 20 20 20  20 20 3a 20 72 65 73 75   ·File      : resu
```

Figure 31: Plaintext communication by inter-domain Hadoop

sidHadoop on the other hand uses the encrypted channel to secure the communication to remote clusters. Figure 32, shows the traffic when sidHadoop communication is performed. One can clearly see that the traffic in case of sidHadoop is encrypted and thus fulfilling the ***CS-1*** objective.


session.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
ip.addr == 192.168.8.195
```

```
No.   Time          Source          Destination      Protocol  Length  Info
  128 18.334211     192.168.8.195   192.168.8.34     TLSv1.2   139 Encrypted Handshake Message
  129 18.334291     192.168.8.34    192.168.8.195    TCP        54 8440 → 51028 [ACK] Seq=13816 Ack=372 Win=65280 Len=0
  130 18.345819     192.168.8.34    192.168.8.195    TLSv1.2    60 Change Cipher Spec
  131 18.346377     192.168.8.34    192.168.8.195    TLSv1.2   139 Encrypted Handshake Message
  132 18.357431     192.168.8.195   192.168.8.34     TLSv1.2   283 Application Data
  133 18.357432     192.168.8.195   192.168.8.34     TCP        54 51028 → 8440 [ACK] Seq=372 Ack=13907 Win=16384 Len=0
  134 18.357532     192.168.8.34    192.168.8.195    TCP        54 8440 → 51028 [ACK] Seq=13907 Ack=601 Win=65280 Len=0
  182 22.145815     192.168.8.195   239.255.255.250  SSDP      318 NOTIFY * HTTP/1.1
  230 24.769603     192.168.8.34    192.168.8.195    TLSv1.2   523 Application Data
  231 24.828050     192.168.8.195   192.168.8.34     TCP        54 51028 → 8440 [ACK] Seq=601 Ack=14376 Win=15872 Len=0
  243 26.968636     192.168.8.195   239.255.255.250  SSDP      318 NOTIFY * HTTP/1.1
  300 32.084695     192.168.8.195   239.255.255.250  SSDP      318 NOTIFY * HTTP/1.1
  313 34.775848     192.168.8.195   192.168.8.34     TLSv1.2   123 Encrypted Alert
```

```
▷ Frame 132: 283 bytes on wire (2264 bits), 283 bytes captured (2264 bits) on interface 0
▷ Ethernet II, Src: IntelCor_5c:48:83 (34:e6:ad:5c:48:83), Dst: LiteonTe_ca:5f:b4 (18:cf:5e:ca:5f:b4)
▷ Internet Protocol Version 4, Src: 192.168.8.195, Dst: 192.168.8.34
◢ Transmission Control Protocol, Src Port: 51028, Dst Port: 8440, Seq: 372, Ack: 13907, Len: 229
```

```
0000  18 cf 5e ca 5f b4 34 e6  ad 5c 48 83 08 00 45 00   ··^·_·4· ·\H···E·
0010  01 0d 35 11 40 00 80 06  32 a4 c0 a8 08 c3 c0 a8   ··5·@··· 2·······
0020  08 22 c7 54 20 f8 34 0a  90 76 6e ed 64 b3 50 18   ·"·T ·4· ·vn·d·P·
0030  00 40 8c 39 00 00 17 03  03 00 e0 34 45 c9 2f 04   ·@·9···· ···4E·/·
0040  12 73 a8 5b 45 74 58 73  69 b9 e8 df 00 84 f2 87   ·s·[EtXs i·······
0050  65 8b e9 12 48 f0 5e fd  4f 0a 9c 5d 2e 46 d9 69   e···H·^· O··].F·i
0060  45 60 14 38 a3 f6 23 7c  90 b9 9f 11 42 77 27 1b   E`·8··#| ····Bw'·
0070  91 f2 a4 d2 3c cc 57 89  6b 0e 64 46 36 b8 39 bb   ····<·W· k·dF6·9·
0080  84 72 32 75 12 5f c8 9b  e4 2c ad 7c a4 5b c8 78   ·r2u·_·· ·,·|·[·x
0090  fa 95 99 40 ab 03 b2 1b  78 34 da ac 57 2c 19 c7   ···@···· x4··W,··
00a0  ba d1 db 0a 74 e9 5d f1  40 7f dd 26 ad 37 e6 40   ····t·]· @··&·7·@
00b0  35 fb 02 c0 30 59 99 39  d9 9f b1 7e f2 70 8f 5c   5···0Y·9 ···~·p·\
00c0  9e 36 f7 c9 27 1b 5a 3e  cc 1c 93 23 31 0d 1e 58   ·6··'·Z> ···#1··X
00d0  82 ef 1d 16 ee ef d5 ab  7b 94 5f fd a8 16 08 8d   ········ {·_·····
00e0  9c 3c 54 09 e6 6d 97 b2  80 cb ce 1e f3 a4 e7 b8   ·<T··m·· ········
00f0  44 df da de 42 a3 19 66  42 e0 a4 08 49 8f 69 17   D···B··f B···I·i·
0100  19 74 c6 93 22 a8 e3 aa  86 0b d2 cb 41 fc 14 c5   ·t··"··· ····A···
0110  c9 0c 68 ab 2c ab a0 81  9e 88 4d                  ··h·,··· ··M
```

