Varnish Web Cache Application Evaluation

Justinus Andjarwirawan, Ibnu Gunawan, and Eko Bayu Kusumo

Informatics Department, Petra Christian University, Surabaya, Indonesia
{justin,ibnu}@petra.ac.id,
m26410006@alumni.petra.ac.id

Abstract. Websites today have static and dynamic contents that concern many people about the performance. People will leave or not returning the website because of loading time that is taking too long. One way to speed up a website access is by utilising web applications using Varnish cache. Varnish will store static content from a website in the memory. Static data is data that changes infrequently or rarely updated. A front end for web administrators was built for Varnish application so that the configuration can be done easily. The configuration that will be simplified through the front end of this Varnish application is detecting SQL injection and Cross Site Scripting (XSS), block files and folders, HTTP header manipulation, detects the original file whether it is deleted or blocked and error handling configuration. The front end application is implemented on Linux platform. Varnish application was tested using ApacheBench where the number of requests from the client and the response time become the main parameters of the test.

Keywords: web cache, reverse proxy, varnish.

1 Introduction

Fast access to websites is everyone's demand at this time. Website visitors leave the website when loading is too long. Websites often have additional content that do not support the main information of the website. This additional content will slow down the access to the website. On a website there are static and dynamic content. Static content is content that is updated infrequently or rarely changes. Dynamic content is content that is frequently updated or there is always a change. Static content such as header, sidebar and footer while dynamic content such as news content, gallery or video. The static content can be stored in a memory so that dynamic content data access to the server faster. The data stored in this memory will speed up static content access on the website so that if a user visits a lot and load dynamic content, server access data will be lighter.

Another thing that becomes a problem is if the main server where the data stored is far from the location of a visitor. The time required to transfer the necessary data will be longer. The data in this memory is needed to accelerate the transfer of this data. For example, someone has a web server in the U.S., but the majority of website visitors are from Indonesia. Data access will be a little longer because the web server

R. Intan et al. (Eds.): ICSIIT 2015, CCIS 516, pp. 404–410, 2015.
DOI: 10.1007/978-3-662-46742-8_37

is away from the users. Data in the memory also helps keep some frequently accessed data.

Web caching is a solution to this problem by using a web cache called Varnish. With Varnish, one can store all the data information from the primary server to the Varnish. Static data such as images, videos and other things related to the website may be stored in this Varnish. Web admin can also set the Time To Live (TTL) of each website fragment by using the concept of Edge Side Includes (ESI). Website content that is rarely updated like the header or footer can be set up its time to live for example like one month. During one month, Varnish automatically update its cache site. Content such as news or advertising, can be set e.g. for 1 hour to be updated. Distribution of the update time is beneficial to reduce data access to major web servers and data received the most recent data.

If a website is accessible to many visitors the loading will be quite long. A Varnish cache site is to help increase the speed of the website access. It is expected to help the problem on the server load and access data through the use of Varnish web cache. Implementing Varnish cache in this research is to define how significant it is to use on highly loaded web servers.

2 Theoretical Background

Varnish cache is a web accelerator application, also known as Hyper Text Transfer Protocol (HTTP) reverse proxy. Varnish was first introduced in 2006 by Poul-Henning Kamp. Reverse proxy is a proxy that is in the front-end of web servers to act as a cache. Varnish work on the front end of an HTTP server and can be configured to cache every website content. Varnish's main working principle is to store the data of a web page in memory, thus reducing the load on the web server loading the same page (Winkler 2012). Here the web application without caching process can be seen in Figure 1:

