УНИВЕРСИТЕТ ИТМО

НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
март–апрель 2022 Том 22 № 2 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
March–April 2022 Vol. 22 No 2 http://ntv.ifmo.ru/en/
ISSN 2226-1494 (print) ISSN 2500-0373 (online)

НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК УНИВЕРСИТЕТ ИТМО
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

doi: 10.17586/2226-1494-2022-22-2-269-278
УДК 004.023

Имитационная модель облачных вычислений со спорадическим механизмом управления параллельным решением задач

Павел Евгеньевич Голосов[1]✉, Иван Михайлович Гостев[2]

[1] Российская академия народного хозяйства и государственной службы при Президенте Российской Федерации (РАНГХиГС), Москва, 119571, Российская Федерация
[2] Институт проблем передачи информации имени А.А. Харкевича РАН, Москва, 127051, Российская Федерация

[1] pgolosov@gmail.com✉, https://orcid.org/0000-0003-4313-0887
[2] igostev@gmail.com, https://orcid.org/0000-0003-4121-1894

Аннотация
Рассмотрена имитационная модель вычислительной системы, построенная в среде Simulink (SimEvent). В соответствии с теорией массового обслуживания система классифицируется как G/G/n/∞. При этом в системе присутствует множество входных потоков, их очередь бесконечна, применяются две обратные связи, отражающие ситуацию повторной обработки в случае отказа или отсутствия решения при первой попытке обработки. Архитектура системы ориентирована на параллельную обработку определенного класса задач, при этом сами задачи независимы по данным. Модель исследована при равномерно распределенных и экспоненциальных входных потоках. Исследована ситуация непрерывных потоков нескольких типов задач, для которых варьируются приоритеты и число фрагментов разбиения. Число фрагментов обуславливает степень параллелизма при выполнении задачи. Показан способ автоматического определения оптимального количества фрагментов задачи для гарантии ее выполнения в директивный срок. Предложено использование механизмов спорадического управления количеством фрагментов задач, поступающих в непрерывном потоке, и управления приоритетами каждого из фрагментов задач. Данный механизм позволит существенно ускорить прохождение задач в директивный срок выполнения. Снижена нагрузка на вычислительную систему и повышена эффективность ее работы. Применение предложенных алгоритмов существенно упростит механизмы планирования в вычислительной системе, что позволит исключить использование планировщика.

Ключевые слова
имитационное моделирование, облачные и распределенные системы, планирование, эффективность выполнения, директивный срок выполнения, спорадическое управление

Ссылка для цитирования: Голосов П.Е., Гостев И.М. Имитационная модель облачных вычислений со спорадическим механизмом управления параллельным решением задач // Научно-технический вестник информационных технологий, механики и оптики. 2022. Т. 22, № 2. С. 269–278. doi: 10.17586/2226-1494-2022-22-2-269-278

Cloud computing simulation model with a sporadic mechanism of parallel task solving control

Pavel E. Golosov[1]✉, Ivan M. Gostev[2]

[1] The Russian Presidential Academy of National Economy and Public Administration (RANEPA), Moscow, 119571, Russian Federation
[2] Institute for Information Transmission Problems of the Russian Academy of Sciences (Kharkevich Institute), Moscow, 127051, Russian Federation

[1] pgolosov@gmail.com✉, https://orcid.org/0000-0003-4313-0887
[2] igostev@gmail.com, https://orcid.org/0000-0003-4121-1894

Abstract
A simulation model of a computer system built in the Simulink (SimEvent) environment is considered. According to the queuing theory, the system is classified as G/G/n/∞. This means that there are multiple input streams in the system, their

queue is infinite, and two feedbacks are applied. These feedbacks reflect the situation of the repeated processing in case of failure or lack of a solution at the first processing attempt. The system architecture under consideration is focused on parallel processing of a certain class of tasks, while the tasks themselves are data-independent. The model is investigated for uniformly distributed and exponential input streams. The situation of continuous streams for several types of tasks is considered, for which priorities and the numbers of partitioning fragments vary. The number of fragments determines the degree of parallelism in the execution of the task. The paper shows a method for automatically determining the optimal number of task fragments to guarantee its completion within the target period. The use of sporadic control mechanisms for a number of the task fragments received in a continuous stream and the priorities managing of each of the task fragments are proposed. The proposed mechanism of the sporadic management made it possible to significantly speed up the tasks completion within the target deadline. As a result, the load on the computing system has been reduced and the efficiency of its operation has been increased. The use of the proposed algorithms significantly simplifies the scheduling mechanisms in the computer system, which allows you to exclude the scheduler.

