Analisis Penggabungan *Delay Scheduling* dan *Fair share Scheduling Algorithm* Dengan beberapa karakteristik *Job* pada Hadoop

Analysis Combination Delay Scheduling and Fair share Scheduling Algorithm with Job Characteristics On Hadoop

Tri Retno Pamungkas Sujarwo[1], Fazmah Arif Yulianto, S.T., M.T.[2], Sidik Prabowo, S.T., M.T.[3]

[1,2,3]Prodi S1 Teknik Informatika, Fakultas Informatika, Universitas Telkom

[1] triretnosujarwo@gmail.com, [2] fazmaharif@telkomuniversity.ac.id ,[3] Pakwowo@telkomuniversity.ac.id

Abstrak

***Scheduling* Hadoop merupakan cara untuk mengatur setiap *job* yang akan berjalan pada sistem Hadoop agar dapat mengelola semua *job* yang ada untuk mendapatkan giliran untuk di eksekusi pada setiap *resource* yang tersedia. Default *Scheduling* pada Hadoop yaitu FIFO yang mempunyai karakteristik untuk setiap *job* yang masuk pertama akan di eksekusi langsung dan berhak memonopoli satu *resource* secara utuh. Namun FIFO memiliki kerugian bagi proses *short job* ketika yang dieksekusi adalah proses *long job*.**

Delay improve Fair share* merupakan *Job scheduler* yang menggunakan metode dengan membagi *job* untuk satu cluster ke dalam beberapa *pool* dan setiap *pool* diberlakukan metode menunda jalannya *jobs* selanjutnya untuk memperbaiki data lokalitas sebelumnya. *Delay improve Fair share* memiliki performasi efektif daripada *Fair share* dan *Delay Scheduling* pada jenis *job* randomtextwriter dengan penurunan 0,3% *job* fail rate dengan nilai *throughput* 2,73 *job*/m dan 273,59 menit lebih cepat dari *Delay Scheduling* dan 128,15 menit lebih cepat dari *Fair share.

Kata kunci: hadoop, hadoop multi-node, *Fair share improve Delay Scheduling, Delay Scheduling*

Abstract

Scheduling Hadoop is a way to set up any job that will run on hadoop system that can manage all the jobs to get a turn in the execution on every resource available. Default Scheduling in Hadoop, FIFO that has characteristics first come first excecution and monopoly of the entire resource. But FIFO has disadvantage for short job that run into starvation when a long job in execution.

Delay improve Fair share is a job scheduler that use a methode to divide job for one cluster in some pool. For each pool that has been divided, put in place a method to Delay the course of the next jobs to improve data locality before. Delay improve Fair share have effective performance than Fair share and Delay Scheduling in the kind of job randomtextwriter with a reduction of fail rate 0,3% job and increase produce of throughput 2,73 job/m and faster 273,59 minutes than Delay Scheduling and 128,15 minutes than Fair share.

Keywords: Fair share improve Delay Scheduling, Delay Scheduling

1. Pendahuluan

Peningkatan akan suatu data pada saat ini akan terus bertambah seiring semakin besarnya suatu kebutuhan. Besarnya suatu data atau *Big Data* justru tidak berjalan seimbang dengan kecepatan bagaimana data itu diolah dan diproses sehingga suatu data dapat diterima dengan baik oleh pengguna data[4]. Hadoop merupakan *framework software* berbasis *java* dan *open-source* yang berfungsi untuk mengolah data yang besar secara terdistribusi dan berjalan di atas *cluster* yang terdiri atas beberapa komputer yang saling terhubung[**1**].

Fair share Scheduling merupakan *job scheduler* yang menggunakan metode dengan membagi *job* untuk satu cluster ke dalam beberapa *pool*, sehingga algoritma ini hanya memiliki keunggulan dalam hal *Fairness resource*. Sedangkan *Delay Scheduling* memiliki kinerja dengan cara menunda jalannya *jobs* selanjutnya untuk memperbaiki data lokalitas sebelumnya. Keuntungan dari penerapan *Delay Scheduling* ini yaitu dapat meningkatkan *locality* data.Algoritma *Fair share improve Delay Scheduling* dapat mencapai kedua keunggulan dari masing-masing algoritma tersebut dengan cara membagi *job* untuk satu cluster ke dalam beberapa pool dan menerapkan penundaan job selanjutnya pada setiap pool untuk memperbaiki lokalitas data.

