Jurnal Riset Teknik Informatika (JURETI)
Vol. 1, No. 2, Desember 2024, pp. 120-124

DOI: .. . ./jureti
E-ISSN: 3063-9662

COMPARISON OF REST API AND GRAPHQL IN THE STRAPI CONTENT MANAGEMENT SYSTEM

Yaser Rayhan[1], Maya Suhayati[2], Beben Sutara[3]

[1,2,3]Informatics, Engineering Faculty, Universitas Sebelas April, Indonesia
Email: [1]a22100135@mhs.stmik-sumedang.ac.id, [2]mayasuhayati@unsap.ac.id , [3]beben@unsap.ac.id

Genesis:(Article received: 18 Desember 2024; Revision: 23 Desember 2024; published: 24 Desember 2024)

Abstract

This study aims to evaluate the performance of two APIs in scenarios involving data retrieval, creation, and updates. REST API is known for its simplicity and high compatibility, whereas GraphQL offers greater flexibility in specifying required data, reducing over-fetching and under-fetching. The study employs a quantitative approach using JMeter to measure throughput, data size, and response time of each API. The results indicate that REST API excels in data retrieval with higher throughput, while GraphQL proves more effective for data creation and updates due to its ability to optimize queries more efficiently. REST API is also more efficient for storing small-sized data; however, GraphQL outperforms it when the total data exceeds 100 units and in handling large data volumes with better network efficiency. In conclusion, REST API is suitable for applications with high data demand and low complexity, whereas GraphQL is recommended for environments requiring large-scale data creation, updates, and dynamic data management.

Keywords: *API performance, GraphQL, REST API, Strapi.*

KOMPARASI REST API DAN GRAPHQL DI CONTENT MANAGEMENT SYSTEM STRAPI

Abstrak

Penelitian ini bertujuan untuk mengevaluasi performa kedua API dalam skenario pengambilan, pembuatan, dan pembaruan data. REST API dikenal dengan kesederhanaan serta kompatibilitas tinggi, sedangkan GraphQL menawarkan fleksibilitas lebih dalam menentukan data spesifik yang dibutuhkan, mengurangi over-fetching dan under-fetching. Studi ini dilakukan melalui pendekatan kuantitatif dengan menggunakan JMeter untuk mengukur throughput, ukuran data, dan respons waktu dari setiap API. Hasil menunjukkan REST API unggul dalam pengambilan data dengan throughput yang lebih tinggi, sementara GraphQL lebih efektif dalam pembuatan dan pembaruan data berkat kemampuan untuk mengoptimalisasi query secara lebih efisien. REST API juga lebih efisien untuk penyimpanan data ukuran kecil, namun GraphQL mengungguli saat total data melebihi 100, serta dalam menangani volume data besar dengan efisiensi penggunaan jaringan yang lebih baik. Kesimpulannya, REST API cocok untuk aplikasi dengan permintaan data tinggi dan kompleksitas rendah, sedangkan GraphQL disarankan untuk lingkungan yang membutuhkan pembuatan dan pembaruan data besar serta manajemen data yang dinamis.

Kata kunci: *GraphQL, Kinerja API, REST API, Strapi.*

1. PENDAHULUAN

Data telah diakui sebagai sumber daya baru yang sangat bernilai di era digital ini. Pengelolaan dan pemanfaatan data secara efektif menjadi kunci dalam pengembangan berbagai aplikasi modern. Dalam konteks ini, proses query menjadi sangat penting karena menentukan bagaimana data diambil, dimanipulasi, dan disajikan kepada pengguna. Content Management System (CMS) menjadi solusi yang populer untuk memudahkan pengelolaan konten pada berbagai jenis aplikasi, baik itu website, aplikasi mobile, maupun aplikasi lainnya. Strapi adalah salah satu CMS yang menawarkan fleksibilitas tinggi dan mendukung berbagai arsitektur API, termasuk REST API dan GraphQL.

