

PEMROSESAN PARALEL MENGGUNAKAN KOMPUTER HETEROGEN

Syarif Hidayat

Jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia

Jl. Kaliurang Km. 14 Yogyakarta 55501

Telp. (0274) 895287 ext. 122, Faks. (0274) 895007 ext. 148

E-mail: syarif@fti.uui.ac.id

ABSTRAKSI

Seiring dengan berkembangnya kemajuan ilmu pengetahuan, jumlah permintaan sumber daya komputer juga semakin meningkat. Beberapa bidang yang membutuhkan komputasi tingkat tinggi tersebut adalah simulasi numerik problem-problem ilmiah dan teknik. Salah satu solusi yang ditawarkan adalah menggunakan pemrosesan paralel. Idennya adalah dengan membagi suatu proses menjadi beberapa bagian untuk kemudian didistribusikan ke dalam beberapa komputer untuk dikerjakan secara simultan. Umumnya, komputer yang digunakan pada pemrosesan paralel adalah komputer yang homogen. Namun pada paper ini akan ditunjukkan pemrosesan paralel menggunakan komputer yang spesifikasinya berbeda. Tujuannya adalah untuk memaksimalkan resource yang telah ada sehingga tidak diperlukan alokasi komputer secara khusus untuk membangun sistem yang akan menjalankan pemrosesan paralel.

Kata kunci: Pemrosesan paralel, sistem paralel, heterogen, MPI, Knoppix

1. PENDAHULUAN

1.1 Latar Belakang Masalah

Seiring dengan kemajuan teknologi maka kebutuhan manusia akan informasi juga akan meningkat. Paradigma baru meramalkan bahwa dunia akan dikendalikan oleh orang-orang yang mempunyai akses informasi paling luas. Adanya World Wide Web membuat Internet yang sebelumnya hanya digunakan oleh kalangan akademisi dan riset menjadi hal penting di dunia bisnis dan media. Hal itu memicu perkembangan Sistem Komputer dan jaringan Internet agar berkembang dengan cepat untuk memenuhi kebutuhan manusia akan informasi. Saat ini hampir seluruh aspek kehidupan dikaitkan dengan sistem komputer dan jaringan internet. Dengan meledaknya perkembangan internet dan semakin perlunya internet dalam kehidupan kita, lalu lintas internet meningkat secara dramatis. Hal ini diindikasikan pula dengan semakin menipisnya persediaan nomor IP sehingga dikembangkan IPv6 yang mampu menyediakan hingga nomor IP. Hal tersebut mengilustrasikan betapa padatnya lalu lintas data yang kelak akan dilakukan sekian banyak nomor IP. Tak terbayangkan beban kerja server yang akan menangani sejumlah besar client terutama web server yang populer.

Selain hal diatas, kemajuan sains di bidang lain juga menuntut adanya sistem komputer yang mampu melakukan perhitungan dengan performa tinggi, baik kecepatan maupun kapasitas perhitungan. Implikasinya, teknologi dari jenis personal komputer hingga Super komputer terus mengalami perkembangan untuk meningkatkan kapasitas dan pengolahan data. Kendala utama superkomputer adalah pada biaya pengadaan, operasional dan pemeliharannya. Selain itu cepatnya perkembangan teknologi di bidang komputer menjadikannya tidak cost-effective.

Munculnya generasi superkomputer yang memiliki kemampuan satu order diatasnya membuat tingkat penyusutan atas nilai investasi awal yang besar menjadi tidak terkendali.

Alternatif yang sedang populer akhir-akhir ini adalah menggunakan pemrosesan paralel yaitu mendistribusikan paket pekerjaan untuk dikerjakan oleh seluruh komputer yang terdapat dalam sistem untuk menyelesaikan satu masalah.

