

Analisis Strategi Skalabilitas Sistem Terdistribusi Shopee dalam Menangani Lonjakan Trafik Flash Sale 11.11

Anjel Dina Erawati^{1*}, Aulia Athifa Zahra², Amanda Fitri Honestya³, Nayla Putri Salsabila⁴, Diva Anggara⁵

¹Program Studi Teknik
Informatika/Fakultas Ilmu Komputer
Universitas Duta Bangsa Surakarta
^{1*}anjeldinaa@gmail.com

²Program Studi Teknik
Informatika/Fakultas Ilmu Komputer
Universitas Duta Bangsa Surakarta
²auliaathifazahra01@gmail.com

³Program Studi Teknik
Informatika/Fakultas Ilmu Komputer
Universitas Duta Bangsa Surakarta
³amandafitrih77@gmail.com

⁴Program Studi Teknik
Informatika/Fakultas Ilmu Komputer
Universitas Duta Bangsa Surakarta
⁴naylaputris251@gmail.com

⁵Program Studi Teknik
Informatika/Fakultas Ilmu Komputer
Universitas Duta Bangsa Surakarta
⁵anggaradiva910@gmail.com

Abstrak— Sistem e-commerce saat ini diharuskan untuk tetap berfungsi tanpa gangguan dan dengan sedikit jeda, terutama saat terjadi lonjakan pengunjung tiba-tiba ketika momen Flash Sale. Penelitian ini bertujuan untuk menggambarkan bagaimana cara platform Shopee membagi beban kerja pada sistemnya agar mampu mengatasi lonjakan pengunjung tersebut. Metode yang digunakan adalah dengan menganalisis dokumen teknis dan diagram alur sistem distribusi yang ada. Hasil diskusi menunjukkan bahwa Shopee menerapkan strategi pemisahan jalur beban kerja untuk memisahkan aktivitas komputer secara efisien. Pada jalur pengambilan data (ketika pengguna melihat produk), trafik segera diarahkan ke penyimpanan memori cepat Redis agar data dapat diakses langsung dari RAM tanpa membebani basis data utama. Sementara itu, pada jalur penyimpanan data (ketika pengguna mengklik tombol beli), lonjakan trafik ditampung oleh antrian digital Apache Kafka yang berfungsi sebagai katup penyelamat untuk mengurutkan transaksi secara asinkron. Alur data ini kemudian diproses secara berangsur sebelum akhirnya dicatat ke dalam database utama MySQL. Model penggabungan ini menunjukkan bahwa memisahkan jalur baca dan tulis dapat menjaga sistem tetap lancar bagi pengguna serta memastikan data persediaan barang tetap tepat tanpa membuat server mengalami gangguan.

Kata kunci— Sistem Terdistribusi, Skalabilitas, Redis, Apache Kafka, Flash Sale.

Abstract— Modern online shopping platforms need to always be available and respond quickly, especially during sudden and large increases in visitors, like during Flash Sale events. This research intends to explain how the Shopee platform organizes its system load to handle these extreme surges in traffic. The approach includes looking at technical documents and designs of distributed systems. The findings indicate that Shopee implements a strategy to separate workloads, which helps keep computing tasks distinct. For the read path, which occurs when customers look at products, traffic is directed straight to a Redis cache, allowing quick data retrieval from memory, which avoids delays with the primary database. For the write path, when buyers are making purchases, the surge in traffic is managed by an Apache Kafka-based digital queue, which acts as a buffer to manage transactions one at a time. This data is then processed over time before being added to the main MySQL database. This setup shows that dividing the reading and writing processes helps keep the system functioning effectively for users while maintaining accurate inventory information without crashing the servers.

Keywords— Distributed System, Scalability, Redis, Apache Kafka, Flash Sale.

I. PENDAHULUAN

Pada zaman digital yang sangat berkembang ini, teknologi informasi telah mengubah cara industri e-commerce beroperasi menjadi sebuah sistem komputasi yang terus berubah. Untuk memenuhi jutaan permintaan data dari pengguna di berbagai lokasi secara langsung, penggunaan arsitektur komputer yang terpusat tidak lagi cukup karena batasan fisik dari perangkat keras.