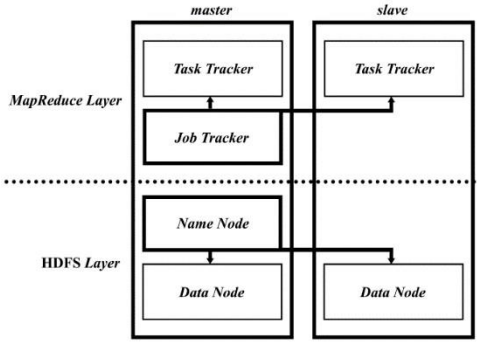
2. Dasar Teori

2.1 Hadoop Single-Node

Hadoop *single-node* merupakan *server* yang hanya memiliki satu *server* yang bekerja 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*. Pada Layer *Hadoop Distributed File System* (HDFS) menjalankan *namenode* dan *datanode* sedangkan layer *MapReduce* menjalankan *jobtracker* dan *tasktracker*. Pada kedua layer ini, bagian yang sangat penting adalah *namenode* dan *jobtracker*. *Jobtracker* merupakan *server* penerima *job* dari *client*, Sedangkan Namenode merupakan *server* yang mengatur *system file namespace* dan hak akses *file* oleh *client*[8].

2.2 Hadoop Multi-Node

Hadoop *multi-node* merupakan gabungan 2 *server single-node* yaitu 1 untuk *server master* dan 1 untuk *server* slave. *Server master* dapat bekerja juga *sebagai server slave* dan *server slave* hanya bekerja sebagai *server slave*.



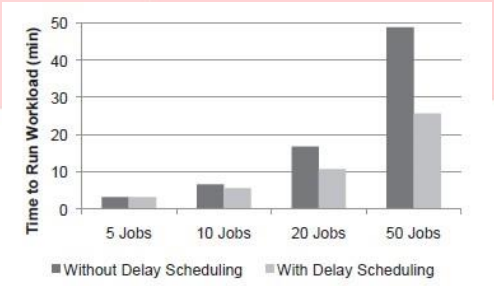
Gambar 2 - 1 Arsitektur Hadoop Multi-Node

2.3 Job Scheduling

Job scheduling adalah suatu cara Bagaimana Hadoop dapat mengelola pemrosesan data agar output yang dikeluarkan sesuai dengan apa yang diharapkan.*Default Scheduling* Hadoop yaitu FIFO, namun *Scheduling* ini dapat di kostum sesuai dengan kebutuhan *user*.

2.4 Delay Scheduling Algorithm

Pada *Delay scheduler* akan menggunakan metode menunda jalannya *jobs* selanjutnya untuk memperbaiki data lokalitas sebelumnya dengan cara pengalokasian data yang hampir optimal dalam mengolah beban kerja sehingga dapat meningkatkan *throughput* hingga 2 sampai 10 kali dengan waktu respon yang rendah[2][3].

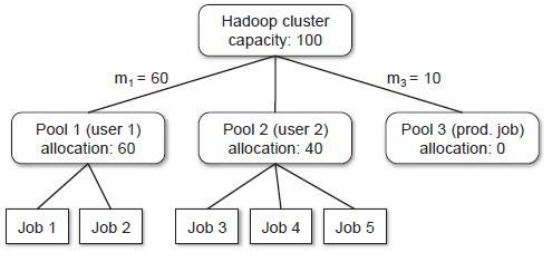


Gambar 2 - 2 peningkatan throughput pada Delay Scheduling[2]

Pada gambar 2 - 4 terdapat 4 kasus berbeda dalam jumlah *job* yang di submit. Pada kasus pertama dengan jumlah 5 *job* tidak terlihat perbedaan yang *significant* antara *scheduler* yang mengimplementasikan *Delay Scheduling* dengan *scheduler* yang tidak mengimplementasikan *Delay scheduler*. Pada kasus kedua dengan jumlah *job* 10 terlihat peningkatan throught 1,1 kali, sedangkan pada kasus ketiga dengan jumlah *job* 20 terjadi peningkatan *Throughput* 1,6 kali, dan pada kasus keempat dengan jumlah *job* 50 terjadi peningkatan *throughput* 2 kali. Hal ini tentunya membawa persentase lokalitas data dari yang semula 27-90% menjadi 99-100% [2].