Dengan komputer paralel ini, biaya investasi bisa ditekan sampai titik terendah, selain nilai penyusutan yang juga menjadi jauh lebih rendah. Lebih penting lagi, sistem ini fleksibel terhadap perubahan teknologi komputer yang sangat cepat. Selain itu bisa juga dilakukan optimalisasi perangkat keras sesuai dengan karakteristik penggunaan. Misalnya untuk aplikasi yang mementingkan kecepatan proses penghitungan cukup dengan prosesor dan memori saja tanpa media penyimpan di setiap PC. Sebaliknya untuk aplikasi yang banyak memproduksi data, penekanan diberikan pada perangkat media penyimpanan. Dengan ini bisa dilakukan efisiensi investasi. Karena PC-PC lama masih bisa didayagunakan sesuai dengan kebutuhan. Selain itu dimungkinkan pula koneksi melalui internet untuk memparalelkan komputer pribadi.

1.2 Batasan masalah

Masalah yang akan dibahas dalam makalah ini adalah bagaimana mengaplikasikan pemrosesan paralel menggunakan komputer yang heterogen (spesifikasinya berbeda-beda). Selain itu, sistem juga tidak akan terhubung ke jaringan Internet.

1.3 Tujuan Pembahasan

Pemrosesan paralel meliki tujuan untuk membagi beban kerja dan mendistribusikannya pada komputer-komputer lain yang terdapat dalam

sistem untuk menyelesaikan suatu masalah. Sistem yang akan dibangun akan tidak akan menggunakan komputer yang didedikasikan secara khusus untuk keperluan pemrosesan paralel melainkan menggunakan komputer yang telah ada. Artinya, sistem ini nantinya akan terdiri dari sejumlah komputer dengan spesifikasi berbeda yang akan bekerjasama untuk menyelesaikan suatu masalah.

2. LANDASAN TEORI

Pemrosesan Paralel adalah komputasi dua atau lebih tugas pada waktu bersamaan dengan tujuan untuk mempersingkat waktu penyelesaian tugas-tugas tersebut dengan cara mengoptimalkan resource pada sistem komputer yang ada.[7] Pemrosesan paralel dapat mempersingkat waktu eksekusi suatu program dengan cara membagi suatu program menjadi bagian-bagian yang lebih kecil yang dapat dikerjakan pada masing-masing prosesor secara bersamaan. Suatu program yang dieksekusi oleh n prosesor diharapkan dapat mempersingkat waktu eksekusi n kali lebih cepat. Salah satu komputer yang menggunakan metode pemrosesan paralel adalah SMP (Single Multi Processor). Satu komputer SMP mempunyai beberapa prosesor yang berbagi memori dan antarmuka bus. Konsepnya dijelaskan dibawah ini:

1. Tugas S1 dimulai pada saat t_1 dan berakhir pada waktu b_1
2. Tugas S2 dimulai pada saat t_2 dan berakhir pada waktu b_2
3. Hendak dilakukan pemrosesan paralel antara S1 dan S2 ($S1||S2$) maka digunakan ketentuan sebagai berikut :
 - a. Begins at $\min(t_1, t_2)$, yang berarti bahwa waktu pemrosesan paralel dihitung mulai dari waktu t_1 atau t_2 yang paling awal.
 - b. Ends at $\max(b_1, b_2)$, yang berarti bahwa waktu pemrosesan paralel dihitung sampai waktu b_1 atau b_2 yang paling akhir.
 - c. Pemrosesan paralel $S1||S2$ sama dengan pemrosesan paralel $S2||S1$

2.1 Tipe-tipe Paralelisme

1. Result Paralelisme

Result Paralelisme yang sering disebut sebagai Embarrassingly Parallel atau Perfect Paralel adalah tipe paralelisme dimana komputasinya dapat dibagi menjadi beberapa tugas independen yang mempunyai struktur sama[7]. Data struktur suatu tugas dibagi menjadi beberapa bagian yang berstruktur sama. Contoh tugas yang bisa diselesaikan dengan Result Parallelism adalah Simulasi Montecarlo.

