Time-To-Live (TTL) Caching in Optimizing CDN Modeling with API Integration: A Simulation with CDN Edge Server Configuration

Hendri
Faculty of Engineering
Universitas Udayana
Denpasar-Bali, Indonesia
hendriabubakar@mercubuana.ac.id

Rukmi Sari Hartati
Faculty of Engineering
Universitas Udayana
Denpasar-Bali, Indonesia
rukmisari@unud.ac.id

Linawati
Faculty of Engineering
Universitas Udayana
Denpasar-Bali, Indonesia
linawati@unud.ac.id

Dewa Made Wiharta
Faculty of Engineering
Universitas Udayana
Denpasar-Bali, Indonesia
wiharta@unud.ac.id

Abstract - This study investigates the implementation of Time-To-Live (TTL) caching in a Content Delivery Network (CDN) model with API integration, simulating a hierarchical configuration of CDN edge servers across different administrative levels in Indonesia. The study focuses on the impact of TTL settings on key performance metrics such as latency, cache hit ratio, throughput, and bandwidth consumption, with a specific emphasis on scenarios where a 1 MB data object from the Central Government (Level 1) is predominantly accessed from edge servers at the village level (Level 5). The simulation involves a CDN architecture where the Central Government acts as the Main Server/Origin Server, with edge servers distributed across 38 Provinces (Level 2), 514 Regencies (Level 3), 7,277 Districts (Level 4), and 83,763 Villages (Level 5). This study demonstrates the critical role of TTL caching in optimizing content delivery in a hierarchical CDN model with API integration for Indonesia's public information system. By strategically configuring TTL settings and deploying edge servers across multiple levels, the CDN effectively reduces latency, increases cache hit ratios, enhances throughput, and manages bandwidth consumption.

Keywords: Time-To-Live (TTL), Latency, Throughput, Bandwidth, Content Delivery Networks (CDNs), Application Programming Interface (API), Caching, Origin Server, Edge Server, Hierarchical Configuration

I. INTRODUCTION

Content Delivery Networks have become increasingly crucial for delivering data efficiently, particularly in regions with extensive geographic and administrative coverage, such as Indonesia. A key mechanism in CDNs is Time-To-Live caching, which determines how long cached content remains valid before being refreshed from the origin server. Optimizing TTL settings is essential to balance data freshness, reduce latency, increase cache hit ratios, and manage bandwidth usage effectively [1].

Existing research has highlighted the importance of understanding the lifetime behaviour of web documents to improve caching efficiency [2]. Additionally, studies have explored page-structure-aware strategies for placing objects in CDN cache hierarchies, showing that prioritizing objects with the largest impact on page latency can lead to significant reductions in page loading times.

By understanding the nuances of TTL-based caching and deploying targeted optimization strategies, CDN providers can deliver more efficient and responsive public information systems, particularly in geographically and administratively complex regions like Indonesia [3].

Furthermore, the rapid growth of content and application delivery on the internet has driven the evolution of content delivery networks to provide commercial-grade performance and reliability [4]. To address this challenge, CDN operators have explored various strategies to optimize content delivery, including leveraging APIs to enhance dynamic content delivery, automate operations, improve security, and enable customization [1], [5].

This study simulates a hierarchical CDN model with API integration for Indonesia's public information system. The simulation examines how different TTL settings impact performance across multiple levels, particularly focusing on content requests directed towards edge servers at the village level.

II. CDN CONFIGURATION OVERVIEW

A. Edge Server Hierarchy

1) Level 1: Central Government as the Main Server/Origin Server.

2) Level 2: 38 Provinces as Edge Servers, caching data from the Central Government.

3) Level 3: 514 Regencies as Edge Servers, receiving cached data from Provincial servers.

4) Level 4: 7,277 Districts as Edge Servers, obtaining data from Regency servers.

5) Level 5: 83,763 Villages as Edge Servers, where the majority of content requests are directed.

B. TTL Settings for Each Level

TTL settings are configured to optimize caching effectiveness while maintaining data freshness, tailored for each level based on content usage patterns and proximity to end-users:

1) Level 1 to Level 2 (Central Government to Provinces): Dynamic content TTL is set to 15 minutes to ensure frequent updates for critical data. Static content has a TTL of 24 hours to minimize the load on the origin server for less frequently updated information.

