Journal : Journal of Computer Science and Information Technology (JCSIT)
Publisher : Yayasan Nuraini Ibrahim Mandiri
ISSN : 3031-8467
Volume / Issue / Year/ Page : Vol. 3 No. 4 (2026): September/ Hal 334-352

IMPLEMENTASI MEKANISME *QUEUE* DAN *WORKER* UNTUK OPTIMASI UTILISASI GPU DALAM PEMROSESAN LAYANAN AI PADA *BACKEND*

Syahrul Ramadhan[1]*, Muhyiddin AM Hayat[2], Rizki Yusliana Bakti[3], Titin Wahyuni[4], Fahrim Irhamna Rachman[5], Darniati[6]
[1,2,3,4,6,] Universitas Muhammadiyah Makassar, Indonesia
[1]105841113722@student.unismuh.ac.id , [2]muhyiddin@unismuh.ac.id,
[3]rizkiyusliana@unismuh.ac.id, [4]titinwahyuni@unismuh.ac.id,
[5]fachrim141020@unismuh.ac.id, [6]darniati@unismuh.ac.id
* Coresponding Author

Info Article

Submission : 08-08- 2026
Review : 11-08-2026
Revised : 28-08-2026
Accepted : 02-09-2006
Published : 13-09-2006

Keywords:
Artificial Intelligence, Worker Pool, Redis, vLLM, Utilisasi GPU

Abstrak

Pengembangan layanan Kecerdasan Buatan (AI) berbasis Large Language Models (LLM) telah meningkatkan kebutuhan terhadap sistem backend yang mampu menangani proses inferensi secara efisien. Proses inferensi yang dijalankan pada Graphics Processing Unit (GPU) membutuhkan komputasi intensif sehingga berpotensi menimbulkan hambatan (bottleneck) dan pemanfaatan GPU yang tidak optimal. Penelitian ini bertujuan untuk mengimplementasikan mekanisme queue dan worker pada backend layanan AI berbasis GPU, menganalisis pengaruh variasi jumlah worker terhadap kinerja sistem dan penggunaan GPU, serta menentukan konfigurasi worker yang optimal. Sistem dikembangkan menggunakan bahasa pemrograman Go dengan layered architecture, Redis sebagai message queue, Asynq sebagai pengelola queue dan worker, serta vLLM sebagai layanan inferensi AI berbasis GPU. Pengujian dilakukan menggunakan K6 dengan beban 50 virtual users selama 60 detik pada konfigurasi 1, 2, 4, 8, dan 16 worker. Parameter yang dianalisis meliputi response time, throughput, error rate, GPU utilization, dan GPU idle time. Hasil penelitian menunjukkan bahwa peningkatan jumlah worker meningkatkan throughput dari 26,60 menjadi 142,91 requests/s, menurunkan response time dari 1.904 menjadi 343,54 ms, serta meningkatkan utilisasi GPU dari 80,21% hingga mencapai 100%. Seluruh konfigurasi menghasilkan error rate sebesar 0%, sedangkan GPU idle time menurun dari 18,3% menjadi 0%. Temuan penelitian menunjukkan adanya pola saturasi GPU akibat peningkatan concurrency, di mana penambahan worker hingga 8 worker mampu meningkatkan pemanfaatan GPU secara optimal, sedangkan penambahan menjadi 16 worker tidak memberikan peningkatan response time yang berarti. Berdasarkan keseluruhan parameter, konfigurasi 8 worker merupakan konfigurasi paling optimal pada lingkungan pengujian penelitian ini.

How to Cite : Ramadhan, S., Hayat, M. A., Bakti, R. Y., Wahyuni, T., Rachman, F. I., Darniati (2026). Implementasi Mekanisme Queue dar Worker untuk Optimasi Utilisasi GPU dalam Pemrosesan Layanan AI pada Backend. Journal of Computer Science anc Information Technology (JCSIT)**,** *3*(4),334–352.

DOI : 10.70248/jcsit.v3i4.4439

PENDAHULUAN

Perkembangan Artificial Intelligence (AI) dalam beberapa tahun terakhir mengalami peningkatan yang sangat pesat, terutama sejak hadirnya Large Language Model (LLM) berbasis arsitektur Transformer yang mampu meningkatkan kemampuan pemahaman dan generasi bahasa secara signifikan. Arsitektur Transformer menjadi salah satu dasar penting dalam perkembangan berbagai model bahasa modern (Vaswani et al., 2023). Perkembangan foundation models kemudian mendorong pemanfaatan LLM pada

Journal : Journal of Computer Science and Information Technology (JCSIT)
Publisher : Yayasan Nuraini Ibrahim Mandiri
ISSN : 3031-8467
Volume / Issue / Year/ Page : Vol. 3 No. 4 (2026): September/ Hal 334-352

berbagai bidang dan aplikasi melalui integrasi Application Programming Interface (API), sehingga layanan AI dapat digunakan tanpa harus melakukan pelatihan model secara mandiri (Bommasani et al., 2022).

Peningkatan adopsi LLM turut meningkatkan kebutuhan terhadap sistem backend yang mampu menangani permintaan inferensi secara efisien. Berbeda dengan aplikasi web konvensional, proses inferensi LLM membutuhkan komputasi paralel yang intensif pada Graphics Processing Unit (GPU). GPU memiliki kemampuan massively parallel processing yang sesuai untuk beban komputasi AI, tetapi performa sistem tetap dipengaruhi oleh keterbatasan sumber daya dan pola distribusi beban kerja (Mittal & Vetter, 2015; Owens et al., 2008). Ketika jumlah permintaan meningkat secara bersamaan, keterbatasan tersebut dapat menyebabkan bottleneck, peningkatan latency, serta penurunan efisiensi pemanfaatan sumber daya.

Salah satu pendekatan untuk mengatasi permasalahan tersebut adalah penerapan pemrosesan asinkron menggunakan message queue dan worker pool. Mekanisme queue memungkinkan proses penerimaan permintaan dipisahkan dari proses eksekusi sehingga pekerjaan dapat diproses secara paralel oleh sejumlah worker. Pendekatan ini juga berkaitan dengan permasalahan tail latency pada sistem berskala besar, di mana peningkatan beban dan konkurensi dapat menyebabkan variasi waktu respons yang semakin besar (Dean & Barroso, 2013). Dengan memisahkan penerimaan permintaan dan proses komputasi, sistem dapat mengatur beban kerja sebelum diteruskan ke sumber daya komputasi yang terbatas.

Namun, pada layanan LLM modern seperti vLLM, optimasi pemrosesan tidak hanya dilakukan pada tingkat backend. vLLM menggunakan PagedAttention untuk mengelola Key-Value (KV) Cache secara lebih efisien dan mendukung pemrosesan banyak permintaan secara bersamaan melalui mekanisme continuous batching. PagedAttention dirancang untuk mengurangi pemborosan memori akibat fragmentasi dan memungkinkan penggunaan memori KV Cache secara lebih fleksibel pada proses LLM serving (Kwon et al., 2023). Dengan pendekatan tersebut, vLLM mampu meningkatkan throughput layanan LLM dibandingkan beberapa sistem serving sebelumnya pada tingkat latency yang sebanding (Kwon et al., 2023).

Meskipun vLLM telah memiliki mekanisme penjadwalan dan batching internal, mekanisme tersebut beroperasi pada tingkat inference engine setelah permintaan diterima oleh layanan vLLM. Sementara itu, queue pada tingkat backend memiliki fungsi yang berbeda, yaitu mengatur penerimaan dan distribusi pekerjaan sebelum permintaan diteruskan ke inference engine. Dengan demikian, backend queue dapat berfungsi sebagai lapisan pengendali beban yang memisahkan request handling dari eksekusi inferensi, mengatur tingkat konkurensi worker, serta membatasi jumlah pekerjaan yang secara bersamaan diteruskan menuju vLLM. Interaksi kedua mekanisme tersebut menjadi penting karena peningkatan konkurensi pada tingkat backend belum tentu menghasilkan peningkatan performa apabila kapasitas komputasi GPU telah mencapai kondisi jenuh.

Penelitian mengenai pemanfaatan GPU sebelumnya telah banyak membahas efisiensi komputasi, paralelisme, dan pengelolaan sumber daya GPU (Hong & Kim, 2009; Mittal & Vetter, 2015). Sementara itu, penelitian Kwon et al. (2023) berfokus pada efisiensi manajemen memori KV Cache dan peningkatan throughput pada sistem LLM serving melalui PagedAttention. Fokus tersebut berbeda dengan penelitian ini yang menempatkan mekanisme queue dan variasi jumlah worker pada tingkat backend sebagai variabel utama yang dievaluasi terhadap performa sistem dan pemanfaatan

Journal : Journal of Computer Science and Information Technology (JCSIT)
Publisher : Yayasan Nuraini Ibrahim Mandiri
ISSN : 3031-8467
Volume / Issue / Year/ Page : Vol. 3 No. 4 (2026): September/ Hal 334-352

GPU. Dengan demikian, research gap penelitian ini terletak pada evaluasi empiris mengenai bagaimana variasi jumlah worker pada backend asynchronous queue berinteraksi dengan kapasitas inference engine vLLM, khususnya dalam menghasilkan keseimbangan antara response time, throughput, utilisasi GPU, dan GPU idle time.

Berdasarkan kesenjangan tersebut, penelitian ini mengimplementasikan mekanisme Redis Queue dan Asynq Worker Pool pada sistem backend layanan AI berbasis GPU. Pengujian dilakukan dengan variasi jumlah worker untuk menganalisis pengaruh tingkat konkurensi terhadap response time, throughput, error rate, utilisasi GPU, dan GPU idle time. Hasil pengujian kemudian digunakan untuk menentukan konfigurasi worker yang paling optimal dalam menyeimbangkan performa sistem dan efisiensi pemanfaatan GPU pada layanan inferensi LLM berbasis vLLM.

