Journal Review

No	Penulis & Tahun	Judul	Sumber (Jurnal/Penerbit)	Tujuan Penelitian	Metode / Dataset	Hasil Utama	Relevansi terhadap Penelitian Saya	Keterbatasan / Celah Penelitian
1	Din et al., 2024	*Locally Abstract, Globally Concrete Semantics of Concurrent Programming Languages*	ACM Transactions on Programming Languages and Systems	Mengembangkan semantik formal modular untuk bahasa pemrograman konkuren yang memungkinkan pemisahan evaluasi lokal dan komposisi global program.	Pendekatan teoritis berbasis *denotational semantics* dan *trace semantics* dengan uji penerapan pada model bahasa seperti C, Java, dan ProMeLa.	Menghasilkan kerangka *Locally Abstract, Globally Concrete (LAGC)* yang memungkinkan definisi semantik modular bagi model konkuren tanpa perlu mengubah keseluruhan framework.	Memberikan dasar teoretis yang kuat bagi analisis dan verifikasi model konkurensi yang dapat diterapkan untuk mengevaluasi performa bahasa modern seperti Go, Rust, Java, dan Python.	Tidak mengukur performa implementasi nyata atau benchmark; fokusnya konseptual dan formal, belum mengaitkan langsung dengan performa eksekusi bahasa nyata.
2	Abhinav et al., 2020	*Concurrency Analysis of Go and Java*	IEEE Conference Paper	Membandingkan fitur dan performa konkuren antara Go dan Java untuk menemukan bahasa paling efisien pada berbagai beban kerja.	Eksperimen dengan algoritma *Matrix Multiplication* dan *PageRank*; analisis waktu kompilasi, waktu eksekusi, ukuran biner, dan efisiensi fitur konkuren.	Go unggul untuk jumlah komputasi kecil, tetapi Java lebih efisien saat skala meningkat. Kedua bahasa memiliki kekuatan khas pada manajemen thread dan pustaka konkuren.	Relevan langsung dengan penelitian performa konkurensi lintas bahasa (Go, Java, Python, Rust). Memberi dasar empiris untuk desain pengujian dan metrik evaluasi performa.	Lingkup terbatas pada dua algoritma dan tidak mencakup pengujian paralelisme skala besar; juga tidak membahas konsumsi memori atau efisiensi CPU secara mendalam.
3	Al-hayanni et al., 2020	*Amdahl's Law in the Context of Heterogeneous Many-Core Systems – A Survey*	IET Computers & Digital Techniques	Mengulas pengembangan hukum Amdahl dan model turunannya untuk memahami batas percepatan komputasi pada sistem multi/heterogeneous core.	Studi pustaka sistematis terhadap model kecepatan eksekusi (*speedup models*) pada sistem multi-core dan heterogeneous.	Menemukan perluasan model Amdahl yang mempertimbangkan heterogenitas sistem, non-linearitas, dan faktor non-prosesor seperti memori dan komunikasi.	Memberi dasar teoretis untuk memahami keterbatasan paralelisme dan korelasi antara efisiensi paralel serta performa bahasa pemrograman multithread.	Tidak menyertakan eksperimen empiris; bersifat tinjauan konseptual, belum memberikan implementasi atau validasi terhadap bahasa tertentu.
4	Arora, Westrick, & Acar, 2021	*Provably Space-Efficient Parallel Functional Programming*	Proceedings of the ACM on Programming Languages (POPL)	Mengembangkan teknik manajemen memori otomatis yang efisien untuk program fungsional paralel bebas kondisi balapan (*determinacy-race-free*).	Pendekatan teoretis + eksperimen implementasi pada MPL Compiler (Parallel ML).	Menunjukkan bahwa program fungsional paralel dapat mencapai efisiensi ruang dan kerja dengan model memori terdistribusi dan penjadwalan adaptif.	Relevan untuk pemahaman efisiensi paralelisme di bahasa fungsional (Rust/Go) serta implikasi pada efisiensi memori dan skalabilitas sistem paralel.	Fokus pada bahasa fungsional; tidak menilai bahasa imperative modern seperti Java atau Python; eksperimen terbatas pada MPL.
