Jurnal Ilmiah Teknologi Informasi Asia
Vol.15, No. 2, Tahun 2021
ISSN: 2580-8397 (O); 0852-730X (P)

Analisis Algoritma Round Robin pada Penjadwalan CPU

Tri Dharma Putra[1], Rakhmat Purnomo[2]
[1,2] Universitas Bhayangkara Jakarta Raya
[1]tri.dharma.putra@dsn.ubharajaya.ac.id, [2]rakhmat.purnomo@dsn.ubharajaya.ac.id

ABSTRAK. Penjadwalan adalah konsep penting dalam sistem operasi multiprosesor dan *multitasking* pada sistem operasi waktu-nyata dengan mengalihkan proses pada CPU. Algoritma Round Robin adalah algoritma yang terkenal pada penjadwalan CPU. Algoritma Round Robin memberikan waktu *quantum* antara pengalihan proses. Memilih waktu *quantum* dalam Algoritma Round Robin sangatlah penting, waktu *quantum* besar akan mengakibatkan *context switching* lebih sedikit, sementara waktu *quantum* lebih kecil akan mengakibatkan *context switching* yang lebih sering. Algoritma Round Robin yang efisien adalah jumlah *context switching* lebih rendah. Untuk waktu tunggu, ide dasarnya adalah untuk mendapatkan waktu tunggu rata-rata yang lebih kecil, sehingga sistem lebih efisien. *Turn around time* juga harus minimum, yang berarti juga lebih efisien. Dua studi kasus didiskusikan untuk memahami algoritma ini dengan lebih mendalam. Dilakukan analisis pada data set proses 1 sampai proses 5 (P1-P5), dengan data set *burstime*: 20mdetik, 34mdetik, 25mdetik, 62mdetik, 67mdetik. Pada analisis studi kasus 1 digunakan *quantum* 25mdetik, dan pada studi kasus kedua digunakan *quantum* 15mdetik. Context switching Pada studi kasus 2 ada sebanyak 17, sementara pada pada studi kasus 1, hanya terdapat 10 context switching. Total burst-time pada studi kasus 1 dan studi kasus 2 adalah 208 mdetik, Waktu tunggu rata-rata pada studi kasus 1 adalah 82 mdetik, sedangkan untuk studi kasus 2 adalah 105,8 mdetik. Rata-rata Turn Around Time pada studi kasus 1 adalah 123,6mdetik, sedangkan pada studi kasus 2 adalah 146,6 mdetik.
Kata kunci: *Algoritma Round Robin, waktu quantum, context switching, rata-rata waktu tunggu, rata-rata turn around time*

ABSTRACT. Scheduling is a key concept in computer multitasking and multiprocessing operating system design and in real-time operating system design by switching the CPU among process. Round Robin Algorithm is a wellknown algorithm in CPU scheduling. Round Robin Algorithm provides a time quantum between context switching. Choosing the time quantum in Round Robin Algorithm is very crucial, large quantum time will result in lower context switching, while small quantum time wil result in higher context switching. For waiting time, the idea for waiting time is to get smaller waiting time which is more efficient. Turn around time also should be minimum, which is more efficient. Two case studies are discussed to understand this algorithm more deeply. We did analysis with data set process 1 till process 5 (P1-P5), with burst time dataset: 20ms, 34ms, 25ms, 62ms, 67ms. In case study 1, we use quantum 25m, and in case study 2, we use quantum 15ms. Context switching in case study 2 is 17 where on case study 1, only 10 context switchings. Total burst-time in case study 1 and case study 2 are 208 ms. Average waiting time in case study 1 is 82 ms, where in case study 2 is 105.8 ms. Average Turn Around Time in case study 1 is 123.6ms, where in case study 2 is 146.6 ms.
Keywords: *Round Robin Algorithm, quantum time, context switching, average waiting time, average turn around time*

1. PENDAHULUAN

Sistem operasi adalah pengelola sumber daya. Sumber daya yang dikelola oleh sistem operasi adalah perangkat keras, unit penyimpanan, perangkat input, perangkat output dan data. Sistem operasi melakukan banyak fungsi seperti mengimplentasikan antar muka pengguna, berbagai perangkat keras antar pengguna, memfasilitasi input/output, perhitungan penggunaan sumber daya, mengatur data, dll. Penjadwalan proses adalah salah satu dari fungsi yang dilakukan oleh sistem operasi. Penjadwalan CPU adalah tugas dalam memilih proses dari antrian siap dan mengalokasikan CPU padanya (Amar Ranjan Dash, 2015). Apabila CPU menjadi *idle*, proses menunggu dari antrian siap dipilih dan CPU dialokasikan padanya.

