Jurnal Penelitian Inovatif (JUPIN)
Vol. 5, No. 4, November 2025, Hal. 3197-3206
https://jurnal-id.com/index.php/jupin

DOI: https://doi.org/10.54082/jupin.1957
p-ISSN: 2808-148X
e-ISSN: 2808-1366

Analisis Komparatif dan Evaluatif terhadap Algoritma First-Come First-Served (FCFS) dalam Penjadwalan CPU di Era Komputasi Modern

Zulfahmi Indra[1], Arsandi Aulia Zidan*[2], Ahmad Affandi Silaen[3], Ahmad Naufal Habibi[4], Kalpin Palendeo Sitepu[5]

[1,2,3,4,5]Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Medan, Indonesia
Email: [1]zulfahmi.indra@unimed.ac.id, [2]aazidan051102@gmail.com, [3]ahmadafandi1330@gmail.com, [4]Ahmadnaufalhabibilbs@gmail.com, [5]Kalpinsitepu20@gmail.com

Abstrak

Tinjauan literatur ini bertujuan untuk mengevaluasi kembali kredibilitas dan relevansi Algoritma First-Come, First-Served (FCFS) dalam konteks lingkungan komputasi modern. Penelitian ini menggunakan pendekatan kajian literatur sistematis (*Systematic Literature Review*) dengan menganalisis literatur ilmiah yang diterbitkan dalam lima tahun terakhir. Analisis komparatif dilakukan dengan membandingkan performa FCFS terhadap algoritma penjadwalan CPU lain yang populer, yaitu Shortest Job First (SJF) dan Round Robin (RR), berdasarkan metrik efisiensi, keadilan, dan kompleksitas implementasi. Hasil kajian menunjukkan bahwa FCFS, meskipun fundamental dan unggul dalam kesederhanaan, memiliki keterbatasan serius di lingkungan multitasking modern akibat efek konvoi yang signifikan. Sementara itu, SJF menawarkan efisiensi waktu tunggu terbaik namun berisiko *starvation*, dan RR memberikan keadilan yang tinggi dengan mengorbankan *overhead context switching*. Temuan ini menegaskan bahwa tidak ada satu algoritma tunggal yang optimal. Setiap algoritma merepresentasikan *trade-off* unik. Kontribusi penelitian ini adalah menyoroti pentingnya pemahaman terhadap FCFS sebagai fondasi konseptual yang berkelanjutan, yang kini bertindak sebagai blok bangunan penting dalam pengembangan algoritma hibrida dan sistem penjadwalan adaptif di era komputasi modern.

Kata Kunci: *First-Come First-Served, Komputasi Modern, Penjadwalan CPU, Round Robin, Shortest Job First, Sistem Operasi.*

Abstract

This literature review aims to re-evaluate the credibility and relevance of the First-Come, First-Served (FCFS) Algorithm within the context of modern computing environments. The study employs a systematic literature review approach by analyzing scientific literature published within the last five years. A comparative analysis was conducted by contrasting FCFS performance against other popular CPU scheduling algorithms—namely Shortest Job First (SJF) and Round Robin (RR)—based on metrics of efficiency, fairness, and implementation complexity. The findings indicate that FCFS, while fundamental and superior in simplicity, has serious limitations in modern multitasking environments due to the significant convoy effect. In contrast, SJF offers the best waiting time efficiency but risks starvation, and RR provides high fairness at the expense of high context switching overhead. These findings affirm that no single scheduling algorithm is optimal. Each algorithm represents a unique trade-off. The contribution of this research is highlighting the sustained conceptual importance of FCFS, which now serves as a crucial building block in the development of hybrid algorithms and adaptive scheduling systems in the modern computing era.

Keywords: *CPU Scheduling, First-Come First-Served, Modern Computing, Operating Systems, Round Robin, Shortest Job First.*

1. PENDAHULUAN

Dalam sejarah ilmu komputer, prinsip First-In, First-Out (FIFO) dan algoritma First-Come, First-Served (FCFS) telah lama dianggap sebagai fondasi fundamental dalam penjadwalan dan manajemen antrean. Keduanya merupakan strategi penjadwalan paling sederhana dan paling awal yang digunakan dalam sistem operasi tradisional (Tani & Amrani, 2016). Prinsip dasar dari kedua algoritma ini terletak pada kesederhanaan operasionalnya, di mana tugas yang datang pertama kali akan diproses pertama kali,

DOI: https://doi.org/10.54082/jupin.1957
p-ISSN: 2808-148X
e-ISSN: 2808-1366

memastikan urutan penanganan yang logis dan adil (Omar dkk., 2021). Seiring dengan evolusi sistem komputasi dari model batch yang sederhana menjadi arsitektur yang sangat kompleks, seperti *multitasking , cloud computing* , dan sistem *real-time*, lingkungan komputasi kontemporer menuntut responsivitas tinggi, pemanfaatan sumber daya yang optimal, dan kemampuan adaptasi terhadap beban kerja yang dinamis (Kruekaew & Kimpan, 2022). Tuntutan ini menuntut pendekatan yang lebih canggih daripada sekadar melayani tugas sesuai urutan kedatangannya (Putra, 2022). Dengan demikian, relevansi historis FIFO/FCFS perlu dievaluasi kembali untuk menentukan perannya yang berkelanjutan dalam lanskap teknologi saat ini, khususnya dalam penjadwalan di lingkungan jaringan dan komputasi (Eltaib dkk., 2022).