5	Arora, Westrick, & Acar, 2023	*Efficient Parallel Functional Programming with Effects*	Proceedings of the ACM on Programming Languages (PLDI)	Mengembangkan model paralel fungsional yang efisien dengan mendukung efek mutasi memori tanpa mengorbankan keamanan data.	Pengembangan model semantik berbasis "entanglement" dan eksperimen pada MPL compiler.	Membuktikan bahwa model *entanglement-aware* memungkinkan efisiensi tinggi tanpa sinkronisasi thread atau promosi objek.	Menjelaskan konsep efisiensi dan isolasi memori yang dapat menginspirasi pengujian paralelisme pada bahasa imperatif modern seperti Rust dan Go.	Fokus pada konteks fungsional; tidak menilai model imperative concurrency; hasil implementasi terbatas pada MPL.
6	Ahmed et al., 2021	*A Parallelization Model for Performance Characterization of Spark Big Data Jobs on Hadoop Clusters*	Journal of Big Data	Mengusulkan model paralelisasi baru untuk memprediksi waktu eksekusi pekerjaan Spark di klaster Hadoop.	Eksperimen empiris dengan *WordCount, SVM, KMeans, PageRank,* dan *Graph (NWeight)* menggunakan HiBench.	Model matematis baru mampu menjelaskan pola performa yang tidak sesuai dengan hukum Amdahl atau Gustafson, dan memberikan prediksi waktu eksekusi akurat.	Memberikan pendekatan kuantitatif untuk karakterisasi performa paralel yang dapat diadaptasi ke konteks bahasa pemrograman umum.	Fokus pada Spark dan Hadoop, bukan bahasa pemrograman; model dibatasi untuk variasi jumlah executor, bukan arsitektur berbeda.
7	Cárdenas, Arroba, &	*Lock-Free Simulation Algorithm to*	Journal of Parallel and	Meningkatkan performa simulasi DEVS dengan	Implementasi dan uji pada xDEVS simulator	Algoritma *lock-free* menghasilkan efisiensi paralel	Sangat relevan untuk memahami	Fokus pada simulator DEVS; tidak mengkaji

No	Penulis & Tahun	Judul	Sumber (Jurnal/Penerbit)	Tujuan Penelitian	Metode / Dataset	Hasil Utama	Relevansi terhadap Penelitian Saya	Keterbatasan / Celah Penelitian
	Risco-Martín, 2025	*Enhance the Performance of Sequential and Parallel DEVS Simulators in Shared-Memory Architectures*	Distributed Computing	algoritma *lock-free* dan *work-stealing* untuk arsitektur berbagi memori.	menggunakan benchmark DEVStone.	tinggi dan mengurangi kontensi thread dibanding pendekatan berbasis *locking*.	strategi optimasi thread dan paralelisme pada arsitektur modern, yang dapat diterapkan untuk evaluasi performa bahasa.	langsung bahasa pemrograman umum; ruang lingkup domain terbatas.
8	Chaudhary, Agrawal, & Ali, 2025	*Modern Programming Languages – Characteristics and Recommendations for Instruction*	Issues in Information Systems	Mengidentifikasi fitur yang mendefinisikan "bahasa pemrograman modern" dan merekomendasikan penggunaannya dalam pendidikan IT.	Analisis literatur dan komparasi karakteristik bahasa modern seperti Python, Julia, dan Rust.	Menemukan bahwa ciri utama bahasa modern mencakup efisiensi, keamanan, paralelisme, dan kemudahan pemeliharaan.	Memberi landasan konseptual tentang bagaimana "modernitas" bahasa dikaitkan dengan fitur concurrency dan parallelism.	Pendekatan normatif, bukan empiris; tidak mengukur performa aktual.
