JEMSI:
Jurnal Ekonomi Manajemen Sistem Informasi
E-ISSN: 2686-5238
P-ISSN: 2686-4916
https://dinastirev.org/JEMSI dinasti.info@gmail.com +62 811 7404 455
DINASTI REVIEW

DOI: https://doi.org/10.38035/jemsi.v6i2
https://creativecommons.org/licenses/by/4.0/

Optimizing CDN Modeling with API Integration Using Time To-Live (TTL) Caching Technique

Hendri Hendri[1], Rukmi Sari Hartati[2], Linawati Linawati[3], Dewa Made Wiharta[4]
[1]Universitas Udayana, Denpasar-Bali, Indonesia, hendriabubakar@mercubuana.ac.id
[2]Universitas Udayana, Denpasar-Bali, Indonesia, rukmisari@unud.ac.id
[3]Universitas Udayana, Denpasar-Bali, Indonesia, linawati@unud.ac.id
[4]Universitas Udayana, Denpasar-Bali, Indonesia, wiharta@unud.ac.id

Corresponding Author: hendriabubakar@mercubuana.ac.id[1]

Abstract: This research examines the implementation of Time-To-Live (TTL) caching within a Content Delivery Network (CDN) model that incorporates API integration, structured to simulate a hierarchical configuration of CDN edge servers across Indonesia's administrative tiers. The analysis centers on the influence of TTL configurations on critical performance metrics—namely latency, cache hit ratio, throughput, and bandwidth consumption. Special focus is placed on scenarios in which a 1 MB data object originating from the Central Government (Level 1) is primarily accessed through edge servers positioned at the village level (Level 5). The simulation envisions a CDN architecture where in the Central Government functions as the Main Server/Origin Server, with edge servers extending across 38 provinces (Level 2), 514 regencies (Level 3), 7,277 districts (Level 4), and 83,763 villages (Level 5).

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

INTRODUCTION

Content Delivery Networks (CDNs) are increasingly vital for the efficient delivery of data, particularly in regions with expansive geographic and administrative scopes, such as Indonesia (Allwörden, 2023; Buyya et al., 2008; Pathan et al., 2014). A fundamental feature of CDNs is Time-To-Live (TTL) caching, a mechanism that dictates the duration cached content remains valid before it is refreshed from the origin server. Optimizing TTL configurations is essential to maintain data freshness, minimize latency, enhance cache hit ratios, and efficiently manage bandwidth (Basu et al., 2018; Elsayed et al., 2024; J. Liu et al., 2022).

Current research emphasizes the significance of understanding the lifespan characteristics of web documents to enhance caching effectiveness (H. Liu & Han, 2021). Furthermore, studies investigating page-structure-aware caching strategies within CDN hierarchies have demonstrated that prioritizing high-impact objects in the cache can substantially reduce page load times (Chen et al., 2021; Kamiyama et al., 2016; H. Liu & Han, 2021). By deepening their understanding of TTL-based caching mechanisms and employing focused optimization approaches (Basu et al., 2018; Elsayed et al., 2024; Goseling & Simeone, 2019; J. Liu et al., 2022), CDN providers can enhance the efficiency and responsiveness of public information systems, which is particularly beneficial in geographically and administratively complex areas like Indonesia.

API Integration Functionality in CDN Modelling

The exponential increase in internet content and application delivery has driven the evolution of content delivery networks (CDNs) to ensure high standards of performance and reliability (Stocker et al., 2017). In response to these demands, CDN operators have adopted diverse strategies aimed at optimizing content distribution, notably through the integration of Application Programming Interfaces (APIs) to improve dynamic content delivery, automate processes, bolster security, and facilitate customization (Allwörden, 2023; Bagga, 2023; Carneiro & Schmelmer, 2016; Chari et al., 2023; De, 2023).

By applying the hierarchical CDN model and relevant sample data, API integration can be effectively utilized to enhance content delivery and performance across various CDN layers. The examples presented here illustrate several scenarios in which APIs enhance CDN functionality, with a particular focus on dynamic content management, automation, security measures, and customizable features (Allwörden, 2023; Chari et al., 2023; Dale, 2019).