2. Specialist Paralelisme

Cara kerja Specialist Paralelisme adalah dengan mengerjakan beberapa tugas secara bersamaan pada prosesor yang berbeda [7]. Setiap komputer mengerjakan tugas tertentu. Contohnya penggunaannya adalah pada simulasi pabrik kimia, satu prosesor mensimulasikan proses sebelum

reaksi kimia, satu prosesor mensimulasikan reaksi pada tahap awal, dan prosesor lainnya mensimulasikan proses penyulingan hasil, dan seterusnya.

3. Agenda Paralelisme

Tipe paralelisme ini mempunyai daftar yang harus dikerjakan oleh sistem komputer [7]. Semua komputer yang terdapat pada sistem dapat mengakses daftar tersebut. Pada Model MW (Manager Worker) terdapat pengelompokan komputer menjadi dua yaitu :

- a. Manager: bertugas memulai perhitungan, memonitor kemajuan tugas, melayani permintaan worker. User berkomunikasi dengan sistem komputer melalui komputer yang berfungsi sebagai manager ini.
- b. Worker: mengerjakan tugas-tugas yang diberikan oleh manager. Kerja komputer ini dimulai setelah ada perintah dari manager dan diakhiri oleh manager.

2.2 Pesan Terdistribusi dan Lingkungan Pemrograman (Distributed Messaging and Programming Environment)

Messages adalah urutan bytes yang dikirimkan antar proses. Pengirim dan penerima harus mempunyai kesepakatan mengenai struktur pesan sehingga isi pesan dapat diterjemahkan dengan benar. Pada prinsipnya cara pengiriman pesan adalah sederhana. Proses A mengirim data buffer sebagai sebuah pesan ke proses B. Pada saat bersamaan proses B menunggu datangnya pesan dari proses A. Ketika pesan tersebut maka proses B akan men-copy pesan tersebut di memori lokalnya.

2.2.1 Metode Pengiriman pesan (message passing)

Terdapat beberapa metode dalam pengiriman pesan yaitu :

a. Synchronous Message Passing

Cara pengiriman menggunakan metode ini ialah pengirim menunggu untuk mengirim pesan sampai penerima siap untuk menerima pesan. Oleh karena itu tidak ada buffering. Selain itu Pengirim tidak bisa mengirim pesan untuk dirinya sendiri.

b. Asynchronous Message Passing

Pengirim akan mengirim pesan kapanpun dia mau. Pengirim tidak peduli ketika penerima belum siap untuk menerima pesan. Oleh karena itu diperlukan buffering untuk menampung pesan sementara sampai penerima siap menerima pesan. Selain itu pengirim dapat pesan untuk dirinya sendiri. Selain berdasarkan metode pengiriman pesan diatas, pengiriman pesan (message passing) dibedakan berdasarkan jumlah penerima pesan yaitu: Point to Point dan Broadcast. Perbedaan mendasar keduanya adalah jumlah penerima yang menerima pesan. Pada Point to Point penerimanya tunggal sedangkan pada broadcast jumlah penerimanya banyak.

2.2.2 PVM dan MPI

Mesin virtual paralel atau Parallel Virtual Machine (PVM) dan Antarmuka Pengiriman Pesan (MPI) adalah kumpulan library yang memungkinkan kita untuk menulis program pengiriman pesan paralel menggunakan bahasa pemrograman C dan FORTRAN agar bisa berjalan pada sistem paralel.

Kemampuan sistem paralel tergantung dari kemampuan pemrogram untuk membuat aplikasi terdistribusi ketika dijalankan pada sistem paralel. Jika node slave mempunyai prosessor lebih dari satu maka pemrogram harus memperhitungkan kemungkinan paralelisme 2 level: Paralelisme di dalam slave node (intra-node parallelism) dan paralelisme antar slave node (inter-node parallelism). Inter-node parallelism menggunakan shared memory dalam node sehingga tidak melakukan pertukaran data secara explicit. Sedangkan Inter-node parallelism melakukan pertukaran data lewat media yang menghubungkan antara node slave yang ada.