Keywords
cloud computing, parallel algorithms, simulation modeling, scheduling, effectiveness of execution, deadline, sporadic control, SimEvent, Simulink

For citation: Golosov P.E., Gostev I.M. Cloud computing simulation model with a sporadic mechanism of parallel task solving control. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2022, vol. 22, no. 2, pp. 269–278 (in Russian). doi: 10.17586/2226-1494-2022-22-2-269-278

Введение

Рассмотрение архитектур и алгоритмов работы различных облачных сервисов, реализованных в виде вычислительных систем, показывает, что в них существует множество проблем, которые до настоящего времени не были рассмотрены [1–6].

Одна из таких проблем — механизм распараллеливания поступающих задач и выполнение их фрагментов на отдельных вычислителях. Другая проблема — обеспечение выполнения задач пользователя за гарантированное время (директивный срок окончания) [7]. В настоящей работе рассмотрены классы задач, выполнение фрагментов которых независимы по данным [8]. В последнее время количество задач такого типа существенно возросло. Это объясняется тем фактом, что увеличивается доля облачных вычислений, в которых параллельно могут выполняться поисковые задачи [9], параллельная обработка больших картографических массивов; решаются задачи шифрования и расшифрования, вычисления хеш-функций; осуществляется перевод документооборота на технологии, основанные на блокчейне и т. п. Для успешного выполнения задач, независимых по данным и обладающих большой трудоемкостью, требуется повышенная мощность вычислительных систем, а значит, увеличение накладных расходов на оборудование, помещения и электроэнергию. Данные условия приводят пользователей к необходимости перехода на облачные вычисления. При таком переходе у пользователей возникает ряд проблем, которые обусловлены спецификой типа задач и необходимостью получить решение за заданное директивное время. Для выполнения данных условий задачу распараллеливают на подзадачи, и, чтобы исключить замедление процесса вычисления [10] в системе, отключают режим вытеснения.

При этом вопрос о том, на какое количество подзадач необходимо разбивать задачу, остается открытым.

Предмет исследования

С точки зрения теории массового обслуживания, поступившая задача должна быть выполнена за некоторое директивное время, независимо от того, на сколько частей ее можно разделить. Очевидно, что если задачу можно разделить на подзадачи и выполнять одновременно на нескольких устройствах, то время ее выполнения сократится пропорционально количеству устройств (ветвей обработки). Заметим, что для такого типа задач интервал времени выполнения может составлять от 0 до полного директивного времени решения. При разделении задачи на подзадачи и получении решения в одной из них, все остальные подзадачи должны быть сняты с вычислений, так как решение уже получено.

В результате архитектура такой облачной вычислительной системы (ОВС) должна быть ориентирована на параллельное выполнение без прерывания (вытеснения) каждой из подзадач. При получении решения одной из подзадач выполнение оставшихся подзадач прекращается. Заметим, что в ОВС поступает поток задач с разным директивным временем выполнения, при этом все требования соблюдать необходимо одновременно. Каждая такая задача имеет приоритет, пропорциональный ее стоимости. Следовательно, ресурсы ОВС должны быть распределены между множеством задач (разделенных на подзадачи) по известным правилам.

Из вышеизложенного требуется определить:

— на какое количество подзадач необходимо разделять каждую задачу для повышения эффективности выполнения как самой задачи, так и всей ОВС;
— как распределять приоритеты для выполнения всех задач в заданное для них директивное время.

Данные требования, на основании теории массового обслуживания, не могут быть выполнены в реальной ситуации.

Это связано с тем, что успешно выполненная подзадача должна воздействовать на всю ОВС для снятия с вычислений остальных подзадач. С другой стороны, любая система имеет известную надежность, т. е. любой вычислительный узел имеет вероятность отказов, возникновение которых приводит к необходимости повторной загрузки подзадачи на выполнение. Также может возникнуть ситуация, когда задание было выполнено полностью, но решение не было найдено.

В этом случае задача требует повторного выполнения с расширенным/измененным набором начальных данных.