Meskipun kesederhanaannya menarik, FCFS/FIFO memiliki kelemahan yang signifikan, yang paling menonjol adalah apa yang dikenal sebagai "*convoy effect*". Efek ini terjadi ketika sebuah tugas dengan waktu eksekusi yang sangat lama berada di awal antrean, memaksa semua tugas lain yang lebih pendek untuk menunggu secara substansial, yang secara signifikan mengurangi efisiensi sistem secara keseluruhan. Lebih lanjut, sifat non-preemptif dari FCFS, di mana sebuah proses yang telah memulai eksekusinya tidak dapat diinterupsi, dapat menyebabkan inefisiensi, terutama dalam kondisi beban kerja tinggi (Shakor, 2021; Thangakumar & Sambath, 2021). Kondisi ini dapat menurunkan kinerja keseluruhan sistem, terutama ketika respons cepat sangat dibutuhkan.

Untuk mengatasi keterbatasan ini, strategi penjadwalan dalam sistem operasi kontemporer telah berkembang mencakup algoritma yang lebih canggih dan dinamis. Algoritma seperti Round Robin (RR) dan Shortest Job Next (SJN) dirancang untuk meningkatkan *throughput* dan mengurangi waktu tunggu rata-rata (Singh dkk., 2021). Solusi-solusi ini bekerja dengan menyesuaikan prioritas berdasarkan karakteristik tugas atau dengan mengalokasikan unit waktu secara bergiliran, memungkinkan sistem untuk lebih responsif (Ju dkk., 2022; Mostafa & Amano, 2020). Transisi dari model FCFS/FIFO yang kaku menuju pendekatan yang lebih fleksibel ini menunjukkan evolusi signifikan dalam teori dan praktik penjadwalan.

Meskipun algoritma yang lebih kompleks kini mendominasi, FCFS dan FIFO tidak sepenuhnya kehilangan relevansinya; sebaliknya, mereka telah menemukan peran baru dalam konteks yang lebih canggih. Beberapa penelitian menunjukkan bahwa algoritma dasar ini tetap melayani peran fundamental, terutama di lingkungan di mana kesederhanaan dan keadilan diprioritaskan (Singh dkk., 2021). Dalam *cloud computing*, misalnya, model FIFO masih dianggap relevan untuk manajemen sumber daya dan tugas, di mana pekerjaan diproses dalam urutan kedatangannya untuk memastikan keadilan alokasi (Arora & Banyal, 2021). Selain itu, studi-studi terbaru telah menyoroti kebutuhan untuk menyempurnakan algoritma tradisional seperti FCFS/FIFO agar dapat beradaptasi dengan tantangan baru, seperti kompleksitas arsitektur *cloud* dan pemrosesan data *real-time* (Abdel-Basset dkk., 2022; Kruekaew & Kimpan, 2022).

Literatur yang ada memberikan gambaran yang jelas mengenai karakteristik dan kelemahan historis FIFO/FCFS. Namun, fokus utamanya seringkali terbagi antara mengkritik ketidakmampuan algoritma ini menghadapi beban kerja modern atau sekadar mendeskripsikannya sebagai konsep historis. Meskipun beberapa penelitian mengakui peran berkelanjutan FIFO/FCFS di lingkungan tertentu, hingga kini belum ada kajian sistematis yang secara komprehensif menilai kembali kredibilitas FCFS dalam konteks sistem operasi modern berbasis multi-core dan komputasi awan. Kekosongan literatur ini menjadi signifikan mengingat FCFS kini lebih sering berfungsi sebagai blok bangunan fundamental di dalam algoritma hibrida (seperti MLFQ) ketimbang sebagai *standalone scheduler* murni. Penilaian ulang ini tidak hanya sekadar mereplikasi temuan sebelumnya, tetapi juga menawarkan perspektif baru tentang nilai fundamental yang masih dipegang oleh FIFO/FCFS.

Oleh karena itu, penelitian ini berupaya mengisi kesenjangan tersebut melalui analisis teoritis dan komparatif. Tujuan utama penelitian ini adalah untuk menilai kembali kredibilitas algoritma FCFS dengan menganalisis perbandingannya terhadap algoritma lain, mengidentifikasi keunggulan dan keterbatasannya, serta menilai relevansinya dalam arsitektur penjadwalan hibrida saat ini. Kebaruan penelitian ini terletak pada pendekatannya yang menempatkan FCFS bukan hanya sebagai model klasik, tetapi sebagai blok bangunan konseptual yang masih esensial untuk algoritma hibrida masa kini (Banerjee & Hecker, 2016). Ruang lingkup penelitian ini terbatas pada analisis teoritis dan komparatif,

DOI: https://doi.org/10.54082/jupin.1957
p-ISSN: 2808-148X
e-ISSN: 2808-1366

tidak mencakup eksperimen empiris atau analisis performa mendalam. Kami akan menggunakan contoh-contoh konseptual untuk mengilustrasikan bagaimana FCFS/FIFO berinteraksi dengan algoritma lain seperti Round Robin dan penjadwalan berbasis prioritas.

