Bab 4

PENJADWALAN PROSES

4.1. Pengertian dan Sasaran Penjadwalan Proses

Penjadwalan proses merupakan kumpulan kebijaksanaan dan mekanisme di sistem operasi yang berkaitan dengan urutan kerja yang dilakukan sistem komputer.

Adapun penjadwalan bertugas memutuskan :

a. Proses yang harus berjalan

b. Kapan dan selama berapa lama proses itu berjalan

Kriteria untuk mengukur dan optimasi kinerje penjadwalan :

a. Adil (fairness)

Adalah proses-proses yang diperlakukan sama, yaitu mendapat jatah waktu pemroses yang sama dan tak ada proses yang tak kebagian layanan pemroses sehingga mengalami kekurangan waktu.

b. Efisiensi (eficiency)

Efisiensi atau utilisasi pemroses dihitung dengan perbandingan (rasio) waktu sibuk pemroses.

c. Waktu tanggap (response time)

Waktu tanggap berbeda untuk :

c.1 Sistem interaktif

Didefinisikan sebagai waktu yang dihabiskan dari saat karakter terakhir dari perintah dimasukkan atau transaksi sampai hasil pertama muncul di layar. Waktu tanggap ini disebut terminal response time.

c.2 Sistem waktu nyata

Didefinisikan sebagai waktu dari saat kejadian (internal atau eksternal) sampai instruksi pertama rutin layanan yang dimaksud dieksekusi, disebut event response time.

d. Turn around time

Adalah waktu yang dihabiskan dari saat program atau job mulai masuk ke sistem sampai proses diselesaikan sistem. Waktu yang dimaksud adalah waktu yang dihabiskan di dalam sistem, diekspresikan sebagai penjumlah waktu eksekusi (waktu pelayanan job) dan waktu menunggu, yaitu : Turn arround time = waktu eksekusi + waktu menunggu.

e. Throughput

Adalah jumlah kerja yang dapat diselesaikan dalam satu unit waktu. Cara untuk mengekspresikan throughput adalah dengan jumlah job pemakai yang dapat dieksekusi dalam satu unit/interval waktu.

Kriteria-kriteria tersebut saling bergantung dan dapat pula saling bertentangan sehingga tidak dimungkinkan optimasi semua kriteria secara simultan.

Contoh : untuk memberi waktu tanggap kecil memerlukan penjadwalan yang sering beralih ke antara proses-proses itu. Cara ini meningkatkan overhead sistem dan mengurangi throughput.

Oleh karena itu dalam menentukan kebijaksanaan perancangan penjadwalan sebaiknya melibatkan kompromi diantara kebutuhan-kebutuhan yang saling bertentangan. Kompromi ini bergantung sifat dan penggunaan sistem komputer.

Sasaran penjadwalan berdasarkan kriteria-kriteria optimasi tersebut :

a. Menjamin tiap proses mendapat pelayanan dari pemroses yang adil.
b. Menjaga agar pemroses tetap dalam keadaan sibuk sehingga efisiensi mencapai maksimum. Pengertian sibuk adalah pemroses tidak menganggur, termasuk waktu yang dihabiskan untuk mengeksekusi program pemakai dan sistem operasi.
c. Meminimalkan waktu tanggap.
d. Meminimalkan turn arround time.
e. Memaksimalkan jumlah job yang diproses persatu interval waktu. Lebih besar angka throughput, lebih banyak kerja yang dilakukan sistem.

4.2 Tipe Penjadwalan

Terdapat 3 tipe penjadwal berada secara bersama-sama pada sistem operasi yang kompleks, yaitu:

1. Penjadwal jangka pendek (short term scheduller)

Bertugas menjadwalkan alokasi pemroses di antara proses-proses ready di memori utama. Penjadwalan dijalankan setiap terjadi pengalihan proses untuk memilih proses berikutnya yang harus dijalankan.

2. Penjadwal jangka menengah (medium term scheduller)

   Setelah eksekusi selama suatu waktu, proses mungkin menunda sebuah eksekusi karena membuat permintaan layanan masukan/keluaran atau memanggil suatu system call. Proses-proses tertunda tidak dapat membuat suatu kemajuan menuju selesai sampai kondisi-kondisi yang menyebabkan tertunda dihilangkan. Agar ruang memori dapat bermanfaat, maka proses dipindah dari memori utama ke memori sekunder agar tersedia ruang untuk proses-proses lain. Kapasitas memori utama terbatas untuk sejumlah proses aktif.