1. Dynamic Content Delivery and Management

Scenario: Delivering real-time emergency alerts to village-level edge servers (Level 5)

a. **API Call:** An API is used to deliver emergency alerts from the Central Government (Level 1) to all village-level edge servers.
b. **Functionality:** The API fetches the latest emergency alert data from the Central Government server and pushes it to the edge servers at each level, ultimately reaching the village-level servers.
c. **API Code:**

```
POST /api/v1/emergency-alerts
Host: centralgov.example.com
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"
}
```

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

2. Automation and Orchestration of CDN Operations

Scenario: Automatically scaling edge servers at the Regency level (Level 3) during a surge in traffic

a. **API Call:** An API is used to monitor traffic levels and automatically provision additional edge servers at the Regency level when traffic exceeds a certain threshold.
b. **Functionality:** The API continuously monitors the number of requests and throughput at the Regency level. When traffic exceeds 90% of capacity, the API triggers an automated scaling operation to add more servers.
c. **API Code:**

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

d. **Response:** The API response confirms the addition of three new edge servers in Regency 45 to handle increased traffic.
e. **Impact:** Automating server scaling through API integration ensures that the CDN can adapt to changing traffic conditions in real time, maintaining performance and availability without manual intervention.

3. Security and Access Control

Scenario: Securing content delivery with token-based authentication for village-level edge servers (Level 5)

a. **API Call:** An API is used to implement token-based authentication for accessing premium content at the village level.
b. **Functionality:** When a user requests premium content, the edge server at the village level uses an API to validate the user’s access token. If the token is valid, the content is delivered; otherwise, access is denied.
c. **API Code:**

```
GET /api/v1/content/validate-token.
Host: auth.example.com
Content-Type: application/json
Authorization: Bearer {user_access_token}
{
"Content_id": "premium123",
"User_id": "user789"
}
```

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

4. Integration with Third-Party Services

Scenario: Integrating CDN with a third-party analytics platform for performance monitoring at the district level (Level 4)

a. **API Call:** An API is used to send performance metrics from district-level edge servers to a third-party analytics platform for real-time monitoring and analysis.
b. **Functionality:** The API collects data on key performance metrics such as latency, cache hit ratio, and throughput from the district-level servers and sends it to the analytics platform for further processing and visualization.
c. **API Code:**

```
POST /api/v1/metrics.
Host: analytics.example.com
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"
}
```

d. **Response:** The API response confirms receipt of the data and provides an acknowledgment ID for tracking.
e. **Impact:** By integrating with third-party analytics tools, the CDN can monitor and analyze performance metrics in real time, identifying bottlenecks and optimizing content delivery strategies based on actual usage patterns.

5. Customization and Flexibility

Scenario: Implementing custom caching rules for frequently accessed content at the province level (Level 2)

a. **API Call:** An API is used to define custom caching rules for specific content types at the province level to optimize cache efficiency and reduce data retrieval from the Central Government.
b. **Functionality:** The API allows CDN operators to set custom caching rules for content such as images, videos, or documents, specifying different TTLs or cache behaviors based on content characteristics or user demand.
c. **API Code:**

```
POST /api/v1/caching-rules.
Host: cdn.example.com
Content-Type: application/json
Authorization: Bearer {access_token}
{
"level": "Province", "content_type": "video",
"Ttl_dynamic": "30 minutes",
"Ttl_static": "48 hours",
"Cache_behavior": "aggressive"
}
```

d. **Response:** The API response confirms that the custom caching rules have been successfully applied to the province level servers.
e. **Impact:** Customizing caching rules through API integration allows the CDN to optimize cache efficiency for different content types, improving performance and reducing the load on higher-level servers.

METHOD

This study presents a simulation of a hierarchical content delivery network (CDN) model integrated with APIs, designed for Indonesia’s public information system. The research investigates the influence of various Time-to-Live (TTL) configurations on performance across multiple CDN layers, with a particular emphasis on content requests routed to edge servers situated at the village level.

In hierarchical CDN architectures, servers are structured in tiers to enhance the efficiency of content distribution across spatially diverse regions. Origin servers, located at the top tier, store the primary content, which is then propagated to cache servers positioned closer to end-users (Boukerche & Gu, 2011; Elsayed et al., 2024; Li & Wang, 2020; H. Liu & Han, 2021). This multi-tiered configuration forms a cascading framework in which each layer retains a cached copy of the data, thereby alleviating demand on the origin servers and accelerating data delivery speeds. Such an architecture effectively balances server load, mitigates network congestion, and improves response times by ensuring that frequently accessed content is readily available in proximity to users (Bolla et al., 1999; Korzun & Gurtov, 2014; Li & Wang, 2020; H. Liu & Han, 2021).

CDN Configuration Overview:

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.

