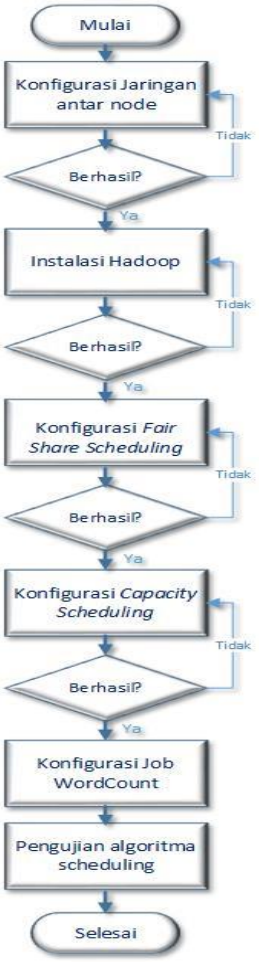
BAB 4 IMPLEMENTASI DAN PENGUJIAN

Pada bab ini akan membahas tentang tahapan-tahapan implementasi pada sistem Hadoop untuk melakukan pengujian terhadap kinerja algoritme *Fair Share Scheduling* dan *Capacity Scheduling*. Langkah-langkah tersebut akan mengacu pada perangkat keras serta perangka lunak yang akan digunakan.

4.1 Implementasi

Pada bab ini akan menjelaskan secara umum proses-proses konfigurasi dam instalasi yang akan dilakukan dalam proses analisis kinerja algoritme *fair scheduling* dengan *Capacity Scheduling* pada Hadoop *cluster multinode* menggunakan parameter *Job Fail Rate, Latency, dan Throughput*sesuai dengan lingkungan perancangan sistem yang diinginkan. Berikut diagram alir dari proses implementasi yang akan dilakukan sebagai berikut.



Gambar 4.1 Diagram Alir Tahap Implementasi

4.2 Konfigurasi Jaringan

Pada bab ini akan dilakukan konfigurasi jaringan sesuai dengan topologi yang telah dibuat pada gambar 3.2. Berikut diagram alir dari proses untuk melakukan konfigurasi jaringan pada Hadoop yang akan dilakukan sebagai berikut.

Gambar 4.2 Diagram Alir Konfigurasi Jaringan

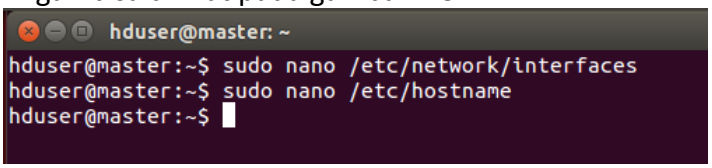
Pada penelitian ini perlu adanya spesifikasi sistem yang dibutuhkan untuk menunjang proses implementasi. Spesifikasi sistem ini meliputi spesifikasi dari komputer *master* maupun komputer *slave*. Berikut adalah rincian dari perangkat yang dibutuhkan yaitu.

Tabel 4.1 Alamat IP pada *Node* Komputer

Node Komputer	RAM	Alamat IP
Master	1 GB	192.168.56.10
Node1	512 MB	192.168.56.11
Node2	513 MB	192.168.56.12
Node3	514 MB	192.168.56.13
Node4	515 MB	192.168.56.14
Node5	516 MB	192.168.56.15

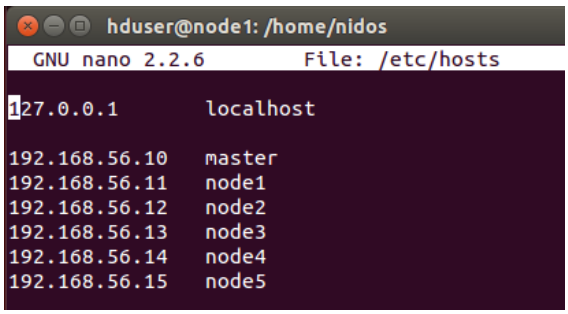
Pada sistem Hadoop ini menggunakan satu komputer *head master* yang terdiri dari enam komputer *virtual* yang digunakan sebagai 1 komputer *master* dan 5 komputer *slave*. Pada setiap *node* perlu dilakukan konfigurasi *network interfaces*

dan hostname. Pada konfigurasi network dilakukan pada perintah “sudo nano /etc/network/interfaces”. Perintah tersebut digunakan untuk konfigurasi alamat IP menggunakan pengalamatan network *static*. Pada pengalamatan IP eth1 dilakukan pada semua keenam *node*. Sedangkan untuk konfigurasi hostname dilakukan pada perintah “sudo nano /etc/hostname”. Pada perintah ini digunakan untuk memudahkan user dalam memberi nama pada sebuah *node*. Perintah konfigurasi jaringan bisa dilihat pada gambar 4.3.



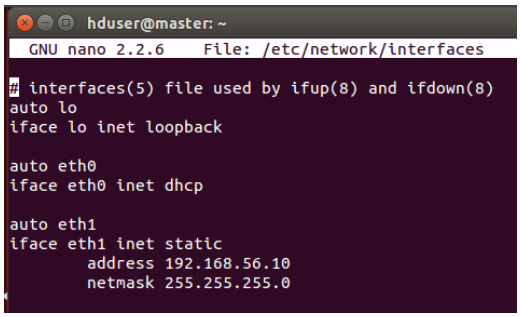
Gambar 4.3 Konfigurasi Jaringan

Setelah melakukan konfigurasi *network interfaces* dan hostname, selanjutnya lakukan konfigurasi hosts pada semua *node* dengan melakukan perintah “sudo nano /etc/hosts”. Pada perintah ini berisi alamat IP dan hostname dari semua *node* dalam satu *cluster*. Untuk mengetahui lebih jelas bisa dilihat pada gambar 4.4.



Gambar 4.4 Konfigurasi *Host*

Pada setiap *node* akan dilakukan konfigurasi *network interfaces* dan hostname. Pada komputer 1 dilakukan konfigurasi dengan mengisi alamat IP pada eth1 dalam kondisi *static*. Alamat IP yang digunakan adalah 192.168.56.10, *netmask* 255.255.255.0. Pengaturan *network interfaces* dapat dilihat pada gambar 4.5.

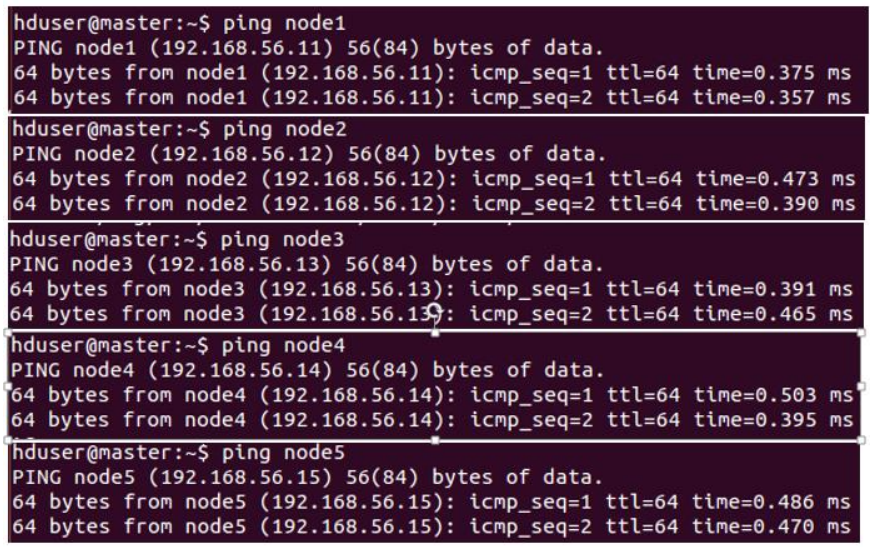


Gambar 4.5 Konfigurasi *Network Interfaces* eth1 pada *Master*

Setelah melakukan konfigurasi *network interfaces* kemudian pada komputer 1 akan dilakukan konfigurasi hostname dengan nama “*master*” dengan melakuakan perintah “sudo nano /etc/hostname”.

Setelah melakukan konfigurasi jaringan pada komputer *master* dengan perintah “sudo nano /etc/network/interfaces”, sudo nano /etc/hostname”, dan “sudo nano /etc/hosts”. kemudian lakukan juga perintah tersebut ke semua komputer *slave* namun dengan menggunakan alamat ip yang berbeda-beda.

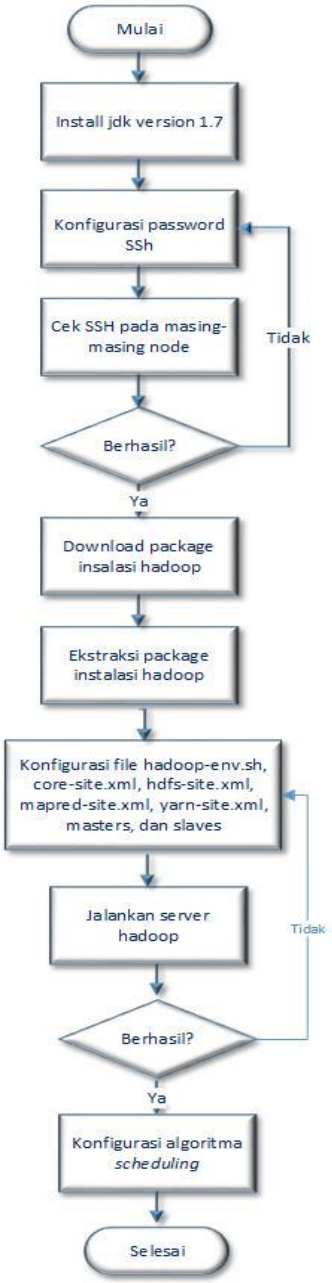
Setelah itu lakukan cek koneksi dengan melakukan perintah “ping” pada komputer *master* ke 5 *node*. Pada perintah ini digunkan untuk memastikan bahwa pada setiap *node* sudah dapat terhubung atau belum. Pada proses cek koneksi ini dapat dilihat pada gambar 4.6.



Gambar 4.6 Cek Koneksi *Master* ke Komputer *Slave*

4.3 Instalasi Hadoop

Pada tahap ini akan menjelaskan secara umum proses instalasi dan konfigurasi Hadoop sistem sebagai wadah untuk melakukan analisis dari kinerja algoritme penjadwalan dengan menggunakan parameter *Job Fail Rate, Latency, dan Throughput* sesuai dengan lingkungan perancangan sistem yang diinginkan. Sebelum melakukan instalasi Hadoop terlebih dahulu siapkan virtual machine sebagai sistem operasi, disini penulis menggunakan virtualBox dengan versi 5.1.10. Berikut diagram alir dari proses instalasi Hadoop yang akan dilakukan yaitu, pada gambar 4.7.



Gambar 4.7 Diagram Alir Proses Instalasi Hadoop

4.3.1 Instalasi Hadoop Multi*node* C*luster*

Pada Hadoop *Multinode* ini menggunakan komputer *master* dan komputer *slave*. Pertama lakukan instalasi Hadoop kedalam komputer *master*. Setelah Hadoop telah terinstal kedalam komputer *master* lalu lakukan cloning ke 5 komputer *slave* dengan melakukan sedikit perubahan dalam beberapa berkas yang tersedia. Pada proses ini menggunakan *virtual machine* Ubuntu versi 14.04.1 sebagai sistem operasinya. Lalu melakukan instalasi serta konfigurasi pada Hadoop versi 2.7.3 sesuai dengan lingkungan perancangan yang diinginkan dengan cara sebagai berikut.

1.	sudo apt-get update
2.	sudo apt-get install default-jdk (cek dengan java -version)
3.	sudo addgroup Hadoop
4.	sudo adduser –ingroup Hadoop hduser
5.	sudo adduser hduser sudo
6.	sudo apt-get install ssh
7.	su hduser
8.	ssh-keygen -t rsa -P ""
9.	Ketikkan “cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys”
10.	wget http://mirror.wanxp.id/apache/Hadoop/common/Hadoop-2.7.3/Hadoop-2.7.3.tar.gz
11.	sudo tar xvzf Hadoop-2.7.3.tar.gz

Setelah berhasil melakukan proses instalasi Hadoop versi 2.7.3 pada komputer *master*, selanjutnya lakukan beberapa proses konfigurasi *file* Hadoop dengan cara sebagai berikut.

1. Memperbaruhi beberapa variabel *export* dengan printah “sudo nano ~/.bashrc”dengan menggunakan editor nano tambahkan beberapa perintah *export* pada baris terakhir di *file* ~/.bashrc. Tujuan penambahan path tersebut digunakan untuk mempermudah ketika melakukan perintah terkait dengan Hadoop dan java pada terminal. Beberapa *path* yang perlu ditambahkan adalah sebagai berikut.

