eBay Distributed Log Processing

A Major Qualifying Project Report

submitted to the Faculty of the

WORCESTER POLYTECHNIC INSITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

By

_____

Patrick Crane

_____

Alyssa Hargraves

Date: March 7, 2009

Approved:

_____

Professor David Finkel, Major Advisor

Abstract

The MQP presented in this document was completed at eBay in San Jose, CA between January and March 2009. eBay Research Labs was using Hadoop to process their log data, and hoped for an easy way to increase their processing speed. To accomplish this, we developed and tested a program to allow users to quickly connect and configure their Windows computers as nodes on eBay's existing Hadoop cluster. Additionally, we created an installer to handle the many prerequisites associated with Hadoop on Windows.

Executive Summary

At the time of our arrival at eBay, eBay Research Labs (eRL) was using the open-source Hadoop distributed computing system to process their daily log entries. The Hadoop cluster ran only on Unix boxes, and developer workstations were left idle at many points throughout the day. eRL wished for a means of taking advantage of those idle workstations by using their processing power in their Hadoop distributed computing cluster.

The goal of our project was to design, develop, and test an easy to use Hadoop node management program to take advantage of idle workstations by allowing users to add and remove their computers from the cluster. This program was required to run on Windows and needed a simple installer to take care of the many prerequisites associated with running Hadoop in a non-Unix environment. Additionally, we were required to test our solution on a cluster setup that is similar to eBay Research Labs' production environment to ensure compatibility with their existing servers.

Over the course of the nine-week program, we created three primary portions: a Hadoop Runner program to deal with the client side portion of node management, a user-friendly installer, and many setup scripts to assist the installer in the setup and configuration process.

The Hadoop Runner program performs the client side actions and is the primary interface between a user and his or her computer's node status. This program was written in Java and provides a graphical user interface to the user. The Hadoop Runner program gives users the option to add or remove their computers from the cluster, lets them specify the amount of hard drive space they are willing to make available to Hadoop, and

shows them basic node status information such as whether or not their node is online and how many jobs it is processing.

The Hadoop Runner program also includes a user-friendly installer created using Nullsoft Scriptable Install System. This installer along with its associated setup scripts sets up the prerequisites of Cygwin, passphraseless secure shell access, and several of the environment variables.

Ultimately, the program we created gives users the freedom to easily add or remove a Windows computer on eBay Research Labs' Hadoop cluster. This provides eBay Research Labs with additional processing power. This processing power can be utilized to perform additional jobs or more complex tasks in a shorter amount of time. Additionally, it opens up the possibility of making Hadoop's log processing power available to other departments within eBay in return for the use of their machines as Hadoop nodes. We feel that this program can be continually modified to handle eBay's growing needs, and hope to see it in use throughout the company.

Table of Contents

1. Introduction

To better compete in an evolving online world, many companies store and analyze log data. This data typically contains user actions and queries, and enables companies to better serve the needs of their clients. eBay stores large anonymous log files on a daily basis, and needs an effective means of analyzing that information. Currently, they use the open source Apache Hadoop project to process their requests, but they wish to improve the speed at which logs are processed.

eBay Research Labs is responsible for running the log storage and processing system, and they were interested in the development of tools for the Hadoop systems. Specifically, eBay wanted the opportunity to run the processing of log files on developer machines to speed up log processing time. This would allow the jobs to complete faster, and as a result, they could free up their servers to perform more jobs in a single day, or more complex job tasks.

To implement this project, we developed both server and client side programs. Additionally, we created installation files to automate the complex setup of Hadoop and its prerequisites. Lastly, we tested the software we created by setting up a small cluster with a Solaris master server and running sample jobs equivalent to those that eBay Research Labs runs on a daily basis.

This paper documents the background information necessary to understand the problem, the tools we used to accomplish our goal, the steps we took to create the program, the testing we performed, and details of the program that resulted.

2. Background

2.1. eBay, Inc.

eBay has been in operation since 1995, and today is the world's largest online marketplace [1]. The company began when its founder, Pierre Omidyar, created his AuctionWeb site and sold a broken laser pointer for $14.83 [2]. Since then, eBay has grown into a multi-billion dollar company. The company's primary site, eBay.com, allows users to list items from a variety of categories to sell at auction or at set prices to other eBay users. The auction site is extremely popular, with approximately 84 million active users worldwide and $60 billon worth of auction sales in 2007 alone [2].

eBay's operations are not limited to auction hosting alone. In past years, eBay has grown to include PayPal, a popular means of online payment for both individuals and businesses [3]. PayPal is a feature seen throughout eBay.com as a means of paying for and accepting payment for its auctions. Paypal allows anyone to pay through credit cards, bank accounts, buyer credit, or account balances without requiring the user to share financial information with the seller and is available in 190 markets and 17 countries [4]. Paypal was acquired by eBay in 2002, and has been deeply integrated throughout eBay's auction site.

eBay's other large company, Skype, operates outside of the auction and payment realm. Instead, Skype focuses on providing voice over internet protocol communications between Skype users to other users via internet, landline, and cell phones [3]. The service is available in 28 languages and generates revenue through premium services

including call forwarding, voicemail, and making and receiving calls to and from landline and mobile phones [5].

2.2. Hadoop

2.2.1. Overview

To meet the needs of an evolving online world and economy, it is necessary for eBay to continually evaluate what users are doing on their site and what their needs are. As a result, eBay's systems create anonymous logs of user actions on a daily basis. These logs are then stored and processed using a distributed stream processing system known as Hadoop.

Before going into detail about eBay's existing Hadoop systems, it is important to understand what Hadoop is. Hadoop is an Apache open-source software project that is based upon the Google Big Table and Google File System projects [6]. It processes large amounts of data by distributing the information among nodes in a cluster, and utilizes a map / reduce algorithm to process that data, which will be explained in more detail in Section 2.2.3. It provides its own open-source distributed file system to store the data on the computing nodes [7]. Hadoop is used by eBay and many other companies to process large files, and these companies often contribute to the open-source project [8]. For example, Amazon uses Hadoop to build their search indices, Facebook uses Hadoop for log storage, and Yahoo! uses Hadoop for its ad system and web search research [9].

2.2.2. Benefits

Hadoop focuses on four especially important benefits that have helped it gain the popularity it has today [10]:

- **Scalability** – Hadoop can store and process petabytes of data.
- **Affordability** – Hadoop distributes the data and processing across clusters of commodity hardware.
- **Efficiency**- Data can be processed in parallel among the nodes.
- **Reliability**- Hadoop automatically maintains multiple copies of the data and automatically redeploys computing tasks in the event of failure.

2.2.3. How It Works

Hadoop utilizes a Map / Reduce algorithm similar to the one developed by Google [11] to process its data. The first portion of this algorithm is the map process, in which the data, in the form of key / value pairs, is split into manageable segments by a master node. Essentially, as described in Hadoop's documentation, "for each input key/value pair (K,V), the map task invokes a user defined map function that transmutes the input into a different key/value pair (K',V')" [6]. The smaller chunks of data are distributed among nodes such that those with similar keys are grouped together. This allows for easy data access based on the keys, and parallel processing of the data at a later state.

Some companies use Hadoop's Map / Reduce to process large log files such as data containing web page information. In this situation, the map state would split the log data into smaller segments with some common feature, which would be the key portion of the key / value pair. For example, internet data could be grouped by domain. The map phase would process this data by grouping it such that the key is the domain and the content is the value. This data would then be split among nodes in the cluster such that data with similar keys were together.

The reduce portion of Map / Reduce first requires some of the data to be processed by a specific job that is run on the nodes. The Hadoop user is responsible for writing the job algorithm, so the actual steps performed during process may vary considerably based on the type of file, the company running the cluster, and their ultimate goals. Once the Map / Reduce steps complete, a resulting output is produced.

When the reduce function begins, the processed output is separated among the nodes in the computing cluster based on how the data was initially distributed (in our previous example, it was by a URL's domain). The purpose of the reduce step is to take all of these individual answers and output a single final result. The reduce step aggregates all of the individual answers and combines the results using a reduction algorithm. The reduction algorithm is a process the Hadoop user writes, so the actual steps involved can vary. The important part is that once those steps complete there is a resulting output that is then appended into one sorted file [12].

