PROGRAM PENGUKURAN DAYAGUNA KOMPUTER PARALEL TOPOLOGI HYPERCUBE.

Meicsy E. I. Najoan

Abstract

Penelitian ini membahas pengukuran unjuk kerja komputer paralel dengan topologi hypercube. Suatu program pengukuran dibuat kemudian digunakan untuk mengevaluasi guna mengetahui performance dari komputer paralel. Dalam mengevaluasi, program ini meng-simulasi-kan suatu program paralel tertentu yang berjalan dalam sistem paralel dan disimulasikan dengan single prosesor (komputer stand-alone). Kuantitas-kuantitas yang diukur nilainya berupa peningkatan kecepatan (speedup) dan efisiensi. Jumlah prosesor, topologi jaringan, mekanisme pengalihan pesan dan jenis program paralel beserta beban tugasnya merupakan parameter-parameter pengamatan pada program ini.

Data hasil pengukuran kemudian dikonversikan kebentuk grafik, sehingga pengguna bisa mendapatkan ungkapan sederhana dari grafik tersebut dan beberapa sifat yang penting. Program ini dapat menjadi suatu tools untuk pengembangan aplikasi paralel, pembelajaran paralel programming tanpa harus menggunakan sistem yang sebenarnya.

Kata Kunci : Paralel Komputer, Dayaguna, Speedup, Efficiency.

I. Pendahuluan.

Komputer paralel yang merupakan solusi untuk mengerjakan beban atau tugas yang besar. Performance dari komputer paralel tergantung disain keseimbangan hardware dan software.

Untuk mengukur performance komputer paralel sebelum disain, dapat dibuat program pengukuran performance dan kemudian dilakukan evaluasi.

Dalam penelitian ini, akan ditunjukkan beberapa parameter yang dipakai untuk mengukur performance dari komputer paralel. Parameter ini merupakan masukan program pengukuran performance dan kemudian akan menghasilkan keluaran yang akan dievaluasi untuk mendapatkan perilaku dari sistem.

II. Ruang Lingkup Permasalahan.

Adapun ruang lingkup pembahasan sebagai berikut.

- Bentuk topologi jaringan yang diamati adalah hypercube
- Pengamatan objek dilakukan dengan mengevaluasi dari program-program paralel.
- Kuantitas-kuantitas yang diukur nilai-nilainya adalah peningkatan kecepatan dan efisiensi.
- Jumlah node yang dimasukkan terbatas pada panjang variabel yang digunakan. Dalam program pengukuran performance ini sudah ditentukan.

III. Tujuan dan Manfaat Penelitian.

Adapun tujuan penelitian ini membuat program pengukuran performance untuk mengevaluasi dayaguna (*performance*) dari arsitektur message-passing multicomputer, dalam hal ini faktor peningkatan kecepatan (*speedup factor*) dan efisiensi (*efficiency*) serta pengaruh-pengaruhnya penurunan dayaguna.

Manfaat dari penelitian ini, merupakan suatu tools yang bermanfaat, dalam hal ini akan memperlihatkan peningkatan kinerja dari paralel komputer umumnya.

IV. Tinjauan Pustaka.

4. 1. Topologi Jaringan

Topologi menunjuk interkoneksi graph dari jaringan. Puncak (vertices) dari graph adalah node/prosesor dalam jaringan, N, dan sisi/tepi (edges) adalah saluran fisik yang menghubungkan node-node.

Beberapa parameter yang dipakai untuk evaluasi suatu topologi : Bisection Width, Degree, Diameter, Wire Length, dan Symmetry.

4.2. E-cube Routing pada Hypercube

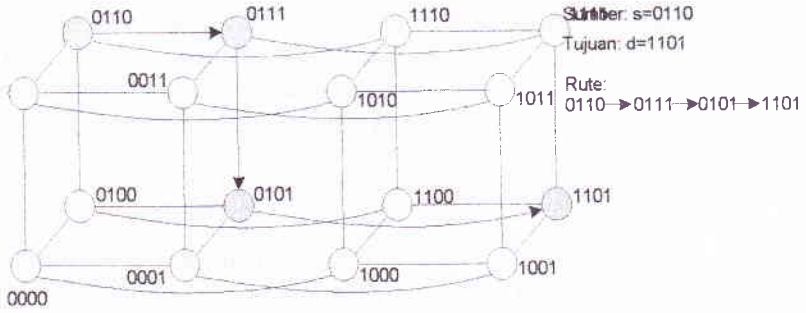
Routing adalah metode yang digunakan untuk memilih jalur terpendek dalam jaringan. Bentuk algoritma pengroutean yang akan

digunakan dalam penelitian ini adalah E-cube Routing pada hypercube.