Figure 32: Encrypted Data using sidHadoop

6.2.2 CS-2 Integrity of data/job

The integrity of data entails that the data is not tampered intentionally or un-intentionally over the channel. To achieve the ***CS-2*** objective Message Authentication Code is also attached with the data. The MAC contains the hash of the message that is to be sent and is concatenated with the original so that the message can be verified by taking the hash of the message and comparing it with the attached hash code. This provides protection from job tempering over the channel. If the reached message and its hash code does not match, then the data has lost it integrity and is discarded.

6.2.3 CS-3 Avoid replay attacks

The SSL authentication and encrypted channel secures the application from ***<u>job tempering</u>***, ***unauthorized access***, ***<u>replay</u>*** and ***<u>MITM</u>*** attacks.

Apart from this, to avoid replay attacks, our application uses Cross-Site Request Forgery CSRF tokens. These tokens are uniquely created for each session and any request containing data also contain a csrf token. The token is provided to the server which verifies the token for the current session. For the new session, a new token is created so no old request will be catered and ***CS-3*** objective is complied.

6.2.4 CS-4 Avoid MITM attacks

The protection from MITM attacks is provided by ssl authentication. In figure 30, where mutual authentication between servers is taking place, the certificates are shared and these certificates are verified before each request. The certificate and a challenge response based key exchange between these servers can only be made possible by having the private key of the certificate. The challenge response mechanism for key-exchange ensures that the traffic is coming from a verified source providing compliance with ***CS-4*** objective.

6.3 Efficiency:

In terms of efficiency, this solution was tested in two scenarios, (i) how much time did a request took when it was ran on a local cluster and (ii) the same when it was run on a remote cluster. The results for these tests revealed that the efficiency is effected because of the remote nature of the other cluster and the request took time to reach the remote domain. The extra time taken by the request is only the time it took on the wire. The following table shows the start time and end time of the request from the requesting server to the accepting server and back.

Table 5: Performance Tests

Model	Take	Cluster	Start	End	Difference
Inter-Domain Hadoop	1	Local	15:20:34.502	15:20:37.633	~ 3 sec
	1	Remote	15:22:19.994	15:22:29.157	~ 9 sec
	2	Local	15:24:27.055	15:24:30.904	~ 3 sec
	2	Remote	15:26:25.265	15:26:28.781	~ 9 sec
Secure Inter-Domain Hadoop	1	Local	15:36:42.138	15:36:45.958	~ 4 sec
	1	Remote	15:38:27.065	15:38:37.916	~ 11 sec
	2	Local	15:40:33.247	15:40:37.950	~ 4 sec
	2	Remote	15:40:27.156	15:40:37.824	~ 11 sec

We ran the request to fetch meta-data on the inter-domain hadoop and secure inter-domain hadoop to find out the performance overhead caused by the security mechanisms implemented. The results shown in table 5 show that a request to local cluster from inter-domain hadoop takes approximate 3 seconds and the same request takes approximate 4 seconds in the secure inter-domain hadoop environment. Similarly, the request to remote cluster took approximate 9 seconds on the inter-domain hadoop and the same request took 11 seconds on the secure inter-domain hadoop. The time taken by requests on the implemented security solution is comparable to the one without security mechanisms. The secure inter-domain hadoop only gives an overhead of 1 second for local cluster requests and overhead of two seconds for remote cluster requests. The overhead caused is very minute and to achieve security, minimal performance overhead is acceptable.

