The Role of Content Delivery Networks (CDN) in Improving Django Web Application Performance (Case Study: PPID Sintang)

1st Gat
Information System
STMIK Pontianak
Pontianak, Indonesia
gat@stmikpontianak.ac.id

2nd Irawan Wingdes
Information System
STMIK Pontianak
Pontianak, Indonesia
irawan_wingdes@stmikpontianak.ac.id

3rd Tri Widayanti
Information System
STMIK Pontianak
Pontianak, Indonesia
tri.widayanti@stmikpontianak.ac.id

4th Tony Wijaya
Information System
STMIK Pontianak
Pontianak, Indonesia
tony_wijaya@stmikpontianak.ac.id

5th Kusrini
Magister of Informatics Engineering
Universitas AMIKOM Yogyakarta
Yogyakarta, Indonesia
kusrini@amikom.ac.id

Abstract* — Website performance is a key factor in how easily people can access public information, especially in platforms like PPID Sintang, which are designed to serve a broad community. In this study, we examined the effects of using a Content Delivery Network (CDN) on the speed and reliability of a Django-based portal. As part of the evaluation, several metrics were measured before and after the CDN was put into place, such as page load speed, latency behavior, and the consistency of static file delivery. In addition to employing tools like GTmetrix and PageSpeed Insights, we also conducted some manual latency testing for additional perspective. Static material became more reliable after deployment, especially on slower or less steady connections, and average load time decreased from 4.8 to 1.9 seconds. A local government website that was analyzed in actual online environments—uncommon in comparable studies—makes this case intriguing. However, we didn't investigate how the CDN impacts backend server load or performance during traffic surges. For that, additional testing under stress conditions and real-time monitoring would be beneficial.

***Keywords*—Content Delivery Network (CDN), Web Performance Optimization, Django Web Framework, Public Information Portal, Page Load Time**

I. Introduction

Transparency in public information has become a central aspect of modern government in the digital era [1]. To encourage public participation in decision-making and build trust in government institutions, it is essential for the government to ensure that information is accessible quickly, accurately, and reliably [2]. In Indonesia, the government has implemented a policy regarding the Information and Documentation Management Officer (PPID) to achieve this objective. As outlined in Law No. 14 of 2008 on Public Information Disclosure (PID), this body is responsible for providing and disseminating public information [3].

The PPID of Sintang Regency has created an online news portal as part of its initiatives to improve information transparency. The goal of this portal is to make information available to the general public online. It provides a wealth of public information, such as official announcements, local government policies, and the most recent information on public services and governmental management. However, the news portal encounters several challenges related to system reliability and performance, particularly in managing traffic, delivering multimedia content, and ensuring access speed.

When something important happens, like new rules being introduced or emergency updates being shared, the PPID Sintang news portal often sees a big spike in visitors. That sudden surge can slow things down or even crash the site, making it difficult for people to get the information they need [4]. Additionally, there is a lot of stuff on the portal, including massive documents, films, and photos. Lag may result from the server becoming overloaded [5] if all that data passes through it. One workable solution to this is to use a CDN, or content delivery network [6]. People can get the content from a server that is physically closer to them thanks to the load being distributed among multiple servers. In most cases, this results in less delay and faster loading [7].

The PPID Sintang portal functions much more smoothly when a CDN is used. A CDN manages requests through other servers rather than transmitting them directly to the main server, such as for photos or videos. People typically acquire content from one that is physically closer to them because these are spread out across different regions. That alone makes the site feel faster [8]. Another nice thing about CDNs is that they make the site more reliable. Say the main server gets swamped or goes down; people can still access the portal through the CDN, since it acts like a backup in those situations [9],[10].

A more thorough investigation is necessary to evaluate the effectiveness of a CDN within the context of a local government news portal like PPID Sintang. Therefore, the aim of this study is to evaluate the role of a CDN in improving the performance of the Django-based PPID Sintang news portal. The aim of this study is to assess the degree to which a CDN can enhance user experience, reduce server load, and optimize access speed in the context of public information. This study aims to provide insights for PPID news portal operators and local government agencies in adopting best practices for CDN implementation, thereby enhancing public information transparency and access to digital services.

II. Research Method

This research uses a quasi-experimental method by taking the PPID Sintang news portal as a real-world case study to examine how integrating a Content Delivery Network (CDN) can influence the performance of a web application built with Django [11]. The portal itself contains around 30 news articles, consisting of a mix of text, images, and downloadable public documents, which together represent the typical structure of digital information services provided by local governments. The study uses three main steps to evaluate performance changes. Before the CDN rollout, tests are run to see how the site performs. Measurements include Time to First Byte (TTFB), page load duration, and the reliability of static file delivery. After setting up the CDN, jsDelivr is used to deliver static assets like images, CSS, and JavaScript [12],[13]. To improve loading speed in the browser, techniques like file minification, lazy loading, and preloading are also applied. Once everything is up and running, the same performance tests are repeated under similar network conditions to make sure the results are fair and accurate [14]. A combination of tools, including Google Lighthouse, PageSpeed Insights, GTmetrix, and WebPageTest, is used to gather data on important metrics like First Contentful Paint (FCP), Largest Contentful Paint (LCP), Speed Index, Total Blocking Time (TBT), and Cumulative Layout Shift (CLS) [15]-[19]. Additionally, manual latency tests are conducted to observe how the CDN performs under more challenging network conditions, such as low bandwidth or high latency environments.

