OPEN ACCESS
Indo-JC
Indonesian Journal of Computing
ISSN 2460-9056
socj.telkomuniversity.ac.id/indojc

Ind. Journal on Computing
Vol. 2, Issue. 1, Maret 2017. pp. 7-20
doi:10.21108/indojc.2017.21.19

Pengembangan dan Pengujian Sistem Grid Computing Menggunakan Globus Toolkit di Universitas Telkom

Izzatul Ummah[#1], Andi Farid Arief Nur[*2]

 # Department of Computational Science, Telkom University
Jl. Telekomunikasi Terusan Buah Batu Bandung 40257 Indonesia

[1] izzatulummah@telkomuniversity.ac.id

** Department of Computational Science, Telkom University*
Jl. Telekomunikasi Terusan Buah Batu Bandung 40257 Indonesia

[2] andifarid@grid.telkomuniversity.ac.id

Abstract

Computational problems nowadays has become more complex and therefore needs more computing resources. In this research, we build a grid computing infrastructure in Telkom University to help solving computational problems. We used Globus Toolkit 6.0 and Condor 8.4.2 as middlewares in developing the grid system. We evaluated the performance of our grid system using parallel matrix multiplication. As a result, we analyze the performance of matrix multiplication process executed in the grid computing system, in which the execution speedup is improved accordingly with the number of processors being added.

Keywords: grid computing, Globus Toolkit, Condor, matrix multiplication

Abstrak

Semakin kompleksnya permasalahan komputasi yang ada saat ini membutuhkan sumberdaya komputasi yang semakin besar. Pada penelitian ini penulis membangun sebuah infrastruktur *grid computing* di Universitas Telkom untuk digunakan dalam menyelesaikan permasalahan komputasi. Middleware yang digunakan untuk membangun grid yaitu Globus Tookit 6.0 dan Condor 8.4.2. Sistem grid yang telah dibangun kemudian diujicoba menggunakan problem komputasi sederhana yaitu perkalian matriks secara paralel. Hasil penelitian ini berupa analisis performansi perkalian matriks yang dijalankan pada sistem grid computing, dengan kecepatan eksekusi meningkat sesuai penambahan jumlah prosesor yang digunakan.

Kata Kunci: grid computing, Globus Toolkit, Condor, perkalian matriks

I. PENDAHULUAN

Kebutuhan akan komputasi kinerja tinggi saat ini telah menjadi semakin meningkat seiring dengan semakin banyaknya problem dengan ukuran data yang semakin besar. Akan tetapi tidak semua pihak dapat membangun lingkungan komputasi kinerja tinggi tersebut, dikarenakan keterbatasan sumber daya. Untuk membangun dan mengelola cluster komputasi kinerja tinggi di *local site*, dibutuhkan keahlian teknis yang cukup spesifik. Salah satu solusi untuk permasalahan ini adalah dengan penggunaan *grid computing*.

Received on Feb 2017. Accepted on Mar 2017

Dengan *grid computing*, peneliti dapat mengakses cluster yang letaknya terpisah secara geografis dan menjalankan program simulasinya di cluster tersebut.

Adapun permasalahan utama yang ingin dijawab dalam penelitian ini yaitu bagaimana menganalisis dan mengevaluasi kinerja sistem *grid computing* tersebut dalam menjalankan dan menyelesaikan problem komputasi. Terdapat dua metode yang sering digunakan dalam penelitian-penelitian sebelumnya, yaitu metode *strong scaling* dan *weak scaling*. Pada *strong scaling*, ukuran problem/data dibuat tetap, dan jumlah prosesor diperbanyak. Pada *weak scaling*, ukuran problem/data yang diassign ke tiap processor dibuat tetap, sehingga penambahan jumlah processor dibuat sebanding dengan penambahan ukuran problem. Dalam penelitian ini penulis menggunakan metode *strong scaling*.

II. DASAR TEORI

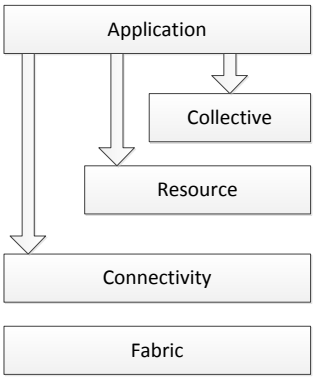
Pada bagian ini akan diuraikan terlebih dahulu mengenai grid computing, Globus Toolkit, serta Condor.

A. *Grid Computing*

Berdasarkan definisi dari IBM Redpaper, *grid* yaitu kumpulan mesin atau node yang berkontribusi membentuk sebuah sistem secara keseluruhan. Beberapa sumberdaya dapat digunakan oleh semua pengguna grid, sementara beberapa sumberdaya lainnya mungkin akan dibatasi hanya untuk grup pengguna tertentu saja [1]. *Grid computing* dapat dipandang sebagai sistem komputasi terdistribusi yang dibawa ke tingkatan yang lebih lanjut. Tujuannya adalah menciptakan sebuah ilusi sistem komputasi virtual yang sangat besar, *powerful*, dan *self-managing*; yang pada kenyataannya sistem itu tersusun dari sekumpulan sistem heterogen yang terhubung dan masing-masingnya mengkontribusikan berbagai macam kombinasi sumberdaya. Sumberdaya yang dimaksud meliputi perangkat keras (*core processor*, RAM, harddisk, dan sebagainya) dan perangkat lunak (*compiler, middleware, libraries*, dan sebagainya).

Grid computing adalah teknologi untuk mengkoordinasikan *resource sharing* dalam skala besar yang melibatkan berbagai *autonomous group* (atau sistem terdistribusi yang terpisah secara geografis), untuk menyelesaikan suatu problem komputasi dalam skala besar di dalam sebuah lingkungan *virtual organization* (VO) yang bersifat dinamis dan multi-institusional [2] [3]. *Resource sharing* ini dikendalikan berdasarkan *rules* (aturan) yang telah disepakati antara penyedia sumberdaya dan pengguna sumberdaya. Kumpulan individu dan/atau institusi yang diatur oleh *rules* bersama itu disebut *virtual organization* (VO) [3].

Sistem *grid computing* menyediakan protokol dan layanan yang terbagi dalam lima lapisan [2], seperti dapat dilihat di Gambar 1.



Gambar 1: Grid Protocol Architecture

Pada lapisan terbawah, yaitu layer *fabric*, grid menyediakan akses ke berbagai jenis sumberdaya misalnya prosesor, *storage*, *network*, repositori kode program, dan sebagainya. Contoh protokol yang bekerja pada layer *fabric* yaitu GARA dan Falkom. Layer *connectivity* bertugas meyediakan protokol jaringan dan sarana komunikasi yang digunakan dalam menghubungkan sistem. Contoh protokol yang bekerja di layer *connectivity* yaitu GSI (Grid Security Infrastructure). Layer *resource* menyediakan protokol untuk *publication*, *discovery*, *negotiation, accounting*, serta *payment* (pembayaran) dari pemakaian sumberdaya tersebut. Contoh protokol yang bekerja di layer *resource* yaitu GRAM (Grid Resource Allocation Management) dan GridFTP. Layer *collective* sama dengan layer *resource*, yang membedakan yaitu layer *collective* mengatur sumber daya yang sifatnya berkelompok. Contoh protokol yang bekerja pada layer *collective* yaitu MDS (Monitoring and Discovery Service), Condor-G, Nimrod-G, MPICH, dan CAS (Community Authorization Service). Layer *application* terdiri dari seluruh aplikasi pengguna serta API (*application programming interface*) yang dibangun di atas sistem grid tersebut, dan dijalankan di lingkungan *virtual organization* (VO) [3].

