Tanggal :
Lokasi :

Susun!

BAGIAN KETUJUH : KASUS PADA KOMPUTER LOKAL (TUNING)

1. Pendahuluan

Ketika komputer masih sederhana, terdapat dua cara mudah untuk mengukur kinerja komputer. Pertama, menggunakan parameter sistem itu sendiri misalnya laju detak prosesor atau jumlah instruksi yang dapat diproses tiap satuan waktu. Jumlah instruksi yang diproses dalam satuan waktu tertentu diekspresikan dalam 'satuan' MIPS (*millions instruction per second*). Ukuran ini menjadi tidak adil digunakan pada komputer yang menggunakan prosesor dengan arsitektur berbeda. Prosesor berarsitektur RISC misalnya, memerlukan lebih banyak instruksi untuk menjalankan suatu tugas (*task*) tertentu dibandingkan dengan prosesor berasitektur CISC. Dengan ukuran MIPS, prosesor RISC akan tampak bekerja jauh lebih cepat (menjalankan lebih banyak instruksi per satuan waktu) dibanding prosesor CISC meskipun lama waktu yang digunakan untuk menyelesaikan satu tugas bisa jadi sama.

Cara kedua adalah dengan menggunakan *benchmark* sintetis. Whetstone dan Dhrystone merupakan contoh *benchmark* sintetis yang banyak digunakan untuk mengukur kinerja komputer. *Benchmark* sintetis berupa program pendek yang dibuat menyerupai tingkah-laku program aplikasi yang ada. Melalui kajian mendalam terhadap berbagai program aplikasi yang ada, dibuat suatu program pendek yang merupakan gabungan dari berbagai komputasi matematis, kalang (*loop*), pemanggilan fungsi, dan sebagainya. Terhadap pengukuran dengan *benchmark* sintetis terdapat dua kelemahan. Selain keraguan apakah program yang disusun benar-benar mewakili program aplikasi yang sebenarnya, keraguan hasil pengukuran juga disebabkan oleh mudahnya teknik pengukuran ini dimanipulasi dengan melakukan optimisasi kompilator (Sharp dan Bacon, 1994:65).

2. Metodologi Optimalisasi

Pemilihan instrumen Pengukuran

Pemilihan ini amat bergantung dengan subyek yang akan di ukur. Berikut ini adalah beberapa contoh metode pengujian bagi komponen komputer

1. Tes CPU

Terdiri dari :

Susun!

1. TES ARITMATIK CPU : Benchmark MP Dhrystone (MIPS) CPU (aritmatik) 32-bit dan Benchmark P Whetstone (MFLOPS) FPU/SSE (Floating Point) 32-bit.
2. Tes CPU MULTIMEDIA : Dilakukan berdasarkan Intel mandelbrot application dan mesin Mandelbrot 4x, terdiri dari :
    a. tes integer : MP SSE2, MP SSE (streaming SIMD), MP Enhanced MMX, MP MMX, MP ALU
    b. Tes floating point : MP SSE2, MP SSE (streaming SIMD), MP 3Dnow! Enhanced MMX, MP 3Dnow!, MP FPU

Gambaran Tes CPU secara detail

1. CPU Tes - Overall

Memberikan tugas khusus yang intensif kepada prosesor yang banyak digunakan pada aplikasi sehari-hari yaitu tes operasi integer dan floating point.

2. CPU Test - JPEG decompression

Tes ini mewakili tugas CPU ketika browsing di web dalam membaca dokumen dengan gambar. Tes ini mecoba medekompresi 3 gambar dalam waktu 10 detik mulai dari ukuran file gambar 149kB, 771kB, dan 889kB. Hasilnya dalam Mpixels/s, berapa banyak rata-rata gambar yang berhasil didekompresi per detik.

Teknis Detil Tes :

Standard JPEG library (version 6b) diambil dari Independent JPEG Group (www.ijg.org). file gambar akan di load ke memori sebelum didewkompresi proses decoding JPEG menggunakan pipeline fixed-point IDCT dan RGB-24 dengan output dalam pixel format, jadi tes ini adalah tes operasi integer.

3. CPU Test - Zlib compression & decompression

Proses kompres dan dekompresi terdapat banyak pada aplikasi sehari-hari. Data yang akan ditransfer biasanya mengalami proses-proses ini ketika ditransfer. Ada 3 file, yaitu 887kB JPG image, 1468kB file text , dan 1280kB file executable, yang akan mengalami kompresi berapa kali dalam waktu 10 detik tergantung kamampuan CPU, lalu dalam putaran 10 detik berikutnya mengalami dekompresi. Hasil dalam bentuk MBytes/s, atau berapa megabytes data yang dapat dikompresi/dekompresi per detik.

Teknis Detil Tes :

Tanggal :
Lokasi :

Susun!

Test ini adalah perhitungan integer. Setiap file tes akan terload dalam 1MB memory buffer sebelum dikompresi. Area buffer lainnya dengan ukuran 500KB digunakan untuk mengkompresi data buffer, dan waktu untuk mengkompresi/dekompresi data dapat diindikasikan sebagai CPU speed. Standar LZ77 metode kompresi didapat dari open source ZLIB (www.gzip.org/zlib/)

4. CPU test - Text search

Text file berukuran besar digunakan dalam tes ini, yang diukur adalah berapa operasi cari yang sukses. Proses pencarian teks ini banyak digunakan pada aplikasi browsing the web, e-mail, dan proses pengolahan dokumen. merupakan operasi integer.

Teknis Detil Tes :

Menggunakan Boyer-Moore algorithm, dipilih karena amat efisien dalam aplikasi sehari-hari. Text file sebesar 1.5MB diload ke sistem memori sebelum pencarian dilakukan. Mengukur banyaknya frekuensi yang tampil dalam menemukan teks yang dicari dalam waktu 10 detik.

