JSCE (Journal of System and Computer Engineering)
Vol.6, No.1, January 2025, pp. 161-169
ISSN (online): 2723-1240
DOI : https://doi.org/10.61628/jsce.v6i1.1637

Research Article Open Access (CC–BY-SA) ■

Performance Analysis of API in Google Cloud Storage Service Integration

Respaty Namruddin [1,a,*]; Rafiqa Mulia Indah Sari Sam [2,b]; Rajul Waahid Syamsuddin [3,c]; Amiruddin A [4,d]; Aang Kunaefi [5,e]
[1234] *Universitas Handayani Makassar, Adhyaksa Baru No. 1, Makassar 90231, Indonesia*
[a] *respatynamruddin@handayani.ac.id*; [b] *rafiqamlyindh@gmail.com*; [c] *rajul.waahid@gmail.com*; [d] *amiruddinardinmks@gmail.com*; [e] *aan.mks@gmail.com*
** Corresponding author*

Abstract

Google Cloud Storage (GCS) is one of the leading cloud storage services that supports large-scale data management through API integration. APIs allow applications to upload, download, and manage data in real-time. This study aims to analyze the performance of APIs in integration with GCS using response time, throughput, and latency parameters. Tests were conducted on various scenarios, including massive data transfer, distributed data management, and caching usage. The results showed that the average API response time reached 120 ms under normal conditions and increased to 180 ms under high load. Throughput reached an average of 400 MB/s, but decreased when the number of simultaneous requests increased. The average server latency was recorded at 60 ms and can be optimized with caching technology. Implementation of strategies such as Content Delivery Network (CDN) and request header optimization can improve performance by up to 30%. This study provides practical guidance for developers to optimally utilize GCS APIs in large-scale data management.

Keywords— *API, Google Cloud Storage, Cloud Storage, API Performance, Data Management*

1. Introduction

In the era of digital transformation, the volume of data generated by various industrial sectors and organizations continues to increase exponentially (Angelia Putriana, 2023). This data originates from diverse sources, such as online transaction activities, Internet of Things (IoT) sensors, and social media-based applications (Aliwijaya, 2023). The massive growth of data presents new challenges in terms of management, storage, and accessibility, requiring reliable, efficient, and flexible storage technologies. Cloud storage has emerged as the primary solution to these challenges, offering scalable internet-based infrastructure that is accessible anytime and anywhere (Rahmah et al., 2024).

Google Cloud Storage (GCS) is a cloud storage service designed to meet the needs of large-scale data management (Hiriyannaiah et al., 2023). Google Cloud Storage (GCS) has emerged as a leading solution for managing large-scale data through its robust object-based storage capabilities (Ita Ita et al., 2023). With its object-based architecture, GCS offers high flexibility in storing and managing various types of data, including documents, images, and large log files. One of GCS's key strengths is its support for API (Application Programming Interface) integration, enabling application developers to access and utilize this service automatically and efficiently. Through APIs, processes such as file uploads, data downloads, and access permission management can be performed more easily and systematically.

However, as cloud storage services scale to accommodate large datasets, API performance becomes a critical challenge. Factors such as network conditions (e.g., bandwidth limitations, latency), server location, and simultaneous user requests can significantly impact performance (Diantono et al., 2024). Additionally, the growing number of simultaneous requests due to user growth can create bottlenecks in the system, thereby reducing API performance. In this context, analyzing API performance is crucial to ensuring that cloud storage services like GCS can continue to meet both business and technical requirements. Furthermore, recent technologies like HTTP/3 and QUIC could potentially enhance communication efficiency and minimize latency.

API performance is influenced by various factors, such as server location, the API technology used (REST or SDK), and caching mechanisms (Mulana et al., 2022). For instance, server location determines the data travel distance between the client and the server, ultimately affecting latency. API technology also plays a crucial role in determining the efficiency of communication between applications and cloud services. REST APIs, which use the HTTP protocol, are widely adopted for their flexibility but incur greater network overhead compared to more integrated SDKs. Meanwhile, caching can help reduce access times by storing copies of data closer to the user.

In this study, Google Cloud Storage (GCS) was selected as a case study to analyze API performance in large-scale data management. GCS offers standout features such as multi-region storage to support data availability across regions, default data encryption for security, and full support for integration with various platforms via APIs. By leveraging these services, organizations can optimize their data management processes, whether for daily operational needs or large-scale data analytics.