REST API (Representational State Transfer) dan GraphQL adalah dua arsitektur API yang populer digunakan dalam pengelolaan konten di CMS. REST API adalah pendekatan tradisional yang menggunakan endpoint yang berbeda untuk setiap jenis permintaan data (GET, POST, PUT, DELETE). Sementara itu, GraphQL adalah bahasa query yang memungkinkan klien untuk meminta data yang spesifik dari server, mengurangi masalah over-fetching dan under-fetching data.

Dengan menggunakan Strapi, pengembang aplikasi dapat memilih dan mengimplementasikan salah satu dari kedua API sesuai dengan kebutuhan proyek mereka. Namun, pemilihan arsitektur API yang tepat membutuhkan pemahaman mendalam mengenai keunggulan, kelemahan, dan performa masing-masing API.

Oleh karena itu, penelitian ini bertujuan untuk menemukan perbedaan antara REST API dan GraphQL dalam konteks implementasi di CMS Strapi. Penelitian ini akan menganalisis performa kedua API dengan cara menilai throughput, ukuran data dan time response API.

2. METODE PENELITIAN

Content management system(CMS) merupakan sebuah aplikasi yang dapat dengan mudah digunakan untuk membuat, mengelola, dan memodifikasi konten digital[1][2][3]. CMS dapat didefinisikan sebagai sistem yang dirancang untuk mendukung pengelolaan dan pengorganisasian informasi yang akan dipublikasikan dalam berbagai bentuk[4]. CMS memiliki beberapa karakteristik utama yang membedakannya dari sistem pengelolaan konten lainnya, yaitu *user-friendly interface, content storage and retrieval, workflow management, version control, role-based access control*[5][6].

Strapi adalah CMS headless yang modern dan open-source, dirancang untuk memberikan fleksibilitas yang tinggi kepada pengembang dan pengguna. Headless CMS adalah tipe CMS yang memisahkan bagian pengelolaan konten (backend) dari bagian penyajian konten (frontend), memungkinkan konten untuk diakses dan digunakan oleh berbagai aplikasi melalui API. Strapi mendukung dua arsitektur API, yaitu REST API dan GraphQL.

REST API (Representational State Transfer) telah menjadi standar API dalam komunikasi antar aplikasi di web. Keunggulannya terletak pada kesederhanaan, kemudahan implementasi, dan kompatibilitas yang luas dengan berbagai teknologi[7][8][9][10]. Sedangkan GraphQL adalah teknologi yang lebih baru yang dikembangkan oleh Facebook untuk mengatasi beberapa kelemahan yang ada pada REST API. GraphQL memungkinkan klien untuk secara spesifik menentukan data yang dibutuhkan, sehingga dapat mengurangi over-fetching dan under-fetching data[11][12].

2.1. Pendekatan Penelitian

Penelitian ini akan menggunakan pendekatan kuantitatif untuk mengukur dan menganalisis perbedaan performa serta keunggulan dan kelemahan REST API dan GraphQL di CMS Strapi. Metode kuantitatif dipilih karena memungkinkan pengukuran yang objektif. Proses pengerjaan penelitian ini dimulai pada instalasi CMS Strapi 4, lalu akan dilanjutkan dengan pembuatan struktur general yang di dalamnya mencakup sebuah data dengan awal sepuluh kolom dan memiliki satu relasi ke data lain. Selama proses pengumpulan data, akan terjadi pengulangan proses input data yang dimana akan menambahkan jumlah data pengujian(melalui API), lalu testing endpoint lainnya dengan masing masing menggunakan kedua metode. Data hasil testing maka akan disimpan lalu akan kembali ke penambahan jumlah data. Apabila waktu pengetesan sudah selesai akan dilanjutkan dengan perhitungan rata rata hasil dari kedua metode API yang nantinya akan dijadikan acuan untuk komparasikan.

2.2. Metode Pengumpulan Data

Data kinerja akan dikumpulkan melalui pengujian yang dilakukan menggunakan JMeter. Beberapa skenario uji akan diterapkan untuk mengukur performa REST API dan GraphQL dalam berbagai kondisi penggunaan.

