11. Alégroth, E., Feldt, R., Ryrholm, L. Visual Gui testing in practice: Challenges, проблемиі обмеження. Empirical Software Engineering. 2015. Vol.20. pp. 694-744.
12. Upton E., Halfacree G. Raspberry Pi user guide. John Wiley & Sons, 2016.

DOI https://doi.org/10.30525/978-9934-26-388-0-3

SERVER CONFIGURATION SOFTWARE OPTIMIZATION FOR IMPROVING WEB APPLICATION PERFORMANCE

ПРОГРАМНА ОПТИМІЗАЦІЯ КОНФІГУРАЦІЇ СЕРВЕРА ДЛЯ ПОКРАЩЕННЯ ПРОДУКТИВНОСТІ РОБОТИ ВЕБДОДАТКУ

Oleshchenko L. M.
Candidate of Technical Sciences, Associate Professor at the Computer Systems Software Department, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute" Kyiv, Ukraine

Олещенко Л. М.
кандидат технічних наук, доцент кафедри програмного забезпечення комп'ютерних систем, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського» м. Київ, Україна

Burchak P. V.
Postgraduate Student at the Department of Computer Systems Software, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute" Kyiv, Ukraine

Бурчак П. В.
аспірант кафедри програмного забезпечення комп'ютерних систем, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського» м. Київ, Україна

Optimizing the server configuration can significantly improve the performance of the web application are developing [1] Maximum productivity web application may be reached by combinations different methods settings web server and others components [2]. CDN network (Content Delivery Network) allows distribute static resource on to everything world, reducing time download for users. Importantly also to provide settings caching for CDNs. Asynchronous loading of resources (images, styles and scripts) avoids blocking the general flow of loading a web page. Caching on different such levels as client's cache, cache on levels proxy server and cache on levels

application provides maximally efficiency using web application. Static caching provides storage of static resources (images, styles, scripts) on the client side or on the server. Dynamic caching is used for data that does not change frequently, on the server or in distributed cache systems. Minimization and compression static resources, the use of sprites is used for merging images and reduction quantity requests. HTTP/2 supports parallel requests and header compression to reduce page load times.

To optimize database, it's a good idea to use indexes for fast data retrieval, optimize SQL queries, and use query result caching where possible. Consider an example of configuring MySQL query caching:

query_cache_type = 1; # we specify the type of query caching , in this case a value of 1 means query caching is enabled, a value of 0 disables caching, and a value of 2 means caching that does not cache queries that have SQL_NO_CACHE in their statement.

query_cache_limit = 2M; # this is a setting to limit the size of query results that can be cached, in this case, the limit is 2 megabytes (2M), if the query result exceeds this limit, it will not be cached.

query_cache_size = 64M; # defines the amount of memory that can be used to store cached query results, in this case the size is 64 megabytes (64M), if the cache exceeds this limit, older results will be deleted to make room for new ones.

The options discussed above are used to control query caching in MySQL, which can improve performance if configured correctly for a specific web application and database usage. Caching efficiency may depend on the nature of requests and the amount of data. Choosing the optimal server hardware for web server is a key step. The server should be configured to limit the number of simultaneous connections and use Keep-Alive to reduce the time it takes to establish new connections. Optimizing worker process pool settings includes adjusting the number and distribution of worker processes for optimal resource utilization. Monitoring tools like New Relic, Datadog or others, optimization should be an ongoing process as traffic volumes and requirements may change.

Optimizing Nginx involves reducing the number of simultaneous connections (concurrent connections):

```
events {
worker_connections 1024; # this value can be increased as needed
}
```

Caching configuration for Nginx looks like this:

```
proxy_cache_path / path / to / cache levels =1:2 keys_zone =my_cache:10m max_size =10g inactive =60m use_temp_path = off;
server {
location / {
```

proxy_cache my_cache; # specifies which cache zone use the rules for this (in this case, my_cache).

proxy_cache_valid 200 302 5m; # determines that responses with status codes 200 and 302 will be stored in the cache within 5 minutes.

proxy_cache_valid 404 1m; # determines that there will be responses with a status code of 404 (not found), also be stored in the cache within 1 minute.

proxy_pass http://backend; # indicates the address to which they will be sent redirected requests after processing by the Nginx web server.

