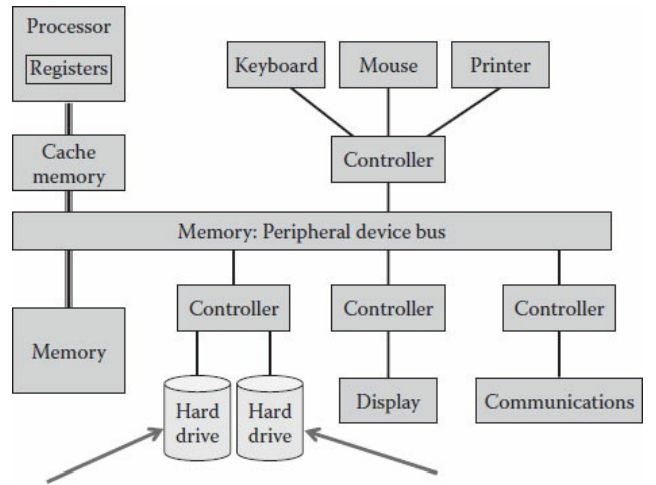
|  |  |  |
| --- | --- | --- |
| **Mata Kuliah** | **:** | **Arsitektur dan Organisasi Komputer** |
| **Bobot Sks** | **:** | **3 sks** |
| **Dosen Pengembang** | **:** | **Catur Nugroho, S.Kom., M.Kom** |
| **Tutor** | **:** |  |
| **Capaian Pembelajaran Mata Kuliah** | **:** | **Mahasiswa mampu menggunakan dan menerapkan konsep & definisi memori eksternal** |
| **Kompetentsi Akhir Di Setiap Tahap (Sub-Cpmk)** | **:** | **Mahasiswa mampu menggunakan dan menerapkan konsep & definisi memori eksternal** |
| **Minggu Perkuliahan Online Ke-** | **:** | **Sesi 6** |

1. **MEMORI EKSTERNAL**

Setiap sistem komputer, seperti makhluk hidup, memiliki sarana untuk menyimpan dan kemudian mengambil data. Seperti pada topik "***Memori***," yang membahas hierarki memori ada berbagai tingkat penyimpanan. Secara umum, baik memori utama atau berbagai level *cache*—dianggap sebagai level utama. Namun, memori ini bersifat fluktuatif dan bersifat sementara dan memelihara data yang disimpan selama tidak diganti atau selama sistem beroperasi. ***Memory volatile*** ini mencakup level pertama dalam hierarki (***Register, Cache***, dan ***Main memory***). Untuk itu, diperlukan perangkat tambahan untuk penyimpanan jangka panjang. Tingkat hierarki berikut adalah untuk media penyimpanan yang tidak perlu dihubungkan ke sumber listrik untuk menjaga data yang disimpan. Selanjutnya, data disimpan jika tidak beroperasi.

Perangkat ***Memory Sekunder*** dibagi menjadi beberapa jenis, berdasarkan akses ke data yang disimpan maupun online, Akses ke perangkat dapat

* ***Serial access***, artinya untuk membaca item data, semua item sebelumnya harus dibaca atau dilewati. Contohnya adalah berbagai perangkat berbasis pita (mirip dengan kaset pita yang digunakan untuk menyimpan informasi audio). Jenis media ini banyak digunakan di masa lalu, terutama untuk tujuan pencadangan, tetapi dalam beberapa tahun terakhir, penggunaannya telah dibatasi.
* ***Random (or direct) access*** di mana dimungkinkan untuk mendapatkan item yang diperlukan secara langsung. Memori, misalnya, adalah perangkat akses langsung. Waktu akses untuk mendapatkan item data menggunakan direct akses hampir serupa untuk semua item, terlepas dari lokasinya.



Gambar6.1. Disk sebagai bagian dari arsitektur

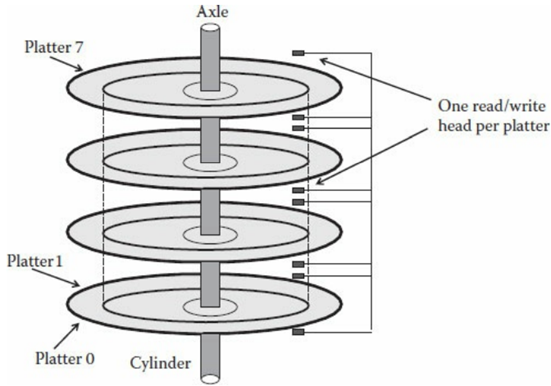
Klasifikasi lain untuk perangkat penyimpanan adalah tingkat ***online*** mereka:

* ***Fully online* (**Sepenuhnya online), artinya perangkat selalu online dan semua datanya selalu tersedia. Sebagian besar disk magnetik yang dipasang di sistem komputer sepenuhnya online.
* ***Partially online*** (Sebagian *online*), yang biasanya melibatkan sistem robot yang menyimpan perpustakaan disk (mis., Disk optik); ketika disk diperlukan, itu akan mengeluarkan perintah untuk mementaskannya. Jenis perangkat ini menyediakan penyimpanan tanpa batas; namun, akses pertama mungkin memerlukan waktu, dan kadang-kadang bahkan waktu yang cukup lama (dalam kasus di mana semua pembaca disk terisi). Seiring kemajuan teknologi dan penurunan harga penyimpanan online, untuk berbagai solusi berbasis cloud, penggunaan dan penyebaran luas penyimpanan online sebagian menurun.
* ***Off-line***, berarti perangkat tidak terhubung ke sistem dan datanya tidak tersedia. Contoh paling sepele mungkin adalah ***disk-on-key***, yang menyimpan data offline, dan ketika diperlukan, disk terhubung ke sistem dan kontennya. Ada organisasi untuk alasan keamanan, menggunakan metode sama tetapi ***hard drive*** yang dapat dilepas. Dalam kedua kasus, ini adalah perangkat offline.