This study also aims to identify the challenges faced in integrating APIs with cloud storage services. One of the primary challenges is ensuring that APIs can handle large volumes of requests without experiencing significant performance degradation. On the other hand, effective optimization strategies are needed to enhance API efficiency, such as using Content Delivery Networks (CDNs) to reduce latency or managing batch requests to improve throughput.

Additionally, this study is designed to provide practical insights for application developers seeking to optimize their use of Google Cloud Storage. Through in-depth testing and analysis, this research will evaluate key API performance parameters, such as response time, throughput, and latency, across various usage scenarios. The study will also propose optimization strategies that can be implemented to improve API performance, enabling cloud storage services to better support large-scale data needs.

With this approach, the study is expected to contribute both academically and practically. Academically, it can serve as a reference for future research focusing on API performance in cloud services. Practically, the findings are anticipated to help organizations and application developers address API performance challenges while maximizing the use of Google Cloud Storage to support their operations more efficiently.

2. Method

2.1 API (*Application Programming Interface*)

An API (Application Programming Interface) is a software interface that enables two systems or applications to communicate and exchange data (Kurnia Bakti et al., 2022). APIs are designed to simplify software development by providing a set of functions accessible to developers without requiring them to understand the technical details of the backend system. In the context of cloud storage, APIs allow applications to access cloud service features such as uploading files, downloading data, creating directories, and managing data access permissions (Bintang Irfansyah et al., 2024).

An Application Programming Interface acts as a programming interface that enables applications to communicate and share information with one another. APIs serve as a bridge

between systems and applications with similar goals but different methods. APIs facilitate standardization in data exchange, allowing developers to provide various types of data that other developers can understand and utilize within their own systems (Firdaus & Afwani, 2024).

Google Cloud Storage supports REST APIs and SDKs for various programming languages, such as Python, Java, and Go, offering flexibility for developers to integrate cloud storage services into their applications.

2.2 Cloud Storage

Cloud storage is a digital data storage model where data is stored on servers accessible via the internet (Purbasari et al., 2024). This service is designed to provide high scalability, flexible access, and enhanced data security compared to local storage. Object-based cloud storage, such as Google Cloud Storage, stores data as objects uniquely identified by URLs. The advantages of cloud storage include (Yoesoep et al, 2023):

1. Scalability: The ability to handle large volumes of data without the need for additional physical infrastructure.
2. Flexibility: Access data anytime and anywhere with an internet connection.
3. Security: Cloud storage services like GCS use high-level encryption to protect data both in transit and at rest.

2.3 API Performance Parameters

In evaluating API performance, three main parameters are analyzed:

1. Response Time:
   Response time refers to the duration required by an API to respond to a client request (Setiawan et al., 2025). The shorter the response time, the faster the API can handle requests. Factors such as server location, internet network, and API operation complexity influence response time.
2. Throughput:
   Throughput measures the amount of data an API can process within a unit of time, typically in megabytes per second (MB/s) (Saputra et al., 2023). High throughput indicates that the API can efficiently handle large volumes of data. However, throughput may decrease when the number of concurrent requests increases.
3. Latency:
   Latency is the time required to transfer data between the client and the server, and vice versa (Abdullah et al., 2024). Low latency is critical for real-time operations, such as video streaming or time-sensitive data transfers.

These parameters serve as key benchmarks in determining the quality of API services across various operational scenarios.

2.4 *Google Cloud Storage*

Cloud storage is a concept and technology that enables the storage, management, and access of data via the internet (Zainul & Romadhan, 2023). In this system, data is stored on servers located in secure data centers and can be accessed by users through an internet connection. Unlike local data storage, cloud storage allows users to store and manage data without relying on physical hardware at their location.

Google Cloud Storage is an object-based storage service offered by Google Cloud Platform (GCP) (Chandra1 et al., 2024). This service is designed to support large-scale data management with features such as data encryption, access management, and cross-regional data replication. Google Cloud Storage provides various storage classes to cater to user needs, including Standard Storage for high-frequency data access, Nearline Storage for monthly access, and Archive Storage for infrequently used archived data. Each class is designed with cost and performance optimized for specific access patterns.