Returning to our website URL example, suppose the original data consisted of logs that stored web addresses and content. The original map process could split the data based on website URL as described previously. Suppose we wanted to solve the problem of finding all web pages that contain information about Hadoop. Each node would search for Hadoop information in its data, and send back a list of applicable key / value pairs that matched our request. The reducer would combine the lists returned by all Hadoop nodes to give a final result containing a sorted list of all web pages containing information about Hadoop.

2.2.4. Hadoop on Windows

Despite its many benefits, Hadoop does not natively support running on Windows. There are plans for this to come in the future, and it is currently listed as task number HADOOP-4998 as a major priority update planned for version 0.21.0. However, the current version is 0.19.0, so this update will likely not come for some time. As a result, there are still a number of prerequisites that must be installed before Hadoop will work with a Windows machine.

First, all machines running Hadoop are required to have Java installed. Java is a programming language, developed by Sun Microsystems, which executes on a local virtual machine. As Hadoop is primarily written in Java code, this is necessary for it to run. It is recommended that users have version 1.6.x installed from Sun [13].

Additionally, Cygwin [14] is required for Windows. Cygwin provides a Linux-like environment for Windows, including a fairly extensive Linux-like API. Hadoop uses Unix commands within its code, which the Windows operating system does not natively support. Due to this dependency, Cygwin is required to allow Hadoop to execute its commands. It is also necessary to include the OpenSSH option during the Cygwin installation to allow for Secure Shell (SSH) access, which is required for Hadoop to communicate between the client and server nodes.

Lastly, the Secure Shell protocol (SSH) [15] must be configured on all machines. SSH is a security protocol that allows secure connections between machines using a terminal. SSH supports several modes of authentication, one of which is the use of public and private security keys to connect. Each client has a public and private key. Authenticating based on public and private keys allows for passwordless authentication,

which is a requirement of running Hadoop. Setting up this system entails generating the public and private key pair, pushing them to the server, and configuring the files that deal with authorized keys and known hosts. This will be further detailed in Section 3.3.

2.3. eBay's Use of Hadoop

At the time this project was completed, eBay was using Hadoop version 0.18.2 to process their daily log files. The logs stored several hundred gigabytes of data per day, and were split up into files of about four hundred to five hundred megabytes per file. Logs were stored on a network file system and their Hadoop nodes that ran a maximum of forty-four map processes at one time. A single Hadoop job on one of the log file portions took about twenty minutes to process.

eBay faced one major bottleneck in the processing of these log files. The logs were stored on a network file system rather than locally on their Unix machines. When many Hadoop nodes tried to read log data at once, they were limited by the output bandwidth of the network file system. As a result, even if eBay were to add additional machines to speed up log processing, their data would not process faster because the bandwidth from the network storage would be split among even more computers.

Ultimately, eBay Research Labs hoped to be able to take advantage of developer workstations when they were not in use. This would provide many more potential nodes, and in turn, would speed up the processing of jobs. Additionally, this would remove the bottleneck by storing the log data locally on the participating nodes. However, eBay Research Labs was aware that this should be a voluntary process. As a result, we were asked to build an interface that allowed users to easily set up their Windows computer as a Hadoop node. This tool was intended to allow for a quick install of all pre-requisites,

including Cygwin and SSH configurations, and a user interface with which users could add or remove their Windows computer from the Hadoop cluster at will.

3. Methodology

3.1. Initial Hadoop Setup

To set up a testing environment, we created a Hadoop cluster on the two Windows machines that were available to us. This involved setting up the prerequisite software and SSH keys as described previously in Section 2.2.4. In the upcoming sections, we will describe the process that we used to set up Hadoop. We found the Hadoop Quick Start manual [13] to be very useful in the setup process, and recommend referencing that guide when setting up Hadoop.

3.1.1. Installing Hadoop

Once the prerequisite software was installed, installing Hadoop itself is very straightforward. First, one must download a stable release from the Hadoop website at http://hadoop.apache.org and unzip it to any directory. We used version 0.18.2, and for simplicity we selected our Hadoop path to be C:/hadoop-0.18.2. However, the choice of where to store Hadoop does not matter, and the software we created accepts other names and storage locations for the main Hadoop directory. In the upcoming sections, we will refer to the directory that Hadoop exists at as Hadoop_Home when specifying where files are located.

3.1.2. Configuring Hadoop

Hadoop has many configurable options. Most of these options do not need to be set when starting a cluster, but some of them are required. The first of the required settings are located in Hadoop_Home/conf/Hadoop-env.sh. The user must modify the existing JAVA_HOME variable to be set to the location of the Java install. For example, on our machines, this line appears as "export JAVA_HOME="C:\Program Files (x86)\Java\jre1.6.0_03."" The user must also modify the HADOOP_IDENT_STRING variable with the name of the Hadoop cluster. On our file, this line was "export HADOOP_IDENT_STRING="ebay"." This should be the same across all nodes, and it is likely that the values for JAVA_HOME and HADOOP_IDENT_STRING on other Hadoop clusters will be different.

Additional Hadoop options are configured in Hadoop_Home/conf/Hadoop-site.xml. This file stores the majority of Hadoop's important settings. The ones that are required when starting a new cluster are fs.default.name (the name of the default file system), mapred.job.tracker (the host and port that the MapReduce job tracker runs at), and dfs.replication (the default block replication for the data) [13]. Many other options are available, and are listed with descriptions in Hadoop_Home/conf/hadoop-default.xml along with their default values.

3.1.3. Starting Hadoop

Hadoop can be started once it has been installed, configured, and all other prerequisites described in Section 2.2.4 are set up. If the desired setup will have slave machines rather than just one machine running Hadoop jobs, then those steps should also be performed on each slave node. Additionally, the IP address of each slave should be

added with one IP per line to the slaves file located in Hadoop_Home/conf/slaves on the master server.

To start the Hadoop cluster, run the following commands [16]:

- bin/hadoop namenode –format (to format a new distributed filesystem)
- bin/start-dfs.sh (to start the Hadoop file system)
- bin/start-mapred.sh (to start map / reduce)
- bin/start-all.sh (to start both the Hadoop file system and map reduce in one command)

The two start commands also reference the slaves file to start the file systems and map reduce operations on the slave nodes. Hadoop can be stopped by replacing the "start" with "stop" in the same commands.

3.2. Node Addition and Removal

Since our project focuses on dynamic addition and removal of nodes, we first looked into what was required for a node to be successfully added and removed from a cluster. For this step we assumed that the node was configured for Hadoop as described in Section 3.1. The steps below describe how a node can be added and removed on a basic Hadoop setup.

3.2.1. Important Files

Hadoop features two files that are central to the addition and removal of nodes on Hadoop. The first of these two is the Hadoop_Home/conf/slaves file. This file on the master server stores a list of all nodes. When Hadoop is started, this file is referenced to start the datanodes and tasktrackers on all slave machines. The datanode process simply

manages the storage of files on a node, and the tasktracker manages the node's jobs. If a node is not listed in the slaves file, its datanodes and tasktrackers will not start, and therefore it will not be used as a node in the cluster.

The Hadoop_Home/conf/exclude file is equally important. This file is used to exclude certain nodes from being started as part of the Hadoop cluster. Even if a node is in the slaves file, it will not be started if it is also part of the exclude file. This file is also necessary for removing nodes while Hadoop is still running. An alternative option to having both a slaves and exclude file would have been to only use a single file listing nodes that should be active, but that option would have made it difficult to track removed or inactive nodes. As a result, using a slaves file for active nodes and an exclude file for inactive nodes makes node management more straightforward.

3.2.2. Node Addition

To add a new Hadoop node to a cluster there are only a few steps that need to be taken. First, the user must add the node's IP address or hostname to the Hadoop_Home/conf/slaves file located on the master server. This information will usually already be there, unless this node has never existed on this cluster previously. Also, the user must make sure that the node is not listed in the exclude file, or it will not be added to the cluster.