2.5 Fair Share Scheduling Algorithm

Fair Scheduler akan menggunakan metode yang menentukan suatu *jobs* akan mendapatkan *resource* yang sama dengan *jobs* yang lain. Maka setiap pekerjaan yang berjalan akan mendapatkan *resource* yang sama. Dengan begitu, *short jobs* tidak akan menunggu lama pada rersource yang sedang digunakan oleh *long jobs*.

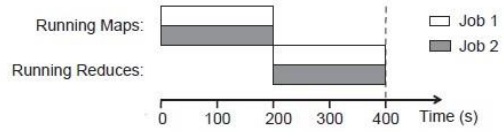


Gambar 2 - 3 Ilustrasi pembagian pool pada Fair share[2]

Gambar diatas menjelaskan bahwa setiap *user* akan memiliki nilai yang dapat dipakai untuk melakukan suatu pekerjaan. Nilai tersebut akan masuk kedalam slot *resource* yang mempunyai batas nilai. Setiap pekerjaan yang telah selesai akan keluar dari slot *resource*, dan *resource* kembali memiliki slot yang kosong, sehingga dapat diisi oleh *user* lain

2.6 Fair Share improve Delay Scheduling

Pada *Fair share improve Delay Scheduling*, metode yang digunakan adalah membagi setiap *job* pada satu *cluster* ke beberapa pool dengan membatasi minimal *Share* untuk setiap pool nya sehingga *resource* yang tersedia dapat terbagi secara adil untuk setiap *job* yang di proses dan pada setiap pool nya akan di implementasikan metode menunda jalannya *jobs* selanjutnya untuk memperbaiki data lokalitas sebelumnya,sehingga dapat meminimalkan waktu respon dan memaksimalkan waktu *throughput*[2].



Gambar 2 - 4 Ilustrasi Fair share Improve Delay Scheduling Scheduler[2]

2.7 Parameter Pengujian

1. *Job* Fail Rate

   Pada parameter ini akan diukur jumlah *job* yang gagal dan telah diulang kembali pekerjaannya atau yang benar-benar [6]. Satuan yang dipakai pada parameter ini adalah *job*.

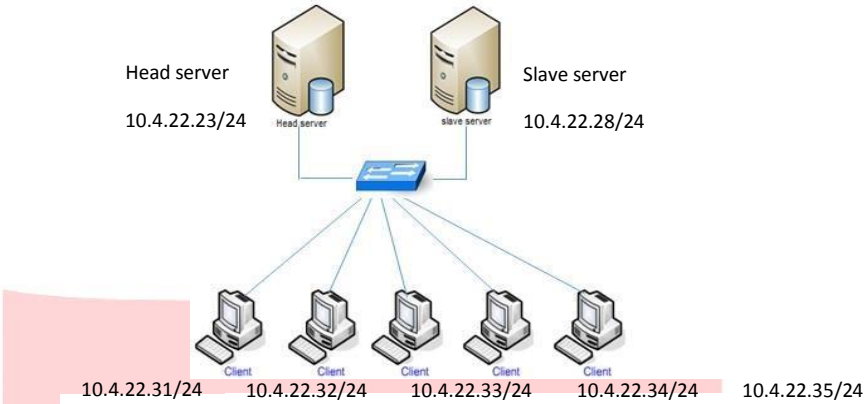
2. *Job Throughput*

   Pada parameter ini akan diukur jumlah *job* yang telah berjalan dan berhasil dalam satuan waktu. Nilai yang dihasilkan diperoleh dari awal suatu *job* itu berjalan hingga suatu *job* tersebut selesai dan dibagi jumlah menit yang dibutuhkan untuk menyelesaikan *job* tersebut[6]. 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[5]. Satuan yang dipakai pada parameter ini adalah menit.