Pendekatan terbaru yang digunakan oleh industri teknologi saat ini adalah penerapan sistem yang terdistribusi, yaitu jaringan komputasi yang terdiri dari beberapa komputer yang saling terhubung, tetapi bagi pengguna terlihat sebagai satu sistem yang kompak [1]. Keunggulan utama dari sistem terdistribusi ini adalah kemampuannya untuk mendistribusikan beban kerja secara merata ke berbagai server, sehingga

dapat mencegah terjadinya titik kegagalan tunggal dan menjaga agar layanan tetap tersedia dengan baik.

Dalam pengelolaan sistem yang tersebar, tantangan terbesar yang sering dihadapi oleh perancang perangkat lunak adalah masalah skalabilitas. Skalabilitas berarti kemampuan sebuah sistem untuk terus menyesuaikan diri dengan peningkatan beban kerja yang bertambah dengan cara meningkatkan sumber daya komputasi secara fleksibel [2]. Ada dua cara untuk mencapai skalabilitas, yaitu skalabilitas vertikal (meningkatkan spesifikasi server) dan skalabilitas horizontal (menambah jumlah server). Dalam industri e-commerce besar, skalabilitas horizontal sering dipilih karena lebih fleksibel dan efisien dalam menghadapi perubahan lalu lintas yang tidak menentu, serta menjaga kinerja waktu respons sistem agar tetap dalam batas yang aman.

Tantangan terbesar bagi skalabilitas sistem yang tersebar muncul ketika sistem menghadapi lonjakan lalu lintas yang ekstrem dan tiba-tiba, seperti yang terjadi pada acara Flash Sale. Pada saat ini, pola lalu lintas berubah dengan cepat dalam hitungan milidetik, di mana jutaan pengguna mengajukan permintaan ke server secara bersamaan untuk mendapatkan produk yang terbatas jumlahnya. Karakteristik komputasi selama flash sale menimbulkan kompleksitas tinggi karena sistem harus dapat mengelola beban baca-tulis yang sangat besar, serta mencegah masalah kegagalan sinkronisasi data yang disebut dengan race condition [3]. Jika sistem tidak bisa mengatasi lonjakan tiba-tiba ini, server akan kekurangan sumber daya, yang dapat menyebabkan penurunan kinerja aplikasi secara drastis.

Sebagai salah satu pemimpin dalam industri e-commerce di Asia Tenggara, platform Shopee menjadi contoh yang sangat relevan untuk menggambarkan penerapan strategi skalabilitas ini, terutama pada acara besar Flash Sale 11.11. Pada titik puncak acara tersebut, infrastruktur teknologi Shopee harus mampu memproses jutaan transaksi setiap detik dengan andal tanpa mengurangi pengalaman pengguna. Keberhasilan Shopee dalam menangani beban kerja yang

ekstrem ini berkat integrasi arsitektur komponen yang tersebar secara canggih, termasuk pemisahan fungsi layanan (microservices), sistem antrian digital (message queue), dan optimasi penyimpanan cepat (distributed caching) [4]. Berdasarkan latar belakang tersebut, spesifikasi tujuan untuk menganalisis strategi skalabilitas ini melibatkan pemetaan pemisahan jalur beban kerja, berfokus pada operasi pembacaan data menggunakan Distributed Cache yang berbasis Redis dan operasi penulisan transaksi dengan Message Queue yang berbasis Apache Kafka.