A new node can be started without affecting the running of the existing cluster. This requires two commands to be executed on the new node's command line. First, the user must navigate to the Hadoop_Home directory. Once there, running "/bin/hadoop-daemon.sh start datanode" and "bin/hadoop-daemon.sh start tasktracker" will start the

data storage and task tracker processes on the new node, therefore adding it to the cluster [17].

3.2.3. Node Removal

Node removal works in a way that is similar to node addition. First, the node must be added to the exclude file. Next, the command "bin/Hadoop dfsadmin –refreshNodes" must be run on the master server. This forces the master server to repopulate the list of valid nodes from the slaves and exclude files. Since the node that is being removed is in the exclude file, the master server will begin a decommissioning process on that node.

Decommissioning a node is meant to prevent data loss and lasts until all blocks on the node are replicated [17]. Depending on the amount of data stored on the system and how often the system updates, this may take some time. To speed the decommission process up, the several variables should be edited in Hadoop_Home/conf/Hadoop-site.xml. First, the dfs.namenode.decommission.interval value specifies the number of seconds between each check for decommissioned nodes. High values would result in "Decommission in Progress" appearing even long after a node has been decommissioned successfully. By default, this value is set to five minutes, but we recommend setting it to a lower value such as 30 seconds. This gives the user a much more reasonable update timeframe, which is important as it allows the user to regain full use of their workspace that much faster.

Additionally, we recommend changing some of the values associated with Map / Reduce. Specifically, mapred.tasktracker.expiry.interval is set by default to be ten minutes. This attribute is defined in the Hadoop-default.xml documentation as "the time-interval, in milliseconds, after which a tasktracker is declared 'lost' if it doesn't send

heartbeats" [18]. With a default value of ten minutes, it is possible for the master to continue to attempt to send jobs to the decommissioning node. By lowering the value, the master will more quickly register the node's tasktracker as dead and will not waste time attempting to assign jobs to it.

Similarly, we recommend changing mapred.task.timeout from its default value of ten minutes to something lower. As stated in Hadoop-default.xml, mapred.task.timeout defines "how long until a task it terminated if it neither reads an input, writes an output, nor updates its status string" [18]. If there is the potential for nodes to frequently be removed it is important that failed jobs due to removed nodes register as quickly as possible so they can be reassigned. In our case, we changed both mapred.task.timeout and mapred.tasktracker.expiry.interval to thirty seconds.

The final value that we recommend editing is heartbeat.recheck.interval in hadoop-site.xml. This attribute specifies how long the master will wait after not receiving a heartbeat from a node before it considers it dead. Given that the slave nodes could be shut down and never have time to properly decommission, this attribute is necessary to quickly register a node as dead.

3.3. Secure Shell Key Generation

Hadoop on Windows requires passphraseless secure shell (SSH) access. However, this is somewhat complicated to set up. As a result, we felt it was necessary to automate the process.

To accomplish this, we created a shell script. This script was adapted from one that already existed at http://www.macosxhints.com/article.php?story=2007091814022049. The script first checks if Digital Signature Algorithm (DSA) keys already exist. If they

do not, the keys are created and stored in the user's .ssh directory. Next, the appropriate permissions are set on the directory. The script does these same steps on the server side to ensure all of the necessary keys are created.

3.3.1. Single User Passphraseless SSH Setup

Figure 1 (below) demonstrates what steps are required to enable passphraseless SSH for a single user. To successfully connect to the master server, the client must to send its public key to the master server's authorized_keys file. The authorized_keys file stores a list of public keys that are allowed to access that machine through SSH. Our script connects using SSH to the master server and concatenates the user's client-side public key to the user's server-side authorized keys file. Likewise, the script retrieves the public key from the server and concatenates it to the user's client-side authorized keys file. The completion of these steps means that the client can successfully SSH to the master server without a password. However, this only works for Hadoop if the administrator has set up a dedicated user account for Hadoop.

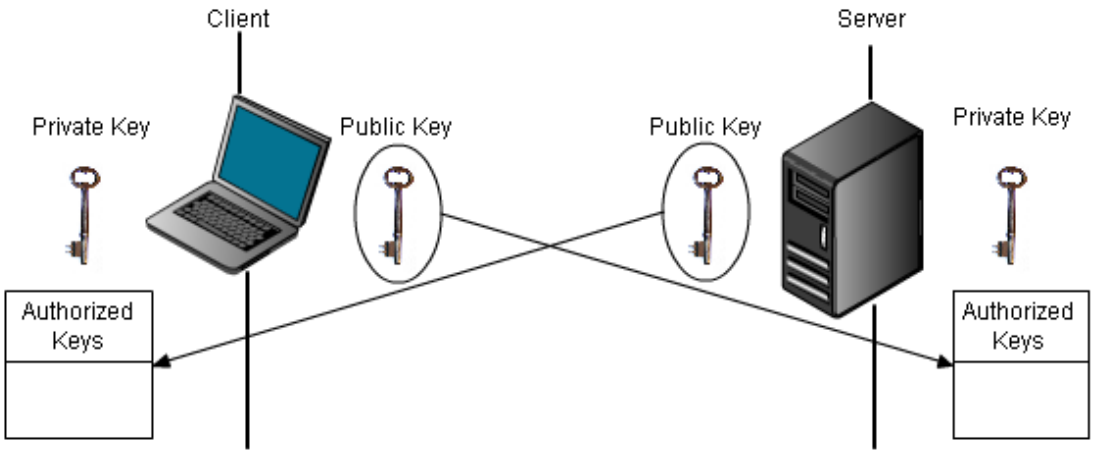

Figure 1: Passphraseless SSH Requirements (single user)

3.3.2. Multi-User Passphraseless SSH Setup

In our case, a dedicated Hadoop user account was not available, so we had to follow additional steps to set up passphraseless SSH. These steps are detailed visually in Figure 2. It is important to note that each SSH user has their own set of public keys, private keys, and authorized keys files. Simply setting up passphraseless SSH for a single user would not work if a separate user account is starting the server and executing the Hadoop commands.

To step around this issue, we added a socket connection between the new user on the server-side and the Hadoop administrative account. This connection sends a command to the Hadoop admin containing the private key. The connection on the Hadoop admin's side writes the key to a Hadoop_Home/.ssh directory. Additionally, it modifies an sshconfig file with lines specifying the hostname, username, and key location for the connection that is being set up. The sshconfig file can be specified as an option when connecting via SSH, which forces the connection to connect using a specific username and key for each host. In doing so, the Hadoop admin is able to connect as a different user for each host successfully without a password.

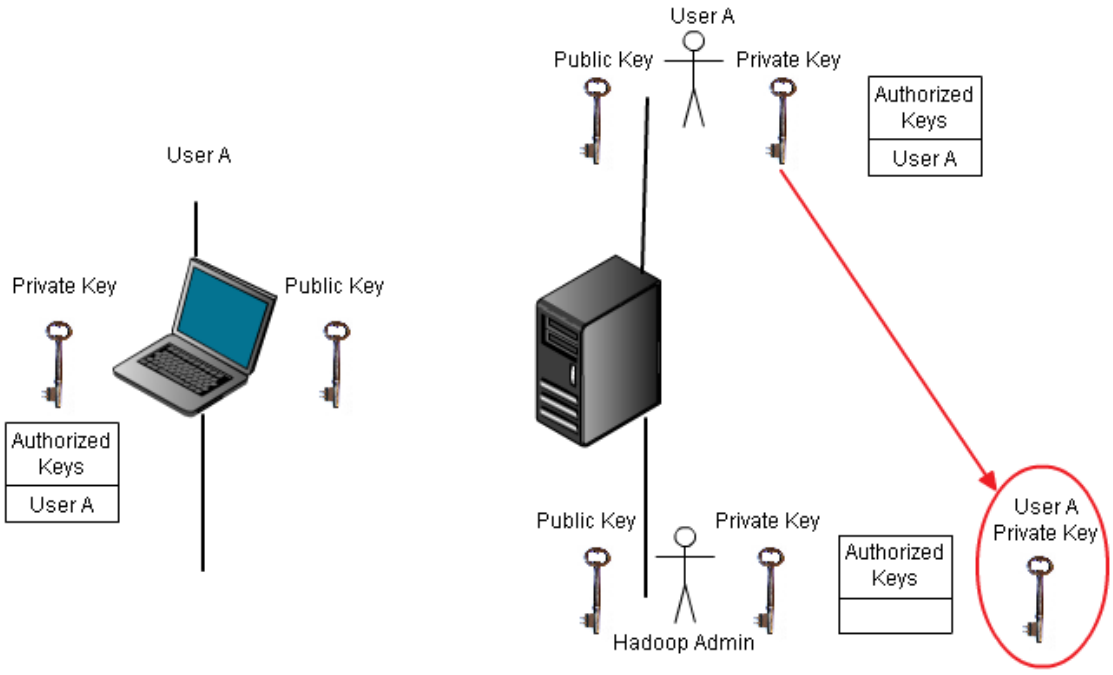

Figure 2: Passphraseless SSH Requirements (multi-user) – Passphraseless SSH must be set up from User A to User A, and their private key made available to the Hadoop Admin.

