Implementasi Sistem *Grid Computing* Berbasis *Cluster* di Prodi Ilmu Komputasi

Cluster-Based Implementation of Grid Computing in Computational Science Andi

Farid Arief Nur[1], Fitriyani[2], Izzatul Ummah[3]

[1]Prodi S1 Ilmu Komputasi, Fakultas Informatika, Universitas Telkom
[2] Prodi S1 Ilmu Komputasi, Fakultas Informatika, Universitas Telkom
[3] Prodi S1 Ilmu Komputasi, Fakultas Informatika, Universitas Telkom
[1]**andifharied@gmail.com**, [2]**fitriyani.y@gmail.com**, [3]**izzatul.ummah@gmail.com**

Abstrak

***Grid computing* adalah teknologi komputasi terdistribusi yang memanfaatkan sumber daya yang terhubung melalui jaringan komputer secara bebas tapi terkoordinasi dengan mekanisme tertentu. Dengan menyediakan sumber daya yang dapat dipakai bersama dapat mempermudah akses dan meningkatkan *Quality of Service.* Pembangunan infrastruktur *Grid computing* tidaklah mudah karena membutuhkan kemampuan dan pengalaman di dalam instalasi dan konfigurasi program berbasis Linux.**
Tujuan akhir dari penelitian ini, penulis membangun infrastruktur *Grid computing* berbasis pada *cluster* yang akan dijadikan sebagai sumber daya *back-end* dengan menggunakan *Globus Tookit* sebagai *midlleware* pengalokasian sumber daya. Penelitian ini menggunakan jaringan lokal Universitas Telkom.
Kata kunci: *grid computing, globus toolkit, cluster*

Abstract

Grid computing is a technology that utilizes a distributed computing resources which connected through a computer network independently but coordinated with a specific mechanism . By providing resources that can be used together to facilitate access and improve Quality of Service . Grid computing infrastructure development is not easy because it requires the ability and experience in the installation and configuration of Linux-based program . The final goal of this study , the authors build a Grid computing infrastructure based on the cluster that will serve as a back -end resource by using Globus tookit as midlleware allocation of resources. This study uses the University of Telkom's local network .
Keyword: *grid computing, globus toolkit, cluster*

1. Pendahuluan

Saat ini kebutuhan akan komputasi kinerja tinggi semakin meningkat seiring dengan semakin banyaknya masalah yang membutuhkan pemrosesan dengan ukuran data yang cukup besar. Yang menjadi permasalahan saat ini, sering kali dalam penyelesaian masalah tersebut terkendala dengan kinerja dari komputer yang masih rendah sehingga membutuhkan waktu yang lama atau bahkan gagal dalam memproses data yang cukup besar. Istilah *Floating Point Operations per Second (FLOPS)* dipakai dalam lingkungan komputasi kinerja tinggi dalam menilai seberapa cepat sistem menyelesaikan pekerjaan dalam waktu singkat, sedangkan kemampuan sistem untuk dapat mengeksekusi pekerjaan dalam kapasitas besar dalam periode waktu yang cukup lama merujuk pada istilah *High Throughput Computing (HTC).*

Pada umumnya pemecahan masalah komputasi selalu tergantung pada *mainframe* yang besar seperti *supercomputer* sehingga dibutuhkan dana yang cukup banyak. *Grid computing* adalah infrastruktur perangkat keras dan perangkat lunak yang dapat menyediakan akses yang bisa diandalkan, konsisten, tahan lama dan tidak mahal terhadap kemampuan komputasi mutakhir yang tersedia [1]. Mengacu pada definisi tersebut, maka sumber daya *grid computing* dapat menyediakan sistem *High Throughput* secara maksimal dan efisien.