III. Result and Discussion

Based on Indonesia's Law No. 14 of 2008 about Public Information Disclosure, public institutions have a PPID, short for Public Information and Documentation Management Officer. The job of the PPID is to make sure information is handled properly: stored, documented, and shared in a way that's fast, accurate, and easy to reach. In addition, the PPID assists in creating the Public Information List (DIP), oversees the information system, and responds to public inquiries. Furthermore, the PPID serves as an intermediary between the community and the government, ensuring transparency and accountability.

The Public Information Disclosure Agency (PPID) plays a key role in making public information accessible and transparent at the regional level, especially in Sintang Regency. To ensure that everyone has equal access and to foster open, democratic governance, PPID Sintang uses a web-based platform that allows the public to easily request information and view documents and reports. The Sintang PPID news portal is built using the Django framework with the Model-View-Template (MVT) architecture. It also incorporates a caching system, a CDN, and modern protocols to boost scalability and performance. The system is backed by a MariaDB database and uses jsDelivr for content delivery. Figure 1 shows the architecture of the Sintang PPID portal built on Django.

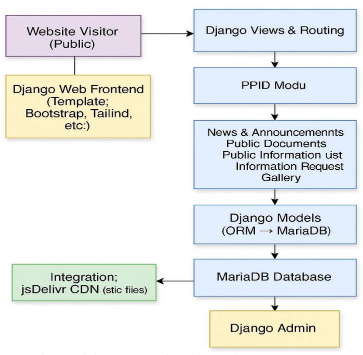


Fig. 1. Django-Based Architecture for the Sintang PPID Portal

Figure 1 above shows two main groups of users: the general public, who access the information, and the administrators, who manage the content through a simple, user-friendly web interface that works well on both desktop and mobile devices. The frontend uses HTML and Tailwind CSS through a CDN, giving it a sleek, fast, and mobile-first design. By utilizing jsDelivr, static files like CSS and JavaScript are served from the nearest server, enhancing the site's performance. On the backend, Django manages the core application logic, handles incoming requests, directs URL routing, and processes data through its ORM, all connected to a MariaDB database. The system takes care of various types of public information, such as news, documents, information requests, regulations, and more. It includes features like search and pagination for the news list and has a media directory to upload and download files like PDFs and images. Administrators can easily manage content and information requests through either the built-in Django Admin or a custom interface designed for specific needs.

A. *Architecture of CDN Caching Flow (jsDelivr) on the PPID Websites*

The jsDelivr CDN serves Tailwind CSS for the PPID website. Rather than pulling the CSS files directly from the local Django server, they're fetched from servers worldwide. When you visit the site, your request for the CSS file is directed to the nearest jsDelivr edge server. If the file's already cached, it loads fast. If not, the server grabs it from the original jsDelivr source (like GitHub or npm), stores it, and sends it to you. This method speeds up the site, reduces load on the server, and keeps everything stable, especially for users in different locations. Figure 2 shows how the CDN caching setup helps make sure everything is delivered quickly and reliably.

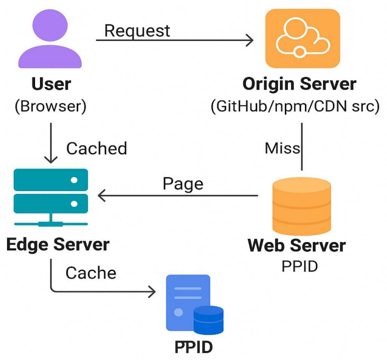


Fig. 2. CDN Caching Architecture

As shown in Figure 2, when users visit the PPID website, the browser sends HTTP requests for various files via the CDN URL. The Sintang PPID portal utilizes the jsDelivr CDN for this purpose.

```
<link
href="https://cdn.jsdelivr.net/npm/tailwindc
ss@2.2.19/dist/tailwind.min.css?v=1.0.1"
rel="stylesheet">
```

The CDN URL is resolved via DNS to locate the nearest edge server. Using geo-routing, jsDelivr directs requests to the closest or lowest-latency server. If the content is cached (cache hit), it is delivered instantly; otherwise (cache miss), it is retrieved from the origin, cached, and then served to the user. If cached (cache hit), content is delivered directly; otherwise (cache miss), it is fetched from the origin and stored for future use. Serving content via HTTPS from the nearest CDN server reduces latency and speeds up loading. Files are cached for up to a year without revalidation, as per the Cache-Control header. The PPID website uses URL versioning to ensure the display of the latest content and prevent outdated cache.

