# ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И СИСТЕМЫ, ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

УДК 519.687.5
DOI: 10.17586/0021-3454-2021-64-11-879-886

## ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ СЕРВЕРОВ С ПРЕРЫВАНИЯМИ В БОЛЬШИХ МНОГОПРОЦЕССОРНЫХ СИСТЕМАХ

П. Е. Голосов[1], И. М. Гостев[2]

[1]*Российская академия народного хозяйства и государственной службы при Президенте РФ, 119571, Москва, Россия*
[2]*Институт проблем передачи информации им. А. А. Харкевича РАН, 127994, Москва, Россия*
*E-mail: igostev@gmail.com*

Рассматривается проблема управления облачной специализированной вычислительной системой, выполняющей разнородные ресурсоемкие задачи, способы выполнения которых можно представить как произвольный перебор большого числа вариантов. Проанализированы процессы распараллеливания заданий по данным в условиях неопределенности. Для решения проблем планирования поступающих входных потоков задач используется метод имитационного моделирования с помощью программного пакета SimEvent/Simulink/MatLab. Рассматривается функционирование системы и особенно серверов, представимых как автоматы конечных состояний. Особенность предложенной модели заключается в возможности прерывания работы сервера при появлении внешнего сигнала. Использование разработанной модели сервера позволяет повысить производительность системы за счет экономии времени выполнения задач каждым сервером и более эффективного распределения задач между серверами всей системы.

***Ключевые слова:*** *облачные вычисления, параллельные алгоритмы, автомат конечных состояний, имитационное моделирование, SimEvent, Simulink*

**Введение**. На современном этапе развития вычислительной техники часто возникают ситуации, связанные с решением разнородных ресурсоемких в вычислительном отношении задач [1—6]. К таким задачам относятся обработка больших массивов в картографии, поиск хешей заданной сложности, работа в информационных системах, основанных на параллельном поиске информации в различных источниках и др. Эти задачи можно объединить в класс задач, в которых решение можно искать параллельно во многих потоках, при этом каждый поток будет независим от остальных по данным [7, 8]. На основании этого такие задачи, как правило, решаются на множестве вычислительных облачных устройств, работающих параллельно [9—11], за счет чего получение решения ускоряется пропорционально количеству устройств ($n$), на которых обрабатывается данная задача.

При имитационном моделировании, например, систем массового обслуживания [12—14] часто возникает необходимость прерывания обслуживания текущей задачи (заявки), что особенно характерно при выполнении некоторой задачи параллельно на нескольких устройствах. Если на одном из них уже получено решение задачи, то на остальных дальнейшее ее выполнение приводит к снижению эффективности всей системы.

В настоящее время для проведения исследований по имитационному моделированию используется большое количество программных продуктов, среди которых один из самых мощных и широко распространенных — MatLab/Simulink [15, 16]. Пакет SimEvent, имеющийся в этой программе, специально предназначен для моделирования работы систем массового обслуживания, ориентированных на различные прикладные направления [17, 18]. Множество элементов этого пакета и их широкий функционал позволяют охватить практически все возможные ситуации, которые возникают в процессе решения задач имитационного моделирования.

Однако существует ряд проблем, первая из которых заключается в том, что на входе мультисерверной системы имеется некоторый поток задач и время поступления конкретного фрагмента задачи на обработку является случайной величиной. Данная ситуация иллюстрирует проблему синхронизации выполнения фрагментов задач во времени. Другая проблема возникает из-за отсутствия в пакете SimEvent таких компонентов, как сервер, допускающий прерывания, т.е. прекращение выполнения некоторой задачи при наличии внешнего сигнала прерывания. Существующие в SimEvent серверы выполняют загруженную задачу либо в соответствии с заданным заранее временем обслуживания, либо до появления на входе сервера задачи с большим приоритетом (в режиме вытеснения — preemption mode).

Поскольку в любой реальной системе существуют прерывания, обусловленные как внутренними, так и внешними факторами, то необходимо создание имитационной модели вычислительной системы, в которой предусмотрена такая возможность. В этой связи было принято решение о разработке модели сервера, в котором в дополнение к уже имеющемуся функционалу будет обеспечена возможность прерывания выполняемой задачи по внешнему сигналу.

