Multiprocessing as a Way to Optimize Queries

Nataliia KHRYSTYNETS, Kateryna MELNYK, Svitlana LAVRENCHUK,
Oksana MISKEVYCH, Serhii KOSTIUCHKO
Lutsk National Technical University, Lutsk 43017, Ukraine
ORCiD ID: Nataliia KHRYSTYNETS https://orcid.org/0000-0002-4836-7632
Kateryna MELNYK https://orcid.org/0000-0002-9991-582X
Svitlana LAVRENCHUK https://orcid.org/0000-0002-5453-3924
Oksana MISKEVYCH https://orcid.org/0000-0002-5009-2391
Serhii KOSTIUCHKO https://orcid.org/0000-0002-1262-6268

Abstract. Developing an effective web application involves the use of various methods and techniques to ensure fast and efficient processing of requests. Sometimes it is not possible to solve the problem of multiprocessing with a single tool, such as a programming language or framework. This work investigates the use of asynchronous methods of processing requests using queues. Job operation in background and non-background modes relative to the main web process is studied. Analytics are provided to analyze a web application with 13,000 requests to process daily. It is proposed to optimize the processing by using the Laravel framework and the Python server dual-tasking using the Supervisor tool on Linux, as well as using a task scheduler for each task. The paper presents positive findings about this algorithm, which contributes to the efficiency of web development and provides a great user experience on the website. Fast processing of web application requests can be a valuable competitive advantage for a business or organization. Research in this field helps to maintain their high competitiveness. In addition, the study of query processing speed is important in scientific research, as it contributes to the development of new algorithms, optimization methods and technologies.

Keywords. Multiprocessing; web-application; query; requests; data processing; server; Laravel

1. Introduction

Web development projects typically involve integration with a large number of external services, platforms, and systems in today's digital world. The operation of web applications in practice is often accompanied by data parsing to obtain, analyze, and process information [1-3]. Commercial websites, news portals, social media platforms, blogs, forums, transportation websites, and more – are just a few examples of areas where the processing of a large number of requests needs to be taken into account.

There are general approaches to implement this task, such as web scraping and using HTTP clients [4]. However, these mechanisms are often not universal. The most famous method, in our opinion, is the ARI public method for executing queries, but excessive requests can lead to IP address blocking or other limitations. Development of web applications [5-8], such as online stores, may require handling a large number of server requests.

This can pose challenges to the performance and scalability of the developed application [9-11]. It is important to utilize effective techniques and tools to optimize server interactions and ensure speed and productivity during development in such situations.

One of the main goals of the research is to improve the quality of user service. A quick response to requests will increase the satisfaction of customers and users of the web resource. Reducing the processing time of requests and the response of the server can help improve the productivity of the workflow and the business in general.

2. Literature review

From a review of literary sources, many ways of improving the operation of web systems are known. In work [12], the authors investigated that "to increase the efficiency of a web server by using a load balancing system to manage a large number of requests," six servers and three load balancing algorithms were proposed. The authors achieved a high response speed and stability of the response time, but the implementation of this method requires expensive equipment - additional servers. In [13], a study was proposed to automate the customer support for inquiries of a business company that currently handles customer support requests manually. Research was conducted using traditional machine learning approaches. These studies are important, they use parameters for each learning algorithm and data set in terms of the query sampling distribution. However, in our opinion, only framework tools can be used for effective traffic of small companies. Asynchronous query optimization methods are explored in [14], but it is not clear whether people can "use asynchrony when updating data, because the user interface is dynamically updated and changes can be difficult to interpret."

From the review of the literature, it can be concluded that the proposed research of the authors is valid, especially in terms of the use of asynchronous methods. In our opinion, it is possible to achieve a reduction in request processing time using multiprocessing tools of the Laravel framework. It fully meets the intended purpose.

3. Research methodology

Parsing research methods can be divided into three categories [15]. In the first, programming languages are used to facilitate the construction of data analysis systems. The second group uses artificial intelligence methods. It should be noted here that machine learning methods are quite time-consuming. The methods of the third category consist in the application of automated systems to detect patterns of query behavior, which include data segmentation and field extraction.

