ITM Web of Conferences **6**, 03009 (2016)
DOI: 10.1051/itmconf/20160603009

Methodology of profiling network operations in software for distributed information systems

Oleg Lukyanchikov[1, a], Vyacheslav Filatov[2], Dmitry Biryukov[1] and Evgeniy Pluzhnik[1]

[1] *Moscow Technological Institute, Leninskiy pr., 38A, Moscow, Russian Federation, 119334*
[2] *Moscow Technological University, Vernadskogo pr., 78, Moscow, Russian Federation, 119454*

Abstract. Network operations take up large amounts of resources in distributed systems and in this case, existing profilers can correctly construct the execution tree of the program code and correctly identify the runtime, so you need other methods that will allow finding bottlenecks in the. The article discusses various approaches on the time efficiency assessment software, and based on a new method for profiling distributed systems, also an example of its use on a system built with agent technology-relational mapping. As a demonstration of the proposed method there's presented an example of the system, built using the technology of agent-relational mapping that performs network operations in distributed transactions. Thus showing that for the ease of using this technology we have to pay with lots of operations to monitor network transactions.

1 Introduction

One of the important qualities of information systems and applications is the temporal efficiency of the implementation. Particularly acute it affects the efficiency indicators of QoS in cloud systems.

Already in the process of developing software the developer has an idea about a possible "narrow" places, as in the limitation of time is taken more convenient implementation, but not more effective. For effectiveness evaluation in the development lifecycle there is only one stage in the course of which it is possible to detect and, if necessary, correct the most resource-intensive places software. Using life cycle according to [1], this stage is the test program, including the conduct of pre-state, interagency, acceptance and other testing and correction of program and program documentation according to test results. At this stage, functional testing, security testing is the performance testing includes load testing, stress testing, stability testing or reliability and volume testing. Profiling relates to load testing.

Profiling is the process of observing and recording metrics about the behavior of regions of code or an entire application. The profiling software may be implemented by the programmer in the process of developing software, implementing in code, fixing the execution time of basic operations, but more convenient after the development using CASE-tools called profilers. Today there are many profiling tools that are built into the composition of development tools and distributed separately (TAU Performance System, MPI Parallel Environment, GProf, Intel VTune Performance Analyzer), which allow us to construct the execution tree for the source code, control operations and even the control of multiple threads. Based on the analysis of bottlenecks using profilers, the developer is taking measures to improve the efficiency of implementation, namely: threading operations between threads, and reduces unnecessary passages cycles, selects specialized data storage structures (lists, queues, hash tables, etc.).

Network operations take up large amounts of resources in distributed systems and in this case, the profilers can correctly construct the execution tree of the program code and the right to determine the time, as it is affected by many factors such as:
- the operation is performed on remote compute nodes, whose characteristics are not always known and at the time of the operation accidentally downloaded, so it is not possible to accurately determine the runtime operations;
- network load isn't taken into account;
- number of users constantly changes.

Therefore, profiling methods are not applicable to distributed systems and other methods are needed that will allow to find the system bottleneck.

2 Methodology of profiling distributed systems

There are three approaches to evaluating execution time of a task on a computing resource[2].

1. Analysis of source code, based on an analytical model of internal system algorithms and key

[a] Corresponding author : bidmal@mail.ru

characteristics of the input data, an example of which is considered in [3].

2. Code profiling, during which run the calculations on plank data sets, and determine the system performance for the studied type of code, such as profilers.

3. Statistical methods of forecasting in which is stored results of past runs of the solving, and then used to predict the time of new work, examples of which are described in [4], and a system database for this purpose is described in [5].

Each approach has its limitations and large errors in the evaluation, so they should complement each other and be used together, especially for evaluating the run-time network operations.

For source code analysis and profiling is cost oriented graph that describes the routes code, where program blocks are nodes connected by edges, if the control can pass from one block to another. This graph is built by analyzing program code and using the Profiler, examples of this are described in [6]. When considering distributed systems, the priority for analysis are network operations, so to simplify the graph minor local operations can be neglected. Also in terms of the popularity of UML is more convenient and more correct instead of count to describe the route code of the activity diagram notations in UML, mainly due to the possibility to indicate parallel processes.

The developer of distributed systems knows about the basic processes occurring in the system, which can describe the sequence of actions when a network share, and build a graph or chart of activity. However, the developer uses third-party technologies and libraries for it can submit a "black box" [5]. Also in the analysis of code a developer has to consider many system operations, or to count them is not substantial, neglecting them, therefore, to analyze code not always immediate right. Hence the required experiments and analyze statistically the collected data, confirming the correctness of earlier estimates of operations.

