Buletin Sistem Informasi dan Teknologi Islam
Vol 7, No 2 (2026), pp 64-70
ISSN: 2721-0901
DOI: https://dx.doi.org/10.33096/busiti.v7i2.3218

# Analisis Performa REST API dan GraphQL pada Aplikasi Berita Digital di Indonesia

## *Performance Analysis of REST API and GraphQL in Digital News Applications in Indonesia*

Ahmad Muzakki Aley [a,1]*, Ade Irma Purnama Sari [a,2], Denni Pratama [b,3], Nining Rahaningsih [c,4], dan Willy Prihartono [c,5]

[a]*Program Studi Teknik Informatika, STMIK IKMI Cirebon, Cirebon, Indonesia*
[b]*Program Studi Manajemen Informatika, STMIK IKMI Cirebon, Cirebon, Indonesia*
[c]*Program Studi Komputerisasi Akuntansi, STMIK IKMI Cirebon, Cirebon, Indonesia*
[1]*ahmadmuzakkialey@gmail.com;* [2]*irma2974@yahoo.com;* [3]*pratamadenni@gmail.com;* [4]*niningr157@yahoo.co.id;* [5]*willyprihartono@gmail.com*
**Corresponding author*

**Informasi Artikel**

Diserahkan : 27 November 2025
Diterima : 12 Februari 2026
Direvisi : 5 Maret 2026
Diterbitkan : 1 Mei 2026

**Kata Kunci:**
REST API
GraphQL
Efisiensi Data
Web Service
Arsitektur Microservices

**ABSTRAK**

Kebutuhan aplikasi berita digital terhadap mekanisme pengambilan data yang cepat dan efisien semakin meningkat. Arsitektur REST API yang umum digunakan masih menghadapi kendala *over-fetching* dan *under-fetching*, terutama pada data bersarang. Penelitian ini bertujuan mengimplementasikan dan membandingkan performa REST API dan GraphQL dalam pengambilan data bersarang pada aplikasi berita digital. Metode yang digunakan adalah eksperimen komparatif menggunakan *Indonesian News Dataset* yang telah direstrukturisasi. Implementasi dilakukan menggunakan Node.js dengan Express.js untuk REST API dan Apollo Server untuk GraphQL. Pengujian performa menggunakan Postman (*baseline*) dan Apache JMeter (*load test*) dengan parameter *latency*, *payload size*, jumlah HTTP *request*, dan *throughput*. Hasil penelitian menunjukkan GraphQL secara signifikan lebih unggul. Rata-rata *latency* GraphQL 29 ms, jauh lebih cepat dibanding REST API 2084 ms. Ukuran *payload* GraphQL rata-rata 253 *byte*, lebih efisien 83% dari REST API (1483 byte). *Throughput* GraphQL mencapai 1.455,39 *request*/detik, melampaui REST API yang hanya 21,77 *request*/detik. GraphQL hanya memerlukan satu HTTP *request* untuk data bersarang, sedangkan REST API memerlukan beberapa *request* terpisah. Disimpulkan bahwa GraphQL merupakan pendekatan yang lebih efisien dan direkomendasikan untuk aplikasi berita digital dengan struktur data kompleks.

***Keywords*:**
*REST API*
*GraphQL*
*Data Efficiency*
*Web Service*
*Microservices Architecture*

***ABSTRACT***

*The demand for fast and efficient data retrieval mechanisms in digital news applications is increasing. The commonly used REST API architecture still faces the challenges of over-fetching and under-fetching, especially with nested data. This study aims to implement and compare the performance of REST API and GraphQL in retrieving nested data in digital news applications. The method used is a comparative experiment using a restructured Indonesian News Dataset. The implementation was carried out using Node.js with Express.js for REST API and Apollo Server for GraphQL. Performance testing was conducted using Postman (baseline) and Apache JMeter (load test) with parameters of latency, payload size, number of HTTP requests, and throughput. The results show that GraphQL is significantly superior. The average latency of GraphQL was 29 ms, much faster than the REST API at 2084 ms. The average payload size of GraphQL was 253 bytes, 83% more efficient than the REST API (1483 bytes). GraphQL's throughput reached 1,455.39 requests/second, surpassing the REST API's 21.77 requests/second. GraphQL only requires one HTTP request for nested data, while the REST API requires several separate requests. It is concluded that GraphQL is a more efficient approach and is recommended for digital news applications with complex data structures.*