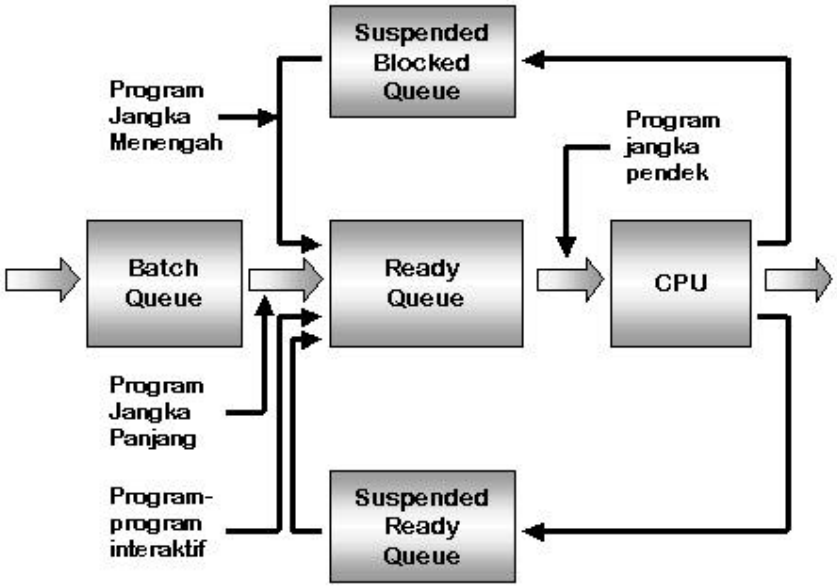
   Aktivitas pemindahan proses yang tertunda dari memori utama ke memori sekunder disebut swapping. Proses-proses mempunyai kepentingan kecil saat itu sebagai proses yang tertunda. Tetapi, begitu kondisi yang membuatnya tertunda hilang dan proses dimasukkan kembali ke memori utama dan ready.

3. Penjadwal jangka panjang (long term scheduller)

   Penjadwal ini bekerja terhadap antrian batch dan memilih batch berikutnya yang harus dieksekusi. Batch biasanya adalah proses-proses dengan penggunaan sumber daya yang intensif (yaitu waktu pemroses, memori, perangkat masukan/keluaran), program-program ini berprioritas rendah, digunakan sebagai pengisi (agar pemroses sibuk) selama periode aktivitas job-job interaktif rendah.

Sasaran penjadwalan berdasarkan tipe-tipe penjadwalan :

a. Memaksimumkan kinerja untuk memenuhi satu kumpulan kriteria yang diharapkan.
b. Mengendalikan transisi dari suspended to ready (keadaan suspend ke ready) dari proses-proses swapping.
c. Memberi keseimbangan job-job campuran.



Gambar 4.1 : Tipe-tipe penjadwalan

4.3 Strategi penjadwalan

Terdapat dua strategi penjadwalan, yaitu :