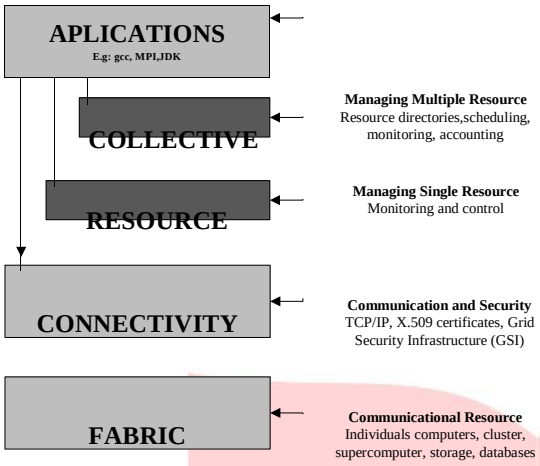
Melihat dari berbagai kebutuhan akademik di Telkom University misalnya, penyelesaian tugas akhir Mahasiswa yang sering kali membutuhkan komputer yang memiliki kinerja yang tinggi dan ketebatasan akses maka dari itu, pada tugas akhir ini akan dibangun sebuah sistem *grid computing* dengan memanfaatkan beberapa unit computer yang akan dijadikan *cluster* sebagai *back-end* di Prodi Ilmu Komputasi sehingga tidak perlu lagi dana yang besar untuk menghasilkan lingkungan komputasi kinerja tinggi dengan memudahkan akses ke sumber daya tersebut. Untuk kedepannya juga diharapkan dapat dimanfaatkan dalam hal pengembangan riset dari Mahasiswa maupun dosen.

2. Dasar Teori

2.1 Grid Computing

Grid computing adalah penggunaan sumber daya yang melibatkan banyak komputer yang terdistribusi dan terpisah secara geografis untuk memecahkan persoalan komputasi dalam skala besar. (http://id.wikipedia.org)

2.1.1 Arsitektur Umum *Grid Computing*



Gambar 2.1: Arsitektur Grid Computing

Pada layer *fabric* menyediakan berbagai kebutuhan yang diperlukan sebagai sumber daya dalam infrastruktur *grid computing*. Layer *connectivity* bertugas meyediakan protocol jaringan dan sarana komunikasi yang digunakan dalam menghubungkan sistem. Layer *resource* bertugas mengatur layanan yang di pakai sumber daya yang tersebar. Fungsi layer *collective* sama dengan layer *resource*, yang membedakan yaitu layer *collective* mengatur sumber daya yang sifatnya berkelompok.

2.1.2 Konsep Dasar *Grid Computing*

1. Sumber daya dikelola dan dikendalikan secara lokal.
2. Sumber daya berbeda dapat mempunyai kebijakan dan mekanisme berbeda, mencakup Sumber daya komputasi dikelola oleh sistem batch berbeda, Sistem storage berbeda pada node berbeda, Kebijakan berbeda dipercayakan kepada user dalam memilih sumber daya yang berbeda pada sistem grid.
3. Sifat alami dinamis: Sumber daya dan pengguna dapat sering berubah
4. Tiga hal yang di-,sharing dalam sebuah sistem grid, antara lain : *Resource*, *Network* dan Proses. Kegunaan / layanan dari sistem grid sendiri adalah untuk melakukan *high throughput computing* dibidang penelitian, ataupun proses komputasi lain yang memerlukan banyak *resource* komputer.

2.1.3 Prinsip Kerja *Grid Computing*

1. Virtualisasi

Setiap sumberdaya (semisal komputer, disk, komponen aplikasi dan sumber informasi) dikumpulkan bersama-sama menurut jenisnya, lalu disediakan bagi *user*. *Virtualisasi* berarti meniadakan koneksi secara fisik antara penyedia dan konsumen sumberdaya, dan menyiapkan sumberdaya untuk memenuhi kebutuhan tanpa konsumen mengetahui bagaimana permintaannya bisa terlayani.

2. Provisioning

Ketika *user* meminta sumberdaya melalui layer *virtualisasi*, sumberdaya tertentu di belakang layer didefinisikan untuk memenuhi permintaan tersebut, dan kemudian dialokasikan ke *user*. Provisioning sebagai bagian dari *grid computing* berarti bahwa sistem menentukan bagaimana cara memenuhi kebutuhan *user* seiring dengan mengoptimasi jalannya sistem secara keseluruhan.

2.2 Globus Toolkit

Globus Toolkit adalah sebuah *software toolkit* yang dibangun oleh The Globus Alliance (2002), yang dapat digunakan untuk membangun *grid system*.

2.2.1 Komponen Globus Toolkit

1. GRAM: Grid Resource Allocation & Management

Komponen ini bertanggung jawab dalam mengelola seluruh sumber daya komputasi yang tersedia dalam sistem *grid computing*. Pengelolaan ini mencakup eksekusi program pada seluruh komputer yang tergabung dalam sistem *grid computing*, mulai dari inisiasi, monitoring, sampai penjadwalan (scheduling) dan koordinasi antar- proses.