**Архитектура Chart Server**. Разработанный Chart-сервер основан на автомате конечных состояний (Discrete Event Server), модуль которого имеется в пакете SimEvent и, в отличие от аналогичного автомата конечных состояний в пакете Simulink/StateFlow, работает с объектами-сущностями (entity). Структурная схема Chart-сервера представлена на рис. 1.

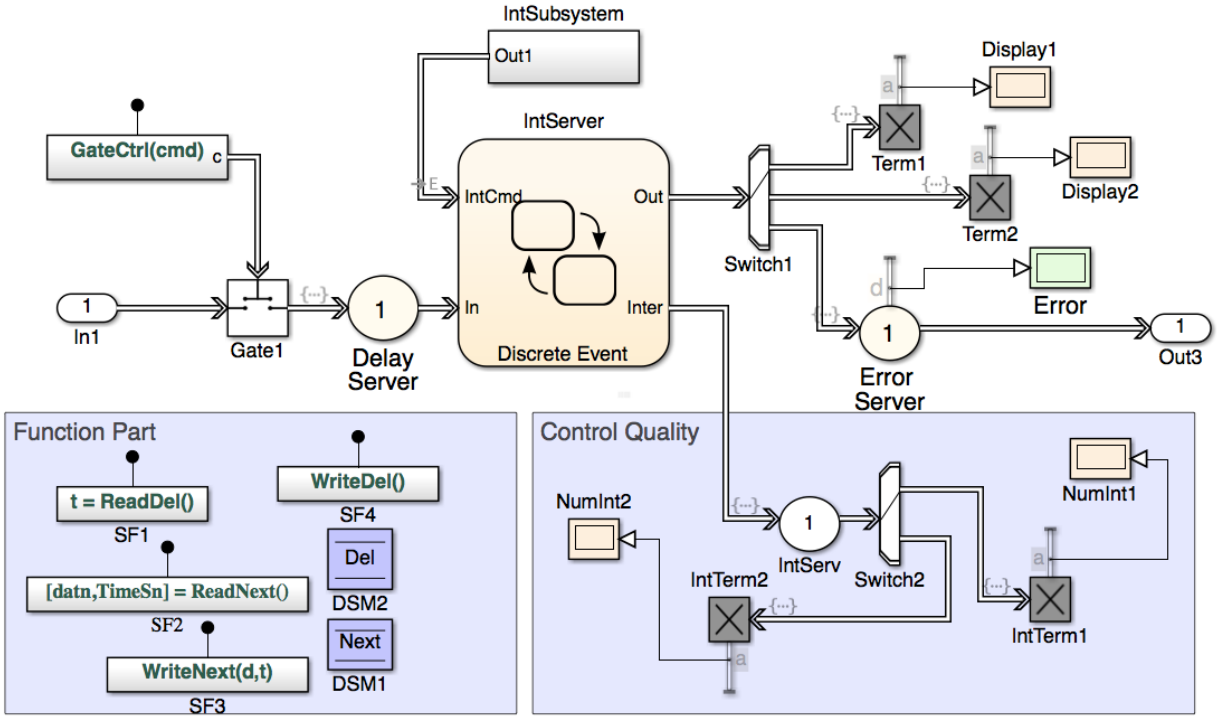


*Рис. 1*

Объекты (сущности), а в данном случае задачи или фрагменты задач, поступают на вход In1 модуля. Далее, через ключ Gate1 следуют на сервер задержки Delay Server. Этот ключ

обеспечивает поступление только одной задачи в данный модуль. После поступления задачи в блок Delay Server вызывает функцию GateCtrl(-1) и ключ отключает вход блока. Далее задача поступает на вход Discrete Event Server. По окончании работы в этом блоке задача поступает на выключатель Switch1 и в зависимости от параметра Term может быть направлена следующим образом:

— на Term1 — в этом случае выполнение задачи (фрагмента) закончено, и если текущий фрагмент содержит решение всей задачи, то счетчик выполненных задач увеличивается;

— на Term2 — это происходит в случае, если для поступившей задачи (фрагмента) уже найдено решение и работа задачи прекращается без выполнения;

— на Error Server — задача поступает на этот сервер в случае, если в процессе ее выполнения на сервере произошла ошибка и задачу (фрагмент) необходимо направить на повторное выполнение через выход Out1.