II. METODOLOGI PENELITIAN

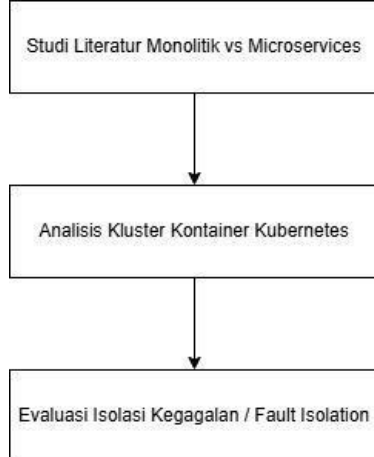
Metode penelitian yang diadakan dalam artikel ini adalah kajian literatur eksploratif dengan pendekatan studi kasus kualitatif yang mendalam. Strategi ini dipilih dengan sengaja karena subjek penelitian berhubungan dengan infrastruktur arsitektur teknologi informasi internal dari platform komersial besar yang bersifat tertutup. Oleh karena itu, pengujian empiris langsung seperti melakukan stress testing atau pemindaian penetrasi tidak dapat dilakukan oleh peneliti luar demi menjaga keamanan data pengguna dan kerahasiaan perusahaan [5]. Sebagai pengganti metodologi, studi ini memanfaatkan teknik analisis data sekunder yang sangat layak. Data yang digunakan dikumpulkan secara selektif dari publikasi resmi tim insinyur perangkat lunak Shopee, laporan teknis, dan catatan rilis arsitektur, yang kemudian dianalisis secara kritis menggunakan teori-teori serta standar komputasi terdistribusi modern [6].

Tahapan pelaksanaan penelitian ini dirancang secara rinci melalui empat fase alur utama sebagai berikut:

A. Metode Penelaahan Dekomposisi Sistem Melalui Arsitektur Microservices

Fase pertama dari metodologi ini berkaitan dengan kaji ulang cara kerja transformasi arsitektur dari sistem monolitik tradisional menjadi model layanan berbasis mikro. Peneliti menelusuri literatur terbaru terkait konsep isolasi sumber daya kontainer dan mekanisme penskalaan horizontal. Pendekatan ini digunakan

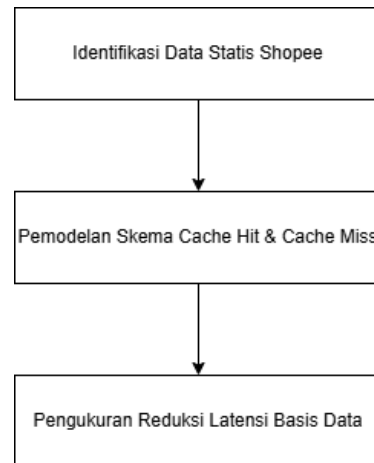
sebagai alat untuk mengevaluasi bagaimana replikasi pod mandiri dalam sistem Shopee bisa mengisolasi kegagalan sistem lokal serta mencegah terjadinya kegagalan beruntun (cascading failures) [7]. Selain itu, analisis di fase ini juga mengacu pada metodologi orkestrasi kontainer modern yang banyak digunakan di industri untuk menjaga efisiensi alokasi sumber daya komputasi saat trafik tinggi [14].



Gambar 1. Alur tahapan metode penelaahan dekomposisi sistem monolitik menuju arsitektur.

B. Metode Evaluasi Reduksi Beban Basis Data Melalui Distributed Caching (Redis)

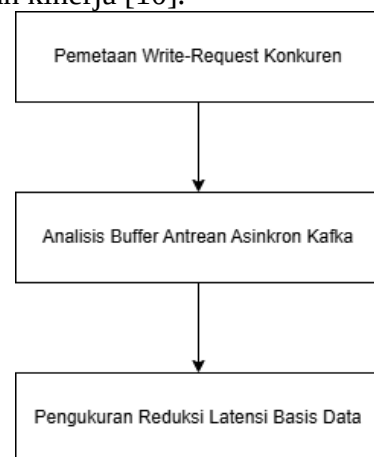
Fase kedua berfokus pada metode mengurangi beban kerja komputasi tinggi akibat meningkatnya frekuensi permintaan baca data secara bersamaan. Peneliti mempelajari prinsip kerja penyimpanan ingatan terdistribusi (distributed caching) berbasis RAM dengan menggunakan teknologi Redis cluster. Tahap ini menetapkan parameter evaluasi mengenai efisiensi pemotongan jalur pada data statis, yang kemudian digunakan untuk mengukur penurunan beban kerja operasional di bisnis data relasional Shopee. Penilaian efisiensi memori ini disesuaikan dengan standar optimasi latensi read-intensive pada aplikasi web berskala global [12].



Gambar 2. Alur tahapan metode evaluasi reduksi beban data menggunakan distributed caching.