Sebuah sistem operasi adalah sebuah program yang mengelola perangkat keras komputer dan menyediakan dasar untuk aplikasi program and bertindak sebagai penengah antara pengguna komputer dan perangkat keras (Md. Sohrawordi, 2019). Sistem operasi menyediakan antarmuka antara pengguna dan perangkat keras komputer yang membantu pengguna menangani sistem dengan cara yang nyaman. Sistem operasi modern menjadi lebih kompleks ketika sistem beralih dari tugas tunggal ke lingkungan multitasking (Tithi Paul, 2019).

Sistem operasi modern mendukung lingkungan *multitasking* dimana proses berjalan bersamaan. Pada sistem dengan prosesor tunggal, hanya satu proses yang dapat berjalan pada CPU pada satu waktu. Proses

lainnya yang berada di antrian siap harus menunggu sampai CPU menjadi bebas. Sistem operasi harus memutuskan melalui *scheduler*, urutan eksekusi dari proses. Objektif dari *multiprogramming* atau *multitasking* adalah untuk membuat beberapa proses berjalan setiap saat. untuk memaksimalkan utilisasi CPU.

Penjadwalan adalah fungsi sistem operasi yang fundamental. Hampir semua sumber daya komputer dijadwalkan sebelum digunakan. CPU, tentu saja, merupakan salah satu sumber daya komputer utama. Karena itu, penjadwalannya menjadi penting sekali pada desain sistem operasi. Penjadwalan CPU menentukan proses mana yang berjalan ketika ada beberapa proses yang dapat dieksekusi. Penjadwalan CPU adalah penting karena dapat memiliki efek pada utilisasi sumber daya dan kinerja keseluruhan sistem. Secara umum kita ingin mengoptimalkan perilaku sistem (Datta, 2015).

Ide dasar *multiprogramming* adalah mengatur beberapa proses untuk dieksekusi oleh CPU. Yaitu ketika proses mengalokasikan sumber daya bagi CPU. Ketika ada lebih dari satu proses yang dapat dieksekusi, hanya satu proses yang akan dialokasikan ke sumber daya. Ada beberapa algoritma, FCFS (First Come First Serve), *Priority Scheduling*, and *Shortest Job First (SJF) Scheduling*, dan Algoritma Round Robin. Penjadwalan CPU adalah masalah dalam memutuskan proses mana yang siap di antrian harus dialokasikan ke CPU (Putra, 2020).

Penjadwalan dapat merupakan *preemptive* atau *nonpreemptive*. Pada penjadwalan nonpreemptive, CPU tidak dapat mengalihkan proses ke proses lain sampai waktu penyelesaian dituntaskan, sedangkan penjadwalan *preemptive*, proses yang sedang dieksekusi mungkin dipaksa untuk mengalihkan proses sedang berjalan oleh proses lain berdasarkan kriteria penjadwalan (Dhruv, 2019). Pada Algoritma Round Robin, proses dialihkan ke proses berikutnya berdasarkan waktu *quantum* yang ditentukan. Sehingga Algoritma Round Robin merupakan algoritma *preemptive.*

Jadi kita dapat menyimpulkan bahwa algoritma penjadwalan yang baik untuk sistem waktu nyata dan time sharing harus memiliki karakteristik berikut:

- *Context switching* yang minimum
- Utilisasi CPU yang maksimum
- *Throughput* yang maksimum
- *Turnaround time* yang minimum
- Waktu tunggu yang minimum, dan
- Waktu respons yang minimum.