Based on the foregoing, it is proposed the following method for profiling distributed systems:

1. Building a directed graph or chart activity notation in UML to describe the routes code.

2. The mathematical formula for calculating average time of the operations.

3. The tests for evaluating run-time operations, under varying number of computing nodes.

4. The correct mathematical formula of the test results.

5. If the test results failed to clearly confirm the correctness of the formula the addition of artificial delays and step 2, otherwise 6.

6. Definition of bottlenecks in the system, thus impacting the system performance.

3 Profiling distributed systems based on agent-relational mapping technologies

For an example of the proposed method, consider profiling a distributed system, built on the basis of agent-relational mapping technology (ARO), the principles of which are described in [7–9]. This technology is a hybrid technology of object-relational mapping (ORM) and remoting technologies (technologies of remote objects, remote procedure call, and are focused on message processing). As a result, the technology has ARO ORM interface and allows you to perform relational commands (select, update, insert, delete) to the database and other remote locations. This technology, due to its versatility, allows you to build almost any system architecture.

For research distributed system will be built consisting of a single database and a variable number of applications distributed between multiple computing nodes, as shown in Fig. 1. System specifications:

– Compute node 1 CPU Pentium Dual-Core E5200 2.5 GHz, 4Gb RAM, Windows 7 operating system on which the application is the initiator of operations, and note the time of operations and PostgreSQL database on 5 thousand records.

– Many computing nodes as virtual machines, raised on a separate server running VMwareEXSi, for each allocated 1 CPU, 1Gb RAM, Windows XP.

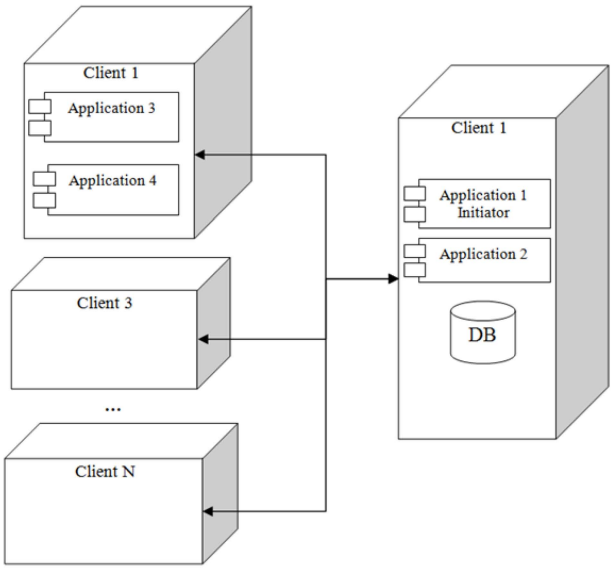

Figure 1. Experimental stand.

Researches on the effectiveness of this architecture are aimed to assess delays in data synchronization during the execution of update operations. This is the following sequence of actions:

– instruction is being generated;

– message with an instruction is sent to all clients and DBs;

– all remote objects on different compute nodes and the application performs the operation asynchronously and if successful, sent back to the initiator a message indicating successful execution, parallel operations are executed on all distributed databases;

– the initiator while finally collecting messages from all clients successfully captures the change and sends all other clients and the DB message with the command commit changes.

Based on the above process the activity diagram notation in UML will be built, as shown in Fig. 2.

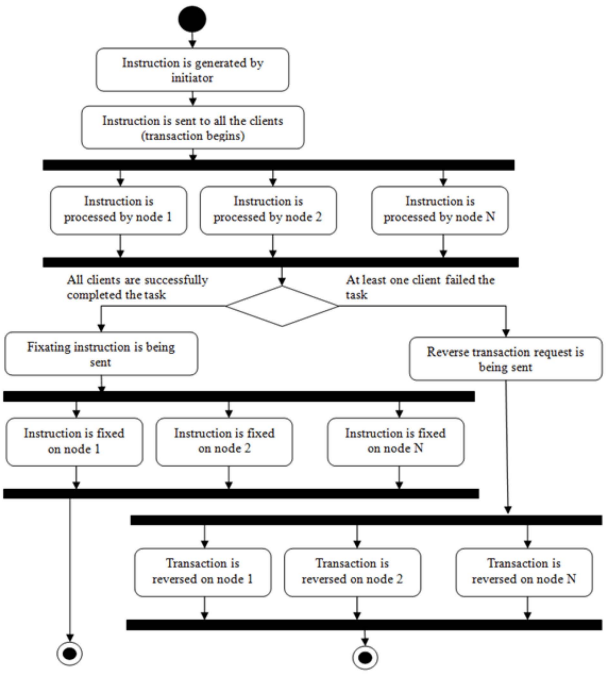

Figure 2. The activity diagram of the update operation for agent-relational mapping technology with UML notation