1. **STRUKTUR DISK**

Cakram magnetik kebanyakan memiliki struktur yang sangat mirip (Gambar 9.2):

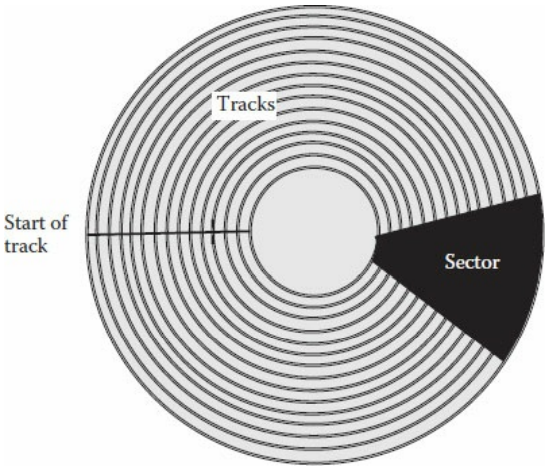
* Disk terdiri dari satu atau lebih piringan bundar pada porosnya, masing-masing dilapisi dengan magnet bahan.
* Masing-masing sisi piringan ini digunakan. Pada disk awal, sisi terluar (bagian atas) piring atas dan bawah piring bawah) tidak digunakan karena lebih banyak masalah fisik terkait dengan kedua sisi tersebut.
* Setiap sisi piringan dibagi menjadi lingkaran konsentris yang disebut trek. Semua trek yang ada di lokasi yang sama pada semua piringan disebut silinder.
* Setiap track dibagi potongan data yang lebih kecil yaitu sektor. Ukuran sektor bervariasi antara 512 byte di disk yang lebih kecil hingga 4096 byte di disk yang lebih besar. Sektor tersebut adalah unit minimal membaca atau menulis ke disk. header dan trailer tambahan, misalnya, *Error Correction Code* (ECC).
* Disk memiliki head *reading/writing* khusus (satu per sisi setiap piringan) yang mampu: membaca dan menulis data. Kepala dipasang pada lengan khusus yang dapat bergerak dari satu melacak ke yang lain. Biasanya, semua kepala terhubung ke rakitan umum sehingga ketika satu kepala bergerak, semua yang lain bergerak secara bersamaan ke silinder yang sama.



Gambar 6.2 Disk structure.

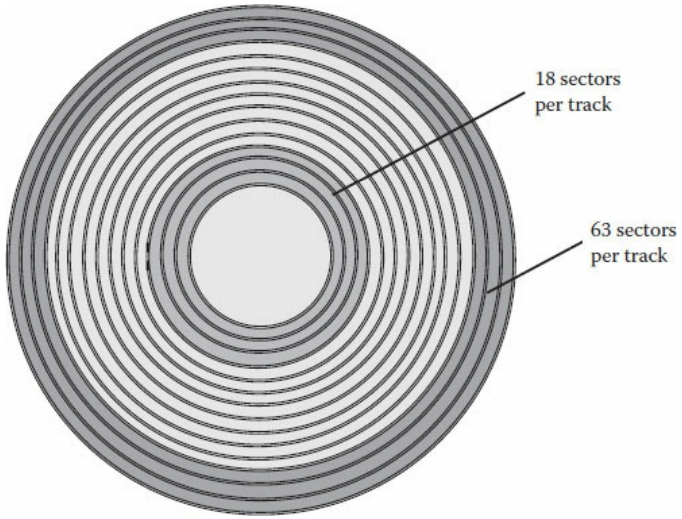
catatan pada disk pertama, hanya sebagian permukaan yang digunakan. Pasalnya, ada perbedaan antara panjang lintasan dalam dan lintasan luar. Disk pertama menggunakan jumlah sektor yang sama per-trek, terlepas dari lokasi trek. Ini dilakukan terutama untuk menyederhanakan tugas pengontrol dalam menghitung alamat sektor. Biasanya, sebuah alamat mencakup nomor disk, nomor silinder, nomor kepala baca atau tulis, dan nomor sektor. Ketika jumlah sektor per trek bervariasi untuk trek yang berbeda, itu akan memerlukan tabel khusus untuk konversi. Untuk mengurangi perbedaan ini, trek direkam hanya di sebagian permukaan. Karena, di satu sisi, teknologi disk maju dan, di sisi lain, persyaratan untuk penyimpanan meningkat, menjadi jelas bahwa mempertahankan sejumlah sektor per trek adalah pemborosan.

Sementara dengan jumlah sektor tetap per trek, perhitungan alamat lebih sederhana, hanya sebagian permukaan yang digunakan, dan trek yang lebih panjang kurang dimanfaatkan. Dengan memberikan pendekatan yang berbeda, jumlah data yang disimpan pada disk dapat ditingkatkan secara dramatis. Pendekatan yang berbeda didasarkan pada peningkatan area yang digunakan untuk trek. Disk permukaan digambarkan secara skematis pada Gambar 6.3.

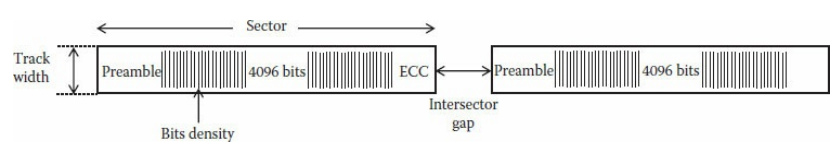


6.3 Tracks and sectors

Solusi yang diterapkan disebut ***Zona Bit Recording*** (ZBR), dan mendefinisikan area yang berbeda (zona) pada disk. Setiap zona mencakup beberapa trek. Trek di zona berbagi sejumlah sektor tetap per-track. *Track* luar, yang lebih panjang, memiliki lebih banyak sektor per trek, sedangkan trek dalam, yang lebih pendek, memiliki lebih sedikit sektor per trek. Tentu saja, mencari sektor menjadi lebih kompleks; namun, algoritme-nya tidak terlalu rumit, dan dilakukan oleh pengontrol. Dalam disk modern, mungkin ada puluhan zona (Gambar 6.4).



Gambar 6.4 Zone bit recording.



Gambar 6.5 Sector structure

Contoh yang ditunjukkan pada Gambar 9.4 menampilkan disk imajiner, zona luarnya memiliki 63 sektor per trek. Zona dalamnya memiliki 18 sektor per trek, dan semua zona di antaranya memiliki nomor yang bervariasi: 18 < n < 63. Sektor memiliki struktur standar (Gambar 9.5), yang dimulai denga pembukaan yang berisi informasi sinkronisasi, seperti alamat sektor. Pembukaan ini menyerupai bit awal dalam transmisi komunikasi asinkron. Pembukaan dibuat saat disk diformat.

Selama operasi baca dan tulis, pengontrol menghitung alamat sektor dan mengirimkan kepala baca dan tulis ke lokasi yang diperlukan. Ketika kepala berada di atas sektor kanan, controller mengeluarkan perintah membaca alamat hanya untuk memastikan bahwa kepala berada di lokasi yang tepat. Jika alamat yang dibaca bukan alamat yang benar, pengontrol mengeluarkan perintah reposisi. Ini dilakukan secara otomatis oleh perangkat keras sementara sistem operasi dan/atau aplikasi tidak menyadarinya masalah. Jika alamat rusak, pengontrol akan terus mencoba untuk mendapatkan alamat yang benar, karena menganggap bahwa masalah mekanis menyebabkan kepala sampai ke lokasi yang salah.