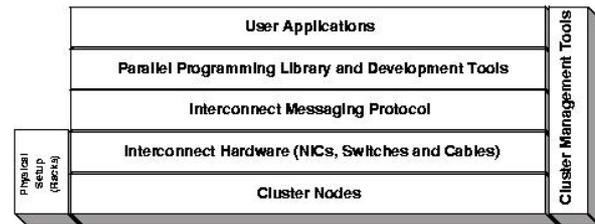
Terdapat tiga metode untuk mengimplementasikan Inter-node parallelism yaitu :

- Dengan cara membuat protokol komunikasi ad hoc level rendah. Contohnya dengan menggunakan socket interface.
- Dengan menggunakan distributed communication library. Contohnya dengan menggunakan Message Passing Interface (MPI) library
- Dengan memanfaatkan layer software dengan maksud untuk menyembunyikan interconnect dari programmer.

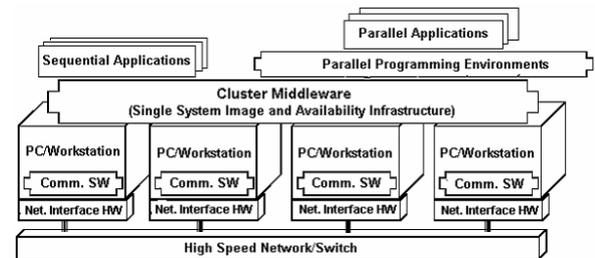
3. METODOLOGI PENELITIAN

Sistem pemrosesan paralel adalah sekumpulan komputer terhubung dan bekerjasama sebagai satu resource komputer yang terintegrasi untuk menyelesaikan suatu tujuan [3]. Sebuah sistem paralel setidaknya terdiri dari Message Passing Interface (MPI) dan sebuah pengatur beban kerja (job scheduler) . Message Passing Interface bertugas untuk mengirim data antar komputer di dalam sistem paralel (biasanya disebut sebagai node atau host). Job scheduler seperti yang tersirat dari namanya bertugas menerima tugas dari user dan menjadwalkan tugas tersebut pada beberapa node didalam sistem paralel sesuai kebutuhan. Gambar 1 memperlihatkan lapisan-lapisan pada arsitektur sistem paralel (LinuxSistem paralels.com).

Gambar 1 memberikan detail lebih mengenai arsitektur sistem paralel seperti yang terlihat pada Gambar 2.



Gambar 1. Arsitektur Sistem paralel



Gambar 2. Detail Arsitektur Sistem paralel

MPI (*Message Passing Interface*) adalah sebuah mekanisme pengiriman instruksi dan data antara dua proses komputasi yang berbeda yang berada pada komputer berbeda pada sistem sistem paralel. Paket-paket yang mempunyai spesifikasi kebutuhan MPI telah banyak beredar di Internet dan telah dilengkapi dengan LAM/MPI [5] dan MPICH [6]. Paket-paket ini telah dilengkapi dengan fungsi-fungsi yang menggunakan library C dan Fortran. Kemampuan MPI digunakan untuk menginterpretasikan bahasa pemrograman matrik kemampuan dynamic linking dari bahasa tersebut. Fungsi library dari paket MPI dapat digabungkan dengan dynamic extension dengan cara menghubungkan bahasa pemrograman tersebut dengan bahasa C, C++, atau FORTRAN. Hal ini telah dilakukan untuk menciptakan toolbox MPI (MPITB) untuk kebutuhan MATLAB, dan bahasa pemrograman GNU Octave oleh Fernandez Baldomero [4]. Pada makalah ini digunakan MPITB dengan pertimbangan fungsionalitas dan kelengkapannya disamping fakta bahwa MPITB dan GNU Octave adalah bebas digunakan bahkan untuk keperluan komersial. Hal ini juga berarti bahwa source code-nya banyak beredar dan dapat dimodifikasi sesuai kebutuhan.

