Jurnal Minfo Polgan
Volume 12, Nomor 2, September 2023 e-ISSN : 2797-3298
DOI : https://doi.org/10.33395/jmp.v12i2.12983 p-ISSN : 2089-9424

Terbit : 01 September 2023

Analisa Perbandingan Metode Penjadwalan Pada Pemrosesan Enkripsi Dan Dekripsi RSA Terdistribusi Dengan Arsitektur Klaster SBC

[1]Arief Prasetyo, [2] Sofyan Noor Arief, [3]Dhebys Suryani Hormansyah
[1, 2, 3]Politeknik Negeri Malang

[1]arief.prasetyo@polinema.com, [2]sofyan@polinema.com, [3]dhebys@polinema.com

ABSTRAK

Implementasi enkripsi menggunakan algoritma RSA memiliki masalah utama yaitu kebutuhan akan sumber daya komputasi yang besar. Optimasi pemrosesan algoritma RSA dapat dilakukan dengan cara membagi komputasinya secara terdistribusi dalam sebuah arsitektur sistem terdistribusi. Penelitian sebelumnya telah mengimplementasikan pemrosesan algoritma RSA secara terdistribusi pada arsitektur cluster Single Board Computer (SBC). Dan juga telah menerapkan algoritma penjadwalan proses FIFO/FCFS pada pemrosesannya. Namun, penerapan algoritma tersebut dirasa masih belum maksimal karena pada penelitian sebelumnya tidak terdapat proses Analisa hasil antar algoritma pemrosesan yang umum digunakan. Pada penelitian ini, empat buah algoritma penjadwalan proses akan diimplementasikan dan dianalisa. Algoritma tersebut antara lain algoritma FCFS/FIFO, algoritma SJF, algoritma Priority Scheduling dan algoritma Multi Level Queue Scheduling. Hasil pengujian yang dilakukan menunjukkan bahwa algoritma penjadwalan proses Shortest Job First (SJF) lebih baik karena dapat menurunkan rata-rata waktu tunggu pemrosesan hingga 29%. Karena adanya penurunan rata-rata waktu tunggu tersebut, algoritma SJF mampu menurunkan rata-rata waktu total pemrosesan hingga 21%.

Kata Kunci: RSA; Distributed; Cluster SBC; Penjadwalan Proses; Shortest Job First;

PENDAHULUAN

Algoritma RSA sampai saat ini masih menjadi metode yang paling populer digunakan untuk menjaga keamanan sebuah data. Kesulitan dalam meretas algoritma ini merupakan alasan utama dipakainya algoritma ini pada berbagai macam kasus penggunaan. Kerumitan persamaan matematis yang digunakan dalam algoritma ini menjadikannya sulit untuk diretas. Namun semakin rumit persamaan matematis yang dipakai maka semakin besar pula sumber daya komputasi yang dibutuhkan untuk memprosesnya. Algoritma RSA juga mempunyai karakteristik di mana algoritma ini memiliki keamanan yang baik bahkan nyaris sempurna, manajemen kunci yang mudah, mudah dalam pengimplementasian dan mudah untuk dipahami. Meneses (2016) mengatakan dalam algoritma ini, kekuatan modular memiliki andil besar dalam mempengaruhi kinerja algoritma, dan menjadi masalah utama yang menghambat penerapannya pada sebuah aplikasi yang lebih kompleks. Oleh karena itu, pemrosesan secara cepat dari algoritma enkripsi RSA (termasuk) telah menjadi fokus dalam berbagai penelitian.

Optimasi dalam pemrosesan secara cepat algoritma RSA dapat dilakukan dengan berbagai macam cara. Salah satunya dengan menjalankannya pada sebuat sistem terdistribusi. Lee (2009) menerangkan bahwa dalam sebuah sistem terdistribusi, sebuah pekerjaan akan dibagi dan dijalankan pada sekumpulan sumber daya komputasi. Sekumpulan sumber daya komputasi tersebut akan saling berkomunikasi dan berkolaborasi dalam memproses sebuah pekerjaan. Pada penelitian

Jurnal Minfo Polgan
Volume 12, Nomor 2, September 2023 e-ISSN : 2797-3298
DOI : https://doi.org/10.33395/jmp.v12i2.12983 p-ISSN : 2089-9424

yang telah dilakukan sebelumnya Arief (2020), sekumpulan komputer dalam bentuk komputer virtual digunakan untuk memproses enkripsi dengan menggunakan algoritma RSA. Dalam penelitian tersebut sebuah pekerjaan enkripsi di distribusikan ke masing-masing komputer pekerja. Pembagian pekerjaan enkripsi tersebut dibagikan secara merata pada setiap pekerja. Hasil dari penelitian tersebut membuktikan bahwa pemrosesan secara terdistribusi dapat meningkatkan kecepatan pemrosesan enkripsi RSA. Namun permasalahan baru muncul Ketika penggunaan komputer virtual tersebut diganti dengan sekumpulan komputer fisik. Penggunaan komputer fisik dalam jumlah yang banyak sangat tidak efisien baik dari segi biaya pembelian alat maupun segi biaya operasional. Oleh karena itu dibutuhkan sebuah sumber daya komputasi yang lebih murah dan rendah biaya operasional.