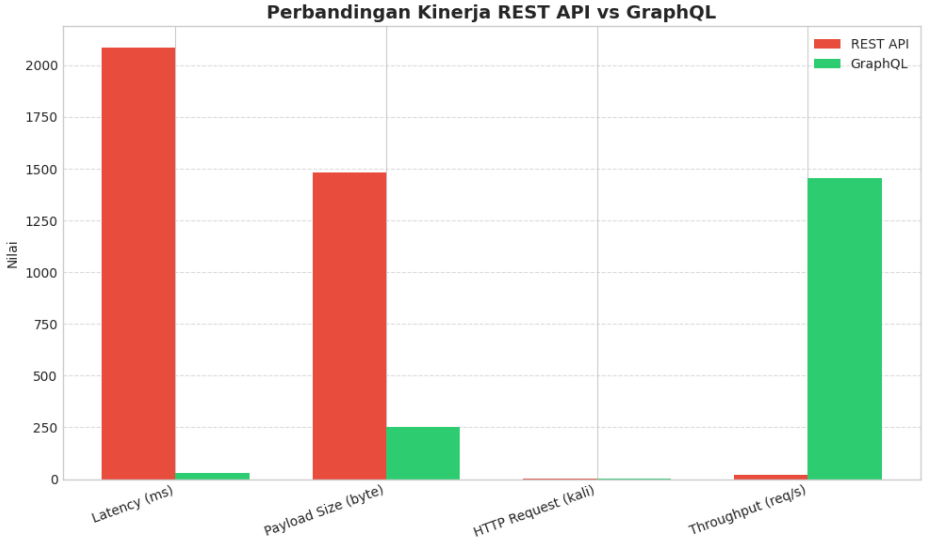


## I. Pendahuluan

Perkembangan teknologi internet yang pesat telah mentransformasi cara pengguna mengakses informasi.

Pengguna internet di Indonesia mencapai lebih dari 221 juta yang mendorong pertumbuhan aplikasi berita digital yang menuntut akses informasi cepat, akurat, dan responsif [1]. *Application Programming Interface* (API) menjadi komponen kunci dalam komunikasi klien-server. Selama dua dekade, *Representational State Transfer* (REST) API menjadi standar dominan karena kesederhanaan dan skalabilitasnya [2]. Namun, REST API memiliki keterbatasan fundamental, seperti *over-fetching* (klien menerima data berlebih) dan *under-fetching* (satu permintaan tidak cukup, butuh beberapa panggilan) [3], terutama ketika menangani struktur data bersarang seperti pada aplikasi berita (artikel, penulis, kategori, komentar). Kondisi ini meningkatkan latensi dan beban jaringan [4].

Sebagai alternatif, Facebook memperkenalkan GraphQL pada 2015 [5]. GraphQL memungkinkan klien menentukan secara presisi data yang dibutuhkan dalam satu permintaan, mengurangi *over-fetching* dan *under-fetching* [6] Perbedaan mendasar arsitektur komunikasi kedua pendekatan ini diilustrasikan pada Gambar 1.




Gambar 1. Perbandingan Arsitektur Komunikasi REST API dan GraphQL

Berbagai penelitian menunjukkan keunggulan GraphQL dalam efisiensi *payload*, latensi lebih rendah, dan *throughput* lebih tinggi pada data kompleks [[7], [8]]. Penelitian terbaru oleh Niswar et al. [9] tentang evaluasi performa komunikasi microservices juga mengonfirmasi keunggulan GraphQL dalam efisiensi data.