Dengan menganggap n-cube dengan $N = 2^n$ node. Node sumber adalah $s = s_{n-1} \cdots s_1 s_0$ dan node tujuan adalah $d = d_{n-1} \cdots d_1 d_0$. Tujuan disini adalah untuk menentukan rute dari s ke d dengan jumlah langkah yang sedikit. Bila $v = v_{n-1} \cdots v_1 v_0$ adalah node-node sepanjang rute yang dilalui, maka rute dengan langkah yang sedikit dapat ditentukan :

Hitung bit $r_i = s_{i-1} \oplus d_{i-1}$ untuk semua n dimensi ($i = 1, \ldots, n$). Mulai hitung dimensi $i = 1$ dan $v = s$. Rute dari node yang sedang dilalui v ke node berikut $v \oplus 2^{i-1}$ jika $r_i = 1$. Jika $r_i = 0$, melompat dari langkah ini. Pindah ke dimensi $i + 1$. Jika $i \leq n$, kembali ke langkah 2, jika tidak keluar. Berikut adalah contoh algoritma E-cube routing :

Bila $n = 4$, $s = 0110$, dan $d = 1101$. Maka $r = r_4 r_3 r_2 r_1 = 1011$. Rute dari s ke $s \oplus 2^0 = 0111$ karena $r_1 = 0 \oplus 1 = 1$. Rute dari $v \oplus 2^1 = 0101$ karena $r_2 = 1 \oplus 0 = 1$. Melompat ke dimensi $i = 3$ karena $r_3 = 1 \oplus 1 = 0$. Rute dari $v = 0101$ ke $v \oplus 2^3 = 1101 = d$ karena $r_4 = 1$. Rute yang dipilih dapat dilihat pada gambar 1.



Gambar 1. E-cube routing pada hypercube komputer dengan 16 node.

4.3. Hukum Amdahl's untuk Speedup.

Untuk mengevaluasi performance dari algoritma paralel untuk suatu problem dalam komputer paralel adalah membandingkan waktu algoritma yang dijalankan secara berurut (sequential) dalam satu prosesor/node dengan waktu dari algoritma paralel yang dijalankan di banyak prosesor/node. Perbandingan ini disebut dengan peningkatan kecepatan (speedup).

Beberapa faktor yang menyebabkan penurunan speedup disebabkan karena faktor serial dari program (seperti ketergantungan data, waktu pemuatan program dan penyumbatan/ bottlenecks I/O), biaya tambahan (overhead) sinkronisasi dan komunikasi.

Amdahl's (1967) mempertimbangkan faktor serial dari program dengan hukum Amdahl's sebagai berikut:

$$S_n = \frac{n}{1 + (n-1)\alpha} \qquad (1)$$

dimana : S_n = peningkatan kecepatan (speedup);
n = jumlah prosesor;
α = the fraction of sequential bottleneck

4.4. Speedup facktor & Efisiensi

Beberapa parameter untuk mengevaluasi perhitungan-perhitungan secara paralel yang sering ditemui dalam aplikasi dan juga merupakan konsep dasar dari pemrosesan secara paralel adalah faktor peningkatan kecepatan (speedup factor) dan efisiensi sistem (efficiency system). Untuk faktor peningakatan kecepatan dapat dihitung dari :

$$S(n) = \frac{T(1)}{T(n)} \qquad (2)$$

S(n) adalah faktor peningkatan kecepatan (speedup factor), T(1) adalah waktu eksekusi prosesor untuk satu prosesor, T(n) adalah waktu eksekusi prosesor untuk n-processor.

Efisiensi sistem untuk suatu n – prosesor dihitung dari :

$$E(n) = \frac{S(n)}{n} = \frac{T(1)}{nT(n)} \qquad (3)$$

Efisiensi, E(n) ; suatu indikasi dari derajat sebenarnya dari performance kecepatan yang dicapai dibanding dengan nilai maksimum.

Batasan nilai kedua faktor adalah: $1 \leq S(n) \leq n$, dan $1/n \leq E(n) \leq 1$.