It is important to note that passphraseless SSH between different users is not recommended. Doing so opens up a security issue in which the Hadoop admin will have SSH access to any client nodes under the user account that ran the installer on the client computer. In our case, a dedicated Hadoop user account was not an option, and connecting through a means other than SSH would be possible only with significant modifications to the Hadoop code, ruining any chances of easily upgrading the Hadoop version. We felt that the ability to upgrade Hadoop was more important, especially since the security concern could be removed if a dedicated Hadoop user account were approved. If at all possible, we strongly recommend a dedicated account to avoid this security and privacy concern.

3.3.3. Expect

This SSH key automation script and the accompanying socket connection did accomplish our goal of preventing the user from having to manually set up their SSH keys while also allowing passphraseless SSH between user accounts, but we also felt that the user should not have to deal with the command prompt requests during the install. In the first version of the script it was possible that during execution the user would be asked if they wanted to add a computer to their known hosts, if they wanted to create their SSH keys, which types of SSH keys they wanted to create, and they were prompted for their password several times. We decided that it was also necessary to automate the process so that the user did not have to worry about any command line input. To accomplish this task, we used a tool known as Expect.

Expect is a government-funded project that is part of the National Institute of Standards and Technology and began in September, 1987 [19]. It is a UNIX program that was designed specifically to interact with interactive programs. A programmer can write a script, usually in Tcl, or "Tool Command Language" [20], describing the interactive dialogue; and the Expect program can then run that program non-interactively [21]. For anyone interested in using Expect with Windows, it is an optional part of the initial Cygwin setup.

We used Expect to control the flow of our SSH key generation script. Using a simple Expect program, we were able to automate the process of key generation and require user input only once to get the username and to get the password. This step was passed on to the installer to allow the user to enter this information in a more visually appealing format than the command line, which later calls the Expect script with that information.

The automation process was performed in a short Expect file of approximately twenty lines, but makes the install process significantly more user-friendly. We simply tell Expect what things the script could potentially ask the user for, and tell it what to give the script in response. For example,

```
expect "Enter passphrase \(empty for no passphrase\):" {
        send "\r"
}
```

tells the program that the script could ask for a passphrase (in this case, it is the passphrase to assign for SSH authentication). Since Hadoop requires passphraseless SSH, Expect simply sends a newline character to the script, and the next operations continue. Similar statements exist for the other output that the key generation script could require. As a result of using Expect, the key generation script completes smoothly while requiring very little from the end user.

3.4. The Hadoop Runner Program

The portion of our project that is visible to the end-user is the Hadoop Runner program. This program was created in Java using both NetBeans and Eclipse and can be divided into three primary sections: user interface, client / server connectivity, and Hadoop property and configuration accessibility.

3.4.1. Key Functionality

Prior to beginning the development of the Hadoop Runner program, we felt it was necessary to determine what features this should support. Through our discussions our

project liaisons, we decided that the Hadoop Runner should support the following basic functions:

- Addition of the node to the Hadoop cluster.
- Removal of the node from the Hadoop cluster.
- Configuring the storage space used by Hadoop.
- Displaying information about the status of the Hadoop node.

With these requirements in mind, we began designing a program that would support these basic features.

3.4.2. User Interface

The user interface portion of Hadoop Runner was designed using NetBeans' Project Matisse Graphical User Interface (GUI) Builder. This GUI Builder, which is included with the NetBeans development environment, makes developing GUIs using Java's Swing components very simple. Further information about this tool can be found at on Netbeans' website [22].

Prior to our preliminary meetings with our liaisons, we created a simple mockup of what we had planned for the GUI (shown below in Figure 3: Initial GUI Mockup). Initially, we had only planned to support adding and removing nodes, showing the current status, and displaying the current job status.

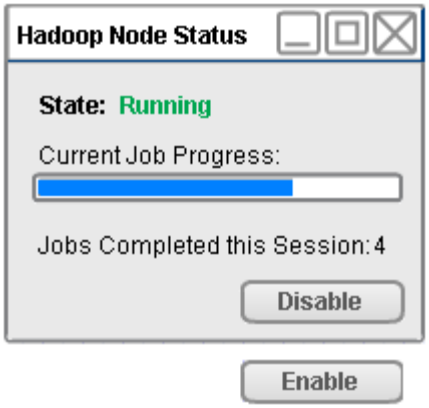

Figure 3: Initial GUI Mockup

After discussing the preliminary UI with our liaisons, we were given feedback and found that the GUI should provide some additional information to the user including basic information about the space occupied by Hadoop. However, we were limited in what we could display because of Hadoop's constraints. Hadoop does not offer any publicly accessible functions that get basic node information such as total completed jobs, job progress, space allocated to Hadoop on a node, and other similar requests. As a result, we were limited to using attributes that were accessible through Hadoop's existing web interface.

Based on our liaisons' feedback and the information we had available through Hadoop's interface, we decided that the user should have the following information available in the GUI:

- Status of the node – Necessary to tell if the node is connected or not.
- Hostname – To make troubleshooting any problems easier for the Hadoop admin.
- Space occupied by the Hadoop Distributed File System (HDFS) – To assist the user in choosing an appropriate amount of allocated space for Hadoop.

- Number of running jobs – To inform the user if any jobs are in progress, which may encourage them to keep their node online.
- Free Space – Only available when editing space allocated to Hadoop, to give the user a value to assist them in choosing an appropriate maximum.
- Space Allocated to Hadoop – Only available when editing space allocated to Hadoop, to keep the user aware of their current settings.

The GUI that we created to meet these requirements can be seen below in Figure 4.