9	Cheeseman et al., 2023	*When Concurrency Matters: Behaviour-Oriented Concurrency*	Proceedings of the ACM on Programming Languages (OOPSLA)	Mengembangkan paradigma baru "Behaviour-Oriented Concurrency (BoC)" untuk mengintegrasikan paralelisme dan koordinasi.	Model formal + implementasi prototipe C++ runtime + evaluasi Savina benchmark suite.	BoC memungkinkan akses eksklusif atomik ke sumber daya paralel, menawarkan alternatif terhadap *actor model* tradisional dengan performa sebanding atau lebih baik.	Relevan untuk desain model concurrency tingkat tinggi dan efisiensi koordinasi antar-thread dalam bahasa seperti Java atau Rust.	Implementasi awal; belum diuji pada bahasa mainstream; kompleksitas teoretis tinggi.
10	Crichton & Krishnamurthi, 2024	*Profiling Programming Language Learning*	Proceedings of the ACM on Programming Languages (OOPSLA)	Menyelidiki proses pembelajaran bahasa pemrograman melalui analisis data kuis interaktif pada buku *The Rust Programming Language*.	Eksperimen longitudinal selama 13 bulan dengan >62.000 peserta dan 1,14 juta respon.	Menemukan bahwa konsep kompleks seperti *ownership* menjadi titik kesulitan utama dalam pembelajaran Rust; intervensi meningkatkan hasil hingga 20%.	Memberi wawasan tentang kesulitan memahami konsep concurrency dan ownership di Rust, berguna untuk konteks pedagogis penelitian bahasa modern.	Fokus pada pembelajaran, bukan performa eksekusi; hasil lebih bersifat edukasional daripada teknis.
11	Gustafson, 1988	*Reevaluating Amdahl's Law*	Communications of the ACM	Mengusulkan model baru yang memperbaiki hukum Amdahl dengan mempertimbangkan skala problem yang bertambah seiring pertumbuhan jumlah prosesor.	Analisis teoretis terhadap model komputasi paralel dengan eksperimen simulasi pada sistem multiprosesor.	Dihasilkan *Gustafson's Law* yang menunjukkan bahwa speedup total dapat meningkat proporsional terhadap jumlah prosesor bila ukuran masalah juga meningkat.	Menjadi dasar konseptual penting untuk menilai skalabilitas performa paralel dalam eksperimen lintas bahasa (Go, Rust, Java, Python).	Model bersifat ideal; tidak memperhitungkan overhead nyata seperti sinkronisasi, latensi memori, atau penjadwalan thread.
12	Jarlow, Stylianopoulos, & Papatriantafilou, 2025	*QPOPSS: Query and Parallelism Optimized Space-Saving for Finding Frequent Stream Elements*	Journal of Parallel and Distributed Computing	Mengembangkan algoritma *Space-Saving* baru yang mendukung eksekusi paralel efisien untuk analisis elemen frekuen pada data streaming.	Desain algoritma + eksperimen empiris pada data streaming dengan berbagai tingkat kontensi thread.	QPOPSS meningkatkan throughput secara linear dengan jumlah thread sambil mempertahankan akurasi tinggi dan efisiensi memori.	Memberikan bukti empiris bahwa desain struktur data paralel yang efisien dapat menjaga akurasi dan kecepatan tinggi — relevan untuk evaluasi mekanisme concurrency modern.	Fokus pada domain streaming; tidak menilai bahasa pemrograman; hasil bergantung pada arsitektur CPU tertentu.
13	Jung et al., 2021	*Safe Systems Programming in Rust*	Communications of the ACM	Menjelaskan prinsip desain Rust yang menggabungkan keamanan memori dan kontrol sistem rendah melalui konsep *ownership* dan *borrowing*.	Analisis konseptual dan formal berbasis proyek RustBelt dengan pembuktian *semantic type soundness*.	Rust membuktikan keseimbangan antara keamanan tinggi dan kendali sumber daya yang sebelumnya dianggap tak mungkin.	Memberikan kerangka teoritis untuk memahami mengapa Rust unggul dalam efisiensi dan keamanan sistem paralel.	Tidak berisi eksperimen performa nyata; fokus pada argumen konseptual dan formal.

