Analisis Penggunaan Algoritma *Delay Scheduling* terhadap Karakteristik *Job Scheduling* pada Hadoop

Komaratih Dian Priharyani[1)], Gandeva Bayu Satrya,ST.,MT[2)], Anton Herutomo, S.T., M.Eng.[3)]

Prodi S1 Teknik Informatika, Telkom School of Computing, Telkom University
Jalan Telekomunikasi no. 1, Dayeuhkolot Bandung 42057 Indonesia

[1]komaratih@gmail.com, [2]gandeva.bayu.s@gmail.com@, [3]herutomo@yahoo.com

Hadoop is a Java-based software framework and open-source that serves to process large amounts of data are distributed and run on a cluster that consists of several computers connected together. Hadoop has advantages in terms of economic because not pay, and can be implemented in hardware with a specification that is not too high. Hadoop architecture consists of two layers are layers and layers of MapReduce Hadoop Distributed File System (HDFS). MapReduce is a framework of distributed applications while the Hadoop Distributed File System (HDFS) is a distributed data.

Delay Scheduling is a job scheduler that is being developed in a multi-node Hadoop system and has the handling characteristics in the queue for job scheduling. Delay Scheduling jobs to apply the method further delay path to improve data locality in advance so that the lower value in the job file. Additionally, perform nearly optimal data allocation so that the effect on the Job Fail Rate, Job Throughput and Response Time. Delay Scheduling algorithm has an effective performance with a reduction in the Job Fail Rate 0.3%, 8.853% increase in job throughput, and faster 142 minutes 45 seconds Response Time with the type of job characteristics Wordcount in the amount of 50 jobs.

Key words**:** *hadoop, hadoop multi-node, delay scheduling, FIFO.*

1. PENDAHULUAN

Hadoop merupakan *framework software* berbasis *java* dan *open-source* yang berfungsi untuk mengolah data yang besar secara terdistribusi dan berjalan diatas *cluster* yang terdiri atas beberapa komputer yang saling terhubung[9]. Hadoop mempunyai kelebihan dari segi ekonomis karena tidak berbayar dan dapat diimplementasikan pada perangkat keras dengan spesifikasi yang tidak terlalu tinggi. Arsitektur Hadoop terdiri dari dua *layer* yaitu *layer MapReduce* dan layer Hadoop *Distributed File System* (*HDFS*). *MapReduce* merupakan *framework* dari aplikasi yang terdistribusi sedangkan Hadoop *Distributed File System* (*HDFS*) merupakan data yang terdistribusi.

Delay Scheduling merupakan *job scheduler* yang sedang berkembang pada sistem Hadoop dan memiliki ciri dalam penanganan antrian untuk *job scheduling* [8]. *Delay Scheduling* menerapkan metode menunda jalannya *jobs* selanjutnya untuk memperbaiki data lokalitas sebelumnya sehingga menurunkan nilai *fail* pada *job*. Selain itu, melakukan pengalokasian data yang hampir optimal sehingga berpengaruh pada *Job Fail Rate, Job Throughput,* dan *Response Time*.

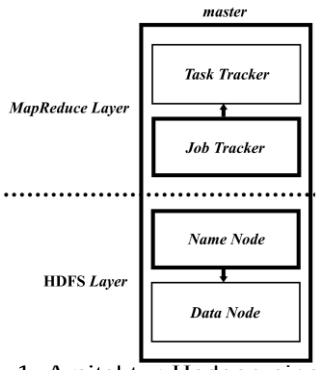
Tugas akhir ini akan menggunakan *FIFO* dan *Delay Scheduling* pada Hadoop sebagai *job scheduler*. Penggunaan ini akan dibandingkan dengan implementasinya pada berbagai jenis karakteristik *job* dengan menggunakan parameter *Job Fail Rate, Job Throughput,* dan *Response Time* sebagai acuan perhitungan performansi sistem. Serta analisis dan klasifikasi *job scheduler* berdasarkan performansi sistem terhadap karakteristik *job* dan jumlah *job* yang dijalankan pada masing - masing *job scheduler*.