B. Globus Toolkit

Globus Toolkit adalah sebuah toolkit perangkat lunak yang dikembangkan oleh The Globus Alliance yang dapat digunakan untuk membangun sistem *grid computing*. Versi terbaru Globus Toolkit saat ini adalah versi 6.0 (GT6). Globus Toolkit bersifat *service oriented architecture*, dan memanfaatkan banyak komponen Web Services yang meliputi XML (Extensible Markup Language), SOAP (Simple Object Access Protocol), dan WSDL (Web Services Definition Language) [4].

Komponen utama Globus Toolkit terdiri dari empat bagian yaitu [4] [5]:

1) **Execution Management**: Komponen ini bertanggung jawab dalam mengatur bagaimana eksekusi dilakukan di atas sumberdaya-sumberdaya komputasi yang tersedia di dalam sistem *grid computing*. Pengelolaan ini mencakup eksekusi program, mulai dari inisiasi, monitoring, sampai penjadwalan dan koordinasi antar-proses.

2) **Data Management**: Komponen ini bertanggungjawab menyediakan mekanisme akses bagi pengguna untuk mengakses data berukuran besar dari sumberdaya komputasi di dalam sistem *grid computing* secara efisien dan *reliable*.

3) **Monitoring and Discovery**: Komponen ini bertanggungjawab menyediakan mekanisme bagi pengguna sistem agar dapat melakukan monitoring terhadap proses komputasi yang sedang dijalankan, sehingga apabila ada masalah yang timbul dapat segera diketahui dan dicari solusinya. Komponen ini juga bertanggung jawab menyediakan mekanisme agar pengguna dapat dengan mudah menemukan serta mengidentifikasi keberadaan suatu sumberdaya komputasi berikut karakteristiknya, secara otomatis dan real time. Apabila ditambahkan sumberdaya baru ke dalam sebuah sistem grid, sumberdaya itu harus dapat segera diidentifikasi dengan mudah oleh para pengguna sistem.

4) **Security**: GSI (Grid Security Infrastructure): Komponen ini bertanggung jawab atas keamanan sistem *grid computing* secara keseluruhan. Komponen ini pula yang merupakan salah satu ciri khas teknologi Globus Toolkit, yang membedakannya dengan teknologi-teknologi pendahulunya seperti PVM atau MPI. Dengan diterapkannya mekanisme keamanan yang terintegrasi dengan komponen-komponen *grid computing* lainnya, maka Globus Toolkit dapat diterapkan di lingkungan jaringan publik (Wide Area Network atau Internet) tanpa menurunkan tingkat keamanannya [5].

C. HTCondor

Condor adalah sistem atau *tools* untuk mengatur beban kerja di lingkungan cluster yang bersifat compute-intensive. Condor dikembangkan oleh Departemen Ilmu Komputer di Wisconsin-Madison University pada tahun 1988. Berkebalikan dengan model umum saat itu yaitu *dominant-centralized control model*, pada model

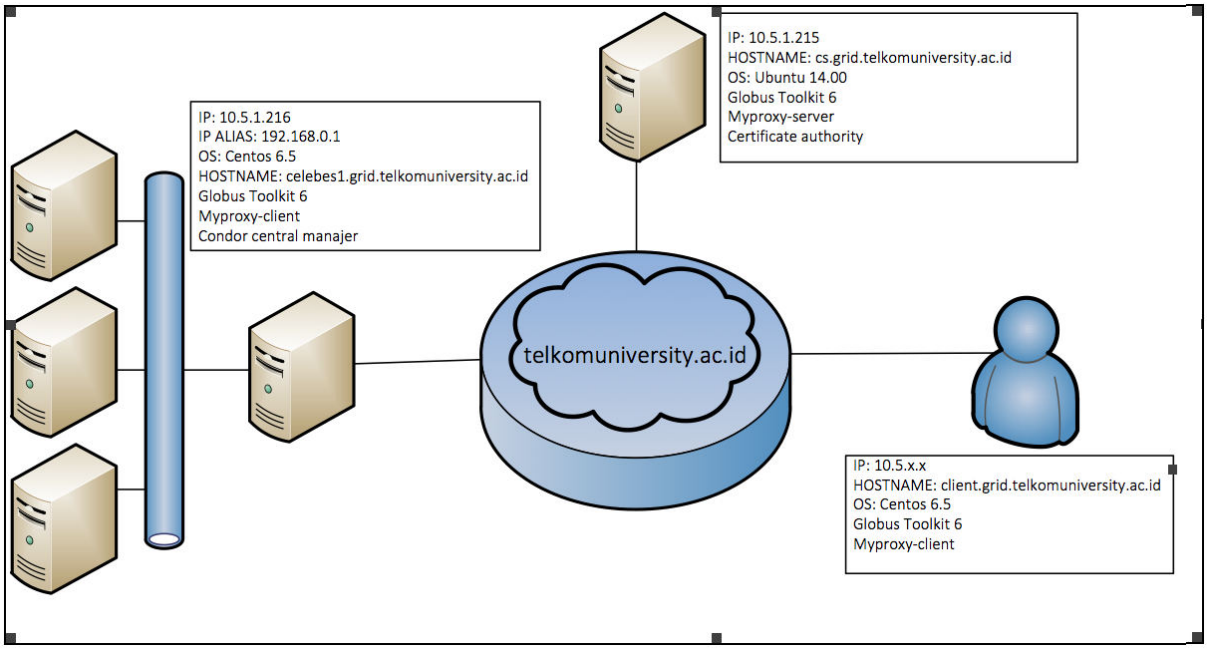
Condor para partisipan (penyedia sumberdaya) di dalam sebuah sistem grid dibebaskan untuk berkontribusi sebanyak atau sesedikit yang mereka kehendaki [6].

Condor berubah nama menjadi HTCondor pada tahun 2012. HTCondor adalah software yang digunakan untuk membangun sebuah lingkungan *high throughput computing* (HTC). Perbedaan *high throughput computing* (HTC) dengan *high performance computing* (HPC) adalah sebagai berikut. Lingkungan HPC bertujuan menyediakan sumberdaya komputasi yang sangat tinggi untuk digunakan dalam waktu yang singkat, dan ukuran yang digunakan untuk mengukur kinerja HPC adalah FLOPS (*floating point operations per second*). Akan tetapi, untuk komunitas pengguna yang lebih luas, seringkali yang menjadi perhatian bukanlah operasi per detik, melainkan operasi per bulan atau bahkan per tahun. Banyak program paralel yang ketika dijalankan di atas sistem paralel membutuhkan waktu berbulan-bulan. Oleh karena itu ukuran yang lebih cocok digunakan untuk mengukur sistem paralel seperti ini bukanlah berapa banyak operasi floating point yang bisa diselesaikan per detiknya (FLOPS), tapi berapa banyak pekerjaan skala besar yang bisa ditangani per bulannya atau bahkan per tahunnya. Sistem HPC yang lebih berorientasi jangka panjang ini disebut *high throughput computing* (HTC).