Local Cluster performance overhead =	~ 1 sec
Remote Cluster performance overhead =	~ 2 sec

6.4 Key Feature in sidHadoop:

Now we will discuss the proposed solution in the light of the feature list in the related work section 2.7. The feature ***minimal changes*** in core Hadoop architecture is fully complied and this solution acts as an add-on for enhancing the

Hadoop capabilities. The hadoop instances that were only confined to in-house deployment are now capable of performing communication with other hadoop intances.

Geo-distributed Hadoop resources, is the main feature that this solution provides. In this solution, geo-distributed resources are turned into independent Hadoop instances. The Hadoop instances communicate with each other over WAN.

Minimal traversal of data is achieved by not moving the data from remote resource to the local storage. The independent Hadoop instances are responsible for their own data and the remote cluster can request to perform operations on that data in the form of MR job. This way the huge volume of raw data is not transferred over the communication medium and only the request and subsequent refined results are returned to the requesting cluster. It will help reduce the traffic and thus improve the overall performance.

The ***security features*** discussed in the feature list are Transaction Security, single-sign on and protection from attacks. ***Transaction Security*** is achieved by using an encrypted channel and 2-way authentication mechanism. As discussed above, the following table lists the features that are provided by sidHadoop.

Table 6: Key Features in sidHadoop

Features	**CD-Map Reduce Execution** [18]	**HOG** [19]	**CD-Map Reduce Frame-work** [38]	**G-Hadoop** [21]	**Security Frame-work in G-Hadoop** [16]	**PigOut** [20]	**sid-Hadoop solution**
Minimal Changes in core Hadoop			✓			✓	✓
Geo-Distributed Hadoop resources	✓	✓		✓			✓
Minimal Data Traversal			✓			✓	✓
Stable and Centralized Namenode	✓	✓		✓			
Site Awareness		✓		✓			
Use of Specialized Hardware	✓	✓		✓			
Use of Specialized Network		✓		✓			
Multi-Layer Security MLS			✓				
Transmission security					✓		✓
Single Sign-On					✓		✓
Protection from attacks			✓		✓		✓

Chapter 7
Conclusion and Future Work

7.1 Conclusion:

Hadoop is the big data analysis engine which is deployed by many large mainstream organizations spread over different industries including High-Tech, Government, Healthcare, Academia, Retail, Financial Services and Manufacturing. Organizations utilize Hadoop according to their business requirements in different capacities. Hadoop technology has been matured over the years and there are different flavors of Hadoop available in the market. Cloudera, Hortonworks, IBM and Pivotal are the few big names of companies which offer customized and upgraded enterprise distributions of Hadoop [58].

Hadoop became popular because of its distributed storage file system and MapReduce which provides extreme parallel processing features over distributed storage. It provided a good solution that could handle the volume of big data in its storage and perform operations on this huge volume of data.

In the beginning, Hadoop lacked simple security features for protection of its stored data. Security solutions were added to Hadoop distributions later which made it ready for the production environment. Kerberos [8], Apache Ranger [10], Apache Sentry [11] and many others provide security architecture for Hadoop. Hadoop is supported by a very large community and a wide amount of development is open-sourced for the better of the community. The present Hadoop has evolved to very high standards of integration with other technologies.

Hadoop security architecture has matured a lot and a lot of work has been done to increase the security of Hadoop. The security solutions in Hadoop are designed in such a way that they comfortably allow addition of new components and integration of security to that component too.

Hadoop is generally used in a private environment, but its use-cases expand to many levels. Many such use-cases emerged where petabytes of data were stored in a geo-distributed cluster environment and complex computations were required to be performed on this data. Moving the data to a pre-built Hadoop cluster was not a solution. So solutions were required to expand the reach of Hadoop to geo-distributed environment. G-Hadoop [21] and HOG [19] are the solutions which proposed the deployment of Hadoop in a geo-distributed environment. G-Hadoop and HOG offered a stable solution to join Hadoop over WAN but they required specialized central server for the role of name node. The stable server could manage the data nodes that were spread across the globe, over a high bandwidth internet connection.

