Analysis of the intermediate layer work in the three-tier architecture “client-server” of automation engineering problems

Analysis of the intermediate layer work in the three-tier architecture "client-server" of automation engineering problems

I A Barabanova[1], O Ja Kravets[1], S A Tkalich[1] and D I Mutin[2]

[1] Voronezh State Technical University, 14, Moscow ave, Voronezh, Russian Federation
[2] Mechanical Engineering Research Institute of the Russian Academy of Sciences, 4, Small Kharitonyevsky lane, Moscow, Russian Federation

E-mail: d.i.mutin@mail.ru, inna_gotovceva@mail.ru

Abstract. The article describes a mathematical model of a three-tier client-server architecture. The importance of using an intermediate server (application server), which provides preliminary data processing, is shown. Analytical conditions are obtained under which a three-tier architecture is more efficient than a two-tier one. Methods are proposed for increasing the performance of client-server systems in the tasks of automating production processes.

1. Introduction

Currently, the client-server technology is widely used in modern society. The main idea of this concept is that the client sends a request to the server, and the server, in its turn, gives a result. For successful application of the client-server technology, appropriate software is required, including client and server parts.

The main functions of the client are as follows:

- Sending requests to the server;
- Interpretation of the results received from the server;
- Presentation of the results to the user in some kind of a form.

The functions of the server part include:

- Receiving requests from client applications;
- Interpretation of requests;
- Optimization and execution of queries to the database;
- Sending the results to the client application;
- Ensuring the security system and access control;
- Database integrity management.

The described model of client and server interaction is a classic two-tier architecture. However, it has a number of drawbacks that are solved by the three-tier client-server architecture (figure 1).

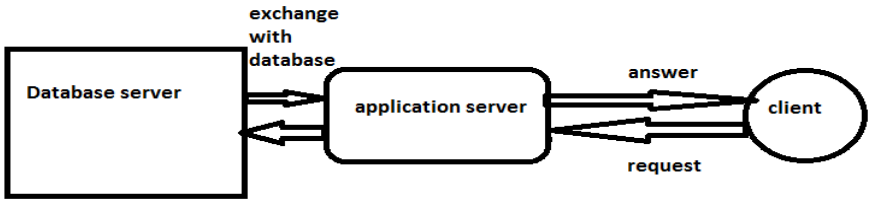

Figure 1. Three-tier client-server architecture.

The main advantages of the three-tier architecture are as follows:

- Scalability (the system allows you to build complex network configurations with many hundreds of users);
- High security (data security is largely dependent on the application task);
- High reliability (the system has a built-in mechanism for working with transactions, including their rollback);
- Configurability (the system is able to reconfigure when failures occur);
- Software replacement or modification at any level is possible without affecting the other levels.

All these advantages are achieved as a result of the introduction of an additional element - an application server. An application server is software that is an intermediate layer between a client and a server. Through a local or distributed network, the application server is physically connected to the client [1].

The most common categories of middleware products are:

- Message orientated – represented by MQseries and JMS;
- Object Broker – broadly represented by CORBA and DCOM;
- Component based – represented by NET and EJB.

2. Three-tier architecture model

It is supposed that a single server has the ability to serve multiple clients. Intermediate link converts protocols and displays some types of database queries to others, as well as combines the results of queries from various data sources. Using an additional server gives you more opportunities, for example, it reduces the load on client computers, since the application server distributes the load and provides protection against failures. The software installed on the database management server begins, in response to client requests, to search for information. Data processing includes its sorting, retrieving the required information and sending it to the user's address. The server software also performs other actions, such as updating, deleting, adding, protecting, etc. [2].

Considering the traffic of queries, it is assumed that a sequence of homogeneous events appear one after another at random points in time (figure 2). And, as a rule, all the work is divided into:

- The process of query receipt by the system;
- The process of query service in the system;
- The discipline of service.

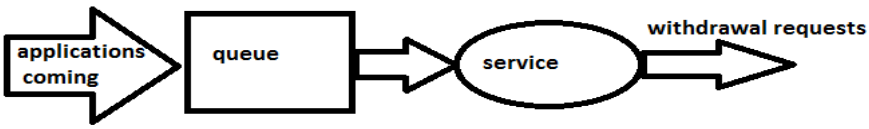

Figure 2. Query service scheme.

In some cases, the traffic of queries may be large at a certain time, and at another point of time it may be completely absent [3]. Therefore, it is necessary to build a model that will allow to calculate the parameters for the most efficient operation. These characteristics include:

- The number of channels;
- The traffic of queries;
- The intensity of query traffic processing;
- The holding time in the model.

All these parameters show how the system can handle the traffic of queries. The time of holding request in the system is particularly interesting, because it determines how quickly the client will receive a response.

