Stochastic Model of a High-Loaded Monitoring System of Data Transmission Network

Kirill S. Shardakov
Department of Information Systems and Technologies, Emperor Alexander I St. Petersburg State Transport University
k.shardakov@gmail.com

Vladimir P. Bubnov
Department of Information Systems and Technologies, Emperor Alexander I St. Petersburg State Transport University
bubnov1950@yandex.ru

Abstract

The article describes a parallel-sequential model of a non-stationary queueing system that simulates the operation of a Zabbix monitoring system based on a distributed architecture. An algorithm for generating a list of the states of such system and its flowchart is given. Derived state transition rules and transition diagram. A sequential algorithm for generating a coefficient matrix for system of ordinary differential equations and its flowchart are given.

1 Introduction

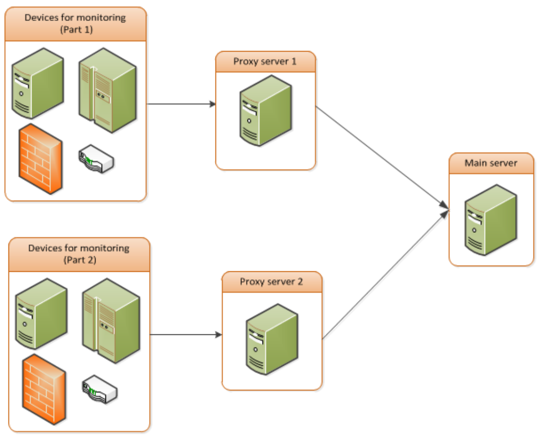
Automated monitoring systems are an important part of any information system. Monitoring is a continuous process of observing and registering the parameters of an object, processing them, comparing them with threshold values. Information systems in transport are developing rapidly, which causes an increase in the amount of net-work devices within the information system. In order to maintain full operability and timely response to problems within the information system, it must be included in the monitoring system. This monitoring system must cope with the increasing load. One of the main tasks is the ability to carry out an analytical calculation of the probabilities of the states of the monitoring system under various loads.

The most popular and easily scalable to adapt to the load freely distributed monitoring system for information systems is Zabbix [Sha18]. Many authors use stationary models of the theory [Zeg12-Upa16]. But of great interest are non-stationary queueing systems (nQS). Examples of works devoted to the non-stationary models are [Bub11, Bub99, Bub15, Bub15]. The disadvantage of the works [Bub11, Bub99] is that they consider only the classical numerical method for solving systems of ordinary differential equations (ODE) - the Runge-Kutta method, but the numerical-analytical method is presented in [Bub15, Ser15] , the speed and accuracy of which, when solving ODE system describing nQS, exceed the most common, when solving this kind of problems, the Runge-Kutta method. The advantage of this method is a recursive algorithm for generating a matrix of coefficients of an ODE system without deriving the general equation of the ODE system. But this algorithm also has significant disadvantages, described in [Sha18]. Also, in [Sha18], a sequential method for generating matrix of coefficients was proposed, without the disadvantages of the recursive method. In [Ser15] an algorithm is presented for the network model of the nQS.

This article describes a previously unreleased parallel-sequential model of the nQS. In constructing the model, modified sequential algorithm for generating a matrix of coefficients was used.

2 General description of the parallel-sequential model

As is known from [Sha18], the Zabbix monitoring system allows to distribute the load and scale through proxy servers using. Each proxy server collects data from its own set of devices, and then sends the data to the main server, which handles the processing. Consider the option in which there are two proxy servers and one main server. Figure 1 shows a general scheme of the interaction of system components.



Devices for monitoring (Part 1)
Proxy server 1
Main server
Devices for monitoring (Part 2)
Proxy server 2

Figure 1: Simplified diagram of the interactions of the monitoring system components

An information system interacting according to such scheme can be considered as queuing system. We divide such system into two subsystems, in the first subsystem there will be a proxy server, in the second - the main server. Both proxy servers have the same bandwidth, each proxy server has a separate independent queue of tasks, since it is engaged in polling values from specific set of devices. In other words, proxy servers work in parallel. After the request is processed by the first subsystem (by any of proxy servers), it immediately goes to the main server for processing. After processing the task by the main server, task is considered as processed. Each server is considered as separate queueing channel. Figure 2 presents a simplified diagram of such queueing system (QS).