Meski banyak penelitian membandingkan REST dan GraphQL, sebagian besar berfokus pada konteks umum dan belum spesifik mengkaji performa pada struktur data bersarang aplikasi berita digital Indonesia. Penelitian ini mengimplementasikan dan membandingkan performa REST API dan GraphQL pada pengambilan data bersarang menggunakan indonesian_news_dataset. Parameter kinerja yang dianalisis meliputi *latency*, *payload size*, jumlah HTTP *request*, dan *throughput*. Hasil penelitian diharapkan menjadi rekomendasi bagi pengembang dalam memilih arsitektur API yang optimal untuk aplikasi berita digital.

Penelitian ini memberikan kontribusi pada tiga aspek utama. Pertama, menyajikan evaluasi komparatif REST API dan GraphQL secara khusus pada konteks aplikasi berita digital Indonesia yang memiliki karakteristik data bersarang kompleks. Kedua, menggunakan Indonesian News Dataset yang direstrukturisasi untuk mensimulasikan skenario dunia nyata. Ketiga, menyediakan analisis performa berbasis pengujian baseline dan load testing yang dapat menjadi referensi praktis bagi pengembang sistem informasi berita digital di Indonesia.

## II. Metode

Penelitian ini menggunakan *comparative experimental design* untuk membandingkan kinerja REST API dan GraphQL dalam kondisi terkendali. Eksperimen dilakukan melalui beberapa tahap utama.

### *A. Spesifikasi Lingkungan Pengujian*

Untuk memastikan validitas data, pengujian dilakukan pada lingkungan terkendali dengan spesifikasi sebagai berikut:

- Sistem Operasi: Windows 11 / Linux Ubuntu 22.04 LTS.
- Runtime: Node.js v20.x (Long Term Support).
- Framework API: Express.js v4.18 untuk REST API dan Apollo Server v4.x untuk GraphQL.

- Database Simulation: File JSON lokal (*articles.json, authors.json, categories.json*) untuk mengeliminasi variabel latensi *database eksternal*.
- Koneksi Jaringan: *Localhost loopback* (127.0.0.1) untuk mengukur performa murni arsitektur tanpa gangguan sinyal internet.

*B. Detail Dataset dan Sampel*

- Ukuran Dataset: Dataset terdiri dari 12.430 artikel, 1.215 penulis, dan 18 kategori. Setiap artikel memiliki minimal 5 komentar simulasi sehingga total data komentar berjumlah 62.150 entri.
- Data Bersarang: Setiap artikel dihubungkan dengan 1 *author_id*, 1 *category_id*, dan minimal 5 data komentar simulasi (*mock data*).
- Ukuran Sampel (Load Test): Pengujian beban menggunakan 10 *concurrent users* dengan total sampel sebanyak 43.899 permintaan untuk GraphQL dan 697 permintaan untuk REST API guna mencapai titik jenuh (*saturation point*) server.

*C. Pertimbangan Etis*

Dataset yang digunakan dalam penelitian ini merupakan dataset publik yang tersedia secara terbuka melalui platform Kaggle dan tidak mengandung data pribadi sensitif. Seluruh data artikel, penulis, dan kategori bersifat umum serta tidak memuat informasi identitas individu yang dilindungi. Data komentar yang digunakan dalam struktur bersarang merupakan data simulasi (*mock data*) yang dibuat untuk keperluan eksperimen, sehingga tidak melibatkan partisipan manusia secara langsung. Oleh karena itu, penelitian ini tidak memerlukan persetujuan etik khusus.