Suatu hal yang menarik dengan sistem *globus toolkit* adalah kemampuannya untuk bekerja sama dengan sistem-sistem pengelolaan sumber daya komputasi yang telah ada sebelumnya seperti Condor, PVM, atau MPI. Dengan mekanisme ini maka program-program yang telah dibangun sebelumnya tidak perlu dibangun ulang atau kalaupun harus dimodifikasi, modifikasinya minimum, jika akan dijalankan dalam lingkungan *grid computing* berbasis *globus toolkit*.

2. RFT/GridFTP: Reliable File Transfer/Grid File Transfer Protocol

Komponen ini memungkinkan pengguna mengakses data yang berukuran besar dari simpul-simpul komputasi yang tergabung dalam sistem *grid computing* secara efisien dan dapat diandalkan. Hal ini penting karena kinerja komputasi tidak saja bergantung pada seberapa cepat komputer-komputer yang tergabung dalam sistem *grid computing* ini mengeksekusi program, tetapi juga seberapa cepat data yang dibutuhkan dalam komputasi tersebut dapat diakses. Perlu diingat bahwa, data yang dibutuhkan oleh suatu proses tidak selalu berada pada komputer yang mengeksekusi proses tersebut.

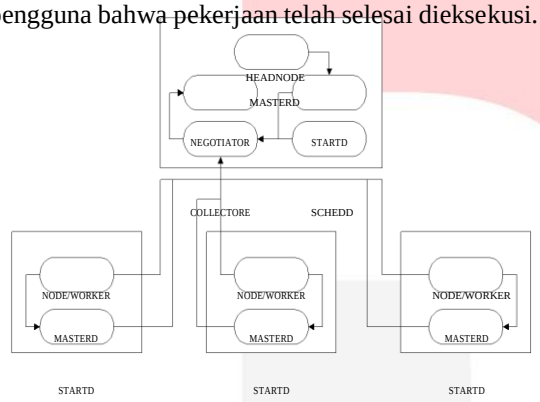
3. GSI: Grid Security Infrastructure

Komponen ini bertanggung jawab atas keamanan sistem *grid computing* secara keseluruhan. Komponen ini pula yang merupakan salah satu ciri pembeda teknologi *globus toolkit* dengan teknologi-teknologi pendahulunya seperti PVM atau MPI. Dengan diterapkannya mekanisme keamanan yang terintegrasi dengan komponen-komponen *grid computing* lainnya, sistem berbasis

teknologi *grid computing* seperti *globus toolkit* dapat diakses oleh publik (WAN) tanpa menurunkan tingkat keamanannya.

2.3 *Condor*

Condor adalah *local resource manager* dalam sistem yang bertugas mengatur beban kerja sekaligus sistem penjadwal dalam *cluster*. Condor dikembangkan oleh Departemen Ilmu Komputer, Universitas Wisconsin-Madison. Condor menyediakan mekanisme pengaturan pekerjaan, aturan penjadwalan, skema prioritas, pemantauan dan pengaturan sumber daya. Pengguna cukup menyerahkan pekerjaan kepada condor dan condor akan menentukan kapan dan dimana pekerjaan tersebut akan dieksekusi, memantau kemajuannya dan pada akhirnya menginformasikan kepada pengguna bahwa pekerjaan telah selesai dieksekusi.



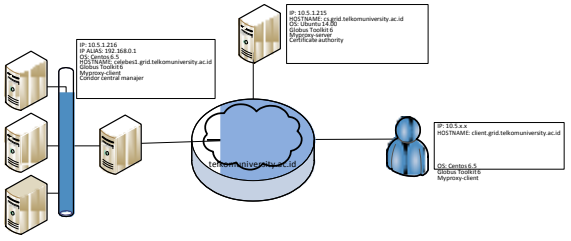
Gambar 2.4: Sistem kerja condor

Condor_master bertugas menyederhanakan proses administrasi sistem yang harus berjalan pada *central manager* dan semua *node*. *Control_startd* bertugas memberikan informasi tentang ketersediaan mesin, *daemon* ini di pakai pada *headnode* dan *node* sehingga masing-masing bias menjalakan eksekusi. *Condor_schedd* bertugas memberikan pekerjaan ke *node* yang tersedia dan telh menjalankan daemon *condor_startd*. *Condor_collector* bertugas mengumpulkan informasi dari semua mesin yang di tampung pada *control manager*. *Control_negotiator* bertugas melakukan pengecekan terhadap status mesin yang bersedia menjalankan proses dengan mempertimbangkan kebutuhan pekerjaan yang akan dieksekusi, daemon ini harus dijalankan pada *control manager*.