2. LANDASAN TEORI

2.1 Hadoop *Single-Node*

Hadoop *single-node* merupakan satu *server* yang berkerja menjadi *master* tetapi tidak bekerja sebagai *slave*. Pada *server* hadoop

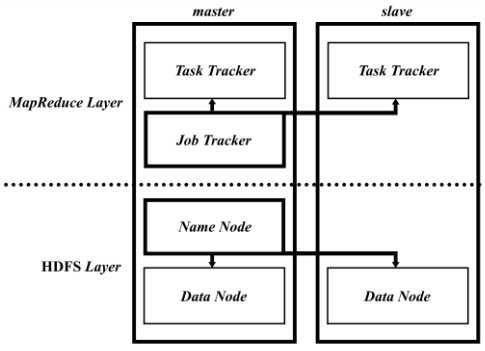
single-node semua proses dilakukan dalam satu *server*. Hadoop terdiri dari dua *layer* yaitu *layer* Hadoop *Distributed File System* (*HDFS*) dan *layer MapReduce*. *Layer* Hadoop *Distributed File System* (*HDFS)* yang menjalankan *namenode* dan *datanode* sedangkan *layer MapReduce* yang menjalankan *jobtracker* dan *tasktracker*. Pada kedua *layer* ini sangat penting aktif yaitu *namenode* dan *jobtracker* karena apabila kedua bagian ada yang tidak jalan maka kerja Hadoop *Distributed File System* (*HDFS)* dan *MapReduce* tidak bisa dijalankan.



Gambar 1. Arsitektur Hadoop *single-node*

2.2 Hadoop *Multi-Node*

Hadoop *multi-node* merupakan gabungan 2 *server single-node* yaitu 1 untuk *server master* dan 1 untuk *server slave*. Kedua *server* ini dikonfigurasikan melalui file *mapred-site.xml*, *core-site.xml*, dan *hdfs-site.xml*. *Server master* dapat berkerja juga sebagai *server slave* dan *server slave* hanya berkerja sebagai *server slave*.



Gambar 2. Arsitektur Hadoop *multi-node cluster*

2.3 *MapReduce*

Model pemrograman *MapReduce* membagi menjadi dua, yaitu *jobtracker* merupakan *server* penerima *job* dari *client* dan mengeksekusi kembali *maps* jika gagal sekaligus akan mendistribusikan *jobs* tersebut ke *tasktracker* yang akan mengerjakan sub-*job* sesuai yang diperintahkan *jobtracker*. Strategi ini akan mendekatkan pengolahan data dengan datanya sendiri, sehingga ini akan sangat signifikan mempercepat proses pengolahan data.

2.4 Hadoop *Distributed File System (HDFS)*

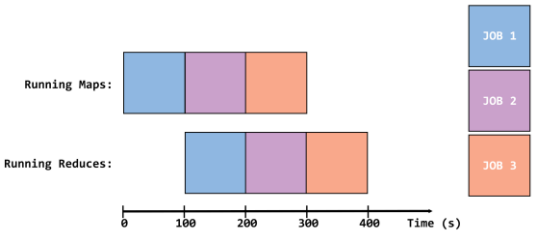
Hadoop *Distributed File System* (*HDFS*) merupakan *filesystem* yang berbasis *java*, yang menyimpan *file* dalam jumlah besar dan disimpan secara terdistribusi didalam banyak komputer yang saling berhubungan[9]. Pada umumnya data replika disimpan kedalam 3 *node* yaitu dua *rack* yang sama dan satu di *rack* yang berbeda, hal ini bertujuan untuk menjaga *realibility* dari Hadoop *Distributed File System* (*HDFS*). Sistem *file* membutuhkan *server* induk yang dinamakan *namenode*. *Namenode* berfungsi untuk menyimpan *metadata* dari data lam yang ada didalam data tersebut. Kemudian disimpan kedalam *server-server* yang dinamakan *datanode*. *Datanode* bisa saling berkomunikasi satu sama lain untuk menjaga konsistensi data dan memastikan proses replika data berjalan dengan baik.