*D. Keterbatasan Penelitian*

Penelitian ini memiliki beberapa keterbatasan. Pertama, pengujian dilakukan pada lingkungan lokal (localhost) sehingga belum sepenuhnya merepresentasikan kondisi server produksi (*production server*) dengan infrastruktur cloud atau distribusi jaringan luas. Kedua, penelitian ini tidak mengukur konsumsi sumber daya sistem seperti penggunaan CPU dan memori yang dapat memberikan gambaran lebih komprehensif terhadap efisiensi arsitektur. Ketiga, meskipun dataset yang digunakan berukuran besar, struktur data masih dalam skala simulasi dan belum mencerminkan kompleksitas penuh sistem berita digital tingkat enterprise. Penelitian selanjutnya dapat menguji performa pada lingkungan cloud serta menambahkan analisis konsumsi sumber daya sistem.

*E. Persiapan Dataset*

Dataset yang digunakan berasal dari indonesian_news_dataset di platform Kaggle. Dataset awal dalam format CSV dikonversi ke JSON. Selanjutnya, data direstrukturisasi menggunakan skrip *Node.js* (*restructure.js*) untuk membentuk struktur data bersarang yang merepresentasikan relasi dunia nyata. Entitas utama yang dibentuk adalah:

1. Articles: berisi artikel dengan *author_id* dan *category_id* sebagai *foreign key*, serta *array comments*.
2. Authors: berisi daftar penulis.
3. Categories: berisi daftar kategori.

Data kosong pada field penulis dan kategori ditangani dengan nilai default. Komentar ditambahkan secara simulasi (*mock data*) untuk melengkapi struktur bersarang. Hasil restrukturisasi disimpan dalam file JSON terpisah (*articles.json, authors.json, categories.json*).

*F. Skenario Pengujian*

Tiga skenario pengujian dirancang dengan kompleksitas berbeda:

1. Skenario 1 (Sederhana): Pengambilan daftar artikel (judul dan ID).
2. Skenario 2 (Menengah): Pengambilan detail satu artikel lengkap.
3. Skenario 3 (Kompleks/Bersarang): Pengambilan artikel beserta data penulis, kategori, dan komentar.

*G. Pelaksanaan Pengujian dan Instrumen*

Pengujian dilakukan dalam dua fase:

1. Pengujian *Baseline*: Menggunakan Postman untuk mengukur *response time* (*latency*) dan *response size* (*payload size*) pada kondisi beban rendah (permintaan tunggal). Setiap skenario dijalankan 5 kali dan dihitung rata-ratanya.

2. Pengujian Beban (*Load Testing*): Menggunakan Apache JMeter untuk mengukur *throughput, average latency,* dan stabilitas *under load*. Konfigurasi JMeter: *Thread Group* (10 pengguna, *ramp-up period* 1 detik), dijalankan hingga total sampel signifikan tercapai. *Listeners* (*Summary Report, Aggregate Report*) digunakan untuk mengumpulkan data.

### *H. Pelaksanaan Pengujian dan Instrumen*

Data kuantitatif dari Postman dan JMeter dikumpulkan dan dianalisis secara deskriptif komparatif. Parameter yang dibandingkan adalah:

1. Rata-rata Latency (ms)
2. Rata-rata Payload Size (byte)
3. Jumlah HTTP Request
4. Throughput (request/detik)

Hasil analisis divisualisasikan dalam bentuk tabel dan grafik untuk memudahkan interpretasi.

## III. Hasil dan Pembahasan

Kedua API berhasil diimplementasikan dan dapat mengembalikan data sesuai skenario. Hasil pengujian baseline (Postman) dan load test (JMeter) menunjukkan perbedaan performa yang signifikan.

### *A. Hasil Pengujian Baseline (Postman)*

Tabel 1 merangkum hasil pengujian baseline (postman) untuk ketiga skenario.