The methods of the first group were used in the research work.

Often, when creating high-capacity web applications, you need to make a large number of requests to a third-party website for monitoring. To do this, you can use different methods of integration with the website, depending on the available tools. For parsing a large number of products, it is recommended to use asynchronous processing of background tasks in Laravel. This will allow efficient processing of a large amount of data without blocking the main flow of web requests.

Technologies and tools for creating web applications were used to achieve the goal. Initially, Laravel tools were used to support multiprocessing and configuration file settings, and PHP language was used to design database library elements. Further editing of the web application scripts was performed using the Python language. The selected tools are free software, so they are universal and accessible to developers from all over the world. The Supervisor toolkit on Linux was used to process and monitor background processes. Classical framework methods and Google Analytics were used for data collection and analysis.

According to this methodology, it is possible to carry out a complex implementation of parsing of web resources and thus get rid of the problem of lengthy processing of requests. First paragraph.

4. Multiprocessing and methods of implementation in web development

Asynchronicity is one way to optimize the handling of multiple database queries in web development. The use of the Queues in the Laravel framework allows efficient management of executing heavy or long operations in the background. This improves the performance of the web application and enhances the user experience. Queues in Laravel operate based on the «producer-consumer» concept. This allows adding job to the queue and then asynchronously executing them using workers. Workers check the queue for pending tasks and perform them in the background. The tasks are implemented following the scheme (Fig. 1).

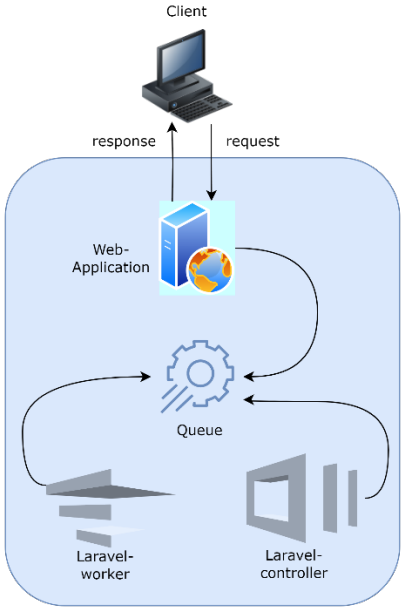
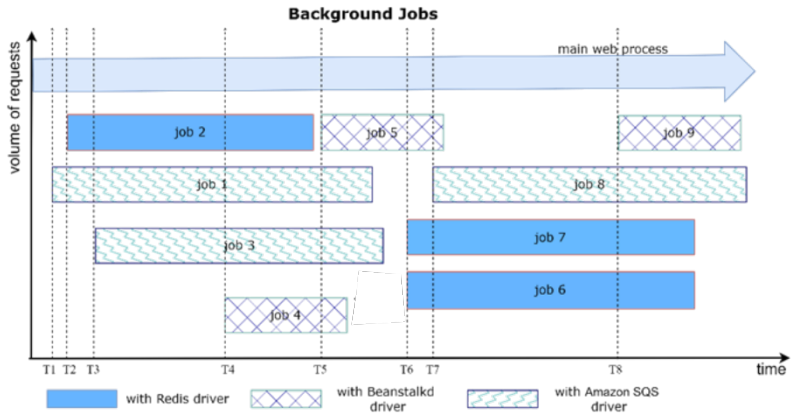


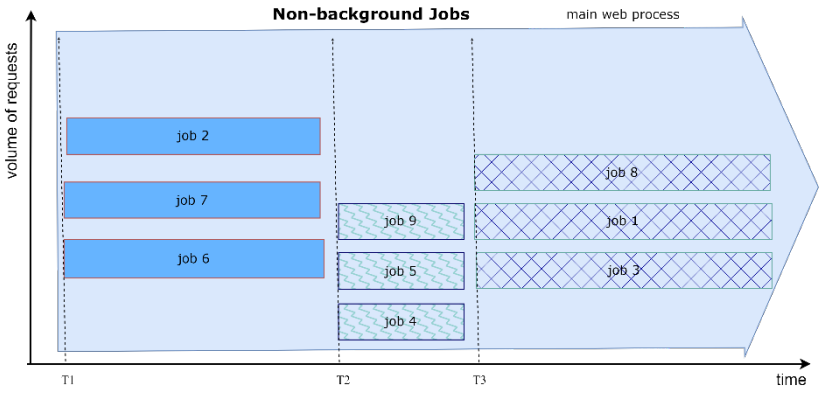
Figure 1. Using the Queues in the Laravel framework