2.5 Algoritma *FIFO*

Algoritma *FIFO* merupakan *default* yang digunakan oleh Hadoop pada proses penjadwalan. Algoritma ini mengatasi permasalahan antrian pada *jobs* dengan cara menjalakan sebuah *jobs* yang datang untuk pertama kali. Algoritma *FIFO* tidak menangani adanya skala prioritas dan perhitungan *long jobs* atau *short jobs*. Sehingga mengakibatkan

penggunaan algoritma *FIFO* kurang efektif. Pada implementasi algoritma *FIFO* dalam *server* berskala besar, algoritma *FIFO* dapat menurunkan performansi dari sisi *server* terutama pada sistem yang mempunyai layanan *sharing* data pada *multiple-user*[8].

FIFO memberikan kesempatan pada *job* untuk menggunakan *resource* yang ada pada Hadoop sesuai dengan waktu kedatangan dari *job*[8]. Apabila suatu *resource* kosong akan dilanjutkan kepada *job* selanjutnya[8]. Berikut ilustrasi dari *FIFO scheduler*:



Gambar 3. Ilustrasi *FIFO*

2.6 Algoritma *Delay Scheduling*

Algoritma *Delay Scheduling* mempunyai struktur yang berbeda dibandingkan dengan penjadwalan pada konfigurasi *default* dari sebuah Hadoop sistem. Pada konfigurasi *default* dari sebuah Hadoop adalah memakai algoritma *FIFO* sebagai teknik utama penjadwalan.

Delay Scheduling tidak menggunakan mekanisme *FIFO* yang memindahankan data pada *virtual hard disk* yang ada pada Hadoop dan memerlukan sinkronisasi terhadap data yang yang dipakai. Sehingga beberapa perpindahan *resource* ini membuat *job* mengalami *fail* dan dilakukan pengulangan *job* tidak digunakan karena pada awal *job* dibuat, *Delay Scheduling* sudah membagi *resource* data terhadap *pool* yang sesuai dengan *resource* data yang ada pada *virtual hard disk* Hadoop karena konfigurasi *pool* merupakan karakterisktik dari *Fair Scheduler* yang dimodifikasi dengan algoritma *Delay Scheduling*. Selain itu *Delay Scheduler* akan menggunakan metode menunda jalannya *jobs* selanjutnya untuk memperbaiki data lokalitas sebelumnya. Sehingga dapat meminimalkan *Response Time* dan memaksimalkan *Job Throughput*[13].

Algorithm 3 Delay Scheduling in HFS

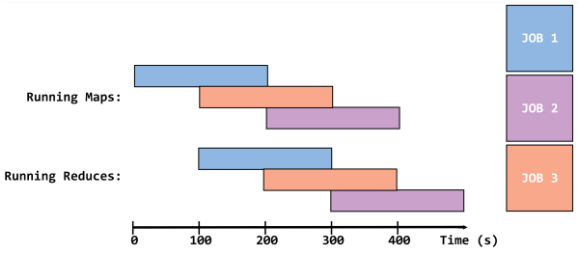
Maintain three variables for each job j, initialized as $j.level = 0$, $j.wait = 0$, and $j.skipped = false$.
when a heartbeat is received from node n:
  for each job j with $j.skipped = true$, increase $j.wait$ by the time since the last heartbeat and set $j.skipped = false$
  if n has a free slot **then**
    sort *jobs* using hierarchical scheduling policy
    for j in *jobs* **do**
      if j has a node-local task t on n **then**
        set $j.wait = 0$ and $j.level = 0$
        return t to n
      else if j has a rack-local task t on n and ($j.level \geq 1$ or $j.wait \geq W_1$) **then**
        set $j.wait = 0$ and $j.level = 1$
        return t to n
      else if $j.level = 2$ or ($j.level = 1$ and $j.wait \geq W_2$) or ($j.level = 0$ and $j.wait \geq W_1 + W_2$) **then**
        set $j.wait = 0$ and $j.level = 2$
        return any unlaunched task t in j to n
      else
        set $j.skipped = true$
      end if
    end for
  end if

Gambar 4. *Psuedocode Delay Scheduling* [11]

Pada gambar 2-2 algoritma berfungsi untuk melakukan pembagian 2 *jobs* saat *cluster* penuh dengan tugas-tugas sehingga mengakibatkan lambatnya suatu proses *job scheduler* saat waktu tunggu W1 akan mencari tugas *node-local* sedangkan waktu tunggu W2 akan mencari tugas *rack-local*.

