EECS 4314: Hadoop Enhancement Report

Mina Zaki
Alexander Aolaritei
Dillon Bondarenko
Camillo John (CJ) D'Alimonte
Jeremi Boston
Daniel Nowak

December 5, 2016

Contents

1 Abstract

In its current stae, multi-node cluster setup in Hadoop is very tedious and takes too long. For first time users, this experience is especially frustrating. To fix this we have proposed two different solutions. The first solutions involves adding a Graphical User Interface to assist first time users with setup. This can reduce the learning time required to get started with using Hadoop. The second solution involves speeds up the process by removing repetitive inputs. This could decrease the time required to perform this tedious task. Due to its ease of use for first time users, we have recommended the first solution.

2 Introduction and Overview

The purpose of this report is to outline our solutions to improving multi-node cluster setup in Hadoop. First we will outline the problem that we will solve with our two solutions. Next we will cover the first solution which is the GUI for multi-node cluster creation. Furthermore, we will investigate the effects of this implementation on the overall architecture of the system. In addition to this, we will weigh the advantages and disadvantages of adding solving the problem with a GUI. Next, we will discuss our second solution which involves using cluster histories for speeding up the process of multi-node cluster creation. We will weight the advantages and disadvantages of this solution and will discuss the potential impact on software architecture. Next, we will formally address the changes to the Software Architecture that our solutions will create. Following the architecture changes, we will discuss the risks and limitations of our solutions on the overall system and how we would test these impacts. Next, we will further explain these two solutions through the use of state and sequence diagrams. This will be followed by discussing concurrency and team issues. Finally we will conclude and dicuss the lessons learned through this exercise.

3 Values and Benefits from the Enhancement

It can be very time consuming setting up a Multi-Node cluster. To setup a Hadoop Multi-Node cluster, Java needs to be installed, PATH and JAVA_HOME variables need to be set, user accounts need to be created, nodes need to be mapped, key based login needs to be configured, Hadoop needs to be installed and configured, Hadoop needs to be installed on the slave servers and configured on the master server, Hadoop services need to be started, a DataNode needs to be added to the Hadoop cluster, users need to be added with SSH access, Hostname of the new node needs to be set, the DataNode on the new node needs to start, and finally a DataNode from the cluster may need to be removed. All of these configurations are primarily done through bash commands and editing configuration files. It can be very confusing for a user to setup a cluster and thus would be easier to have it done through a user interface or instead having a method in which previous cluster setup histories are saved in a file and could then be reused when creating a new cluster. For instance, editing the wrong line in a configuration file during the cluster setup phase can produce errors that the user wouldn't be aware of right away. Having a user interface that would edit that configuration file for the user based off of a given input would minimize this type of error. Another benefit of this enhancement is that it would reduce setup time. The user would be allowed to quickly navigate through each phase of the setup process or simply copy over a past cluster setup history file when creating a new cluster. Lastly, it would reduce user confusion. New Hadoop developers would be able to simply navigate through a user friendly interface with prompts that alert the user of what setup phase is mandatory and if there are any problems.

4 Stakeholders

Stakeholders	Non-functional requirements and priorities
Users	Ease of use, Performance, Reliability
Developers	Maintainability, Scalability, Bug Free
Customers	Reliability, Improves previous results

Table 1: Stakeholders description

In this system there are three main stakeholders. The users, developers and customers. The users will benefit since this will increase the ease of use of the system. The developers will need to ensure the system is bug free. The customers will benefit since it will decrease the time taken to get results.

5 Approach 1 User Interface Breakdown

5.1 GUI Concept

The first proposed enhancement for Hadoop is a method of easing cluster creation. The reason for such an enhancement is due to the many steps required to manually set up a Hadoop cluster. As an example, currently one must first create an account for the master node and all the slave nodes, map the nodes by editing host files and XML files, set up ssh for every node so that they can communicate with each other freely without the use of passwords, and etc. There is a long list of tasks which must be performed to set up a Hadoop Cluster, requiring some sort of knowledge of using a command line, and the syntax and format of certain files. Therefore, this enhancement would greatly improve the process for a large number of users including those which may not be able to perform some of the manual steps. Not only will a user interface increase the user-base, but it will also streamline the process of cluster creation allowing for fast and proper installation. The suggested interface should go through the process with the user step-by-step providing simple "fill in the blank" requests. A basic idea of the user interface may look something like what Figure 1 proposes. The first frame presents the user with 4 text boxes all clearly labeled explaining what is needed to progress. Here the User will input the Cluster name, and create a Master node with a user name and password for added security, as well as the IP address to setup the node.

