International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 05 | May 2020 www.irjet.net p-ISSN: 2395-0072

Study on Performance Tuning of Web Applications using Ad-Hoc Techniques

Arnab Jana[1], Raunak Kedia[2], Merin Meleet[3]

[1]*Student, Dept. of Information Science and Engineering, R.V College of Engineering, Karnataka, India*
[2]*Student, Dept. of Computer Science and Engineering, Birla Institute of Technology Mesra, Jharkhand, India*
[3] *Professor, Dept. of Information Science and Engineering, R.V College of Engineering, Karnataka, India*

Abstract - *Internet is the connection of millions of networks and a billion of Web Pages. Every web page hosted on the Internet are being hit by users across the globe. In such a scenario it is really important to have Web app with great performance if it has to be honoured by its users.*
This review paper highlights some of the mechanisms for improving the performance of a Web Application. Here performance can be regarded in terms of metrics like Response Time and Storage improvements. Response time is the time lapse between the submit of request and the first response recorded. In general the most theoretically best web site should render all the components in no time, but this isn't practical ever. When a user submits some request to a website, the request is marshalled from the client side, DNS mapping is done to obtain the server IP, request travels through network channel, Received in the Server end, Unmarshalled and finally received by the Web Server. The response journey happens the same way but in opposite direction. This overall request-response cycle takes a prominent amount of time hindering the web page to load quickly.
The above process can't be neglected, but though the web applications performance can be improved using some techniques like Caching, Aggregating response, Decreasing Contents etc. Various mechanisms and relevant ways for making API Calls are also discussed. By using certain tweaks and following certain conventions for writing the Web languages like HTML, CSS and JavaScript can show significant improvement in the response time of the application. Techniques like Compression of the Media contents, Progressive rendering, Minifying Scripts etc are discussed in brief.

Key Words: **Response time, Optimization, HTML, CSS, XHTML, Progressive Rendering**

1.INTRODUCTION

From the client point of view web performance is measured as the load time of the page. That is the time lapse between a user accessing a new page and the first instance of the browser rendering that website. Fast web pages render progressively. That is, their content is shown incrementally, when the browser loads it. A web page that gradually renders visual feedback to the user that the page is being loaded, and provides the user with the information they have requested as soon as it is available. Yahoo and Google has also recommended best practices for having web pages rendered progressively, such as adding style sheets in the header of the text.

You may apply some additional best practices to optimize progressive rendering for most websites. First, a simple page will make the content first accessible to the user, and then make the content off-screen (i.e. the content outside the current scroll region) later. Second, before loading and rendering heavyweight resources like images and video, a quick page might also load and render the lightweight resources such as text. Some Techniques on the other hand are known to do progressive rendering in the background. Applying style sheets late in the document can also prevent progressive rendering, even if those style sheets aren't required for the initial page load.

Sometimes it is also needed to store some amount of data that is repeatedly requested by the user. This technique is called Caching. Many recent browsers do enable caching at various levels. Two prominent caching techniques used for reducing the response time are Browser Caching(Cache maintained by the browser) and Content Caching (where caching is done at the ISP level during DNS mapping).

One of the other key aspects that affects the performance of the web app is the heaviness of the application. Heaviness is measured in terms of the data, dependencies or modules required by the application. Sometimes the application makes use of large number of dependencies and modules and then the developer should do a trade off on what all dependencies are needed to be packaged with the application files and what all to be received using CDNs (content delivery network). It is recommended to go for the dependencies packaged only when it is used extensively within the application. If the dependency or plugin is used to aid some smaller changes, going for a CDN is a wise decision.

Next section discusses various techniques using which we can improve the performance of a web application.

2. PROPOSED SOLUTION

2.1 Amortizing the API calls

Any web app generally is a composure of multiple components that are displayed on a unified interface which tends to interact with its user. Sometimes these components are standalone instructions to its users and sometimes they are informative with a power of data boosted to them. When a user interacts with any component like buttons it triggers some event which calls a function or makes a series of API calls to complete its request.

So, in the overall lifetime in which a user interacts with the app, they tend to make a large number of API calls unknowingly. If on an average each of the request takes around 2 sec to respond and if there are around 10 API calls then for 20 sec almost, the user is bound to wait. This type of interaction on a live web page accessed by millions of users can lead to very bad impression regarding the site, hence could lead to a bad User Experience.

