Analisis Skalabilitas Message Broker Terdistribusi dalam Pemrosesan Katalog Visual pada Platform Shopee

Alwi Sofyan Affandi[1*], Nisa Aulia Ramadhani[2], Andriyan Susetyo[3], Pradina Aulia Resti Prasmita[4]

[1*]*Teknik Informatika*
Universitas Duta Bangsa
[1*]240103124@mhs.udb.ac.id

[2]*Teknik Informatika*
Universitas Duta Bangsa
[2]240103143@mhs.udb.ac.id

[3]*Teknik Informatika*
Universitas Duta Bangsa
[3]240103125@mhs.udb.ac.id

[4]*S1 Teknik Informatika*
Universitas Duta Bangsa
[4]240103144@mhs.udb.ac.id

***Abstrak*— Platform *e-commerce* berskala besar seperti Shopee mengelola jutaan produk visual secara *real-time*, sehingga membutuhkan infrastruktur pemrosesan data yang andal, latensi rendah, dan mampu menangani lonjakan beban tinggi. Penelitian ini bertujuan mengevaluasi skalabilitas *message broker* terdistribusi menggunakan RabbitMQ dalam mendukung aliran data katalog visual pada platform *e-commerce*, khususnya dalam aspek *throughput*, latensi, dan efisiensi distribusi beban kerja. Metode yang digunakan meliputi pengujian skalabilitas berbasis skenario beban bertahap (120 aset gambar produk elektronik) dengan membandingkan Skenario A (tunggal/*single node*) dan Skenario B (multi-worker terdistribusi paralel). Arsitektur pemrosesan katalog ini diintegrasikan dengan modul visi komputer untuk otomatisasi penamaan katalog visual (*auto-rename*). Hasil pengujian menunjukkan bahwa arsitektur terdistribusi paralel mampu memangkas waktu pemrosesan secara signifikan dan mencegah terjadinya penumpukan antrean (*bottleneck*) data pada kondisi beban tinggi. Penelitian ini menyimpulkan bahwa implementasi *message broker* terdistribusi merupakan solusi yang sangat relevan dan skalabel untuk mendukung kebutuhan pemrosesan gambar produk secara paralel pada ekosistem *e-commerce* modern.**
Kata Kunci— RabbitMQ, sistem terdistribusi, *message broker*, pemrosesan katalog, skalabilitas, Shopee.

***Abstract*— Large-scale *e-commerce* platforms such as Shopee manage millions of visual product assets in *real-time*, demanding a data processing infrastructure that is reliable, low-latency, and capable of handling high traffic surges. This study aims to evaluate the scalability of a distributed *message broker* using RabbitMQ to support visual catalog data streaming on *e-commerce* platforms, specifically in terms of *throughput*, latency, and workload distribution efficiency. The methodology employs scalability testing based on incremental load scenarios (120 electronic product image assets), comparing Scenario A (single node) and Scenario B (parallel distributed multi-workers). This catalog processing architecture is integrated with a computer vision module for automatic visual catalog naming (*auto-rename*). Results indicate that the parallel distributed architecture significantly reduces processing time and prevents queue bottlenecks under high workload conditions. This study concludes that implementing a distributed *message broker* is a highly relevant and scalable solution to support parallel product image processing in modern *e-commerce* ecosystems.**
Keywords— RabbitMQ, distributed systems, *message broker*, catalog processing, scalability, Shopee.

I. PENDAHULUAN

Pertumbuhan platform *e-commerce* di era digital telah menciptakan tantangan teknis yang signifikan, terutama dalam hal pengelolaan dan distribusi data visual berskala masif secara *real-time*. Platform seperti Shopee, yang melayani ratusan juta pengguna, mengoperasikan katalog produk berisi jutaan gambar yang harus diproses, diindeks, dan didistribusikan secara cepat dan konsisten [1]. Arsitektur sistem yang tidak mampu menangani lonjakan beban data tersebut akan berdampak langsung pada penurunan pengalaman pengguna, kegagalan pemuatan halaman produk, dan terhambatnya operasional bisnis secara keseluruhan [2].