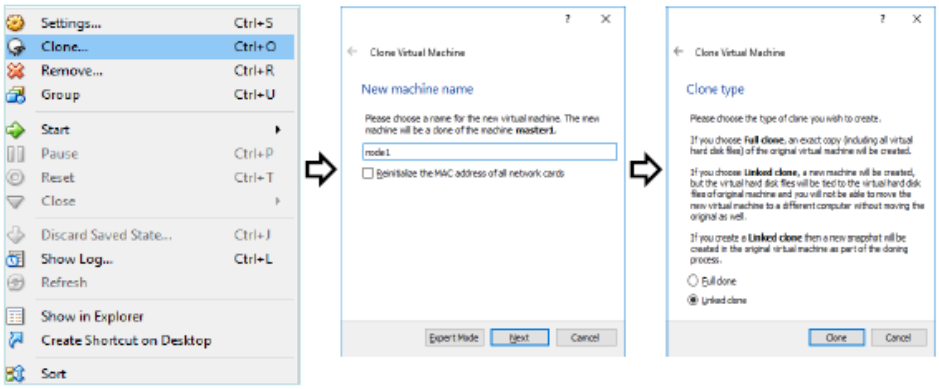
```
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export HADOOP_INSTALL=/usr/local/Hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
export HADOOP_CLASSPATH=$JAVA_HOME/lib/tools.jar
export HADOOP_CLASSPATH=/usr/lib/jvm/java-7-openjdk-amd/lib/tools.jar
```

2. Ketik perintah “source ~/.bashrc”. Perintah ini digunakan untuk menerapkan perubahan pada *file* .bashrc yang telah dilakukan sebelumnya.
3. Kemudian ketik “sudo nano /usr/local/Hadoop/etc/Hadoop/Hadoop-env.sh”. Pada perintah tersebut ubah baris *file* export JAVA_HOME=” ” menjadi export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
4. Setelah itu ketik perintah “sudo nano /usr/local/Hadoop/etc/Hadoop/core-site.xml” dan tambahkan konfigurasi pada isi *file* pada lampiran B.
5. Kemudian lakukan *copy file* mapred-site dengan perintah “cp /usr/local/Hadoop/etc/Hadoop/mapred-site.xml.template /usr/local/Hadoop/etc/Hadoop/mapred-site.xml”

6. Selanjutnya pada konfigurasi MapReduce yang digunakan pada *file* mapred-site.xml pada perintah "sudo nano /usr/local/Hadoop/etc/Hadoop/mapred-site.xml". Pada konfigurasi MapReduce yang digunakan adalah YARN. Adapun beberapa konfigurasi *file* mapred-site.xml dapat dilihat pada lampiran B.
7. Kemudian buka *file* hdfs-site.xml dengan perintah "sudo nano /usr/local/Hadoop/etc/Hadoop/hdfs-site.xml. pada konfigurasi ini menunjukkan banyaknya replikasi data pada HDFS, tambahkan beberapa baris konfigurasi *file* bisa dilihat pada lampiran B.
8. Kemudian buka *file* yarn-site.xml lakukan pada perintah "sudo nano /usr/local/Hadoop/etc/Hadoop/yarn-site.xml" dan tambahkan beberapa baris konfigurasi *file* bisa dilihat pada lampiran B.

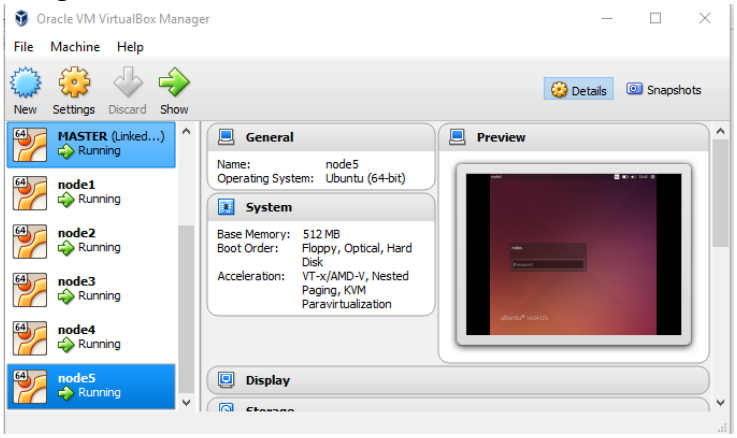
Setelah melakukan konfigurasi Hadoop pada komputer *master*, kemudian lakukan cloning pada PC *node*1, Pc *node*2, PC *node*3, PC *node*4, PC *node*5 dengan cara clone dari PC *master*.

Lakukan hal berikut (shutdown komputer *master*, kemudian klik kanan, klik *Clone*, berinama *node*1, klik Next, Pilih *Linked*, Klik *Clone*):



Gambar 4.8 *Cloning Master* Hadoop

Setelah komputer *slave* berhasil di *cloning* melalui komputer *master*, kemudian klik tombol "start" pada kelima *node* tersebut untuk menjalankan *node* tersebut seperti gambar 4.9.



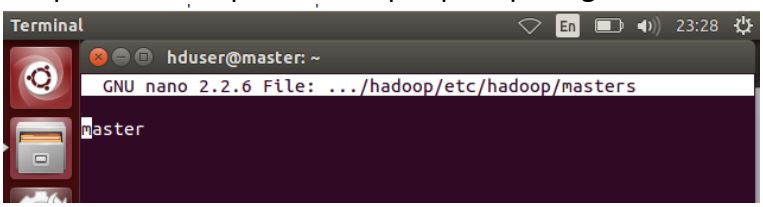
Gambar 4.9 Jalankan Semua *Node*

4.3.2 Konfigurasi Komputer *Master*

Setelah melakukan instalasi Hadoop pada komputer *master* dan melakukan konfigurasi jaringan pada semua *node cluster* hingga terhubung. Selanjutnya pada tahap ini akan dilakukan perubahan konfigurasi pada Hadoop multi*node cluster*. Pada sub bab ini akan dilakukan konfigurasi komputer *master* yang bertujuan untuk memperkenalkan pada seluruh *node cluster* bahwa ini merupakan komputer *master* yang merupakan pusat dari server Hadoop. Pada tahap ini akan dilakukan modifikasi isi baris pada beberapa *file*. Pada komputer *master* lakukan pengaturan untuk membuat direktori Hadoop_tmp seperti pada perintah dibawah ini. Pada langkah pertama hapus direktori Hadoop_tmp, setelah itu pada langkah kedua buat folder name*node* pada direktori Hadoop_tmp, selanjutnya pada langkah ketiga dengan menggunakan perintah chown hduser yang berarti bahwa merubah kepemilikan folder name*node* pada user hduser.

```
1.  sudo rm -rf /usr/local/Hadoop_tmp/
2.  sudo mkdir -p /usr/local/Hadoop_tmp/hdfs/namenode
3.  sudo chown -R hduser /usr/local/Hadoop_tmp
```

Kemudian setelah membuat folder name*node* lakukan konfigurasi *file master*s dengan mengisi nama "*master*" sebagai inisialisasi bahwa ini merupakan komputer *master* pada Hadoop seperti pada gambar 4.10.



Gambar 4.10 Inisialisasi *File* M*asters*

Setelah melakukan inisialisi pada *file master*s, kemudian lakukan perubahan konfigurasi beberapa isi baris pada *file* Hadoop yaitu, pada *file* hdfs-site.xml. Pada *file* hdfs-site ini perubahan yang dilakukan pada jumlah replica dfs yang berjumlah 5 dan nama direktori dfs pada folder name*node* seperti pada lampiran B.

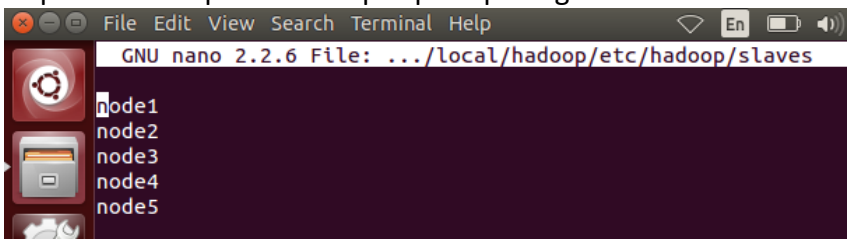
4.3.3 Konfigurasi Komputer *Slave*

Pada sub bab ini akan dilakukan konfigurasi komputer *slave* yang bertujuan untuk memperkenalkan pada komputer *master* bahwa pada komputer ini akan bertindak sebagai *slave* pada Hadoop *cluster*. Pada komputer *slave* ini berjumlah lima buah yaitu, (*node*1, *node*2, *node*3, *node*4, *node*5). Pada tahap ini akan dilakukan modifikasi isi baris pada beberapa *file*. Pada komputer *slave* lakukan pengaturan untuk membuat direktori Hadoop_tmp seperti pada perintah dibawah ini. Pada langkah pertama hapus direktori Hadoop_tmp, setelah itu pada langkah kedua buat folder data*node* pada direktori Hadoop_tmp, selanjutnya pada langkah ketiga dengan menggunakan perintah chown hduser yang berarti bahwa merubah kepemilikan folder data*node* pada user hduser.

```
1.  sudo rm -rf /usr/local/Hadoop_tmp/
```

2.	sudo mkdir -p /usr/local/Hadoop_tmp/hdfs/data*node*
3.	sudo chown -R hduser /usr/local/Hadoop_tmp

Kemudian pada komputer *slave* lakukan konfigurasi *file slave*s dengan mengisi nama *node* pada komputer *slave* sebagai inisialisasi bahwa ini merupakan daftar komputer *slave*s pada Hadoop seperti pada gambar 4.11.



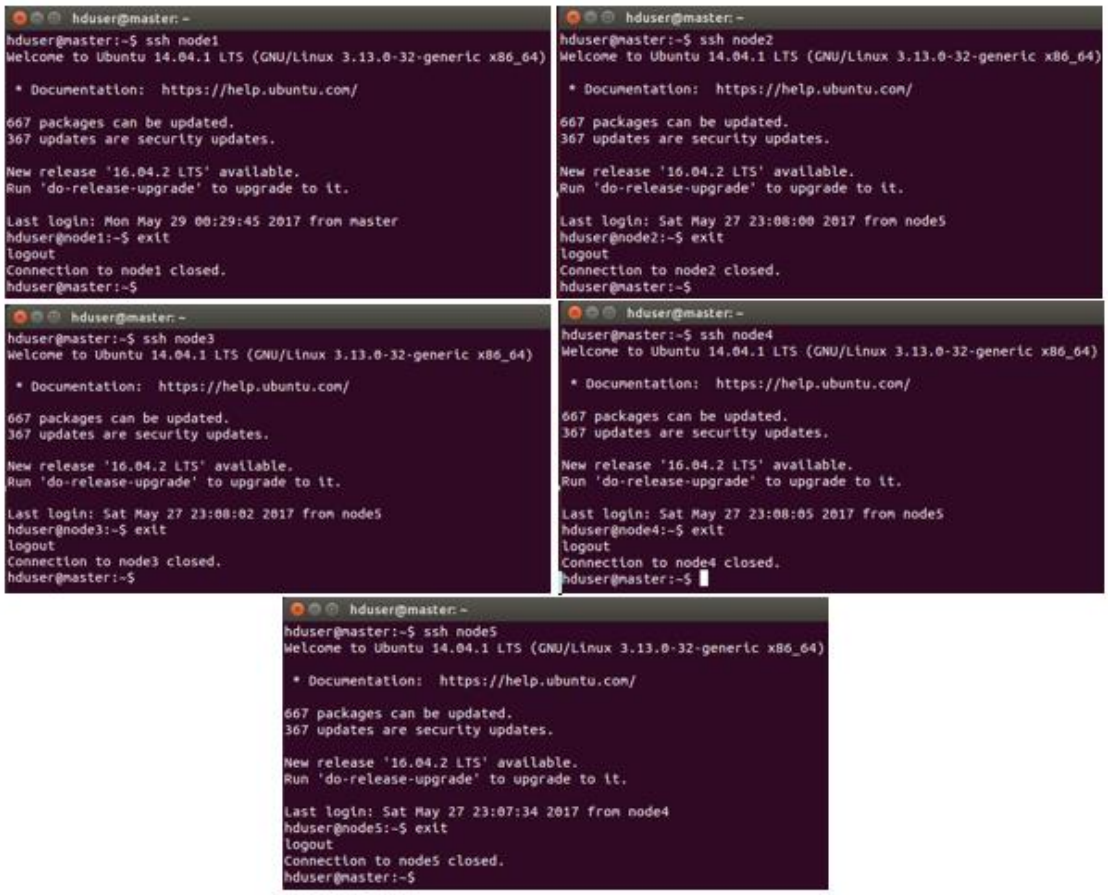
Gambar 4.11 Inisialisasi *File Slaves*

Setelah melakukan inisialisi pada *file slave*s, kemudian lakukan perubahan konfigurasi beberapa isi baris pada *file* Hadoop yaitu, pada *file* hdfs-site.xml. Pada *file* hdfs-site ini perubahan yang dilakukan pada jumlah replica dfs yang berjumlah 5 dan nama direktori dfs pada folder data*node* seperti pada lampiran B.

Setelah berhasil melakukan konfigurasi beberapa *file* Hadoop pada komputer *slave*s. Pada langkah konfigurasi ini dilakukan pada semua komputer *slave*s yang terdapat pada *cluster* Hadoop yaitu, pada komputer *node*1, *node*2, *node*3, *node*4, dan *node*5.

4.3.4 Menjalankan Perintah SSH

Kemudian pada langkah berikutnya lakukan perintah SSH atau Secure Shell merupakan protokol jaringan yang digunakan untuk pengguna membuka akses pada komputer lokal yang terhubung pada komputer server. Pada komputer *master* melakukan call ssh ke semua komputer *slave* dengan melakukan perintah seperti pada gambar 4.12.