This scheme will be used to solve the problem of multi-threaded processing of requests. For this, the Laravel framework supports asynchronous multiprocessing

through the use of functional built-in mechanisms. However, the PHP development language itself is not multi-threaded. Therefore, to manage the queue of tasks in the database, we will use the Jobs table, which stores information about tasks awaiting execution [16]. For the efficiency of their use and communication, we will use the mechanisms connects. It should be noted here that the Jobs implementation mechanism is possible in the Background Jobs and Non-Background/Synchronous Jobs mode [17]. The difference between them is schematically presented in Fig. 2. In the background mode (Fig. 2a) tasks are performed asynchronously and run separately from the main web process, with the support of various drivers. In non-background mode (Fig. 2b), tasks are performed synchronously, which means that they are performed directly in the context of a web request.



a)



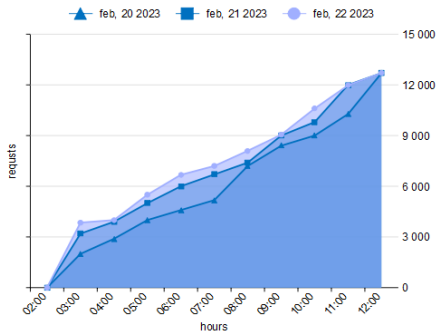
b)

Figure 2. Temporary implementation of requests in: a) background and b) synchronous mode

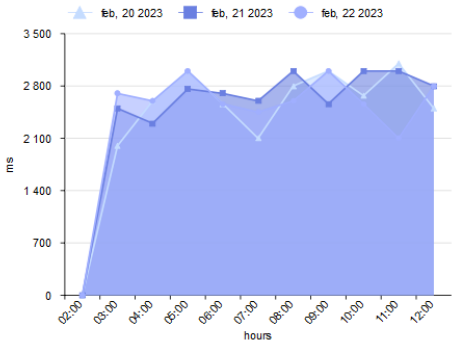
All data that was passed to the constructor of this class is in JSON format [18]. Fields that allow you to store and track information about tasks waiting to be executed in the database queue are important for performing further tasks of multiprocessor processing. Among them are the «payload» table field, which contains serialized data,

and the «queue» field – a queue field that distinguishes each job. The default value of the «queue» field for a task in the queue is specified in the application settings in Laravel. When «queue» is set to «sync», tasks in the queue are executed synchronously. If the process is asynchronous, we can configure these queues.

Parsing of a well-known online store with tens of thousands of products was provided, and these products were periodically updated and changed. Background tasks were running to update the goods, the update process was started every night at 02:00. Initially, the application processed only one job for each time period, and in this mode, updating 13,000 products (since such many of requests were planned for daily processing) took too long. Analytical data of request processing was analyzed, average indicators were calculated according to tabular data and, as a result of calculations, the graphic representation of traffic has the following form:



a)



b)

Figure 3. Analysis of request processing in the period from February 20 to February 23, 2023: a) processing time; b) average request time

As can be seen from Fig. 3 a), about 13,000 requests were processed in three days, and the time of complete processing was about 10 hours. The duration of the request (Fig. 3b) is from 2 to 3 seconds. Such a process is quite slow and needs improvement for the efficiency of the developed application.

