

Pengembangan Input Generator untuk Simulasi Penjadwalan *MPI Job* dengan metode *First Come First Serve-Ejecting Based Dynamic Scheduling (FCFS-EDS)* dalam Sistem Grid

Rusydi Umar

Teknik Informatika S2, Universitas Ahmad Dahlan
Jl Prof Supomo, Janturan, Yogyakarta
e-mail: rusydi_umar@rocketmail.com

Abstrak

Penjadwalan *MPI job* dalam sistem grid berdasarkan reservasi yaitu *First Come First Serve – Ejecting Based Dynamic Scheduling (FCFS-EDS)* masuk ke dalam kategori *flexible advance reservation*, bersifat dinamis dalam sudut pandang logika. Karena sifat penjadwalannya yang dinamis dan sudut pandangnya adalah logik maka sulit dipahami bagaimana proses penjadwalannya. Simulasi Penjadwalan *MPI job* berdasarkan reservasi dengan metode *FCFS-EDS* telah dibuat, dan input dari simulasi tersebut adalah file teks. File teks tersebut berisi spesifikasi input dari program simulasi dan rangkaian spesifikasi dari *job* yang datang untuk dijadwalkan. Selama ini file input dibuat secara manual, oleh karena itu sangat rentan terjadi kesalahan. Di makalah ini akan menjelaskan bagaimana caranya membuat program aplikasi input generator yang sudah ditentukan spesifikasi inputnya diawal, sehingga file teks tersebut dapat dibuat dengan dengan berbagai macam variasi spesifikasi input dan akan terhindar dari kesalahan. Dari hasil pengujian, perangkat lunak ini layak untuk diimplementasikan.

Kata kunci: reservasi, penjadwalan, *FCFS-EDS*, input generator

Abstract

Scheduling *MPI job* in a grid system, based on reservation, *First Come First Serve - Ejecting Based Dynamic Scheduling (FCFS-EDS)* comes into a *flexible advance reservation*, it is dynamic in a logical view. It will be difficult to learn and imagine how dynamic it is because it is in a logical view. The program simulation of this scheduling system has been made, and the input of this program is a text file consists of the specification of the input and the series of *MPI job* specification that comes for scheduling. So far, this text file has been made manually, so it is slow and error-prone. In this paper, we will explain how to make an input generator application with a certain specification of a series of *MPI job*. With this application, we can generate input quickly without any error on it. From the testing result, this application is worth to be implemented.

Keywords: maximum of 5 keywords from paper

1. Pendahuluan

Aplikasi paralel membutuhkan sumber daya yang sangat besar, sumber daya ini berasal dari banyak komputer paralel, menjalankan aplikasi paralel dalam waktu bersamaan. *Message Passing Interface (MPI)* adalah spesifikasi API yang memungkinkan aplikasi paralel untuk saling berkomunikasi dengan mengirimkan pesan. Biasanya, *MPI* digunakan oleh program paralel yang dijalankan dalam sebuah *cluster*, *super computer* atau dalam sistem *grid*. Aplikasi paralel ini bisa disebut sebagai aplikasi *MPI* atau (*MPI job*) pekerjaan *MPI*. Jika pekerjaan *MPI* membutuhkan *n* node daya komputasi untuk dieksekusi, maka *n* pekerjaan ini harus dieksekusi dalam waktu yang bersamaan. Aplikasi lain yang membutuhkan sumber daya dari banyak tempat adalah aplikasi alur kerja (*workflow application*), di mana pekerjaan dalam aplikasi alur kerja harus menunggu

pekerjaan lain sebelum dapat dieksekusi, sehingga dalam aplikasi alur kerja itu terjadi dependensi. *Workflow* adalah aplikasi paralel yang digabungkan yang terdiri dari serangkaian tugas komputasi yang dihubungkan oleh data dan dependensi [1].

