MODIFIKASI ALGORITMA ROUND ROBIN DENGAN *DYNAMIC QUANTUM TIME* DAN PENGURUTAN PROSES SECARA *ASCENDING*

Gortap Lumbantoruan
Program Studi Komputerisasi Akuntansi, Universitas Methodist Indonesia
Email: lumbantoruan.gortap@gmail.com

ABSTRACT

Algorithm Round Robin scheduling algorithm is one of the widely used process in the CPU scheduling. Round Robin algorithm using a timesharing system with a static quantum time for each process to be executed CPU. This algorithm depends on the size of a given quantum time. If the quantum time is too large, the response time for the processes are too high. Conversely, if the quantum time is too small, it can lead to overhead on the CPU in which the context switching of the process becomes larger. In this study, to improve CPU performance by reducing waiting time and turnaround time modification of Round Robin algorithm using dynamic quantum time and sorting in ascending process. Conducted testing of the queue process and results using the modified round robin algorithm is obtained average waiting time and average turnaround time is much smaller than algorithms using Round Robin Classic.

Keywords: *Quantum Time, Time Quantum Static, Dynamic Quantum Time, Ascending, Average Waiting Time, Average Turnaround Time*

ABSTRAK

Algoritma Round Robin merupakan salah satu algoritma penjadwalan proses yang digunakan secara luas didalam penjadwalan CPU. Algoritma Round Robin menggunakan sistem time sharing dengan static quantum time untuk setiap proses yang akan dieksekusi CPU. Algoritma ini tergantung pada ukuran quantum time yang diberikan. Jika quantum time terlalu besar, maka respons time untuk proses-proses terlalu tinggi. Sebaliknya, jika quantum time terlalu kecil, maka dapat mengakibatkan overhead pada CPU dimana context switching dari proses menjadi lebih besar. Pada penelitian ini, untuk meningkatkan performa CPU dengan memperkecil waiting time dan turnaround time dilakukan modifikasi terhadap algoritma Round Robin dengan menggunakan dynamic quantum time serta sorting proses secara ascending. Dilakukan pengujian terhadap antrian proses dan hasilnya dengan menggunakan algoritma Round Robin yang dimodifikasi ini didapat average waiting time dan average turnaround time yang lebih kecil dibandingkan menggunakan algoritma Round Robin Klasik.

Kata kunci: *Quantum Time, Static Quantum Time, Dynamic Quantum Time, Ascending, Average Waiting Time, Average Turnaround Time*

PENDAHULUAN

Sistem Operasi merupakan *software* yang menghubungkan antara *user hardware* komputer. Sistem operasi berfungsi umtuk mengelola *hardware* komputer, dan menjadi sarana untuk *user* dimana user dapat menjalankan program aplikasi atau mengeksekusi program dengan cara nyaman dan efisien.[1]

Salah satu misi yang dijalankan oleh sistem operasi adalah efisiensi penggunan waktu ketika terjadi *multiprogramming*. Memperbolehkan beberapa program berjalan pada saat yang hampir bersamaan atau secara bersamaan mengakibatkan terjadinya *multipro- cessing* yang membuat CPU harus melaksanakan eksekusi terhadap sejumlah proses tersebut.[2]

Tujuan dari *multiprogramming* adalah untuk memiliki beberapa proses yang berjalan setiap saat, untuk memaksimalkan penggunan CPU. Tujuan dari pembagian waktu adalah untuk mengganti CPU diantara proses begitu sering bahwa pengguna dapat berinteraksi dengan setiap program ketika sedang berjalan.[3]

Karena banyaknya jumlah proses yang akan dieksekusi oleh CP mengalami kompleksitas sendiri karena alokasi waktu yang sangat terbatas. Efisiensi penggunaan waktu eksekusi beberapa program dipengaruhi oleh kecepatan CPU dalam mengeksekusi sejumlah proses. Oleh karena itu sistem operasi harus membagi waktu CPU untuk mengeksekusi sejumlah proses tersebut dengan mengatur penjadwalan eksekusi untuk masing-masing proses. Penjadwalan CPU adalah dasar dari sistem *multiprogramming.* Penjadwalan CPU mengacu pada aturan dan mekanisme untuk mengontrol urutan pekerjaan yang harus dilakukan oleh CPU. Hal ini dibuat oleh bagian dari sistem operasi yang disebut *scheduler,* dengan menggunakan algoritma penjadwalan.[4]

Dari ulasan tentang penelitan-peneltian diatas, diperoleh informasi bahwa kinerja algoritma Round Robin diharapkan dapat ditingkatkan dengan menerapkan nilai *quantum time* yang dinamis.