2. METODE PENELITIAN

Metode merupakan teknik yang digunakan pada pemecahan suatu masalah untuk menghasilkan hasil yang sesuai dengan yang diharapkan. Penulis menganalisis algoritma round robin dalam jurnal ini berdasarkan berbagai sumber, jurnal dan sumber-sumber lain untuk digunakan dalam penelitian ini, semuanya terpercaya dan digunakan dalam penulisan jurnal penelitian ini, semuanya tentu saja berkaitan dengan algoritma round robin yang kita bahas di sini.

Metode penelitian pada jurnal ini adalah studi pustaka yang mendalam. Literatur studi dilakukan dari berbagai sumber literatur dan jurnal. Beberapa penelitian terkait dibandingkan berdasarkan literatur. Ada beberapa macam varian Algoritma Round Robin di literatur, salah satunya adalah dengan waktu *quantum* yang bervariasi. Namun dalam studi kasus dalam jurnal ini dilakukan dengan waktu *quantum* yang tetap. Dua studi kasus Algoritma Round Robin dibahas secara mendalam satu persatu.

Round robin adalah metode pada penjadwalan proses yang membagi semua proses dengan porsi waktu yang sama. Proses dimana waktu prosesnya lebih pendek akan selesai lebih dahulu, daripada proses yang waktu prosesnya lebih lama. Metode ini membagi proses dengan *quantum* yang sama pada setiap proses. Berapa lama nilai *quantum* terhadap proses telah ditentukan sebelumnya. Tidak ada prioritas dalam proses dan diberikan *quantum* atau *time-slice* yang sama. Round robin dibuat sehingga memenuhi kebutuhan time-sharing. Dalam jurnal ini semua proses memiliki waktu *quantum* yang sama.

Keuntungan Algoritma Round Robin adalah sebagai berikut (J. R. Indusree, 2017):

- *Starvation* pada proses dapat dihindari, karena setiap proses mendapatkan giliran untuk dieksekusi oleh CPU.
- Proses dengan *burst-time* lebih kecil mendapat keuntungan karena dieksekusi lebih dulu dibandingkan dengan algoritma yang lain.
- Algoritma Round Robin adalah *preemptive*.

Tujuan penjadwalan sistem adalah untuk :

- *Throughput*: Throughput adalah jumlah proses yang diselesaikan dalam satu unit waktu. Algoritma penjadwalan harus didesain sedemikian rupa sehingga *throughput* sistem menjadi maksimal.
- Utilisasi CPU: Ini adalah persentase waktu bahwa CPU sibuk dalam mengeksekusi proses. Tujuan dasar penjadwalan adalah menjaga prosesor sibuk sepanjang waktu.
- *Fairness*: Semua proses dalam sistem harus diperlakukan dalam cara yang sama kecuali ada beberapa proses yang disukai atau diprioritaskan. Jika seperti ini, semua proses dengan prioritas lebih rendah harus diabaikan untuk menghindari *starvation*.
- *Context Switching*: Adalah prosedur menyimpan status proses aktif dan memulihkan status proses lain untuk CPU ketika sistem harus mulai mengeksekusi proses berikutnya. *Context switch*ing adalah *overhead* dari sistem dan membuat waktu tidak efektif dari CPU. Algoritma penjadwalan harus didesain untuk meminimalkan *context switch*ing.

Jurnal ini diatur dalam bab-bab berikut ini. Jurnal ini dibagi menjadi lima bab. Bab 1 adalah Pendahuluan. Di bab satu ini dibahas konsep penjadwalan sistem operasi secara umum. Bab dua adalah Metode Penelitian. Pada bab ini dibahas metode penelitian yang dilakukan. Yaitu literatur studi yang mendalam. Bab 3 merupakan pembahasan. Pada pembahasan ini dianalisis dua case studi untuk Algoritma Round Robin. Bab 5 adalah Kesimpulan. Dan yang terakhir adalah Daftar Pustaka.