2. METODE PENELITIAN

2.1. Desain Penelitian

Penelitian ini menggunakan Kajian Literatur Sistematis (*Systematic Literature Review*—SLR) sebagai desain penelitian utama. Pendekatan SLR dipilih karena memungkinkan identifikasi, evaluasi, dan sintesis temuan dari semua literatur yang relevan secara sistematis dan transparan (Kitchenham & Charters, 2007). Metode analisis data yang digunakan adalah *Narrative Synthesis* dan *Content Analysis* kualitatif untuk mengidentifikasi pola, *trade-off* utama, dan peran FCFS sebagai fondasi algoritma hibrida.

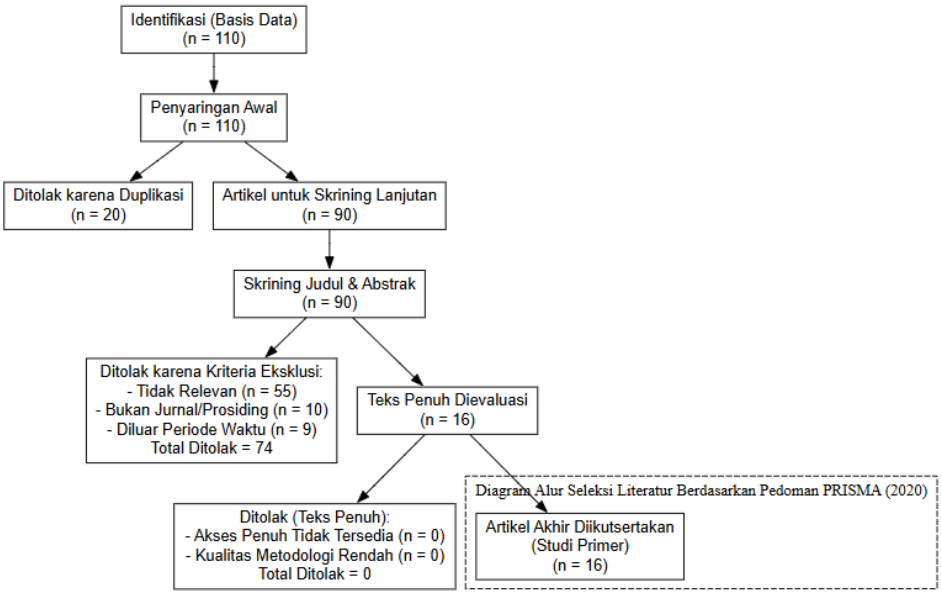
2.2. Prosedur Pengumpulan dan Seleksi Data

Prosedur pengumpulan dan seleksi data mengikuti protokol Kajian Literatur Sistematis (SLR) yang ketat. Pencarian literatur dilakukan secara sistematis pada basis data utama, meliputi IEEE Xplore, ACM Digital Library, Scopus, dan Google Scholar. Untuk menjamin relevansi temuan dengan tren komputasi terkini, periode pencarian literatur ditetapkan dari Januari 2021 hingga Mei 2025.

Proses pencarian menggunakan kombinasi Boolean dengan dua set kata kunci utama: Pertama, kombinasi ("FCFS" ATAU "FIFO") AND ("CPU Scheduling" ATAU "Task Scheduling") AND ("Modern Computing" OR "Cloud Computing" OR "Multicore"). Kedua, kombinasi ("FCFS" AND "Round Robin") AND ("SJF" AND "Trade-off").

Seleksi artikel dilakukan dalam dua tahap oleh dua penelaah (penulis) untuk meningkatkan keandalan. Kriteria Inklusi diterapkan untuk menerima artikel jurnal dan prosiding konferensi yang telah melalui *peer-review*, membahas FCFS/FIFO, relevan dengan algoritma hibrida, diterbitkan dalam periode 2021-2025, dan memiliki akses penuh. Sebaliknya, Kriteria Eksklusi digunakan untuk menolak tesis, disertasi, artikel non-ilmiah, artikel dengan fokus penjadwalan non-CPU, serta artikel yang diterbitkan di luar periode atau memiliki kualitas metodologi yang diragukan.

Dari proses skrining awal yang menemukan 110 artikel, dan setelah diterapkan kriteria inklusi dan eksklusi, sebanyak 16 artikel akhir (studi primer) dipilih untuk dianalisis secara mendalam. Alur proses seleksi literatur ini diilustrasikan secara visual dalam bentuk diagram alur PRISMA (Gambar 1).



Gambar 1. Diagram Alur Seleksi Literatur

DOI: https://doi.org/10.54082/jupin.1957
p-ISSN: 2808-148X
e-ISSN: 2808-1366

2.3. Teknik Analisis Data dan Keandalan

Data dari artikel terpilih diekstraksi dan diringkas berdasarkan Penulis, Tahun Publikasi, Tujuan Studi, Temuan Utama, dan Keterbatasan (disajikan secara detail dalam Tabel 1. Ringkasan Karakteristik Studi yang Direview). Analisis dilakukan melalui dua tahapan: (1) *Content Analysis* (Analisis Konten): Mengidentifikasi dan mengkategorikan konsep-konsep inti, seperti definisi *convoy effect*, metrik performa SJF/RR, dan arsitektur algoritma hibrida yang menggunakan FCFS. (2) *Narrative Synthesis* (Sintesis Naratif): Melakukan sintesis temuan dari berbagai studi menjadi sebuah narasi yang koheren untuk menjawab pertanyaan penelitian.

