Aplikasi Kombinatorial untuk Menentukan Arah Perkembangan Cache

Jonathan Sudibya (13512093)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
jonathans121@students.itb.ac.id

***Abstract*—Pada makalah ini akan dibahas salah satu bahasan studi Matematika Diskrit yaitu kombinatorial. Banyak sekali penggunaan kombinatorial, yang terutama adalah penggunaannya di bidang statistika dan peluang. Namun, pada makalah ini akan lebih ditekankan pada bagaimana aplikasi kombinatorial digunakan untuk menghitung arah perkembangan cache berikutnya. Cache sendiri adalah komponen penting yang meningkatan performa sebuah komputer. Cache digunakan sebagai memori sementara untuk mengolah data sehingga tidak diperlukan waktu yang lama hanya mengakses memori utama atau bahkan tempat penyimpanan utama.**

Di sini akan dibahas bagaimana jenis-jenis cache saat ini dan bagaimana kombinatorial dapat membatu menentukan perkembangan cache selanjutnya.

***Index Terms*—Cache, Cache Miss, Kombinatorial.**

I. Introduction

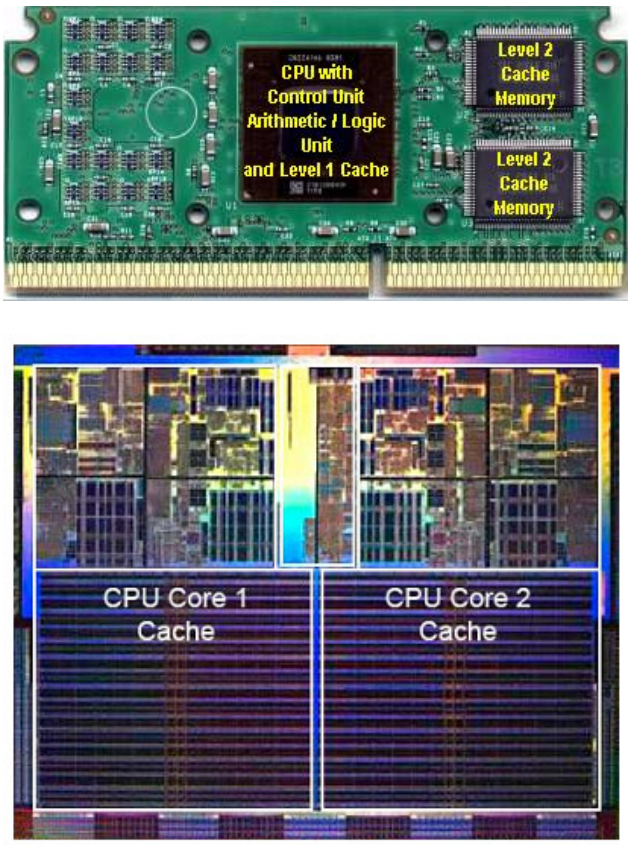
Cache merupakan salah satu tempat penyimpanan data yang terletak sangat dekat dengan CPU (Computer Processing Unit). Berbeda dengan tempat penyimpanan lain, cache tidak bias diakses layaknya memori utama ataupun disk storage. Walaupun tidak bisa diakses pengguna, kecepatan akses cache oleh CPU lebih cepat dari pada akses langsung ke memori utama ataupun ke disk storage sekalipun. Kemampuan cache ini menyebabkan harganya mahal, sehingga cache hanya berukuran kecil dibandingkan yang lain (rata-rata sekitar 3 MB untuk computer standar di rumah).

Jika performa sebuah CPU diukur oleh banyaknya instruksi yang dapat dikerjakan dalam 1 detik, cache diukur dengan banyaknya data yang dapat diakses dalam satu cycle (putaran). Dengan menggunakan teknik kombinatorial dapat dihitung banyaknya putaran yang diperlukan sebuah cache untuk mengakses data yang dimilikinya.