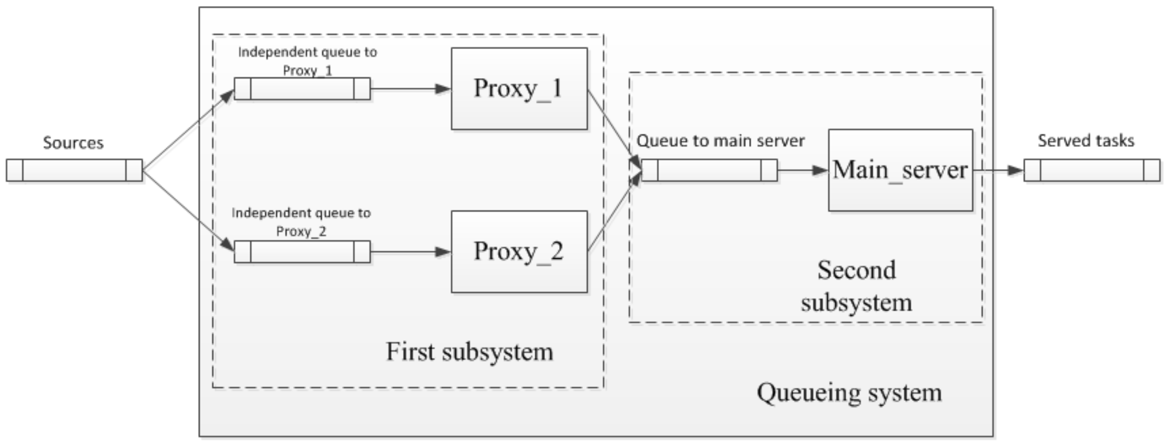


Figure 2: Simplified diagram of parallel-sequential QS

At any time, such QS can be described by the vectors "in_system" = [in_1, in_2, in_3] and "served" = [out_proxy, out_main]. The "in_system" vector contains three values, each value is equal to the number of tasks in the channel with corresponding number, while ||in_system|| = $\overline{0, N}$. The "served" vector contains two values, where each value is equal to the number of tasks served by corresponding subsystem, while ||served|| = $\overline{0, N - ||in\_system||}$. Where N is the total number of tasks that can enter to the system. At the input, system receives successive which depend on the number of tasks. Tasks are served in the first subsystem with intensities {μ1, μ2, … , μN}, and in the second subsystem with intensities {μ_main1, μ_main2, … , μ_main N}, which also depends on the number of tasks.

3 Modification of the state list generation algorithm

For a parallel-sequential model, the number of possible states is Ns = ((N+1)*(N+2)*(N+3)*(N+4))/24, and depends on the number of tasks which can enter to the system. With a successive increase in the number of such tasks by 1, a sequence of numbers is formed from the values of Ns, which is on line 5 of Pascal's triangle (sequence A000332 in the OEIS).
The next step is to divide the states into N+1 groups so that each group has a constant value of "served" [out_main] and other parameters can be changed. Each group must be divided into N+1 subgroups so that each subgroup has a constant value of "served" [out_proxy] and remaining parameters can be changed. Each subgroup must be divided into N+1 subgroups of the second order so that in each subgroup of second order there is a constant value ||in_system|| and other parameters can be changed. Create an empty structure to store the list of states. Iterate through all possible combinations of parameters describing the state of the system. We consider a combination as new state if it satisfies the following conditions simultaneously: (1) in_1 + in_2 + out_proxy <= N; (2) in_3 = out_proxy - out_main. In this case, the generated state is placed in the list of states in the subgroup of the second order according to the order of grouping at the

address "states" [out_main] [out_proxy] [||in_system||]. After list of states generating using this algorithm, all states will be listed in order in sequential numbering, which allows not to sort the list and the matrix of coefficients additionally and provides convenient way to describe possible transitions between states. Figure 3 shows the flowchart of the algorithm for generating a list of states for a parallel-sequential model.