Tabel 1. Ringkasan Karakteristik Studi yang Direview

Penulis (Tahun)	**Tujuan Studi (Sintesis)**	**Temuan Utama (Relevansi dengan FCFS)**	**Keterbatasan (Inferred)**
Shakor (2021)	Kajian algoritma penjadwalan dan sinkronisasi OS.	Sifat non-preemptif FCFS dapat menyebabkan inefisiensi; FCFS jarang digunakan sebagai solusi tunggal di sistem modern.	Lebih bersifat survei umum tanpa eksperimen performa spesifik.
Ghaffar et al. (2021)	Mengevaluasi kinerja FCFS dan mengusulkan algoritma baru (MSA).	Mengidentifikasi *convoy effect* sebagai kelemahan kritis FCFS yang secara dramatis memperbesar waktu tunggu rata-rata.	Fokus kuat pada kelemahan FCFS tanpa mengeksplorasi peran FCFS sebagai komponen hibrida.
Omar et al. (2021)	Analisis komparatif algoritma penjadwalan CPU esensial.	Tidak ada satu algoritma yang sempurna; setiap algoritma merepresentasikan *trade-off* unik antara efisiensi, keadilan, dan kompleksitas.	Analisis lebih bersifat umum (*comparative survey*) tanpa studi kasus implementasi yang mendalam.
Singh et al. (2021)	Kajian *Metaheuristics* untuk penjadwalan tugas *Heterogeneous* di *Cloud*.	Prinsip FCFS tetap relevan di lingkungan yang memprioritaskan keadilan (*fairness*) dan mencegah *starvation*.	Fokus utama adalah evaluasi *metaheuristics*, bukan tinjauan mendalam FCFS.
Abdel-Basset et al. (2022)	Pendekatan penjadwalan tugas di *Cloud* menggunakan *Hybrid Differential Evolution*.	FCFS memiliki kelemahan yang menuntut pengembangannya menjadi model hibrida adaptif untuk tantangan *cloud* dan *real-time*.	Kurangnya analisis mendalam tentang FCFS murni; hanya digunakan sebagai pembenaran untuk algoritma hibrida.
Dirdal et al. (2024)	Simulasi algoritma penjadwalan multiprosesor (FCFS/SJF).	SJF secara teoretis mampu meminimalkan waktu tunggu; FCFS bermanfaat dalam sistem *batch* dengan tugas berdurasi serupa.	Penekanan pada SJF, analisis FCFS cenderung sebagai pembanding klasik.
Salamun et al. (2023)	Survei model *Weakly Hard Real-Time* untuk *Control Systems*.	FCFS sering digunakan di antrean prioritas terendah MLFQ untuk menjamin penyelesaian tugas (*guaranteed completion*).	Kontekstualisasi FCFS terbatas pada sistem kendali dan MLFQ.
Arora & Banyal (2021)	Menganalisis penjadwalan *workflow* di *Cloud Computing*.	Model FIFO masih relevan untuk manajemen sumber daya dan tugas di *cloud* untuk memastikan keadilan alokasi.	Fokus pada algoritma hibrida berbasis optimasi, bukan analisis mendalam FCFS murni.
Mahmood et al. (2021)	Penjadwalan *Real-Time* Dinamis pada *Multicore Processors*.	FCFS masih efektif dan umum digunakan untuk penjadwalan I/O karena kesederhanaan dan keadilannya.	Fokus utamanya pada penjadwalan *multicore* / *power asymmetric*.
Farid et al. (2020)	Survei persyaratan QoS pada teknik penjadwalan *workflow*.	Sistem harus menyeimbangkan kepuasan pengguna (*fairness*) dengan metrik kinerja (*throughput*).	Fokus pada *workflow* di *cloud*, bukan penjadwalan CPU internal OS.

DOI: https://doi.org/10.54082/jupin.1957
p-ISSN: 2808-148X
e-ISSN: 2808-1366

Stan et al. (2021)	Evaluasi algoritma penjadwalan di lingkungan *Heterogeneous Computing*.	FCFS tidak optimal di lingkungan dengan tugas beragam, tidak memperhitungkan prioritas, menyebabkan tugas kritis tertunda.	Analisis terbatas pada lingkungan *heterogeneous*, tidak mencakup arsitektur *real-time* secara luas.
Jamil et al. (2024)	Mengembangkan algoritma Round Robin yang ditingkatkan.	Kinerja RR sangat bergantung pada ukuran *time quantum*; *quantum* yang terlalu besar dapat membuat RR berperilaku mirip FCFS.	Fokus spesifik pada optimasi RR; peran FCFS hanya sebagai titik referensi degenarasi.