Table 1. Rangkuman Hasil Pengujian Baseline dengan Postman

| Skenario Pengujian | Metode | Response Time (ms) | Response Size (KB) | Keterangan |
|---|---|---|---|---|
| Pengambilan Data Sederhana (List) | REST API | 267,44 | 91,91 | REST stabil pada data datar |
| | GraphQL | 103,32 | 91,92 | GraphQL lebih cepat (-61%) |
| Pengambilan Data Detail (1 Artikel) | REST API | 100,98 | 3,49 | Terjadi *over-fetching* |
| | GraphQL | 14,89 | 3,07 | GraphQL jauh lebih cepat |
| Pengambilan Data Bersarang (*Nested*) | REST API | 326,34 (Total 3 *Request*) | 3,81 (Total) | Membutuhkan 3 request terpisah (*under-fetching*) |
| | GraphQL | 120,98 | 0,55 | Hanya 1 *request*, *payload* lebih kecil |

Pada skenario sederhana, GraphQL 61% lebih cepat. Pada skenario bersarang, REST API memerlukan tiga permintaan terpisah (total latency 326,34 ms), sementara GraphQL hanya satu permintaan (120,98 ms). Ukuran payload GraphQL untuk data bersarang juga jauh lebih kecil (0,55 KB vs 3,81 KB), menunjukkan efisiensi dalam menghindari *over-fetching*.

### *B. Hasil Pengujian Beban (JMeter)*

Pengujian beban dengan JMeter mensimulasikan kondisi tekanan tinggi. REST API hanya dapat memproses 697 sampel, sedangkan GraphQL berhasil memproses 43.899 sampel. Hasil performa keseluruhan dirangkum dalam Tabel 2.

Table 2. Rangkuman Hasil Pengujian Beban dengan JMeter

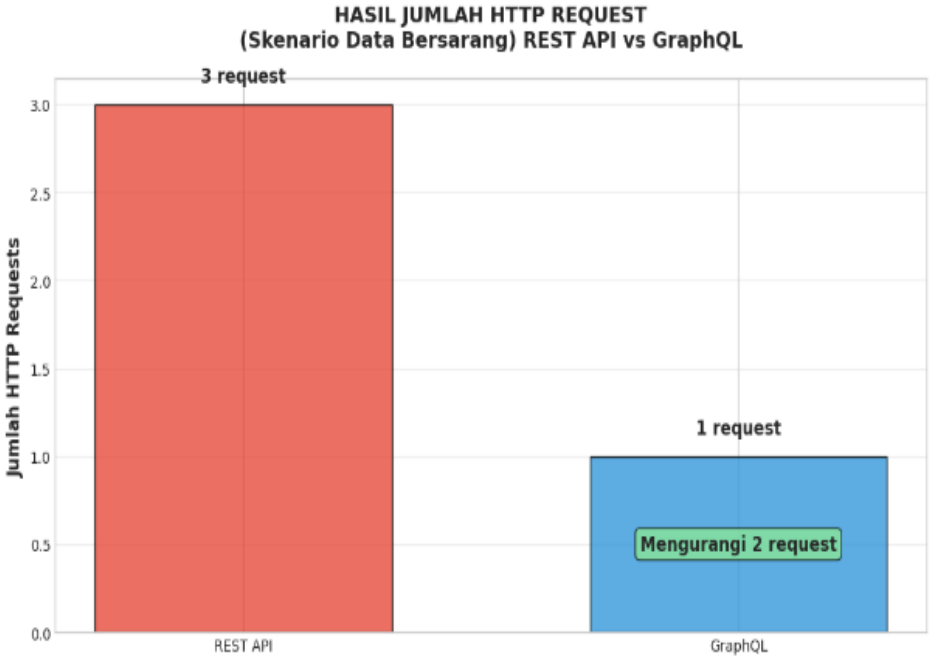
| Parameter Uji | REST API | GraphQL | Interpretasi |
|---|---|---|---|
| # Samples | 697 | 43.899 | Banyaknya permintaan yang berhasil diproses |
| Average (Response Time) | 2084 ms | 29 ms | Waktu rata-rata server merespons |
| Throughput | 21,77 req/s | 1455,39 req/s | Kapasitas pemrosesan permintaan per detik |
| Received KB/sec | 32,8 | 285,4 | Kecepatan transfer data (indikasi efisiensi payload) |

GraphQL menunjukkan throughput yang sangat tinggi (1455,39 req/detik) dibandingkan REST API (21,77 req/detik). Average latency GraphQL (29 ms) secara signifikan lebih rendah daripada REST API (2084 ms).