1. Penjadwalan nonpreemptive (run to completion)

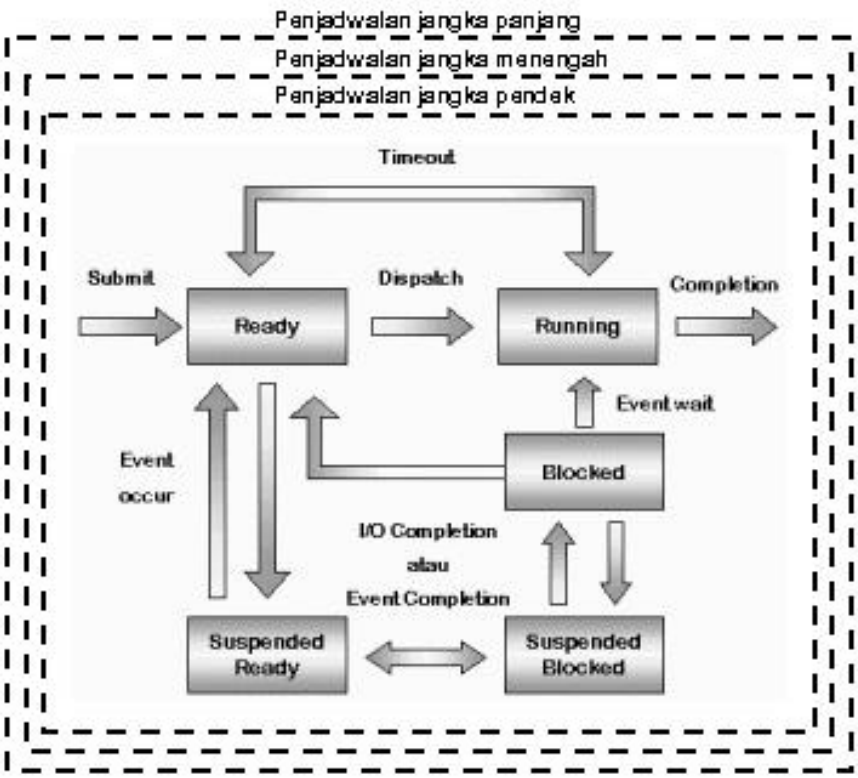
   Proses diberi jatah waktu oleh pemroses, maka pemroses tidak dapat diambil alih oleh proses lain sampai proses itu selesai.

2. Penjadwalan preemptive

   Proses diberi jatah waktu oleh pemroses, maka pemroses dapat diambil alih proses lain, sehingga proses disela sebelum selesai dan harus dilanjutkan menunggu jatah waktu pemroses tiba kembali pada proses itu. Berguna pada sistem dimana proses-proses yang mendapat perhatian/tanggapan pemroses secara cepat, misalnya :

   a. Pada sistem realtime, kehilangan interupsi (tidak layani segera) dapat berakibat fatal.
   b. Pada sistem interaktif, agar dapat menjamin waktu tanggap yang memadai. Penjadwalan secara preemptive baik tetapi harus dibayar mahal. Peralihan proses memerlukan overhead (banyak tabel yang dikelola). Supaya efektif, banyak proses harus berada di memori utama sehingga proses-proses tersebut dapat segera running begitu diperlukan. Menyimpan banyak proses

tak running benar-benar di memori utama merupakan suatu overhead tersendiri.



Gambar 4.2 : Tipe-tipe penjadwalan dikaitkan dengan diagram state

4.4 Algoritma-algoritma Penjadwalan

Berikut jenis-jenis algoritma berdasarkan penjadwalan :

1. Nonpreemptive, menggunakan konsep :

a. FIFO (First In First Out) atau FCFS (First Come First Serve)
b. SJF (Shortest Job First)
c. HRN (Highest Ratio Next)
d. MFQ (Multiple Feedback Queues)

2. Preemptive, menggunakan konsep :

a. RR (Round Robin)

b. SRF (Shortest Remaining First)

c. PS (Priority Schedulling)

d. GS (Guaranteed Schedulling)

Klasifikasi lain selain berdasarkan dapat/tidaknya suatu proses diambil secara paksa adalah klasifikasi berdasarkan adanya prioritas di proses-proses, yaitu :

1. Algoritma penjadwalan tanpa berprioritas.
2. Algoritma penjadwalan berprioritas, terdiri dari :
   a. Berprioritas statik
   b. Berprioritas dinamis

4.5 Algoritma Preemptive

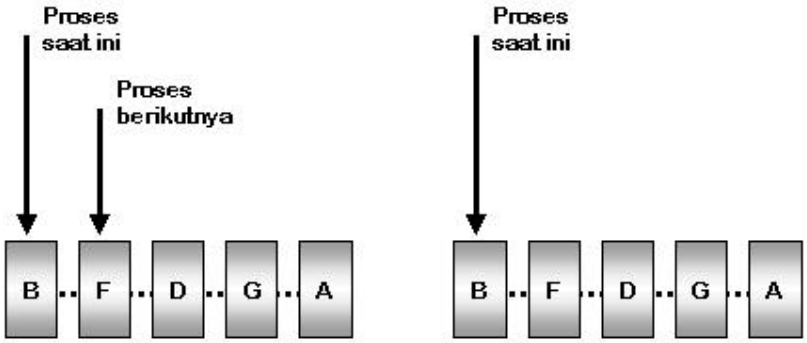
A. Round Robin (RR)

