PERANCANGAN ALGORITMA *LOAD BALANCING* PADA TOPOLOGI *DYNAMIC TREE* JARINGAN *GRID COMPUTING*

Irfan Darmawan[(1)], Kuspriyanto[(2)], Yoga Priyana[(3)]
[(1)] Teknik Elektro dan Informatika, Universitas Siliwangi,
[(2)(3)] Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Email: dirfand@yahoo.com

ABSTRAK
Beban kerja dan manajemen infrastruktur merupakan fungsi utama yang diperlukan dalam suatu layanan infrastruktur komputasi grid. Dalam meningkatkan throughput infrastruktur grid, beban kerja (workload) suatu infrastruktur dalam suatu jaringan perlu diperhatikan. Melihat perubahan pada topologi secara dynamic yaitu dengan adanya penambahan atau pengurangan infrastruktur. Untuk merealisasikan tujuan di atas, strategi dan algorima load balancing telah direalisasikan. Beberapa strategi yang telah dibuat dengan mengasusikan sekumpulan infrastruktur yang homogen yang dihubungkan dengan jaringan homogen. Dalam komputasi grid harus diperhatikan masalah-masalah heterogeneity, scalability and adaptability yang merupakan persoalan utama dalam suatu proses penentuan beban kerja suatu infrastruktur. Topologi dynamic adalah topologi yang terjadi karena adanya perubahan pada struktur arsitektur jaringan yang diakibatkan penambahan dan pengurangan infrastruktur, yang akan mempengaruhi terhadap routing data. Pada paper ini, akan dibangun suatu algoritma beban kerja pada komputasi grid. Didasarkan pada model topologi tree, algoritma yang akan dibangun memiliki cirri-ciri: (i) topologi tree berlapis, (ii) mendukung heterogenitas dan scalable, dan (iii) secara umum tidak tergantung pada arsitektur grid pada umumnya.

Kata kunci : Beban Kerja, Komputasi Grid, Dynamic Topologi

1. PENDAHULUAN

Grid Computing adalah infrastruktur komputasi yang menyediakan akses berskala besar terhadap sumber daya komputasi yang tersebar secara secara geografis namun saling terhubung menjadi satu kesatuan fasilitas. Sumber daya ini termasuk antara lain supercomputer, sistem penyimpanan, sumber-sumber data, dan instrument- instrument.

Secara umum, elemen-elemen dari infrastruktur Grid adalah

- Hardware/Sumber daya (Dibuat tersedia dari site-site berbeda yang terdistribusi secara geografis, mencakup CPU/Storage/Instruments, dll…)
- Software: Sesuatu yang menghubungkan bersama-sama semua sumber daya ini: middleware. Beberapa aplikasi untuk menggunakan sumber daya komputasi yang dibuat tersedia.

Kontribusi utama pada makalah ini difokuskan pada dua kegiatan. Pertama membuat topologi arsitektur grid dengan memfokuskan pada permasalah beban kerja. Karakteristik model yang dirancang terdiri dari tiga bagian: (i)secara hirarki untuk mendapatkan beban kerja seminimal mungkin; (ii)model berdasarkan pada penggunaan struktur transformasi grid pada arsitektur tree dengan level yang digunakan sebanyak 4 lapis (*layer*); (iii) Topologi yang dibangun tidak bergantung pada architektur grid lainnya.

Kontribusi yang ke dua adalah membangun suatu algoritma beban kerja pada setiap lapis dengan strategi yang disesuaikan dengan topologi yang dirancang. Strategi yang dibuat megacu pada tiap lapis (*intra-network, intra-cluster*, dan *intra-grid*).

2. LOAD BALANCING

Penyeimbang beban (*Load Balancing*) adalah suatu cara untuk membagi/ menyeimbangkan pekerjaan (workload) antara dua atau lebih komputer/ infrastruktur (*resource*), jaringan computer, CPU, yang terhubung dalam suatu jaringan untuk mendapatkan infrastruktur yang optimal, throughput yang maksimum, dan waktu respon yang kecil. Dengan menggunakan algoritma load balancing diharapkan dapat meningkatkan reliabilitas dan redundancy.