Для контроля за количеством такого типа решений использованы дисплеи — Display1, Display2 и Error.

В случае когда в процессе выполнения задачи происходит прерывание, инициируемое сигналом IntCmd, поступающим из подсистемы IntSubsystem на вход In сервера Discrete Event, задача направляется на выход Inter сервера Discrete Event. Для контроля за правильностью выполнения прерываний (блок Control Quality) используется механизм фиксации правильно и неправильно выполненных прерываний, основанный на присваивании параметру Term обрабатываемой задачи значения 1 или 2. В первом случае прерывание происходит в соответствии с поступившим сигналом IntCmd. Во втором случае, в силу нарушения синхронизации работы системы, возможно возникновение прерывания, инициированного для следующей задачи в результате сигнала прерывания для предыдущей.

В состав функционального блока Function Part включены процедуры и локальные переменные, необходимые для работы сервера Discrete Event. По окончании обработки текущей задачи происходит вызов функции GateCtrl(1) и ключ подключает вход блока для поступления следующей задачи.

**Блок IntSubsystem**. Для обеспечения прерываний работы сервера Discrete Event был разработан блок IntSubsystem, представленный на рис. 2. Он состоит из импульсного генератора Pulse Generator, блока вызова функций MatLab — Function Caller, блока Message Send и функции Simulink — SigInt(). Генерируемые импульсы непрерывно поступают на блок вызова функций, который вызывает функцию SigInt(). Частота следования импульсов определяется длительностью выполнения задач и выбирается равной примерно 10 % от длительности самой короткой задачи. В функции SigInt() происходит сравнение параметров текущей задачи (из локальной переменной Next), выполняемой в Discrete Event Server, и параметров уже выполненных задач из глобальной переменной Br. При обнаружении совпадений функция возвращает единичное значение сигнала, далее поступающего на блок Message Send, который и генерирует сигнал прерываний IntCmd.

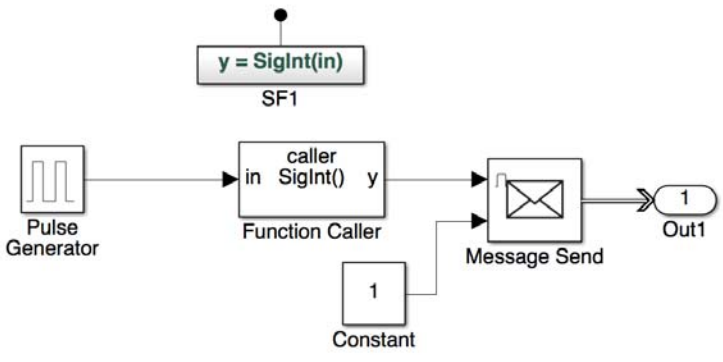


*Рис. 2*

**Блок IntServer**. Блок IntServer представляет собой автомат конечных состояний, в котором роль сигналов выполняют сами задачи (entity). Его схема представлена на рис. 3, а работа основана на переходе задачи между несколькими состояниями по некоторым правилам.

