Vol.7, No.2, Desember 2024
Available online at: http://jurnal.kdi.or.id/index.php/bt

Perbandingan Efisiensi Waktu dan Memori Pada C# dan Java dengan Metode Benchmarking

Alvin Hadiewijaya[1)*], Budi Wasito[2)]

[1)2)]*Teknik Informatika, Fakultas Informatika dan Komunikasi, Institut Bisnis dan Informatika Kwik Kian Gie Jakarta Utara, Indonesia*

[1)]56210008@student.kwikkiangie.ac.id

[2)]budi.wasito@kwikkiangie.ac.id

Article history:

Received 09 Des 2024;
Revised 12 Des 2024;
Accepted 19 Des 2024;
Available online 27 Des 2024

Keywords:

Benchmarking
C#
Efisiensi Memori
Java
Waktu Eksekusi

Abstrak

Penelitian ini bertujuan untuk membandingkan efisiensi waktu eksekusi dan penggunaan memori antara dua bahasa pemrograman, yaitu C# dan Java, menggunakan metode benchmarking. Penelitian ini menggunakan dataset berukuran kecil (1 juta data), sedang (5 juta data), dan besar (10 juta data) yang berisi ID, Nama, dan Alamat. Implementasi algoritma pencarian *string* dilakukan menggunakan *binary search* pada Java dan C#, dengan pengukuran waktu eksekusi menggunakan fungsi *System.nanoTime()* (Java) dan *Stopwatch* (C#). Konsumsi memori diukur menggunakan class *Process* pada C# dan fungsi *totalMemory()* serta *freeMemory()* pada Java. Data diuji dua kali untuk memastikan konsistensi hasil. Hasil pengujian menunjukkan bahwa C# memiliki keunggulan dalam kecepatan waktu eksekusi pada dataset kecil dan sedang, sedangkan Java lebih stabil untuk dataset besar. C# secara konsisten menunjukkan efisiensi dalam pengelolaan *thread* dan *runtime* di platform .NET, sedangkan Java memanfaatkan optimasi *Just-In-Time* (JIT) pada *Java Virtual Machine* (JVM) untuk performa yang lebih stabil pada *dataset* besar. Penelitian ini memberikan wawasan penting bagi pengembang perangkat lunak dalam memilih bahasa pemrograman berdasarkan kebutuhan aplikasi mereka. Penelitian lebih lanjut disarankan untuk mencakup evaluasi penggunaan CPU serta pengujian dengan algoritma lainnya untuk memberikan analisis yang lebih komprehensif. Temuan ini diharapkan dapat membantu pengembang dalam membuat keputusan strategis terkait pemilihan teknologi.

I. PENDAHULUAN

Dalam perkembangan perangkat lunak yang terus berkembang, faktor utama dalam implementasi sistem perangkat lunak adalah pemilihan bahasa pemrograman yang tepat untuk memenuhi kebutuhan aplikasi tertentu. Pada pemilihan ini, langkah penting bagi pengembang adalah mempertimbangkan beberapa faktor yang dapat memengaruhi performa perangkat lunak secara keseluruhan. Keputusan ini tidak hanya berdampak pada efisiensi waktu eksekusi, tetapi juga pada stabilitas sistem yang telah dibangun. Stabilitas sistem ini menjadi sangat penting dalam pengembangan perangkat lunak yang digunakan untuk menangani aplikasi berskala besar maupun yang bersifat kompleks. Oleh karena itu, diperlukan pendekatan yang tepat untuk mengetahui performa dari bahasa pemrograman yang digunakan.

Penelitian ini menggunakan metode *benchmarking* untuk membandingkan performa sistem, yang sering dilakukan oleh pengembang perangkat lunak untuk mengevaluasi waktu eksekusi dan efisiensi memori [1]. Penggunaan metode *benchmarking* memungkinkan evaluasi sistem secara menyeluruh, sehingga memudahkan pengembang perangkat lunak dalam mengidentifikasi area yang perlu ditingkatkan untuk mencapai efisiensi yang optimal [2]. Dalam penggunaan metode *benchmarking*, juga diberikan gambaran komprehensif terkait kelebihan dan kelemahan masing-masing bahasa pemrograman dalam menangani kebutuhan aplikasi tertentu. Hal ini memberikan keuntungan tersendiri bagi pengembang perangkat lunak, karena mereka dapat memahami bahasa pemrograman secara lebih spesifik serta karakteristiknya secara mendalam.

Dalam pembuatan sistem perangkat lunak, bahasa C# dan Java dapat mendukung berbagai kebutuhan aplikasi. Namun, di antara kedua bahasa pemrograman tersebut terdapat perbedaan yang signifikan dalam menangani

* Corresponding author

ISSN 2622-2728 (online) 2622-271X (print)
doi: 10.32877/bt.v7i2.1906

dataset dengan ukuran yang berbeda, sehingga bagi pengembang perangkat lunak, hal ini menimbulkan tantangan untuk menentukan bahasa mana yang paling sesuai dengan kebutuhan spesifik mereka. Untuk hal ini, pemilihan bahasa pemrograman sering kali dipengaruhi oleh jenis aplikasi yang ingin dikembangkan, seperti aplikasi yang memiliki dataset besar atau aplikasi yang beroperasi secara real-time.