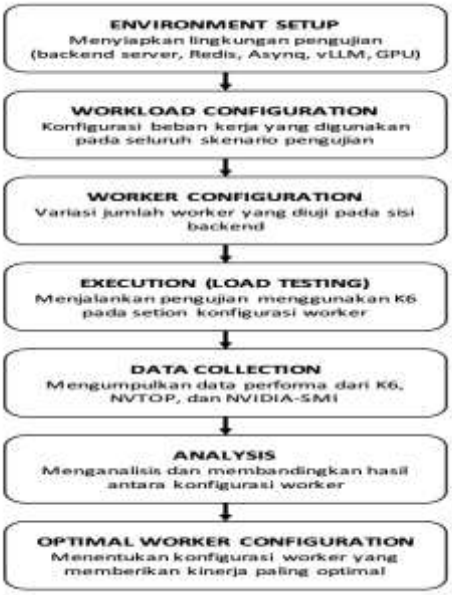
METODE PENELITIAN

Metode penelitian ini menggunakan pendekatan eksperimen komparatif untuk menganalisis pengaruh variasi jumlah worker terhadap kinerja sistem backend layanan Artificial Intelligence berbasis GPU. Penelitian diawali dengan perancangan dan implementasi sistem menggunakan layered architecture, Redis Queue, dan worker pool berbasis Asynq. Pemanfaatan message queue memungkinkan komunikasi antar komponen perangkat lunak dilakukan secara asynchronous melalui media antrean bersama (Maharjan et al., 2023). Selanjutnya dilakukan pengujian pada lima konfigurasi worker (1, 2, 4, 8, dan 16 worker) menggunakan beban kerja yang sama, yaitu 50 virtual users selama 60 detik. Data hasil pengujian dianalisis berdasarkan parameter response time, throughput, error rate, utilisasi GPU, dan GPU idle time untuk mengevaluasi pengaruh jumlah worker serta menentukan konfigurasi yang paling optimal dalam meningkatkan kinerja sistem backend dan efisiensi pemanfaatan GPU (Mittal & Vetter, 2015).

1. Perencanaan Sistem

Perencanaan sistem pada penelitian ini dilakukan untuk menyiapkan lingkungan eksperimen dan alur pengujian dalam menganalisis pengaruh variasi jumlah worker terhadap kinerja sistem backend layanan Artificial Intelligence berbasis GPU. Perencanaan mencakup penyiapan lingkungan sistem, penetapan workload, konfigurasi jumlah worker, pelaksanaan load testing, pengumpulan data kinerja sistem dan GPU, serta analisis hasil pengujian untuk menentukan konfigurasi worker yang paling optimal.

Journal : Journal of Computer Science and Information Technology (JCSIT)
Publisher : Yayasan Nuraini Ibrahim Mandiri
ISSN : 3031-8467
Volume / Issue / Year/ Page : Vol. 3 No. 4 (2026): September/ Hal 334-352



GAMBAR 1. *Experimental Setup* dan *Execution Pipeline*

Diagram pada **Gambar 1** menunjukkan Experimental Setup and Execution Pipeline yang digunakan dalam penelitian. Proses dimulai dari tahap Environment Setup, yaitu menyiapkan seluruh komponen lingkungan pengujian yang terdiri atas backend server, Redis, Asynq, layanan worker, layanan inferensi vLLM, serta GPU sebagai sumber daya komputasi. Penggunaan GPU sebagai sumber daya komputasi untuk beban kerja yang membutuhkan pemrosesan paralel telah banyak diterapkan pada sistem komputasi berkinerja tinggi (Owens et al., 2008). Seluruh komponen sistem dikonfigurasi secara konsisten pada setiap skenario pengujian agar perbedaan hasil yang diperoleh dapat dikaitkan dengan variasi jumlah worker.

Tahap berikutnya adalah Workload Configuration, yaitu menetapkan beban kerja yang digunakan pada seluruh skenario pengujian. Beban kerja dibuat sama untuk setiap konfigurasi agar hasil pengujian dapat dibandingkan secara objektif. Pada penelitian ini, pengujian dilakukan menggunakan K6 dengan beban 50 virtual users selama 60 detik. Selain itu, jumlah pengguna dan durasi pengujian dipertahankan agar skenario pengujian tetap konsisten.

Setelah workload ditetapkan, dilakukan Worker Configuration dengan menentukan jumlah worker yang akan diuji pada sisi backend. Variasi konfigurasi yang digunakan terdiri atas 1, 2, 4, 8, dan 16 worker. Variasi tersebut digunakan untuk mengatur tingkat concurrency dalam pemrosesan permintaan layanan AI. Pengaturan jumlah proses atau pekerja yang berjalan secara paralel merupakan salah satu aspek yang berkaitan dengan pemanfaatan sumber daya pada sistem komputasi heterogen (Mittal & Vetter, 2015).

Tahap selanjutnya adalah Execution (Load Testing). Pada tahap ini, setiap konfigurasi worker dijalankan menggunakan K6 dengan skenario dan workload yang sama. Permintaan yang diterima oleh backend diproses melalui mekanisme Redis Queue dan worker pool Asynq secara asinkron. Mekanisme message queue memungkinkan tugas ditempatkan pada antrean sehingga proses pengirim dan penerima dapat bekerja secara lebih independen (Maharjan et al., 2023). Worker mengambil job dari antrean dan meneruskan ke layanan vLLM untuk menjalankan proses inferensi menggunakan GPU. Pendekatan pemrosesan asynchronous melalui background job dan message

Journal : Journal of Computer Science and Information Technology (JCSIT)
Publisher : Yayasan Nuraini Ibrahim Mandiri
ISSN : 3031-8467
Volume / Issue / Year/ Page : Vol. 3 No. 4 (2026): September/ Hal 334-352

queue dapat digunakan untuk memisahkan pekerjaan komputasi dari proses utama aplikasi (Topalidi, n.d.).

Setelah proses pengujian selesai, dilakukan Data Collection untuk mengumpulkan data dari sistem backend dan GPU. Data kinerja diperoleh dari hasil pengujian K6, meliputi response time, throughput, dan error rate. Sementara itu, pemantauan GPU dilakukan menggunakan NVIDIA-SMI dan NVTOP untuk memperoleh nilai GPU utilization dan GPU idle time. Data tersebut kemudian digunakan sebagai dasar untuk mengevaluasi hubungan antara jumlah worker, kinerja sistem, dan pemanfaatan sumber daya GPU.

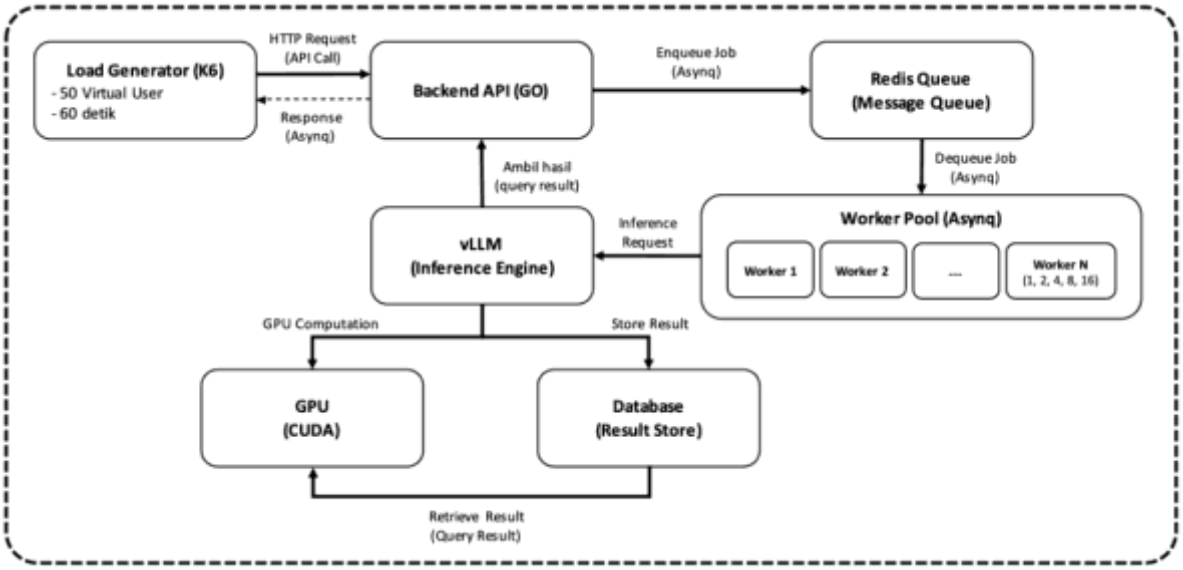
Tahap berikutnya adalah Analysis, yaitu menganalisis dan membandingkan hasil dari seluruh konfigurasi worker. Analisis dilakukan berdasarkan lima parameter utama, yaitu response time, throughput, error rate, GPU utilization, dan GPU idle time. Response time digunakan untuk mengetahui kecepatan sistem dalam memberikan respons, throughput untuk mengetahui kemampuan sistem dalam memproses permintaan, error rate digunakan untuk mengukur tingkat keberhasilan pemrosesan permintaan, sementara GPU utilization dan GPU idle time digunakan untuk mengetahui tingkat pemanfaatan sumber daya GPU. Evaluasi kinerja sistem yang melibatkan CPU dan GPU perlu mempertimbangkan pemanfaatan sumber daya serta keseimbangan beban kerja pada perangkat komputasi (Mittal & Vetter, 2015).

