Methods and tools for profiling and control of distributed systems

R Sukharev¹, O Lukyanchikov¹, E Nikulchev¹,² * D Biryukov² and I Ryadchikov³

¹Moscow Technological University (MIREA) Moscow, Russia
²Moscow Technological Institute, Moscow, Russia
³Kuban State University, Krasnodar, Russia

*E-mail: nikulchev@mail.ru

Abstract. This article is devoted to the topic of profiling and control of distributed systems. Distributed systems have a complex architecture, applications are distributed among various computing nodes, and many network operations are performed. Therefore, today it is important to develop methods and tools for profiling distributed systems. The article analyzes and standardizes methods for profiling distributed systems that focus on simulation to conduct experiments and build a graph model of the system. The theory of queueing networks is used for simulation modeling of distributed systems, receiving and processing user requests. To automate the above method of profiling distributed systems the software application was developed with a modular structure and similar to a SCADA-system.

1. Introduction

All complex systems tend to be separated into smaller modules that perform autonomous tasks. Each developed software module may be included in different programs, if the conditions of its use are met, declared in the documentation for this module. Thus, a software module can be considered as a tool to deal with the complexity of the programs, and as a tool to combat duplication in programming. But modular programming has failed to fully implement all its concepts in full [1]. Therefore, with the development of networks and distributed systems, modules became separate network applications, which formed a service-oriented system Software-as-a-Service (SaaS) and more complex cloud systems. However, the complexity of the analysis of such systems also increased, since operations are performed on different computing nodes, and for each service a different load is applied. So today it is important to develop methods and tools for profiling and management of distributed systems.

All existing applications for the analysis of distributed systems are based on the analysis of logs gathered from all services, such as ELK [2, 3] or Dapper [4] by Google. These tools allow finding errors or bottlenecks using the logs provided by the developers, but do not allow to effectively take corrective measures, that is, perform a control action on the system. Therefore, systems like SCADA (Supervisory Control And Data Acquisition) are preferred for profiling and management of distributed systems. The term "SCADA-system" is used when it comes to automated systems, that is, systems of control and management, carried out with the participation of a person – the dispatcher (operator). The information systems produce a great amount of operations. In result, the operator is unable to keep up, so it is preferable to only show false and time-consuming operations, this will help finding the bottlenecks in the system and reconfigure it if necessary.
2. Methods for Profiling of Distributed Systems

Today there is no widely accepted methods and tools for profiling of distributed systems, but there is a number of proposed methods in [5, 6]. Both of these methods for analysis and profiling of distributed systems are focused on simulation to conduct experiments and build the graph model of a system. After combining these methods, it is possible to identify the main stages of profiling distributed systems: (1) To form the graph representation of the process of solving functional problems as a coherent set of query processing. (2) To build and configure the simulation model of query processing for a specific version of a computer network and the distributed data warehouse. (3) To carry out the experiment on the simulation model to evaluate the execution time of the queries. (4) To determine the bottlenecks in the system, dramatically affecting the performance of the distributed system.

In these methods the profiling results depend on the simulation model, which determines the load on the system with a certain requests’ flow intensity. When profiling a distributed system using the calculation indicators for monitoring the system's effectiveness, following should be considered:

- a plurality of storage devices in the queuing systems (QS). This factor leads to a consideration of the QS as a system with a total lineup of applications, eliminating the receipt of the individual characteristics of the queues;
- generally non-deterministic time for the service of application unit. The time spent on application maintenance depends on the total volume of the package, with the result that there is a mutual dependence of the average time of service applications and the number of requests waiting for service;
- correlations between the time of receipt of applications coming from different elementary sources, leading to lack of steady-state flow.

Having calculated performance of individual QS’s, it is possible to calculate the performance of stochastic queuing network (SQN), defined by a set of characteristics described in [6]: (1) set \{S_1, S_2, ..., S_n\}, forming a network; (2) number of channels \(K_1, K_2, ..., K_n\) in the systems \(S_1, S_2, ..., S_n\), respectively; (3) trajectory matrix of requests \(R=||r_{ij}||\), where \(r_{ij}\) is the QS number, to which the request is addressed on the \(i\)-th path in the \(j\)-th service phase during the deterministic routing procedure; or matrix of the probabilities of the requests’ transfer from one QS to another: \(P=||p_{ij}||\), where \(p_{ij}\) is the probability that a request leaving \(S_i\) goes to \(S_j\); (4) number of requests circulating in an isolated network \(Z\); (5) intensity of request sources in an open network \(\Lambda=\{\lambda_i(k)\}\), where \(i\) is the type of the request, and \(k\) is the category of urgency; (6) time distribution laws \(F_1(t), F_2(t), ..., F_n(t)\) and request maintenance disciplines in the systems \(S_1, S_2, ..., S_n\). The systems \(S_1, S_2, ..., S_n\) and the connections between them determine the network structure. The intensity of the request sources \(\lambda_i(k)\), the service intensity \(\mu_i(k)\), the queue lengths, and the operating mode of the devices characterize the load and performance of the SQN.