Masalah baru muncul ketika tugas untuk melakukan pekerjaan enkripsi datang secara bersamaan. Proses pekerjaan yang tidak terkendali dapat menyebabkan kerusakan data enkripsi yang berujung pada pemrosesan yang tak kunjung berakhir (terjadi pemrosesan secara terus menerus) pada komputer pekerja (baik berupa komputer personal tunggal maupun SBC). Pemrosesan secara terus menerus dan tak kunjung berakhir dapat menyebabkan penurunan performa pemrosesan enkripsi secara menyeluruh. Selain itu dapat menyebabkan kerusakan perangkat komputer yang dipakai untuk memproses enkripsi tersebut.

Penjadwalan proses merupakan pendekatan yang dapat digunakan untuk mengatasi permasalahan tersebut. Dengan adanya penjadwalan proses, sebuah proses akan dijalankan secara terstruktur dan terkendali. Sehingga akan meminimalisir adanya pemrosesan tak berujung. Banyak sekali algoritma penjadwalan proses yang saat ini ada, seperti algoritma First-Come First-Served (FCFS), algoritma Shortest-Job-Next (SJN), algoritma Priority Scheduling, dan algoritma Multiple-Level Queues Scheduling. Algoritma penjadwalan proses sendiri dilakukan dengan membentuk sebuah antrian (atau biasa disebut queue). Tanenbaum (2011) menjelaskan bahwa pada algoritma FIFO atau FCFS, proses antrian dibuat dengan meletakkan pekerjaan/proses berdasarkan waktu kedatangan. Sehingga pekerjaan/proses yang diterima pada waktu (t)-1 akan diproses terlebih dahulu dibandingkan dengan pekerjaan/proses yang diterima pada waktu (t). Algoritma SJN yang dijelaskan oleh Tanenbaum (2011), mengatakan bahwa pekerjaan akan diantrikan berdasarkan lama waktu penyelesaian yang dibutuhkan. Pekerjaan akan diurutkan berdasarkan pekerjaan yang mempunyai waktu pemrosesan paling kecil. Sednagkan menurut Tanenbaum (2011), algoritma Priority Scheduling mengurutkan pekerjaan berdsarkan tingkat prioritas dari pekerjaan yang datang. Maksudnya adalah jika ada dua buah pekerjaan yang datang, algoritma ini akan memeriksa skala proritas dari masing-masing pekerjaan tersebut. Jika prioritas dari sebuah pekerjaan lebih tinggi dibandingkan lainnya maka pekerjaan tersebut akan diantrikan paling awal. Menurut Tanenbaum (2011), diantara ketiga algoritma diatas, masing-masing memiliki kelebihan dan kekurangan masing-masing. Oleh karena itu muncullah sebuah algoritma baru yang menggabungkan beberapa dari algoritma penjadwalan proses yang telah ada. Algoritma tersebut kemudian disebut dengan Multiple-Level Queue Scheduling.

Pada penelitian yang telah dilakukan oleh Arief (2022), algoritma penjadwalan proses First In First Out (FIFO) atau bisa juga disebut algoritma First Come First Serve (FCFS) telah diterapkan dan mampu menyelesaikan permasalahan pemrosesan pekerjaan yang datang secara bersamaan pada satu satuan waktu. Namun realita yang ada dalam sebuah pekerjaan komputasi tidak hanya sebatas itu saja. Bagaimana cara menyelesaikan pekerjaan yang datang secara bersamaan pada satuan waktu namun memiliki skala prioritas masih belum terpecahkan. Selain itu, tingkat keberhasilan dalam mempercepat waktu respon masih belum bisa diperlihatkan karena pada penelitan tersebut hanya ada satu algoritma saja yang diterapkan.