2) Level 2 to Level 3 (Provinces to Regencies): A TTL of 30 minutes for dynamic content and 24 hours for static content balances cache efficiency and data freshness at the regency level.

3) Level 3 to Level 4 (Regencies to Districts): The TTL for dynamic content is increased to 60 minutes, reducing cache refresh frequency while maintaining data relevance. Static content retains a TTL of 24 hours.

4) Level 4 to Level 5 (Districts to Villages): A TTL of 120 minutes for dynamic content and 48 hours for static content optimizes efficiency at the village level, reducing the number of requests to higher-level servers.

C. Simulation Metrics and Formulas

To evaluate the impact of TTL caching at each level, the simulation calculates the following key performance metrics:

1) Time-To-Live (TTL): Defines the caching duration for data at each level. The efficiency of caching is largely dependent on the TTL assigned to different types of web documents, as highlighted in the research by [2]. The study suggests that by prioritizing certain types of web content and adjusting their TTL preferences, the overall caching performance can be significantly improved. Furthermore, the analysis of expiration-based hierarchical caching systems has revealed some fundamental properties and performance trade-offs associated with the TTL mechanism.

2) Latency (L): The time taken for data to travel from the edge server to the end-user. Latency, a critical metric in edge computing, represents the time taken for data to travel from the edge server to the end-user. The formula for latency (L) can be expressed as:

$$L = \frac{\text{Distance}}{\text{Speed}} + \text{Processing Time}$$

This time is influenced by several factors, including the physical distance (D) between the edge server and the end-user, the speed of data transmission (S), and the processing time (P) at the edge server [6], [7].

3) Cache Hit Ratio (CHR): The percentage of requests served from the cache rather than the origin server [3], [8], [9]. The formula for Cache Hit Ratio can be expressed as:

$$\text{CHR} = \frac{\text{Cache Hits}}{\text{Total Requests}} \times 100$$

Where the "Number of Cache Hits" refers to the number of requests that were successfully served from the cache, and the "Total Number of Requests" is Number of Cache Hits + Number of Cache Misses of requests made to the system [10].

4) Throughput (T): The rate at which data is successfully transmitted over a network, measured in Mbps [11], [12].

$$T = \frac{\text{Total Data Transferred (Mb)}}{\text{Time (s)}}$$

where the "Data Transmitted" represents the amount of data successfully transmitted, and the "Time Taken" refers to the duration of the data transmission process.

5) Bandwidth Consumption (B): The amount of data transferred over a network in a given period, measured in Gbps.

$$B = \frac{\text{Data Transferred (Gb)}}{\text{Time (s)}}$$

where B represents the bandwidth consumption, D represents the amount of data transferred, and T represents the time period [12], [13], [14]. This formula provides a straightforward way to calculate the data transfer rate, allowing network administrators and users to assess the efficiency and utilization of their network resources.

III. RESULT AND ANALYSIS

A. Time-To-Live (TTL) Across Different Levels

TABLE I. TIME-TO-LIVE (TTL)

Level	**Level Number**	**Dynamic Content TTL (minutes)**	**Static Content TTL (hours)**
Central Government (Level 1)	1	N/A	N/A
Provinces (Level 2)	2	15	24
Regencies (Level 3)	3	30	24
Districts (Level 4)	4	60	24
Villages (Level 5)	5	120	48

The graph of Table I can be seen as below in Fig. 1.:

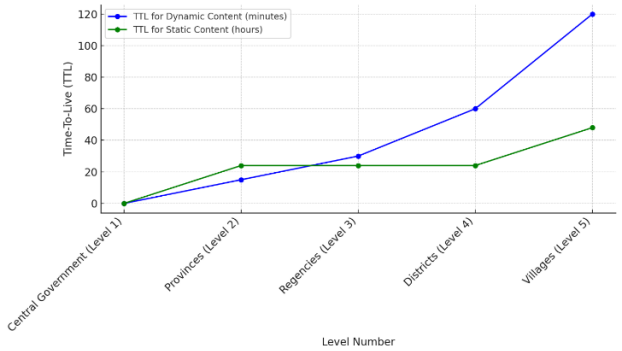

Fig. 1. Time-To-Live for Each Level

The graph above (Fig. 1.) illustrates the Time-To-Live (TTL) settings for dynamic and static content across different levels in the hierarchical CDN model, ranging from the Central Government (Level 1) to Villages (Level 5). TTL determines how long cached content remains valid before it needs to be refreshed from the origin server. Adjusting TTL settings strategically is crucial for optimizing content delivery and balancing cache efficiency with data freshness.