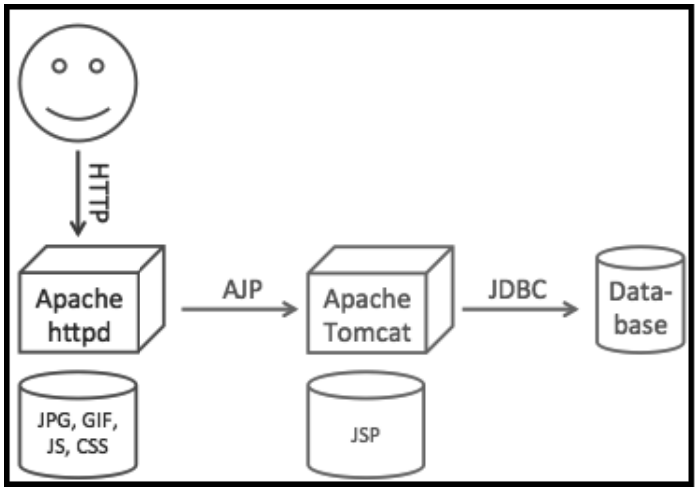

Fig. 1. Web application without cache

The three main components of the web application is Apache HTTPD, Apache Tomcat and database. Apache set static requests for images, scripts and others. In addition there is a forward requests for pages Hyper Text Markup Language (HTML) to application servers such as Apache Tomcat and then request to the database. After successfully getting the information request, it will be returned to Apache and then to the user. The red component is the slowest component in its performance due to the long process of searching data for this component. (Winkler 2012)

If researchers apply the varnish on a website, Varnish HTTP accelerator will serve as the store (cache) a copy of each page of the website. When a user accesses the website back, Varnish will provide a copy of the data of the Apache server. Varnish will greatly assist access to a website with a system of storing data in the cache memory of this site. The following implementation of Varnish web cache can be seen in Figure 2:

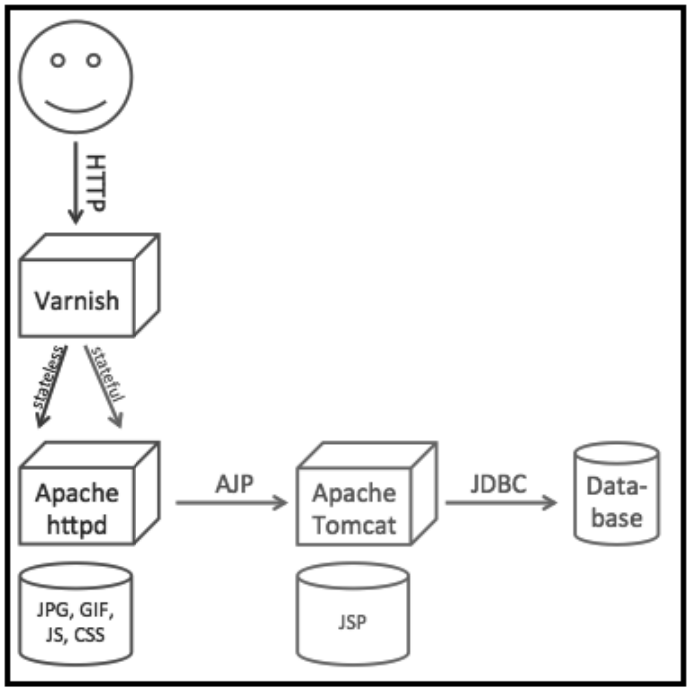

Fig. 2. Web server with Varnish web cache

Varnish is a web cache that will be divided into two: stateless and stateful. Stateless is data with static content, while stateful is data with dynamic content. Varnish only serves requests stateless whereas for stateful request will be returned to Apache web server.

VCL (Varnish Configuration Language) is a programming language used in the Varnish web cache. VCL syntax is similar to programming in C language.