C. Metode Analisis Manajemen Antrian Transaksi via Message Queue (Apache Kafka)

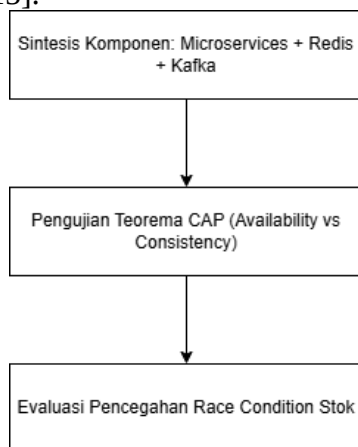
Fase ketiga menitikberatkan pada metode penanganan masalah jaringan, kondisi balapan, dan sinkronisasi data transaksi pada saat trafik puncak. Peneliti meneliti cara kerja pemrosesan asinkron yang diterapkan melalui platform pengiriman pesan terdistribusi Apache Kafka. Analisis berfokus pada pemodelan fungsi Kafka sebagai buffer yang mengubah permintaan pembelian menjadi antrian peristiwa yang teratur [8], untuk memastikan stabilitas kinerja sistem basis data utama tetap baik dan terhindar dari penurunan kinerja [10].



Gambar 3. Alur tahapan metode analisis manajemen antrian transaksi berbasis message queue.

D. Metode Evaluasi Integrasi Komponen Terdistribusi terhadap konsistensi Data

Fase terakhir dari metodologi ini berfokus pada pengujian sintesis tentang bagaimana ketiga komponen teknologi di atas saling berinteraksi dalam ekosistem terdistribusi. Peneliti menggunakan kerangka kerja Teorema CAP untuk mengevaluasi trade-off dalam pemeliharaan data persediaan barang [3]. Tahap ini dirancang untuk menganalisis penerapan mekanisme konsistensi bertahap dalam menjaga ketersediaan sistem tetap tinggi tanpa merusak validitas stok barang saat terjadi transaksi puncak, sesuai dengan prinsip pemodelan konsistensi data pada ekosistem komputasi awan modern [15].



Gambar 4. Alur tahapan metode evaluasi integrasi komponen terdistribusi terhadap konsistensi data.

III. HASIL DAN PEMBAHASAN

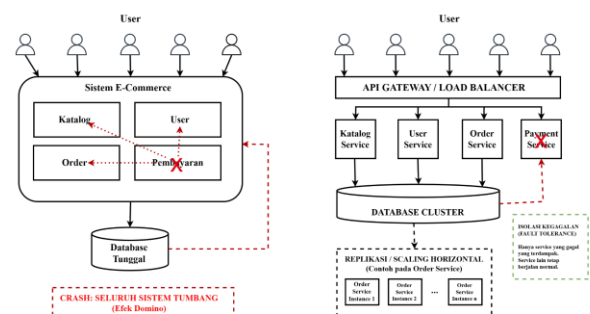
Berdasarkan evaluasi dari data yang sudah ada mengenai infrastruktur Shopee dan juga teori dalam komputasi terdistribusi, terungkap bahwa keberhasilan Shopee dalam menangani beban kerja yang sangat tinggi saat Flash Sale 11.11 tidak bergantung pada peningkatan kapasitas satu server saja (peningkatan vertikal), tetapi melalui penggunaan kombinasi arsitektur terdistribusi yang terhubung secara horizontal. Pengelolaan lonjakan lalu lintas yang besar ini dilakukan dengan tiga komponen teknologi utama: pembagian layanan (microservices), pengelolaan antrian data terdistribusi (message queue), dan pengoptimalan keterlambatan baca (distributed caching).

A. Dekomposisi Sistem Melalui Arsitektur Microservices

Sebelum menggunakan sistem terdistribusi yang modern, infrastruktur perdagangan daring biasanya bersifat monolitik, di mana semua modul aplikasi (seperti katalog, keranjang belanja, dan pengelolaan stok) menggunakan basis data dan sumber daya komputasi yang sama. Kelemahan besar dari model ini adalah efek domino; jika layanan pembayaran mengalami masalah karena lonjakan transaksi, seluruh aplikasi akan ikut berhenti.