Bahasa pemrograman Java memiliki keunggulan dalam pengelolaan data berbasis objek, menjadikannya pilihan yang sangat ideal untuk menangani aplikasi yang membutuhkan pengolahan dataset berskala besar. Dalam aplikasi ini, Java menawarkan runtime yang sangat optimal, yang mampu meningkatkan kecepatan eksekusi sehingga mendukung perangkat lunak secara efisien [3]. Namun, dalam aplikasi yang digunakan untuk mengolah data secara real-time pada performa Java tidak seefisien C#, terutama dalam hal kecepatan waktu eksekusi. Kelebihan Java adalah stabilitasnya, yang menjadi daya tarik tersendiri bagi pengembang yang ingin memastikan aplikasi mereka tetap dapat diandalkan saat mengolah dataset besar maupun yang bersifat kompleks.

Di sisi lain, bahasa pemrograman C# menunjukkan performa yang bagus dalam pengembangan berbasis algoritma. C# sering digunakan dalam aplikasi mengelola data secara real-time dengan tingkat efisiensi yang tinggi [4]. Meskipun C# unggul dalam mengolah dataset kecil hingga sedang dalam hal kecepatan stabilitas performa C# dapat menjadi tantangan ketika mengolah dataset berskala besar. Hal ini dapat menjadi pertimbangan penting bagi pengembang perangkat lunak, terutama jika mereka membutuhkan performa yang konsisten pada data dengan volume yang besar.

Penelitian ini bertujuan untuk memberikan analisis komprehensif terhadap performa kedua bahasa pemrograman, yaitu Java dan C#, dengan memfokuskan pada perbandingan waktu eksekusi dan penggunaan memori menggunakan metode *benchmarking*. Penelitian ini memilih metode benchmarking karena mampu memberikan hasil yang baik dan objektif terkait performa kedua bahasa pemrograman dalam melakukan pencarian pada berbagai skala dataset. Hasil dari penelitian ini diharapkan dapat memberikan wawasan bagi pengembang perangkat lunak dalam memilih bahasa pemrograman yang sesuai, berdasarkan efisiensi waktu, performa efisiensi memori, dan aplikasi yang ingin dikembangkan. Dengan pemahaman yang mendalam tentang kedua bahasa pemrograman, baik dari segi kelebihan maupun kelemahannya, pengembang perangkat lunak dapat mempertimbangkan keputusan yang lebih terinformasi dan strategis sesuai dengan kebutuhan spesifik aplikasi mereka.

II. TINJAUAN PUSTAKA

Algoritma pencarian atau *searching* adalah suatu proses untuk menemukan data tertentu dalam sekumpulan data yang bertipe sama. Terdapat banyak algoritma pencarian yang dapat diimplementasikan ke sistem, seperti *sequential search, binary search*, dan sebagainya. Pada penelitian yang dilakukan oleh Nurul I. et al [5] yang membandingkan kinerja antara *sequential* dan *binary search* membuktikan bahwa *binary search* lebih cepat dalam melakukan pencarian kata. Meski begitu, hasil penelitian tersebut juga menunjukkan bahwa algoritma *sequential search* lebih efisien dalam penggunaan memori. Karena itu, kali ini penelitian dilakukan menggunakan algoritma *binary search* untuk membandingkan performa waktu eksekusi bahasa pemograman C# dan Java.

Penelitian yang dilakukan oleh M. I. D. Figueroa et al [6] membandingkan performa *image processing* dari C# dan Java. Penelitian tersebut mengukur waktu eksekusi kedua bahasa pemograman dalam mengolah data matriks berukuran 512x512 hingga 16834x16834. Hasil dari penelitian tersebut menunjukkan bahwa Java sedikit lebih cepat dalam mengolah ukuran matriks kecil, dari 512x512 hingga 1024x1024. Untuk ukuran menengah seperti 2048x2048, keduanya menunjukkan hasil waktu eksekusi yang hampir sama. Sementara untuk ukuran besar, dari 4096x4096 ke atas, C# menjadi jauh lebih efisien dari Java karena cara pengelolaan sumber daya oleh CLR lebih optimal untuk pengolahan berskala besar.

Penelitian berjudul "Perbandingan Subprogram pada Bahasa C dan Java" [7] berkesimpulan bahwa fitur *subprogram* yang dimiliki oleh Java lebih banyak dari C karena OOP (*Object Oriented Programming*) yang lebih kompleks. Namun, terdapat perbedaan fitur-fitur lain, misalnya *subprogram* di Java tidak terpengaruhi *subprogram* lain seperti di C. Java juga tidak memiliki tipe data *pointer* seperti bahasa pemograman C. Namun, penelitian tersebut tidak membandingkan efek fitur-fitur tersebut dalam performa waktu eksekusi kedua bahasa.