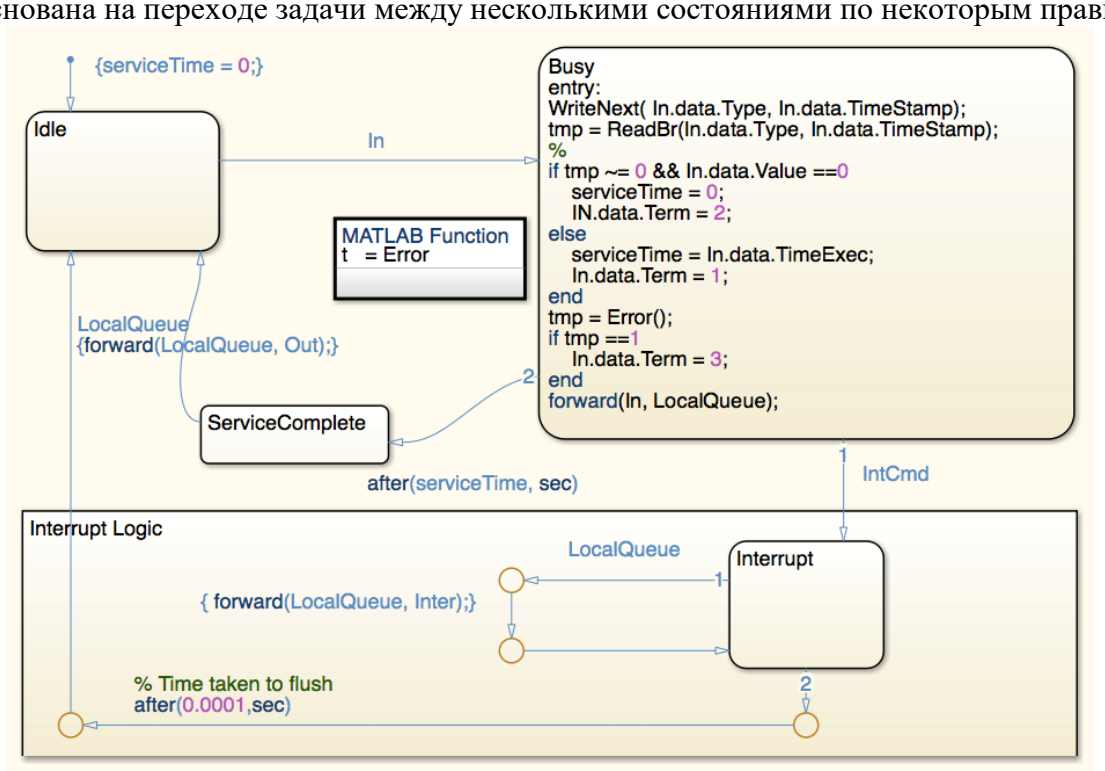


*Рис. 3*

На входе блока расположен объект Idle, на который поступает входящая задача. Из этого блока задача передается на объект Busy, в котором осуществляются все основные операции по обслуживанию задачи:

— запись параметров текущей задачи в локальную переменную Next;

— проверка параметров текущей задачи с целью определения, не была ли она выполнена ранее; в случае ее успешного выполнения на другом сервере время выполнения устанавливается равным нулю, а параметр Term, равным 2;

— вызов функции Error(), которая предназначена для имитации неисправности сервера IntServer с некоторой вероятностью; в этом случае параметр Term=3.

Далее задача поступает в локальную очередь размерности 1 — LocalQueue, и после ожидания времени обслуживания serviceTime поступает на объект ServiceComplete (обслуживание завершено) и далее на выход Out.

В случае возникновения прерывания, т.е. сигнала IntCmd, задача из локальной очереди поступает на объект Interrupt, откуда направляется на выход Inter, свидетельствующий, что задача была прервана. При этом после объекта Interrupt блок переходит к объекту Idle, т.е. возвращается в исходное состояние.

**Общая структура системы**. Общая структурная схема системы представлена на рис.4. Ее состав и функциональность детально описаны в работе [19]. Эта система состоит из следующих объектов:

— блок генераторов задач Block_of_Generators, включающий несколько генераторов с разным временем выполнения задач и интервалами их следования;

— очереди Entity_Queue, где задачи ожидают выполнения;

— блоки Subsystem1 и Subsystem3, используемые для выполнения задач, каждый из них содержит по 8 одинаковых блоков типа Chart-сервер;

— Function Block, содержащий все необходимые для работы системы функции и глобальные переменные; параметры выполненных задач записываются в глобальную переменную Br, а их количество (по типам) отражается на дисплеях Task1, Task2, Task3;

— блок ControlNumPart, предназначенный для статического или динамического разделения задач на подзадачи в зависимости от нагрузки всей системы.