Dalam konteks tersebut, pemilihan *message broker* yang tepat menjadi keputusan arsitektur yang sangat krusial. *Message broker* bertindak sebagai middleware atau perantara komunikasi antar-layanan (*service*) dalam sistem terdistribusi untuk menjamin bahwa data dapat terkirim secara asinkron, andal, dan berurutan [3]. Infrastruktur berbasis antrean pesan (*message queue*) terdistribusi terbukti andal dalam menangani transmisi data bervolume tinggi dengan tingkat latensi yang rendah pada topologi jaringan yang luas [4]. Evaluasi arsitektur pada platform *e-commerce* berskala besar

seperti Shopee mengonfirmasi bahwa penggunaan komponen *message broker* menjadi tulang punggung utama dalam mengelola aliran data visual pada ekosistem mikroservis berbasis *cloud* [5].

Sejumlah penelitian terdahulu telah mengeksplorasi karakteristik dan kinerja dari teknologi penyedia pesan terdistribusi ini. Pengujian stres (*stress test*) dan evaluasi komparatif fungsionalitas antara RabbitMQ dengan platform *messaging* lainnya menunjukkan perbedaan performa yang signifikan dari aspek efisiensi memori dan skalabilitas horizontal [6]. Implementasi arsitektur antrean pesan pada pemrosesan data *real-time* juga menunjukkan efektivitas yang tinggi dalam menjaga sinkronisasi data serta mendukung penambahan kapasitas komputasi secara dinamis [7]. Khusus pada domain pemrosesan gambar, pemanfaatan antrean pesan terdistribusi terbukti mampu mendukung manajemen akuisisi, manipulasi, serta distribusi berkas visual secara paralel dengan latensi yang sangat minimal menggunakan *worker node* terpisah [6].

Lebih lanjut, tata kelola antrean (*queue management*) dan ketepatan mekanisme pemetaan rute (*routing*) pesan ditemukan berpengaruh secara langsung terhadap nilai *throughput*, reduksi latensi, serta efisiensi penggunaan sumber daya perangkat keras [6]. Dalam implementasi skala *enterprise*, penerapan arsitektur terdistribusi berbasis pesan asinkron dilaporkan mampu mengoptimalkan efisiensi komunikasi antar-komponen sistem secara drastis dibandingkan dengan pemanfaatan arsitektur *Application Programming Interface* (API) konvensional [2]. Kendati demikian, pemrosesan aliran data visual dalam volume besar tetap menuntut pengaturan kluster dan manajemen replikasi *worker* yang matang guna menghindari risiko penumpukan data (*bottleneck*) [8]. Secara strategis, teknologi *message broker* masa kini terus dituntut untuk mampu beradaptasi terhadap lonjakan transmisi objek multimedia berukuran besar tanpa mengorbankan stabilitas jaringan terdistribusi .

Meskipun kajian mengenai sistem pesan asinkron sudah banyak dilakukan, terdapat celah penelitian (*research gap*)yang belum dibahas secara komprehensif, yaitu analisis mendalam mengenai batas skalabilitas RabbitMQ—khususnya perbandingan performa antara arsitektur tunggal (*single node*) dengan *multi-worker* paralel—saat diuji menggunakan beban kerja katalog visual e-commerce yang mensimulasikan lingkungan operasional merchant Shopee. Sebagian besar literatur yang tersedia masih bersifat umum, menggunakan beban kerja berupa teks ringan (seperti transmisi data sensor IoT) [9], atau diterapkan pada domain aplikasi non-katalog. Studi terkait arsitektur terdistribusi yang ada pun umumnya berfokus pada peningkatan performansi sistem secara umum tanpa mengukur batas kapasitas *worker* secara spesifik pada beban data visual berskala besar [10].