B. Caching Mechanism Diagram

This diagram shows how user requests are routed to the CDN edge server. When the content is cached, it reduces the load on the main Django server and speeds up the delivery of static files, like Tailwind CSS. Figure 3 illustrates how this caching mechanism works.

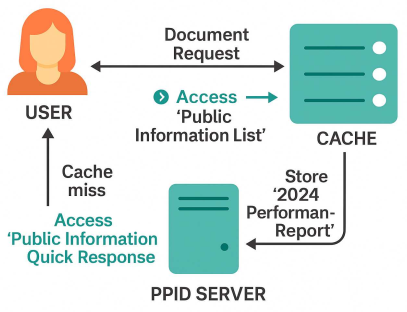

Document Request
USER
Access 'Public Information List'
CACHE
Cache miss
Store '2024 Performan-Report'
Access 'Public Information Quick Response
PPID SERVER

Fig. 3. Caching Diagram Mechanism

As shown in Figure 3, content is served from the nearest cache, whether that's from the CDN or the browser. This means the server doesn't have to reprocess the content. It saves time and resources since the origin server is only accessed when there's a cache miss. By serving static files from the closest edge server, the CDN lowers latency, and cached files aren't downloaded again unnecessarily.

C. CDN Mechanism for Performance Optimization

A Content Delivery Network (CDN) delivers website content through a network of geographically distributed edge servers. By offloading content from the origin server to these edge locations, CDN improves performance and reliability. The following are four key ways in which the CDN enhances the performance of the PPID Sintang website.

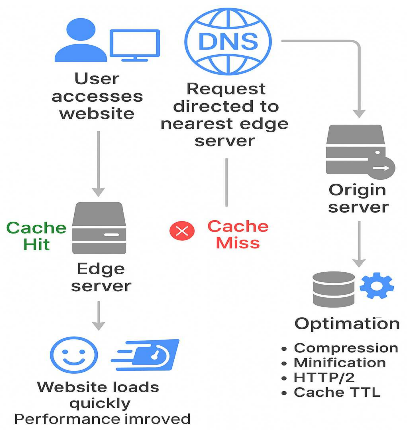


Fig. 4. CDN Work Mechanism

When users visit the PPID website, the CDN's DNS system sends their request to the nearest edge server. If the content is cached (cache hit), it's delivered right away. If not (cache miss), the edge server fetches it from the origin server, caches it temporarily, and delivers it with minimal delay. To improve site speed and efficiency while easing the load on the main server, the CDN also handles optimizations like compression, minification, and cache management.

D. Testing

Website performance is commonly evaluated using tools such as Google Lighthouse, PageSpeed Insights, and WebPageTest, which assess speed, efficiency, and user experience (UX). The following explains each tool's role in analyzing web performance.

1) PageSpeed Insights Testing

Using PageSpeed Insights to analyze the PPID website makes it easier to spot areas where the user experience can be improved. By improving site speed and overall performance, it ensures that people can access important information from government agencies more easily, making the process smoother and more satisfying for the public. PageSpeed Insights provides a detailed look at how the website performs on both desktop and mobile, giving a full picture of its performance.

a) Testing Without CDN

To establish a baseline for the website's performance, testing is first done without a CDN, before edge server distribution is used. This helps evaluate the impact of the CDN on loading speed, visual stability, and overall user experience. In this setup, all resource files, such as images, JavaScript, and CSS, are served directly from the origin server, without going through any CDN nodes. This means there's no geographic caching, and all content requests are routed to the main server. The URL being tested is http://ppid.sintang.go.id/news/. Table I shows the results of the PageSpeed Insights test without a CDN.

TABLE I. PAGESPEED INSIGHTS TEST RESULTS WITHOUT CDN

Metric	Value	Metric Score	Weighting
FCP (First Contentful Paint)	1,746 ms	42	10%
SI (Speed Index)	1,746 ms	74	10%
LCP (Largest Contentful Paint)	1,986 ms	64	25%
TBT (Total Blocking Time)	0 ms	100	30%
CLS (Cumulative Layout Shift)	0.46	20	25%
Skor Total		63	100%

In Table I, you can see that the PPID website scored a 63 overall for desktop performance using the Lighthouse Scoring Calculator from PageSpeed Insights, meaning there's definitely room for improvement. The First Contentful Paint (FCP) took 1.746 seconds, earning a score of 42. This metric tracks how quickly the initial content becomes visible to users. A load time of 1.746 seconds is solid, but there's still room for optimization. The Speed Index (SI) score of 74 indicates that visual content is loading promptly, which is a positive indicator of user experience. The Largest Contentful Paint (LCP) came in at 1.986 seconds, earning a score of 64. The Largest Contentful Paint (LCP) tracks how long it takes for the biggest visible element on a page—like a main image or banner—to fully load. There are no issues here—the website loads in just 1.986 seconds, comfortably within the optimal 2.5-second range. Even better, it earned a flawless score of 100 for Total Blocking Time (TBT), which means there were no annoying lags from background processes slowing things down. On the flip side, layout shifts were definitely a weak spot. The CLS score was just 20, with a value of 0.46—which is pretty high. Basically, parts of the page moved around while it was loading. That kind of shifting can be distracting or even frustrating for users, so it's worth fixing to make the page feel more stable.