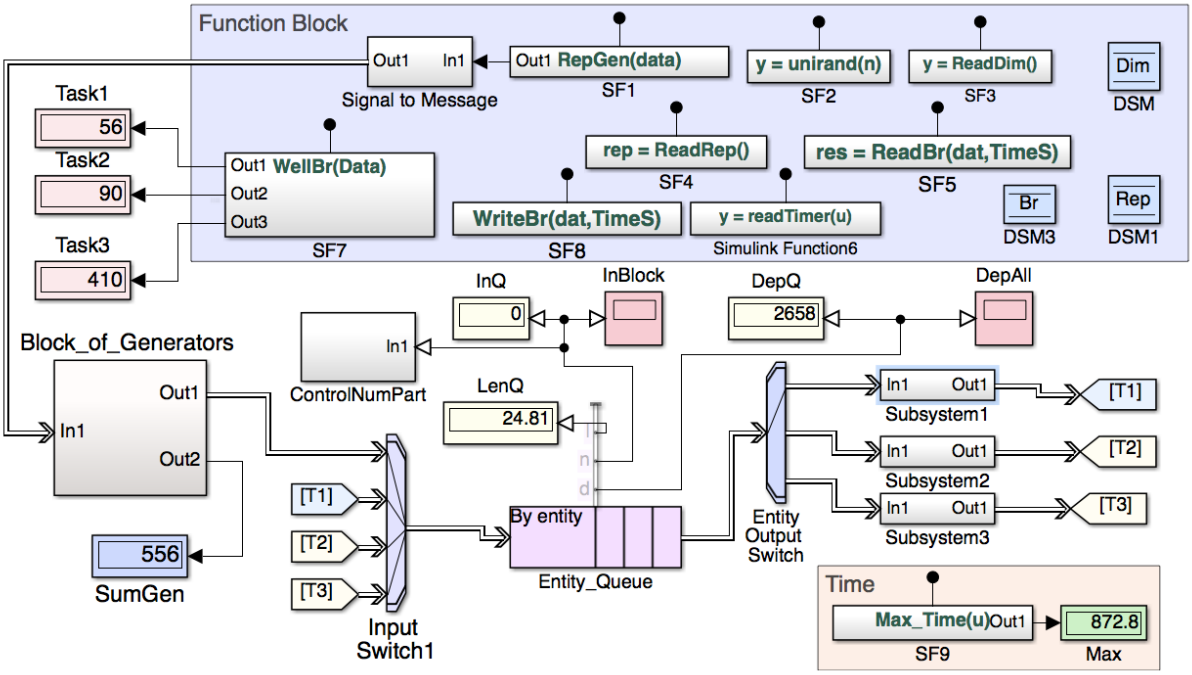


*Рис. 4*

**Результаты моделирования.** Для проведения экспериментов были использованы три типа задач:

— задачи с большим временем выполнения и малой интенсивностью следования;

— задачи среднего размера и со средним интервалом генерации;

— задачи небольшой длительностью, но с интенсивным потоком следования.

Генерация трех типов задач осуществлялась по экспоненциальному закону с параметром λ, равным 12, 8, 2 соответственно, а длительность выполнения задач составляла 96, 64 и 32 условных такта соответственно. Как видно из рис. 4, за время моделирования было выполнено 56, 90 и 410 задач первого, второго и третьего типа, при этом общее число подзадач — 2658. Эта величина обусловлена тем, что количество подзадач, на которое разбивалась задача, определялось динамически. Для этого была использована система ControlNumPart, в которой на основании оценки количества задач в очереди Entity_Queue выбиралось количество подзадач. Общее время генерации составило 1000 тактов, из которых в течение только 800 генераторы работали. Динамика изменения количества задач в очереди показана на рис. 5, где $n$ — количество тактов, $N$ — количество задач.