II. Cache

Pada awalnya computer pada umumnya hanya memiliki tiga tingkat penyimpanan, yaitu CPU Register, memori utama dan disk storage. Namun, performa ketiganya sangat jauh sehigga sering terjadi bottleneck di CPU. Untuk menghindari hal tersebut diciptakan sebuah tempat penyimpanan sementara yang disebut Cache.

Cache ini bertipe SRAM (Static Random Access Memory) berbeda dengan memori utama yang bertipe DRAM (Dynamic Random Access Memory). Kesamaan keduanya adalah isinya akan "hilang" ketika tidak dialiri listrik, sehingga tidak cocok untuk menjadi tempat penyimpanan jangka panjang (seperti Disk Storage atau Flash Drive).



Gambar 0. Bentuk fisik Cache

Cache terletak diantara CPU Registers dan memori utama. Besar dari cache tidak terlalu besar, yaitu sekitar 32-64 KB di L1,sekitar 256 KB di L2, dan sekitar 3 MB di L3. Walaupun kecil, namun letaknya yang dekat dengan CPU, menguragi bottleneck yang sering terjadi antara memori utama dan CPU Registers.

Ketika CPU ingin mendapatkan atau mengirim data. Cache mencari apakah data dengan address tertentu telah

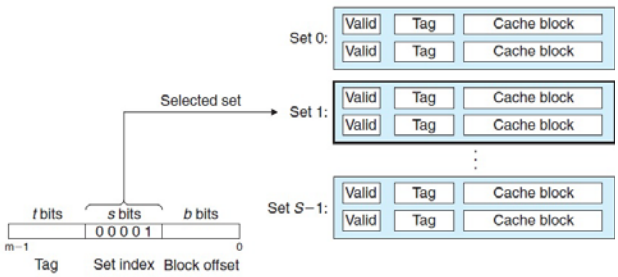
ada di cache. Jika tidak, maka kejadian tersebut disebut cache miss Ada 3 jenis cache miss, yaitu cold miss/compulsory miss, conflict miss, dan capacity miss.

Cold/compulsory miss terjadi ketika cache "kosong" saat akan baca. Maka diputaran kedua cache akan ditulis dengan data yang dibutuhkan.

Conflict miss terjadi ketika salah satu set cache penuh sedangkan data yang diinginkan harusnya berada pada set tersebut. Maka cache harus mengganti salah satu data dengan data baru yang ingin di baca.

Capacity miss terjadi ketika data yang dinginkan tidak cukup untuk disikan ke cache. Biasanya terjadi akibat iterasi data yang cukup besar.

Pemasukkan data ke cache baik dari CPU maupun main memori sama, hanya cache dirancang unik untuk setiap cache 'slot'-nya. Secara umum pemasukkan data ke cache melihat addres si pengirim. Address tersebut dibagi ke dalam 3 sesi, yaitu tag bits, set bits, dan block bits. Ketiganya bergantung pada besarnya cache dan panjangnya address.



Gambar 1 Pemasukan Cache (Cache dengan 2-way associative)

Pemasukkan suatu data ke cache tidak dengan byte per byte, namun seluruh block cache tersebut. Misalkan ada cache yang memiliki 10 byte block. Maka jika hanya ingin menggunakan salah satu dari byte block tersebut, kesepuluh block tersebut harus tetap dibawa ke cache.

Pengisian cache ada berbagai macam cara, 3 yang terbesar adalah direct map, n-way associative, dan full associative. Direct map membuat seluruh baris dari cache menjadi set individu. Sedangkan n-way associative membuat n buah baris menjadi satu set. Full associative menjadikan seluruh cache menjadi 1 set dengan kata lain cache tidak memiliki set (sudah menjadi satu kesatuan).

III. KOMBINATORIAL

A. Prinsip Inklusi-Eksklusi

Prinsip ini menyebabkan beberapa kejadian dimasukan dan sebagian yang lain dibuang dari perhitungan. Contoh nyatanya ketika menghitung jumlah kemungkinan nomor plat untuk sebuah mobil di daerah X. Dengan demikian karena setiap daerah memiliki kode daerah masing-masing maka nomor plat yang dimasukan hanyalah nomor plat yang berkode daerah tersebut dan daerah yang lain tidak dimasukan. Prinsip ini kemudian memiliki dua metode perhitungan yaitu kombinasi dan permutasi.