Validitas dan Keandalan Data: Untuk memastikan keandalan temuan, kami menerapkan triangulasi sumber, yaitu membandingkan hasil temuan dari berbagai basis data dan tipe artikel (jurnal vs. konferensi). Penggunaan kriteria inklusi dan eksklusi yang ketat, serta konsensus bersama oleh dua penelaah pada tahap skrining, meningkatkan objektivitas dan keandalan seleksi literatur.

3. HASIL DAN PEMBAHASAN

3.1. Hasil Analisis: Implementasi, Keunggulan, dan Kelemahan FCFS

FCFS adalah algoritma penjadwalan non-preemptif yang paling sederhana. Implementasinya hanya memerlukan struktur data antrean (*Queue*), menjadikannya sangat efisien dalam hal *overhead* implementasi dan sumber daya komputasi. Di lingkungan komputasi modern, prinsip FCFS tetap digunakan secara fundamental dalam manajemen *resource queue* tingkat rendah, seperti *queue* I/O (Input/Output) atau *buffer* antrean pesan, di mana keadilan akses adalah prioritas utama (Arora & Banyal, 2021).

Oleh karena minimnya kebutuhan informasi *a priori* (sebelumnya) tentang waktu eksekusi tugas, FCFS menawarkan solusi yang paling andal dalam hal kesederhanaan. Karakteristik ini menjadikannya pilihan ideal untuk sistem *embedded* atau server *queue* dengan beban konstan di mana *overhead* yang disebabkan oleh algoritma penjadwalan harus diminimalisir. Nilai intinya terletak pada jaminan keadilan akses dasar; proses pertama yang meminta sumber daya adalah proses pertama yang akan dilayani. Inilah alasan mengapa FCFS tetap menjadi model konseptual penting dalam pengajaran sistem operasi, terlepas dari kelemahan kinerjanya.

Kelemahan paling signifikan FCFS adalah *convoy effect* (efek konvoi). Analisis literatur menegaskan bahwa efek ini terjadi ketika proses dengan waktu eksekusi yang sangat panjang (*CPU burst time*) tiba di awal antrean. Proses-proses yang lebih pendek harus menunggu, yang secara dramatis meningkatkan Waktu Tunggu Rata-Rata (WTR) sistem secara keseluruhan (Ghaffar dkk., 2021; Shakor, 2021). Dampak *convoy effect* ini menjadi semakin parah dalam lingkungan *multitasking* dengan tugas-tugas yang memiliki variasi waktu eksekusi yang tinggi.

Secara spesifik, *convoy effect* tidak hanya menurunkan efisiensi CPU, tetapi juga mengurangi responsivitas sistem secara keseluruhan, yang krusial bagi pengalaman pengguna di lingkungan *time-sharing*. Kerugian utama ini yang memberikan pembenaran metodologis bagi pengembangan algoritma penjadwalan yang lebih canggih dan adaptif. Algoritma tersebut, seperti Shortest Job First (SJF) dan Round Robin (RR), dirancang untuk secara fundamental mengatasi kelemahan FCFS ini. Oleh karena itu, *convoy effect* adalah alasan utama mengapa FCFS hampir tidak pernah digunakan sebagai penjadwal tunggal pada sistem operasi umum.

3.2. Perbandingan Komparatif Algoritma Penjadwalan Modern

Untuk menilai kredibilitas FCFS, tinjauan ini membandingkannya dengan dua algoritma penjadwalan CPU paling populer: Shortest Job First (SJF) dan Round Robin (RR). Perbandingan ini bertujuan mengidentifikasi *trade-off* yang mendasari desain penjadwalan CPU.

DOI: https://doi.org/10.54082/jupin.1957
p-ISSN: 2808-148X
e-ISSN: 2808-1366

3.2.1. FCFS vs. Shortest Job First (SJF)

SJF terbagi menjadi SJF non-preemptif dan Shortest Remaining Time First (SRTF, preemptif SJF). Secara teoretis, SJF non-preemptif menawarkan WTR minimal di antara semua algoritma. Meskipun demikian, analisis kritis menemukan bahwa SJF memiliki dua kelemahan fatal dalam praktik: Pertama, SJF sulit diimplementasikan secara murni di sistem *real-time* karena ia membutuhkan pengetahuan *a priori* (sebelumnya) tentang waktu eksekusi tugas, yang seringkali tidak tersedia. Implementasi praktis selalu mengandalkan prediksi heuristik. Kedua, SJF berisiko tinggi menyebabkan *starvation* bagi tugas-tugas panjang yang terus-menerus didahului oleh tugas-tugas pendek yang baru tiba (Dirdal dkk., 2024).

Perbandingan ini secara tegas menunjukkan *trade-off* mendasar dalam penjadwalan CPU: SJF memaksimalkan efisiensi waktu dengan mengorbankan keadilan dan kepastian penyelesaian tugas. FCFS, sebaliknya, menjamin keadilan deterministik, meskipun mengorbankan efisiensi waktu tunggu rata-rata. Dengan demikian, SJF hanya optimal digunakan dalam lingkungan *batch processing* di mana waktu eksekusi tugas diketahui sebelumnya dan risiko *starvation* dapat diatasi melalui mekanisme *aging* atau prioritas dinamis. Kesenjangan fundamental ini adalah alasan utama mengapa SJF, meskipun optimal secara matematis, tidak pernah menjadi penjadwal CPU universal.