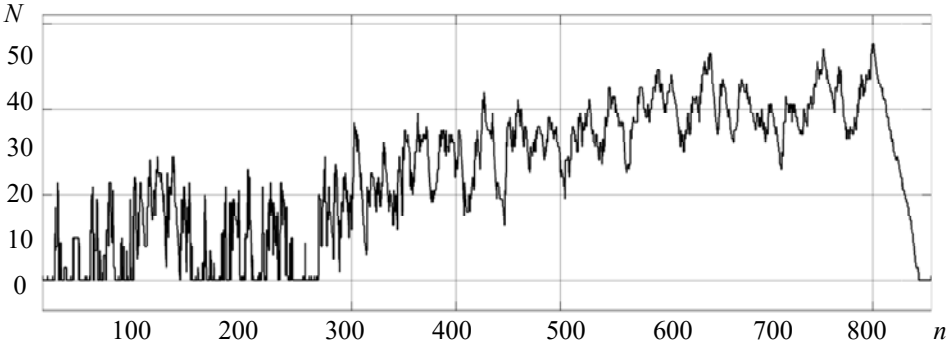


*Рис. 5*

Примеры результатов моделирования Chart-сервера показаны на рис. 1, они означают следующее:

— количество выполненных подзадач — на дисплее Display1 (101);

— количество подзадач, снятых с выполнения, поскольку решение уже было найдено, — на дисплее Display2 (2);

— количество ошибок на сервере, которое потребовало повторного выполнения подзадачи, — на дисплее Error (1);

— количество подзадач, прерванных во время выполнения, поскольку решение было найдено на другом сервере во время выполнения, — на дисплее NumInt1 (14);

— количество подзадач, ошибочно прерванных, — на дисплее NumInt2 (0).

Как видно из результатов, количество задач, прерванных во время моделирования на этом сервере, составляет примерно 10 % от общего количества выполненных задач. В общем случае количество прерванных задач колеблется от 5 до 12 %, что обеспечивает существенную экономию по времени выполнения и уменьшение нагрузки всей системы.

На основании этого можно сделать вывод о целесообразности замены сервера из пакета SimEvent на Chart-сервер, реализованный автоматом конечных состояний.

**Заключение.** Имитационное моделирование позволяет быстро и с минимальными затратами производить оценку качества различных архитектур вычислительных систем и разрабатывать системы более экономные в расходовании ресурсов.

Разумеется, рассмотренный способ моделирования предназначен для определенной категории задач, но предложенные способы моделирования можно расширить и на другие классы задач. Вопросы, связанные с влиянием количества подзадач на производительность и эффективность работы всей системы, а также вопросы эффективности выполнения задач в гетерогенной системе, где существуют группы серверов различной архитектуры и, следовательно, производительности, являются предметом дальнейших исследований.

## СПИСОК ЛИТЕРАТУРЫ

1. *Козлов М. В., Малашенко Ю. Е., Назарова И. А.* и др. Анализ режимов управления вычислительным комплексом в условиях неопределенности. М.: ВЦ РАН, 2011.
2. Sourcebook of Parallel Computing / Eds: *J. Dongarra, I. Foster*. San Francisco: Morgan Kaufmann Publ. Inc., 2003.
3. *Malashenko Yu. E., Nazarova I. A.* Control of resource intensive computations under uncertainty. I. Multiparametric model // J. Computer and Systems Sciences International. 2014. Vol. 53, N 4. P. 497—510.
4. *Golosov P. E., Kozlov M. V., Malashenko Yu. E.* et al. Analysis of computer job control under uncertainty // J. Computer and Systems Sciences International. 2012. Vol. 51, N 1. P. 49—64.
5. *Vdovin P. M., Kostenko V. A.* Algorithm for resource allocation in data centers with independent schedulers for different types of resources //J. Computer and Systems Sciences International. 2014. Vol. 53, N 6.
6. *Loginovsky O. V., Shestakov A. L., Shinkarev A. A.* Supercomputing technologies as drive for development of enterprise information systems and digital economy // Supercomputing Frontiers and Innovations. 2020. Vol. 7, N 1.
7. *Баранов А. В., Киселев А. В., Корнеев Б. В.* и др. Программный комплекс „Пирамида“ организации параллельных вычислений с распараллеливанием по данным // Тр. Междунар. суперкомпьютерной конф. и конф. молодых ученых „Научный сервис в сети Интернет: Суперкомпьютерные центры и задачи“. М.: МГУ, 2010. С. 299—302.
8. *Максимов К. В.* Эффективность использования облачных вычислений: методы и модели оценки // Прикладная информатика. 2016. Т. 11, № 1 (61).
9. *Talia D., Trunfio P., Marozzo F.* Data Analysis in the Cloud Models, Techniques and Applications. Elsevier, 2015.
10. Cloud Computing: Concepts, Technology & Architecture / Ed. *T. Erl.* Pearson, 2013.
11. *Sosinsky B.* Cloud Computing Bible. John Wiley & Sons, 2011.