Pengetesan API akan dibagi menjadi dua batch, per satu batch nya akan terjadi penambahan seratus lalu dua ratus data. Setiap batch dimulai dengan penambahan data sebanyak seratus lalu dua ratus data, lalu akan dilanjut dengan pengambilan data dan dilanjutkan dengan edit data.

3. HASIL DAN PEMBAHASAN

Data yang dikumpulkan dari pengujian kinerja REST API dan GraphQL dalam CMS Strapi akan disajikan dalam bentuk tabel untuk memudahkan pembacaan dan analisis lebih lanjut. Tabel-tabel berikut menampilkan metrik kinerja utama yang diukur selama penelitian, termasuk time response, throughput, dan ukuran data yang dikirim maupun yang diterima.

Tabel 1. Hasil Testing Rest API

Jenis	Sample	Time Response				Data / detik (KB)			Total Data	Total User
		Average	Median	Min	Max	Troughput / detik	Diterima	Dikirim		
Create	100	11	9	7	63	101.5	119.37	73.76	100	10
Get - Relasi	100	260	264	50	555	28.6	5585.57	6.06	100	10
Update	100	30	28	16	72	82.9	97.79	61	100	10
Create	200	42	40	11	84	141.9	166.9	103.13	300	20
Get - Relasi	200	1150	1233	108	1578	15.4	6021.89	3.67	300	20
Jenis	**Sample**	**Time Response**				**Data / detik (KB)**			**Total Data**	**Total User**
		Average	**Median**	**Min**	**Max**	**Troughput / detik**	**Diterima**	**Dikirim**		

Get - Relasi	300	1717	1829	151	2378	10.6	6177.75	2.51	300	20
Update	300	109	99	17	271	101	115.80	74.38	300	20

Tabel 2. Hasil Testing GraphQl

Jenis	Sample	Time Response				Data / detik (KB)			Total Data	Total User
		Average	Median	Min	Max	Troughput / detik	Diterima	Dikirim		
Create	100	14	13	8	44	93.5	115.46	70.43	100	10
Get - Relasi	100	1309	1362	450	1456	7.1	481.25	4.52	100	10
Update	100	29	28	13	64	86.3	106.86	66.23	100	10
Create	200	26	26	7	65	184.2	227.50	138.66	300	20
Get - Relasi	200	5520	5591	1119	6062	3.5	472.38	2.23	300	20
Get - Relasi	300	8638	8639	2241	10151	2.3	453.91	1.43	300	20
Update	300	84	76	12	193	116.8	114.35	89.67	300	20

Data yang berhasil dikumpulkan pada penelitian ini mencakup time response, throughput dan juga ukuran data saat mengirim request lalu ukuran data response nya. Pada tabel 1 dan tabel 2 terdapat kolom jenis yang merepresentasikan metode yang sedang dijalankan. Sample merupakan jumlah data yang diambil atau dibuat dalam sekali batch testing, selanjutnya kolom time response menyimpan data waktu yang dibutuhkan untuk sebuah proses selesai. Selanjutnya pada kolom throughput / detik, menyimpan data rata rata throughput dalam satu detik selama proses batch berlangsung. Kolom data / detik menyimpan rata rata data yang dikirim dan diterima selama batch. Lalu total data merupakan jumlah data yang tersimpan pada database setelah batch dijalankan. Terakhir terdapat total user, kolom ini mempresentasikan jumlah user yang aktif secara bersamaan untuk menjalankan proses yang sama selama batch berlangsung.

3.1. Hasil Time Response

Perbandingan rata rata time response, dinyatakan bahwa Rest API unggul dibandingkan dengan graphQl dalam studi kasus mendapatkan data atau proses HTTP GET. Namun sebaliknya graphQl dapat mengungguli rest API dalam kasus membuat data baru dan mengupdate data.

