IMPLEMENTASI GRID COMPUTING DENGAN MENGGUNAKAN PENGALAMATAN IPv6

Ahmad Makhsun[1], Idris Winarno, SST, M.Kom.[2]
[1]Mahasiswa Jurusan Teknik Informatika, [2]Dosen Jurusan Teknik Informatika
Jurusan Teknik Informatika
Politeknik Elektronika Negeri Surabaya
Institut Teknologi Sepuluh Nopember
Kampus ITS Sukolilo, Surabaya 60111, Indonesia
Tel: +62 (31) 594 7280; Fax: +62 (31) 594 6114
e-mail : makhsun@student.eepis-its.edu, idris@eepis-its.edu

Abstrak

Untuk menghasilkan karya yang lebih dibutuhkan juga perangkat komputasi yang setara dengan karya tersebut. Habisnya IPv4 valid juga menjadi kendala tersendiri di era sekarang. Implementasi grid computing dengan menggunakan pengalamatan IPv6 menjadi solusi dari masalah yang ada. Grid Computing mampu melakukan komputasi dalam skala besar yang terdistribusi dan terpisah secara geografis. Grid Computing diimplementasikan pada 9 komputer yang membentuk 1 server dan 2 cluster dimana semua node menggunakan pengalamatan IPv6. Implementasi cluster menggunakan openmpi, job scheduler menggunakan condor, dan grid engine menggunakan globus toolkit. Perkalian matrik dan Prime sum digunakan untuk menguji coba kecepatan dari sistem grid computing untuk menunjukkan seberapa cepat sistem grid computing mengeksekusi aplikasi (program parallel). Hasil dari uji coba menunjukkan bahwa implementasi openmpi menggunakan IPv6 mampu meningkatkan komputasi lebih dari 5 kali dibandingkan dengan komputasi pada 1 komputer, condor tidak bisa diimplementasikan karena versinya belum memenuhi, dan globus tidak cocok diintegrasikan dengan openmpi. Dua problem tersebut digantikan dengan Portal PHP yang difungsikan sebagai administrasi jobs. Hasilnya, aplikasi Prime Sum dapat dijalankan pada Portal untuk diteruskan ke semua node pada cluster.

Kata Kunci : IPv6, Grid Computing, Komputasi Parallel

1. Latar Belakang

Komputasi Grid adalah penggunaan sumber daya yang melibatkan banyak komputer yang terdistribusi dan terpisah secara geografis untuk memecahkan persoalan komputasi dalam skala besar[1]. Grid Computing erat kaitannya dengan metode komputasi paralel. Metode ini dapat membagi kerja komputer menjadi beberapa bagian sehingga, tidak memberatkan kerja komputer itu sendiri dan mempercepat kerja komputer. Sebagai contoh, bila ada suatu perintah untuk mencari satu angka dari 100 angka, komputer tersebut memiliki 10 processor. Dengan adanya komputasi paralel, komputer tersebut dapat memecah kerja menjadi 10 bagian untuk mencari angka tersebut. hal ini tentu saja dapat mempercepat dan memperingan kerja komputer. Tentu saja masalah pembagian kerja komputer tersebut dalam skala kecil. Tapi dari sinilah grid computing dikembangkan. Grid computing semakin dikembangkan dengan adanya jaringan dan internet. Dengan jaringan, kerja komputer terbagi-bagi di satu tempat dan tempat lain, namun pekerjaannya tetap satu atau terhubung. Grid Computing memanfaatkan kekuatan pengolahan berbagai unit komputer, dan menggunakan kekuatan proses untuk menghitung satu pekerjaan. Pekerjaan itu sendiri dikontrol oleh satu komputer utama, dan dipecah menjadi beberapa tugas yang dapat dilaksanakan secara bersamaan pada komputer yang berbeda. Tugas-tugas ini tidak perlu saling eksklusif, meskipun itu adalah skenario yang ideal. Sebagai tugas lengkap pada berbagai unit komputasi, hasil dikirim kembali ke unit pengendali, yang kemudian collates itu membentuk keluaran kohesif. Satu masalah akan kurangnya sumber daya untuk komputasi tinggi sudah terpenuhi dengan kehadiran grid computing. Namun masalah tidak berhenti di situ saja. Salah satu komponen yang terpenting juga dalam grid computing adalah konektifitas atau jaringan. Tidak akan membentuk sebuah grid computing kalau tidak ada jaringan. Didalam sebuah jaringan, tidak asing lagi dengan penggunaan IP Address. Lebih dari 20 tahun manusia menggunakan IPv4 sebagai protokol jaringan. Namun, jumlah IPv4 yang mencapai 4,3 milyar sudah