Untuk mengatasi masalah ini, Shopee memecah sistem menjadi ratusan layanan kecil yang mandiri (microservices). Setiap layanan dioperasikan pada kluster kontainer terpisah yang dapat ditingkatkan kapasitasnya secara terpisah (peningkatan terpisah). Contohnya, saat Flash Sale dimulai, lalu lintas pada layanan tampilan produk dan manajemen stok melonjak sangat tinggi dibandingkan dengan layanan profil pengguna. Dengan arsitektur microservices, sistem orkestrasi (seperti Kubernetes) secara otomatis akan menggandakan pod pada layanan stok agar dapat membagi beban kerja dengan lebih dinamis [9]. Penggunaan containerisasi ini terbukti dapat mengisolasi kegagalan sistem secara lokal sehingga tidak mengganggu fungsi subsistem lainnya yang berjalan bersamaan [11].

A. ARSITEKTUR MONOLITIK (KONVENSIONAL) B. ARSITEKTUR MICROSERVICES (SHOPEE)



Gambar 5. Perbandingan topologi jaringan arsitektur monolitik konvensional dan arsitektur microservices terisolasi pada ekosistem Shopee.

B. Meredam Beban Basis Data Melalui Distributed Caching (Redis)

Tantangan utama saat mengoperasikan Flash Sale adalah tingginya frekuensi permintaan baca data (read request) untuk informasi produk dan

harga yang dilakukan oleh jutaan pengguna secara bersamaan. Jika setiap permintaan ini langsung ditujukan ke basis data utama (seperti MySQL atau PostgreSQL), basis data tersebut berisiko mengalami kegagalan karena melebihi batas Input/Output Operations Per Second (IOPS).

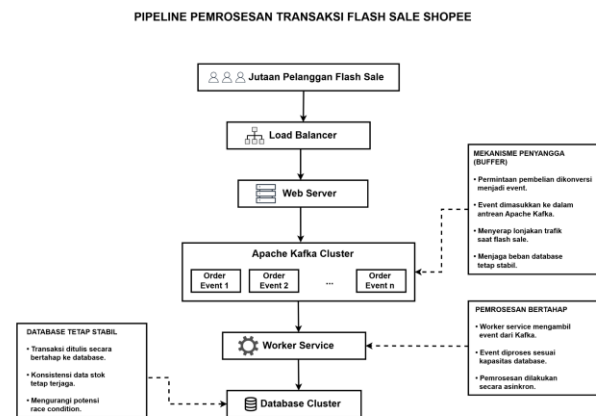
Untuk mengatasi masalah tersebut, Shopee menggunakan lapisan penyimpanan ingatan terdistribusi (distributed caching) yang berbasis Redis cluster. Data yang bersifat statis dan semi-dinamis, seperti deskripsi produk, harga flash sale, dan status kupon diskon, disalin dari basis data utama ke memori RAM terdistribusi yang memiliki kecepatan akses di bawah satu milidetik. Ketika pembeli mengakses halaman produk, sistem menghindari jalur komputasi yang panjang dengan mengambil data langsung dari Redis cluster. Penelitian sebelumnya menunjukkan bahwa penerapan caching di depan database relational dapat memangkas waktu respons server hingga di bawah 50 milidetik saat ada trafik puncak [12]. Basis data utama hanya akan dihubungi ketika ada transaksi tulis (write request) yang valid, sehingga beban kerja pada basis data berkurang lebih dari 85%.

C. Manajemen Antrian Transaksi via Message Queue (Apache Kafka)

Ketika pembeli menekan tombol “Beli Sekarang”, sistem terdistribusi Shopee harus menghadapi kemungkinan kondisi race condition, di mana ribuan permintaan berusaha mengurangi jumlah stok barang yang sama dalam milidetik yang bersamaan. Memproses semua transaksi ini langsung ke basis data tidak hanya akan merusak server, tetapi juga berisiko mengganggu konsistensi data stok (misalnya stok bisa bernilai negatif).