Menurut penelitian dari M. A. Jauhari et al [8], stabilitas eksekusi kode dan efisiensi waktu dapat dievaluasi juga menggunakan algoritma *sorting* seperti *bubble sort* dan *insertion sort*, yang sering digunakan dalam studi komparasi. Penelitian tersebut menemukan bahwa bahasa pemograman Go memiliki waktu eksekusi lebih cepat dari C#, terutama dalam kondisi data kompleks seperti *reversed data*. Untuk pengukuran 100 elemen menggunakan algoritma *bubble sort*, rata-rata waktu eksekusi Go adalah 31,42 ms, sedangkan C# membutuhkan 599,4 ms. Ini menunjukkan keunggulan Go dalam efisiensi eksekusi kode. Penelitian ini menjadi dasar penting untuk melanjutkan studi perbandingan efisiensi waktu antara bahasa pemograman lainnya, seperti C# dan Java yang juga memiliki karakteristik unik dalam pengelolaan memori dan eksekusi kode.

Penelitian yang dilakukan oleh N. G. C. Wicaksono et al [9] menunjukkan bahwa Java unggul dalam pemrosesan *string* dibandingkan Kotlin, dengan jumlah karakter yang dapat diproses lebih besar dalam waktu yang sama. Namun, efisiensi sumber daya seperti penggunaan memori dan CPU cenderung lebih tinggi dari

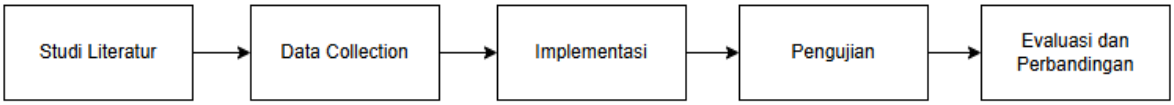
Kotlin. Ini menjadikan Java pilihan yang tepat untuk aplikasi yang membutuhkan performa tinggi dalam pengolahan data berbasis *string*.

Sebagai bahasa pemograman yang terkenal karena fleksibilitasnnya, Java tidak hanya berperforma tinggi dalam mengolah data *string*. Penelitian berjudul "Analisis Algoritma Bahasa Pemrograman Java dengan Bahasa Pemrograman Visual Basic Program Aplikasi Perkalian Matriks" [10] menunjukkan bahwa kecepatan eksekusi Java untuk program aplikasi perkalian matriks mencapai rata-rata 0,03342 detik dengan penggunaan memori sebesar 1672 KB. Meskipun hasilnya sedikit lebih lambat dibandingkan Visual Basic pada kasus serupa, Java tetap menjadi pilihan populer karena sifatnya yang universal dan mendukung pengembangan aplikasi berskala besar dengan sintaks yang konsisten. Hal ini menjadikan Java relevan untuk berbagai kebutuhan pemrograman yang memerlukan stabilitas dan kompatibilitas lintas platform.

III. METODE

Penelitian ini menggunakan metode *benchmarking* yang bertujuan untuk mengetahui mana yang memiliki performa lebih unggul dari kedua bahasa pemograman yang diuji. Penelitian ini dilakukan menggunakan salah satu algoritma *search* yang paling popular, yaitu *binary search*. Alur penelitian ini dimulai dengan studi literatur untuk mengulas hasil penelitian terdahulu sebagai landasan dari penelitian yang dilakukan [11].

Setelah mengumpulkan beberapa penelitian sebelumnya, dilakukan pengumpulan data sebagai bahan yang akan diolah oleh kedua bahasa pemograman. *Dataset* yang dikumpulkan terbagi menjadi tiga segmentasi, yaitu kecil, sedang, dan besar. Lalu, dilakukan implementasi, yaitu memulai pengolahan data yang telah dikumpulkan menggunakan bahasa C# dan Java. Setelah selesai, dilakukan pengujian untuk mengukur waktu eksekusi kedua bahasa pemograman tersebut. *Flowchart* alur penelitian yang dilaksanakan dapat dilihat pada Gambar 1:



Gambar 1 Metode Penelitian

Pengumpulan *dataset* dilakukan menggunakan *dummy* data yang berisi id, nama, dan alamat yang mengacu pada kota-kota besar di seluruh dunia. Terdapat tiga skala *dataset* yang dikumpulkan untuk diimplementasi. Ketiga *dataset* ini disimpan dalam format .txt menggunakan Notepad dengan jumlah data yang berbeda-beda. Total data di ketiga *dataset* ini dapat dilihat pada Tabel 1:

TABEL 1
UKURAN DATASET

Label	Jumlah
Kecil	1.000.000
Sedang	5.000.000
Besar	10.000.000

Selanjutnya, dilakukan pengolahan data atau implementasi data ke algoritma *binary search* menggunakan C# dan Java. Format file *dataset* .txt tadi di-*load* ke *source code* dan diolah menjadi data bertipe *list.* Setelah itu, peneliti membuat aplikasi *command line interface* (CLI) simple untuk menguji pencarian data.