12. *Chaturvedi D. K.* Modeling and Simulation of Systems Using MatLab and Simulink. CRC Press. 2009.

13. *Stallings W.* Queuening Analysis 2000 [Электронный ресурс]: <from WilliamStallings.com/StudentSupport.htm>.

14. *Baker K. R., Trietsch D.* Principles of Sequencing and Scheduling. John Wiley & Sons, 2009.

15. Simulink® User's Guide. MathWorks®, Release R2016a. Natick, MA, 2016.

16. SimEvents®, User's Guide. MathWorks®, Release R2016a, Natick, MA, 2016.

17. *Wei Li, Ramamurthy Mani, Mosterman P. J.* Extensible discrete-event simulation framework in simevent // Proc. of the Winter Simulation Conf., Dec. 2016. P. 943—954.

18. *Brandberg C., Di Natale M.* A SimEvents model for the analysis of scheduling and memory access delays in multicores // Proc. of the 13th Intern. Symp. on Industrial Embedded Systems (SIES), IEEE. 2018.

19. *Golosov P. E., Gostev I. M.* About one cloud computing simulation model // Systems of Signals Generating and Processing in the Field of on Board Communications: Conf. Proc., Moscow, 2021. DOI: 10.1109/IEEECONF51389.2021.9416100.

***Сведения об авторах***

***Павел Евгеньевич Голосов*** — канд. техн. наук; Российская академия народного хозяйства и государственной службы при Президенте РФ, факультет информационных технологий и анализа данных; декан; E-mail: golosov-pe@ranepa.ru

***Иван Михайлович Гостев*** — д-р техн. наук; ИППИ РАН, центр распределенных вычислений, ведущий научный сотрудник; E-mail: igostev@gmail.com

Поступила в редакцию
18.08.2021 г.

**Ссылка для цитирования:** *Голосов П. Е., Гостев И. М.* Имитационное моделирование серверов с прерываниями в больших многопроцессорных системах // Изв. вузов. Приборостроение. 2021. Т. 64, № 11. С. 879—886.

## SIMULATION OF SERVERS WITH INTERRUPTS IN LARGE MULTIPROCESSOR SYSTEMS

**P. E. Golosov[1], I. M. Gostev[2]**

*[1]Russian Academy of National Economy and Public Administration under the Russian President, 119571, Moscow, Russia*
*[2]Kharkevich Institute for Information Transmission Problems of the RAS, 127994, Moscow, Russia*
*E-mail: igostev@gmail.com*

The problem of managing a specialized cloud computing system performing heterogeneous resource-intensive tasks is considered when methods of the task execution can be represented as an arbitrary sorting out of a large number of options. Data-based parallelization of tasks under uncertainty is analyzed. To solve the problems of scheduling incoming input streams of tasks, the simulation is performed using the SimEvent/Simulink/MatLab software package. The functioning of the system and especially servers, represented as finite state machines, is analyzed. The peculiarity of the proposed model is the possibility of interrupting the server when an external signal appears. Using the developed server model makes it possible to improve system performance by saving time for each server to complete tasks and more efficiently distributing tasks between the servers of the entire system.