No	Penulis & Tahun	Judul	Sumber (Jurnal/Penerbit)	Tujuan Penelitian	Metode / Dataset	Hasil Utama	Relevansi terhadap Penelitian Saya	Keterbatasan / Celah Penelitian
14	Klabnik & Nichols, 2018	*The Rust Programming Language*	No Starch Press	Menyediakan panduan komprehensif untuk pemrograman Rust termasuk konsep *fearless concurrency* dan *ownership system.*	Studi literatur dan pendekatan tutorial dengan proyek nyata (multithreaded web server).	Menunjukkan bagaimana Rust menerapkan model *message passing* dan *ownership* untuk menghindari *data race*.	Memberikan pemahaman praktis tentang implementasi dan perilaku concurrency di Rust — sangat relevan untuk analisis empiris lintas bahasa.	Bersifat deskriptif dan instruksional; tidak menyajikan analisis performa formal atau kuantitatif.
15	Liu, Millstein, & Musuvathi, 2021	*Safe-by-default Concurrency for Modern Programming Languages*	ACM Transactions on Programming Languages and Systems	Mengusulkan paradigma *volatile-by-default* untuk menjamin keamanan memori pada concurrency tanpa mengorbankan performa sepenuhnya.	Implementasi dan uji pada Java HotSpot dan Julia JIT Compiler; evaluasi performa pada x86-64 dan ARM-v8.	Menunjukkan bahwa *sequential consistency (SC)* bisa dipertahankan dengan overhead performa yang masih dapat diterima.	Sangat relevan untuk memahami kompromi antara keamanan dan efisiensi dalam desain model concurrency bahasa modern.	Fokus pada JVM dan Julia; tidak mencakup bahasa non-VM seperti Go dan Rust.
16	Menard et al., 2023	*High-Performance Deterministic Concurrency Using Lingua Franca*	ACM Transactions on Architecture and Code Optimization	Menunjukkan bahwa deterministik concurrency dapat dicapai tanpa kehilangan performa melalui model *reactor*.	Implementasi *Lingua Franca* dan uji dengan Savina benchmark dibandingkan Akka dan CAF.	LF mengungguli Akka dan CAF hingga 1.86× dalam waktu eksekusi dengan tetap mempertahankan determinisme.	Menunjukkan pendekatan concurrency deterministik yang efisien; relevan sebagai pembanding konsep model actor pada Go atau Java.	Fokus pada model *reactor*, bukan bahasa umum; tidak menilai implikasi di sistem produksi besar.
17	Morshed & Roy, 2024	*Go vs. Java: A Detailed Performance Analysis in Concurrent Programming*	Journal of Computer Science Innovations	Membandingkan performa concurrency Go dan Java pada berbagai tugas komputasi paralel.	Benchmark empiris mencakup *Matrix Multiplication*, *Sorting*, dan *File Processing*; uji pada CPU multicore.	Go unggul pada beban ringan-menengah, sementara Java unggul pada optimasi jangka panjang dengan *JIT compilation*.	Relevan langsung untuk evaluasi performa lintas bahasa — salah satu studi pembanding utama bagi penelitian Anda.	Tidak mencakup Rust dan Python; hanya menguji subset kecil beban kerja dan metrik performa terbatas.
18	Newhall et al., 2025	*An Introductory-Level Undergraduate CS Course that Introduces Parallel Computing*	Journal of Parallel and Distributed Computing	Mendesain kurikulum pengantar sistem komputer dengan integrasi awal konsep paralelisme dan distribusi.	Studi pedagogis dengan pendekatan *active learning*; evaluasi hasil belajar mahasiswa.	Mahasiswa menunjukkan peningkatan pemahaman PDC dan kepercayaan diri dalam menerapkan paralelisme.	Memberikan konteks pendidikan untuk memahami pentingnya paralelisme sejak awal — mendukung pendekatan metodologis penelitian.	Fokus pendidikan, bukan performa teknis; tidak menguji bahasa atau sistem.