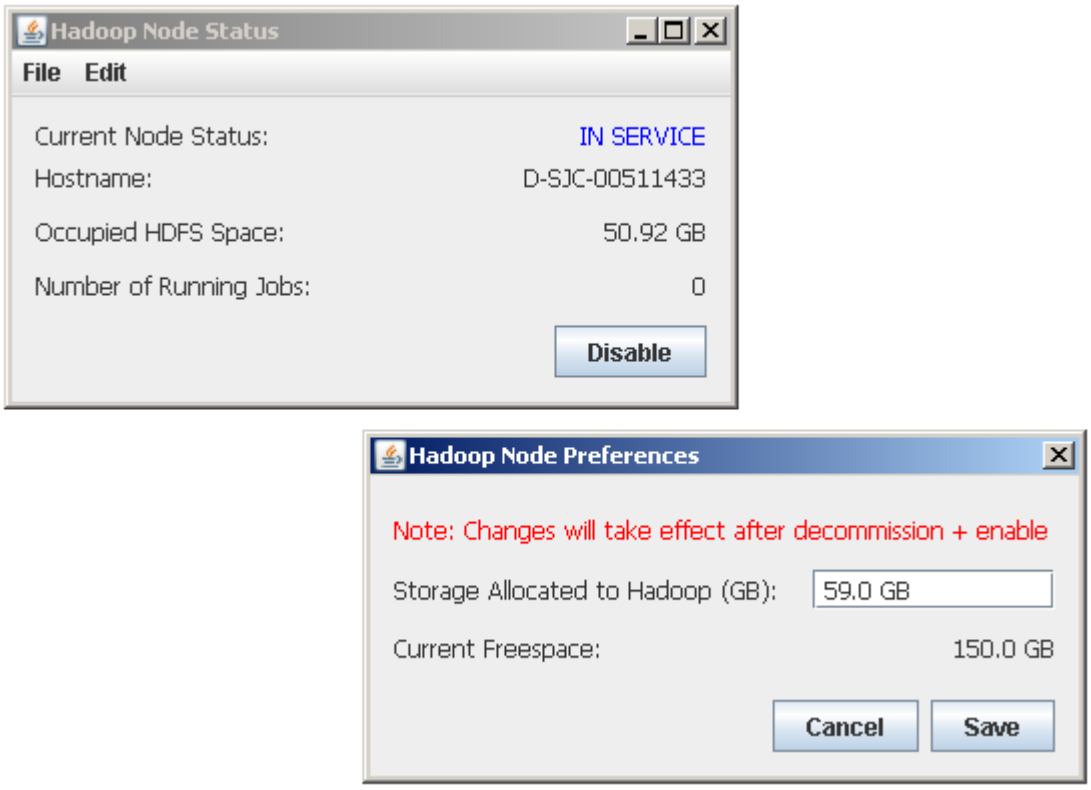

Figure 4: Hadoop Node GUI

Figure 4's screenshot of the GUI shows the two windows that the user is likely to see. Other windows exist, but are only used to convey error messages. In Figure 4 the window in the background is the primary user interface. The window in the foreground, Hadoop Node Preferences, is only visible if the user chooses Edit -> Preferences from the

main GUI menu. Both display helpful information to the user about their computer's status and characteristics as a node on the Hadoop cluster.

3.4.3. Client / Server Connectivity

One primary requirement of the Hadoop Node program is that the client and server be able to communicate so instructions can be passed for operations such as node addition and removal. To accomplish this task, we created a socket connection and a protocol for communication between the client and the server. The client and server may need to communicate for the following reasons:

General Hadoop Functionality:

- Client prompting server to be added to the list of nodes.
- Client prompting server to be removed from the list of nodes.

Setup:

- Client prompting server to add its key to the authorized keys, sshconfig, and .ssh folder.

As a result, we created a basic protocol to support this functionality as well as the initiation of the connection between the client and server. Additionally, we synchronized access to functions that modify files to avoid concurrent write attempts to the same file.

3.4.4. Hadoop Property and Configuration Accessibility

The Hadoop Runner program also requires frequent reading and accessing of various Hadoop related properties. To deal with this, we created a PropertyReader and PropertyWriter factory. These allow for easy reading and writing of XML files and INI files. The XML files must store information with <name> and <value> tags for the

reader and writer to function properly. Similarly, any INI files must have a name = value pair on each line.

Within this project, the PropertyReader and PropertyWriter classes are used to access the setup.ini and hadoop-site.xml properties. Setup.ini is primarily read from and written to during setup to save important information including the location of Hadoop, the master server address, and the location of Cygwin. Hadoop-site.xml is read from and written to when any configuration changes within the settings of Hadoop are made, or when the program needs information about Hadoop's configuration. The overall purpose of the PropertyReader and PropertyWriter is to have an easy to use interface between the Hadoop Runner program and its configuration files.

3.5. The Installer

3.5.1. NSIS

The Nullsoft Scriptable Installation System (NSIS) is a scripting framework for building installer programs for Windows machines [23]. This was used to create the actual installer for our project. To go from a clean machine to a running node on a Hadoop cluster using our software, several steps by an installer program are necessary. The installer includes the setup files as well as the Cygwin and Hadoop installations to use. The user has the option of not including the setup files, Cygwin, or Hadoop; if Cygwin and/or Hadoop are not included, the program will ask the user to specify the directory of a working version of the omitted program.

4. Testing and Results

Much of our testing was done at the component and unit levels. Component level testing is a method of testing in which the software is subdivided into similar parts, stimuli is applied to the component and the result is evaluated through a state change either within the component or elsewhere in the system [24]. In our case, the components we tested fit into three main parts: command line level testing during initial setup, testing of the HadoopRunner java components, and testing of the installer and its associated setup scripts.

4.1. Command Line Component Testing

The earliest testing that we performed was through the command line during initial setup. This testing had little to do with the program we ultimately created, but was instead needed to ensure that the prerequisites of Cygwin, Hadoop, and node addition and removal were working as expected. During this portion of the project, we set up Cygwin, passphraseless SSH, and Hadoop for distributed operation. Testing was performed at the following checkpoints:

- Following the setup of the SSH server, to ensure a user could log in with a passphrase.
- Following the setup of passphraseless SSH, to ensure the user could log in without a passphrase.
- Following the setup of Hadoop on a single node, to ensure Hadoop could start and run a sample job.

- Following the setup of Hadoop on a distributed system (multiple nodes) to ensure Hadoop was properly starting its nodes, running jobs, and assigning tasks.

These basic tests enabled us to be confident that the software we needed to work with was properly configured.

Also, we needed to be sure that node addition and removal worked successfully prior to attempting to automate the process. To accomplish this, we first tested node addition and removal at the command line level. We manually followed the node addition and removal steps described previously in Section 3.2 and were able to confirm that nodes were successfully added and removed when those steps were followed by viewing the Hadoop Administration webpage. With this knowledge, we had a clear plan for automating the node addition and removal process, and were able to begin working our program.

4.2. Hadoop Runner Testing

We tested the Hadoop Runner program and its associated Java files primarily through unit testing and component testing. Unit testing is a process of testing in which one takes the smallest piece of testable software in the application, isolates it from the remainder of the code, and determines whether it behaves exactly as expected [25]. We performed unit tests as individual portions of code were finished. We primarily performed these tests manually, though in hindsight, we should have used an easily automated test suite to accomplish unit testing more effectively.

We also tested our software at the component level. The Java portion of the project could be split up into three separate sections – the initial setup related functions, the SSH setup related classes, and the Hadoop Runner portion. To test the initial setup functions,

we manually ran the main class and checked each file that it creates to ensure that the proper output was produced. For the SSH related classes, we ran the setup file along with the script that calls the class and attempted to passphraseless SSH directly after. A successful attempt to connect using passphraseless SSH meant that this section worked successfully.

Lastly, the Hadoop Runner program was tested both client-side and server side. We tested node addition by attempting to add a node, verifying it was added on the server through the web interface, and verifying that its tasktracker and datanode were started properly through the logs. For node removal, we verified that it was decommissioned successfully through the web interface, and watched the client side log output to ensure that the datanode and tasktrackers were stopped as expected.

4.3. Hadoop Runner Job Testing

When the Hadoop Runner program was completed and the individual functions and components had been tested, we felt that it was necessary to test it on a cluster similar to the one used currently at eBay Research Labs. To do so, we set up a Solaris system to be the master server, and set up the test node as the slave. We began with basic example jobs included with Hadoop, and attempted simple node addition and removal to ensure those ran without error. When those completed successfully, we began more in-depth testing.

Our primary goals of testing were as follows:

- To ensure a node could be added mid-job.
- To ensure a node could be repeatedly added and removed without failure.
- To ensure the jobs completed successfully, even with mid-job node removal.

- To ensure data was not lost from node removal.

To make these tests as realistic as possible, we copied some of eBay's sample log data to our Hadoop file system, and the jobs we ran were the same as those that eBay runs on its existing Hadoop server.

Our first tests involved basic node addition and removal. We wanted to be sure that nodes could be added to the server, and quickly used to the advantage of the cluster. To test this, we began a job on the master server while our node was offline. We then enabled our node using the client side program. We monitored the Hadoop administrative web interface and saw that within seconds of being added it was listed as in service and had been assigned a task.

Our next test was to ensure that node addition and removal worked properly. Prior to this, we had only tested each on a singular basis, and never repeatedly. We repeatedly removed and added the node to make sure that the slaves and exclude files appeared normal at the end of each attempt. In doing so, we uncovered a bug in which the node was added to the slaves file after each addition resulting in many duplicate entries, and were able to promptly correct it.