**Keywords:** cloud computing, parallel algorithms, finite state machine, simulation modeling, SimEvent, Simulink

### REFERENCES

1. Kozlov M.V., Malashenko Yu.E., Nazarova I.A. et al. *Analiz rezhimov upravleniya vychislitel'nym kompleksom v usloviyakh neopredelennosti* (Analysis of Control Modes of a Computing Complex under Conditions of Uncertainty), Moscow, 2011. (in Russ.)
2. Dongarra J., Foster I., eds., *Sourcebook of Parallel Computing,* San Francisco, Morgan Kaufmann Publ. Inc., 2003.
3. Malashenko Yu.E., Nazarova I.A. *J. Computer and Systems Sciences International,* 2014, no. 4 (53), pp. 497–510.
4. Golosov P.E., Kozlov M.V., Malashenko Yu.E. et al. *J. Computer and Systems Sciences International,* 2012, no. 1(51), pp. 49–64.

5. Vdovin P.M., Kostenko V.A. *J. Computer and Systems Sciences International,* 2014, no. 6(53).
6. Loginovsky O.V., Shestakov A.L., Shinkarev A.A. *Supercomputing Frontiers and Innovations,* 2020, no. 1(7).
7. Baranov A.V., Kiselev A.V., Korneev B.V. et al. *Nauchnyy servis v seti Internet: Superkomp'yuternyye tsentry i zadachi* (Scientific Service on the Internet: Supercomputer Centers and Problems), Proceedings of the International Supercomputer Conference and Conference of Young Scientists, Moscow, 2010, pp. 299–302. (in Russ.)
8. Maksimov K.V. *Journal of Applied Informatics,* 2016, no. 1(11). (in Russ.)
9. Talia D., Trunfio P., Marozzo F. *Data Analysis in the Cloud Models, Techniques and Applications,* Elsevier Science Publishers, Amsterdam, 2015, 150 p.
10. Erl T., ed., *Cloud Computing: Concepts, Technology & Architecture,* Pearson, 2013.
11. Sosinsky B. *Cloud Computing Bible,* John Wiley & Sons, 2011.
12. Chaturvedi D.K. *Modeling and Simulation of Systems Using MATLAB and Simulink,* CRC Press, 2009.
13. Stallings W. *Queuening Analysis,* 2000, WilliamStallings.com/StudentSupport.htm.
14. Baker K.R., Trietsch D. *Principles of Sequencing and Scheduling,* John Wiley & Sons, 2009.
15. *MathWorks. 2016. Simulink® User's Guide,* MathWorks®, Release R2016a, Natick, MA.
16. *MathWorks. 2016. SimEvents®, User's Guide,* MathWorks®, Release R2016a, Natick, MA.
17. Wei Li, Ramamurthy Mani, Pieter J. Mosterman, *Proceedings of the 2016 Winter Simulation Conference IEEE Press,* 2016, pp. 943–954, DOI:10.1109/WSC.2016.7822155.
18. Brandberg C., Di Natale M. *2018 IEEE 13th International Symposium on Industrial Embedded Systems (SIES),* 2018, pp. 1–10, DOI: 10.1109/SIES.2018.8442094.
19. Golosov P.E., Gostev I.M. *Systems of Signals Generating and Processing in the Field of on Board Communications,* 2021, DOI: 10.1109/IEEECONF51389.2021.9416100.

***Data on authors***

***Pavel E. Golosov*** — PhD; Russian Academy of National Economy and Public Administration under the Russian President, Faculty of Information Technologies and Data Analysis; Dean of the Faculty; E-mail: golosov-pe@ranepa.ru

***Ivan M. Gostev*** — Dr. Sci.; Kharkevich Institute for Information Transmission Problems of the RAS, Distributed Computing Center; Leading Researcher; E-mail: igostev@gmail.com

**For citation**: Golosov P. E., Gostev I. M. Simulation of servers with interrupts in large multiprocessor systems. *Journal of Instrument Engineering*. 2021. Vol. 64, N 11. P. 879—886 (in Russian).

DOI: 10.17586/0021-3454-2021-64-11-879-886