Pencarian data dilakukan dua kali dengan pencarian semua data, dari segmen ID, nama, dan alamat. Pengukuran waktu eksekusi dilakukan menggunakan alat *benchmarking* seperti *System.Diagnostics.Stopwatch* untuk C# dan *System.nanoTime()* untuk Java. Lalu, waktu eksekusinya dikumpulkan dan dirata-rata untuk mendapatkan hasil pengukuran performa dari kedua bahasa pemrograman tersebut.

Sementara, pengukuran penggunaan memori (RAM) di bahasa pemograman C# menggunakan *class Process*, yaitu *WorkingSet64* dan *PrivateMemorySize64.* Kedua *function* tersebut akan menampilkan jumlah memori fisik dam memori *private* yang digunakan untuk menjalankan program dalam satuan *byte.* Di sisi lain, pengukuran penggunaan memori di Java menggunakan *runtime* yaitu *totalMemory()* untuk menampilkan total memori yang dialokasikan oleh JVM untuk program dan *freeMemory()* untuk menampilkan total memori yang tidak digunakan di JVM. Lalu, data yang dihasilkan oleh *function freeMemory()* dikurangkan dengan *totalMemory()* untuk mencari *usedMemory.*

Hasil dari pengujian lalu dianalisis untuk mengevaluasi perbedaan efisiensi waktu dan memori antara C# dan Java. Perbandingan dilakukan berdasarkan grafik yang menampilkan waktu eksekusi dan memori untuk setiap kali uji dalam kedua bahasa permograman. Grafik ini memvisualisasikan data sehingga memudahkan pembaca untuk memahami perbandingan secara kuantitatif.

IV. HASIL

Hasil dari penelitian ini adalah data perbandingan waktu eksekusi dan penggunaan memori dari bahasa pemograman Java dan C# yang dikategorikan berdasarkan ukuran dataset. Pengujian dilakukan dengan membagi dataset ke dalam tiga kategori: Data Kecil, Data Sedang, dan Data Besar, serta diukur pada dua pengujian berbeda untuk setiap kategori.

A. Hasil Pengujian Waktu Eksekusi dan Penggunaan Memori Pada Bahasa Pemograman C#

Pengujian dilakukan dengan frekuensi dua kali di setiap dataset. Hasil pengujian waktu eksekusi C# pada tiga *dataset* tersebut dapat dilihat pada tabel 2 berikut ini:

TABEL 2
HASIL PENGUJIAN WAKTU EKSEKUSI C# (DALAM DETIK)

Pengujian 1 Data Kecil	Pengujian 2 Data Kecil	Pengujian 1 Data Sedang	Pengujian 2 Data Sedang	Pengujian 1 Data Besar	Pengujian 2 Data Besar
0,122335	0,074571	0,412794	0,336001	0,788658	0,689677

Berdasarkan hasil pengujian waktu eksekusi C#, terlihat perbedaan performa untuk berbagai ukuran data dan frekuensi pengujian. Pada pengujian dengan data kecil, waktu eksekusi rata-rata untuk pengujian pertama adalah 0,123335 detik, sedangkan Pengujian 2 memiliki waktu yang lebih cepat yaitu 0,074571 detik. Ini menunjukkan bahwa pengujian kedua lebih efisien untuk data kecil dibandingkan pengujian pertama.

Untuk data sedang, pengujian pertama mencatat waktu eksekusi rata-rata sebesar 0,412794 detik, sedikit lebih lama dibandingkan pengujian kedua yang mencatat waktu 0,336001 detik. Hasil yang sama terlihat pada data besar, waktu eksekusi pengujian pertama tercatat sebesar 0,788658 detik, lebih lambat dibandingkan pengujian kedua yang memiliki waktu rata-rata 0,689677 detik.

Dari hasil tersebut, dapat disimpulkan bahwa pengujian pertama secara konsisten memberikan performa yang lebih baik dibandingkan pengujian kedua untuk semua ukuran data. Selain itu, peningkatan ukuran data berbanding lurus dengan peningkatan waktu eksekusi. Ini mengindikasikan adanya pengaruh signifikan dari kompleksitas data terhadap efisiensi eksekusi program di bahasa pemograman C#. Sementara, untuk hasil pengujian memori dapat dilihat pada tabel 3 berikut ini:

TABEL 3
HASIL PENGUJIAN PENGGUNAAN MEMORI C# (DALAM SATUAN BYTE)

Pengujian 1 Data Kecil	Pengujian 2 Data Kecil	Pengujian 1 Data Sedang	Pengujian 2 Data Sedang	Pengujian 1 Data Besar	Pengujian 2 Data Besar
436367360	436412416	2019069952	4025876480	4083156992	4047437824

Berdasarkan hasil pengujian penggunaan memori C#, terlihat bahwa konsumsi memori meningkat seiring dengan bertambahnya ukuran data. Untuk data kecil, pengujian 1 mencatat penggunaan memori sebesar 43.636.736 byte, sedangkan pengujian 2 sedikit lebih rendah dengan nilai 43.641.216 byte. Pada data sedang, terjadi peningkatan signifikan dengan pengujian 1 menggunakan 2.019.069.952 byte dan pengujian 2 mencatat nilai yang lebih besar, yaitu 4.025.876.480 byte. Untuk data besar, pengujian 1 menunjukkan penggunaan memori sebesar 4.083.156.992 byte, sementara pengujian 2 sedikit lebih rendah dengan 4.047.437.824 byte.