5. CPU test - Audio Conversion

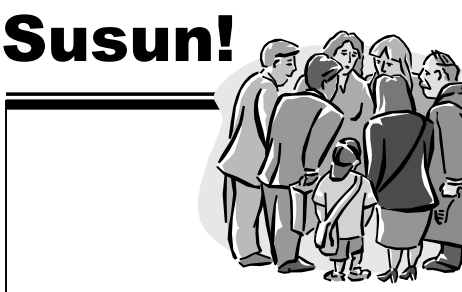
Kompresi MP3 file audio menggunakan algoritma public audio compression format Ogg Vorbis (http://www.gnu.org/directory/oggvorbis.html). Microsoft MP3 decoder (terdapat pada DirectShow filter DirectX) digunakan untuk dekompresi. Untuk MP3 encoder menggunakan Ogg Vorbis encoder. Tes ini mengukur CPU workload dari MP3 playback dan kompresi.

Teknis Detil Tes :

30 detik 128kbit/s MP3 stream (500kb ukuran terkompresi) didekompresi secara simultan ke 44kHz/16bit PCM format dan dalam Ogg Vorbis format. DirectShow digunakan untuk membuat filter graph agar kinerja dan waktu yang digunakan dapat dihitung. Filter yang dipakai dalam graf tersebut : File Source (Async.), MPEG-I Stream Splitter, MPEG Layer-3 Decoder, Vorbis Stream Encoder, AVI Mux dan Null Renderer,

6. CPU test – Kalkulasi 3D Vector

Simulasi rambut pada kepala manusia merupakan salah satu teknik advanced real-time 3D demo. Ini banyak digunakan pada game terakhir. Obyek pada rambut terdiri dari banyak sekali polygon. Yang membentuk suatu hubungan yang amat kecil satu sama lain. Kalkulasi floating point 3D transformation yang sangat kompleks dibutuhkan untuk dapat menampilkan hal ini. Tes



berlangsung dalam 10 detik, hasilnya menunjukkan berapa fram yang berhasil ditampilkan dalam masa tersebut.

Teknis Detil Tes :

Test simulasi keadaan physics dan pencahayaan pada rambut. Rambut tersebut dijadikan model menggunakan garis polygon dengan 7300 rambut 8 node, menkonsumsi 7300 * (24 + 8*24) = 1576800 ~= 1.5 MB memory. Kode fisik rambut diproses memperhatikan 3 hal : gravitasi, kelurusan rambut dan kriting. Teknik pencahayaan adalah anisotropic, dikodekan dengan MAX-FX's vector template library. Kode terdiri dari penambahan vektor, produk titik dan garis silang dan normalisasi vektor yang menggunakan nilai 32 bit floating-point.

2. TES MEMORI

Kondisi Pengukuran :

- Multiprosesor sampai 32 CPU dan SMT.
- Benchmark operasi aritmatik (assignment, skala, penambahan, triad)
- Tes perlu 50% lebih memori fisik yang tidak terpakai.
- Tutup semua program, terlebih jika memorinya lebih kecil dari 16MB.
- Perlu 2 menit pada pada mesin kelas P6 dengan memori 64 MB.

Metode tes :

- ALU/FPU tes memori
- MMX tes memori (integer)
- I/F SSE tes memori – SGI bandwidth memori (integer dan Float)
- I/F EMMX/SSE tes memori – AMD bandwidth memori (integer dan Float)
- MP/MT
- Mengecek informasi di DMI/SMBIOS pada Motherboard

Gambaran Tes Memori secara detail

1. Memory Tests – Overall
Tes ini mengukur kinerja subsistem memori, L2 dan L1 cache dengan memberikan operasi read, write, read-modify-write, dan random access, suatu operasi yang sering terjadi di kebanyakan aplikasi. Tes ini untuk mendapatkan maximum throughput pada satu bagian proses saja, dengan catatan hanya satu tugas saja ketika terjadi transfer di memori.

2. Memory tests - Raw Access and Random Access
Memori di tes dengan memberikan ukuran operasi yang berbeda untuk operasi read, write, and read-modify-write dan mengambil data di daftar STL.

Tanggal :
Lokasi :

Susun!

Teknis Detil Tes :

Operasi Raw read, write, dan read-modify-write sangat baik dimulai dari 3072 kilobytes array yang berkurang sampai 1536 kB, 384 KB, 48 KB dan akhirnya 6 KB. Setiap ukuran blok dites sebanyak 2 detik dan jumlah data yang terakses yang diukur. Dalam tes STL daftar dari elemen 116 byte dibangun dan diurutkan oleh kunci integer pseudo-random. Daftar tersebut menunjukkan berapa waktu yang mungkin untuk 2 detik dan berapa total akses terhadap elemen data yang didapatkan. Ada 6 tes, yaitu dengan 24576 item dalam data 1536 kB, berkurang sampai 12288 item (768 kB), 6144 item (384 kB), 1536 item (96 kB), 768 item (48 kB) dan 96 item yang paling kecil berkorespondensi ke 6 kB of total data.

3. Tes Video memory sub-sistem

membebani memori dengan aplikasi desktop windows. Tes akan mendorong memori internal kartu grafis dan bandwith dengan kecepatan transfer pada AGP. Sesuatu beban yang biasa terjadi pada saat aplikasi melakukan scrolling dokumen dan moving/resizing windows.

Teknis Detil Tes :

Tes dimulai dengan membuat back buffered primary surface pada resolusi 1024x768 32 bit di DX exclusive mode dan satu off-screen bidang kerja dalam format pixel yang sama. Bidang kerja tersebut di-update dengan setiap frame yang ditransfer dari bus AGP, yang mewakili kecepatan scrolling (1, 4, 16 and 32 scan lines / frame). Bidang kerja akan terisi setiap frame yang kemudian akan membebani bandwith internal memori. Setiap kecepatan transfer tersebut dicatat setiap 3 detik dan berapa total frame yang berhasil di-update itulah hasil yang didapatkan.