Gambar 4.12 Melakukan *Call SSH* pada Komputer *Master*

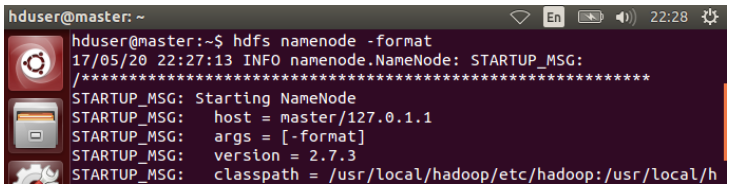
Jika pada saat melakukan call ssh terjadi error seperti pesan “ssh: connect to host *node*2 port 22: No route to host” maka solusinya bisa melakukan cek status ssh dengan menggunakan perintah “sudo service ssh status” kemudian akan muncul pesan OK jika ssh start/running, process 790. Jika muncul pesan ERROR maka bisa perintah “sudo apt-get remove openssh-*slave* openssh-server” lalu ketik perintah “sudo apt-get install openssh-*slave* openssh-server”. Setelah itu ketik perintah “sudo service status”, maka akan muncul pesan sukses ssh start/running. Process 3084 yang menunjukkan bahwa *master* telah berhasil melakukan call ssh ke komputer *slave*.

Kemudian pada setiap komputer *slave* juga melakukan lakukan call ssh ke komputer *master* dan komputer *slave* lainnya seperti. Pada komputer *slave* (*node*1) akan melakuakn call ssh ke komputer *master* dan komputer *slave* lainnya yaitu, (*node*2, *node*3, *node*4, *node*5) dengan melakukan perintah seperti pada komputer master.

Setelah *node*1 berhasil melakukan call ssh ke *node* lainnya, maka selanjutnya lakukan juga perintah tersebut untuk melakuka call ssh pada komputer *node*2, *node*3, *node*4, dan *node*5 seperti pada komputer *node*1 diatas.

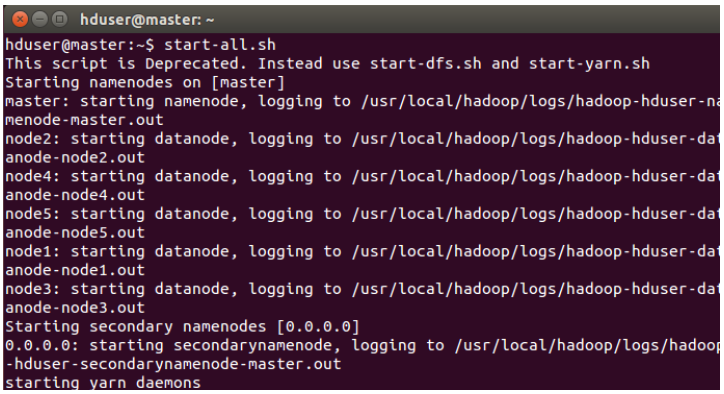
4.3.5 Menjalankan dan Menghentikan Hadoop

Setelah berhasil melakukan konfigurasi pada Hadoop *cluster* multi*node*. Selanjutnya sebelum menjalankan perintah Hadoop lakukan format name*node* dengan perintah "hdfs name*node* –format" seperti pada gambar 4.13.



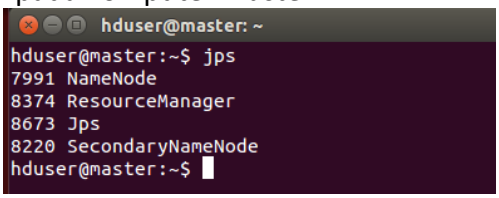
Gambar 4.13 Format Name*node*

Setelah berhasil melakukan format name*node*, kemudian jalankan Hadoop dengan melakukan perintah "start-all.sh" dari komputer *master* seperti pada gambar 4.14. Pada perintah ini digunakan untuk menjalankan perintah yarn.sh dan dfs.sh sekaligus.



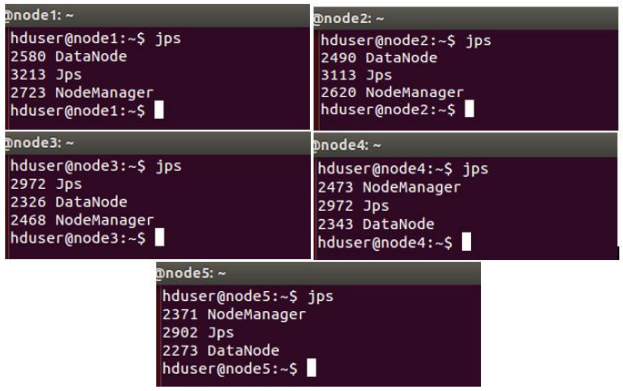
Gambar 4.14 Perintah Untuk Menjalankan Hadoop

Selanjutnya setelah Hadoop telah berhasil dijalankan kemudian untuk mengetahui informasi mengenai proses-proses Hadoop apa saja yang telah berjalan dapat diketahui dengan menggetik perintah "jps" Pada gambar 4.15 perintah jps dilakukan pada komputer *master*.



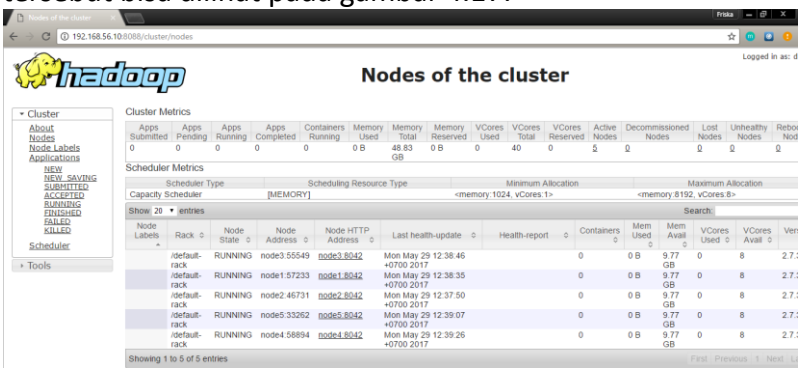
Gambar 4.15 Cek JPS pada Komputer *Master*

Sedangkan pada kompter *slave* untuk mengetahui informasi proses Hadoop yang telah berjalan dapat dilakukan pada perintah jps. Perintah ini lakukan pada komputer *node*1, *node*2, *node*3, *node*4, dan *node*5. Pada perintah ini menunjukkan bahwa pada komputer *slave* proses yang berjalan adalah Data*node*, jps, dan *Node*Manager. Perintah ini dapat dilihat pada gambar 4.16.



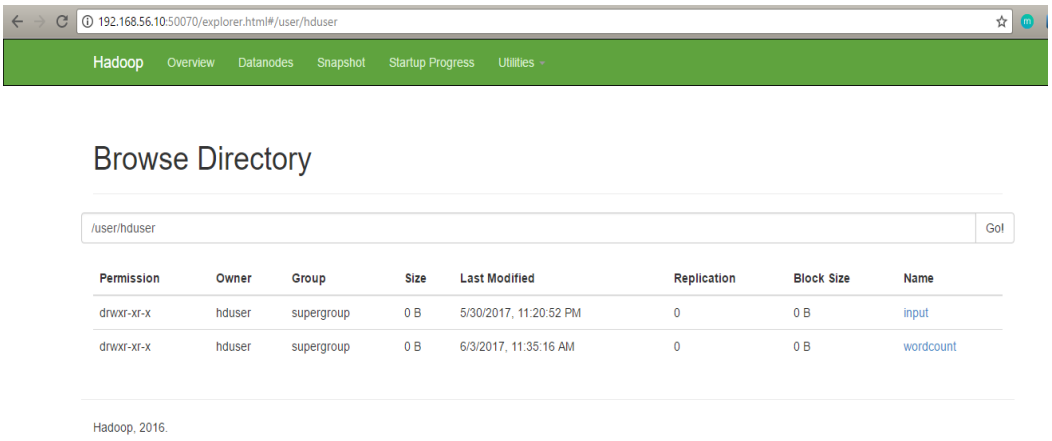
Gambar 4.16 Cek JPS pada Komputer *Slave*

Setelah Hadoop berhasil dijalankan kemudian untuk melakukan monitoring pada kinerja Hadoop dapat dilihat dari web UI Hadoop. Untuk melihat web UI Hadoop dapat diakses pada http://192.168.56.10:8088 untuk melihat *Resource* Manager. Pada antarmuka ini kita bisa mengetahui *cluster* metric dan tools. Pada *cluster* ini kita bisa melihat scheduler apa yang telah berjalan pada sistem Hadoop, pada scheduler ini yang digunakan untuk melakukan monitoring kinerja *scheduler* berdasarkan pada pengiriman *job* yang dilakukan oleh user. Informasi tersebut bisa dilihat pada gambar 4.17.



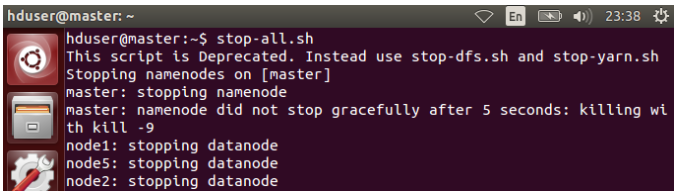
Gambar 4.17 Antarmuka Hadoop pada Resource Manager

Untuk melihat name*node* manager pada web UI Hadoop dapat diakses pada alamat http://192.168.56.10:50070. Pada web UI ini terdapat keterangan informasi *master* yang telah aktif. Setelah berhasil menjalankan service HDFS, selanjutnya pada direktori HDFS ini kita bisa membuat beberapa *file* penyimpanan data yang akan diproses pada sistem Hadoop. Pada pembuatan direktori tersebut bisa dilakukan dengan menggunakan perintah "bin/hdfs dfs -mkdir -p /user/hduser". Pada direktori ini berisi input data *file* yang akan diproses dan output hasil dari eksekusi perintah *job* yang telah berhasil dijalankan. Untuk melakukan cek direktori HDFS dapat dilihat pada tab "Utilities", kemudian pilih "Browse the *File* System" seperti pada gambar 4.18.



Gambar 4.18 Antarmuka Hadoop pada Direktori HDFS

Setelah melakukan semua proses pada Hadoop, untuk menghentikan Hadoop dapat dilakukan dengan mematikan proses yarn dan dfs dengan mengetik perintah "stop-all.sh". Hasil dari menghentikan proses HDFS dan YARN dapat dilihat pada gambar 4.19.



Gambar 4.19 Menghentikan Proses Hadoop

4.4 Konfigurasi *Fair Share Scheduler*

Untuk menerapkan *Fair Share Scheduling*, administrator perlu melakukan konfigurasi pada *file* yarn-site.xml dan fair-scheduler.xml untuk menempatkan *job* yang diajukan kedalam antrian sesuai dengan pengujian yang dilakukan. Untuk melakukan konfigurasi *file* tersebut perlu dilakukan penambahan beberapa parameter *property.* Berikut diagram alir dari proses untuk melakukan konfigurasi algoritme *Fair Share Scheduling* pada Hadoop yang akan dilakukan sebagai berikut.

Gambar 4.20 Diagram Alir Konfigurasi *Fair Share Scheduling*

4.4.1 Instalasi *Fair Share Scheduling*

Untuk dapat menjalankan algoritme *Fair Share Scheduling* pada sistem Hadoop, perlu dilakukan intalasi pada *file* yarn-site.xml dengan menambahkan parameter "yarn.*resource*manager.scheduler.class" kedalam *file* konfigurasi untuk mengelola scheduler Hadoop pada *resource* manager. Secara *default* pada Hadoop versi 2 menggunakan *Capacity Scheduling*, namun administrator dapat mengubah nilai parameter ini kedalam fair scheduler. Konfigurasi parameter ini bisa dilihat sebagai berikut.

```
<property>
<name>yarn.resourcemanager.scheduler.class</name>
<value>org.apache.Hadoop.yarn.server.resourcemanager.scheduler.fair.FairSc
heduler</value>
</property>
```

Selanjutnya untuk mengatur pada parameter "yarn.scheduler.fair. allocation.*file*" digunakan untuk menentukan lokasi *file* yang telah dikonfigurasi oleh administrator pada *file* fair-scheduler.xml. Untuk mengetahui parameter konfigurasi ini bisa dilihat sebagai berikut.

```
<Property>
    <name> yarn.scheduler.fair.allocation.file </ name>
  <value>/usr/local/Hadoop/etc/Hadoop/fair-scheduler.xm</value> </property>
```

Selain melakukan konfigurasi diatas, administrator juga bisa melakukan control lebih terhadap *property* pada *file* yarn-site.xml. Misalnya untuk menambahkan *control* lebih terhadap penugasan container bisa dilihat lebih lanjut pada konfigurasi parameter pada yarn Hadoop.