Tahap terakhir adalah Optimal Worker Configuration, yaitu menentukan konfigurasi worker yang memberikan kinerja paling optimal berdasarkan hasil perbandingan seluruh parameter pengujian. Penentuan konfigurasi optimal mempertimbangkan keseimbangan antara response time yang rendah, throughput yang tinggi, error rate yang rendah, GPU utilization yang tinggi, dan GPU idle time yang rendah. Dengan demikian, konfigurasi yang dipilih tidak hanya berdasarkan satu parameter kinerja, tetapi berdasarkan keseluruhan hasil pengujian dan keseimbangan antara performa backend dengan efisiensi pemanfaatan GPU.

2. Perencanaan Desain Sistem

Perancangan desain sistem dilakukan untuk menggambarkan arsitektur dan alur eksekusi sistem backend dalam memproses permintaan layanan Artificial Intelligence berbasis GPU menggunakan mekanisme queue dan worker. Sistem dirancang dengan memisahkan proses penerimaan permintaan, pengelolaan antrean, pemrosesan tugas, proses inferensi, dan penyimpanan hasil sehingga pemrosesan dapat dilakukan secara asinkron serta memungkinkan pengaturan tingkat concurrency melalui variasi jumlah worker. Pemisahan proses melalui mekanisme message queue dapat mendukung komunikasi asynchronous dan memungkinkan komponen sistem bekerja secara lebih independen (Maharjan et al., 2023).

Journal : Journal of Computer Science and Information Technology (JCSIT)
Publisher : Yayasan Nuraini Ibrahim Mandiri
ISSN : 3031-8467
Volume / Issue / Year/ Page : Vol. 3 No. 4 (2026): September/ Hal 334-352



GAMBAR 2. Arsitektur Sistem dan Alur Eksekusi

Sistem backend yang dirancang terdiri atas tujuh komponen utama, yaitu Load Generator (K6), Backend API (Go), Redis Queue, Worker Pool (Asynq), vLLM sebagai inference engine, GPU, dan Database. Penggunaan GPU sebagai perangkat komputasi paralel sesuai untuk beban kerja yang membutuhkan kemampuan pemrosesan komputasi secara intensif (Owens et al., 2008).

a. Load Generator (K6) mengirimkan permintaan layanan AI kepada Backend API melalui HTTP request. Pada proses pengujian, K6 digunakan dengan beban 50 virtual users selama 60 detik. Setelah permintaan diproses, hasil atau respons dikembalikan kepada Load Generator melalui Backend API.
b. Backend API (Go) menerima permintaan dari Load Generator dan melakukan pengelolaan terhadap permintaan tersebut. Permintaan yang akan diproses kemudian dimasukkan ke dalam Redis Queue sebagai job menggunakan mekanisme enqueue melalui Asynq. Penggunaan antrean memungkinkan tugas ditampung sebelum diproses oleh komponen pekerja dan mendukung komunikasi asynchronous antar komponen sistem (Maharjan et al., 2023).
c. Redis Queue berfungsi sebagai media penyimpanan sementara job yang menunggu untuk diproses. Setiap job yang masuk ke dalam antrean akan menunggu hingga tersedia worker yang dapat memprosesnya. Dengan demikian, proses penerimaan permintaan pada Backend API dapat dipisahkan dari proses pemrosesan komputasi yang membutuhkan sumber daya GPU (Topalidi, n.d.).
d. Worker Pool (Asynq) mengambil job dari Redis Queue melalui mekanisme dequeue. Worker Pool terdiri atas sejumlah worker yang jumlahnya divariasikan dalam penelitian, yaitu 1, 2, 4, 8, dan 16 worker. Setiap worker mengambil job yang tersedia dan mengirimkan request ke vLLM. Penggunaan beberapa pekerja untuk menangani tugas secara asynchronous memungkinkan pemrosesan beberapa pekerjaan secara paralel (Topalidi, n.d.).
e. vLLM (Inference Engine) menerima inference request dari worker dan menjalankan proses inferensi menggunakan model LLM. Proses komputasi inferensi memanfaatkan GPU sebagai sumber daya komputasi utama. Penggunaan GPU untuk pemrosesan paralel dapat meningkatkan kemampuan sistem dalam menangani beban komputasi yang tinggi (Owens et al., 2008).
f. GPU digunakan sebagai sumber daya utama untuk menjalankan proses inferensi

Journal : Journal of Computer Science and Information Technology (JCSIT)
Publisher : Yayasan Nuraini Ibrahim Mandiri
ISSN : 3031-8467
Volume / Issue / Year/ Page : Vol. 3 No. 4 (2026): September/ Hal 334-352

model AI melalui vLLM. Selama proses pengujian, pemanfaatan GPU dipantau untuk mengetahui tingkat penggunaan sumber daya GPU pada setiap konfigurasi worker. Pemanfaatan GPU dan pengelolaan beban kerja pada lingkungan CPU-GPU merupakan aspek penting dalam evaluasi sistem komputasi heterogen (Mittal & Vetter, 2015).

g. Database digunakan sebagai result store untuk menyimpan hasil pemrosesan yang telah dilakukan oleh layanan inferensi. Hasil tersebut kemudian dapat diambil kembali oleh Backend API melalui proses query result dan diberikan kepada pengguna sebagai hasil permintaan.

Secara keseluruhan, mekanisme queue dan worker pada sistem bekerja dengan memisahkan proses penerimaan permintaan dari proses komputasi inferensi. Backend API bertugas menerima permintaan dan memasukkannya ke Redis Queue, sedangkan Worker Pool bertugas mengambil dan memproses job secara paralel sesuai jumlah worker yang dikonfigurasi. Selanjutnya, worker meneruskan permintaan ke vLLM untuk diproses menggunakan GPU, kemudian hasil pemrosesan disimpan pada database. Hasil tersebut dapat diambil kembali melalui Backend API sehingga alur komunikasi antara komponen sistem terstruktur dan proses komputasi dapat dikendalikan berdasarkan tingkat concurrency yang digunakan. Pendekatan message queue dan background processing dapat digunakan untuk memisahkan pekerjaan komputasi dari proses utama aplikasi sehingga komponen sistem dapat bekerja secara asynchronous (Maharjan et al., 2023; Topalidi, n.d.).

3. Lingkungan Pengujian

Pengujian penelitian dilaksanakan pada lingkungan server GPU Universitas Muhammadiyah Makassar sebagai infrastruktur utama untuk menjalankan sistem backend, layanan queue-worker, serta proses inferensi model Artificial Intelligence. Penggunaan GPU sebagai perangkat komputasi utama sesuai untuk pemrosesan yang membutuhkan kemampuan paralel dan komputasi intensif (Owens et al., 2008).

TABEL 1. Spesifikasi Lingkungan Pengujian

Komponen	Spesifikasi
Server	Server GPU Universitas Muhammadiyah Makassar
CPU	Intel(R) Xeon(R) Gold 5418Y, 64 Core, 64 thread
RAM	251 GiB
GPU	NVIDIA L40S
VRAM	46,068 MiB
Operating System	Ubuntu 22.04.5 LTS
CUDA	V13.2.78
Go	go1.26.4
vLLM	0.23.0
Model	meta-llama/Llama-2-7b-chat-hf
Model Size	7B parameters
Tensor Parallel Size	1
GPU Memory Utilization	0,85
Load Testing	K6
Virtual Users	50 User
Durasi	60 Detik

Berdasarkan **Tabel 1**, server yang digunakan memiliki prosesor Intel(R) Xeon(R)

Journal : Journal of Computer Science and Information Technology (JCSIT)
Publisher : Yayasan Nuraini Ibrahim Mandiri
ISSN : 3031-8467
Volume / Issue / Year/ Page : Vol. 3 No. 4 (2026): September/ Hal 334-352

Gold 5418Y dengan 64 core dan 64 thread, RAM sebesar 251 GiB, serta GPU NVIDIA L40S dengan VRAM sebesar 46.068 MiB. Sistem operasi yang digunakan adalah Ubuntu 22.04.5 LTS dengan CUDA versi 13.2.78 dan bahasa pemrograman Go versi 1.26.4. Lingkungan tersebut digunakan untuk menjalankan sistem backend mekanisme queue dan worker dalam memproses permintaan layanan AI secara asinkron. Pemanfaatan CPU dan GPU secara bersamaan merupakan karakteristik umum dalam sistem komputasi heterogen yang bertujuan memanfaatkan karakteristik masing-masing perangkat untuk memperoleh kinerja yang lebih baik (Mittal & Vetter, 2015).

Pada sisi layanan inference, penelitian menggunakan vLLM versi 0.23.0 dengan model meta-llama/Llama-2-7b-chat-hf berukuran 7B parameter dan tensor parallel size sebesar 1. Penggunaan GPU sebagai sumber daya komputasi untuk menjalankan model AI memungkinkan proses komputasi paralel dilakukan pada perangkat yang memiliki kemampuan pemrosesan tinggi (Owens et al., 2008). Pengujian dilakukan menggunakan K6 dengan beban 50 virtual users selama 60 detik pada setiap konfigurasi worker. Parameter lingkungan dan beban pengujian dibuat sama pada seluruh konfigurasi worker sehingga perbedaan hasil dapat dikaitkan dengan variasi jumlah worker.

4. Teknik Pengujian Sistem

Pengujian sistem pada penelitian ini menggunakan metode eksperimen komparatif untuk menganalisis pengaruh variasi jumlah worker terhadap kinerja sistem backend dan pemanfaatan Artificial Intelligence pada layanan AI berbasis GPU. Penelitian dilakukan dengan membandingkan lima konfigurasi jumlah worker, yaitu 1, 2, 4, 8, 16. Setiap konfigurasi diuji menggunakan lingkungan sistem, model AI, serta beban kerja yang sama sehingga hasil pengujian dapat dibandingkan berdasarkan jumlah worker yang digunakan.