The problem of such lengthy processing was identified. Note that the problem of such a plan often occurs when organizing multiprocessing. A site with a large number of products is a powerful application. It was possible to use traffic sniffers to monitor network traffic that passes through the web application interface. Then it would be possible to access the data packets sent over the network and analyze their content and make decisions about optimizing their work [19]. Different frameworks allow you to solve this problem with your own methods. Solving the issue is possible through the use of a Python library as an implementation of the Laravel framework for developing web applications that interact with scripts written in the Python programming language. Python is not the fastest programming language, and PHP, as a language specialized in web development, has some advantages in the speed of executing web queries and interacting with databases. Therefore, for an ideal solution, it was worth rewriting the site library with a huge number of products in PHP. Or, in the case of not fully using the functionality, such as only updating prices, availability of goods and some categories, solve the issue by contacting the IP of your own development using the Job class (Fig. 4)

```php
namespace App\Jobs;

use App\Models\Product;
use Illuminate\Bus\Queueable;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Foundation\Bus\Dispatchable;
use Illuminate\Queue\InteractsWithQueue;
use Illuminate\Queue\SerializesModels;

class UpdateProductAvailabilityJob implements ShouldQueue  2 usages
{
    use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;

    public $tries = 1;  no usages
    public $timeout = 3600;

    private const LIMIT = 1000;  2 usages


    public function handle()
    {
        $offset = 0;

        $productIds = $this->getProductIds($offset);

        while(count($productIds)) {
            foreach ($productIds as $productId) {
                ProductAvailabilityJob::dispatch($productId)->onQueue('update-one-product');
            }

            $offset += self::LIMIT;

            $productIds = $this->getProductIds($offset);
        }
    }

    private function getProductIds($offset): array  2 usages
    {
        return Product::where( column: 'sync_at',  operator: '<', date( format: 'Y-m-d'))
            ->whereNotNull( columns: 'sync_at')
            ->offset($offset)
            ->limit(self::LIMIT)
            ->pluck('id')
            ->toArray();
    }
}
```

Figure 4. A code snippet of a program to make a request to your own IP address using the Job class

Let's move on to solving the issue of multiprocessing and optimization of the web application developed on Laravel [20]. With the help of this framework, it is possible to make settings so that in the Config directory, where the configuration file is located, a

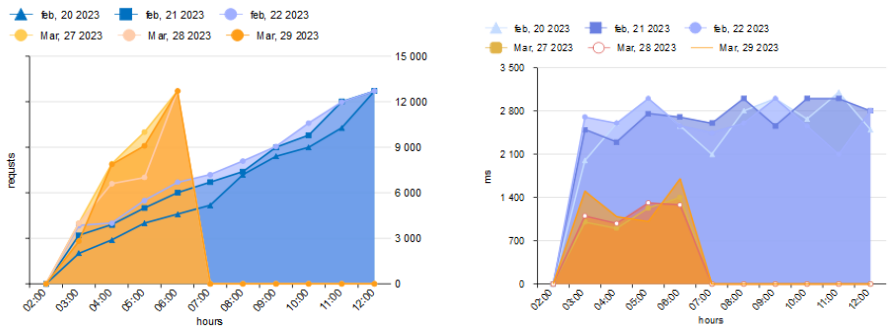
separate configuration file database_connect.php was specified, where the settings for a new connection will be specified in this file, so that the job is executed on a specific connection.

The work uses the popular Supervisor tool for managing background processes on a server running the Linux OS. It allows us to monitor the state of processes, monitor their execution, automatically restart them in case of a crash. The Supervisor is a powerful tool for ensuring the stability and continuity of background processes on the server, in particular for managing the background tasks of web applications that use Laravel or other frameworks.

The Supervisor's role in resolving the issue is as follows: When the Supervisor starts a queue process, it caches the code and settings used to process the queue. When we change the code or setting for a particular queue, we must restart the corresponding queue process in Supervisor for the changes to take effect. This is because the Supervisor is responsible for managing background processes, including queue handling. If we do not restart the queues, then processing will be performed with old data, and this will be incorrect relative to the expected results. Note that restart is a Supervisor setting .