3 Research Model and Hypotheses

3.1 System Analysis

Varnish that has been installed can be accessed via the default port 80. Varnish will be at the front end admin so that requests data going through Varnish will be checked first before heading to Apache. Request a stateless data will be managed by Varnish while stateful data request will be returned to Apache. If there is a request then the request will be checked by Varnish whether it is already cached. If not cached, then requests will be forwarded to the Apache. If there is a page-request it will weigh on the performance of Apache. Varnish will assist in arranging Apache request the same page with the system cache in-memory store. The system after using Varnish can be seen in Figure 3:

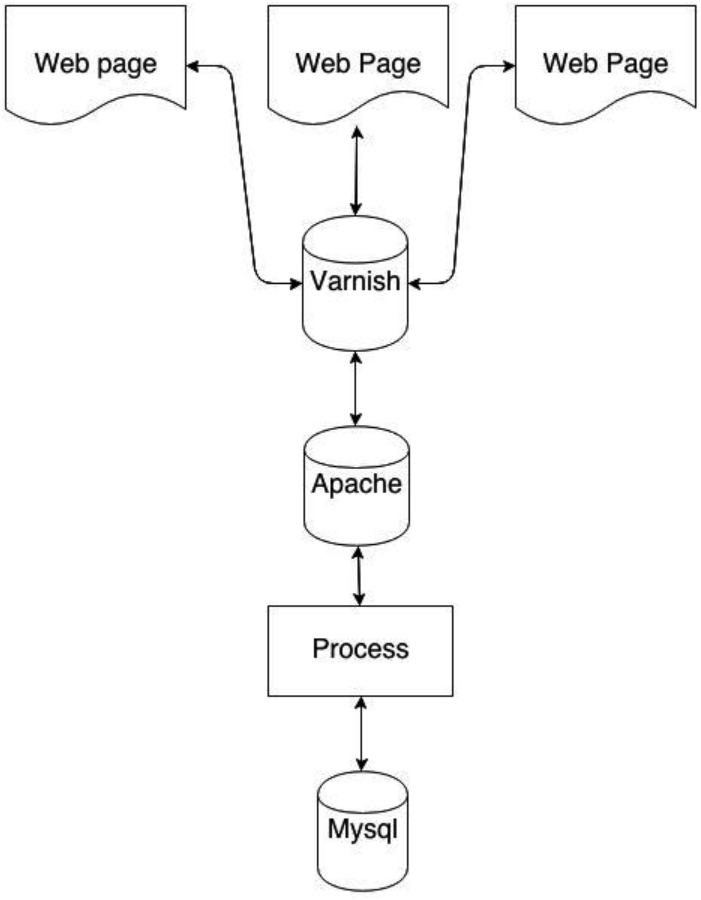

Fig. 3. System after using Varnish

The analysis test bed is to separate static and dynamic content to decide which part of the web to be cached by Varnish. The separation needs to be done manually by the web administrator.

3.2 Design of Experiment

Three websites were developed for the test. One site has 10 kilobytes of static content, the second site with 4 kilobytes and third site 15 kilobytes with mixed static and dynamic content with PHP backend. These three websites were placed locally on the same machine as the benchmark tool ApacheBench.

4 Web Cache Performance Test

Gnuplot is used for the visualization in the form of graphs. In Figure 4 it can be seen that the response time benchmarks without Varnish starts to weigh when there are 40 incoming requests. While the response time benchmarks via Varnish remains stable during the 40 incoming request. Response time is the longest of the benchmark, without Varnish is 5048ms and reponse time longest of benchmarks with Varnish is 264ms with a maximum of 100 requests. Graph on figures are generated using Gnuplot [2].