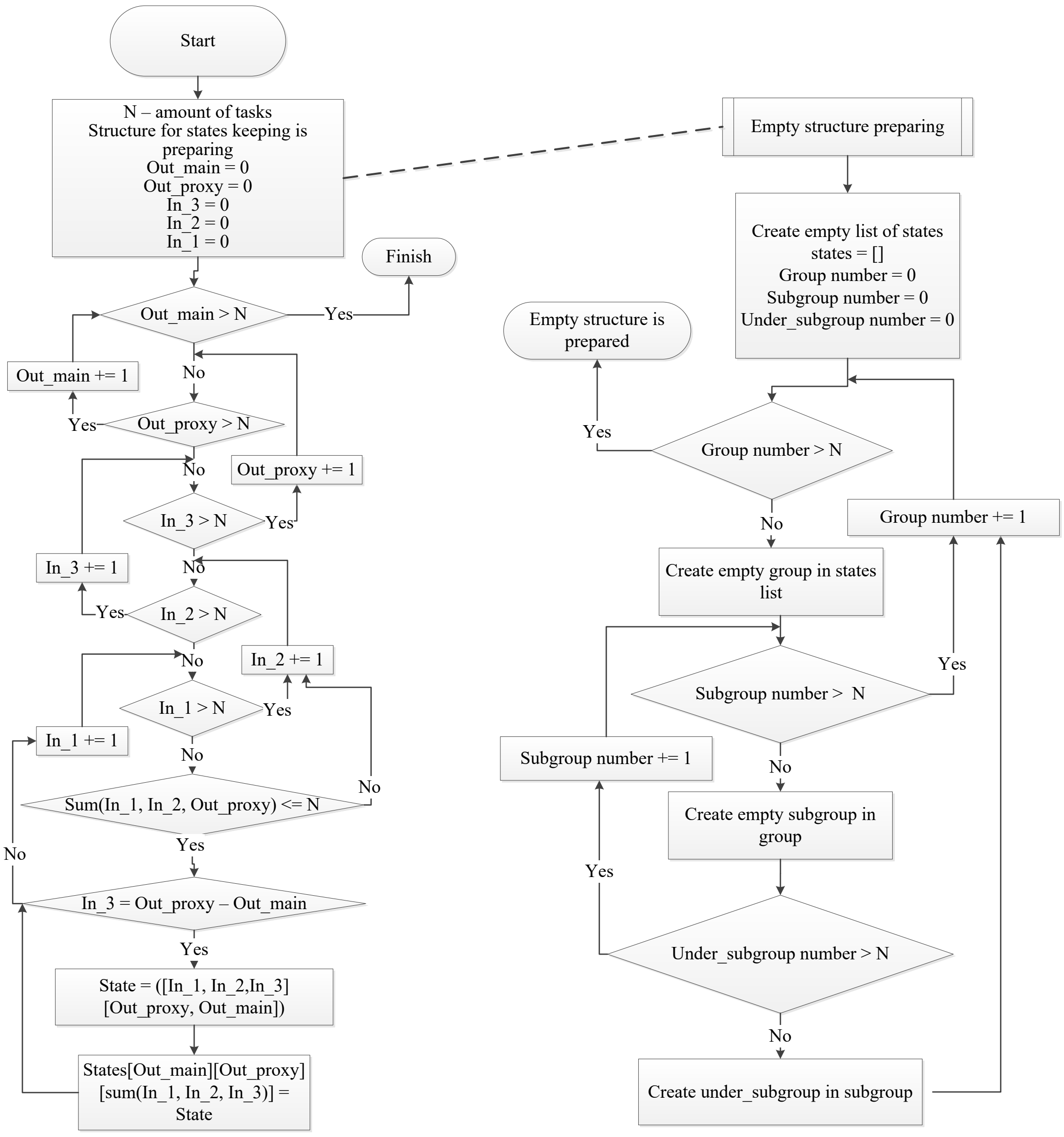

Start
N – amount of tasks
Structure for states keeping is preparing
Out_main = 0
Out_proxy = 0
In_3 = 0
In_2 = 0
In_1 = 0
Finish
Out_main > N
Yes
No
Out_main += 1
Out_proxy > N
Out_proxy += 1
In_3 > N
In_3 += 1
In_2 > N
In_2 += 1
In_1 > N
In_1 += 1
Sum(In_1, In_2, Out_proxy) <= N
In_3 = Out_proxy – Out_main
State = ([In_1, In_2,In_3] [Out_proxy, Out_main])
States[Out_main][Out_proxy] [sum(In_1, In_2, In_3)] = State
Empty structure preparing
Create empty list of states
states = []
Group number = 0
Subgroup number = 0
Under_subgroup number = 0
Empty structure is prepared
Group number > N
Group number += 1
Create empty group in states list
Subgroup number > N
Subgroup number += 1
Create empty subgroup in group
Under_subgroup number > N
Create under_subgroup in subgroup