Modeling of distributed information based on SQN system described in [7–9].

After building the simulation model, for profiling of a distributed system it is necessary to conduct a series of experiments with different system architectures, in which the number of calculated nodes, their location, data flow, and other parameters change [10, 11]. SCADA-like software will allow automating the construction of these systems, immediate implementation and the collection of data based on the results of the experiment.

3. Development Tools for Control and Profiling

Tools that automate profiling of distributed systems are implemented in hmiSCADA system (https://github.com/RSukharev/hmiSCADA), written in C++, using the Qt libraries. This application uses the principle of modular construction. The system consists of a main module, providing a complete set of basic functions for processing and visualization of information, as well as additional functional modules, implementing the data collection functions and optional monitoring and control functions in the application. Additional modules for data collection and management are included in the source code of the profiled system.
The core module’s functionality includes: (1) obtaining data from external modules; (2) data visualization; (3) options to manage the software profiling process; (4) simulation.

Additional modules’ functionality includes: (1) data collection on the times of start and end of the test piece of code; (2) sending profiling data over the network to the main module of the system; (3) execution of the commands, received from the visualization and control modules.

The main difference of this system from other SCADA systems is that to render data received from an operator, it is not required to build the mnemonic of what should be displayed in advance. The mnemonic is generated automatically based on the data obtained. However, due to the modular structure of the system it is possible in the future to bring the graphical user interface to a common standard. It is also possible to provide compatibility with the data transmission standards for SCADA systems (OPC UA). Enhanced system functionality occurs as new modules are written and connected.

Modules are connected dynamically and have following interfaces: (1) request, intended for sending data; (2) processData, designed to execute commands and process the data received.

The work of the system is based on the exchange of messages between modules. The messages are exchanged using a data structure consisting of fields: «senderID» (identifier of the sending plugin), «receiverID» (identifier of the receiving plugin), «command» (the command to execute on the side of the receiving plugin), and «value» (forwarded data).

The architecture of the system allows following the process, in which messages may pass through a series of modules, each of which performs its work on data processing and forwards the result further, modifying and extending the functionality of the system as a whole.

The components of the system are shown in Figure 1:

Figere 1. Diagram of components of the hmiSCADA system.

The main component in the diagram (hmiSCADA) consists of the following elements:

- Controller (message manager and provides data exchange between system modules);
- Visualiser (visualization module, provides a visualization of the received data as a mnemonic scheme and provides a graphical user interface for controlling the profiling process);
- PluginUDP (network module, provides the reception of messages over the network protocol UDP from the module AgentUDP and redirection to the Controller component, as well as sending over the network of control commands from the Controller to the AgentUDP module);
- Syslog (the log analysis module is used to obtain profiling data based on log analysis of the system);
- Model (module designed for simulation);
- Additional modules of the system are represented in the diagram by components:
- Agent (data acquisition and management module for the profiled system);
- SysLog source (any data source that is compatible with the Syslog format).

The collection of information is ensured by including in the source code a profiled system of the Profiler class and implemented through following macros:

```
define PROFILE_BEGIN(nodeinfo, moduleinfo)
    _profiler_.profiler_(nodeinfo, moduleinfo, __FILE__);
#define PROFILE_APPEND(...) _profiler_.start(__FILE__, __LINE__, ##__VA_ARGS__);
#define PROFILE_STOP()
    _profiler_.stop(profileBlock, __FILE__, __LINE__);
#define ON_EXIT(...) _profiler_.onExit();
```

Macros call functions of the Profiler class, with parameters: the name of the file with the source code and the line number of the profiled block.

