BAB II

TINJAUAN PUSTAKA

2.1 Algoritma Penjadwalan

Penjadwalan CPU (*Central Processing Unit*) adalah pemilihan proses dari antrian *ready* untuk dapat dieksekusi. Penjadwalan CPU merupakan konsep dari *multiprogramming*, dimana CPU digunakan secara bergantian untuk proses yang berbeda (Fauzi, 2013). Jenis-jenis algoritma berdasarkan penjadwalan dibagi menjadi dua (Dicky Faizal, t. t) yaitu:

- *Preemptive*

Algoritma penjadwalan memungkinkan beberapa proses yang sedang *running*, bisa dihentikan sementara. Algoritma ini bertujuan agar sistem lebih responsif dan dapat mengerjakan proses yang lain secara bergantian. Yang termasuk algoritma *preemptive scheduling* menggunakan konsep :

1. FIFO (*First In First Out*) atau FCFS (*First Come First Serve*)
2. SJF (*Shortest Job First*)
3. HRN (*Hight Ratio Next*)
4. MFQ (*Multiple Feedback Queues*)

- *Non-Preemptive*

Suatu algoritma penjadwalan dimana proses yang sedang *running* tidak bisa dihentikan sementara, jadi harus *running* terus sampai selesai. Yang termasuk algoritma *non preemptive scheduling* meggunakan konsep:

1. RR (*Round Robin*)

2. SRF (*Short Remaining First*)

3. PS (*Priority Scheduling*)

4. GS (*Guaranteed Scheduling*)

2.2 Kriteria Penjadwalan

Terdapat enam buah kriteria penjadwalan yang dipakai sebagai pertimbangan pemakaian sebuah penjadwalan (Novalianty, 2004), yaitu:

- **Adil (*fairness)*** adalah dimana proses-proses diperlakukan sama yaitu mendapat jatah waktu pemroses yang sama dan tidak ada proses yang tidak kebagian layanan pemroses sehingga mengalami *starvation*. Sasaran penjadwalan harus menjamin tiap proses mendapat pelayanan dari pemroses yang adil.
- **Efisiensi atau utilisasi pemroses** dihitung dengan perbandingan (rasio) waktu sibuk pemroses. Sasaran penjadwalan adalah menjaga agar pemroses tetap dalam keadaan sibuk sehingga efisiensi mencapai maksimum.
- **Waktu tanggap (*response time*)** adalah waktu yang dibutuhkan oleh suatu proses dari minta dilayani hingga ada respon pertama yang menanggapi permintaan tersebut. Sasaran penjadwalan adalah meminimalkan waktu tanggap.
- ***Turnaround time*** adalah waktu yang dihabiskan dari saat program atau *job* mulai masuk ke sistem sampai proses diselesaikan sistem. Sasaran penjadwalan adalah meminimalkan *turnaround time*.

- ***Waiting time*** adalah waktu yang diperlukan oleh suatu proses untuk menunggu di *ready queue*. *Waiting time* ini tidak mempengaruhi eksekusi proses dan penggunaan I/O.
- ***Throughput*** adalah jumlah kerja yang dapat diselesaikan dalam satu unit waktu. Lebih tinggi angka *throughput*, lebih banyak kerja yang dilakukan sistem.

2.3 *Shortest Job First Scheduling*

Algoritma *Shortest Job First Scheduling* sangat optimal (Nugrahanto, 2002), karena memberikan rata–rata waktu tunggu lebih kecil dibandingkan algoritma penjadwalan yang lain dengan cara memindahkan *job–job* pendek di depan *job-job* yang panjang, sehingga akan mengurangi waktu tunggu. Untuk memperjelasnya dapat dilihat di contoh berikut.

Misalkan ada empat *job* yaitu A, B, C, D masing-masing waktu kedatangan sama, yaitu pada t = 0, dan lama proses *job* berturut-turut : 8, 4, 4, 4.

Tabel 2.1 Contoh Penjadwalan *Shortest Job First* Jika t = 0

Proses	Waktu
A	8
B	4
C	4
D	4

Jika urutan pengerjaannya :

a. *Job* A, B, C, D

b. *Job* B, C, D, A

Maka proses pengerjaannya adalah sebagai berikut:

(a)

8 A	4 B	4 C	4 D

(b)

4 B	4 C	4 D	8 A

Dengan pengerjaan *job* berdasarkan urutan (a) maka berturut-turut waktu yang dibutuhkan untuk proses A, B, C, D adalah 8, 12, 16, 20 sehingga dapat dihitung waktu rata-rata = (8 + 12 + 16 + 20) / 4 = 14.

Bila *job* yang dikerjakan berdasarkan (b), yaitu dengan *shortest job first*, maka waktu yang dibutuhkan untuk proses B, C, D, A adalah 4, 8, 12, 20 atau rata-rata = (4 + 8 + 12 + 20) / 4 = 11.

Pada algoritma ini setiap proses yang ada di *ready queue* akan dieksekusi berdasarkan *burst time* terkecil (Fajaryanti, 2005). Hal ini mengakibatkan *waiting time* yang pendek untuk setiap proses dan karena hal tersebut maka *waiting time* rata-ratanya juga menjadi pendek, sehingga dapat dikatakan bahwa algoritma ini adalah algoritma yang optimal.

Burst time adalah asumsi berapa lama sebuah proses membutuhkan CPU (*Central Processing Unit*) di antara proses menunggunya I/O. Hal ini tidak dapat diprediksi secara tepat sebelum dimulainya sebuah proses. Artinya jumlah waktu yang dibutuhkan sebuah proses dalam menggunakan CPU dalam sebuah satuan waktu. Sebuah proses dapat menggunakan CPU selama beberapa kali selama *task* yang diberikan belum diselesaikan.