3. GRID ARSITEKTUR

3.1. Topologi Grid

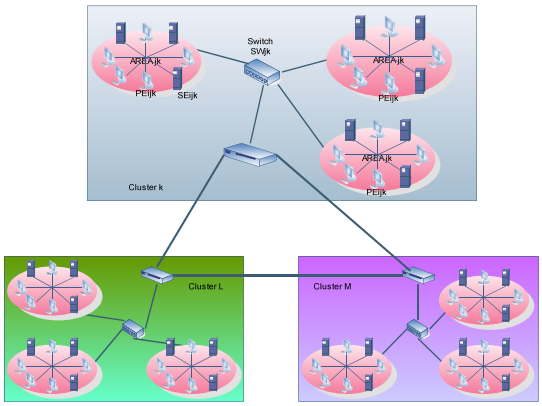
Suatu set G yang terdiri dari cluster (C_k) yang dihubungkan oleh gate (GT_k), dimana k Є {0,…,G-1} , dimana setiap cluster terdiri dari sejumlah Network (N_{jk}) yang dihubungkan oleh Switch (SW_{jk}) dan setiap area terdiri dari beberapa Processing Element (PE_{ijk}) yang terhubung dalam suatu local domain seperti LAN (*Local Area Network*). Tampak pada gambar 1. merupakan topologi grid.

3.2. Spesifikasi pada Komputasi Grid

Pekerjaan utama dalam analisis algoritma beban kerja yang akan dibuat adalah dengan mengasumsikan keseragaman set/node yang

 ISSN: 1907-5022

terhubung pada suatu jaringan yang homogen dan cepat (*fast networks*) [3].



Gambar 1. Topologi Grid

Jika asumsi ini dapat digunakan dalam suatu sistem terdistribusi biasanya, sedangkan dalam arsitektur grid tidak dapat dilakukan, dikarenakan karakeristik suatu grid adalah sebagai berikut [3]:

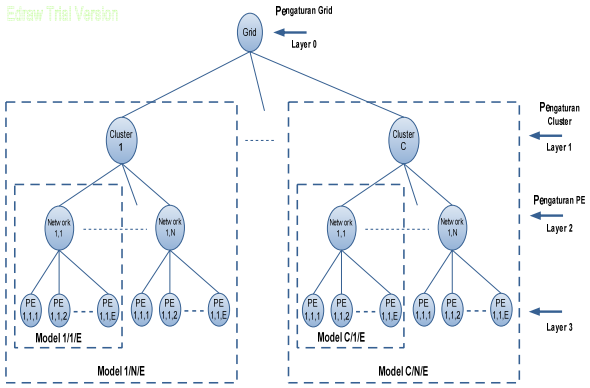
- *Heterogeneity*: Grid yang terdiri dari berbagai resource dengan sifat yang berbeda-beda baik hardware maupun software serta domain administrasi
- *Scalability*: Grid diharapkan dapat dikembangkan dari sumberdaya yang sedikit sampai yang besar
- *Adaptability*: Dapat menyesuaikan terhadap lingkungan yang dinamis, dalam suatu grid suatu sumberdaya mengalami kegagalan merupakan ketentuan yang paling penting. Dalam hal ini kemungkinan beberapa sumberdaya mengalami kegagala/kerusakan akan sangat tinggi

3.3. Load Balancing Model

Model yang dirancang merupakan model topologi yang merupakan penggabungan dari beberapa topologi tree. Pertama untuk setiap domain jaringan dibuat dua buah sub-tree. Ujung dari setiap topologi tree adalah *Processing Element* (PE) yang terdapat pada suatu domain jaringan (N), dan root utama adalah virtual node yang menghubungkan antara domain jaringan. Sub-tree yang terhubung dalam suatu domain jaringan merupakan suatu cluster (C) yang merupakan kumpulan dari sub-tree pada lapisan ke tiga. Selanjutnya beberapa sub-tree tadi terhubung pada lapisan ke empat dan disebut dengan *model penyeimbang beban berlapis* (*Load Balancing Layerd Model*).

Model ini dapat disimbolkan dengan C/N/E, dimana C adalah jumlah cluster pada suatu grid, N jumlah network dalam cluster pada arsitektur grid, E adalah jumlah sumberdaya/prosesing element di setiap jaringan/ network.

Model arsitektur ini dapat di transformasikan ke dalam tiga bentuk model : C/N/E, 1/N/E dan 1/1/E, tergantung dari nilai C dan S. Model grid secara keseluruhan merupakan suatu hubungan model graph yang terpisah dimana setiap lapis/layer mempunyai fungsi yang berbeda.