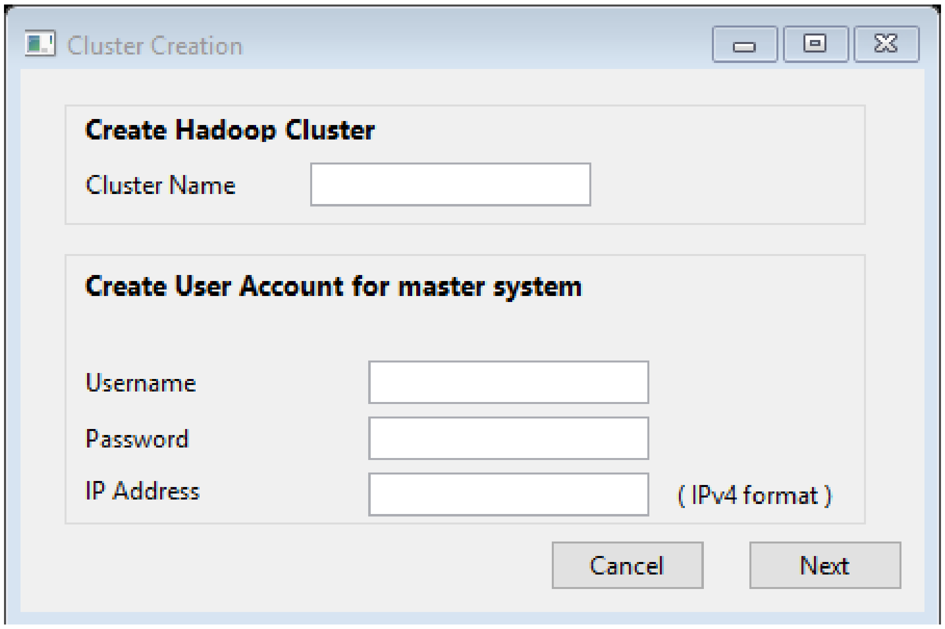
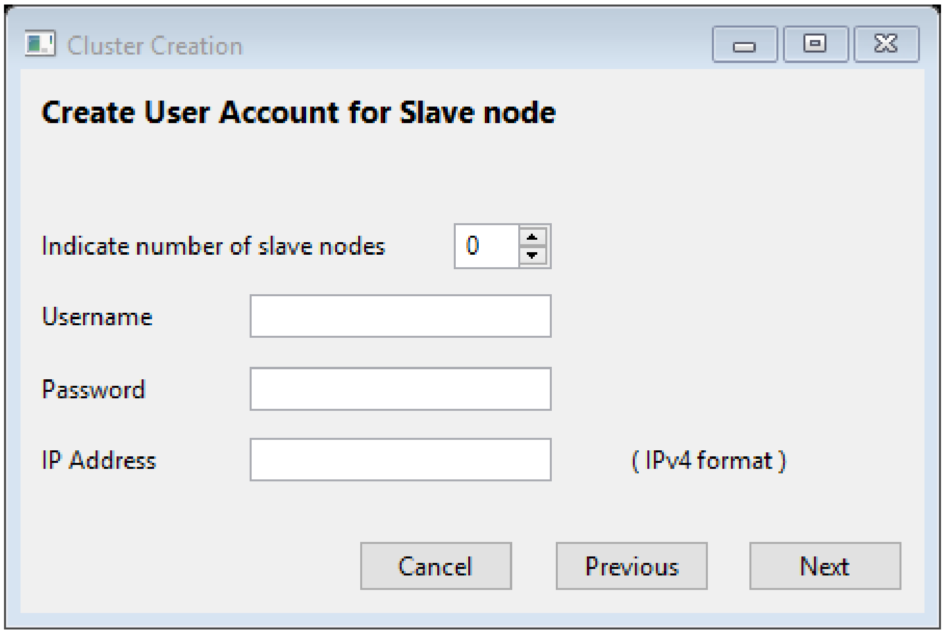


Figure 1: Possible representation of a Hadoop Cluster Creation User Interface

The next frame in Figure 2 shows a similar step as with the first prompting the user to first indicate the amount of slave nodes which will be require, and second the username, password, and IP address. Depending on the indicated amount the GUI will most likely present the user with a similar frame indicating the slave number, with the username password and IP address again, but excluding the desired number of slaves prompt.