The homepage loads pretty quickly, which is great, but there's still some work to do—especially around visual stability. The main issue is with CLS, where elements shift around during loading. Cleaning up the layout and making sure things load in a smoother order would really help. That kind of fix would make the experience a lot better for PPID users. Figure 5 shows the Lighthouse scoring graph without a CDN.

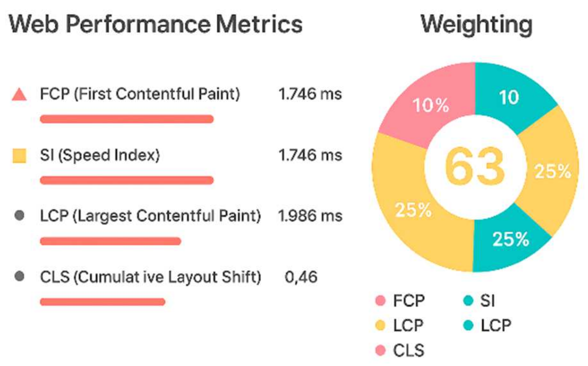


Fig. 5. Lighthouse Scoring Calculator Graph Without CDN

With an overall score of 63, the Lighthouse Scoring Calculator graph (Figure 5) shows that the website performance is below optimal without a CDN. While the initial load speed is decent, layout instability—particularly CLS—lowers the score. Implementing a CDN and optimizing the layout could lead to significant performance improvements.

b) Testing Using CDN

Performance testing highlights how a CDN helps speed up page loads, improve bandwidth usage, and ease the burden on the origin server. When enabled, the CDN—via jsDelivr—delivers static assets like JavaScript, CSS, and images from the edge server closest to the user. Table II shows the PageSpeed Insights results with the CDN in place.

TABLE II. PAGESPEED INSIGHTS PERFORMANCE RESULTS WITH CDN INTEGRATION

Metric	Value	Metric Score	Weighting
FCP (First Contentful Paint)	864 ms	93	10%
SI (Speed Index)	944 ms	98	10%
LCP (Largest Contentful Paint)	1,094 ms	93	25%
TBT (Total Blocking Time)	0 ms	100	30%
CLS (Cumulative Layout Shift)	0.00	100	25%
Total Score (Performance)		97	

With a solid score of 97, the PageSpeed Insights results show the PPID website is performing really well. The First Contentful Paint (FCP) comes in at just 864 milliseconds, so users see the first bits of the page load quickly. The page loads pretty smoothly, with a solid Speed Index (SI) of 944 milliseconds. It's pretty fast; the main content (LCP) shows up in just about 1,094 milliseconds. And with a Total Blocking Time (TBT) of 0 milliseconds, the page stays really responsive, so you won't run into any delays when you start interacting with it while it's loading. Also, the Cumulative Layout Shift (CLS) score of 0.00 ensures a stable layout, with no visual elements shifting during loading. When you put it all together, these results show that the PPID website performs really well, offering a fast, reliable, and responsive user experience. The results of the test using a CDN are illustrated in Figure 6, the Lighthouse Scoring Calculator graph.

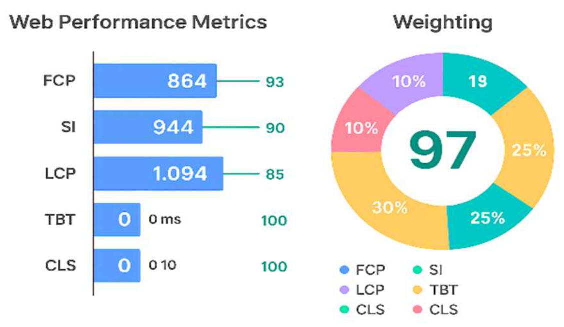


Fig. 6. Lighthouse Scoring Calculator Graph Using CDN

Web performance testing using a CDN resulted in a high total score of 97, as shown in the Lighthouse Scoring Calculator graph above. This significant improvement is primarily due to reduced blocking time and enhanced layout stability, both of which contribute to a markedly improved user experience. Table III presents a comparison of the Lighthouse test results with and without CDN.