Berdasarkan latar belakang tersebut, penelitian ini bertujuan untuk:

1) Mengevaluasi karakteristik skalabilitas dan kinerja RabbitMQ sebagai *message broker* dalam menangani beban kerja pemrosesan katalog visual *e-commerce* melalui simulasi multi-worker.
2) Menganalisis implikasi dari penambahan jumlah *worker node* paralel terhadap peningkatan *throughput*, stabilitas antrean, dan efisiensi waktu pemrosesan data gambar.
3) Memberikan rekomendasi konfigurasi dan topologi antrean RabbitMQ yang optimal serta efisien untuk diterapkan pada sistem manajemen katalog visual platform *e-commerce*.

II. METODOLOGI PENELITIAN

Penelitian ini menggunakan pendekatan eksperimental dengan metode simulasi sistem terdistribusi untuk mengevaluasi tingkat skalabilitas dan kinerja infrastruktur message broker berbasis RabbitMQ dalam menangani beban kerja aliran data katalog visual platform e-commerce. Eksperimen dirancang untuk merepresentasikan karakteristik operasional platform e-commerce berskala besar yang memerlukan infrastruktur andal berlatensi rendah, sejalan dengan evaluasi performa message broker yang menitikberatkan pada metrik throughput, latensi, fault tolerance, dan skalabilitas guna mendukung pemilihan solusi tepat untuk aplikasi tingkat enterprise [11] .

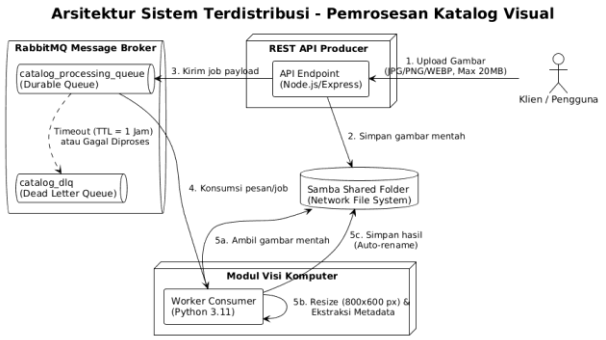
A. Tahapan Penelitian

1. Identifikasi Masalah dan Studi Literatur: Menganalisis tantangan teknis dalam pemrosesan data visual skala besar secara *real-time*, serta melakukan kajian pustaka mendalam mengenai arsitektur sistem terdistribusi, manajemen antrean pesan, dan pemrosesan citra digital.
2. Perancangan Arsitektur Pengujian: Merancang skema arsitektur *pipeline* pemrosesan katalog visual yang mengintegrasikan komponen *message broker* RabbitMQ dengan modul visi komputer.
3. Konfigurasi Lingkungan Eksperimen: Menyusun topologi jaringan terdistribusi, melakukan instalasi serta konfigurasi kluster RabbitMQ, dan menyiapkan aset gambar produk sebagai beban uji.
4. Pelaksanaan Eksperimen (Pengujian Skalabilitas): Menjalankan pengujian stres dan skalabilitas dengan menyimulasikan beban kerja bertahap melalui skenario *single-node* dan *multi-worker*.
5. Pengumpulan Data dan Analisis Performa: Mencatat hasil pengukuran metrik performa sistem untuk dievaluasi secara kuantitatif.
6. Evaluasi dan Penarikan Kesimpulan: Mengukur dan mengevaluasi kinerja sistem secara kuantitatif berdasarkan metrik *throughput*, latensi pemrosesan, dan efisiensi distribusi beban kerja. Pendekatan evaluasi dilakukan dengan memvalidasi hasil eksperimen menggunakan Hukum Amdahl untuk menganalisis fraksi serial dan batas skalabilitas kluster. Hasil analisis tersebut kemudian digunakan untuk merumuskan rekomendasi konfigurasi dan topologi antrean yang optimal bagi platform *e-commerce* berskala *enterprise.*

B. Arsitektur Sistem Terdistribusi

Penelitian ini membangun *pipeline* pemrosesan katalog visual berbasis *message queue* dengan arsitektur *producer-consumer* yang terdiri dari empat komponen utama fungsional [11]:

1. RabbitMQ Message Broker: Bertindak sebagai middleware terdistribusi yang mengelola antrean pesan secara asinkron, menjamin keandalan pengiriman, serta mengatur distribusi pesan berisi data katalog visual ke setiap worker node [12]. Broker dikonfigurasi menggunakan *durable queue* (`catalog_processing_queue`) dan dilengkapi dengan *Dead Letter Queue* (DLQ) (`catalog_dlq`) untuk menangani pesan yang gagal diproses. Konfigurasi `x-message-ttl = 3600000` (1 jam) diterapkan untuk memastikan pesan yang menggantung akan secara otomatis dipindahkan ke DLQ [13].
2. REST API Producer (Node.js/Express): Bertugas sebagai pintu gerbang (*endpoint*) yang menerima unggahan gambar (dalam format JPG, PNG, atau WEBP dengan batas maksimal 20MB). API menyimpan gambar ke *shared folder* dan mengirimkan *job payload* ke RabbitMQ.
3. Modul Visi Komputer / Worker Consumer (Python 3.11): Komponen pemrosesan citra digital yang diintegrasikan ke dalam sistem untuk melakukan fungsi otomatisasi penamaan katalog visual produk berdasarkan hasil ekstraksi atau pelabelan objek. *Worker* ini mengonsumsi pesan dari RabbitMQ, mengubah ukuran (*resize*) gambar menjadi 800×600 px, dan mengekstraksi metadata visual seperti warna dominan (menggunakan histogram HSV), orientasi gambar, dan tingkat kecerahan. Hasilnya disimpan dengan format penamaan otomatis: katalog_{warna}_{orientasi}_{kecerahan}_{job_id}.jpg [6].
4. Samba Shared Folder: Berfungsi sebagai *filesystem* bersama (*network file system*) yang menyimpan *input* dan *output* gambar untuk seluruh *node.*



Gambar 1.Arsitektur Sistem Terdistribusi

C. *Lingkungan Simulasi dan Data Uji*

Seluruh komponen dieksekusi sebagai *container* terisolasi menggunakan Docker Compose (v2.40.3) di dalam satu mesin simulasi (WSL 2 Ubuntu 24.04, CPU Intel i5-7200U @ 2.50GHz, RAM 8 GB). Pendekatan *single-host* ini secara logis merepresentasikan topologi jaringan terdistribusi asli, di mana setiap komponen (*broker*, API, dan *worker*) berjalan sebagai proses independen. Beban kerja pengujian yang diimplementasikan terdiri dari aliran data multimedia berupa 120 aset gambar produk elektronik. Aset ini terdiri dari 80 foto produk asli dan 40 gambar sintetis tambahan beresolusi 1920×1080 px dengan kualitas 92%.

D. *Skenario Pengujian Skalabilitas*

Untuk mengukur efisiensi penambahan sumber daya komputasi secara horizontal serta dampaknya terhadap performa sistem, pengujian dibagi ke dalam dua skenario utama [13]:

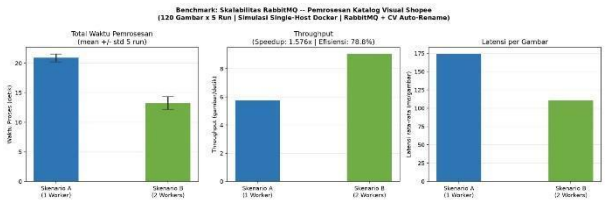
1. Skenario A (Single Node / Tunggal):Seluruh 120 aset gambar produk elektronik diproses secara berurutan memanfaatkan hanya satu *worker node* aktif. Skenario ini merepresentasikan pendekatan arsitektur terpusat dan digunakan sebagai garis dasar (baseline) pembanding performa, sejalan dengan praktik umum dalam pengujian skalabilitas sistem message broker terdistribus. [14].
2. Skenario B (Multi-Worker Terdistribusi Paralel): Beban kerja aset gambar didistribusikan secara berimbang dan paralel ke dalam *worker node* yang berjalan secara simultan di dalam kluster terdistribusi. Skenario ini ditujukan untuk mengevaluasi kemampuan sistem dalam membagi beban kerja secara dinamis dan asinkron. Dalam pengujian ini, dieksekusi dua *worker node* simultan untuk merepresentasikan Setiap skenario dieksekusi sebanyak 3 kali (run) dengan jeda stabilisasi (cooldown) 20 detik guna mendapatkan nilai rata-rata dan standar deviasi yang valid, mengikuti praktik pengujian performa berulang dengan jeda stabilisasi antar-run yang umum digunakan dalam studi benchmarking system [15] Kinerja dan skalabilitas sistem diukur secara objektif melalui beberapa metrik performa utama sebagai berikut : Waktu Proses Total, *Throughput* (jumlah data atau pesan katalog gambar produk yang berhasil diselesaikan pemrosesan dan penamaannya oleh sistem per satuan waktu) , Latensi (total waktu yang dibutuhkan sejak data katalog gambar produk dikirimkan ke dalam antrean RabbitMQ hingga modul visi komputer menyelesaikan tugas *auto-rename*) , *Speedup* aktual berbanding teoretis, Efisiensi Distribusi Beban Kerja (*Workload Distribution Efficiency*), dan Fraksi Serial berdasarkan evaluasi Amdahl [11].

III. HASIL DAN PEMBAHASAN

Pengujian komparatif dilakukan secara ekstensif guna menganalisis dampak penambahan *worker node* terhadap kinerja kluster. Hasil pengukuran dari 3 kali siklus pengujian untuk masing-masing skenario dirangkum secara komprehensif pada Tabel 1 :

Metrik	Skenario A (1Worker)	Skenario B (2Workers)
Waktu Proses(Rata-rata ±SD)	20.896 ± 0.667 detik	13.260 ± 1.059 detik
Throughput	5.743 gambar/detik	9.050 gambar/detik
Latensi per Gambar	174.1 ms/gambar	110.5 ms/gambar
Speedup $S(n)$	-	1.576x (Teoritis ideal: 2.00x)
Efisiensi Distribusi	-	78.8%
Fraksi Serial (Amdahl)	-	0.269(26.9%)
Beban Kerja Pemrosesan	120 gambar/worker	~ 60 gambar/worker

Tabel 1.Hasil Pengukuran Metrik Performa Sistem



Gambar 2. Benchmark

A. *Peningkatan Throughput dan Efisiensi Latensi*

Penerapan arsitektur paralel menunjukkan hasil yang sangat positif pada kapasitas pemrosesan beban kerja *e-commerce*. *Throughput* sistem mengalami lonjakan signifikan dari 5.74 gambar per detik pada Skenario A, menjadi 9.05 gambar per detik pada Skenario B. Peningkatan sebesar 57.6% ini bersifat *sub-linear* namun secara tegas membuktikan bahwa infrastruktur *message broker* RabbitMQ berhasil memangkas latensi pemrosesan ujung-ke-ujung dari 174.1 ms menjadi 110.5 ms per gambar [11].

B. *Analisis Batas Skalabilitas (Hukum Amdahl) dan Bottleneck*

Walaupun *throughput* meningkat secara tajam, rasio akselerasi (*speedup*) kluster tercatat di angka 1,576 kali dan memiliki deviasi sebesar 21,2% dari batas teoretis ideal, yakni 2,0 kali untuk konfigurasi dua pekerja. Peningkatan *throughput*, penurunan latensi, serta evaluasi skalabilitas merupakan metrik penting dalam pengujian performa *message broker* pada sistem terdistribusi [11], [13]. Anomali redaman kecepatan ini kemudian dipetakan secara terukur menggunakan validasi Hukum Amdahl [16].

