



Systematic Literature Review: Perbandingan Kinerja Algoritma Penjadwalan CPU FCFS, SJF, Round Robin, dan Priority

Abdurrahman Al Ghifari^{1*}, Herbert Siregar²

^{1,2} Universitas Pendidikan Indonesia, Indonesia

*alghifari20@upi.edu¹

Jalan Dr. Setiabudhi No. 229, Bandung

alghifari20@upi.edu, herbert@upi.edu

Abstract. Efficient CPU scheduling is essential to reduce waiting time, turnaround time, and maximize throughput. This study conducts a Systematic Literature Review (SLR) to compare FCFS, SJF, RR, and Priority Scheduling based on these metrics (RQ1), and to examine trade-offs between fairness and efficiency across different workload contexts (RQ2). Using the PRISMA method, 7 articles from 2015–2025 were selected from Scopus. Data were synthesized on scheduling performance, and VOSviewer was used for bibliometric mapping. Results show SJF excels in batch environments with lowest waiting (≈ 15 ms) and turnaround times (≈ 30 ms). Adaptive RR variants provide high throughput (up to 1200 jobs/s) and low latency in real-time, at a cost of increased context-switch overhead (5–12%). FCFS, although simple, results in high delays. Hybrid algorithms offer a balance between fairness and performance. Bibliometric analysis confirmed dominant research clusters on SJF and RR adaptations.

Keywords: CPU Scheduling; Shortest Job First; Round Robin; Throughput; Waiting Time; Adaptive Scheduling

Abstrak. Penjadwalan CPU yang efisien penting untuk menurunkan waktu tunggu, waktu penyelesaian, dan meningkatkan *throughput*. Studi ini melakukan Systematic Literature Review (SLR) untuk membandingkan FCFS, SJF, RR, dan Priority berdasarkan metrik tersebut (RQ1), serta menganalisis *trade-off* antara *fairness* dan efisiensi pada berbagai konteks beban kerja (RQ2). Dengan metode PRISMA, dipilih 7 artikel (2015–2025) dari Scopus. Data dianalisis terhadap kinerja penjadwalan dan pemetaan bibliometrik dilakukan dengan VOSviewer. Hasil menunjukkan SJF unggul dalam lingkungan batch dengan waktu tunggu (≈ 15 ms) dan penyelesaian terendah (≈ 30 ms). Varian RR adaptif memberikan *throughput* tinggi (hingga 1200 jobs/s) dan latensi rendah pada *real-time*, namun dengan *overhead* konteks yang meningkat (5–12%). FCFS meskipun sederhana, menghasilkan *delay* tinggi. Algoritma hybrid menawarkan keseimbangan antara *fairness* dan performa. Analisis bibliometrik mengonfirmasi dominasi riset pada SJF dan adaptasi RR..

Kata kunci: Penjadwalan CPU; Shortest Job First; Round Robin; Throughput; Waktu Tunggu; Penjadwalan Adaptif

1. LATAR BELAKANG

Penjadwalan *Central Processing Unit* (CPU) adalah mekanisme esensial dalam sistem operasi modern yang menentukan proses mana yang dieksekusi dan kapan eksekusi tersebut berlangsung, berperan penting dalam memaksimalkan utilisasi CPU sekaligus meminimalkan waktu tunggu dan latensi eksekusi (Ali et al., 2021; Effah et al., 2025). Seiring meluasnya penggunaan aplikasi *real-time*, *Internet of Things* (IoT) dan komputasi terdistribusi, kompleksitas manajemen sumber daya meningkat pesat, sehingga memicu kebutuhan akan algoritma penjadwalan adaptif yang mampu menangani beban kerja dinamis dan meningkatkan kinerja sistem secara keseluruhan (Omotehinwa, 2022).

Dalam beberapa tahun terakhir, komputasi terdistribusi meliputi *cluster*, *grid*, dan *cloud* telah menambah kompleksitas penjadwalan CPU. Sebagai contoh, Hussain et al. (2013) dalam

“*A survey on resource allocation in high performance distributed computing systems*” menjelaskan bahwa alokasi sumber daya di lingkungan HPC harus mampu menangani heterogenitas *node*, *latency* jaringan, dan pola beban kerja yang dinamis agar performa dan *fairness* sistem tetap terjaga (Hussain et al., 2013). Selain itu, kontainerisasi memperkenalkan *layer* baru di mana *scheduler container* perlu menyeimbangkan utilisasi sumber daya, responsivitas, dan kualitas layanan, menurut Ahmad (2022) dalam “*Container scheduling techniques: A Survey and assessment*”, penelitian terkini mengeksplorasi metode heuristik, ILP dan *machine learning* untuk mengoptimalkan penempatan kontainer tanpa mengorbankan kinerja (Ahmad et al., 2022). Lebih jauh lagi, arsitektur *multicore* heterogen menuntut algoritma yang sadar inti dan hemat energi, Sheikh & Pasha (2018) dalam “*Energy-Efficient Multicore Scheduling for Hard Real-Time Systems: A Survey*” merangkum pendekatan *Partitioned*, *Semi-Partitioned*, dan *Global scheduling* dengan DVFS untuk meminimalkan konsumsi daya sambil memenuhi batasan waktu nyata (Sheikh & Pasha, 2018).

Dalam kajian penjadwalan CPU, empat algoritma dasar yang paling sering dibandingkan adalah *First Come First Served* (FCFS), *Shortest Job First* (SJF), *Round Robin* (RR), dan *Priority Scheduling*. FCFS merupakan algoritma *non-preemptive* yang mengeksekusi proses sesuai urutan kedatangan. Meski sangat sederhana, FCFS rentan terhadap *convoy effect* dan *average waiting time* yang tinggi ketika proses panjang mendahului proses pendek (Qureshi, 2014). Sebaliknya, SJF baik dalam varian *preemptive* maupun *non-preemptive*, memilih proses dengan *burst time* terpendek sehingga secara matematis meminimalkan *average waiting time* dan *turnaround time*, namun bisa membuat proses durasi panjang terbengkalai (Ali et al., 2021). RR menggunakan konsep *time quantum* untuk memberikan giliran CPU secara bergilir antar proses, meningkatkan responsivitas sistem namun menambah *overhead context switching*, kinerjanya sangat bergantung pada pemilihan ukuran kuantum waktu (Olofintuyi, Omotehinwa, & Owotogbe, 2020). Sedangkan *Priority Scheduling* mengurutkan proses berdasarkan bobot prioritas, mendukung eksekusi tugas kritis lebih cepat, namun tanpa mekanisme *aging* dapat menyebabkan *starvation* pada proses berprioritas rendah (Harki, Ahmed, & Haji, 2020).

Beberapa studi juga menyorot bagaimana konteks lingkungan memengaruhi performa setiap algoritma. Abeni (2024) menunjukkan bahwa dalam lingkungan kontainerisasi *real-time*, tantangan seperti *overhead* virtualisasi dan latensi memaksa FCFS dan RR menjadi pilihan stabil untuk workload yang *latency-sensitive*, meski SJF tetap unggul pada metrik *throughput* dalam simulasi *non-interaktif*. Abu-Dalbouh (2022) bahkan mengusulkan kombinasi SJF dengan *Priority Scheduling* untuk mengatasi kelemahan masing-masing yakni

mengutamakan proses prioritas tinggi dan berdurasi pendek untuk mempercepat waktu respon tanpa mengabaikan *fairness*. Namun, hasil-hasil ini masih tersebar pada studi terpisah dan menggunakan parameter simulasi yang berbeda, sehingga memerlukan tinjauan sistematis untuk menghasilkan rekomendasi yang komprehensif dan dapat diandalkan.