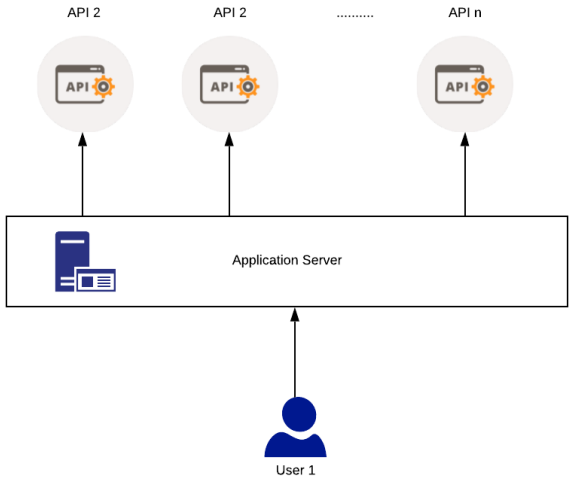

Fig -1: Shows the general architectural design of any conventional Application.

One of the solutions to tackle such a scenario is to amortize the average response time experienced by a user.

In this solution, suppose there exist a fast retrieval storage called a Data Cache which is a large number of flat files in some Structured or Semi-Structured format like JSON or XML. Then these files will hold the most recent updated results of all the possible API calls that can be made to that web app.

Any possible event made by the user will then fetch the details from this Data Cache which corresponds to the same information being fetched from its equivalent API call at a definite time stamp.

As the time required to do read operations on a file residing in the server takes lesser time than any API request(where the time taken depends upon the third party who created that API endpoint), it can drastically improve the overall experience of any user using that web app.

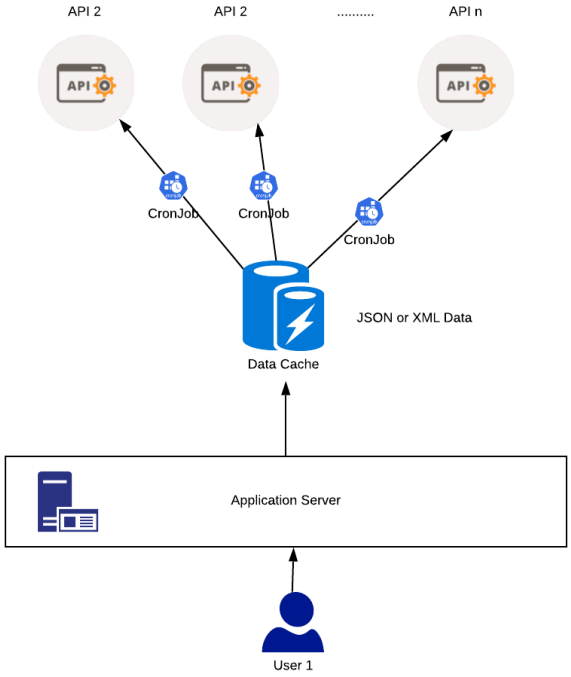

Fig -2: Addition of a Data Cache

Here one of the key factors that impacts the performance is the *time interval* chosen for *refreshing the Data Cache*. This mostly depends upon the business use case of the application. One sample scenario may be:

If the web app's data is changing at an interval of some days then a 12hr interval would be descent to go for updating the Data Cache contents as it will utilize the unproductive time i.e when people are sleeping to update the Cache to latest contents.

Few of the key note is that this technique can only be beneficial if :

1. The data contents of the application is more of static in nature than volatile:

That is if an API is continuously updated with new content, the app will show the information pertaining to the last timestamp in which data was refreshed. Hence information displayed is a stale information.

2. The Data in semi-structured formatted is as minimal as required for the proper working of the application:

As the data obtained from API calls in most of the cases are too bulky and only a subset of the information is actually needed. So only that subset is stored in the data cache which is actually needed by the application can reduce down the access time further with a bit of improvement on the storage also.

2.2 Contents Caching

Contents Caching is a way of storing the most recent results in a cache server which renders the data quickly as compared to the data received from the original source server.

When a user makes request to a webpage rendering static contents like images, videos, stylesheets then the browser stores some information in its local cache. If the data isn't available there then the user browser submits request to the CDN. CDN uses Anycast DNS to route to the closes server which most of the times stores some request results locally within. Hence if the results are found from this Cache server it loads instantly as compared to the requests being served by Original server.

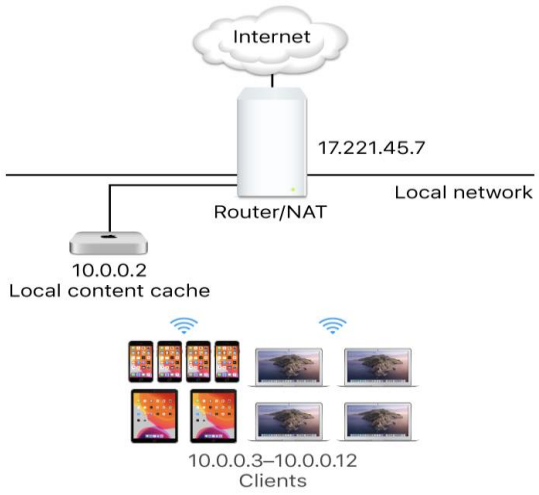

Fig -3: Contents Caching working

Thus Content Caching can also improve the response activity of the web pages. Fig. 2.3 shows how contents caching works for Apple based system.