Next, we tested node removal during a job. Since nodes are likely to be removed while jobs are running, it is important that Hadoop recognizes this quickly and redistributes the tasks accordingly. On our initial runs, we realized that tasks were not quickly redistributed. To correct this, we reviewed the Hadoop documentation and found that we had to modify some of the Hadoop-site.xml configuration options. The changes that we made to correct this issue are documented in Section 3.2.3. On subsequent

attempts, nodes were quickly identified after going offline and tasks were reassigned as expected.

Lastly, we tested to make sure that data was never lost during node addition and removal. To accomplish this, we began a job and decommissioned a node partly through the run. Prior to decommissioning, we ran the command "hadoop fsck folderName" whose output is shown in Figure 5.

```
Target Replicas is 2 but found 1 replica(s).
Status: HEALTHY
 Total size:    3865732200 B
 Total dirs:    0
 Total files:   7
 Total blocks (validated):      59 (avg. block size 65520884 B)
 Minimally replicated blocks:   59 (100.0 %)
 Over-replicated blocks:        0 (0.0 %)
 Under-replicated blocks:       41 (69.49152 %)
 Mis-replicated blocks:         0 (0.0 %)
 Default replication factor:    2
 Average block replication:     1.0
 Corrupt blocks:                0
 Missing replicas:              41 (69.49152 %)
 Number of data-nodes:          1
 Number of racks:               1


The filesystem under path '/tmp' is HEALTHY
-bash-3.00$ hadoop fsck /tmp
.......Status: HEALTHY
 Total size:    3865732200 B
 Total dirs:    0
 Total files:   7
 Total blocks (validated):      59 (avg. block size 65520884 B)
 Minimally replicated blocks:   59 (100.0 %)
 Over-replicated blocks:        0 (0.0 %)
 Under-replicated blocks:       0 (0.0 %)
 Mis-replicated blocks:         0 (0.0 %)
 Default replication factor:    2
 Average block replication:     1.6949153
 Corrupt blocks:                0
 Missing replicas:              0 (0.0 %)
 Number of data-nodes:          2
 Number of racks:               1
```

Figure 5: Hadoop FSCK Output

The output reported that all data was accounted for. After decommissioning, we ran the check again. We found that the data still existed, but was under-replicated. This was due to the fact that replication was set to two and only one node was online. In an actual Hadoop cluster, Hadoop would recognize the under replication automatically and redistribute the data accordingly.

5. Recommendations and Conclusions

The Hadoop Runner program and installer work to correctly, quickly, and easily add a node to an existing Hadoop cluster. The resulting program met the requirements that our mentors had set at the start of the project. The nodes can be successfully added and removed at will, the setup scripts deal with the initial configuration, the installer handles the install of all necessary prerequisites, and the users have the freedom to choose how much space Hadoop is using.

Additionally, this project offers other potential uses aside from the current plans of adding nodes to the existing cluster. First, the installer is flexible enough to be configured for use on a different Hadoop cluster. As a result, it is possible that eBay could set up another master server and use the installer to quickly and automatically configure its data nodes. Previously, configuring its nodes would have taken several hours or days of work on Windows.

Also, this project opens up the possibility of using Hadoop to those at eBay who otherwise may have never been introduced to it. For example, employees in a less technical department may not have the resources or expertise to set up a Hadoop cluster and run jobs, but they would likely have use for the information that could come from the daily log data. eBay Research Labs could open up their Hadoop systems to be a company wide tool available to run specific jobs in exchange for a certain number of hours of node usage. With a simple installer, anybody can quickly configure a computer for Hadoop with little time investment, so this prospect is an intriguing option.

Although our program is certainly useful, there is also room for improvement that could be added in a future project. For example, a scheduler system is one feature that

we would have liked to implement but did not have the time to do so. This feature would allow users to specify when to activate and decommission their computer as a node. This would be especially helpful when encouraging users to add their node while they are away at meetings, gone for the day, or for overnight job runs.

Another feature that we would have liked to see added was client-based job submission. This option would fit well with the job requests for node usage tradeoff that was described earlier, as it would allow departments that are not programming-savvy to easily submit requests for data through a user interface. Currently, job algorithms must be written in java and the jobs must be run through the command-line while connected to the Hadoop admin account on the master server. However, simple jobs could be run though a visual interface that would help the less technical departments get useful information.

Overall, the creation of an easy to use node addition and removal tool opens many new doors. Faster processing of log data and the ability to run more complicated jobs can give eBay new insight into what users are doing on their site, and in turn, can help shape the company's future. We hope to see this tool continue to be improved upon and ultimately used throughout eBay to take advantage of their existing company resources.

Appendices

5.1. Maintainer Documentation

5.1.1. Hadoop Upgrade / Reinstallation Requirements

We avoided modifying Hadoop's Java code to allow for easy updates from one version of Hadoop to another. However, there are some changes that will need to be made when updating Hadoop. These changes were made to the hadoop-daemons.sh and hadoop-daemon.sh scripts, path variables, and SSH options. By default, Hadoop looks for Hadoop in the same location on every client machine when starting and stopping. However, since this location can vary depending on the username it is stored under, the startup scripts had to be modified and a HADOOP_HOME environment variable had to be added.

In hadoop-daemon.sh, the final line was modified as follows:

Original:

```
#exec "$bin/slaves.sh" --config $HADOOP_CONF_DIR cd
"$HADOOP_HOME" \; "$bin/hadoop-daemon.sh" --config
$HADOOP_CONF_DIR "$@"
```

Modified:

```
exec "slaves.sh" --config $HADOOP_CONF_DIR
"hadoop-daemon.sh" --config $HADOOP_CONF_DIR "$@"
```

Also, be sure to add the HADOOP_HOME environment variable to the master server as well as HADOOP_HOME \bin to the PATH variable to ensure proper server-side startup.

The hadoop-env.sh file located in HADOOP_HOME /conf should be modified to account for the use of a configuration file in the SSH options. The new SSH_OPTS parameter value should be "-F $HADOOP_HOME/sshconfig." Unfortunately, Hadoop is not consistent in its naming of this parameter. In the hadoop-env.sh file, it is specified as SSH_OPTS, but in the slaves script in bin/slaves.sh, it is specified as HADOOP_SSH_OPTS. It must be updated to "SSH_OPTS" in the slaves.sh script for the changes to properly work.

Finally, the hadoop-daemons.sh file should be modified. The line HADOOP_CONF_DIR=$HADOOP_HOME/conf should be added early in the script before the hadoop_rotate_log() function begins. This sets the client-side Hadoop configuration directory to prevent the client computer from looking in the same directory as the server. Additionally, the line "if kill -0 `cat $pid > /dev/null 2>&1; then" should be changed to "if kill -0 `cat $pid| perl -ne '$_=~ s/^.*?(\d+).*$/$1/g; print;'` > /dev/null 2>&1; then" to account for Cygwin's addition of Windows line breaks at the end of the process ID file.

5.1.2. Server Updates

As explained in Section 3.3.3, we used Expect to automate the setup of passphraseless SSH. However, server updates have the potential to break this part of the installer. When we switched from a Windows Cygwin master server for testing to a Solaris master server, we discovered that the exact wording of the prompts varied slightly. Accounting for those variations was a simple matter of adding additional cases to the Expect script that we had written. If a server update changes the wording of the prompts during SSH related configurations, the Expect script will need to be updated to reflect those changes.

5.1.3. GUI Updates

The GUI for the Hadoop Runner program was created using Netbeans' GUI Builder. As a result, we recommend continuing any GUI-related changes by editing those files in NetBeans. Attempting to manually edit the code in an editor such as Eclipse could result in the file not being readable by NetBeans' GUI Editor. It is certainly still possible to work with the code in Eclipse, and working with non-GUI code will have no ill effects. However, editing GUI related code in another editor is likely to make it unreadable by the Netbeans GUI Builder, and would ruin the possibility of taking advantage of its simple to use features. Ultimately it is up to the decision of the software's administrator, but we recommend using the GUI Builder because of its simplicity and time-saving nature.