Pengguna mensubmit *job* ke HTCondor. Lalu HTCondor akan mencarikan mesin yang tersedia di sistem melalui jaringan, dan kemudian mendelegasikan *job* itu ke mesin tersebut. HTCondor mampu mendeteksi apabila sebuah mesin tidak lagi available.

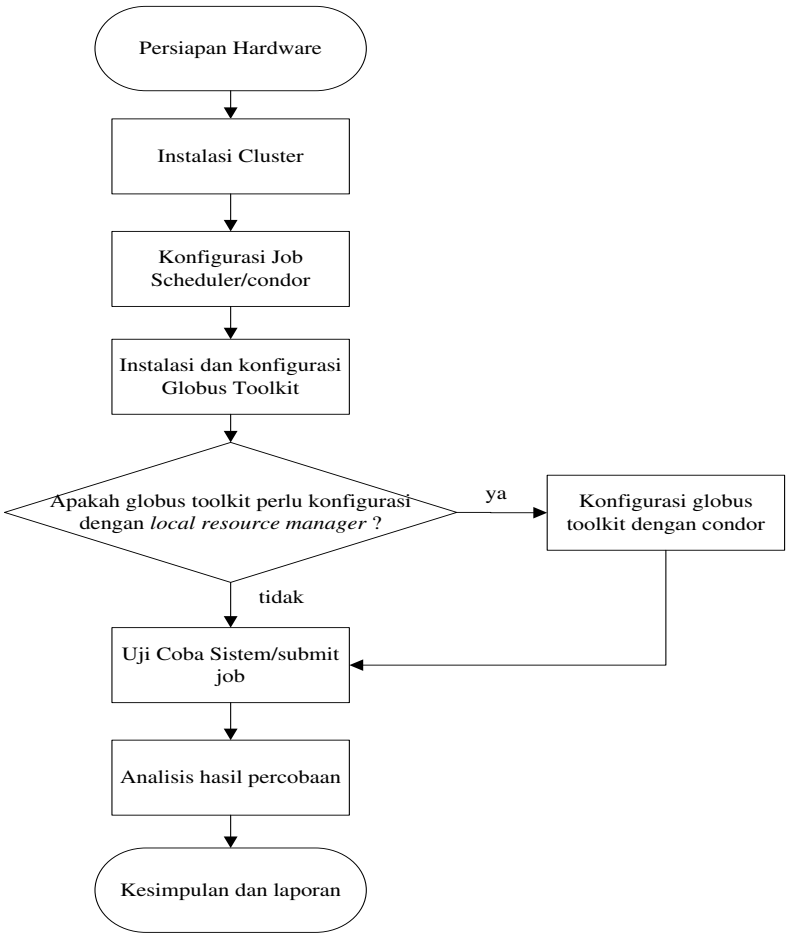
III. PERANCANGAN DAN IMPLEMENTASI SISTEM

Perancangan dan implementasi sistem *grid computing* dilakukan di lingkungan Lab High Performance Computing, Program Studi Ilmu Komputasi, Universitas Telkom. Pada penelitian ini digunakan satu buah cluster (Celebes Cluster) sebagai sumber daya *grid computing*. Celebes cluster merupakan *dedicated cluster* yang terdiri dari 3 buah node dan 1 headnode. Terdapat pula 1 buah server sebagai portal dan *certificate authority manager* dari sistem *grid computing* ini. Antar sistem tersebut dihubungkan menggunakan jaringan lokal kampus Universitas Telkom. Rancangan topologi sistem dapat dilihat di Gambar 2.



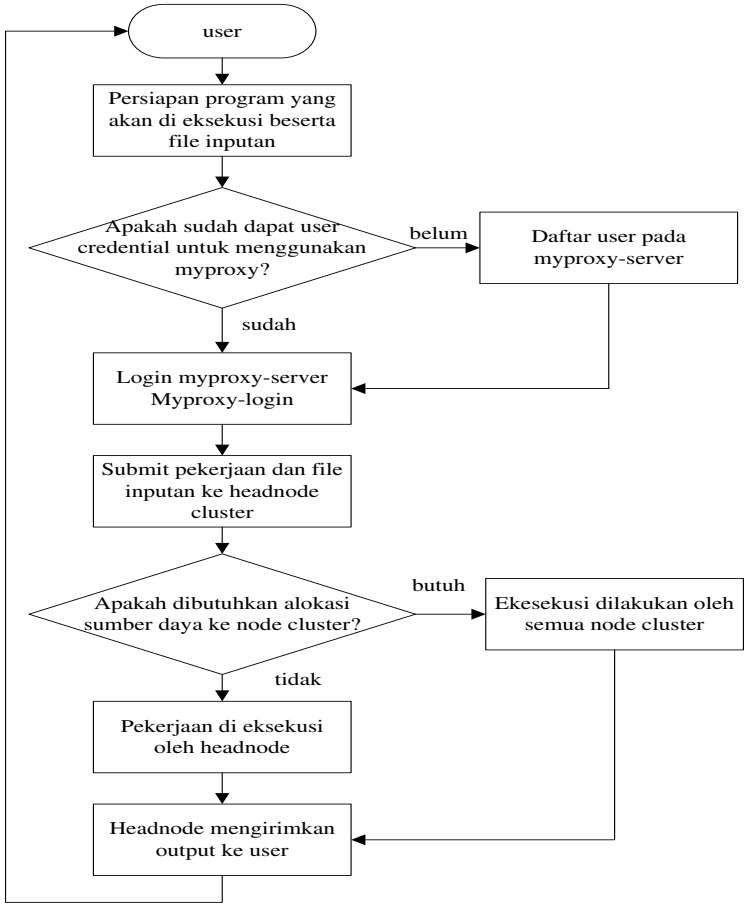
Gambar 2. Skema lingkungan grid computing

Alur perancangan sistem grid computing pada penelitian ini dapat dilihat pada Gambar 3. Langkah pertama yaitu instalasi cluster, kemudian lakukan konfigurasi job scheduler (Condor) pada cluster tersebut. Berikutnya yaitu install dan konfigurasi Globus Toolkit pada node yang berbeda. Konfigurasi tambahan perlu dilakukan untuk mensinkronkan Globus Toolkit dengan Condor. Setelah sistem selesai dibangun, dilakukan ujicoba dan analisis hasil percobaan.