Cluster Creation

Create User Account for Slave node

Indicate number of slave nodes 0

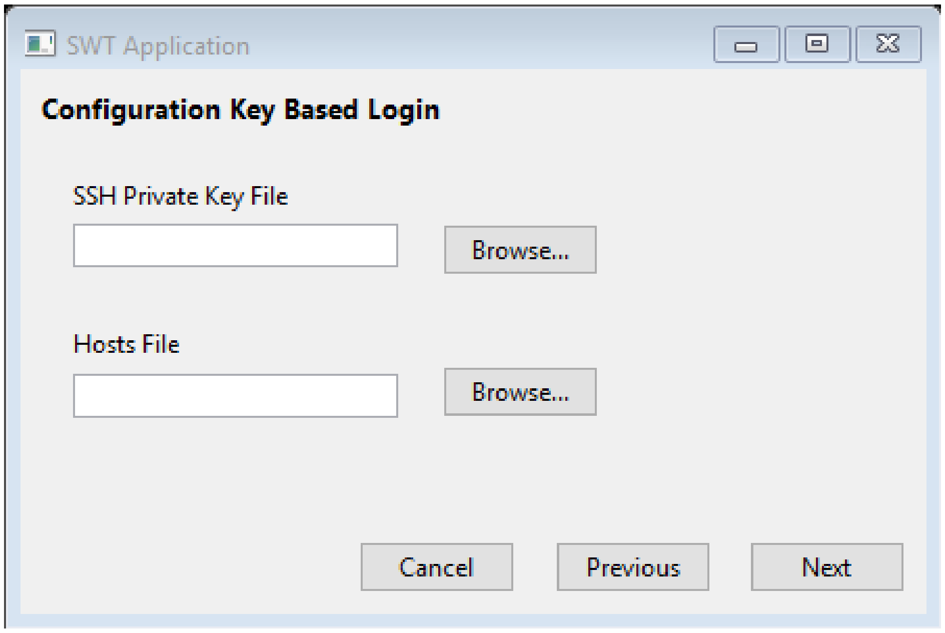
Username

Password

IP Address (IPv4 format)

Figure 2: GUI frame used to set up slave node servers

Moving on to Figure 3 showing two text boxes and browse buttons prompting for SSH Private Key File along with the Hosts File. Rather than the user having to modify and edit files or type in commands, this can be handled through the interface easing the process. The Browse button would pull up a file browser interface where the user may input the required files so that the program can properly modify them.


SWT Application

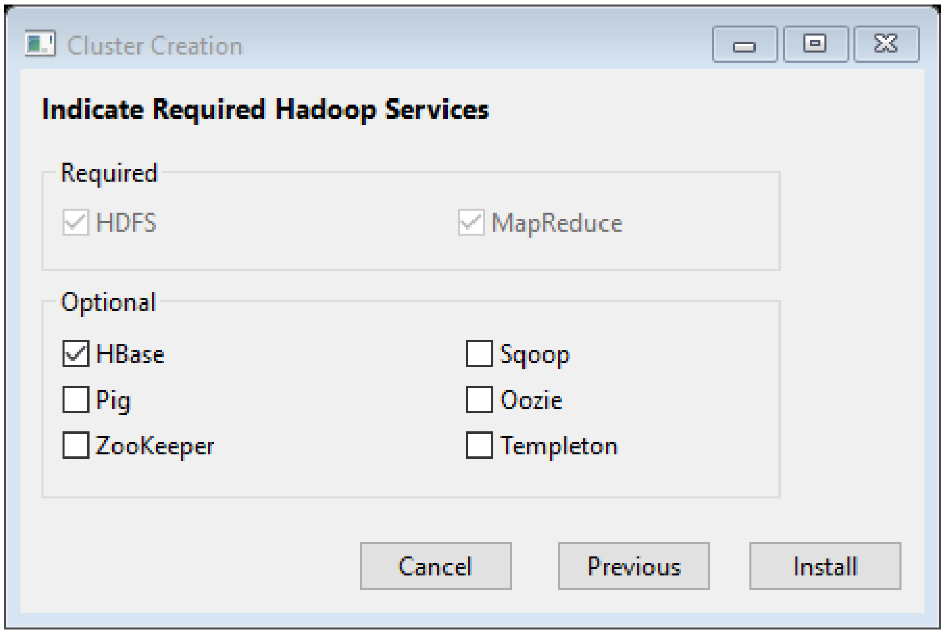
Configuration Key Based Login

SSH Private Key File:

Hosts File:

Figure 3: GUI frame used to modify hosts file and SSH Private Key file for the user

The next frame in Figure 4 presents the user with added services which may also be installed on the servers and the client PC to provide an overall better experience working with Hadoop.


Cluster Creation

Indicate Required Hadoop Services

Required
- [✓] HDFS
- [✓] MapReduce

Optional
- [✓] HBase
- [ ] Sqoop
- [ ] Pig
- [ ] Oozie
- [ ] ZooKeeper
- [ ] Templeton

Figure 4: GUI frame prompting user to select desired additional services

In order to make sure Hadoop is properly installed, the Hadoop Distributed File System and MapReduce are required to be installed and cannot be unchecked. This safeguard is to prevent an uninformed user from not installing one of the main components of Hadoop. The remaining services are completely optional and may be selecting by click the checkbox. Some of these services include HBase which is a non-relational database, Pig a platform to analyze large data sets, ZooKeeper used for configuration management and distribution synchronization, and etc.