5.2. Server Side Configuration

For the Hadoop Runner program to work properly on client side computers, the socket listener must be started on the server. Please follow the steps below to ensure the server is properly configured and the socket listener is running. These steps should be performed after Hadoop has been set up and verified to be functioning. It is not necessary for Hadoop to be started at this point.

5.2.1. Jar File Setup

First, copy the Hadoop Runner jar file to a location on the server, and copy the Apache Log4j jar file to a directory entitled "lib" within that same folder. These copies can be done with the Unix commands "scp HadoopRunner.jar username@master:/home/username" and "scp –r lib username@master:/home/username."

5.2.2. Setup.ini Configuration

Next, configure the server's setup.ini file. This file stores important information such as the Hadoop home directory, the master server, and the Java home directory. On the client machines, this file is created automatically during installation. However, since the installer will never run on the server, the Hadoop administrator must set this up manually. It is important that setup.ini be stored in the same location as HadoopRunner.jar. Setup.ini should contain information similar to the following:

```
[Hadoop_Config]
Hadoop_Location=/home/ahargraves/hadoop-0.18.2
JAVA_HOME=/usr/java/jre
HADOOP_IDENT_STRING=eBay, Inc.
hadoop.tmp.dir=/home/ahargraves/Hadoop-tmp
MASTER=erl09.arch.ebay.com
```

Your setup.ini file should be edited based on your server's configuration.

- HADOOP_LOCATION should be changed to reflect the location of your Hadoop install.
- JAVA_HOME should represent the location of your Java install.
- HADOOP_IDENT_STRING should reflect the name of your Hadoop cluster (this should be the same as what you have specified in
- HADOOP_HOME/conf/Hadoop-env.sh).
- Hadoop.tmp.dir should reflect the temporary storage space for Hadoop.
- MASTER should reflect the full hostname of the master server.

5.2.3. Hadoop-Site.xml Requirements

In the case where nodes may be frequently added and removed, there are additional Hadoop-Site.xml configuration parameters that we recommend to ensure quick updates to node status changes. These were detailed in Section 3.2 and should be referenced when configuring the server. Once the HadoopRunner jar files have been transferred, the setup.ini file is set up, and the hadoop-site.xml contains valid information, the server-side socket listener can be started.

5.2.4. Starting Server Side Socket

The server-side socket listener should be started whenever client machines have the potential to be added or removed, or when the installer is running on a client computer. We recommend running the server side socket whenever Hadoop is running. To start the socket listener, follow these steps:

1. SSH to the master server (ssh username@master)
2. Change directory (cd) to the location of the Hadoop install
3. Once in the directory that contains HadoopRunner.jar, type

   "java –jar HadoopRunner.jar HadoopMultiServer"

At this point, the server-side listener has been started. If the command line window remains open, you will see the output of any errors that occur during the client / server communication, as well as information when a node is added or removed.

5.3. HadoopRunner.jar Parameters

HadoopRunner.jar is the responsible for several things. It contains the server-side program, the client side Hadoop Runner program, and two different setup programs. In

this section, we will detail what each program is responsible for and how to begin each one:

- Initial Setup:
  - Run by the installer (java –jar HadoopRunner.jar InitialSetup)
  - Responsible for setting up the environment variables HADOOP_HOME and setting the PATH variable to include HADOOP_HOME/bin. Also edits the /etc/passwd file to remove spaces from the home directory, to avoid problems that Cygwin has interpreting spaces.
- SSH Setup:
  - Run by the script ssh-keys.sh, called within the host_config_expect script portion of the installer (java –jar HadoopRunner.jar SSHSetup client_hostname)
  - Responsible for transferring the installing user's server-side private key to the Hadoop admin. Required for multi-user passphraseless SSH implementations.
- Hadoop Multi Server:
  - The server-side listener as described in Section 5.2.4. This should always be active, but must be manually started on the server side.
- Main GUI:
  - The client-side Hadoop Runner program (java –jar HadoopRunner.jar MainGUI) that should be called by the shortcut created by the installer.

5.4. Updating the Installer

As with any software project, the ability to update the final product is not only a good idea but necessary to continual functioning. To this end, there are two ways of adding to the installer. The first is through the 'installer.ini' file. This file lists out properties and their values, which the installer will read during compilation and export out to the 'setup.ini' file, which is used during installation to run the Java program and some of the Cygwin setup scripts. This does not allow for major customization; rather it allows a user to set the value of certain variables which are likely to change, such as the address of the master Hadoop node and the identification token of the Hadoop cluster.

To edit the installer itself, the NSIS compiler is required. This process is more involved, as the installer has its own scripting language which a user would need to learn before significant changes could be made. The original files used to compile the script were provided to eBay for this purpose.

The configuration scripts run by both the installer and the Java programs can also be updated. These scripts were written in Cygwin shell code, and as such should not be difficult to update or expand upon by a user with knowledge of Cygwin or UNIX systems.

The Java client allows for updates, but in most cases doing so requires changing of the Java code and recompiling of the JAR. Some properties, such as those specified in the setup.ini file of the Hadoop Runner directory can be changed without affecting the source code, but aside from that source code changes would need to be made. These changes are likely to be necessary if Hadoop updates the commands used to decommission or enable nodes, or if it adds or removes information from the online status pages. Since Hadoop has no way to get node status aside from the web-based utilities, we needed to parse the HTML of the website to get the information we needed. Changes to the structure of the page could affect the GUI accuracy, and would need to be updated. The source code for the Java programs was provided to eBay to allow them to made further modifications as necessary.

5.5. Common Problems and Their Solutions

In this section, we detail problems that we encountered frequently over the course of this project, as well as ways to solve these issues. We realize that eBay may hope to upgrade their servers to a later version of Hadoop. Although we avoided modifying

Hadoop's source code with the exception of the startup scripts, it is still likely that small changes may need to be made to our program to keep it up to date. Additionally, some of the issues we describe are general Hadoop setup problems that anyone setting up a Hadoop cluster has the potential to face. We hope that by detailing problems we frequently encountered we can make Hadoop setup and modification of our code more straightforward. In general, utilizing the Hadoop log files, which are stored in the Logs folder of the Hadoop install by default, can easily identify most problems. Additionally, it is important to check log files on both the master and the problematic slave computers because they store different information. Hadoop's error messages are occasionally unclear, but the Hadoop user mailing lists are an excellent resource in the event that an unclear error comes up.

5.5.1. Invalid Namespace ID

The "Invalid Namespace ID" error was an error that we encountered frequently, especially when first starting the datanodes and tasktrackers on a newly formatted Hadoop cluster. Although it was unclear what was causing this error, it has a very simple solution, and in our experience it did not reoccur unless starting from a newly formatted cluster. To solve the Invalid Namespace ID error, first locate the log file that displays this error on the master and slave PCs. The error will display what namespace was expected, as well as what the computer has set currently. Simply navigate to the location that you specified for Hadoop's storage, and open logs/hadoop-clustername-datanode-hostname.log. Copy the expected Hadoop namespace ID from the log and paste it into the version file stored in Hadoop's distributed file system. Saving the changes on this and any other affected nodes should solve the issue.

5.5.2. Spaces in Filepaths

Many of the problems that we encountered were not caused by misconfigured settings, and as a result were not easy to diagnose. Ultimately, we discovered that spaces in the filepaths were the root cause of many of our issues. For example, attempting to start Hadoop if the Hadoop home location had a space, or attempting to start SSH if the user's home location had a space would frequently result in errors. Eventually, we discovered that Cygwin was not always able to properly work with files that had spaces in their location name. As a result, we opted to keep our user's home directories as /home/username rather than C:/Documents and Settings/…etc. Also, we chose to restrict the options that the user had when installing Hadoop and Cygwin to avoid these problems. We recommend keeping these restrictions in place to avoid potentially difficult to solve errors.

5.6. Installation

In this section we detail how to install in the various cases where some of the pre-requisites already exist on the computer.

Installation in the ideal case is fairly simple. The ideal case would be a computer that has a Java JDK in a file path that does not have spaces (C:\Program Files\Java would not work, but C:\Java would) and has neither Cygwin nor Hadoop installed, though Hadoop installations will not conflict. In this case the installer can install and run all scripts normally, and the installation files will be contained within the C:\Hadoop_Installer\Hadoop_Install folder.