Gambar 2. *Beban kerja Layerd Model* pada topologi Grid, dengan ketiga jenis varian: 1/1/E, 1/N/E dan C/N/E

- ***Layer 0***: Pada lapisan pertama ini merupakan lapisan pertama (lapisan paing atas), dirancang suatu virtual node yang digunakan sebagai root pada topologi tree. Node ini menghubungkan dalam suatu jaringan grid dan mempunyai dua buah fungsi utama: (i) mengatur informasi beban kerja pada jaringan grid; (ii) menentukan, pada saat menerima tugas/pekerjaan dari user, dimana tugas dapat di sebarkan untuk diproses tergantung pada konektifitas beban kerja jaringan grid.
- ***Layer 1***: Pada lapisan ini G merupakan virtual node, yang akan menghubungkan seluruh cluster pada grid, dalam strategi beban kerja, node ini mempunyai tanggung jawab untuk mengatur beban dari setiap jaringan
- ***Layer 2***: lapisan ketiga ini terdiri dari node-node N yang menghubungkan antara jaringan dalam suatu cluster pada arsitektur grid In this third level, we find *S* nodes associated. Fungsi utama node ini adalah untuk mengatur beban *processing element*.
- ***Layer 3***: Lapisan terakhir merupakan lapisan yang paling ujung, kita tentukan M buah Processing Element pada suatu jaringan yang terdapat dalam tiap cluster yang terhubung dengan lingkungan grid.

3.4. Karakteristik Model Topologi

Karakteristik beban kerja topologi grid yang akan dibahas adalah sebagai berikut:

1) Bersifat hirarki: karakteristik ini digunakan untuk memberikan rute aliran informasi dan strategi untuk menentukan pesan jalur beban kerja yang baik dalam topologi yang dirancang.

 ISSN: 1907-5022

2) Mendukung *heterogeneity* and *scalability* pada lingkunagan Grid: dengan adanya penambahan dan pengurangan/pelepasan (elemen pemroses, domain network, atau cluster) akan sangat mudah dalam system yang dirancang (penambahan atau pengurangan node atau sub-tree)
3) Tidak bergantung pada arsitektur grid lainnya; Perubahan arsitektur pada suatu grid ke dalam bentuk topologi tree merupakan perubahan yang univocal. Setiap grid akan saling berkomunikasi dengan satu atau lebih cluster.

4. PERANCANGAN DYNAMIC TOPOLOGI

4.1. Graph Topologi

Algoritma penyeimbang beban dapat didefinisikan dengan meperhatikan ketentuan sebagai berikut [6]:

- Information policy: mengumpulkan spesifikasi informasi apa yang menentukan beban lebih dan dari mana informasi tersebut didapat.
- Triggering policy: menentukan waktu yang tepat untuk memulai operasi Load Balancing.
- Resource type policy: mengelompokkan sumber daya yang digunakan sebagai server atau receiver dari setiap tugas (task) berdasarkan status dari sumber yang ada.
- Location policy: penggunaan hasil dari setiap jenis aturan sumber daya untuk mencari hubungan yang baik antara server atau receiver.
- Selection policy: menentukan tugas (task) yang harus dipindahkan dari sumber yang memiliki beban lebih (source) ke sumber yang masih diam/idle (receiver).

5. PENERAPAN LOAD BALANCING

5.1. Strategi Pengaturan

Sesuai dengan model topologi yang telah dijelaskan, strategi beban kerja bersifat hirarki. Dalam penerapan algoritma beban kerja di atas, node utama merupakan pengatur bagi subtree yang ada di bawahnya. Dalam perancangan ini beban kerja dibagi tiga macam, yaitu:

1) Beban kerja antar jaringan (*Intra-network/domain*):
   Pada lapisan pertama, routing tergantung pada banyaknya PE dan topologi jaringan yang digunakan.
2) Beban kerja antar cluster (*Intra-cluster*):
   Pada lapisan kedua, beban kerja hanya difokuskan pada setiap cluster, Ck. Dimana jaringan local merupakan node dari jaringan tree yang terdiri dari domain jaringan.
3) Beban kerja antar grid (*Intra-grid*):
   Beban kerja pada lapisan ini hanya digunakan untuk mendeteksi perubahan jaringan setiap cluster yang terhubung ke dalam jaringan grid

Tujuan utama dari strategi yang dirancang yaitu dengan diutamakannya beban kerja antara jaringan lokal (antar jaringan, antar cluster, dan yang terakhir beban kerja untuk keseluruhan jaringan grid)