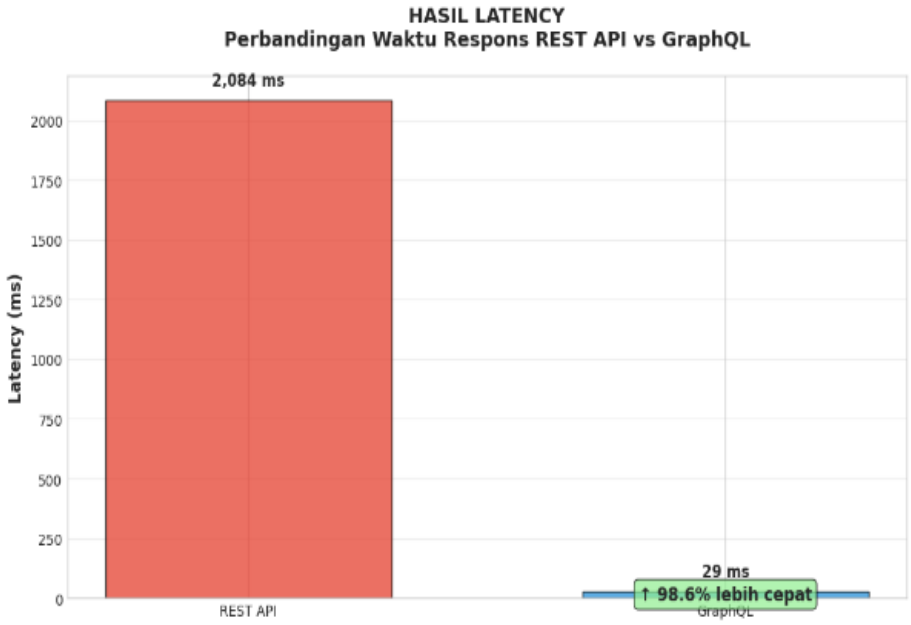
### *C. Pembahasan*

Hasil eksperimen konsisten menunjukkan keunggulan GraphQL pada semua parameter yang diukur, terutama untuk pengambilan data bersarang. Jumlah HTTP Request dan Latency: Pada skenario bersarang, REST API memerlukan tiga permintaan terpisah, menyebabkan akumulasi overhead protokol HTTP (TCP handshake, dll) pada setiap permintaan. Hal ini secara langsung berkontribusi pada latency tinggi (2084 ms pada load test). Sebaliknya, GraphQL mengkonsolidasikan kebutuhan data ke dalam satu permintaan,

menghilangkan round-trip tambahan dan menghasilkan latency yang jauh lebih rendah (29 ms). Temuan ini sejalan dengan penelitian [6] yang menyatakan GraphQL mengurangi jumlah round-trip.