Merupakan :

- ☑ Penjadwalan yang paling tua, sederhana, adil,banyak digunakan algoritmanya dan mudah diimplementasikan.
- ☑ Penjadwalan ini bukan dipreempt oleh proses lain tetapi oleh penjadwal berdasarkan lama waktu berjalannya proses (preempt by time).
- ☑ Penjadwalan tanpa prioritas.
- ☑ Berasumsi bahwa semua proses memiliki kepentingan yang sama, sehingga tidak ada prioritas tertentu.

Semua proses dianggap penting sehingga diberi sejumlah waktu oleh pemroses yang disebut kwanta (quantum) atau time slice dimana proses itu berjalan. Jika proses masih running sampai akhir quantum, maka CPU akan mempreempt proses itu dan memberikannya ke proses lain.

Penjadwal membutuhkannya dengan memelihara daftar proses dari runnable. Ketika quantum habis untuk satu proses tertentu, maka proses tersebut akan diletakkan diakhir daftar (list), seperti nampak dalam gambar berikut ini :



Gambar 4.3

(a) : Daftar proses runnable.

(b) : Daftar proses runnable sesudah proses b habis quantumnya.

Algoritma yang digunakan :

1. Jika kwanta habis dan proses belum selesai, maka proses menjadi runnable dan pemroses dialihkan ke proses lain.
2. Jika kwanta belum habis dan proses menunggu suatu kejadian (selesainya operasi I/O), maka proses menjadi blocked dan pemroses dialihkan ke proses lain.
3. Jika kwanta belum habis tetapi proses telah selesai, maka proses diakhiri dan pemroses dialihkan ke proses lain.

Diimplementasikan dengan :

1. Mengelola senarai proses ready (runnable) sesuai urutan kedatangan.
2. Ambil proses yang berada di ujung depan antrian menjadi running.
3. Bila kwanta belum habis dan proses selesai, maka ambil proses di ujung depan antrian proses ready.
4. Jika kwanta habis dan proses belum selesai, maka tempatkan proses running ke ekor antrian proses ready dan ambil proses di ujung depan antrian proses ready.

Masalah yang timbul adalah menentukan besar kwanta, yaitu :

☑ Kwanta terlalu besar menyebabkan waktu tanggap besar dan turn arround time rendah.

☑ Kwanta terlalu kecil menyebabkan peralihan proses terlalu banyak sehingga menurunkan efisiensi proses.

Switching dari satu proses ke proses lain membutuhkan kepastian waktu yang digunakan untuk administrasi, menyimpan, memanggil nilai-nilai register, pemetaan memori, memperbaiki tabel proses dan senarai dan sebagainya. Mungkin proses switch ini atau konteks switch membutuhkan waktu 5 msec disamping waktu pemroses yang dibutuhkan untuk menjalankan proses tertentu.
Dengan permasalahan tersebut tentunya harus ditetapkan kwanta waktu yang optimal berdasarkan kebutuhan sistem dari hasil percobaan atau data historis. Besar kwanta waktu beragam bergantung beban sistem. Apabila nilai quantum terlalu singkat akan menyebabkan terlalu banyak switch antar proses dan efisiensi CPU akan buruk, sebaliknya bila nilai quantum terlalu lama akan menyebabkan respon CPU akan lambat sehingga proses yang singkat akan menunggu lama. Sebuah quantum sebesar 100 msec merupakan nilai yang dapat diterima.

Penilaian penjadwalan ini berdasarkan kriteria optimasi :

· Adil

Adil bila dipandang dari persamaan pelayanan oleh pemroses.

· Efisiensi

Cenderung efisien pada sistem interaktif.

· Waktu tanggap

Memuaskan untuk sistem interaktif, tidak memadai untuk sistem waktu nyata.

· Turn around time

Cukup baik.

· Throughtput

Cukup baik.

Penjadwalan ini :

a. Baik untuk sistem interactive-time sharing dimana kebanyakan waktu dipergunakan menunggu kejadian eksternal.
   Contoh : text editor, kebanyakan waktu program adalah untuk menunggu keyboard, sehingga dapat dijalankan proses-proses lain.
b. Tidak cocok untuk sistem waktu nyata apalagi hard-real-time applications.