TTL Settings for Each Level:

TTL configurations are systematically optimized to balance caching efficiency with data freshness, customized to content demand and user proximity at each level.

For Level 1 to Level 2 (Central Government to Provinces), dynamic content is assigned a TTL of 15 minutes, enabling timely updates for critical information, while static content is set to 24 hours to reduce origin server load for infrequently updated data.

From Level 2 to Level 3 (Provinces to Regencies), TTLs are set at 30 minutes for dynamic and 24 hours for static content, achieving an equilibrium between cache effectiveness and data freshness at the regency level.

At Level 3 to Level 4 (Regencies to Districts), dynamic content has a TTL of 60 minutes, allowing less frequent cache refreshes while retaining data relevance, with static content continuing at 24 hours.

Finally, for Level 4 to Level 5 (Districts to Villages), dynamic content is given a TTL of 120 minutes, and static content 48 hours, maximizing cache efficiency at the village level and decreasing requests to higher-level servers.

Simulation Metrics and Formulas

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

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 (Xiangping Chen & Mohapatra, 1999). 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.

Latency (L) is 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:

$$L = \frac{Distance}{Speed} + Processing\ Time \qquad (1)$$

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 (Rao et al., 2021).

Cache Hit Ratio (CHR) is the percentage of requests served from the cache rather than the origin server (Maffeis, 1993). The formula for Cache Hit Ratio can be expressed as:

$$CHR = \frac{Cache\ Hits}{Total\ Request} \times 100\% \qquad (2)$$

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 (Abrahamsson & Nordmark, 2012).

Throughput (T) is the rate at which data is successfully transmitted over a network, measured in Mbps (Feamster & Livingood, 2019; Khalil & Khan, 2019).

$$T = \frac{Total\ Data\ Transferred\ (Mb)}{Time\ (s)} \qquad (3)$$

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

Bandwidth Consumption (B) is the amount of data transferred over a network in a given period, measured in Gbps.

$$B = \frac{Data\ Transferred\ (Gb)}{Time\ (s)} \qquad (4)$$

where B represents the bandwidth consumption, D represents the amount of data transferred, and T represents the time period (Khalil & Khan, 2019; Pariag & Brecht, 2017). 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.

RESULTS AND DISCUSSION

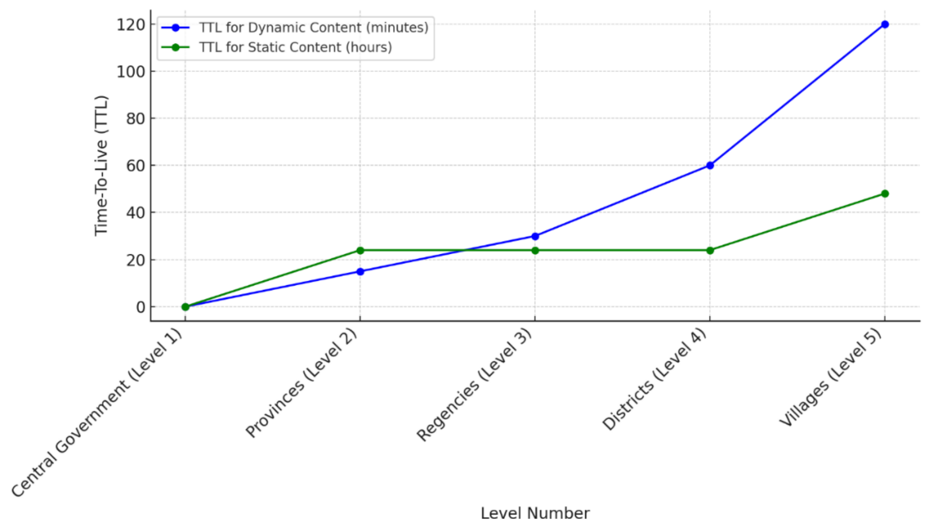
1. Time-To-Live (TTL) Across Different Levels:

Table 1. Time-To-Live (TTL)
Simulation Result Across Different Levels

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

Source: Research Data

Graph: Time-To-Live for Each Level in Simulation
X-Axis: Level Number (Central Government to Villages)
Y-Axis: TTL in minutes (dynamic) and hours (static)



Source: Research Result

Figure 1. Time-To-Live for Each Level

Figure 1 illustrates the Time-To-Live (TTL) settings applied to dynamic and static content across varying hierarchical levels within a CDN model, extending from the Central Government (Level 1) down to Villages (Level 5). TTL defines the duration that cached content remains valid before requiring a refresh from the origin server. Strategically configuring TTL is vital for enhancing content delivery performance while balancing cache efficiency and data freshness.