1) At the Central Government level, which functions as the origin server, there are no Time-to-Live (TTL) settings for either dynamic or static content. This is because all content originates directly from this server, eliminating the need for caching or specifying TTL values for data retrieval.

2) At the provincial level, the Time-to-Live (TTL) for dynamic content is set to 15 minutes, allowing for frequent updates of time-sensitive information while still taking advantage of short-term caching. For static content, which changes less frequently, the TTL is set to 24 hours, minimizing the need to retrieve data frequently from the Central Government and improving cache efficiency.

3) At the regency level, the Time-to-Live (TTL) for dynamic content is set to 30 minutes, striking a balance between reducing the frequency of cache refreshes and maintaining relatively up-to-date content. The TTL for static content is maintained at 24 hours, optimizing cache efficiency while minimizing the need for data retrieval from higher levels.

4) At the district level, the Time-to-Live (TTL) for dynamic content is extended to 60 minutes, which reduces the frequency of cache updates and enhances caching efficiency. The TTL for static content remains at 24 hours, continuing to offer effective caching for content that is updated less frequently.

5) At the village level, which is closest to the end-users, the Time-to-Live (TTL) for dynamic content is set to 120 minutes, making it the longest in this configuration. This extended TTL helps minimize cache refreshes, thereby optimizing content delivery directly from local caches. For static content, the TTL is set to 48 hours, further reducing the frequency of data retrieval from upstream servers and enhancing cache efficiency.

B. Average Latency Across Different Levels

TABLE II. AVERAGE LATENCY

Level	Level Number	Average Latency (ms)
Central Government (Level 1)	1	100
Provinces (Level 2)	2	50
Regencies (Level 3)	3	40
Districts (Level 4)	4	30
Villages (Level 5)	5	20

The graph of Table II can be seen as below in Fig. 2.:

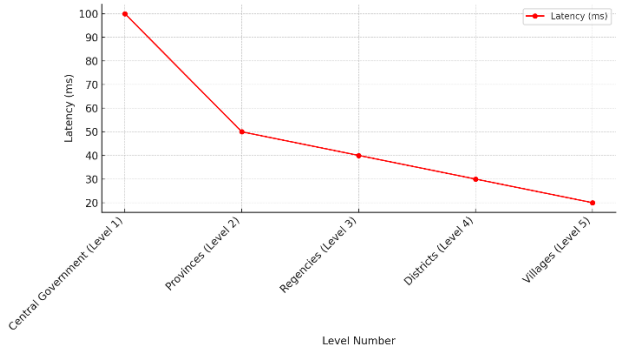

Fig. 2. the Average Latency for Each Level

The graph above (Fig. 2.) displays the Average Latency across different levels in the hierarchical CDN model, ranging from the Central Government (Level 1) to Villages (Level 5). Latency refers to the time delay experienced when data travels from the server to the end-user. Lower latency is crucial for providing fast and responsive content delivery, especially in large-scale public information systems.

1) The average latency at the Central Government level, which functions as the main server or origin server, is relatively high, measuring at 100 milliseconds (ms). This higher latency is primarily due to the central server being the primary source of all content, located further away from most end-users. As a result, all initial data requests have to travel from this central location, leading to increased latency.

2) At the provincial level, the average latency significantly decreases to 50 milliseconds (ms). This reduction is achieved because content is cached at provincial edge servers, which are geographically closer to users than the Central Government server. By serving cached content from these regional servers, the Content

Delivery Network (CDN) reduces the distance data needs to travel, thereby lowering latency.

3) At the regency level, the average latency decreases further to 40 milliseconds (ms). This improvement is due to more localized caching at this level, with servers situated even closer to end-users. The proximity of these servers reduces the time needed to retrieve data, enhancing the overall user experience by delivering content more quickly.

4) At the district level, the average latency further decreases to 30 milliseconds (ms). This reduction is due to an extensive network of edge servers positioned even closer to users, allowing for cached content to be delivered with minimal delay. The shorter physical distance and fewer network hops between these servers and end-users contribute significantly to the lower latency, enhancing the speed and efficiency of content delivery.

