Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2019, 6(8):100-104

Research Article ISSN: 2394 - 658X

Performance Optimization in Web Applications

Prathyusha Kosuru

Project Delivery Specialist

ABSTRACT

Application performance optimization is an essential process that helps improve the usability of web applications. This document aims to discuss various ways of increasing both the front-end and back-end performance levels. It includes basic front-end optimizations such as lazy loading and caching and back-end optimizations including database indexing and query optimization. By using these methods, developers can increase the effectiveness of work and efficiency of web applications (Ahmed et al., 2016)

Keywords: Caching, Lazy Loading, Database Indexing, Content Delivery Networks, Web Application Responsiveness.

INTRODUCTION

Performance optimization is a set of approaches to improving application performance, and application speed when used on the World Wide Web. This implies the reduction of time taken to load a given page or to respond to user requests and minimizing use of resources. Optimization is not a single facet issue, but it entails consideration of both front-end and back-end factors for the client and server systems, respectively. With the help of various optimization techniques, web applications can be designed and developed that are not only faster but also more reliable and scalable (Bader et al., 2016).

INTRODUCTION TO WEB APPLICATION PERFORMANCE OPTIMIZATION

Web application performance and particularly its response time plays a major role on how users will interact and remain engaged on the application. The applications that load slowly will mean that users are bouncing away, users are not engaging with the app and at the end, fewer conversions are made. Further, it can impact the basic functionality and interaction by making a web application less user-friendly and approachable to frustration. Given the high competition in most markets, any form of performance enhancement will always be desirable and give a competitive advantage besides improving search engine standing, user satisfaction, and business prowess. That is why performance optimization falls under the major goal of providing the best user experience and meeting business needs (Cui et al., 2018).

IMPORTANCE OF WEB APPLICATION SPEED AND RESPONSIVENESS

1. Database Indexing: When dealing with databases, indexing is used to enhance the process of searching and make the process faster. Indexes are created on the columns that are searched or used in join clauses so that the data can be retrieved easily.

2. Query Optimization: Simplify and improve query performance by using the EXPLAIN statement and identifying and altering bad queries. The use of SELECT * must be discouraged since it brings in all the fields from the table, making data transfer unnecessary in most cases. Use pagination and restrict the results when running a query to enhance query execution speed.

3. Connection Pooling: Design and use of connection pool to help to control the connection to the database. Connection pools in turn provide an added advantage of conservation of connections that have been opened so that it can help in less overhead of opening or closing down connections.

4. Load Balancing: Spread incoming requests across different servers using load balancers. Load balancing makes the website more scalable and available because no member server can be overloaded.

5. Server-Side Caching: Employ caching techniques on the server side to cache data that is frequently accessed by the application in memory. Cache can be employed by using technologies such as Redis or to store query results or session data to avoid making numerous database inquiries (Godfrey & Zulkernine, 2013).

BACK-END OPTIMIZATION TECHNIQUES FOR WEB APPLICATIONS

Front-end optimizations are concerned with enhancing the functionality of a Web application client-side components. Key techniques include:

1. Lazy Loading: Use lazy loading optimization technique, which helps to avoid loading unnecessary elements until they are required. This lessens the first page load time, and the time taken in rendering important content on a page. Lazy loading is used for images, videos and other media where only the content that can be displayed on a page is loaded at first.

2. Caching: Use this caching on the browser end and the server side to make frequent accesses of similar resources faster. Due to caching, subsequent request demands to the server are eliminated hence reducing the load time on the server. Caching can also be done through the setting of cache headers (i. e., Cache-Control) and use of CDN to cache items such as images, stylesheets, and scripts.

3. Minification and Compression: Compression and reduction of CSS, Javascript, and HTML formats to make the size small and enable fast loading. Minification means stripping out unnecessary characters and excessive white spaces while compression employs tools such as GZIP to further compress the size of the files.

4. Asynchronous Loading: Include the JavaScript files at the bottom of the page to avoid the scripts to delay the rendering of the page. In the script tags, you can use the `async` or the `defer` attributes for managing the loading of the JS resources.