At the Central Government level, which functions as the origin server, TTL settings are unnecessary as all content is sourced directly from this point without caching.

Moving to the Provincial level, a TTL of 15 minutes is assigned to dynamic content, supporting frequent updates for time-sensitive data while still enabling short-term caching. Static content, which experiences less frequent changes, is assigned a TTL of 24 hours to minimize the need for recurrent data retrieval from the Central Government.

The Regency level features a TTL of 30 minutes for dynamic content, striking a balance between reducing cache refresh intervals and maintaining reasonably fresh content. Static content remains at a 24-hour TTL, optimizing cache performance by reducing the frequency of data retrieval from higher tiers.

Further at the District level, dynamic content TTL is further extended to 60 minutes, decreasing cache update frequency to improve caching efficiency, while the static content TTL continues at 24 hours to maintain effective caching of less frequently updated data.

At the Village level, which is positioned nearest to end-users, dynamic content is configured with a TTL of 120 minutes, the longest duration in this model. This extended TTL reduces the need for frequent cache updates, facilitating content delivery directly from local caches. For static content, the TTL is set to 48 hours, further enhancing cache efficiency by minimizing upstream data retrieval.

2. Average Latency Across Different Levels:

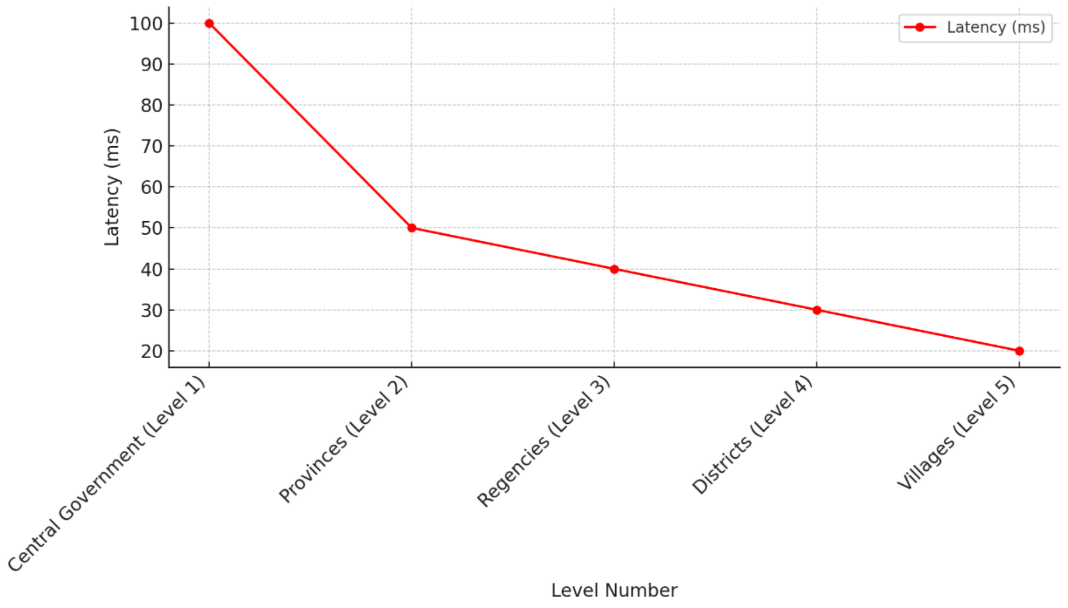
Table 2. Average Latency Simulation Result Across Different Levels

Level	Level Number	Average Latency (ms)
Central Government (Level 1)	1	100

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

Source: Research Data

Graph: Average Latency for Each Level in Simulation
X-Axis: Level Number (Central Government to Villages)
Y-Axis: Average Latency in milliseconds (ms)



Source: Research Result

Figure 2. The Average Latency for Each Level

Figure 2 illustrates the Average Latency across various levels within a hierarchical Content Delivery Network (CDN) model, encompassing layers from the Central Government (Level 1) to Village Level (Level 5). Latency, defined as the delay in data transmission from server to end-user, is a critical metric in ensuring efficient and responsive content delivery, especially in large-scale public information networks.

At the Central Government level, which functions as the primary or origin server, latency is relatively elevated at 100 ms. This higher latency arises because the central server acts as the main content source, located at a considerable distance from most end-users. Initial data requests must thus traverse from this central node, incurring greater latency.