Terdapat 3 kasus yang memiliki perbedaan jumlah *jobs*. Kasus pertama *jobs* berjumlah 3 *maps*, sedangkan kasus kedua *jobs* berjumlah 10 *maps*, dan kasus ketiga *jobs* berjumlah 100 *maps*. Kemudian setiap kasus akan diimplementasikan dengan menggunakan *Delay Scheduling* dan tidak menggunakan *Delay Scheduling*.

Delay scheduling meningkatkan *Job Throughput* 1.2 kali pada kasus pertama yang memiliki *jobs* berjumlah 3 *maps*, 1.7 kali pada kasus kedua yang memiliki *jobs* berjumlah 10 *maps*, dan 1.3 kali pada kasus ketiga yang memiliki *jobs* berjumlah 100 *maps*. Berikut ilustrasi dari *Delay Scheduler*:



Gambar 5. Ilustrasi *Delay Scheduling*

2.7 Parameter Pengujian

Parameter yang dipakai yaitu *Job Fail Rate*, *Job Throughput*, dan *Response Time*. Rincian parameter yang digunakan adalah sebagai berikut:

1) *Job Fail Rate*

Pada parameter ini akan diukur jumlah *job* yang gagal dan telah diulang kembali pekerjaannya atau yang benar-benar gagal dan merupakan kesalahan dari *resource* dari *server* Hadoop[10]. Satuan yang dipakai pada parameter ini adalah *job*.

2) *Job Throughput*

Pada parameter ini akan diukur jumlah *job* yang telah berjalan dan berhasil dalam satuan waktu. Nilai yang dihasilkan akan diperoleh dari awal suatu *job* itu berjalan hingga suatu *job* tersebut selesai dan dibagi jumlah menit yang dibutuhkan untuk menyelesaikan *job* tersebut[10]. Satuan pada parameter ini adalah *job*/menit.

3) *Response Time*

Pada parameter ini akan diukur dari waktu yang dibutuhkan dari suatu *job* ke *job*. Nilai ini akan diperoleh dari jumlah keseluruhan waktu yang dibutuhkan untuk menyelesaikan *job* yang masuk pada satu antrian[7]. Satuan yang dipakai pada parameter ini adalah menit.

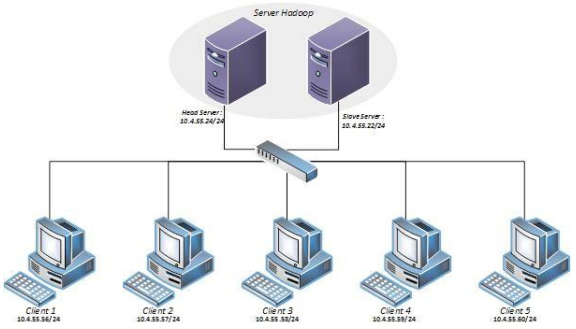
3. PERANCANGAN SISTEM

3.1 Perancangan Sistem

Perancangan sistem tugas akhir ini difokuskan pada pengujian performansi dari *job scheduling* yang dipakai pada sistem *cluster server* Hadoop. Parameter yang digunakan sebagai pembanding performansi dari *job scheduling* adalah *Job Fail Rate, Job Throughput,* dan *Response Time*. Penjelasan konfigurasi jaringan, perangkat keras, dan perangkat lunak yang digunakan dalam tugas akhir ini akan dijelaskan dibawah ini:

3.1.1 Konfigurasi Jaringan

Konfigurasi jaringan dari tugas akhir ini akan diimplementasikan dengan satu *cluster server* Hadoop yang terdiri dari 2 *server* yaitu satu *server head* dan satu *server slave* yang akan ditanamkan sistem Hadoop dan 10 komputer yang menjadi *user* akan terhubung dengan satu *switch* sebagai perantarannya. Berikut gambar rancangan jaringan yang dipakai:



Gambar 6. Rancangan jaringan

Pada gambar 3 - 1, *server* Hadoop terdiri dari dua komputer yang akan di-*cluster*. Kedua *server* tersebut berperan sebagai *master* dan *slave* kemudian dikonfigurasikan menjadi *mutli-node* dimana *head server* didesain sebagai *master* tapi dapat bekerja juga menjadi *slave*, sedangkan *slave server* didesain sebagai *slave* yang berkerja menjadi *slave*. Head *server* memiliki *Internet Protocol* (*IP*) *address* yaitu 10.4.55.22 sedangkan *slave server* memiliki *Internet Protocol* (*IP*) *address* yaitu 10.4.55.25. Pada *head server* akan ada penyettingan *Domain Name System* (*DNS*) *server* berfungsi untuk para *client* bisa meng-*submit job-job* kepada *master* dan *slave* pada Hadoop. *Client* terdiri dari 5 komputer yang setiap *client* bisa meng-*submit* lebih dari 1 *job*. Jenis *job* yang digunakan ada tiga yaitu *job wordcount, job grep*, dan *job randomtextwriter*. *Job wordcount* bekerja menghitung tingkat kemiripan dari setiap kata pada data yang sudah ditentukan *user* dalam sebuah *plaintext*, *job grep* bekerja mencari kata yang ditentukan *user* pada data yang sudah ditentukan *user* dalam sebuah *plaintext*, sedangkan *job randomtextwriter* menghitung tingkat kemiripan dari setiap kata pada data *random* dalam sebuah *plaintext*.

3.1.2 Komponen Perangkat Keras dan Perangkat Lunak

Pada topologi jaringan yang dibangun untuk tugas akhir ini seperti gambar 3-1, maka dibutuhkan komponen perangkat keras dan perangkat lunak yang mendukung. Berikut penjelasan mengenai perangkat keras dan lunak yang digunakan:

3.1.2.1 Komponen Perangkat Keras

Perangkat keras yang digunakan memiliki spesifikasi sebagai berikut:

1) Satu komputer *server* sebagai *head server* dengan prosesor *Intel Core* i7-3770 3.40Ghz, RAM 4GB dan *hard disk* 310GB.
2) Satu komputer *server* sebagai *slave server* dengan prosesor AMD Athlon 7750 Dual-Core 2.7GHz, RAM 2GB dan *hard disk* 300GB.
3) 5 buah komputer fisik maupun *virtual* dengan minimum RAM sebesar 500MB. Komputer ini akan dipakai sebagai *client* yang akan memberikan *job* pada skenario pengujian.
4) Sebuah *switch* Allied Telesis 24 port.
5) Kabel UTP kategori 5 sebagai media transmisi pada jaringan.

3.1.2.2 Komponen Perangkat Lunak

Perangkat lunak yang digunakan pada tugas akhir ini adalah sebagai berikut:

1) Sistem Operasi
   Sistem operasi yang digunakan pada *head server, slave server* dan komputer *client* adalah Linux Ubuntu 12.04.
2) *Server* Hadoop
   Server Hadoop yang dipakai adalah versi hadoop-1.2.1 STABLE *version*.
3) Java
   Untuk menunjang sistem, Hadoop memerlukan Java *Run Time Environment* dan Java *Development Kit*. Versi yang dipakai adalah Java 1.6.0_33.
4) OpenSSH
   Pada sistem Hadoop memerlukan jaringan yang aman untuk koneksinya. Oleh karena itu menggunakan OpenSSH sebagai otentikasi.
5) Bind9
   Menggunakan *server* DNS untuk penamaan dari *head server* dan *slave server* yang menggunakan perangkat lunak Bind9.
6) Maven
   Untuk build Hadoop versi hadoop-1.2.1 setelah di-*patch* atau perubahan kodingan java dalam mengeksekusi *job scheduler* pada Hadoop.
7) Mozzila FireFox
   Untuk portal dari sistem Hadoop sendiri menggunakan *browser* yaitu Mozzila FireFox. Portal disini dapat berupa *monitoring* jaringan yang ada maupun aktivitas yang dilakukan Hadoop secara *real-time*.