Figure 3: The flowchart of the sequential algorithm

4 Formation of transition rules and matrix of coefficients

In the previous step, a structured list of all possible states was obtained. Thanks to this grouping, you can easily derive the rules for transitions between states. Need to notice that the basic rules of transition between states can be represented as the life cycle of task that passes through the described QS:

1. The new task enters the system on any of proxy servers, **transition from the current state**;
2. The new task enters the system on any of proxy servers, **transition to the next state**;
3. Task is served in the first subsystem (by any proxy server), **transition from the current state**;
4. Task is served in the first subsystem (by any proxy server), **transition to the next state**;
5. Task is served in the second subsystem (main server), **transition from the current state**;
6. Task is served in the second subsystem (main server), **transition to the next state**.

Also, by grouping the list of states, we can derive several rules:

1. Transition within one subgroup of the second level is impossible;
2. Within the subgroup "subgroup", transition from the second level subgroup "under_subgroup"

to the state “i” and “i”+1 (if they exist) of the next second level subgroup “under_subgroup”+1 is possible, if that "under_subgroup" is not the last nonempty subgroup of the second level of the subgroup "subgroup". The trigger for such transition is the receipt of new task in the first subsystem;

3. Within one group "group" transition is possible from any subgroup "subgroup" to subgroup "subgroup"+1. In this case, the transition is possible from the state "i" of the second level subgroup "under_subgroup" to the states "i" and "i"-1 (if they exist) the second level subgroup "under_subgroup". The trigger of such transition is the task completion in first subsystem and its transition to second subsystem;

4. Inside the list of states "states" transition is possible from group "group" to group "group"+1. The transition is possible from the state "i" of the subgroup of the second level "under_subgroup" of the subgroup "subgroup" to state "i" of the second level subgroup "under_subgroup" of the subgroup "subgroup". Such transition is im-possible from the first non-empty subgroups in the group. The trigger of such transition is the task completion in the main server.

Figure 4 shows a simplified diagram of transitions between states for a parallel-sequential model with N = 3. As you can see, all states are irretrievable, which corresponds to the goal.