2.5 API Optimization Strategies

Optimization strategies implemented include::

1. Caching:
   Caching is a technique for storing frequently accessed data in local memory or nearby servers to reduce response time (Hidayatut & Harianto, 2023). By using caching, API requests do not always need to access the original data on the main server, thereby reducing network load and speeding up response times.
2. Content Delivery Network (CDN):
   A CDN is a globally distributed network of servers that store copies of data closer to end users (Rahmadyanto & Chandra, 2023). Using a CDN can reduce latency by avoiding long-distance data transfers.
3. Batch Processing:
   This technique combines multiple API requests into a single request to reduce overhead and improve throughput (Khairy et al., 2023). Batch processing is highly effective in scenarios where many requests need to be processed simultaneously.
4. Request Header Optimization:
   Adjusting API request headers to include information such as ETags or other metadata can speed up response times, especially for repeated requests.

3. Results And Discussion

3.1 API Performance Testing

1. Response Time :
   a. Normal load (low latency, high bandwidth): Average of 120 ms.
   b. High load (1,000 simultaneous requests): Increased to 180 ms.
   c. Low-bandwidth, high-latency conditions: Response time exceeded 300 ms.
2. Throughput :
   a. Normal load: 400 MB/s.
   b. High load: Decreased to 320 MB/s.
   c. Optimized with batch processing: Increased throughput to 450 MB/s.
3. Latency :
   a. Normal conditions: 60 ms.
   b. High-latency network: Increased to 90 ms.
   c. Optimized with CDN: Reduced latency to 40 ms.

3.2 Resource Efficiency

The study measured CPU and memory utilization during testing. Results showed that under high loads, CPU utilization increased by 30%, while memory usage grew by 25%. Optimization strategies such as caching reduced CPU utilization by 15%.

3.3 Real-World Simulation

Simulated scenarios included IoT data uploads and media streaming applications. Results indicated significant performance degradation without optimization, particularly for latency-sensitive operations like live streaming. CDN integration improved playback latency by 20%.

3.4 Comparative Analysis

Performance benchmarks against Amazon S3 showed that GCS achieved comparable response times but higher throughput due to batch processing optimization.

3.5 Statistical Analysis

A t-test confirmed the significance of improvements after optimization, with p-values < 0.05 for response time and throughput metrics.

3.6 Impact on End-User Experience

End-user feedback highlighted noticeable improvements in application responsiveness and data loading times. The study underscores the importance of maintaining low latency and high throughput for enhancing user satisfaction.

4. Conclusions

4.1 API Performance Testing Results

Testing was conducted by measuring three key API performance parameters: response time, throughput, and latency. The tests were performed under two conditions, normal load and high load, using tools such as Postman for response time, Apache JMeter for load simulation, and cloud monitoring for network analysis.

1. Response Time:
   Response time is calculated as the time difference between the client sending a request to the API and receiving a response from the server.

Figure 1. Average Response Time Across Three Scenarios.

Formula:
Response Time = Response Reception Time − Request Transmission Time

Calculation:
If the request transmission time is 10:00:00.000 and the response reception time is 10:00:00.120, then:

$$\text{Response Time} = 120\ ms$$

Results:

- Normal load: an average of 120 ms.
- High load (1,000 simultaneous requests): the average increased to 180 ms.

2. Throughput:
   Throughput is calculated as the amount of data processed within a unit of time.

Figure 2. Average Throughput Across Various Scenarios.

Formula:

Calculation:
If 1 GB (1,024 MB) of data is processed in 2.5 seconds, then:

Results:
- Normal load: an average of 400 MB/s.
- High load: throughput decreased to 320 MB/s due to the large number of simultaneous requests.

3. Latency:
   Latency refers to the time it takes for data to travel from the client to the server and back to the client.

Figure 3. Average Latency Across Various Scenarios.

Formula:

Calculation**:**
If the round-trip time (RTT) is 120 ms, then:

Results:

- Normal load: an average latency of 60 ms.
- High load: latency increased to 90 ms due to request queuing.