Sebuah hipotesis dikemukakan oleh peneliti untuk menjawab permasalahan percepatan waktu respon dalam pemrosesan enkripsi yang datang secara bersamaan dengan berbagai macam besaran proses dan bentuk prioritasnya, yaitu dengan menerapkan dan membandingkan algoritma penjadwalan proses yang telah ada. Pada kasus enkripsi RSA pada arsitektur klaster SBC, algoritma penjadwalan yang paling relevan untuk diterapkan adalah algoritma FIFO (First In First Out) atau algoritma First-Come First-Served (FCFS), algoritma Shortest-Job-First (SJF), algoritma Priority Scheduling, dan algoritma Multiple-Level Queues Scheduling. Oleh karena itu pada

Jurnal Minfo Polgan
Volume 12, Nomor 2, September 2023 e-ISSN : 2797-3298
DOI : https://doi.org/10.33395/jmp.v12i2.12983 p-ISSN : 2089-9424

penelitian ini algoritma-algoritma tersebut akan diterapkan, diuji, dan dianalisa untuk mencari waktu respon terbaik. Sehingga nantinya dapat ditarik kesimpulan algoritma mana yang paling cocok diterapkan pada pemrosesan enkripsi dan dekripsi RSA secara terdistribusi pada arsitektur klaster SBC.

TINJAUAN PUSTAKA

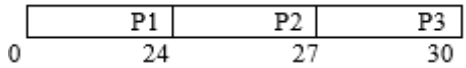
Algoritma FIFO / FCFS

Tanenbaum (2011) menjelaskan bahwa dalam algoritma penjadwalan proses FIFO, proses akan dilakukan secara berurutan sesuai dengan waktu kedatangan proses. Namun jika ada proses/pekerjaan yang datang bersamaan, akan dibuat mekanisme antrian. Misalnya, jika ada tiga pekerjaan P1, P2 dan P3 dengan lamanya waktu kerja CPU (CPU *Burst-time)* masing-masing sebagaimana pada Tabel 1 berikut :

Tabel 1. Proses waktu kerja CPU

Process	Burst Time
P1	24
P2	3
P3	3

Jika prosesnya berada dalam urutan P1, P2, P3 dan disajikan dengan algoritma FIFO maka dapat dijelaskan oleh *Bagan Gantt* sebagai berikut:



Gambar 1. Bagan Gantt

Jadi, jika waktu menunggu proses P1 adalah 0 milidetik. Sedangkan untuk proses P2, waktu tunggu adalah 24 milidetik. Dan untuk proses P3, waktu tunggu adalah 27 mili detik. Hal ini terjadi karena dalam algoritma FIFO, proses P2 dijalankan setelah proses P1 selesai selama 24 milidetik. Hal yang sama berlaku untuk proses P3. Waktu tunggu proses P3 adalah jumlah lamanya waktu proses yang sebelumnya berjalan, yaitu P1 dan P2. Jadi, dapat disimpulkan bahwa waktu tunggu rata-rata dari tiga proses adalah 17 milidetik. Selain waktu tunggu, parameter penting yang biasa dihitung dalam algoritma FIFO adalah waktu penyelesaian kerja (*Turn Around Time).* Waktu penyelesaian pekerjaan dihitung menggunakan rumus:

$$\text{TA} = Waiting\ Time + Length\ Of\ Execution.$$

Dimana *Waktu tunggu* adalah waktu tunggu proses, dan Lama eksekusi adalah lamanya waktu eksekusi proses. Jadi, dapat disimpulkan, Turn *Around Time* untuk proses P1 adalah 24 milidetik. Turn Around *Time* untuk proses P2 dan P3 masing-masing adalah 27 milidetik dan 30 milidetik. Rata-rata waktu turn-around dari tiga proses adalah 27 milidetik.

Algoritma SJF

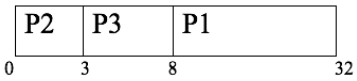
Tanenbaum (2011) menjelaskan bahwa dalam algoritma penjadwalan proses SJF, proses akan dilakukan dengan memproses terlebih dahulu pekerjaan-pekerjaan yang memiliki estimasi waktu pemrosesan paling pendek. Misalnya, jika ada tiga pekerjaan P1, P2 dan P3 dengan lamanya waktu kerja CPU (CPU *Burst-time)* masing-masing sebagaimana pada Tabel 2 berikut :

Tabel 2. Proses waktu kerja CPU

Process	Burst Time
P1	24
P2	3
P3	5

Jika prosesnya berada dalam urutan P1, P2, P3 dan disajikan dengan algoritma SJF maka dapat dijelaskan oleh *Bagan Gantt* sebagai berikut:

Jurnal Minfo Polgan
Volume 12, Nomor 2, September 2023 e-ISSN : 2797-3298
DOI : https://doi.org/10.33395/jmp.v12i2.12983 p-ISSN : 2089-9424



Gambar 2. Bagan Gantt

Jadi, jika waktu menunggu proses P1 adalah 8 milidetik. Sedangkan untuk proses P2, waktu tunggu adalah 0 milidetik. Dan untuk proses P3, waktu tunggu adalah 3 milidetik. Hal ini terjadi karena dalam algoritma SJF, proses P2 dijalankan terlebih dahulu kemudian dilanjukan dengan proses P3. Kemudian baru proses P1 akan dijalankan. Waktu tunggu proses P1 adalah jumlah lamanya waktu proses yang sebelumnya berjalan, yaitu P2 dan P3. Jadi, dapat disimpulkan, Turn *Around Time* untuk proses P1 adalah 32 milidetik. Turn Around *Time* untuk proses P2 dan P3 masing-masing adalah 3 milidetik dan 8 milidetik.