3.2.2. FCFS vs. Round Robin (RR)

RR dirancang untuk mengatasi inefisiensi dan *starvation* dengan memperkenalkan *time quantum* dan *preemption*. RR sangat unggul dalam hal keadilan (*fairness*) dan responsivitas, menjadikannya pilihan utama untuk sistem *time-sharing*. Kelemahan utama RR terletak pada *overhead context switching*. Setiap kali sebuah proses diinterupsi oleh *time quantum* (kuanta waktu) yang berakhir, sistem harus menyimpan *state* CPU dan memuat *state* proses berikutnya. Jika *time quantum* terlalu kecil, *overhead* ini akan menghabiskan sebagian besar waktu CPU, mengurangi efisiensi bersih. Sebaliknya, jika *time quantum* terlalu besar, RR akan mendegenerasi dan berperilaku mirip FCFS (Jamil dkk., 2024).

Analisis komparatif menunjukkan bahwa RR secara efektif menghilangkan *convoy effect* FCFS dan mencapai *fairness* melalui rotasi waktu yang terukur. Namun, tantangan utama RR adalah menemukan titik optimal untuk *time quantum* guna menyeimbangkan *overhead* dan efisiensi. Ironisnya, kegagalan dalam optimasi ini dapat mengembalikan RR ke mode perilaku FCFS, membuktikan bahwa prinsip FCFS adalah batas atas (*upper bound*) dari algoritma *time-sharing* tanpa *overhead context switching*. Oleh karena itu, RR adalah solusi superior untuk responsivitas, namun implementasi FCFS tetap menjadi tolok ukur kesederhanaan dan efisiensi *overhead* yang ekstrem.

Untuk memvisualisasikan secara ringkas *trade-off* yang teridentifikasi dalam sintesis literatur ini, Tabel 2 menyajikan analisis komparatif terperinci antara FCFS, SJF, dan RR berdasarkan kriteria metrik kunci dan implikasi desain dalam konteks komputasi modern.

Tabel 2. Analisis Komparatif Trade-off Algoritma Penjadwalan CPU di Era Komputasi Modern

Kriteria Metrik / Desain	First-Come, First-Served (FCFS)	Shortest Job First (SJF / SRTF)	Round Robin (RR)
Implikasi Efisiensi Waktu	Waktu Tunggu Rata-Rata (WTR) tidak optimal; sangat rentan terhadap *Convoy Effect* yang memperlambat sistem.	WTR dan *Turnaround Time* minimal (*optimal* secara teoretis); efisiensi throughput tinggi.	WTR moderat; responsivitas tinggi (*low response time*) untuk tugas interaktif/pendek.
Risiko Diskriminasi (*Fairness*)	Keadilan tertinggi (*deterministic*); setiap proses dijamin selesai sesuai waktu kedatangan.	Risiko *Starvation* akut bagi proses yang membutuhkan *burst time* panjang.	Keadilan tinggi; distribusi waktu CPU merata di antara semua proses aktif.
Overhead Komputasi	Overhead terendah; implementasi hanya memerlukan *queue* sederhana.	Overhead sedang hingga tinggi (*preemptive*); memerlukan prediksi *burst time* (heuristik).	Overhead *Context Switching* paling tinggi; berbanding terbalik dengan ukuran *quantum*.
Persyaratan *A Priori*	Tidak memerlukan informasi tugas sebelumnya.	Membutuhkan perkiraan waktu eksekusi yang	Membutuhkan penentuan *time quantum* yang optimal (tantangan utama desain).

DOI: https://doi.org/10.54082/jupin.1957
p-ISSN: 2808-148X
e-ISSN: 2808-1366

		akurat (sulit dalam praktik).	
Relevansi Modern (*Blok Bangunan*)	Ideal sebagai tingkat antrean terendah dalam algoritma hibrida (*MLFQ*) untuk mencegah *starvation*.	Relevan untuk penjadwalan *real-time* atau *batch* jika *burst time* dapat diprediksi.	Ideal untuk sistem *time-sharing* dan lingkungan yang menuntut pengalaman pengguna responsif.

3.3. Diskusi dan Implikasi Relevansi FCFS dalam Komputasi Modern

Temuan dari sintesis literatur ini secara konsisten menegaskan hipotesis awal: tidak ada algoritma penjadwalan tunggal yang mampu mengoptimalkan semua metrik performa secara bersamaan. Setiap algoritma adalah cerminan dari *trade-off* inheren antara efisiensi, keadilan, dan kompleksitas.

Kontribusi dan *novelty* utama dari tinjauan ini terletak pada penekanan peran FCFS bukan lagi sebagai *standalone scheduler* utama, melainkan sebagai *blok bangunan konseptual* dalam desain algoritma hibrida modern. Arsitektur canggih seperti Multi-Level Feedback Queue (MLFQ) secara eksplisit memanfaatkan FCFS di lapisan antrean prioritas terendah. Dalam MLFQ, FCFS di tingkat terakhir memastikan bahwa, meskipun sebuah proses memiliki prioritas rendah atau telah dieksekusi lama, ia pada akhirnya akan selesai (*guaranteed completion*), sekaligus mencegah *starvation* yang merupakan kelemahan SJF murni. Analisis ini memperluas pandangan terhadap FCFS dari sekadar model klasik menjadi komponen strategis dalam sistem penjadwalan adaptif.