5.2 Effects of Implementation

Since the proposed enhancement would be a simple add-on with a GUI subsystem most likely using "java.swing" there would not be much of an effect on the overall architecture of the system with the exception of an added subsystem. There may be some dependencies created with the distributed file system or MapReduce when trying install the system for the user, however with proper installation there is minimal risks to the integrity of the system. The design pattern used to implement the GUI will require the use of the "Model View Controller" design pattern. This means that the GUI will be separated into three components where the View contains all the code required to implement the user interface, the Model holds all the logic and data require to fulfill tasks, while the Controller takes input from the user and allows communication between the Model and the View. The GUI would most likely need to be maintained throughout it's lifetime applying updates along with the newest version of Hadoop. These updates could be something as simple as improving user comprehension of certain steps by adding tags, or larger tasks for example providing greater customizations or more services. A good way to check performance would be to try a user test group with average computer users to test if it is possible for the user to create a Hadoop cluster. If sufficient amount of users are unable, a questionnaire may be provided to indicate what is lacking and not understood in the interface to improve the system.

5.3 Advantages and Disadvantages

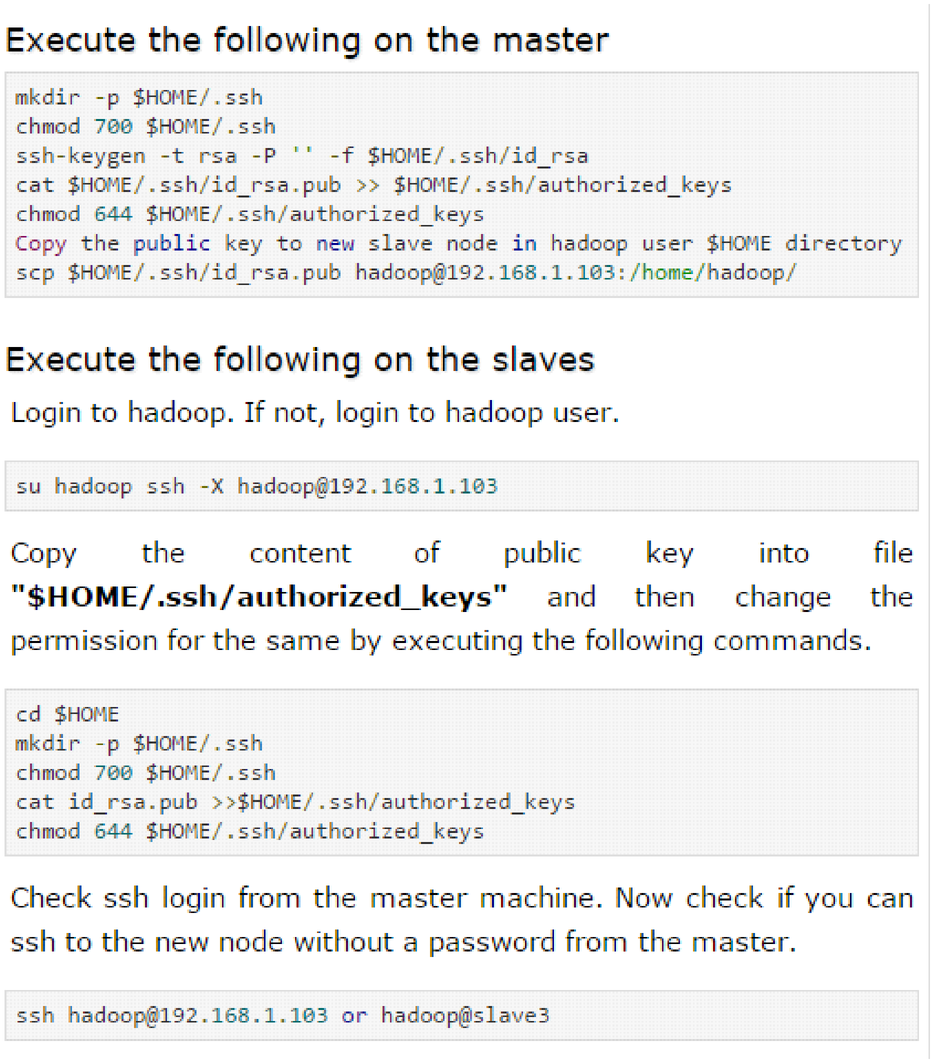
The main advantage of such an enhancement would be user friendly aspect of the system. Allowing the average computer user to be able to install a Hadoop cluster and in turn use the system for their own uses would greatly increase the user base. This would allow many small businesses to gather data and use it to better manage their own business. Since Hadoop is already geared towards perhaps smaller businesses without the resources to have massive servers to process data this could be a very useful tool. There is no doubt that some users may be uninterested in trying Hadoop due to the very time consuming and tedious set-up which is very inconvenient. However, as with all enhancements there are some disadvantages as well, for example the customizability and maintainability. For the experienced Hadoop user, the interface may not present all possible implementations or services required. In such a case the experience user would need to implement these changes themselves or suggest the addition to the interface in a newer version. This presents the problem of maintainability where the tool needs to be upgraded to keep up with the new versions and 3rd party services provided for Hadoop.