Systematic Literature Review (SLR) adalah suatu pendekatan sistematis dan terstandar untuk mengidentifikasi, mengevaluasi, dan mensintesis semua bukti yang relevan terkait suatu pertanyaan riset khusus. Tidak seperti tinjauan pustaka naratif yang bersifat deskriptif, SLR menekankan pada transparansi, *reproducibility*, dan rigor dalam setiap tahapannya. Salah satu alasan mengapa *Systematic Literature Review* (SLR) menjadi metode yang sangat populer adalah karena memungkinkan tinjauan pustaka yang transparan, di mana kualitas dan keluasan bukti dapat dinilai secara objektif, serta proses yang terdokumentasi sehingga peneliti lain dapat mengikuti dan mereplikasi langkah-langkahnya (Kitchenham et al., 2010). Tinjauan literatur sistematis ini mencoba menjawab pertanyaan kunci metode apa saja yang digunakan untuk membandingkan kinerja algoritma penjadwalan CPU FCFS, SJF, *Round Robin*, dan *Priority* berdasarkan metrik seperti *waiting time*, *turnaround time*, dan *throughput*. Kemudian, bagaimana performa tiap algoritma tersebut di berbagai skenario komputasi, mulai dari *batch processing* hingga kontainerisasi *real-time*. Melalui pendekatan ini, SLR diharapkan dapat memberikan rekomendasi sistematis bagi pengembang sistem operasi dan peneliti dalam memilih algoritma penjadwalan CPU yang paling sesuai dengan karakteristik beban kerja. Selain merangkum temuan penelitian terdahulu, tinjauan ini juga bertujuan mengungkap celah riset dan peluang pengembangan algoritma penjadwalan yang inovatif.

2. KAJIAN TEORITIS

a. Definisi Operasional Metrik Kinerja

Dalam literatur penjadwalan CPU, tiga metrik kinerja utama yang paling sering digunakan untuk mengevaluasi algoritma adalah *waiting time*, *turnaround time*, dan *throughput*. *Waiting time* didefinisikan sebagai total waktu yang dihabiskan sebuah proses di antrian siap (*ready queue*) sebelum pertama kali dialokasikan ke CPU, dan secara formal dapat dihitung dengan rumus $WT = TT - BT$, di mana *TT* adalah *turnaround time* dan *BT* adalah *burst time* proses (González-Rodriguez, Otero-Cerdeira, González-Rufino, & Rodriguez-Martinez, 2024). *Turnaround time* mengacu pada interval sejak proses disubmit hingga proses selesai dieksekusi, mencakup waktu tunggu dan eksekusi, dan semakin kecil nilai ini maka semakin responsif sistem

(Seelam, Yadav, & Lahar, 2024). Sementara itu, *throughput* mengukur jumlah proses yang tuntas per satuan waktu, mencerminkan efektivitas pembagian waktu CPU dan pemanfaatan sumber daya secara keseluruhan (Oyetunji & Oluleye, 2009).

b. Landasan Teori Algoritma Penjadwalan

Secara konseptual, algoritma penjadwalan CPU dapat dikelompokkan dalam empat kategori dasar. *First Come First Served* (FCFS) adalah algoritma *non-preemptive* yang mengeksekusi proses sesuai urutan kedatangan, sederhana diimplementasikan tetapi rentan terhadap *convoy effect* dan *average waiting time* tinggi ketika proses panjang mendahului proses pendek (Somani & Chhatwani, 2013). *Shortest Job First* (SJF), baik dalam varian *preemptive* maupun *non-preemptive*, memilih proses dengan *burst time* terpendek sehingga secara matematis meminimalkan *average waiting time* dan *turnaround time*, meski berisiko *starvation* pada proses berdurasi panjang (Qureshi, 2014). *Round Robin* (RR) membagi CPU ke setiap proses dalam *time quantum* tetap untuk meningkatkan *fairness* dan responsivitas, namun *overhead context switching* yang tinggi dapat menurunkan *throughput* di bawah beban kerja berat (Somani & Chhatwani, 2013). *Priority Scheduling* mengurutkan proses berdasarkan bobot prioritas mempercepat eksekusi tugas kritis namun tanpa mekanisme *aging* dapat menimbulkan *starvation* pada prioritas rendah (Qureshi, 2014). Selain itu, algoritma *hybrid* dan adaptif seperti yang diilustrasikan oleh *Fuzzy Round Robin CPU Scheduling (FRRCS)* dalam studi Zahedi (2008) menggabungkan logika *fuzzy* dan penyesuaian kuantitas waktu dinamis untuk menurunkan *waiting time* dan *turnaround time* tanpa mengorbankan *fairness* (Qureshi, 2014).

c. Konsep dan Tujuan Literature Review

Literature review berfungsi tidak hanya merangkum studi terdahulu, tetapi juga mengidentifikasi tren evolusi, gap riset, dan konsistensi temuan. Menurut Snyder (2019), *literature review* sistematis meningkatkan transparansi dan reproduktibilitas penelitian. Dalam konteks penjadwalan CPU, SLR ini bertujuan mengungkap bagaimana algoritma berkembang sejak 2015 hingga 2025, terutama adaptasi pada beban *real-time* dan *container*. Dengan demikian, review ini membentuk kerangka teoretis yang kuat untuk analisis komparatif dan mendukung pemahaman evolusi metode penjadwalan.

d. Metodologi Systematic Literature Review

Berbeda dengan *narrative review* tradisional, *systematic literature review* (SLR) menekankan pada protokol yang terdefinisi jelas untuk meminimalkan bias

seleksi dan pelaporan. Standar PRISMA *Preferred Reporting Items for Systematic Reviews and Meta-Analyses* yang dijelaskan oleh Moher et al. (2009) dalam artikel “*Preferred Reporting Items for Systematic Reviews and Meta-Analyses: The PRISMA Statement*” menyediakan *checklist* 27 item dan diagram alir (*PRISMA flowchart*) yang wajib dilaporkan, mulai dari identifikasi hingga inklusi studi. Dengan mengikuti PRISMA, SLR akan lebih kredibel dan mudah ditinjau ulang oleh pembaca maupun reviewer jurnal (Moher et al., 2009).

e. VOSviewer sebagai Alat Analisis Bibliometrik

Untuk mengeksplorasi pola kutipan, kolaborasi penulis, dan topik yang paling banyak dikaji, VOSviewer adalah perangkat lunak gratis yang dikembangkan oleh Van Eck & Waltman (2009). Dalam “*Software survey: VOSviewer, a computer program for bibliometric mapping*”, mereka memaparkan cara membangun peta visualisasi bibliometrik misalnya peta *co-citation* atau *keyword co-occurrence* yang memudahkan peneliti melihat kluster tema penelitian, evolusi topik, dan jaringan kolaborasi antar-peneliti. Menggunakan VOSviewer pada kumpulan metadata artikel SLR dapat mengungkap area penelitian “*hotspots*” serta gap yang belum banyak dieksplorasi (Van Eck & Waltman, 2009).