Таким образом, параллелизм выполнения задач, наличие сложных обратных связей в ней, а также неизвестный закон поступления задач в ОВС и неопределенное время обработки задач разных типов не позволяют применить для решения этих проблем, как аппарат теории массового обслуживания, так и классические методы планирования управления заданиями. В общем виде такого типа задачи в терминах теории массового обслуживания, которые в нотации Кендалла имеют вид G/G/n/∞ (с множеством входных потоков с произвольным распределением, бесконечной очередью и несколькими обратными связями), практически не имеют формального описания.

Для решения таких задач можно применить методы имитационного моделирования [11–13]. С этой целью необходимо сконструировать архитектуру вычислительной системы и в процессе моделирования обеспечить различные законы поступления входных задач и механизмы их обслуживания. В качестве инструментария для решения поставленной задачи используем MATLAB/Simulink с пакетом SimEvent [14, 15].

В настоящей работе поставлена задача исследования возможности управления ОВС при непрерывном потоке задач с целью уточнения минимизации времени исполнения поставленных задач, для гарантии обеспечения директивных сроков и повышения эффективности работы всей системы. Для достижения этих целей разработаны следующие спорадические механизмы:

— изменение количества подзадач в зависимости от темпа и типа поступающих задач в ОВС, что позволит повысить эффективность работы всей системы и увеличить ее пропускную способность;
— управление приоритетами фрагментов выполняемых задач для обеспечения гарантированного их выполнения за директивное время и повышения эффективности работы всей системы.

Архитектура облачной вычислительной системы

Структура и алгоритм функционирования имитационной модели ОВС были рассмотрены в работах [16, 17]. Имитационная модель разработана специально для решения задач, независимых по данным, показана на рис. 1.

Имитационная модель включает в себя:
— подсистему генераторов задач — Generators;
— очередь задач — Entity_Queue;
— четыре обслуживающие подсистемы Subsystem, выполняющие роль исполнительных систем, каждая из которых содержит восемь подсистем, представляющих собой серверные узлы, основанные на автомате конечных состояний и представленные на рис. 2;
— систему управления количеством подзадач — ControlNumPart;
— блок анализа среднего времени вычисления каждого типа задач — Avg;
— блок Function Block, содержащий глобальные переменные, дисплеи и необходимые функции для работы модели.

Приведем описание работы отдельных блоков имитационной модели.