Pada penelitian ini, penulis meneliti solusi peningkatan kinerja algoritma Round Robin dengan pengurutan proses dan penetuan *quantum time* yang dinamis dan diharapkan dapat memperkecil *averange waiting time* dan *averange turnaround time* dan proses yang terdapat pada antrian.

Sistem Operasi

Sistem operasi adalah *software* yang menangani *hardware* komputer dan menyediakan sumber daya kepada program yang akan berkoperasi. Dan juga sebagai penghubung antara *user* dengan sistem komputer[1]. Sistem operasi adalah sebuah program yang mengontrol eksekusi dari program aplikasi dan bertindak sebagai penghubung antara program aplikasi dengan *hardware* komputer.[3]

Proses

Secara informal, proses adalah program daam eksekusi. Suatu proses adalah lebih dari kode program, dimana kadang kala proses dikenal sebagai bagian tulisan. Proses juga termasuk aktivitas yang sedang terjadi, sebagaimana digambambarkan oleh nilai pada *program counter* dan isi dari daftar *processor's register.* Suatu proses umumnya juga termasuk *process stack,* yang berisikan data tempor (seperti parameter metoda, *address* yang kembali, dan variabel lokal) dan sebuah *data section,* yang berisikan variabel global. Sebagaimana proses bekerja, maka proses tersebut merubah state (keadaan statis/asal). Status dari sebuah

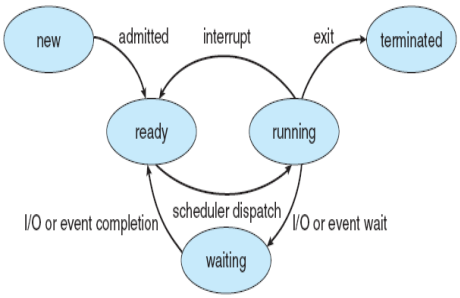
proses didefenisikan dalam bagian oleh aktivitas yang ada pada proses tersebut.[5]

Konsep kunci dalam semua sistem operasi adalah proses. Sebuah proses pada dasarnya adalah program yang dieksekusi[6]. Proses terkait dengan address space, atau daftar lokasi memori mulai dari ruang 0 sampai keruang maksimm, dan proses dapat membaca dan menulis pada *address space* tersebut. *Address space* berisi program yang dieksekusi, data dan *stack.* Proses membutuhkan sejumlah sumber daya, umumnya termasuk register (*program counter* dan *stack pointer*, *file*, proses lain yang terkait dengan proses tersebut, dan semua informasi lainnya yang dibutuhkan untuk menjalankan program tersebt. Proses menampung semua informasi yang dibutuhkan untuk mengeksekusi program.[6]

Keadaan Proses

Semua proses berada dalam keadaan seperti berikut ini [7]:

a. *New*
   Proses sedang dikerjakan/dibuat.
b. *Running*
   Intruksi sedang dikerjakan.
c. *Waiting*
   Proses sedang menunggu sejumlah kejadian untuk terjadi (seperti sebuah penyelesaian I/O atau penerimaan sebuah tanda/signal).
d. *Ready*
   Proses sedang menunggu untuk ditugaskan pada sebuah prosesor.
e. *Terminated*
   proses telah selesai melaksanakan tugasnya/ mengeksekusi



Gambar 1. Keadaan Proses

Penjadwalan Proses

Tujuan dari *Multi -programing* adalah untuk mempunyai proses berjalan secara bersamaan, untuk memaksimalkan kinerja dari CPU.[8] Untuk sistem *uniprosesor*, tidak pernah ada proses yang berjalan lebih dari satu. Bila ada proses yang lebih dari satu maka yang lain harus mengantri sampai CPU bebas.

Tujuan penadwalan adalah untuk meminimalkan total biaya layanan komputer dan waktu tunggu *user*.[9] Dalam prakteknya ada dua masalah yang sering dihadapi, waktu pelayanan terhadap proses dapat dikurangi dengan memperhatikan waktu prosesor hilang akibat intervensi *user*, perangkat keras yang lambat , dan *multiplexing* sumber daya. Hal ini biasanya memiliki pengaruh yang drastis pada mode operasi ditawarkan pada *user*.