B. Priority Schedulling (PS)

Adalah tiap proses diberi prioritas dan proses yang berprioritas tertinggi mendapat jatah waktu lebih dulu (running). Berasumsi bahwa masing-masing proses memiliki prioritas tertentu, sehingga akan dilaksanakan berdasar prioritas yang dimilikinya. Ilustrasi yang dapat memperjelas prioritas tersebut adalah dalam komputer militer, dimana proses dari jendral berprioritas 100, proses dari kolonel 90, mayor berprioritas 80, kapten berprioritas 70, letnan berprioritas 60 dan seterusnya. Dalam UNIX perintah untuk mengubah prioritas menggunakan perintah nice.

Pemberian prioritas diberikan secara :

a. Statis (static priorities)

Berarti prioritas tidak berubah.

Keunggulan :

- ☑ Mudah diimplementasikan.
- ☑ Mempunyai overhead relatif kecil.

Kelemahan :

- ☑ Tidak tanggap terhadap perubahan lingkungan yang mungkin menghendaki
- ☑ penyesuaian prioritas.

b. Dinamis (dynamic priorities)

Merupakan mekanisme untuk menanggapi perubahan lingkungan sistem beroperasi. Prioritas awal yang diberikan ke proses mungkin hanya berumur pendek setelah disesuaikan ke nilai yang lebih tepat sesuai lingkungan.

Kelemahan :

- Implementasi mekanisme prioritas dinamis lebih kompleks dan mempunyai overhead lebih besar. Overhead in diimbangi dengan peningkatan daya tanggap sistem.

Contoh penjadwalan berprioritas :

Proses-proses yang sangat banyak operasi masukan/keluaran menghabiskan kebanyakan waktu menunggu selesainya operasinya masukan/keluaran. Proses-proses ini diberi prioritas sangat tinggi sehingga begitu proses memerlukan pemroses segera diberikan, proses akan segera memulai permintaan masukan/keluaran berikutnya sehingga menyebabkan proses blocked menunggu selesainya operasi masukan/keluaran. Dengan demikian pemroses dapat

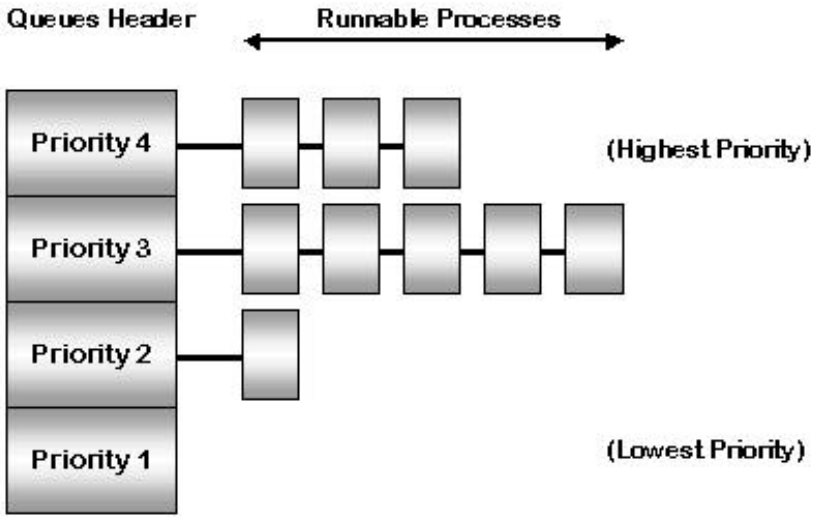
dipergunakan proses-proses lain. Proses-proses I/O berjalan paralel bersama proses-proses lain yang benar-benar memerlukan pemroses, sementara proses-proses I/O itu menunggu selesainya operasi DMA.
Proses-proses yang sangat banyak operasi I/O-nya, kalau harus menunggu lama untuk memakai pemroses (karena prioritas rendah) hanya akan membebani memori, karena harus disimpan tanpa perlu proses-proses itu dimemori karena tidak selesai-selesai menunggu operasi masukan dan menunggu jatah pemroses.