Tepat setelah pembukaan ada 4096 bit (sektor 512-byte). Byte ini berisi data yang disimpan di sektor ini. Setelah data, akan ada ECC untuk memastikan integritas data. Setelah ECC, akan ada celah kecil (intersector gap) diikuti oleh sektor baru.

Biasanya, pembacaan dari atau penulisan ke disk dilakukan dalam data blok. Setiap blok dapat berisi satu/lebih sektor. Bahkan aplikasi mengeluarkan pembacaan untuk blok yang lebih kecil, pengontrol akan membaca setidaknya seluruh sektor. Untuk membaca dari atau menulis ke disk, aplikasi harus menyediakan parameter berikut:

* *Disk number*
* *Cylinder number*
* *Head number* (atau sebagai alternatif nomor permukaan, karena setiap kepala melayani permukaan tertentu)
* *Sector number*
* *Number of bytes to read or write*

Disk adalah salah satu dimensi sektor logis. Sektor-sektor tersebut dipetakan secara berurutan berdasarkan ***silinder,*** dimulai dari sektor nol, yang merupakan sektor pertama pada track pertama (track terluar, dekat dengan keliling platter). Pemetaan berlanjut hingga sektor terakhir dari lintasan, dan kemudian berlanjut dengan sektor pertama di lintasan pertama tetapi di permukaan kedua. Kemudian berlanjut ke trek pertama di permukaan ketiga, dan seterusnya.

Hanya setelah semua track pertama pada semua permukaan telah diakses akan controller kemudian melanjutkan dengan sektor pertama di track kedua dari permukaan pertama. Ide mekanisme ini adalah untuk meminimalkan gerakan heads kepala untuk membaca atau menulis berurutan Salah satu jenis disket yang sudah sangat berguna, namun menghilang karena kelemahan teknologinya, adalah ***floppy disk*** (atau ***disket***). Disk biasa disebut sebagai ***hard drive***, maka ***removable disk*** "baru" disebut ***floppy disk***. Selain nama ada perbedaan kinerja, struktur ***floppy disk*** identik dengan struktur ***hard drive.*** Floppy disk hanya memiliki satu piringan (dua permukaan) dan dua kepala baca dan tulis. Seperti halnya disket biasa, disket mengalami beberapa generasi yang mempengaruhi ukuran dan kapasitas fisiknya.

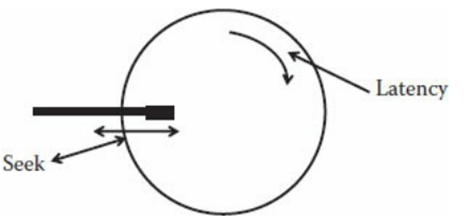
***Floppy disk*** pertama yang menggunakan 8 in. disk dikembangkan oleh IBM dengan tujuan yang sangat berbeda, Selama akhir 1960-an, IBM mengembangkan kegagalan komputer baru (IBM 370) untuk menggantikan keluarga sebelumnya (IBM 360). Salah satu perbedaan utama antara dua desain adalah bahwa keluarga baru menggunakan ***instruksi mikrokode***. Menggunakan mikrokode untuk menentukan instruksi dapat diubah saat sistem berada di situs pelanggan. IBM membutuhkan mekanisme mengirimkan mikrokode (atau firmware), dilakukan dengan menggunakan teknologi "baru", yang sebagian besar merupakan perangkat *seluler* dan *murah*.

Versi ***floppy disk*** yang paling terkenal adalah *disk* 3,5 inci dengan kapasitas 1,44 MB. Disket ini memiliki 80 track dengan 18 sektor *per-track*. Ukuran sektor adalah 512 byte, yang menghasilkan kapasitasnya (2 sisi \* 80 trek per sisi \* 18 sektor per trek \* 512 byte per trek = 1,44 MB). ***Floppy disk*** sangat lambat, dan kecepatan rotasinya enam putaran per-detik. Kemampuannya menyediakan cara sederhana untuk menyimpan dan membawa data, kelemahannya (keandalan, kapasitas, kecepatan transfer lambat) memaksa industri untuk mencari solusi tambahan, dan disket praktis menghilang.

1. **DISK SPEED**

Waktu respons disk ditentukan oleh tiga faktor utama (Gambar 9.6) :

* ***Seek time***, yaitu waktu yang dibutuhkan head (atau rakitan lengan yang menahan head) untuk mencapai silinder yang dibutuhkan.
* ***Latency,*** yaitu waktu yang dibutuhkan disk untuk berputar hingga sektor yang diminta berada di bawah head. Rata-rata, ini adalah separuh waktu yang dibutuhkan untuk satu putaran. Waktu ini secara langsung dipengaruhi oleh kecepatan rotasi disk.
* ***Reading or writing time***, yaitu waktu yang dibutuhkan kepala untuk membaca atau menulis data. Biasanya, kali ini jauh lebih rendah dibandingkan dengan dua kali lainnya. Sementara dua waktu pertama bergantung pada operasi mekanis, waktu membaca atau menulis bergantung pada transfer data elektronik (kepala berada di atas lokasi yang diminta, dan yang harus dilakukan hanyalah membaca atau menulis data).



Gambar 6.5 Sector structure

Misalnya, dengan asumsi

* S adalah ***(seek time)*** waktu pencarian
* L adalah waktu ***latensi*** (*rata-rata setengah putaran*)
* T adalah transfer time

kemudian, keseluruhan waktu yang diperlukan untuk membaca dari atau menulis ke disk ditentukan oleh:



Rumus ini berlaku jika tidak ada permintaan tambahan di disk. Pada kenyataannya, *driver* perangkat disk serta pengontrol harus menanggapi beberapa permintaan, sehingga waktu keseluruhan mungkin lebih lama dan termasuk waktu tunggu juga. Ada beberapa algoritma yang mungkin diterapkan sistem operasi dalam mencoba mengoptimalkan waktu akses, dan dapat memengaruhi waktu tunggu.

Selain itu, meskipun dapat dipahami dari rumus di atas bahwa aktivitas dilakukan secara berurutan, pada pengontrol modern tidak demikian. Disk modern biasanya sebagian tumpang tindih dengan dua gerakan mekanis; namun, aktivitas ketiga hanya dapat dimulai setelah dua aktivitas pertama selesai.