Dalam aplikasi multimedia dan waktu nyata (*real time application*), untuk menjamin penyiaran video dan audio dalam jaringan dengan sempurna, seperti konferensi video, diperlukan sejumlah *bandwidth* tertentu. Dengan contoh ini, kita perlu reservasi untuk penjadwalan, untuk menjamin bahwa sumber daya tertentu akan siap digunakan untuk aplikasi yang diperlukan di masa mendatang [2].

Di sebagian besar sistem grid dengan penjadwal tradisional, semua pekerjaan yang diterima dimasukkan ke dalam antrian jika sumber daya yang diperlukan tidak tersedia. Setiap sistem grid menggunakan metode penjadwalan yang berbeda seperti *First Come First Serve* (FCFS), *Shortest Job First* (SJF), *Earliest Deadline First* (EDF), atau *EASY Backfilling* [3]. Semua metode penjadwalan di atas melaksanakan pekerjaan berdasarkan tiga parameter, yaitu jumlah sumber daya yang dibutuhkan, waktu mulai dan durasi waktu pekerjaan. Dengan semua metode penjadwalan di atas tidak ada jaminan kapan pekerjaan ini akan dieksekusi [4]. *Advance reservation* adalah proses meminta sumber daya untuk digunakan pada waktu tertentu di masa yang akan datang [2]. Sumber daya umum yang dapat dipesan atau diminta adalah elemen penyimpanan, *bandwidth* jaringan, *node* komputasi atau kombinasi dari semua itu.

FCFS-EDS masuk dalam *advance reservation* yang fleksibel, dinamis dalam pandangan logik [5]. Karena berada pada sudut pandang logik, maka agar mudah dipahami, telah dibuat perangkat lunak simulasi untuk mensimulasikan metode penjadwalan FCFS-EDS untuk Pekerjaan MPI dalam Sistem Grid [6,7]. Program simulasi ini membutuhkan input berupa rangkaian kedatangan MPI job yang akan dijadwalkan dari waktu ke waktu. File input ini masih dibuat secara manual. Oleh karena itu dibutuhkan program input generator yang dapat diumpungkan ke program simulasi tersebut, agar dapat dibuat dengan cepat dengan berbagai macam spesifikasi input. File input yang dihasilkan adalah serangkaian kedatangan MPI job dengan spesifikasi yang sudah ditentukan sebelumnya.

2. Metodologi

Platform komputasi Grid [8,9] memungkinkan pembagian, pemilihan, dan kombinasi sumber daya heterogen yang didistribusikan secara geografis (sumber data dan komputer) milik organisasi manajerial yang berbeda (juga disebut Organisasi Virtual (VO)) untuk menjawab masalah skala besar dalam bidang teknik, perdagangan, dan ilmu.

Tujuan utama untuk membuat dan menggunakan komputer paralel adalah bahwa paralelisme adalah salah satu cara terbaik untuk mengatasi penyumbatan kecepatan prosesor tunggal [10]. Ada tiga pendekatan untuk membuat aplikasi paralel. Pendekatan pertama didasarkan pada paralelisasi otomatis. Dengan pendekatan ini, seorang programmer tidak perlu khawatir tentang bagaimana caranya memparalelkan *task*/tugas. Yang kedua didasarkan pada penggunaan *library* paralel. Dengan pendekatan ini kode paralel yang sama untuk beberapa aplikasi diletakkan di *library* paralel. Pendekatan ketiga adalah pengodean ulang atau menulis kode dari awal dalam membuat aplikasi paralel. Programmer bebas memilih bahasa dan model pemrograman yang digunakan untuk membuat aplikasi paralel [11].

Message Passing Interface (MPI) adalah spesifikasi API yang memungkinkan aplikasi paralel untuk saling berkomunikasi dengan mengirimkan pesan. Biasanya, MPI digunakan oleh program paralel yang dijalankan dalam sebuah *cluster*, *super computer* atau dalam sistem grid. Aplikasi paralel ini bisa disebut sebagai aplikasi MPI atau pekerjaan MPI. Jika pekerjaan MPI membutuhkan n node daya komputasi untuk dieksekusi, maka n pekerjaan ini harus dieksekusi dalam waktu yang bersamaan.