3. TES MEDIA PENYIMPANAN

3.1. TES HARD-DISK

Terdiri dari pengukuran :

- DTR (data Transfer rate)
- Tes kinerja file sistem pada media penyimpanan.
- Tes baca : buffer, random, sekuensial
- Tes tulis : buffer, random, sekuensial
- Tes cari

Gambaran Tes harddisk secara detail

Susun!

1. HDD Tests – Overall

Mengukur kinerja dengan mengukur operasi Read dan write menggunakan cache dan uncache. Tes ini mensimulasikan aplikasi real seperti menulis, membaca dan meng-copy file. Pertama kali eksekusi program, semua bagian logikal disk akan diuji, termasuk space yang masih kosong.

Teknis Detil Tes :

Directori sementara "__PCMARK" akan dibuat dalam root directory pada disk yang dipilih. Akan diulang pembuatan sebanyak 8x8x8 subdirectory pada directory sementara tersebut. Waktu untuk membuat struktur directory ini tidak diukur.

2. File write test

18 file dibuat dengan ukuran bervariasi dari 1KB sampai 128MB, dengan total data yang ditulis sampai 256MB. Rata-rata penulisan (MB/s) yang diambil sebagai hasil. Untuk mensimulasi cara penulisan random, 18 file tersebut ditempatkan secara random di directory yang dibuat. 1MB buffer memori digunakan untuk membuat raw data untuk ditulisi. Untuk file yang lebih besar dari 1 MB digunakan buffer yang sama secara bersamaan. Penulisan file dites dengan 2 cara, yaitu mode cache dan mode uncache. Pada mode uncache, file buffer terus diisi ketika penulisan dilakukan. Dalam mode cache, setiap file ditulisi sekali dan file-file sistem kemudian men-copy data ke file cache sebelum dicopy ke hard-disk. Penulisan dengan cache lebih lambat dari metode uncache.

3. Tes File Read

Pembacaan berdasarkan directory yang sudah dibuat dan file yang sudah ditulisi. 1MB bufferjuga digunakanuntuk menyimpan data sementara, dan file dibaca dalam random. Rata-rata reading rate (MB/s) adalah hasilnya. Tes baca juga menggunakan cache dan non-cache mode. Dalam mode cache pembacaan file sistem akan mengcopy data dari harddisk ke file cache sebelum data berada di buffer memori.

4. Tes File copy

Tes ini dilakukan dengan cara meng-copy ke directory yang didefinisikan terlebih dahulu, dengan nama lain. Analisisnya dilakukan dengan mengkombinasikan tes read dan write. Setelah operasi copy dilakukan, file sementara dihapus. Rata-rata transfer rate data (MB/s) adalah hasilnya.

File System sangat berpengaruh pada score

Susun!

Windows akan menyiapkan blok memori sebagai cache operasi file. Cache dapat lebih besar dari 80% total memori sistem. Sebelum semua tes tersebut, penggunaan memori buffer diminimalkan terlebih dahulu.

3.2. TES CD-ROM/DVD

Tes perbandingan CD-ROM/DVD drive dan controller :
· tes baca : Buffer, Random, Sequential
· tes cari

Benchmark yang dilakukan dapat terdiri atas 2 macam tes :

- Kinerja Data CD : Disk dengan kapasitas data 600MB+ dibutuhkan minimal (64MB+) file data., termasuk pula data MPEG, MOV, AVI dan database yang besar.
- Kinerja VideoCD / DVD : Untuk tes VideoCD, perlu 300MB+ VideoCD rekomendasi 600MB+ VideoCD

· Tes biasanya memerlukan waktu 10 menit pada mesin kelas P6 dengan 32x CD-ROM.

Metode tes :

1. Mem-bypass Windows Cache : menggunakan mode write through untuk bypass cache, normalnya menggunakan cache.
2. Mengaktifkan Multi processing / multi threading Benchmark, normalnya single threaded mikroprosesor sistem.
3. Menggunakan statik MP load balance, normalnya selalu melakukan kalibrasi CPU sebelum eksekusi dan menggunakan unit variabel kerja.
4. Mengambil ekstra informasi disc.

4. Tes Sistem Video (tampilan)

Terdiri dari 4 Macam tes Video:

- Video encoding, melakukan kompresi ASF (Advanced Streaming Format, a type of MPEG4).
- ASF playback, memainkan clip ASF yang dibuat pada tes encoding.
- DVD playback, memainkan clip DVD (a type of MPEG2)dan mengukur CPU utilization.
- Video quality, testing DVD playback quality.

Catatan : encoding ASF dan tes playback membutuhkan Microsoft Windows Media Encoder 7.1 dan Microsoft Windows Media Player 7.1. Playback DVD dan tes kualitas video membutuhkan program DVD playback, antara lain :

- Cyberlink PowerDVD

Susun!

- Intervideo WinDVD
- Ravisent Cinemaster

Gambaran Tes Video secara detail

1. Kompresi ASF

ASF video clip dibuat dari MPEG1 clip, menggunakan Windows Media Encoder. Dua clip yang dibuat : clip low-resolution dengan resolusi 352x288, dan clip high-resolution dari hasil DivX, dengan resolusi 640x480.

2. ASF playback

Dua clip yang dimainkan : clip low-resolution dengan resolusi 352x288, dan clip high-resolution dari hasil DivX, dengan resolusi 640x480.

3. DVD playback

Mengukur CPU load dengan menjalankan playback 2 clip DVD dengan resolusi 720x480, pada 4 Mbit/s bit rate dan pada variabel 7-12 Mbit/s bit rate.

4. Video quality

DVD playback quality dites berdasarkan tes de-interlace. De-interlacing adalah proses transformasi content video interlaced (modus bergerak) ke format yang dapat ditampilkan pada layar monitor.