4. IMPLEMENTASI

4.1 Desain Jaringan

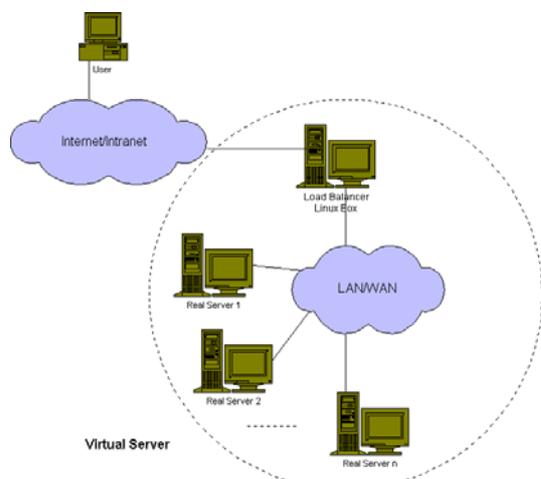
Secara garis besar, mekanisme pemberian layana publik bagi user di luar jaringan adalah sebagai berikut:

- User diluar jaringan diatas melakukan suatu request tugas, misalnya `tracetest_example.m`
- Request diterima oleh Load balancer/Linux box untuk kemudian diolah dan dibagi menjadi proses yang relatif lebih kecil
- Proses yang telah berukuran kecil tersebut diolah oleh masing-masing node/slave untuk diselesaikan.
- Setelah selesai melakukan tugasnya, node/slave mengirimkan kembali hasilnya ke

Load balancer untuk kemudian disusun kembali.

e. Hasilnya dikirimkan kembali ke user.

Dalam penelitian ini hanya akan digunakan lima buah komputer karena adanya keterbatasan resource yang ada. Sistem sistem paralel ini tidak terhubung ke Internet. Diasumsikan bahwa user menggunakan komputer Server/Master. Jikapun ada user yang berada dalam jaringan internet menginginkan untuk menggunakan sistem sistem paralel ini, maka ia dapat mengakses komputer server/master menggunakan program aplikasi Telnet/ssh. Gambar 3 memperlihatkan skema jaringan sistem paralel.



Gambar 3. Skema jaringan sistem paralel

4.2 Spesifikasi Hardware

1. *Spesifikasi Hardware Komputer Master/Server*
CPU AMD Athlon XP 1.8 GHz, Motherbord Albatron KX 18D Pro II, RAM 256 MB 168-pin PC2700 Registered MCPro, kartu Jaringan Realtek rtl8139, Hardisk 40 GB Seagate/Baracuda 7200 RPM, CD ROM Asus 52x, Floppy 1.4 MB, VGA ATI Radeon 9200 64MB PCI, Casing Simbada Full Tower
2. *Komputer/Node Slave 1*
CPU AMD Athlon XP 1.6 GHz, Motherbord VIA KT400 chipset AD77, RAM 32 MB 168-pin PC-2100 Registered Apacer, kartu Jaringan Realtek rtl8139, Hardisk 40 GB Seagate 7200 RPM, CD ROM Samsung 52x, Floppy 1.4 MB, VGA Geforce2 MX400 64 MB PCI, Casing Simbada Full Tower
3. *Komputer/Node Slave 2*
CPU AMD Athlon 950 MHz, Motherboard Biostar M7VIB, RAM 128 MB 168-pin PC-2100 Registered Apacer, kartu Jaringan Realtek rtl8139, Hardisk 20 GB Maxtor 5400 RPM, CD ROM Samsung 52x, Floppy 1.4 MB, VGA Geforce2 MX400 64 MB PCI, Casing ATX Full Tower.
4. *Komputer/Node Slave 3*
CPU Intel Pentium III Coppermine 533 GHz, Motherboard BX Master, RAM 128 MB 168-pin PC-133 Registered Visipro, kartu Jaringan

Realtek rtl8139, Hardisk 20 GB Quantum 5400 RPM, CD ROM Acer 50x, Floppy 1.4 MB, VGA Nvidia Riva TNT 64 MB PCI, Casing ATX Full Tower