FCFS-EDS masuk dalam kategori *flexible advance reservation*, dinamis dalam pandangan logik [5]. Strategi ini memanfaatkan pergeseran reservasi sebelumnya yang dibuat (tunduk pada batasan fleksibilitas yang diberikan) untuk memberikan ruang bagi permintaan reservasi yang baru masuk. Namun perencanaan untuk reservasi dalam model tersebut memberikan pandangan logik yang bertentangan dengan pandangan fisik yang dilaporkan dalam literatur. Oleh karena itu, disebut metode itu sebagai strategi FCFS-EDS.

2.1. Analisis dan Desain

Untuk percobaan, kita memerlukan beban kerja (*workload*) atau permintaan pengguna (*user requests*) (sebagian dipengaruhi oleh [12]) memiliki karakteristik berikut:

- 1) Laju permintaan reservasi yang masuk diasumsikan mengikuti distribusi Poisson.
- 2) Waktu eksekusi (t_e) untuk permintaan reservasi adalah antara 5 hingga 15 *timeslots* didistribusikan secara seragam.
- 3) Waktu mulai paling awal (t_{es}) untuk permintaan reservasi adalah antara 0 hingga 24 *timeslots*, didistribusikan secara seragam.
- 4) Persentase permintaan pengguna yang untuk *flexible advance reservation* (dipilih secara acak).
- 5) *Relax time* (t_r) untuk permintaan reservasi adalah antara 1 hingga 12 *timeslots* didistribusikan secara seragam dan $t_{ls} = t_{es} + t_r$
- 6) Jumlah node komputasi yang dibutuhkan (*numCN*) adalah antara 1 hingga 5 node komputasi dan didistribusikan secara seragam.
- 7) Dalam percobaan diasumsikan bahwa slot waktu sama dengan 5 menit (waktu jam).

Untuk memenuhi persyaratan di atas kita harus mengembangkan Algoritma untuk menghasilkan urutan pekerjaan yang masuk yang mengikuti persyaratan di atas. Algoritma dapat dilihat sebagai berikut:

- A. Kita harus menginisialisasi / mengatur parameter input program simulasi penjadwalan pekerjaan MPI di FCFS-EDS:
 - 1) *minDur* adalah nilai minimum waktu eksekusi (t_e), diinisialisasi dengan 5 *timeslots*
 - 2) *maxDur* adalah nilai maksimum waktu eksekusi (t_e), diinisialisasi dengan 15 *timeslots*
 - 3) *minBookAhead* adalah nilai minimum jarak antara waktu mulai dan waktu reservasi yang masuk, diinisialisasi dengan 0 *timeslots*
 - 4) *maxBookAhead* adalah nilai maksimum jarak antara waktu mulai dan waktu reservasi yang masuk, diinisialisasi dengan 0 *timeslots*
 - 5) *meanP* adalah rata-rata jumlah permintaan reservasi masuk di setiap *timeslots*, diinisialisasi dengan 3.0 permintaan reservasi masuk
 - 6) *persenFlex* adalah persentase *flexible advance reservation*, diinisialisasi dengan 25%
 - 7) *lasOcc* adalah *timeslots* terakhir yang terjadi pembalikan, diinisialisasi dengan 300
 - 8) *minFlex* adalah nilai minimum *relax time*, yang diinisialisasi dengan 1
 - 9) *maxFlex* adalah nilai minimum *relax time*, yang diinisialisasi dengan 1
 - 10) *minCN* adalah jumlah node komputasi yang dibutuhkan, diinisialisasi dengan 1
 - 11) *maxCN*, adalah jumlah node komputasi yang dibutuhkan, diinisialisasi dengan 5
 - 12) *userID*, adalah identitas pengguna, diinisialisasi dengan 1
 - 13) *file* adalah nama file output, diinisialisasi dengan "MPIInput3-25-1.txt"
- B. Setiap pekerjaan yang masuk memiliki karakteristik seperti jumlah *timeslot*, *userID*, *eStartTime*, *lastStartTime*, *execTime*, dan *numCN* yang dibutuhkan. Dengan karakteristik itu kita membuat matriks (*input matrix*) dengan baris adalah pekerjaan dan kolom adalah karakteristik pekerjaan. Ukuran matriks adalah $(meanP + 2) * lasOcc$ baris dan 7 kolom. Setiap baris berisi satu pekerjaan kolom-kolomnya adalah sebagai berikut: Kolom 0 berisi nomor *timeslot*
 - Kolom 1 berisi ID pengguna
 - Kolom 2 berisi *eStartTime*
 - Kolom 3 berisi *LastStartTime*
 - Kolom 4 berisi *exectime*
 - Kolom 5 adalah untuk pengocokan
 - Kolom 6 berisi *numCN*
- C. Bangkitkan banyaknya pekerjaan di setiap *timeslot* dengan distribusi *poisson* dan letakkan di vektor kedatangan
- D. Hitung jumlah pekerjaan yang dibuat di semua *timeslot* dan simpan ke *numOfJobs*