Going back to multiprocessing, the following was done to speed up query processing. When processing the queue, the simultaneous execution of five jobs is planned. This means that in the Non-Background mode (which is schematically presented earlier in Fig. 2b), the parallel execution of several jobs works simultaneously, which helps to reduce the execution time of the entire queue.

As a result of such planning, the update time was reduced from 10 to 4 hours, which is a significant positive indicator. Now, when the process was started at 2 in the morning, it was completed by 6 in the morning (Fig. 5a).



a) b)

Figure 5. Comparative analysis of request processing in the period 27-29 March 2023: a) processing time; b) average request time

Reducing the update time has its advantages, especially for an online store where the timeliness of information is important. At the same time, the average request time was significantly reduced (Fig. 5b): from 1900-3000 ms to 800-1450 ms. To execute the update command in the online store, the following software structure was used in the Laravel console (Fig. 6):

```php
class ProductAvailabilityJob implements ShouldQueue  1 usage
{
    use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;

    public $tries = 1;  no usages
    public $timeout = 60 * 15;
    public int $productId;
    public Product $product;

    public function __construct(int $productId)  no usages
    {
        $this->productId = $productId;
    }

    public function handle()
    {
        $this->product = Product::find($this->productId);

        $code = trim($this->product->code);

        if (Str::is('*-*', $this->product->code)) {
            $code = substr($code,  offset: 0, strpos($code,  needle: "-"));
        }

        if (strlen($code) < 5) {
            return;
        }

        $availability = app(ProductAvailability::class)->getAvailability($code);

        if (is_null($availability)) {
            $this->product->update([
                'quantity' => 0,
                'sync_at' => date( format: 'Y-m-d H:i:s')
            ]);
            return;
        }

        $this->product->update([
            'price' => $availability['price'],
            'quantity' => $availability['quantity'],
            'sync_at' => date( format: 'Y-m-d H:i:s')
        ]);
    }
}
```

Figure 6. Job code that is dispatched to the queue

Laravel can execute its commands in the console using a built-in engine (Artisan). This command is implemented programmatically as follows. Laravel's built-in task scheduler mechanism in the developed project automatically starts it for execution every day at a fixed time, at 02:00. The planned and implemented mechanism is flexible, and the logic of the command can be adjusted according to the requirements and the type of

job to be called. Therefore, planning organized in this way allows you to optimize the operation of the application according to your own needs.

5. Conclusion

This article solves the problem of long data processing in the process of parsing a web application. The initial data of processing 40 thousand requests with a request duration of 2 to 3 seconds showed an unsatisfactory result of contacting the server - the request processing time lasted for 10 hours. This indicated a rather slow process that needed improvement. As a result of the study, multiprocessing was applied to the original data set using the proposed methodology using the specified tools.

Six experiments were conducted during February-March 2023. They were defined as follows: the original data set in the form of requests to the server was launched at a fixed time of 02:00. For these experiments, the data was pre-processed by the Supervisor using the data caching method. As a result of the programmed elements of the system, five jobs simultaneously participated in the processing of requests. This significantly reduced the processing time of requests from 10 to 4 hours.

The reliability and accuracy of the results are ensured by the analytical data of the framework and the Google Analytics audit. That is, the proposed method is actually confirmed by a series of six experiments, which allows you to see the stability and effectiveness of multiprocessing as a result of web application server parsing.

The results of the study are a significant acceleration of the processing of requests. Improvements in long-running query processing can have practical applications in real-world scenarios, such as computing systems, databases, Internet searches, and medical research. The practical significance of the conducted experiments, the popularization of research and the publication of its results will contribute to the improvement of the field and increase the level of knowledge, encouraging other scientists to further research and development.

References

[1] Amini, Mahyar, et al. Mahamgostar.com as a case study for adoption of laravel framework as the best programming tools for php based web development for small and medium enterprises. Journal of Innovation & Knowledge, ISSN (2021): 100-110.

[2] Laaziri, M., Benmoussa, K., Khoulji, S., & Kerkeb, ML (2019). A comparative study of PHP frameworks performance. Procedia Manufacturing, 32, 864-871.