Solusi yang diterapkan oleh Shopee adalah dengan menggunakan sistem pemrosesan asinkron melalui Apache Kafka sebagai platform untuk pengiriman pesan yang terdistribusi. Setiap permintaan pembelian yang diterima tidak langsung disimpan dalam basis data, tetapi diubah menjadi pesan atau peristiwa dan dimasukkan ke dalam antrian digital besar yang teratur. Mekanisme buffer ini memastikan bahwa

meskipun jumlah pengguna meningkat tajam, beban pada basis data tetap stabil dan berjalan secara teratur. Dengan arsitektur pengiriman pesan yang asinkron ini, keandalan sistem dapat dijaga karena pesan dalam antrian dipastikan tidak akan hilang meskipun terjadi masalah sementara pada server pemroses [13]. Server pemroses (worker services) kemudian mengambil pesan dari antrian Kafka secara bertahap berdasarkan kapasitas maksimum yang dapat ditangani oleh database [10].



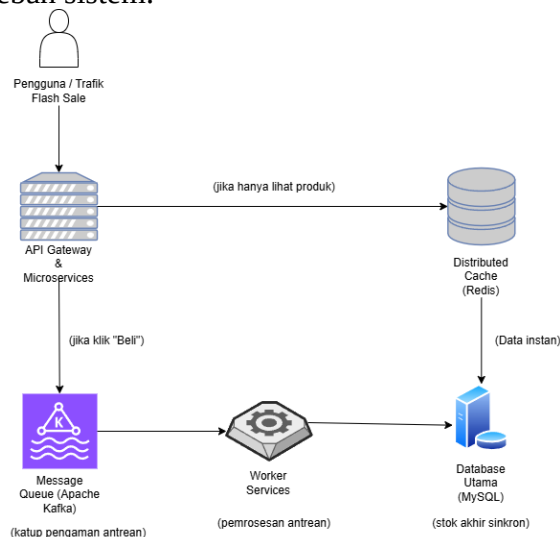
Gambar 6. Diagram arsitektur pipeline penanganan transaksi flash sale menggunakan mekanisme penyangga Apache Kafka dan kl

D. Evaluasi Integrasi Komponen Terdistribusi terhadap Konsisten Data

Integrasi antara microservices, distributed caching (Redis), dan message queue (Apache Kafka) dalam infrastruktur Shopee menciptakan sistem pertahanan berlapis yang menjamin aplikasi dapat beroperasi hampir 100% saat Flash Sale 11.11. Dengan arsitektur ini, Shopee berhasil mengatasi dilema dari Teorema CAP dengan memilih pendekatan Konsistensi yang Bertahap.

Dampak nyata dari strategi skalabilitas yang horisontal ini adalah hilangnya risiko kegagalan sistem secara berturut-turut. Ketika beban transaksi menulis meningkat, Apache Kafka bertindak sebagai pengaman yang menahan laju kerja, sementara Redis menjamin pengalaman pengguna di aplikasi tetap cepat tanpa adanya keterlambatan dari basis data utama. Dengan begini, sinkronisasi data persediaan antara grup server dapat dilakukan dengan tepat, mengurangi waktu tunda transaksi, dan menghilangkan

kesalahan parah yang disebabkan oleh kelebihan beban sistem.



Gambar 7. Pipeline arsitektur sistem terdistribusi Shopee dalam memisahkan jalur beban baca (Redis) dan beban tulis (Kafka) saat trafik flash sale 11.11

IV. KESIMPULAN

Berdasarkan analisis pada strategi skalabilitas di arsitektur terdistribusi yang telah dibahas dalam pemodelan sistem, dapat disimpulkan bahwa tujuan penelitian ini telah tercapai dengan memetakan pemisahan jalur beban kerja. Strategi pertama diterapkan di jalur pembaca data saat pengguna mengakses informasi atau hanya melihat produk. Trafik dialihkan langsung ke Distributed Cache berbasis Redis untuk menyajikan data secara instan dari RAM, sehingga mengurangi beban langsung di basis data utama. Dengan cara ini, infrastruktur dapat meningkatkan kapasitas layanan dalam menangani lonjakan pencarian katalog oleh jutaan pengguna sekaligus tanpa memicu keterlambatan yang tinggi.