3.2. Hasil Throughput per Detik

Perbandingan Throughput per Detik, dinyatakan bahwa Rest API memiliki keunggulan throughput saat mengambil data, selain itu Rest API dapat mengungguli saat penyimpanan data sebanyak 100 data pertama. Sebaliknya graphQl dapat mengungguli saat penyimpanan data saat totalnya diatas 100, selain itu graphQl unggul saat update data.

3.3. Hasil Ukuran Data

Perbandingan Ukuran Data, dinyatakan bahwa graphQl sangat mengungguli rest API pada metriks ukuran data, hasil yang terlihat jauh terdapat pada pengetesan studi kasus pengambilan data, baik pengambilan seratus, dua ratus maupun tiga ratus data.

3.4. Pembahasan Hasil Penelitian

Penelitian ini mengevaluasi kinerja Rest API dan GraphQL dalam berbagai skenario, yaitu pengambilan data, pembuatan data baru, dan pembaruan data. Hasil penelitian menunjukkan perbedaan yang signifikan dalam performa kedua teknologi tersebut berdasarkan beberapa metriks yang digunakan. Pada pengambilan data (*HTTP GET*), hasil penelitian menunjukkan bahwa *Rest API* unggul dalam studi kasus pengambilan data atau proses *HTTP GET*. Keunggulan *Rest API* dalam skenario ini dapat disebabkan oleh struktur endpoint yang spesifik, yang memungkinkan pengambilan data dilakukan dengan lebih efisien. *Throughput* yang tinggi pada *Rest API* saat pengambilan data menandakan bahwa *Rest API* mampu memproses permintaan *GET* dengan cepat, yang penting dalam aplikasi dengan permintaan data yang tinggi. Sedangkan dalam skenario pembuatan data baru dan pembaruan data, GraphQL menunjukkan keunggulan dibandingkan dengan *Rest API*. Hal ini mungkin karena kemampuan *GraphQL* untuk mengirim permintaan yang lebih spesifik dan kompleks dalam satu operasi, mengurangi jumlah permintaan yang diperlukan untuk memodifikasi data. Oleh karena itu, dalam kasus di mana pembuatan dan pembaruan data adalah operasi yang dominan, *GraphQL* mungkin menjadi pilihan yang lebih baik. Penelitian ini juga menemukan bahwa *Rest API* memiliki keunggulan *throughput* saat mengambil data serta dalam penyimpanan data untuk 100 data pertama. Namun, ketika total data yang disimpan melebihi 100, *GraphQL* mulai mengungguli Rest API. Hal ini menunjukkan bahwa *Rest API* lebih efisien untuk operasi dengan jumlah data yang lebih kecil, sedangkan *GraphQL* lebih baik untuk operasi dengan jumlah data yang besar.

4. DISKUSI

Keunggulan GraphQL dalam menangani volume data yang lebih besar mungkin disebabkan oleh kemampuannya untuk mengoptimalkan kueri dan pengambilan data secara lebih efektif

dibandingkan Rest API. Selain itu, GraphQL unggul dalam pembaruan data, menunjukkan fleksibilitas dan efisiensi yang lebih tinggi dalam menangani perubahan data yang sering. Dan dalam hal ukuran data, GraphQL sangat mengungguli Rest API, terutama dalam studi kasus pengambilan data dengan ukuran yang bervariasi (100, 200, hingga 300 data). Hal ini bisa karena kemampuan GraphQL untuk mengontrol dengan tepat data apa saja yang dibutuhkan, sehingga mengurangi jumlah data yang dikirim melalui jaringan. Efisiensi ini sangat penting dalam aplikasi yang memerlukan transmisi data yang besar, karena dapat mengurangi latensi dan bandwidth yang dibutuhkan.