4.2 Optimization Implementation

To improve the performance of the Google Cloud Storage API, several optimization strategies were implemented and tested:

1. *Caching*:
   Caching stores frequently accessed data in cache so that subsequent requests do not need to retrieve the original data from the server. This is implemented by leveraging HTTP headers like Cache-Control and ETag to ensure API responses are stored in local or intermediary server caches. Caching successfully reduced the average response time from 120 ms to 85 ms for repeated requests.
2. *Content Delivery Network* (CDN):
   A CDN distributes copies of data to servers geographically closer to users. Implementation involved configuring Google Cloud Storage buckets with Cloud CDN, allowing data to be accessed from edge servers closer to clients. CDN reduced the average latency from 60 ms to 40 ms for users in remote locations.
3. *Batch Processing*:
   Batch processing combines separate API requests into a single batch to reduce network overhead. Implementation used Google Cloud SDK to create batched requests, such as uploading multiple files simultaneously. Results showed throughput increased from 320 MB/s to 450 MB/s with batch processing.
4. Request Header Optimization:

Request header optimization aims to improve API headers, such as including metadata to avoid unnecessary data retrieval. This was implemented by adding the If-Modified-Since header to ensure only modified data is transmitted. Optimization reduced response times by up to 25% for rarely updated data requests.

4.3 Discussion

The test results indicate that the performance of the Google Cloud Storage API is significantly influenced by the number of simultaneous requests, server location, and the use of optimization technologies. Strategies such as caching and CDN were highly effective in reducing response times and latency. Batch processing also delivered substantial improvements in throughput, making it a strong choice for large-scale data transfer scenarios.

However, these optimization strategies have limitations. For instance, CDN is more effective for data that changes infrequently, while caching requires additional configuration to maintain data consistency. Therefore, the choice of strategy must align with the type of operations and application needs. Overall, API optimization provides significant performance improvements, making Google Cloud Storage a reliable solution for large-scale data management across various scenarios.

5. Conclusion

The findings of the study titled *"Performance Analysis of API in Google Cloud Storage Service Integration"* can be summarized as follows:

1. The performance of the GCS API is influenced by factors such as server location, caching, and the number of simultaneous requests.
2. Optimization strategies like using CDN and caching headers can improve API efficiency by up to 30%.
3. Google Cloud Storage delivers consistent API performance in large-scale data scenarios.

References

Abdullah, M., Matni, B., Musyaffa, R. A., Hamzah, U., Hayani, A. N., & Aribowo, D. (2024). Simulasi Penggunaan Cisco Packet Treaser Untuk Protokol TCP dan UDP Dalam Topologi Jaringan Ring. *Jurnal Teknik Mesin, Industri, Elektro Dan Informatika (JTMEI)*, *3*(2), 240–246.

Aliwijaya, A. (2023). Peluang Pemanfaatan Big Data di Perpustakaan: Sebuah Kajian Literatur. *Media Informasi*, *32*(2), 214–222. https://doi.org/10.22146/mi.v32i2.6388

Angelia Putriana. (2023). Analisis Strategi Bisnis di Era Transformasi Digital. *MUKASI: Jurnal Ilmu Komunikasi*, *2*(3), 223–232. https://doi.org/10.54259/mukasi.v2i3.2105

Bintang Irfansyah, M., Noor Arief, S., & Satya Dian Nugraha, B. (2024). Desain Dan Arsitektur Serverless Cloud Computing Pada Aplikasi Penghitung Kalori Makanan Berbasis Mobile Menggunakan Layanan Google Cloud Platform. *JATI (Jurnal Mahasiswa Teknik Informatika)*, *8*(4), 6090–6097. https://doi.org/10.36040/jati.v8i4.10180

Chandra1, C., Wijaya2, F., Gunawan3, J., Andrew, A., Lee4, Rafferty, J., & Maulana, A. (2024). *Perancangan dan Implementasi RESTful API untuk Aplikasi Mobile Pembelajaran Flora dan Fauna pada Google Cloud Platform*. *4*(1), 58–69. https://doi.org/10.54259/satesi.v4i1.2850

Diantono, P., Susanto, A., Supriyono, A. R., & Prasetyanti, D. N. (2024). *Perbandingan Kinerja Antara Gatling dan Apache JMeter pada Uji Beban RESTful API*. *15*(01), 211–215. https://doi.org/10.35970/infotekmesin.v15i1.2176

Firdaus, M., & Afwani, R. (2024). PENGEMBANGAN RESTFUL API UNTUK APLIKASI KLASIFIKASI JENIS TANAH BERBASIS MOBILE PADA GOOGLE CLOUD (

Restful Api Development For Mobile-Based Soil Type. *Jurnal Teknologi Informasi, Komputer Dan Aplikasinya (JTIKA)Vol. 6, No. 1, Maret 2024, (Terakreditasi Sinta-4, SK No:164/E/KPT/2021)ISSN:2657-0327http://Jtika.If.Unram.Ac.Id/Index.Php/JTIKA/275*, *6*(1), 275–287.

