

# Implementasi Algoritma Penjadwalan untuk pengelolaan Big Data dengan Hadoop

Sidik Prabowo <sup>#1</sup>, Maman Abdurohman <sup>\*2</sup>

*School of Computing, Telkom University*

*Jalan Telekomunikasi no 1, Bandung, Jawa Barat. Indonesia 40257*

pakwowo@telkomuniversity.ac.id

abdurohman@telkomuniversity.ac.id

## Abstract

This paper proposes a hadoop scheduler scheme on completing the appropriate job type for Hadoop performance improvement. The suitability of the scheduler type and the type of job done can increase throughput and decrease the average job completion time. The main problem with job execution is the mismatch between the scheduler and the type of job being done. In this paper we have tested several Hadoop scheduler algorithms namely FIFO, Fair, SARS and COSHH scheduler with some kind of job which handled in hadoop environment. The types of jobs that are tested are word count, random text writer and grep. The test is done two scenarios, namely homogeneous job (one type) and heterogeneous (some kind of job) done together. The results show that SARS algorithm suitable for use on job completion which is homogeneous. Meanwhile, COSHH algorithm is suitable for heterogeneous combined job.

**Keywords** : Hadoop, scheduler, SARS,FAIR, COSHH, dan job.

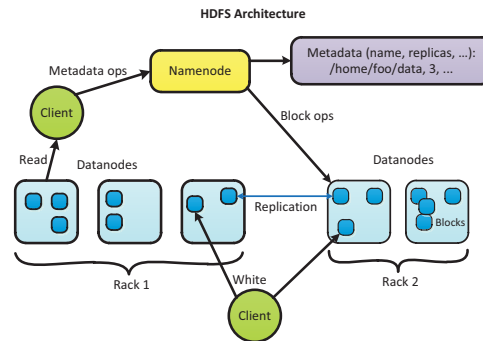
## Abstrak

Paper ini mengusulkan skema *scheduler* hadoop pada penyelesaian tipe *job* yang sesuai untuk peningkatan kinerja Hadoop. Kesesuaian jenis *scheduler* dan tipe *job* yang dikerjakan dapat meningkatkan *throughput* dan menurunkan waktu rata-rata penyelesaian *job*. Masalah utama pada eksekusi *job* adalah ketidaksesuaian antara *scheduler* dengan tipe *job* yang dikerjakan. Pada paper ini telah dilakukan pengujian terhadap beberapa algoritma *scheduler* Hadoop yaitu FIFO, Fair, SARS dan COSHH *scheduler* dengan beberapa jenis *job* yang ditangani dalam lingkungan hadoop. Jenis-jenis *job* yang diujikan adalah *word count*, *random text writer* dan *grep*. Pengujian dilakukan dua skenario, yaitu *job homogen* (satu jenis) dan *heterogen* (beberapa jenis *job*) dikerjakan bersama. Hasil pengujian menunjukkan bahwa algoritma SARS cocok digunakan pada penyelesaian *job* yang sifatnya *homogen*. Sementara itu algoritma COSHH cocok digunakan pada penyelesaian *job* gabungan yang *heterogen*.

**Kata kunci**: Hadoop, *scheduler*, SARS, COSHH, dan *job*.

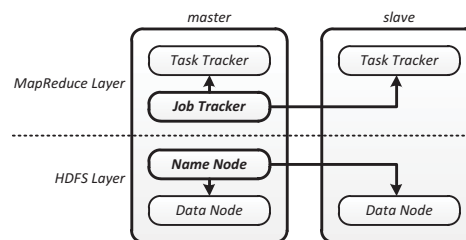
## I. PENDAHULUAN

**H**ADOOP merupakan salah satu sistem file terdistribusi yang diperuntukan untuk mengerjakan *job-job* untuk data yang masuk kategori besar (*Big Data*). Arsitektur *Hadoop Distributed File System* (HDFS) yang ditunjukkan pada gambar 1. terdiri dari dua bagian yaitu *Namenode* dan *Datanode*. *Namenode* merupakan *node* utama yang berkomunikasi dengan *client* untuk menyelesaikan *job*.