A.1. Permutasi

Permutasi adalah jumlah urutan berbeda dari pengaturan objek-objek.

$$P(n,r) = \frac{n!}{(n-r)!}$$

A.2 Kombinasi

Kombinasi adalah jumlah pengaturan dari objek-objek tanpa melihat urutannya.

$$C(n,r) = \frac{n!}{(n-r)!\,r!}$$

B.Metode perkalian

Jika ada n percobaaan masing-masing dengan P_i maka hasinya adalah

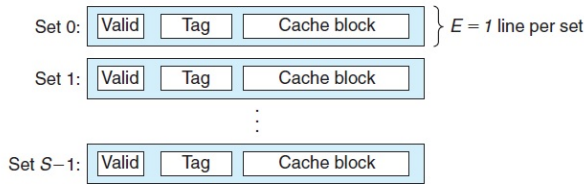
$$X = P1 \times P2 \times P3 \times \ldots \times Pi$$

IV. ANALISA PERKEMBANGAN CACHE

Misalkan cache yang digunakan adalah cache L1-d dengan besar 32 KB (32768 Bytes) dan cache block sebesar 64 byte.

A.Jenis Cache

A.I. Direct Map



Gambar 2. Direct map Cache

Rumus untuk menghitung jumlah set yang dimiliki cache adalah

$$S = \frac{N}{B}$$

S = jumlah set bits
N = besar cache
B = jumlah cache blok

Dengan rumus demikian jumlah set yang dimiliki cache tersebut adalah

$$S = \frac{32768}{64}$$
$$S = 512\ set$$

Dengan demikian cara pemasukaan data ke dalam cache adalah sebagai berikut

B = 64 byte

Maka, block bits yang digunakan adalah 5 bit ($2^{(5+1)}$ = 64). Sedangkan untuk set bits yang digunakan adalah 8 bit ($2^{(8+1)}$ = 512). Dan sisanya digunakan untuk tag bits (19 bit).

Compulsory miss untuk direct map dapat dihitung dengan banyaknya set yang dimiliki oleh cache tersebut. Jadi compulsory miss maksimum yang dimiliki oleh cache ini adalah 512 cold miss.

Conflict miss secara umum adalah banyaknya address yang dapat masuk ke dalam satu set cache (jumlah variasi tag). Maka, banyaknya conflict miss pada cache ini adalah

– – – – – – – – – – – – – – – – – – –

Setiap tag bit hanya memiliki 2 kemungkinan yaitu 0 atau 1. Maka banyaknya kemungkinan untuk tag tersebut adalah

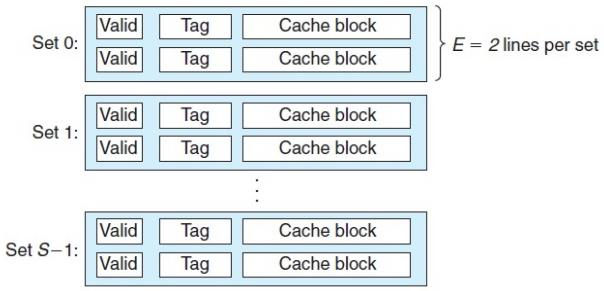
$$X = 2 \times 2 \times 2 \times 2 \times ..\times 2 \text{ (sebanyak 19)}$$

Atau

$$X = 2^{19}$$
$$X = 524288$$

Sedangkan Capacity Miss tidak akan dihitung karena bergantung pada data yang dikerjakan.