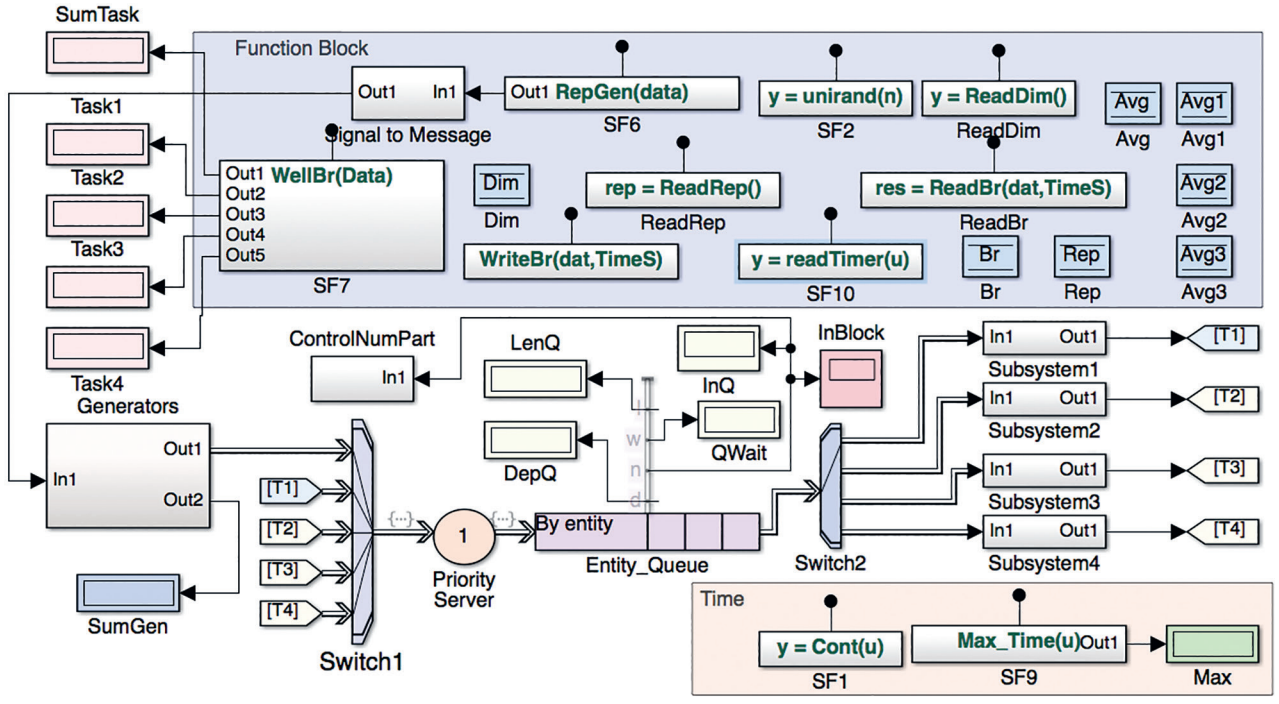


Рис. 1. Общая блок-схема имитационной модели
Fig. 1. Simulation model general block diagram

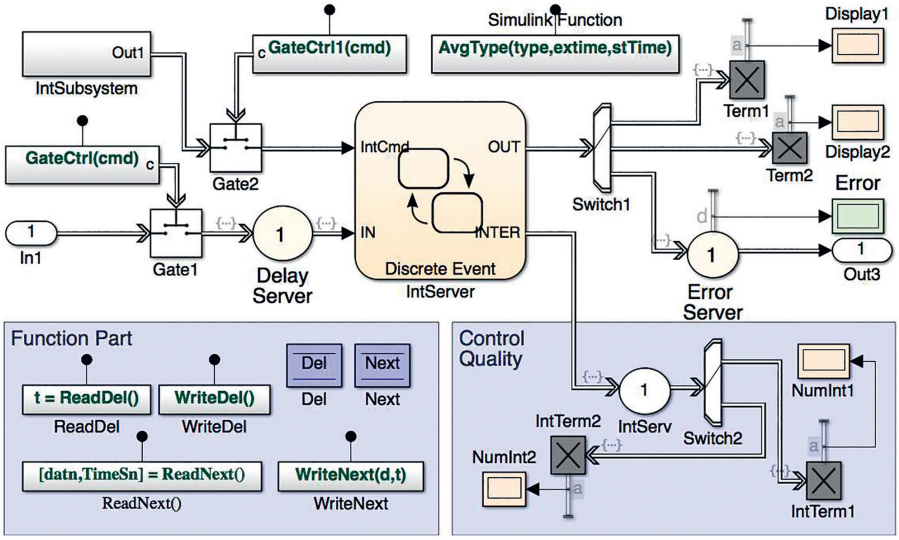


Рис. 2. Схема Chart сервера

Fig. 2. Chart server diagram

Блок генераторов. Блок генераторов Generators был модифицирован таким образом, чтобы генерировать задачи в течение времени меньшего, чем общее время моделирования. Данное условие связано с оценкой общей эффективности работы ОВС при гарантированном выполнении всех поступивших задач. В данной работе генераторы задач работают в интервале [0, 500], в то время как общее время моделирования составляет 700 условных секунд.

Серверный блок. Разработанный Chart-сервер основан на автомате конечных состояний (Discret 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).

Отличие предложенного сервера от полного описания, приведенного в работе [17], заключается в добавлении ключа Gate2, который открывается только во время работы сервера и необходим для синхронизации работы сервера и блока прерываний. Структура сервера показана на рис. 2. Данное отличие важно при возникновении ситуации, когда подзадача выполнена, и сгенерирован сигнал прерывания для снятия этой подзадачи со счета, но одновременно на сервер поступила подзадача другого типа. Тогда за счет сигнала прерывания подзадача будет снята со счета, хотя и не была выполнена.