habis tanggal 15 April 2011 [2]. Tentu saja hal ini menjadi kendala bagi pengguna internet, khususnya grid computing ini yang juga membutuhkan IP Address valid untuk konektifitasnya. Muncullah protokol jaringan baru yang merupakan pengganti dari IPv4 yang sudah habis yaitu IPv6. Dengan protokol ini, pengguna internet tidak perlu khawatir lagi akan kebutuhan penggunaan IP Address. Atas dasar itulah pada proyek akhir ini, penulis mengimplementasikan grid computing dengan menggunakan pengalamatan IPv6.

2. Perumusan Masalah

Beberapa lingkup permasalahan yang akan dibahas dalam penelitian proyek akhir ini diantaranya:

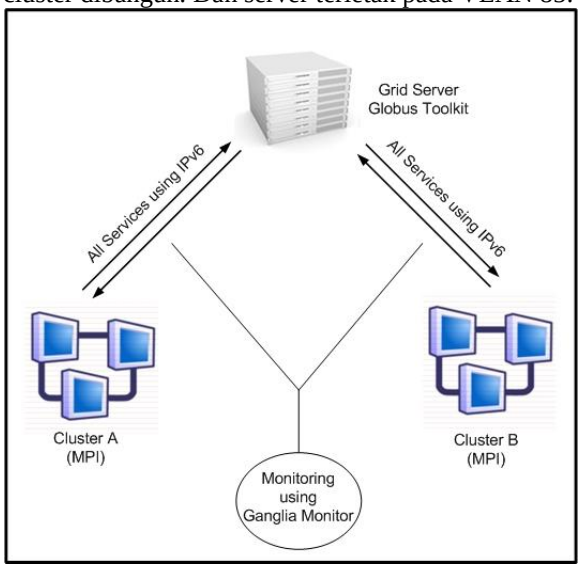
1. Membangun Cluster Computer, yang merupakan bagian terpenting dari grid computing yang akan mengerjakan job-job yang ada.
2. Membangun Headnode, untuk menghubungkan cluster-cluster yang sudah dibangun sekaligus sebagai pengatur job yang akan dikerjakan.
3. Mengimplementasikan IPv6 pada service yang ada dalam grid computing untuk menggantikan IPv4 valid yang sudah habis.
4. Mengintegrasikan Cluster, Condor dan Globus untuk membentuk Grid Computing.

3. Batasan Masalah

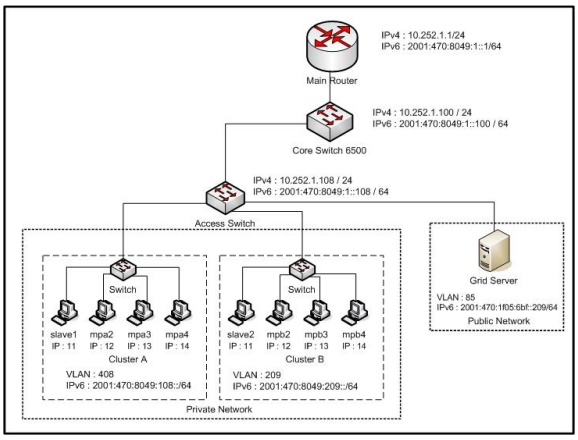
1. Sistem operasi yang akan digunakan adalah Linux Debian Squeeze.
2. Aplikasi yang digunakan untuk membentuk lingkungan cluster adalah OpenMPI.
3. Aplikasi yang digunakan untuk Grid Engine menggunakan Globus Toolkit.
4. Implementasi IPv6 pada semua node untuk melihat kompabilitasnya.