3. HASIL DAN PEMBAHASAN

Algoritma Round Robin didesain khususnya untuk sistem *time-sharing*. Masing-masing proses ditetapkan interval waktu, yang disebut waktu *quantum*. Masing-masing proses proses disediakan waktu tetap untuk mengeksekusi sesuai waktu *quantum*. Waktu *quantum* umumnya adalah 10 hingga 100 mdetik. Antrian siap diperlakukan sebagai antrian sirkuler (Arpita Sharma, 2015). Sehingga proses dialihkan berurutan dari P1 hingga P5 dan kembali ke P1 hingga ke P5 kembali secara sirkuler.

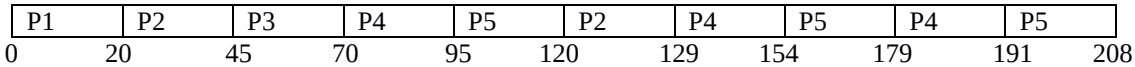
a. Studi Kasus 1

Diberikan lima proses P1 sampai P5 pada tabel 1 di bawah ini. Dengan *burst-time* masing-masingnya. Pada studi kasus ini diberikan waktu *quantum* sama dengan 25 mdetik.

Tabel 1. Lima Proses dan masing-masing *Burst-Time* CPU dengan waktu *quantum* = 25 mdetik

Nama Proses	Burst-Time CPU (mdetik)
P1	20
P2	34
P3	25
P4	62
P5	67

Perhatikan Gambar 1. Gantt Chart studi kasus 1. Proses akan bergiliran dari P1 ke P5 dan kembali ke P1 lagi, dengan urutannya. Proses dimulai dengan P1 dengan *burst-time* adalah 20. Setelah itu adalah P2, dengan *burst-time* adalah 34, sehingga tersisa 9, dengan *quantum* 25 sehingga sampai di 45. P3, 45 ditambah 25 menjadi 70, dengan sisa *burst-time* 0, berarti proses P3 selesai. Kemudian P4 masuk sampai 95. Pada P4 sisa proses adalah 37. Kemudian beralih ke P5 sampai ke 120, dengan sisa *burst-time* 42. Dari P5 beralih ke P1 kembali, namun P1 sudah selesai dieksekusi, sehingga masuk P2, sisa P2 adalah 9, hingga sampai ke 129. Dari P2 beralih ke P3, namun P3 sudah selesai, sehingga lanjut ke P4. P4 bersisa 37, sehingga sampai ke 154, dengan sisa P4 adalah 12. Dari P4 beralih ke P5, sisa P5 adalah 42, sehingga dikurangi 25 menjadi 17. P4 beralih ke P5 pada 179. Sistem secara sirkuler berlanjut ke P1 kembali. P1 sudah selesai, lanjut P2, P2 juga sudah selesai, sehingga lanjut urutan ke P3, P3 juga sudah selesai lanjut ke P4. Menjadi 191, karena sisa P4 adalah 12. Akhirnya P5 masuk dengan menjadi 208.



Gambar 1. Gantt Chart Studi Kasus 1 dengan *quantum* = 25

Untuk menghitung waktu tunggu setiap proses perhatikan Tabel 2.:

Tabel 2. Menghitung waktu tunggu rata-rata proses

Prosess	Waiting Time
P1	0
P2	20 + (120-45)= 95
P3	45
P4	70 + (129-95)+(179-154)=129
P5	95+(154-120)+(191-179)=141

Sehingga waktu tunggu rata-rata adalah (0+95+45+129+141)/5=82 mdetik. Untuk menghitung rata-rata Turn Around Time perhatikan tabel 3. Di bawah ini:

Tabel 3. Menghitung rata-rata Turn Around Time

Proses	Waktu tiba	Burst-time	Waktu mulai	Waktu selesai	Turn Around Time
P1	0	20	0	20	20
P2	0	34	20	129	129
P3	0	25	45	70	70
P4	0	62	70	191	191
P5	0	67	95	208	208

Sehingga rata-rata Turn Around Time adalah (20+129+70+191+208)/5=123,6 mdetik.

b. Studi Kasus 2

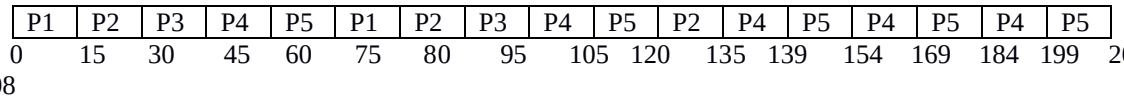
Pada studi kasus 2 ini, perhatikan Tabel 4. di bawah ini. Proses dan *burst-time* sama, namun waktu *quantum* yang digunakan adalah 15. Lebih sedikit waktu *quantum* maka *context switching* akan menjadi lebih banyak.