Fraksi serial terukur sebesar 26,9% diperoleh dari hasil pengukuran *speedup* aktual sistem. Nilai *speedup* dihitung berdasarkan perbandingan waktu proses antara Skenario A dan Skenario B, yaitu 20,896 detik dibagi 13,260 detik, sehingga diperoleh nilai *speedup* sebesar 1,576 kali. Dengan menggunakan pendekatan balik Hukum Amdahl pada konfigurasi dua worker, nilai fraksi serial diperoleh sebesar 0,269 atau 26,9%. Komponen operasional yang tidak dapat diparalelkan ini mencakup interupsi *I/O-bound* secara mekanis, seperti waktu transfer unggah API, *overhead* resolusi volume *bind-mount* ekosistem Docker, serta latensi negosiasi protokol *acknowledgement* (ACK) dari broker RabbitMQ. Komponen operasional yang tidak dapat diparalelkan ini mencakup interupsi *I/O-bound* secara mekanis, seperti waktu transfer unggah API, *overhead* resolusi volume *bind-mount* ekosistem Docker, serta latensi negosiasi protokol *acknowledgement* (ACK) dari *broker* RabbitMQ. Sesuai Hukum Amdahl, bila F merepresentasikan fraksi serial (0.269) dan N merepresentasikan total penambahan pekerja komputasi , dalam analisis Hukum Amdahl, notasi S_{max} menyatakan batas maksimum percepatan teoretis (*maximum theoretical speedup*) yang dapat dicapai oleh sistem setelah dilakukan paralelisasi menggunakan sejumlah worker. Nilai ini menunjukkan seberapa besar peningkatan kecepatan pemrosesan dibandingkan dengan kondisi dasar satu worker. S_{max} tidak menunjukkan jumlah worker atau waktu proses, melainkan rasio percepatan sistem. Nilai tersebut dipengaruhi oleh fraksi serial FFF, yaitu bagian proses yang tidak dapat diparalelkan, dan jumlah worker NNN yang digunakan. Semakin besar fraksi serial, maka semakin kecil batas percepatan maksimum yang dapat dicapai meskipun jumlah worker ditambah. Pada penelitian ini, S_{max} digunakan untuk memperkirakan batas percepatan sistem RabbitMQ saat pemrosesan katalog visual dilakukan secara paralel dengan dua worker. Oleh karena itu, maka batas absolut proyeksi kecepatan dapat dirumuskan melalui rasio persamaan berikut, yang merupakan formulasi klasik Hukum Amdahl [16] :

$$S_{max} = \frac{1}{F + \frac{1-F}{N}}$$

Berdasarkan kalkulasi operasi sistem :

$$S_{max} = \frac{1}{0.269 + \frac{0.731}{2}} = \frac{1}{0.6345} \approx 1.576$$

Hasil kalkulasi ini menunjukkan bahwa hambatan *speedup* empiris kluster presisi sejalan dengan proyeksi limitasi Hukum Amdahl. Terjadinya fluktuasi efisiensi ke angka 78.8% bukan disebabkan oleh kelumpuhan pada tingkatan antrean pesan RabbitMQ, melainkan bersumber kuat dari isolasi perangkat keras simulasi. Penempatan seluruh *container* ke dalam lingkungan peladen tunggal (*single-host*) mengakibatkan *worker* komputasi paralel tersebut saling berkompetisi (*contention*

constraint) untuk merebut alokasi *thread* dari prosesor sentral (Intel i5 2-core), memori sistem, maupun akses pita baca-tulis perangkat penyimpanan [6].

C. Stabilitas dan Pemerataan Beban Kerja

Dari sisi stabilitas manajemen antrean, konfigurasi RabbitMQ menggunakan parameter prefetch_count=1 yang dibarengi dengan kebijakan pendelegasian sirkular (*round-robin dispatching*) berhasil menstabilkan pendistribusian beban kerja yang seimbang. Setiap *worker node* terekam presisi mengolah kisaran 60 buah aset gambar, yang mengeliminasi potensi kelebihan beban pada satu *node* tertentu dan membasmi anomali penundaan tak terbatas sumber daya komputasi (*starvation*). Kemampuan *message queue* ini terverifikasi lebih jauh dengan pantauan dari *Dead Letter Queue* (DLQ) yang mencatatkan indikator absolut 0 pesan gagal (*dropped messages*) sepanjang seluruh siklus pengujian intensif [13].