3. Perancangan Sistem

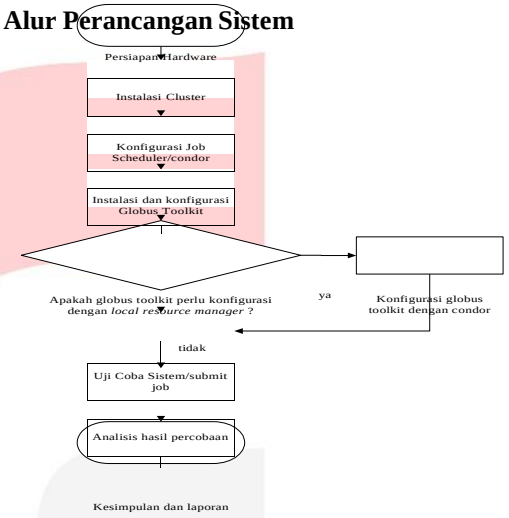
3.1 Deskripsi Sistem

Perancangan dan implementasi instalasi sistem *grid computing* dilakukan di lingkungan Universitas Telkom. Pada tugas akhir ini menggunakan satu buah *cluster* sebagai sumber daya grid computing yaitu *celebes cluster*. *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. Dalam menghubungkan antar sistem tersebut digunakan jaringan lokal kampus Universitas Telkom.



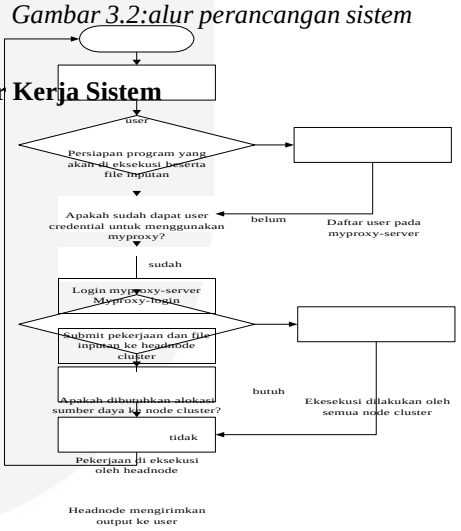
Gambar 3.1: Skema lingkungan grid computing

3.2 Alur Perancangan Sistem



Gambar 3.2:alur perancangan sistem

3.3 Alur Kerja Sistem



Gambar 3.2: Alur kerja sistem

4. Implementasi Sistem

4.1 Pengujian Sistem

Pada tahapan uji coba ini dilakukan dengan manjalankan program perkalian dua matriks misalnya perkalian dari dua matriks persegi antara matriks A dan matriks B dengan elemen sehingga hasil perkalian yaitu matriks C dengan NxN elemen juga. Dapat dituliskan dengan persamaan sebagai berikut:

Dari persamaan tersebut dibuat kedalam algoritma yang menunjukkan beberapa perulangan yang harus dijalankan, algoritma 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];
```

Terdapat berbagai cara dalam mengoptimalkan kinerja algoritma perkalian matriks ini, salah satunya dengan menyesuaikan dengan arsitektur dan struktur data bahasa pemrograman yang dipakai. Namun, dalam tugas akhir ini optimalisasi tidak dilakukan pada algoritma sehingga tidak menjadi pertimbangan dalam percobaan yang dilakukan.

Selanjutnya menerapkan algoritma tersebut kedalam bahasa pemrograman C++. Dalam pengujian akan diserahkan pekerjaan dengan skala matriks yang berbeda yaitu , , , , , , , , , dan .

4.2 Asumsi Pengujian Sistem

1. Submit pekerjaan dilakukan dari komputer *user* yang telah terinstall *globus toolkit*. Dalam tugas akhir 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 mendafatarkan *user credential*.

4.3 Pengujian Menggunakan *Fork*

Penyerahan pekerjaan dari komputer *user* mendefinisikan *local resource manager fork* yang akan dipakai dalam eksekusi pekerjaan di *resource* 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:

Table 4.1: Waktu eksekusi menggunakan fork (*second)*

N	Waktu Eksekusi
500	0.77
1000	10.02
1500	45.53
2000	120.59
2500	225.67
3000	361.34
3500	602.17
4000	893.66
4500	1277.24
5000	1728.1

Dari data yang dihasilkan dapat disimpulkan bahwa banyaknya N elemen matriks berbanding lurus dengan waktu eksekusi program, semakin banyak jumlah N elemen maka semakin lama waktu yang dibutuhkan untuk mengeksekusi program. Hal ini tentu saja dipengaruhi oleh keterbatasan *processor* dalam mengeksekusi program.

4.4 Pengujian Menggunakan *Condor*

Dalam percobaan ini dilakukan pendefinisian *local resource manager condor* yang akan dipakai dalam proses penyeraha pekerjaan ke *resource* 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 jumlah *core CPU* yang berbeda yaitu 1, 10, 20, dan 30. *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 4.2: 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 1 *core CPU (single)* dengan *multiple*. Waktu eksekusi meningkat lebih dari dua kali lipat. Sedangkan ketika menjalankan *multiple*, semakin banyak jumlah *core CPU* yang dipakai maka semakin cepat waktu eksekusi. Hal ini disebakan 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 kinerja dari *core CPU* tersebut.

4.5 Optimalisasi Sistem

Untuk mendapatkan alokasi sumber daya yang maksimal diperlukan beberapa konfigurasi dalam sistem. Dalam tugas akhir ini, optimalisasi sistem dilakukan dengan memanfaatkan fitur dari condor. Condor memungkinkan penyedia mengatur prioritas pekerjaan yang akan di eksekusi 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.

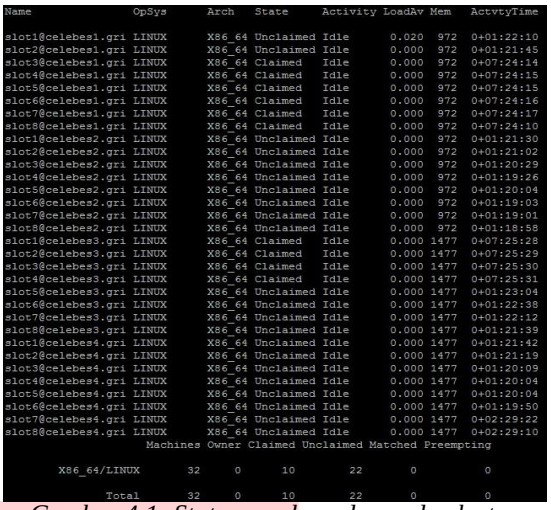
Dalam sistem ini diperlukan perubahan dalam berkas konfigurasi lokal pada condor dengan menentukan ekspresi sebagai berikut:

```
## 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*.

4.6 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*.

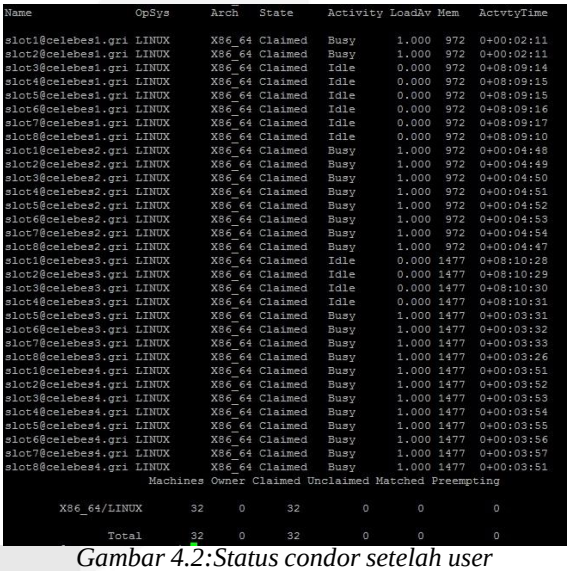


Name	OpSys	Arch	State	Activity	LoadAv	Mem	ActvtyTime
slot1@celebes1.gri	LINUX	X86_64	Unclaimed	Idle	0.020	972	0+01:22:10
slot2@celebes1.gri	LINUX	X86_64	Unclaimed	Idle	0.000	972	0+01:21:45
slot3@celebes1.gri	LINUX	X86_64	Claimed	Idle	0.000	972	0+07:24:14
slot4@celebes1.gri	LINUX	X86_64	Claimed	Idle	0.000	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 4.1: Status awal condor pada cluster

Gambar 4.1 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 di esksekusi.