Installation on a computer with Hadoop is not much more complex. The user will need to deselect the "Hadoop" checkbox during section selection. This will add a page to the process allowing you to specify the directory of Hadoop you want to use. After this, installation will continue and the scripts will run normally. One thing to watch out for in this case is that the installer's version of Hadoop (0.18.3 as the time of writing) has some modifications to some of the Hadoop startup scripts and configuration files. The post-installation scripts will change certain values in files in the conf directory of your Hadoop installation, however the files containing potentially necessary modifications in the /bin directory will not be copied over and the changes are not made by the scripts. The directory set by the installer for HDFS will be in the C:/Hadoop_Installer/ Hadoop_Install/DFS directory, by default.

Installation on a computer with a pre-existing Cygwin installation can become fairly difficult. It is recommended that if the user already has a Cygwin installation that they not install the installer's version of Cygwin. Also of note is that the Perl, awk, expect, inetutils and OpenSSH plugins are required for the installation scripts and Hadoop to run. The installer will ask for Cygwin's home directory and use this in executing the setup scripts.

The first script (mkpassgrp.sh) will generate group and passwd files if they do not already exist, and set permissions on the /var directory to 711 using chmod, a change necessary for the later SSH scripts. The next step in the installer modifies the home directory of the current user in Cygwin to /home/<username>. This is to change the default Windows home where the path contains spaces (Documents and Settings), which need to be removed. The third step taken is to append three lines to the .bashrc file in the

home directory, as well as a single line to the bash_profile file in the same directory. These additions are appended and will not overwrite any existing data in the files. The fourth and final step will use Expect to run the SSH configuration for the client and the server. This uses ssh-keys.sh, as well as ssh-host-config -y, to set up. This will attempt to create an SSHD service called 'sshd' and .ssh files in the user's home directory, located in the /home/<username> folder on the local machine and the server. As a cleanup step after this, the initial login information used by the installer is removed from setup.ini after SSH setup.

If Cygwin was pre-installed on the computer but the user had the installer install its own version of Cygwin regardless, then an important case arises. The installer will, as part of Cygwin installation, check for C:/Cygwin; if this exists, it will rename cygwin1.dll to cygwin1.dll_ and alert the user to the change. This is necessary, since if multiple versions of Cygwin exist on a machine they will fail to start until there is only a single 'cygwin1.dll' file on the system. If the user has a pre-existing Cygwin in another location, then they must perform this change manually.

5.7. Uninstallation

Uninstallation of the program can be potentially difficult due to some of the actions performed by the setup scripts. In this section, we detail the steps that should be taken if a user wishes to uninstall the Hadoop Runner program and remove modifications made by the setup scripts.

In the ideal case where the installer installs all the necessary components, this process is the least complex. Here, there are three steps. First, open up Cygwin and enter the 'cygrunsrv -L' command. This will list the Cygwin services which are installed; in this

list should be a service 'sshd'. To remove this service, run the command 'cygrunsrv -R sshd'. Next, exit the Cygwin window. For the second step, delete the Hadoop_Installer directory to clean out all of the files that were installed. Finally, to remove the Cygwin path from the Windows PATH variable, right click on My Computer, go to Properties->Advanced->Environment Variables and find the "PATH" entry in the second list. In this string delete the entry corresponding to the installed Cygwin folder, generally of the form ";C:/Hadoop_Installer/Hadoop_Install/Cygwin/bin". There will also be a path similar to the previous but ending in "Hadoop/bin"; this can be removed as well.

If Hadoop was already installed, the process is much the same as above; Hadoop versions do not conflict, and the Hadoop installation only modifies the installed Cygwin's .bashrc folder to have an entry Hadoop_Home, as well as the Windows PATH.

If Cygwin was not installed and a pre-existing installation was used, reversions to the effects of the scripts need to occur. As Cygwin installations can be very non-standard there is no definite guide to doing this, however the actions the scripts perform are well-documented and the scripts themselves can be found in the installation directory (C:/Hadoop_Installer/Hadoop_Install). Changes to keep in mind are the addition of an 'sshd' service, modifications to the user's home directory and SSH key setup on both the client's machine as well as the central server.

If Cygwin was installed with the installer (and Hadoop was not), follow the actions for an ideal install above. Keep in mind that if a pre-existing Cygwin installation was on the machine it will need to have its cygwin1.dll_ file renamed to the original name of cygwin1.dll.

Works Cited

[1] eBay, Inc. “eBay Media Center: About eBay.” eBay. http://news.ebay.com/about.cfm (accessed January 21, 2009).

[2] eBay, Inc. “eBay Media Center: Our History.” eBay. http://news.ebay.com/history.cfm (accessed January 21, 2009).

[3] Yahoo!. “EBAY: Profile for eBay Inc.” Yahoo!. http://finance.yahoo.com/q/pr?s=ebay (accessed January 21, 2009).

[4] PayPal. “About Us – Paypal.” https://www.paypal-media.com/aboutus.cfm (accessed January 21, 2009).

[5] Skype Technologies. “About Skype.” http://about.skype.com/ (accessed January 21, 2009).

[6] Apache. “ProjectDescription – Hadoop Wiki.” http://wiki.apache.org/hadoop/ProjectDescription (accessed January 21, 2009).

[7] Apache. “Frontpage – Hadoop Wiki.” http://wiki.apache.org/hadoop/FrontPage (accessed January 21, 2009).

[8] Apache. “Who Are We.” http://hadoop.apache.org/who.html (accessed January 21, 2009).

[9] Apache. “PoweredBy – Hadoop Wiki.” http://wiki.apache.org/hadoop/PoweredBy (accessed January 21, 2009).

[10] Apache. “Welcome to Hadoop!” http://hadoop.apache.org/core/ (accessed January 21, 2009).

[11] Dean, Jeffrey and Sanjay Ghemawat. “MapReduce: Simplified Data Processing on Large Clusters” (paper presented at the Sixth Symposium on Operating System Design and Implementation, San Francisco, California, December 2004).

[12] Apache. “HadoopMapReduce.” http://wiki.apache.org/hadoop/HadoopMapReduce (accessed January 21, 2009).

[13] Apache. “Hadoop Quick Start.” http://hadoop.apache.org/core/docs/current/quickstart.html (accessed January 21, 2009).

[14] “Cygwin Information and Installation.” http://www.cygwin.com/ (accessed February 2, 2009).

[15] 360 Information Security. “SSH” http://www.360is.com/04-ssh.htm (accessed

February 28, 2009).

[16] Apache. “Hadoop Cluster Setup.” http://hadoop.apache.org/core/docs/current/cluster_setup.html (accessed January 29, 2009).

[17] Apache. “Hadoop FAQ” http://wiki.apache.org/hadoop/FAQ (accessed January 29, 2009).

[18] Apache. “Hadoop-Default” http://hadoop.apache.org/core/docs/r0.18.2/hadoop-default.html (accessed February 12, 2009).

[19] Libes, Don. “Expect,” National Institute of Standards and Technology. http://expect.nist.gov/ (last edited October 20, 2006).

[20] Tcl Developer Exchange. “Tcl Developer Site,” http://www.tcl.tk/ (accessed February 2, 2009).

[21] Libes, Don. *Exploring Expect: A Tcl-Based Toolkit for Automating Interactive Programs*. O’Reilly Media, Inc, 1994. Section 1.1.

[22] NetBeans. “NetBeans IDE 5.0 GUI Building Resources,” http://www.netbeans.org/kb/articles/matisse.html (accessed February 12, 2009).

[23] “Nullsoft Scriptable Install System,” http://nsis.sourceforge.net/Main_Page (accessed March 2, 2009).

[24] IBM. “Component Testing with IBM Rational Application Developer for WebSphere Software.” http://www.ibm.com/developerworks/rational/library/05/kelly-stoker (accessed February 19, 2009).

[25] Microsoft. “Unit Testing”. http://msdn.microsoft.com/en-us/library /aa292197(VS.71).aspx (accessed February 28, 2009).