Hasil ini menunjukkan bahwa ukuran data sangat memengaruhi jumlah memori yang digunakan selama proses pengujian. Selain itu, terdapat sedikit variasi antara hasil pengujian 1 dan pengujian 2, yang kemungkinan disebabkan oleh perbedaan alokasi memori dinamis atau faktor internal sistem selama pengujian. Secara umum, tren peningkatan penggunaan memori terlihat konsisten dengan peningkatan ukuran data.

B. Hasil Pengujian Waktu Eksekusi dan Penggunaan Memori Pada Bahasa Pemograman Java

Pengujian dilakukan dengan metode yang sama dengan C#. Hasil pengujian waktu eksekusi dapat dilihat pada tabel 3 berikut ini:

TABEL 4
HASIL PENGUJIAN WAKTU EKSEKUSI JAVA (DALAM DETIK)

Pengujian 1 Data Kecil	Pengujian 2 Data Kecil	Pengujian 1 Data Sedang	Pengujian 2 Data Sedang	Pengujian 1 Data Besar	Pengujian 2 Data Besar
0,124598	0,218753	0,405421	0,380939	0,641134	0.754705

Berdasarkan hasil pengujian waktu eksekusi pada Java, terlihat bahwa waktu eksekusi meningkat seiring dengan bertambahnya ukuran data. Pada pengujian pertama untuk data kecil, waktu eksekusi tercatat sebesar 0,124598 detik, sedangkan pada pengujian 2 untuk data kecil meningkat menjadi 0,218753 detik. Untuk data sedang, pengujian pertama mencatat waktu eksekusi sebesar 0,405421 detik, sedikit lebih tinggi dibandingkan pengujian kedua dengan waktu eksekusi 0,380939 detik.

Pada data besar, waktu eksekusi untuk pengujian pertama adalah 0,641134 detik, sementara pengujian kedua menunjukkan nilai tertinggi dengan waktu eksekusi sebesar 0,754705 detik. Dari hasil ini, dapat disimpulkan bahwa ukuran data memengaruhi waktu eksekusi, dengan data yang lebih besar membutuhkan waktu lebih lama. Selain itu, terdapat variasi kecil dalam hasil pengujian antara pengujian 1 dan 2, yang mungkin disebabkan oleh faktor teknis atau kondisi sistem selama pengujian. Untuk hasil pengujian penggunaan memori dapat dilihat pada tabel 5 berikut:

TABEL 5
HASIL PENGUJIAN PENGGUNAAN MEMORI JAVA (DALAM SATUAN BYTE)

Pengujian 1 Data Kecil	Pengujian 2 Data Kecil	Pengujian 1 Data Sedang	Pengujian 2 Data Sedang	Pengujian 1 Data Besar	Pengujian 2 Data Besar
449839104	444596224	2044725248	2231369728	4083156992	3952523160

Hasil pengujian penggunakan memori terhadap bahasa Java menunjukkan bahwa pada data kecil, pengujian pertama menggunakan memori sebesar 44.983.910 byte, sedikit lebih tinggi dibandingkan pengujian kedua yang menggunakan 44.459.624 byte. Hal ini mengindikasikan adanya sedikit perbedaan dalam efisiensi memori di antara kedua pengujian, meskipun perbedaannya cukup kecil.

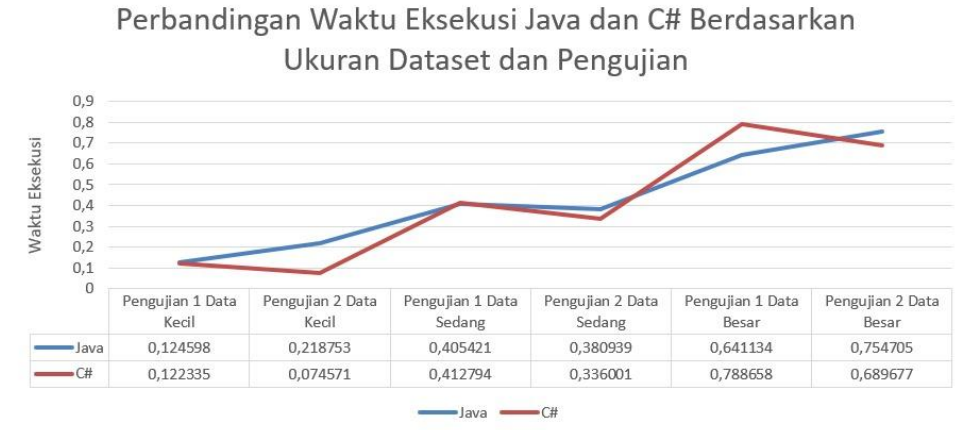
Untuk kategori data sedang, pengujian pertama menunjukkan penggunaan memori sebesar 204.472.548 byte, sementara pengujian kedua mencatat penggunaan sebesar 223.136.728 byte. Perbedaan ini menunjukkan bahwa pada data sedang, pengujian kedua cenderung memakan memori lebih besar dibandingkan pengujian pertama.