4. Perancangan System

Grid Computing terdiri dari 3 bagian utama yaitu cluster, condor, dan globus. Cluster merupakan lingkungan komputasi paralel sehingga komputer saling berkoordinasi untuk menyelesaikan sebuah job, condor digunakan untuk penjadwalan job sekaligus penghubung antar cluster-cluster yang ada, dan globus merupakan grid engine yang membangun lingkungan grid. Disain sistem dari grid computing adalah seperti pada gambar 1. Dari desain tersebut, terlihat 3 komponen penting yang sudah dijelaskan sebelumnya yaitu cluster, condor (include di grid server), grid (include beserta monitoring). Desain diatas diimplementasikan pada jaringan eepis yang sudah ada sehingga topologi yang terbentuk seperti pada gambar 2. Pada gambar tersebut terlihat VLAN 408 dan VLAN 209 merupakan segmen tempat 2 cluster dibangun. Dan server terletak pada VLAN 85.



Gambar 1. Desain Grid Computing dengan IPv6



Gambar 2. Topologi EEPIS Net dengan Grid IPv6

Berikut adalah rincian dari desain diatas yang diimplementasikan pada jaringan eepis net seperti pada tabel dibawah ini :

Tabel 1. Detail NamaMesin dan Alokasi IP Address

Hostname	IP Address	Keterangan
condormaster.grid-eepis.edu	2001:470:1f05:6bf::209 / 64	Headnode / central manager
slave1.grid-eepis.edu	2001:470:8049:108::11 / 64	CLUSTER A
mpa2.grid-eepis.edu	2001:470:8049:108::12 / 64	
mpa3.grid-eepis.edu	2001:470:8049:108::13 / 64	
mpa4.grid-	2001:470:8049:108	

eepis.edu	::14 / 64	
slave2.grid-eepis.edu	2001:470:8049:209 ::11 / 64	CLUSTER B
mpb2.grid-eepis.edu	2001:470:8049:209 ::12 / 64	
mpb3.grid-eepis.edu	2001:470:8049:209 ::13 / 64	
mpb4.grid-eepis.edu	2001:470:8049:209 ::14 / 64	

Seluruh aplikasi yang digunakan dalam Infrastruktur dan aplikasi dalam proyek akhir ini, berjalan pada sistem operasi Linux dengan spesifikasi:

- Sistem Operasi : Debian Squeeze
- Cluster :
  - libopenmpi1.3
  - openmpi-common
  - libopenmpi-dev
  - openmpi-bin
  - mpi-default-dev
  - ssh
  - g++
  - gcc
  - gfortran
- Job Scheduler : Condor-7.5.6-1lenny
- Grid Engine :
  - Globus Toolkit 4.0.7
  - PostgreSQL
  - Apache Ant
  - libiodbc

5. Pengujian

Pada bagian ini akan dilakukan pengujian yaitu pengujian secara normal dan pengujian secara tidak normal. Berikut adalah uraian dari pengujian yang akan dilakukan :

5.1 Pengujian Normal

Pengujian ini dilakukan sesuai dengan prosedur yang sesuai yaitu file yang digunakan berekstensi c dan mengandung pustaka MPI. Pengujian ini akan menghasilkan output sesuai dengan file dan parameter yang diinputkan.

5.1.1 MPI

Pada pengujian MPI ini akan diuji apakah komputasi parallel mampu mereduksi waktu dalam proses penyelesaian jobs. Proses pengujian dilakukan dengan cara melakukan perhitungan jumlah bilangan prima antara 2 sampai 500000 dan perkalian matrix 1000 x 1000. Dari proses ini akan didapat perbandingan antara komputasi serial dengan komputasi parallel. Berikut adalah proses pengujian bilangan prima pada lingkungan mpi dengan perintah mpicc dan mpirun :

$mpicc prime_sum.c -o prime_sum

$mpirun -np 1 prime_sum

Keluaran dari perintah diatas untuk komputasi serial adalah seperti gambar 3 dibawah ini :

```
condor@condormaster:~/jobs$ mpirun -np 1 prime_sum

PRIME_SUM - Master process:
  C/MPI version
  Add up the prime numbers from 2 to 500000.
  The number of processes available is 1.

  Process 0, N_LO = 2   N_HI = 500000   Total = 1324301603   Time = 48.522960

  The total sum is 1324301603
```

Gambar 3. Output Pengujian Komputasi Serial

Dari gambar diatas, terlihat bahwa proses yang dihasilkan hanya 1 yang merupakan hasil dari pengerjaan komputasi serial dan waktu yang dibutuhkan 48.52 detik