Untuk menjaga konsistensi workload, setiap permintaan pengujian menggunakan prompt dengan panjang input yang dibuat sama dan parameter keluaran model yang dikonfigurasi secara tetap. Layanan inference menggunakan model AI yang sama pada seluruh skenario pengujian sehingga perbedaan hasil dapat dikaitkan dengan variasi jumlah worker.

Pengujian dilakukan menggunakan K6 dengan beban 50 virtual users (VUs) selama 60 detik pada setiap konfigurasi worker. Setiap konfigurasi worker diuji menggunakan skenario dan workload yang sama sehingga data yang diperoleh dapat dibandingkan secara objektif. Parameter yang diukur meliputi response time, throughput, error rate, GPU utilization, dan GPU idle time. Response time digunakan untuk mengetahui kecepatan sistem dalam memberikan respons, throughput digunakan untuk mengukur jumlah permintaan yang berhasil diproses per satuan waktu, sedangkan error rate digunakan untuk mengukur jumlah permintaan yang mengalami kegagalan pemrosesan. Evaluasi performa sistem dengan mempertimbangkan latency dan throughput merupakan pendekatan yang umum digunakan dalam pengukuran kinerja layanan pemrosesan secara daring (Maharjan et al., 2023).

Pemantauan GPU utilization dan GPU idle time dilakukan menggunakan NVTOP dan NVIDIA-SMI selama proses pengujian berlangsung. Data dari setiap konfigurasi kemudian dikumpulkan dan dianalisis untuk mengetahui hubungan antara tingkat concurrency worker, kinerja sistem backend, dan pemanfaatan sumber daya GPU. Selanjutnya, hasil dari seluruh konfigurasi dibandingkan untuk menentukan konfigurasi worker yang memberikan keseimbangan terbaik antara response time, throughput, stabilitas sistem, GPU utilization, dan GPU idle time. Evaluasi pemanfaatan CPU dan GPU

Journal : Journal of Computer Science and Information Technology (JCSIT)
Publisher : Yayasan Nuraini Ibrahim Mandiri
ISSN : 3031-8467
Volume / Issue / Year/ Page : Vol. 3 No. 4 (2026): September/ Hal 334-352

serta efisiensi penggunaan sumber daya merupakan bagian penting dalam analisis sistem komputasi heterogen (Mittal & Vetter, 2015).

5. Teknik Analisis Data

Analisis data dilakukan untuk mengetahui pengaruh variasi jumlah worker terhadap kinerja sistem backend dan utilisasi GPU berdasarkan sejumlah parameter kinerja utama, yaitu:

a. **Waktu Respons (*Response Time*):** Mengukur rata-rata waktu yang dibutuhkan sistem backend untuk memberikan respons terhadap setiap permintaan layanan AI. Nilai response time yang lebih rendah menunjukkan sistem mampu merespons permintaan dengan lebih cepat. Pengukuran waktu respons merupakan salah satu parameter penting dalam evaluasi kinerja layanan komputasi (Maharjan et al., 2023).
b. **Throughput:** Mengukur jumlah permintaan layanan AI yang berhasil diproses sistem backend dalam satuan waktu tertentu. Nilai throughput yang lebih tinggi menunjukkan kemampuan sistem dalam menangani beban permintaan yang lebih besar. Throughput dapat digunakan sebagai salah satu indikator untuk membandingkan kemampuan pemrosesan sistem pada kondisi beban yang berbeda (Maharjan et al., 2023) .
c. **Stabilitas Sistem (Error Rate):** Mengukur tingkat kegagalan sistem dalam memproses permintaan selama pengujian berlangsung. Stabilitas sistem dianalisis berdasarkan persentase permintaan yang gagal diproses terhadap total permintaan yang diberikan.
d. **Utilisasi GPU (%):** Mengukur persentase penggunaan GPU selama proses pengujian pada masing-masing skenario. Nilai utilisasi GPU digunakan untuk mengetahui tingkat efektivitas pemanfaatan sumber daya komputasi dalam proses layanan AI. Pemanfaatan GPU secara efektif berkaitan dengan kemampuan sistem dalam memanfaatkan karakteristik pemrosesan paralel GPU untuk beban komputasi (Owens et al., 2008).
e. **GPU Idle Time (%):** Mengukur persentase waktu GPU berada dalam kondisi tidak aktif selama pengujian berlangsung. Nilai idle time yang rendah menunjukkan distribusi tugas yang lebih optimal dan pemanfaatan GPU yang lebih efisien. Pengurangan waktu idle pada perangkat komputasi merupakan salah satu aspek yang perlu diperhatikan dalam optimalisasi pemanfaatan sumber daya CPU-GPU (Mittal & Vetter, 2015).
f. **Analisis Konfigurasi Worker Optimal:** Analisis dilakukan dengan membandingkan hasil dari setiap variasi jumlah worker untuk menentukan konfigurasi yang memberikan performa paling optimal. Evaluasi dilakukan berdasarkan keseimbangan antara response time, throughput, stabilitas sistem, utilisasi GPU, dan GPU idle time. Konfigurasi worker yang optimal ditentukan dari skenario yang mampu memberikan utilisasi GPU yang tinggi dengan latency dan error rate yang tetap rendah.

Hasil analisis dari seluruh parameter tersebut digunakan untuk mengevaluasi pengaruh mekanisme queue dan worker terhadap performa layanan AI berbasis GPU, serta menentukan konfigurasi worker yang paling efektif dalam meningkatkan efisiensi dan stabilitas sistem.

HASIL DAN PEMBAHASAN

Hasil penelitian ini diperoleh melalui tahapan implementasi dan pengujian sistem *backend* layanan *Artificial Intelligence* berbasis *queue* dan *worker* yang memanfaatkan

Journal : Journal of Computer Science and Information Technology (JCSIT)
Publisher : Yayasan Nuraini Ibrahim Mandiri
ISSN : 3031-8467
Volume / Issue / Year/ Page : Vol. 3 No. 4 (2026): September/ Hal 334-352

GPU sebagai media komputasi inferensi. Setelah sistem berhasil diimplementasikan, dilakukan serangkaian pengujian menggunakan variasi jumlah *worker* untuk menganalisis pengaruhnya terhadap kinerja sistem. Hasil yang diperoleh meliputi implementasi arsitektur sistem, mekanisme pemrosesan permintaan secara asinkron, serta hasil pengujian yang dianalisis berdasarkan parameter *response time*, *throughput*, *error rate*, utilisasi GPU, dan *GPU idle time*. Seluruh hasil tersebut digunakan untuk mengevaluasi pengaruh variasi jumlah *worker* terhadap performa sistem sekaligus menentukan konfigurasi *worker* yang paling optimal dalam mendukung layanan *Artificial Intelligence* berbasis GPU.

1. Implementasi Sistem

Implementasi sistem merupakan tahap penerapan hasil perancangan ke dalam bentuk sistem backend yang digunakan untuk memproses layanan Artificial Intelligence (AI) berbasis GPU. Sistem dikembangkan menggunakan bahasa pemrograman Go dengan menerapkan mekanisme queue dan worker pool untuk memisahkan proses penerimaan permintaan dari proses eksekusi inferensi. Mekanisme tersebut memungkinkan permintaan yang diterima backend diproses secara asinkron melalui antrean Redis dan dijalankan secara paralel oleh sejumlah worker.

Berdasarkan **Gambar 2**, sistem terdiri atas tujuh komponen utama, yaitu load generator K6, backend API, Redis Queue, worker pool Asynq, vLLM sebagai inference engine, GPU sebagai sumber daya komputasi, dan database sebagai media penyimpanan hasil pemrosesan. Setiap komponen memiliki fungsi yang berbeda dan saling terintegrasi untuk membentuk alur pemrosesan layanan AI. K6 digunakan untuk menghasilkan beban permintaan selama pengujian, sedangkan backend API, Redis Queue, dan worker pool berperan dalam mengatur penerimaan, antrean, dan eksekusi tugas sebelum diteruskan ke layanan inferensi vLLM.

Alur pemrosesan sistem pada **Gambar 2** dimulai ketika load generator K6 mengirimkan HTTP request ke backend API. Permintaan tersebut kemudian dimasukkan ke dalam Redis Queue sebagai job melalui mekanisme enqueue. Redis berfungsi sebagai media antrean yang menyimpan job hingga tersedia worker untuk memprosesnya. Selanjutnya, worker pool Asynq mengambil job dari Redis melalui mekanisme dequeue dan menjalankan tugas secara paralel sesuai dengan jumlah worker yang dikonfigurasikan.