Hidayatut, A., & Harianto, T. (2023). *Analisis Kinerja Mekanisme Caching Redis pada Moodle*. *7*(6), 2889–2894. http://j-ptiik.ub.ac.id

Hiriyannaiah, S., G M, S., & K G, S. (2023). Cloud-based Multi-Modal Information Analytics. In *Cloud-based Multi-Modal Information Analytics*. https://doi.org/10.1201/9781003215974

Ita Ita, Muhlis Tahir, & Edem Vincentius. (2023). Analisis Komparatif Layanan Cloud : Microsoft Azure, Aws, Dan Google Cloud Platform (GCP). *Jurnal Informasi, Sains Dan Teknologi*, *6*(02), 30–43. https://doi.org/10.55606/isaintek.v6i02.127

Khairy, M. D., Defitri, A. N., Hadi, A., Nuzuli, A., Dyakiyyah, A. L., & Heryanto, A. (2023). NetPLG Journal of Network and Computer Applications Pengolahan Data Sensor IOT Dengan Apache Spark Menggunakan Metode Batch Processing. *Journal of Network and Computer*, *2*, 87–106. https://jurnal.netplg.com/jnca

Kurnia Bakti, V., Sutanto, A., & Rizal Arfani, M. (2022). Penerapan Tuya Application Programming Interface (API) pada Sistem IoT Monitoring Suhu Ruang Server. *Jurnal Informatika: Jurnal Pengembangan IT*, *8*(1), 45–49. https://doi.org/10.30591/jpit.v8i1.4764

Mulana, L., Prihandani, K., Rizal, A., Singaperbanga, U., & Abstract, K. (2022). Analisis Perbandingan Kinerja Framework Codeigniter Dengan Express.Js Pada Server RESTful Api. *Jurnal Ilmiah Wahana Pendidikan*, *8*(16), 316–326. https://doi.org/10.5281/zenodo.7067707

Purbasari, N. P., Syakirah, T., & Wijaya, N. (2024). *Pengaruh Kinerja dan Keamanan Penyimpanan Cloud Computing Pada Google Drive*. *3*(5), 1119–1124.

Rahmadyanto, E. P., & Chandra, D. W. (2023). Analisis keamanan Content Delivery Network (CDN) Cloudflare (Studi kasus: Web Hakazon). *Jutisi: Jurnal Ilmiah Teknik Informatika Dan Sistem Informasi*, *12*(3), 1818–1929.

Rahmah, S. A., Elyas, A. H., Informasi, T., Teknik, F., Dahrmawangsa, U., Informasi, S., Teknik, F., & Dahrmawangsa, U. (2024). *EFEKTIVITAS CLOUD COMPUTING DALAM PENYIMPANAN DATA BERBASIS SEKOLAH*. *5*(3), 789–796. https://doi.org/10.46576/djtechno

Saputra, F., Cut, B., & Nilamsari, F. (2023). Analisis Perbandingan Tiga Software Terhadap Pengukuran Quality Of service (QoS) Pada Pengukuran Jaringan Wireless Internet. *Jurnal Teknologi Informasi*, *2*(1), 33–40. http://jurnal.utu.ac.id/JTI/article/view/7275

Setiawan, A., Alexandro, Y., Parhusip, J., Informatika, T., Palangkaraya, U., Yos, J., & Palangkaraya, S. (2025). *ANALISIS DISTRIBUSI RATA-RATA WAKTU RESPON APLIKASI BERBASIS WEB MENGGUNAKAN KURVA NORMAL DAN SIMPANGAN BAKU*. *9*(1), 211–216.

Yoesoep Edhie Rachmad, Rizki Dewantara, Robet, Satrio Junaidi, Mohamad Firdaus, Sulistianto SW, Sepriano, E. (2023). *MASTERING CLOUD COMPUTING (Foundations and Applications Programming)*. PT. Sonpedia Publishing Indonesia.

Zainul, Z., & Romadhan, N. H. (2023). Cloud Storage sebagai Pengganti Arsip Manual dalam Penunjang Aktifitas Sehari-hari. *Kohesi: Jurnal Multidisiplin Saintek*, *1*(6), 10–20.