Блок управления количеством подзадач ControlNumPart. Для управления количеством подзадач в систему встроен модуль ControlNumPart, который показан на рис. 3. Входной сигнал на данный блок поступает со статистического выхода очереди Entity Queue, показывающего количество задач в очереди. Далее он направляется на три компаратора со значениями уставок в 64, 128, 256. Выходы компараторов попадают на переключатели, которые определяют конечное значение выходного сигнала блока ControlNumPart, служащее входным параметром функции DimPar(). Функция DimPar() задает значение глобальной переменной системы Dim, величина которой в зависимости от входных значений функции DimPar() имеет значения: 24, 20, 18, 16. Таким образом, значение переменной

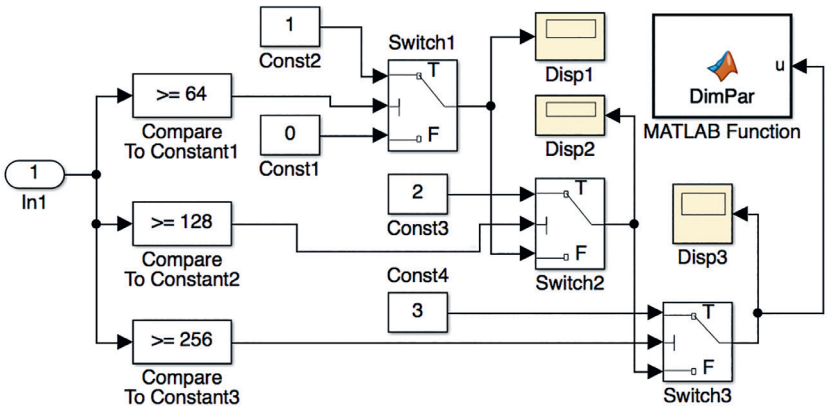


Рис. 3. Схема блока управления количеством подзадач

Fig. 3. Diagram of the control unit for the number of subtasks

Dim определяет количество подзадач, на которое будет разделяться генерируемая задача. Начальное значение глобальной переменной Dim равно 24. T и F — состояния переключателей; u — управляющий сигнал функции DimPar (принимает значения из множества [1–3]).

Таким образом, алгоритм работы блока ControlNumPart заключается в регулировании общего количества подзадач в ОВС в зависимости от длины очереди. Чем больше длина очереди, тем на меньшее количество подзадач будут разбиваться поступающие задачи.

Дополнительные блоки. В представленной модели в ОВС добавлены следующие компоненты:

— AvgControl — блок, предназначенный для вычисления среднего времени выполнения по каждому типу задачи, показан на рис. 4, *a*. Значение времени выполнения задачи считывается со значений глобальных переменных Avg1–Avg4, вычисленных усредняющими функциями Avg_Time1()–Avg_Time4(), и далее поступает на дисплеи Avg Type1–Avg Type4. Эти глобальные переменные инициализируются при помощи функций Avg Type(), вызываемыми из каждого блока Chart Server.

— Сервер PriorityServ – предназначен для изменения значения приоритетов задач в зависимости от их среднего времени выполнения. Программный код управления приоритетами показан на рис. 4, *b*. При стремлении времени выполнения к директивному, ее приоритет существенно повышается (в системе Simulink наивысший приоритет равен 0). Таким образом, увеличивая приоритет задачи, добиваемся ускорения прохождения задачи через очередь, тем самым обеспечивая ее выполнение в директивное время.

— Блок Time (рис. 1) предназначен для расчета максимального времени вычисления всех задач, которое отображается на дисплее Max. Для этого используется функция Max_Time().

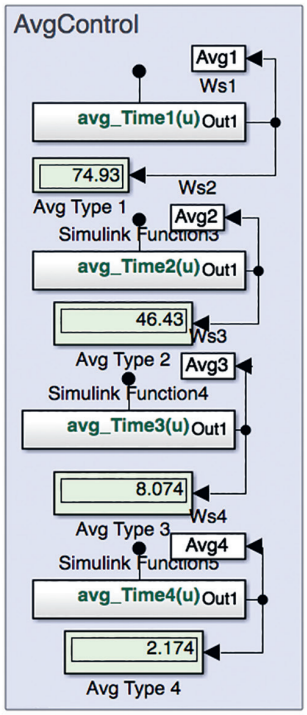
Результаты моделирования

Процесс моделирования выполнен в несколько этапов для двух типов входных потоков – экспоненциального и равномерного. Время выполнения четырех типов задач с помощью блока AvgControl неизменно и составило 160, 128, 64 и 32 условных единиц. При экспоненциальном законе поступления задач в ОВС выбраны следующие значения параметра λ равные 12, 10, 6, 2, при равномерном – интервал генерации задач составил: [0, 20], [0–15], [0–15], [0–8].