Pada tahap pemrosesan, worker mengirimkan inference request ke vLLM yang berperan sebagai inference engine. vLLM kemudian meneruskan proses komputasi ke GPU untuk menjalankan inferensi model AI. Penggunaan GPU memungkinkan proses inferensi dilakukan menggunakan sumber daya komputasi paralel yang sesuai dengan karakteristik beban kerja model AI. Setelah proses inferensi selesai, hasil pemrosesan disimpan ke dalam database sebagai result store. Hasil tersebut selanjutnya dapat diambil oleh backend API untuk diberikan kembali sebagai response kepada K6.

a. ***Load Generator (K6)*****:** Load generator K6 digunakan untuk menghasilkan beban permintaan secara simultan terhadap backend API. Pada penelitian ini, K6 dikonfigurasikan dengan 50 virtual users (VUs) selama 60 detik. Setiap VU menghasilkan permintaan ke endpoint layanan inferensi sehingga sistem memperoleh beban kerja yang konsisten pada setiap konfigurasi jumlah worker yang diuji.
b. ***Backend API (Go)*****:** Backend API dikembangkan menggunakan bahasa pemrograman Go dan berfungsi sebagai titik masuk permintaan dari K6. Backend menerima HTTP request, melakukan validasi terhadap permintaan, kemudian membuat job yang akan

Journal : Journal of Computer Science and Information Technology (JCSIT)
Publisher : Yayasan Nuraini Ibrahim Mandiri
ISSN : 3031-8467
Volume / Issue / Year/ Page : Vol. 3 No. 4 (2026): September/ Hal 334-352

diproses oleh worker. Job tersebut selanjutnya dikirimkan ke Redis Queue melalui mekanisme enqueue. Dengan pendekatan ini, proses penerimaan permintaan dapat dipisahkan dari proses inferensi yang membutuhkan waktu komputasi lebih besar.

c. **Redis *Queue*:** Redis digunakan sebagai message queue untuk menyimpan job yang menunggu proses eksekusi. Setelah backend API memasukkan job ke dalam antrean, Redis mempertahankan job hingga diambil oleh worker. Mekanisme ini memungkinkan proses penerimaan permintaan dan proses eksekusi berjalan secara terpisah sehingga beberapa permintaan dapat menunggu di antrean tanpa harus dieksekusi secara langsung oleh backend API.
d. ***Worker Pool (Asynq)*:** Worker pool merupakan komponen yang bertugas mengambil job dari Redis dan menjalankan proses pemrosesan secara paralel. Pada penelitian ini, worker diimplementasikan menggunakan Asynq dengan variasi jumlah 1, 2, 4, 8, dan 16 worker. Variasi tersebut digunakan untuk mengamati pengaruh tingkat concurrency terhadap kinerja sistem. Setiap worker yang memperoleh job akan mengirimkan inference request ke vLLM untuk diproses menggunakan GPU.
e. **vLLM (Inference Engine):** vLLM digunakan sebagai inference engine yang menerima permintaan dari worker dan menjalankan proses inferensi menggunakan model AI. Pada sistem ini, vLLM menjadi lapisan yang menghubungkan permintaan dari worker dengan sumber daya komputasi GPU. Hasil proses inferensi kemudian dikembalikan ke sistem untuk disimpan sebagai hasil pemrosesan.
f. **GPU:** GPU digunakan sebagai sumber daya komputasi utama untuk menjalankan proses inferensi model AI melalui vLLM. Pada lingkungan pengujian, GPU digunakan untuk memproses permintaan inferensi secara paralel. Tingkat pemanfaatan GPU menjadi salah satu parameter utama dalam penelitian untuk mengevaluasi seberapa efektif variasi jumlah worker dalam memasok pekerjaan ke proses inferensi.
g. Database (Result Store): Database digunakan sebagai result store untuk menyimpan hasil pemrosesan yang dihasilkan oleh layanan inferensi. Penyimpanan hasil memungkinkan data hasil inferensi tersedia untuk proses pengambilan kembali melalui backend API. Dengan demikian, hasil pemrosesan tidak harus diperoleh melalui pengulangan proses inferensi.

Backend pada penelitian ini dikembangkan menggunakan bahasa pemrograman Go (Golang) dengan menerapkan pola *Layered Architecture*. Pola ini dipilih karena mampu memisahkan tanggung jawab setiap komponen sistem sehingga kode program menjadi lebih terstruktur, mudah dikembangkan, dan lebih mudah dipelihara. Selain itu, pemisahan logika aplikasi ke dalam beberapa lapisan juga memudahkan proses pengujian dan pengembangan fitur baru tanpa memengaruhi komponen lainnya.

Alur pemrosesan pada *backend* dimulai ketika *controller* menerima permintaan dari *client*. Permintaan tersebut diteruskan ke *service* untuk diproses sesuai kebutuhan bisnis. Selanjutnya *service* dapat berinteraksi dengan *repository* untuk pengelolaan data atau mengirimkan tugas ke *queue* untuk diproses oleh *worker* secara asinkron. Pendekatan ini memungkinkan setiap lapisan memiliki tanggung jawab yang jelas sehingga sistem menjadi lebih modular dan mudah dikembangkan.

Implementasi *queue* pada penelitian ini bertujuan untuk memisahkan proses penerimaan permintaan dan proses eksekusi layanan *Artificial Intelligence* sehingga pemrosesan dapat dilakukan secara asinkron. Mekanisme ini memungkinkan *backend* API tetap responsif dalam menerima permintaan baru tanpa harus menunggu proses inferensi AI selesai dijalankan. Pada penelitian ini, Redis digunakan sebagai media antrean (*message broker*), sedangkan *Asynq* digunakan sebagai pustaka (*library*) untuk

Journal : Journal of Computer Science and Information Technology (JCSIT)
Publisher : Yayasan Nuraini Ibrahim Mandiri
ISSN : 3031-8467
Volume / Issue / Year/ Page : Vol. 3 No. 4 (2026): September/ Hal 334-352

mengelola proses *enqueue* dan *dequeue* tugas.

Redis dipilih sebagai media antrean karena memiliki performa tinggi berbasis penyimpanan *in-memory* sehingga mampu menangani operasi baca dan tulis dengan latensi rendah. Selain itu, Redis telah didukung secara langsung oleh *Asynq* sehingga mempermudah implementasi sistem antrean pada aplikasi *backend* berbasis Golang.

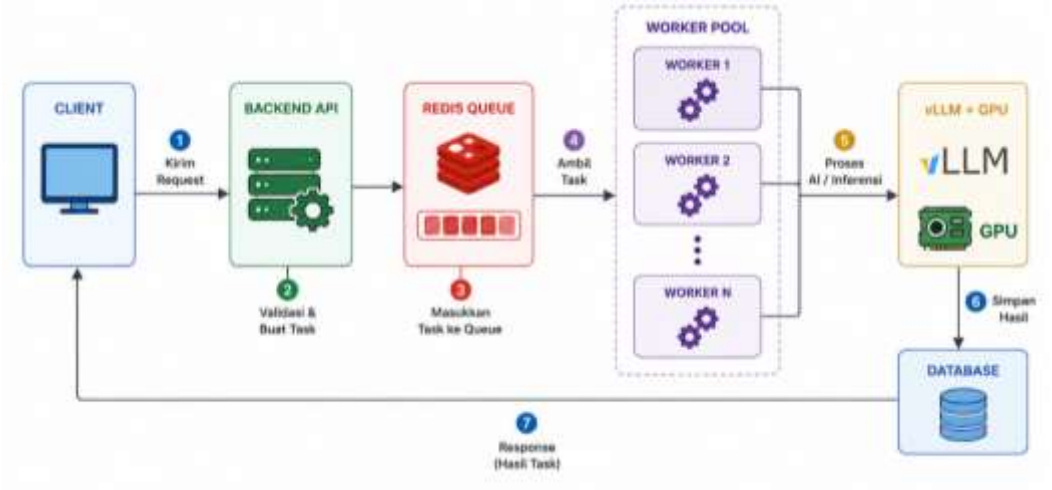
Setiap permintaan yang diterima *backend* dikonversi menjadi sebuah *task* yang berisi informasi yang dibutuhkan *worker* untuk menjalankan proses inferensi AI. *Task* tersebut kemudian diserialisasi dalam format JSON sebelum dimasukkan ke dalam antrean Redis.

TABEL 2. *Payload* yang dikirim ke *queue*

Field	Keterangan
prompt	*Input* pengguna berupa pertanyaan atau perintah yang akan diproses oleh layanan *Artificial Intelligence*
model	Nama model *Artificial Intelligence* yang digunakan untuk melakukan proses inferensi

Setelah permintaan berhasil divalidasi oleh *backend*, sistem akan membuat objek *task* yang berisi *payload* permintaan pengguna. *Task* tersebut kemudian dimasukkan ke dalam Redis *Queue* menggunakan fungsi *Enqueue()* dari *Asynq* sehingga dapat diproses oleh *worker* secara asinkron.

Alur pemrosesan permintaan pada sistem yang diimplementasikan menggunakan mekanisme *queue* dan *worker* dilakukan secara asinkron. Mekanisme ini memisahkan proses penerimaan permintaan dari proses eksekusi layanan *Artificial Intelligence* sehingga *backend* tetap responsif meskipun terdapat banyak permintaan yang masuk secara bersamaan.



GAMBAR 3. Alur Pemrosesan Menggunakan Queue dan Worker

Proses sistem dimulai ketika *client* mengirimkan permintaan melalui *backend API*. Permintaan yang diterima divalidasi, kemudian dibentuk menjadi *task* dan dimasukkan ke dalam Redis Queue. Selanjutnya, *worker* mengambil *task* dari antrean dan mengirimkannya ke layanan VLLM untuk dilakukan proses inferensi menggunakan GPU. Hasil inferensi kemudian disimpan ke dalam basis data dan dikembalikan kepada *client* melalui *backend API*.

Implementasi sistem menggunakan mekanisme worker pool untuk memisahkan proses penerimaan permintaan dan proses komputasi AI. Pendekatan ini memungkinkan *backend* tetap responsif terhadap permintaan pengguna sekaligus

Journal : Journal of Computer Science and Information Technology (JCSIT)
Publisher : Yayasan Nuraini Ibrahim Mandiri
ISSN : 3031-8467
Volume / Issue / Year/ Page : Vol. 3 No. 4 (2026): September/ Hal 334-352

mendukung pemrosesan tugas secara paralel. Jumlah *worker* yang digunakan menentukan tingkat konkurensi sistem, sehingga berpengaruh terhadap *response time*, *throughput*, dan pemanfaatan sumber daya GPU.