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 4.2:Status condor setelah user menyerahkan pekerjaan

Setelah *user* menyerahkan pekerjaan ke *resource* terdapat perubahan status *core CPU* dari *unclaimed* menjadi *claimed*, lebih banyak dari sebelumnya artinya proses sudah diserahkan dan sudah siap dieksekusi. 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 penyerahan pekerjaan dengan mendefinisikan *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*.

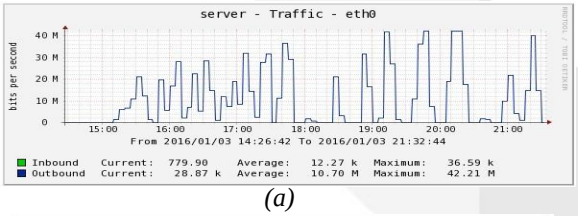
```
-- Schedd: celebes1.grid.telkomuniversity.ac.id : <10.5.1.216:7961?...
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
 365.0   celebes         1/3  23:17   0+00:02:06 R  0   0.0  data
 365.1   celebes         1/3  23:17   0+00:02:06 R  0   0.0  data
 365.2   celebes         1/3  23:17   0+00:02:05 R  0   0.0  data
 365.3   celebes         1/3  23:17   0+00:02:05 R  0   0.0  data
 365.4   celebes         1/3  23:17   0+00:02:05 R  0   0.0  data
 365.5   celebes         1/3  23:17   0+00:02:05 R  0   0.0  data
 365.6   celebes         1/3  23:17   0+00:02:05 R  0   0.0  data
 365.7   celebes         1/3  23:17   0+00:02:05 R  0   0.0  data
 365.8   celebes         1/3  23:17   0+00:02:06 R  0   0.0  data
 365.9   celebes         1/3  23:17   0+00:02:05 R  0   0.0  data
 365.10  celebes         1/3  23:17   0+00:02:05 R  0   0.0  data
 365.11  celebes         1/3  23:17   0+00:02:05 R  0   0.0  data
 365.12  celebes         1/3  23:17   0+00:02:05 R  0   0.0  data
 365.13  celebes         1/3  23:17   0+00:02:05 R  0   0.0  data
 365.14  celebes         1/3  23:17   0+00:02:05 R  0   0.0  data
 365.15  celebes         1/3  23:17   0+00:02:05 R  0   0.0  data
 365.16  celebes         1/3  23:17   0+00:02:05 R  0   0.0  data
 365.17  celebes         1/3  23:17   0+00:02:05 R  0   0.0  data
 365.18  celebes         1/3  23:17   0+00:02:05 R  0   0.0  data
 365.19  celebes         1/3  23:17   0+00:02:05 R  0   0.0  data
 365.20  celebes         1/3  23:17   0+00:02:05 R  0   0.0  data
 365.21  celebes         1/3  23:17   0+00:02:06 R  0   0.0  data
 365.22  celebes         1/3  23:17   0+00:00:00 I  0   0.0  data
 365.23  celebes         1/3  23:17   0+00:00:00 I  0   0.0  data
 365.24  celebes         1/3  23:17   0+00:00:00 I  0   0.0  data
 365.25  celebes         1/3  23:17   0+00:00:00 I  0   0.0  data
 365.26  celebes         1/3  23:17   0+00:00:00 I  0   0.0  data
 365.27  celebes         1/3  23:17   0+00:00:00 I  0   0.0  data
 365.28  celebes         1/3  23:17   0+00:00:00 I  0   0.0  data
 365.29  celebes         1/3  23:17   0+00:00:00 I  0   0.0  data

30 jobs; 0 completed, 0 removed, 8 idle, 22 running, 0 held, 0 suspended
```

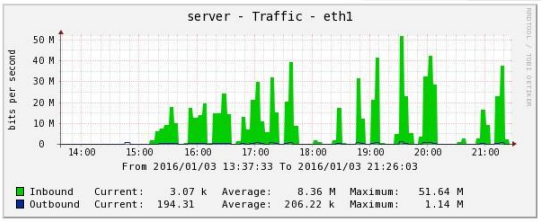
Gambar 4.3: Status antrian

4.7 Monitoring Sistem

Dengan memanfaatkan *network monitoring system* yang telah diimplementasikan pada *cluster* dapat jadikan acuan bahwa sistem *grid computing* telah berjalan. *Monitoring* ini dilakukan dalam jangka waktu tertentu saat *user* mengirimkan pekerjaan ke *resource*.