Algoritma Round Robin

Algoritma Round Robin dirancang untuk sistem *time sharing.* Algoritma ini mirip dengan penjadwalan FCFS, namun preemption ditambahkan untuk switch antara proses.[10] Antrian *reaady* dan mengalokasikan masing-masing proses untuk interval waktu tertentu sampai satu *time slicel quantum*.[11]

Kriteria yang biasanya digunakan dalam memilih penjadwalan adalah:[12]

1. *CPU utilization* kita ingin menjaga CPU sesibuk mungkin. CPU *(time*

slicel quantum) umumnya antara 10-100 milli detik.
   a. Setelah *quantum time* maka proses akan disk-preemp-dan dipindahkan ke antrian *ready*
   b. Proses ini adil dan sangat sederhana
2. Jika terdapat n proses di "antrian *ready"* dan *quantum time* q (milli detik), maka:
   a. Maka setiap proses akan mendapatkan 1/n dari waktu CPU
   b. Proses tidak akan menunggu lebih lama dari: (n-1)q *tine units*
3. Kinerja dari algoritma ini tergantung dari ukuran *quantum time*
   a. *Quantum time* dengan ukuran yang besar maka akan sama dengan FCFS
   b. *Quantum time* dengan ukuran yang kecil maka *quantum time* harus diubah ukurannya lebih besar dengan respect pada ali konteks sebaliknya akan memerlukan ongkos yang besar.

METODE PENELITIAN

Metodologi

Metode merupakan suatu cara atau teknik yang sistematik untuk memecahkan suatu kasus sehingga memberikan hasil sesuai dengan yang diharapkan. Peneliti menggunakan studi kepustakaan (*library research*), yaitu menggunakan sumber-sumber melalui buku, jurnal serta sumber-sumber lain yang relevan untuk digunakan dalam penelitian ini. Studi kepustakaan dalam penelitian ini adalah hal-hal yang berkaitan dengan sistem operasi dan algoritma penjadwalan proses.

Data yang Digunakan

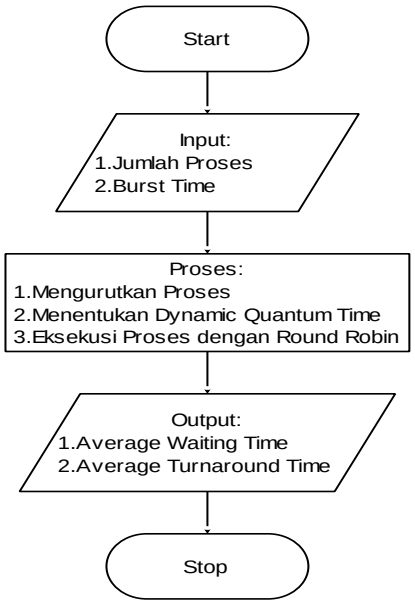
Pada penelitian ini, penulis membutuhkan beberapa data input yang terdiri dari:

1. Jumlah Proses
   Jumlah proses dalam hal ini adalah banyaknya jumlah proses yang sedang mengantri dalam satu tumpukan.
2. *Arrival Time*
   Adalah urutan kedatangan sebuah proses yang akan menunggu untuk di eksekusi
3. *Burst Time*
   Adalah alokasi lamanya waktu eksekusi yang telah di alokasikan kepada masing-masing proses sejak proses itu dibuat.

Prosedur Penyelesaian Masalah

Prosedur kerja

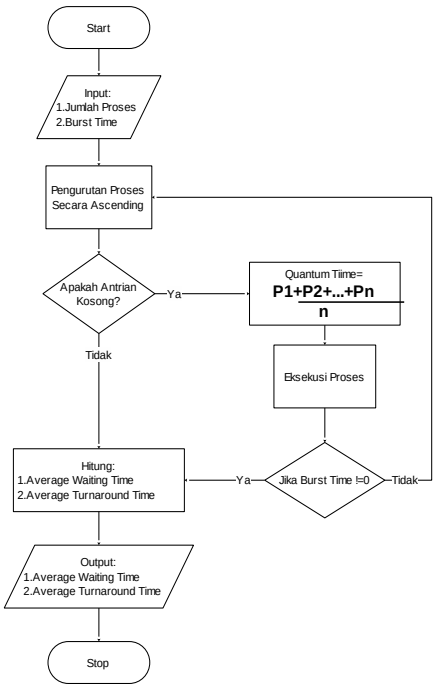
Prosedur kerja modifikasi algoritma Round Robin dengan meetode pengurutan proses secara *ascending* dapat dijelaskan dengan diagram alir pada gambar berikut:



Gambar 2. Prosedur Kerja

Perancangan algoritma

Perancangan algoritma modifikasi algoritma Round Robin dengan metode pengurutan proses secara *ascending* seperti pada gambar berikut ini:



Gambar 3. Perancangan Algoritma

HASIL DAN PEMBAHASAN

Hasil Uji Coba

Pada penelitian ini, penentuan *quantum time* didapat dengan melakukan perhitungan rata-rata *burst time* dari seluruh proses yang ada. Dan proses yang terlebih dahulu dilayani oleh CPU adalah proses yang memilki *busrt time* yang paling kecil dari proses yang ada. Pengujian dilakukan dengan menggunakan algoritma penjadualan round robin klasik dan menggunakan algoritma penjadualan round robin yang dimodifikasi dengan menggunakan *quantum time* yang dinamis dari hasil rat-rata *burst time* seluruh proses yang akan dilayani CPU.