The requirement of specialized environment pushed us towards achieving the same by using multiple Hadoop instances. In this solution, the reach of Hadoop is being expanded over WAN but, not by deploying a single Hadoop instance with centralized name node to handle the whole cluster. Instead, this solution proposed to build multiple Hadoop clusters on different locations where the data is situated, or using already existing Hadoop instances to communicate with each other.

In Secure Inter-domain Hadoop, multiple Hadoop instances in different administrative domains will be able to interact and collaborate with each other. The inter-domain Hadoop is the proposed concept which cannot exist without implementing security for protection of the Hadoop resources. The security implementation for sidHadoop ensures that the web service is not accessed for unauthorized usage. Only the Hadoop clusters which have shared their sidHadoop credentials with each other have visibility of each other servers.

The sidHadoop solution fulfills the security requirements discussed in the beginning of this research. The sidHadoop servers authenticate each other at the start of every request first, to make sure that the authorized service is only allowed to access the Hadoop cluster. The channel, in use, is encrypted which secures the communication from channel sniffers. According to the features of the SSL protocol, SSL has the ability to avoid the man-in-the-middle (MITM) attack, version roll back attack, delay attack and replay attack.

The jobs and results are all communicaed over the encrypted channel which provides us the feature of confidentiality. It also provides security from other attacks like replay and MITM attacks. sidHadoop provides a solution for securely sharing resources of Hadoop clusters over insecure public network. Different Hadoop instances will be able to communicate with each other, once sidHadoop is correctly deployed with the Hadoop clusters. It provides a management UI for handling multiple Hadoop clusters spread over different geographical locations.

7.2 Future Work:

Although security issues are dealt with in sidHadoop, but the scope of this solution was only limited to transfer of jar and transfer of results back. Currently, sidHadoop only allows primitive MR job execution over already existing data between multiple clusters, but other tools like Hive, Pig, Storm associated with Hadoop are not yet supported by this solution. Support for other components of Hadoop can be included in the future work to be performed in sidHadoop. Finally, the results are returned from remote cluster and stored in the local server, whereas the results from local cluster are also stored in the local server. In the next version of sidHadoop, it will also provide the final reduce function through which results from multiple clusters will be compiled to give one final results file.

Bibliography

[1] R. L. Villars and C. W. Olofson, "White Paper Big Data : What It Is and Why You Should Care Information Everywhere , But Where's The Knowledge ?," 2014.

[2] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," *Proc. Ninet. ACM Symp. Oper. Syst. Princ. - SOSP '03*, p. 29, 2003.

[3] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Proc. 6th Symp. Oper. Syst. Des. Implement.*, pp. 137–149, 2004.

[4] T. White, *Hadoop: The definitive guide*, vol. 54. 2015.

[5] Apache Hadoop, "Hadoop Wiki." [Online]. Available: https://wiki.apache.org/hadoop. [Accessed: 06-Jul-2018].

[6] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop distributed file system," *2010 IEEE 26th Symp. Mass Storage Syst. Technol. MSST2010*, pp. 1–10, 2010.

[7] D. Borthakur, "The hadoop distributed file system: Architecture and design, Hadoop Project Website," pp. 1–14, 2007.

[8] O. O. Malley, "Integrating Kerberos into Apache Hadoop," 2010.

[9] S. Narayanan, *Securing Hadoop*. 2014.

[10] Apache Ranger, "Apache Ranger – Introduction." [Online]. Available: http://ranger.apache.org/. [Accessed: 06-Jul-2018].

[11] Apache Sentry, "Sentry Tutorial - Apache Sentry - Apache Software Foundation." [Online]. Available: https://cwiki.apache.org/confluence/display/SENTRY/Sentry+Tutorial. [Accessed: 06-Jul-2018].

[12] S. Priya and C. Navdeti, "Securing Big Data Hadoop : A Review of Security Issues , Threats and Solution," *Int. J. Comput. Sci. Inf. Technol.*, vol. 5, no. 2, p. 1, 2015.