19	Pearce, 2021	*A Lightweight Formalism for Reference Lifetimes and Borrowing in Rust*	ACM Transactions on Programming Languages and Systems	Menyusun formalitas ringan untuk menjelaskan sistem *lifetime* dan *borrowing* dalam Rust.	Pengembangan *flow-sensitive type system* dengan pembuktian formal dan pengujian model menggunakan 500B program.	Menegaskan keamanan memori Rust secara formal serta efektivitas *borrow checker*.	Memberikan dasar logis untuk menjelaskan stabilitas dan keamanan model concurrency Rust dibandingkan bahasa lain.	Tidak menganalisis performa eksekusi atau komputasi paralel secara empiris.
20	Nigro, 2022	*Performance of Parallel K-Means Algorithms in Java*	MDPI Algorithms	Mengevaluasi dua pendekatan paralel K-Means di Java: *parallel streams* dan *Theatre actor system.*	Eksperimen komputasi intensif pada multicore shared-memory machine.	Implementasi *Theatre actor system* memberikan kinerja lebih tinggi daripada parallel streams.	Memberikan bukti empiris tentang keunggulan model *actor* dalam Java — relevan untuk membandingkan efisiensi mekanisme thread antarbahasa.	Fokus tunggal pada Java; tidak mempertimbangkan bahasa atau model paralelisme lain.
21	Poolla & Saxena, 2023	*On Extending Amdahl's Law to Learn Computer Performance*	Journal of Parallel and Distributed Computing	Mengembangkan perluasan hukum Amdahl agar dapat diterapkan pada sistem dengan beberapa sumber	Analisis teoretis dan regresi multivariat berbasis data benchmark (SPEC CPU2017,	Model multivariabel menunjukkan akurasi prediksi performa di atas 95%, menggabungkan	Menyediakan kerangka empiris-kuantitatif untuk memodelkan percepatan	Hanya menguji arsitektur CPU, tidak mengevaluasi pengaruh model bahasa atau runtime;

No	Penulis & Tahun	Judul	Sumber (Jurnal/Penerbit)	Tujuan Penelitian	Metode / Dataset	Hasil Utama	Relevansi terhadap Penelitian Saya	Keterbatasan / Celah Penelitian
				daya konfigurabel melalui pendekatan pembelajaran mesin.	PCMark 10) pada empat arsitektur CPU berbeda.	efek multi-resource dalam estimasi *speedup*.	eksekusi paralel pada sistem modern; dapat dijadikan basis analisis untuk evaluasi lintas bahasa.	pendekatan ML belum diuji pada sistem heterogeneous.
22	Quislant, Gutierrez, & Plata, 2024	*Exploring Multiprocessor Approaches to Time Series Analysis*	Journal of Parallel and Distributed Computing	Mengeksplorasi efisiensi pendekatan multiprosesor pada algoritma *Matrix Profile* untuk analisis deret waktu berskala besar.	Eksperimen dengan implementasi SCAMP menggunakan *Hardware Transactional Memory (HTM)*, *locks*, dan *barriers* di multiprosesor.	HTM dan tiling optimization memberikan *speedup* hingga 100× dibanding sequential dan 3× dibanding baseline paralel sambil menjaga efisiensi memori.	Relevan untuk memahami implementasi konkuren efisien dengan sinkronisasi spekulatif yang dapat diterapkan pada studi performa Go/Rust.	Terbatas pada domain time-series; tidak mengevaluasi efek pada bahasa umum; hasil tergantung pada konfigurasi CPU.
