Data distribution system: preparation of server stations data

E O Vikulov, O V Denisov and L A Denisova

Omsk State Technical University, 11, Mira ave., Omsk, 644050, Russia
e-mail: vikuloveo@gmail.com, denisova@asoju.com

Abstract. In this article the issue of server stations status data processing for effective distribution of client data on the server stations in high load information systems is discussed. We have analyzed the existing methods for distributing data on server stations of high-loaded web applications, as well as solutions that improve the speed of data delivery to the user. Among the considered methods, we identified a prototype for development and proposed an approach to the design of data distribution systems. We have developed a model that describes the state of the servers, and proposed a structure for representing the parameters of the system. According to the obtained experimental data we have constructed graphs that reflect the dependence of server state parameters on time. In the future, we are planning to use this data about server states, as well as the data about end users (uploading or browsing files), in order to optimize server selection and ensure fast provision of information to the user.

1. Introduction

One of the actual problems of efficient use of computing resources in high-load information systems is to ensure the rapid provision of information to the user by means of the Internet. To increase the speed of data delivery to the user, the following technologies are used: data delivery and distribution networks, data balancing (distribution) systems.

Data delivery and distribution networks (CDN) [1] consist of geographically distributed multifunctional platforms. The interaction of these platforms allows CDN to process data efficiently. This allows CDN to speed up the user's requests for data.

Load balancing systems [2] are technologies based on the data distribution through the server stations. This technology is one of the most successful and is often used in high-load web applications. There are a few algorithms of load balancing through the server stations e.g. DNS Round-robin [3], weighted Round-robin, least and weighted-least connection, agent-based adaptive load balancing, and etc. [3].

Most simple popular algorithm of load balancing is Round-robin. This algorithm allows to system normally distribute tasks between server stations. It helps to optimize resources and decreases processing time of each request [3, 4].

Both methods have drawbacks. The use of data delivery networks significantly increases the cost of the developed application. The cost increases in direct proportion to an increase in the amount of data. When using the same data balancing systems, the load is distributed between the server stations, which allows us to increase the speed, but does not give a significant increase, as it is often performed by the Round-robin distribution algorithm. Also, the consistent distribution of data does not take into account the information about the data itself, for example, the type of data, the amount of data, and the information about the user who downloads this data. This information helps to determine on which server station balancer should distribute the user file in order to speed up the further operation of the web application. To take into account the current state of the system, you must use adaptive balancing based on the agent. The agent is a software tool that collects and processes data about the current state of the server stations. This type of balancing will allow system to take into account the current state of used server stations and the information about the clients of the system.

2. The problem statement
It is required to develop a data distribution system for server stations, one of the main functions of which is the processing of information about the status of server stations. In addition, the system should implement the following functions: downloading, viewing and issuing data that users work with (images, text, video, etc.). Let's construct the structural scheme of the developed system. This scheme allows us to understand the interaction of users with the system, as well as visualizes the data flows within the application. Here we can see: distributed server stations, application balancer, users who download (on the one side) and scan (on the other side) files using the application (figure 1). The balancer plays a major role in the data transfer chain from the user who has uploaded them to the user downloading or viewing the file. Thus, when downloaded data by the user, the server-balancer must select one of the server stations most suitable for the downloaded data [5, 6]. As a selection criterion, we can use different indicators.

So, the server-balancer can select the required server station based on the information about the downloaded data (such as type and size) and about the user (such as location, system usage time, previously downloaded files). In order to justify the selection of the quality criterion for the data balancing, it is necessary to carry out the series of experiments confirming the significance of the chosen criterion. As the most simple and affordable, we consider the criterion of the distance between user, downloading files and the server station from the array of servers. Thus, we need to develop a data distribution system for server stations, the tasks of that will be collecting and processing of server stations status data.

3. The development of data distribution system

3.1. System operation algorithm

Let's consider how the developed system should function and imagine the algorithm for data distribution on server stations. At the beginning, the user selects the data that he needs to transfer. After that, the client application sends data to the balancer, which determines the most suitable server for receiving data. Server determination based on server stations status data analysis and user data analysis. After making the decision, the balancer sends the user uploaded data to the selected server and waits for a response. After loading the data, the server responds that the download is successful. The received response is processed and displayed to the user. The block diagram of the algorithm is shown in figure 2.

![Figure 1. Structural diagram of the system being developed](image-url)
Let's consider the operation of the balance server in accordance with blocks 4 and 5 of the algorithm: data collection and data preparation for server selection.

3.2. Preparation data for server selection

Let’s consider how we should process the data to make it easy to select a server. We will form an array of data on the status of server stations. Based on this data, the server balancer should make a conclusion about which of the server stations is suitable to transfer of the downloaded file. The data obtained are represented in the form of a multidimensional matrix whose number of columns is equal to the number used in the system parameters and the number of rows is equal to the number of requests sent to the load data. Also we take into account the number of requests for uploading files. To solve the problem of optimizing the choice of the server station, it is important to select the information that will allow the optimal distribution of data on the nodes of the system. The initial data for this problem can be represented in the form of array of size $l \times n \times m$.

$$X = \begin{bmatrix}
        x_{11}^l & x_{12}^l & \ldots & x_{1m}^l \\
        x_{21}^l & x_{22}^l & \ldots & x_{2m}^l \\
        \ldots & \ldots & \ldots & \ldots \\
        x_{n1}^l & x_{n2}^l & \ldots & x_{nm}^l \\
        \end{bmatrix},$$

$$X = \begin{bmatrix}
        x_{11}^q & x_{12}^q & \ldots & x_{1m}^q \\
        x_{21}^q & x_{22}^q & \ldots & x_{2m}^q \\
        \ldots & \ldots & \ldots & \ldots \\
        x_{n1}^q & x_{n2}^q & \ldots & x_{nm}^q \\
        \end{bmatrix}.$$  