Uji Coba dengan 5 Proses

Pada pengujian yang dilakukan oleh penulis dengan parameter input adalah jumlah proses sebanyak 5 proses, *burst time* dari tiap proses dengan *quantum time* 2 milisecond (ms). Proses yang akan dieksekusi oleh CPU adalah seperti pada tabel berikut ini:

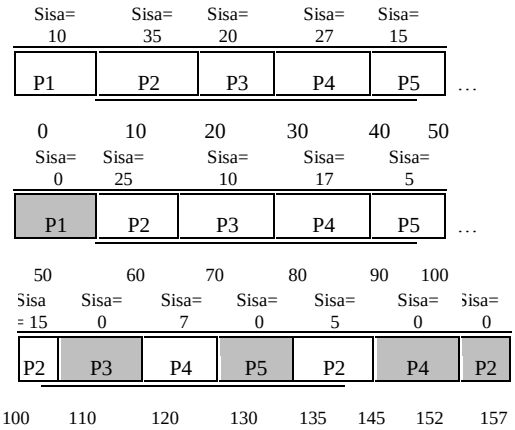
Tabel 1. Uji Coba 5 Proses

Proses	*Burst time* (ms)
P1	20
P2	45
P3	30
P4	37
P5	25

Hasil pengujian 5 proses dengan algoritma Round Robin klasik

Berikut adalah hasil pengujian dari parameter pengujian berdasarkan table 1 diatas:

Gantt Chart



Gambar 4. Gantt Chart Eksekusi Proses dengan Algoritma Round Robin Klasik

Average waiting time (AWT)

Tabel 2. *Waiting time* dengan Round Robin Klasik

Proses	*Waiting time* (ms)
P1	50-10 = 40
P2	152-40 = 112
P3	110-20 = 90
P4	145-30 = 115
P5	130-20 = 110

Dari tabel diatas maka *average waiting time* (AWT) yaitu:

AWT =(WTP1 + WTP2 + WTP3 + WTP4 + WTP5) / 5
= (40 + 112 + 90 + 115 + 100) / 5
= 467/5
= **93.4**

***Average turnaround time* (ATT)**

Untuk *turnaround time* didapat dengan menjumlah *waiting time* dengan *burst time. Turnaround time* dari tiap proses diatas dapat dilihat seperti pada tabel berikut ini:

Tabel 3.*Turnaround Time* dengan Round Robin Klasik

Proses	***Burst Time***	***Waiting Time* (WT)**	***Turnaround Time (TT)***
P1	20	40	60
P2	45	112	157
P3	30	90	120
P4	37	115	152
P5	25	110	135

Maka *average tunaround time* (ATT) untuk seluruh proses tersebut adalah sebagai berikut :

ATT = (TT P1 + TT P2 + TT P3 + TT P4 + TT P5) / 5
=(60 + 157 + 120 + 152 + 135) / 5
= 624 / 5
= **124.8**

Hasil pengujian 5 proses dengan algoritma Round Robin yang dimodifikasi

Uji coba dengan memodifikasi algoritma round robin, yang terlebih dahulu dilakukan adalah melakukan sorting terhadap proses-proses secara *ascending* dan menentukan *quantum time* dengan mencari nilai rata-rata *burst time* dari seluruh proses.

Berikut adalah hasil pengujian terhadap parameter pada tabel diatas menggunakan algoritma round robin dengan *quantum time* berbasis rata-rata dan dengan proses sorting secara *ascending.*

a. Eksekusi Proses

Untuk memulai eksekusi proses, hal yang pertama kali dilakukan adalah pengurutan terhadap proses-proses berdasarkan *busrt time* secara *ascending*.