Efisiensi rendah terjadi pada kasus dimana seluruh kode program yang dimasukan

dieksekusi secara berurutan pada satu prosesor. Efisiensi maksimum dicapai ketika semua n-prosesor secara penuh digunakan untuk periode eksekusi.

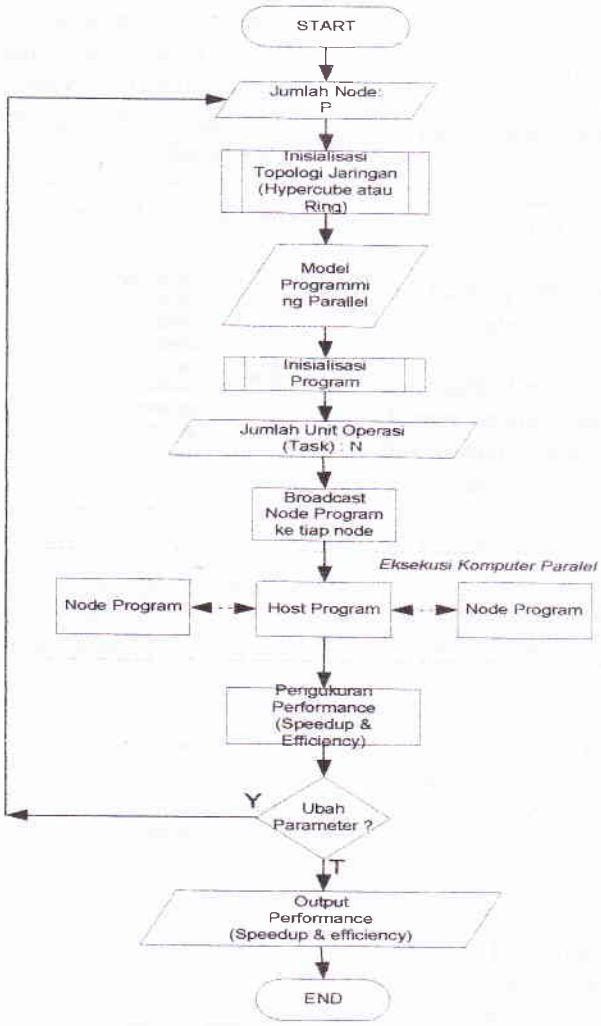
V. Perancangan Program Pengukuran Performance.

Pada pemrograman ini akan menggunakan kemampuan dari C++ Builder yaitu pemrograman berorientasi objek (Object Oriented Programming/ OOP). Beberapa asumsi akan didefinisikan sendiri untuk men-simulasi-kan sistem yang akan dirancang. Misalnya untuk setiap instruksi yang dikenakan pada node program dan waktu penundaan yang terjadi pada saat pengiriman pesan antara node (communication delay) akan didefinisi-kan sendiri clock cycle. Dengan demikian dapat ditentukan waktu eksekusi dari masing-masing prosesor dan waktu penundaan pada jaringan. Variabel global akan diinisialisasikan terlebih dahulu sebagai variabel yang akan digunakan sebagai tempat pelewatan pesan (message-passing).

Adapun bentuk sistem program dari simulator message-passing multi-computer seperti Gambar 2

5.1. Model Program Paralel.

Dalam mengukur performance dari sistem ini, perlu adanya model program paralel. Dalam sistem ini diambil dua model program paralel untuk mengevaluasi keluaran dari sistem. Model program pertama berupa perhitungan terdistribusi (distributed computing) dengan multi-komputer. Model yang kedua adalah program pengurutan (sorting) dalam hal ini pengurutan gabungan (merge-sort). Berikut adalah penjelasan dari kedua model program diatas.



Gambar 2. Sistem Program Pengukuran Performance Komputer Paralel.

5.1.1 Perhitungan Terdistribusi (distributed computing).

Model program paralel ini dijalankan secara bersama-sama untuk perhitungan terdistribusi arsitektur message-passing. Program ini akan mengevaluasi π pada area dibawah kurva f(x) antara 0 dan 1.

Dengan menggunakan aturan rectangle dari bentuk integral diskrit berikut ini :

$$\pi = \int_0^1 f(x)\,dx = \int_0^1 \frac{4}{1+x^2}\,dx = h\sum_{i=1}^{n} f(x_i) \quad (4)$$

dengan h = 1/n dari lebar panel, x_i = h (i – 0.5) dari titik tengah, dan n adalah jumlah panel (rectangle) yang akan dihitung.