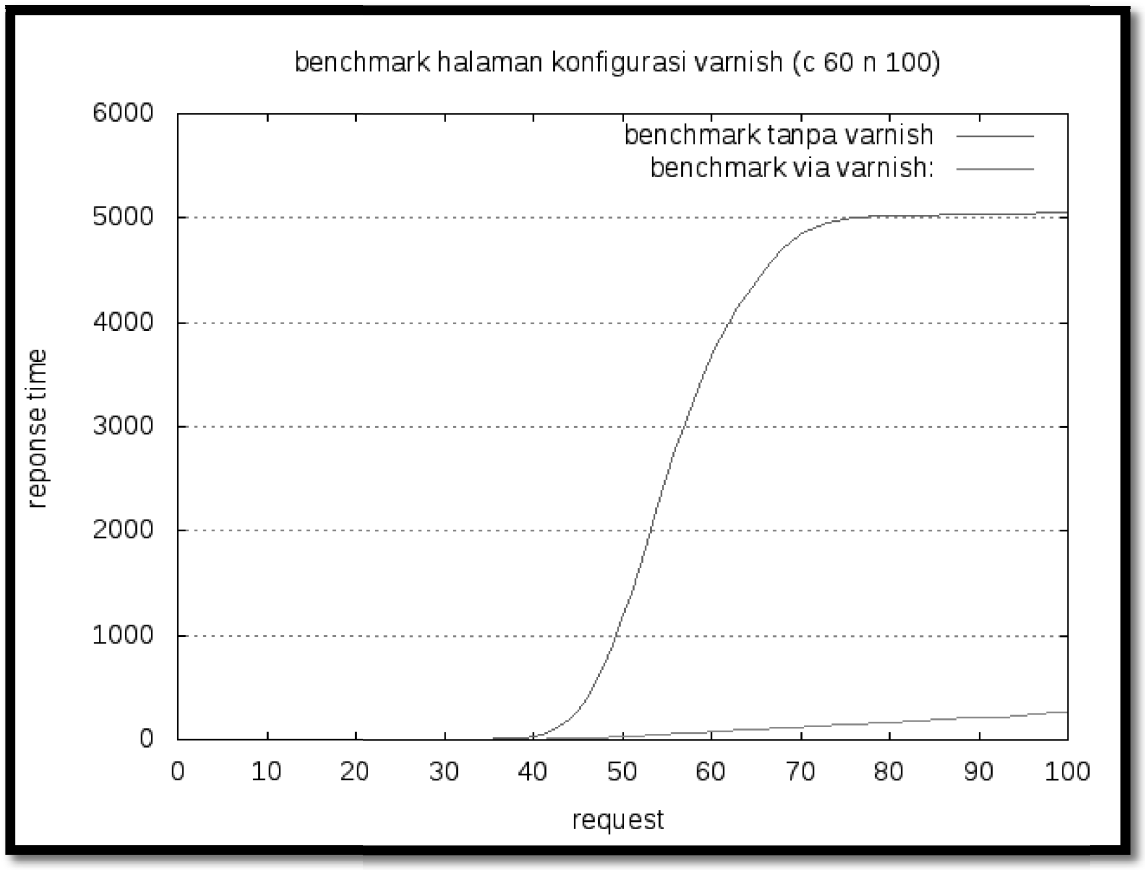

Fig. 4. Benchmark test

Using ApacheBench for benchmarking, setting a concurrent users value to 296 and number of requests 300, the following table shows the connection time (ctime), processing time (dtime), total time (ttime) and waiting time:

Table 1. Varnish benchmark test

starttime	seconds	ctime	dtime	ttime	wait	id user
Wed May 28 10:42:45 2014	1401248565	0	1	1	1	1
Wed May 28 10:42:45 2014	1401248565	0	2	2	2	2
Wed May 28 10:42:45 2014	1401248565	0	2	2	2	3
Wed May 28 10:42:45 2014	1401248565	0	3	3	3	4
Wed May 28 10:42:45 2014	1401248565	8	2	11	1	5
Wed May 28 10:42:45 2014	1401248565	8	2	11	2	6
Wed May 28 10:42:45 2014	1401248565	8	2	11	2	7
Wed May 28 10:42:45 2014	1401248565	8	2	11	2	8
Wed May 28 10:42:45 2014	1401248565	8	3	11	2	9
Wed May 28 10:42:45 2014	1401248565	8	3	11	3	10
Wed May 28 10:42:45 2014	1401248565	9	2	11	2	11
Wed May 28 10:42:45 2014	1401248565	8	3	11	3	12