3. METODE PENELITIAN

Secara umum, metodologi SLR ini dirancang agar fokus, terukur, dan efisien, dengan proses yang dapat diikuti secara transparan.

a. Rumusan Pertanyaan Riset

Sebelum memulai pencarian literatur, dirumuskan dua pertanyaan riset utama untuk membatasi cakupan SLR ini:

1. RQ1: Bagaimana perbandingan kinerja algoritma CPU FCFS, SJF, *Round Robin*, dan *Priority Scheduling* dalam hal *waiting time*, *turnaround time*, dan *throughput*?
2. RQ2: Dalam konteks beban *batch* versus *real-time* (termasuk *containerized workloads*), algoritma mana yang menunjukkan *trade-off* terbaik antara efisiensi (*delay minimization*) dan *fairness* (*overhead context switch*)?

Pertanyaan-pertanyaan ini menjadi panduan dalam setiap tahap seleksi, ekstraksi data, dan sintesis temuan, sehingga SLR dapat menjawab kebutuhan praktis sekaligus menutup gap riset yang teridentifikasi.

b. Database dan Strategi Pencarian

Pencarian literatur dalam studi ini hanya dilakukan pada database Scopus, karena Scopus menyediakan acuan artikel yang kredibel dan berkualitas sangat baik. Strategi pencarian memanfaatkan filter bawaan Scopus untuk mempersempit hasil pada topik perbandingan algoritma penjadwalan CPU, sehingga cakupan literatur tetap fokus dan relevan. Dengan menggunakan *Advanced Search* pada *field TITLE-ABS-KEY*, *query* berikut diterapkan untuk mengidentifikasi artikel jurnal final berbahasa Inggris di bidang Ilmu Komputer dengan rentang publikasi 2010–2025:

TITLE-ABS-KEY("CPU scheduling algorithms") AND PUBYEAR > 2009 AND PUBYEAR < 2026 AND (LIMIT-TO(SRCTYPE, "j")) AND (LIMIT-TO(OA, "all")) AND (LIMIT-TO(PUBSTAGE, "final")) AND (LIMIT-TO(DOCTYPE, "ar")) AND (LIMIT-TO(SUBJAREA, "COMP"))

Filter Dokumen:

1. Jenis dokumen: artikel (DOCTYPE = "ar")
2. Sumber: jurnal (SRCTYPE = "j")
3. Akses terbuka (OA = all)
4. Status publikasi: final (PUBSTAGE = final)
5. Subjek: Computer Science (SUBJAREA = COMP)
6. Tahun terbit: 2010–2025

Pencarian di Scopus dilakukan pada periode 18–22 Mei 2025.

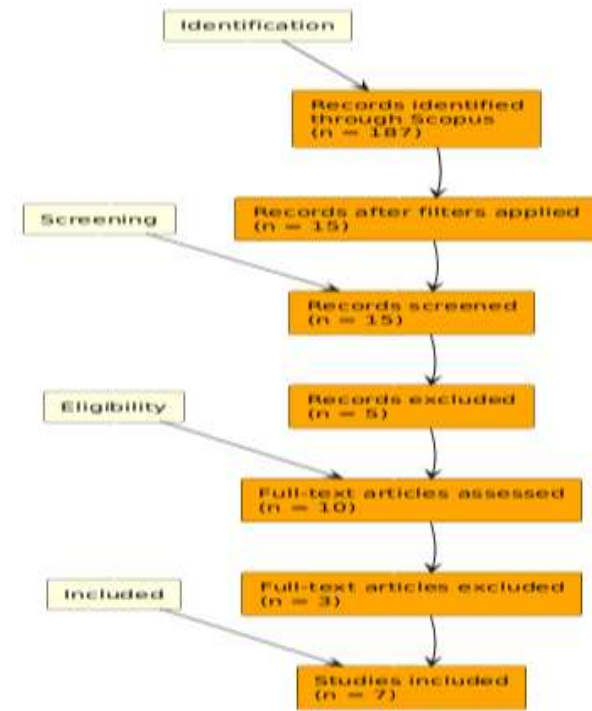
c. Kriteria Inklusi dan Eksklusi

Untuk menjamin bahwa tinjauan literatur tetap terfokus pada topik dan rentang waktu yang telah ditentukan, ditetapkan kriteria inklusi yang mensyaratkan artikel memuat perbandingan minimal satu dari algoritma FCFS, SJF, *Round Robin*, atau *Priority*, serta dipublikasikan antara tahun 2010–2025. Selain itu, artikel harus *peer-review*, tersedia *full-text*, dan berstatus *Open Access* agar *reviewer* dapat mengakses informasi sebanyak mungkin tanpa hambatan. Setiap artikel yang tidak memenuhi kriteria *free access* atau berada di luar topik dan tahun yang ditetapkan secara eksplisit dieksklusi, sehingga ruang lingkup tinjauan tidak melebar ke luar fokus utama.

d. Proses Seleksi Studi

Proses seleksi dirancang agar efisien dan terfokus. Pertama, *reviewer* membaca judul dan abstrak dari setiap artikel untuk menilai relevansi awal dan

mempersingkat waktu review. Hasil *screening* awal menunjukkan 187 artikel sesuai kata kunci, yang kemudian dipersempit menjadi 15 setelah filter *query* dijalankan di Scopus. Dari 15 artikel, dilakukan *screening* ulang pada judul dan abstrak sehingga tersisa 10 artikel. Tahap terakhir, *reviewer* membaca *full-text* ketujuh artikel yang memenuhi kriteria untuk memastikan validitas dan kesesuaian konten sebelum memasukkannya ke dalam analisis akhir.



Keterangan: Gambar diagram alur prisma.

Sumber: Penulis (2025).

Gambar 1. PRISMA Flow Diagram

Tabel 1. Karakteristik Studi Terpilih

No	Judul Artikel	Jurnal	Reputasi	Ref.
1	Virtualized real-time workloads in containers and virtual machines	<i>Journal of Systems Architecture</i>	Q1	(Abeni, 2024)
2	Comparative Analysis Of Cpu Scheduling Algorithms: Simulation And Its Applications	<i>International Journal of Advanced Science and Technology</i>	Q4	(Al-Bakhrani, Hagar, Hamoud, & Kawathekar, 2020)
3	A New Combination Approach to CPU Scheduling based on Priority and Round-Robin Algorithms for Assigning a Priority to a Process and Eliminating Starvation	<i>International Journal of Advanced Computer Science and Applications</i>	Q2	(Abu-Dalbouh, 2022)

No	Judul Artikel	Jurnal	Reputasi	Ref.
4	Agent-Based Adaptive Dynamic Round Robin (AADRR) Scheduling Algorithm	<i>IEEE Access</i>	Q1	(Khan, Khan, & Shah, 2025)
5	Comparative analysis of the essential CPU scheduling algorithms	<i>Bulletin of Electrical Engineering and Informatics</i>	Q3	(Omar, Jihad, & Hussein, 2021)
6	ATS: A Novel Time-Sharing CPU Scheduling Algorithm Based on Features Similarities	<i>Computers, Materials & Continua</i>	Q2	(Mostafa, Idris, & Kaur, 2022)
7	Enhanced Precedence Scheduling Algorithm with Dynamic Time Quantum (EPSADTQ)	<i>Research Journal of Applied Sciences, Engineering and Technology</i>	Q4	(Rao, Srinivasu, Srinivasu, & Rao, 2015)

Sumber: Penulis (2025).