Beberapa sistem operasi (***Windows oleh Microsoft***) tidak meminta pengguna untuk menentukan ukuran file yang ingin dia tulis. Untuk setiap blok tambahan yang ditambahkan ke file, sistem operasi mencari sektor gratis dan tersedia dan menetapkannya ke file. Saat file bertambah, ada kemungkinan yang meningkat bahwa sektor yang baru ditugaskan tidak akan secara fisik dekat dengan sektor sebelumnya. ***Sistem operasi*** menangani situasi ini mungkin memiliki implikasi kinerja yang parah. Jika file ditulis secara berurutan, ada sedikit gerakan kepala. Kepala terletak di atas silinder dan menulis data tanpa gerakan mekanis apa pun. Di sisi lain, jika file ditulis sedemikian rupa sehingga sektor tersebar di seluruh disk, disk diakses secara acak dan kepala harus dipindahkan untuk setiap sektor. Sementara di disk modern, sekuensial.

Membaca (atau menulis) dapat memakan waktu sekitar satu *milidetik*. Jika dilakukan secara acak, itu akan membutuhkan sekitar 20 milidetik, dan terkadang bahkan lebih. Ini tentu saja tergantung pada disk dan atributnya dan jumlah fragmentasi. Fragmen ini, yang mewakili sektor berurutan milik file yang sama tetapi ditulis di lokasi yang berbeda pada disk, adalah produk sampingan dari cara kerja sistem (seperti halnya dalam mengelola *Random Access Memory* [RAM]). Jika sistem tidak memaksakan deklarasi terlebih dahulu tentang ukuran file, kemudian seiring waktu, fragmentasi meningkat. Masalah terkait dengan tingkat fragmentasi yang lebih tinggi adalah meningkatkan waktu respons. Solusi yang diusulkan oleh sistem adalah menjalankan utilitas defrag, yang mencoba menyalin file ke sektor yang berurutan

1. **DI**SK **CAPACITY**

Sebagian besar hard drive yang tersedia saat ini adalah ***disk Winchester*** yang berarti bahwa piringan tertutup rapat baik piringan maupun kepala yang bergerak tidak terlihat dari luar. Ini diperlukan untuk memastikan bahwa debu atau partikel lain tidak akan masuk ke rakitan disk. Materi eksternal semacam itu menimbulkan bahaya nyata bagi disk, mengingat kecepatan putarannya yang tinggi.

Yang pertama dari jenis disk ini dikembangkan oleh IBM pada tahun 1973. Disk, yang disegel, dibagi menjadi dua bagian: bagian tetap dan bagian yang dapat dilepas. Masing-masing dimaksudkan untuk menyimpan 30 MB. Disk dinamai senapan yang terkenal *Winchester* 30-30. Kapasitas disk ditentukan oleh pabrikannya, biasanya mencakup semua bit yang ada pada disk. Namun, sebelum dapat digunakan, disk harus diformat, yang antara lain menulis alamat sektor. Oleh karena itu, kapasitas aktual disk yang dapat digunakan selalu lebih rendah dari kapasitas yang dikutip. Misalnya, pertimbangkan disk dengan karakteristik berikut :

* 8 (23 ) platters
* 2097152 (2 21 ) tracks per surface
* 128 (27 ) sectors per track (rata-rata)
* 4096 (212 ) bytes per sector

Pabrikan akan menggambarkan disk sebagai 8,8 TB B (23 \* 221 \* 27 \* 212 = 243 ). Pada kenyataannya, sekitar 10% dari bit akan digunakan untuk mendengar (*intersector gapes, preamble, ECC*), sehingga dalam situasi optimal, disk akan berisi kurang dari 8 TB, Perkembangan teknologi disk dapat dibagi menjadi dua bagian. Sejak diperkenalkan pada awal 1970-an dan hingga 1990-an, tujuan utamanya adalah untuk meningkatkan kapasitas. Ini dicapai dengan meningkatkan kerapatan perekaman.

Selama waktu densitas perekaman meningkat dengan faktor 40-50, sedangkan kapasitas meningkat dengan faktor 80-100. Hasilnya adalah bahwa disk awal, yang secara fisik besar (ukuran mesin cuci rata-rata), bertambah besar, dan pada 1990-an, mereka seukuran lemari es. Sejak tahun 1990-an, penekanan yang lebih besar diberikan pada pembuatan disk yang lebih kecil. Akibatnya, ada periode pada akhir 1990-an di mana, alih-alih peningkatan kapasitas, itu sebenarnya berkurang, tetapi ukuran disk secara signifikan lebih kecil. Rekaman kepadatan, yang sekitar 3 megabit per inci persegi, meningkat menjadi beberapa ***terabits*** per inci persegi.

Berlawanan dengan perkembangan pesat dalam teknologi disk yang terkait dengan kepadatan perekaman, kecepatan rotasi disk, secara langsung mempengaruhi kinerja disk, telah konstan selama beberapa waktu. Ada disk cepat dengan kecepatan putaran 10.000 rpm; namun, disk yang berputar paling cepat, seperti ***Cheetah by Seagate***, memiliki kecepatan putaran 15.000 rpm. ***Disk Cheetah*** diperkenalkan beberapa tahun yang lalu, dan perkembangan teknologi tambahan terutama di bidang peningkatan kapasitasnya, tetapi kecepatan rotasinya tetap tidak berubah. Keterbatasan ini, dikombinasikan dengan perkembangan ***SolidStateDisk*** (SSD), telah mengubah fokus, dan saat ini untuk perkembangan baru, terutama untuk peralatan seperti tablet dan ponsel, SSD adalah yang umum. Disk ini juga muncul perlahan di sistem komputer biasa, meskipun hanya digunakan untuk tugas yang harus sangat cepat, seperti menyimpan halaman aplikasi. Ini karena masih ada perbedaan biaya per megabyte yang besar antara SSD dan disk biasa.

1. **PENINGKATAN KINERJA**

***Hard drive***, seperti perangkat periferal lainnya, terhubung ke bus komputer melalui pengontrol. Pada tahap awal, pengontrol terutama digunakan untuk menjembatani antara protokol bus dan protokol perangkat. Seiring kemajuan teknologi disk, pengontrol mengambil peran yang lebih penting. Misalnya, dengan penerapan ***zone bit recording*** (ZBR), pengontrol, melalui lapisan logika tambahan, yang mengelola perhitungan alamat. Kecerdasan tambahan ditambahkan untuk mencapai tingkat optimalisasi dan kinerja yang lebih tinggi. Logika baru ini dilakukan oleh perangkat keras (pengontrol) tanpa gangguan pada sistem operasi atau aplikasi.