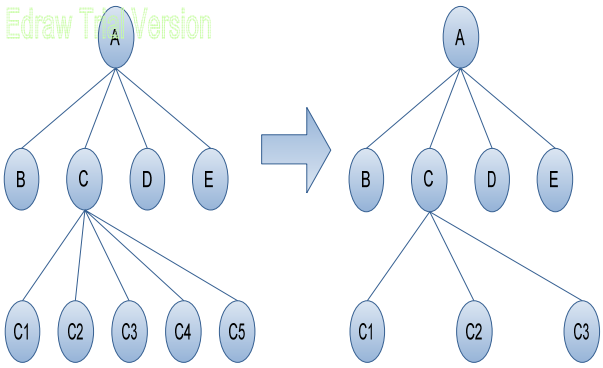
5.2. Algoritma

Dari strategi yang telah di jelaskan, algoritma beban kerja didefinikan dalam tiga lapis: *intra-site*, *intra-cluster* and *intra-grid*.

1) Notasi: symbol-simbol yang digunakan dalam algoritma adalah sebagai berikut :

Ck = kth banyaknya kluster; *Njk = jth* banyaknya network dalam *kth* kluster; *PEijk = ith* banyaknya element pemroses (processing element) yang terdapat dalam *jth* network dalam *kth* kluster; T=factor toleransi (%); *Lijk*= nilai beban kerja dari setiap *PEijk*; *Ljk*= nilai beban kerja dari setiap *Njk*; *Lk*= nilai beban kerja dari setiap *Ck*; *Avrg* = rata-rata beban kerja jaringan Grid; *net-max* = parameter untuk menentukan proses load balancing dalam suatu kluster; *clu-max* = parameter untuk menentukan proses load balancing antar kluster.

2) Algoritma loadbalancing intra-network:
Algoritma ini berfungsi jika pengaturan PE menemukan sesuatu yang tidak beban yang tidak seimbang antara prosesor elemen yang berada dalam lingkungannya. Untuk membuat informasi tersebut, manager akan menerima secara berkala/periodik informasi beban lebih dalam suatu network dari setiap prosesing elemen.
Analisa akan menentukan proses penyeimbangan beban diantara local PE pada suatu network, atau akan memberikan informasi kepada manager network (c*luster*) bahwa network mengalami beban yang sangat berat (*overload*).



a. Menambahkan node b.Menghapus Node

Gambar 5. Graph Penambahan dan Penghapusan Node Dalam Dynamic Topologi

```
BEGIN
For setiap PEijk AND Periodik do
   -Merubah aktual workload Lijk dari
setiap PEijk
   -Mengirim informasi ke PE's manager Njk
end For
- Update aktual load Ljk of site Njk
- Kirimkan informasi beban ke network
manager Ck
- Menerima rata-rata beban dari Ck
If ( |Ljk-Avrg|Ljk > T) then
   imbalance state
else
   kembali
end If
[Mengelompokkan PE dalam Njk menjadi
overloaded, underloaded and balanced]

CES ← Ø; CER← Ø; CEN ← Ø

For setiap PEijk of Njk do
   Switch
    - Lijk > Avrg + T :
    CES ← CES ∪ {CEijk} - Lijk < Avrg:
    CER ← CER ∪ {CEijk} - Avrg ≤ Lijk ≤
Avrg + T:
    CEN ← CEN ∪ {ECijk}
   end Switch
end For

While (CES _= Ø AND CER _= Ø) do

   - Mengurutkan nilai CES dari yang
terbesar Lijk
   - Mengurutkan nilai CER dari yang
terkecilLijk
   - PEsjk ← PE most overloaded;
   - ECrjk ← PE most lightly overloaded
   - Load offered ECrjk = Avrg – Lrjk
   - Memindahkan Tasks dari PEsjk ke PErjk
   - Update current workload of PEsjk,
PErjk
   - Update niliai CES , CER dan CEN
done
If (card(CES) ≥ sit-max) then
   -Intra-Site load balancing gagal
   -Memanggil algoritma intra-cluster load
balancing
else
   load balancing berhasil
end If
END.
```

