Simulation of Tasks Distribution in Horizontally Scalable Management System

D Kustov[1], A Sherstneva[2], I Botygin[3]

[1] E-learning Institute, National Research Tomsk Polytechnic University, Lenin Avenue, 30, Tomsk, 634050, Russia
[2]Department of Higher Mathematics, Institute of Physics and Technology, National Research Tomsk Polytechnic University, Lenin Avenue, 30, Tomsk, 634050, Russia
[3]Department of Computer-Aided Designed System, Institute of Cybernetics, National Research Tomsk Polytechnic University, Lenin Avenue, 30, Tomsk, 634050, Russia

Abstract. This paper presents an imitational model of the task distribution system for the components of territorially-distributed automated management system with a dynamically changing topology. Each resource of the distributed automated management system is represented with an agent, which allows to set behavior of every resource in the best possible way and ensure their interaction. The agent work load imitation was done via service query imitation formed in a system dynamics style using a stream diagram. The query generation took place in the abstract-represented center – afterwards, they were sent to the drive to be distributed to management system resources according to a ranking table.

1. Introduction

Currently, there have been significant changes in automated systems infrastructure for various complicated organization-technical complexes. Constant hardware modernization, production number increase and cost reduction are not the only reasons behind it. The difficulty of data processing itself has increased (especially in scientific research applications), while applied problems are solved in a distributed manner.

The amount of stored and processed data is yet another factor that actively influences the research and development of the new information-calculation structures. The modern management systems are often using a territorially distributed structure, formed of tens of thousands computers under the concepts of meta-computing and grid-technologies. Sketchily, such structure includes computing nodes functioning as calculators and data storages, communication network, middleware management software multi-layered architecture, various targeted domain-functional modules and systems. While I do not claim it as an in-depth description, the main features of such infrastructures are their large scale defined by the territorial distribution and decentralized management due to non-deterministic process management type.

Even those features listed allow to conclude that it’s difficult, if not impossible, to use distributed automated management systems for strict formal mathematical methods.

Such circumstance largely defines the special role of the imitational modeling during the design of distributed automated management systems (DAMS) [1]. Only an imitational model allows to adequately represent DAMS primary components, their interaction and the dynamically changing system condition.

Therefore, it is possible to replay all the DAMS functioning process over time, accounting for its designed logic structure, various management factors, random events and limitations. Let's point out the prospect of using imitational models for integration with decision support systems. In this case, it is possible to search for optimal variants of distributed automated management system at every step of development.

Various concepts, technologies and implementation methods are used to solve technological and methodological issues of DAMS development, for planning and implementing the rational use of its resources [2]. One of the approaches that provides the necessary DAMS functioning efficiency, is, definitely, load balancing of its software-hardware components. The solution for this problem is targeted towards providing the necessary technological interaction between integral components in DAMS functioning process. The load balancing (equalization) ensures optimized calculations for distributed architecture of informational and computational resources of complex systems, and increases their reliability as well.

2. Simulation model of load balancing

The current work researches the load balancing algorithm [3] developed by authors. The testing was done in a model experiment imitating the distributed automated management system functioning with a more than 500 interacting components. Regarding the model itself, DAMS was represented as a generalized mass-service system with a multiple channels, service errors, non-homogenous service query streams, with a multiphase service query processing, unlimited queues, open-type system with limited reliability. All the DAMS hardware-technical and software-informational infrastructure components are acting as the generic model objects.

The imitational model development uses universal programming languages (C, C++, C#, Java, Fortran etc.), specialized languages (GPSS/H, GPSS World, SIMSCRIPT, SIMPLEX3,GASP, SIMULA, SLAM etc.) [4-15], specialized modeling environments (MicroSaint, iGrafx Process, SIMUL8, VisSim, Extend, Arena, AnyLogic, Enterprise Dynamics etc.) [16-22]. Using the universal programming languages most certainly allows to receive the models most adequately resembling the real objects. Specialized modeling environments, however, allow to significantly reduce the model development time and easily modify them if necessary.