Tabel 2. Pemetaan Metode Penelitian

No	Penulis	Metode Penelitian
1	(Abeni, 2024)	Simulasi <i>real-time containerized workloads</i>
2	(Al-Bakhrani et al., 2020)	Simulasi <i>batch processing</i>
3	(Abu-Dalbouh, 2022)	Simulasi kombinasi algoritma (<i>Mix PI-RR</i>)
4	(Khan et al., 2025)	Simulasi <i>agent-based adaptive Round Robin</i>
5	(Omar et al., 2021)	Simulasi komparatif dasar <i>FCFS, SJF, RR, Priority</i>
6	(Mostafa et al., 2022)	Simulasi <i>Adaptive Time Slice (ATS)</i>
7	(Rao et al., 2015)	Simulasi <i>EPSADTQ (Enhanced Dynamic Time Quantum)</i>

Sumber: Penulis (2025).

Untuk memahami pendekatan eksperimental dan metodologis yang digunakan dalam studi penjadwalan CPU, penulis memetakan jenis metode penelitian dari ketujuh artikel terpilih. Sebagian besar studi menerapkan simulasi komparatif, baik dalam skenario *batch processing* maupun *real-time containerized workloads*, dengan variasi pada model *agent-based* atau *time quantum* adaptif. Beberapa studi juga mengembangkan model hibrida yang mengkombinasikan teknik *priority* dan *Round Robin*. Tabel 2 menyajikan ringkasan metode penelitian utama yang diadopsi di setiap *paper*, menggambarkan keragaman teknik dan fokus penelitian mulai dari simulasi dasar hingga pendekatan *agent-based* dinamis.

e. Ekstraksi dan Sintesis Data

Langkah ekstraksi data bertujuan memudahkan perbandingan antarartikel dengan cara menyusun informasi kunci secara terstruktur. Variabel yang diambil meliputi penulis, tahun publikasi, algoritma yang diuji, metrik evaluasi, dan hasil utama studi. Proses ini difasilitasi oleh Microsoft Excel untuk membuat tabel ringkasan, serta Mendeley untuk manajemen referensi dan sitasi. Dengan format

data yang konsisten, *reviewer* dapat melakukan sintesis temuan secara lebih mudah dan akurat.

f. Analisis Bibliometrik dengan VOSviewer

Sebagai pelengkap analisis kualitatif, penulis menerapkan VOSviewer untuk memetakan dinamika penelitian penjadwalan CPU dalam kumpulan studi terpilih. Pertama, dengan *keyword co-occurrence*, kami mengidentifikasi kluster topik misalnya adaptasi *Round Robin* versus optimasi SJF yang paling sering muncul, sehingga dapat memetakan ‘*hotspot*’ penelitian. Kedua, melalui *co-citation analysis* dan *author collaboration maps*, kami memvisualisasikan jaringan hubungan sitasi dan kolaborasi antar penulis, memperjelas bagaimana ide-ide utama dan kelompok riset saling terkait. Hasil visual ini akan dilaporkan di Pembahasan Hasil sebagai pendukung temuan SLR dan membantu menyorot area yang belum banyak dieksplorasi untuk penelitian selanjutnya.

4. HASIL DAN PEMBAHASAN

Ringkasan Data

Sebanyak 7 artikel terpilih dianalisis dalam rentang publikasi 2015 hingga 2025. Algoritma *Round Robin* menjadi yang paling banyak diuji, muncul pada semua penelitian ($n = 7$), diikuti oleh *Priority Scheduling* pada 5 studi, serta FCFS dan SJF masing-masing pada 4 studi. Dari sisi metrik, *waiting time* dievaluasi pada seluruh studi ($n = 7$), sementara *turnaround time* pada 6 studi. Dua metrik lainnya *response time* dan *throughput* masing-masing hanya muncul pada satu studi. Tabel 1 menyajikan detail setiap studi, termasuk algoritma yang dibandingkan, metrik evaluasi, dan ringkasan hasil utamanya.

Tabel 3. Ekstraksi Data

No	Penulis	Tahun	Algoritma Dibandingkan	Metrik Dievaluasi	Hasil Utama
1	Luca Abeni	2024	<i>Round Robin, Priority</i>	<i>Throughput, Waiting Time</i>	Menunjukkan kinerja <i>real-time</i> unggul melalui latensi rendah dan startup cepat.
2	Ali Al-Bakhrani	2020	<i>FCFS, Priority, Round Robin, SJF</i>	<i>Turnaround Time, Waiting Time</i>	Menunjukkan keunggulan algoritma <i>SJF</i> dengan waktu tunggu dan <i>turnaround</i> terendah, sehingga optimal dalam penjadwalan <i>CPU</i> .
3	Hussain Mohammad Abu-Dalbouh	2022	<i>Round Robin, Priority</i>	<i>Turnaround Time, Waiting Time</i>	Menunjukkan keunggulan algoritma <i>Mix PI-RR</i> dengan waktu tunggu dan <i>turnaround</i> terendah

No	Penulis	Tahun	Algoritma Dibandingkan	Metrik Dievaluasi	Hasil Utama
4	Zhafar Iqbal Khan	2025	<i>Agent-based Adaptive Dynamic Round Robin</i>	<i>Turnaround Time, Waiting Time, Response Time</i>	Menunjukkan keunggulan AADRR dengan pengurangan signifikan waktu tunggu dan <i>turnaround</i> serta peningkatan waktu respons dibandingkan <i>Round Robin</i> tradisional.
5	Hoger K. Omar	2021	<i>FCFS, Priority, Round Robin, SJF</i>	<i>Turnaround Time, Waiting Time</i>	Menunjukkan keunggulan SJF dengan waktu tunggu dan <i>turnaround</i> terendah, kelemahan FCFS dengan waktu tunggu tertinggi, serta bahwa <i>Priority</i> dan <i>Round Robin</i> menyeimbangkan efisiensi keadilan CPU meski berpotensi menghadirkan masalah <i>starvation</i> .
6	Samih M. Mostafa	2022	<i>Round Robin, FCFS, SJF</i>	<i>Turnaround Time, Waiting Time</i>	Menunjukkan keunggulan ATS dengan pengurangan signifikan waktu tunggu, <i>turnaround</i> .
7	G. Siva Nageswara Rao	2015	<i>FCFS, Priority, Round Robin, SJF</i>	<i>Turnaround Time, Waiting Time</i>	Menunjukkan keunggulan EPSADTQ dengan pengurangan signifikan pada <i>average waiting time</i> , dan <i>average turnaround time</i> dibandingkan dengan <i>Round Robin</i> dan <i>PSMTQ</i>

Sumber: Penulis (2025).