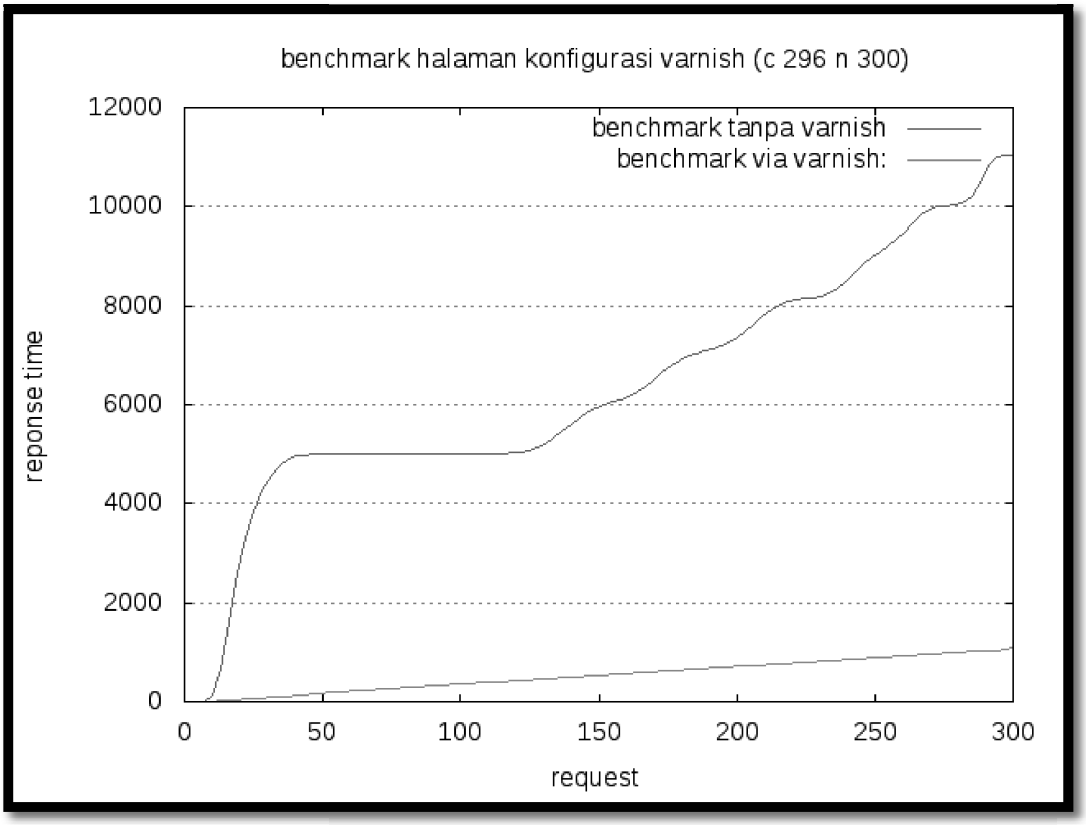

Fig. 5. Benchmarking Varnish

The table above is shown as figure in Figure 5:

The longest response time on the 300 users configuration without using Varnish is 11047ms compared to 1087ms with using Varnish.

The remaining tests with various numbers of concurrent users and requests showed similar conditions from Figure 5.

Varnish has a built-in script feature called VCL (Varnish Configuration Language) for making its configuration and also a security feature to prevent XSS (Cross Site Scripting).

5 Conclusion

The evaluation showed that Varnish is accepted significantly on static contents of websites. Not very useful for websites which have frequent updates or very dynamic content changes. Varnish still need cache clearing mechanism to delete cached contents in order to retrieve updated content after a period of time. It can also be concluded that Varnish is very recommended to implement on large scale web servers with high concurrency load.

References

1. Christian, W.: Ultra-Performant Dynamic Websites with Varnish (2012), http://blog.mgm-tp.com/2012/01/varnish-web-cache/ (accessed November 18, 2013)
2. Gnuplot, Gnuplot Homepage (2013), http://gnuplot.info/ (accessed November 18, 2013)
3. Silicon-Press. Web Caching: What is Web Caching?) (2013), http://www.silicon-press.com/briefs/brief.webcaching/brief.pdf (accessed November 18, 2013)
4. Varnish Varnish Book (2012), https://www.varnish-software.com/static/book/ (accessed November 18, 2013)