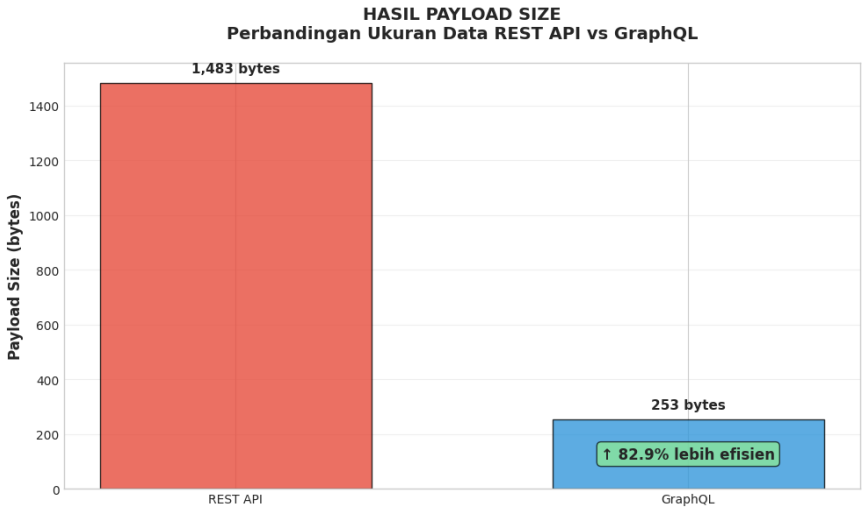



Gambar 2. Perbandingan Jumlah HTTP *Request*



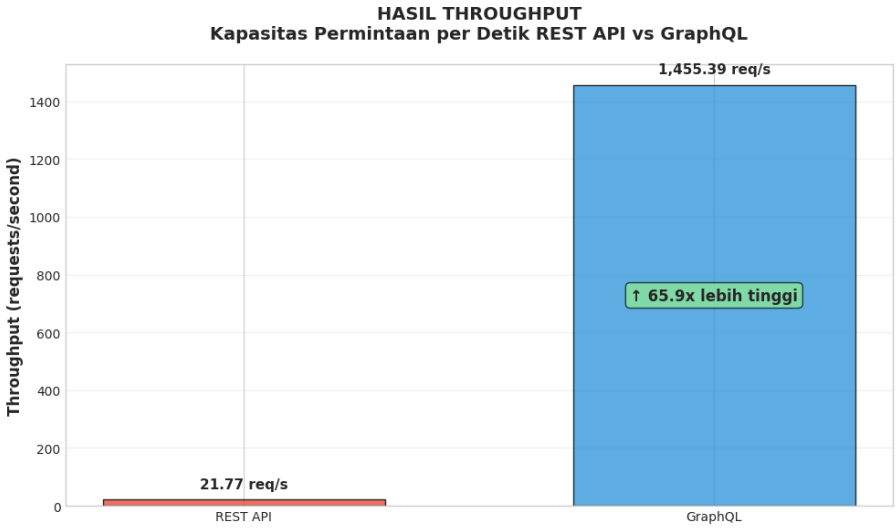

Gambar 3. Perbandingan *Latency*

Ukuran Payload: Mekanisme field selection pada GraphQL memungkinkan klien meminta hanya data yang dibutuhkan. Pada skenario bersarang, GraphQL menghasilkan payload 253 byte (rata-rata), 83% lebih kecil dari REST API (1483 byte). Ini menghindari over-fetching, di mana REST API mengembalikan semua field pada sebuah endpoint meski tidak semuanya dibutuhkan klien. Efisiensi payload ini mempercepat transmisi data dan menghemat bandwidth, mendukung temuan [10].




Gambar 4. Perbandingan Payload size

Throughput: Kemampuan GraphQL menangani 1455,39 permintaan per detik, jauh melampaui REST API (21,77 req/detik), menunjukkan ketahanannya di bawah beban tinggi. Pada REST API, beban server meningkat non-linear karena setiap data bersarang membutuhkan beberapa permintaan independen, menyebabkan bottleneck lebih cepat. GraphQL, dengan arsitektur single-endpoint, memproses data secara terpadu, mengurangi overhead koneksi secara keseluruhan. Hasil ini konsisten dengan studi [8] yang menemukan GraphQL lebih stabil pada beban kompleks.




Gambar 5. Perbandingan Throughput

Keunggulan GraphQL dalam mengurangi under-fetching dan over-fetching berkontribusi langsung pada peningkatan efisiensi komunikasi data. Bagi aplikasi berita digital, pengurangan latency yang drastis berarti waktu muat halaman yang lebih cepat, yang secara positif mempengaruhi pengalaman pengguna dan retensi [4]. Efisiensi bandwidth juga penting, terutama bagi pengguna dengan konektivitas terbatas.