Pada kategori data besar, hasil menunjukkan bahwa pengujian pertama menggunakan memori sebesar 408.315.692 byte, yang lebih tinggi dibandingkan pengujian kedua dengan penggunaan memori sebesar 395.252.360 byte. Perbedaan ini menunjukkan bahwa pengujian pertama lebih memakan memori dibandingkan pengujian kedua untuk data besar.

Secara keseluruhan, hasil pengujian menunjukkan bahwa penggunaan memori cenderung meningkat seiring dengan ukuran data yang diuji. Selain itu, terdapat variasi kecil dalam hasil antara pengujian pertama dan kedua pada setiap kategori data, yang dapat diakibatkan oleh faktor seperti kondisi lingkungan *runtime* atau perbedaan kondisi *hardware* dalam proses pengolahan data.

V. PEMBAHASAN

Hasil penelitian yang disajikan menunjukkan variasi performa antara bahasa pemograman C# dan Java berdasarkan ukuran *dataset* dan frekuensi pengujian. Grafik perbandingan antara waktu eksekusi C# dan Java dapat dilihat pada Gambar 2:



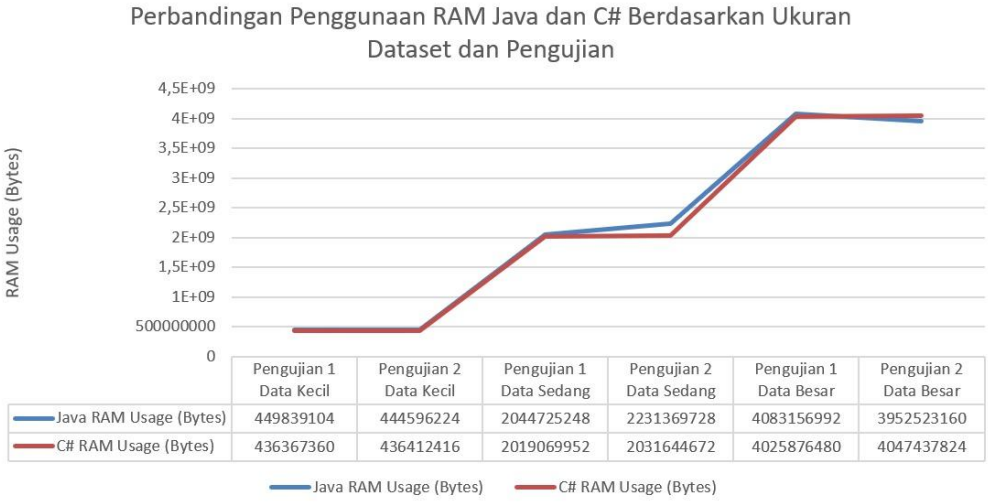
Gambar 2 Perbandingan Waktu Eksekusi Java dan C# Berdasarkan *Dataset* dan Pengujian

Baik pengujian pertama maupun kedua pada dataset kecil menunjukkan bahwa waktu eksekusi C# lebih cepat dibandingkan Java. Pada pengujian pertama, waktu eksekusi C# tercatat sebesar 0,122335 detik, sedangkan Java membutuhkan 0,142598 detik. Hal serupa terjadi pada pengujian kedua, di mana C# mencatat waktu 0,074571 detik, lebih rendah dibandingkan Java yang memerlukan 0,218753 detik.

Untuk dataset sedang, perbedaan waktu eksekusi menjadi lebih signifikan. Pada pengujian pertama, waktu eksekusi C# adalah 0,412794 detik, sementara Java mencatat waktu 0,405421 detik, menunjukkan performa yang hampir setara. Namun, pada pengujian kedua, waktu eksekusi C# (0,336001 detik) tetap lebih cepat dibandingkan Java (0,380399 detik).

Pada dataset besar, C# mempertahankan keunggulan performanya dibandingkan Java. Pada pengujian pertama, waktu eksekusi C# tercatat sebesar 0,788658 detik, sedangkan Java membutuhkan 0,641134 detik. Pengujian kedua juga menunjukkan hal yang sama, dengan C# mencatat waktu 0,689677 detik dan Java 0,754705.