Semua tes (kecuali tes kualitas gambar) diukur berdasarkan rata-rata frame rate (rendered frames / detik), Semua tes performance dijalankan dengan cara:

1. Menginisialisasi scene
2. Merender 1-3 (termasuk dalam tes) frame untuk pemanasan yaitu mendownload tekstur yang dibutuhkan ke memori grafik card.
3. Memulai perhitungan timer
4. Merender sebanyak-banyaknya frame yang mungkin dalam n detik.
5. ketika n detik tercapai, gambarkan black triangle setelah semua frame dalam pipeline, untuk meyakinkan semua frame sudah berhasil ditampilkan.
6. Mengunci isi frame buffer
7. Men-stop timer
8. Membebaskan frame buffer
9. Menghitung hasilnya berdasarkan NumberOfFramesRendered / Time
10. De-inisialisasi scene

Tanggal :
Lokasi :

Susun!

5. Tes Network / LAN Bandwidth

Menggunakan interface ICMP (ping / Echo) untuk mengukur waktu respon dan lebar bandwith transfer ke suatu host.

Metode tes :

- Ping untuk status koneksi.
- Melihat statistik transfer.
- Mengecek IPX/SPX
- Mengecek informasi domain dan workgroup.

Library winsock – fungsi pada winsock (versi 16-bit dan 32-bit) :

1. Status library
2. Maksimum soket proses yang dapat digunakan, menunjukkan berapa koneksi yang dapat dilakukan dalam satu waktu.
3. Maksimum ukuran UDP datagram (KB) dari paket yang dapat dikirim UDP.
4. Host name (nama domain, komputer), IP address, IPX address (alamat hardware)
5. Informasi service : nama, port, protokol : FTP, telnet, HTTP.

6. Tes Crunch

Tes ini menjalankan 3 tes secara simultan :

- CPU Test - JPEG Decompression, membebani sebagian besar CPU dan juga memory subsystem.
- Memory Test - Raw 3072 kB modifikasi Blok, membebani sebagian besar memory subsystem, begitu juga CPU.
- Memory Test - Video - 1 scanline, membebani memori grafik card bandwidth dan bus AGP.

Tes ini dilakukan untuk mengecek seberapa kuat sistem menjalankan tugas secara simultan dan dibebani oleh beberapa macam elemen yang berbeda.

Perencanaan dan tahap pengukuran

- Memanfaatkan monitor software sampling (secara batch atau secara online)
    1. Ekstraktor memanfaatkan time interval (jam, clock system) untuk menginterupsi aktivitas sistem pada interval waktu reguler bagi ekstraksi data yang berfokus kepada keadaan sistem.
    2. Analizer dipanggil secara periodik untuk melakukan analisa terhadap respon sistem terhadap ditugas-tugas yang diberikan kepadanya.

Tanggal :
Lokasi :

Susun!

- Menugaskan CPU kepada program dengan prioritas lebih besar atau program yang memiliki beban kerja (work load) yang lebih berat.
- Memeriksa frekuensi penggunaan resources sistem seperti memori pada even-even aktivitas sistem tersebut.

3. Contoh Pengukuran

Pengukuran Kinerja Komputer dan Permasalahannya

Tulisan berikut ini akan mengulas salah satu *benchmark* yang banyak dipergunakan, yakni SPEC, yang dikembangkan oleh System Performance Evaluation Corporation. Uraian tentang *benchmark* SPEC ini dimaksudkan untuk menjelaskan cara pengukuran kinerja komputer dengan membandingkan waktu eksekusi yang diperlukan oleh suatu komputer untuk satu program tertentu dengan waktu eksekusi pada komputer rujukan. Pada bagian berikutnya diuraikan mengenai kelemahan pengukuran kinerja berbasis 'waktu eksekusi program' tersebut dan alternatif lain yang dapat dilakukan untuk memperbaiki pengukuran kinerja komputer tersebut.

SPEC95

Sebagai upaya untuk mendapatkan tolok-ukur baku agar dapat membandingkan kinerja berbagai sistem komputer, sekelompok perusahaan besar antara lain: DEC, Hewlett-Packard, IBM, Intel, dan Sun sepakat membentuk lembaga non-profit yang diberi nama System Performance Evaluation Corporation (Sharp dan Bacon, 1994:66; Reilly, 1995). Lembaga ini ditugasi untuk mengembangkan dan memberi dukungan terhadap pembakuan *benchmark* kinerja komputer.

Sebelum membuat program untuk mengukur kinerja komputer, SPEC telah mempelajari sejumlah program yang umum dipakai, menganalisis algoritma dan bahasa mesinnya, menentukan cara mengukur kinerja komputer, dan menentukan rumusan untuk membuat rerata skor kinerja komputer dari skor-skor yang diperoleh masing-masing elemen *benchmark*. *Benchmark* SPEC terdiri atas dua kelompok program. Satu kelompok merupakan program-program yang dititik-beratkan pada operasi atas bilangan *integer* dan satu kelompok lainnya dititikberatkan pada operasi atas bilangan *floating-point*.

Perangkat *benchmark* pertama yang dibuat diperkenalkan pada tahun 1989, karenanya disebut SPEC89. Pada tahun 1992 dimunculkan versi baru, dan dengan demikian SPEC89 tdak digunakan lagi. SPEC92 terdiri atas 20 program yang terbagi menjadi dua kelompok, yakni untuk operasi bilangan *integer* dan untuk operasi bilangan *floating-point*. Saat ini, SPEC92 juga sudah tidak digunakan lagi karena telah dimunculkan perangkat *benchmark* baru yakni SPEC95. Pada SPEC95 ini, komputer rujukan yang digunakan

Susun!

sebagai pembanding berubah dari semula VAX-11/780 menjadi Sun SPARCstation 10/40. Dengan demikian, bila dikatakan skor SPECint95 adalah 5.0, maka berarti sistem yang diuji 5 kali lebih cepat dibanding Sun SPARCstation 10/40.