Gambar 3. Diagram alur perancangan sistem

Diagram alur kerja sistem dapat dilihat pada Gambar 4 berikut ini. User harus mempersiapkan program yang akan dieksekusi di grid terlebih dahulu, kemudian merequest user credential dan login ke myproxy server. Selanjutnya user mensubmit job dan file data dan binary program yang akan dieksekusi ke grid. Job tersebut kemudian akan dialokasikan dan dieksekusi ke cluster (compute nodes). Apabila proses komputasi sudah selesai dijalankan, headnode grid akan memberikan kembali hasil eksekusi tersebut ke user.



Gambar 4. Diagram alur kerja sistem

IV. PENGUJIAN DAN EVALUASI KINERJA SISTEM

Bab ini menjelaskan pengujian dan evaluasi kinerja sistem. Grid computing yang sudah dibangun diuji dengan program perkalian matriks NxN.

A. Perkalian Matriks NxN

Sistem grid computing yang telah dibangun selanjutnya diujicoba dengan program perkalian matriks, yaitu perkalian dari dua matriks persegi antara matriks A dan matriks B yang menghasilkan matriks C. Ketiga matriks ini berukuran NxN elemen. Operasinya dapat dituliskan dengan persamaan sebagai berikut:

$$C_{ij} = \sum_{k} A_{ik} B_{kj}$$

Dari operasi sederhana di atas, dapat dibuat algoritma perkalian matriks sebagai berikut:

```
for (int i = 0; i < N; i++)
    for (int j = 0; j < N; j++)
       for (int k = 0; k < N; k++)
          C[i][j] = A[i][k] * B[k][j];
```

Kompleksitas waktu asimptotik untuk algoritma di atas adalah $T(n) \in O(n^3)$.

Pengujian dilakukan dengan mensubmit jobs secara bertahap, dengan memperbesar ukuran matriks yang digunakan, yaitu 500x500, 1000x1000, 1500x1500, 2000x2000, 2500x2500, 3000x3000, 3500x3500, 4000x4000, 4500x4500, dan 5000x5000. Dalam penelitian ini, tidak dilakukan optimalisasi lebih lanjut terhadap algoritma dan struktur data yang digunakan dalam proses perkalian matriks tersebut.

Adapun asumsi yang digunakan dalam proses pengujian sistem ini adalah:
1) Submit pekerjaan dilakukan dari komputer user yang telah terinstall globus toolkit. Dalam penelitian ini digunakan dari komputer dengan hostname: client1.grid.telkomuniversity.ac.id.
2) Host telah memiliki *credential* yang valid sebagai syarat menggunakan layanan globus toolkit.
3) User telah memiliki *certificate authority* untuk melakukan login ke layanan myproxy. Layanan myproxy-server yang dapat dipakai dalam sistem ini yaitu cs.grid.telkomuniversity.ac.id dengan memasukkan *passphrase* yang dipakai saat mendaftarkan *user credential*.

Ada 2 skenario pengujian yang dilakukan:
1) Pengujian menggunakan fork (jobtype: single)
2) Pengujian menggunakan condor (jobtype: multiple)

B. Pengujian Menggunakan Fork

Ketika proses *submit job*, pengguna mendefinisikan *local resource manager*: *fork* yang akan dipakai untuk mengeksekusi pekerjaan di sumberdaya yang dituju, dalam hal ini celebes1.grid.telkomuniversity.ac.id. Fork hanya mendukung *jobtype single* dan tidak mendukung *dedicated resource* sehingga pekerjaan hanya akan diproses pada headnode cluster dengan satu processor. Dari hasil percobaan tersebut didapatkan data waktu eksekusi pekerjaan yaitu sebagai berikut (lihat Tabel I):

TABLE I. Waktu eksekusi menggunakan fork

I	N_i	N_i / N_1	$(N_i / N_1)^3$	T_i (detik)	T_i / T_1
1	500	1	1	1.17	1.00
2	1000	2	8	10.02	8.56
3	1500	3	27	45.53	38.91
4	2000	4	64	120.59	103.07
5	2500	5	125	225.67	192.88
6	3000	6	216	361.34	308.84
7	3500	7	343	602.17	514.68
8	4000	8	512	893.66	763.81
9	4500	9	729	1277.24	1091.66
10	5000	10	1000	1728.10	1477.01

Dari data yang dihasilkan terlihat bahwa pertumbuhan Ti/T1 masih sebanding dengan $O(n^3)$. Hal ini karena eksekusi program menggunakan *fork* hanya dijalankan di atas 1 prosesor saja.

C. Pengujian Menggunakan Condor

Dalam percobaan ini dilakukan pendefinisian *local resource manager: condor* yang akan dipakai dalam proses submit job ke sumberdaya komputasi yang tersedia. Sebelumnya diperlukan konfigurasi *condor* dalam cluster sehingga semua core CPU dari semua node terdeteksi sebagai *dedicated resource*. Condor mendukung jobtype *multiple* sehingga dilakukan percobaan dengan menggunakan metode strong scaling, di mana jumlah core CPU akan diperbanyak untuk setiap kali percobaan yaitu 1, 10, 20, dan 30 cores. Jobtype multiple dalam hal ini menjalankan proses yang sama pada processor yang berbeda dalam cluster bukan dikerjakan secara parallel sehingga akan dilihat kemampuan condor dalam melakukan penjadwalan pekerjaan ke processor yang

tersedia ketika dilakukan penyerahan pekerjaan dari globus toolkit. Dari hasil percobaan ini didapatkan rata-rata waktu eksekusi program pada processor yang berbeda dalam sekali submit program dari globus toolkit. Data yang dihasilkan sebagai berikut:

TABLE II. Waktu eksekusi menggunakan condor (second)

NP \ N	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
1	0.76	8.01	43.75	104.2	204.97	355.6	580.03	869.79	1240.68	1682.6
10	2.054	19.732	71.086	176.334	334.925	592.196	939.36	1411.258	2045.674	2688.96
20	1.6875	18.321	65.845	160.8835	316.361	540.291	868.6945	1305.154	1862.749	2468.628
30	1.610667	18.24361	65.19567	160.205	313.9907	539.7187	860.6863	1293.065	1840.062	2454.886

Dari data yang dihasilkan dapat dilihat perbedaan waktu eksekusi ketika menggunakan mode jobtype single (hanya menggunakan satu core CPU) dengan mode jobtype *multiple* (menggunakan beberapa core CPU). Pada penggunaan 1 core saja, maka waktu eksekusi meningkat sebanding dengan $O(n^3)$. Sedangkan ketika program dijalankan dengan mode *jobtype multiple*, semakin banyak jumlah core CPU yang dipakai maka semakin cepat waktu eksekusi. Hal ini disebabkan oleh proses *matchmaking* yang dilakukan condor, ketika membutuhkan jumlah core CPU yang banyak maka condor membagi pekerjaan ke semua core CPU yang tersedia tanpa mempertimbangkan spesifikasi dari core-core CPU tersebut.