Результаты моделирования приведены в таблице, где приняты следующие обозначения: T_{max} и TQ_{max} — максимальное время выполнения всех задач и нахождения задач в очереди; Q_{max} — максимальное количество задач в очереди; LenQ — среднее значение числа задач в очереди; WaitQ — среднее время ожидания задачи в очереди; Avg1, Avg2, Avg3, Avg4 — среднее время выполнения задач 1–4 типов. Значения 1, 4, 8, 16, 24 соответствуют количеству подзадач. Столбцы var1 и var2 содержат данные количества задач с включенной системой управления и со спорадическим режимом управления приоритетами.

На первом этапе моделирования приоритеты всех задач равны 30.

a



b

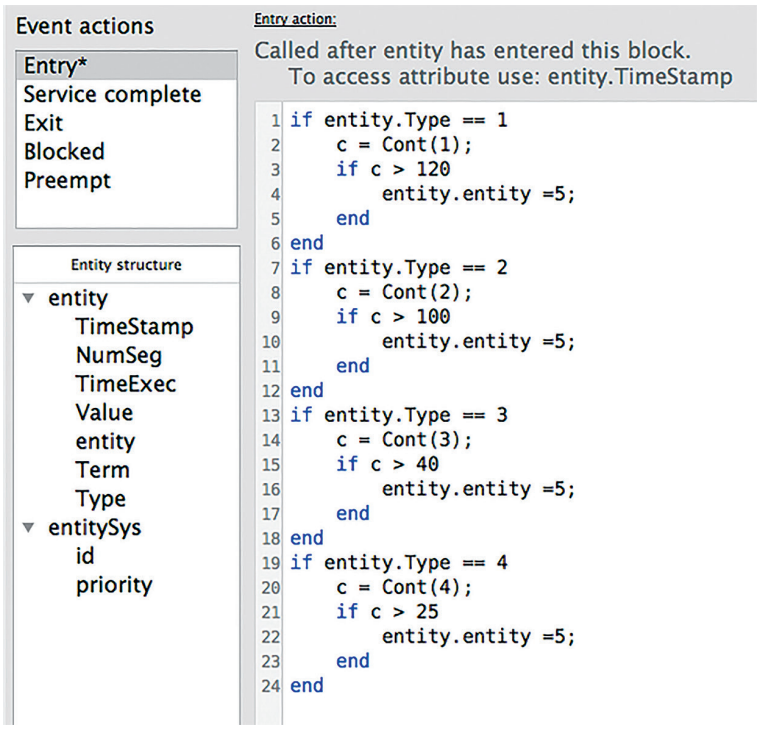


Рис. 4. Схема блока AvgControl (*a*) и код спорадического управления приоритетами задач в сервере PriorServ (*b*)

Fig. 4. Diagram of the AvgControl block (*a*) and the code for the sporadic task priority management in the PriorServ server (*b*)

Значения времени выполнения задач и параметры законов генерации выбраны таким образом, чтобы при отсутствии деления задач на части система была мало нагружена, и среднее значение количества задач в очереди равно 1. Такое состояние ОВС позволяет выдерживать директивные сроки выполнения каждой из задач, однако, работа всей системы при такой нагрузке будет малоэффективна. Также отсутствуют гарантии, что директивные сроки выполнения каждой из задач будут соблюдены. С увеличением количества подзадач очередь будет увеличиваться, при этом теоретически время выполнения каждой из задач должно сокращаться пропорционально количеству частей задачи. Однако с ростом очереди возникает и конкуренция за ресурсы (серверы) системы. Вследствие этого не все директивные сроки выполнения задач будут соблюдаться, как видно из таблицы.

На рис. 5 (кривая *1*) показано количество задач в очереди при фиксированном количестве фрагментов (24).

При подключении системы регулирования количества фрагментов (блок ControlNumPart) размер очереди задач уменьшился (рис. 5, кривая *2*), но директивные сроки выполнения задач — не выдержаны (таблица, var1). При переводе очереди в приоритетный режим работы и использовании сервера PriorityServ, для при-

Таблица. Результаты моделирования для экспоненциального и равновероятного законов входного потока задач

Table. Simulation results for exponential and equiprobable laws of the input stream of tasks