Gambar 1. Arsitektur HDFS [1]

Pada satu sistem terdapat sejumlah *datanodes* yang didalamnya data disimpan secara *redundan*. Dalam sistem Hadoop terdapat dua lapisan yaitu lapisan *MapReduce* dan lapisan HDFS. Gambar 2. menunjukan lapisan pada Hadoop. Pada masing-masing lapisan terapat peran *node master (masternode)* dan *node slave (datanode)*. Pada lapisan *MapReduce* terdapat dua proses penting yaitu *Map* dan *Reduce*. Kedua proses ini merupakan keunggulan sistem Hadoop dibandingkan dengan sistem terdistribusi yang lain.



Gambar 2. Lapisan pada Hadoop [1]

Setiap job yang masuk ke dalam master akan dilakukan proses *map* dan *reduce*. Pada proses ini diperlukan *job scheduler* sebagai sistem manajemen penjadwalan eksekusi job. Terdapat beberapa algoritma yang dibangun untuk memastikan bahwa proses ini berjalan cepat dan akurat. Melihat karakteristik *Big Data* saat ini, dimana baik ukuran maupun variasi jenis dan tipe data yang sangat besar, hal ini akan menyebabkan terdapatnya sekumpulan *job* yang harus diselesaikan oleh *master* node dalam waktu yang hampir bersamaan. *Job* dalam kategorisasinya dapat dikelompokkan menjadi *homogen (job yang sama)* atau dilakukan secara bersamaan antar *job* yang *homogen (job yang berbeda)*. Kedua karakteristik *job* ini memerlukan algoritma yang berbeda sehingga diperoleh hasil proses yang efektif dilihat dari sisi *throughput* dan waktu rata-rata pengerjaan *job*. Semakin besar *throughput* dan semakin kecil rata-rata pengerjaan *job*, menunjukkan sistem semakin berjalan dengan baik.

Pada paper ini dilakukan pengujian algoritma scheduler yang tersedia untuk dijalankan diatas Hadoop yaitu: FIFO, Fair, SARS dan COSHH. Kedua scheduler pertama yaitu FIFO dan Fair adalah scheduler asli yang dibawa pada Hadoop, sementara itu SARS dan COSHH merupakan algoritma pengembangan dari algoritma scheduler sebelumnya. Pengujian dilakukan pada dua tipe job yaitu homogen (satu jenis job) dan heterogen (beberapa jenis job berbeda yang dijalankan secara bersamaan). Hasil pengujian ini merupakan rekomendasi penggunaan scheduler yang sesuai untuk tipe job yang dijalankan pada Hadoop.

II. TINJAUAN PUSTAKA

*Scheduler* pada Hadoop bertujuan untuk mengelola pencarian maupun pemrosesan data supaya dapat dieksekusi lebih cepat. Secara default Hadoop menggunakan First In First Out (FIFO) dan Fair untuk penjadwalan eksekusi *Job*.

A. FIFO (*First In First Out*)

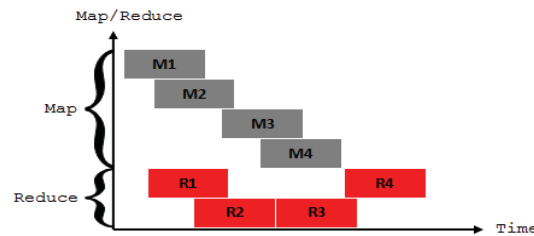
Mekanisme dari metode penjadwalan FIFO adalah dengan mengeksekusi *job* yang masuk kedalam antrian lebih dahulu, diikuti *job* selanjutnya sesuai dengan urutan kedatangan. Algoritma FIFO tidak menjalankan penanganan skala prioritas serta perhitungan baik dari kategorisasi *long jobs* atau *short jobs* maupun parameter yang lain, dimana hal ini mengakibatkan penggunaan algoritma FIFO kurang efektif untuk beberapa jenis *job* tertentu[2]. FIFO akan tetap mempertahankan *job* yang sedang dieksekusi pertama kali saat *job* yang lain akan masuk[3]. FIFO akan memberikan kesempatan *job* untuk menggunakan *resource* yang ada sesuai dengan waktu kedatangan dan apabila *resource* kosong, nantinya akan dilanjutkan oleh *job* selanjutnya [2]. Penggunaan MapReduce pada algoritma FIFO juga mengikuti pertama kali *map* dimulai[3]. Berikut ilustrasi dari FIFO berdasarkan fungsi reduce pada hadoop:



Gambar 3. Ilustrasi FIFO untuk *reduce* Hadoop[2]

B. Fair

Fair *scheduler* bekerja dengan cara mendistribusikan *resource* secara sama rata (*Fair*), untuk setiap *job* yang masuk kedalam antrian. Ketika hanya terdapat satu *Job* yang sedang berjalan, *job* tersebut akan menggunakan keseluruhan sumber daya yang tersedia. Ketika terdapat *job* baru masuk untuk dieksekusi, sumber daya digunakan oleh *job* sebelumnya akan dibebaskan alokasinya dan ditugaskan untuk eksekusi *job* yang baru, sehingga setiap *job* pada akhirnya mendapatkan jumlah sumber daya yang kira-kira sama. Tidak seperti FIFO, yang membentuk antrian *Job*, pada metode ini memungkinkan untuk waktu eksekusi *job* yang pendek selesai dieksekusi dalam waktu singkat, tanpa mengorbankan *job* yang lebih panjang.

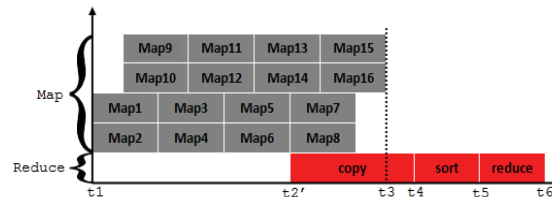


Gambar 4. Ilustrasi Fair *Scheduler* untuk *reduce* Hadoop[2]

Fair juga mampu melakukan pembagian sumber daya yang adil ,serta dapat bekerja dengan memberikan prioritas *job* yang selanjutnya dapat digunakan sebagai bobot untuk menentukan bagian dari total sumber daya yang harus diperoleh masing-masing *job*.

C.SARS (*Self-Adaptive Scheduling Algorithm for Seduce Start Time*)

Algoritma penjadwalan SARS dapat mengurangi *completion time* dan waktu yang dibutuhkan untuk proses *reduce* pada slot *resources* [3]. Penyimpanan slot ini tidak terjadi apada algoritma FIFO. Pengurangan *completion time* ini akan memaksimalkan *response time* [3]. Dengan semakin kecilnya *response time* yang diperoleh maka akan berpengaruh pada parameter *job throughput* yang semakin maksimal [4]

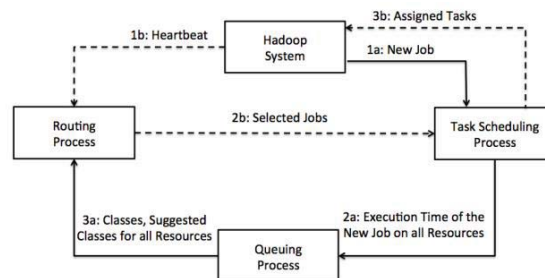


Gambar 5. Ilustrasi Scheduler SARS

Penggunaan *MapReduce* pada algoritma SARS akan dimulai cukup jauh atau setengah dari dimulainya proses map, hal ini dikarenakan algoritma SARS akan men-*delay* waktu *reduce*, akan tetapi sebelum map selesai yang nantinya dapat mengurangi *response time* [5]. Atas *delay* yang dilakukan pada *reduce task*, akan mengakibatkan *jobs* dalam slot *resources* tidak bekerja secara normal [5]. Dikarenakan menyesuaikan pengurangan waktu penjadwalan, metode ini akan mengurangi waktu yang terbuang untuk replikasi data dan mengoptimalkan pemanfaatan slot *resources* yang tersedia secara lebih efektif dibandingkan dengan FIFO.