SPEC95, yang diperkenalkan pada bulan Agustus 1995, merupakan perangkat *benchmark* yang terdiri atas dua bagian, yakni CINT95 (ditulis dalam bahasa C) dan CFP95 (ditulis dalam bahasa Fortran). CINT95 merupakan bagian dari perangkat SPEC95 yang mengukur kinerja komputer terhadap operasi bilangan *integer*, yang diasumsikan mewakili program aplikasi bisnis. Bagian lain, yakni CFP95, mengukur kinerja komputer terhadap operasi bilangan *floating-point* yang diasumsikan mewakili program aplikasi ilmiah-numerik.

Tabel 1 :

Elemen-elemen CINT95

Nama Program	Deskripsi	Waktu Referensi (Detik)
099.go	Program kecerdasan buatan, menjalankan program permainan Go melawan dirinya sendiri	4600
124.m88ksim	Simulator cip mikroprosesor Motorola 88100, menjalankan program uji Dhrystone dan program uji memori	1900
126.gcc	Melakukan kompilasi untuk bahasa mesin prosesor SPARC berbasis kompilator GNU C versi 2.5.3.	1700
129.compress	Program pemampat (compress) dan pengurai (decompress) file teks sebesar 16 MB yang bekerja dalam-memori (in-memory), dengan metode pengkodean Limpel-Ziv adaptif	1800
130.li	Interpreter bahasa LISP	1900
132.ijpeg	Program pemampat dan pengurai citra (image) dengan berbagai parameter, yang bekerja dalam-memori.	2400
134.perl	Melakukan manipulasi numerik dan teks (anagram dan pemfaktoran bilangan prima)	1900
147.vortex	Membuat dan memanipulasi tiga basis data yang saling berkaitan	2700

Susun!

Tabel 2 :

Elemen-elemen CFP95

Nama Program	Deskripsi	Waktu Referensi (Detik)
101.tomcatv	Mewakili program bidang dinamika fluida/ translasi geometris. Membangkitkan sistem koordinat dua-dimensi pada domain geometris umum.	3700
102.swim	Model perairan dangkal dengan 1024x1024 kisi	8600
103.su2cor	Simulasi Monte Carlo, bidang fisika kuantum. Melakukan perhitungan massa partikel elementer berdasarkan teori Quark-Gluon	1400
104.hydro2d	Bidang astrofisika, menyelesaikan persamaan hidrodinamik Navier Stokes untuk menghitung semburan galaktik	2400
107.mgrid	Bidang elektromagnetis, memecahkan persoalan medan potensial 3 dimensi multikisi (multigrid)	2500
110.applu	Bidang dinamika fluida/matematika, menyelesaikan sistem matriks dengan *pivoting*	2200
125.turb3d	Simulasi turbulensi homogen isotropik dalam kubus	4100
141.apsi	Pemecahan persoalan berkaitan dengan temperatur, angin, serta kecepatan dan distribusi polutan	2100
145.fppp	Menyelesaikan derivasi multielektron , bidang kimia kuantum	9600
146.wave5	Penyelesaian persamaan Maxwell pada jaring Cartesian	3000

Bencmark SPEC95 mengukur dan membandingkan kinerja komputer dalam tiga kategori pilihan:

1. Kinerja terhadap bilangan integer versus floating point
2. Kinerja dengan kompilasi agresif versus kompilasi konservatif
3. Kecepatan versus *throughput*

Berdasarkan pilihan-pilihan tersebut, SPEC95 memungkinkan dibuatnya 'ukuran' komposit sebagai berikut:

Tanggal :
Lokasi :

Susun!

Tabel 3.
Pilihan 'Ukuran' Kinerja Komposit

Metode Kompilasi	Kecepatan	Throughput
Kompilasi Agresif	SPECint95 SPECfp95	SPECint_rate95 SPECfp_rate95
Kompilasi Konservatif	SPECint_base95 SPECfp_base95	SPECint_rate_base95 SPECfp_rate_base95

Yang dimaksud ukuran komposit adalah bahwa skor indivual dihitung untuk tiap-tiap elemen program (8 elemen pada CINT95 dan 10 elemen pada CFP95) dalam CINT95 atau CFP95 dan hasilnya digunakan untuk menghitung ukuran komposit ini.

Untuk pengukuran kecepatan, setiap elemen *benchmark* memiliki SPECratio. SPECratio adalah referensi waktu SPEC dibagi dengan waktu-pelaksanaan (*runtime*) tiap-tiap elemen program pada sistem yang diukur.

Ukuran komposit kecepatan dapat dihitung sebagai berikut:

1. SPECint95 : Rerata geometris dari 8 buah SPECratio (satu untuk tiap elemen program CINT95) bila masing-masing elemen program dikompilasi dengan optimisasi agresif.

2. SPECint_base95 : Rerata geometris dari 8 buah SPECratio (satu untuk tiap elemen program CINT95) bila masing-masing elemen program dikompilasi dengan optimisasi konservatif

3. SPECfp95 : Rerata geometris dari 10 rasio ternormalisasi (satu untuk tiap elemen program CFP95) bila masing-masing elemen program dikompilasi dengan optimisasi agresif.

4. SPECfp_base95 : Rerata geometris dari 10 rasio ternormalisasi (satu untuk tiap elemen program CFP95) bila masing-masing elemen program dikompilasi dengan optimisasi agresif.

Untuk mendapatkan ukuran *throughput* atau laju eksekusi program, yang disebut juga 'metode kapasitas homogen' beberapa salinan elemen program tertentu dijalankan. Metode ini terutama cocok untuk sistem multiprosesor. Hasilnya, yang disebut laju SPEC (*SPECrate*) menggambarkan berapa banyak elemen program yang dapat dijalankan pada satu waktu tertentu. Dengan

demikian laju SPEC menggambarkan kapasitas sistem untuk tugas-tugas yang sama karakteristiknya dengan program uji (Dixit dan Reilly, 1995).