- E. Dalam matriks input, tetapkan nilai *timeslot number* dan *userID* ke dalam kolom 0 dan 1 untuk setiap pekerjaan dalam vektor kedatangan
- F. Sejauh ini kita memiliki *timeslot* yang ditentukan dan *jobId* akan menjadi indeks, kemudian menetapkan:
 - a. *eStartTime* pada kolom 2 didistribusikan secara seragam
 - b. *lastStartTime* di kolom 3 diisi dengan *eStartTime*, artinya semua tidak fleksibel
 - c. *execTime* pada kolom 4 didistribusikan secara seragam
 - d. *numCN* pada kolom 6 didistribusikan secara seragam
- G. Hitung jumlah *flexible advance reservation* (*numFlex*) dengan *numofJob* * *persenFlex*
- H. Inisialisasi kolom 5 dalam matriks input dengan *jobId*, kemudian kocok kolom 5 saja, sehingga data lain di kolom lain utuh
- I. Di antara *numofJob*, pilih *numFlex* pekerjaan untuk menjadi *flexible advance reservation*, yaitu *jobId* dengan data di kolom 5 berisi angka dari 0 hingga *numflex*, yang disebut *flexible advance reservation*
- J. Untuk semua *flexible advance reservation*, ubah *lasStarttime* dengan menambahkan dengan nilai antara *minFlex* dan *numFlex* yang didistribusikan secara seragam
- K. Tulis semua karakteristik ke dalam *file* dan diikuti oleh input matriks sehingga dapat dimasukkan ke program simulasi.

3. Results and Analysis

Telah dikembangkan perangkat lunak untuk generator input untuk perangkat lunak simulasi untuk mensimulasikan metode penjadwalan FCFS-EDS untuk Pekerjaan MPI dalam Sistem *Grid*. Perangkat lunak ini kemudian disebut sebagai *Input Generator* untuk Simulasi Penjadwalan FCFS-EDS untuk Pekerjaan MPI. Output dari program ini menjadi input dari perangkat lunak simulasi untuk mensimulasikan metode penjadwalan Dinamis Penjadwalan Berbasis Awal (FCFS-EDS) untuk Pekerjaan MPI dalam Sistem Grid [7].