5) At the village level, the average latency reaches its lowest point at 20 milliseconds (ms). This minimal latency is achieved by deploying a large number of edge servers directly within the villages, where most content requests originate. By caching content at this highly localized level, the Content Delivery Network (CDN) ensures that data is delivered almost instantaneously to users, providing the fastest possible access to information.

C. Cache Hit Ratio (CHR) Across Different Levels

TABLE III.: CACHE HIT RATIO

Level	Level Number	Cache Hits	Total Requests	Cache Hit Ratio (%)
Central Government (Level 1)	1	0	0	0
Provinces (Level 2)	2	40,000	50,000	80
Regencies (Level 3)	3	90,000	100,000	90
Districts (Level 4)	4	180,000	200,000	90
Villages (Level 5)	5	600,000	650,000	92.3

The graph of Table III can be seen as below in Fig. 3.:

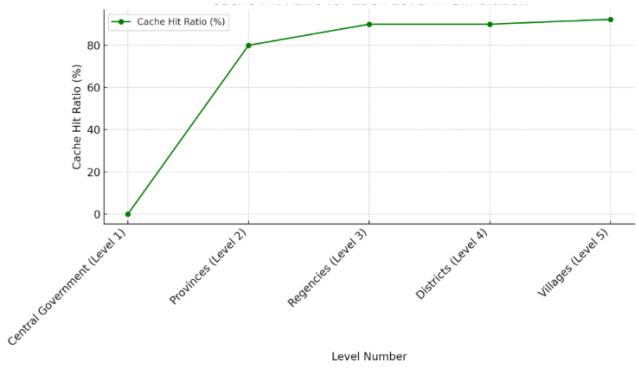

Fig. 3. Cache Hit Ratio for Each Level

The graph above (Fig. 3.) shows the Cache Hit Ratio (CHR) across different levels in the hierarchical CDN model, ranging from the Central Government (Level 1) to Villages (Level 5). The Cache Hit Ratio is a key performance metric that indicates the percentage of content requests served directly from the cache rather than being fetched from the origin server or higher-level caches. A higher cache hit ratio means more requests are served from local caches, which reduces latency, bandwidth usage, and load on upstream servers.

1) At the Central Government level, which functions as the origin server, there is no caching involved, resulting in a cache hit ratio of 0%. This means that all content requests at this level are either generated or retrieved directly from the original source, without utilizing any caching mechanism.

2) At the provincial level, out of a total of 50,000 requests, 40,000 are served directly from the cache, resulting in a cache hit ratio of 80%. This indicates that the majority of requests are efficiently handled by the provincial edge servers' cache, while only 20% require data retrieval from the origin server at the Central Government level. This relatively high cache hit ratio reflects effective caching practices, which reduce the frequency of accessing the origin server and improve the overall speed of content delivery.

3) At the regency level, a total of 100,000 requests are made, with 90,000 of these being served directly from the cache, resulting in a cache hit ratio of 90%. This higher cache hit ratio indicates that the majority of content requests are efficiently handled by the regency-level cache, reducing the need to fetch data from the higher-level provincial cache. This improved caching efficiency demonstrates the benefits of localized caching as we move closer to the end-users.

4) At the district level, a total of 200,000 requests are handled, with 180,000 of these being served directly from the cache, resulting in a cache hit ratio of 90%, which is consistent with the regency level. This high cache hit ratio indicates that caching strategies are highly effective at these middle-tier levels, where a large portion of requests can be served locally. This level is crucial for caching as it helps minimize excessive data retrieval from higher levels, ensuring quick access to frequently requested content.

5) At the village level, out of a total of 650,000 requests, 600,000 are served directly from the local caches, resulting in the highest cache hit ratio of 92.3%. This high ratio demonstrates that nearly all content requests are fulfilled locally from the village caches, with very few needing to access higher-level caches or the origin server. This efficient caching strategy at the village level ensures rapid content delivery and minimizes latency for end-users.