A.II. n-way Associative



Gambar 3. 2-way associative cache

Dengan rumus di atas maka jumlah set yang dimiliki n-way associative berubah, yaitu

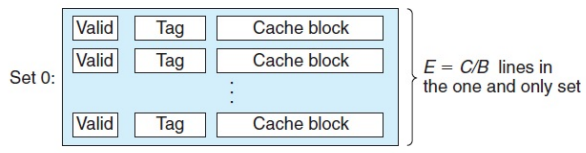
$$S = \frac{32768}{(64 \times n)}$$

Jumlah compulsory miss maksimum tetap sama karena jumlah block size dan cache size tidak berubah. Jadi compulsory miss untuk n-way associative tetap 512.

Jumlah conflict miss maksimum adalah

$$X = 2^{(19+n)} - n$$

A.III. Full Associative



Gambar 4. Full Associative Cache

Jenis Cache ini hanya menggunakan satu set cache. Sehingga cache tersebut layaknya tempat parkir.

Jumlah compulsory miss maksimum yang terjadi adalah 512 (tetap) sedangkan jumlah conflict miss yang mungkin adalah

$$X = 2^{27} - \frac{N}{B}$$

B.Besar Cache dan Besar Block

B.I Besar Cache

Misalkan besar cache diperbesar 2 x menjadi 64 KB (65536 byte). Jika kita analisa efeknya menggunakan direct map, maka jumlah set yang dimilikinya adalah

$$S = \frac{65536}{64}$$
$$S = 1024$$

Cold miss yang terjadi akan meningkat seiring dengan jumlah set yang dimiliki oleh cache tersebut.

Sedangkan untuk conflict miss maksimum yang terjadi adalah

$$X = 2^{18}$$

Sedangkan jika diperkecil 2 x menjadi 16 KB (16384 byte) maka jumlah set dan cold miss yang dimiliki oleh cache tersebut adalah

$$S = \frac{16384}{64}$$
$$S = 256$$

Conflict miss maksimum pada cache ini menjadi

$$X = 2^{20}$$

B.II. Besar Block

Misalkan besar cache diperbesar 2 x menjadi 128 Byte
Maka jumlah set yang dimilikinya adalah

$$S = \frac{32768}{128}$$
$$S = 256$$

Dengan demikian cold miss maksimum yang terjadi akan sebanyak 256. Sedangkan capacity miss untuk direct map adalah sebanyak.

$$X = 2^{20}$$

Misalkan besar cache diperkecil 2 x menjadi 32 byte
Maka jumlah set yang dimiliki adalah

$$S = \frac{32768}{32}$$
$$S = 1024$$

Dengan demikian cold miss yang terjadi maksimum sebanyak 1024. Sedangkan, conflict miss yang terjadi

sebanyak

$$X = 2^{18}$$

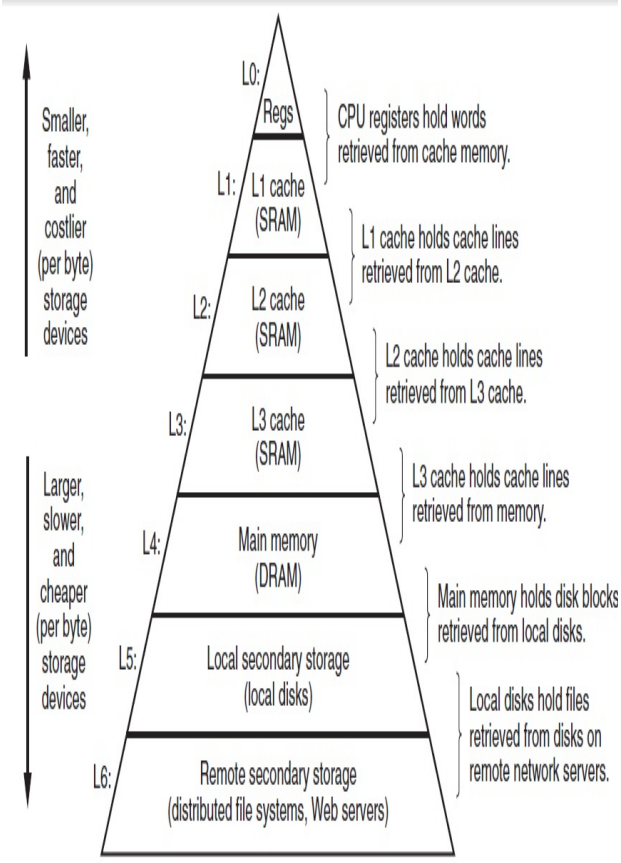
C.Analisa data

Secara umum data di align sesuai dengan Operating System. Maka data yang digunakan sering di align menjadi 32 atau 64 byte. Maka agar semua byte optimal digunakan dalam satu cache saja maka block size yang baik digunakan adalah 32 atau 64 byte.