Sama seperti ukuran kecepatan, SPEC mendefinisikan rerata ukuran laju sebagai berikut:

1. SPECint_rate95 = Rerata geometris dari 8 buah SPECrate ternormalisasi (satu untuk tiap elemen program CINT95) bila masing-masing elemen program dikompilasi dengan optimisasi agresif.
2. SPECint_rate_base95 = Rerata geometris dari 8 buah SPECrate ternormalisasi (satu untuk tiap elemen program CINT95) bila masing-masing elemen program dikompilasi dengan optimisasi konservatif.
3. SPECfp_rate95 = Rerata geometris dari 10 buah SPECrate ternormalisasi (satu untuk tiap elemen program CFP95) bila masing-masing elemen program dikompilasi dengan optimisasi agresif.
4. SPECfp_rate_base95 = Rerata geometris dari 8 buah SPECrate ternormalisasi (satu untuk tiap elemen program CINT95) bila masing-masing elemen program dikompilasi dengan optimisasi konservatif.

Benchmark lainnya

Selain SPEC95, berbagai *benchmark* dikembangkan untuk mengukur kinerja komputer. Berikut ini disajikan beberapa contoh beserta deskripsi singkatnya untuk memberikan gambaran mengenai perbedaan masing-masing.

Whetstone merupakan *benchmark* sintetik yang dikembangkan oleh Curnow dan Wichman pada tahun 1976 (Sharp dan Bacon, 1994: 68). *Benchmark* ini dimaksudkan untuk mengukur kinerja komputer dalam mengolah bilangan *floating point* dan digunakan untuk membandingkan arsitektur maupun kompilator teroptimisasi yang dijalankannya. Program semula dibuat dalam bahasa Algol dengan kompilator Algol 60 yang menterjemahkannya menjadi instruksi untuk mesin Whetstone imajiner (Sill, 1996). Kelemahan *benchmark* Whetstone adalah kecilnya ukuran modul/program benchmark sehingga sistem memori di luar *cache* tidak teruji, dan dengan optimisasi kompilator dengan mudah didapatkan skor *benchmark* tinggi tanpa mengubah sistem yang diuji (Sill, 1996).

Dhrystone juga merupakan *benchmark* sintetik yang dikembangkan oleh Reinhold Weicker pada awal tahun 1980-an dan difokuskan untuk mengukur kinerja komputer atas bilangan integer dan string (Sharp dan Bacon, 1994:68). Program asli ditulis dalam bahasa Ada, dan kemudian diterjemahkan ke dalam bahasa-bahasa lain. Sama seperti Whetstone, program *benchmark* Dhrystone memiliki ukuran yang terlalu kecil (sekitar 1,5 KB) sehingga tidak dapat menguji sistem di luar *cache*. Optimisasi kompilator juga dapat dilakukan untuk mempertinggi skor perolehan (Sill, 1996).

Susun!

Susun!

Linpack yang dikembangkan oleh Jack Dongarra, kernelnya dikembangkan dari rutin program aplikasi aljabar linier (Sharp dan Bacon, 1994:68; Sill, 1996). Semula ditulis dan digunakan dalam lingkungan bahasa program Fortran namun tersedia juga versi bahasa C. Sebagian besar waktu uji merupakan waktu eksekusi subrutin yang menjalankan operasi matriks y(i) = y(i) + a * x(i). Versi standar bekerja dengan matriks ukuran 100x100, tetapi juga tersedia versi dengan ukuran matriks 300x300 dan 1000x100 dengan aturan optimisasi yang berbeda. Kelemahannya, program hanya mewakili tipe komputasi matriks yang memang banyak digunakan dalam bidang bidang sains (Sill, 1996).

NAS Parallel Benchmark (NPB), yang dikembangkan oleh peneliti di NAS, yakni cabang dari NASA Ames Research Lab, untuk mengukur kinerja komputer paralel. NPB dikembangkan khusus untuk mengukur kinerja komputer paralel, yang memerlukan penulisan ulang program agar dapat secara efektif dan efisien membagi beban komputasi di antara prosesor-prosesornya (Sharp dan Bacon, 1994:68).

Para peneliti di Universitas Illinois, yang telah lama bekerja dengan superkomputer, tidak puas dengan berbagai teknik yang dipakai dalam mengembangkan berbagai benchmark. Mereka mengembangkan benchmark yang disebut Perfect Club, yang memiliki pendekatan pengukuran mirip dengan SPEC. Perfect Club merupakan gabungan dari aplikasi-aplikasi riil yang disumbangkan oleh kelompok-kelompok peminat komputasi dan diorganisasi sedemikian rupa sehingga menjadi satu benchmark. Benchmark Perfect Club terutama mengukur kinerja atas bilangan floating-point dan biasanya dieksekusi pada superkomputer. Tujuan utama proyek Perfect Club adalah mengkarakterisasi program aplikasi dalam hal perilaku algoritmanya sehingga memungkinkan pemakai memperoleh prediksi kinerja yang diharapkan untuk program aplikasi yang dikembangkannya (Sharp dan Bacon, 1994:68).

Benchmark iCOMP dikembangkan oleh Intel untuk membandingkan kinerja prosesor-prosesor yang ada di pasaran. Ketika prosesor generasi 486 diperkenalkan, di pasaran muncul berbagai versi mulai 486SX, 486DX2, dan sebagainya. Agar calon pembeli memiliki gambaran ringkas kinerja prosesor, Intel mengembangan angka indeks yang merupakan angka kinerja prosesor dibandingkan dengan kinerja prosesor rujukan. Benchmark ini khusus mengukur kinerja prosesor dan tidak mencerminkan kinerja komputer secara keseluruhan (Sharp dan Bacon, 1994:68). Popularitas iCOMP terutama karena benchmark ini hampir selalu menjadi ukuran kinerja prosesor-prosesor Intel dalam iklan-iklannya (setidaknya sampai generasi Pentium).

Susun!