} }

This code is part of configurations web server Nginx defines settings caching for proxy requests: proxy_cache_path specifies the path to the directory where it will be stored cache.

levels =1:2 indicates the internal structure of the cache directory.

keys_zone =my_cache:10m sets name zones cache and its size (in this case, 10 megabytes).

max_size =10g sets maximum size cache directory (in this case, 10 gigabytes).

inactive =60m indicates the idle time for determination inactive elements cache (60 minutes).

use_temp_path = off turns off using temporary directory for storage temporary files.

server defines a configuration block for a specific virtual server.

location / indicates that _ this rule applies to all URLs that begin with /.

Settings buffers memory:

key_buffer_size = 256M; # this parameter for MyISAM, which indicates size buffer remember that is used for caching indices and intermediate results, in this case size buffer is 256 megabytes (256M).

innodb_buffer_pool_size = 512M; # is an option for InnoDB that specifies the size of the memory buffer used to cache InnoDB data and indexes, in this case, the size of the buffer is 512 megabytes (512M).

MySQL configuration and defines memory buffers for two different types of data stores in MySQL: MyISAM (which uses key_buffer_size) and InnoDB (which uses innodb_buffer_pool_size).

Both considered parameters are very important for MySQL performance: key_buffer_size is important for optimizing MyISAM tables, and a large buffer can improve performance, especially for large read operations; innodb_buffer_pool_size is important for optimizing InnoDB tables. A memory buffer is used to cache blocks of data and indexes, which can significantly improve performance for read and write operations. Large values of these parameters can improve system performance, provided the server has enough memory to support them. Setting the optimal parameters for a specific amount of data and load:

innodb_flush_log_at_trx_commit = 2; # this option determines how often InnoDB transaction log entries will be synced to disk. A value of 2 means that transaction log entries are not synchronized every time after each transaction, but only once per second. This allows to reduce disk load and improve performance, provided we are willing to accept some risk of data loss in the event of a system failure.

innodb_log_file_size = 256M; # this parameter specifies the size of the InnoDB transaction log file. A larger file size allows for more transaction history, but also requires more resources. In this case, the log file size is set to 256 megabytes (256M), which may be efficient for many applications.

innodb_file_per_table = 1; # this parameter specifies whether each table has its own data and index file, or whether they should be stored in a single shared file. A value of 1 means that a separate file will be created for each table, which can make database optimization and management easier, especially if we have many tables. These settings can be useful for improving performance and optimizing transactions in an InnoDB environment.

In conclusion, modern problems of web server software optimization include the need to effectively cope with high loads and peak data volumes, ensure response speed to improve user experience, consider security aspects to protect against various types of attacks, optimize resource-intensive operations and effectively use caching.

The issues of scalability, integration with CDN, optimization of work with the database and improvement of support for the latest technologies also remain important. The process of finding solutions for these challenges requires constant improvement and implementation of the latest technologies to ensure the efficiency and reliability of the web server in today's Internet environment.

According to conducted research, improving the performance of a web application through the optimization of the web server can lead to a significant enhancement in efficiency, and in some cases, it can exceed 20–30%.

Bibliography:

1. Kuipers J., Ueda T. and Vermaseren J.A.M. J. Code optimization in FORM. ComputerPhysics Communications. 2015. 189, 6, 1-19.

2. Zengyu Cai, Jingxiao, Li Jianwei Zhang, Jianwei Zhang. Research on Performance Optimization of Web Application System based on JAVA EE. Journal of Physics Conference Series. 2020. 1437(1): 012039. DOI: 10.1088/1742-6596/1437/1/012039.