IV. KESIMPULAN

Dari perhitungan-perhitungan diatas dapat diketahui bahwa secara umum meningkatkan kapasitas cache akan mengurangi miss rate dari (terutama compulsory) cache tersebut. Sedangkan mengecilkan besar block size juga akan memberikan pengaruh yang sama kepada cache tersebut. Sedangkan menggunakan berbagai metode memasukan data. Tampak bahwa cache dengan jumlah set yang banyak akan lebih sedikit mengalami conflict miss dibandingkan yang lain. Namun, dapat diketahui bahwa dengan meningkatnya jumlah associative membantu untuk mengolah data dengan kebutuhan yang berulang dan panjang. Contohnya pada pengolahan array.

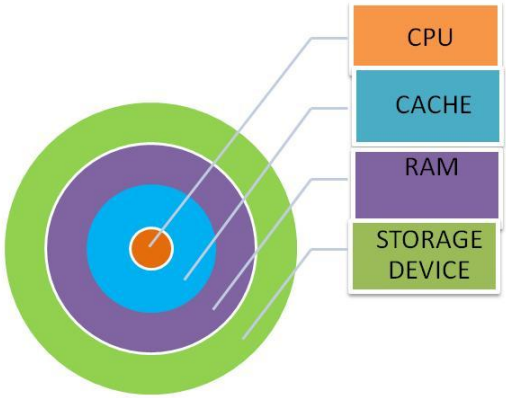
IV. FUN FACT



Gambar 5. Hirarki memori

Type	What cached	Where cached	Latency (cycles)	Managed by
CPU registers	4-byte or 8-byte word	On-chip CPU registers	0	Compiler
TLB	Address translations	On-chip TLB	0	Hardware MMU
L1 cache	64-byte block	On-chip L1 cache	1	Hardware
L2 cache	64-byte block	On/off-chip L2 cache	10	Hardware
L3 cache	64-byte block	On/off-chip L3 cache	30	Hardware
Virtual memory	4-KB page	Main memory	100	Hardware + OS
Buffer cache	Parts of files	Main memory	100	OS
Disk cache	Disk sectors	Disk controller	100,000	Controller firmware
Network cache	Parts of files	Local disk	10,000,000	AFS/NFS client
Browser cache	Web pages	Local disk	10,000,000	Web browser
Web cache	Web pages	Remote server disks	1,000,000,000	Web proxy server

Gambar 6. Kecepatan (cycle) beberapa tempat penyimpanan



Gambar 7. Alokasi memori jika di ilustrasikan sebagai sebuah piringan

REFERENCES

[1] Byrant, O'Hallaron, *Computer Systems : A Programmer's Perspective 2nd edition*. Prentice hall,2011, halaman 559 - 614

[2] Munir, Rinaldi, *Kombinatorial (Slide kuliah)*.

[3] "Secret part of PC Memory : Part 1" http://www.bit-tech.net/hardware/memory/2007/11/15/the_secrets_of_pc_memory_part_1/2; Diakses pada 17 desember 2013 pukul 09.00

[4] Davesh,"How Cache Works", http://www.engineersgarage.com/mygarage/how-cache-memory-works; diakses pada 17 desember 2013 pukul 09.05

[5] "Microprocessor", http://www.kids-online.net/learn/click/details/micropro.html ; diakses pada 17 desember 2013 pukul 09.10

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 17 Desember 2013

Jonathan Sudibya 13512093