Secara keseluruhan, C# menunjukkan performa waktu eksekusi yang lebih baik dibandingkan Java, terutama pada dataset kecil dan sedang. Namun, pada dataset besar, meskipun C# masih lebih cepat, selisih waktu antara kedua bahasa semakin kecil. Hal ini mengindikasikan bahwa Java memiliki skalabilitas yang baik untuk dataset yang lebih besar. Sementara untuk efisiensi memori, grafik hasil perbandingan dapat dilihat pada Gambar 3 berikut ini:



	Pengujian 1 Data Kecil	Pengujian 2 Data Kecil	Pengujian 1 Data Sedang	Pengujian 2 Data Sedang	Pengujian 1 Data Besar	Pengujian 2 Data Besar
Java RAM Usage (Bytes)	449839104	444596224	2044725248	2231369728	4083156992	3952523160
C# RAM Usage (Bytes)	436367360	436412416	2019069952	2031644672	4025876480	4047437824

Gambar 3 Perbandingan Penggunaan Memori (RAM) Java dan C# Berdasarkan *Dataset* dan Pengujian

Pada *dataset* kecil, C# menunjukkan keunggulan yang konsisten dibandingkan Java, demikian pula pada pengujian kedua. Ini dapat dikaitkan dengan efisiensi pengaturan memori internal dan *garbage collection* dalam *runtime environment* .NET yang digunakan oleh C# [12].Meski begitu, dalam penelitian sebelumnya, Java menunjukkan stabilitas *runtime* yang lebih baik, didukung oleh optimasi *Just-In-Time* (JIT) di *Java Virtual Machine* (JVM).

JVM memiliki mekanisme pengoptimalan runtime adaptif yang memungkinkan Java mempertahankan efisiensi pada ukuran *dataset* tertentu. Sementara itu, C# unggul dalam pengelolaan thread dan waktu eksekusi, terutama pada tugas dengan granularitas tinggi. Hal ini terkait erat dengan efisiensi platform .NET dan arsitektur task-centric pada runtime-nya [13].

Pada *dataset* sedang, hasil penelitian menunjukkan bahwa performa kedua bahasa relatif seimbang. Ini menunjukkan bahwa untuk ukuran data yang lebih besar, efisiensi *runtime* Java dan C# mendekati keseimbangan. Perbedaan kecil dalam hasil pengujian dapat disebabkan oleh variasi implementasi algoritma atau faktor-faktor lain, seperti efisiensi pengaturan *multi-threading* dan optimisasi *compiler* masing-masing platform [14].

Pada *dataset* besar, hasil pengujian menunjukkan pola yang lebih fluktuatif. Pada pengujian pertama, Java lebih unggul dibandingkan C#. Sedangkan pada pengujian kedua, C# kembali memimpin. Fluktuasi ini dapat dipengaruhi dari pengelolaan memori *heap*, optimasi *runtime*, atau cara kedua *platform* menangani alokasi memori untuk proses skala besar. JVM sering kali dioptimalkan untuk performa jangka panjang dengan teknik seperti *Just-In-Time (JIT) compilation* yang bekerja lebih baik pada aplikasi yang berjalan dalam durasi lama. Sementara itu, *runtime* .NET MVC 5 dapat memanfaatkan optimasi berbasis hardware yang spesifik untuk lingkungan Windows [15]. Ini memberikan keunggulan pada bahasa C# dalam pengujian dengan scenario tertentu.

Hasil penelitian ini sejalan dengan penelitian sebelumnya, seperti yang dilakukan oleh M. I. D. Figueroa et al. [6], yang menunjukkan bahwa C# lebih efisien dalam mengolah data berukuran besar. Sedangkan, Java sedikit lebih unggul pada ukuran matriks kecil hingga menengah. Meskipun penelitian tersebut hanya menguji kedua bahasa pemograman dengan frekuensi yang hanya satu kali *run*, hasil dari penelitian tersebut masih sejalan dengan penelitian ini.

Sementara itu, penelitian oleh N. G. C. Wicaksono et al. [9] juga mendukung temuan ini dengan menunjukkan keunggulan Java dalam performa pemrosesan *string*, meskipun dalam konteks yang berbeda. Penelitian ini menegaskan bahwa pemilihan bahasa pemrograman sangat bergantung pada kebutuhan spesifik aplikasi dan karakteristik *dataset* yang akan diolah.

VI. KESIMPULAN

Penelitian ini berhasil membandingkan waktu eksekusi dan penggunaan memori antara dua bahasa pemrograman, yaitu Java dan C, menggunakan metode *benchmarking* dengan tiga kategori *dataset*. Hasil pengujian menunjukkan bahwa C# unggul dalam kecepatan waktu eksekusi pada *dataset* kecil dan sedang dibandingkan dengan Java. Sementara itu, Java memberikan performa yang lebih stabil pada *dataset* besar.

Keunggulan C# pada dataset kecil dan sedang dapat dijelaskan oleh efisiensi pengelolaan memori dan pengaturan thread yang lebih baik pada platform .NET. Sementara itu, Java menunjukkan performa yang lebih stabil pada dataset besar berkat optimasi *Just-In-Time (JIT)* pada *Java Virtual Machine* (JVM), yang memungkinkan peningkatan performa seiring waktu.