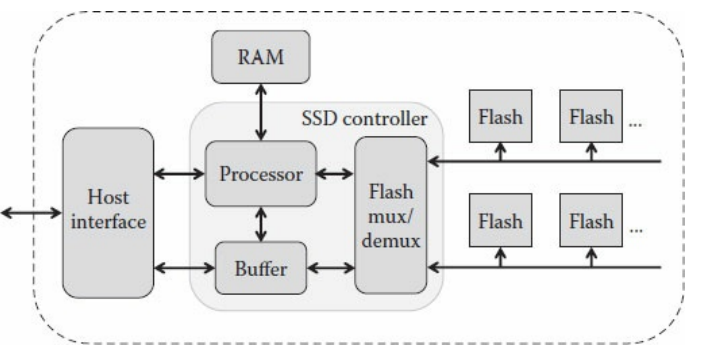
Kerja sama antara pengontrol dan fitur tambahan dalam sistem operasi menghasilkan peningkatan kinerja dan keandalan yang signifikan. Beberapa teknik yang digunakan adalah :

* Blok yang lebih besar: Pengontrol dapat membaca jumlah data yang lebih besar dari disk (dibandingkan dengan jumlah yang diminta oleh aplikasi). Data tambahan disimpan dalam buffer di dalam pengontrol. Saat berikutnya sistem meminta data ini, pengontrol akan mengambilnya tanpa perlu mengakses disk mekanis yang lambat. Ini berarti bahwa pengontrol harus mengelola buffer, karena mungkin pengontrol membaca data berikut tetapi aplikasi tidak membutuhkannya. Dalam hal ini, dan setelah beberapa waktu yang telah ditentukan berlalu, buffer akan dibebaskan sehingga dapat dipindahkan. Kontroler modern sebenarnya mewakili sistem tertanam dengan sistem komputernya sendiri. Perangkat lunak yang disertakan dengan pengontrol terkadang cukup canggih karena banyaknya kasus khusus yang harus ditangani. Salah satu contoh yang dimaksudkan untuk meningkatkan kehandalan disk adalah terkait dengan ***bad sector***. Di masa lalu, jika sektor buruk terdeteksi, itu bisa mencegah sistem menggunakan drive. Sistem operasi mencoba untuk menandai beberapa sektor sebagai sektor buruk sehingga tidak akan mencoba untuk menulis di atasnya. Kontroler modern memiliki area khusus pada disk yang digunakan sebagai cadangan. Setiap kali titik buruk, atau sektor buruk, terdeteksi, pengontrol menggantinya dengan lokasi lain di area cadangannya. Ketika sistem operasi atau aplikasi mencoba mengakses lokasi yang sama, pengontrol akan mengganti alamat asli dengan alamat baru. Ini dilakukan dengan cepat tanpa memberi tahu sistem operasi atau aplikasi. Perangkat lunak yang tertanam di pengontrol bukan bagian dari sistem operasi, dan itu dikembangkan dan dipasok oleh produsen disk.
* *Pre-fetch* (atau baca dulu): Ini sudah dibahas. Pengontrol dapat mengidentifikasi pola perilaku aplikasi. Saat aplikasi membaca sektor secara berurutan, pengontrol dapat membaca blok berikutnya bahkan sebelum diminta. Blok baca-depan ini disimpan sementara di buffer pengontrol. Ide ini berasal dari pemahaman bahwa membaca sektor berikut saat kepala diposisikan dengan benar membutuhkan waktu yang sangat sedikit (tidak perlu mencari dan latensi). Mekanisme ini membutuhkan peningkatan ukuran memori pengontrol.
* ***Access algorithms*** : Ini biasanya diimplementasikan oleh sistem operasi dan disertakan dalam bab ini untuk memberikan penjelasan terperinci dan pandangan yang lebih lengkap tentang metode peningkatan kinerja.
* ***Disk array*** : Ini adalah teknologi yang meningkatkan ukuran disk tunggal serta kecepatan, keandalan, dan kemampuan bertahannya.

1. ***SOLID STATE DISK* (SSD)**

Perkembangan pesat dalam teknologi memori diamati dengan jelas oleh produsen ***hard disk***, yang menghadapi peningkatan permintaan untuk kinerja disk yang ditingkatkan. Salah satu implementasi pertama adalah ***disk-on-key***, yang menggunakan ***flash memori***.\* ***Disk-on-key*** memasuki pasar dalam permintaan dan dengan cepat menggantikan media pemindahan file yang tersedia. Sejauh menyangkut komputer, ada kebutuhan yang berkembang untuk membawa informasi. Bahkan dengan diperkenalkannya laptop, pengguna masih membutuhkan perangkat sederhana untuk menyimpan informasi. Awalnya, ini dilakukan dengan menggunakan disket. Disket,sangat terbatas dan tidak dapat diandalkan jika dilakukan di lingkungan yang terkendali. Untuk waktu singkat, disket digantikan oleh CD, yang masih digunakan untuk musik, tetapi ketika ***disk-on-key*** tiba, media yang paling banyak digunakan untuk menampung file. selanjutnya, berdasarkan pengalaman pengguna, adalah menggunakan teknologi untuk penggantian disk. Perlu dicatat bahwa ada produk yang menerapkan teknologi ini, bahkan di awal 1990-an; Namun, hanya dalam dekade terakhir teknologi telah memasuki komputasi mainstream. Keuntungan signifikan yang terkait dengan SSD bahwa sebenarnya adalah perangkat memori tanpa bagian yang berputar. Dengan demikian, perangkat ini bisa sangat cepat (tanpa *pencarian dan latensi*) dan relatif kecil dengan konsumsi daya yang rendah

***SolidStateDisk***(SSD), seperti disk modern, bukan hanya penyimpanan tetapi juga sistem yang mencakup prosesor, bus, dan memori (Gambar 6.6). Dibandingkan dengan mekanis disk, ***SolidStateDisk***(SSD) sangat cepat, terutama untuk pembacaan secara acak. Seperti dinyatakan sebelumnya, gagasan mengganti disk berputar dengan memori sebagai sarana untuk meningkatkan kinerja bukanlah hal baru, dan telah diimplementasikan, misalnya oleh ***Control Data*** selama tahun 1980-an. Memori saat itu secara fisik besar, sehingga tidak dapat digunakan sebagai perangkat ***mobile*** ***off-line***; memori digunakan sebagai perangkat penyimpanan sistem terutama untuk memutar proses masuk dan keluar serta untuk menyalin segmen memori yang besar.



Gambar 6.6 contoh arsitektur SSD

Perkembangan teknologi baru ***SolidStateDisk***(SSD), dan ukurannya yang kecil mendorongnya dan penggunaannya semakin cepat. Penggunaan utama SSD adalah untuk aplikasi server, di mana waktu respons sangat penting. Misalnya, data yang disimpan di SSD akan menjadi data yang menyebabkan kemacetan, seperti bagian dari ***database, file swap***, dan ***file paging***. Industri komputasi terkunci dalam persepsi disk yang berputar selama lebih dari 60 tahun, dan pengenalan teknologi baru semacam itu membutuhkan pemikiran yang *out-of-the-box*. Pada tahun 2007, ASUS memperkenalkan ***notebook*** dengan ***solid-state disk***. Meskipun bobotnya sangat ringan, ia diterima dengan sangat hati-hati (awalnya ia menjalankan Linux), meskipun beberapa tahun kemudian ia menyertakan disk biasa (berputar) dan menjalankan Windows. Butuh beberapa tahun sebelum menemukan jalannya organisasi teknologi.

