BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi komputasi modern ditandai dengan meningkatnya kebutuhan akan sistem yang mampu memanfaatkan arsitektur multi-core processor secara optimal. Dalam konteks ini, konsep concurrency (eksekusi tugas secara bersamaan) dan parallelism (eksekusi simultan menggunakan beberapa inti prosesor) menjadi sangat penting untuk meningkatkan performa aplikasi. Keduanya telah menjadi fondasi pengembangan perangkat lunak berskala besar, mulai dari web server berperforma tinggi, big data processing systems, hingga komputasi ilmiah di bidang high-performance computing (HPC). Optimalisasi concurrency dan parallelism memungkinkan perangkat lunak untuk mengurangi waktu eksekusi, meningkatkan throughput, serta memanfaatkan sumber daya perangkat keras secara efisien (Lee & Aggarwal, 1987).

Berbagai bahasa pemrograman modern menawarkan pendekatan berbeda dalam mendukung concurrency dan parallelism. Go, yang dikembangkan Google, dikenal dengan goroutines dan channels yang ringan serta terintegrasi dengan runtime scheduler, sehingga memudahkan pengembang menulis kode konkuren dengan overhead minimal (Castro et al., 2019). Rust, sebagai bahasa sistem modern, mengusung ownership model yang menjamin keamanan memori sekaligus menghindari data race, serta mendukung asynchronous programming melalui pustaka seperti Tokio (Klabnik & Nichols, 2018). Java, yang telah lama menjadi standar industri, memiliki ekosistem matang melalui ExecutorService, Fork/Join Framework, dan Parallel Streams, dengan dukungan just-in-time compilation (JIT)

serta garbage collector yang terus berevolusi (Lea, 2000). Sementara itu, Python, meskipun populer untuk komputasi ilmiah, memiliki keterbatasan mendasar berupa Global Interpreter Lock (GIL) yang membatasi eksekusi paralel berbasis threads pada CPU-bound tasks. Upaya mitigasi biasanya dilakukan dengan modul multiprocessing atau pustaka eksternal seperti NumPy yang memanfaatkan BLAS (Van der Walt & Aivazis, 2011). Perkembangan terbaru melalui PEP 703 bahkan membuka arah baru dengan opsi build Python tanpa GIL, yang berpotensi meningkatkan performa paralel di masa depan (Gross, 2023).

Sejumlah penelitian terdahulu telah membandingkan performa concurrency dan parallelism pada beberapa bahasa pemrograman. Penelitian oleh Abhinav et al., (2020) mengevaluasi Go dan Java menggunakan matrix multiplication serta PageRank, dan menemukan bahwa Go unggul dalam efisiensi goroutine dan kecepatan kompilasi, sementara Java lebih baik dalam eksekusi matriks besar berkat optimasi ExecutorService. Sementara itu, Costanza et al., (2019) membandingkan Go, Java, dan C++17 pada pipeline bioinformatika (elPrep) dan menemukan bahwa Go memberikan keseimbangan terbaik antara runtime dan konsumsi memori, Java lebih cepat namun boros memori, sedangkan C++17 mengalami overhead signifikan karena reference counting.

Studi terbaru oleh Diehl et al. (2023) membandingkan sepuluh bahasa pemrograman menggunakan 1D heat equation solver dan menemukan variasi performa yang signifikan, dengan Python sebagai yang paling lambat dan C++, Rust, serta Chapel sebagai yang tertinggi. Sementara itu, Menard et al. (2023) membuktikan bahwa deterministic concurrency model seperti Lingua Franca dapat

mengungguli framework aktor non-deterministik seperti Akka (1.86×) dan CAF (1.42×) pada Savina benchmark suite. Temuan ini menunjukkan bahwa faktor-faktor seperti determinisme, model konkurensi, dan mekanisme scheduling memiliki dampak signifikan terhadap performa yang belum banyak dieksplorasi dalam konteks bahasa pemrograman tingkat tinggi seperti Go, Rust, Java, dan Python secara komprehensif.