4.4.2 Membuat Daftar Antrian

Selain melakukan konfigurasi pada *file* yarn-site.xml, administrator juga melakukan konfigurasi pada *file* fair-scheduler.xml. Pada konfigurasi *file* fair-scheduler.xml terdapat beberapa parameter yang digunakan untuk mengatur elemen penyusun pada setiap antrian. Pada antrian *pool1* merupakan antrian induk dan terdapat beberapa *property* penyusun yang terdiri dari min*Resource*s, max*Resource*s, maxRunningApps, aclSubmitApps, weight, dan schedulingPolicy. Pada setiap antrian terdapat nilai yang ditentukan oleh administrator sesuai dengan pengujian yang digunakan. Berikut adalah elemen konfigurasi yang digunakan pada antrian pool1 sebagai berikut.

```
<queue name="pool1">
    <minResources>2000 mb, 1 vcores</minResources>
    <maxResources>5000 mb, 1 vcores</maxResources>
    <maxRunningApps>10</maxRunningApps>
    <aclSubmitApps>*</aclSubmitApps>
   <weight>2.0</weight>
   <schedulingPolicy>fair</schedulingPolicy>
```

Setelah membuat antrian induk pada pool1 beserta elemen-elemen penyusunnya, selanjutnya pada antrian pool1 terdapat sub organisasi yang terdiri dari antrian pool1a dan antrian pool1b. Pada kedua antrian sub organisasi tersebut terdaoat beberapa komponen penyusun elemen-elemen *property* yang dibutuhkan untuk mengatur jalannya *job* yang terdiri dari *queue* name, aclSubmitApps, dan min*Resource*s. Berikut adalah elemen konfigurasi yang digunakan pada antrian pool1 sebagai berikut.

```
<queue name="pool1">
  <minResources>5000 mb,4vcores</minResources>
  <maxResources>30000 mb,4vcores</maxResources>
  <maxRunningApps>50</maxRunningApps>
  <weight>1.0</weight>
  <schedulingPolicy>fair</schedulingPolicy>
  <aclSubmitApps>*</aclSubmitApps>
  <aclAdministerApps>*</aclAdministerApps>
  <minSharePreemptionTimeout>1</minSharePreemptionTimeout> </queue>
```

4.4.3 Konfigurasi Alokasi *File*

Setelah melakukan konfigurasi pada setiap antrian pada fair scheduler, selanjutnya lakukan konfigurasi user elemen yang digunakan untuk mengatur user yang dapat menjalankan *job* pada suatu pool. Pada elemen ini berisi komponen userMaxAppsDefault, fairSharePreemption*Timeout*, dan default*Queue*Scheduling Policy. Berikut adalah elemen konfigurasi yang digunakan untuk menentukan elemen *use*r sebagai berikut.

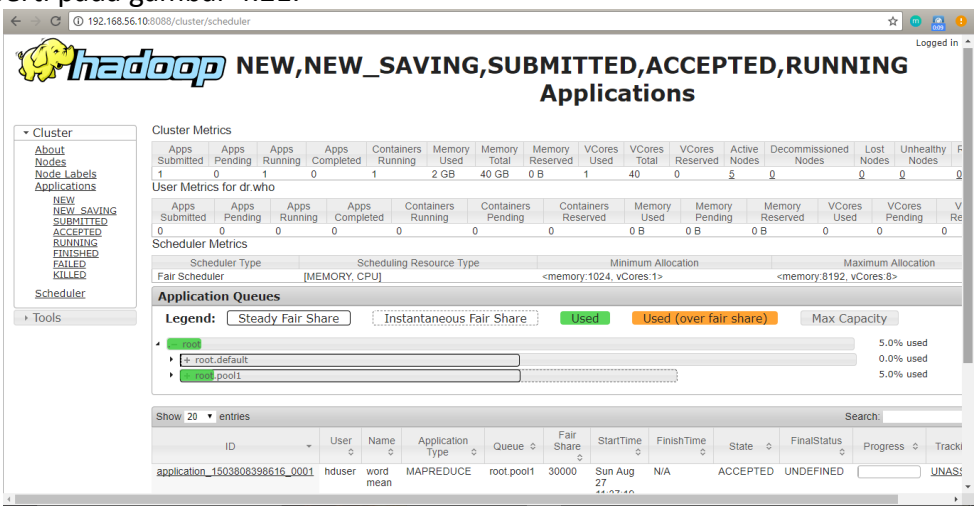
```
<userMaxAppsDefault>3</userMaxAppsDefault>
 <fairSharePreemptionTimeout>1</fairSharePreemptionTimeout>
 <defaultQueueSchedulingPolicy>fair</defaultQueueSchedulingPolicy>
```

Setelah melakukan konfigurasi pada antrian, *Fair Share Scheduling* terdapat aturan-aturan yang digunakan untuk memberitahu *scheduler* tentang bagaimana menempatkan sebuah *job* yang akan masuk kedalam antrian. Atruran ini ditentukan oleh konfigurasi pada *queue*PlacementPolicy. Dalam aturan ini terdapat argument rule name dimana sebuah *job* akan ditempatkan dalam antrian.Secara default argumen rule name ini akan ditempatkan di pool1, Aturan-aturan tersebut berisi komponen dibawah ini :

```
<queuePlacementPolicy>
  <rule name="default" queue="pool1"/>
 </queuePlacementPolicy> </allocations>
```

4.4.4 Melihat Daftar Antrian

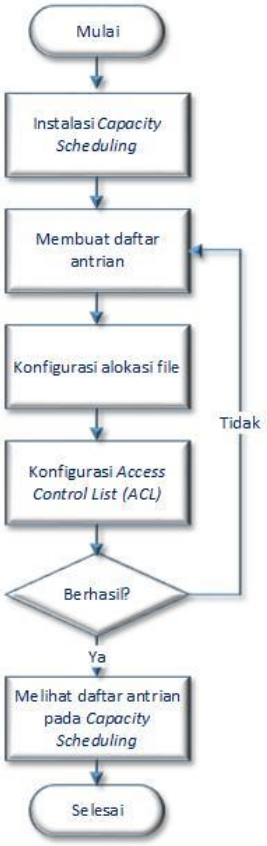
Pada *Fair Share Scheduling* untuk melihat informasi pada daftar antrian bisa dilakukan dengan membuka antarmuka dari YARN *resource*manager. Untuk membuka antrarmuka tersebut dengan cara masuk ke link <*resource*manager-hostname>:8088. Berikut adalah tampilan antarmuka dengan *fair scheduling* seperti pada gambar 4.21.



Gambar 4.21 Daftar Antrian *Fair Share Scheduling* pada Resource Manager

4.5 Konfigurasi *Capacity Scheduling*

Dalam melakukan konfigurasi pada *Capacity Scheduling*, administrator perlu melakukan pengaturan konfigurasi pada *file* yarn-site.xml dan capacity-scheduler.xml. Untuk melakukan konfigurasi pada *file* tersebut perlu melakukan edit pada beberapa parameter *property* sesuai dengan lingkungan pengujian yang diinginkan. Berikut diagram alir dari proses untuk melakukan konfigurasi algoritme *Capacity Scheduling* pada Hadoop yang akan dilakukan sebagai berikut.



Gambar 4.22 Diagram Alir konfigurasi *Capacity Scheduling*

4.5.1 Instalasi *Capacity Scheduling*

Sebelum mengatur antrian dalam *Capacity Scheduling*, pertama-tama lakukan konfigurasi pada *file* yarn-site.xml untuk mengatur jumlah memori maksimum yang akan digunakan pada *node*manager YARN. Pada penelitian ini menggunkan jumlah memori sebanyak 10GB untuk utilisasi pengelola *node* manager YARN pada setiap komputer *slave*. Konfigurasi parameter yang perlu diedit seperti pada *property* dibawah ini.

```
<property>
    <name> yarn.nodemanager.resource.memory-mb </name>
    <value> 10000 </value>  </property>
```

Setelah itu untuk memastikan YARN menggunakan *Capacity Scheduling*, pada komputer *master* seorang administrator harus menambahkan parameter "yarn.*resource*manager.scheduler.class". Namun secara default pada Hadoop versi 2 telah menggunakan *Capacity Scheduling*, jadi bisa langsung menentukan antrian pada *cluster*.

```
<property>
    <name> yarn.resourcemanager.scheduler.class </name>
    <value>org.apache.Hadoop.yarn.server.resourcemanager.scheduler.capacit
y.CapacityScheduler </value>
```

```
</property>
```

Setelah melakukan konfigurasi pada *file* yarn-site.xml dan nilai yang diinginkan telah ditentukan oleh administrator, maka perlu melakukan restart pada layanan YARN agar perubahan yang terjadi bisa aktifkan.

4.5.2 Membuat Daftar Antrian

Selain melakukan konfigurasi pada *file* yarn-site.xml, administrator juga melakukan konfigurasi pada *file* capacity-scheduler.xml. Administrator perlu menentukan struktur antrian yang digunakan pada pengujian untuk pengiriman *job* . Pada *Capacity Scheduling* memiliki fitur antrian hierarki yang diatur dalam antrian root. Semua antrian dalam sistem adalah sub organisasi dari antrian root yang bisa mengeksekusi *job* . Untuk membuat antrian organisasi ini menggunakan parameter "yarn.scheduler.capacity.root.*queue*s". Pada penelitian ini menggunakan 1 buah antrian organisasi dengan nama *queue*1. Pada konfigurasi antrian ini bisa dilihat sebagai berikut.

```
<property>
    <name>yarn.scheduler.capacity.root.queues</name>
    <value>queue1 </value>
 </property>
```

Setelah membuat antrian, administrator bisa melakukan konfigurasi pada struktur antrian yang lebih spesifik. Untuk membuat antrian lebih spesifik menggunakan sintak yang menunjukkan parameter apa saja yang perlu dikonfgurasikan. Berikut adalah contoh penulisan sintak yang umum digunakan untuk melakukan konfigurasi pada *file* capacity-scheduler.xml yaitu, "yarn.scheduler.capacity.<*queue*-path>.<parameter>" dimana :
<*queue*-path> : digunakan untuk mengidentifikasi nama antrian
<parameter> : digunakan untuk mengidentifikasi parameter yang nilainya ditetapkan.

4.5.3 Konfigurasi Alokasi *File*

Setelah membuat sebuah antrian, kemudian pada antrian tersebut terdapat presentasi kapasitas *resource* yang harus dialokasikan. Untuk melakukan konfigurasi kapasitas *resource* pada setiap antrian, administrator perlu mengedit nilai parameter dari "yarn.scheduler.capacity.root.<*queue*-path>.capacity". Pada penelitian ini terdapat satu antrian organisasi yaitu, *queue*1. Pada *queue*1 menggunakan kapasitas *resource* sebanyak 100% dari total keseluruhan *resource* yang dialokasikan sebelumnya yang ditetapkan oleh *node*manager pada "yarn.*node*manager.*resource*.memory-mb" pada *file* yarn-site.xml. Untuk konfigurasi presentasi *resource* pada antrian bisa dilihat sebagai berikut.

```
<property>
   <name>yarn.scheduler.capacity.root.queue1.capacity</name>
   <value>100</value>
  </property>
```

Pada parameter "yarn.scheduler.capacity.root.<*queue*-path>.state" digunakan untuk mengetahui status *job* atau aplikasi yang telah disubmit oleh *node slave* supaya antrian tersebut bisa berjalan. Untuk setiap *job* yang dieksekusi status antrian harus "RUNNING", jika *job* atau aplikasi tidak berjalan, maka user akan menerima pesan kesalahan "STOPPED"yang menyatakan bahwa antrian terhenti. "RUNNING dan STOPPED" adalah nilai yang diizinkan pada parameter ini. Berikut konfigurasi parameter antrian untuk administasi dan permission yaitu.

```
<property>
   <name>yarn.scheduler.capacity.root.queue1.state</name>
   <value>RUNNING</value> </property>
```

4.5.4 Konfigurasi Access Control List (ACL)

Hal pertama yang harus dilakukan administrator untuk mengatur ACL pada parameter "yarn.scheduler.capacity.root.acl_submit_applications" adalah untuk mengaktifkan pengguna tertentu supaya bisa mengirimkan *job* atau aplikasi ke antrian tertentu menggunakan parameter "yarn.scheduler.capacity.root.<*Queue*-path>.acl_submit_applications". Pada parameter ini administrator harus menentukan nama pengguna atau grup yang dipisah dengan menggunakan tanda koma. Namun khusus pada nilai "*" yang menyatakan bahwa semua pengguna bisa mengirimkan *job* ke atrian.

```
<property>
<name>yarn.scheduler.capacity.root.queue1.acl_submit_applications</name>
  <value>*</value>
  </property>
```

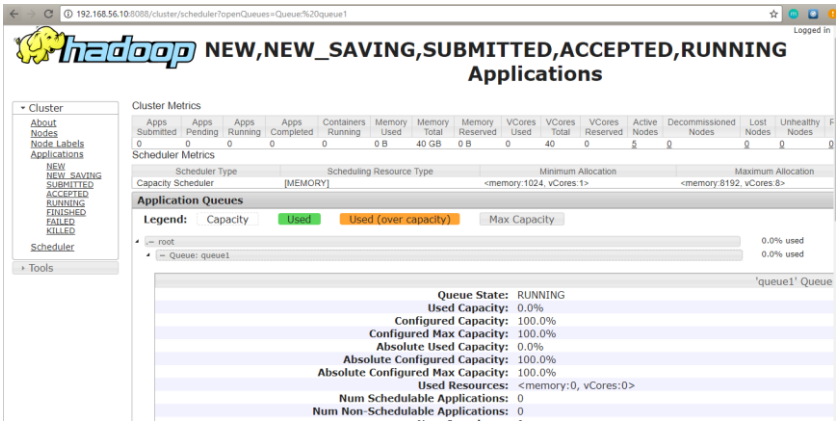
Selain dapat menentukan daftar pengguna yang bisa mensubmit *job* . Parameter"yarn.scheduler.capacity.<*Queue*-path>.acl_administer_*queue*" digunakan untuk mengatur daftar administrator yang bisa mengelola aplikasi pada suatu atrian. Jika menggunakan nilai khusus "*" yang berarti bahwa semua administrator bisa mengelola *job* yang sedang berjalan pada antrian. Konfigurasi parameter tersebut dapat sebagai berikut.

```
<property>
<name>yarn.scheduler.capacity.root.queue1.acl_administer_queue</name>
<value>*</value> </property>
```

4.5.5 Melihat Daftar Antrian

Setelah melakukan konfigurasi pada *file* yarn-site.xml dan capacity-scheduler.xml, selanjtnya jalankan perintah " yarn rmadmin –refresh*Queues*" untuk mengaktifkan perubahan konfigurasi yang telah dilakukan sebelumnya. Untuk melihat daftar antrian pada capcity scheduler dapat dilihat dengan membuka Antarmuka dari YARN *resource* manager. Untuk membuka antarmuka tersebut dengan cara masuk ke link <*resource*manager-hostname>:8088. Dimana <*resource*manager-hostname> merupakan alamat IP dari *master* Hadoop dan

8088 merupakan port default unutk membuka antarmuka dari *resource*manager tersebut. Berikut adalah tampilan antarmuka dengan *Capacity Scheduling* seperti pada gambar 4.23.