2.3 Compressing the Media Objects

Image files often contains extra information embedded in the files. Sometimes their contextual use demands their sizes to be small and hence having that much detailed information is not needed. The images can then be compressed using some lossless compression to achieve the same output but with improved loading time. Examples like JPEG image files mostly contains the name of the program name used to write them. Similarly other file formats like GIF and PNG files can be made smaller by making use of the way they are encoded. Certain techniques like Huffman encoding, Arithmetic encoding and Run length encoding can be used for the compression.

2.3 Minified JavaScript and CSS files

These days most of the 3rd Party libraries and Frameworks are written and packaged along with their minified versions. These minified code are almost similar to the original code with the unwanted items removed from the original doc. Examples like min.js and min.css files are the same representation of the JS and CSS files but with the spaces and tabs removed from them and only retaining the useful code.

2.4 Progressive Rendering

Sometimes the content of the web pages involves too many components which gets revealed upon various actions by the user. These contents most of the time are not required to be rendered at the same time as other core components and hence can be processed in intervals or only upon their requirement. Some of the examples of this may be Showing rest of the Web Page Contents upon scrolling, Button triggered data rending via AJAX calls etc.

Thus Progressive Rendering can bring down the number of components being rendered at the moment of time and hence improves the response time of the websites.

3. CONCLUSION

The size of the average web page has more than tripled in the last five years and the number of external objects has almost doubled. Although broadband users encountered quicker loading times, users of narrowband were left behind. For the typical web page hosting more than 50 external objects, most web page delays are now controlled by object overhead.

Lesser the number of HTTP requests, lesser multimedia contents combining JavaScript or CSS, while still retaining the attractiveness of the web page reduces the load on the rendering of the web page. Maintaining this trade-off is

really crucial for creating a good content for the web site. It is evident that most of the website which shows minimal required data with the most attractive designs and animation attracts the users view a lot.

4. REFERENCES

[1] V. Jain and A. Kolambkar, "Modeling Web Attachment Storage for Web Applications," 2014 21st Asia-Pacific Software Engineering Conference, Jeju, 2014, pp. 98-102, doi: 10.1109/APSEC.2014.24.

[2] G. Jiang and S. Jiang, "A Quick Testing Model of Web Performance Based on Testing Flow and its Application," 2009 Sixth Web Information Systems and Applications Conference, Xuzhou, Jiangsu, 2009, pp. 57-61, doi: 10.1109/WISA.2009.16.

[3] K. Gupta and M. Mathuria, "Improving performance of web application approaches using connection pooling," 2017 International conference of Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, 2017, pp. 355-358, doi: 10.1109/ICECA.2017.8212833.

[4] T. Gao, Y. Ge, G. Wu and J. Ni, "A Reactivity-based Framework of Automated Performance Testing for Web Applications," 2010 Ninth International Symposium on Distributed Computing and Applications to Business, Engineering and Science, Hong Kong, 2010, pp. 593-597, doi: 10.1109/DCABES.2010.127.

[5] A. Puliafito, M. Scarpa, A. Zaia and M. Villari , "A modeling technique for the performance analysis of Web searching applications," in IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 11, pp. 1339-1356, Nov. 2004, doi: 10.1109/TKDE.2004.65.

[6] Z. Zhu, X. Fei and Y. Gaizhen, "Research on Performance Optimization for the Web-Based University Educational Management Information System," 2011 International Conference on Intelligence Science and Information Engineering, Wuhan, 2011, pp. 261-264, doi: 10.1109/ISIE.2011.53.

[7] J. Chen and S. Yang, "A Study of Security and Performance Issues in Designing Web-based Applications," IEEE International Conference on e-Business Engineering (ICEBE'07), Hong Kong, 2007, pp. 81-88, doi: 10.1109/ICEBE.2007.44.

[8] P. Davies, N. Naik, D. Newell, P. Jenkins, and, "Native Web Communication Protocols and Their Effects on the Performance of Web Services and Systems," 2016 IEEE International Conference on Computer and Information Technology (CIT), Nadi, 2016, pp. 219-225, doi: 10.1109/CIT.2016.100.