As the modeling environment, the authors used the domestic object-oriented imitational modeling environment AnyLogic [23, 24]. The combine approach, including agent modeling and system dynamics, was used as a modeling method. The system dynamics was used for distributed system components abstraction, and agent model was employed to adequately represent the functional software-hardware components. In other words, for the studied problem, every distributed automated management system component was represented as an agent, which is the best way to set a behavior for each resource, to provide a proper way for their interaction.

The resource (agent) load imitation was modeled via service query generation formed with the system dynamics style using a stream diagram to create a determined and alternative operations passing algorithm. The query generation is done via abstract-view center; then, queries are delivered to the drive (distributor) that distributes queries to DAMS resources. The query distribution is executed according to the following principle:

Step 1. If any service queries remain in the drive, the execution priority is given to the resources that aren't processing any queries at the moment, i.e. are currently free.

Step 2. If there are no free resources at the moment, the busyness coefficient of every node is calculated $L = P*(Q+1)$, where P is resource performance, Q the number of queries in queue for this resource. The query is sent to the resource with the lowest busyness coefficient. Specifically, it is sent to queue. Afterwards, the busyness coefficient for this resource is recalculated accordingly.

Step 3. If the query has finished being processed, it is deleted.

For this model to resemble a real system more closely, the following randomly generated events have been included:

- The possibility of DAMS resource refusing to accept query;

- The possibility of resource failure at any moment of time;
- The possibility of resending processed request to another DAMS resource for its further processing.

The AnyLogic Statechart class object were chosen as the primary blocks of the imitational model. Those objects can determine a state diagram that describes system functioning as a set of states with different readiness coefficients and state transitions. Additionally, every transition has a percentage rate assigned statically or dynamically calculated during modeling. Every resource used one of the following states:

- Resource initial state – «FreeState» (DAMS resource is ready to receive service queries);
- «BusyState» – DAMS resource is busy processing the incoming queries;
- «BreakState» – DAMS resource is not available (break/failure).

«FreeState» is pretty simple and always relates to an initial state for any distributed automated management system resource. In such condition DAMS resource is always ready for incoming queries.

After receiving a request, resource enters the «busyState» and starts processing. The processing duration can be calculated as a query processing time multiplied by resource components performance. By default, the query processing time is equal to a single unit of modeling time. DAMS resource performance is determined as generalized characteristics performance for every component that belongs to this resource. Resource performance is correlated by its load and can be defined by dimensionless coefficient between 0 and 1. The processing duration may vary from 1 up to 10 modeling time units. After query processing is finished, there is a chance to redirect query to another random system resource (the probability is 0.05 by default). Otherwise, the query is deemed as processed. The processed query counter increases by 1, resource enters «freeState».

The «breakState» is necessary for modeling breaks/failures of distributed automated management system. The chance to enter break/failure state is defaulted to 0.1 and calculated for every unit of modeling time. Entering this state will return the query in process to the query pool. The duration of break/failure state varies between 1 and 100 modeling time units. After break/failure state expires, the resource enters the initial «freeState».

The following parameters were chosen as DAMC imitational model input data:

- The total number of DAMC resources (agents).
- Service query generation intensity.
- The time required to process a single service query.
- The resource break/failure rate, defined by Bernoulli distribution.
- The resource break/failure duration, determined as a random value within minimum and maximum boundaries.
- The chance to resend a service query from one resource to another.

All of the default data above can be set for imitational modeling with model interface. The work end condition was time limit. The experiment used time limit of 200 modeling time units, which is sufficient to receive a complete work load data for each resource and system as a whole.

There was a series of experiments with different input data values. The varying parameters were: query generation intensity and system resource number. The query stream intensity varied from 1 to 150 per modeling time unit. The number of DAMC resources was determined by the following array a = {5, 10, 20, 30, 50, 75, 100, 125, 150, 200, 250, 300, 350, 400, 500}.

As a result of DAMC imitational model work, the following numerical criteria were chosen for estimation:

- The total number of queries processed by DAMC.
- The total number of DAMC breaks/failures.
- The total DAMC load coefficient.
- The performance of every DAMC resource.
- The number of processed queries for each DAMC resource.

- The number of failures for each DAMC resource.
- The load coefficient for each DAMC resource per unit of modeling time.
- The number of queries in queue for every DAMC resource per unit of modeling time.