Kritik terhadap *Benchmark*

Pengukuran kinerja komputer dengan *benchmark* yang ada saat ini banyak dikritik karena seperti orang buta meraba gajah. Bergantung pada bagian yang dipegang, gajah bisa didefinisikan sebagai tinggi seperti pohon kelapa, panjang dan kecil seperti ular, atau lebar dan tipis seperti kipas (Gustafson dan Todi, 1998). Benchmark yang ada cenderung mengukur satu aspek dari kinerja komputer dan hasilnya digunakan untuk menggeneralisasi kinerja keseluruhan.

Menyadari masalah itu, benchmark Perfect Club dan SPEC mengembangkan apa yang disebut *suite* aplikasi (yakni sekumpulan aplikasi terdiri atas elemen-elemen program yang masing-masing mengukur berbagai aspek perilaku sistem agar diperoleh gambaran sistem lebih lengkap). *Suite* semacam itu biasanya sulit dipangkalkan (*porting*) ke komputer lain, terutama komputer paralel, dan bila berhasil dipangkalkan diperlukan waktu berjam-jam untuk mengeksekusinya (Gustafson dan Todi, 1998).

Menurut Gustafson dan Todi (1998) aplikasi semacam itu tetaplah hanya mengukur satu titik sampel dari kinerja komputer sementara yang diperlukan adalah mengukur seluruh rentang kinerja komputer. *Benchmark-benchmark* tersebut memiliki satu karakteristik yang sama, yakni ukuran masalahnya dibuat tetap. Komputer-komputer dibandingkan dari segi berapa lama diperlukan waktu untuk menyelesaikan masalah tersebut. Karena *benchmark* harus dapat dieksekusi pada banyak jenis komputer, maka ukuran masalah dipilih sedemikian rupa sehingga diharapkan semua komputer sasaran uji dapat menjalankan *benchmark* dengan baik. Ukuran masalah pada NAS Parallel Benchmark misalnya, disesuaikan dengan kapasitas komputer Cray X-MP yang ada di NASA Ames Research Center meskipun banyak komputer paralel yang memiliki lebih banyak memeri dan dirancang untuk masalah-masalah yang jauh lebih besar. Meskipun para pengembang NAS Parallel Benchmark telah membuat lima ukuran masalah yang berbeda tetapi tetap tidak menemukan cara untuk membandingkan komputer dengan kapasitas memori yang sangat berbeda. Oleh karena itulah, dukungan terhadap penggunaan NAS Parallel Benchmark makin lama makin surut (Gustafson dan Todi, 1998).

McMahon, yang merancang *benchmark* Livermore Loop menunjukkan upaya awal untuk mengeksplorasi penggunaan ukuran masalah yang berbeda dalam satu *benchmark*. Panjang vektor diubah-ubah pada suatu rentang tertentu untuk merepresentasikan berbagai rejim (konfigurasi dan kapasitas) memori sistem komputer. Hanya saja rentang variasi panjang vektor tidak cukup besar untuk mengakomodasi rejim memori (untuk tahun 1997 saja rasio ukuran yang cukup representatif adalah 1.000.000 : 1) dan *benchmark* Livermore Loop dinyatakan dengan satu skor tunggal sehingga meniadakan perbedaan struktur uji yang dilakukan.

Susun!

Benchmark Whetstone, yang dibuat oleh Curnow dan Wichman, juga dimaksudkan untuk menghindari sindrom 'orang buta dan gajah' dengan memberikan bobot yang dapat diatur untuk aspek-aspek komputasi yang berbeda, misalnya matematika bilangan integer, pemanggilan subrutin, fungsi-fungsi khusus, percabangan, dan sebagainya (Gustafson dan Todi, 1998). Pengguna *benchmark* ini dapat memilih sendiri bobot yang dipakai untuk aplikasi target, mengatur bobot *benchmark*, dan menggunakan skor total sebagai prediktor kinerja komputer tersebut. Meskipun demikian, *benchmark* Whetstone dengan bobot yang telah disediakan di dalamnya sebagai bobot *default* lebih sering digunakan sebagai perbandingan.

Hal lain yang tidak masuk dalam perhitungan para perancang *benchmark* adalah fakta bahwa menurut hukum Moore, kinerja komputer bertambah sebesar 60% setiap tahun. Semua *benchmark* mendasarkan diri pada masalah berukuran tetap (*fixed-size* problem) sehingga relatif terhadap kinerja yang terus meningkat, masalah yang diujikan sebagai pengukur kinerja menjadi sangat kecil dalam beberapa tahun kemudian. *Benchmark* LINPACK, misalnya, mulai dengan menentukan bahwa ukuran matriks harus 100x100. Ketika kinerja komputer bertambah besar dan penghitungan matriks berukuran 100x100 terselesaikan lebih cepat dari waktu yang diperlukan pemrogram untuk menekan tombol ENTER, ukuran masalah diperbesar dengan matriks 300x300 dan kemudian 1000x1000. Bahkan selanjutnya perancang LINPACK mengijinkan versi 'sebesar yang dapat ditampung oleh memori'.

Meski LINPACK berupaya melakukan penskalaan ukuran masalah terhadap peningkatan kinerja, ada hal lain yang juga perlu diperhatikan. Ukuran memori pada sistem komputer meningkat 4 kali lipat setiap 3 tahun. Jumlah operasi yang dijalankan pada sistem komputer meningkat dengan faktor 64. Bila kecepatan pemrosesan juga dianggap mengikuti hukum Moore dengan peningkatan sebesar 4 kali, maka berarti waktu yang diperlukan untuk menjalankan program 'LINPACK terskala' tersebut meningkat dengan faktor 16. Dengan kondisi seperti itu, 'LINPACK terskala' praktis sama dengan LINPACK berukuran tetap, relatif terhadap peningkatan instrinsik sistem komputer tersebut di atas. Satu hal yang pasti adalah bahwa pengukuran kinerja komputer sebenarnya ditentukan oleh batas waktu yang dapat ditolerir manusia untuk menunggu hasil eksekusi program pengukur, dan ini merupakan batas ukuran masalah yang dapat diberikan kepada komputer sebagai penguji kinerja.