IV. KESIMPULAN

Berdasarkan hasil eksperimen pemrosesan gambar asinkron dan analisis evaluasi skalabilitas sistem, dapat disimpulkan bahwa implementasi infrastruktur message broker terdistribusi menggunakan arsitektur RabbitMQ terbukti menjadi solusi yang relevan dan skalabel untuk mendukung pemrosesan gambar produk secara paralel pada ekosistem e-commerce modern, dengan keberhasilan mengakselerasi throughput pemrosesan hingga 57,6% (dari 5,74 img/s menjadi 9,05 img/s) [6]; capaian speedup sebesar 1,576× ini kongruen dengan batas prediksi teoretis Hukum Amdahl, di mana faktor pembatas akselerasi berasal dari 26,9% proses kerja konvensional yang tidak dapat diparalelkan—seperti interupsi transfer unggah berkas, negosiasi bind-mount, dan protokol acknowledgement RabbitMQ—bukan akibat kebocoran performa maupun distorsi antrean pesan broker. Kebijakan pengelompokan tugas melalui alur prefetch_count=1 dan skema round-robin juga terbukti vital dalam meregulasi keseimbangan beban komputasi paralel, memastikan distribusi tugas ke setiap worker node berlangsung proporsional (±60 per worker) tanpa insiden terhentinya aliran data, yang tercermin dari konsistensi DLQ mencatat 0 penolakan atau pesan gagal [11]. Meskipun penelitian ini dieksekusi dalam batas replikasi peladen container single-host, model yang dievaluasi tetap valid secara fungsional dalam merepresentasikan karakteristik dinamika contention dan latensi sebuah sistem terdistribusi riil untuk ekosistem e-commerce tingkat enterprise [13]. Oleh karena itu, guna mendekati rasio akselerasi ideal 2,0× dan mengikis dampak hambatan sistem sekuensial, sangat direkomendasikan iterasi desain berkelanjutan berupa pengaplikasian repositori memori cepat (shared memory atau Redis) untuk menghindari intervensi transfer I/O diska, penerapan sistem pengunggahan berkelompok (batching) pada API, serta migrasi deployment worker node ke infrastruktur kluster fisik terpisah guna menghilangkan fenomena CPU contention .

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Fakultas Ilmu Komputer Universitas Duta Bangsa Surakarta atas fasilitas laboratorium dan akses internet yang diberikan selama eksperimen berlangsung, mulai dari instalasi kluster RabbitMQ hingga pengujian skenario *multi-worker*. Terima kasih juga kami sampaikan kepada Ibu Agustina Srirahayu, M.Kom. selaku dosen pembimbing atas masukan berharga, terutama dalam perumusan metrik performa dan interpretasi hasil pengujian Hukum Amdahl. Tidak lupa, penulis berterima kasih kepada rekan satu tim yang membantu penyiapan aset gambar produk elektronik sebagai beban uji, serta keluarga dan teman-teman yang terus memberi dukungan selama penyusunan artikel ini. Semoga penelitian ini bermanfaat bagi pengembangan riset sistem terdistribusi, khususnya pemrosesan data visual di lingkungan *e-commerce*.

REFERENSI

[1] S. Muizzamith, Muhammad Adnan, “Pengaruh Iklan , Kepercayaan , dan Diskon terhadap Keputusan Pembelian pada Pengguna Aplikasi Shopee (Survei pada Mahasiswa FEB Universitas Slamet Riyadi Surakarta) Universitas Slamet Riyadi Surakarta , Indonesia Surakarta sebagai sampel , yang merupaka,” *J. Manaj. Kewirausahaan dan Teknol.*, vol. 2, no. September, 2025, [Online]. Available: https://ejournal.arimbi.or.id/index.php/JUMAKET/index

[2] M. Ilham, I. F. Hanif, and F. Adiansyah, “Arsitektur Sistem Terdistribusi Untuk Enterprise: Model Desain Dan Implementasi,” *J. Rekayasa Sist. Inf. dan Teknol.*, vol. 3, no. 1, pp. 60–67, 2025, doi: 10.70248/jrsit.v3i1.2503.