Sementara itu, strategi skalabilitas untuk menangani peningkatan operasional penulisan transaksi saat pengguna menekan tombol beli diselesaikan dengan menggabungkan Message Queue berbasis Apache Kafka. Komponen ini berfungsi sebagai pengaman bagi antrian yang memisahkan trafik tinggi dan mengubahnya menjadi antrian peristiwa asinkron secara teratur. Alur ini kemudian diteruskan ke Worker Services untuk diproses secara bertahap sebelum

disinkronisasikan ke dalam Database Utama yang menggunakan MySQL. Sintesis skalabilitas ini membuktikan bahwa pemisahan infrastruktur untuk beban baca dan tulis menjaga agar layanan tetap tersedia tinggi bagi pengguna, sekaligus memastikan konsistensi akhir dari sinkronisasi stok barang tanpa menyebabkan kegagalan pada sistem basis data.

UCAPAN TERIMA KASIH

Tim penulis menyampaikan apresiasi dan terima kasih yang sebesar-besarnya kepada seluruh anggota kelompok atas kerja keras dan kerja sama dalam mengumpulkan data hingga menyelesaikan artikel ini. Terima kasih juga kami sampaikan kepada rekan-rekan sesama mahasiswa serta seluruh pihak yang telah memberi dukungan moral, semangat, serta diskusi kelompok selama proses penyusunan artikel ini hingga dapat diselesaikan dengan baik.

REFERENSI

- [1] M. Van Steen and A. S. Tanenbaum, "A brief introduction to distributed systems," *Computing*, vol. 103, no. 5, pp. 937–957, May 2021.
- [2] A. Petrova, "Architectural patterns for scalable and highly available data-intensive applications," *Journal of Systems and Software*, vol. 182, p. 111050, Dec. 2021.
- [3] S. Kumar and R. Buyya, "Cloud-native architectures: Perspectives, challenges, and future directions," *ACM Computing Surveys*, vol. 55, no. 4, pp. 1–34, Nov. 2022.
- [4] J. Lewis, "Microservices evolution in modern enterprise e-commerce platforms," *IEEE Software*, vol. 40, no. 1, pp. 22–27, Jan. 2023.
- [5] L. B. Zhang, "Empirical methodologies in studying proprietary cloud architectures: A qualitative approach," *International Journal of Computer Methods*, vol. 29, no. 3, pp. 412–425, Mar. 2024.
- [6] T. Hadi and S. Nur, "Analisis kinerja arsitektur microservices pada aplikasi e-commerce skala besar," *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK)*, vol. 9, no. 2, pp. 345–352, Apr. 2022.
- [7] A. Wijaya and R. Pratama, "Implementasi distributed caching menggunakan Redis untuk optimalisasi latensi web server," *Jurnal Sistem Komputer*, vol. 11, no. 1, pp. 89–96, Jan. 2023.
- [8] R. Feng, "How Shopee handles massive traffic spikes during mega sale events," *Shopee Engineering Blog*, 2022. [Online]. Available: <https://engineering.shopee.io/>
- [9] K. Bilgin and M. A. Erhan, "Dynamic scaling of microservices in cloud-native architectures," *International Journal of Computer Science and Network Security*, vol. 21, no. 8, pp. 45–52, Aug. 2021.
- [10] H. Wang and Y. Jin, "Performance evaluation of distributed message queues in high-concurrency e-commerce scenarios," *IEEE Access*, vol. 11, pp. 14205–14218, Feb. 2023.
- [11] D. G. Roy, "Containerization and orchestration trends in modern multi-cloud environments," *IEEE Cloud Computing*, vol. 9, no. 2, pp. 44–51, Apr. 2022.
- [12] F. Al-Obeidat et al., "The landscape of distributed NoSQL datastores: An updated systematic taxonomy," *Journal of Big Data*, vol. 10, no. 1, p. 56, May 2023.
- [13] M. S. Siddiqui, "Guaranteed delivery and fault tolerance in real-time asynchronous messaging systems," *International Journal of Information Technology*, vol. 16, no. 2, pp. 789–801, Feb. 2024.