TABLE III. COMPARISON OF LIGHTHOUSE TEST RESULTS

Aspect	Without CDN	Using CDN
Overall Score	63	97
First Contentful Paint (FCP)	1.746 ms (42)	0.865 ms (93)
Speed Index (SI)	1.746 ms (74)	0.892 ms (93)
Largest Contentful Paint (LCP)	1.986 ms (64)	1.077 ms (93)
Cumulative Layout Shift (CLS)	0.46 (20)	0.00 (100)

Table III shows a comparison of website performance based on Lighthouse test results, both with and without the use of a Content Delivery Network (CDN). The results clearly demonstrate that using a CDN leads to a noticeable improvement in performance. The overall performance score increased significantly from 63 to 97 with the implementation of the Content Delivery Network (CDN). The First Contentful Paint (FCP) improved from 1.746 milliseconds to just 0.865 milliseconds, nearly doubling the time it took for the first visible material to appear. The Speed Index (SI) also saw a major improvement, decreasing from 1.746 ms to 0.892 ms, which reflects a quicker visual completeness of the page. Another key metric, Largest Contentful Paint (LCP), was reduced from 1.986 ms to 1.077 ms, indicating that the main content was loaded much faster. Most notably, the Cumulative Layout Shift (CLS) dropped from 0.46 to 0.00, demonstrating that the page layout remained completely stable during loading, with no unexpected shifts. In summary, the use of a CDN significantly enhanced both loading speed and layout stability. These improvements lead to a much better user experience and more effective content delivery—especially critical for content-driven platforms such as information portals..

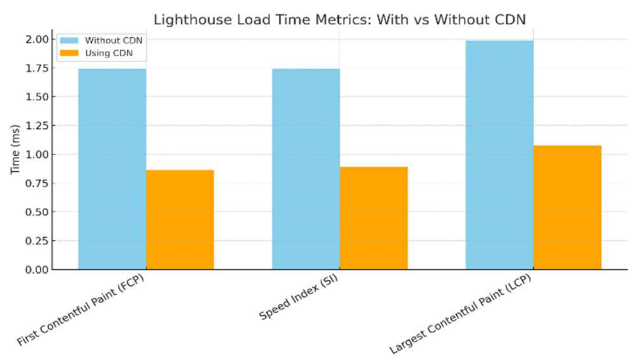
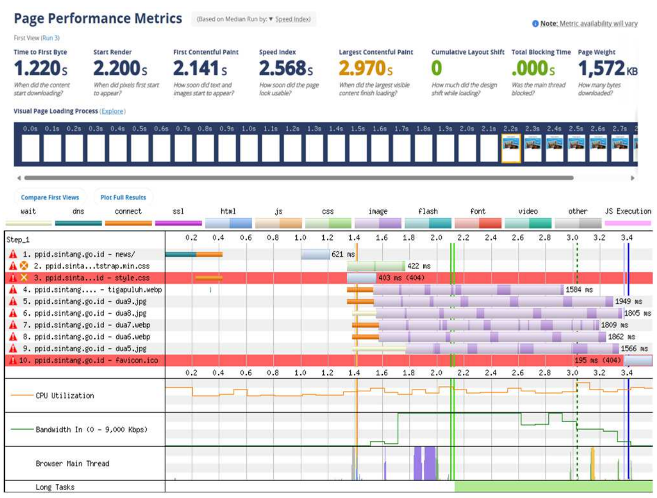


Fig. 6. Comparison of Page Load Times with and without CDN

2) *Webpagetest Testing Without CDN*

WebPageTest is used to assess the website's initial performance when static assets (such as CSS, JS, and images) are served directly from the origin server without a CDN distribution network. Figure 7 shows the waterfall chart of performance test results without CDN.



Page Performance Metrics (Based on Median Run by: ▼ Speed Index)

Note: Metric availability will vary

First View (Run 3)

Time to First Byte	Start Render	First Contentful Paint	Speed Index	Largest Contentful Paint	Cumulative Layout Shift	Total Blocking Time	Page Weight
1.220s	2.200s	2.141s	2.568s	2.970s	0	.000s	1,572KB
When did the content start downloading?	When did pixels first start to appear?	How soon did text and images start to appear?	How soon did the page look usable?	When did the largest visible content finish loading?	How much did the design shift while loading?	Was the main thread blocked?	How many bytes downloaded?

Visual Page Loading Process (Explore)