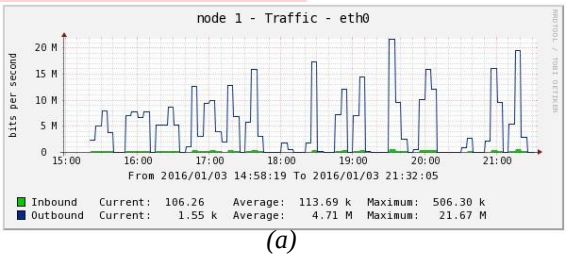
(a)



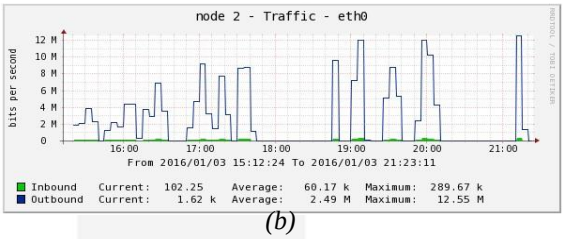
(b)

Gambar 4.4:(a)eth0 (b)eth1 Network traffic pada headnode

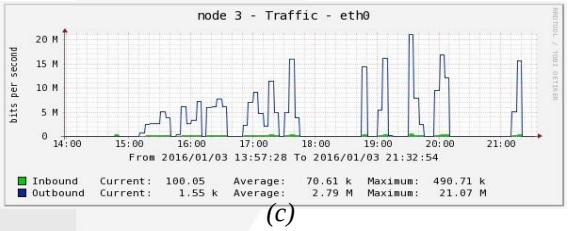
Dapat dilihat pada *Gambar 4.4 (a) interface eth0* yang merupakan *interface* yang terhubung langsung dengan sistem *grid computing*. Terlihat grafik data yang masuk sangat kecil yaitu rata-rata 12,27 kB, itu menunjukkan *headnode* hanya menerima data dalam bentuk *script RSL* yang berisi perintah eksekusi 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. Pada *Gambar 4.4(b)* menunjukkan proses data yang berlangsung di *interface eth1* yang merupakan *interface* yang terhubung dengan internal *cluster*. 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, data yang dikirimkan juga siginfikan tinggi.



(a)



(b)

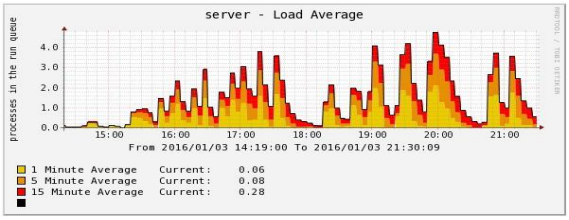


(c)

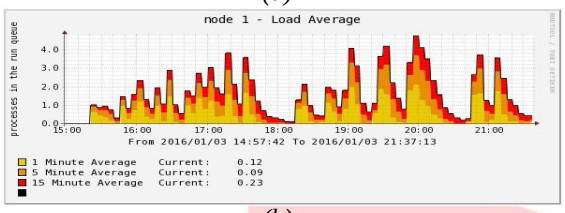
Gambar 4.5:(a)node 1 (b)node 2 (c)node 3 Network traffic pada node

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*.

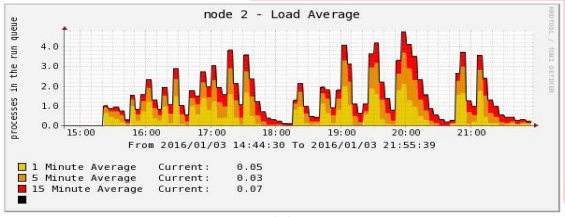
Ketika *user* menyerahkan pekerjaan dengan beban yang besar maka terjadi peningkatan *load average* pada *headnode* dan *node*. *Load average* ini berdasarkan berapa banyak proses yang menunggu untuk dijalankan dan aktivitas dari *input output* media penyimpanan.