Example of use:
```
#include "Profiler/profiler.h"
std::string appID = qPrintable(QString("mini") + QString::number(getTimeNS()));

#define PROFILE_ENABLE
#define PROFILE_BEGINNER PROFILE_BEGIN(appID, " ");
    PROFILE_START(Q_FUNC_INFO);
#define PROFILE_ENDER PROFILE_STOP();

void mini::on_pbSend_released() {
    PROFILE_BEGINNER;
    sendMessage();
    PROFILE_ENDER;
}
```

Between PROFILE_BEGINNER and PROFILE_ENDER macros, an arbitrary number of lines of code can be placed, the main requirement is only the general execution context (within one function or within one block). The Agent module is the network and simultaneously managing the profiler module.

AgentUDP sends a message to the address and port of the main module of the system over the network. The recipient in the message is the Visualiser module. PluginUDP gives the data received via the network to the Controller class, adding its own identifier to it as a parameter «pluginID» (repeater plugin ID). Controller redirects the data to the module specified in the message as the recipient, i.e. Visualiser. In the message, the sender specifies the identity of the AgentUDP module, so if you need to send a response or a control command, the Visualiser module will generate a message with this ID as the recipient. The Visualiser message will be sent to the Controller message manager and then forwarded to the relay module (PluginUDP), which will forward it over the network to the AgentUDP module with the corresponding identifier.

The results of profiling a simple chat sending message by UDP are shown in Figure 2.
Figures 2. Visualization of profiling results of two instances of the chat application.

Instances of profiled applications (Figure 2) are marked with an icon showing the workstation. Functional blocks are marked with gray squares with the name of the profiled function. The sequence of transitions between the functional blocks is indicated by arrows. When you hover the cursor on the function block, a pop-up window appears with information about the execution time. Active functions and transitions between them are highlighted in green. The menu for controlling the instance of the profiled application is called up by right-clicking on the icon. By default, there are options for displaying information about the application and completing the application.

As a result, this SCADA-type profiler allows not only operating as a sniffer and showing packets in the system, but also visually shows the messages passing in the system at the function level and notices the time of their execution, thus when messages are delayed or lost they quickly find problematic places. This visual approach to mapping processes in the system and the approach of modular integration into existing systems can solve the problems posed with the complexity of analysis and profiling of distributed systems.

4. Conclusion

This article describes the methods and SCADA-like tools for profiling and management of distributed systems. Those allow automating the process of profiling systems, which makes it possible to locate bottlenecks and errors in the system.

An important factor in the development and promotion of these SCADA-like systems is the ability to integrate them into functioning distributed systems. That’s why the prototype hmiSCADA proposes modular architecture. With this architecture, third-party developers can develop their modules for hmiSCADA. Costs for the development of these modules are easily compensated by the time saved on profiling and analysis of the system.

Acknowledgments

The work was carried out within the framework of the state task of the Ministry of Education and Science, project No. 8.2321.2017 “Development and adaptation of control systems for compensation of dynamic deflecting effects on mobile objects in a state of dynamic equilibrium"
References
[1] Scott W R and Davis G F 2015 Organizations and organizing: Rational, natural and open systems perspectives (Routledge)
[2] Reelsen A 2014 Using elasticsearch, logstash and kibana to create realtime dashboards Available: https://secure.trifork.com/dl/goto-berlin-2014/GOTO_Night/logstash-kibana-intro.pdf, 2014.
[3] Bagnasco S et al 2015 Journal of Physics: Conference Series 608 012016.
[4] Sigelman B H et al 2010 Dapper, a large-scale distributed systems tracing infrastructure (Technical report, Google)
[5] Lukyanchikov O, Filatov V et al 2016 ITM Web of Conferences 6 03009
[6] Ivanov A Y, Gorshkov V S Methods for assessing the effectiveness of distributed information stores Available: http://vestnik.igps.ru/wp-content/uploads/V22/7.pdf [In Russian]
[7] Gurvich I, Van Mieghem J A 2014 Manufacturing & Service Operations Management 17 16.
[8] Hershey P C, Rao S, Silio C B, Narayan A 2015 IEEE Systems Journal 9 212.
[9] Bramson M. 2008 Stability of queueing networks (Springer Berlin Heidelberg).
[10] Nikulchev E, Pluzhnik E et al 2015 International Journal of Advanced Computer Science and Applications 6(6) 22.
[11] Nikulchev E, Pluzhnik E et al 2016 International Journal of Advanced Computer Science and Applications 7(1) 30.