Adapun program untuk menghitung daerah diatas dibagi dalam dua bagian :

Host Program	Node Program
Input(n)	p = numnodes()
Send(n,allnodes)	me = mynode()
Recv(Pi)	recv(n)
Output(Pi)	h = 1.0 / n
	sum = 0
	Do i = me + 1, n, p
	x = h x (i – 0.5)
	sum = sum + f(x)
	End Do
	pi = h x sum
	gop('+',Pi,host)

5.1.2. Pengurutan Gabungan (Merge-Sort).

Algoritma pengurutan gabungan dapat dijelaskan sebagai berikut.

Diberikan vector R berisi N record, algoritma pengurutan dari vector dengan urutan menaik adalah dengan berturut-turut memanggil prosedur MERGE_PASS. Sebuah vector C dibutuhkan sebagai vector bantuan dengan ukuran yang sama dengan R. Variabel L menetapkan sejumlah elemen dari masing-masing sub-tabel untuk digabungkan selama suatu pass khusus. Adapun kode programnya adalah :

1. [Melakukan sort]
   for L = 1, 2, 4, …, $2^{\lceil \log_2 N \rceil - 1}$
   if $\log_2$ L adalah genap
   then Call
   MERGE_PASS(R,N,C,L)
   else Call
   MERGE_PASS(C,N,R,L)
2. [Menyalin jika diperlukan]
   if $\lceil \log_2 N \rceil$ adalah ganjil
   then for I = 1, 2, …, N
   R[I] ←C[I]
3. [Selesai] Exit

Jadi ada $\lceil \log_2 n \rceil$ pass yang diperlukan dalam mengurutkan dan total jumlah perbandingan yang diperlukan adalah O(n $\log_2$ n).

VI. Pengukuran Hasil

Pengukuran yang dilakukan untuk melihat performance (dayaguna) sistem yang dirancang. Dari pengukuran ini dilakukan beberapa analisa untuk melihat performance sistem, penyebabnya dan apa yang harus dilakukan untuk meningkatkan performance.

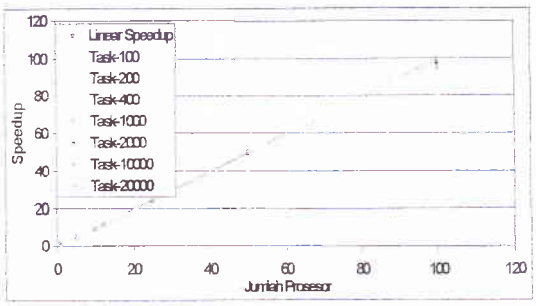
Adapun hal-hal yang diuji adalah speedup factor (faktor peningkatan kecepatan) dan efisiensi. Dua model program yakni perhitungan terdistribusi dan gabungan pengurutan akan digunakan untuk melakukan pengukuran.

6.1. Pengamatan Program Perhitungan Terdistribusi (Distributed Computing).

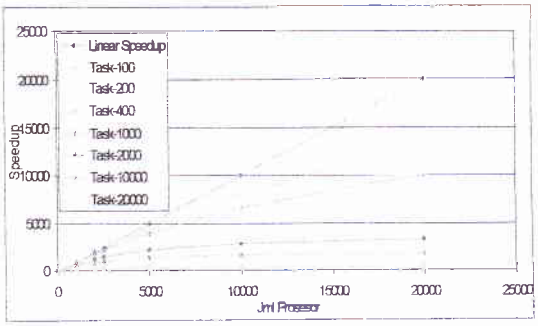
Pengamatan yang akan dilakukan adalah waktu pemrosesan, faktor kecepatan dan efisiensi sistem dengan kecepatan eksekusi prosesor yang berbeda.

Pengamatan dilakukan dengan beban kerja (task) yang berbeda-beda serta penambahan jumlah prosesor pada system yang akan diamati.