D. COSHH (A classification and optimization based scheduler for heterogeneous Hadoop systems)

COSHH adalah sebuah algoritma *scheduling* dalam Hadoop yang meminimalkan *mean/average completion time* dari *job* yang masuk dalam antrian[6]. *Scheduler* bawaan dari hadoop sendiri terlihat mengabaikan aspek heterogenitas *job* yang mungkin dieksekusi[6], dimana salah satu aspek yang penting dari *scheduling* adalah heterogenitas. Maka dari itu COSHH dibuat untuk memaksimalkan heterogenitas pada level cluster. Seperti yang dijabarkan sebelumnya COSHH dapat mengurangi *Average Completion Time* dibanding dengan *Scheduler* yang lain. Adapun arsitektur dari COSHH sendiri adalah sebagai berikut:

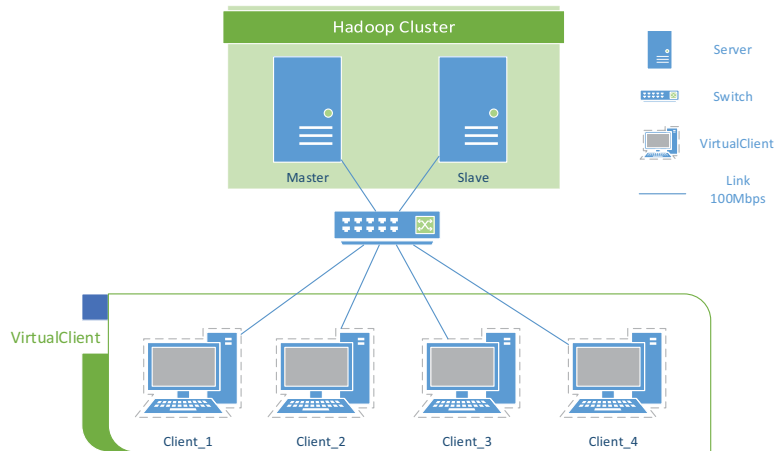


Gambar 6. Arsitektur COSHH[6]

Terdapat empat komponen utama penyusun COSHH yaitu; *Task Scheduling Process*, *The Queuing Process*, dan *Routing Process*. Hadoop *Scheduler* pada umumnya menerima dua *message* dari sistem Hadoop yaitu *message signal* dari user dan *heartbeat message* dari *resource* yang bebas. Ketika menerima suatu *job* baru maka *scheduler* akan memasukan *job* tersebut ke *queuing process* untuk menyimpan *job* yang datang menuju *queue* yang tepat[6]. Ketika menerima *heartbeat message*, *scheduler* akan memasukannya ke *routing process* untuk menempatkan *job* ke *resource* yang bebas. Pada gambar 6 alur *job* ditandai dengan garis sedangkan alur *heartbeat* ditandai dengan garis putus – putus.

III. PERANCANGAN SISTEM

Pada penelitian ini fokus pada performansi metode *scheduler* untuk system *cluster* server Hadoop. Penelitian dilakukan lingkungan terbatas dengan menggunakan arsitektur *MultiNode* Hadoop dengan topologi sebagai berikut :



Gambar 7 . Topologi Jaringan

Dalam lingkungan uji coba, terdiri atas dua buah server, dimana salah satunya berfungsi sebagai *Master node* dan yang lainnya berfungsi sebagai *Slave Node*. Konfigurasi hadoop yang digunakan adalah *multi-node*, dimana *Slave* hanya akan berfungsi sebagai *slave server* saja, sedangkan untuk *Master node* dapat berfungsi sebagai *Master server* dan juga dapat bertindak sebagai *slave server* dalam waktu yang bersamaan. Fungsi *client* adalah mengirimkan job yang selanjutnya akan dieksekusi disisi server. Jenis *job* yang dikerjakan dalam penelitian ini adalah seperti pada tabel 1 berikut.

Tabel 1. Karakteristik Job

Jenis Job	Karakteristik Job
<i>WordCount</i>	Menghitung jumlah kata yang unik dalam sebuah data text berukuran besar. Output dari <i>job</i> ini adalah daftar kata beserta jumlah kemunculan kata tersebut dalam sebuah file.
<i>Grep</i>	Mencari kata yang ditentukan oleh user, sehingga data hasil dari <i>grep</i> hanya kata yang dicari user saja.
<i>RandomTextWriter</i>	Akan membuat suatu file text yang berukuran tertentu yang sudah didefinisikan oleh <i>user</i> dan untuk kemudian disimpan pada Hadoop. <i>Job</i> ini digunakan saat kegiatan generate data secara random dalam ukuran besar untuk keperluan penelitian selanjutnya.