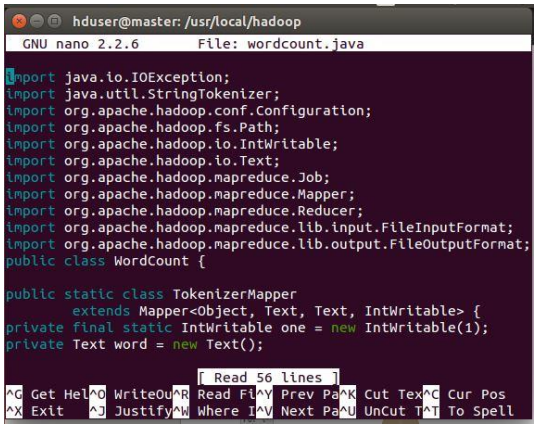
Gambar 4.23 Daftar antrian *Capacity Scheduling* pada Resource Manager

4.6 Konfigurasi *job wordcount*

Pada sub bab ini akan dijelaskan tentang cara pengiriman *job* yang akan digunakan pada penilitian ini. Salah satu jenis *job* yang digunakan adalah *job wordcount*. Untuk dapat menjalankan *job wordcount* perlu dilakukan beberapa tahapan seperti seperti melakukan *compile* terhadap kode program, input *dataset* ke folder hdfs, serta menjalankan *job* wordcount ke suatu antrian Hadoop yang telah dikonfigurasikan. Untuk lebih jelasnya tahapan-tahapan tersebut akan dilakukan seperti berikut.

4.6.1 Melakukan *Compile* Kode Program

Untuk dapat menjalankan program wordcount, hal yang pertama dilakukan adalah dengan menjalankan server Hadoop. Selanjutnya melakukan *compile* pada kode program *wordcount* yang telah dibuat sebelumnya. Untuk kode program lebih lengkap pada *file wordcount* java dapat dilihat pada lampiran. Selankutnya kode program *wordcount* yang telah di *compile* akan di format dalam bentuk jar untuk dapat dieksekusi.



Gambar 4.24 Kode Program *Wordcount*

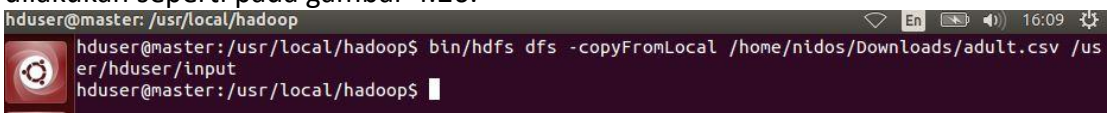
Setelah membuat kode program *wordcount* dan disimpan dalam *file* java, selanjutnya kode program tersebut di *compile* ke format jar dengan menggunakan sintak "bin/hdfs com.sun.tools.javac.Main WordCount.java" lalu ketik perintah "jar df wc.jar WordCount*.class". setelah itu pada folder Hadoop akan muncul folder seperti gambar 4.25.

Gambar 4.25 Hasil *Compile* Program *Wordcount*

4.6.2 Membuat Direktori HDFS

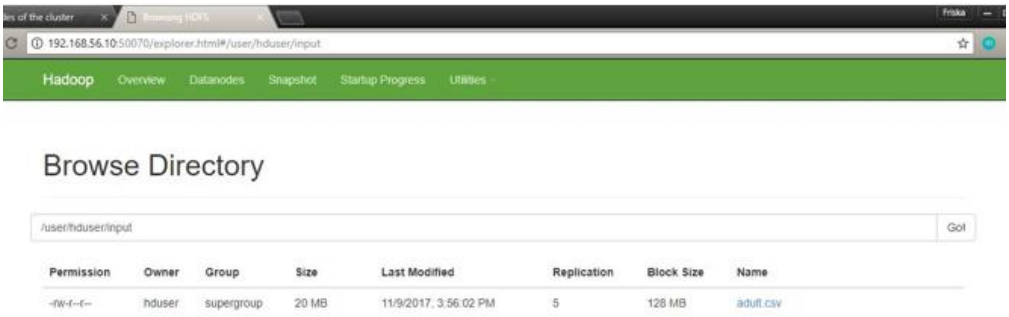
Untuk menjalankan *job* pada Hadoop diperlukan data yang diambil dari direktori HDFS yang telah disediakan pada komponen Hadoop. Data yang digunakan ini dapat dilakukan dengan membuat variasi *dataset* yang berbeda ukuran sesuai dengan pengujian yang dibutuhkan. Untuk membuat direktori pada HDFS dengan menggunakan perintah "bin/hdfs dfs –mkdir –p /user/hduser/input".

Setelah membuat direktori input pada folder HDFS, setelah itu akan dilakukan copy *file* adult.csv yang berada pada direktori /home/nidos/Downloads/adult.csv kedalam direktori HDFS /user/hduser/input. Data yang digunakan pada Hadoop ini berupa plaintext yang berukuran 20 MB. Perintah yang digunakan untuk melakukan copy *file* kedalam folder HDFS dapat dilakukan seperti pada gambar 4.26.



Gambar 4.26 *Menyalin File* pada Direktori HDFS

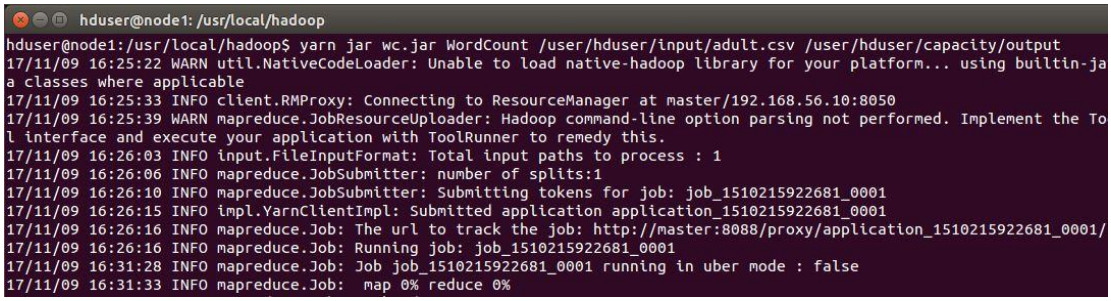
Setelah berhasil menjalankan perintah tersebut, selanjutnya pada web UI dapat dilihat pada url "localhost:50070" untuk melihat direktori HDFS apakah *file* adult.csv telah berhasil dimasukkan atau tidak seperti pada gambar 4.27.



Gambar 4.27 Tampilan Direktori HDFS

4.6.3 Menjalankan *Job Wordcount*

Setelah melakukan semua tahapan tersebut, maka *job* wordcount akan siap dijalankan. Untuk menjalankan *job* wordcount perlu beberapa perintah. Pada perintah pertama "yarn" karena pada komponen yarn terdapat fungsi scheduling. Untuk perintah "jar wc.jar" berisi kode program wordcount yang disimpan dengan format jar. Untuk perintah "WordCount" adalah jenis *job* wordcount yang akan dijalankan dengan nama class WordCount pada isi kode program. Selanjutnya perintah "/user/hduser/input/adult.csv" digunakan untuk memilih input data yang diletakkan pada direktori HDFS dengan nama data adult.csv. Untuk perintah "/user/hduser/capacity/output" digunakan untuk meletakkan hasil output dari *job* wordcount tersebut yang diletakkan pada direktori HDFS. Untuk melihat perintah menjalankan *job* wordcount lebih jelas bisa dilihat pada gambar 4.28.

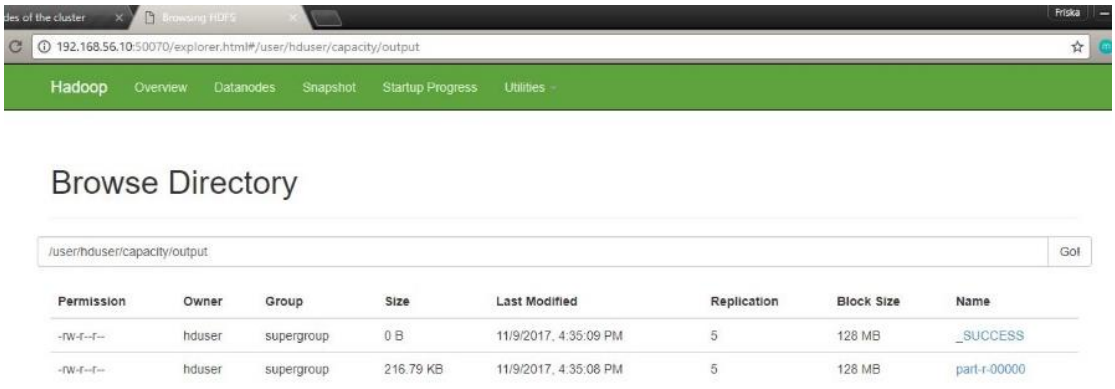


Gambar 4.28 Perintah Menjalankan *Job Wordcount*

Untuk menjalankan *job wordcount*, Hadoop menyediakan antarmuka tampilan pada *resource manager* yang bisa diakses dengan alamat url "192.168.56.10:8088" 192.168.56.10 merupakan alamat ip yang dimiliki oleh komputer *master* pada *server* Hadoop. Sedangkan 8088 merupakan *port* pada *resource manager*. Pada tampilan *resource manager* ini, bila mengklik scheduler akan muncul tampilan dari beberapa informasi seperti *cluster metrics, scheduler metrics, application queues*, dll. Beberapa informasi tersebut dapat dilihat pada gambar 4.29.

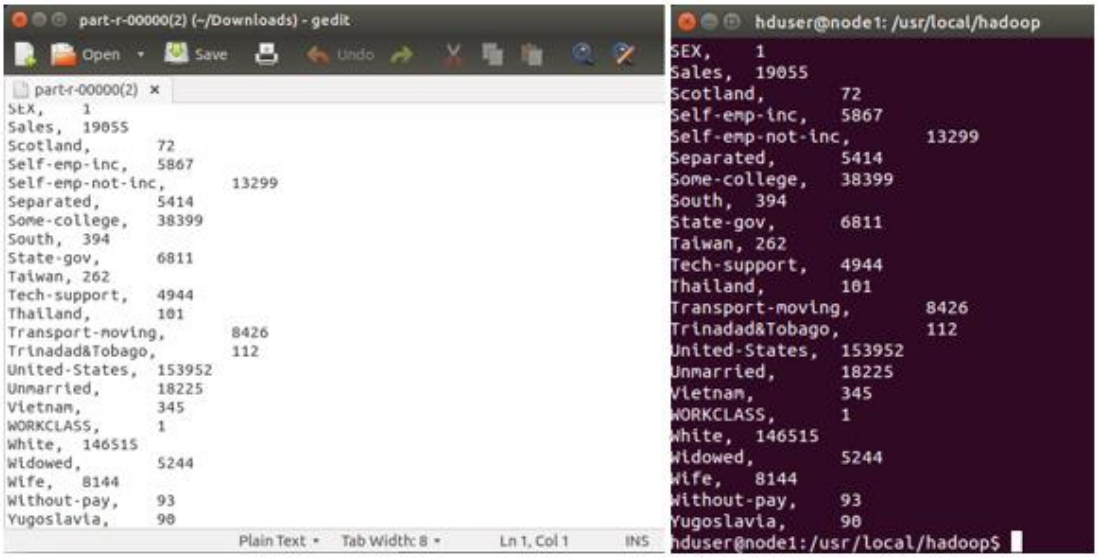
Gambar 4.29 Tampilan *Resourcemanage*r Untuk Monitoring *Job Wordcount*

Setelah menjalankan *job* wordcount, maka keluaran hasil dapat dilihat pada direktori HDFS yang terletak pada "/user/hduser/capacity/output" pada direktori tersebut terdapat 2 *file* yang terdiri dari *file* pertama yaitu, "_SUCCESS" yang berisi keterangan bahwa *job* wordcount tersebut telah berhasil dijalankan, dan *file* kedua yaitu, "part-r-00000" yang berisi hasil keluaran *job* wordcount pada *file* adult.csv. Dan pada direktori ini terdapat beberapa informasi lainnya yang dapat dilihat pada gambar 4.30.