6 Approach 2 Cluster History Breakdown

A typical process for creating clusters in Hadoop is to do it manually through the use of shell commands on each of the slave nodes. The client must specify many parameters individually through the basic command line interface. This can include processes such as linking to the master, grabbing HostNames, setting up account priveledges, and writing to the DataNode. When deploying many nodes at the same time, this process can get tedious through the initialization process of each and every slave required. This approach allows users to deploy clusters that have similar attributes at the same time with the ease of a script command on the master.

6.1 Typical Slave Commands

A tutorial for creating clusters is given to the user which addresses all of the necessary steps to install Hadoop and configure it on the master server and slave nodes. The steps mainly go over installing Java, the Hadoop source files, and setting up an account for your server. However, after that there are many commands that must be done to each of the slave nodes for configuration. Figure 5 shows an example of the commands that must be run on slaves.



Execute the following on the master

```
mkdir -p $HOME/.ssh
chmod 700 $HOME/.ssh
ssh-keygen -t rsa -P '' -f $HOME/.ssh/id_rsa
cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
chmod 644 $HOME/.ssh/authorized_keys
Copy the public key to new slave node in hadoop user $HOME directory
scp $HOME/.ssh/id_rsa.pub hadoop@192.168.1.103:/home/hadoop/
```

Execute the following on the slaves

Login to hadoop. If not, login to hadoop user.

```
su hadoop ssh -X hadoop@192.168.1.103
```

Copy the content of public key into file **"$HOME/.ssh/authorized_keys"** and then change the permission for the same by executing the following commands.

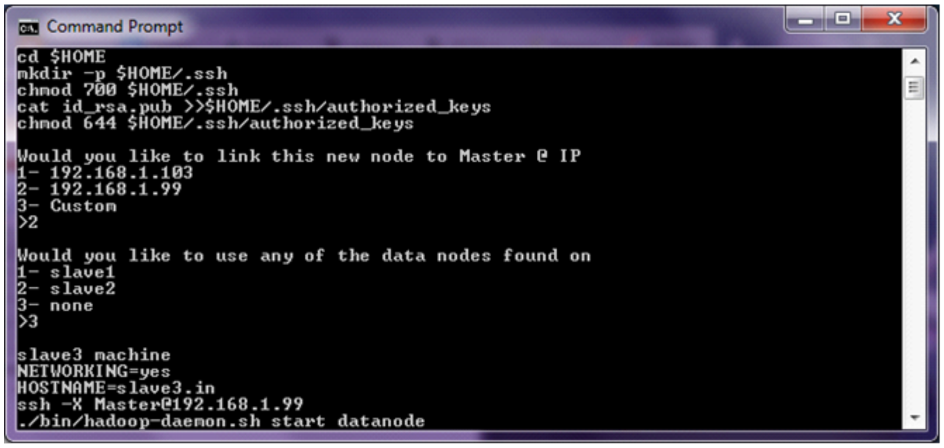
```
cd $HOME
mkdir -p $HOME/.ssh
chmod 700 $HOME/.ssh
cat id_rsa.pub >>$HOME/.ssh/authorized_keys
chmod 644 $HOME/.ssh/authorized_keys
```

Check ssh login from the master machine. Now check if you can ssh to the new node without a password from the master.

```
ssh hadoop@192.168.1.103 or hadoop@slave3
```

Figure 5: Various commands that must be run on each slave

In this approach, a history of these commonly used commands could be stored in the master server for use when creating new nodes so users would not have to manually type in these commands for each of their new slaves. Linking slave nodes to master's hosts could be done automatically. Parameters such as IP addresses, account information, app config files (3rd party, or not) could be copied over to new slaves. In Figure 6: a hypothetical approach to creating new nodes is applied.



```
cd $HOME
mkdir -p $HOME/.ssh
chmod 700 $HOME/.ssh
cat id_rsa.pub >>$HOME/.ssh/authorized_keys
chmod 644 $HOME/.ssh/authorized_keys

Would you like to link this new node to Master @ IP
1- 192.168.1.103
2- 192.168.1.99
3- Custom
>2

Would you like to use any of the data nodes found on
1- slave1
2- slave2
3- none
>3

slave3 machine
NETWORKING=yes
HOSTNAME=slave3.in
ssh -X Master@192.168.1.99
./bin/hadoop-daemon.sh start datanode
```

Figure 6: Previously used attributes would be accessible to new slaves

The user would be prompted whether they would like to use previously entered information to ease setup times. Slaves that have similar attributes can be created easily without tedious repetitiveness.

6.2 Pros and Cons

The biggest advantage to this approach is exactly the main reason behind it: clients are no longer setting up passwords, hosts, config files, and such on every slave node manually. This would reduce the risk of typos and provides an easy way to add new slaves even after the initial master server setup. Applications that have been installed on other clusters would be accessible to new slaves, similarities could be created such as "I want a slave that behaves like X, but has some stuff like Y". Sharing history between other users can be seen as both an advantage and a disadvantage. On one hand, being able to share the configurations would allow teams more communication, but the other side to this is that privacy of the information could be at risk. Also note that maybe the client doesn't always want to setup clusters that are similar to each other. Then the client is back at square one where they are manually inputting all these parameters by hand again.