Skenario yang dijalankan terdiri atas 3 skenario *job* utama, yang kemudian di kombinasikan berdasarkan karakteristik *job* yang berbeda. Hal ini ditujukan untuk melihat pengaruh *scheduler* terhadap type *job* yang *homogen* maupun *heterogen*. Tabel 2 menunjukkan kombinasi job yang digunakan dalam penelitian ini.

Tabel 2. Skenario Job

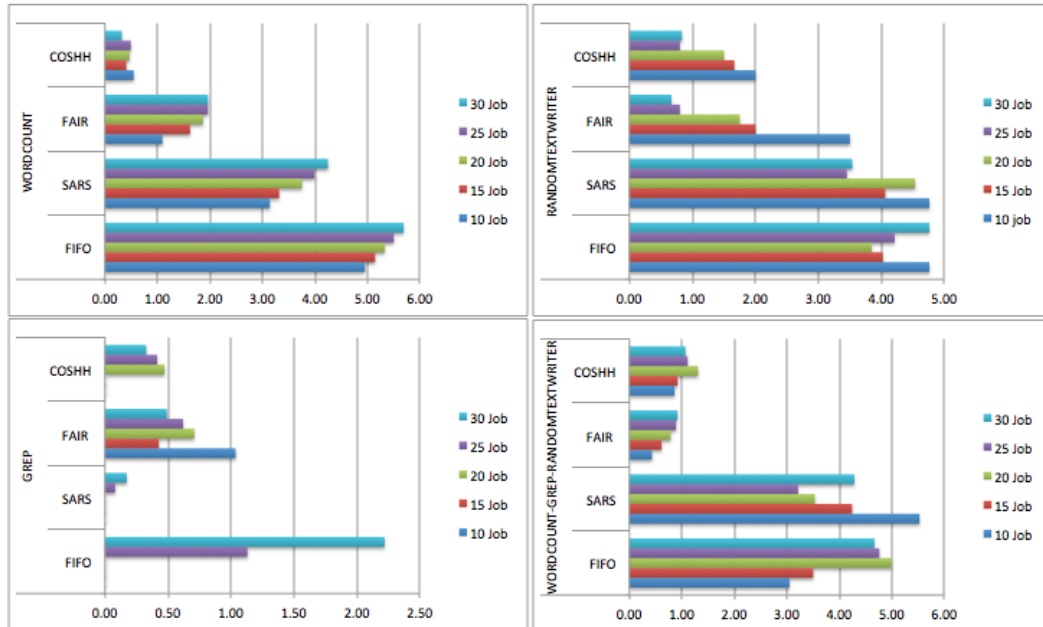
Skenario	Jenis Job	Job
1	Homogen	Wordcount
2		Grep
3		RandomTextWriter
4	Heterogen	WordCount-Grep-RandomTextWriter

Dari dataset yang disediakan, setiap skenario akan di jalankan dengan variasi jumlah *job* yaitu : 10,15,20,25 dan 30 *job* untuk setiap client nya.

IV. HASIL PENGUJIAN

Dari percobaan yang telah dilaksanakan, perbandingan performa 4 algoritma penjadwalan yang diimplementasikan diatas hadoop, akan dibandingkan dengan 3 parameter sebagai berikut :

- *Task Failure Rate*  
 Mengukur jumlah *job* yang gagal serta akan diulang kembali eksekusinya atau yang benar-benar gagal , kegagalan eksekusi *job* ini merupakan salah satu akibat dari kegagalan Hadoop dalam menyediakan resource yang diperlukan untuk eksekusi *job*. Satuan yang digunakan adalah job

