**Testing Hadoop / HDFS (CDH3u2) Multi-users with Kerberos on a Shared Environment**

Tak-Lon (Stephen) Wu and Zhenhua (Gerald) Guo

1. **Introduction**

The original Hadoop, before version 0.20, supports multi-users simultaneously read/write with looking up permissions based on a plaintext group table. The users listed on this table do not require being real Linux users. After version 1.0 (developed based on hadoop 0.20.205), integrating with Kerberos, it provides an optional security interface to support secured file I/O and job submission.

Our work here aims to test these features to support multi-users mode on a shared cluster environment. Since Hadoop 1.0 was released on 27th Dec, 2011, the official website does not provide enough information/tutorial about the system configuration, we use the open source Cloudera's Distribution Including Apache Hadoop version 0.20u2 (CDH3u2) from Cloudera [1]. This CDH is currently the main track of developing Hadoop I/O security of Apache Hadoop project.

1. **Configuration**

This test was made on local nodes machine, the multiple nodes setting should be similar. There are several ways to setup the environment, mainly, it is related to the owner and starter of the hadoop daemons, table 1 shows the relationships between four groups of Linux/Unix and the normal hadoop daemons:

|  |  |  |
| --- | --- | --- |
|  | **HDFS** | **MapReduce** |
| **Users** | **NameNode** | **DataNode** | **2ndary Namnode** | **JobTracker**  | **TaskTracker** |
| hdfs | Owner & Starter | Owner | Owner & Starter |  |  |
| mapred |  |  |  | Owner & Starter | Owner & Starter |
| Root |  | starter |  |  |  |
| anyone with Kerberos authentication |  | Reader/ Writer |  | Job Submitter |  |

groupd on the groups the user belongs to at server side?Table 1. Relationship between Linux system users and hadoop services

As shown from table 1, hdfs and mapred are two Linux/Unix users that own and start the service. According to the Cloudera tutorial [2], it is more secured to have multiple level controls between HDFS and JobTracker. For our convenience, in our test, we merge these two accounts into only hdfs to start all the services. In addition, these starters/owners must be authenticated with Kerberos by creating a secured session with their Kerberos private keys loaded before starting any Hadoop daemons. As our experience, noted that, DataNode daemon requires root-level permission to be started; this is due to the hard-coded issue with the CDH3u2 package, this may be improved if we use Hadoop 1.0 or change their source code.

For other authenticated users (here, we mainly use user “mapred”) with the Kerberos database, they can read and write files to HDFS with their own namespaces and also submit job to the Hadoop JobTracker. Noted that, we loose the tasktracker security without using org.apache.hadoop.mapred.LinuxTaskController class as suggested from the tutorial (we got map tasks creation error even if we can submit the job). We will also discuss about this setting in the below session.

1. **Local / Remote HDFS Files I/O and MapReduce** **Job Submissions**

In our test, we tried to upload files from inside and outside of the Hadoop server. Table 2 shows the test cases:

|  |  |  |
| --- | --- | --- |
|  | HDFS Files I/O | MapReduce Job Submission |
| **Users** | Local | Remote (same host domain) | Remote (different host domain) | Local | Remote (same/diff host domain) |
| hdfs/(main/slave)  | Y | Y | Y | Y | Y |
| mapred/(main/slave) | Y | Y | Y | Y | Y |
| User w/o Kerberos authentication | N | N | N | N | N |

Table 2. Local / Remote Files I/O and job submission

So, as shown in table 2, we have tested the client side I/O from the local and remote side. We have generated four sets of Kerberos keys with format as username/hostDomain, e.g. hdfs/main, hdfs/slave, mapred/main and mapred/slave, whose Kerberos authentications are loaded to Hadoop Server when started. The domain of key is used for Kerberos security to identify the trusted machines; such keys can be created either on the server or remotely from client with admin privileges. Here, we assume these private keys are extracted and are safely sent to users. Authenticated user creates Kerberos secured session by loading the private key with using a command line tool. CDH3u2 Hadoop only allows operations performed by user with authorized session.

Rather than running Hadoop operations within the Hadoop cluster, as mentioned above, user can also do it remotely from other machine. Noted that, this remote machine must be able to communicate with our Kerberos server in order to load and match the user database.

There are two scenarios to remotely do HDFS file I/O and MapReduce Job submission:

* 1. User creates Kerberos session with different host domain
	2. User creates Kerberos session with same host domain

Scenario A is suggested by the Cloudera tutorial as it is secured and is less hassle with Kerberos; User on each machine creates individual Kerberos connection by loading an authenticated key with its own host domain. For Scenario B, although it’s doable, Kerberos KDC authentication server may reject the incoming requests. This is due to a security thread that many requests generated by same identity on numerous host domains (machines).

Now, we discuss about the DefaultTaskController and LinuxTaskController. These java classes are used to identify the user-level permission when a new map/reduce task are attempting to be created. MapReduce I/O operations includes the job staging cache files I/O, temporary intermediate files I/O, and any runtime I/Os will call this class to check whether the current job user has the right permissions to perform related actions. Here, DefaultTaskController has less restriction on these operations; LinuxTaskController strickly follows the ACLs for every task-level I/O related operations.

The previous local cache I/O permission error with using LinuxTaskController has been fixed by setting the mapred.local.dir to a specific directory on /tmp/hadoop-share/mapred/local (written in both mapred-site.xml and taskcontroller.cfg).

1. **File system Features**

After experiencing the CHD3u2, we summarize its file system features as following:

* Multi-users

It fully supports users in a shared cluster environment, as long as the users are authenticated with both Linux (normal system setup with LDAP) and Kerberos user space. We didn’t merge the public/private keys between Linux and Kerberos, so, if we need to go further, we need expertise from other system admins.

* ACL

It’s the same as normal Linux ACL file control, but the user must be existed on both Linux and Kerberos database.

* Disk quota

N/A for HDFS, but it is limited with the HDFS owner disk quota.

1. **Drawback**

We discuss some drawbacks from the experience we had:

* In secured Kerberos public/private keys

As mentioned above, the Kerberos public/private keys cannot be created with password protection (in our test, it’s a random key); this is due to the CDH source code does not provide this function. If someone’s Kerberos private are stolen, it’s easy to pretend as the stolen users.

* No real time users update mechanism

If there is any added new users, the current Hadoop System need to be fully restarted.

1. **Related system services**

DNS, Kerberos, LDAP

1. **Conclusion**

Overall, the multi-user mode works and can support Kerberos authorization and authentication, but there are few hard coded issues which need root level permission to edit/load files before/during starting up the services. In addition, this approach still has room for improvement before deploying on a larger environment, such as adding password protection to Kerberos keys, real time or periodic user update, etc. We suggest to run this on a small environment (2 ~ 4 nodes) on a real cluster such as Alamo or India.

**Reference**s

1. Cloudera official website: <https://www.cloudera.com/>
2. Cloudera CHD3 tutorial: [https://ccp.cloudera.com/display/CDHDOC/Configuring+Hadoop+Security+in+CDH3](https://ccp.cloudera.com/display/CDHDOC/Configuring%2BHadoop%2BSecurity%2Bin%2BCDH3)