[3] Sunardi, Andri, et al. MVC architecture: a comparative study between Laravel framework and Slim framework in freelancer project monitoring system web based. Procedia Computer Science, 2019, 157: 134-141.

[4] Yamao, Yasuo, et al. Protocol to acquire time series data on adverse reactions following vaccination using a smartphone or web-based platform. STAR protocols, 2023, 4.2: 102284.

[5] Mourato, Sandra, et al. An interactive Web-GIS fluvial flood forecast and alert system in operation in Portugal. International Journal of Disaster Risk Reduction, 2021, 58: 102201.

[6] Permana, Putu Adi Guna; Triandini, Evi. Performance with Eloquent and Query Builder in Crowdfunding System with Laravel Framework. 2021.

[7] Rojas, Hesmeralda ; ARIAS, Kevin A.; Renteria, Ronald. Service-oriented architecture design for small and medium enterprises with infrastructure and cost optimization. Procedia Computer Science, 2021, 179: 488-497.

[8] Bottani, Eleonora, et al. Wearable and interactive mixed reality solutions for fault diagnosis and assistance in manufacturing systems: Implementation and testing in an aseptic bottling line. Computers in Industry, 2021, 128: 103429.
[9] Pulmano, Christian E., et al. MIND Plug-in: Development of an Electronic Medical Record based data collection tool for research in autism and neurodevelopmental disorders. Procedia Computer Science, 2019, 164: 646-653.
[10] Antunes, Mário; Maximiano, Marisa; GOMEZ, Ricardo. A Customizable Web Platform to Manage Standards Compliance of Information Security and Cybersecurity Auditing. Procedia Computer Science, 2022, 196: 36-43.
[11] Fawzy, Ahmed Hany; Wassif, Khaled; MOUSSA, Hanan. Framework for automatic detection of anomalies in DevOps. Journal of King Saud University-Computer and Information Sciences, 2023, 35.3: 8-19.
[12] Chyrvon, Andrii & Lisovskyi, Kostiantyn & Kyryndas, Nikita. (2023). The Main Methods of load Balancing on the Nginx Web Server. 10.36074/logos-26.05.2023.040.
[13] Arias-Barahona, María & Arteaga Arteaga, Harold & Orozco Arias, Simon & Flórez-Ruíz, Juan & Valencia-Díaz, Mario & Tabares Soto, Reinel. (2023). Requests classification in the customer service area for software companies using machine learning and natural language processing. PeerJ Computer Science. 9. e1016. 10.7717/peerj-cs.1016.
[14] Sahu, Divya Rishi; Tomar, Deepak Singh. DNS pharming through PHP injection: Attack scenario and investigation. International Journal of Computer Network and Information Security, 2015, 7.4: 21-28.
[15] Raza, Arslan & Habib, Asad & Ashraf, Jawad & Javed, Muhammad. (2017). A Review on Urdu Language Parsing. International Journal of Advanced Computer Science and Applications. 8. 10.14569/IJACSA.2017.080413.
[16] Parreira, Vinícius da Silva Coutinho, et al. ExVe: the knowledge base of orthologous proteins identified in fungal extracellular vesicles. Computational and Structural Biotechnology Journal, 2021, 19: 2286-2296.
[17] Sonmez, Rifat; Ahmadisheykhsarmast, Salar; Güngör, Aslı Akçamete. BIM integrated smart contract for construction project progress payment administration. Automation in Construction, 2022, 139: 104294.
[18] Fotakis, Emmanouil A., et al. VectorMap -GR: A local scale operational management tool for entomological monitoring, to support vector control activities in Greece and the Mediterranean Basin. Current Research in Parasitology & Vector-Borne Diseases, 2021, 1: 100053.
[19] ROY, Sujit; Kabir, Md Humaun; Ahmed, Md Tofail. Design and Implementation of Web-based Smart Class Routine Management System for Educational Institutes. 2022.
[20] L i Dalu, L i Rui, S un Yuanzhang. Data compatibility analysis of WAMS/SCADA hybrid measurements state estimation[J]. Proceedings of the CSEE, 2010, 30(16): 60-66.