D. Optimalisasi Sistem

Untuk mendapatkan alokasi sumber daya yang maksimal diperlukan beberapa konfigurasi dalam sistem. Dalam penelitian ini, optimalisasi sistem dilakukan dengan memanfaatkan fitur dari condor. Condor memungkinkan penyedia mengatur prioritas pekerjaan yang akan dieksekusi berdasarkan *jobtype* dari pekerjaan tersebut. Konfigurasi condor dalam cluster dapat diatur dengan memilih salah satu dari condor_policy sebagai berikut:

1. Hanya menjalankan pekerjaan yang sifatnya dedicated.
2. Dapat menjalankan semua jenis pekerjaan tapi lebih memprioritaskan yang sifatnya dedicated.
3. Selalu menjalankan pekerjaan yang sifatnya dedicated, tapi bisa juga menjalankan pekerjaan yang non-dedicated ketika terdapat waktu jeda dan tidak ada pekerjaan dedicted dalam antrian.

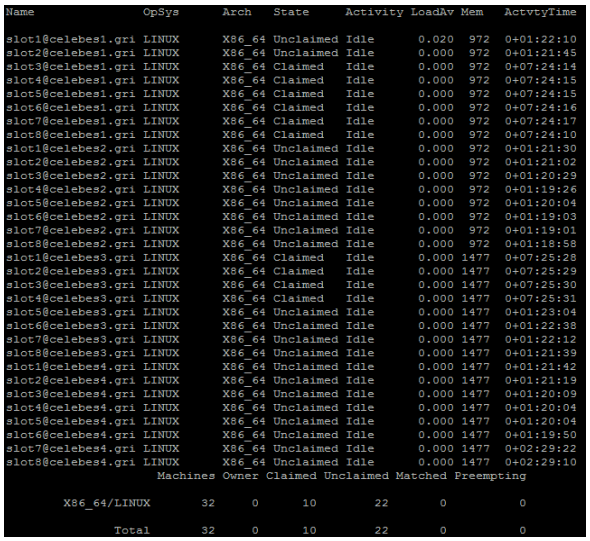
Dalam sistem ini diperlukan perubahan dalam berkas konfigurasi lokal pada condor dengan menentukan ekspresi sebagai berikut:

```
## 2) Always run jobs, but prefer dedicated ones
##--------------------------------------------------------------------
START           = True
SUSPEND         = False
CONTINUE        = True
PREEMPT         = False
KILL            = False
WANT_SUSPEND    = False
WANT_VACATE     = False
RANK            = Scheduler =?= $(DedicatedScheduler)
```

Ekspresi START bernilai TRUE, artinya pekerjaan akan otomatis dieksekusi ketika beban kerja CPU < 0.3 dalam kondisi idle atau sedang menjalankan pekerjaan dengan syarat beban CPU belum overload. Ekspresi CONTINUE bernilai TRUE, artinya ketika pekerjaan sedang ditunda, dapat otomatis dijalankan oleh sistem ketika syarat START sudah terpenuhi. Sedangkan ekspresi RANK menunjukkan bahwa sistem selalu menjalankan pekerjaan yang dedicated.

E. Antrian pada Condor

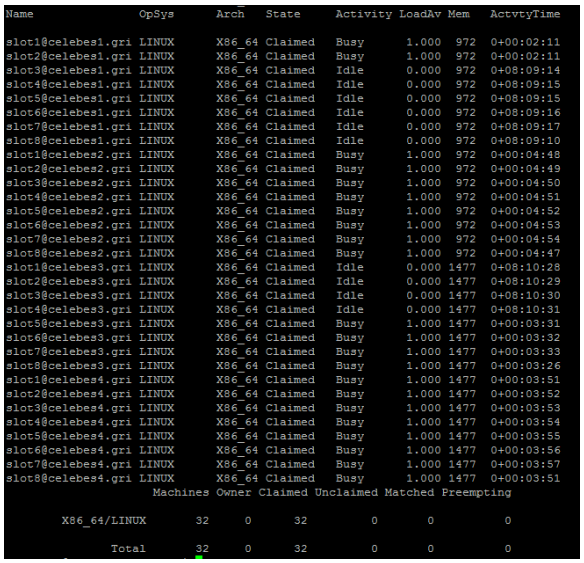
Pengujian yang telah dilakukan bertujuan untuk mengamati dan melakukan uji coba terhadap implementasi sistem yang telah dibangun. Dalam skenario percobaan yang telah dilakukan akan dianalisis tingkat keberhasilan sistem dalam melayani permintaan dari user. Oleh karena itu akan dinilai keberhasilan sistem saat user menyerahkan pekerjaan ke *resource* melalui layanan Globus Toolkit.



```
Name               OpSys      Arch   State     Activity LoadAv Mem   ActvtyTime

slot1@celebes1.gri LINUX      X86_64 Unclaimed Idle      0.020  972  0+01:22:10
slot2@celebes1.gri LINUX      X86_64 Unclaimed Idle      0.000  972  0+01:21:45
slot3@celebes1.gri LINUX      X86_64 Claimed   Idle      0.000  972  0+07:24:14
slot4@celebes1.gri LINUX      X86_64 Claimed   Idle      0.000  972  0+07:24:15
slot5@celebes1.gri LINUX      X86_64 Claimed   Idle      0.000  972  0+07:24:15
slot6@celebes1.gri LINUX      X86_64 Claimed   Idle      0.000  972  0+07:24:16
slot7@celebes1.gri LINUX      X86_64 Claimed   Idle      0.000  972  0+07:24:17
slot8@celebes1.gri LINUX      X86_64 Claimed   Idle      0.000  972  0+07:24:10
slot1@celebes2.gri LINUX      X86_64 Unclaimed Idle      0.000  972  0+01:21:30
slot2@celebes2.gri LINUX      X86_64 Unclaimed Idle      0.000  972  0+01:21:02
slot3@celebes2.gri LINUX      X86_64 Unclaimed Idle      0.000  972  0+01:20:29
slot4@celebes2.gri LINUX      X86_64 Unclaimed Idle      0.000  972  0+01:19:26
slot5@celebes2.gri LINUX      X86_64 Unclaimed Idle      0.000  972  0+01:20:04
slot6@celebes2.gri LINUX      X86_64 Unclaimed Idle      0.000  972  0+01:19:03
slot7@celebes2.gri LINUX      X86_64 Unclaimed Idle      0.000  972  0+01:19:01
slot8@celebes2.gri LINUX      X86_64 Unclaimed Idle      0.000  972  0+01:18:58
slot1@celebes3.gri LINUX      X86_64 Claimed   Idle      0.000 1477  0+07:25:28
slot2@celebes3.gri LINUX      X86_64 Claimed   Idle      0.000 1477  0+07:25:29
slot3@celebes3.gri LINUX      X86_64 Claimed   Idle      0.000 1477  0+07:25:30
slot4@celebes3.gri LINUX      X86_64 Claimed   Idle      0.000 1477  0+07:25:31
slot5@celebes3.gri LINUX      X86_64 Unclaimed Idle      0.000 1477  0+01:23:04
slot6@celebes3.gri LINUX      X86_64 Unclaimed Idle      0.000 1477  0+01:22:38
slot7@celebes3.gri LINUX      X86_64 Unclaimed Idle      0.000 1477  0+01:22:12
slot8@celebes3.gri LINUX      X86_64 Unclaimed Idle      0.000 1477  0+01:21:39
slot1@celebes4.gri LINUX      X86_64 Unclaimed Idle      0.000 1477  0+01:21:42
slot2@celebes4.gri LINUX      X86_64 Unclaimed Idle      0.000 1477  0+01:21:19
slot3@celebes4.gri LINUX      X86_64 Unclaimed Idle      0.000 1477  0+01:20:09
slot4@celebes4.gri LINUX      X86_64 Unclaimed Idle      0.000 1477  0+01:20:04
slot5@celebes4.gri LINUX      X86_64 Unclaimed Idle      0.000 1477  0+01:20:04
slot6@celebes4.gri LINUX      X86_64 Unclaimed Idle      0.000 1477  0+01:19:50
slot7@celebes4.gri LINUX      X86_64 Unclaimed Idle      0.000 1477  0+02:29:22
slot8@celebes4.gri LINUX      X86_64 Unclaimed Idle      0.000 1477  0+02:29:10
                     Machines Owner Claimed Unclaimed Matched Preempting

        X86_64/LINUX       32     0      10        22       0          0

               Total       32     0      10        22       0          0
```