4.1. Perbandingan Dengan Penelitian Sebelumnya

Perbandingan dengan penelitian sejenis dari jurnal/conference terdahulu

1. Pada penelitian yang berjudul "*Can GraphQL Replace REST? A Study of Their Efficiency and Viability*", menunjukkan bahwa kedua paradigma API, baik REST maupun GraphQL, memiliki kelebihan dan kelemahannya masing-masing. Studi ini menyimpulkan bahwa, setidaknya dalam waktu dekat, satu paradigma tidak dapat sepenuhnya menggantikan yang lain[13].
2. Pada penelitian "*Evaluating GraphQL and REST API Services Performance in a Massive and Intensive Accessible Information System*", menunjukkan bahwa REST masih lebih cepat hingga 50.50% dalam waktu respons dan 37.16% dalam throughput, sementara GraphQL sangat efisien dalam pemanfaatan sumber daya, yaitu 37.26% untuk beban CPU dan 39.74% untuk penggunaan memori. Oleh karena itu, GraphQL adalah pilihan yang tepat ketika kebutuhan data sering berubah dan pemanfaatan sumber daya merupakan pertimbangan yang paling penting. REST digunakan ketika beberapa data sering diakses dan diminta oleh banyak permintaan[14].
3. Pada penelitian "*Comparative Analysis of Rest and GraphQL Technology on Nodejs-Based Api Development*", menunjukkan bahwa REST memiliki kinerja yang lebih baik dibandingkan GraphQL dalam hal kecepatan respons. Di sisi lain, GraphQL unggul dalam penyajian data berdasarkan permintaan aplikasi klien, yang membantu mengoptimalkan bandwidth yang tersedia[15].

5. KESIMPULAN

Penelitian ini telah mengevaluasi kinerja Rest API dan GraphQL dalam berbagai skenario operasi seperti pengambilan data (HTTP GET), pembuatan data baru, dan pembaruan data. Berdasarkan hasil penelitian, dapat diambil beberapa kesimpulan sebagai berikut:

5.1. Keunggulan Rest API

Pengambilan Data (HTTP GET): Rest API menunjukkan keunggulan dalam pengambilan data atau proses HTTP GET. Keunggulan ini mencakup throughput yang lebih tinggi, yang mengindikasikan kemampuan Rest API dalam memproses permintaan GET dengan lebih cepat dan efisien.

Throughput Awal Penyimpanan Data: Rest API juga unggul dalam throughput saat menyimpan 100 data pertama. Ini menunjukkan efisiensi Rest API dalam menangani operasi penyimpanan untuk jumlah data yang relatif kecil.

5.2. Keunggulan GraphQL

Pembuatan dan Pembaruan Data: GraphQL menunjukkan performa yang lebih baik dalam skenario pembuatan data baru dan pembaruan data. Hal ini mungkin disebabkan oleh kemampuan GraphQL untuk mengirim permintaan yang lebih spesifik dan kompleks dalam satu operasi, yang mengurangi jumlah permintaan yang diperlukan untuk memodifikasi data.

Penyimpanan Data dalam Jumlah Besar: GraphQL unggul dalam penyimpanan data saat total data melebihi 100. Kemampuan GraphQL untuk mengoptimalkan kueri dan pengambilan data secara lebih efektif menjadi faktor kunci dalam skenario ini.

Ukuran Data: Dalam hal ukuran data, GraphQL sangat mengungguli Rest API, terutama dalam pengambilan data dengan ukuran yang bervariasi. Efisiensi ini penting untuk mengurangi latensi dan bandwidth yang dibutuhkan, terutama dalam aplikasi yang memerlukan transmisi data yang besar.

5.3. Rekomendasi Pemilihan Teknologi

Berdasarkan hasil dan kesimpulan di atas, pemilihan antara Rest API dan GraphQL harus disesuaikan dengan kebutuhan spesifik aplikasi:

Rest API lebih sesuai untuk aplikasi yang membutuhkan throughput tinggi dan efisiensi dalam pengambilan data, terutama untuk jumlah data yang lebih kecil.

GraphQL lebih unggul dalam skenario yang melibatkan pembuatan, pembaruan, dan pengambilan data dalam jumlah besar, serta dalam mengelola ukuran data yang ditransfer.