23	Rogowski et al., 2023	*mpi4py.futures: MPI-Based Asynchronous Task Execution for Python*	IEEE Transactions on Parallel and Distributed Systems	Mengembangkan *mpi4py.futures* untuk mengeksekusi tugas asinkron di Python menggunakan MPI sebagai backend paralel.	Implementasi *drop-in replacement* untuk `concurrent.futures`; uji performa pada sistem Cray XC40 dan shared-memory.	Meningkatkan throughput hingga 3.7× dan bandwidth hingga 2.9× dibanding `concurrent.futures` dan outperform Dask di sebagian besar kasus.	Memberi bukti empiris bahwa Python dapat mencapai skalabilitas tinggi melalui model MPI — mendukung relevansi Python dalam konteks parallelism modern.	Terbatas pada task-level parallelism; tidak mengevaluasi parallelism berbasis thread internal; tidak membahas overhead GIL sepenuhnya.
24	Schryen, 2024	*Speedup and Efficiency of Computational Parallelization: A Unifying Approach and Asymptotic Analysis*	Journal of Parallel and Distributed Computing	Menyusun model unifikasi hukum *speedup* (Amdahl, Gustafson, dsb.) untuk memahami efisiensi paralel dalam konteks asimtotik.	Model matematis generik diuji dengan eksperimen pada *matrix multiplication* dan *LU decomposition*.	Menghasilkan taksonomi baru *scalability cases* (sublinear, linear, superlinear) dan menjelaskan batas teoritis efisiensi paralel.	Menyediakan dasar teoretis universal untuk menilai hasil *benchmark concurrency* bahasa modern seperti Go, Rust, Java, dan Python.	Fokus pada sistem homogen; tidak mempertimbangkan overhead nyata seperti komunikasi dan cache-coherency.
25	Wang et al., 2025	*Optimizing General Sparse Matrix-Matrix Multiplication on the GPU*	ACM Transactions on Architecture and Code Optimization	Mengoptimalkan SpGEMM (Sparse Matrix-Matrix Multiplication) untuk GPU dengan pendekatan pembelajaran mesin dan heuristik adaptif.	Implementasi adaptif GPU kernel dengan dataset SuiteSparse dan analisis kinerja terhadap cuSPARSE, Kokkos, spECK, dan lainnya.	Model adaptif menunjukkan peningkatan performa signifikan dibanding state-of-the-art; mencapai rata-rata *speedup* 4.91–5.67×.	Menunjukkan bagaimana kombinasi optimisasi ML dan paralelisme GPU dapat meningkatkan efisiensi komputasi — mendukung aspek *performance benchmarking*.	Spesifik pada GPU; tidak menilai CPU parallelism atau bahasa pemrograman; fokus komputasi ilmiah.
26	Yuan & Yang, 2020	*Effective Concurrency Testing for Distributed Systems*	ACM ASPLOS	Mengembangkan *Morpheus*, alat pengujian konkurensi berbasis *partial order sampling* untuk sistem terdistribusi.	Implementasi dan uji pada 4 sistem Erlang nyata (RabbitMQ, Mnesia, dll).	Morpheus menemukan 11 bug baru di sistem nyata dan memperlihatkan efisiensi deteksi kesalahan yang tinggi dengan biaya uji rendah.	Relevan untuk aspek keandalan sistem konkuren; menunjukkan pentingnya *testing framework* dalam evaluasi bahasa berorientasi concurrency.	Fokus pada testing correctness, bukan performa; terbatas pada bahasa Erlang dan sistem terdistribusi.
27	Zeng, 2023	*Performance Analysis of Parallel Programming Models for C++*	IOP Journal of Physics: Conference Series	Membandingkan performa model paralel di C++: *std::thread*, *OpenMP*, dan *Pthreads* pada operasi matriks.	Eksperimen empiris dengan ukuran matriks berbeda dan variasi jumlah thread.	*OpenMP* unggul untuk matriks besar; *std::thread* efisien untuk ukuran kecil; performa sangat bergantung pada ukuran data dan jumlah thread.	Memberi kerangka empiris untuk menganalisis perbedaan model paralel antar bahasa — relevan untuk metodologi pengujian lintas bahasa (Go, Rust, Java, Python).	Fokus tunggal pada C++; tidak menganalisis arsitektur multi-core heterogen atau bahasa lain.