Moving to the Provincial level, average latency is reduced significantly to 50 ms. This decrease can be attributed to content caching on provincial edge servers, which are geographically closer to end-users compared to the central server. By distributing cached content through these regional nodes, the CDN diminishes the physical distance data must travel, thereby lowering latency.

At the Regency level, latency drops further to 40 ms, highlighting the positive impact of increasingly localized caching. Edge servers at the regency level, situated even nearer to users, allow for quicker data retrieval, thereby enhancing user experience with more rapid content delivery.

Down to the District level, average latency falls to 30 ms, owing to the widespread deployment of edge servers that cache and serve content closer to users. The reduced physical distance and fewer network hops required between servers and end-users contribute to this lower latency.

Lastly, at the Village level, latency reaches its lowest point of 20 ms. This minimal latency is achieved by deploying numerous edge servers directly within villages, where the majority of content requests originate. Through localized caching at this most granular level, the CDN enables near-instantaneous data delivery, providing users with the swiftest access to information.

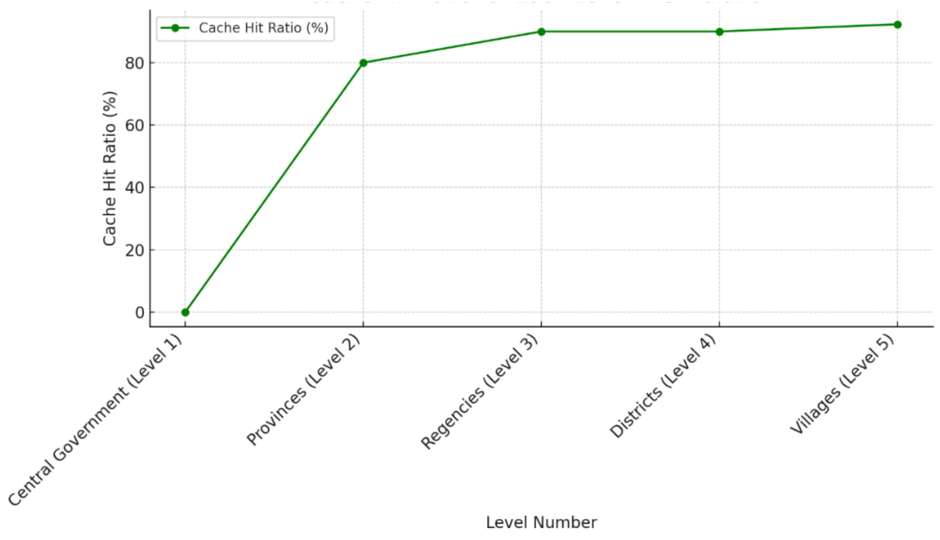
3. Cache Hit Ratio (CHR) Across Different Levels:

Table 3. Cache Hit Ratio (CHR)
Simulation Result Across Different Levels

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

Source: Research Data

Graph: Cache Hit Ratio for Each Level in Simulation
X-Axis: Level Number (Central Government to Villages)
Y-Axis: Cache Hit Ratio (%)



Source: Research Result
Figure 3. Cache Hit Ratio for Each Level in Simulation

Figure 3 illustrates the Cache Hit Ratio (CHR) across various levels within a hierarchical Content Delivery Network (CDN) model, spanning from the Central Government (Level 1) down to Village-level caches (Level 5). The CHR is a critical performance metric that quantifies the proportion of content requests served directly from a local cache, as opposed to being retrieved from the origin server or from caches at higher levels. A higher CHR signifies that a greater number of requests are fulfilled by local caches, thereby reducing latency, conserving bandwidth, and decreasing the load on upstream servers.

At the Central Government level, which functions as the origin server, there is no caching implemented; consequently, the CHR at this level is 0%. All content requests at this tier are either generated or fetched directly from the original source without the intervention of a caching mechanism.

Moving to the Provincial level, the CHR reaches 80%, indicating that 80% of content requests are served directly from the cache of the provincial edge servers, with only 20% requiring data retrieval from the origin server at the Central Government level. This relatively high CHR suggests efficient caching processes, which reduce the dependency on the origin server and enhance the speed of content delivery.

At the Regency level, the CHR increases to 90%, demonstrating that a larger proportion of content requests are effectively served from the regency-level cache. This leaves only 10% of requests that necessitate fetching data from the Provincial level, highlighting a further improvement in caching efficiency as content distribution moves closer to end-users.