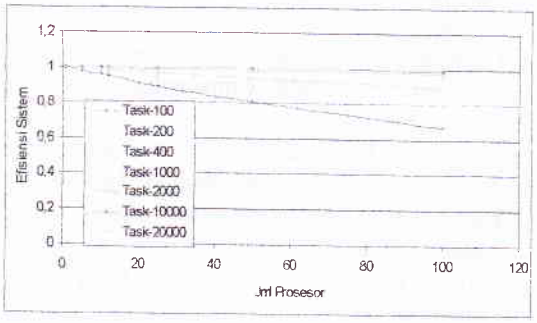
Grafik perbandingan dapat dilihat pada Gambar 3 s/d Gambar 6.



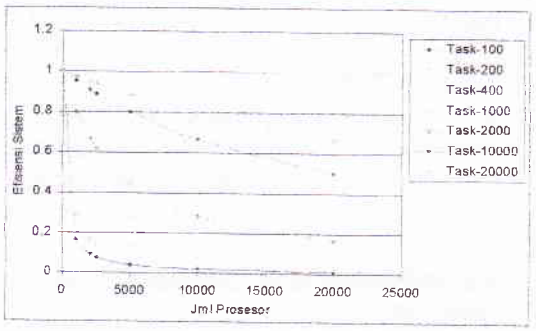
Gambar 3. Speedup vs Jml Prosesor, Program Distributed Computing beban kerja/task berbeda, Jml Prosesor=100.



Gambar 4. Speedup vs Jml Prosesor, Program distributed Computing, beban kerja/task berbeda, Jml Prosesor=20000



Gambar 5. Efisiensi Sistem , Program Distributed Computing, beban kerja/task berbeda, Jml Prosesor=100.



Gambar 6. Efisiensi Sistem , Program Distributed Computing, beban kerja/task berbeda, Jml Prosesor=20000.

6.2. Pengamatan Program Gabungan Pengurutan (Merge-Sort).

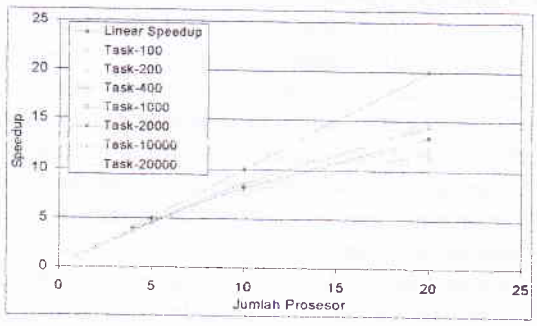
Untuk menyelesaikan gabungan pengurutan ada O(n $\log_2$ n) total jumlah perbandingan. Dengan membagi data menjadi potongan kecil dengan dua record, kemudian kedua record diurutkan secara mengulang tiap potongan kecil, akhirnya menggabungkan data tersebut.

Pengamatan yang akan dilakukan adalah waktu pemrosesan, faktor kecepatan dan efisiensi sistem dengan waktu tunda komunikasi (communication delay) yang berbeda. Untuk mengubah waktu tunda komunikasi akan dikalikan dengan variabel (d). Waktu tunda akan bertambah bila d diperbesar.

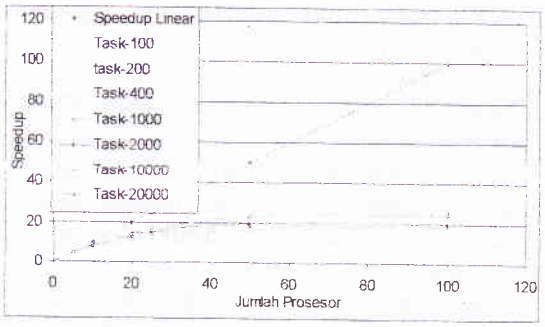
6.2.1. Data Pengamatan - Program Merge-Sort Topologi Hypercube

Beban kerja (task) yang berbeda-beda dengan penambahan jumlah prosesor system yang akan diamati.

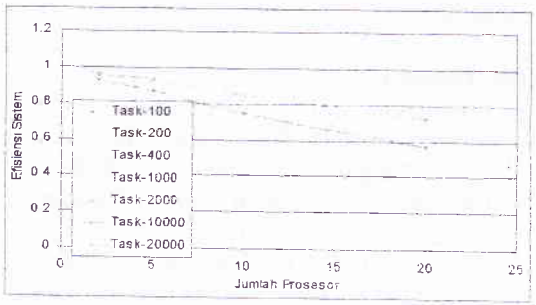
Grafik perbandingan tersebut dapat dilihat di Gambar 7 s/d Gambar 10.