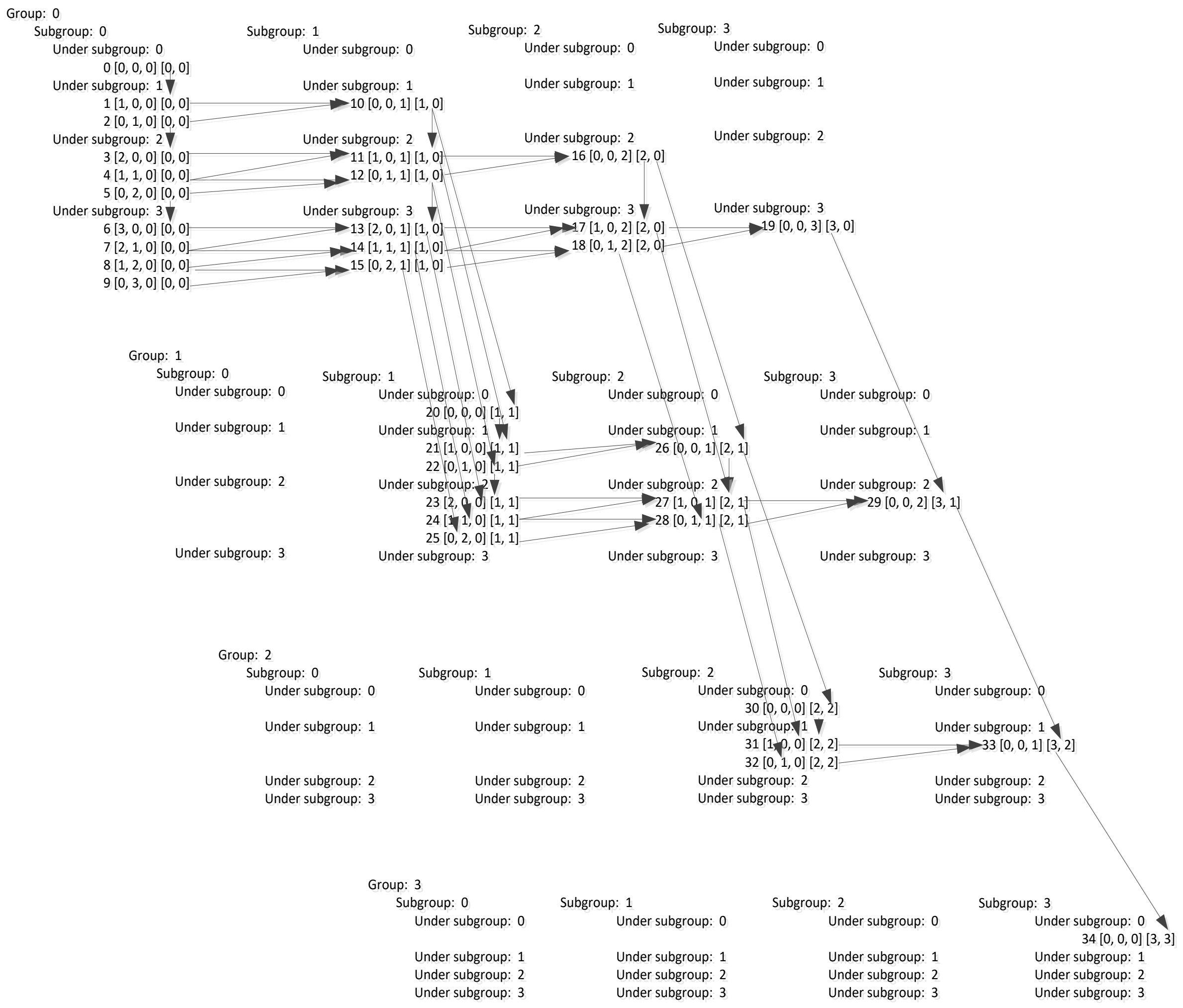


Figure 4: Simplified state transition diagram with N = 3

Consistently going through the states from generated list of states and applying the detected rules to each condition that suits the conditions, it becomes possible to simultaneously fill the matrix of coefficients. Each state is checked for compliance with certain conditions. The required changes are made to the coefficient matrix on the run. After making all changes in the matrix of coefficients, it remains only to solve the system of ordinary differential equations by a numerical-analytical method.

Figure 5 shows the flowchart for generating “matrixA” coefficient matrix.