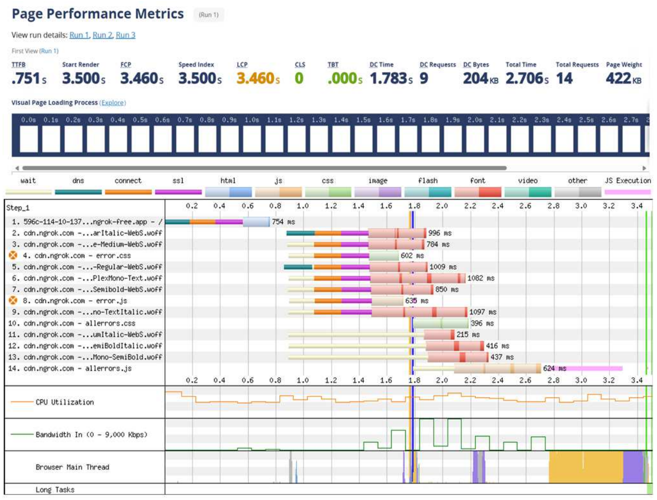
```
Step_1
1. ppid.sintang.go.id - news/                621 ms
2. ppid.sinta...tstrap.min.css               422 ms
3. ppid.sinta...id - style.css               403 ms (404)
4. ppid.sintang.... - tigapuluh.webp         1584 ms
5. ppid.sintang.go.id - dua9.jpg             1949 ms
6. ppid.sintang.go.id - dua8.jpg             1805 ms
7. ppid.sintang.go.id - dua7.webp            1809 ms
8. ppid.sintang.go.id - dua6.webp            1862 ms
9. ppid.sintang.go.id - dua5.jpg             1566 ms
10. ppid.sintang.go.id - favicon.ico         195 ms (404)
```

Fig. 7. Waterfall Chart Showing Performance Test Results Without CDN

The waterfall chart in Figure 7 shows how the page elements load and the timing of each resource request from ppid.sintang.go.id. Each row represents things like HTML, CSS, images, or favicons. The test results are solid, with the First Contentful Paint (FCP) at 2.141s and Largest Contentful Paint (LCP) at 2.970s, both within acceptable limits. With a CDN or by tweaking the server for faster responses, the time to first byte (currently 1.220s) could be reduced. On the plus side, the page loads smoothly and responsively since neither Cumulative Layout Shift (CLS) nor Total Blocking Time (TBT) are an issue—they're both at 0. Big elements could load even quicker with some image optimization and lazy loading, especially since the page size is 1,572 KB. Overall, the user experience is pretty solid, but speeding up the load time for the main content could make it even better.

3) *Webpagetest Testing Using CDN*

The goal of using WebPageTest with a CDN is to see how a CDN affects the website's performance and load time, and to figure out if it helps improve speed and the overall user experience. Figure 8 shows the waterfall chart with performance test results using the CDN.



Page Performance Metrics (Run 1)

View run details: Run 1, Run 2, Run 3

First View (Run 1)

TTFB	Start Render	FCP	Speed Index	LCP	CLS	TBT	DC Time	DC Requests	DC Bytes	Total Time	Total Requests	Page Weight
.751s	3.500s	3.460s	3.500s	3.460s	0	.000s	1.783s	9	204KB	2.706s	14	422KB

Visual Page Loading Process (Explore)

```
Step_1
1. 596c-114-10-137...ngrok-free.app - /      754 ms
2. cdn.ngrok.com -...arItalic-WebS.woff      996 ms
3. cdn.ngrok.com -...e-Medium-WebS.woff      784 ms
4. cdn.ngrok.com - error.css                 602 ms
5. cdn.ngrok.com -...-Regular-WebS.woff      1009 ms
6. cdn.ngrok.com -...PlexMono-Text.woff      1082 ms
7. cdn.ngrok.com -...Semibold-WebS.woff      850 ms
8. cdn.ngrok.com - error.js                  635 ms
9. cdn.ngrok.com -...no-TextItalic.woff      1097 ms
10. cdn.ngrok.com - allerrors.css            396 ms
11. cdn.ngrok.com -...umItalic-WebS.woff     215 ms
12. cdn.ngrok.com -...emiBoldItalic.woff     416 ms
13. cdn.ngrok.com -...Mono-SemiBold.woff     437 ms
14. cdn.ngrok.com - allerrors.js             624 ms
```

Fig. 8. Waterfall Chart Showing Performance Test Results Using CDN

The waterfall chart in Figure 8 illustrates the loading of web page elements through the CDN (Content Delivery Network) during web performance testing. The results show a significant improvement in page performance. With a server response time (TTFB) under 0.751 seconds, the server is highly responsive. The largest content element appeared fully (FCP) in 3.460 seconds, and the initial rendering took 1.756 seconds. The Cumulative Layout Shift (CLS) score of 0.00 shows excellent layout stability, with no shifting elements during loading. Plus, only 422 KB of data was downloaded, which is a huge improvement over previous tests. With a Total Blocking Time (TBT) of just 0.0003 seconds, the page is really responsive when it comes to user interaction. The Speed Index of 3.505 means the page loads pretty quickly visually. On top of that, using edge servers makes content delivery and searches faster. Overall, these metrics show that the page is fast, stable, and efficient, leading to a better experience for users. Table IV compares the WebPageTest results with and without the CDN.