The average mathematical performance evaluation of a distributed system with distributed databases using the ARO will have the following form:

$$\overline{T} = \overline{t}_g + 2\sum_{i=1}^{N+M}\left(\overline{t}_{z_i}\right) + \max\left(\max_{i=1,N}\left(k\overline{T}_{o_i} + \overline{t}_{a_i}\right), \max_{i=1,M}\left(\overline{T}_{db_i} + \overline{t}_{a_i}\right)\right) + \overline{t}_c,$$

where $\overline{T}$ – average estimated time for a single operation; $\overline{t}_z$ – average time before request for operation is sent; $\overline{t}$ – average time of generating SQL request; $\overline{T}_o$ – average time of processing the operation by the object; $\overline{t}$ – average time for reply with operation results; $\overline{T}_{db}$ – average time of processing request on DBMS; $\overline{t}_c$ – average time of operation fixation; N – number of remote nodes; k – number of remote objects on a node; M – number of DBs.

This estimation does not include the loading units, location storing a record in a data structure, the probability of unsuccessful operations and other factors, so all times are the average value. To confirm the mathematical evaluation of the execution time of the update operation (1), conducted a series of experiments, the average results of which are shown in Fig. 3.

As can be seen from the graphs the mathematical evaluation of (1) is true. The deviation of graph values without delay from the values of schedule delay of the operation is different about the forced time delay, thus confirming that this operation will be executed in parallel on all the compute nodes.

A completely different kind of schedule has a delay of data transmission, it has a linear increase as in arithmetical progression, thus showing that the operation data is "narrow" place in the system.

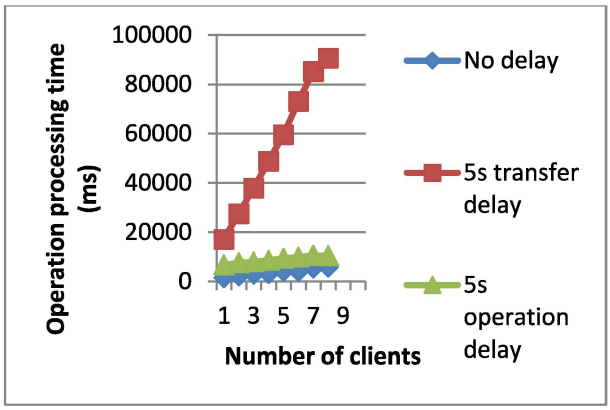

Figure 3. The results of the experiments to assess changes in run-time modification operations from changing the number of clients.

4 Conclusion

The article discusses various approaches on the time efficiency assessment software, and based on this a new method for profiling distributed systems. An example of application using this method is given on a distributed system containing one DB and many clients, built on the basis of technology ARO. The proposed method will more accurately assess the existing architecture of distributed systems, to identify among them the most efficient one and to generate recommendations for the design of new systems.

References

1. ISO/IEC 12207 (2008)
2. M.A. Iverson, F. Ozguner and L.C. Potter, *Proc. Eighth Heterogeneous Computing Workshop (HCW'99)*, pp. 99–111 (1999).
3. D.J. Kerbyson, H.J. Wasserman and A. Hoisie, *International Workshop on Innovative Architecture for Future Generation High Performance Processors and Systems*, pp. 27–37 (2002)
4. M.V. Devarakonda and R.K. Iyer, IEEE Trans. on Software Engineering, **15**, 1579 (1989)
5. E. Pluzhnik; E. Nikulchev and S. Payain, *2014 IEEE World Congress on Services (SERVICES)*, pp. 458-461 (2014)
6. S. Hao, D. Li, W. G. Halfond, and R. Govindan, *2013 35th International Conference on Software Engineering (ICSE)*, pp. 92-101 (2013).
7. E. Nikulchev, E. Pluzhnik, D. Biryukov O. Lukyanchikov and Simon Payain, International Journal of Advanced Computer Science and Applications, **6**, 22 (2015)
8. E. Nikulchev, O. Lukyanchikov, E. Pluzhnik and D. Biryukov, International Journal of Advanced Computer Science and Applications, **7**, 30 (2016)
9. M. Aubakirov and E. Nikulchev, International Journal of Advanced Computer Science and Applications, **7** (2016)