Gambar 1. adalah *file* input untuk program simulasi. Kita dapat melihat dari Gambar 1 bahwa kita menulis semua karakteristik input, dan kemudian diikuti oleh jumlah timeslot. Di bawah timeslot ada 6 angka di setiap baris yang mewakili sebagai *userId*, *jobId*, *eStartTime*, *lastStartTime*, *Duration* (*execTime*), and jumlah *node* komputasi yang dibutuhkan. Jika ada x baris berturut-turut dengan 6 angka tersebut berarti bahwa timeslot yang sesuai memiliki x pekerjaan yang masuk.

```

File Edit Format View Help
Minimum Duration          5
Maximum Duration         15
Minimum Book Ahead        0
Maximum Book Ahead       24
Average of arrival        3.0
Percentage of Flexible AR 25
Last Occurrence           300
Minimum Flexibility        1
Maximum Flexibility       12
Minimum CN needed         1
Maximum CN needed         5
User ID                   1
Below is the data of incoming reservation
0
1 0 7 7 11 4
1 1 7 7 14 2
1
1 2 16 16 12 2
1 3 3 3 8 2
2
1 4 17 17 8 2
1 5 26 26 15 1
3

```

Gambar 1. *File* output dari program input generator

4. Kesimpulan

Setelah melakukan pengujian *black box*, kami menemukan bahwa perangkat lunak berjalan dengan sempurna. Kesimpulan yang dapat diambil dari penelitian ini: Telah dikembangkan sebuah program untuk menghasilkan input atau pekerjaan yang masuk untuk program simulasi FCFS-EDS dengan karakteristik tertentu. Kita dapat mengubah karakteristik dengan mengubah inisialisasi parameter dalam program. Dengan cara itu kita dapat membuat beberapa *file* input dengan parameter berbeda, sehingga dapat dengan mudah digunakan untuk mempelajari perilaku program simulasi FCFS-EDS.

Daftar Pustaka

- [1] Juve, Gideon, et al. "Scientific workflow applications on Amazon EC2." *2009 5th IEEE international conference on e-science workshops*. IEEE, 2009.
- [2] Smith, Warren, Ian Foster, and Valerie Taylor. "Scheduling with advanced reservations." *Proceedings 14th International Parallel and Distributed Processing Symposium. IPDPS 2000*. IEEE, 2000.
- [3] Mu'alem, Ahuva W., and Dror G. Feitelson. "Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling." *IEEE transactions on parallel and distributed systems* 12.6 (2001): 529-543.
- [4] Sulistio, Anthony, and Rajkumar Buyya. "A grid simulation infrastructure supporting advance reservation." *16th International Conference on Parallel and Distributed Computing and Systems (PDCS 2004)*. Vol. 11. 2004.
- [5] Umar, Rusydi, Arun Agarwal, and C. Raghavendra Rao. "Advance planning and reservation in a grid system." *International Conference on Networked Digital Technologies*. Springer, Berlin, Heidelberg, 2012.
- [6] Umar, Rusydi, and Ardi Pujiyanta. "Media Pembelajaran untuk Metode Penjadwalan First Come First Serve-Ejecting Based Dynamic Scheduling (FCFS-EDS) untuk MPI Job dalam Sistem Grid." *Annual Research Seminar (ARS)*. Vol. 3. No. 1. 2017.
- [7] Umar, Rusydi, and Ardi Pujiyanta. "Development of First Come First Serve-Ejecting Based Dynamic Scheduling (FCFS-EDS) Simulation Scheduling Method for MPI Job in a Grid System." *Journal of Engineering and Applied Sciences* 12.8 (2017): 1972-1978.
- [8] Daniel, Minoli. "A Networking Approach to Grid Computing, Hoboken." (2005).
- [9] Foster, Ian, and Carl Kesselman, eds. *The Grid 2: Blueprint for a new computing infrastructure*. Elsevier, 2003.
- [10] Hwang, Kai, and Zhiwei Xu. *Scalable parallel computing: technology, architecture, programming*. McGraw-Hill, Inc., 1998.
- [11] Tiemeyer, Matthew P., and Johnny SK Wong. "A task migration algorithm for heterogeneous distributed computing systems." *Journal of Systems and Software* 41.3 (1998): 175-188.
- [12] Sulistio, Anthony, Kyong Hoon Kim, and Rajkumar Buyya. "Using revenue management to determine pricing of reservations." *Third IEEE International Conference on e-Science and Grid Computing (e-Science 2007)*. IEEE, 2007.