Gambar 5. Status awal condor pada cluster

Gambar 5 menunjukkan status dari *resource* yang akan mengeksekusi pekerjaan. Terdapat 32 slot core CPU yang tersedia dengan 22 status *unclaimed* artinya terdapat 22 Core CPU yang siap mengeksekusi pekerjaan. Ketika user mengirimkan pekerjaan menggunakan globus toolkit maka tugas condor menampung antrian kemudian melakukan penjadwalan terhadap pekerjaan yang akan dieksekusi.



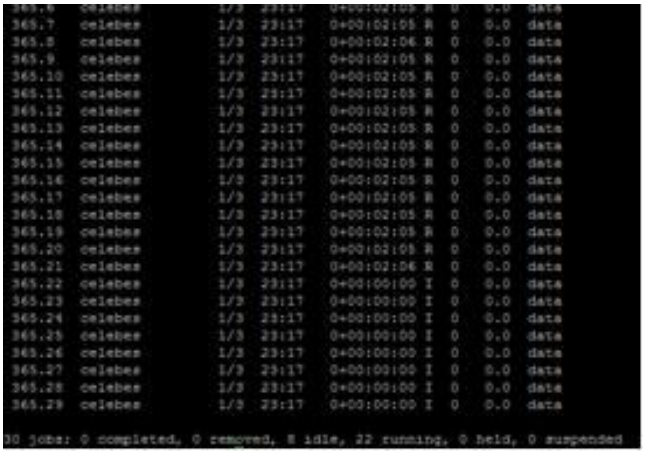
```
Name               OpSys      Arch   State     Activity LoadAv Mem   ActvtyTime

slot1@celebes1.gri LINUX      X86_64 Claimed   Busy      1.000  972  0+00:02:11
slot2@celebes1.gri LINUX      X86_64 Claimed   Busy      1.000  972  0+00:02:11
slot3@celebes1.gri LINUX      X86_64 Claimed   Idle      0.000  972  0+08:09:14
slot4@celebes1.gri LINUX      X86_64 Claimed   Idle      0.000  972  0+08:09:15
slot5@celebes1.gri LINUX      X86_64 Claimed   Idle      0.000  972  0+08:09:15
slot6@celebes1.gri LINUX      X86_64 Claimed   Idle      0.000  972  0+08:09:16
slot7@celebes1.gri LINUX      X86_64 Claimed   Idle      0.000  972  0+08:09:17
slot8@celebes1.gri LINUX      X86_64 Claimed   Idle      0.000  972  0+08:09:10
slot1@celebes2.gri LINUX      X86_64 Claimed   Busy      1.000  972  0+00:04:48
slot2@celebes2.gri LINUX      X86_64 Claimed   Busy      1.000  972  0+00:04:49
slot3@celebes2.gri LINUX      X86_64 Claimed   Busy      1.000  972  0+00:04:50
slot4@celebes2.gri LINUX      X86_64 Claimed   Busy      1.000  972  0+00:04:51
slot5@celebes2.gri LINUX      X86_64 Claimed   Busy      1.000  972  0+00:04:52
slot6@celebes2.gri LINUX      X86_64 Claimed   Busy      1.000  972  0+00:04:53
slot7@celebes2.gri LINUX      X86_64 Claimed   Busy      1.000  972  0+00:04:54
slot8@celebes2.gri LINUX      X86_64 Claimed   Busy      1.000  972  0+00:04:47
slot1@celebes3.gri LINUX      X86_64 Claimed   Idle      0.000 1477  0+08:10:28
slot2@celebes3.gri LINUX      X86_64 Claimed   Idle      0.000 1477  0+08:10:29
slot3@celebes3.gri LINUX      X86_64 Claimed   Idle      0.000 1477  0+08:10:30
slot4@celebes3.gri LINUX      X86_64 Claimed   Idle      0.000 1477  0+08:10:31
slot5@celebes3.gri LINUX      X86_64 Claimed   Busy      1.000 1477  0+00:03:31
slot6@celebes3.gri LINUX      X86_64 Claimed   Busy      1.000 1477  0+00:03:32
slot7@celebes3.gri LINUX      X86_64 Claimed   Busy      1.000 1477  0+00:03:33
slot8@celebes3.gri LINUX      X86_64 Claimed   Busy      1.000 1477  0+00:03:26
slot1@celebes4.gri LINUX      X86_64 Claimed   Busy      1.000 1477  0+00:03:51
slot2@celebes4.gri LINUX      X86_64 Claimed   Busy      1.000 1477  0+00:03:52
slot3@celebes4.gri LINUX      X86_64 Claimed   Busy      1.000 1477  0+00:03:53
slot4@celebes4.gri LINUX      X86_64 Claimed   Busy      1.000 1477  0+00:03:54
slot5@celebes4.gri LINUX      X86_64 Claimed   Busy      1.000 1477  0+00:03:55
slot6@celebes4.gri LINUX      X86_64 Claimed   Busy      1.000 1477  0+00:03:56
slot7@celebes4.gri LINUX      X86_64 Claimed   Busy      1.000 1477  0+00:03:57
slot8@celebes4.gri LINUX      X86_64 Claimed   Busy      1.000 1477  0+00:03:51
                     Machines Owner Claimed Unclaimed Matched Preempting

        X86_64/LINUX       32     0      32         0       0          0

               Total       32     0      32         0       0          0
```

Gambar 6. Status condor setelah user melakukan submit job

Setelah user melakukan submit job ke sumberdaya grid, terdapat perubahan status core CPU dari unclaimed menjadi claimed, lebih banyak dari sebelumnya artinya proses sudah diserahkan dan sudah siap

dieksekusi (lihat Gambar 6). Satu core CPU mampu melakukan beberapa proses tergantung seberapa besar proses yang dikerjakan. Core CPU yang telah menjalankan suatu proses akan menunjukkan status aktivitas buzy dan load average menunjukkan angka 1. Saat menjalankan percobaan yang dilakukan ini, terdapat status aktifitas idle dan buzy. Hal ini menunjukkan bahwa sistem condor menyerahkan pekerjaan dengan melihat performansi dari core CPU, ketika sedang menjalankan proses yang lain dan beban kerjanya sudah maksimum (menunjukkan angka 1.00) maka pekerjaan tidak akan diserahkan ke Core CPU tersebut.