Gambar 4.30 Antarmuka Direktori *Output* pada *Job Wordcount*

Selanjutnya untuk melihat hasil keluaran pada *job* wordcount tersebut bisa dengan melakukan klik pada *file* part-r-00000 dan klik download atau dengan ketik perintah "bin/hdfs dfs –cat /user/hduser/capacity/output" pada terminal. Pada *job* wordcount digunakan untuk menghitung kemiripan jumlah kata yang terdapat pada sebuah *file* berdasarkan nilai *key-value*. Hasil keluaran *job* wordcount yang dilakukan pada eksekusi *file* adult.csv dapat dilihat pada gambar 4.31.



Gambar 4.31 Hasil Keluaran *Job Wordcount* pada *File* adult.csv

4.7 Pengujian

Pengujian pada algoritme penjadwalan *Fair Share Scheduling* dan *Capacity Scheduling* ini dilakukan untuk mengetahui perbandingan kinerja dari kedua algoritme tersebut berdasarkan pada pengiriman *job* yang diajukan pada *cluster* Hadoop. Pengujian ini dilakukan dengan 3 skenario pengujian yang berbeda berdasarkan pada variabel bebas dan variabel tetap. Untuk skenario pengujian pertama menggunakan variabel bebas berdasarkan variasi ukuran data, skenario pengujian kedua menggunakan variabel bebas berdasarkan variasi jumlah *job*, dan skenario pengujian ketiga menggunakan variabel bebas berdasarkan variasi jenis *job*. Pada pengujian ini akan menggunakan parameter pembanding yaitu, *Job Fail Rate, Latency, dan Throughput* untuk mendapatkan hasil berupa grafik nilai yang dilakukan untuk perbandingan kinerja antara kedua algoritme tersebut.

4.7.1 Alat pengujian

Pada penelitian ini, penulis menggunakan menggunakan perangkat lunak Apache Hadoop versi 2. Apache Hadoop merupakan *framework open source* berbasis Java yang digunakan untuk melakukan komputasi data dengan ukuran skala yang sangat besar secara terdistribusi. Sistem kerja dari apache Hadoop terdiri dari 3 komponen utama yang terdiri dari HDFS, MapReduce, dan YARN. Pada HDFS digunakan untuk sebagai media penyimpanan data dan mengelola metadata dari kluster, Mapreduce bertugas untuk membagi data kedalam ukuran yang lebih kecil kedalam bentuk tupel (kombinasi antara *key-value*), dan YARN yang bertanggung jawab untuk mengatur *resources* dalam *cluster*.

Untuk melakukan pengujian Apache Hadoop, pada YARN terdapat komponen *resource* manager yang digunakan untuk menyediakan antarmuka web UI untuk memudahkan user melakukan *monitoring* pada proses pengiriman *job* pada *Fair Share Scheduling* dan *Capacity Scheduling.* Pada penelitian ini *user*

melakukan *monitoring job* untuk mendapatkan data yang akan digunakan untuk membandingkan kesua algoritme tersebut dengan menggunakan parameter *Job Fail Rate, Latency, dan Throughput.*

4.7.2 Pengujian variasi ukuran data

Pada skenario pengujian ini dilakukan dengan menggunakan variabel bebas berupa variasi ukuran data, sedangkan variabel tetap berupa jenis *job* dan jumlah *job*. Variasi ukuran data yang digunakan adalah 1,5 GB, 286,81 MB, dan 20 MB. Sedangkan jenis *job* yang digunakan adalah *job* wordcount dan jumlah *job* sebanyak 5 *job*. Pada skenario pengujian ini menggunakan 5 buah *node slave* untuk melakukan pengiriman *job* pada masing-masing algoritme *Fair Share Scheduling* dan *Capacity Scheduling.* Dari hasil pengujian pada skenario variasi ukuran data ini bertujuan untuk mengetahui nilai dari parameter pengujian sebagai bahan acuan pembanding yaitu, *Job Fail Rate, Latency, dan Throughput.*

4.7.2.1 *Fair Share Scheduling*

Pada skenario pengujian dengan variasi ukuran data, untuk menjalankan *job* wordcount dilakukan pada salah satu komputer *slave* dengan mengirimkan 5 buah *job* sekaligus ke dalam antrian *Fair Share Scheduling* pada sistem Hadoop. Pada perintah ini menggunakan data facebook-names-unique.txt dengan ukuran data sebesar 1,5 GB. Perintah tersebut sebagai berikut.

```
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario1/fair/output1 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario1/fair/output2 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario1/fair/output3 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario1/fair/output4 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario1/fair/output5
```

Pada perintah kedua ini menjalankan *job* wordcount dilakukan pada salah satu komputer *slave* dengan mengirimkan 5 buah *job* sekaligus ke dalam antrian *Fair Share Scheduling* pada sistem Hadoop. Pada perintah ini menggunakan data facebook-f.last-withcount.txt dengan ukuran data sebesar 286,81 MB. Perintah tersebut sebagai berikut.

```
yarn jar wc.jar WordCount /user/hduser/input/facebook-f.last-withcount.txt /user/hduser/skenario1/fair/output6 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-f.last-withcount.txt /user/hduser/skenario1/fair/output7 &
```

```
yarn jar wc.jar WordCount /user/hduser/input/facebook-f.last-withcount.txt /user/hduser/skenario1/fair/output8 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-f.last-withcount.txt /user/hduser/skenario1/fair/output9 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-f.last-withcount.txt /user/hduser/skenario1/fair/output10
```

Pada perintah ketiga ini menjalankan *job* wordcount dilakukan pada salah satu komputer *slave* dengan mengirimkan 5 buah *job* sekaligus ke dalam antrian *Fair Share Scheduling* pada sistem Hadoop. Pada perintah ini menggunakan data adult.csv dengan ukuran data sebesar 20 MB. Perintah tersebut sebagai berikut.

```
yarn jar wc.jar WordCount /user/hduser/input/adult.csv /user/hduser/skenario1/ fair/output11 &
yarn jar wc.jar WordCount /user/hduser/input/adult.csv /user/hduser/skenario1/ fair/output12 &
yarn jar wc.jar WordCount /user/hduser/input/adult.csv /user/hduser/skenario1/ fair/output13 &
yarn jar wc.jar WordCount /user/hduser/input/adult.csv /user/hduser/skenario1/ fair/output14 &
yarn jar wc.jar WordCount /user/hduser/input/adult.csv /user/hduser/skenario1/ fair/output15
```

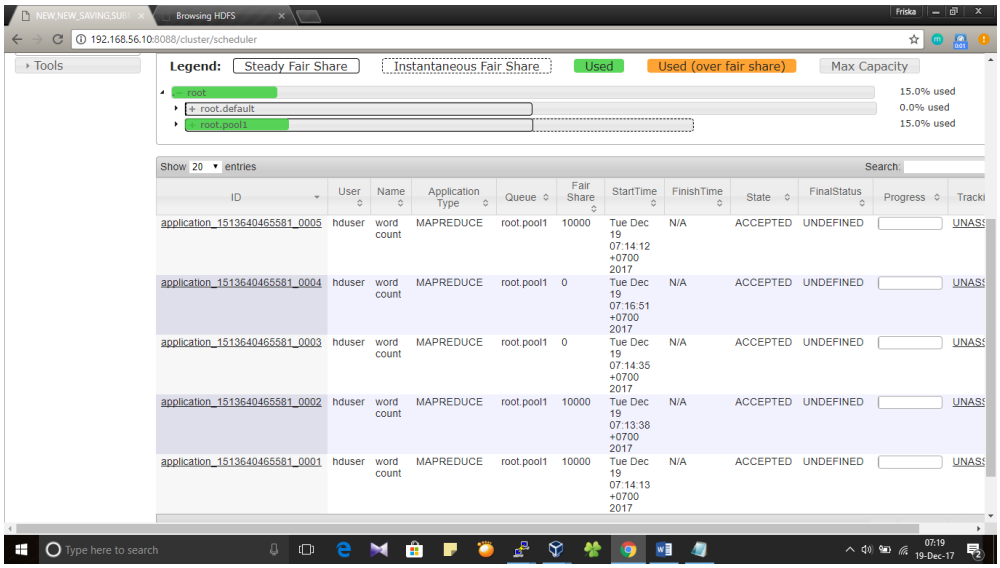
Pada perintah tersebut, komponen “yarn” disini terdapat fungsi scheduling. Untuk perintah “jar wc.jar” berisi kode program wordcount yang disimpan dengan format jar. Untuk perintah “WordCount” adalah jenis *job* wordcount yang akan dijalankan dengan nama class WordCount pada isi kode program.

Untuk perintah 1 dengan jumlah ukuran data sebesar 1,5 GB menggunakan perintah “/user/hduser/input/ facebook-names-unique.txt” digunakan sebagai input data yang diletakkan pada direktori HDFS. Untuk perintah 2 dengan jumlah ukuran data sebesar 286,81 MB menggunakan perintah “/user/hduser/input/facebook-f.last-withcount.txt” digunakan sebagai input data yang diletakkan pada direktori HDFS. Untuk perintah 3 dengan jumlah ukuran data sebesar 1,5 GB menggunakan perintah “/user/hduser/input/adult.csv” digunakan sebagai input data yang diletakkan pada direktori HDFS.

Untuk perintah “/user/hduser/skenario1/fair/output1” digunakan untuk meletakkan hasil output dari *job* wordcount tersebut yang diletakkan pada direktori HDFS. Untuk tanda “&” digunakan untuk tanda pemisah *job* satu dengan *job* lainnya agar bisa dilakukan proses *multiple job.*

Untuk menjalankan *job wordcount*, Hadoop menyediakan antarmuka tampilan pada *resource manager* yang bisa diakses dengan alamat url “192.168.56.10:8088” 192.168.56.10 merupakan alamat ip yang dimiliki oleh komputer *master* pada *server* Hadoop. Sedangkan 8088 merupakan *port* pada

resource manager. Untuk mengetahui jalannya *job* pada antrian algorima *Fair Share Scheduling* dapat dilakukan dengan mengklik menu scheduler dan akan muncul seperti gambar 4.32.



Gambar 4.32 Monitoring *Job* Variasi Ukuran Data pada *Fair Share Scheduling*

4.7.2.2 *Capacity Scheduling*

Pada skenario pengujian dengan variasi ukuran data, untuk menjalankan *job* wordcount dilakukan pada salah satu komputer *slave* dengan mengirimkan 5 buah *job* sekaligus ke dalam antrian *Capacity Scheduling* pada sistem Hadoop. Pada perintah ini menggunakan data facebook-names-unique.txt dengan ukuran data sebesar 1,5 GB. Perintah tersebut sebagai berikut.

```
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario1/capacity/output1 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario1/capacity/output2 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario1/capacity/output3 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario1/capacity/output4 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario1/capacity/output5
```

Pada perintah kedua akan menjalankan *job* wordcount yang dilakukan pada salah satu komputer *slave* dengan mengirimkan 5 buah *job* sekaligus ke dalam antrian *Capacity Scheduling* pada sistem Hadoop. Pada perintah ini menggunakan facebook-f.last-withcount.txt dengan ukuran data sebesar 286,81 MB. Perintah tersebut sebagai berikut.

```
yarn jar wc.jar WordCount /user/hduser/input/facebook-f.last-withcount.txt /user/hduser/skenario1/capacity/output6 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-f.last-withcount.txt /user/hduser/skenario1/capacity/output7 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-f.last-withcount.txt /user/hduser/skenario1/capacity/output8 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-f.last-withcount.txt /user/hduser/skenario1/capacity/output9 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-f.last-withcount.txt /user/hduser/skenario1/capacity/output10
```

Pada perintah ketiga akan menjalankan *job* wordcount yang dilakukan pada salah satu komputer *slave* dengan mengirimkan 5 buah *job* sekaligus ke dalam antrian *Capacity Scheduling* pada sistem Hadoop. Pada perintah ini menggunakan adult.csv dengan ukuran data sebesar 20 MB. Perintah tersebut sebagai berikut.

```
yarn jar wc.jar WordCount /user/hduser/input/adult.csv /user/hduser/skenario1/ capacity/output11 &
yarn jar wc.jar WordCount /user/hduser/input/adult.csv /user/hduser/skenario1/ capacity/output12 &
yarn jar wc.jar WordCount /user/hduser/input/adult.csv /user/hduser/skenario1/ capacity/output13 &
yarn jar wc.jar WordCount /user/hduser/input/adult.csv /user/hduser/skenario1/ capacity/output14 &
yarn jar wc.jar WordCount /user/hduser/input/adult.csv /user/hduser/skenario1/ capacity/output15
```

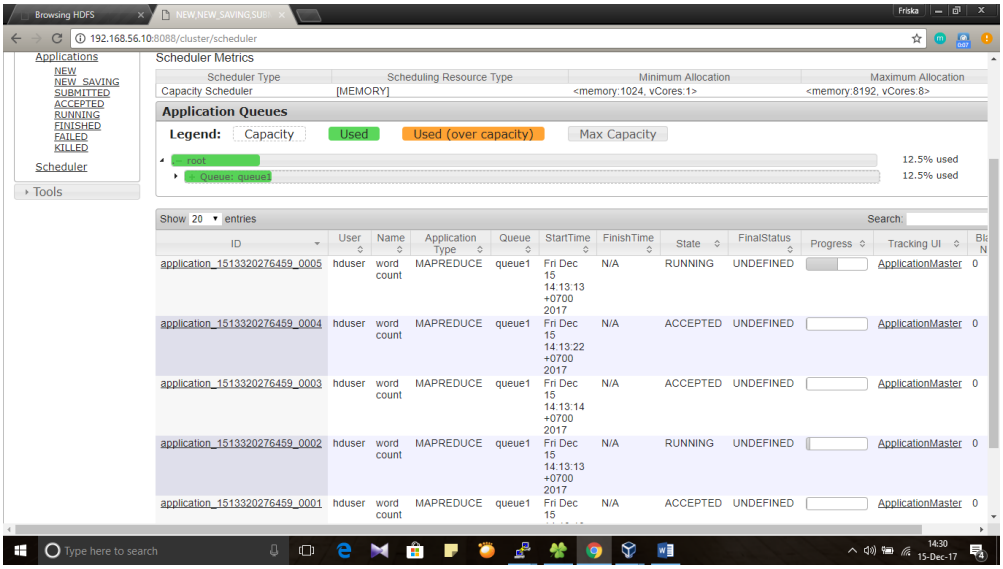
Pada perintah tersebut, komponen “yarn” disini terdapat fungsi scheduling. Untuk perintah “jar wc.jar” berisi kode program wordcount yang disimpan dengan format jar. Untuk perintah “WordCount” adalah jenis *job* wordcount yang akan dijalankan dengan nama class WordCount pada isi kode program.