Selain itu, penelitian tentang matrix-matrix multiplication menunjukkan bahwa kernel komputasi ini merupakan beban kerja representatif untuk mengukur kemampuan parallelism. Michailidis dan Margaritis (2012) menekankan bahwa optimasi algoritmik seperti blocking dan loop interchange sangat berpengaruh terhadap performa pada multi-core. Temuan lebih mutakhir oleh Akoushideh dan Shahbahrami (2022) menunjukkan bahwa kombinasi SIMD, OpenMP, dan OpenCL dapat meningkatkan performa hingga 32 kali lipat dibanding implementasi serial pada CPU. Namun, mayoritas studi tersebut lebih berfokus pada model pemrograman paralel (misalnya OpenMP atau OpenCL) daripada membandingkan langsung bahasa pemrograman tingkat tinggi.

Dari literatur tersebut dapat dilihat bahwa studi yang ada masih terbatas pada kasus tertentu, baik berupa algoritma spesifik atau aplikasi domain tertentu, dan umumnya hanya membandingkan sebagian kecil bahasa pemrograman. Meskipun benchmark suite standar seperti Savina telah digunakan untuk evaluasi actor frameworks (Menard et al., 2023), evaluasi komprehensif yang mencakup berbagai model konkurensi pada bahasa tingkat tinggi seperti Go, Rust, Java, dan Python masih terbatas. Belum banyak kajian yang melakukan evaluasi menyeluruh

terhadap performa concurrency dan parallelism pada beberapa bahasa modern sekaligus dengan metodologi benchmarking yang terstandar. Mengingat perbedaan mendasar dalam model eksekusi, runtime, serta strategi pengelolaan memori antar bahasa, diperlukan analisis komprehensif untuk memahami kelebihan dan keterbatasan masing-masing. Evaluasi lintas bahasa dengan mengukur metrik seperti waktu eksekusi, pemanfaatan CPU, konsumsi memori, serta skalabilitas pada berbagai jumlah inti prosesor akan memberikan gambaran lebih objektif mengenai performa concurrency dan parallelism.

Oleh karena itu, penelitian ini dilakukan untuk mengevaluasi dan membandingkan performa concurrency dan parallelism pada empat bahasa pemrograman modern, yaitu Go, Rust, Java, dan Python. Studi ini menggunakan benchmark tasks representatif seperti matrix multiplication dan parallel sorting untuk menguji skenario komputasi intensif, serta pola producer-consumer untuk mengevaluasi concurrency pada beban kerja ringan. Dengan mengukur metrik kunci pada lingkungan multi-core processor, penelitian ini diharapkan dapat memberikan kontribusi ilmiah berupa pemahaman yang lebih mendalam mengenai performa concurrency dan parallelism pada bahasa pemrograman modern, sekaligus memberikan wawasan praktis bagi pengembang, peneliti, maupun industri dalam memilih bahasa yang sesuai untuk aplikasi berbasis komputasi paralel.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan, maka permasalahan penelitian ini dapat dirumuskan sebagai berikut:

1. Bagaimana perbedaan performa concurrency dan parallelism pada bahasa pemrograman Go, Rust, Java, dan Python ketika diuji menggunakan benchmark tasks representatif seperti matrix multiplication, parallel sorting, dan pola producer–consumer?
2. Sejauh mana masing-masing bahasa pemrograman mampu memanfaatkan prosesor multi-inti dalam hal skalabilitas, waktu eksekusi, penggunaan CPU, dan konsumsi memori?
3. Faktor apa saja yang mempengaruhi kelebihan dan keterbatasan setiap bahasa dalam mendukung concurrency dan parallelism, ditinjau dari aspek model eksekusi, manajemen memori, serta mekanisme runtime?
4. Bahasa pemrograman mana yang menunjukkan efisiensi terbaik untuk skenario komputasi intensif dan beban kerja konkuren, serta bagaimana rekomendasi pemanfaatannya pada konteks pengembangan perangkat lunak modern?

1.3 Batasan Masalah

Agar penelitian ini lebih terarah dan fokus, maka ruang lingkup penelitian dibatasi sebagai berikut:

1.3.1 Bahasa Pemrograman yang Diteliti

Penelitian hanya mencakup empat bahasa pemrograman modern, yaitu Go, Rust, Java, dan Python. Bahasa lain seperti C/C++, C#, atau Erlang tidak termasuk dalam lingkup penelitian ini.