Penelitian ini menggunakan lima konfigurasi jumlah *worker*, yaitu 1, 2, 4, 8, dan 16 worker. Seluruh konfigurasi diuji menggunakan beban kerja yang sama untuk memperoleh data yang konsisten mengenai *response time*, *throughput*, *error rate*, *GPU utilization*, dan *GPU idle time*. Hasil pengujian kemudian dianalisis untuk mengetahui pengaruh variasi jumlah *worker* terhadap kinerja sistem.

Layanan *Artificial Intelligence* diimplementasikan menggunakan VLLM sebagai *inference engine* yang berjalan pada server GPU. Setiap *worker* mengirimkan *prompt* ke VLLM melalui API lokal, kemudian VLLM melakukan proses inferensi menggunakan GPU dan mengembalikan hasilnya kepada *worker* untuk disimpan ke dalam basis data sebelum diteruskan kepada pengguna.

Seluruh pengujian dilakukan menggunakan model AI yang sama pada satu unit GPU NVIDIA L40S sehingga lingkungan pengujian tetap konsisten. Dengan demikian, perbedaan performa sistem dan tingkat utilisasi GPU pada setiap skenario pengujian dapat dikaitkan secara langsung dengan variasi jumlah *worker*, sehingga konfigurasi *worker* yang paling optimal dapat ditentukan secara objektif.

Untuk memastikan proses inferensi layanan *Artificial Intelligence* dijalankan pada satu GPU yang telah ditentukan, dilakukan pemantauan menggunakan aplikasi NVTOP. *Server* yang digunakan memiliki dua buah GPU NVIDIA L40S, namun pada penelitian ini hanya satu GPU yang digunakan sebagai sumber daya komputasi untuk menjalankan layanan AI. Hal tersebut bertujuan untuk menjaga konsistensi lingkungan pengujian serta mempermudah analisis hubungan antara jumlah *worker* dan tingkat utilisasi GPU.

Penggunaan satu GPU pada penelitian ini bertujuan untuk memperoleh hasil pengujian yang lebih konsisten. Dengan memusatkan seluruh proses inferensi pada satu GPU yang sama, pengaruh variasi jumlah *worker* terhadap parameter *response time*, *throughput*, GPU *utilization*, dan GPU *idle time* dapat dianalisis secara lebih objektif tanpa dipengaruhi oleh distribusi beban ke beberapa GPU.

2. Skenario Pengujian

Pengujian sistem dilakukan pada lingkungan *backend* layanan *Artificial Intelligence* yang diimplementasikan menggunakan mekanisme *queue* dan *worker*. Seluruh komponen sistem dijalankan pada *server* GPU milik Universitas Muhammadiyah Makassar yang dilengkapi dengan dua GPU NVIDIA L40S. Namun, untuk menjaga konsistensi hasil pengujian, penelitian ini hanya memanfaatkan satu GPU sebagai sumber daya komputasi utama. Pengujian dilakukan menggunakan aplikasi K6 untuk menghasilkan beban permintaan secara simultan, sedangkan Redis digunakan sebagai media antrean dan PostgreSQL digunakan untuk penyimpanan data hasil inferensi.

Pengujian dilakukan untuk menganalisis pengaruh variasi jumlah *worker* terhadap performa sistem *backend* dan tingkat utilisasi GPU. Beberapa parameter yang diamati meliputi *response time*, *throughput*, *error rate*, GPU *utilization*, dan GPU *idle time*. Parameter tersebut digunakan untuk menentukan konfigurasi *worker* yang paling optimal dalam memanfaatkan sumber daya GPU.

Pengujian dilakukan menggunakan beberapa konfigurasi jumlah *worker* untuk mengetahui pengaruh tingkat konkurensi terhadap kinerja sistem *backend* dan tingkat pemanfaatan GPU. Variasi konfigurasi yang digunakan meliputi 1, 2, 4, 8, dan 16 *worker*. Setiap konfigurasi diuji menggunakan model *Artificial Intelligence*, lingkungan sistem, serta beban kerja yang sama, yaitu 50 *virtual users* selama 60 detik, sehingga hasil

Journal : Journal of Computer Science and Information Technology (JCSIT)
Publisher : Yayasan Nuraini Ibrahim Mandiri
ISSN : 3031-8467
Volume / Issue / Year/ Page : Vol. 3 No. 4 (2026): September/ Hal 334-352

pengujian dari masing-masing konfigurasi dapat dibandingkan secara objektif. Data yang diperoleh kemudian dianalisis berdasarkan parameter *response time*, *throughput*, *error rate*, GPU *utilization*, dan GPU *idle time* untuk menentukan konfigurasi *worker* yang memberikan kinerja sistem dan efisiensi pemanfaatan GPU yang paling optimal.

Beban pengujian diberikan menggunakan aplikasi K6 dengan mengirimkan permintaan secara simultan ke *endpoint* layanan AI. Setiap variasi jumlah *worker* diuji menggunakan skenario beban yang sama untuk menjaga konsistensi hasil pengujian. Beban diberikan selama 60 detik dengan jumlah *virtual user* (VUs) tertentu sehingga dapat diamati pengaruh jumlah *worker* terhadap *response time*, *throughput*, *error rate*, serta utilisasi GPU.

Penggunaan 50 virtual users (VUs) ditetapkan sebagai beban pengujian yang sama untuk seluruh konfigurasi worker agar perbandingan performa dapat dilakukan secara objektif. Jumlah tersebut digunakan untuk menghasilkan permintaan secara simultan sehingga mekanisme queue dan worker pool dapat bekerja dalam kondisi konkurensi yang cukup untuk mengamati pengaruh perubahan jumlah worker terhadap performa sistem. Dengan mempertahankan jumlah VUs, durasi pengujian, endpoint, model AI, dan workload yang sama, perubahan response time, throughput, error rate, dan utilisasi GPU dapat lebih langsung dikaitkan dengan variasi jumlah worker.

Perlu ditegaskan bahwa 50 VUs dalam penelitian ini merupakan parameter controlled workload untuk eksperimen komparatif dan bukan parameter untuk menentukan batas maksimum (stress limit) sistem. Oleh karena itu, error rate sebesar 0% menunjukkan bahwa sistem mampu menyelesaikan seluruh permintaan pada beban pengujian yang digunakan, tetapi belum dapat digunakan untuk menyimpulkan kapasitas maksimum sistem. Pengujian stress limit dapat dilakukan pada penelitian lanjutan dengan meningkatkan jumlah VUs secara bertahap hingga terjadi degradasi performa, peningkatan error rate, atau tercapainya batas sumber daya sistem.

TABEL 3. Konfigurasi Beban Pengujian

Parameter	**Nilai**
Tool Pengujian	K6
Enpoint	*/api/v1/inference*
Virtual User (VUs)	50
Durasi Pengujian	60 detik
Payload	*prompt*
Model AI	Llama2
Jumlah *Worker*	1, 2, 4, 8, 16

3. Analisis Pengaruh Jumlah *Worker* terhadap Performa Sistem

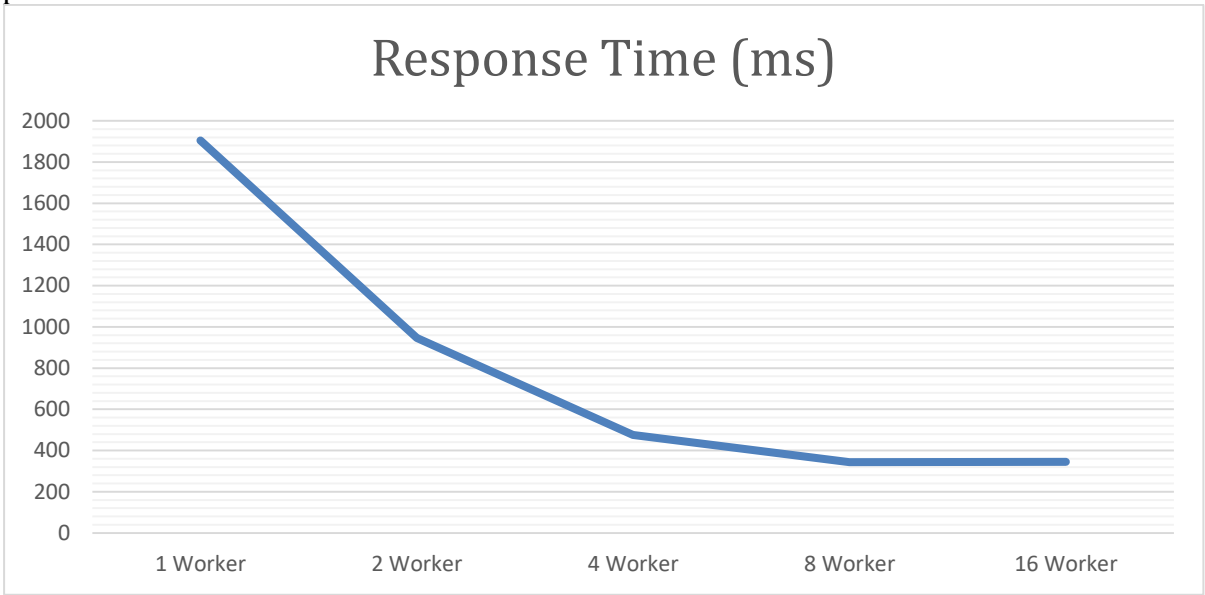
TABEL 4. Hasil Pengujian Konfigurasi *Worker*

Jumlah *Worker*	***Response Time* (ms)**	***Throughput* (request/s)**	***Error Rate* (%)**	**Utilisasi GPU (%)**	**GPU *Idle Time* (%)**
1 *worker*	1904	26,60	0	80,21	18,3
2 *worker*	946,13	52,78	0	90,65	8,3
4 *worker*	475,30	104,86	0	96,60	3,3
8 *worker*	343,54	131	0	100	0
16 *worker*	345,70	142,91	0	99,30	0