Langkah 1 : Sorting Proses

Hasil sorting terhadap proses dapat dilihat seperti pada tabel berikut:

Tabel 4. Sorting Proses secara *Ascending*

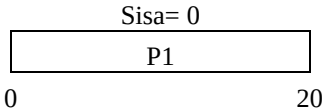
Proses	***Burst Time* (BT)**
P1	20
P5	25
P3	30
P4	37
P2	45

Langkah 2 : Menentukan *Quantum time*

Quantum time untuk proses P1 yang terdapat pada tabel diatas adalah sebagai berikut :

QT = (BT P1 + BT P5 + BT P3 + BT P4 + BT P2) / 5
= (20 + 25 + 30 + 37 + 45) / 5
= 157 / 5
= **31.4**

Langkah 3: Ekseskusi Proses P1 dengan *Quantum time* 31.4 ms



Langkah 4: Sisa Proses

Sisa proses yang belum dieksekusi adalah seperti pada tabel berikut :

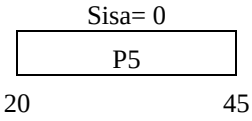
Tabel 5. Sisa Proses Setelah P1 Selesai Dieksekusi

Proses	***Burst Time* (BT)**
P5	25
P3	30
P4	37
P2	45

Langkah 5: Menentukan *Quantum time*
Quantum time untuk proses yang belum dieksekusi adalah sebagai berikut :

QT = (BTP5 + BTP3 + BTP4 + BT P2) / 4
=(25 + 30 + 37 + 45) / 4
=137/ 4
=**34.25**

Langkah 6: Ekseskusi Proses P5 dengan *Quantum time* 34.25 ms



Langkah 7: Sisa Proses
Sisa proses yang belum dieksekusi adalah seperti pada tabel berikut:

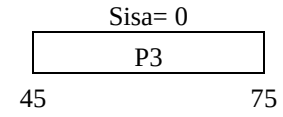
Tabel 6. Sisa Proses Setelah P5 Selesai Dieksekusi

Proses	***Burst time* (BT)**
P3	30
P4	37
P2	45

Langkah 8: Menentukan *Quantum time*
Quantum time untuk proses yang belum dieksekusi adalah sebagai berikut:

QT =(BT P3 + BT P4 + BT P2) / 3
=(30 + 37 + 45) / 3
=112 / 3
=**37.33**

Langkah 9: Ekseskusi Proses P3 dengan *Quantum time* 37.33 ms



Langkah 10: Sisa Proses
Sisa proses yang belum dieksekusi adalah seperti pada tabel berikut:

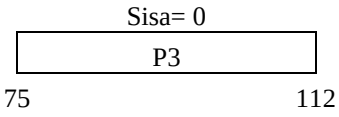
Tabel 7. Sisa Proses Setelah P3 Selesai Dieksekusi

Proses	***Burst time* (BT)**
P4	37
P2	45

Langkah 11: Menentukan *Quantum time*
Quantum time untuk proses yang belum dieksekusi adalah sebagai berikut

QT = (BT P4 + BT P2) / 2
= (37 + 45) / 2
= 82 / 2
= **41**

Langkah 12: Ekseskusi Proses P4 dengan *Quantum time* 41 ms



Langkah 13: Sisa Proses
Sisa proses yang belum dieksekusi adalah seperti pada tabel berikut :

Tabel 8. Sisa Proses Setelah P4 Selesai Dieksekusi

Proses	***Burst time* (BT)**
P2	45

Langkah 14: Menentukan *Quantum time*
Quantum time untuk proses yang belum dieksekusi adalah sebagai berikut:

QT = (BT P2) / 1
= (45) / 1
= 45 / 1
= **45**

Langkah 15: Ekseskusi Proses P2 dengan *Quantum time* 45 ms

Sisa= 0

P2

112 157

b. *Average waiting time* (AWT)

Waiting time dari tiap proses adalah seprti tabel berikut ini :

Tabel 9. *Waiting time* Proses dengan Algoritma Round Robin yang Dimodifikasi

Proses	*Waiting time*
P1	0
P2	112
P3	45
P4	75
P5	20

Dari tabel diatas maka *average waiting time* (AWT) yaitu:

AWT = (WTP1 + WT P2 + WT P3 + WT P4 + WT P5) / 5
= (0 + 112 + 45 + 75 + 20) / 5
= 252 / 5
= **50.4**

c. *Average turnaround time* (ATT)

Turnaround time didapat dengan menjumlah *waiting time* dengan *burst time*. *Turnaround time* dari tiap proses diatas dapat dilihat seperti pada tabel berikut ini:

Tabel 10. *Turnaround Time* Proses dengan Algoritma Round Robin yang Dimodifikasi