## 1. Kesimpulan dan saran

Berdasarkan hasil penelitian, dapat disimpulkan bahwa implementasi REST API dan GraphQL berhasil dilakukan menggunakan Node.js dengan dataset berita Indonesia yang direstrukturisasi. GraphQL menunjukkan performa yang jauh lebih unggul dibandingkan REST API dalam pengambilan data bersarang. GraphQL memiliki *latency* rata-rata 29 ms (vs 2084 ms pada REST), *payload size* 253 *byte* (vs 1483 *byte*), *throughput* 1455,39 req/detik (vs 21,77 req/detik), dan hanya memerlukan satu HTTP *request* untuk data bersarang (vs beberapa *request* pada REST). GraphQL merupakan arsitektur API yang lebih optimal untuk aplikasi berita digital dengan struktur data kompleks dan bersarang, karena efisiensinya dalam mengurangi *latency*, ukuran *payload*, dan jumlah permintaan, sekaligus meningkatkan *throughput*. Berdasarkan kesimpulan

tersebut, disarankan bagi pengembang, GraphQL dapat diadopsi untuk modul aplikasi berita digital yang membutuhkan pengambilan data bersarang yang efisien. Penelitian lanjutan dapat mengeksplorasi performa dengan dataset lebih besar, kedalaman relasi lebih variatif, serta analisis konsumsi sumber daya (CPU, memori). Untuk implementasi produksi, perlu diterapkan mekanisme keamanan pada GraphQL seperti *rate limiting*, *authorization*, dan *query depth limiting*.

## Ucapan Terima Kasih

Penulis mengucapkan terima kasih kepada STMIK IKMI Cirebon yang telah mendukung pelaksanaan penelitian ini.

## Daftar Pustaka

[1] APJII, "APJII Jumlah Pengguna Internet Indonesia Tembus 221 Juta Orang," 2024. [Online]. Available: https://apjii.or.id/berita/d/apjii-jumlah-pengguna-internet-indonesia-tembus-221-juta-orang. [Accessed: Feb. 7, 2024].

[2] R. T. Fielding, "Architectural Styles and the Design of Network-Based Software Architectures," Ph.D. dissertation, Univ. California, Irvine, CA, USA, 2000. [Online]. Available: https://ics.uci.edu/~fielding/pubs/dissertation/top.htm

[3] R. N. Muzaki and A. Salam, "Reducing under-fetching and over-fetching in REST API with GraphQL for web-based software development," J. Tek. Inform., vol. 5, no. 2, pp. 447–453, Apr. 2024, doi: 10.52436/1.jutif.2024.5.2.1725.

[4] A. Kumar and R. Singh, "The impact of API response time on user retention in mobile applications," J. Syst. Softw., vol. 185, pp. 111–125, 2022, doi: 10.1016/j.jss.2021.111125.

[5] F. Engineer, "GraphQL: A data query language," 2015. [Online]. Available: https://engineering.fb.com/2015/09/14/core-infra/graphql-a-data-query-language/

[6] A. Quiña-Mera, F. Faculty, and U. Técnica, "GraphQL: A systematic mapping study," vol. 55, no. 10, 2023, doi: 10.1145/3561818.

[7] M. Śliwa and B. Pańczyk, "Performance comparison of programming interfaces on the example of REST API, GraphQL and gRPC," vol. 21, pp. 356–361, 2021.

[8] A. Lawi, B. Panggabean, and T. Yoshida, "Evaluating GraphQL and REST API services performance in a massive and intensive accessible information system," Comput., vol. 10, p. 138, 2021, doi: 10.20944/preprints202109.0386.v1.

[9] M. Niswar, R. A. Safruddin, A. Bustamin, and I. Aswad, "Performance evaluation of microservices communication with REST, GraphQL, and gRPC," Int. J. Electron. Telecommun., 2024, doi: 10.24425/ijet.2024.149562.

[10] S. Kanthed, "REST vs. GraphQL: Comparative analysis of API design approaches," Int. J. Multidiscip. Res. Growth Eval., 2023, doi: 10.54660/.ijmrge.2023.4.1.984-991.