Tabel 4. Lima Proses dan masing-masing *Burst-Time* CPU dengan waktu *quantum* = 15

Nama Proses	Burst-Time CPU
P1	20
P2	34
P3	25
P4	62
P5	67

Proses P1 dimulai, karena quantum 15 maka proses bersisa 5. Kemudian proses P2 masuk, dan proses P2 bersisa 19. Proses P3 masuk dan bersisa 10. Kemudian P4 masuk dan bersisa 47. Setelah itu proses P5 masuk dan bersisa 52. Proses kemudian kembali ke P1 lagi, secara sirkuler. P1 masuk dan tidak bersisa, dari 75 menjadi 80 karena P1 hanya bersisa 5. Kemudian P2 masuk dari 19 menjadi bersisa 4, dengan data di gantt

chart menjadi 95. Kemudian P3 masuk sebanyak 10, gantt chart menjadi 105. Berikutnya P4 masuk, sisa sebelumnya 47, sehingga gantt chart menjadi 120. Dengan proses bersisa 32. Berikutnya P5 masuk, status sekarang adalah 52, sehingga sisa menjadi 37, pada gantt chart menjadi 135. Kemudian proses beroperasi secara sirkuler, kembali ke P1. P1 status tidak ada lagi proses tersisa sehingga dilewatkan. P2 masuk dengan *burst-time* tersisa 4, sehingga di gantt chart menjadi 139 dari 135. Kemudian proses beralih ke P3. P3 ternyata sudah selesai tidak ada *burst-time* tersisa, sehinggga lanjut ke P4. P4 masuk, dari tersisa 32 menjadi tersisa 17, di gantt chart menjadi 154, yaitu ditambahkan 139 tambah 15 menjadi 154. Berikutnya P5 masuk dengan sisa yang ada 37, sehingga sisa menjadi 22, di gantt chart menjadi 169. Kemudian proses secara sirkuler kembali ke P1. Pada P1 proses sudah selesai, sehingga lanjut ke P2. P2 juga sudah selesai sehingga lanjut ke P3, P3 juga sudah selesai, sehingga dilanjutkan ke P4. Pada P4 sisa *burt-time* adalah 17, sehingga di gantt chart dari 169 menjadi 184. Sisa *burst-time* pada P4 ini adalah 2. Proses kemudian berlanjut ke P5. Pada P5, sisa proses yang ada 22, sehingga di gantt chart menjadi 199, dengan *burst-time* tersisa 7. Proses kemudian secara sirkuler balik lagi ke P1. P1 sudah selesai, sehingga lanjut ke P2. P2 sudah selesai sehingga lanjut ke P3, P3 sudah selesai sehingga lanjut P4. Pada P4 *burt-time* tersisa 2, sehingga di gantt chart menjadi 201 dari 199. Berikutnya P5, yang hanya tersisa 7, sehingga di gantt chart menjadi 208 dari 201. Akhirnya semua proses selesai.