3. Perancangan dan Implementasi

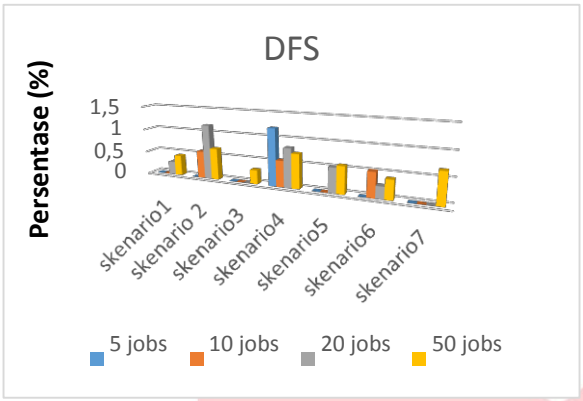


Gambar 3 -1 Arsitektur fisik jaringan

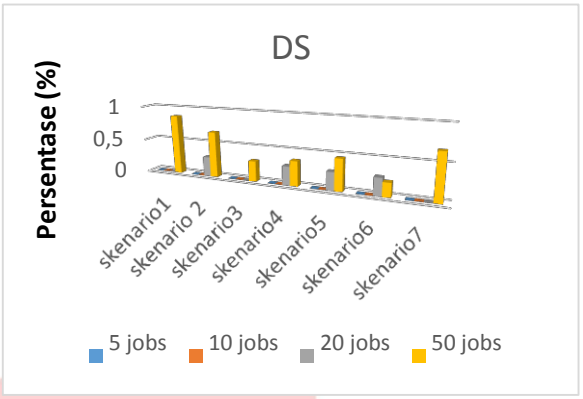
Pada gambar 3 - 1, *server* Hadoop terdiri dari 2 buah komputer yang akan dijadikan sebuah cluster. Kedua *server* tersebut akan berperan sebagai master dan slave kemudian akan dikonfigurasikan menjadi Hadoop multi-node cluster, dimana Head *server* akan bekerja sebagai master dan slave sedangkan slave *server* akan bekerja sebagai slave. Hadoop head *server* memiliki IP *(Internet Protocol*) 10.4.22.23/24 dan slave *server* memiliki IP 10.4.22.28/24. Client terdiri dari 5 buah komputer yang digunakan sebagai pengirim *job* pada *server*. Komputer client dapat mengirim lebih dari satu *job*. *Job* yang dikirimkan client ada tiga jenis yaitu *job wordcount* yang bekerja menghitung jumlah kata yang sama pada data yang sudah ditentukan *user* dalam sebuah plaintext, *job grep* bekerja mencari kata yang ditentukan *user* pada data yang sudah di tentukan *user* dalam sebuah plaintext, sedangakan *job randomtextwriter* bekerja menuliskan 10 GB *random* data pada DFS Map/Reduce

4. Hasil Pengujian

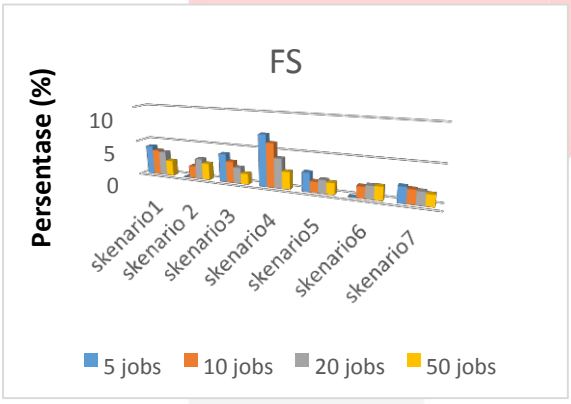
Dalam pengujian ini akan dibandingkan tiga *scheduler* yaitu *Delay improve Fair share*, *Delay Scheduler*, dan *Fair share. Resource* data dalam pengaksesan file setiap *job* ini beragam yaitu 2.4GB, 1.6GB, 743MB dan 20 MB dengan jumlah *job* yang beragam yaitu 5 *jobs*, 10 *jobs*, 20 *jobs*, dan 50 *jobs*. 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* yatu 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, jon randomtextwriter.*