Berdasarkan hasil pengujian pada **Tabel 4**, peningkatan jumlah worker memberikan pengaruh terhadap performa sistem, terutama pada response time, throughput, dan

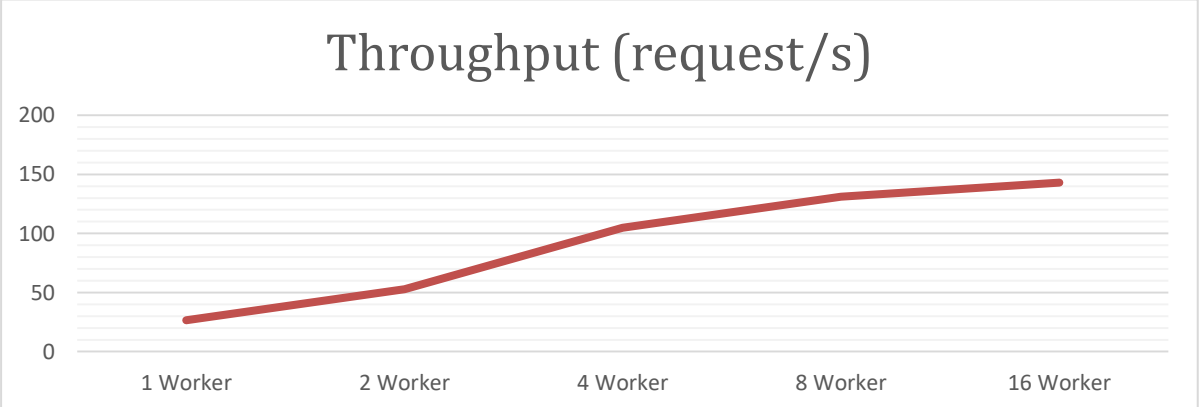
Journal : Journal of Computer Science and Information Technology (JCSIT)
Publisher : Yayasan Nuraini Ibrahim Mandiri
ISSN : 3031-8467
Volume / Issue / Year/ Page : Vol. 3 No. 4 (2026): September/ Hal 334-352

pemanfaatan GPU. Pada konfigurasi 1 worker, sistem menghasilkan response time sebesar 1.904 ms dengan throughput sebesar 26,60 request/s. Ketika jumlah worker ditingkatkan menjadi 2 dan 4, response time menurun masing-masing menjadi 946,13 ms dan 475,30 ms, sedangkan throughput meningkat menjadi 52,78 request/s dan 104,86 request/s. Penurunan response time dan peningkatan throughput tersebut menunjukkan bahwa mekanisme worker pool mampu meningkatkan tingkat konkurensi pemrosesan sehingga permintaan yang masuk dapat diproses secara lebih paralel.



GAMBAR 4. Grafik Response Time terhadap Jumlah Worker

Perubahan paling signifikan terjadi hingga konfigurasi 8 worker. Pada konfigurasi tersebut, response time mencapai nilai terendah sebesar 343,54 ms, sedangkan throughput meningkat menjadi 131 request/s. Pada saat yang sama, utilisasi GPU mencapai 100% dan GPU idle time mencapai 0%. Kondisi tersebut menunjukkan bahwa peningkatan jumlah worker dari 1 hingga 8 mampu meningkatkan suplai pekerjaan ke proses inferensi sehingga GPU dapat dimanfaatkan secara lebih kontinu. Dengan demikian, peningkatan worker pada rentang tersebut tidak hanya meningkatkan kemampuan pemrosesan permintaan, tetapi juga mengurangi waktu GPU berada dalam kondisi tidak aktif.



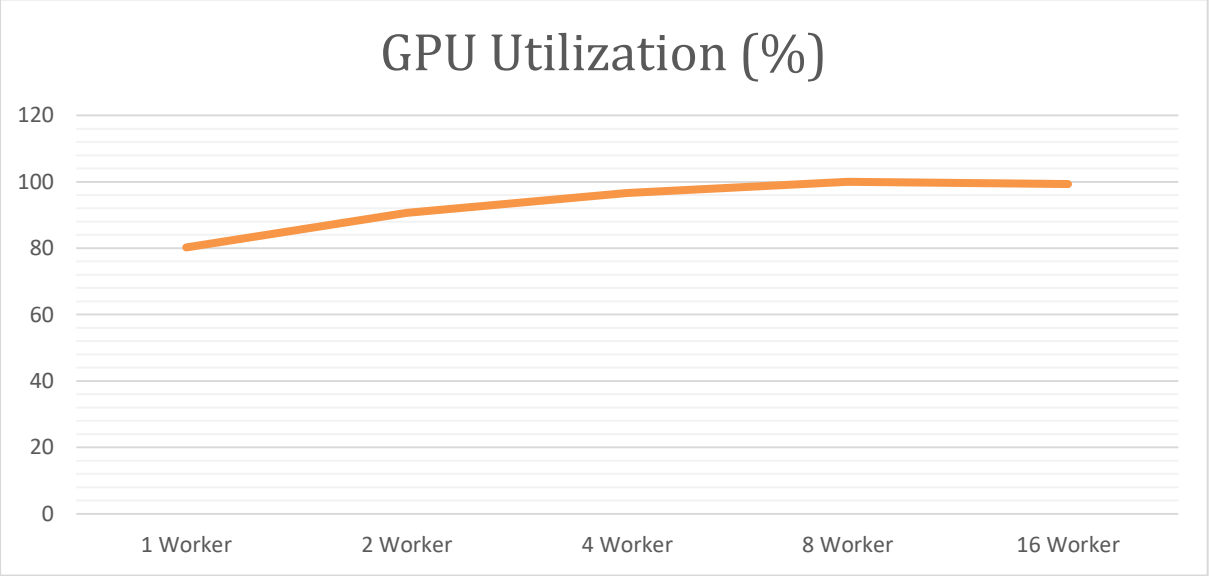
GAMBAR 5. Grafik Throughput terhadap Jumlah Worker

Journal : Journal of Computer Science and Information Technology (JCSIT)
Publisher : Yayasan Nuraini Ibrahim Mandiri
ISSN : 3031-8467
Volume / Issue / Year/ Page : Vol. 3 No. 4 (2026): September/ Hal 334-352

Namun, pola peningkatan performa mulai berubah ketika jumlah worker ditingkatkan dari 8 menjadi 16. Throughput memang meningkat dari 131 menjadi 142,91 request/s atau sekitar 9,09%, tetapi response time justru meningkat dari 343,54 ms menjadi 345,70 ms atau sekitar 0,63%. Pada kondisi ini, GPU telah mencapai utilisasi 100% pada konfigurasi 8 worker, sehingga penambahan worker tidak lagi dapat meningkatkan kapasitas komputasi GPU secara proporsional. Penambahan worker memungkinkan lebih banyak pekerjaan tersedia secara bersamaan, tetapi kapasitas komputasi GPU tetap menjadi faktor pembatas dalam proses inferensi.

Fenomena tersebut menunjukkan adanya trade-off antara latency dan throughput. Pada konfigurasi 16 worker, peningkatan jumlah permintaan yang dapat diproses menghasilkan throughput yang lebih tinggi, tetapi tidak menghasilkan penurunan response time. Sebaliknya, terjadi sedikit peningkatan response time. Secara fenomenologis, kondisi ini mengindikasikan bahwa sistem telah memasuki kondisi mendekati GPU saturation, yaitu ketika penambahan tingkat konkurensi pada sisi worker tidak lagi menghasilkan peningkatan kecepatan pemrosesan secara signifikan. Pada kondisi tersebut, proses penjadwalan dan pengelolaan pekerjaan oleh worker dan inference engine vLLM berpotensi menambah waktu tunggu pemrosesan, sementara kapasitas komputasi GPU tetap terbatas. Dengan demikian, peningkatan throughput pada 16 worker lebih mencerminkan peningkatan kapasitas pemrosesan permintaan dibandingkan peningkatan kecepatan penyelesaian setiap permintaan.

Selama seluruh pengujian, setiap konfigurasi worker menghasilkan error rate sebesar 0%. Hasil tersebut menunjukkan bahwa seluruh permintaan yang diberikan selama skenario pengujian berhasil diproses tanpa menghasilkan kegagalan pada sisi aplikasi. Namun, nilai error rate 0% tidak digunakan untuk menyimpulkan bahwa sistem telah mencapai batas stress limit atau tidak memiliki batas kapasitas. Pengujian dalam penelitian ini dirancang sebagai eksperimen komparatif dengan beban yang sama pada setiap konfigurasi worker, sehingga error rate digunakan sebagai salah satu indikator stabilitas untuk membandingkan konsistensi sistem antar-konfigurasi.



GAMBAR 6. Grafik Utilisasi GPU terhadap Jumlah Worker

Dari perspektif pemanfaatan sumber daya, utilisasi GPU meningkat dari 80,21% pada 1 worker menjadi 90,65% pada 2 worker, 96,60% pada 4 worker, dan mencapai 100% pada 8 worker. Pada 16 worker, utilisasi GPU tetap sangat tinggi, yaitu 99,30%. Pola tersebut memperkuat indikasi bahwa konfigurasi 8 worker telah mencapai kondisi

Journal : Journal of Computer Science and Information Technology (JCSIT)
Publisher : Yayasan Nuraini Ibrahim Mandiri
ISSN : 3031-8467
Volume / Issue / Year/ Page : Vol. 3 No. 4 (2026): September/ Hal 334-352

pemanfaatan GPU yang sangat tinggi. GPU idle time juga mengalami penurunan dari 18,3% pada 1 worker menjadi 8,3% pada 2 worker, 3,3% pada 4 worker, dan 0% pada 8 serta 16 worker. Artinya, penambahan worker hingga 8 mampu mengurangi waktu GPU tidak aktif, sedangkan penambahan worker setelah kondisi tersebut tidak lagi memberikan keuntungan pada parameter GPU idle time.