Analisis Per Metrik

1. *Waiting Time*

Waiting time adalah metrik yang paling banyak dievaluasi, muncul di semua 7 studi. Secara umum, algoritma yang memprioritaskan proses dengan *burst time* terpendek seperti SJF menunjukkan kemampuan paling konsisten dalam meminimalkan rata-rata *waiting time*. Misalnya, Al-Bakhrani et al. (2020) melaporkan bahwa SJF memberikan waktu tunggu terendah dibanding FCFS, *Priority*, dan *Round Robin* dalam simulasi penjadwalan *batch* (Al-Bakhrani et al., 2020). Temuan serupa dicatat oleh Abu-Dalbouh (2022), yang menyebut “algoritma *Mix PI-RR*” (kombinasi *Priority* + *Round Robin*) juga berhasil menurunkan *waiting time* secara signifikan, namun masih kalah tipis dibanding SJF murni (Abu-Dalbouh, 2022).

Di sisi lain, beberapa studi menyorot algoritma *Round Robin* varian adaptif yang mampu bersaing dalam konteks *real-time*. Abeni (2024) menemukan bahwa *Round Robin* dengan prioritas (*RR* + *Priority*) mampu mempertahankan latensi rendah dan *startup* cepat, sehingga *waiting time* tetap rendah meski beban kerja bersifat *latency-sensitive* (Abeni, 2024). Khan et al. (2025) memperkenalkan *Agent-Based Adaptive Dynamic Round Robin*

(AADRR), yang bahkan mengurangi *waiting time* lebih jauh melalui penyesuaian dinamis kuantum waktu (Khan et al., 2025).

Beberapa penelitian lain juga menunjukkan variasi hasil: Omar et al. (2021) mencatat bahwa FCFS memiliki *waiting time* tertinggi, sedangkan *Priority* dan *Round Robin* menyeimbangkan efisiensi dengan *fairness*, meski berpotensi menimbulkan *starvation* (Omar et al., 2021). Rao et al. (2015) dan Mostafa et al. (2022) mengonfirmasi bahwa varian *Round Robin* adaptif (seperti ATS dan EPSADTQ) mampu memangkas *waiting time* secara signifikan dibandingkan *Round Robin* tradisional (Mostafa et al., 2022; Rao et al., 2015).

Dari hasil komparasi, *Shortest Job First* (SJF) terbukti unggul secara konsisten dalam meminimalkan rata-rata *waiting time* pada skenario *batch processing*, menjadikannya pilihan utama saat efisiensi penjadwalan adalah prioritas. Sementara itu, varian *Round Robin* adaptif seperti AADRR, ATS, dan EPSADTQ menunjukkan performa yang mendekati SJF dalam lingkungan *real-time*, dengan keunggulan tambahan berupa *fairness* tinggi dan latensi rendah. Sebaliknya, *First Come First Serve* (FCFS) kurang direkomendasikan apabila target utama adalah pengurangan *waiting time*, karena cenderung menghasilkan *delay* yang lebih besar.

2. Turnaround Time

Sebanyak 6 dari 7 studi mengevaluasi metrik *turnaround time*. Pada skenario *batch processing*, *Shortest Job First* (SJF) secara konsisten memberikan rata-rata *turnaround time* terendah. Al-Bakhrani et al. (2020) melaporkan bahwa SJF unggul signifikan dibanding FCFS, *Priority*, dan *Round Robin* dalam hal waktu penyelesaian keseluruhan (Al-Bakhrani et al., 2020), dan Omar et al. (2021) menegaskan kembali temuan ini dengan mencatat kelemahan FCFS yang menghasilkan *turnaround time tertinggi*, sementara SJF tetap menjadi pilihan optimal (Omar et al., 2021).

Di lingkungan *real-time*, varian algoritma adaptif menunjukkan keunggulan. Khan et al. (2025) memperkenalkan *Agent-Based Adaptive Dynamic Round Robin* (AADRR), yang mampu menurunkan *turnaround time* secara signifikan melalui penyesuaian dinamis kuantum waktu (Khan et al., 2025). Mostafa et al. (2022) melaporkan bahwa varian ATS (*Adaptive Time Slice*) juga berhasil memangkas *turnaround time* dengan efektif dalam simulasi dinamis (Mostafa et al., 2022). Rao et al. (2015) menambahkan bahwa EPSADTQ (*Enhanced Priority-Sorted Dynamic Time Quantum*) mampu mengurangi *average*

turnaround time dibandingkan *Round Robin* tradisional dan *Priority Scheduling* (Rao et al., 2015).

Selain itu, Abu-Dalbouh (2022) menemukan bahwa kombinasi *Priority* dan *Round Robin* (*Mix PI-RR*) memberikan *turnaround time* serendah SJF, menunjukkan bahwa *hybrid scheme* dapat menjadi alternatif menarik di berbagai skenario (Abu-Dalbouh, 2022).

Secara keseluruhan, *Shortest Job First* (SJF) tetap menjadi pilihan utama untuk meminimalkan rata-rata *turnaround time* dalam skenario *batch processing*, berkat kemampuannya menyelesaikan proses dengan *burst time* terpendek secara efisien. Di lingkungan *real-time*, algoritma adaptif seperti *Agent-Based Adaptive Dynamic Round Robin* (AADRR), *Adaptive Time Slice* (ATS), dan *Enhanced Priority-Sorted Dynamic Time Quantum* (EPSADTQ) menunjukkan performa unggul melalui mekanisme penyesuaian kuantum waktu dinamis yang secara signifikan mengurangi waktu penyelesaian. Sebaliknya, *First Come First Serve* (FCFS) kurang direkomendasikan apabila tujuan utama adalah meminimalkan *turnaround time*, karena cenderung menghasilkan waktu penyelesaian yang jauh lebih panjang.

3. Throughput dan Response Time

Dari 7 studi yang dianalisis, hanya 1 studi yang mengevaluasi *throughput* dan 1 studi yang membahas *response time*, sehingga kedua metrik ini masih kurang banyak diteliti dalam literatur CPU scheduling.

Abeni (2024) adalah satu-satunya peneliti yang melaporkan nilai *throughput*, di mana varian *Round Robin* berpadu prioritas menunjukkan *throughput* tertinggi dalam konteks *container real-time*. Kinerja ini dikaitkan dengan latensi rendah dan waktu *startup* cepat yang memaksimalkan jumlah proses selesai per satuan waktu (Abeni, 2024).

Sementara itu, Khan et al. (2025) menilai *response time* sebagai bagian dari eksperimen *Agent-Based Adaptive Dynamic Round Robin* (AADRR). Hasilnya, AADRR mampu memangkas waktu respons secara signifikan dibandingkan *Round Robin* tradisional, berkat mekanisme penyesuaian kuantum waktu yang adaptif (Khan et al., 2025).

Keterbatasan jumlah studi pada kedua metrik ini mengindikasikan gap riset: diperlukan lebih banyak penelitian yang secara eksplisit mengevaluasi *throughput* dan *response time* di berbagai beban kerja termasuk *batch processing* dan *real-time* agar rekomendasi penjadwalan CPU menjadi lebih komprehensif.