Grafik 4 – 1 Grafik Rangkuman Parameter Fail Rate DFS

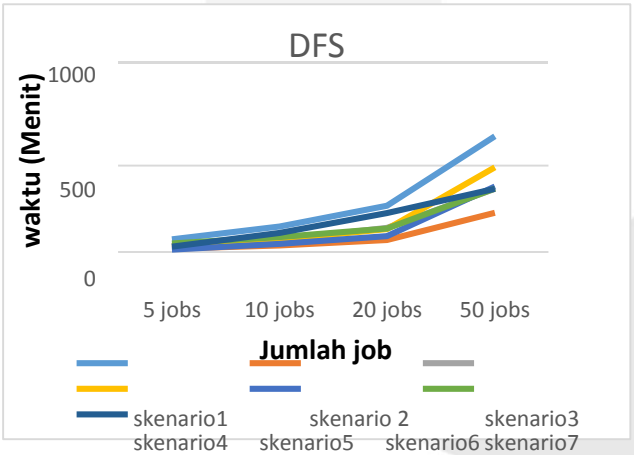


Grafik 4 – 2 Rangkuman Parameter Fail Rate DS

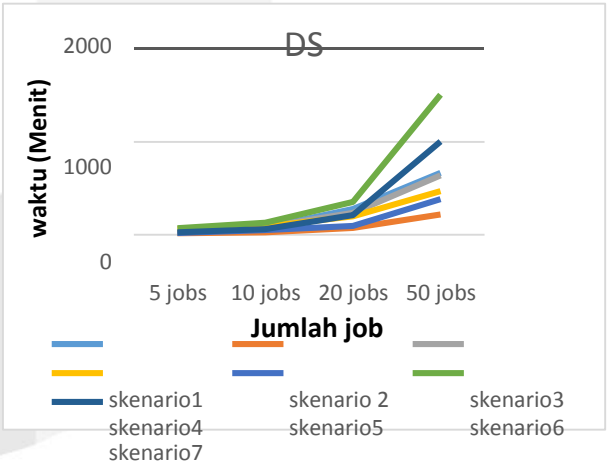


Grafik 4 – 3 Grafik Rangkuman Parameter Fail Rate DFS

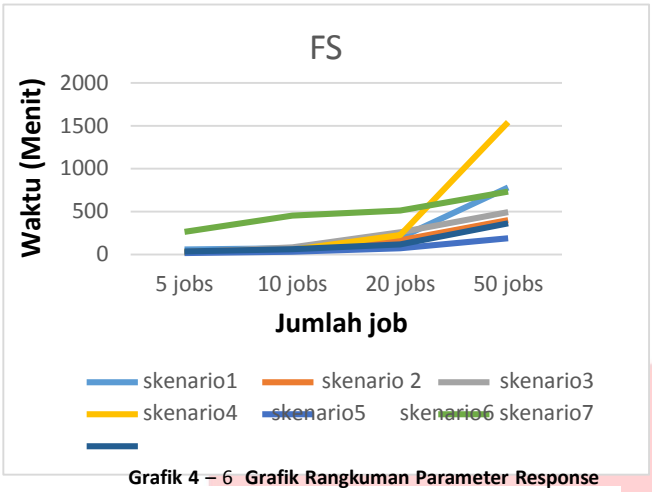
kemunculan fail pada algoritma *Delay improve Fair share* cenderung merata pada peningkatan jumlah *jobs*, hal ini dikarenakan karakteristik dari *Fair share* yang sudah mempunyai nilai fail pada setiap *job* yang akan masuk queue, dan tentu berbeda dengan *Delay Scheduling* karakteristiknya yang masih menggunakan pola FIFO (First In First Out), sehingga nilai kemunculan fail nya cenderung muncul pada *job* yang mendekati jumlah *job* Grafik