3.2 Proses Instalasi dan Konfigurasi

Dalam tahap pembuatan jaringan yang sesuai dengan rancangan diatas, diperlukan proses instalasi dan proses konfigurasi berguna untuk *server* berjalan sesuai dengan lingkungan yang telah diinginkan. Adapun proses instalasi dan konfigurasi sebagai berikut

3.2.1 Konfigurasi DNS *Server*

Untuk pengadaan jaringan pada Hadoop akan digunakan DNS *server* untuk penamaan *head server* dan *slave server*. Hal ini diperlukan untuk koneksi antara *head server* dan *slave* server serta dari *client* menuju server Hadoop. *Head* server dengan ip 10.4.55.22 akan dinamakan *head*, sedangkan pada *slave server* dengan ip 10.4.55.25 akan dinamakan *slave*. Konfigurasi yang dilakukan pada DNS *server* ada pada lampiran A.

3.2.2 Instalasi dan Konfigurasi Hadoop *Server Default* Algoritma *FIFO*

Instalasi dilakukan pada *head server* dan *slave server* dengan prosedur yang sama, tetapi mempunyai perbedaan pada saat konfigurasi.

3.2.3 Konfigurasi Hadoop C*luster* antara *Head Server* dengan *Slave Server*

3.2.3.1 Konfigurasi *Head Server*

Konfigurasi pada tahap ini bertujuan untuk mengenalkan seluruh *cluster* dan lingkungan luar *cluster* bahwa satu komputer ini merupakan pusat dari *server* Hadoop ini. Konfigurasi ini merupakan kelanjutan dari konfigurasi DNS *server* serta konfigurasi *server* Hadoop pada sub bab sebelumnya.

3.2.3.2 Konfigurasi *Slave Server*

Konfigurasi pada tahap ini hanya akan dilakukan pada komputer yang bertindak sebagai *slave server*. Adapun langkah-langkah yang dikonfigurasi pada lampiran D.

3.2.4 Konfigurasi Hadoop *Server* dengan Algoritma *Delay Scheduling*

Konfigurasi pada tahap ini bertujuan untuk mengubah *job scheduler* pada Hadoop dari algoritma *FIFO* (*default* Hadoop) menjadi *job scheduler* yang menggunakan algoritma *Delay Scheduling*. Adapun langkah-langkah yang dikonfigurasi pada lampiran E.

3.3 Skenario Pengujian Performansi Sistem

Skenario pengujian pada tugas akhir ini secara garis besar terbagi menjadi 7 skenario. Pembagian ini berpengaruh dari karakteristik *job* yaitu jenis *job* dan jumlah *job* yang diproses. Tujuan dilakukannya skenario ini adalah untuk mengukur performansi dari *job scheduler* pada masing masing karakteristik *job*. Pada setiap skenario akan diperoleh data acuan performansi sistem yaitu *Job Fail Rate*, *Job Throughput*, dan *Response Time*. Penjelasan pada skenario yang dibangun adalah sebagai berikut:

3.3.1 Penentuan Skenario

Pada tahap ini akan dilakukan penetapan skenario berdasarkan karakteristik *job* sebagai bahan uji dalam tugas akhir ini. Skenario yang akan dibuat salah satunya akan bergantung pada jumlah *job* yang diproses. Pembagian berdasarkan *resource* ini akan mengacu pada kelemahan dan kelebihan dari masing-masing jenis *job scheduler* yang dipakai.

```
hduser@head:/usr/local/hadoop$ bin/hadoop dfs -ls /
Found 7 items
-rw-r--r--   2 hduser supergroup 2410990224 2014-12-20 17:45 /data1.txt
-rw-r--r--   2 hduser supergroup 1609962544 2014-12-20 17:49 /data2.txt
-rw-r--r--   2 hduser supergroup  743503440 2014-12-20 17:56 /data3.txt
-rw-r--r--   2 hduser supergroup   20163420 2014-12-20 18:05 /data4.txt
drwxr-xr-x   - hduser supergroup          0 2014-12-23 12:26 /home
drwxr-xr-x   - hduser supergroup          0 2014-12-23 12:26 /user
```

Delay Scheduler menggunakan metode menunda jalannya *jobs* selanjutnya untuk memperbaiki data lokalitas sebelumnya. Sehingga dapat meminimalkan *Response Time* dan memaksimalkan *Job Throughput*[13]. Sedangkan pada *FIFO* akan diberlakukan antrian berdasarkan waktu datang. Maka penggunaan *resource* pada *FIFO* tidak akan terganggu dengan *job* yang lain.