5. Image Optimization: Reduce the image size from its original dimensions, reduce its size in kilobytes, or utilize more progressive file formats such as WebP. Choose images that react to the screen size and choose the right dimension for images in order to avoid the overloading of the device.

6. Content Delivery Networks (CDNs): Use CDNs to spread static assets over the geographically distributed servers. CDNs also make content load faster since they make clients download content from servers that are near them.

Through these front-end optimization practices developers will be able to improve the standards of the Web applications (Herbaut et al., 2016).

LEVERAGING LAZY LOADING FOR FASTER PAGE RENDERING

Lazy loading is a method of delaying the loading of resources that are not required at the moment in the web application to improve the speed. Lazy loading also allows developers to address rendering of important content first, the values of which many users seem to find most relevant on web pages during their first visits.

1. Images and Media: As for images and videos, lazy loading makes it possible for media elements to load only when they appear on the user's screen. This decreases the initial payload and enhances the rate of page loading. Some tools like the `loading="lazy"` attribute of HTML or employing the help of specific JavaScript libraries, for example, Lazysizes, can help in that.

2. JavaScript Modules: Lazy loading can be performed with the use of JavaScript modules and components to enhance performance. Webpack supports code splitting, and even React has a dynamic import function that can utilize it. This would mean loading only the required code and delaying unnecessary components to load for faster rendering of a page.

3. Frameworks and Libraries: Use frameworks and libraries that come with features for lazy loading components, for instance, Angular modules using lazy loading or Vue's asynchronous component loading. These tools help to make this task smoother and easily blend into today's development environments (Hasslinger et al., 2017).

IMPLEMENTING CACHING STRATEGIES FOR OPTIMAL PERFORMANCE

Caching is a feature that can greatly enhance a site's efficiency because less repeated server accesses and, therefore, less load times are required. Effective caching strategies include:

1. Browser Caching: Browser caching must also be optimized through the use of cache-control headers especially with reference to images, CSS and JavaScript files. Use expiration times to force the browser to cache resources and use the cached resources instead of sending out new requests on the next visits.

2. Server-Side Caching: Use server-side caching where it is appropriate to store data that frequently needs to be retrieved. They consist of page caching, object caching, and fragment caching. There is special software such as Redis which can help to cache results of database queries to decrease the intensity of requests on the backend.

3. Content Delivery Networks (CDNs): Employ CDNs to cache the static content across the multiple servers, in the different geographical regions. CDNs store content closer to users which reduces latency time to deliver the content (Odell & Odell, 2014).

MINIFYING JAVASCRIPT AND CSS FOR ENHANCED LOAD TIMES

Minification specifically refers to the process of eliminating unnecessary characters, white spaces and comments from the actual JavaScript and CSS code without compromising on functionality. This decreases the size of files that have to be downloaded and can speed up the process. Key practices include:

1. Minification Tools: Minification can be done manually using services like UglifyJS, Terser, or CSSNano in order to optimize the workflow. To be specific it means these tools effectively minify JavaScript and CSS files and this process results in less and faster downloading of the files.

2. Build Processes: Minification should be included as a part of the build process with the help of utilities, such as Gulp or Grunt, or as a part of bundlers such as Webpack. By integrating this step into the build process, it is guaranteed that a file is always delivered in the smallest size possible.

3. Compression: Combine minification with the compression methods such as GZIP or Brotli. Compression also helps reduce file sizes during the transmission process, which increases the performance and minimizes the amount of time it takes to load the information.

Removing the bytes from JavaScript and CSS files is one of the mainstream optimizations that help in making page loads faster and providing faster user engagement (Qu et al., 2018).

IMAGE OPTIMIZATION: REDUCING FILE SIZE FOR FASTER LOADING

Optimizing images is a critical issue to address as far as enhancing the performance of the web application is concerned since big image files place a lot of burden on the load time. Effective image optimization techniques include:

1. Compression: Optimize image files by resizing and compressing them with tools such as ImageMagick, TinyPNG or JPEGoptim. Compression shrinks image's file size down making images load faster.