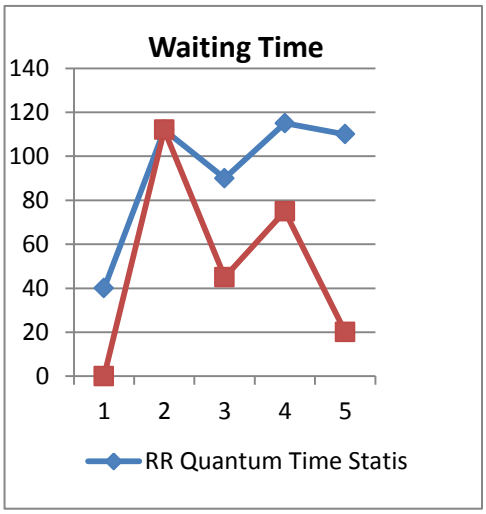
Proses	*Burst time* (BT)	*Waiting time* (WT)	*Turnaroud Time* (TT)
P1	20	0	20
P2	45	112	157
P3	30	45	75
P4	37	75	112
P5	25	20	45

Maka *average turnaround time* (ATT) untuk seluruh proses tersebut adalah sebagai berikut:

ATT = (TT P1 + TT P2 + TT P3 + TT P4 + TT P5) / 5
= (20 + 157 + 75 + 112 + 45) / 5
= 409 / 5
= **81.8**

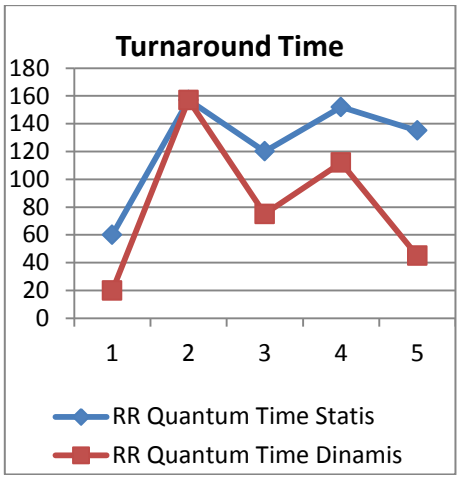
Grafik Waiting Time dan Turnaround Time Round Robin klasik dan Round Robin modifikasi untuk 5 proses

Perbandingan *waiting time* dengan menggunakan algoritma Round Robin klasik dan Round Robin yang dimodifikasi dapat dilihat seperti pada gambar berikut ini:



Gambar 5. Grafik Perbandingan

Waiting Time Round Robin Klasik dan Round Robin Modifikasi untuk 5 Proses Perbandingan *turnaround time* dengan menggunakan algoritma Round Robin klasik dan Round Robin yang dimodifikasi dapat dilihat seperti pada gambar berikut ini:



Gambar 6. Grafik Perbandingan *Turnaround Time* Round Robin Klasik dan Round Robin Modifikasi untuk 5 Proses

Uji Coba dengan 10 Proses

Pada pengujian yang dilakukan oleh penulis dengan parameter input adalh jumlah proses sebanyak 10 proses, *burst time* dari tiap proses dengan *quantum time* 10 milisecond (ms). Proses yang akan dieksekusi oleh CPU adalah seperti pada tabel berikut ini:

Tabel 11. Uji Coba 10 Proses

Proses	*Burst time* (BT)
P1	20
P2	45
P3	30
P4	37
P5	25
P6	23
P7	27
P8	32
P9	40
P10	19

Hasil pengujian 10 proses dengan algoritma Round Robin klasik

Berikut adalah hasil pengujian dari parameter pengujian berdasarkan tabel diatas:

Tabel 12. Hasil Pengujian 10 Proses dengan Algoritma Round Robin klasik

Proses	*Burst time* (BT)	*Waiting Time* (WT)	*Average waitingTime* (AWT)	*Turnaround Time (TT)*	*Average Turnaround Time (ATT)*
P1	20	90	**210**	110	**239.8**
P2	45	253		298	
P3	30	189		219	
P4	37	244		281	
P5	25	209		234	
P6	23	214		237	
P7	27	217		244	
P8	32	251		283	
P9	40	253		293	
P10	19	180		199	

Hasil pengujian 10 proses dengan algoritma Round Robin yang dimodifikasi

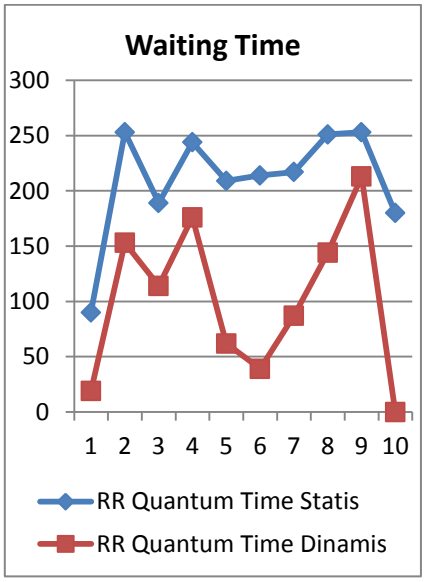
Berikut adalah hasil pengujian dengan pengujian 10 proses berdasarkan tabel diatas :

