A. Gudkov

ASYNCHRONOUS APPROACH TO THE DEVELOPMENT OF MODERN BACK-END APPLICATIONS

This article explores the recent trend towards asynchronous approaches of Back-End applications development, highlighting the numerous advantages they offer over traditional synchronous models. Also explores why NGINX web servers became more preferred over Apache, and the advantages of asynchronous database queries.

Introduction

Nowadays the world of web development is constantly evolving. From the begging of 21-th century the amount of people using Web services have been growing exponentially which led to the adaptation of services to such kinds of loads. And one of the most recent trends in this sphere is about asynchronous techniques of Back-End applications, which includes asynchronous server modeling and handling of asynchronous database operations.

I. Asynchronous server model

For better understanding let's begin from classic synchronous server model (blocking I/O model). In this model, the server creates a new thread for each incoming request which is responsible for handling this request and sending the response back to the client. While the thread is processing the request, it awaits, it means that the thread is blocked and can't handle any other requests until the I/O operation for the current request is completed (figure 1).

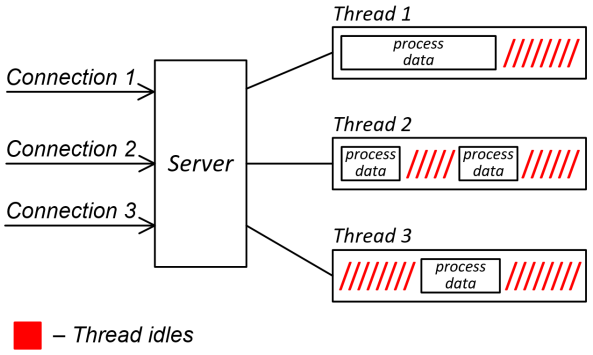

Fig. 1 Synchronousserver model

However, it can be inefficient in handling large numbers of concurrent requests because creating and managing a large number of threads requires a significant amount of system resources. The rapid growth of incoming requests can lead to the "thread hunger", the situation, when all threads fulfilled the memory they use and the only thing the processor can do is just switching between threads without allowing them to perform any tasks. There also can be a situation when thread resources were allocated by OS, but it idles. Sometimes "deadlocks"can occur – when multiple threads refer to the same memory or storage resources.

Based on this model, the most popular Apache web server was developed in 1995 [2]. For about 20 years it coped with his task perfectly as during that time amount of web-sites clients was not so big. But the time passed, the number of users has grown significantly and Apache faced problems which led to the creation of a new server model.

Asynchronous server model (non-blocking I/O) allows the server not to wait for a client request to complete and moves to the next operation, which allows multiple requests to be processed simultaneously even in one main thread.

To achieve non-blocking I/O, asynchronous servers use an event loop to handle incoming requests [4]. The event loop continuously checks for new requests and processes them using callback functions. When a request requires I/O operations, the server delegates the task to a separate worker (another thread, process, external service), allowing the event loop to continue processing other requests and not to block the main thread. Once the I/O operation is complete, it invokes the appropriate callback function to handle the request result (figure 2).

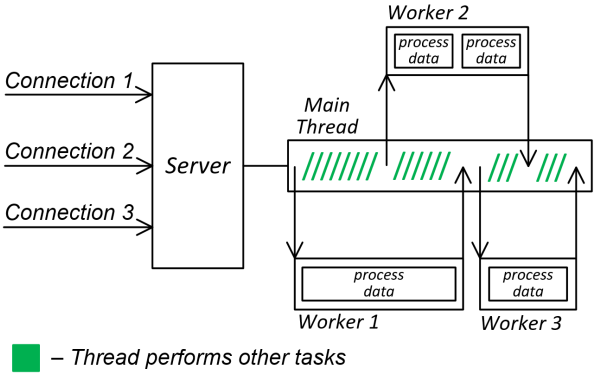

Fig. 2 Asynchronousserver model

By using asynchronous model applications can handle large numbers of concurrent connections with low resource consumption, reducing latency and improving response times. This is especially important for real-time Web applications, such as online gaming platforms, financial trading systems, and social media networks, which should always be very responsive to user and where delays can be a major issue.