4.3 Instalasi Sistem Operasi dan software

4.3.1 Instalasi Sistem Operasi

Setelah menentukan perangkat keras yang akan dipakai dalam sistem paralel maka langkah berikutnya adalah meng-install Linux. Ada berbagai macam distro Linux yang ditawarkan. Namun yang akan dipergunakan dalam makalah ini adalah Parallel Knoppix [1,2]. Parallel Knoppix adalah varian dari distro Linux Knoppix yang merupakan live-linux-on-CD. Hal itu berarti disini Linux dijalankan pada CD tanpa harus menginstal-nya pada hardisk. Selain itu script OpenMosix Terminal Server pada ParallelKnoppix telah dimodifikasi sehingga memudahkan dalam penggunaannya. Dengan menggunakan ParallelLinux, node/slave tidak perlu menggunakan CD ParallelLinux karena dapat melakukan booting jaringan dari komputer master. Untuk itu, motherboard dan kartu jaringan yang digunakan harus mendukung PXE (Preboot Execution Environment). Karena motherboard pada node/slave tidak mendukung PXE maka digunakan networkbootdisk.

Cara membuat networkbootdisk adalah sebagai berikut:

- a. Download driver kartu jaringan di <http://rom-o-matic.net> Dalam makalah ini digunakan kartu jaringan Realtek 8139 sehingga didapatkan `eb-5.2.0-rtl8139.zdisk`
- b. Masukkan disket pada floppy disk drive dan masukkan perintah: `$ cat eb-5.2.0-rtl8139.zdisk > /dev/fd0.`

4.3.2 Instalasi Software

Pada makalah ini digunakan MPITB dengan pertimbangan fungsionalitas dan kelengkapannya disamping fakta bahwa MPITB dan GNU Octave adalah bebas digunakan bahkan untuk keperluan komersial. Hal ini juga berarti bahwa source code-nya banyak beredar dan dapat dimodifikasi sesuai kebutuhan. GNU Octave adalah bahasa pemrograman untuk menginterpretasikan matrik. Sintaks GNU Octave pada umumnya compatible dengan MATLAB. Program yang ditulis pada MATLAB pada umumnya dapat dijalankan oleh Octave meskipun terkadang perlu dilakukan sedikit penyesuaian. Parallel Knoppix telah mempunyai paket-paket diatas didalam distribusinya sehingga memudahkan dalam penelitian. Untuk keterangan lebih lanjut kunjungi situsnya di <http://www.octave.org>. Pada makalah ini digunakan MPITB dengan pertimbangan fungsionalitas dan kelengkapannya disamping fakta bahwa MPITB dan GNU Octave adalah bebas digunakan bahkan untuk keperluan komersial. Hal ini juga berarti bahwa source code-nya banyak beredar dan dapat dimodifikasi sesuai kebutuhan. GNU Octave adalah

bahasa pemrograman untuk menginterpretasikan matrik. Sintaks GNU Octave pada umumnya compatible dengan MATLAB. Program yang ditulis pada MATLAB pada umumnya dapat dijalankan oleh Octave meskipun terkadang perlu dilakukan sedikit penyesuaian. Parallel Knoppix telah mempunyai paket-paket diatas didalam distribusinya sehingga memudahkan dalam peneliatian. Untuk keterangan lebih lanjut kunjungi situsnya di <http://www.octave.org>.

5. HASIL DAN PEMBAHASAN

Simulasi Montecarlo melakukan eksperimen random berkali-kali pada kondisi yang sama. Listing 1 pada Gambar 4 menunjukkan fungsi penghitungan statistic trace test. Fungsi ini membantu menjelaskan fungsi simulasi Montecarlo. Fungsi tersebut menerima sebuah argumen dan menghasilkan sebaris vektor sebagai hasil simulasi random. Argumen tersebut berupa array yang berisi panjang rangkaian pada posisi pertama dan nomer rangkaian pada posisi kedua. Fungsi tersebut manghasilkan hasil random dan menghasilkan report berupa vektor baris. Proses selanjutnya yang memanggil fungsi ini saling independent dengan proses sebelumnya dan tentu saja bisa dijalankan pada mesin/komputer yang berbeda.