Jenis *job* yang dipakai juga sangat berpengaruh pada penentuan skenario yang akan dibuat. *Job* yang digunakan yaitu *job wordcount*, *job grep*, dan *job randomtextwriter*. Setiap *job* memiliki karakteristik seperti *job wordcount*, *job grep*, dan *job randomtextwriter* memiliki waktu eksekusi yang berbeda. Sehingga ketiga *job* ini akan didapatkan variable waktu eksekusi *job* yang berbeda. *Resource* data dalam pengaksesan file diambil dari:

Gambar 7. *Resource* data

Skenario *job* pada tugas akhir ini dibedakan menjadi dua bagian yaitu *job* yang menggunakan *default scheduler* Hadoop yaitu *FIFO* dan *job* yang menggunakan algoritma *Delay Scheduling*.

Berikut tabel skenario 1 sampai dengan skenario 7 dengan spesifikasi jumlah *job* dan jenis *job* yang akan dipakai pada masing-masing skenario.

Pada skenario 1, 2, dan 3 menggunakan masing-masing 1 jenis *job* yaitu *job wordcount, job grep*, dan *job randomtextwriter*. Kemudian skenario 4, 5, dan 6 menggunakan kombinasi 2 jenis *job* yaitu antara *job wordcount* dengan *job grep*, *job wordcount* dengan *job randomtextwriter*, dan *job grep* dengan *job randomtextwriter*. Sedangkan skenario 7 menggunakan kombinasi 3 jenis *job* yaitu *job wordcount, job grep*, dan *job randomtextwriter*.

Semua skenario terdiri dari lima client yang jumlah *job*-nya diberikan tidak harus sama setiap *user* karena jumlah *user* tidak mempengaruhi performansi *scheduling* melainkan jumlah *job*. Hal ini dilakukan untuk dapat meneliti antara antrian yang memiliki *job* berjenis sama dan antrian yang memilik i jenis *job* yang berbeda yang penempatan jenis *job* dilakukan secara acak.

4. ANALISIS HASIL PENGUJIAN PERFORMANSI SISTEM

Pada tahap ini akan dibahas mengenai analisis dari hasil pengujian sesuai dengan skenario yang telah dibuat. Pengujian dilakukan bertujuan untuk mengetahui *job scheduler* yang cocok untuk beberapa macam karakterisitik *job* yang dimunculkan pada skenario. Untuk parameter yang dipakai dalam penentuan performansi sistem yaitu *Job Fail Rate*, *Job Throughput*, dan *Response Time*. Dalam pengujian ini akan dibandingkan dua *job scheduler* antara *FIFO* dan *delay scheduling*.

4.1 Analisis Keseluruhan Pengujian

Analisis berisikan rangkuman *job scheduler* dari seluruh skenario antara *FF* (*FIFO*) dan DS (*Delay Scheduling)* sesuai dengan parameter yang digunakan.

Karakteristik dari *Delay Scheduling* yaitu memperbaiki data *locality job* namun semakin banyak jumlah *jobs* maka performasi *Job Fail Rate* menurun dengan munculnya *fail*. Pada *FIFO* nilai maksimal *Job Fail Rate* yaitu 10% sedangkan *Delay Scheduling* nilai maksimal *Job Fail Rate* yaitu 1%.

Karakteristik jenis *job wordcount* yaitu mengeksekusi *job* ke *job* selanjutnya lebih cepat dari *job* sebelumnya dan pengaruh nilai *fail* yang kecil mengakibat *jobtracker* tidak banyak mengulang pengeksekusi *maps* yang *fail* sehingga meminimalkan waktu *running* setiap *job*. Pada jenis *job grep* yaitu memiliki jumlah *job* dua, *grep-search* dan *grep-sort*. Dimana *grep-sort* akan masuk antrian *jobs* setelah *grep-search* selesai dieksekusi dan membutuhkan waktu tambah dalam proses *grep-sort* masuk kedalam antrian. Sedangkan pada *jenis job randomtextwriter* yaitu mengeksekusi *job* secara *random* dengan total *maps* selalu sama pada setiap *job* sehingga adanya kestabilan jumlah *maps* pada setiap *job*.