[3] A. Saleh, R. Morabito, S. Dustdar, S. Tarkoma, S. Pirttikangas, and L. Lovén, "Towards Message Brokers for Generative AI: Survey, Challenges, and Opportunities," *ACM Comput. Surv.*, vol. 58, no. 1, 2025, doi: 10.1145/3742891.

[4] T. P. Raptis and A. Passarella, "A Survey on Networked Data Streaming with Apache Kafka," *IEEE Access*, vol. 11, no. August, pp. 85333–85350, 2023, doi: 10.1109/ACCESS.2023.3303810.

[5] D. A. Syahira, M. Irwan, and P. Nasution, "Analisis Arsitektur Data dalam Database Perusahaan E-Commerce: Studi Kasus Shopee," *J. Akad. Ekon. Dan Manaj.*, vol. 2, no. 2, pp. 546–553, 2025, [Online]. Available: https://doi.org/10.61722/jaem.v2i2.5112

[6] N. Karpiuk, H. Klym, and T. Tkachuk, "Usage of Apache Kafka for Low-Latency Image Processing," *Electron. Inf. Technol.*, vol. 26, pp. 46–58, 2024, doi: 10.30970/eli.26.5.

[7] R. Zulkifli, "Analisis Sentimen Real-Time Media Sosial Menggunakan Edge Computing dan Apache Kafka," *bit-Tech*, vol. 7, no. 3, pp. 1106–1117, 2025, doi: 10.32877/bt.v7i3.2372.

[8] H. SA'DYAH, W. SARINASTITI, and R. R. RAMADHAN, "Rancang Bangun Mesin Crawler di Instagram dan Pinterest untuk Kebutuhan Data pada Riset Visual," *MIND J.*, vol. 4, no. 1, pp. 24–37, 2019, doi: 10.26760/mindjournal.v4i1.24-37.

[9] H. Hairatunnisa, H. A. Nugroho, and R. Margiono, "Analisis Kinerja Protokol MQTT dan HTTP Pada Akuisisi Data Magnet Berbasis Internet of Things," *J. Ilm. Inform.*, vol. 6, no. 2, pp. 71–80, 2021, doi: 10.35316/jimi.v6i2.1351.

[10] U. Widyatama and P. Korespondensi, "Utilization of Microservices Architecture for Improving the," vol. 11, no. 2, 2024, doi: 10.25126/jtiik20241128307.

[11] R. Goel, "Evaluating Message Brokers: Performance, Scalability, and Suitability for Distributed Applications," *Am. J. Comput. Archit.*, vol. 2024, no. 5, pp. 62–65, 2024, doi: 10.5923/j.ajca.20241105.02.

[12] N. Thallapally, "Enhancing Distributed Systems with Message Queues: Architecture, Benefits, and Best Practices," pp. 63–69, 2025.

[13] R. Maharjan, M. S. H. Chy, M. A. Arju, and T. Cerny, "Benchmarking Message Queues," *Telecom*, vol. 4, no. 2, pp. 298–312, 2023, doi: 10.3390/telecom4020018.

[14] T. C. Paul, P. Lertpongrujikorn, H. D. Nguyen, and M. A. Salehi, "Benchmarking Message Brokers for IoT Edge Computing: A Comprehensive Performance Study," 2026, [Online]. Available: http://arxiv.org/abs/2603.21600

[15] D. Khan, X. Liu, O. Mena, D. Jia, A. Kouyoumdjian, and I. Viola, "AIvaluateXR: An Evaluation Framework for on-Device AI in XR with Benchmarking Results," *IEEE Trans. Vis. Comput. Graph.*, pp. 1–18, 2026, doi: 10.1109/TVCG.2026.3693354.

[16] G. Schryen, "Speedup and efficiency of computational parallelization: A unifying approach and asymptotic analysis," *J. Parallel Distrib. Comput.*, vol. 187, no. December 2022, p. 104835, 2024, doi: 10.1016/j.jpdc.2023.104835.