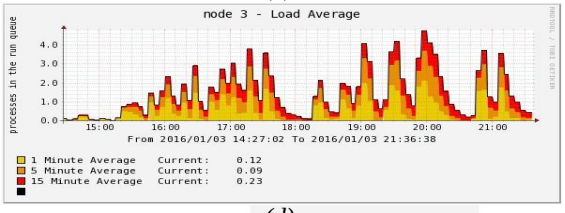
(a)



(b)



(c)



(d)

Gambar 4.6:(a)server (b)node 1 (c)node 2 (d)node 3 Load average pada cluster

Beban kerja yang mampu diproses setiap *node* tergantung dari jumlah *core CPU* yang dimiliki. Dari grafik *load average* tersebut menunjukkan 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 merupakan rata-rata beban kerja yang diproses tiap *core CPU* masih dibawah 1.00. Maksimal angka yang ditunjukkan saat proses normal yaitu 1.00, sistem akan *overload* jika angka yang ditunjukkan lebih dari 1.00.

Oleh karena itu, waktu proses komputasi yang di hasilkan masih sangat kecil. Artinya setiap kali *user* mengirimkan pekerjaan dari skenario yang dicobakan masih mampu di ekesekusi karena *core CPU* mempunyai beban kerja yang belum melewati batas maksimal.

5. Kesimpulan dan Saran

5.1 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. Optimalisasi sumber daya dilakukan dengar mengatur konfigurasi *condor_policy* pada *cluster* agar condor memprioritaskan antrian yang bersifat *dedicated* atau antrian yang membutuhkan lebih dari satu *core CPU* sehingga semua *core CPU* dapat terpakai.
3. Pada kasus perkalian matriks yang dianalisis, sistem *grid computing* berhasil dijalankan.
4. Konfigurasi *globus toolkit* dengan *condor* telah berhasil dan sudah berjalan dengan baik sehingga lingkungan *grid computing* yang dibangun telah menyediakan *resource* berbasis *cluster.*
5. Penyerahan pekerjaan/submit job dapat dilakukan oleh *user* dengan terlebih dahulu mendefinisikan kebutuhan, misalnya *jobtype* dan *local resource manager* yang dibutuhkan.

a. Saran

1. Pada 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. Untuk menunjang *Quality of Service* diharapkan penelitian ini dapat dikembangkan dengan menyediakan *web interface* yang mendukung sistem ini.
3. Untuk penelitian selanjutnya diharapkan terhubung dengan jaringan *public* sehingga dapat berbagi *resource* dari luar lingkungan Universitas Telkom.

Daftar Pustaka:

[1] Bertis, Victors. (2002). "Fundamentals of *Grid computing*". *Red Books Paper.*1-28.

[2] Baker,Mark., Buyya,Rajkumar. Laforenza,Domenico. (2002). "Grids and Grid Technologies for Wide-Area Distributed Computing". *Software-Practice and Experience.* 1-30

[3] Foster,Ian, (2002). "What is the Grid? A Three Point Checklist". *Grid Today*. http://wwwfp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf.

[4] Parastatidis,Savas., Watson,Paul. and Webber,Jim. (2005). *GRID COMPUTING USING WEB SERVICE.* Newcastle: University of Newcastle Upon Time.

[5] Chuan,Lin Lai., Chao,Tung Yang. (2003). "Construct a *Grid computing* Environment on Multiple Linux PC Clusters". *Tunghai Science.* 5,107-124.

[6] Rumagit, Arthur. "Implementasi *Grid computing* Untuk High Throughput Computing". 25 Oktober 2014.

http://download.portalgaruda.org/article.php?article=81612&val=1029.

[7] Budiharto,Widodo. "Implementasi dan Evaluasi Penerapan Globus Toolkit Untuk Aplikasi *Grid computing*". 10 Oktober 2014. http://library.binus.ac.id/eColls/eJournal/68.TI%20Widodo_Budiharto%20%20Implementasi%20dan%20Evaluasi%20Penerapan%20Globus%20Toolkit%20-Ok.pdf.

[8] Waskito, Wahyu. (2010). "*Grid computing* dan Aplikasinya" 13 Oktober 2014. http://wahyuwaskito.wordpress.com/2010/10/21/grid-computing-dan-aplikasinya-2/ .

[9] Anonim. (2014). "*Grid computing* pada Pengembangan e-sciense". 30 Oktober 2014. http://aboutkuliah.blogspot.com/2014/06/grid-computing-pada-pengembangan-e.html.

[10] Nazief,Bobby. (2006)."RI-GRID: Usulan Pengembangan Infrastruktur Komputasi Grid Nasional". *e-Indonesia Initiative.*

[11] Josef. (2006)."Panduan Instalasi dan Konfigurasi Condor High-Throughput Computing System". Fakultas Ilmu Komputer Universitas Indonesia.