Untuk perintah 1 dengan jumlah ukuran data sebesar 1,5 GB menggunakan perintah “/user/hduser/input/ facebook-names-unique.txt” digunakan sebagai input data yang diletakkan pada direktori HDFS. Untuk perintah 2 dengan jumlah ukuran data sebesar 286,81 MB menggunakan perintah “/user/hduser/input/facebook-f.last-withcount.txt” digunakan sebagai input data yang diletakkan pada direktori HDFS. Untuk perintah 3 dengan jumlah ukuran data sebesar 1,5 GB menggunakan perintah “/user/hduser/input/adult.csv” digunakan sebagai input data yang diletakkan pada direktori HDFS.

Untuk perintah “/user/hduser/skenario1/capacity/output1” digunakan untuk meletakkan hasil output dari *job* wordcount tersebut yang diletakkan pada

direktori HDFS. Untuk tanda "&" digunakan untuk tanda pemisah *job* satu dengan *job* lainnya agar bisa dilakukan proses *multiple job.*

Untuk menjalankan *job wordcount*, Hadoop menyediakan antarmuka tampilan pada *resource manager* yang bisa diakses dengan alamat url "192.168.56.10:8088" 192.168.56.10 merupakan alamat ip yang dimiliki oleh komputer *master* pada *server* Hadoop. Sedangkan 8088 merupakan *port* pada *resource manager*. Untuk mengetahui jalannya *job* pada antrian algorima *Capacity Scheduling* dapat dilakukan dengan mengklik menu scheduler dan akan muncul seperti gambar 4.33.



Gambar 4.33 Monitoring *Job* Variasi Ukuran Data pada *Capacity Scheduling*

4.7.3 Pengujian variasi jumlah *job*

Pada skenario pengujian ini dilakukan dengan menggunakan variabel bebas berupa variasi jumlah *job*, sedangkan variabel tetap berupa jumlah ukuran data dan jenis *job*. Variasi jumlah *job* yang digunakan adalah 5 *job*, 10 *job*, dan 15 *job*. Sedangkan jumlah ukuran data yang digunakan adalah 1,5 GB dan jenis *job* yang digunakan adalah *job* wordcount. Pada skenario pengujian ini menggunakan 5 buah *node slave* untuk melakukan pengiriman *job* pada masing-masing algoritme *Fair Share Scheduling* dan *Capacity Scheduling.* Dari hasil pengujian pada skenario variasi ukuran data ini bertujuan untuk mengetahui nilai dari parameter pengujian sebagai bahan acuan pembanding yaitu, *Job Fail Rate, Latency, dan Throughput.*

4.7.3.1 *Fair Share Scheduling*

Pada skenario pengujian dengan variasi jumlah *job*, untuk menjalankan *job* wordcount dilakukan pada salah satu komputer *slave* dengan mengirimkan 5 buah *job* sekaligus ke dalam antrian *Fair Share Scheduling* pada sistem Hadoop. Pada

perintah ini menggunakan data facebook-names-unique.txt dengan ukuran data sebesar 1,5 GB. Perintah tersebut sebagai berikut.

```
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output1 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output2 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output3 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output4 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output5
```

Pada perintah kedua ini akan menjalankan *job* wordcount dilakukan pada salah satu komputer *slave* dengan mengirimkan 10 buah *job* sekaligus ke dalam antrian *Fair Share Scheduling* pada sistem Hadoop. Pada perintah ini menggunakan data facebook-names-unique.txt dengan ukuran data sebesar 1,5 GB. Perintah tersebut sebagai berikut.

```
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output6 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output7 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output8 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output9 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output10 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output11 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output12 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output13 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output14 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output15
```

Pada perintah ketiga ini akan menjalankan *job* wordcount dilakukan pada salah satu komputer *slave* dengan mengirimkan 15 buah *job* sekaligus ke dalam antrian *Fair Share Scheduling* pada sistem Hadoop. Pada perintah ini menggunakan data facebook-names-unique.txt dengan ukuran data sebesar 1,5 GB. Perintah tersebut sebagai berikut.

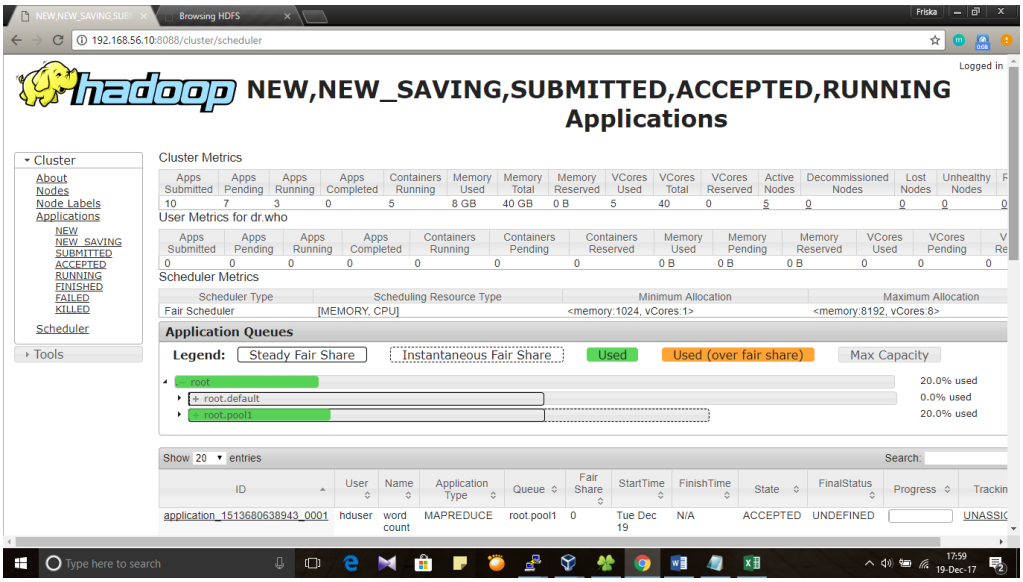
```
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output16 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output17 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output18 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output19 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output20 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output21 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output22 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output23 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output24 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output25 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output26 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output27 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output28 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output29 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/fair/output30
```

Pada perintah tersebut, komponen “yarn” disini terdapat fungsi scheduling. Untuk perintah “jar wc.jar” berisi kode program wordcount yang disimpan dengan format jar. Untuk perintah “WordCount” adalah jenis *job* wordcount yang akan dijalankan dengan nama class WordCount pada isi kode program.

Untuk perintah 1 mengirimkan *job* sebanyak 5 *job* dengan jumlah ukuran data sebesar 1,5 GB menggunakan perintah “/user/hduser/input/ facebook-names-unique.txt” digunakan sebagai input data yang diletakkan pada direktori HDFS. Untuk perintah 2 mengirimkan *job* sebanyak 10 *job* dengan jumlah ukuran data sebesar 1,5 GB menggunakan perintah “/user/hduser/input/ facebook-names-unique.txt”. Untuk perintah 3 mengirimkan *job* sebanyak 15 *job* dengan jumlah ukuran data sebesar 1,5 GB menggunakan perintah “/user/hduser/input/ facebook-names-unique.txt”.

Untuk perintah "/user/hduser/skenario1/fair/output1" digunakan untuk meletakkan hasil output dari *job* wordcount tersebut yang diletakkan pada direktori HDFS. Untuk tanda "&" digunakan untuk tanda pemisah *job* satu dengan *job* lainnya agar bisa dilakukan proses *multiple job.*

Untuk menjalankan *job wordcount*, Hadoop menyediakan antarmuka tampilan pada *resource manager* yang bisa diakses dengan alamat url "192.168.56.10:8088" 192.168.56.10 merupakan alamat ip yang dimiliki oleh komputer *master* pada *server* Hadoop. Sedangkan 8088 merupakan *port* pada *resource manager*. Untuk mengetahui jalannya *job* pada antrian algorima *Fair Share Scheduling* dapat dilakukan dengan mengklik menu scheduler dan akan muncul seperti gambar 4.34.



Gambar 4.34 Monitoring *Job* Variasi Jumlah *Job* pada *Fair Share Scheduling*

4.7.3.2 *Capacity Scheduling*

Pada skenario pengujian kedua ini, untuk menjalankan *job* wordcount dilakukan pada salah satu komputer *slave* dengan mengirimkan 5 buah *job* sekaligus ke dalam antrian *Capacity Scheduling* pada sistem Hadoop. Pada perintah ini menggunakan data facebook-names-unique.txt dengan ukuran data sebesar 1,5 GB. Perintah tersebut sebagai berikut.

```
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output1 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output2 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output3 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output4 &
```

```
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output5
```

Pada perintah kedua ini akan menjalankan *job* wordcount dilakukan pada salah satu komputer *slave* dengan mengirimkan 15 buah *job* sekaligus ke dalam antrian *Capacity Scheduling* pada sistem Hadoop. Pada perintah ini menggunakan data facebook-names-unique.txt dengan ukuran data sebesar 1,5 GB. Perintah tersebut sebagai berikut.

```
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output6 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output7 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output8 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output9 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output10 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output11 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output12 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output13 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output14 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output15
```

Pada perintah ketiga ini akan menjalankan *job* wordcount dilakukan pada salah satu komputer *slave* dengan mengirimkan 15 buah *job* sekaligus ke dalam antrian *Capacity Scheduling* pada sistem Hadoop. Pada perintah ini menggunakan data facebook-names-unique.txt dengan ukuran data sebesar 1,5 GB. Perintah tersebut sebagai berikut.

```
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output16 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output17 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output18 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output19 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output20 &
```

```
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output21 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output22 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output23 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output24 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output25 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output26 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output27 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output28 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario2/capacity/output29&
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario1/capacity/output30
```

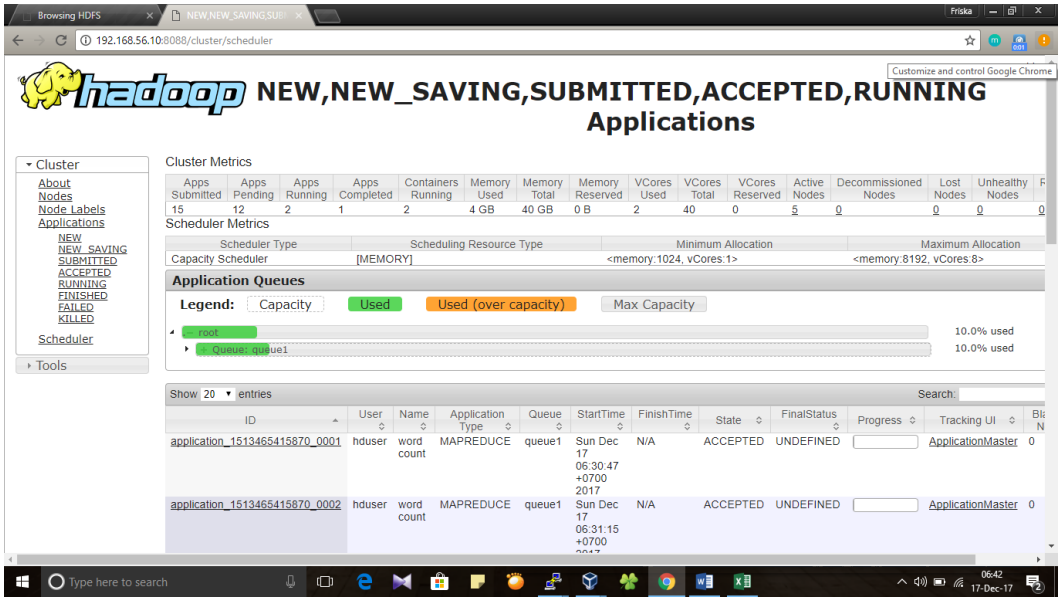
Pada perintah tersebut, komponen “yarn” disini terdapat fungsi scheduling. Untuk perintah “jar wc.jar” berisi kode program wordcount yang disimpan dengan format jar. Untuk perintah “WordCount” adalah jenis *job* wordcount yang akan dijalankan dengan nama class WordCount pada isi kode program.

Untuk perintah 1 mengirimkan *job* sebanyak 5 *job* dengan jumlah ukuran data sebesar 1,5 GB menggunakan perintah “/user/hduser/input/ facebook-names-unique.txt” digunakan sebagai input data yang diletakkan pada direktori HDFS. Untuk perintah 2 mengirimkan *job* sebanyak 10 *job* dengan jumlah ukuran data sebesar 1,5 GB menggunakan perintah “/user/hduser/input/ facebook-names-unique.txt”. Untuk perintah 3 mengirimkan *job* sebanyak 15 *job* dengan jumlah ukuran data sebesar 1,5 GB menggunakan perintah “/user/hduser/input/ facebook-names-unique.txt”.