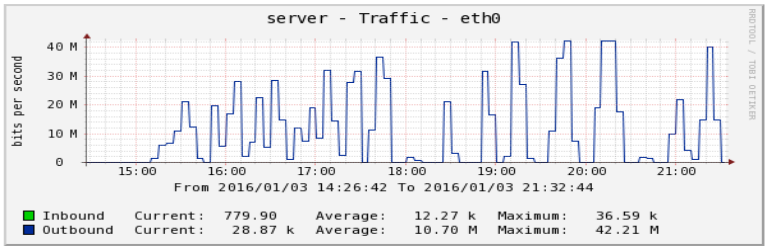
Saat user melakukan *submit job* dengan jobtype *multiple* maka proses tersebut akan dijalankan berulang pada tiap core CPU yang berbeda. Misalnya user mendefinisikan jobtype multiple dengan menggunakan 30 jumlah processor, maka terdapat 30 antrian yang didefinisikan oleh condor (lihat Gambar 7).



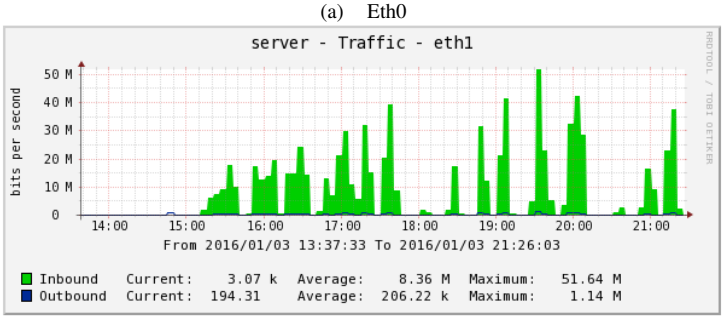
Gambar 7. Status antrian

F. Monitoring Sistem

Dengan memanfaatkan *network monitoring system* yang telah diimplementasikan pada cluster, dapat diketahui bahwa sistem *grid computing* telah berjalan sesuai yang diinginkan. Monitoring ini dilakukan dalam jangka waktu tertentu saat user melakukan submit job ke sumberdaya grid.



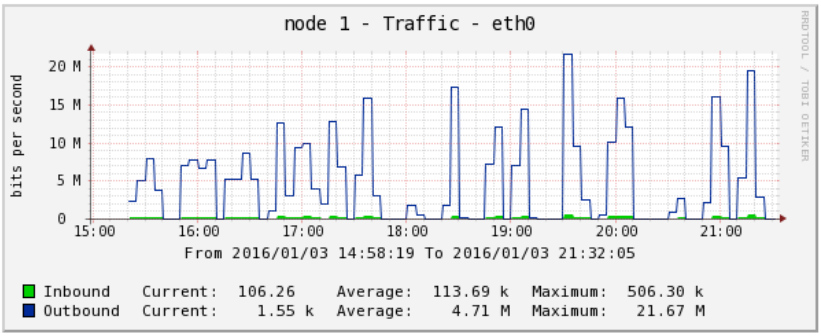
(a) Eth0



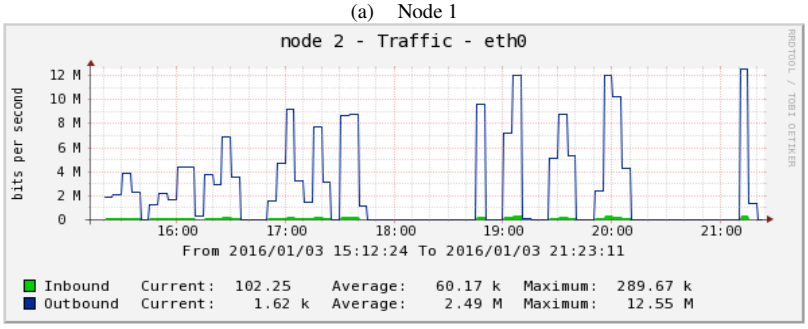
(b) Eth1

Gambar 8. Network traffic pada headnode

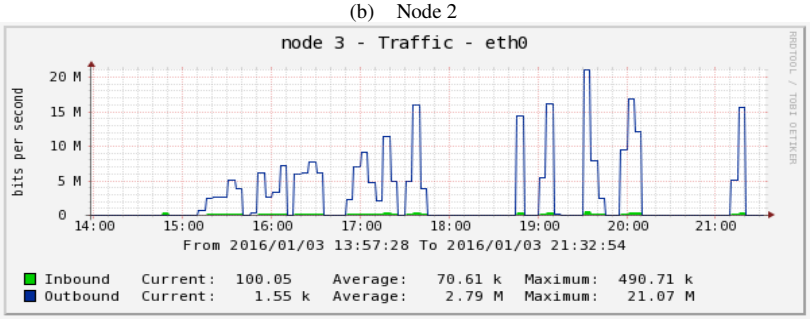
Dapat dilihat pada Gambar 8, interface eth0 yang merupakan interface yang terhubung langsung dengan sistem grid computing. Terlihat grafik data yang masuk sangat kecil yaitu rata-rata 12,27 kB, itu menunjukkan headnode hanya menerima data dalam bentuk script RSL yang berisi perintah eksekusi program. Sedangkan grafik data yang keluar sangat tinggi yaitu rata-rata 10,70 mB, itu menunjukkan bahwa hasil eksekusi dikirim kembali ke mesin user dalam bentuk keluaran dari program yang dieksekusi. Dari Gambar 8 dapat diketahui bahwa proses transfer data berlangsung di interface eth1 yang merupakan interface yang terhubung dengan internal cluster. Transfer data yang signifikan tinggi terlihat pada data yang masuk yaitu rata-rata 8,36 mB. Data tersebut merupakan data yang diterima oleh headnode setelah node mengirimkan hasil eksekusi program yang diberikan. Hal ini dapat dilihat pada grafik data ketiga node tersebut, di mana data yang dikirimkan juga cukup tinggi *traffic*nya. Dengan adanya communication overhead ini, maka penambahan jumlah prosesor hanya akan efektif sampai titik tertentu, dan setelah melampaui titik itu maka *communication overhead* dan transfer data akan menjadi sangat besar sehingga akan memperlambat eksekusi program.