Параметры моделирования	Закон входного потока задач	Количество подзадач					Var1	Var2
		1	4	8	16	24		
T_{max}	экспоненциальный	582,2	612,8	623,5	595,9	578	600	569,7
	равновероятный	615,4	627,2	628,5	609	571,3	565	574
TQ_{max}	экспоненциальный	496,3	587,4	619,2	588	573,6	592	539
	равновероятный	499	601,2	615,2	605,2	570,7	563	563,9
Q_{max}	экспоненциальный	3	231	56,3	859	1186	907	373
	равновероятный	1	195	461	823	847	573	325
LenQ	экспоненциальный	0,07	75,46	199	270,7	35,8	270	110
	равновероятный	0	72,8	176,6	291,9	267,6	201,8	134,6
WaitQ	экспоненциальный	0	27,64	38,42	24,82	21,34	20,87	7,468
	равновероятный	0	38,2	47,2	38,38	22,10	20,73	13,09
Avg_1	экспоненциальный	110,6	106,6	127,1	84,42	71,38	91,46	74,93
	равновероятный	120,6	114,8	114,5	102,6	69,79	60,52	82,45
Avg_2	экспоненциальный	122,8	99,36	123,6	65,47	64,02	71,36	46,43
	равновероятный	123,1	119,1	124,1	111,1	77,93	71,36	16,35
Avg_3	экспоненциальный	58,46	78,67	124,2	91,36	77,21	96,14	8,074
	равновероятный	61,56	108	116,1	106,2	72,96	63,64	5,097
Avg_4	экспоненциальный	31,8	97,17	68,9	90,98	76,46	95,94	2,74
	равновероятный	16,25	106,2	118,2	106,7	74,87	64,04	0,9162

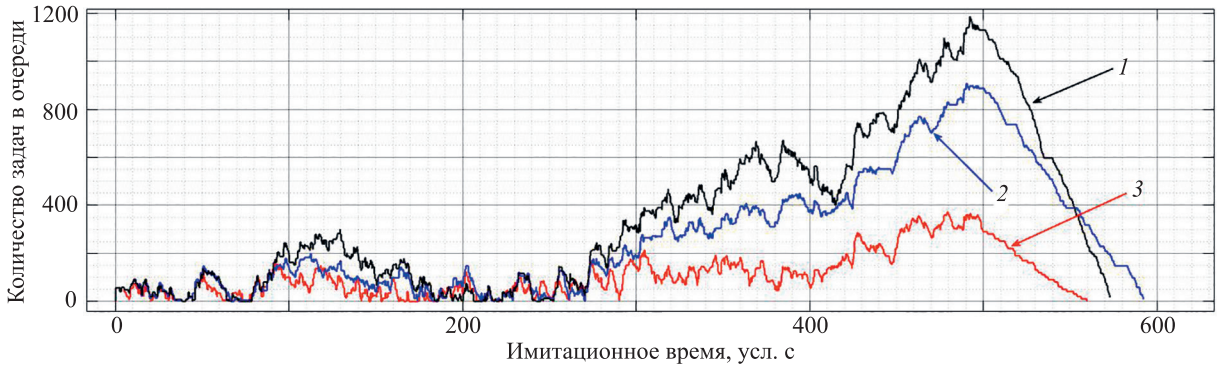


Рис. 5. Поведение очереди задач при различных режимах моделирования: статическое деление задачи на 24 части (кривая *1*); динамическое разделение управления без приоритета (кривая *2*) и с приоритетом (кривая *3*)

Fig. 5. The behavior of the task queue for various simulation modes: static task sectioning into 24 parts (curve *1*); dynamic control sectioning without priority (curve *2*) and with priority (curve *3*)