[13] B. Spivey and J. Echeverria, *Hadoop Security*. 2015.

[14] Apache Hadoop, "PoweredBy - Hadoop Wiki." [Online]. Available: https://wiki.apache.org/hadoop/PoweredBy. [Accessed: 06-Jul-2018].

[15] B. Sheppard, "Get started with Hadoop - O'Reilly Media," 2011. [Online]. Available: https://www.oreilly.com/ideas/getting-started-with-hadoop. [Accessed: 08-Jul-2018].

[16] J. Zhao *et al.*, "A security framework in G-Hadoop for big data computing across distributed Cloud data centres," *J. Comput. Syst. Sci.*, vol. 80, no. 5, pp. 994–1007, 2014.

[17] K. Yang and X. Jia, "Security for Cloud Storage Systems," vol. 5, no. 6, pp. 7359–7362, 2014.

[18] Y. Luo, Z. Guo, Y. Sun, B. Plale, J. Qiu, and W. W. Li, "A hierarchical framework for cross-domain MapReduce execution," *Proc. Second Int.*

Work. Emerg. Comput. methods life Sci. - ECMLS '11, pp. 15–22, 2011.

[19] C. He, D. Weitzel, D. Swanson, and Y. Lu, "HOG: Distributed hadoop MapReduce on the grid," *Proc. - 2012 SC Companion High Perform. Comput. Netw. Storage Anal. SCC 2012*, pp. 1276–1283, 2012.

[20] K. Jeon, S. Chandrashekhara, F. Shen, S. Mehra, O. Kennedy, and S. Y. Ko, "PigOut: Making multiple Hadoop clusters work together," *Proc. - 2014 IEEE Int. Conf. Big Data, IEEE Big Data 2014*, pp. 100–109, 2015.

[21] L. Wang *et al.*, "G-Hadoop: MapReduce across distributed data centers for data-intensive computing," *Futur. Gener. Comput. Syst.*, vol. 29, no. 3, pp. 739–750, 2013.

[22] R. K. G. Michael, "A Research on Secure Shell (SSH) Protocol," vol. 116, no. 16, pp. 559–564, 2017.

[23] S. Li, T. Zhang, J. Gao, and Y. Park, "A Sticky Policy Framework for Big Data Security," *2015 IEEE First Int. Conf. Big Data Comput. Serv. Appl.*, no. March, pp. 130–137, 2015.

[24] N. Miloslavskaya, M. Senatorov, A. Tolstoy, and S. Zapechnikov, "Big Data Information Security Maintenance," 2014.

[25] O. O'Malley, K. Zhang, and S. Radia, "Hadoop security design," *Yahoo, Inc., Tech. Rep*, no. October, pp. 1–19, 2009.

[26] Apache Hadoop, "Transparent Encryption in HDFS." [Online]. Available: https://hadoop.apache.org/docs/r2.7.0/hadoop-project-dist/hadoop-hdfs/TransparentEncryption.html. [Accessed: 08-Jul-2018].

[27] P. Adluru, S. S. Datla, and X. Zhang, "Hadoop Eco System for Big Data Security and Privacy," 2015.

[28] C. Swarna and Z. Ansari, "Apache Pig - A Data Flow Framework Based on Hadoop Map Reduce," vol. 50, no. 5, pp. 271–275, 2017.

[29] C. Olston, B. Reed, R. Kumar, and A. Tomkins, "Pig Latin : A Not-So-Foreign Language for Data Processing."

[30] A. Thusoo *et al.*, "Hive – A Petabyte Scale Data Warehouse Using Hadoop."

[31] S. S. Aravinth, "An Efficient HADOOP Frameworks SQOOP and Ambari for Big Data Processing," vol. 1, no. 10, pp. 252–255, 2015.

[32] OSG, "The Open Science Grid Project," *J. Phys. Conf. Ser. J. Phys. Conf. Ser*, pp. 1–4, 2007.

[33] O. Tatebe and K. Hiraga, "Gfarm Grid File System," vol. 28, pp. 257–275, 2010.

[34] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. Ullah Khan, "The rise of 'big data' on cloud computing: Review and open research issues," *Inf. Syst.*, vol. 47, pp. 98–115, 2015.