$mpirun -np 4 --host slave1,slave2,mpa2,mpb2 prime_sum

Keluaran dari perintah diatas untuk komputasi parallel adalah seperti Gambar 4 dibawah ini :

```
condor@condormaster:~/jobs$ mpirun -np 4 --host slave1,slave2,mpa2,mpb2 prime_sum

PRIME_SUM - Master process:
  C/MPI version
  Add up the prime numbers from 2 to 500000.
  The number of processes available is 4.

  Process 0, N_LO = 2   N_HI = 125000   Total = 695396285   Time = 3.403582
  Process 1, N_LO = 125001   N_HI = 250000   Total = 1927634866   Time = 9.410022
  Process 2, N_LO = 250001   N_HI = 375000   Total = -1217095014   Time = 15.041990
  Process 3, N_LO = 375001   N_HI = 500000   Total = -81634534   Time = 20.583992

  The total sum is 1324301603
```

Gambar 4. Output Pengujian Komputasi Parallel

Dari gambar diatas, terlihat bahwa proses yang dihasilkan ada 4 yang didistribusikan pada 4 mesin yang ditunjuk sehingga pengerjaannya secara parallel dan waktu yang dibutuhkan lebih cepat yaitu 20.58 detik.

Berikut adalah proses pengujian bilangan prima pada lingkungan mpi dengan perintah mpicc dan mpirun :

$mpicc matrik_parallel.c -o matrik_parallel

$mpirun -np 1 matrik_parallel

Keluaran dari perintah diatas untuk komputasi serial adalah seperti gambar 5 dibawah ini :

```
condor@condormaster:~/jobs$ mpirun -np 1 matrik_parallel

Mulai menghitung perkalian Matrix Paralel ukuran = 1000 x 1000 dimensi

                ... sedang proses ...


 Selesai.
 Total Waktu menggunakan [ 1] prosesor : [16.682694] seconds.
```

Gambar 5. Output Pengujian Komputasi Serial

Dari gambar diatas, terlihat bahwa jumlah prosesor hanya 1 yang merupakan hasil dari pengerjaan komputasi serial dan waktu yang dibutuhkan 16.68 detik.

$ mpirun -np 4 --host slave1,slave2,mpa2,mpb2 matrik_parallel

Keluaran dari perintah diatas untuk komputasi parallel adalah seperti Gambar 6 dibawah ini :

```
condor@condormaster:~/jobs$ mpirun -np 4 --host slave1,slave2,mpa2,mpb2
 matrik_parallel

Mulai menghitung perkalian Matrix Paralel ukuran = 1000 x 1000 dimensi

                ... sedang proses ...



 Selesai.
 Total Waktu menggunakan [ 4] prosesor : [8.853404] seconds.
```

Gambar 6. Output Pengujian Komputasi Parallel

Dari gambar diatas, terlihat bahwa jumlah prosesor ada 4 dimana setiap mesin melakukan perhitungan juga sehingga pengerjaannya secara parallel dan waktu yang dibutuhkan lebih cepat yaitu 8.85 detik.

Dari 2 percobaan diatas, komputasi parallel mampu mereduksi waktu yang dibutuhkan untuk menyelesaikan jobs hingga lebih dari 5 kali dibandingkan komputasi serial.

5.1.2 MPI dan Globus

Pengujian pada lingkungan ini tidak bisa dilakukan karena integrasi antara openmpi dan globus gagal. Beberapa file dari openmpi tidak terbaca meskipun sudah lengkap. Berikut adalah gambar dari hasil pengujian.

```
[condormaster.grid-eepis.edu:08407] [[INVALID],INVALID] ORTE_ERROR_LOG: Not found
in file ../../../../../../orte/mca/ess/hnp/ess_hnp_module.c at line 161
--------------------------------------------------------------------------
It looks like orte_init failed for some reason; your parallel process is
likely to abort.  There are many reasons that a parallel process can
fail during orte_init; some of which are due to configuration or
environment problems.  This failure appears to be an internal failure;
here's some additional information (which may only be relevant to an
Open MPI developer):

  orte_plm_base_select failed
  --> Returned value Not found (-13) instead of ORTE_SUCCESS
--------------------------------------------------------------------------
[condormaster.grid-eepis.edu:08407] [[INVALID],INVALID] ORTE_ERROR_LOG: Not found
in file ../../../orte/runtime/orte_init.c at line 132
--------------------------------------------------------------------------
It looks like orte_init failed for some reason; your parallel process is
likely to abort.  There are many reasons that a parallel process can
fail during orte_init; some of which are due to configuration or
environment problems.  This failure appears to be an internal failure;
here's some additional information (which may only be relevant to an
Open MPI developer):

  orte_ess_set_name failed
  --> Returned value Not found (-13) instead of ORTE_SUCCESS
--------------------------------------------------------------------------
[condormaster.grid-eepis.edu:08407] [[INVALID],INVALID] ORTE_ERROR_LOG: Not found
in file ../../../../../../orte/tools/orterun/orterun.c at line 543
```