Down to the District level maintains a similarly high CHR of 90%, aligning with the Regency level. This consistency implies that caching strategies at these intermediate levels are robust, enabling a substantial percentage of requests to be satisfied locally. As such, the District level acts as a pivotal caching layer, mitigating the need for frequent data retrieval from higher levels and facilitating rapid access to frequently requested content.

Finally, the Village level achieves the highest CHR at 92.3%, indicating that nearly all content requests are met by local village caches, with minimal reliance on higher-level caches or the origin server. This elevated CHR underscores the effectiveness of the caching strategy at the village level, ensuring prompt content delivery and reduced latency for end-users.

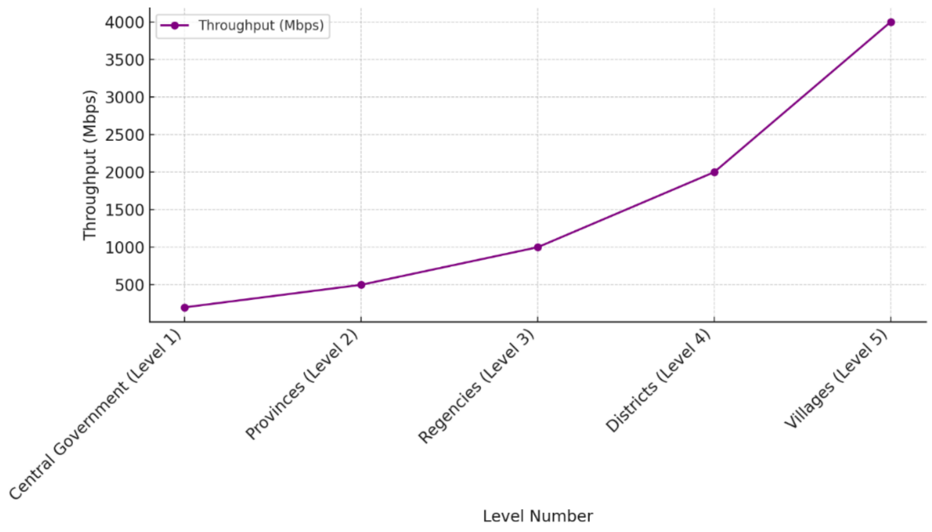
4. Throughput Across Different Levels:

Table 4. Throughput
Simulation Result Across Different Levels

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

Source: Research Data

Graph: Average Throughput for Each Level in Simulation
X-Axis: Level Number (Central Government to Villages)
Y-Axis: Average Throughput in Mbps



Source: Research Result

Figure 4. Average Throughput for Each Level in Simulation

The graph in Figure 4 illustrates the Average Throughput across varying tiers within a hierarchical Content Delivery Network (CDN) structure, spanning from the Central Government (Level 1) down to Village-level servers (Level 5). Throughput, measured in megabits per second (Mbps), quantifies the rate of successful data transmission within the network. A higher throughput value corresponds to more efficient data transfer, which is essential for rapid content delivery and an optimized user experience.

At the Central Government level, designated as the origin server, the throughput is recorded at 200 Mbps. This level is primarily responsible for content distribution to provincial edge servers, serving as the source for data within the CDN hierarchy. Due to its focus on upstream content distribution rather than direct end-user service, the throughput here is relatively moderate compared to downstream edge servers, which cater more closely to user demands.

The throughput at the Provincial level rises to 500 Mbps, reflecting the increased capacity of provincial edge servers to serve cached content efficiently within their regional boundaries. By locally caching frequently accessed content, these servers minimize repeated data requests to the Central Government level, thus enhancing data transmission rates.

At the Regency level, throughput doubles to 1,000 Mbps, representing the efficiency gains achieved by further decentralizing content closer to end-users. Regency-level servers leverage more localized caching, allowing them to support higher data transmission volumes and reduce latency, thereby improving the overall user experience.

Throughput continues to increase at the district level, reaching 2,000 Mbps. The CDN infrastructure at this stage benefits from an even finer distribution of edge servers across districts, which enables more substantial local caching capabilities. The resulting high throughput reflects the CDN's improved capacity to fulfill data requests directly from local caches, reducing the reliance on upstream data retrieval and providing users with quicker access.

Finally, at the Village level, throughput peaks at 4,000 Mbps. This substantial throughput rate is a result of the extensive deployment of edge servers at the local level, where the majority of content requests are satisfied. Village-level caching allows data to be delivered directly and at high speeds from local servers, significantly enhancing user experience while alleviating the overall network load.