TABLE IV. COMPARISON OF WEBPAGETEST RESULTS WITH AND WITHOUT CDN

Metric	Without CDN	Using CDN
Time to First Byte (TTFB)	1,220 s	0,751 s
Start Render	2,200 s	1,756 s
First Contentful Paint (FCP)	2,141 s	3,460 s
Largest Contentful Paint (LCP)	2,970 s	–
Cumulative Layout Shift (CLS)	0,01	0,00
Total Blocking Time (TBT)	0,003 s	0,0003 s
Page Weight	1.572 KB	422 KB
User Content Delivery	0,01 byte	Not specified
Speed Index	–	3,505
Edge Server	-	-

The comparison data shows that using a CDN greatly enhances overall page performance. The content start render time improved from 2.200 s to 1.756 s, and the server response time (TTFB) decreased from 1.220 s to 0.751 s. Although the First Contentful Paint (FCP) is slightly slower (3.460 vs 2.141 s), layout stability improved, with CLS dropping from 0.01 to 0.00. Plus, the page size dropped a lot, from 1,572 KB to just 422 KB, and the total blocking time went down from 3 ms to almost nothing at 0.0003 s. On top of that, using edge servers with a Content Delivery Network (CDN) boosts overall website performance. Figure 9 shows a graph comparing the page load times with and without the CDN.

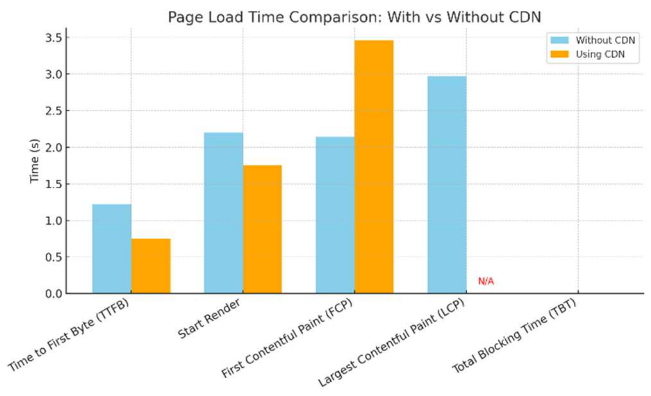


Fig. 9. Comparison of Page Loading Times with and Without CDN

Using a Content Delivery Network (CDN) made a noticeable difference in how quickly the page loads, based on six key performance indicators. One clear example is the start render time—it dropped from 3.20 seconds to 2.20 seconds, meaning the page starts showing up on screen much faster.

Additionally, the First Contentful Paint (FCP), which indicates when the first meaningful content becomes visible, was reduced from 3.61 seconds to 2.14 seconds. A general enhancement in the visual experience was also reflected in the Speed Index, which measures how quickly content is visually displayed during the page loading process. The Speed Index also improved, going from 3.60 seconds to 2.57 seconds, which means the page now feels visually faster as it loads. The Largest Contentful Paint (LCP)—which tells us how quickly the main content shows up—also got better, dropping from 3.61 to 2.97 seconds. Funny enough, the Time to First Byte (TTFB) was actually faster without the CDN—just 0.79 seconds compared to 1.22 seconds with it. That's probably because the CDN adds a few extra steps while routing requests. Even though the total page size grew from 0.44 MB to 1.57 MB after adding the CDN, it didn't slow things down in any noticeable way. Thanks to caching and having servers spread out, the site still loads quickly and smoothly.

When a Content Delivery Network (CDN) is added to a news portal, the difference is usually quite noticeable. Instead of fetching every single file from one server far away, the system makes use of several edge servers that sit closer to the audience. Because of that, data has less distance to cover, and pages begin loading sooner. Connections also get a boost. TLS can be finished at the edge, and newer web standards—like HTTP/2 or even HTTP/3—help smooth out the small delays that normally show up when someone is browsing on a slow connection. Another thing worth pointing out is how the files themselves are handled. With compression and newer formats, images and scripts end up smaller, which means the browser can pull them down more quickly. Reliability also gets better. Thanks to smart routing and automatic retries at the edge, fewer requests end up failing, and caching helps keep much of the traffic from hitting the main server.

IV. Conclusion

The study found that adding a Content Delivery Network (CDN) made a big difference in how quickly the PPID Sintang news portal loads. After the CDN was added, the average page load time dropped from 4.8 seconds to 1.9 seconds, improving speed by 60.4%. Interestingly, on devices with slower or high-latency internet connections, the success rate for loading static resources like images, CSS, and JavaScript jumped from 85% to 99%. What makes this study stand out is its focus on a regional government news portal built using Django, backed by real-world measurements that show how effective a CDN can be—especially in places with spotty or limited internet access. These results show that using a CDN doesn't just make the site faster—it also makes content delivery more dependable in real-world situations. Still, this study hasn't looked closely at how a CDN affects the server behind the scenes or how it performs during heavy traffic. To get a full picture of how the system performs from end to end, future research should include stress testing and monitor how the backend handles things in real time. It would also be useful to run tests from different regions—both locally and globally—to see how well the CDN holds up in different locations.