Dalam algoritma berprioritas dinamis dituntun oleh keputusan untuk memenuhi kebijaksanaan tertentu yang menjadi tujuan. Layanan yang bagus adalah menset prioritas dengan nilai 1/f, dimana f adalah ration kwanta terakhir yang digunakan proses.
Contoh :

· Proses yang menggunakan 2 msec kwanta 100 ms, maka prioritasnya50.
· Proses yang berjalan selama 50 ms sebelum blocked berprioritas 2.
· Proses yang menggunakan seluruh kwanta berprioritas 1.

Kebijaksanaan yang diterapkan adalah jaminan proses-proses mendapat layanan adil dari pemroses dalam arti jumlah waktu pemroses yang sama. Keunggulannya penjadwalan berpriorita adalah memenuhi kebijaksanaan yang ingin mencapai maksimasi suatu kriteria diterapkan. Algoritma ini dapat dikombinasikan, yaitu dengan mengelompokkan proses-proses menjadi kelas-kelas prioritas. Penjadwalan berprioritas diterapkan antar kelas-kelas proses itu.

Algoritma penjadwal akan menjalankan : proses runnable untuk prioritas 4 lebih dulu secara round robin, apabila kelas 4 semua sudah diproses, selanjutnya akan menjalankan proses runnable untuk prioritas 3 secara round robin, apabila kelas 3 semua sudah diproses (habis), selanjutnya akan menjalankan proses runnable untuk prioritas 2 secara round robin, dan seterusnya, seperti dalam gambar berikut :



Gambar 4.4 : Skedul algoritma denga empat klas prioritas

C. Multiple Feedback Queues (MFQ)

Merupakan :

☑ Penjadwalan berprioritas dinamis

Penjadwalan ini untuk mencegah (mengurangi) banyaknya swapping dengan proses-proses yang sangat banyak menggunakan pemroses (karena menyelesaikan tugasnya memakan waktu lama) diberi jatah waktu (jumlah kwanta) lebih banyak dalam satu waktu. Penjadwalan ini juga menghendaki kelas-kelas prioritas bagi proses-proses yang ada. Kelas tertinggi berjalan selama satu kwanta, kelas berikutnya berjalan selama dua kwanta, kelas berikutnya berjalan empat kwanta, dan seterusnya.

Ketentuan yang berlaku adalah sebagai berikut

☑ Jalankan proses pada kelas tertinggi.

☑ Jika proses menggunakan seluruh kwanta yang dialokasikan, maka diturunkan kelas prioritasnya.

☑ Proses yang masuk untuk pertama kali ke sistem langsung diberi kelas tertinggi.

Mekanisme ini mencegah proses yang perlu berjalan lama swapping berkali-kali dan mencegah proses-proses interaktif yang singkat harus menunggu lama.

D. Shortest Remaining First (SRF)

Merupakan :

- ☑ Penjadwalan berprioritas.dinamis.
- ☑ Adalah preemptive untuk timesharing
- ☑ Melengkapi SJF

  Pada SRF, proses dengan sisa waktu jalan diestimasi terendah dijalankan, termasuk proses-proses yang baru tiba.
- ☑ Pada SJF, begitu proses dieksekusi, proses dijalankan sampai selesai.
- ☑ Pada SRF, proses yang sedang berjalan (running) dapat diambil alih proses baru dengan sisa waktu jalan yang diestimasi lebih rendah.

Kelemahan :

- ☑ Mempunyai overhead lebih besar dibanding SJF. SRF perlu penyimpanan waktu layanan yang telah dihabiskan job dan kadang-kadang harus menangani peralihan.
- ☑ Tibanya proses-proses kecil akan segera dijalankan.
- ☑ Job-job lebih lama berarti dengan lama dan variasi waktu tunggu lebih lama dibanding pada SJF.

SRF perlu menyimpan waktu layanan yang telah dihabiskan , menambah overhead. Secara teoritis, SRF memberi waktu tunggu minimum tetapi karena overhead peralihan, maka pada situasi tertentu SFJ bisa memberi kinerja lebih baik dibanding SRF.