The use of the Little formula (1), which is applicable to the client-server system, was previously discussed. It can be used to calculate the average service time per a request (t). It is assumed that a client is logged in at a random point in time and is waiting to be serviced. He will leave the system as soon as he is served.

The formula (1) shows that for any time distribution between two arrival events and any service time distribution, the average number of tasks is equal to the product of the arrival intensity λ by the average holding time in the model.

$$\bar{N} = \lambda \bar{t} \tag{1}$$

Then, the average service time can be represented by the following formula, where μ is the average performance of the servicing device (database server):

$$\bar{t} = \frac{1}{\mu - \lambda} \tag{2}$$

The relation for the average service time is as follows:

$$\bar{t} = \frac{1}{\mu_1 - \lambda} + \frac{b_1}{\mu_2 - b_1 \lambda} \tag{3}$$

This equation is represented as the sum of two hyperbolic functions. Achieving the average intensity λ of the minimum of the values $(\mu_1, \mu_2 / b_1)$ will be the boundary condition. Then, we denote this value as $k = \min(\mu_1, \mu_2 / b_1)$ and the average performance of the μ_2 database server will be considered to be the value unchanged. Thus, we have created a situation in which system performance depends only on the parameters of the intermediate server. Of course, if the boundary conditions for the intermediate server and the database server coincided, this would ensure the most efficient use of the performance of each of the links. However, a deviation from this principle can give an advantage, due to the less efficient use of one or several links that make up the system and thereby increase its performance in general [4].

$$\mu_2 = b_1 \mu_1 \tag{4}$$

Figure 3 shows the dependence of the average response time on the intensity of arrival events under the condition $\mu_1 > \mu_2 / b_1$. When the average intensity of arrival events is achieved, the intermediate server capabilities will not be fully utilized.

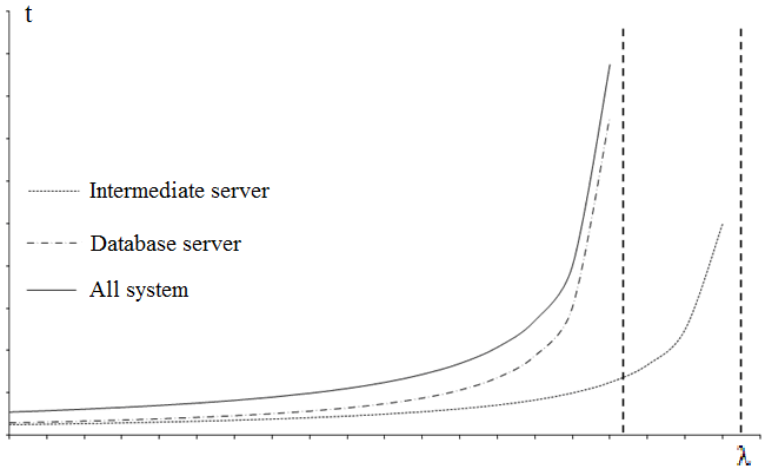

Figure 3. Dependence of the average response time on the intensity of arrival events.

A detailed analysis of the gain in the system performance with the addition of an intermediate layer has already been described in [5, 6]. With the help of the formulas (3) and (2) we define the condition under which the average response time of a two-tier system is larger than the average response time of a three-tier system. Since we consider that the database server in both cases is the same element, the formula (2) is as follows:

$$\bar{t} = \frac{1}{\mu_2 - \lambda} \tag{5}$$

Then the condition of greater efficiency of the three-tier system with respect to the two-tier is as follows:

$$\frac{1}{\mu_2 - \lambda} \geq \frac{1}{\mu_1 - \lambda} + \frac{b_1}{\mu_2 - b_1\lambda} \tag{6}$$

It is necessary to take into account the limitations imposed by the physical working conditions of the system:

$$\begin{aligned} &\lambda > 0 \\ &\lambda < \mu_2 \\ &\lambda < \mu_1 \\ &\lambda < \frac{\mu_2}{b_1} \end{aligned} \tag{7}$$

After some algebraic transformations, the formula (6) allows us to obtain an equation that determines the intersection points of the graphs of the two-tier and three-tier systems:

$$\lambda^2 - 2\mu_2\lambda + \frac{\mu_2}{b_1}\left(\mu_2 + b_1\mu_1 - \mu_1\right) = 0 \tag{8}$$

Intersection of graphs is possible only if there are real roots of equation (8). Therefore, we get the following condition:

$$\mu_2^2 - \frac{\mu_2}{b_1}\left(\mu_2 + b_1\mu_1 - \mu_1\right) \geq 0 \tag{9}$$

Since all the parameters in the equations are positive, we get the following:

$$(b_1 - 1)(\mu_2 - \mu_1) \geq 0 \tag{10}$$

The probability of successful completion of a transaction lies in the range from 0 to 1, therefore, the condition (10) is as follows:

$$\mu_1 \geq \mu_2 \tag{11}$$

Taking into account this formula, in order to get an advantage when adding an intermediate server, its performance should be higher than the performance of the database server.