2. Responsive Images: Make use of the 'srcset' html attribute to make images responsive. This will make sure that right sizes of images are delivered based on the device type of the users and their screen resolutions thus saving on the amount of traffic to be covered on the internet.

3. Modern Formats: Migrate images to formats that are new and more efficient, like WebP that provides better image compression than other formats like JPEG or PNG. WebP images are significantly smaller and take less time as compared to other images to load, thereby enhancing performance (Odell & Odell, 2014).

BACK-END OPTIMIZATION TECHNIQUES FOR WEB APPLICATIONS

Back-end optimization is about enhancing the performance of the server side in order to enable efficient processing and delivery of data. Key techniques include:

1. Efficient Code Practices: Provide well-tuned code on the server to handle requests. Lessen computational complexity and attempt to use optimal strategies of data processing.

2. Load Balancing: In load balancing, the incoming request is divided between a set of servers to optimize resource utilization and prevent overloading of any particular server. This keeps the load uniformly balanced, enhancing the capability for better system reliability and scalability, hence making the handling of traffic easier and improving application responsiveness and uptime.

3. Asynchronous Processing: For long running operations, it is advisable for the application to employ asynchronous processing in order to avoid blocking. There are a number of mechanisms that can be used to handle such tasks as message queues, for instance, RabbitMQ and background workers (Ahmed et al., 2016).

DATABASE INDEXING: IMPROVING QUERY PERFORMANCE

Indexing is one of the most important options for improving the efficiency of queries in a database and minimizing response time. Key practices include:

1. Creating Indexes: Determine which columns are most often used in inquiries and establish indexes for quick access. It has been established that the general execution time of a query can be greatly reduced when appropriate indexing is applied.

2. Analyzing Query Plans: Logically, use database query analyzer to analyze the execution plan and look for inefficient queries. Tune queries and indexes based on said analyses.

3. Maintaining Indexes: Indexes should be constantly checked to determine their efficiency and effectiveness, and necessary actions taken. To enhance the performance, it is good to eliminate indexes that are not frequently used in the database (Bader et al., 2016).

OPTIMIZING SQL QUERIES FOR FASTER DATA RETRIEVAL

The efficient optimization of SQL queries is crucial in enhancing the speeds at which web applications retrieve data. Key strategies include:

1. Indexing: Index the columns that are used in search conditions frequently or are included in joins. Indexing helps in fast retrieval since the database can quickly identify the necessary rows from the table.

2. Query Refinement: All the fields don't need to be fetched in a query instead of fetching all fields using * since it causes unnecessary transportation of large data. Also, use WHERE clauses to filter data to enhance the query performance and speed.
3. Execution Plans: Deconstruct statements and queries and analyze the database for executing them and coming up with a query plan. There are options that, for instance, MySQL `EXPLAIN` or PostgreSQL's `EXPLAIN ANALYZE` that can give hints on how a particular query is executed and may be useful in adjusting it to one's performance parameters.
4. Avoiding Complex Joins: Avoid complex joins and subqueries as much as possible. Expounding complex queries into simple ones may help in minimizing the time to execute and thus enhance its efficiency.
It has been observed that the use of these optimization techniques helps in reducing the time needed to retrieve data within web applications and thus enhance their performance (Cui et al., 2018).

UTILIZING CONTENT DELIVERY NETWORKS (CDNS) TO REDUCE LATENCY

CDNs are a valuable tool as they help to minimize the latency and increase speed of delivery of web assets.
1. Global Distribution: CDNs replicate content on servers based in different geographical locations around the world. From this, it makes certain that the content that is being delivered is from the nearest server hence there is little delay in the loading of the content.
2. Static Asset Caching: One should also utilize CDN to help in caching of the static assets such as images, CSS and JavaScript. This helps to decrease the number of requests to the origin server and speeds up the delivery of content.
3. Dynamic Content Acceleration: There are CDNs that have dynamic content acceleration where the delivery of dynamic content is made faster through the use of certain methods such as TCP optimizations and edge computing (Godfrey & Zulkernine, 2013).