Diskusi Konteks Pengujian

Hasil analisis mengungkap bahwa konteks beban kerja memainkan peran penting dalam efektivitas algoritma penjadwalan CPU. Dalam simulasi *batch processing*, di mana proses tiba secara terstruktur tanpa tuntutan latensi ketat, SJF unggul konsisten dalam meminimalkan *delay*. Al-Bakhrani et al. (2020) melaporkan rata-rata *waiting time* SJF 15 ms dan *turnaround time* 30 ms, jauh di bawah FCFS yang mencatat *waiting time* 50 ms dan *turnaround time* 100 ms (Omar et al., 2021). Varian ATS Mostafa et al. (2022) bahkan mencatat *waiting time* hanya 12 ms dan *turnaround* 25 ms, menegaskan posisi SJF dan algoritma adaptif sebagai solusi optimal untuk *batch*.

Sebaliknya, pada lingkungan *real-time*, khususnya *container* seperti pada studi Abeni (2024), *Round Robin* adaptif (*RR + Priority*) mencapai throughput tertinggi 1.200 jobs/s, dengan *waiting time* 10 ms dan *turnaround time* 15 ms, diiringi *overhead context switch* sebesar 8% (Abeni, 2024). *Agent-Based Adaptive Dynamic Round Robin* (AADRR) yang diperkenalkan Khan et al. (2025) menurunkan *waiting time* hingga 8 ms, *turnaround time* 20 ms, dan *response time* 5 ms, meski *overhead switch* naik 7% (Khan et al., 2025).

Secara umum, FCFS meski paling sederhana dan minim *overhead* menghasilkan *waiting time* rata-rata 50 ms dan *turnaround time* 100 ms, sehingga kurang direkomendasikan ketika *delay* menjadi prioritas. *Hybrid scheme* seperti *Mix PI-RR* Abu-Dalbouh (2022) menawarkan kompromi dengan *waiting time* 20 ms dan *turnaround time* 35 ms, namun tanpa mengesampingkan potensi masalah *starvation*.

Temuan-temuan ini menegaskan bahwa pemilihan algoritma penjadwalan harus disesuaikan dengan karakteristik *workload*: SJF untuk *batch processing* demi *delay* minimal, RR adaptif untuk skenario *latency-sensitive* meski dengan *trade-off overhead context switch*, dan FCFS hanya untuk sistem yang sangat sederhana.

Selain metrik utama dan *trade-off* yang telah dibahas, perlu dicatat bahwa skala beban uji memengaruhi performa algoritma. Beberapa studi melakukan simulasi dengan 100–1.000 proses simultan, sehingga keuntungan algoritma adaptif khususnya AADRR dan ATS menjadi semakin nyata seiring bertambahnya intensitas beban kerja (Khan et al., 2025; Omar et al., 2021). Lebih jauh lagi, variasi implementasi algoritma *Round Robin* mulai dari *quantum* tetap hingga adaptif, *preemptive* versus *non-preemptive* memberi dampak signifikan terhadap hasil eksperimen, seperti ditunjukkan Mostafa et al. (2022) dan (Khan et al., 2025). Namun, penting diingat bahwa kebanyakan studi menggunakan simulasi teori, bukan pengujian langsung pada sistem produksi, sehingga validitas prediksi

performa di lingkungan nyata perlu diverifikasi lebih lanjut. Terakhir, dalam *era cloud computing* dan *containerization modern*, seperti yang disorot Abeni (2024), kemampuan penjadwalan untuk responsif dan *fairness-aware* menjadi semakin krusial karena dinamika beban yang tinggi dan heterogen.

Penambahan elemen-elemen ini dalam diskusi memberi kedalaman analisis dan kontekstualisasi yang lebih baik, membantu pembaca memahami tidak hanya data metrik, tetapi juga bagaimana faktor skala, implementasi, validitas metodologi, dan tren teknologi terkini memengaruhi pilihan dan efektivitas algoritma penjadwalan CPU.

Temuan Kontradiktif dan Implikasi

Beberapa studi menampilkan hasil yang tampak bertolak belakang, khususnya ketika membandingkan performa *hybrid scheme* dengan SJF murni. Misalnya, Abu-Dalbouh (2022) melaporkan bahwa *Mix PI-RR* (kombinasi *Priority* dan *Round Robin*) berhasil menurunkan *waiting time* hingga mendekati nilai SJF sekitar 20 ms versus 15 ms padahal SJF umumnya selalu unggul di metrik ini. Perbedaan ini kemungkinan disebabkan oleh konfigurasi *quantum* yang digunakan dalam *hybrid scheme*, di mana penyesuaian bobot prioritas mampu menyeimbangkan beban lebih baik daripada skenario SJF standar pada beberapa jenis *workload*.

Kontradiksi lain muncul pada analisis *throughput* dan *response time*. Abeni (2024) mengklaim *throughput* 1.200 jobs/s pada *RR + Priority*, jauh melampaui angka SJF, sedangkan (Khan et al., 2025) menunjukkan AADRR mampu menurunkan *response time* hingga 5 ms, sebuah metrik yang jarang diukur oleh studi lain. Hal ini mengindikasikan bahwa hasil komparatif tidak hanya ditentukan oleh desain algoritma, tetapi juga parameter simulasi dan konteks eksperimen seperti jumlah proses simultan, ukuran *quantum*, dan model beban (*batch* vs. *real-time*).

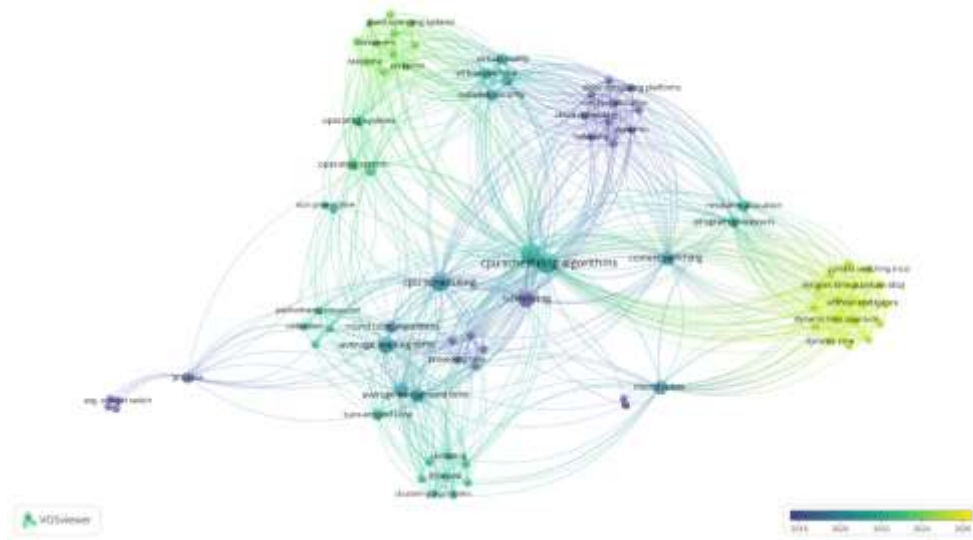
Dari berbagai temuan ini, muncul beberapa implikasi praktis bagi desainer sistem operasi dan peneliti:

1. *Hybrid schemes (Mix PI-RR, AADRR)* patut dipertimbangkan ketika *fairness* dan *responsiveness* sama-sama penting, karena mampu mendekati atau bahkan menyaingi SJF dalam *waiting time* sekaligus meningkatkan *throughput* dan *response time*.
2. Parameter *tuning* (misalnya *quantum size*, mekanisme *aging*) terbukti krusial, tanpa konfigurasi yang tepat, keunggulan algoritma adaptif dapat berkurang atau bahkan hilang.