1.3.2 Jenis Benchmark

Benchmark yang digunakan terbatas pada tugas komputasi representatif, yaitu matrix multiplication, parallel sorting, serta pola

producer–consumer. Benchmark lain seperti graf komputasi, simulasi numerik kompleks, atau GPU-based workloads tidak dibahas.

1.3.3 Lingkungan Pengujian

Eksperimen dilakukan pada lingkungan multi-core CPU dengan konfigurasi perangkat keras dan perangkat lunak yang dikontrol (misalnya versi kompiler, interpreter, runtime, serta sistem operasi). Pengujian pada GPU, cluster, atau arsitektur heterogen tidak termasuk dalam penelitian ini.

1.3.4 Implementasi Program

Implementasi benchmark di setiap bahasa menggunakan pustaka standar atau mekanisme bawaan bahasa. Optimalisasi tingkat lanjut dengan pustaka eksternal pihak ketiga (misalnya NumPy untuk Python atau BLAS untuk Java/Rust) hanya dipertimbangkan secara terbatas untuk perbandingan, tetapi tidak menjadi fokus utama.

1.3.5 Metrik yang Dianalisis

Metrik performa yang dianalisis meliputi waktu eksekusi, pemanfaatan CPU, konsumsi memori, serta skalabilitas terhadap jumlah inti prosesor. Metrik lain seperti energy consumption, ukuran kode, atau kompleksitas pengembangan tidak menjadi fokus utama penelitian ini.

1.4 Tujuan Penelitian

Penelitian ini bertujuan untuk mengevaluasi dan membandingkan performa concurrency dan parallelism pada beberapa bahasa pemrograman modern, yaitu Go, Rust, Java, dan Python, dengan menggunakan benchmark representatif pada lingkungan prosesor multi-inti.

Secara lebih rinci, penelitian ini memiliki tujuan khusus sebagai berikut:

1. Mengukur dan menganalisis perbedaan waktu eksekusi, pemanfaatan CPU, konsumsi memori, serta skalabilitas dari implementasi concurrency dan parallelism pada Go, Rust, Java, dan Python.
2. Mengevaluasi efisiensi setiap bahasa pemrograman dalam menyelesaikan benchmark tasks komputasi intensif seperti matrix multiplication dan parallel sorting, serta beban kerja konkuren seperti pola producer–consumer.
3. Mengidentifikasi faktor-faktor teknis yang memengaruhi kinerja setiap bahasa, termasuk model eksekusi, mekanisme runtime, dan strategi manajemen memori.
4. Memberikan rekomendasi pemilihan bahasa pemrograman yang sesuai untuk kebutuhan aplikasi yang menuntut concurrency dan parallelism tinggi, baik dalam konteks akademis maupun industri.

1.5 Manfaat Penelitian

1.5.1 Manfaat Teoritis

Penelitian ini diharapkan dapat menambah khazanah literatur akademis di bidang ilmu komputer, khususnya pada topik concurrency dan parallelism dalam bahasa pemrograman modern. Hasil penelitian dapat menjadi rujukan bagi peneliti lain yang ingin mengembangkan studi sejenis, serta memperkaya pemahaman mengenai perbedaan model eksekusi, strategi runtime, dan pengelolaan memori antar bahasa.

1.5.2 Manfaat Praktis

Hasil penelitian dapat memberikan panduan bagi pengembang perangkat lunak dalam memilih bahasa pemrograman yang sesuai dengan

kebutuhan aplikasi, khususnya aplikasi yang menuntut concurrency tinggi maupun pemrosesan paralel berskala besar. Dengan demikian, penelitian ini dapat membantu meningkatkan efisiensi pengembangan dan performa aplikasi nyata.

1.6 Sistematika Penulisan

1. BAB I PENDAHULUAN

   Bab ini menguraikan latar belakang penelitian yang menjelaskan urgensi evaluasi performa concurrency dan parallelism pada bahasa pemrograman modern. Selanjutnya dipaparkan rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian baik secara teoritis maupun praktis, serta sistematika penulisan yang menggambarkan struktur keseluruhan penelitian.