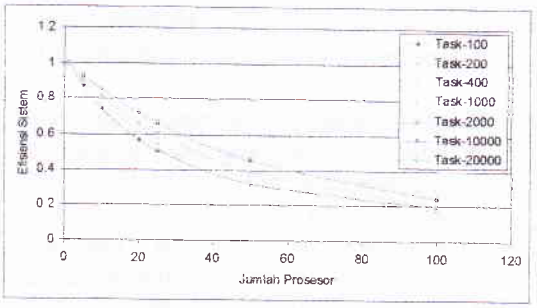
Gambar 7. Speedup vs Jml Prosesor, Program Merge-sort , Topology Hypercube, beban kerja/task berbeda, Jml Prosesor = 20.



Gambar 8. Speedup vs Jml Prosesor, Program Merge-sort , Topology Hypercube, beban kerja/task berbeda, Jml Prosesor = 100.



Gambar 9. Efisiensi Sistem, Program Mergesort, Topologi Hypercube, beban kerja/task berbeda, Jml Prosesor = 20.



Gambar 10. Efisiensi Sistem, Program Merge-sort, Topologi Hypercube, beban kerja/task berbeda, Jml Prosesor = 100.

VII. Penutup / Kesimpulan.

- Program pengukuran performance komputer paralel ini dapat dijadikan disain pendahuluan (preliminary design) untuk merancang arsitektur dari komputer paralel.
- Beban kerja tidak sama pada masing-masing node, sehingga menurunkan faktor speedup.
- Penurunan speedup disebabkan karena ada bagian serialisasi program. Hal ini sesuai dengan hukum Amdahl's yang menyatakan adanya sequential bottleneck dalam program akan menurunkan speedup, dan ini tak dapat diselesaikan hanya dengan menambah jumlah prosesor dalam sistem.
- Peningkatan performance terjadi bila jumlah data atau tugas bertambah. (grafik pengamatan).
- Bila delay jaringan kecil maka faktor kecepatan akan meningkat (mendekati kecepatan linear).
- Bentuk topologi jaringan mempengaruhi delay jaringan. Bila diameter jaringan besar delay pada jaringan akan meningkat sehingga speedup turun.
- Penambahan jumlah data/task akan mengurangi faktor serialisasi dalam eksekusi program.
- Topologi hypercube akan menghasilkan performance yang baik, bila jumlah prosesor diperbanyak.

Daftar Pustaka:

[1] Covington. R.G, Dwarkada, Jump, Sinclair, Madala, The Efficient Simulation of Paralel Computer Systems, International Journal in Computer Simulation, Vol 1, 1991.

[2] Dally William, J. Fiske, R. Lethin, J. Keen,M.D.Noakes, P.R.Nuth, The Message-Driven Processor: A Multi-computer Processing Node with Efficient Mechanisms, Journal IEEE, April 1992.

[3] Dally William, Network and Processor Architecture for Message-Driven Computers, Lab AI-MIT, Massachusetts, 1988

[4] Hwang, Kai, Advanced Computer Architecture Paralelism, Scalability, Programmability, McGraw-Hill, Singapore, 1993

[5] Hwang, Kai. dan Douglas, DeGroot., Paralel Processing for Supercomputers & Artificial Intelligence, McGraw-Hill, Singapore, 1989.

[6] Law, Averill.M. dan Kelton, David.W., Simulation Modeling & Analysis, McGraw-Hill, Industrial Engineering Series, 1982.

[7] Legedza Ulana, Reducing Synchronization Overhead in Paralel Simulation, Master Thesis, MIT, 1995

[8] Mehdi, R. Zargham, Computer Architecture Single and Paralel Systems, Prentice-Hall, Inc., New Jersey, 1996

[9] Miano John, Thomas Cabanski, Harold Howe, Borland C++ Builder How-To, California, 1997.

[10] Raj, Jain, The Art Of Computer Systems Performance Analysis, John Wiley & Sons, Inc, USA, 1991.

[11] Trembaly, Jean-Paul dan Sorenson, Paul.G, An Introduction to Data Structures with Applications, McGraw-Hill, Singapore, 1984

[12] Wu Xingfu, Performance Evaluation, Prediction and Visualization of Paralel Systems, Kluwer Academic Publishers, Boston/ Dordrecht /London, 1999.

[13] http://citeseer.nj.nec.com/dally98jmachine.html

[14] http://citeseer.nj.nec.com