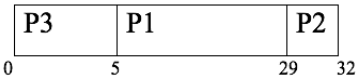
Algoritma Priority Scheduling

Tanenbaum (2011) menjelaskan bahwa dalam algoritma penjadwalan proses Priority Scheduling, proses akan dilakukan dengan memproses terlebih dahulu pekerjaan-pekerjaan yang memiliki prioritas paling mendesak. Misalnya, jika ada tiga pekerjaan P1, P2 dan P3 dengan lamanya waktu kerja CPU (CPU *Burst-time)* dan prioritas masing-masing pekerjaan adalah sebagaimana pada Tabel 3 berikut :

Tabel 3. Proses waktu kerja CPU

Process	Burst Time	Prority
P1	24	2
P2	3	3
P3	5	1

Jika prosesnya berada dalam urutan P1, P2, P3 dan disajikan dengan algoritma Priority Scheduling maka dapat dijelaskan oleh *Bagan Gantt* sebagai berikut:



Gambar 3. Bagan Gantt

Jadi, jika waktu menunggu proses P1 adalah 5 milidetik. Sedangkan untuk proses P2, waktu tunggu adalah 29 milidetik. Dan untuk proses P3, waktu tunggu adalah 0 milidetik. Hal ini terjadi karena dalam algoritma Priority Scheduling, proses P3 dengan prioritas paling tinggi dijalankan terlebih dahulu kemudian dilanjukan dengan proses P1. Kemudian baru proses P2 akan dijalankan. Waktu tunggu proses P1 adalah jumlah lamanya waktu proses yang sebelumnya berjalan, yaitu P2 dan P3. Jadi, dapat disimpulkan, Turn *Around Time* untuk proses P1 adalah 29 milidetik. Turn Around *Time* untuk proses P2 dan P3 masing-masing adalah 32 milidetik dan 0 milidetik.

Algoritma Multiple Lever Scheduling

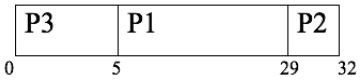
Tanenbaum (2011) menjelaskan bahwa dalam algoritma Multiple-Level Queue Scheduling berkerja dengan menggabungkan 2 atau lebih algoritma penjadwalan proses yang ada. Misalkan ketika ada beberapa pekerjaan yang datang secara bersamaan dengan prioritas yang berbeda dan ada pekerjaan lain yang datang kemudian dengan prioritas yang sama dengan prioritas salah satu pekerjaan sebelumnya sedangakan algoritma campuran yang digunakan adalah FCFS-Priority. Maka pekerjaan tersebut akan dibuatkan antrian berdasarkan waktu kedatangannya terlebih dahulu baru berdasarkan prioritasnya. Misalnya, jika ada tiga pekerjaan P1, P2 dan P3 dengan lamanya waktu kerja CPU (CPU *Burst-time)*, prioritas dan waktu kedatangan dari masing-masing pekerjaan adalah sebagaimana pada Tabel 4 berikut :

Tabel 4. Proses waktu kerja CPU

Process	Arrival	Burst Time	Prority
P1	0	24	2
P2	4	3	1
P3	0	5	1

Jurnal Minfo Polgan
Volume 12, Nomor 2, September 2023 e-ISSN : 2797-3298
DOI : https://doi.org/10.33395/jmp.v12i2.12983 p-ISSN : 2089-9424

Jika prosesnya berada dalam urutan P1, P2, P3 dan disajikan dengan algoritma Multi Level Queue Scheduling maka dapat dijelaskan oleh *Bagan Gantt* sebagai berikut:



Gambar 4. Bagan Gantt

Jadi, jika waktu menunggu proses P1 adalah 5 milidetik. Sedangkan untuk proses P2, waktu tunggu adalah 29 milidetik. Dan untuk proses P3, waktu tunggu adalah 0 milidetik. Hal ini terjadi karena dalam algoritma Multi Level Queue Scheduling, proses P3 dengan prioritas lebih tinggi dan datang lebih dahulu dijalankan terlebih dahulu . Kemudian dilanjukan dengan proses P1 dnegan waktu kedatangan yang sama dengan P3 namun memiliki prioritas yang lebih rendah. Kemudian baru proses P2 akan dijalankan. Waktu tunggu proses P1 adalah jumlah lamanya waktu proses yang sebelumnya berjalan, yaitu P2 dan P3. Jadi, dapat disimpulkan, Turn *Around Time* untuk proses P1 adalah 29 milidetik. Turn Around *Time* untuk proses P2 dan P3 masing-masing adalah 32 milidetik dan 0 milidetik.