2. BAB II TINJAUAN PUSTAKA

   Bab ini menyajikan landasan teori yang mencakup konsep dasar concurrency dan parallelism, model pemrograman paralel dan konkuren (thread-based, message-passing, asynchronous, data-parallel, dan hybrid model), hukum Amdahl dan Gustafson sebagai dasar teoritis pengukuran performa sistem paralel, karakteristik empat bahasa pemrograman yang diteliti (Go, Rust, Java, dan Python), serta benchmark komputasi paralel yang digunakan dalam penelitian. Bab ini juga mengulas penelitian-penelitian terdahulu yang relevan sebagai dasar komparasi dan pengembangan metodologi penelitian.

3. BAB III METODE PENELITIAN

Bab ini menjelaskan jenis dan pendekatan penelitian yang digunakan, yaitu penelitian eksperimental dengan pendekatan kuantitatif. Kemudian dipaparkan kerangka kerja penelitian yang terdiri dari lima tahapan utama: studi literatur, perancangan eksperimen, implementasi benchmark, pelaksanaan pengujian, dan analisis data. Bab ini juga menjelaskan tempat penelitian beserta spesifikasi perangkat keras dan perangkat lunak yang digunakan, serta teknik analisis data yang meliputi analisis kuantitatif deskriptif, analisis perbandingan (speedup dan efficiency), analisis skalabilitas berdasarkan hukum Amdahl dan Gustafson, serta visualisasi dan interpretasi hasil.

4. BAB IV ANALSIS DAN PERANCANGAN (untuk laporan akhir)

Bab ini membahas analisis hasil eksperimen serta rancangan sistem pengujian yang digunakan dalam penelitian. Analisis difokuskan pada evaluasi performa concurrency dan parallelism pada empat bahasa pemrograman (Go, Rust, Java, dan Python) berdasarkan metrik waktu eksekusi, pemanfaatan CPU, konsumsi memori, dan skalabilitas. Bagian perancangan menjelaskan desain sistem *benchmarking*, alur eksekusi, serta rancangan algoritma untuk setiap *benchmark* (matrix multiplication, parallel sorting, dan producer–consumer pattern), disertai diagram UML dan rancangan keluaran hasil uji yang menggambarkan struktur serta hubungan antar komponen sistem.

5. BAB V HASIL DAN PEMBAHASAN (untuk laporan akhir)

Bab ini akan menyajikan data hasil pengujian performa concurrency dan parallelism pada empat bahasa pemrograman yang diteliti. Hasil eksperimen akan disajikan dalam bentuk tabel, grafik, dan visualisasi lainnya. Pembahasan

akan menganalisis perbandingan performa antar bahasa, skalabilitas sistem terhadap penambahan jumlah thread, serta interpretasi hasil berdasarkan teori yang telah dipaparkan pada Bab II.

6. BAB VI KESIMPULAN DAN SARAN (untuk laporan akhir)

Bab ini akan menyimpulkan hasil penelitian secara menyeluruh dengan menjawab rumusan masalah yang telah ditetapkan. Kesimpulan akan mencakup perbandingan performa keempat bahasa pemrograman, identifikasi faktor-faktor yang memengaruhi kinerja masing-masing bahasa, serta rekomendasi pemilihan bahasa pemrograman untuk berbagai skenario aplikasi. Selain itu, bab ini juga akan menyajikan saran untuk pengembangan penelitian selanjutnya, termasuk kemungkinan perluasan cakupan benchmark, pengujian pada arsitektur perangkat keras berbeda, atau eksplorasi teknik optimasi lanjutan.

7. DAFTAR PUSTAKA

Bagian ini memuat seluruh referensi yang digunakan dalam penelitian, meliputi buku, artikel jurnal ilmiah, prosiding konferensi, dan sumber online yang relevan. Penulisan daftar pustaka mengikuti standar sitasi yang berlaku.

8. LAMPIRAN (untuk laporan akhir)

Lampiran akan memuat informasi pendukung seperti source code implementasi benchmark untuk keempat bahasa pemrograman, hasil log pengujian lengkap, tabel data mentah hasil eksperimen, serta dokumentasi teknis lainnya yang diperlukan untuk verifikasi dan replikasi penelitian.