The following values of μ_2 will be solutions to the equation (8):

$$\lambda = \mu_2 \pm \sqrt{\frac{\mu_2}{b_1}(1 - b_1)(\mu_1 - \mu_2)} \tag{12}$$

The intersection point of the graphs in figure 4 will correspond to the smaller of the solutions:

$$\lambda = \mu_2 - \sqrt{\frac{\mu_2}{b_1}(1 - b_1)(\mu_1 - \mu_2)} \tag{13}$$

The positivity of the average intensity of arrival events in the system gives us the following relationship between μ_1 and μ_2:

$$\mu_2 \geq (1 - b_1)\mu_1 \tag{14}$$

When the equality $\mu_2 = (1- b_1) \mu_1$ is performed, the three-tier system will have an advantage over the two-tier in the whole domain of definition λ, which can be seen in figure 4.

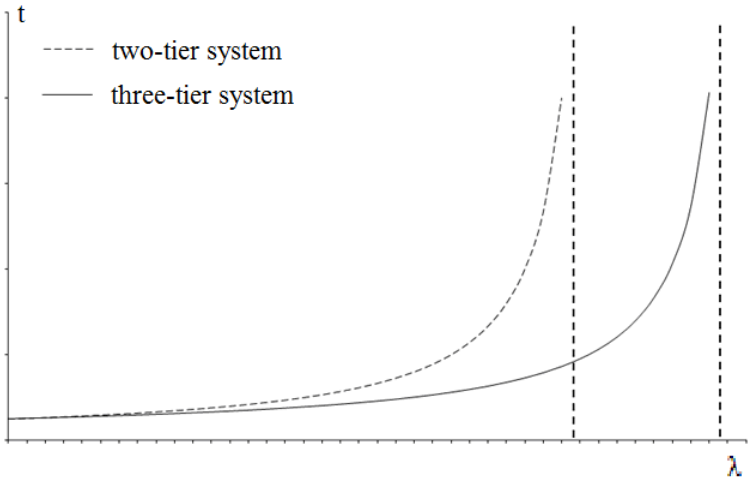

Figure 4. Comparison of the two-tier and the three-tier systems.

Undoubtedly, the three-tier architecture is not perfect and has its drawbacks, which, for example, include [7]:

- Complexity of creating applications;
- Complexity in deployment and administration;
- Cost of software, hardware and maintenance;
- Strict requirements to the performance of application servers and database servers and, as a result, high cost of server hardware;
- strict requirements to the speed of the channel (network) between the database server and application servers.

These drawbacks can be eliminated by moving to a system with more than three tiers and, therefore, by appropriately distributing the layers of the application software across different machines. And, therefore, there will arise issues connected to the behavior of the parameters of the architecture, increased by one more tier.

3. Conclusion

We have considered the behavior of the client server system when maintaining the middleware (application server). The variants of the system performance gain as a whole are analyzed. Further, it is important to study the behavior of the system parameters with an increase in the "client-server" architecture.

References

[1] Oluwatosin H S 2014 Client-Server Model *IOSR Journal of Computer Engineering* **16(1)** 67-71
[2] Mishra D and Alok M 2008 Design issues in client-server software maintenance *ACM SIGSOFT Software Engineering Notes* **33(5)** doi: 10.1145/1402521.1402524
[3] Kriegel H-P, Kröger P, Kunath P *et al* 2007 Proximity queries in large traffic networks *15th ACM International Symposium on Geographic Information Systems, ACM-GIS 2007, November 7-9, 2007, Seattle, Washington, USA* doi: 10.1145/1341012.1341040
[4] Higham D G, Hopkins W G, Pyne D B *et al* 2014 Relationships between rugby sevens performance indicators and international tournament outcomes. *Journal of Quantitative Analysis in Sports* **10(1)** 81-7 doi: 10.1515/jqas-2013-0095
[5] Sheme E, Frashëri N 2015 Software Tools and Techniques for Energy Efficiency in Data Centers *International Journal on Information Technologies and Security* **3(7)** 13-22
[6] Qamar E, Sajjad F, Kouser S *et al* 2017 An Optimized Handover Management System in 3G/4G-WLAN Using Genetic Algorithm *International Journal on Information Technologies and Security* **3(9)** 19-30
[7] Zeeshan A 2010 Proposing LT based Search in PDM Systems for Better Information Retrieval *International Journal of Computer Science & Emerging Technologies* **1(4)** 86-100