Gambar 6. Output Pengujian MPI dan Globus

Dari gambar diatas terlihat beberapa file dari openmpi tidak ada. Padahal file yang dimiliki sudah lengkap. Error diatas menunjukkan jika openmpi dan globus tidak cocok diintegrasikan.

5.1.3 Portal PHP

Portal PHP ini dibuat untuk menggantikan peran dari globus dalam proses administrasi jobs. Proses pengujian portal ini dilakukan dengan cara melakukan perhitungan jumlah bilangan prima antara 2 sampai 500000 dan perkalian matrik. Dari proses ini akan didapat perbandingan antara komputasi serial dengan komputasi parallel. Berikut adalah proses submit job melalui portal PHP.

Output dari hasil eksekusi pada komputasi serial bilangan prima seperti gambar 7 dibawah ini :

```
Hasil Eksekusi :

PRIME_SUM - Master process:
C/MPI version
Add up the prime numbers from 2 to 500000.
Compiled on Jul 19 2011 at 21:15:01.
The number of processes available is 1.

Process 0, N_LO = 2 N_HI = 500000 Total = 1324301603 Time = 48.357914

The total sum is 1324301603
```

Gambar 7. Hasil Eksekusi pada Komputer Serial

Output dari hasil eksekusi pada komputasi parallel bilangan prima seperti gambar 8 dibawah ini:

```
Hasil Eksekusi :

PRIME_SUM - Master process:
C/MPI version
Add up the prime numbers from 2 to 500000.
Compiled on Jul 19 2011 at 21:17:04.
The number of processes available is 4.

Process 0, N_LO = 2 N_HI = 125000 Total = 695396285 Time = 3.408305
Process 1, N_LO = 125001 N_HI = 250000 Total = 1927634866 Time = 9.404242
Process 2, N_LO = 250001 N_HI = 375000 Total = -1217095014 Time = 15.042698
Process 3, N_LO = 375001 N_HI = 500000 Total = -81634534 Time = 20.563483

The total sum is 1324301603
```

Gambar 8. Hasil Eksekusi pada Komputer Parallel

Output dari hasil eksekusi pada komputasi serial perkalian matrik seperti gambar 8 dibawah ini :

```
Hasil Eksekusi :

Mulai menghitung perkalian Matrix Paralel ukuran = 1000 x 1000 dimensi

... sedang proses ...


Selesai.
Total Waktu menggunakan [ 1] prosesor : [16.853002] seconds.
```

Gambar 8. Hasil Eksekusi pada Komputer Serial

Output dari hasil eksekusi pada komputasi parallel perkalian matrik seperti gambar 9 dibawah ini :

```
Hasil Eksekusi :

Mulai menghitung perkalian Matrix Paralel ukuran = 1000 x 1000 dimensi

... sedang proses ...


Selesai.
Total Waktu menggunakan [ 4] prosesor : [8.448449] seconds.
```

Gambar 9. Hasil Eksekusi pada Komputer Parallel

Dari pengujian diatas, terlihat bahwa portal php mampu berperan sebagai globus untuk menjalankan aplikasi prime sum dan perkalian matrik untuk diteruskan ke node dari cluster yang ditunjuk.

5.2 Pengujian Tidak Normal

Terdapat beberapa penanganan kesalahan input oleh user yang sudah diterapkan dalam aplikasi ini dan prosedur yang tidak tepat yang mengakibatkan Portal PHP tidak berjalan dengan sesuai. Berikut adalah beberapa kesalahan yang dilakukan oleh user :

5.2.1 Nilai Iterasi Kosong

Untuk menjalankan jobs melalui Portal PHP, user harus mengisikan berapa nilai iterasi. Dari nilai yang dimasukkan, akan menentukan berapa banyak jobs akan dipecah. Sehingga akan berpengaruh pada penggunaan mesin dan waktu yang dihasilkan.