Banyak organisasi teknologi, terdapat infrastruktur komputasi yang sangat besar yang mendukung organisasi tersebut. Infrastruktur ini biasanya mencakup sejumlah besar penyimpanan disk, dan menggantinya dengan ***SolidStateDisk***(SSD) baru sangat mahal, karena SSD masih jauh lebih mahal dibandingkan dengan disk mekanis. Di sisi lain, kemungkinan baru teknologi bagi perusahaan yang tidak aktif di industri *mechanical-disk*, seperti Intel, yang saat ini menawarkan *solid-state disk*.

Faktor signifikan lain yang mendukung ***SolidStateDisk***(SSD) adalah konsumsi daya yang lebih rendah dan ketahanannya lebih tinggi terhadap benturan/guncangan; ini membuat lebih tahan untuk laptop, notebook, dan sebagainya. Terlepas dari kelebihannya dan biayanya mahal saat ini ***SolidStateDisk***(SSD) akan digunakan untuk memperluas hierarki memori dengan menambahkan level. Penting untuk dicatat teknologi ***SolidStateDisk***(SSD) memiliki beberapa kelemahan.

Hal ini untuk beberapa waktu, lebih mahal daripada disk mekanis biasa. Selain itu, seperti produk berbasis flash lainnya, disk dapat ditulis dalam jumlah terbatas. untuk penulisan dengan jumlah besar dapat menyebabkan masalah. Ini terjadi karena, seperti disk biasa, ***SolidStateDisk***(SSD) menggantikan ***bad sector*** dengan sektor dari area cadangannya. Ada beberapa aplikasi yang menggunakan penulisan, atau sistem yang sangat sibuk untuk paging dan dapat mencapai batas bawaan, Oleh karena itu, ***SolidStateDisk***(SSD), setidaknya pada kondisi saat ini, tidak dapat digunakan untuk penyimpanan file jangka panjang yang sedang dimodifikasi pada frekuensi tinggi

1. ***ACCE***SS ***ALGORITHM***S

Dalam mengimplementasikan berbagai algoritma akses, sistem operasi bertujuan untuk meningkatkan kinerja disk. Biasanya, faktor dominan yang membatasi kinerja dalam operasi cakram mekanis adalah gerakan lengan (kepala). Jika algoritma untuk mengoptimalkan gerakan dapat diterapkan, akan mengurangi waktu yang dihabiskan untuk menggerakkan lengan, yang akan meningkatkan kinerja disk secara keseluruhan. Inilah alasan mengapa berbagai algoritma dikembangkan.

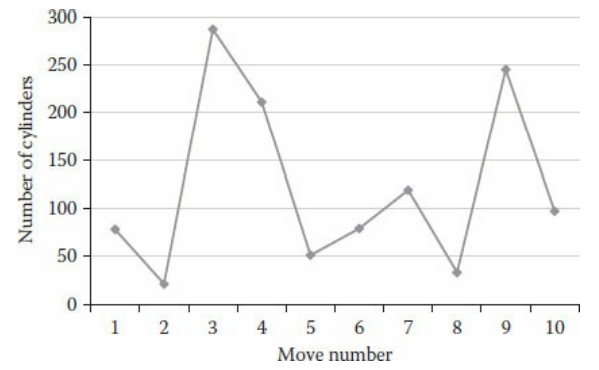
Contoh visual untuk menjelaskan algoritma yang berbeda, di asumsikan bahwa :

* Disk yang digunakan dalam contoh memiliki 300 silinder (0–299)
* Posisi kepala awal adalah silinder 103
* Antrian permintaan mencakup perpindahan ke silinder 78, 21, 287, 211, 51, 79, 119, 3, 245, dan 97

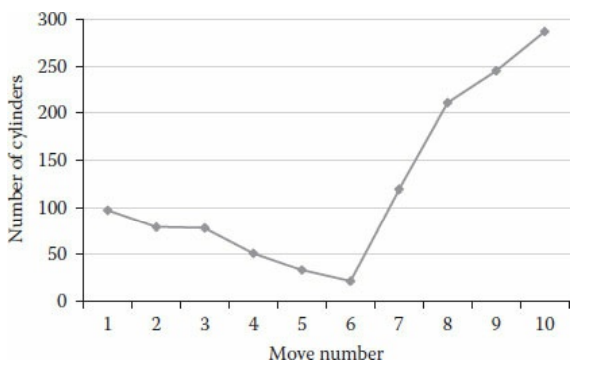
Beberapa algoritma akses yang digunakan untuk meningkatkan kinerja meliputi.

* ***First come first served*** (FCFS), menangani permintaan berdasarkan pesanan. Permintaan pertama akan dilayani terlebih dahulu, diikuti oleh permintaan kedua, dan seterusnya. Dalam kasus khusus ini, kepala pertama akan pindah ke silinder 78 dan, setelah menyelesaikan permintaan ini, itu akan pindah ke silinder 21 dan seterusnya, seperti yang terlihat pada Gambar 6.7. Jumlah silinder yang digerakkan lengan untuk memenuhi permintaan adalah 1098.
* ***Shortest time first*** (STF), di mana permintaan ditangani berdasarkan jaraknya dari lokasi kepala saat ini. Permintaan pertama yang ditangani adalah yang paling dekat. Dalam contoh khusus ini, permintaan pertama yang diproses adalah permintaan yang berhubungan dengan silinder 97 (paling dekat dengan lokasi awal—103). Permintaan berikutnya adalah yang mengakses silinder 79, dan seterusnya, seperti yang digambarkan pada Gambar 6.8. Sebelum gerakan kepala berikutnya, sistem operasi menghitung gerakan minimal dan membuat daftar prioritas. Dalam kasus khusus ini, karena kepala berada di silinder 103.

daftar akan terdiri dari



Gambar 6.7. FCFS head movement.