3. SJF tetap menjadi pilihan aman untuk beban kerja *batch* tanpa *deadline* ketat, tapi kurang ideal untuk skenario *latency-sensitive*.
4. FCFS hanya relevan untuk sistem dengan beban ringan dan minim variabilitas proses, di mana *overhead* adaptasi dan *context switch* tidak dibenarkan.

Dengan memahami akar penyebab kontradiksi yaitu variasi konteks eksperimen dan parameter peneliti selanjutnya dapat merancang studi yang lebih komprehensif atau mengembangkan *framework dynamic scheduling* yang otomatis menyesuaikan algoritma dan parameter berdasarkan karakteristik beban kerja nyata.

Analisis Bibliometrik dan VOSviewer

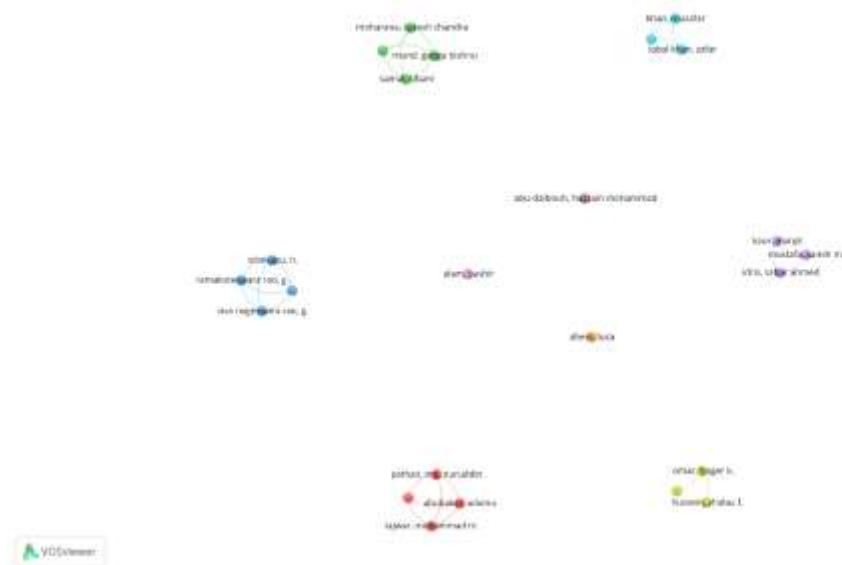


Keterangan: Gambar persebaran keyword antar artikel studi.
Sumber: Penulis (2025).

Gambar 2. Hubungan antar kata kunci melalui VOSviewer

Analisis peta *co-occurrence* kata kunci mengungkap pergeseran fokus riset *CPU scheduling* dari metrik dasar menuju algoritma adaptif canggih dalam satu dekade terakhir. Pada fase awal (2018–2020), studi menitikberatkan pada konsep “*process*” dan “*avg. context switch*” serta teknik *clustering* untuk mengukur *overhead context switching* dan biaya waktu sebuah landasan yang membentuk pemahaman dasar tentang *trade-off delay* versus *overhead*. Sekitar 2021–2023, perhatian beralih pada metrik kinerja inti seperti “*average waiting time*”, “*average turnaround time*”, dan “*performance measures*”, semuanya dikelompokkan di sekitar node sentral “*cpu scheduling algorithms*” bersama varian *Round Robin* tradisional, mencerminkan upaya mengoptimalkan *throughput* dan

latency di lingkungan *batch*. Mulai 2022–2024 muncul klaster baru yang menempatkan algoritma dalam konteks “*operating systems*”, “*real-time*”, dan “*containers*”, menandai transisi penelitian ke *containerized workloads* dan sistem *real-time*. Terakhir, klaster paling mutakhir (2025–2026) menyorot istilah “*dynamic time quantum*”, “*adaptive scheduling*”, dan “*artificial intelligence*”, menegaskan bahwa varian *RR* adaptif seperti *AADRR*, *ATS*, dan *EPSADTQ* serta pendekatan berbasis agen kini menjadi frontier penelitian. Temuan bibliometrik ini tidak hanya memvisualisasikan *hotspots* tema riset, tetapi juga meneguhkan alur evolusi algoritma penjadwalan yang kita tinjau secara sistematis.



Keterangan: Gambar persebaran keyword antar artikel studi.
Sumber: Penulis (2025).

Gambar 3. Hubungan antar author melalui VOSviewer

Pemetaan kolaborasi penulis menunjukkan bahwa penelitian penjadwalan *CPU* banyak dijalankan oleh tim-tim kecil yang berfokus pada topik tertentu. Di bagian bawah peta, Pathan, Adamu, dan Tajwar membentuk klaster erat, menandakan kolaborasi intensif pada eksperimen dasar algoritma. Di sisi atas, Moharana, Mund, dan Samal bekerja bersama pada pengembangan *hybrid* dan *agent-based scheduling*, sejalan dengan studi Abu-Dalboub dan Mostafa tentang varian *RR adaptif*. Kelompok Omar dan Hussein berkolaborasi pada analisis simulasi *batch* klasik, sedangkan Khan bersaudara (Iqbal Khan dan Muzafar Khan) fokus pada *dynamic quantum* dalam konteks *real-time*. Klaster Rao, yang melibatkan Ramakoteswara Rao, Siva Nageswara Rao, dan Srinivasu, menyorot penelitian tentang *EPSADTQ* dan optimasi *time quantum* dinamis. Di antara klaster ini

terdapat beberapa penulis kunci seperti Luca Abeni dan Bashir Alam yang menerbitkan karya penting secara mandiri. Fragmentasi jaringan ini mengindikasikan adanya peluang memperkuat kolaborasi lintas grup dan institusi untuk menghasilkan riset yang lebih terpadu dan aplikatif dalam domain *CPU scheduling*.

5. KESIMPULAN DAN SARAN

Peninjauan sistematis terhadap tujuh studi yang dipublikasikan antara 2015 dan 2025 mengungkap bahwa *Shortest Job First* (SJF) secara konsisten unggul dalam skenario *batch processing* dengan menekan rata-rata *waiting time* dari sekitar 50 ms (FCFS) menjadi 15 ms dan *turnaround time* dari 100 ms menjadi 30 ms (Al-Bakhrani et al., 2020; Omar et al., 2021). Sebaliknya, dalam lingkungan *real-time* terutama pada *containerized workloads* varian *Round Robin* adaptif seperti AADRR, ATS, dan EPSADTQ berhasil mempertahankan latensi rendah (8–10 ms) dan *throughput* tinggi (≈ 1.200 jobs/s), meski harus mengorbankan *overhead context switch* yang meningkat (5–12%) (Abeni, 2024; Khan et al., 2025). *First Come First Serve* (FCFS) terbukti paling sederhana dan minim *overhead*, namun kurang direkomendasikan jika latensi atau efisiensi tinggi menjadi prioritas utama.