Grafik 4 – *4* Grafik Rangkuman Parameter Response Time DFS

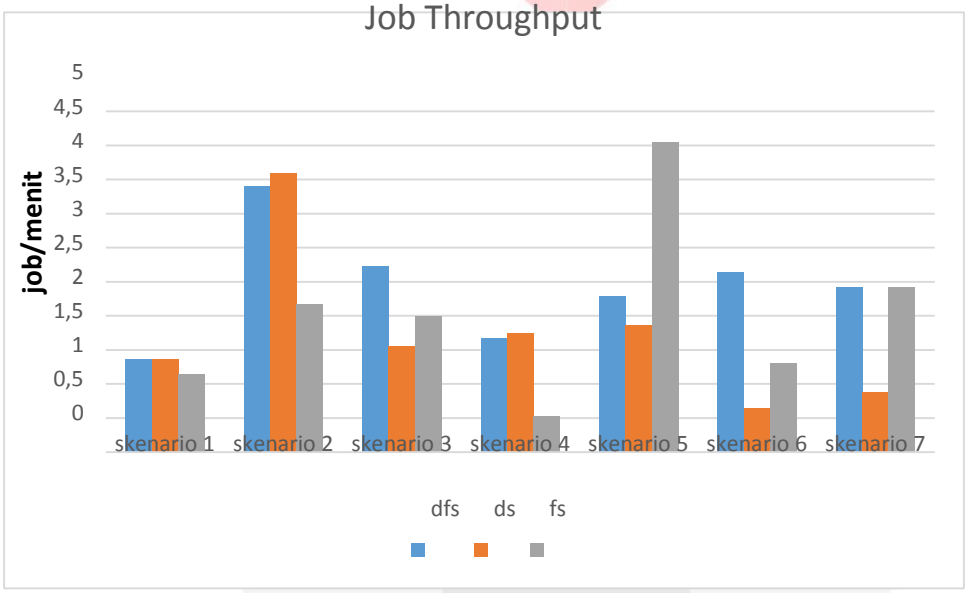


Grafik 4 – *5* Grafik Rangkuman Parameter Response Time DS



Grafik 4 – 6 Grafik Rangkuman Parameter Response

Pada *scheduler* DFS untuk parameter Response Time, apabila dibandingkan secara keseluruhan dari skenario satu sampai tujuh dengan DS dan FS, DFS cenderung lebh banyakmenghabiskan waktu dengan range antara 200 menit hingga 600 menit, hal ini terjadi karena karakteristik dari *Delay improve Fair share* yang bekerja secara paralel dan memperbaiki data localtiyy *job* sebelumnya sehingga *job*tracker tidak perlu banyak mengulang *job* yang fail, sehingga waktu yang dibutuhkan untuk mengerjakan keseluruhan *job* kecil.



Grafik 4 – 7 Grafik Rangkuman Parameter Job Throughput

Pada Parameter *job throughput*, *Delay improve Fair share* nilai *job throughput* yang paling tinggi yakni 3,9 *job* per menit yang didapat dari skenario ke 2. Sedangkan untuk DS dan FS memiliki nilai *job throughput* 4,08 *job* per menit dan 4,54 *job* per menit. Waktu yang dibutuhkan dalam menyelesaikan *job* keseluruhan tentunya akan mempengaruhi nilai *job throughput* yang didapat. Hal ini bisa terjadi karena adanya pengaruh dari jumlah banyak tidaknya *job* yang gagal dan waktu yang dibutuhkan untuk menyelesaikan *job*. Semakin banyak *job* yang gagal dan lama waktu untuk menyelesaikan *job*, maka nilai *job throughput* akan semakin kecil, begitupun sebaliknya.

5. Kesimpulan dan Saran

Berdasarkan tujuan serta hasil dari pengujian dan analisis yang telah dilakukan, maka dapat ditarik kesimpulan sebagai berikut:

5.1 Kesimpulan

1. Pada parameter *Fail Rate*, *Scheduler Delay improve Fair share* lebih baik bila dibandingkan dengan *Scheduler Fair Share* yang memiliki nilai *fail* lebih kecil.
2. Pada parameter *Response Time*, waktu yang diperlukan untuk menyelesaikan *job* keseluruhan skenario pada *Delay improve Fair share* memiliki maksimum response time yang lebih rendah dan rentang waktu keseluruhan yang juga lebih kecil.
3. Pada parameter *Job throughput*, nilai *job throughput* pada *Delay improve Fair share* cenderung tidak stabil, hal ini dikarenakan perbedaan karakteristik *job* yang menyebabkan naik turunnya nilai *fail* dan *response time*.
4. *Delay Scheduling improve Fair share* lebih baik digunakan untuk mengerjakan jenis *job* yang memiliki karakteristik menulis data seperti *randomtextwriter*.

5.2 Saran

Pengembangan dari tugas akhir ini dapat dilakukan dengan memodifikasi Algoritma *job scheduler* lain, mengubah jumlah server hadoop menjadi lebih banyak, serta menambahkan keragaman *job* yang dapat merepresentasikan *job* pada keadaaan yang sebenarnya.

6. Daftar Pustaka

[1] Intel Corporation 2012, "Planning Guide Getting started with Hadoop," 2012.

[2] R. Aysan dan D. G.Down, Guidelines for Selecting Hadoop Scheduler based on System Heterogenity, Hamilton.

[3] Bughin, Chin dan Manyika, "Big Data and Analytics : Startegic and Organizational Impacts," 2010.

[4] P. Dijcks dan Jean, "Oracle : Big Data for the Enterprise," 2013.

[5] Y. Dongjin dan S. Mong kwang, "A Comparative Review of Job Scheduling for Mapreduce," 2002.

[6] P. Jorda, C. Claris, C. David, B. Yolanda, W. Ian, S. Malgorzata dan T. Jordi, "Resource-aware Adaptive Scheduling for Mapreduce Clusters," 2010.

[7] Z. Matei, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker dan I. Stoica, "Delay Scheduling : A simple Technique foe Achieving Locality and Fairness in Cluster Scheduling".

[8] Z. Matei, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker dan I. Stoica, "Job Scheduling for Multi-User Mapreduce Cluster," 2009.