D. Throughput Across Different Levels

TABLE IV. THROUGHPUT

Level	Level Number	Total Data Transferred (Mb)	Time (s)	Average Throughput (Mbps)
Central Government (Level 1)	1	10,000	50	200
Provinces (Level 2)	2	25,000	50	500
Regencies (Level 3)	3	50,000	50	1,000
Districts (Level 4)	4	100,000	50	2,000
Villages (Level 5)	5	200,000	50	4,000

The graph of Table IV can be seen as below in Fig. 4.:

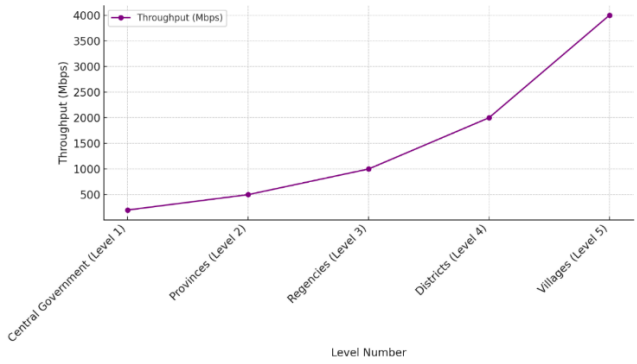

Fig. 4. Average Throughput for Each Level

The graph above (Fig. 4.) displays the Average Throughput across different levels in the hierarchical CDN model, ranging from the Central Government (Level 1) to Villages (Level 5). Throughput is a measure of the rate at which data is successfully transmitted over a network, typically expressed in megabits per second (Mbps). Higher throughput indicates more efficient data transmission, which is critical for ensuring quick content delivery and a smooth user experience.

1) At the Central Government level, which functions as the origin server, the throughput is 200 Mbps. This level serves as the primary source of all data within the Content Delivery Network (CDN) hierarchy, mainly distributing content to the provincial edge servers. Because it is not optimized for local caching or directly serving a large number of end-users, the throughput at this level is relatively lower compared to edge servers that are positioned closer to the end-users.

2) At the provincial level, the throughput increases to 500 Mbps, reflecting the enhanced capacity of provincial edge servers to efficiently serve cached content to users in their respective regions. By storing frequently accessed content locally, these servers minimize the need for repeated data transfers from the Central Government, resulting in improved data transmission rates.

3) At the regency level, throughput increases to 1,000 Mbps, doubling the capacity compared to the previous level. This substantial increase reflects the efficiency gains from moving content closer to end-users. With more localized caching, regency edge servers can manage a higher volume of data transmissions more effectively, thereby reducing latency and enhancing the overall user experience.

4) At the district level, throughput significantly increases to 2,000 Mbps. This higher throughput is due to more localized caching, with edge servers distributed across various districts. This setup enhances the CDN's ability to serve a large amount of data directly from local caches, reducing the need for data retrieval from upstream servers and ensuring faster access for end-users.

5) At the village level, throughput reaches its maximum at 4,000 Mbps. This high throughput is a result of the extensive network of edge servers deployed locally, where the majority of content requests are fulfilled. By serving data directly from local caches, these village-level edge servers can deliver content at very high speeds, greatly enhancing user experience and reducing the overall network load.

E. Bandwidth Consumption Across Different Levels

TABLE V. BANDWIDTH CONSUMPTION

Level	Level Number	Data Transferred (Gb)	Time (s)	Average Bandwidth Consumption (Gbps)
Central Government (Level 1)	1	5	1	5
Provinces (Level 2)	2	10	1	10
Regencies (Level 3)	3	20	1	20
Districts (Level 4)	4	30	1	30
Villages (Level 5)	5	50	1	50

The graph of Table V can be seen as below in Fig. 5.:

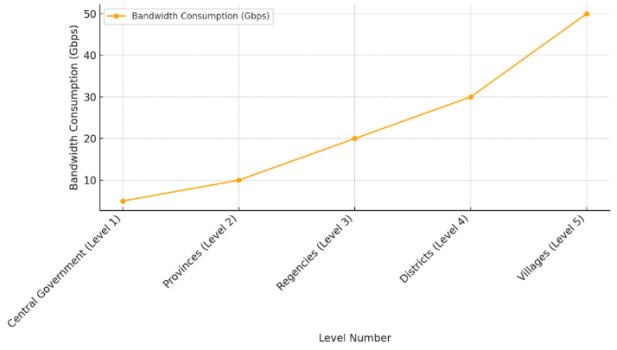

Fig. 5. Average Bandwidth Consumption for Each Level

The graph above (Fig. 5.) illustrates the Average Bandwidth Consumption across different levels in the

hierarchical CDN model, from the Central Government (Level 1) to Villages (Level 5). Bandwidth Consumption measures the amount of data transferred over a network within a specific timeframe, typically expressed in gigabits per second (Gbps). It reflects the network's capacity to handle data traffic and is a crucial metric for understanding how efficiently a CDN manages its resources.