All the received data was recorded in a structured manner in an external file. The imitational experiments have shown that the researched load balancing algorithm allows to determine the optimal amount of queries per unit of time to equally load all DAMC resources for any component infrastructure (up to 500 agents), component load or query routing. Moreover, after the received data analysis, the DAMC functioning quality can be estimated.

It is based upon such general information-communicational environment parameters as flow coefficient, average query processing time, average query processing delay, failure rate etc. The recommendations for increasing DAMC resilience have been created, which allows to reduce time and computation costs. Finally, the criteria in use allowed to get a complete statistic on distributed automated management system model functioning.

References

[1] Devyatkov V.V. Metodologiya i tehnologiya imitatsionnyh issledovaniy slozhnyh sistem: sovremennoe sostoyanie i perspektivy razvitiya. - M.: INFRA-M, 2013. - 448 p.
[2] Sherstnev V. S. , Sherstneva A. I. , Botygin I. A. , Kustov D. A. Distributed information system for processing and storage of meteorological data // Key Engineering Materials . - 2016 - Vol. 685. - p. 867-871.
[3] Botygin I.A., Tartakovsky V.A., Sherstneva A.I. Algoritm balansirovki nagruzki v raspredelennyh sistemah upravleniya // Informatizatciya i sviaz. – 2015. – № 1. – P. 28-31.
[4] Jefferson D.R. Virtual Time / Jefferson D.R. // Assoc. Comput. Mach. Trans. Programming Languages and Systems, 1985. – № 7. P. 404-425.
[5] Computer Simulation // Minuteman Software, available at http://www.minutemansoftware.com, 2015.
[6] GPSS World // Elina-Computer, available at http://elina-computer.ru/static/gpss-world.html, 2015.
[7] GPSS/H Software Products // Wolverine Software, available at http://www.wolverinesoftware.com/GPSSHProducts.htm, 2015.
[8] SIMSCRIPT III // CACI Advanced Simulation Lab, available at http://www.simscript.com/products/products.html, 2015.
[9] Jalote P. Fault Tolerance in Distributed Systems // P. Jalote. USA: Prentice Hall, New Jersey, 1994. – 448 p.
[10] What is Simplex3? // SOI-Software, available at http://www.simplex3.net/Body/Introduction/English/indexAbstract.html, 2015.
[11] Philip J. Kiviat, A. Colker A General Activity Simulation Program. – RAND Corporation, Document Number: P-2864, 1964. – 9 p.
[12] Sklenar J. Introduction to OOP in SIMULA, available at http://staff.um.edu.mt/jskl1/talk.html, 2015.
[13] Birtwistle, G.M., O.-J. Dahl, B. Myhrhaug and K. Nygaard: SIMULA Begin, AUERBACH Publishers Inc, 1973.
[14] Pooley, R.J.: An Introduction to Programming in SIMULA, Oxford, Blackwell Scientific Publications, 1987.
[15] Kirkerud, B.: Object-Oriented Programming with SIMULA, Addison-Wesley, 1989.
[16] About Micro Saint Sharp // Alion Science and Techology, available at http://www.microsaintsharp.com, 2015.
[17] iGrafx Process 2015 // iGrafx, LLC, available at http://www.igrafx.com/products/process-modeling-analysis/process, 2015.
[18] SIMUL8 // SIMUL8 Corporation, available at http://www.simul8.com, 2015.
[19] VisSim. A graphical language for simulation and model-based embedded development // Visual

Solutions, available at http://www.vissim.com, 2015.
[20] Simulation with ExtendSim // Imagine That Inc., available at http://www.extendsim.com/prods_overview.html, 2015.
[21] Arena // Arena Simulation Software, available at https://www.arenasimulation.com, 2015.
[22] Enterprise Dynamics // INCONTROL Simulation Solutions, available at http://www.incontrolsim.com/en/enterprise-dynamics/enterprise-dynamics.html, 2015.
[23] AnyLogic // AnyLogic, available at http://www.anylogic.ru/overview, 2015.
[24] Kelton W., Law A. Simulation Modeling and Analysis. McGraw Hill Higher Education, 3rd edition, 2000 – 784 p.