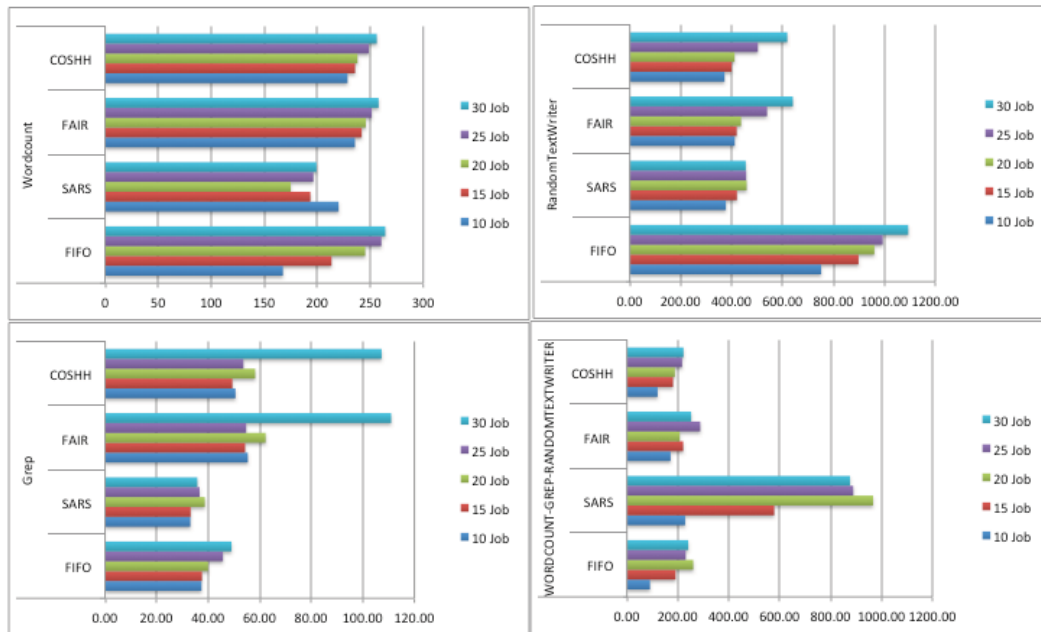


Gambar 8. Task Fail Rate

Secara keseluruhan scenario *job* yang dieksekusi, baik *homogen* maupun *heterogen*, terlihat bahwa Algoritma COSHH memiliki rata-rata nilai *Task Failure Rate* yang paling kecil, kecuali untuk jenis *job*

*grep* dimana algoritma SARS, dimana perbedaanya berkisar antara 0-0.5 *job*. Hal ini dipengaruhi oleh karakteristik COSHH, dimana dalam eksekusinya mengatur jumlah *pool* atau *queue* sebelum mengeksekusi *job*. Hal inilah yang menyebabkan prosentasi kemungkinan terjadinya *Failure* pada *job* yang akan dieksekusi menjadi berkurang.

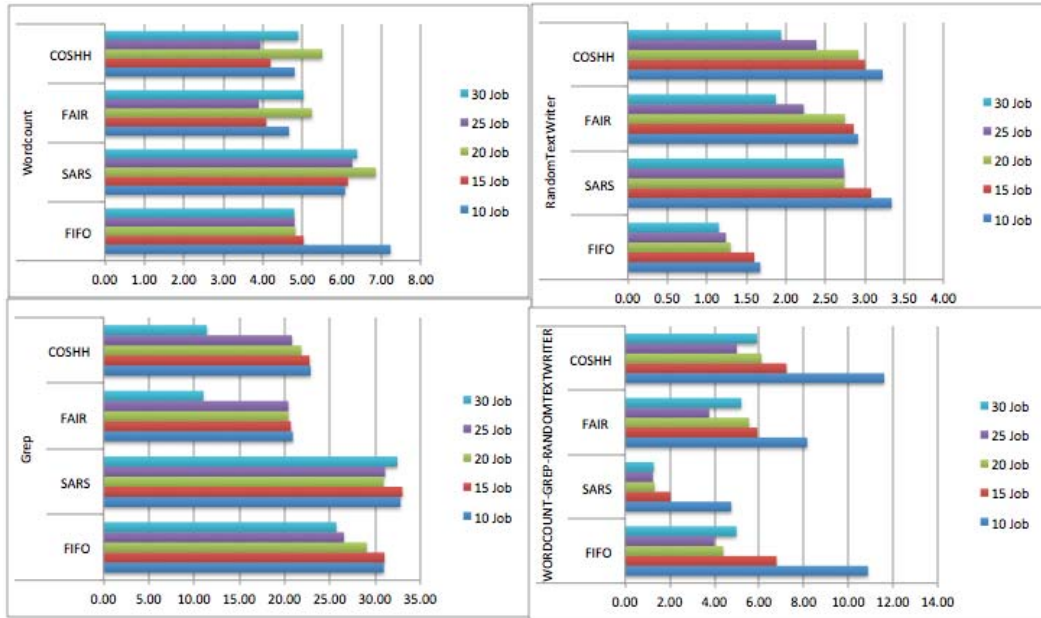
- Average Completion Time**  
 Parameter mengukur rata – rata dari waktu yang dibutuhkan dari suatu *job*. Nilai yang diperoleh merupakan total keseluruhan waktu yang dibutuhkan untuk menyelesaikan keseluruhan *job* dibagi dengan jumlah *job* yang dikerjakan [4]. Satuan yang digunakan adalah detik / *second*.