Delay Scheduling memiliki performansi efektif daripada *FIFO* pada jenis *job wordcount* dengan dengan penurunan 0.3% *Job Fail Rate*, peningkatan 8.853% *Job Throughput*, dan lebih cepat 142 menit 45 detik *Response Time* pada jumlah *job* 50 *jobs*.

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan tujuan serta hasil dari pengujian dan analisis yang telah dilakukan pada penggunaan *FIFO* dan *Delay Scheduling* sebagai *job scheduling* pada Hadoop, maka dapat diambil beberapa kesimpulan sebagai berikut:

1. *Delay Scheduling* memiliki performansi efektif daripada *FIFO* pada jenis *job wordcount* dengan dengan penurunan 0.3% *Job Fail Rate*, peningkatan 8.853% *Job Throughput*, dan lebih cepat 142 menit 45 detik *Response Time* pada jumlah *job* 50 *jobs*.
2. *Delay Scheduling* memiliki performansi *Job Fail Rate* maksimal 1% sedangkan FIFO memiliki performansi Job Fail Rate maksimal 10% dari setiap skenario.
3. *Delay Scheduling* pada jenis *job* yang mengandung *job grep* memiliki nilai maksimal *Job Throughput* pada jumlah *job* 10 *jobs* kemudian nilai *Job Throughput* turun seiring pertambahan jumlah *job*.

5.2 Saran

Pengembangan dari tugas akhir ini dapat dilakukan dengan menambahkan jumlah *server* Hadoop, menambahkan jumlah *jobs*, menggunakan jenis job lain, serta memodifikasikan algoritma *job scheduler* lain yang dapat diimplementasi pada *server* Hadoop.

6. REFERENSI

[1] B.Thirumala Rao, Dr. L.S.S.Reddy, *Survey on Improved Scheduling in Hadoop MapReduce in Cloud Environments*, India, 2011

[2] Guo Zhenhua, Fox Geoffrey, Zhou Mo, *Investigation of Data Locality in MapReduce*, USA

[3] Hammound Mohammad, Sakr F. Majd, *Locality-Aware Reduce Task Scheduling for MapReduce*, State of Qatar

[4] He Chen, Ying Lu, David Swanson, *Matchmaking: A New MapReduce Scheduling Technique*, Nebraska

[5] Kim Woo-Cheol, Changryong Baek, Dongwon Lee, *Measuring the Optimality of Hadoop Optimization*, Seoul, 2013

[6] L. Abad Cristina, Lu Yi, H. Campbell Roy, *Dare: Adaptive Data Replication for Efficient Cluster Scheduling*, Urbana-Champaign

[7] Rasooli Aysan, Douglas G. Down, *A Hybrid Scheduling Approach for Scalable Heterogeneous Hadoop Systems*, Canada

[8] Rasooli Aysan, Douglas G. Down, *Guidelines for Selecting Hadoop Schedulers based on System Heterogeneity*, Hamilton

[9] White Tom, *Hadoop: The Definitive Guide, Third Edition*, United States of America,2009

[10] XIA Yang, Lei WANG, Qiang ZHAO, Gongxuan ZHANG, *Research on Job Scheduling Algorithm in Hadoop*, China, 2011

[11] Yoo Dongjin, Sim Mong Kwang, *A Comparative Review of Job Scheduling for MapReduce*, Republic of Korea, 2002

[12] Zaharia Matei, Dhruba Borthakur, Joydeep Sen Sarma, Khaled Elmeleegy, Scott Shenker, Ion Stoica, *Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling*, California

[13] Zaharia Matei, Dhruba Borthakur, Joydeep Sen Sarma, Khaled Elmeleegy, Scott Shenker, Ion Stoica, *Job Scheduling for Multi-User MapReduce Clusters*, California, 2009