6.3 Potential Software Architecture

A simple addition of a new class to attach to the master system is enough to store parameters of the currently created clusters. Values such as IP addresses, passwords, installed applications and such would be readily accessible to any new cluster created. Pointers would be used to give a path that the new slave can use to read values that were stored in the history on the master, or potentially just another pointer that can be followed further to extract relevant information from older slaves. The Figure 7 shows the hypothetical model that can be implemented.

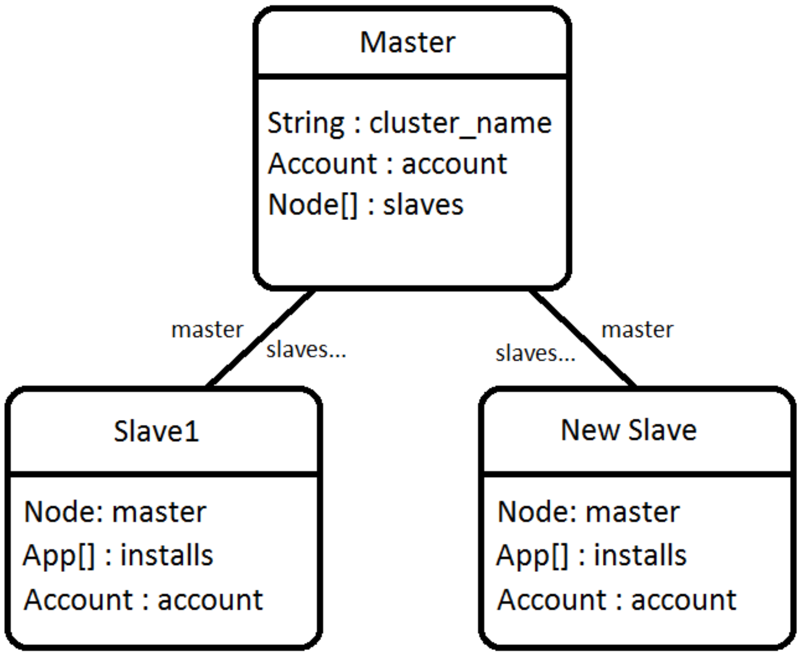


Figure 7: New slaves would be able to follow pointer to the master to extract needed data from old slaves

6.4 Applications Docker and Kubernetes

The applications Docker and Kubernetes can be brought to YARN to allow users to manipulate common resources. It can be seen that the approach described in this section has a lot of similarities to these apps. However, both Docker and Kubernetes manipulate data on clusters that have already been created, whereas the history script approach is only used during initialization of said clusters. Perhaps Docker may be updated in the future to include features such as the ones described in this section.

7 Effects of enhancement

7.1 Effects of User Interface on Architecture

Since the addition to the system is a UI, this will not majorly affect the architecture of Hadoop. This is merely an overlay which will connect to HDFS to support in node creation. The following diagram displays the architecture change necessary to implement the UI approach on Hadoop. This displays the conceptual architecture, concrete architecture and the architecture following the enhancement on Hadoop.

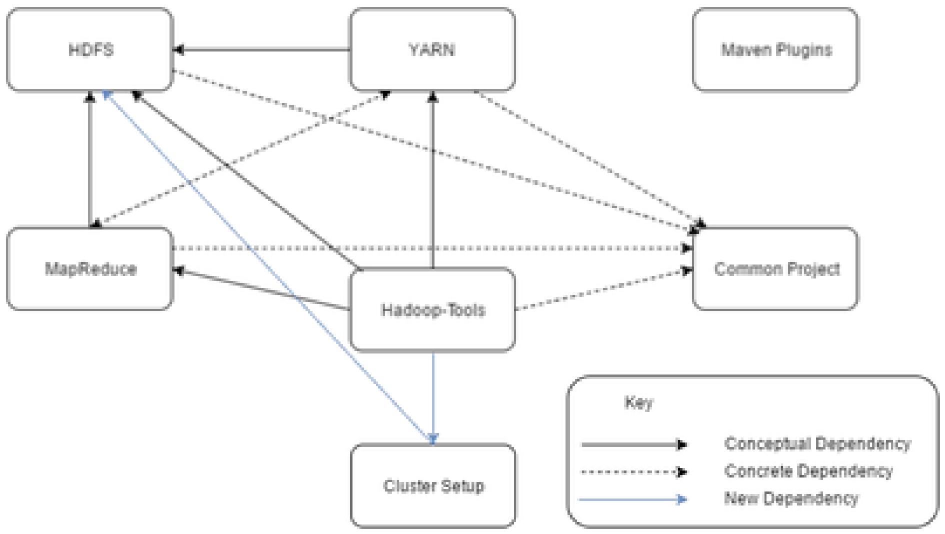


Figure 8: Architecture Diagram contrasting conceptual, concrete and the new architecture

7.2 Effects of Cluster History on Architecture

Since the cluster history must save the previous nodes, HDFS will have to be adjusted in order to add this new feature. Apart from HDFS nothing else must be adjusted and no new subsystems need to be added.