[35] Garrett M. Morris *et al.*, "AutoDock — AutoDock." [Online]. Available: http://autodock.scripps.edu/. [Accessed: 20-Jul-2018].

[36] G. Von Laszewski *et al.*, "Design of the Futuregrid experiment management framework," *2010 Gatew. Comput. Environ. Work. GCE*

2010, 2010.

[37] P. H. Beckman, “Building the TeraGrid,” *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.*, vol. 363, no. 1833, pp. 1715–1728, 2005.

[38] T. D. Nguyen, M. A. Gondree, J. Khosalim, and C. E. Irvine, “Towards a cross-domain MapReduce framework,” *Proc. - IEEE Mil. Commun. Conf. MILCOM*, pp. 1436–1441, 2013.

[39] B. Hicks, S. Rueda, T. Jaeger, and P. Mcdaniel, “Integrating SELinux with Security-typed Languages.”

[40] C. Hanson, “SELinux and MLS : Putting the Pieces Together.”

[41] A. C. Sekhar and R. P. Sam, “Grid Theory And Grid Security Infrastructure (GSI),” pp. 184–187, 2015.

[42] K. Piirainen, R. A. Gonzalez, and G. Kolfschoten, “Quo Vadis, Design Science? - A Survey of Literature,” *Glob. Perspect. Des. Sci. Res.*, vol. 6105, pp. 93–108, 2010.

[43] S. T. March and G. F. Smith, “Design and natural science research on information technology,” *Decis. Support Syst.*, vol. 15, no. 4, pp. 251–266, 1995.

[44] P. J. Denning, “A new social contract for research,” *Commun. ACM*, vol. 40, no. 2, pp. 132–134, 1997.

[45] J. Ernst Van Aken, “Management research based on the paradigm of the design sciences: The quest for field-tested and grounded technological Rules Design science (as research approach) in entrepreneurship and innovation mngt View project,” *Artic. J. Manag. Stud.*, no. December, 2001.

[46] H. Takeda, P. Veerkamp, T. Tomiyama, and H. Yoshikawa, “Modeling Design Processes,” *AI Mag.*, vol. 11, no. 4, pp. 37–48, 1990.

[47] V. Kuechler, B. Vaishnavi, “Design science research in information systems.,” *URI http//www. desrist. org/design-research-in-information-Syst.*, 2004.

[48] R. Khanna and P. Bhalla, “Study of spring framework,” no. 10, pp. 419–423, 2014.

[49] M. Keen, S. Baber, and H. Cui, “Developing Enterprise JavaBeans Applications,” 2012.

[50] Pivotal Software, “Web MVC framework.” [Online]. Available: https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html. [Accessed: 16-Jul-2018].

[51] Pivotal Software, “Spring Security Reference.” [Online]. Available: https://docs.spring.io/spring-security/site/docs/current/reference/htmlsingle/. [Accessed: 16-Jul-2018].

[52] J. Franks *et al.*, “HTTP Authentication: Basic and Digest Access Authentication,” Jun. 1999.

[53] D. Ahrens and S. Bremer, “HTTP Digest Access Authentication,” Sep. 2015.

[54] S. Sejwani and S. Tanwar, "Implementation of X . 509 Certificate for Online Applications," vol. 2, no. 3, pp. 250–254, 2014.

[55] H. Johner, L. Brown, F. Hinner, W. Reis, and J. Westman, "Understanding LDAP," *Contract*, p. 177, 1998.

[56] E. Tsyrklevich and V. Tsyrklevich, "Single Sign-On for the Internet: A Security Story," *BlackHat USA, Las Vegas*, p. 11, 2007.

[57] Pivotal Software, "Cross Site Request Forgery (CSRF)." [Online]. Available: https://docs.spring.io/spring-security/site/docs/current/reference/html/csrf.html. [Accessed: 16-Jul-2018].

[58] Brandon Butler, "The top 5 Hadoop distributions, according to Forrester | Network World." [Online]. Available: https://www.networkworld.com/article/3024812/big-data-business-intelligence/the-top-5-hadoop-distributions-according-to-forrester.html. [Accessed: 21-Jul-2018].