Gambar 9. Average Completion Time

Algoritma SARS memiliki nilai yang paling dijalankan pada skenario satu sampai skenario lima , atau untuk skenario *job homogen*. Namun hal berlawanan pada skenario dimana *job* yang dieksekusi adalah *job* yang bertipe *homogen*. Hal ini dipengaruhi karakteristik SARS yang mengatur jumlah *maps* serta *reduce* pada *tasktracer*, hal ini memerlukan komputasi yang lebih kompleks ketika dikerjakan pada skenario dimana karakteristik *job* yang dieksekusi bermacam-macam atau *heterogen*.

- Job Throughput**  
 Parameter ini mengukur jumlah *job* yang telah berhasil dieksekusi dalam suatu satuan waktu. Satuan yang dipakai adalah *job* per menit. Seperti ditunjukkan pada gambar 11, terlihat bahwa pada parameter ini diperoleh hasil bahwa SARS memiliki *job* Throughput yang paling tinggi untuk seluruh skenario yang sifat *job* yang homogen. Sedangkan untuk *job* yang sifatnya *Heterogen*, COSHH memiliki *throughput* yang paling tinggi diantara ke empat algoritma yang diuji cobakan.



Gambar 10. Job Throughput

## V. KESIMPULAN

Hasil pengujian menunjukkan bahwa algoritma SARS cocok digunakan pada penyelesaian *job* yang sifatnya homogen/sejenis. Hal ini ditunjukkan dengan tingginya *throughput* dan *average completion time* pada saat menyelesaikan *job* sejenis baik *word count*, *random text writer* maupun *grep*. Sementara itu algoritma COSHH cocok digunakan pada penyelesaian *job* gabungan yang *heterogen*. Hal ini terlihat dari tingginya *throughput* dan kecilnya *average completion time* pada saat menyelesaikan *job heterogen* gabungan *word count*, *random text writer* dan *grep*.

## REFERENCES

- [1] D. Borthakur, "HDFS Architecture Guide," 2008.
- [2] A. Rasooli and D. G. Down, "Guidelines for Selecting Hadoop Schedulers Based on System Heterogeneity," *J. Grid Comput.*, vol. 12, no. 3, pp. 499–519, Sep. 2014.
- [3] S. R. Pakize, "A Comprehensive View of Hadoop MapReduce Scheduling Algorithms," *Int. J. Comput. Networks Commun. Secur.*, vol. 2, no. 9, pp. 308–317, 2014.
- [4] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, and S. Shenker, "Job Scheduling for Multi-User MapReduce Clusters," 2009.
- [5] Z. Tang, L. Jiang, J. Zhou, K. Li, and K. Li, "A self-adaptive scheduling algorithm for reduce start time," *Futur. Gener. Comput. Syst.*, vol. 43–44, pp. 51–60, 2015.
- [6] A. Rasooli and D. G. Down, "COSHH: A classification and optimization based scheduler for heterogeneous Hadoop systems," *Futur. Gener. Comput. Syst.*, vol. 36, no. 36, pp. 1–15, Jul. 2014.