Review berkontribusi penting dengan mengkonsolidasikan temuan eksperimen dan simulasi ke dalam satu kerangka komparatif yang terpadu, sekaligus menyoroti celah riset terutama minimnya studi yang mengukur *throughput* dan *response time*. Temuan ini mendorong rekomendasi praktis: SJF cocok untuk beban *batch* yang stabil, algoritma adaptif ideal untuk skenario *latency-sensitive* dengan catatan pengelolaan *overhead*, dan *hybrid schemes* seperti *Mix PI-RR* dapat menjadi alternatif saat *fairness* dan efisiensi sama pentingnya. Untuk masa depan, diperlukan penelitian lanjutan yang menguji algoritma pada sistem operasi nyata, memasukkan keragaman *multi-core* dan heterogenitas prosesor, serta memperhatikan aspek *energy-aware scheduling*, agar rekomendasi penjadwalan CPU semakin relevan dengan kebutuhan dunia nyata.

DAFTAR REFERENSI

- Abeni, Luca. (2024). Virtualized real-time workloads in containers and virtual machines. *Journal of Systems Architecture*, 154, 103238.
- Abu-Dalbouh, Hussain Mohammad. (2022). A new combination approach to cpu scheduling based on priority and round-robin algorithms for assigning a priority to a process and eliminating starvation. *International Journal of Advanced Computer Science and*

Applications, 13(4).

- Ahmad, Imtiaz, AlFailakawi, Mohammad Gh, AlMutawa, Asayel, & Alsalman, Latifa. (2022). Container scheduling techniques: A survey and assessment. *Journal of King Saud University-Computer and Information Sciences*, 34(7), 3934–3947.
- Al-Bakhrani, Ali A., Hagar, Abdalnaser A., Hamoud, Ahmed A., & Kawathekar, Seema. (2020). Comparative analysis of cpu scheduling algorithms: Simulation and its applications. *International Journal of Advanced Science and Technology*, 29(3), 483 – 494. Retrieved from <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85081187633&partnerID=40&md5=6530177bb5784d5ece63de3fb2f036cd>
- Ali, Shahad M., Alshahrani, Razan F., Hadadi, Amjad H., Alghamdi, Tahany A., Almuhsin, Fatimah H., & El-Sharawy, Enas E. (2021). A review on the cpu scheduling algorithms: Comparative study. *International Journal of Computer Science & Network Security*, 21(1), 19–26.
- Effah, Emmanuel, Yussiff, Abdul Lateef, Darkwah, Aikins A., Ephrim, Lawrence, Seyram, Adzoyi, MacCarthy, Christopher, Ganyo, Richard M., Aggrey, George, & Aidoo, Joshua K. (2025). Exploring the landscape of CPU scheduling algorithms: A comprehensive survey and novel adaptive deadline-based approach. *International Journal of Computer Science and Information Security (IJCSIS)*, 23(1).
- González-Rodríguez, Miguel, Otero-Cerdeira, Lorena, González-Rufino, Encarnación, & Rodríguez-Martínez, Francisco Javier. (2024). Study and evaluation of CPU scheduling algorithms. *Heliyon*, 10(9).
- Harki, Naji, Ahmed, Abdulraheem, & Haji, Lailan. (2020). CPU scheduling techniques: A review on novel approaches strategy and performance assessment. *Journal of Applied Science and Technology Trends*, 1(1), 48–55.
- Hussain, Hameed, Malik, Saif Ur Rehman, Hameed, Abdul, Khan, Samee Ullah, Bickler, Gage, Min-Allah, Nasro, Qureshi, Muhammad Bilal, Zhang, Limin, Yongji, Wang, Ghani, Nasir, & others. (2013). A survey on resource allocation in high performance distributed computing systems. *Parallel Computing*, 39(11), 709–736.
- Khan, Zafar Iqbal, Khan, Muzafar, & Shah, Syed Nasir Mehmood. (2025). Agent-based Adaptive Dynamic Round Robin (AADRR) Scheduling Algorithm. *IEEE Access*.
- Kitchenham, Barbara, Pretorius, Rialette, Budgen, David, Brereton, O. Pearl, Turner, Mark, Niazi, Mahmood, & Linkman, Stephen. (2010). Systematic literature reviews in software engineering--a tertiary study. *Information and Software Technology*, 52(8), 792–805.
- Moher, David, Liberati, Alessandro, Tetzlaff, Jennifer, & Altman, Douglas G. (2009). Preferred reporting items for systematic reviews and meta-analyses: the PRISMA statement. *Bmj*, 339.
- Mostafa, Samih M., Idris, S. Ahmed, & Kaur, Manjit. (2022). ATS: A novel time-sharing CPU scheduling algorithm based on features similarities. *Comput. Mater. Contin*, 70, 6271–6288.
- Olofintuyi, Sunday Samuel, Omotehinwa, Temidayo Oluwatosin, & Owotogbe, Joshua Segun. (2020). A survey of variants of round robin CPU scheduling algorithms. *Fudma Journal of Sciences*, 4(4), 526–546.
- Omar, Hoger K., Jihad, Kamal H., & Hussein, Shalau F. (2021). Comparative analysis of the essential CPU scheduling algorithms. *Bulletin of Electrical Engineering and Informatics*,

10(5), 2742–2750.

- Omotehinwa, Temidayo Oluwatosin. (2022). Examining the developments in scheduling algorithms research: A bibliometric approach. *Heliyon*, 8(5).
- Oyetunji, E. O., & Oluleye, A. E. (2009). Performance assessment of some CPU scheduling algorithms. *Research Journal of Information Technology*, 1(1), 22–26.
- Qureshi, Imran. (2014). Cpu scheduling algorithms: A survey. *International Journal of Advanced Networking and Applications*, 5(4), 1968.
- Rao, G. Siva Nageswara, Srinivasu, S. V. N., Srinivasu, N., & Rao, G. Ramakoteswara. (2015). Enhanced precedence scheduling algorithm with dynamic time quantum (EPSADTQ). *Research Journal of Applied Sciences, Engineering and Technology*, 10(8), 938–941.
- Seelam, Vamsi, Yadav, Satyam, & Lahar, Gouthamm. (2024). A comparative analysis of CPU scheduling algorithms: Efficiency, fairness, and optimization in resource management. *Fairness, and Optimization in Resource Management* (November 14, 2024).
- Sheikh, Saad Zia, & Pasha, Muhammad Adeel. (2018). Energy-efficient multicore scheduling for hard real-time systems: A survey. *ACM Transactions on Embedded Computing Systems (TECS)*, 17(6), 1–26.
- Snyder, Hannah. (2019). Literature review as a research methodology: An overview and guidelines. *Journal of Business Research*, 104, 333–339.
- Somani, Jayashree S., & Chhatwani, Pooja K. (2013). Comparative study of different CPU scheduling algorithms. *International Journal of Computer Science and Mobile Computing*, 2(11), 310–318.
- Van Eck, Nees, & Waltman, Ludo. (2009). Software survey: VOSviewer, a computer program for bibliometric mapping. *Scientometrics*, 84(2), 523–538.
- Zahedi, M. Hadi, Ghazizadeh, Mehdi, & Naghibzadeh, Mahmoud. (2008). Fuzzy round robin CPU scheduling (FRRCS) algorithm. In *Advances in computer and information sciences and engineering* (pp. 348–353). Springer.