METODE PENELITIAN

Desain Pengujian Testbed

Pada penelitian ini, sebuah skenario pengujian dibuat untuk mendapatkan data hasil pemrosesan yang sesuai dengan dunia nyata. Adapun skenario dibuat dengan menerapkan beberapa aspek penting yaitu variasi waktu kedatangan, variasi ukuran berkas, dan variasi prioritas pemrosesan. Adapun skenarioya tampak pada tabel 5 berikut.

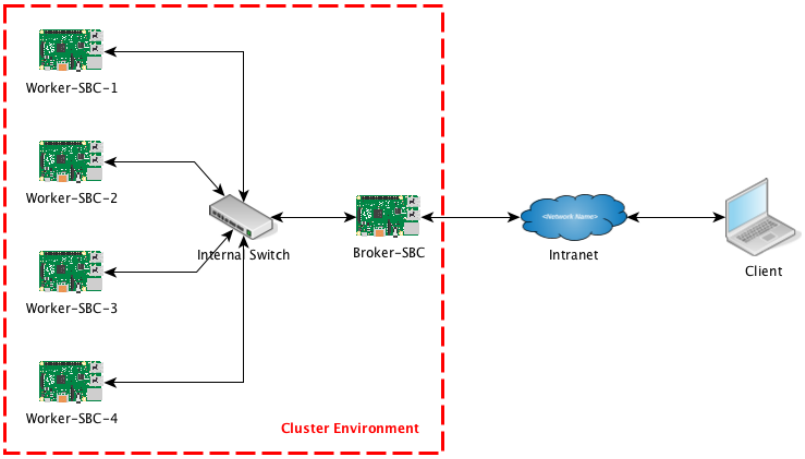
Tabel 5. Skenario Pengujian

Skenario	Waktu Kedatangan	Ukuran Pekerjaan	Skala Prioritas Pekerjaan
Skenario 1	0	10 MB	1
	1	30 MB	2
	0	50 MB	3
	1	80 MB	2
	0	100 MB	1
Skenario 2	0	100 MB	1
	1	80 MB	2
	0	50 MB	3
	1	30 MB	2
	0	10 MB	1
Skenario 3	0	10 MB	1
	1	50 MB	2
	0	30 MB	3
	1	80 MB	2
	0	100 MB	1
Skenario 4	0	100 MB	1
	1	80 MB	2
	0	30 MB	3
	1	50 MB	2
	0	10 MB	1
Skenario 5	0	10 MB	1
	1	80 MB	2
	0	100 MB	3
	1	30 MB	2
Skenario 6	0	10 MB	1
	1	30 MB	2
	0	100 MB	3
	1	80 MB	2
	0	50 MB	1
Skenario 7	0	30 MB	1
	1	10 MB	2
	0	100 MB	3
	1	50 MB	2
	0	80 MB	1
Skenario 8	0	30 MB	1
	1	50 MB	2
	0	100 MB	3
	1	10 MB	2
	0	80 MB	1
Skenario 9	0	100 MB	1
	1	50 MB	2
	0	10 MB	3
	1	30 MB	2
	0	80 MB	1
Skenario 10	0	100 MB	1
	1	30 MB	2
	0	10 MB	3
	1	50 MB	2

Jurnal Minfo Polgan
Volume 12, Nomor 2, September 2023 e-ISSN : 2797-3298
DOI : https://doi.org/10.33395/jmp.v12i2.12983 p-ISSN : 2089-9424

	0	50 MB	1		0	80 MB	1

Skenario tersebut akan diujikan pada arsitektur yang sama dengan penelitian sebelumnya yaitu sebagai berikut:



Gambar 5. Lingkungan Pengujian

Masing-masing SBC dalam arsitektur tersebut memiliki spesifikasi sebagai berikut:

1. Device Type : Raspberry Pi 4B 4GB
2. Processor : Broadcom BCM2711, Quad core Cortex-A72 SoC @ 1.5GHz
3. RAM : 4GB LPDDR4-3200 SDRAM
4. Storage : 16 GB
5. NIC : 10/100/1000 Mbps Ethernet
6. Sistem Operasi : RaspiOS