ENHANCING SERVER-SIDE PERFORMANCE WITH LOAD BALANCING

Load balancing improves the performance on the server side by distributing the frequent traffic over multiple servers so that no server is burdened, which then ensures optimum response times and reliability. Key strategies include:
1. Round-Robin Distribution: When it comes to addressing the incoming requests, one should establish round-robin load balancing that spreads all the requests in turn.
2. Health Monitoring: Employ load balancers to constantly check the health status and manage traffic flow to eliminate damaged servers. This ensures high availability and reliability of the system, maintaining consistent performance and uptime.
3. Scalability: Load balancing makes it easy to scale through increased servers in order to handle the load of traffic. It also guarantees that the application can expand its capacity to handle users' needs as they increase.
Load balancing, thus, can help web applications improve server-side performance, scalability, and availability of the application (Herbaut et al., 2016).

CASE STUDY: BOOSTING PERFORMANCE FOR AN E-COMMERCE PLATFORM WITH LAZY LOADING AND QUERY OPTIMIZATION

In a case study involving an e-commerce platform, performance was significantly improved through the implementation of lazy loading and query optimization:
1. Lazy Loading: The site incorporated the use of lazy loading for images and product information so as to minimize the time taken to download the pages at initial stage. For better perception and user experience, images were only loaded as the user scrolled the page.
2. Query Optimization: Optimizations to common SQL queries included making indexes on indexes that were searched for in columns and basic paradigm queries. This enabled the system to retrieve data in a faster manner and also improve the overall response time of the applications (Hasslinger et al., 2017).

CONCLUSION

In conclusion, performance optimization is very important to be worked out on both client and server sides. By applying front-end strategies, such as lazy loading or caching, and some back-end improvements, such as SQL query optimization and load balancing, it's possible for developers to considerably increase the efficiency of an application. Moreover, CDNs ensure things run even smoother, while the efficiency of server-side resources does its magic in rendering a smooth and friendly web experience. All these efforts taken together interrelate to ensure that web applications can deliver speed, reliability, and quality performance, which will lead to increased user satisfaction and more successful businesses (Odell & Odell, 2014).

REFERENCES

[1]. Ahmed, T. M., Bezemer, C. P., Chen, T. H., Hassan, A. E., & Shang, W. (2016, May). Studying the effectiveness of application performance management (apm) tools for detecting performance regressions for web applications: an experience report. In Proceedings of the 13th International Conference on Mining Software Repositories (pp. 1-12).

[2]. Bader, A., Ghazzai, H., Kadri, A., & Alouini, M. S. (2016). Front-end intelligence for large-scale application-oriented internet-of-things. IEEE Access, 4, 3257-3272.

[3]. Cui, S., Asghar, M. R., & Russello, G. (2018). Multi-CDN: Towards privacy in content delivery networks. IEEE Transactions on Dependable and Secure Computing, 17(5), 984-999.

[4]. Godfrey, M., & Zulkernine, M. (2013, June). A server-side solution to cache-based side-channel attacks in the cloud. In 2013 IEEE Sixth International Conference on Cloud Computing (pp. 163-170). IEEE.

[5]. Herbaut, N., Négru, D., Chen, Y., Frangoudis, P. A., & Ksentini, A. (2016, December). Content delivery networks as a virtual network function: A win-win ISP-CDN collaboration. In 2016 IEEE Global Communications Conference (GLOBECOM) (pp. 1-6). IEEE.

[6]. Hasslinger, G., Ntougias, K., Hasslinger, F., & Hohlfeld, O. (2017). Performance evaluation for new web caching strategies combining LRU with score based object selection. Computer Networks, 125, 172-186.

[7]. Odell, D., & Odell, D. (2014). Boosting JavaScript Performance. Pro JavaScript Development: Coding, Capabilities, and Tooling, 91-118.

[8]. Qu, C., Calheiros, R. N., & Buyya, R. (2018). Auto-scaling web applications in clouds: A taxonomy and survey. ACM Computing Surveys (CSUR), 51(4), 1-33.