No	Penulis & Tahun	Judul	Sumber (Jurnal/Penerbit)	Tujuan Penelitian	Metode / Dataset	Hasil Utama	Relevansi terhadap Penelitian Saya	Keterbatasan / Celah Penelitian
28	Pearce, 2021	*A Lightweight Formalism for Reference Lifetimes and Borrowing in Rust*	ACM Transactions on Programming Languages and Systems	Mengembangkan formalitas ringan (FR) untuk memodelkan sistem *ownership*, *borrowing*, dan *lifetime* pada Rust agar mudah dipahami dan diuji secara formal.	Pendekatan teoretis dengan kalkulus *flow-sensitive type system*; model formal dan pembuktian *type soundness*; diuji dengan *model checking* dan *fuzz testing* pada 500 miliar program.	FR berhasil menangkap aspek penting Rust seperti *move semantics*, *mutable/immutable borrowing*, *partial moves*, dan *lifetimes*, sekaligus membuktikan keamanan memori formal.	Menyediakan dasar teoretis untuk memahami model *ownership* dan *borrowing* Rust yang menjadi fondasi efisiensi dan keamanan dalam concurrency — sangat relevan bagi analisis performa Rust.	Tidak menilai performa eksekusi nyata; fokus pada model formal; tidak mencakup aspek concurrency eksplisit atau runtime parallelism.
29	Pons, Petit, & Sahuquillo, 2025	*Advanced Resource Management: A Hands-on Master Course in HPC and Cloud Computing*	Journal of Parallel and Distributed Computing	Mendeskripsikan pengembangan kurikulum pascasarjana berbasis praktik untuk manajemen sumber daya dan evaluasi performa sistem HPC dan cloud modern.	Studi deskriptif berbasis *course design research* dengan modul teoretis dan eksperimen praktis pada Proxmox VE, Perf, dan Intel VTune Profiler.	Kursus ini meningkatkan pemahaman mahasiswa terhadap manajemen sumber daya komputasi paralel serta analisis performa *workloads* nyata seperti SPEC CPU dan Tailbench.	Relevan sebagai referensi metodologis untuk desain eksperimental dan pengukuran performa sistem dalam studi concurrency dan parallelism lintas bahasa.	Fokus pendidikan, bukan penelitian empiris performa; hasil tidak mencakup pengujian bahasa pemrograman atau analisis sistem multithread secara langsung.

1. Pola Umum

Aspek	Pola yang Ditemukan
Topik Umum	Mayoritas penelitian fokus pada **evaluasi performa, efisiensi paralelisme, dan keamanan eksekusi konkuren** pada sistem modern (terutama bahasa pemrograman dan platform komputasi).
Pendekatan	Umumnya menggabungkan **pendekatan teoretis dan empiris**. Artikel seperti *Arora et al. (2021, 2023)* dan *Pearce (2021)* membangun dasar formal (model semantik, teori tipe), sedangkan *Morshed & Roy (2024)*, *Nigro (2022)*, dan *Rogowski et al. (2023)* berfokus pada pengukuran empiris (benchmark dan throughput).
Metode Dominan	- Benchmark eksperimental (*Matrix Multiplication*, *K-Means*, *File Processing*). - Analisis teoretis (*Amdahl's Law*, *Gustafson's Law*, *semantik concurrency*). Simulasi skala besar untuk mengukur *speedup*, *efficiency*, dan *scalability*.
Objek Penelitian	Bahasa pemrograman modern (Go, Rust, Java, Python, C++), arsitektur multi-core dan GPU, serta paradigma concurrency (actor model, lock-free, task-based).