HASIL DAN PEMBAHASAN

Uji coba dilakukan sesuai dengan skenario yang telah dipaparkan pada bab sebelumnya. Uji coba tersebut dilakukan pada testbed yang sudah ditentukan. Implementasi algoritma penjadwalan diterapkan dalam kode program dibawah ini.

```
25  def check_available_job(algo):
26      conn = check_mysql_conn()
27      if conn is not None:
28          cur = conn.cursor()
29          if(algo is "FCFS"):
30              sql = 'SELECT t_jobs.*, t_user_files.size_of_user_files, t_user.type_of_user FROM t_jobs INNER JOIN
                    t_user_files ON t_jobs.t_user_files_id_user_files = t_user_files.id_user_files INNER JOIN t_user ON
                    t_user_files.t_user_id_user = t_user.id_user WHERE t_jobs.type_of_jobs=0 OR t_jobs.type_of_jobs=3 ORDER BY
                    date_enter_jobs ASC;'
31          elif(algo is "SJF"):
32              sql = 'SELECT t_jobs.*, t_user_files.size_of_user_files, t_user.type_of_user FROM t_jobs INNER JOIN
                    t_user_files ON t_jobs.t_user_files_id_user_files = t_user_files.id_user_files INNER JOIN t_user ON
                    t_user_files.t_user_id_user = t_user.id_user WHERE t_jobs.type_of_jobs=0 OR t_jobs.type_of_jobs=3 ORDER BY
                    size_of_user_files ASC;'
33          elif(algo is "PS"):
34              sql = 'SELECT t_jobs.*, t_user_files.size_of_user_files, t_user.type_of_user FROM t_jobs INNER JOIN
                    t_user_files ON t_jobs.t_user_files_id_user_files = t_user_files.id_user_files INNER JOIN t_user ON
                    t_user_files.t_user_id_user = t_user.id_user WHERE t_jobs.type_of_jobs=0 OR t_jobs.type_of_jobs=3 ORDER BY
                    type_of_user ASC;'
35          elif(algo is "MLQS"):
36              sql = 'SELECT t_jobs.*, t_user_files.size_of_user_files, t_user.type_of_user FROM t_jobs INNER JOIN
                    t_user_files ON t_jobs.t_user_files_id_user_files = t_user_files.id_user_files INNER JOIN t_user ON
                    t_user_files.t_user_id_user = t_user.id_user WHERE t_jobs.type_of_jobs=0 OR t_jobs.type_of_jobs=3 ORDER BY
                    type_of_user, size_of_user_files, date_enter_jobs ASC;'
37          cur.execute(sql)
38          res = cur.fetchone()
39          if res is not None:
40              return res
41          else:
42              return None
```

Jurnal Minfo Polgan
Volume 12, Nomor 2, September 2023 e-ISSN : 2797-3298
DOI : https://doi.org/10.33395/jmp.v12i2.12983 p-ISSN : 2089-9424

Gambar 6. Kode Program Penerapan Algoritma Penjadwalan

Dalam kode program tersebut, database digunakan sebagai alat bantu pencatatan pekerjaan yang akan dikerjakan. Dari database tersebut dapat dipilih pekerjaan mana yang harus dikerjakan sekarang dan jumlah pekerjaan yang harus dikerjakan berikutnya. Pemilihan pekerjaan yang akan dikerjakan didasarkan pada prilaku dari algoritma penjadwalan proses yang dipakai.

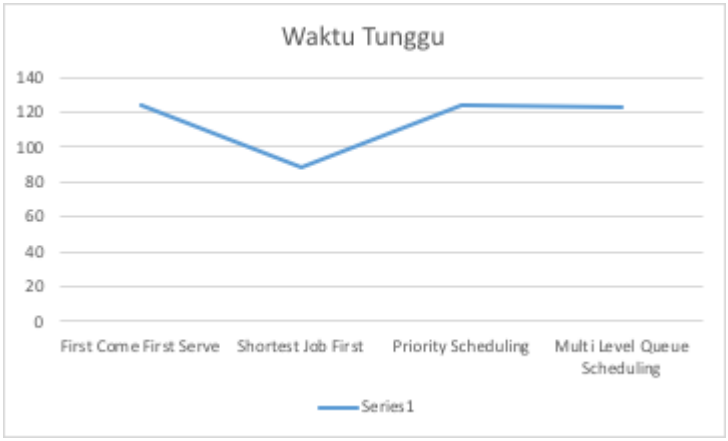
Untuk penjadwalan dengan algoritma First Come First Serve, pemilihan pekerjaan dilakukan dengan melakukan pengurutan pekerjaan yang ada pada database berdasarkan waktu kedatangan pekerjaan tersebut (date_enter_jobs). Sedangkan pada penjadwalan Shortest Job First, pemilihan pekerjaan dilakukan dengan melakukan pengurutan yang ada pada database berdasarkan ukuran berkasnya. Hal ini didasarkan pada prinsip bahwa semakin kecil ukuran berkas, maka semakin kecil pula waktu pemrosesannya.

Untuk algoritma Priority Scheduling, data dalam database diurutkan berdasarkan skala prioritasnya. Dan untuk algoritma yang terakhir yaitu Multi Level Queue Scheduling, data pada database pekerjaan, diurutkan berdasarkan level kriteria yang dipakai. Dalam penelitian ini, MLQS diterapkan dengan mendahulukan prioritas pekerjaan terlebi dahulu, kemudian ukuran pekerjaan dan terakhir waktu masuk pekerjaan.