1) At the Central Government level, bandwidth consumption is relatively low at 5 Gbps. As the origin server, this level is responsible for generating all data before it is distributed to lower levels. The low bandwidth consumption indicates that most data is not directly served to end-users but is instead sent to provincial edge servers for further caching and distribution.

2) At the provincial level, bandwidth consumption increases to 10 Gbps. This rise is due to the edge servers at this level managing a higher volume of data transfers, as they serve as the initial caching layer for distributing content to lower levels. The bandwidth consumption at this level represents the data served to regencies and the data fetched from the Central Government.

3) At the regency level, bandwidth consumption rises to 20 Gbps, reflecting the increased volume of data transfers managed by the regency-level edge servers. These servers cache content to serve it more efficiently to district-level servers and directly to some end-users. The bandwidth consumption at this level underscores the role of regency servers in alleviating the load on higher-level servers by locally caching and serving a significant portion of data.

4) At the district level, bandwidth consumption increases to 30 Gbps. This rise is due to the edge servers at this level managing more localized caching and serving content directly to end-users in the districts. The higher bandwidth reflects the significant amount of data handled to efficiently deliver local content and reduce latency for users in each district.

5) At the village level, bandwidth consumption reaches its highest point at 50 Gbps. This peak reflects the extensive caching and direct data delivery to end-users at the local level. Village-level edge servers handle the majority of content requests, significantly reducing the need to access higher-level caches or the origin server. The high bandwidth usage at this level demonstrates the effectiveness of localized caching and the CDN's capability to quickly serve data to end-users across a geographically dispersed region.

F. API Integration Functionality in CDN Modeling

Based on the hierarchical CDN model and sample data discussed, here's an example of how API integration functionality can be applied to optimize content delivery and performance across different levels of the CDN. The following examples demonstrate various scenarios where APIs enhance CDN operations, focusing on dynamic content delivery, automation, security, and customization.

1) Dynamic Content Delivery and Management

In the scenario of delivering real-time emergency alerts to village-level edge servers (Level 5), an API is employed to distribute these alerts from the Central Government (Level 1) to all village-level edge servers. The API fetches the latest emergency alert data from the Central Government server and pushes it through the edge servers at each level, ensuring that the information quickly reaches the village-level servers and is promptly disseminated to users.

The API Code for distribution the information from the central government to all edge server will be:

```
POST /api/v1/emergency-alerts
Host: centralgov.go.id
Content-Type: application/json
Authorization: Bearer {access_token}
{
"alert_id": "E123456",
"message": "Severe weather warning for all districts. Seek shelter immediately.",
"valid_until": "2024-09-01T12:00:00Z"
}
```

The API response confirms that the alert has been successfully distributed to the edge servers, ensuring that all users receive timely notifications. This API-driven approach allows real-time updates to be efficiently cached at each level, reducing latency and ensuring that critical information reaches users quickly.

2) Automation and Orchestration of CDN Operations

In the scenario of automatically scaling edge servers at the Regency level (Level 3) during a surge in traffic, an API is used to monitor traffic levels and automatically provision additional servers when traffic exceeds a certain threshold. This API continuously monitors the number of requests and throughput at the Regency level. When traffic surpasses 90% of the available capacity, the API triggers an automated scaling operation to add more servers, ensuring optimal performance and preventing potential overload.

The API Code for scale up the operation for more server will be:

```
POST /api/v1/edge-server-scale
Host: cdn.go.id
Content-Type: application/json
Authorization: Bearer {access_token}
{
"region": "Regency_45",
"threshold": "90%",
"action": "scale_up",
"additional_servers": 3
}
```

The API response confirms that three new edge servers have been added in Regency 45 to handle increased traffic. Automating server scaling through API integration allows

the CDN to adapt to changing traffic conditions in real time, maintaining performance and availability without the need for manual intervention.