Pergeseran peran ini menunjukkan nilai FCFS yang tak tergantikan dalam memastikan integritas sistem. Dalam desain arsitektur hibrida, FCFS berfungsi sebagai *safety net* atau kebijakan *fail-safe* untuk menjamin keadilan waktu yang ketat. Ketersediaannya sebagai modul *plug-in* dengan *overhead* terendah memungkinkannya diintegrasikan ke dalam berbagai lapisan penjadwalan. Oleh karena itu, penelitian ini mengubah interpretasi FCFS, dari hanya sebagai referensi sejarah, menjadi mekanisme aktif yang menjamin keandalan dan *fairness* fundamental pada sistem multi-core kontemporer.

Temuan ini memiliki implikasi praktis yang signifikan bagi perancang sistem operasi dan pengembang *cloud computing*. Implikasi utama adalah bahwa desainer harus berhenti mencari solusi tunggal yang sempurna. Sebaliknya, mereka harus fokus pada solusi hibrida yang mengintegrasikan FCFS di mana keadilan akses sumber daya (seperti pada *locking* atau *resource allocation* awal) lebih penting daripada efisiensi mutlak, dan menggabungkannya dengan SJF (untuk *real-time* atau tugas pendek) atau RR (untuk responsivitas *time-sharing*). Penggunaan FCFS di server *queue* dengan beban konstan atau di sistem *embedded* yang menuntut *deterministic scheduling* dengan *overhead* minimal tetap terbukti optimal.

Oleh karena itu, implikasi desainnya adalah perlunya adopsi model adaptif yang fleksibel dalam memilih algoritma penjadwalan pada lapisan yang berbeda. Keputusan desain harus didasarkan pada analisis beban kerja (*workload analysis*), yang menentukan apakah sistem membutuhkan latensi rendah (memilih RR/SJF) atau keadilan yang ketat (*strict fairness*) dengan *overhead* minimal (memilih FCFS). Dengan menyadari *trade-off* ini, desainer dapat merancang sistem operasi modern yang lebih efisien dan stabil, memanfaatkan kesederhanaan FCFS di tempat yang paling dibutuhkan.

4. KESIMPULAN

Kajian literatur sistematis ini menegaskan kembali posisi Algoritma First-Come, First-Served (FCFS) dalam konteks sistem operasi dan komputasi modern. Temuan utama menunjukkan bahwa FCFS, meskipun sangat sederhana dan efisien dari sisi *overhead* implementasi, memiliki keterbatasan signifikan sebagai solusi penjadwalan tunggal, terutama kerentanannya terhadap *convoy effect* yang secara substansial meningkatkan Waktu Tunggu Rata-Rata. Di sisi lain, algoritma modern seperti SJF dan RR menawarkan kinerja waktu yang lebih optimal (efisiensi) dan responsif (keadilan), namun dengan konsekuensi risiko *starvation* (SJF) atau *overhead context switching* yang tinggi (RR). Oleh karena itu, prinsip fundamentalnya tetap relevan: Pemilihan algoritma penjadwalan harus selalu disesuaikan dengan tujuan sistem, mempertimbangkan *trade-off* kritis antara efisiensi, keadilan, dan kompleksitas.

DOI: https://doi.org/10.54082/jupin.1957
p-ISSN: 2808-148X
e-ISSN: 2808-1366

Penelitian ini berkontribusi secara signifikan dalam memberikan pemahaman konseptual baru mengenai posisi FCFS sebagai komponen integral dari algoritma penjadwalan hibrida, bukan sekadar model klasik. Implikasi teoretis dan praktis menunjukkan bahwa desainer sistem operasi dan *cloud computing* tidak lagi harus memilih salah satu algoritma, melainkan mengintegrasikan FCFS secara strategis. Prinsip FCFS sangat penting dalam desain arsitektur Multi-Level Feedback Queue (MLFQ) sebagai mekanisme antrean tingkat terendah yang menjamin penyelesaian tugas (*guaranteed completion*) dan mencegah *starvation*. Dengan demikian, FCFS berperan sebagai fondasi yang menjamin *fairness policy* dasar dalam sistem *multi-core* yang kompleks.

Untuk melanjutkan kajian ini, disarankan agar penelitian di masa mendatang memperluas analisis melalui simulasi empiris. Penelitian lanjutan dapat berfokus pada evaluasi performa hibridisasi FCFS dalam arsitektur *multi-core* di bawah beban kerja dinamis. Selain itu, eksplorasi potensi integrasi FCFS dengan algoritma penjadwalan berbasis *Deep Optimization* atau *Reinforcement Learning* (RL) sangat disarankan, guna mengkaji bagaimana prinsip keadilan klasik dapat diterapkan secara cerdas dalam paradigma penjadwalan masa depan.