6. DAFTAR PUSTAKA

[1] N. A. N. Sobri, M. A. H. Abas, A. I. M. Yassin, M. S. A. M. Ali, N. M. Tahir, A. Zabidi, and Z. I. Rizman, "Comparison between Headless CMS and Backend-as-a-Service Products for E-Suripreneur

Backend", Mathematical Statistician and Engineering Applications, vol. 71, no. 3s2, pp. 928–938, 2022.

[2] V. Sindekar and A. Pandey, "Comparison Between Traditional Web Design and CMS", SAMRIDDHI: A Journal of Physical Sciences, Engineering and Technology. [Online]. Available: https://doi.org/10.18090/samriddhi.v13is1.11. Accessed: Dec. 15, 2024.

[3] G. Sharma, A. Kute, N. Jadhav, N. Kolhe, and A. Tiwari, "Review of Web Content Management Systems and their Increasing Demand in Market", International Journal for Research in Applied Science and Engineering Technology. [Online]. Available: https://doi.org/10.22214/ijraset.2022.48167. Accessed: Dec. 15, 2024.

[4] M. Hembram, "Comparative Study of Open Source Content Management Systems (CMS) in Digital Era", Asian Journal of Electrical Sciences. [Online]. Available: https://doi.org/10.51983/ajes-2022.11.1.3184. Accessed: Dec. 15, 2024.

[5] B. Boiko, Content Management Bible. John Wiley & Sons, 2005.

[6] M. Student and M. Chandrika, "Content Management Systems (CMS): Traditional model vs CMS in managing corporate websites," 2021.

[7] Prayogi A A, Niswar M., Indrabayu and Rijal M.,"Design and implementation of REST API for academic information system", in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, 2020, p. 012047.

[8] R. Huang, M. Motwani, I. Martinez, and A. Orso, "Generating REST API Specifications through Static Analysis", in 2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE), pp. 1311–1323, 2024. [Online]. Available: https://doi.org/10.1145/3597503.3639137. Accessed: Dec. 15, 2024.

[9] E. Viglianisi, M. Dallago, and M. Ceccato, "RESTTESTGEN: Automated Black-Box Testing of RESTful APIs", in 2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST), pp. 142–152, 2020. [Online]. Available: https://doi.org/10.1109/icst46399.2020.00024. Accessed: Dec. 15, 2024.

[10] A. Golmohammadi, M. Zhang, and A. Arcuri, "Testing RESTful APIs: A Survey", ACM Transactions on Software Engineering and Methodology, vol. 33, pp. 1–41, 2022. [Online]. Available: https://doi.org/10.1145/3617175. Accessed: Dec. 15, 2024.

[11] A. Quiña-Mera, P. Fernández, J. García, and A. Ruiz-Cortés, "GraphQL: A Systematic Mapping Study", ACM Computing Surveys, vol. 55, pp. 1–35, 2022. [Online]. Available: https://doi.org/10.1145/3561818. Accessed: Dec. 15, 2024.

[12] T. Díaz, F. Olmedo, and É. Tanter, "A mechanized formalization of GraphQL", in Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, 2020. [Online]. Available: https://doi.org/10.1145/3372885.3373822. Accessed: Dec. 15, 2024.

[13] S. L. Vadlamani et al., "Can GraphQL replace REST? A study of their efficiency and viability", in 2021 IEEE/ACM 8th International Workshop on Software Engineering Research and Industrial Practice (SER&IP), IEEE, 2021, pp. 10–17.

[14] A. Lawi, B. L. Panggabean, and T. Yoshida, "Evaluating GraphQL and REST API services performance in a massive and intensive accessible information system," Computers, vol. 10, no. 11, p. 138, 2021.

[15] G. S. M. Diyasa, G. S. Budiwitjaksono, H. A. Ma'rufi, and I. A. W. Sampurno, "Comparative analysis of REST and GraphQL technology on Node.js-based API development," Nusantara Science and Technology Proceedings, pp. 43–52, 2021.