E. Guaranteed Scheduloing (GS)

Penjadwalan ini memberikan janji yang realistis (memberi daya pemroses yang sama) untuk membuat dan menyesuaikan performance adalah jika ada N pemakai, sehingga setiap proses (pemakai) akan mendapatkan 1/N dari daya pemroses CPU. Untuk mewujudkannya, sistem harus selalu menyimpan informasi tentang jumlah waktu CPU untuk semua proses sejak login dan juga berapa lama pemakai sedang login. Kemudian jumlah waktu CPU, yaitu waktu mulai login dibagi dengan n, sehingga lebih mudah menghitung rasio waktu CPU. Karena jumlah waktu pemroses

tiap pemakai dapat diketahui, maka dapat dihitung rasio antara waktu pemroses yang sesungguhnya harus diperoleh, yaitu 1/N waktu pemroses seluruhnya dan waktu pemroses yang telah diperuntukkan proses itu.

Rasio 0,5 berarti sebuah proses hanya punya 0,5 dari apa yang waktu CPU miliki dan rasio 2,0 berarti sebuah proses hanya punya 2,0 dari apa yang waktu CPU miliki. Algoritma akan menjalankan proses dengan rasio paling rendah hingga naik ketingkat lebih tinggi diatas pesaing terdekatnya. Ide sederhana ini dapat diimplementasikan ke sistem real-time dan memiliki penjadwalan berprioritas dinamis.

4.6 Algoritma Nonpreemptive

A. First In First Out (FIFO)

Merupakan :

· Penjadwalan tidak berprioritas.

FIFO adalah penjadwalan paling sederhana, yaitu :

· Proses-proses diberi jatah waktu pemroses berdasarkan waktu kedatangan.

· Pada saat proses mendapat jatah waktu pemroses, proses dijalankan sampai selesai.

Penilian penjadwalan ini berdasarkan kriteria optimasi :

· *Adil*

Adil dalam arti resmi (proses yang datang duluan akan dilayani lebih dulu), tapi dinyatakan tidak adil karena job-job yang perlu waktu lama membuat job-job pendek menunggu. Job-job yang tidak penting dapat membuat job-job penting menunggu lama.

· *Efisiensi*

Sangat efisien.

· *Waktu tanggap*

Sangat jelek, tidak cocok untuk sistem interaktif apalagi untuk sistem waktu nyata.

· *Turn around time*

Jelek.

· *Throughtput*

Jelek.

FIFO jarang digunakan secara mandiri, tetapi dikombinasikan dengan skema lain, misalnya : Keputusan berdasarkan prioritas proses. Untuk proses-pross berprioritas sama diputuskan berdasarkan FIFO.

Penjadwalan ini :

a. Baik untuk sistem batch yang sangat jarang berinteraksi dengan pemakai.
   Contoh : aplikasi analisis numerik, maupun pembuatan tabel.
b. Sangat tidak baik (tidak berguna) untuk sistem interaktif, karena tidak memberi waktu tanggap yang baik.
c. Tidak dapat digunakan untuk sistem waktu nyata (real-time applications).

B. Shortest Job First (SJF)

Penjadwalan ini mengasumsikan waktu jalan proses sampai selesai diketahui sebelumnya. Mekanismenya adalah menjadwalkan proses dengan waktu jalan terpendek lebih dulu sampai selesai, sehingga memberikan efisiensi yang tinggi dan turn around time rendah dan penjadwalannya tak berprioritas.

Contoh :

Terdapat empat proses (job) yaitu A,B,C,D dengan waktu jalannya masing-masing adalah 8,4,4 dan 4 menit. Apabila proses-proses tersebut dijalankan, maka turn around time untuk A adalah 8 menit, untuk B adalah 12, untuk C adalah 16 dan untuk D adalah 20. Untuk menghitung rata-rata turn around time seluruh proses adalah dengan menggunakan rumus :

$$(4a + 3b + 2c + 1d) / 4$$

Dengan menggunakan rumus, maka dapat dihitung turn around time-nya sebagai berikut (belum memperhatikan shortest job first, lihat gambar a) :

$$= (4a + 3b + 2c + 1d) / 4$$
$$= (4x8 + 3x4 + 2x4 + 1x4) / 4$$
$$= (32 + 12 + 8 + 4) / 4$$
$$= 56 / 4$$
$$= 14 \text{ menit}$$