Adapula beberapa kekurangan dari algoritma ini yaitu:

1. Susahnya untuk memprediksi *burst time* proses yang akan dieksekusi selanjutnya.
2. Proses yang mempunyai *burst time* yang besar akan memiliki *waiting time* yang besar pula karena yang dieksekusi terlebih dahulu adalah proses dengan *burst time* yang lebih kecil.

2.4 Round Robin Scheduling

Penjadwalan *round robin* adalah penjadwalan proses yang menerapkan strategi *preemptive*, bukan di-*preempt* oleh proses lain tapi terutama oleh penjadwal berdasarkan jatah waktu pemroses yang disebut kwanta (Novalianty, 2004). Ketentuannya adalah jika kwanta habis dan proses belum selesai, maka pemroses dialihkan ke proses lain dan jika kwanta belum habis tapi proses telah selesai, maka proses diakhiri dan pemroses dialihkan ke proses lain.

Round robin merupakan salah satu algoritma penjadwalan yang paling sederhana untuk proses dalam sistem operasi (Tintus Sony Pratama, 2013). *Round robin* ditugaskan untuk membagi waktu setiap proses pada porsi yang sama dan dalam urutan melingkar, menjalankan semua proses tanpa prioritas (dikenal juga sebagai eksekutif siklik). Penjadwalan *round robin* itu sederhana, mudah diterapkan, dan bebas *starvation*. Penjadwalan *round robin* juga dapat diterapkan untuk masalah penjadwalan lainnya, seperti penjadwalan paket data dalam jaringan komputer.

Round robin dirancang untuk sistem *time sharing*. Algoritma ini mirip dengan penjadwalan FCFS (*First Come First Served*), namun *preemption* ditambahkan untuk *switch* (peralihan proses) antara proses. Setiap proses akan

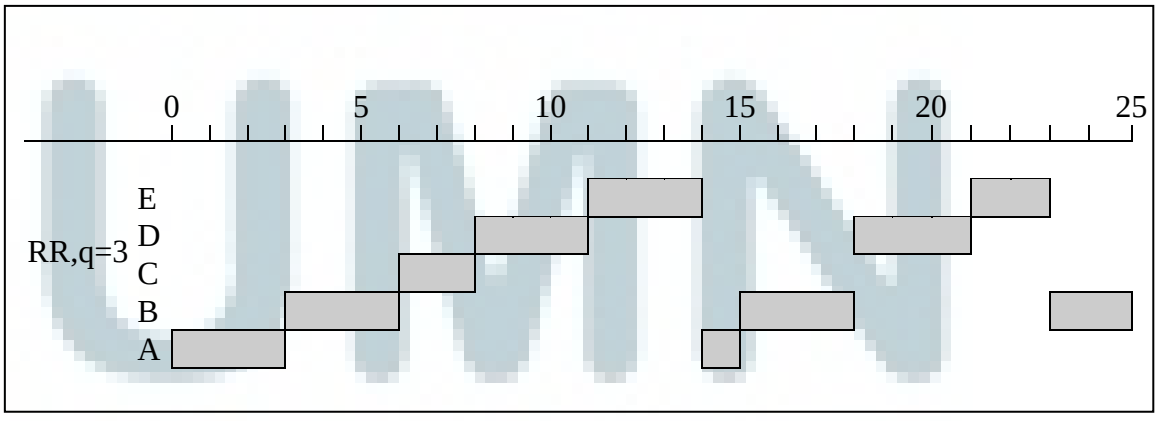
mendapat jatah sebesar *time quantum*. Jika *time quantum*-nya habis atau proses sudah selesai, CPU akan dialihkan ke proses yang selanjutnya.

Dengan proses ini memberikan keadilan pada setiap proses, karena tak ada proses yang diprioritaskan, semua proses mendapat jatah waktu yang sama dari CPU yaitu (1/n), dan tak akan menunggu lebih lama dari (n-1)q dengan q adalah lama 1 quantum.

Tabel 2.2 Contoh Lima Buah Proses dengan *Burst Time*

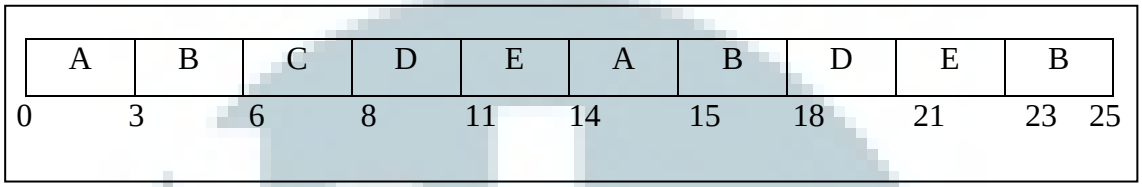
Process	*Burst Time*
A	4
B	8
C	2
D	6
E	5

Gambar 2.1 menunjukkan penyelesaian penjadwalan proses jika menggunakan penjadwalan *Round robin* dengan nilai kwanta = 3 (Novalianty, 2004).



Gambar 2.1 Grafik Proses Penyelesaian Penjadwalan *Round Robin* dengan Kwanta 3 (Sumber: Novalianty, 2004)

Gambar 2.2 menunjukkan penyelesaian penjadwalan *Round Robin* dalam bentuk *gantt chart.*



Gambar 2.2 *Gantt Chart* Penjadwalan Proses *Round Robin* (Sumber: Novalianty, 2004)