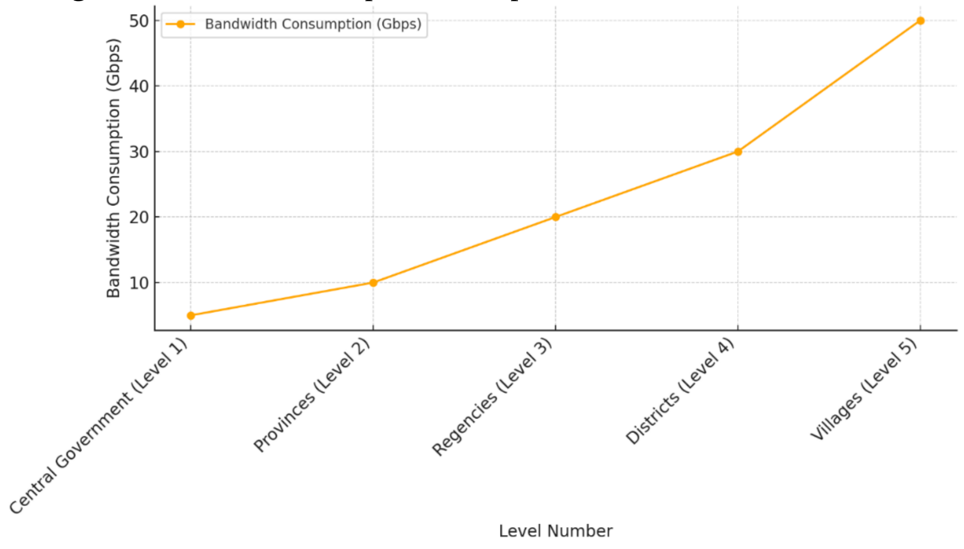
5. Bandwidth Consumption Across Different Levels:

Table 5. Bandwidth Consumption Simulation Result Across Different Levels

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

Source: Research Data

Graph: Average Bandwidth Consumption for Each Level in Simulation
X-Axis: Level Number (Central Government to Villages)
Y-Axis: Average Bandwidth Consumption in Gbps



Source: Research Result
Figure 5. Average Bandwidth Consumption for Each Level in Simulation

Figure 5 presents the Average Bandwidth Consumption across various hierarchical levels within a Content Delivery Network (CDN) model, spanning from the Central Government (Level 1) to Villages (Level 5). Bandwidth Consumption represents the volume of data transferred over a network within a designated timeframe, generally measured in gigabits per second (Gbps). This metric is essential for assessing a network's data-handling capacity and gauging the CDN’s efficiency in resource management.

At the Central Government level, bandwidth consumption is relatively modest, measured at 5 Gbps. Serving as the origin server, this level is responsible for data origination before distribution to subordinate levels. The limited bandwidth here reflects the minimal direct service to end-users, as the data is primarily channeled to provincial edge servers (Level 2) for additional caching and distribution.

The Provincial level shows a doubling in bandwidth consumption to 10 Gbps, driven by the edge servers managing a larger volume of data transfers. As the initial caching layer, this level is tasked with forwarding content to lower tiers, which accounts for the increased bandwidth. This level thus serves both regencies and the Central Government in data retrieval and distribution.

At the Regency level, bandwidth consumption rises notably to 20 Gbps, underscoring the substantial data transfers handled by the regency-level edge servers. These servers cache content to optimize delivery to district-level servers (Level 4) and, in certain cases, directly to end-users. This bandwidth increase highlights the role of regency servers in alleviating the burden on higher-tier servers through local caching.

The District level experiences a further increase in bandwidth consumption, reaching 30 Gbps. Edge servers at this level focus on localized caching and direct content delivery to end-users within each district. This heightened bandwidth signifies the intensive data processing required to effectively distribute local content, thereby minimizing latency for district users.

Bandwidth consumption peaks at the Village level, recorded at 50 Gbps. This level exhibits the highest bandwidth usage across all tiers, reflecting the extensive caching and direct data provisioning to end-users at the local level. Village-level edge servers manage the bulk of content requests, significantly reducing the need for higher-level cache or origin server access. This level’s high bandwidth consumption underscores the efficacy of localized caching and the CDN's capacity to deliver data swiftly to end-users across a widespread geographical area.

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.

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. 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.