Keterbatasan dalam penelitian ini terletak pada fokus pengukuran waktu eksekusi dan memori tanpa mempertimbangkan penggunaan CPU. Untuk penelitian selanjutnya, disarankan untuk memberikan analisis yang lebih komprehensif, termasuk evaluasi penggunaan CPU. Selain itu, pengujian dengan pencarian lainnya dapat dilakukan untuk mengevaluasi performa kedua bahasa pemrograman secara menyeluruh.

DAFTAR PUSTAKA

[1] M. Khusien Bagaskoro, M. A. Chakim, M. N. Hilal, and O. Thowimma, "BENCHMARKING METODE RANCANG BANGUN WATERFALL DAN PEMODELAN BERBASIS OBJEK," vol. 15, no. 2, 2021, doi: 10.47111/JTI.

[2] M. Cahyanti and M. Lamsani, "PERANCANGAN SISTEM INFORMASI JASA LAYANAN PENCUCIAN KENDARAAN BERMOTOR," *Sebatik*, vol. 25, no. 2, pp. 639–648, Dec. 2021, doi: 10.46984/sebatik.v25i2.1530.

[3] Yuliana Monika Liwu, Kristianus Jago Tute, and Benediktus Yoseph Bhae, "Perancangan dan Implementasi Sistem Informasi Manajemen Panti Asuhan Menggunakan Pemrograman Java Netbeans," *SATESI: Jurnal Sains Teknologi dan Sistem Informasi*, vol. 2, no. 2, pp. 108–116, Oct. 2022, doi: 10.54259/satesi.v2i2.1126.

[4] R. Selviana and D. Bastha Fiastat Lugata, "Implementasi Generate Map dan Pemunculan Objek secara Acak pada Game 3D menggunakan Bahasa C# dan Metode Perlin Noise di Unity: Studi Kasus pada Game Development," 2023.

[5] N. Imamah and M. I. Bahari, "PERBANDINGAN ALGORITMA SEQUENTIAL SEARCH DAN ALGORITMA BINARY SEARCH PADA APLIKASI KAMUS BAHASA INDONESIA MENGGUNAKAN PHP DAN JQUERY," *Jurnal Informatika-COMPUTING*, vol. 08, no. 1, pp. 1–6, Jun. 2021.

[6] M. Isabel, D. Figueroa, and D. Rodríguez, "Image Processing using Java and C#: A Comparison Approach," 2014.

[7] R. Selamet, "Perbandingan Subprogram Pada Bahasa C dan Java," *Jurnal Media Informatika*, vol. 17, no. 2, pp. 1–4, 2018.

[8] M. A. Jauhari, D. Hamidin, and M. Rahmatuloh, "Komparasi Stabilitas Eksekusi Kode Bahasa Pemrogrman .Net C# Versi 4.0.3019 Dengan Google Golang Versi 1.4.2 Menggunakan Algoritma Bubble Sort dan Insertion Sort," 2017.

[9] N. Galih, C. Wicaksono, and H. Leong, "PERBANDINGAN PERFORMA BAHASA PEMROGRAMAN JAVA DAN KOTLIN PADA ANDROID APPS," *Jurnal Proxies*, vol. 4, no. 2, p. 2021, 2021.

[10] F. Shodik, "Analisis Algoritma Bahasa Pemrograman Java dengan Bahasa Pemrograman Visual Basic Program Aplikasi Perkalian Matriks," *Journal Of Mathematics UNP*, vol. 6, no. 3, pp. 1–6, 2021.

[11] N. Firdha, Damira, R. Fitri, G. Hijrah Selaras, and I. G. N. Saputra, "Studi Literatur Tentang Peningkatan Kompetensi Belajar Peserta Didik Melalui Kegiatan Pembelajaran Kolaboratif Berbasis Lesson Study," *Universitas Negeri Padang*, vol. 1, 2021.

[12] Kartini, "MEMBANGUN APLIKASI KASIR BERBASIS ORACLE (STUDY KASUS TOKO MEGA CELLULAR BEKASI)," *Jurnal Ilmu Komputer*, vol. 11, no. 2, pp. 1–9, Sep. 2015.

[13] J. Ogala, B. Ogala, and J. Onyarin, "COMPARATIVE ANALYSIS OF C, C++, C# AND JAVA PROGRAMMING LANGUAGES," 2020, [Online]. Available: www.globalscientificjournal.com

[14] K. Sari *et al.*, "Pengoptimalan Algoritma Genetika dengan Multithreading untuk Pencarian Kalimat Optimizing Genetic Algorithms with Multithreading for Sentence Retrieval," vol. 14, no. 1, 2024, doi: 10.30700/sisfotenika.v14i1.410.

[15] E. Christanto and T. Wibowo, "ANALISIS KOMPARASI PERFORMA WEB APPLICATION: STUDI KASUS ASP.NET MVC DAN ASP.NET CORE," *Jurnal UIB*, vol. 1, no. 1, pp. 1–8, Aug. 2020, [Online]. Available: http://journal.uib.ac.id/index.php/cbssit