Untuk perintah “/user/hduser/skenario1/capacity/output1” digunakan untuk meletakkan hasil output dari *job* wordcount tersebut yang diletakkan pada direktori HDFS. Untuk tanda “&” digunakan untuk tanda pemisah *job* satu dengan *job* lainnya agar bisa dilakukan proses *multiple job.*

Untuk menjalankan *job wordcount*, Hadoop menyediakan antarmuka tampilan pada *resource manager* yang bisa diakses dengan alamat url “192.168.56.10:8088” 192.168.56.10 merupakan alamat ip yang dimiliki oleh komputer *master* pada *server* Hadoop. Sedangkan 8088 merupakan *port* pada *resource manager*. Untuk mengetahui jalannya *job* pada antrian algorima *Capacity*

Scheduling dapat dilakukan dengan mengklik menu scheduler dan akan muncul seperti gambar 4.35.



Gambar 4.35 Monitoring *job* variasi jumlah *job* pada *Capacity Scheduling*

4.7.4 Pengujian variasi jenis *job*

Pada skenario pengujian ini dilakukan dengan menggunakan variabel bebas berupa variasi jenis *job*, sedangkan variabel tetap berupa jumlah ukuran data dan jumlah *job*. Variasi jenis *job* yang digunakan adalah *job* wordcount dan *job* wordmean. Sedangkan jumlah ukuran data yang digunakan sebesar 1,5 GB dan jumlah *job* yang digunakan sebanyak 5 *job*. Pada skenario pengujian ini menggunakan 5 buah *node slave* untuk melakukan pengiriman *job* pada masing-masing algoritme *Fair Share Scheduling* dan *Capacity Scheduling.* Dari hasil pengujian pada skenario variasi ukuran data ini bertujuan untuk mengetahui nilai dari parameter pengujian sebagai bahan acuan pembanding yaitu, *Job Fail Rate, Latency, dan Throughput.*

4.7.4.1 *Fair Share Scheduling*

Pada skenario pengujian dengan variasi jenis *job*, menjalankan *job* wordcount. Pada *job* ini dikerjakan pada salah satu komputer *slave* dengan mengirimkan 5 buah *job* sekaligus ke dalam antrian *Fair Share Scheduling* pada sistem Hadoop. Pada perintah ini menggunakan data facebook-names-unique.txt dengan ukuran data sebesar 1,5 GB. Perintah tersebut sebagai berikut.

```
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt
/user/hduser/skenario3/fair/output1 &
```

```
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario3/fair/output2 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario3/fair/output3 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario3/fair/output4 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario3/fair/output5
```

Pada perintah kedua ini akan menjalankan *job* wordmean. Pada *job* ini dikerjakan pada salah satu komputer *slave* dengan mengirimkan 5 buah *job* sekaligus ke dalam antrian *Fair Share Scheduling* pada sistem Hadoop. Pada perintah ini menggunakan data facebook-names-unique.txt dengan ukuran data sebesar 1,5 GB. Perintah tersebut sebagai berikut.

```
yarn jar wordmean.jar WordMean /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario1/fair/output6 &
yarn jar wordmean.jar WordMean /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario1/fair/output7 &
yarn jar wordmean.jar WordMean /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario1/fair/output8 &
yarn jar wordmean.jar WordMean /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario1/fair/output9 &
yarn jar wordmean.jar WordMean /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario1/fair/output10
```

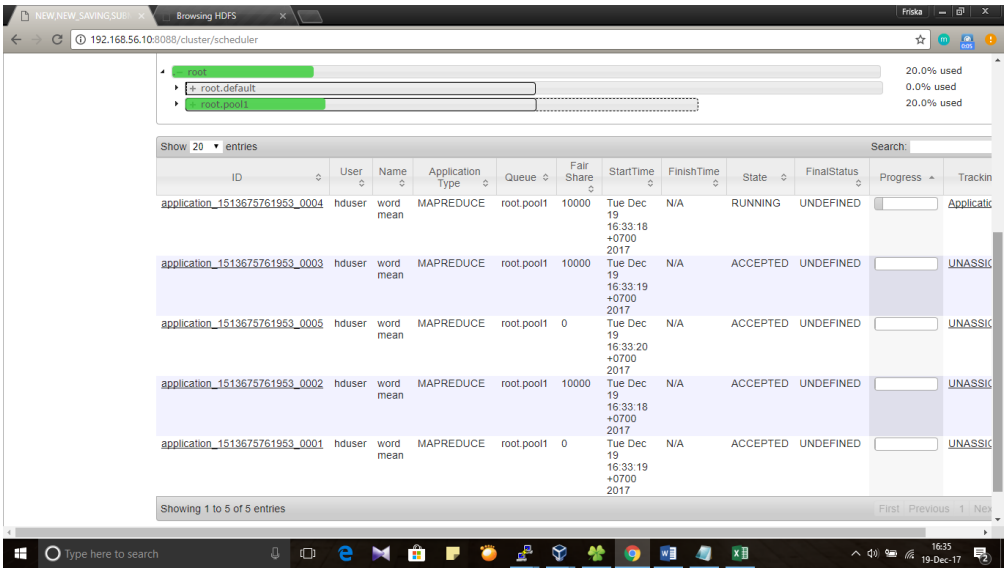
Pada skenario variasi jenis *job* dengan menggunakan algoritme *Fair Share Scheduling*, komponen "yarn" disini terdapat fungsi scheduling. Untuk perintah pertama, perintah "jar wc.jar" berisi kode program wordcount yang disimpan dengan format jar. Untuk perintah "WordCount" adalah jenis *job* wordcount yang akan dijalankan dengan nama class WordCount pada isi kode program. Pada perintah ini akan mengirimkan *job* sebanyak 5 *job* dengan jumlah ukuran data sebesar 1,5 GB menggunakan perintah "/user/hduser/input/facebook-names-unique.txt" digunakan sebagai input data yang diletakkan pada direktori HDFS.

Untuk perintah kedua, perintah "jar wordcount.jar" berisi kode program wordcount yang disimpan dengan format jar. Untuk perintah "WordCount" adalah jenis *job* wordcount yang akan dijalankan dengan nama class WordCount pada isi kode program. Pada perintah ini akan mengirimkan *job* sebanyak 5 *job* dengan jumlah ukuran data sebesar 1,5 GB menggunakan perintah "/user/hduser/input/facebook-names-unique.txt" digunakan sebagai input data yang diletakkan pada direktori HDFS.

Untuk perintah "/user/hduser/skenario1/fair/output1" digunakan untuk meletakkan hasil output dari *job* wordcount tersebut yang diletakkan pada

direktori HDFS. Untuk tanda "&" digunakan untuk tanda pemisah *job* satu dengan *job* lainnya agar bisa dilakukan proses *multiple job.*

Untuk menjalankan *job wordcount*, Hadoop menyediakan antarmuka tampilan pada *resource manager* yang bisa diakses dengan alamat url "192.168.56.10:8088" 192.168.56.10 merupakan alamat ip yang dimiliki oleh komputer *master* pada *server* Hadoop. Sedangkan 8088 merupakan *port* pada *resource manager*. Untuk mengetahui jalannya *job* pada antrian algorima *Fair Share Scheduling* dapat dilakukan dengan mengklik menu scheduler dan akan muncul seperti gambar 4.36.



Gambar 4.36 Monitoring *Job* Variasi Jenis *Job* pada *Fair Share Scheduling*

4.7.4.2 *Capacity Scheduling*

Pada skenario pengujian dengan variasi jenis *job*, menjalankan *job* wordcount. Pada *job* ini dikerjakan pada salah satu komputer *slave* dengan mengirimkan 5 buah *job* sekaligus ke dalam antrian *Capacity Scheduling* pada sistem Hadoop. Pada perintah ini menggunakan data facebook-names-unique.txt dengan ukuran data sebesar 1,5 GB. Perintah tersebut sebagai berikut.

```
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario3/capacity/output1 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario3/capacity/output2 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario3/capacity/output3 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario3/capacity/output4 &
yarn jar wc.jar WordCount /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario3/capacity/output5
```

Pada perintah kedua ini akan menjalankan *job* wordmean. Pada *job* ini dikerjakan pada salah satu komputer *slave* dengan mengirimkan 5 buah *job* sekaligus ke dalam antrian *Capacity Scheduling* pada sistem Hadoop. Pada perintah ini menggunakan data facebook-names-unique.txt dengan ukuran data sebesar 1,5 GB. Perintah tersebut sebagai berikut.

```
yarn jar wordmean.jar WordMean /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario1/capacity/output6 &
yarn jar wordmean.jar WordMean /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario1/capacity/output7 &
yarn jar wordmean.jar WordMean /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario1/capacity/output8 &
yarn jar wordmean.jar WordMean /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario1/capacity/output9 &
yarn jar wordmean.jar WordMean /user/hduser/input/facebook-names-unique.txt /user/hduser/skenario1/capacity/output10
```

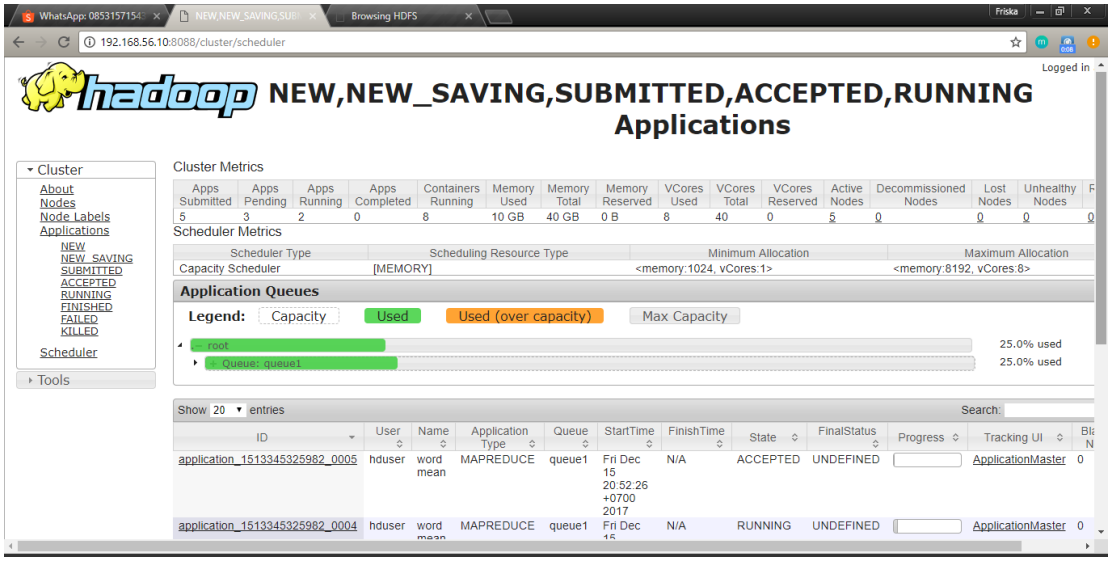
Pada skenario variasi jenis *job* dengan menggunakan algoritme *Capacity Scheduling*, komponen "yarn" disini terdapat fungsi scheduling. Untuk perintah pertama, perintah "jar wordmean.jar" berisi kode program wordmean yang disimpan dengan format jar. Untuk perintah "WordMean" adalah jenis *job* wordmean yang akan dijalankan dengan nama class WordMean pada isi kode program. Pada perintah ini akan mengirimkan *job* sebanyak 5 *job* dengan jumlah ukuran data sebesar 1,5 GB menggunakan perintah "/user/hduser/input/facebook-names-unique.txt" digunakan sebagai input data yang diletakkan pada direktori HDFS.

Untuk perintah kedua, perintah "jar wordmean.jar" berisi kode program wordmean yang disimpan dengan format jar. Untuk perintah "WordMean" adalah jenis *job* wordmean yang akan dijalankan dengan nama class WordMean pada isi kode program. Pada perintah ini akan mengirimkan *job* sebanyak 5 *job* dengan jumlah ukuran data sebesar 1,5 GB menggunakan perintah "/user/hduser/input/ facebook-names-unique.txt" digunakan sebagai input data yang diletakkan pada direktori HDFS.

Untuk perintah "/user/hduser/skenario1/capacity/output1" digunakan untuk meletakkan hasil output dari *job* wordmean tersebut yang diletakkan pada direktori HDFS. Untuk tanda "&" digunakan untuk tanda pemisah *job* satu dengan *job* lainnya agar bisa dilakukan proses *multiple job.*

Untuk menjalankan *job wordcount*, Hadoop menyediakan antarmuka tampilan pada *resource manager* yang bisa diakses dengan alamat url "192.168.56.10:8088" 192.168.56.10 merupakan alamat ip yang dimiliki oleh komputer *master* pada *server* Hadoop. Sedangkan 8088 merupakan *port* pada *resource manager*. Untuk mengetahui jalannya *job* pada antrian algorima *Capacity*

Scheduling dapat dilakukan dengan mengklik menu scheduler dan akan muncul seperti gambar 4.37.



Gambar 4.37 Monitoring *Job* Variasi Jenis *Job* pada *Fair Share Scheduling*