7.3 Effects of Enhancement on system properties

	Effects of Enhancement
Maintainability	decrease
Evolvability	increase
Testability	decrease
Performance of the System	increase

Table 2: Effects of Enhancement on Key System Properties

Maintainability will decrease since this adds another interface which must be correct of defects, or repaired. Evolvability will increase since this adds the capability for User Interfaces to be added to the Hadoop system, instead of only command line interactions. Testability will decrease since finding faults in the user interface will be difficult. This enhancement will not make testing any easier. Performance of the system will increase since this adds a new interface which will make creation of multi-node clusters both easier and less tedious.

7.4 Impacted Directories and files

The main impacted directories and files are related to HDFS. This is the only subystem that interacts with this enhancement. Therefore, only the directories and files associated with HDFS will be impacted.

8 Risks and Limitations

- Although the enhanced feature allows for the simplification of the process in which cluster services are added, the deletion and removal of such clusters remains dependent on command line arguments. The process to remove a cluster is as follows:
  - Identify the different components for the service.

    - Identify the hosts where the components are running.
    - Stop the service.
        * Through the command line, the user will need to locate the hosts where certain components are running in case some component(s) do not stop when instructed to. This process will need to be repeated for each service component, regardless of how many services fail to terminate execution. The GUI, currently, does not allow the user to traverse a list of services; the user must rely on command line arguments.
    - Stop the components of service, if required.
        * Verification as to whether or not the service and its components stop must be completed in the command line using the curl command.
    - Delete the service using DELETE API Call. [1]

- There is a limited ability to customize the graphical user interface and its features for the various Hadoop Distributions used. While a base model can work on all core Hadoop distributions, the clustering setup will vary depending on the distribution type (i.e.: clustering management and installation varies between Cloudera and MapR). Further development of derivative features will increase the effectiveness and efficiency of the feature while greatly improving the user's experience.

- Security concerns are still prevalent with the proposed interface. Even in a secure cluster, parts of the data path from the client to the server are usually writeable, often leading to the unintentional creating of insecure paths. Permissions have to be explicitly asked for as there is no inheritance structure between nodes. Node permissions need to be locked down manually as to allow only authenticated users the ability to read and manipulate secret data written by specific cluster services. [2]

9 Testing the Impact of Interactions

The tool MiniKDC allows for integrated Hadoop Security Unit Testing. The test runner allows for functional testing to be performed against a Hadoop Cluster. Although easy to run, the tests are often slow and there is considerable overhead in the copying of files into HDFS. Retrieving and interpreting results are difficult as it often involves manually going to YARN RM to get through the log files. Despite these concerns, MiniKDC remains the most appropriate choice in testing the security of the proposed system. [3]

10 Diagrams further explaining approaches

10.1 User Interface Approach: State Diagram

The User Interface approach to ease cluster setup begins with launching the UI application. Afterwards, each next state represents a screen within the application that will prompt the user for necessary information within that setup phase. For example, there will be a screen asking the user for information about user account creation and another screen configuring Hadoop. Once all the configurable data is entered, the final step is to save that data within HDFS so that it can be accessed for cluster creation. The state diagram for the User Interface approach can be observed in Figure 9.

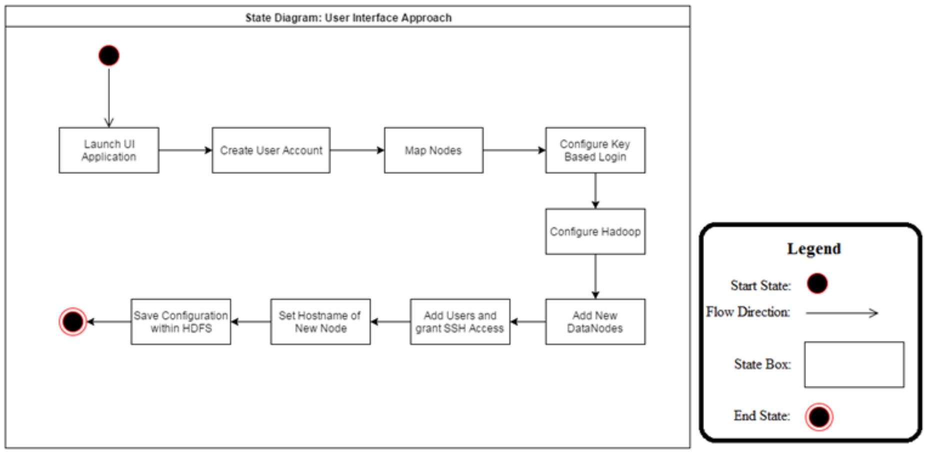


Figure 9: User Interface State Diagram

10.2 Cluster History Application Approach: Sequence Diagram

The history file approach to ease cluster setup begins with creating a Multi-Node cluster using bash commands and editing configurations but at the same time saving these configurations within HDFS. Afterwards, the Cluster History Application will access HDFS when a new cluster needs to be created with similar configurations as an existing cluster. This helps avoid the tedious task of reconfiguring attributes that you have already configured in the past. The last phase for the Cluster History Application is to create the new cluster. The sequence diagram for the Cluster History Application approach can be observed in Figure 10.