One disadvantage of asynchronous servers is that they can be more difficult to program and

debug than synchronous servers. Asynchronous code can be complex and difficult to understand, and the use of callback functions can lead to nested and convoluted code.

Based on this model NGINX web server was developed [1]. It also offers superior performance for static content. This is due to its use of a highly optimized HTTP engine, which is specifically designed for serving static files. Additionally, NGINX has a smaller memory footprint than Apache, making it ideal for use in resource-constrained environments (figure 3).

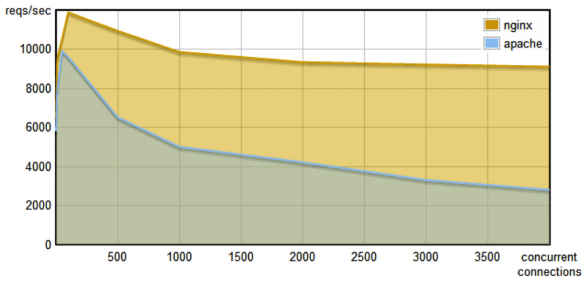

Fig. 3 Amount of available requests/second per concurrent connections count

As NGINX was developed based on modern web requirements it includes a set of features and modules such as load balancing, caching, SSL encryption, proxying, etc. However, Apache is still a good choice for servers with a small number of incoming requests.

II. Asynchronous database operations

The idea of asynchronous database operations is similar to asynchronous server model – you get more performance by concurrency. A single database connection can't execute instructions both synchronously and others asynchronously. You need to specify the SQL connection executing mode in your data source [3].

In asynchronous execution mode, you instruct the database engine to perform an operation, then the database engine works in the background while the application keeps running. When the operation finishes the database engine dispatches an event to alert you to that fact. The key benefit of asynchronous execution is that you release the business logics main thread while the database is doing its job (in the background).

This is especially valuable when the operation takes a notable amount of time to run. On the other hand, in synchronous execution mode operations don't run SQL instructions in the background. You tell the database engine to perform an operation and the code pauses at that point. When the operation completes, execution continues with the next line of your code.

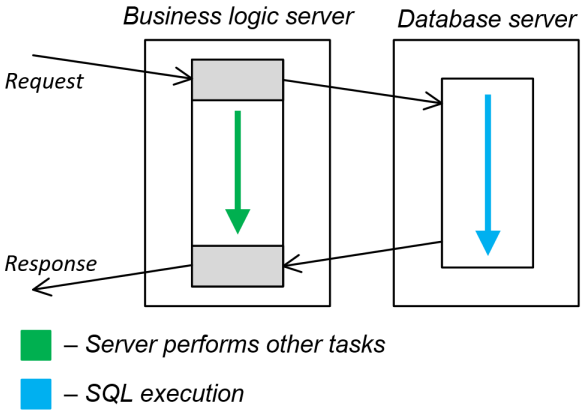

Fig. 4 Process of asynchronous database operation

III. Conclusion

Asynchronous server models and asynchronous database operations are becoming increasingly important for the development of high-performance, scalable, and fault-tolerant modern Back-End applications. They are particularly useful in real-time Web applications, such as online gaming platforms, financial trading systems, and social media networks, where those parameters play significant role. As the demands of modern systems continue to grow, it is likely that we will see even more widespread adoption of these approaches in the coming years.

1. DeJonghe D. The Complete NGINX Cookbook: Advanced Recipes for High-Performance Load Balancing / D. DeJonghe. O'Reilly Media Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, USA, 2019. – 181 p.
2. Laurie B. Apache: The Definitive Guide / B. Laurie, P. Laurie. – O'Reilly Media Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, USA, 2002. – 536 p.
3. Molinaro A. SQL Cookbook: Query Solutions and Techniques for All SQL Users / A. Molinaro, R. de Graaf. – O'Reilly Media Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, USA, 2020. – 567 p.
4. Parker D. JavaScript with Promises: Managing Asynchronous Code / D. Parker. – O'Reilly Media Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, USA, 2015. – 92 p.

Gudkov Alexey, 3-rd grade student in the Faculty of Information Technology and Management of BSUIR, gudkov.bsuir@gmail.com.

Scientific supervisor: Trofimovich Alexey, Senior Lecturer in the Faculty of Information Technology and Management of BSUIR, trofimaf@bsuir.by.