Listing 2 pada Gambar 5 menunjukkan script Octave yang menjalankan tracetest yang menggunakan metode Montecarlo dengan cara mengevaluasi fungsi tracetest.m berulang kali. Hal yang perlu diperhatikan pada script ini adalah pada baris ke-7 dan ke-10 memanggil fungsi montecarlo. Pada baris ke 7 hanya ada 3 argumen pada pemanggilan fungsi Montecarlo yang berarti bahwa program dijalankan secara serial pada komputer yang menjalankan octave. Sedangkan pada baris ke-10 terdapat 4 argumen sebagai masukan bagi fungsi Montecarlo. Hal ini berarti bahwa proses dijalankan secara paralel dengan jumlah slave sebanyak masukan pada argumen ke-4. Saat menjalankan simulasi Montecarlo, user dapat melihat jumlah slave yang tersedia pada sistem sistem paralel. Dengan begitu user dapat menentukan secara eksplisit pada programnya jumlah slave yang hendak ia gunakan.

```
# Simulates trace test for cointegration
# Can be used to tabulate empirical distribution
# Ref. Doornik et. al. (2002)
# "Computationally-intensive Econometrics using
# a Distributed Matrix-programming Language",
# Philosophical Transactions of the Royal Society
of London,
# Series A, 360, 1245-1266.
function test_stat = tracetest(args)
    t = args{1};
    n = args{2};
    e = randn(t,n);
    p = inv(chol(e'*e/t));
    e = e*p; s = lag(cumsum(e),1);
    s(1,:) = s(1,:) - s(1,:); # lag fills with 1, test
needs 0
    fac = e'*s;
    ev = eig(fac*inv(s'*s)*fac'/t);
    test_stat = -t*sum(log(1 - ev/t));
endfunction
```

Gambar 4. Listing 1. Tracetest.m

```
# number of monte carlo replications
reps = 100000;

# specifics of test
T = 1000;
dim = 5;

# run on master only
output = montecarlo("tracetest",{T, dim}, reps);
hist(output, 30);

# run on master and one slave
output = montecarlo("tracetest",{T, dim}, reps, 1);
figure(2);
hist(output, 30);
```

Gambar 5. Listing 2. Tracetest_example.m

Untuk mengenerate waktu proses dijalankan program pada listing 3 pada Gambar 6. Script ini menjalankan replikasi montecarlo sebanyak 100.000 kali menggunakan 1 sampai dengan 4 komputer. Waktu yang dihasilkan dari script pada listing 3 dapat dilihat pada Tabel 1. Kolom pertama didapat dari percobaan orang lain di Internet menggunakan SMP (Symetric MultiProcessor) dengan spesifikasi 3,06 GHz Xeon, masing-masing menggunakan 512 KB level 2 cache, dan 2 GB RAM. Selain itu komputer ini juga dilengkapi dengan teknologi Hyperthreading sehingga SMP dapat diasumsikan mempunyai 4 CPU virtual. Kolom kedua didapat dari hasil percobaan sendiri menggunakan Pentium III 1 GHz, 256 KB leve2 cache, dan 128 MB RAM.

```
outfile = "SMP-";
maxslaves = 4;
T = 1000;
dim = 5;
reps = 100000;
for nslaves = 0:maxslaves
tic;
montecarlo("tracetest", {T,dim}, reps, nslaves);
t = toc;
results(nslaves+1,:) = [nslaves, T, dim, reps, t];
endfor
eval (sprintf("save\%stracetest-%d-%d-
%d_results.out"results", outfile, T, dim, reps));
#print timing results to file
```