(1)

superscript $l = \overline{1,q}$ is an number of file upload request, $q$ is amount of requests. Subscripts $i,j$ describe the states of the servers, $i = \overline{1,n}$ is an number of server stations, $n$ is amount of server stations, $j = \overline{1,m}$ is an number of operation parameters, $m$ is an amount of parameters. System operation parameters: $x_{li}^1$ is a hard drive usage of server station number $i$, $x_{ij}^l$ is a distance between a user and a server station number $i$, $x_{im}^l$ is an information about the uploaded file, file size, from a request number $l$. This data structure will allow us to apply clustering methods to break down subsequent requests to load data into groups and select an appropriate server station.

**Figure 2.** The block diagram of system operation algorithm
3.3. Server status data collecting procedure
In order to implement the data collection procedure, a special program was created. This program consists of three main components: agent, model, router. Let’s consider each of components. The agent is a subroutine that polls the server about its state. The model generates the data received from the agent in the required form, corresponding to the expression (1). Router provides data to server balancer. Code snippet of each component is shown in figure 3.

![Code snippets](image)

Each code snippet is structured to the best performance its function. As you can see, for the agent subroutine, we needed to organize nested calls to the query functions of the server parameters, the model was given a schema with the names of the database fields and their types, an http library for sending and receiving requests was attached to the router.

To develop a program we choose a programming language JavaScript Node.js [7]. Node.js is a runtime environment for executing JavaScript code on the server side. Node.js enables JavaScript to be used for server side scripting, and runs scripts server side to produce dynamic web page content before the page is sent to the user's web browser. Node.js adds to JavaScript ability to interact with input/output devices through the interface (developed on C++). Also Node.js allows us to include another external libraries, written on another programming languages. It provides them JavaScript abilities.

We made a research of different Node.js frameworks and chose ExpressJs. This framework is a modern high performance development tool.

As a database we used MongoDB [8]. This is one of the most developed open-source NoSQL solution. Also this database is a cross-platform, easy to use and provides high performance and availability.

4. The results of the experimental studies
For the experimental research, the developed program on the existing three server stations was installed. After the installation and launch of the program, data on the status of the server stations, in the processed form, was saved in the database of our system. Thus, we got a complete history of the status of each server station after the installation of the program. Figure 4 shows an example of displaying saved parameters (we chose such parameters as the percentage of hard drive usage and the percentage of the central process unit (CPU) usage) for three server stations.

![Figure 4. Example of displaying server station parameters](image)

The received data represents the dependences of parameters of functioning of server stations on time. This data can be used to analyze the status of server stations, as well as to perform the distribution of user requests for file downloads. So, we will use the server state data presented in the form (1) at the next stage of the system: at the server selection stage.

5. The discussion of the results

The proposed approach to processing data on the state of server stations differs from the known approaches [9] that the data obtained has a structure suitable for further optimization of server selection. Also the records recorded in the database completely reflect the state of the system at a particular point in time. We believe that it is appropriate to use the multidimensional statistical procedure of cluster analysis [10] for the final processing of data. This procedure will allow us to split the received data into groups with similar parameters and add each new request to one of the groups, thereby determining the appropriate server for it.

6. Conclusion

The article suggests an approach to information processing when designing data distribution systems for highly loaded web applications. A model has been developed and it allows describing the status of server stations at the current time. A structure for presenting the parameters of the current state of the system suitable for further optimization of the server selection has been proposed. As a direction for the further research, we are planning to optimize the distribution of the user data on server stations. To do this, we will use not only the received data on the parameters of the server stations, but also the
data about the end users who perform uploading or viewing downloaded files. The task of data distribution will be solved using the methods of cluster analysis.

References
[1] Suherman, Zarlis M, Stirous S And Al-Akaidi M 2016 Applying a rateless code in content delivery networks IOP Conference Series: Materials Science and Engineering 237 conf 1
[2] What is load balancer? How load balancers work. URL: https://www.nginx.com/resources/glossary/load-balancing/ (reference date 12.10.2017)
[3] Membrey P, Plugge E and Hows D 2013 Practical Load Balancing: Ride the Performance Tiger (Expert’s Voice in Networking)
[4] Load Balancing Techniques and Algorithms URL: https://kemptechnologies.com/load-balancer/load-balancing-algorithms-techniques/ (reference date 12.10.2017)
[5] Server load balancing | Akamai URL: https://www.akamai.com/uk/en/resources/server-load-balancing.jsp (reference date 23.10.2017)
[6] Wilson S, Prakash P and Deepalakshmi P 2017 Server-based Dynamic Load Balancing IEEE Networks & Advances in Computational Technologies (NetACT) International Conference (Thiruvanthapuram)
[7] Node.js URL: https://nodejs.org (reference date 20.09.2017)
[8] MongoDB for GIANT Ideas | MongoDB URL: https://mongodb.com (reference date 20.09.2017)
[9] Sahoo J, Salahuddin M and Roch G 2016 A survey on Replica Placement Algorithms for Content Delivery IEEE Communications Surveys and Tutorials 19 pp 1002–1026
[10] Knyazev N L and Denisova L A 2018 The detection methods of dynamic objects J. Phys.: Conf. Ser. 944 012051