Setelah implementasi algoritma dilakukan, pengujian dilakukan dan menghasilkan data waktu tunggu pemrosesan sebagai berikut.

Tabel 6. Waktu Tunggu Pemrosesan

Skenario	First Come First Serve	Shortest Job First	Priority Scheduling	Multi Level Queue Scheduling
1	118,6	92	139,4	130,6
2	156,8	100,2	116	111,4
3	99,2	85,2	126,4	130
4	113,6	86,8	136,6	133,2
5	111,6	89,2	89,2	95,4
6	124,2	87	105,6	96,6
7	146,4	87,4	108,2	112
8	134	87,2	113,6	110,4
9	129,4	86,6	165	163,6
10	107,6	77,2	140	142,4
Rata-Rata	124,14	87,88	124	122,56



Gambar 7. Grafik Rata-Rata Waktu Tunggu Pemrosesan

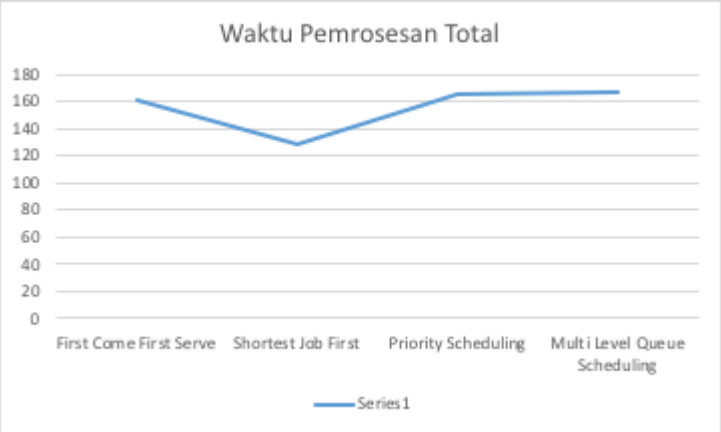
Jurnal Minfo Polgan
Volume 12, Nomor 2, September 2023 e-ISSN : 2797-3298
DOI : https://doi.org/10.33395/jmp.v12i2.12983 p-ISSN : 2089-9424

Dari grafik diatas terlihat bahwa rata-rata waktu tunggu paling kecil dari pengujian yang telah dilakukan adalah 87,9 milidetik. Rata-rata waktu tunggu tersebut didapatkan dengan implementasi algoritma pemrosesan Shortest Job First. Perbedaan waktu rata-rata algoritma tersebut dengan algoritma lainnya adalah 35,7 milidetik. Atau dapat disimpulkan bahwa terjadi penurunan waktu tunggu hingga 29%.

Selain waktu tunggu pemrosesan, data waktu pemrosesan total juga didapatkan dari hasil analisa data pengujian. Adapun data pemrosesan total disajikan pada tabel 7.

Tabel 7. Waktu Total Pemrosesan

Skenario	First Come First Serve	Shortest Job First	Priority Scheduling	Multi Level Queue Scheduling
1	157,6	136	186,8	178,6
2	207,2	151,6	152,4	150,2
3	134,6	123,6	168,4	174,6
4	150,2	127,6	180,2	178,6
5	148,4	129,6	129,6	141
6	161,2	127,6	148,6	142
7	182,8	127,8	151	157
8	170,6	127,2	156,6	155,2
9	166	126,6	207	208
10	134,2	107	173	178,2
Rata-Rata	161,28	128,46	165,36	166,34



Gambar 7. Grafik Rata-Rata Waktu Total Pemrosesan

Dari grafik total waktu pemrosesan diatas terlihat bahwa rata-rata waktu pemrosesan total paling kecil adalah 128,46 milidetik. Waktu paling kecil tersebut didapatkan dari pemrosesan menggunakan algoritma Shortest Job First. Perbedaan waktu rata-rata algoritma tersebut dengan algoritma lainnya adalah 35,9 milidetik. Atau dapat disimpulkan bahwa terjadi penurunan waktu total pemrosesan hingga 22%.

Berdasarkan hasil analisa data rata-rata waktu tunggu dan rata-rata waktu pemrosesan total dapat disimpulkan bahwa penerapan algoritma Shortest Job First lebih menguntungkan jika dibandingkan dengan algoritma-algoritma yang lain. Keuntungan yang didapatkan dari penerapan algoritma ini adalah penurunan waktu tunggu sebesar 29% dan penurunan waktu pemrosesan total sebesar 22%.