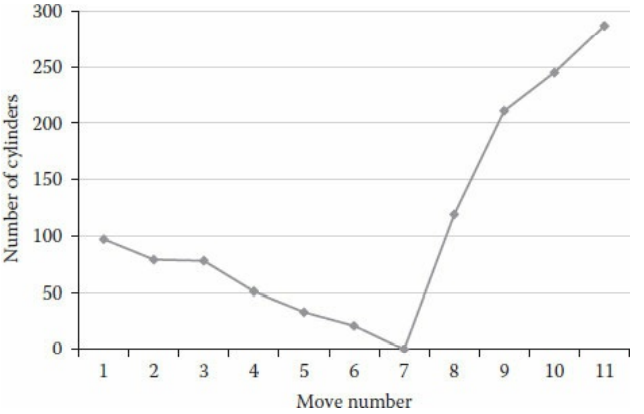
Gambar 6.8. STF head movement



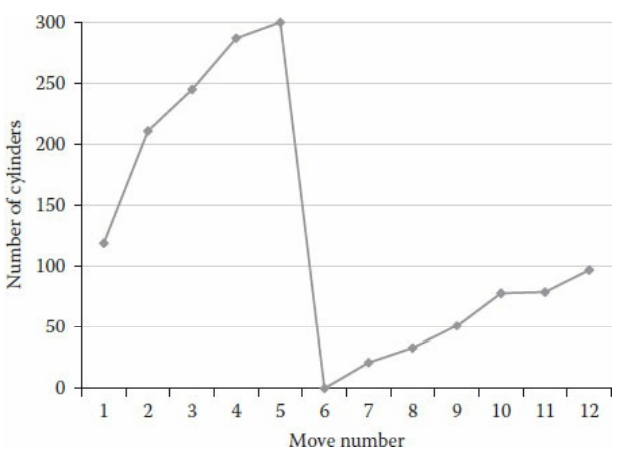
Untuk menyelesaikan semua permintaan, *head* harus menggerakkan 266 silinder, yang menyiratkan bahwa algoritma ini lebih efisien daripada algoritma FCFS sebelumnya. Sayangnya, menerapkan algoritme apa adanya, tanpa tindakan pencegahan tambahan mungkin dapat mengakibatkan ***unfulfilled***, di mana beberapa permintaan tidak terpenuhi. Ini terjadi jika banyak permintaan tertunda semuanya dalam jarak dekat, dan permintaan baru juga untuk silinder yang dekat, tetapi ada satu permintaan untuk silinder yang jauh. Permintaan menunggu selamanya, karena akan selalu ada permintaan yang lebih dekat untuk dilayani lebih awal.

***Scan*** (atau algoritma elevator), di mana ***head*** dimulai di salah satu ujung disk (silinder pertama) dan, saat bergerak ke ujung yang lain, menangani semua permintaan yang ditemukan di jalan. Sebagai bagian dari algoritme, sistem operasi mengurutkan permintaan dan menanganinya sesuai dengan urutan yang diurutkan. Ketika kepala mencapai ujung disk, ia mulai bergerak mundur mengikuti mekanisme yang sama, hanya saja kali ini nomor yang diurutkan ditangani dari yang terbesar ke bawah. Dengan asumsi kepala bergerak menuju awal disk, maka permintaan pertama yang ditangani adalah yang mengakses silinder 97, diikuti oleh 79, dan seterusnya hingga mencapai awal disk. Kemudian akan berubah arah, menuju ke ujung disk dan dimulai dengan permintaan yang mengakses silinder 119, diikuti oleh 211, dan seterusnya, seperti yang dijelaskan pada Gambar 6.9. Untuk menyelesaikan semua permintaan, kepala harus bergerak lebih dari 372 silinder.

C-Scan (atau pemindaian melingkar, Gambar 6.10), yang sangat mirip dengan SCAN dengan sedikit perubahan. Kepala dimulai dari satu ujung (biasanya di awal disk) dan, saat bergerak ke ujung disk, menangani semua permintaan yang ditemukannya. Setelah sampai ke akhir disk, itu dimulai sekali lagi dari awal. Dalam kasus khusus ini, karena kepala berada di silinder 103 bergerak maju, permintaan pertama yang dilayani adalah silinder 119, diikuti oleh 211, 245, dan 287. Kemudian, kepala akan pindah ke ujung piringan, kembali ke awal, dan mulai maju mencari permintaan di jalannya. Jumlah total silinder yang digerakkan adalah 594 (termasuk 300 silinder dalam perjalanan kembali dari ujung piringan ke awal).

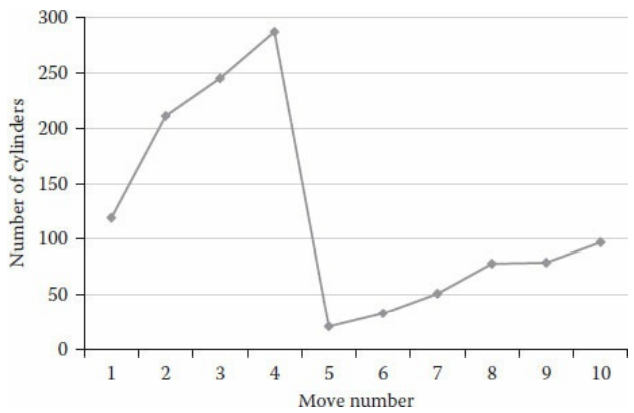


Gambar 6.9. Scan head movement.



Gambar 6.10 C-Scan head movement.

C-Look (Gambar 6.11) merupakan peningkatan dari C-Scan yang dijelaskan sebelumnya. Tidak seperti C-Scan, yang bergerak dari awal disk ke akhir (bahkan jika tidak ada permintaan tambahan di dekat akhir), di C-Look, kepala bergerak maju sampai menemukan permintaan dengan nomor silinder yang lebih besar. Maknanya tidak ada permintaan tambahan untuk silinder yang letaknya lebih jauh, jadi tidak perlu maju-maju, dan head bisa kembali ke awal. Namun, ketika pindah ke awal disk, karena pengontrol mengetahui silinder mana yang diminta, itu tidak harus memindahkan kepala ke awal disk tetapi ke silinder yang diminta pertama. Dalam contoh khusus ini, setelah kepala mencapai silinder 287, kepala tidak melanjutkan gerakannya ke ujung piringan tetapi langsung menuju silinder 21. Jumlah total silinder yang dipindahkan dalam hal ini adalah 526.



Gambar 6.11C-Look head movement.

1. **DISK CONTROLLER**

***Disk controller*** bertanggung jawab atas beberapa peningkatan kinerja. Salah satu aktivitas ini adalah mengelola buffer internal serta seluruh struktur memori, termasuk menyimpan data dan mengeluarkannya perintah *read-ahead* untuk data yang akan diambil sebelumnya. Dalam banyak kasus, pengontrol mengimplementasikan pembacaan ke depan dengan mengalokasikan beberapa buffer dan menggunakan buffer ganda atau mekanisme buffer melingkar. Dalam arti, mirip dengan mekanisme pipa yang digunakan oleh prosesor untuk meningkatkan kinerja karena eksekusi paralel. Saat aplikasi memproses satu blok data, blok berikutnya sudah tersedia di buffer pengontrol, dan blok tambahan mungkin sedang dimuat dari disk.