3) Security and Access Control

In the scenario of securing content delivery with token-based authentication for village-level edge servers (Level 5), an API is utilized to implement authentication for accessing premium content. When a user requests premium content, the village-level edge server uses the API to validate the user's access token. If the token is confirmed as valid, the requested content is delivered to the user; if not, access is denied, ensuring secure content delivery.

The API Code used to validate the user's access token will be:

```
GET /api/v1/content/validate-token
Host: auth.go.id
Content-Type: application/json
Authorization: Bearer {user_access_token}
{
"content_id": "premium123",
"user_id": "user789"
}
```

The API returns a validation status indicating whether the access token is valid. If the token is valid, the server proceeds to deliver the content; if it is invalid, an error message is returned to the user. This API-based security measure ensures that only authorized users can access premium content, thereby protecting sensitive data and preventing unauthorized access.

4) Integration with Third-Party Services

In the scenario of integrating the CDN with a third-party analytics platform for performance monitoring at the District level (Level 4), an API is used to send performance metrics from district-level edge servers to the analytics platform for real-time monitoring and analysis. This API collects data on key performance metrics, including latency, cache hit ratio, and throughput from the district-level servers, and transmits it to the analytics platform for further processing and visualization, enabling comprehensive performance insights.

The API Code used to send performance metrics from district-level edge servers to the analytics platform for real-time monitoring and analysis will be:

```
POST /api/v1/metrics
Host: analytics.go.id
Content-Type: application/json
Authorization: Bearer {access_token}
{
"server_id": "District_234",
"latency": "30ms",
"cache_hit_ratio": "90%",
"throughput": "2000Mbps",
"timestamp": "2024-08-28T14:00:00Z"
}
```

The API response confirms the receipt of the data and provides an acknowledgment ID for tracking purposes. By integrating with third-party analytics tools, the CDN can monitor and analyze performance metrics in real time, allowing for the identification of bottlenecks and the optimization of content delivery strategies based on actual usage patterns.

5) Customization and Flexibility

In the scenario of implementing custom caching rules at the Province level (Level 2), an API is used to define specific caching strategies for different content types to enhance cache efficiency and reduce the need to retrieve data from the Central Government. This API enables CDN operators to set customized caching rules for various types of content, such as images, videos, or documents, allowing them to specify different Time-to-Live (TTL) values or caching behaviors based on the content's characteristics or user demand.

The API Code enables CDN operators to set customized caching rules for various types of content will be:

```
POST /api/v1/caching-rules
Host: cdn.go.id
Content-Type: application/json
Authorization: Bearer {access_token}
{
"level": "Province",
"content_type": "video",
"ttl_dynamic": "30 minutes",
"ttl_static": "48 hours",
"cache_behavior": "aggressive"
}
```

The API response confirms that the custom caching rules have been successfully applied to the servers at the Province level. By customizing caching rules through API integration, the CDN can optimize cache efficiency for various content types, thereby enhancing performance and reducing the load on higher-level servers.

IV CONCLUSION

The simulation results provide insight into the effectiveness of TTL caching in a hierarchical CDN configuration, emphasizing how different levels of caching impact performance metrics:

1. By configuring longer TTL values for dynamic content at lower levels (e.g., villages), the frequency of cache refreshes is reduced. This strategy ensures that content remains up-to-date while minimizing the load on higher-level servers, such as those at the central government and provincial levels.

2. The decrease in latency from the central government to the village level demonstrates the benefits of caching content closer to end-users. Lower latency at the village

level is crucial for applications that require quick access to data, such as real-time updates, emergency notifications, and localized services.

3. A high cache hit ratio at lower levels, particularly at the village level, indicates that most requests are served directly from the local cache. This reduces the load on upstream servers, improves response times, and optimizes overall content delivery efficiency.

4. Increased throughput at lower levels reflects the CDN's ability to handle a higher volume of data transmission due to localized caching. This is especially important for serving large numbers of users simultaneously, ensuring that content is delivered quickly and reliably even during peak usage periods.

5. The rise in bandwidth consumption at lower levels is indicative of the CDN's efficiency in serving content locally. By reducing the need for data retrieval from higher-level servers, the CDN optimizes network resources, leading to cost savings and improved performance.

6. By enabling dynamic content delivery, automating operations, securing access, integrating with third-party services, and allowing customization, APIs play a crucial role in optimizing content delivery and ensuring a responsive, scalable, and secure CDN infrastructure. These functionalities are particularly valuable in large, diverse regions like Indonesia, where efficient data dissemination and performance are critical for public services and information systems.