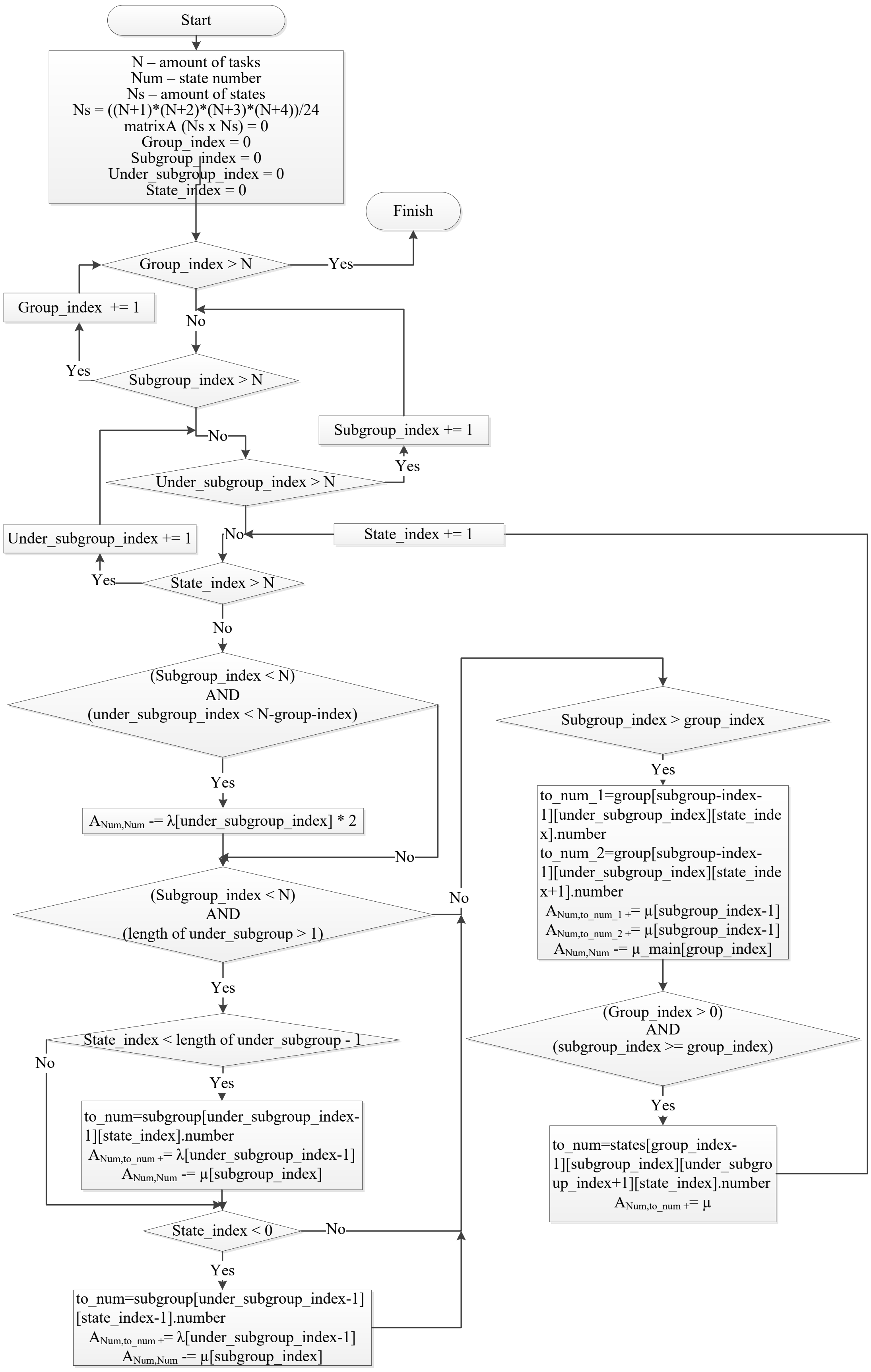

Start
N – amount of tasks
Num – state number
Ns – amount of states
Ns = ((N+1)*(N+2)*(N+3)*(N+4))/24
matrixA (Ns x Ns) = 0
Group_index = 0
Subgroup_index = 0
Under_subgroup_index = 0
State_index = 0
Finish
Group_index > N
Yes
Group_index += 1
No
Yes
Subgroup_index > N
Subgroup_index += 1
No
Yes
Under_subgroup_index > N
Under_subgroup_index += 1
No
State_index += 1
Yes
State_index > N
No
(Subgroup_index < N) AND (under_subgroup_index < N-group-index)
Yes
A_Num,Num -= λ[under_subgroup_index] * 2
No
(Subgroup_index < N) AND (length of under_subgroup > 1)
No
Yes
State_index < length of under_subgroup - 1
No
Yes
to_num=subgroup[under_subgroup_index-1][state_index].number
A_Num,to_num += λ[under_subgroup_index-1]
A_Num,Num -= μ[subgroup_index]
State_index < 0
No
Yes
to_num=subgroup[under_subgroup_index-1][state_index-1].number
A_Num,to_num += λ[under_subgroup_index-1]
A_Num,Num -= μ[subgroup_index]
Subgroup_index > group_index
Yes
to_num_1=group[subgroup-index-1][under_subgroup_index][state_index].number
to_num_2=group[subgroup-index-1][under_subgroup_index][state_index+1].number
A_Num,to_num_1 += μ[subgroup_index-1]
A_Num,to_num_2 += μ[subgroup_index-1]
A_Num,Num -= μ_main[group_index]
(Group_index > 0) AND (subgroup_index >= group_index)
Yes
to_num=states[group_index-1][subgroup_index][under_subgroup_index+1][state_index].number
A_Num,to_num += μ

Figure 5: The flowchart of the algorithm for generating matrix of coefficients

5 Conclusion

For the first time, a parallel-sequential model of a non-stationary queue system was built, which simulating the operation of the Zabbix monitoring system based on a distributed architecture. To build the model, modified sequential algorithm for matrix of coefficients generating of ordinary differential equations was used, which accelerated the process of matrix formation in several times. In the modified algorithm, state transition rules derived in the paper were used.