Tabel 13. Hasil pengujian 10 Proses dengan Algoritma Round Robin yang Dimodifikasi

Proses	*Burst time* (BT)	*Waitning time* (WT)	*Average waiting time*(AWT)	*Turnaround time* (TT)	*Average Turnaround time* (ATT)
P1	20	19	**100.7**	39	**130.5**
P2	45	153		198	
P3	30	114		144	
P4	37	176		213	
P5	25	62		87	
P6	23	39		62	
P7	27	87		114	
P8	32	144		176	
P9	40	213		253	
P10	19	0		19	

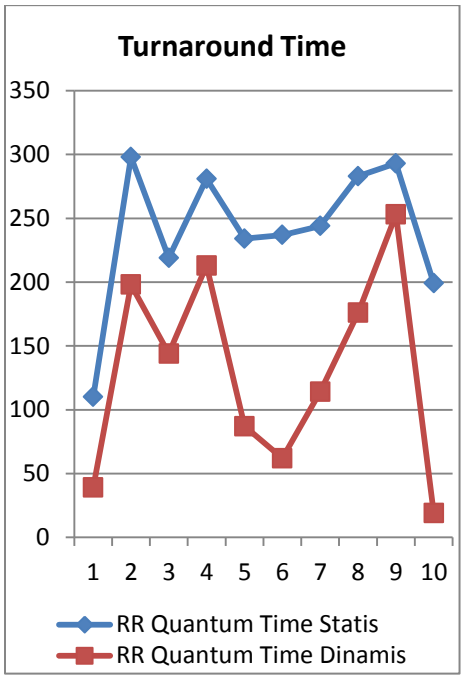
Grafik Waiting Time dan Turnaround Time Round Robin klasik dan Round Robin modifikasi

Perbandingan *waiting time* dengan menggunakan algoritma Round Robin klasik dan Round Robin yang dimodifikasi dapat dilihat seperti pada gambar berikut ini:



Gambar 7. Grafik Perbandingan Waiting Time Round Robin Klasik dan Round Robin Modifikasi untuk 10 Proses

Perbandingan *turnaround time* dengan menggunakan algoritma Round Robin klasik dan Round Robin yang dimodifikasi dapat dilihat seperti pada gambar berikut ini:



Gambar 8. Grafik Perbandingan *Turnaround Time* Round Robin Klasik dan Round Robin Modifikasi untuk 10 Proses

Persentase Penurunan *average Time* dan *average Turnaround Time*

Pada penelitian ini, penulis melakukan analisis dengan membandingkan *average waiting time* dan *average waiting time* dengan menggunakan algoritma Round Robin yang menggunakan *quantum time* statis dan algoritma Round Robin yang menggunakan *quantum time* dinamis. Pada tabel berikut ini ditampilkan hasil dengan menggunakan algoritma Round Robin yang menggunakan *quantum time* dinamis serta persentasi penurunan *average waiting time* dan *average waiting time* dengan jumlah proses 5 proses, 10 proses dan 25 proses.

Tabel 12. Persentase Penurunan AWT dan ATT

Jumlah Proses	RR Klasik		RR Modifikasi		Persentase Penurunan	
	AWT	ATT	AWT	ATT	AWT (%)	ATT (%)
5	93,4	124,8	50,4	81,8	**53,96**	**65,54**
10	201,0	239,8	100,7	130,5	**47,95**	**54,42**
25	715,08	759,52	415,16	459,6	**58,06**	**60,51**

Dari hasil yang penelitian yang telah dilakukan seperti yang telah dipaparkan diatas dapat disimpulkan bahwa algoritma Round Robin dengan menggunakan *quantum time* dinamis memiliki *average waiting time* dan *average waiting time* lebih kecil dibandingkan dengan Round Robin yang menggunakan *quantum time* statis.