Gambar 6. Algoritma loadbalancing intra-network

3) Algoritma loadbalancing intra-cluster: dalam algoritma ini digunakan untuk : a) mengatur beban kerja yang terjadi antara PE (*Processing Element*) dalam suatu jaringan local. b) Mengetahui nilai beban kerja suatu network, dimana network manager dapat mendistribusikan beban kerjanya ke network yang lainnya.

```
BEGIN
For setiap network Njk dari Ck AND Periodik
do
Merubah current workload Ljk dan
megirimkannya ke network manager Ck
end For
   - Merubah actual load Lk dalam Ck
   - Mengirim it to grid manager
   - Menerima beban rata-rata grid dari
grid manager
If (Overloaded sites number ≥ sit-max) then
   Ck pada posisi imbalance state
else
   Kembali
end If
[Mengelompokkan network menjadi Ck
overloaded, underloaded, dan balance
network]

   SS ← Ø; SR ← Ø; SN ←Ø

For setiap Njk dari Ck do
Switch
   -Ljk > Avrg + T:
   SS ← SS ∪ {Sjk} -Ljk < Avrg :
   SR ← SR ∪ {Sjk} -Avrg ≤ Ljk ≤ Avrg + T:
   SN ← SN ∪ {Sjk}
 end Switch
end For

While (SS _= Ø AND SR _= Ø) do

  - Mengurutkan nilai SS dari beban yang
   terbesar Ljk
  - Mengurutkan nilai SR dari beban yang
terkecil Ljk
  - Slk ← overloaded network ;
  -Srk ← underloaded network
  - Load offered by Srk = Avrg – Lrk
  - Memindahkan Tasks dariSlk ke Srk
  - Kerjakan algorithm Intra-networkload
balancing
done

If (card(SS) ≥ clu-max) then
   - Load balancing Intra-Cluster gagal
   -panggil algoritma load balancing intra-
grid
else
   prosess load balancing selesai
end If
END.
```

Gambar 7. Algoritma loadbalancing *intra-cluster*

4) Algoritma loadbalancing intra-grid: algoritma yang ke tiga merupakan algoritma untuk menentukan infrastruktur yang mempunyai beban lebih yang dilakukan oleh setiap kluster dalam grid arsitektur.

```
BEGIN
For setiap Ck AND Periodik do
   - Merubah nilai workload Lk
   - Mengirim nilai ke grid manager
end For
   - Merubah global grid workload
   - Membandingkan rata-rata beban grid
   - Mengirim informasi beban ke seluruh
kluster
If (overloaded clusters number ≥ clu-max)
then
   Grid tidak seimbang / imbalance
else
   Kembali
end If
[Mengelompokkan Grid menjadi  overloaded,
underloaded dan balanced clusters]


CS ← Ø; CR ← Ø; CN ← Ø


For Every Ck do
Switch
   Lk > Avrg + T :
   CS ← CS ∪ {Ck} Lk < Avrg :
   CR ← CR ∪ {Ck} Avrg ≤ Lk ≤ Avrg+T:
   CN ← CN ∪{Ck}
 end Switch
end For


While (CS _= Ø AND CR _= Ø) do


  - Mengurutkan nilai CS dari beban yang
terbesar Lk
  - Mengurutkan nilai CR dari beban yang
terbesar Lk
  - Cs ← overloaded cluster
  - Cr ← underloaded
  - Mengirimkan Tasks dariCs ke Cr
  - Menjalankan algoritma intra-Cluster load
balancing
done
END.
```

Gambar 8. Algoritma loadbalancing *intra-grid*

6. ANALISIS

6.1. Modelling Parameters

Untuk mengevaluasi algoritma yang dirancang, model algoritma tersebut dibuat dalam simulator grid. Simulator dibangun dengan menggunakan Java dengan parameter-parameter sebagai berikut

1) ***PE's parameters***: parameter yang memberikan informasi tentang adanya PE dalam proses *load balancing*, seperti (i) jumlah network; (ii) jumlah PE dari setiap network; (iii) Kecepatan PE; (iv) waktu untuk mengirimkan informasi kelebihan beban dari seiap PE;(v) factor toleransi

2) ***Tasks parameters***: parameter ini terdiri dari: (i) banyaknya jumlah pekerjaan (task) dari setiap PE; (ii) jadwal task yang diberikan; (iii) jumlah instruksi setiap task; (iv) lebar task; dan (v) prioritas.

3) ***Network parameter***: lebar bandwidth.