Trade-off Latency dan Throughput pada GPU Saturation

Perbandingan konfigurasi 8 dan 16 worker menunjukkan titik penting dalam hubungan antara tingkat konkurensi, latency, dan kapasitas pemrosesan. Konfigurasi 8 worker menghasilkan response time sebesar 343,54 ms dengan throughput 131 request/s, sedangkan konfigurasi 16 worker menghasilkan response time 345,70 ms dengan throughput 142,91 request/s. Dengan demikian, penambahan 8 worker hanya menghasilkan peningkatan throughput sebesar 9,09%, sementara response time meningkat sebesar 0,63%.

Kondisi tersebut menunjukkan bahwa peningkatan jumlah worker setelah GPU mencapai utilisasi maksimum tidak lagi menghasilkan peningkatan performa yang linear. Pada 8 worker, GPU telah mencapai utilisasi 100% dan GPU idle time 0%, sehingga sumber daya komputasi GPU telah dimanfaatkan secara maksimal selama pengujian. Ketika jumlah worker ditingkatkan menjadi 16, permintaan yang tersedia untuk diproses secara bersamaan bertambah, tetapi kapasitas komputasi GPU tidak ikut bertambah. Akibatnya, tambahan konkurensi lebih banyak digunakan untuk meningkatkan jumlah pekerjaan yang dapat diproses dalam satuan waktu, sementara waktu penyelesaian masing-masing permintaan relatif tidak mengalami perbaikan.

Dari perspektif inference engine vLLM, kondisi tersebut dapat mengindikasikan meningkatnya tekanan konkurensi pada proses penjadwalan dan pengelolaan permintaan ketika jumlah worker melebihi titik pemanfaatan GPU yang efektif. Meskipun penelitian ini tidak melakukan pengukuran langsung terhadap queueing delay, context switching, atau penggunaan memori internal vLLM, perubahan pola response time dan throughput menunjukkan bahwa penambahan worker dari 8 menjadi 16 telah memberikan marginal performance gain. Oleh karena itu, konfigurasi 8 worker lebih sesuai dipilih sebagai konfigurasi optimal apabila tujuan sistem adalah memperoleh keseimbangan antara latency, throughput, dan efisiensi pemanfaatan GPU.

Penentuan Konfigurasi *Worker* Optimal

Berdasarkan keseluruhan parameter pengujian, konfigurasi 8 *worker* memberikan keseimbangan performa terbaik. Pada konfigurasi ini diperoleh *response time* terendah sebesar 343,54 ms, *throughput* sebesar 131 *request/s*, *error rate* 0%, utilisasi GPU mencapai 100%, dan *GPU idle time* sebesar 0%. Hasil tersebut menunjukkan bahwa sistem mampu memanfaatkan sumber daya GPU secara optimal dengan waktu respons yang paling rendah.

Meskipun konfigurasi 16 *worker* menghasilkan *throughput* yang sedikit lebih tinggi, peningkatan tersebut tidak sebanding dengan kenaikan *response time* dan kondisi GPU yang telah mencapai titik jenuh (*GPU saturation*). Oleh karena itu, konfigurasi 8 *worker* dipilih sebagai konfigurasi yang paling optimal karena mampu memberikan keseimbangan terbaik antara kecepatan respons, kapasitas pemrosesan, stabilitas sistem, dan efisiensi pemanfaatan sumber daya GPU.

Journal : Journal of Computer Science and Information Technology (JCSIT)
Publisher : Yayasan Nuraini Ibrahim Mandiri
ISSN : 3031-8467
Volume / Issue / Year/ Page : Vol. 3 No. 4 (2026): September/ Hal 334-352

KESIMPULAN

Berdasarkan hasil perancangan, implementasi, dan pengujian, mekanisme Redis Queue dan worker pool pada backend layanan Artificial Intelligence mampu meningkatkan kemampuan sistem dalam menangani permintaan inferensi secara asinkron. Hasil eksperimen menunjukkan bahwa peningkatan jumlah worker memberikan peningkatan kinerja hingga mencapai 8 worker, yang menjadi titik optimal pada lingkungan pengujian. Pada kondisi tersebut, utilisasi GPU mencapai 100% dan GPU idle time 0%. Penambahan worker menjadi 16 tidak lagi memberikan peningkatan response time dan throughput yang sebanding. Temuan ini menunjukkan adanya titik jenuh (saturation point) GPU, ketika kapasitas komputasi GPU telah dimanfaatkan secara maksimal sehingga penambahan concurrency cenderung meningkatkan beban antrean dan waktu tunggu pemrosesan tanpa memberikan peningkatan kinerja yang signifikan.

Secara praktis, hasil penelitian menunjukkan bahwa konfigurasi jumlah worker perlu disesuaikan dengan kapasitas komputasi GPU agar tercapai keseimbangan antara latency, throughput, dan efisiensi pemanfaatan sumber daya. Namun, penelitian ini memiliki keterbatasan karena pengujian dilakukan pada satu GPU, satu model LLM, dan beban 50 virtual users, sehingga hasilnya belum dapat digeneralisasikan pada lingkungan multi-GPU atau beban yang lebih tinggi. Penelitian selanjutnya dapat mengembangkan pengujian pada arsitektur multi-GPU serta menerapkan mekanisme dynamic batching untuk mengevaluasi kemampuan sistem dalam meningkatkan efisiensi pemrosesan ketika jumlah permintaan dan concurrency meningkat.

PERNYATAAN PENGGUNAAN KECERDASAN BUATAN

Kecerdasan buatan (AI) digunakan dalam penyusunan naskah ini. Penulis menggunakan alat-alat AI semata-mata untuk membantu penyempurnaan bahasa, pemeriksaan tata bahasa, dan penyuntingan guna memastikan kejelasan dan keterbacaan naskah. Penulis tetap bertanggung jawab penuh atas keakuratan, orisinalitas, integritas, dan isi akademis naskah ini. Seluruh isi yang dihasilkan dengan bantuan AI telah ditinjau dan diverifikasi oleh penulis sebelum naskah ini diajukan.

KONFLIK KEPENTINGAN

Penulis menyatakan tidak ada konflik kepentingan

KONTRIBUSI PENULIS

Konseptualisasi, SR dan FIR; pengumpulan data, SR; validasi data, SR; analisis formal, SR dan MAH; penulisan—penyusunan draf awal, SR dan RYB; penulisan—peninjauan dan penyuntingan, SR dan TW; visualisasi, SR dan D. Semua penulis telah membaca dan menyetujui versi akhir naskah ini.

UCAPAN TERIMA KASIH

Ucapan terima kasih yang tulus disampaikan kepada COCONUT Computer Club dan Universitas Muhammadiyah Makassar atas dukungan fasilitas dan infrastruktur yang menunjang proses pengembangan serta pengujian sistem, khususnya dalam pemanfaatan sumber daya komputasi berbasis GPU. Terima kasih juga disampaikan kepada rekan-rekan yang telah memberikan dukungan, bantuan, dan masukan selama proses penelitian dan pengembangan sistem. Apresiasi yang sebesar-besarnya turut disampaikan kepada keluarga dan rekan kerja atas dukungan moral, motivasi, serta

Journal : Journal of Computer Science and Information Technology (JCSIT)
Publisher : Yayasan Nuraini Ibrahim Mandiri
ISSN : 3031-8467
Volume / Issue / Year/ Page : Vol. 3 No. 4 (2026): September/ Hal 334-352

semangat yang diberikan sehingga penelitian dan penyusunan karya ini dapat diselesaikan dengan baik.

DAFTAR PUSTAKA

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N., Chen, A., Creel, K., Davis, J. Q., Demszky, D., … Liang, P. (2022). *On the Opportunities and Risks of Foundation Models*. http://arxiv.org/abs/2108.07258

Dean, J., & Barroso, L. A. (2013). The tail at scale. *Communications of the ACM*, *56*(2), 74–80. https://doi.org/10.1145/2408776.2408794

Hong, S., & Kim, H. (2009). *An Analytical Model for a GPU Architecture with Memory-level and Thread-level Parallelism Awareness* (Vol. 37). ACM SIGARCH Computer Architecture News. https://doi.org/10.1145/1555815.1555775

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J., Zhang, H., & Stoica, I. (2023). Efficient Memory Management for Large Language Model Serving with PagedAttention. *SOSP 2023 - Proceedings of the 29th ACM Symposium on Operating Systems Principles*, 611–626. https://doi.org/10.1145/3600006.3613165

Maharjan, R., Chy, M. S. H., Arju, M. A., & Cerny, T. (2023). Benchmarking Message Queues. *Telecom*, *4*(2), 298–312. https://doi.org/10.3390/telecom4020018

Mittal, S., & Vetter, J. S. (2015). A survey of CPU-GPU heterogeneous computing techniques. *ACM Computing Surveys*, *47*(4). https://doi.org/10.1145/2788396

Owens, J. D., Houston, M., Luebke, D., Green, S., Stone, J. E., & Phillips, J. C. (2008). GPU computing. *Proceedings of the IEEE*, *96*(5), 879–899. https://doi.org/10.1109/JPROC.2008.917757

Topalidi, A. (n.d.). Asynchronous processes and message queues in Ruby applications: efficiency analysis of Sidekiq and RabbitMQ. *International Journal on Science and Technology*. Retrieved www.ijsat.org•

Vaswani, A., Brain, G., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2023). *Attention Is All You Need*.