© UNIVERSAL RESEARCH REPORTS | REFEREED | PEER REVIEWED
ISSN : 2348 - 5612 | Volume : 05 , Issue : 01 | January – March 2018

Study of web application performance and techniques for acceleration and optimization

[1]Indumati, [2]Dr. Pankaj Verma
[1]Research Scholar, [2]A.P., Department of CSA, Niilm University

Abstract : Improving web application performance is more critical than ever. The share of economic activity that's online is growing; more than 5% of the developed world's economy is now on the Internet. And our always-on, hyper-connected modern world means that user expectations are higher than ever. If your site does not respond instantly, or if your app does not work without delay, users quickly move on to your competitors.

Key Words : Web Application, Performance, Etc

Introduction : A study done by Amazon almost 10 years ago proved that, even then, a 100-millisecond decrease in page-loading time translated to a 1% increase in its revenue. Another recent study highlighted the fact that that more than half of site owners surveyed said they lost revenue or customers due to poor application performance.

How fast does a website need to be? For each second a page takes to load, about 4% of users abandon it. Top ecommerce sites offer a time to first interaction ranging from one to three seconds, which offers the highest conversion rate. It's clear that the stakes for web application performance are high and likely to grow. In this paper we outline some of the techniques for improvements in security that you can gain along the way.

Improving performance and optimization

Accelerate and Secure Applications with a Reverse Proxy Server

If your web application runs on a single machine, the solution to performance problems might seem obvious: just get a faster machine, with more processor, more RAM, a fast disk array, and so on. Then the new machine can run your WordPress server, Node.js application, Java

application, etc., faster than before. (If your application accesses a database server, the solution might still seem simple: get two faster machines, and a faster connection between them.)

Trouble is, machine speed might not be the problem. Web applications often run slowly because the computer is switching among different kinds of tasks: interacting with users on thousands of connections, accessing files from disk, and running application code, among others. The application server may be thrashing – running out of memory, swapping chunks of memory out to disk, and making many requests wait on a single task such as disk I/O.

Instead of upgrading your hardware, you can take an entirely different approach: adding a reverse proxy server to offload some of these tasks. A reverse proxy server sits in front of the machine running the application and handles Internet traffic. Only the reverse proxy server is connected directly to the Internet; communication with the application servers is over a fast internal network.

Using a reverse proxy server frees the application server from having to wait for users to interact with the web app and lets it concentrate on building pages for the reverse proxy server to send across the Internet. The application server, which no longer has to wait for client responses, can run at speeds close to those achieved in optimized benchmarks.

Adding a reverse proxy server also adds flexibility to your web server setup. For instance, if a server of a given type is overloaded, another server of the same type can easily be added; if a server is down, it can easily be replaced.

Because of the flexibility it provides, a reverse proxy server is also a prerequisite for many other performance-boosting capabilities, such as:

Load balancing – A load balancer runs on a reverse proxy server to share traffic evenly across a number of application servers. With a load balancer in place, you can add application servers without changing your application at all.

Caching static files – Files that are requested directly, such as image files or code files, can be stored on the reverse proxy server and sent directly to the client, which serves assets more quickly and offloads the application server, allowing the application to run faster.
Securing your site – The reverse proxy server can be configured for high security and monitored for fast recognition and response to attacks, keeping the application servers protected.

The performance improvements that are available for any one web application vary tremendously, and actual gains depend on your budget, the time you can invest, and gaps in your existing implementation.

Reverse proxy server and load balancing – No load balancing, or poor load balancing, can cause episodes of very poor performance. Adding a reverse proxy server, , can prevent web applications from thrashing between memory and disk. Load balancing can move processing from overburdened servers to available ones and make scaling easy. These changes can result in dramatic performance improvement, with a 10x improvement easily achieved compared to the worst moments for your current implementation, and lesser but substantial achievements available for overall performance.

Caching dynamic and static content – If you have an overburdened web server that's doubling as your application server, 10x improvements in peak-time performance can be achieved by caching dynamic content alone. Caching for static files can improve performance by single-digit multiples as well.

Compressing data – Using media file compression such as JPEG for photos, PNG for graphics, MPEG-4 for movies, and MP3 for music files can greatly improve performance. Once these are all in use, then compressing text data (code and HTML) can improve initial page load times by a factor of two.

Optimizing SSL/TLS – Secure handshakes can have a big impact on performance, so optimizing them can lead to perhaps a 2x improvement in initial responsiveness, particularly for text-heavy sites. Optimizing media file transmission under SSL/TLS is likely to yield only small performance improvements.

Implementing HTTP/2 and SPDY – When used with SSL/TLS, these protocols are likely to result in incremental improvements for overall site performance.

References :

1. J. P. Bigham, C. M. Prince, and R. E. Ladner. Addressing performance and security in a screen reading web application that enables accessibility anywhere. In 8th Int'l Conf. on Web Engineering (ICWE), 2008.
2. J. Chen, D. Hutchful, W. Thies, and L. Subramanian. Analyzing and accelerating web access in a school in peri-urban India. In 20th Int'l Conf. on World Wide Web (WWW), 2011.
3. J. Chen, L. Subramanian, and J. Li. RuralCafe: Web search in the rural developing world. In 18th Int'l Conf. on World Wide Web (WWW), 2009.
4. E. Cuervo, A. Balasubramanian, D.-K. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. MAUI: Making smartphones last longer with code offload. In 8th Int'l Conf. on Mobile Systems, Applications, and Services (MobiSys), 2010.
5. B. Du, M. Demmer, and E. Brewer. Analysis of WWW traffic in Cambodia and Ghana. In 15th Int'l Conf. on World Wide Web (WWW), 2006.