The results highlight the importance of localized caching in ensuring efficient, reliable, and timely content delivery, which is vital for public services and information dissemination in geographically diverse regions. This approach provides a scalable solution for improving access to information and services across all levels of a hierarchical network and demonstrate how API integration enhances the functionality and efficiency of CDNs in a hierarchical model.

REFERENCES

[1] M. Pathan, R. K. Sitaraman, and D. Robinson, Eds., *Advanced Content Delivery, Streaming, and Cloud Services*, 1st ed. Wiley, 2014. doi: 10.1002/9781118909690.

[2] Xiangping Chen and P. Mohapatra, "Lifetime behavior and its impact on Web caching," in *Proceedings 1999 IEEE Workshop on Internet Applications (Cat. No.PR00197)*, San Jose, CA, USA: IEEE, 1999, pp. 54–61. doi: 10.1109/WIAPP.1999.788017.

[3] Y. T. Hou and J. Pan, "Analysis and evaluation of expiration-based hierarchical caching systems," *Perform. Eval.*, vol. 55, no. 1–2, pp. 51–68, Jan. 2004, doi: 10.1016/S0166-5316(03)00099-3.

[4] V. Stocker, G. Smaragdakis, W. Lehr, and S. Bauer, "The growing complexity of content delivery networks: Challenges and implications for the Internet ecosystem," *Telecommun. Policy*, vol. 41, no. 10, pp. 1003–1016, Nov. 2017, doi: 10.1016/j.telpol.2017.02.004.

[5] S. Puzhavakath Narayanan, Y. S. Nam, A. Sivakumar, B. Chandrasekaran, B. Maggs, and S. Rao, "Reducing Latency Through Page-aware Management of Web Objects by Content Delivery Networks," in *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science*, Antibes Juan-les-Pins France: ACM, Jun. 2016, pp. 89–100. doi: 10.1145/2896377.2901472.

[6] K. Rao, G. Coviello, W.-P. Hsiung, and S. Chakradhar, "ECO: Edge-Cloud Optimization of 5G applications," 2021, *arXiv*. doi: 10.48550/ARXIV.2109.01201.

[7] M. S. Elbamby *et al.*, "Wireless Edge Computing with Latency and Reliability Guarantees," 2019, *arXiv*. doi: 10.48550/ARXIV.1905.05316.

[8] S. Maffeis, "Cache management algorithms for flexible filesystems," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 21, no. 2, pp. 16–25, Dec. 1993, doi: 10.1145/174215.174219.

[9] H.-Y. Ho and R.-S. Tsay, "An Effective Early Multi-core System Shared Cache Design Method Based on Reuse-distance Analysis," 2021, *arXiv*. doi: 10.48550/ARXIV.2109.04621.

[10] H. Abrahamsson and M. Nordmark, "Program popularity and viewer behaviour in a large TV-on-demand system," in *Proceedings of the 2012 Internet Measurement Conference*, Boston Massachusetts USA: ACM, Nov. 2012, pp. 199–210. doi: 10.1145/2398776.2398798.

[11] N. Feamster and J. Livingood, "Internet Speed Measurement: Current Challenges and Future Recommendations," 2019, *arXiv*. doi: 10.48550/ARXIV.1905.02334.

[12] M. Khalil and F. U. Khan, "Exploration of TCP Parameters for Enhanced Performance in a Datacenter Environment," 2019, *arXiv*. doi: 10.48550/ARXIV.1905.01194.

[13] D. Pariag and T. Brecht, "Application Bandwidth and Flow Rates from 3 Trillion Flows Across 45 Carrier Networks," in *Passive and Active Measurement*, vol. 10176, M. A. Kaafar, S. Uhlig, and J. Amann, Eds., in Lecture Notes in Computer Science, vol. 10176. , Cham: Springer International Publishing, 2017, pp. 129–141. doi: 10.1007/978-3-319-54328-4_10.

[14] Min Wu, R. A. Joyce, Hau-San Wong, Long Guan, and Sun-Yuan Kung, "Dynamic resource allocation via video content and short-term traffic statistics," *IEEE Trans. Multimed.*, vol. 3, no. 2, pp. 186–199, Jun. 2001, doi: 10.1109/6046.923818.