Misalnya, dengan asumsi

* + P adalah waktu pemrosesan satu blok
  + R adalah jumlah waktu yang diperlukan untuk membaca satu blok
  + n adalah jumlah blok yang akan diproses (baca atau tulis)

Maka total waktu yang diperlukan untuk pekerjaan serial menggunakan buffer tunggal diberikan oleh :

Namun, ketika bekerja dengan dua buffer atau lebih (dengan asumsi R > P), total waktu diberikan oleh :

Ini menyerupai pipa prosesor, di mana pembacaan sebenarnya dari blok berikutnya dilakukan sementara unit pemrosesan pusat (CPU) memproses yang sebelumnya. Seperti halnya pipeline prosesor, dalam hal ini dimungkinkan karena dua komponen berbeda (buffer) terlibat. Kegiatan paralel ini secara drastis dapat mengurangi waktu aplikasi. Manfaat tambahan, yang telah disebutkan, menggunakan pengontrol terkomputerisasi adalah kemampuannya untuk meningkatkan keandalan.

Meskipun disk tertutup rapat, mereka berputar dengan kecepatan yang sangat tinggi (hingga 15.000 rotasi per menit dalam kasus disk Cheetah). Karena itu, terkadang ada titik cacat yang menyebabkan bad sector. Bintik-bintik buruk ini disebabkan oleh masalah manufaktur atau masalah yang terjadi selama operasi. Sektor buruk seperti itu biasanya merupakan area pada disk yang tidak dapat diakses.

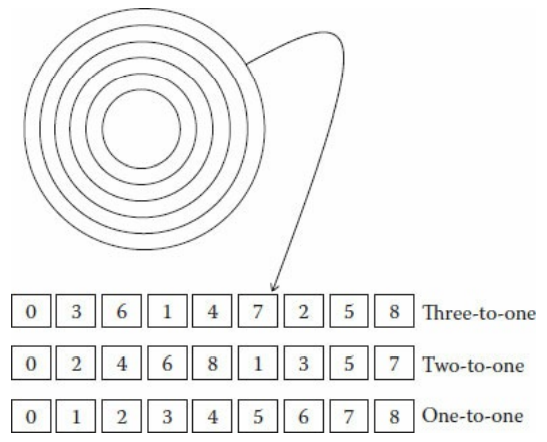
Ini mungkin berisi beberapa data, dan dalam kasus seperti itu, file yang berisi data yang ada di bad sector tidak akan terbaca (*corrupted file*). Biasanya, sebagai bagian dari proses jaminan kualitas, disk diuji sebelum meninggalkan fasilitas manufaktur. Salah satu tes ini adalah untuk memastikan bahwa tidak ada sektor buruk atau bahwa mereka jumlahnya kurang dari ambang batas yang ditentukan oleh produsen.

Bertahun-tahun yang lalu, ada label yang melekat pada disk dengan informasi yang relevan, termasuk jumlah bad sector yang dimilikinya. Saat ini, semua disk menyertakan beberapa sektor cadangan tambahan di setiap trek. Setiap kali ada bad sector ditemukan, secara otomatis dipetakan ke lokasi yang berbeda. Lokasi baru ini adalah salah satu dari sektor cadangan tambahan. Ini dilakukan sebagai bagian dari proses pembuatan dan pengujian, dan itu berlanjut selama operasi normal. Pengontrol menyimpan informasi pemetaan pada silinder (tersembunyi) terpisah, yang digunakan untuk tujuan itu dan bukan merupakan bagian dari kapasitas disk yang dihitung.

Menggunakan mekanisme ini, semua disk meninggalkan fasilitas manufaktur tanpa bad sector. Sebagian besar disk, dan terutama yang modern, menggunakan akses memori langsung (DMA), yang dijalankan oleh pengontrol. Untuk lebih memahami perkembangan teknologi di daerah itu, penting untuk dicatat bahwa pada awalnya sektor-sektor tersebut tidak ditulis secara berurutan.

Prosesor yang menggunakan input/output (I/O) terprogram tidak cukup cepat untuk menangani satu sektor saat disk berputar. Ini berarti prosesor membaca sektor pertama dan kemudian harus menulisnya ke dalam memori. Pada saat prosesor siap untuk sektor kedua, disk sudah diputar, dan prosesor harus menunggu rotasi baru.

Untuk memperbaiki masalah, sektor-sektor dipecah untuk memberikan waktu tambahan. Teknik ini disebut interleaving (seperti halnya memory interleaving), dan dapat dilakukan dengan menggunakan beberapa variasi (Gambar 6.12). Gambar 6.12 menunjukkan metode *interleaving* menggunakan tiga variasi. Bagian atas mewakili disk dan trek yang direkam, sedangkan sisi bawah mewakili tiga contoh. Contoh pertama, yang menunjukkan interleaving tiga-ke-satu, menggunakan perbedaan tiga. Alih-alih perekaman berurutan (satu-ke-satu) yang normal, dalam hal ini sektor-sektor direkam sehingga, sementara prosesor menangani sektor pertama, disk terus berputar. Ketika prosesor siap untuk mengakses sektor berikutnya, dua sektor telah berlalu, dan sektor berikutnya yang berurutan siap untuk membaca. Dengan kata lain, prosesor memiliki lebih banyak waktu untuk menangani sektor baca. Jumlah waktu ditentukan oleh waktu yang dibutuhkan disk untuk memindahkan dua sektor.



Gambar 6.12 *Sector interleaving*.

Dalam *interleaving* dua-ke-satu, perbedaan waktu lebih pendek. Sementara prosesor menangani sektor baca, sektor berikutnya berputar. Ketika prosesor siap untuk sektor berikutnya, itu tersedia untuk dibaca.

Sistem modern yang menggunakan metode lain untuk I/O (DMA atau I/O berbasis interupsi) cukup cepat untuk membaca dan menangani sektor tanpa penundaan (interleaving satu-ke-satu), terutama karena pengenalan pengontrol cerdas dan penggunaan transfer DMA, sehingga interleaving tidak diperlukan lagi.

**REFERENSI/DAFTAR PUSTAKA**

1. Andrew S. Tanenbaum, Structured Computer Organization Fifth Edition, Pearson Prentice Hall 2005
2. Willam Stallings, Organisasi&ArsitekturKomputerEdisikeenam, Prentice Hall 2003
3. Syahrul, Organisasi dan Arsitektur Komputer, Andi offset 2010,