Fokus Evaluasi	1️⃣ Efisiensi eksekusi paralel 2️⃣ Manajemen memori dan keamanan data race 3️⃣ Skalabilitas sistem saat beban meningkat 4️⃣ Keseimbangan antara performa dan determinisme

2. Tren Riset

Tren	Deskripsi
a. Dominasi Rust sebagai objek penelitian baru	Rust sering digunakan untuk mengeksplorasi *safe concurrency*, dengan fokus pada *ownership model*, *borrow checker*, dan *type safety* (*Jung et al., 2021; Pearce, 2021*). Rust diposisikan sebagai bahasa "post-C++" yang menjembatani performa tinggi dan keamanan memori.
b. Penggabungan teori klasik dengan pendekatan modern	Banyak penelitian memperluas hukum klasik seperti *Amdahl* dan *Gustafson* untuk konteks heterogen (*Al-hayanni, 2020; Poolla & Saxena, 2023; Schryen, 2024*), menunjukkan tren menuju **generalized speedup modeling**.
c. Eksperimen empiris lintas bahasa	Studi komparatif (*Abhinav et al., 2020; Morshed & Roy, 2024; Zeng, 2023*) menilai Go, Java, Python, dan C++ untuk concurrency. Tren ini menyoroti pergeseran ke **evaluasi praktis performa nyata, bukan hanya teori.**
d. Fokus pada deterministik concurrency	Riset seperti *Menard et al. (2023)* dan *Cheeseman et al. (2023)* menekankan *deterministic parallelism* dan *behavior-oriented concurrency* untuk mengatasi nondeterminisme akibat *thread interleaving*.
e. Integrasi pembelajaran mesin dan HPC	Studi seperti *Wang et al. (2025)* dan *Pons et al. (2025)* menggabungkan AI/HPC untuk optimisasi otomatis (*machine learning-assisted optimization*). Ini menjadi tren baru dalam riset performa sistem paralel.

3. Perbedaan Antar Penelitian

Aspek	Variasi yang Ditemukan
Pendekatan Teori vs Praktik	- *Pearce (2021)*, *Arora (2021, 2023)* → formal dan matematis. - *Morshed & Roy (2024)*, *Nigro (2022)* → eksperimental dan aplikatif.
Objek Penelitian	- *Bahasa pemrograman*: Go, Java, Rust, Python, C++. - *Arsitektur*: CPU multicore, GPU, cluster, hingga cloud HPC (*Pons, 2025*).
Level Analisis	- Mikro (kode, thread, borrow checker).

Aspek	Variasi yang Ditemukan
Fokus Tujuan	- Sebagian menekankan **keamanan dan determinisme** (*Rust, Lingua Franca*). - Sebagian lain menekankan **performa dan efisiensi** (*Go, Java, C++*).
Metode Evaluasi	- Formal proof dan model checking (Rust). - Benchmark empiris (*SPEC CPU, Tailbench, K-Means*). - Simulasi matematika (Amdahl's extension).

4. Research Gap (Celah Penelitian)

Area	Celah yang Belum Tersentuh Secara Mendalam
Integrasi antara efisiensi performa dan keamanan	Banyak studi membahas performa (Go, Java, Python) **atau** keamanan memori (Rust), tapi jarang yang **menggabungkan keduanya** dalam konteks empiris terukur.
Analisis biaya dan efisiensi ekonomi	Hampir semua riset fokus pada kecepatan dan efisiensi teknis. **Aspek ekonomi** (biaya energi, operasional, infrastruktur — seperti yang diangkat dalam *MetPen02*) belum banyak dikaji.
Evaluasi paralelisme lintas platform nyata (heterogeneous)	Studi-studi concurrency sering terbatas pada satu platform (CPU/GPU). Belum banyak penelitian yang membandingkan **kinerja bahasa lintas arsitektur heterogen (CPU + GPU + Cloud)**.
Keterhubungan antara *theoretical model* dan hasil empiris	Ada kesenjangan antara model teoretis seperti *Amdahl's/Gustafson's extensions* dan performa riil program di Go, Rust, Java. Dibutuhkan penelitian yang memvalidasi teori menggunakan data empiris nyata.
Pengukuran concurrency behavior pada Python modern	Python masih jarang dianalisis secara mendalam dari sisi *asynchronous performance* (mis. asyncio, multiprocessing, MPI4Py) pada konteks industri atau HPC.