Apabila keempat proses tersebut menggunakan penjadwalan shortest job fisrt (lihat gambar b), maka turn around time untuk B adalah 4, untuk C adalah 8, untuk D

adalah 12 dan untuk A adalah 20, sehingga rata-rata turn around timenya adalah sebagai berikut :

= (4a + 3b + 2c + 1d) / 4
= (4x4 + 3x4 + 2x4 + 1x8) / 4
= (16 + 12 + 8 + 8) / 4
= 44 / 4
= 11 menit

Tidak memperhatikan SJF

```
Posisi   : a     b   c   d
Priority : 4     3   2   1
Job      : A     B   C   D

        +-----------------+
        :  8  : 4 : 4 : 4 :
        +-----------------+
```

(a)

Memperhatikan SJF

```
  a   b   c    d
  4   3   2    1
  B   C   D    A

      +-----------------+
      : 4 : 4 : 4 :  8  :
      +-----------------+
```

(b)

Jelas bahwa a memberikan nilai kontribusi yang besar, kemudian b, c dan d. Karena SJF selalu memperhatikan rata-rata waktu respon terkecil, maka sangat baik untuk proses interaktif. Umumnya proses interaktif memiliki pola, yaitu menunggu perintah, menjalankan perintah, menunggu perintah dan menjalankan perintah, begitu seterusnya.

Masalah yang muncul adalah :

· Tidak mengetahui ukuran job saat job masuk.
  Untuk mengetahui ukuran job adalah dengan membuat estimasi berdasarkan kelakukan sebelumnya.
· Proses yang tidak datang bersamaan, sehingga penetapannya harus dinamis.
  Penjadwalan ini jarang digunakan, karena merupakan kajian teoritis untuk pembandingan turn around time.

C. Highest Ratio Next (HRN)

Merupakan :

- ☑ Penjadwalan berprioritas dinamis.
- ☑ Penjadwalan untuk mengoreksi kelemahan SJF.
- ☑ Adalah strategi penjadwalan dengan prioritas proses tidak hanya merupakan fungsi waktu layanan tetapi juga jumlah waktu tunggu proses. Begitu proses mendapat jatah pemroses, proses berjalan sampai selesai.

Prioritas dinamis HRN dihitung berdasarkan rumus :
Prioritas = (waktu tunggu + waktu layanan) / waktu layanan
Karena waktu layanan muncul sebagai pembagi, maka job lebih pendek berprioritas lebih baik, karena waktu tunggu sebagai pembilang maka proses yang telah menunggu lebih lama juga mempunyai kesempatan lebih bagus.
Disebut HRN, karena waktu tunggu ditambah waktu layanan adalah waktu tanggap, yang berarti waktu tanggap tertinggi yang harus dilayani.

4.7 Variasi yang diterapkan pada sistem waktu nyata (real time)

Karena sistem waktu nyata sering mempunyai deadline absolut, maka penjadwalan dapat berdasarkan deadline. Proses yang dijalankan adalah yang mempunyai deadline terdekat. Proses yang lebih dalam bahaya kehilangan deadline dijalankan lebih dahulu. Proses yang harus berakhir 10 detik lagi mendapat prioritas di atas proses yang harus berakhir 10 menit lagi.
Penjadwalan in disebut Earliest Deadline First (EDF).

4.8 Schedulling mechanism VS schedulling policy

Ada perbedaan antara schedulling mechanism dengan schedulling policy.
Skedul algoritma adalah dengan pemakaian nilai-nilai dalam parameter, dimana nilai-nilai parameter tersebut dapat diisi (set/change) oleh sebuah proses.
Kernel menggunakan algoritma schedulling priority dengan menyediakan sebuah system call dimana sebuah proses dapat diset dan diubah prioritasnya.

Metode ini dapat membantu proses induk (parent process) sehingga dapat mengontrol skedul anak prosesnya (child process). Disini mekanismenya adalah dalam kernel dan policy adalah penetapan nilai (set) oleh proses pemakai.

KUIS :

1. Definisikan perbedaan antara penjadwalan secara preemptive dan non-preemptive? dan berikan contoh-contohnya penjadwalan tersebut?

2. Jelaskan cara kerja dari penjadwalan Multiple Feedback Quenes (MFQ) dan Round Robin (RR)?