The further development of the model can be the introduction of a not instantaneous transition of each task to the second subsystem after servicing in the first subsystem. In this case, task will accumulate in the queue to the main server and arrive at it for processing by several pieces at once. Such addition will allow to more accurately simulate actual behavior of Zabbix monitoring system when using distributed architecture and increase the non-stationarity of system.

References

[Sha18] Shardakov K..: Comparative analysis of the popular monitoring systems for network equipment distributed under the GPL license. Intellectual Technologies on Transport 2018(1), pp. 44–48 (2018). (in Russ)

[Zeg12] Zegzhda P., Zegzhda D., Nikolskiy A.: Using graph theory for cloud system security modeling. Lecture Notes in Computer Science (including subseries Lecture Notes in Arti-ficial Intelligence and Lecture Notes in Bioinformatics), pp. 309–318 (2012).

[Oso13] Osogami T., Raymond R.: Analysis of transient queues with semi definite optimization. Queueing Systems, vol. 73. pp. 195–234 (2013).

[Upa16] Upadhyaya S.: Queueing systems with vacation: an overview. International journal of mathematics in operational research, vol. 9, issue 2. pp. 167–213 (2016).

[Bub11] Bubnov V., Khomonenko A., Tyrva A.: Software reliability model with coxian distribution of length of intervals between errors detection and fixing moments. International Computer Software and Applications Conference, pp. 310-314 (2011).

[Bub99] Bubnov V., Safonov V.: Razrabotka dinamicheskih modelej nestacionarnyh system obsluzhivaniya. [Developing dynamic modeling of non-stationary systems.] Saint-Petersburg, (1999). (in Russ)

[Bub15] Bubnov V., Eremin A., Sergeev S.: Osobennosti programmnoj realizacii chislenno analiticheskogo metoda raschyota modelej nestacionranyh sistem obsluzhivaniya. [Features of the software implementation of numerical-analytical method of calculation models non-stationary service systems.] SPIIRAS Proceedings. 2015(1), pp. 218-232 (2015). (in Russ)

[Bub15] Bubnov V., Khomonenko A., Sergeev S.: Recursive method for generating the coefficient matrix of the system of homogeneous differential equations describing nonstationary system maintenance. Proceedings of International Conference on Soft Computing and Measurements. SCM 2015, pp. 75-77 (2015).

[Ser15] Sergeev S.: Method for compilation of the system of homogeneous differential equations for calculation probability-time characteristics which describing non stationary systems. Intellectual Technologies on Transport 2015(2), pp. 32-42, (2015). (in Russ)

[Sha18] Shardakov K., Bubnov V., Pavlov A.: Generating of the Coefficient Matrix of the System of Homogeneous Differential Equations. Workshop Computer Science and Engineering in the framework of the 5th International Scientific-Methodical Conference "Problems of Mathematical and Natural-Scientific Training in Engineering Education", pp. 42-47, (2018).

[Wol14] Wolff R., Yao Y.: Little’s law when the average waiting time is infinite. Queueing Systems, vol. 76. pp. 267–281 (2014).

[Sud13] R. Sudhesh, K.V. Vijayashree Stationary and transient analysis of M/M/1 G-queues. Int. J. of Mathematics in Operational Research, 2013. vol. 5. no 2. Pp. 282–299.

[Sud13] R. Sudhesh, L. Francis Raj Stationary and transient solution of Markovian queues — an alternate approach. Int. J. of Mathematics in Operational Research, 2013. vol. 5. no. 3. Pp. 407–421.

[Bub14] Bubnov V., Tyrva A., Eremin A.: A set of non-stationary queuing system models with phase-type distributions. SPIIRAS Proceedings, vol. 6, pp. 61–71 (2014).

[Feh70] Fehlberg E.: Low-order classical Runge—Kutta formulas with step size control and their application to some heat transfer problems. NASA Technical Report 315 (1969), extract published in Computing, vol. 6, pp. 61–71 (1970)