Gambar 2. Gantt Chart Studi Kasus 2 dengan waktu *quantum* = 15

Untuk perhitungan waktu tunggu rata-rata perhatikan Tabel 5. di bawah ini:

Tabel 5. Menghitung waktu tunggu rata-rata proses

Nama Proses	Waktu Tunggu
P1	0+(75-15)=60
P2	15+(80-30)+(135-95)=109
P3	30+(95-45)=80
P4	45+(105-60)+(139-120)+(169-154)+(199-184)=139
P5	60+(120-75)+(154-135)+(184-169)+(201-199)=141

Maka waktu tunggu rata-rata adalah (60+109+80+139+141)/5=105,8 mdetik. Untuk menghitung rata-rata *Turn Around Time* perhatikan Tabel 6. di bawah ini:

Tabel 6. Menghitung rata-*rata Turn Around Time*

Proses	Waktu Tiba	Burst-time	Waktu mulai	Waktu selesai	Turn Around Time
P1	0	20	0	80	80
P2	0	34	15	139	139
P3	0	25	30	105	105
P4	0	62	45	201	201
P5	0	67	60	208	208

Sehingga rata-rata Turn Around Time adalah (80+139+105+201+208)/5=146,6 mdetik.

4. KESIMPULAN

Context switching yang lebih banyak terjadi pada waktu *quantum* = 15. Pada studi kasus 2 dengan waktu *quantum* 15, ada sebanyak 17 *context switch*ing, sementara pada waktu *quantum* = 25, pada studi kasus 1, hanya terdapat 10 *context switch*ing.

Total *burst-time* untuk waktu *quantum* = 25 pada studi kasus 1 adalah 208 mdetik, sedangkan pada waktu *quantum* = 15, adalah juga sama 208 mdetik, nilai ini harus sama, yang membuktikan bahwa perhitungan gantt chart sudah benar pada kedua studi kasus.

Waktu tunggu rata-rata untuk waktu *quantum*=25 pada studi kasus 1 adalah 82 mdetik, sedangkan waktu tunggu rata-rata untuk studi kasus 2, dengan waktu *quantum* 15 adalah 105,8 mdetik. Sehingga waktu tunggu yang lebih efisien adalah pada *quantum* yang sama dengan 25. Karena waktu tunggunya lebih kecil.

Rata-rata *Turn Around Time* untuk waktu *quantum* = 25 pada studi kasus 1 adalah 123,6, sedangkan rata-rata *Turn Around Time* untuk waktu *quantum* = 15 pada studi kasus 2 adalah 146,6 mdetik. Sehingga rata-rata *Turn Around Time* yang lebih efisien adalah pada waktu *quantum* yang sama dengan 25, karena rata-rata *Turn Around Time*-nya lebih kecil.

5. SARAN

Diusulkan untuk studi kasus Algoritma Round Robin berikutnya adalah dengan menggunakan waktu *quantum* yang bervariasi, sehingga lebih mengefisienkan sistem.

DAFTAR RUJUKAN

Amar Ranjan Dash, S. K. (2015). An Optimized Round Robin CPU Scheduling Algorithm with Dynamic Time Quantum. *International Journal of Computer Science, Engineering and Information Technology, Vol. 5, No. 1*, 7-26.

Arpita Sharma, M. G. (2015). Analysis of Adaptive Round Robin Algorithm and Proposed Round Robin Remaining Time Algorithm. *International Journal of Computer Science and Mobile Computing, Vol 4. Issue 12*, 139-147.

Datta, L. (2015). Efficient Round Robin Scheduling Algorithm with Dynamic Time Slice. *International Journal of Education and Management Engineering*, 10-19.

Dhruv, R. (2019). Round Robin Scheduling Algorithm Based on Dynamic Time Quantum. *International Journal of Engineering and Advanced Technology, Volume X, Issue X*, 593-595.

J. R. Indusree, B. P. (2017). Enhanced Round Robin CPU Scheduling with Burst Time Based Time Quantum. *IOP Conference Series: Material Science and Engineering*, 1-8.

Md. Sohrawordi, U. A. (2019). A Modified Round Robin CPU Scheduling Algorithm with Dynamic Time Quantum. *International Journal of Advanced Research (IJAR)*, 422-429.

Putra, T. D. (2020). Analysis of Preemptive Shortest Job First (SJF) Algorithm in CPU Scheduling. *International Journal of Advanced Research in Computer and Communication Engineering, Vol 9, Issue 4, April*, 41-45.

Tithi Paul, R. H. (2019). Improved Round Robin Scheduling Algorithm with Progressive Time Quantum. *International Journal of Computer Application Volume 178, No. 49*, 30-36.