Mengembangkan *benchmark* tanpa memperhitungkan laju peningkatan kinerja dapat diibaratkan mematok harga barang tanpa memperhitungkan laju inflasi (dan perlu diingat, 'inflasi' kinerja komputer adalah 60% pertahun). Ketika LINPACK petama kali digunakan, waktu yang diperlukan dengan matriks 100x100 dikumpulkan dari berbagai sistem komputer yang ada di berbagai instutusi dan dilaporkan secara rutin. Pada taraf presisi 32 bit, matriks

Tanggal :
Lokasi :

Susun!

100x100 tersebut hanya memerlukan 40.000 byte memori dan 670.000 operasi *floating-point*.Komputer VAX-11/780 memerlukan beberapa detik untuk menjalankan program tersebut - cukup untuk memberi kesempatan manusia melakukan pengukuran dengan cermat. Komputer CRAY Y-MP8/832 hanya memerlukan waktu eksekusi 1/300 detik, jauh lebih cepat dari kemampuan monitor menyegarkan (*to refresh*) tampilan untuk memberitahukan bahwa proses komputasi telah selesai (Gustafson, dkk, 1996). Bahkan dengan taraf presisi 64 bit, CRAY hanya memerlukan 1/30.000 dari kapasitas memorinya untuk menyelesaikan persoalan itu. Berikutnya, LINPACK dikembangkan dengan matriks 300x300, kemudian 1000x1000. Variasi taraf presisi dan optimisasi yang diijinkan menyebabkan skor *benchmark* LINPACK menjadi relatif, bergantung pada persoalan dan taraf presisi yang digunakan saat pengujian.

Benchmark dengan Pendekatan Lain

Sebagai upaya untuk mengembangkan *benchmark* yang lebih baik, peneliti di Ames Laboratory merancang program yang disebut sebagai SLALOM (*The Scalable, Language-independent, Ames Laboratory One-minute Measurement*). *Benchmark* SLALOM mengukur kinerja komputer dengan pendekatan waktu-tetap (*fixed-time*), bukan ukuran-masalah tetap (*fixed-size problem*). Dengan prinsip waktu-tetap, dimungkinkan membandingkan berbagai jenis komputer, dari komputer personal sampai komputer paralel berkemampuan besar. SLALOM, yang secara otomatis menyesuaikan dengan daya komputasi (*computing power*) yang ada dan memperbaiki kekurangan berbagai *benchmark* sebelumnya, memiliki sifat-sifat: sangat terskala, memecahkan persoalan riil, memperhitungkan juga waktu yang diperlukan untuk unit masukan dan keluaran, dan dapat dijalankan pada komputer paralel serta menggunakan berbagai bahasa yang ada (Gustafson, dkk, 1996).

Meskipun dirancang untuk memanfaatkan memori sesuai dengan kecepatan komputer, SLALOM tidak dapat dijalankan selama satu menit (waktu yang diperlukan untuk pengukuran) pada komputer yang kapasitas memorinya relatif kurang terhadap kecepatannya. Akibatnya, komputer dengan kapasitas memori kecil tidak dapat diukur kinerjanya dengan SLALOM.

Benchmark dengan pendekatan lain juga dibuat oleh peneliti di Ames Laboratory, dikenal dengan nama HINT (*Hierarchial INTegration*). Benchmark ini menghasilkan satuan ukuran yang disebut QUIPS (*Quality Improvement Per Second*) dan tidak menggunakan ukuran-masalah tetap ataupun waktu-tetap. HINT dikembangkan berdasarkan *benchmark* SLALOM tetapi bekerja lebih cepat. Bedanya, HINT tidak mematok ukuran masalah (*problem size*) maupun waktu komputasi. QUIPS merupakan satuan yang digunakan untuk mengukur banyaknya usaha yang dilakukan komputer pada rentang waktu tertentu. Gustafson, peneliti yang mengembangkan HINT, tidak menginginkan waktu yang terlalu singkat untuk melakukan pengukuran

Susun!

karena kebanyakan komputer bekerja sangat cepat pada awalnya, lalu mulai menurun kecepatannya setelah banyak terjadi kemelesetan (*miss*) *cache* dan mulai menggunakan memori utama atau bahkan harus mengakses data pada *harddisk* (Gustafson dan Snell, 1997).

Meskipun belum sepopuler *benchmark* lain, misalnya SPEC95, HINT diklaim sebagai *benchmark* yang memungkinkan pembandingan yang adil terhadap perbedaan-perbedaan ekstrim dalam hal arsitektur komputer, kineja absolut, kapasitas memori, dan taraf presisi komputasi. HINT merupakan perbaikan dari SLALOM dalam hal linieritas (kualitas penyelesaian masalah, pemakaian memori, dan banyaknya operasi, semuanya proporsional), mudah dikonversi ke komputer dengan arsitektur berbeda, serta menyatukan taraf presisi dan ukuran memori ke dalam satu kinerja. Sampai saat ini HINT masih belum banyak diuji dan belum cukup populer untuk digunakan sebagai tolok-ukur kinerja komputer universal.

Penutup

Perancangan pengukur kinerja komputer yang universal semakin sulit dilakukan akibat semakin bervariasinya arsitektur komputer, konfigurasi dan kapasitas memori, taraf presisi sistem, maupun teknik optimisasi kompilator. Berbagai pendekatan ilmiah dilakukan dalam upaya merancang *benchmark* yang memperhitungkan semua aspek di atas, tetapi tetap mudah digunakan. Sampai saat ini, pemakai sendirilah yang harus menentukan *benchmark* pilihannya, sekedar untuk mendapatkan prediksi kinerja komputer bila diberi beban kerja sesuai dengan pemakaian yang sebenarnya. Dengan tetap memperhatikan 'kelemahan' *benchmark* yang dipakai, setidaknya dapat diperoleh gambaran awal kinerja komputer atas beban-kerja yang akan diberikan kepadanya.