REFERENCE

Allwörden, T. M. V. (2023). *Content and API Acceleration Using Content Delivery Networks* [PDF]. https://doi.org/10.2313/NET-2023-11-1_11

Bagga, J. (2023). Introduction to APIs. In J. Bagga, *Introduction to Integration Suite Capabilities* (pp. 1–8). Apress. https://doi.org/10.1007/978-1-4842-9630-1_1

Basu, S., Sundarrajan, A., Ghaderi, J., Shakkottai, S., & Sitaraman, R. (2018). Adaptive TTL-Based Caching for Content Delivery. *IEEE/ACM Transactions on Networking*, *26*(3), 1063–1077. https://doi.org/10.1109/TNET.2018.2818468

Bolla, R., Davoli, F., & Ricciardi, S. (1999). A hierarchical control structure for multimedia access networks. *1999 IEEE International Conference on Communications (Cat. No. 99CH36311)*, 1320–1325. https://doi.org/10.1109/ICC.1999.765556

Boukerche, A., & Gu, Y. (2011). Hierarchically distributed tree. *2011 IEEE Symposium on Computers and Communications (ISCC)*, 91–96. https://doi.org/10.1109/ISCC.2011.5984033

Buyya, R., Pathan, M., & Vakali, A. (Eds.). (2008). *Content Delivery Networks* (Vol. 9). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-77887-5

Carneiro, C., & Schmelmer, T. (2016). Optimizing Your APIs. In C. Carneiro & T. Schmelmer, *Microservices From Day One* (pp. 83–101). Apress. https://doi.org/10.1007/978-1-4842-1937-9_7

Chari, G., Sheffer, B., Branavan, S. R. K., & D'ippolito, N. (2023). Scaling Web API Integrations. *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 13–23. https://doi.org/10.1109/ICSE-SEIP58684.2023.00007

Chen, Q., Wang, W., Chen, W., Yu, F. R., & Zhang, Z. (2021). Cache-Enabled Multicast Content Pushing With Structured Deep Learning. *IEEE Journal on Selected Areas in Communications*, *39*(7), 2135–2149. https://doi.org/10.1109/JSAC.2021.3078493

Dale, R. (2019). Five Tips for a Successful API. *Natural Language Engineering*, *25*(06), 769–772. https://doi.org/10.1017/S1351324919000536

De, B. (2023). Introduction to APIs. In B. De, *API Management* (pp. 1–26). Apress. https://doi.org/10.1007/979-8-8688-0054-2_1

Elsayed, K. S., Geyer, F., & Rizk, A. (2024). *Utility-driven Optimization of TTL Cache Hierarchies under Network Delays* (Version 1). arXiv. https://doi.org/10.48550/ARXIV.2405.04402

Goseling, J., & Simeone, O. (2019). Soft-TTL: Time-Varying Fractional Caching. *IEEE Networking Letters*, *1*(1), 18–21. https://doi.org/10.1109/LNET.2018.2883245

Kamiyama, N., Nakano, Y., & Shiomoto, K. (2016). Cache Replacement Based on Distance to Origin Servers. *IEEE Transactions on Network and Service Management*, *13*(4), 848–859. https://doi.org/10.1109/TNSM.2016.2600240

Korzun, D., & Gurtov, A. (2014). Hierarchical architectures in structured peer-to-peer overlay networks. *Peer-to-Peer Networking and Applications*, *7*(4), 359–395. https://doi.org/10.1007/s12083-013-0200-z

Li, Y., & Wang, X. (2020). Hierarchical Information-Centric Networking Framework. *International Journal of Wireless Information Networks*, *27*(1), 184–196. https://doi.org/10.1007/s10776-019-00477-0

Liu, H., & Han, R. (2021). A Hierarchical Cache Size Allocation Scheme Based on Content Dissemination in Information-Centric Networks. *Future Internet*, *13*(5), 131. https://doi.org/10.3390/fi13050131

Liu, J., Wan, X., Zhu, Q., Peng, T., & Hu, X. (2022). Research on Adaptive Cache Mechanism Based on TTL. *2022 2nd International Conference on Networking, Communications and Information Technology (NetCIT)*, 507–511. https://doi.org/10.1109/NetCIT57419.2022.00125

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

Stocker, V., Smaragdakis, G., Lehr, W., & Bauer, S. (2017). The growing complexity of content delivery networks: Challenges and implications for the Internet ecosystem. *Telecommunications Policy*, *41*(10), 1003–1016. https://doi.org/10.1016/j.telpol.2017.02.004