DAFTAR PUSTAKA

Abdel-Basset, M., Mohamed, R., Abd Elkhalik, W., Sharawi, M., & Sallam, K. M. (2022). Task Scheduling Approach in Cloud Computing Environment Using Hybrid Differential Evolution. *Mathematics*, *10*(21), 4049. https://doi.org/10.3390/math10214049

Arora, N. K., & Banyal, R. K. (2021). Workflow Scheduling Using Particle Swarm Optimization and Gray Wolf Optimization Algorithm in Cloud Computing. *Concurrency and Computation Practice and Experience*, *33*(16). https://doi.org/10.1002/cpe.6281

Banerjee, S., & Hecker, J. P. (2016). *A Multi-Agent System Approach to Load-Balancing and Resource Allocation for Distributed Computing*. 41–54. https://doi.org/10.1007/978-3-319-45901-1_4

Dirdal, D. O., Vo, D., Feng, Y., & Davidrajuh, R. (2024). Developing a Platform Using Petri Nets and GPenSIM for Simulation of Multiprocessor Scheduling Algorithms. *Applied Sciences*, *14*(13), 5690. https://doi.org/10.3390/app14135690

Eltaib, M. O., Alshammari, H., Boukrara, A., Gasmi, K., & Hrizi, O. (2022). Choosing the Best Quality of Service Algorithm Using OPNET Simulation. *International Journal of Electrical and Computer Engineering (Ijece)*, *12*(4), 4079. https://doi.org/10.11591/ijece.v12i4.pp4079-4089

Ghaffar, A., Akbari, M., & Yousufzai, M. (2021). Milad Scheduling Algorithm (MSA). *Kjet*. https://doi.org/10.31841/kjet.2022.22

Jamil, B., Yar, A., & Ijaz, H. (2024). Dynamic Time Quantum Computation for Improved Round Robin Scheduling Algorithm Using Quartiles and Randomization (IRRQR). *Sukkur Iba Journal of Computing and Mathematical Sciences*, *7*(2), 25–37. https://doi.org/10.30537/sjcms.v7i2.1340

Ju, L., Yin, Z., Zhou, Q., Liu, L., Pan, Y., & Tan, Z. (2022). Near-Zero Carbon Stochastic Dispatch Optimization Model for Power-to-Gas-Based Virtual Power Plant Considering Information Gap Status Theory. *International Journal of Climate Change Strategies and Management*, *15*(2), 105–127. https://doi.org/10.1108/ijccsm-02-2022-0018

Kitchenham, B., & Charters, S. (2007). *Guidelines for performing Systematic Literature Reviews in Software Engineering | BibSonomy* (2.3). https://www.bibsonomy.org/bibtex/aed0229656ada843d3e3f24e5e5c9eb9

Kruekaew, B., & Kimpan, W. (2022). Multi-Objective Task Scheduling Optimization for Load Balancing in Cloud Computing Environment Using Hybrid Artificial Bee Colony Algorithm With Reinforcement Learning. *Ieee Access*, *10*, 17803–17818. https://doi.org/10.1109/access.2022.3149955

Mostafa, S. M., & Amano, H. (2020). Dynamic Round Robin CPU Scheduling Algorithm Based on K-Means Clustering Technique. *Applied Sciences*, *10*(15), 5134. https://doi.org/10.3390/app10155134

DOI: https://doi.org/10.54082/jupin.1957
p-ISSN: 2808-148X
e-ISSN: 2808-1366

Omar, H. K., Jihad, K. H., & Hussein, S. F. (2021). Comparative Analysis of the Essential CPU Scheduling Algorithms. *Bulletin of Electrical Engineering and Informatics*, *10*(5), 2742–2750. https://doi.org/10.11591/eei.v10i5.2812

Putra, T. D. (2022). Analysis of Priority Preemptive Scheduling Algorithm: Case Study. *Ijarcce*, *11*(1). https://doi.org/10.17148/ijarcce.2022.11105

Shakor, M. Y. (2021). Scheduling and Synchronization Algorithms in Operating System: A Survey. *Journal of Studies in Science and Engineering*, *1*(2), 1–16. https://doi.org/10.53898/josse2021121

Singh, H., Tyagi, S., Kumar, P., Gill, S. S., & Buyya, R. (2021). Metaheuristics for Scheduling of Heterogeneous Tasks in Cloud Computing Environments: Analysis, Performance Evaluation, and Future Directions. *Simulation Modelling Practice and Theory*, *111*, 102353. https://doi.org/10.1016/j.simpat.2021.102353

Tani, H. G., & Amrani, C. E. (2016). Cloud Computing CPU Allocation and Scheduling Algorithms Using CloudSim Simulator. *International Journal of Electrical and Computer Engineering (Ijece)*, *6*(4), 1866. https://doi.org/10.11591/ijece.v6i4.pp1866-1879

Thangakumar, J., & Sambath, M. (2021). Performance Analysis of CPU Scheduling Algorithms – A Problem Solving Approach. *International Journal of Science and Management Studies (Ijsms)*, 411–416. https://doi.org/10.51386/25815946/ijsms-v4i4p138

Jurnal Penelitian Inovatif (JUPIN)
Vol. 5, No. 4, November 2025, Hal. 3197-3206
https://jurnal-id.com/index.php/jupin
DOI: https://doi.org/10.54082/jupin.1957
p-ISSN: 2808-148X
e-ISSN: 2808-1366

Halaman Ini Dikosongkan