Jurnal Minfo Polgan
Volume 12, Nomor 2, September 2023 e-ISSN : 2797-3298
DOI : https://doi.org/10.33395/jmp.v12i2.12983 p-ISSN : 2089-9424

KESIMPULAN

Berdasarkan pengujian yang telah dilakukan, dapat ditarik kesimpulan bahwa dengan menerapkan algoritma Shortest Job First terjadi penurunan waktu tunggu sebesar 29% jika dibandingkan dengan algoritma lain yang diujikan. Dengan menerapkan algoritma Shortest Job First terjadi penurunan waktu pemrosesan total sebesar 22% jika dibandingkan dengan algoritma lain yang diujikan. Penerapan algoritma Shortest Job First lebih menguntungkan jika dibandingkan dengan algoritma-algoritma yang lain.

UCAPAN TERIMA KASIH

Terima kasih kepada tempat mengabdi kami di Politeknik Negeri Malang yang sudah memberikan motivasi terhadap kami sekaligus mendanai penelitian ini dan terima kasih kepada Jurusan Teknologi Informasi yang telah menyediakan tempat dan sarana penunjang lain yang kami gunakan dalam melaksanakan penelitian ini.

REFERENSI

Andrews, G.R., 1999. Foundations of Multithreaded, Parallel, and Distributed Programming, 1st ed. Addison-Wesley.

Arief, S.N., Firdaus, V.A.H., Prasetyo, A., 2020. Optimization of RSA encryption and decryption process with distributed computing method. IOP Conference Series: Materials Science and Engineering 830. https://doi.org/10.1088/1757-899X/830/2/022090

Błażewicz, Jacek., 2001. Handbook on Scheduling From Theory to Applications. 1st ed. Springer.

Devha, C., 2013. Pengamanan Pesan Rahasia Menggunakan Algoritma Kriptografi Rivest Shank Adleman (RSA). Universitas Pendidikan Indonesia 39–73.

Irvi, E., 2019. Rancang Bangun dan Evaluasi Kinerja Raspberry Pi Cluster sebagai Platform Penerapan Pembelajaran Mesin.

Lee, S., Lee, E., 2009a. Distributed Adaptation System for Quality Assurance of Web Service in Mobile Environment. JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 25, 403–417.

Lee, S., Lee, E., 2009b. Distributed Adaptation System for Quality Assurance of Web Service in Mobile Environment. JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 25, 403–417.

Meneses, F., Fuertes, W., Salvador, S., Flores, D., Aules, H., Castro, F., Torres, J., Miranda, A., Nuela, D., 2016. RSA Encryption Algorithm Optimization to Improve Performance and Security Level of Network Messages 8.

Munir, R., 2008. Pengantar Ilmu Kriptografi. https://doi.org/10.1017/CBO9781107415324.004

Prasetyo, A., Arief, S.N., Wakhidah, Rokhimatul, 2021. OPTIMASI PEMROSESAN ENKRIPSI DAN DEKRIPSI RSA PADA SINGLE BOARD COMPUTER (SBC) DENGAN PEMBAGIAN BEBAN KOMPUTASI DALAM SISTEM TERDISTRIBUSI. JIP (Jurnal Informatika Polinema).

Sukoco, H., Solahudin, M., Iqbal, M., 2015. Perbandingan Kinerja Pemrosesan Paralel Pada PC dan Raspberry Pi Untuk Pendeteksian Gulma Pada Lahan Pertanian Menggunakan Fraktal.

Stallings, William, 2009. Operating Systems Internals and Design Principles, 1st ed. Pearson Prentice Hall.

Tanenbaum, A.S., 2008. Modern Operating Systems, 1st ed. Pearson Prentice Hall.

Tanenbaum, A.S., Woodhull, A.S., 2011. Operating Systems Design and Implementation, 3rd ed. Pearson Education.

Basford, P. J., Johnston, S. J., Perkins, C. S., Garnock-Jones, T., Tso, F. P., Pezaros, D., … Cox, S. J. (2020). Performance analysis of single board computer clusters. Future Generation Computer Systems, 102. https://doi.org/10.1016/j.future.2019.07.040

Ching, P. L., Mutuc, J. E., & Jose, J. A. (2019). Assessment of the quality and sustainability implications of FIFO and LIFO inventory policies through system dynamics. Advances in Science, Technology and Engineering Systems, 4(5). https://doi.org/10.25046/aj040509

Hazay, C., Mikkelsen, G. L., Rabin, T., Toft, T., & Nicolosi, A. A. (2019). Efficient RSA Key Generation and Threshold Paillier in the Two-Party Setting. Journal of Cryptology, 32(2). https://doi.org/10.1007/s00145-017-9275-7

Zhong, X., & Liang, Y. (2016). Raspberry Pi: An effective vehicle in teaching the internet of things in computer science and engineering. Electronics (Switzerland). https://doi.org/10.3390/electronics5030056