KESIMPULAN

Berdasarkan pembahasan dan hasil uji coba yang telah dilakukan dalam penelitian ini maka dapat disimpulkan bahwa:

1. Algoritma Round Robin bergantung pada nilai *quantum time* yang dipilih. Jika *quantum time* yang dipilih terlalu kecil, maka akan menambah *context switching* dan jika *quantum time* yang dipilih terlalu besar, maka akan meningkatkan *respons time*
2. Dengan melakukan pengurutan proses secara *ascending* terhadap proses, maka proses yang memiliki *burst time* yang lebih kecil dilayani terlebih dahulu sehingga memperkecil *context switching* dan akan mengurangi *overhead* pada CPU
3. *Turnaround Time* yang dibutuhkan untuk mengeksekusi sejumlah proses yang mengantri di dalam CPU akan lebih kecil jika nilai *quantum time* yang diberikan dinamis, tetapi dalam urutan eksekusi yang sama proses akan memiliki *turnaround time* yang sama jika *burst time* proses yang dieksekusi CPU menggunakan algoritma Round Robin Klasik lebih besar dibandingkan dengan *burst time* proses yang menggunakan algoritma Round Robin Modifikasi.
4. *Average waiting time* dengan menggunakan *quantum time* berbasis rata-rata dan sorting secara *ascending*, lebih kecil dibandingkan menggunakan *quantum time* yang dinamis, tetapi dalam urutan eksekusi yang sama proses akan memiliki *waiting time* yang sama jika *burst time* proses yang dieksekusi CPU menggunakan algoritma Round Robin Klasik lebih besar dibandingkan dengan *burst time* proses yang menggunakan algoritma Round Robin Modifikasi.
5. *Waiting time dan turnaround time* dipangaruhi oleh urutan proses yang dieksekusi CPU
6. Penerapan algoritma Round Robin yang menggunakan *quantum time* berbasis rata-rata dan sorting terhadap proses secara *ascending*, sangat berpengaruh terhadap kinerja CPU dan sistem operasi. Proses-proses yang mengantri dapat diselesaikan dengan menggunakan waktu lebih sedikit dibanding dengan Round Robin klasik.

Saran

Adapun saran yang dapat peneliti berikan untuk pengembangan penelitian yang berkaitan dengan penjadwalan CPU perlu dilakukan berbagai cara lain agar lebih meningkatkan kinerja CPU dengan *waiting time, turnaround time,*

context switching yang lebih kecil lagi tanpa memperbesar *respons time*.

DAFTAR PUSTAKA

[1]Silberschatz, A., Galvin, P.B. & Gagne, G. 2012. *Operating System Concept*. 9th Edition. Jhon Wiley & Sons: Hoboken.

[2]Tanenbaum, A.S. 2009. *Modern Operating System*. 3rd Edition. Prentice Hall: Upper Saddle River.

[3]Stallings, W. 2012. *Operating System: Internal and Design Principles*. 7th Edition. Prentice Hall : Upper Saddle River.

[4]Abdulrahim, A., Abdullahi, S.E. & Salahu, J.B. 2014. A New Improved Round Robin (NIRR) CPU Scheduling Algorithm. *International Journal of Computer Aplication.* **90**: 27-33.

[5]Behera, H.S., Mohanty, R. & Nayak, D. 2010. A New Proposed Dynamic Quantum Time With Re-Adjusted Round Robin Scheduling Algorithm and Its Performance Analysis. *International Journal of Computer Aplication*. **5:**10-15.

[6]Cormen, T.H., Leiserson, C.E., Rivest, R.L. & Clifford, S. 2009. *Introduction To Algorithms*. The MIT Press: Massacusetts.

[7]Dawood, A.J. 2012. Improving Efficiency of Round Robin Scheduling Using Ascending Quantum and Minimum-Maximum Burst Time. *Journal of University of Anbar for Pure Science*. **6**: 1-5.

[8]Dhamdhere, D.M. 2006. *Operating System: A Concept Based Approach*. 2nd Edition. Tata McGraw-Hill Education: New Delhi.

[9]Hansen, P.B. 2001. *Operating Systems Principles*. Prentice Hall: New Jersey.

[10]Kumar, M.R., Rajenbra, B., Sreenatha, M. & Niranjan, C.K. 2014. An Improved Approach To Minimize Context Switching In Round Robin Scheduling Algorithm Using Optimize Technique. *International Journal of Research Engeenering and Technology*. **3**(4): 804-808.

[11]Nayak, D., Malla, S.K. & Debadarshini, D. 2012. Improved Round Robin Scheduling Using Dynamic Time Quantum. *International Journal of Computer Aplications*. **38**: 34-38.

[12]Noon, A., Kalakech, A. & Kadry, S. 2011. A New Round Robin Scheduling Algorithm For Operating System: Dynamic Quantum Using Mean Average. *International Journal of Computer Science Issue*. **3**(4): 224-229.