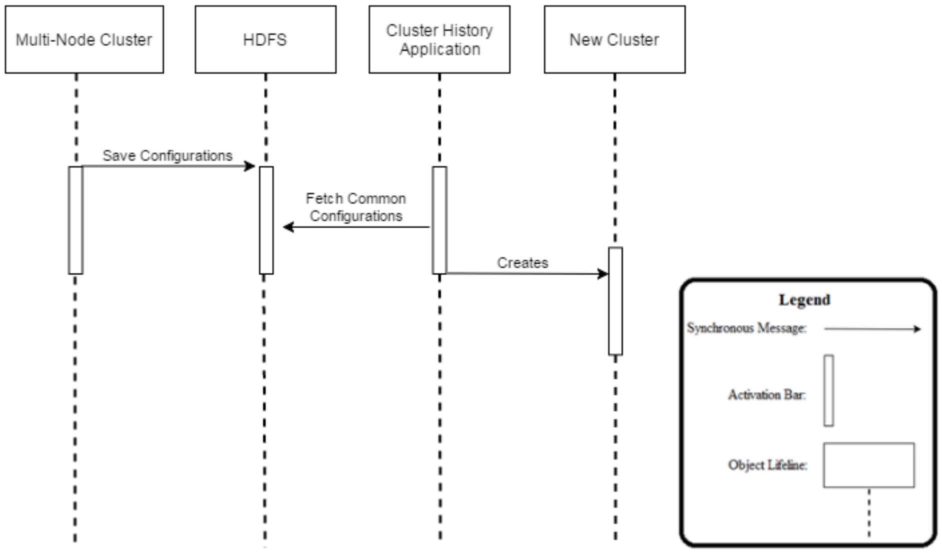


Figure 10: Cluster History Application Sequence Diagram

11 Concurrency and Team Issues

Depending on which approach is adopted different issues may arise. In the case of the first approach, where a UI is implemented, the difficulty barrier for creating a cluster is reduced thus making it easier to delegate cluster creation to a team member. However, there exists the potential for redundancy within this approach. Namely, if multiple users are operating on the UI simultaneously, unaware of the other, they could both create the same cluster. This will cause unnecessary bloat to the system , and result in the need to backtrack so that the issue can be fixed. A possible solution to this issue could be to lock new cluster creation while a user is in the process of doing so. This would stop others from adding to the bloat, and when the UI refreshes they will see that their new cluster has already been created. In the case of the second approach, where a cluster creation history is implemented, the histories of multiple users will be available to others. This aspect can both be harmful and helpful. Harmful in the sense that it adds clutter to user histories, which results in irrelevant histories being available to some user. While being helpful in the sense that it increases interconnectedness between the team. This issue can be solved in multiple ways such as a potential naming convention for the histories or having histories being attached to specific users.

12 Conclusion

In conclusion, there it is very time consuming to set-up a multi-node cluster in Hadoop. This process can be very tedious and is not easy for first time users of the system. One possible solution to this issue is adding a GUI that can assist users with the set-up process. The second solution could be recalling histories for creation. The first solution is more effective since it is able to assist beginners with using Hadoop. There are possible security concerns with this system and a failure to address these concerns would harm Hadoop more than help.

13 Lessons Learned

During the process of producing this assignment multiple instances occurred in which the group acquired noteworthy lessons. The first being that improving on an established product is more of a creative process than it is technical. This lesson originating from the fact that the longest process of this assignment was coming up with an original idea which could successfully build upon Hadoop and improve it. However, through this time researching this idea a more in depth knowledge of cluster creation within Hadoop was gained. Finally, we learnt a very important lesson which will surely apply to our future careers in software development, which is that developing even a small feature is quite a demanding task. Something the group would have done differently we the lessons we have learned from this project would have been to spend more time using Hadoop more extensively so as to figure out what it truly needs.

References

[1] `http://zdatainc.com/2015/10/remove-or-delete-service-from-ambari/`

[2] `https://steveloughran.gitbooks.io/kerberos`$_{a}nd_{h}adoop/content/sections/zookeeper.html$

[3] `https://steveloughran.gitbooks.io/kerberos`$_{a}nd_{h}adoop/content/sections/testing.html$

[4] Tutorial for setting up the cluster,
`https://www.tutorialspoint.com/hadoop/hadoop`$_{m}ulti_{n}ode_{c}luster.htm$

[5] Information on Docker and Kubernetes,
`http://hortonworks.com/blog/docker-kubernetes-apache-hadoop-yarn/`