4) **Indek beban lebih (*Workload)***: suatu PE dikatakan mempunyai kelebihan beban dengan melihat ratio pekerjaan, *workload=instspeed*, dimana inst menunjukkan jumlah instruksi yang menunggu/antri yang terdapat dalam setiap PE, speed adalah kecepatan akses setiap PE.

5) ***Performance Parameter***: parameter yang diguakan dalam model ini adalah difokuskan pada rata-rata waktu respon task dan biaya komunikasi.

6.2. Hasil Analisa

Tampak pada table 1 dan 2 yang memperlihatkan waktu respon rata-rata sebelum dan sesudah menggunakan algoritma *load balancing*, yang mana performance (%) dan waktu (detik) sangat memberikan pengaruh.

Tabel 1. Waktu Respos terhadap Jumpah PE

Jumlah PE	Waktu Respon (sec)			Cost (sec)
	Sebelum	Sesudah	Gain (%)	
50	817	730	10.65%	41
100	409	353	13.69%	50
150	273	230	15.75%	49
200	126	99	21.43%	47
250	164	139	15.24%	49

Tabel 2. Waktu Respos terhadap Jumlah Task

Jumlah Task	Waktu Respon (sec)			Cost (sec)
	Sebelum	Sesudah	Gain (%)	
1000	113	95	15.93%	22
1250	134	114	14.93%	33
1500	145	122	15.86%	37
1750	156	132	15.38%	47
2000	164	139	15.24%	49

Seperti tampak pada table, dapat disimpulkan bahwa strategi load balancing telah berhasil:

1) Untuk jumlah task sebanyak 2000 dan jumlah PE yang berbeda mulai dari 50 sampai 250, terlihat pencapaian antara 10,65% dan 21,43%, dimana biaya komunikasi dapat diabaikan.

2) Untuk jumlah PE sebanyak 250 dan untuk jumlah task yang berbeda dari 1000 sampai 2000, terlihat pencapaian antara 14,93% dan 15,93%

3) Dari hasil penelitian ini, kami menghendaki bahwa pencapaian hasil terbaik jika grid dalam keadaan stabil (tidak ada yang overload atau sama sekali diam (idle).

7. KESIMPULAN

Dalam makalah ini, telah kita bahas algoritma untuk mengatur beban kerja dalam jaringan komputasi grid. Algoritma ini digunakan dalam komputasi grid dengan beban kerja antara jaringan (intra-network, intra-cluster, intra-grid).

Model algoritma yang dirancang memungkinkan untuk merubah arsitektur tree menjadi suatu topologi tree sebanyak empat lapis (layer). Dari topologi

tersebut dapat dipecah lagi menjadi beberapa sub-model: intra-network, intra-kluster, intra-grid. Dengan adanya model ini, strategi yang dibangun adalah strategi hirarki load balancing yang memberikan prioritas load balancing pada jaringan lokal.

Dengan pembangunan prototype algoritma berlapis yang telah diimplementasikan dan di analisis pada simulator grid dengan kondisi/parameter sebenarnya. Hasil yang didapat dari penelitian ini menunjukkan bahwa model yang dibuat dapat membuat proses load balancing lebih baik antara PE pada grid tanpa memerlukan pengeluaran yang besar.

PUSTAKA

1. I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1998.
2. Rinaldi Munir, "*Diktat Kuliah IF2251: Strategi Algoritmik*", Program Studi Teknik Informatika, STEI ITB, 2006.
3. E. Deelman A.Chervenak and al. *High performance remote access to climate simulation data: a challenge problem for data grid technologies*. In *Proceeding. of 22th parallel computing*, volume 29(10), pages 13–35, 1997.
4. Jose Duato, "*Interconnection Networks: An Engineering Approach*", Morgan Kaufmann Publisher, 2003.
5. F. Berman, G. Fox, and Y. Hey. *Grid Computing: Making the Global Infrastructure a Reality. Wiley Series in Communications Networking & Distributed Systems*, 2003.
6. H.D. Karatza. *Job scheduling in heterogeneous distributed systems*. Journal. of Systems and Software, 56:203–212, 1994.
7. B. Yagoubi. *Dynamic load balancing for beowulf clusters*. In Proceeding of the 2005 International Arab Conference On information Technology, pages 394–401, Israa University, Jordan, December 6th 8th 2005.
8. W. Leinberger, G. Karypis, V. Kumar, and R. Biswas. *Load balancing across near-homogeneous multi-resource servers*. In 9th Heterogeneous Computing Workshop, pages 60–71, 2000.