Gambar 6. Listing 3. Tracetest_print.m

Ditetapkan bahwa lower limit runtime proses paralel adalah runtime roses serial dibagi jumlah node yang digunakan dalam sistem sistem paralel. Gambar dibawah menunjukkan runtime yang sesungguhnya dan lower limit fungsi

tracetest.m pada simulasi Montecarlo. Dari gambar diatas bahwa komputasi paralel sangat efisien, dilihat dari runtime yang mirip dengan lower limit. Hal ini didasarkan pada tingkat efisiensi LAM/MPI ketika menggunakan protokol TCP/IP dan implementasi MPITB yang efisien. Tabel 3 berisi laporan mengenai speedup (runtime pemrosesan serial dibagi runtime pemrosesan paralel) dan efisiensi (speedup dibagi jumlah node). Gambar 4.20 berikut merupakan hasil eksekusi program menggunakan satu buah komputer slave. Percobaan dilanjutkan dengan cara menambah jumlah komputer slave sampai berjumlah 3. Setiap penambahan slave dicatat peningkatan kecepatan proses. Hasilnya dapat dilihat pada Tabel 1. Perbandingan Runtime SMP dengan Sistem paralel.

Spesifikasi Komputer SMP (Single Multi-Processor) yang digunakan adalah satu komputer SMP dengan dua prosesor Xeon 3,06 GHz yang masing-masing mempunyai cache 512 KB level 2. Komputer ini dilengkapi dengan RAM sebesar 2 GB. Komputer ini mengaktifkan teknologi Hyperthreading sehingga seolah-olah mempunyai 4 prosesor (virtual CPU). Sebagai perbandingan maka 4 virtual CPU tersebut dibandingkan dengan Sistem paralel yang terdiri dari satu master dan 3 slave. Urutan penelitian pada Sistem paralel sesuai urutan spesifikasi hardware diatas.

suatu masalah lebih cepat dari suatu komputer tunggal.

DAFTAR PUSTAKA

- [1] Creel, Michael (2004), "ParallelKnoppix - Create a Linux Cluster for MPI Parallel Processing in 15 Minutes", <http://pareto.uab.es/mcreel/ParallelKnoppix/>
- [2] Creel, Michael, 2004, Parallel-Knoppix –Rapid Creation of a Linux Cluster for MPI Parallel Processing Using Non-Dedicated Computers, <http://pareto.uab.es/mcreel/ParallelKnoppix/>
- [3] Kant, Chander, 2002, Introduction to Clusters, <http://LinuxCluster.com>.
- [4] Fernández Baldomero, J. (2004), "LAM/MPI Parallel Computing under GNU Octave", <http://atc.ugr.es/javier-bin/mpitb>.
- [5] LAM team (2004), "LAM/MPI Parallel Computing", <http://www.lam-mpi.org/>
- [6] Gropp, W., E. Lusk, N. Doss and A. Skjellum (1996), "A high-performance, portable implementation of the MPI message passing interface standard", *Parallel Computing*, 22,789-828
- [7] Mateti, Prabhaker, 2005, "Cluster Computing with Linux".

Tabel 1. Perbandingan Runtime SMP dengan Sistem paralel

Node	SMP (runtime)	Parallel (runtime)
1	363,6	610,5
2	209,0	309,8
3	209,3	211,3
4	213,2	165,6

Tabel 2. Speedup dan Efisiensi

Node	Speedup	Efisiensi
1	1	1
2	1,97	0,98
3	2,89	0,96
4	3,68	0,92

Segala aktivitas yang dilakukan oleh sluster dapat diamati menggunakan software Ganglia.

6. KESIMPULAN

Dari perbandingan diatas dilihat bahwa sistem paralel dapat dibangun dari kumpulan komputer dengan spesifikasi yang beraneka macam. Hal itu ditunjang dengan fakta bahwa sistem paralel dapat memanfaatkan sumber daya komputer yang telah ada walaupun spesifikasinya berbeda-beda. Selain itu pada jumlah komputer, sistem paralel dapat menyelesaikan suatu