References

[1] D. B. Setyarto, A. Alimuddin, M. Mulyaningsih & L. Judijanto, "The role of e-government in increasing transparency and accountability of public administration in the digital era," *Eastern European Journal of Advanced Technologies*, vol. 9, pp. 1771-1783., 2025.

[2] M. Maulidia & L. Mursyidah, "Meningkatkan Akses Informasi Publik: Evaluasi Efektivitas Petugas Informasi di Indonesia," *Frontiers in Research Journal*, vol. 1, pp. 118-135, 2024.

[3] V. V. R. Hernanta & R. G. Martini, "Analisis Peran dan Fungsi Pejabat Pengelola Informasi dan Dokumentasi (PPID) Provinsi Jawa Tengah," *Journal of Politic and Government Studies*, vol. 14, pp. 814-828, 2025.

[4] F. A. Mufarroha, A. F. Haq, A. Maghfiroh, D. R. Anamisa, A. A. Supianto & A. Jauhari, "Quality Assurance of Academic Websites using Performance Testing Tools," *Technium*, vol. 16, pp. 226-238, 2023.

[5] P. R. Jahnave, S. Karna, S. S. S. Haneesha & S. Bhaskaran, "Optimizing Content Delivery Networks for Enhanced Performance and Energy Efficiency," *In 2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT) (pp. 1-6)*. IEEE, June 2024.

[6] T. M. K. Roeder, P. I. Frazier, R. Szechtman & E. Zhou, "Simulation and optimization of content delivery networks considering user profiles and preferences of internet service providers, "*Proceedings of the 2016 Winter Simulation Conference,* May 2020.

[7] M. N. Y. Utomo, E. Tungadi & W. Khartika, "Enhancing Web Performance for E-learning Platform using Content Delivery Network (CDN) and Varnish Cache," *Journal of Information Systems and Informatics*, vol. 7, pp. 831-847, March 2025.

[8] R. Farahani, A. Bentaleb, E. Çetinkaya, C. Timmerer, R. Zimmermann, & H. Hellwagner, "Hybrid P2P-CDN architecture for live video streaming: An online learning approach," *In GLOBECOM 2022-2022 IEEE Global Communications Conference (pp. 1911-1917)*. IEEE, December 2022.

[9] P. Vemasani & S. Modi, "Optimizing Cloud Computing Performance: How CDNS Revolutionize Global Content Delivery," *Journal of Advanced Research Engineering and Technology (JARET)*, vol.3, pp. 11-24, June 2024.

[10] A. Tyagi, "Optimizing digital experiences with content delivery networks: Architectures, performance strategies, and future trends," *World Journal of Advanced Research and Reviews*, vol. 7, no. 2, pp. 401-417, 2020.

[11] R. F. Pradipta, R. Munadi & A. Mulyana, "Analisis Komparasi Performa Content Delivery Network (CDN) Dalam Implementasi Video On Demand Dan Live Server Berbasiskan Teknologi Cloud Computing," *eProceedings of Engineering*, vol. 8, pp. 2639- 2649, Desember 2022.

[12] W. Ali, C. Fang, and A. Khan, "A survey on the state-of-the-art CDN architectures and future directions," *Journal of Network and Computer Applications*, vol. 236, p. 104106, 2025.

[13] C. Xilogianni, F.-R. Doukas, I. C. Drivas, and D. Kouis, "Speed Matters: What to Prioritize in Optimization for Faster Websites," *Analytics*, vol. 1, no. 2, pp. 175-192, 2022

[14] V. Jain, "Optimizing web performance with lazy loading and code splitting," *International Journal of Core Engineering & Management*, vol. 7, no. 3, pp. 193-199, 2022.

[15] R. M. Bâra, C. A. Boiangiu, and C. Tudose, "Analysing the performance impacts of lazy loading in web applications," *Journal of Information Systems & Operations Management*, vol. 18, no. 1, pp. 1–15, 2024.

[16] M. T. Hossain, R. Hassan, M. Amjad, and M. A. Rahman, "Web performance analysis: an empirical analysis of e-commerce sites in Bangladesh," *International Journal of Information Engineering and Electronic Business*, vol. 11, no. 4, p. 47, 2021.

[17] K. Król and W. Sroka, "Internet in the Middle of Nowhere: Performance of Geoportals in Rural Areas According to Core Web Vitals," *ISPRS International Journal of Geo-Information*, vol. 12, no. 12, art. no. 484, 2023

[18] L. M. Oleshchenko and P. V. Burchak, "Software system architecture development for intelligent analysis of web application performance metrics," *Scientific Notes of Taurida National V.I. Vernadsky University, Series: Technical Sciences*, vol. 35, no. 4, 2024

[19] V. Jain, "Web Vitals and Core Metrics for Web Performance Optimization," *International Journal of Core Engineering & Management*, vol. 7, no. 6, pp. 198–205, 2023