(a) Node 1



(b) Node 2

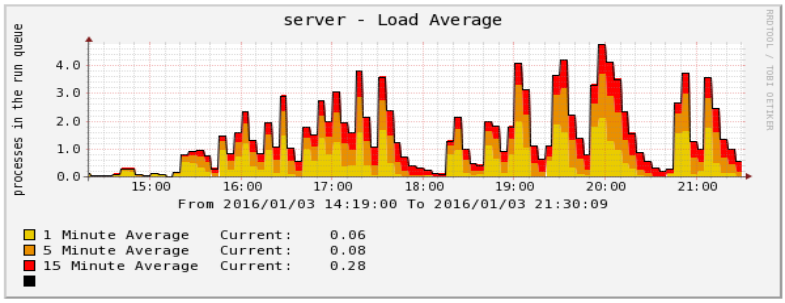


(c) Node 3

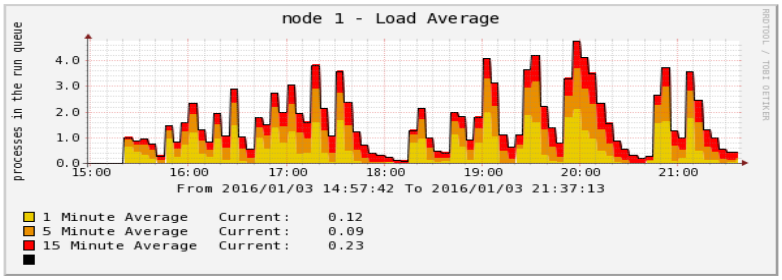
Gambar 9. Network traffic pada node

Melihat dari network traffic pada cluster membuktikan bahwa komunikasi data internal maupun eksternal cluster sudah berjalan dengan baik ketika user melakukan submit program dengan mendefinisikan cluster sebagai resource dan condor sebagai local resource manager.

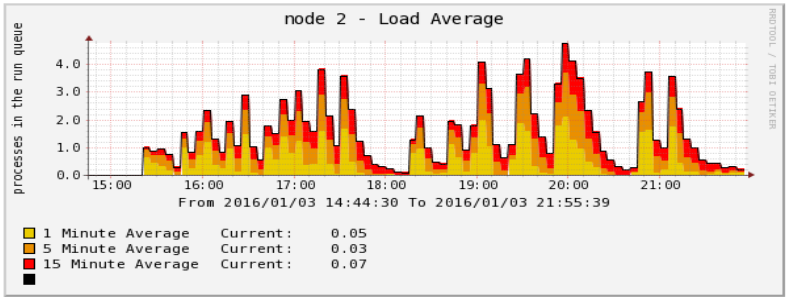
Ketika user menyerahkan pekerjaan dengan beban yang besar maka terjadi peningkatan load average pada headnode dan node. Load average ini berdasarkan pada berapa banyak proses yang menunggu untuk dijalankan dan aktivitas dari input output media penyimpanan.



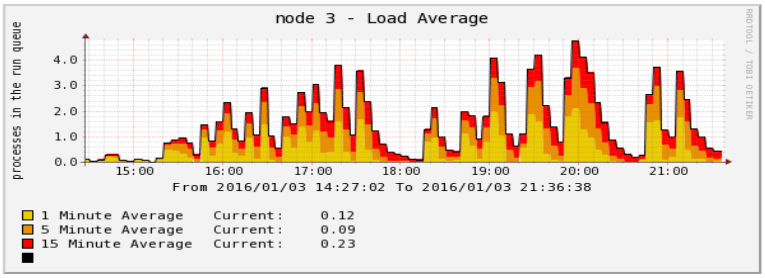
(a) Node 1



(b) Node 2



(c) Node 3



(d) Node 4

Gambar 10. Load average pada cluster

Beban kerja yang mampu diproses setiap node tergantung dari jumlah core CPU yang dimiliki. Dari grafik *load average* tersebut dapat dilihat bahwa skenario uji coba yang dilakukan masih mampu dieksekusi oleh node karena belum terjadi overload baik pada menit 1, menit 5 dan menit 15. Data yang ditunjukkan adalah rata-rata beban kerja yang diproses tiap core CPU masih dibawah 1.00. Maksimal angka yang ditunjukkan saat proses normal yaitu 1.00, dan sistem akan overload jika angka yang ditunjukkan lebih dari 1.00.

Oleh karena itu, waktu proses komputasi yang dihasilkan masih sangat kecil. Artinya setiap kali user mengirimkan pekerjaan dari skenario yang diujicobakan masih mampu diekesekusi karena core CPU mempunyai beban kerja yang belum melewati batas maksimal.

V. KESIMPULAN DAN SARAN

A. *Kesimpulan*

1) Sistem *grid computing* dengan memanfaatkan cluster dapat dibangun dengan melakukan konfigurasi Condor sebagai *local resource manager* pada cluster, dan konfigurasi Globus Toolkit sebagai pengalokasi sumber daya sistem *grid computing*.

2) Pada penelitian ini, konfigurasi Globus Toolkit dengan Condor telah berhasil dan sudah berjalan dengan baik sehingga lingkungan *grid computing* yang dibangun telah menyediakan sumberdaya berbasis cluster yang dapat diakses oleh pengguna. Performansi sistem *grid computing* juga telah diujicoba menggunakan studi kasus perkalian matriks secara parallel, dengan metode *strong scaling* di mana ukuran problem/data dibuat tetap dan jumlah prosesor diperbanyak.

3) Sistem *grid computing* yang dibangun menunjukkan bahwa eksekusi program dapat dipercepat dengan menambah jumlah prosesor, namun tidak berbanding lurus dengan jumlah penambahan tersebut karena adanya *communication overhead*.

B. *Saran*

1) Pada penelitian ini penulis hanya berhasil mengujicoba *jobtype single* dan *multiple*. Untuk penelitian selanjutnya diharapkan sistem grid computing ini dapat dijalankan dengan sistem parallel yaitu dengan menggunakan *jobtype* MPI, karena dalam penelitian ini penulis masih menemukan kendala dalam konfigurasi MPI dengan sistem.

2) Ukuran problem (data matriks) perlu diperbesar.

3) Untuk penelitian selanjutnya diharapkan terhubung dengan jaringan *public* sehingga dapat berbagi resource dari luar lingkungan Universitas Telkom.

REFERENSI

[1] Viktors Berstis, "IBM Redpaper: Fundamentals of Grid Computing", IBM Redbooks Paper, 2003.
[2] Shruti N. Pardeshi, Chitra Patil, and Snehal Dhumale, "Grid Computing Architecture and Benefits", International Journal of Scientific and Research Publications, Volume 3, Issue 8, August 2013, ISSN 2250-3153.
[3] Ian Foster, Carl Kesselman, Steven Tuecke, "The Anatomy of Grid: Enabling Scalable Virtual Organization", International Journal of Supercomputer Applications, 2001.
[4] Ian Foster, "A Globus Primer", Early Draft, http://toolkit.globus.org/toolkit/docs/development/4.1.3/key/GT4_Primer_0.6.pdf.
[5] Bobby Nazief, "RI-GRID: Usulan Pengembangan Infrastruktur Komputasi Grid Nasional", Prosiding e-Indonesia Initiative 2006.
[6] Douglas Thain, Todd Tannenbaum, and Miron Livny, "Condor and the Grid", in Fran Berman, Anthony J.G. Hey, Geoffrey Fox, editors, Grid Computing: Making The Global Infrastructure a Reality, John Wiley, 2003. ISBN: 0-470-85319-0.
[7] http://toolkit.globus.org/toolkit/
[8] https://research.cs.wisc.edu/htcondor/index.html