5.2.2 File tidak berekstensi C dan tidak menggunakan pustaka MPI

File Jobs yang akan di olah harus menggunakan pustaka MPI karena akan menetukan program tersebut bisa dikerjakan secara serial atau parallel. Selain harus menggunakan pustaka MPI, file harus berekstensikan C. Jika 2 syarat diatas tidak terpenuhi, maka jobs tidak akan dikerjakan atau tidak bisa mendapatkan hasil yang diinginkan.

6. Analisa

Hasil dari uji coba submit job diatas belum memenuhi tujuan dari Implementasi Grid Computing dengan menggunakan pengalamatan IPv6. Berikut adalah beberapa analisa yang didapatkan adalah :

1. Implementasi IPv6 pada openmpi telah berhasil dilakukan dan mampu membentuk komputasi parallel.
2. Portal PHP bisa menggantikan peran globus untuk menjalankan aplikasi prime sum dan perkalian matrik.
3. Condor belum bisa diimplementasikan karena versi 7.6.1 yang sekarang rilis stabil belum support IPv6. Sedangkan versi yang support IPv6 adalah 7.7.x akan mulai dikembangkan pada musim panas. Disarankan untuk menggunakan paket job scheduler lain seperti oracle grid engine, dll.
4. MPICH2 versi 1.3 masih terdapat bug sehingga tidak bisa digunakan untuk membentuk komputasi parallel dan sebagai alternatif pengganti openmpi.
5. Integrasi Globus dan openmpi tidak bisa diimplementasikan sehingga tidak terbentuk lingkungan grid.

7. Kesimpulan

Berdasarkan hasil pengujian dan analisa yang telah di bahas pada bab sebelumnya maka dapat diberikan beberapa kesimpulan sebagai berikut :

1. Implementasi Grid Computing menggunakan pengalamatan IPv6 belum berhasil karena beberapa faktor yang sudah disebutkan pada bab sebelumnya. Sedangkan Implementasi Grid Computing menggunakan pengalamatan IPv4 berhasil sesuai dengan hasil yang didapat dari bab IV.
2. Pada Implementasi menggunakan IPv6, penggunaan globus digantikan oleh web php yang digunakan untuk administrasi job berbasis web.
3. Disain grid yang terdiri atas banyak cluster mempunyai kelebihan dalam hal biaya serta kemudahan dalam penambahan ataupun pengurangan node-node eksekusinya.
4. Eksekusi program di lingkungan yang berbeda memberikan hasil seperti yang diharapkan, sehingga penambahan cluster ataupun node-node eksekusi pada lingkungan komputasi bisa dilakukan tanpa mengganggu jalannya proses eksekusi.
5. Globus toolkit dapat diandalkan, memberikan lingkungan yang aman, serta bagus pada sisi layanan dan dukungan terhadap local job scheduler, juga adaptif merespon kebutuhan pengembangan sistem aplikasi grid yang bersifat open source.

8. Daftar Pustaka

[1] Komputasi Grid, Wikipedia Bahasa Indonesia, ensiklopedia bebas. From : http://id.wikipedia.org/wiki/Komputasi_grid, (diakses 9 Desember 2010)

[2] IPv6 Forum :: Driving IPv6 Deployment. From : http://www.IPv6forum.com/, (diakses 12 Desember 2010)

[3] Intro to Parallel Computing, E. Van Den Berg. From : http://www.nus.edu.sg/ACES/seminars/2001/ewout/index.html, (diakses 20 Desember 2010)

[4] Rusadi, Fajran Imam, Nazief, Ph.D, Bobby (2006). Evaluasi Lingkungan Pemrograman Paralel Berbasis Message Passing Interface dalam infrastruktur Komputasi Grid di Universitas Indonesia, Fasilkom UI, Jakarta

[5] Computer Cluster, Wikipedia English International. From : http://en.wikipedia.org/wiki/Computer_cluster, (diakses 23 Desember 2010)

[6] Condor Mailing Lists. From : http://www.cs.wisc.edu/condor/mail-lists/

[7] GT 4 Admin Guide. From : http://globus.org/toolkit/docs/4.0/admin/docbook/index.html, (diakses 12 Januari 2011)

[8] Qomari A.W, M. Nur, Winarno SST,M.Kom, Idris (2009). Studi, Disain, dan Implementasi Lingkungan Komputasi Grid di Kampus PENS Menggunakan Globus, Teknik Informatika, Surabaya