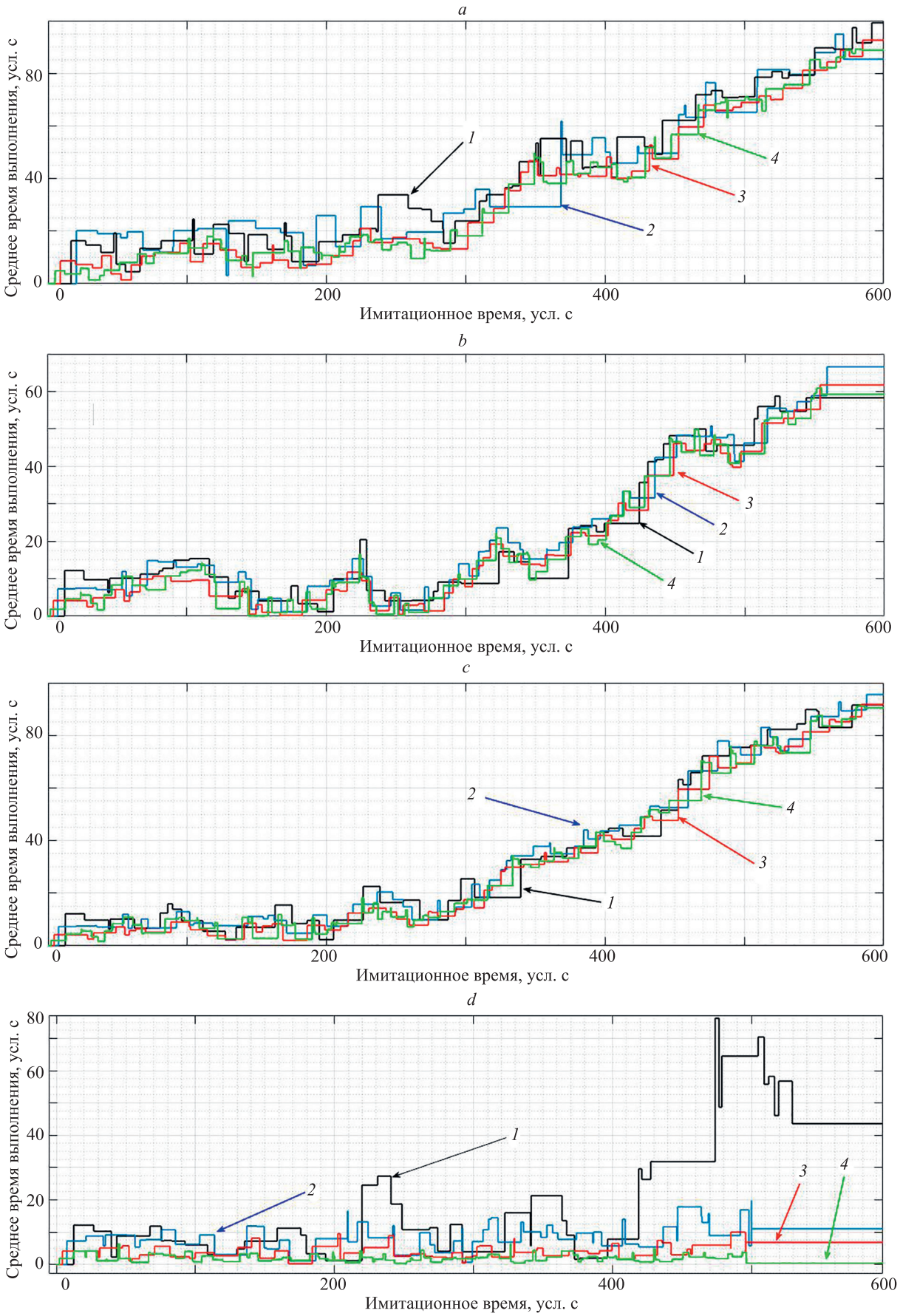


Рис. 6. Средние значения времени выполнения типов задач при различных режимах работы облачной вычислительной системы: 1 типа (кривая *1*), 2 типа (кривая *2*), 3 типа (кривая *3*), 4 типа (кривая *4*)

Fig. 6. Average task execution time for different operating modes of a cloud computing system: type 1 (curve *1*), type 2 (curve *2*), type 3 (curve *3*), type 4 (curve *4*)

оритетов 50, 40, 30 и 20 (задачи типа 1–4 соответственно) — очередь значительно уменьшилась (рис. 5, кривая *3*) и директивные сроки выдержаны с большим запасом (таблица, var2).

При этом в таблице показаны средние значения времени, которые не могут гарантировать выполнение директивных сроков. Покажем поведение параметров моделирования в различных режимах (рис. 6). На рис. 6, *a* продемонстрировано поведение среднего времени выполнения при отсутствии регулирования и разбиении задачи на 8 подзадач, на рис. 6, *b* — на 24 подзадачи. Также показано поведение среднего времени выполнения при динамическом разбиении задачи без управления приоритетами (рис. 6, *c*) и для задач при динамическом разбиении и приоритетном спорадическом управлении (рис. 6, *d*).

В результате видно, что только случай на рис. 6, *d* обеспечивает заданное директивное время выполнения с большим запасом. При этом размеры очереди существенно уменьшились (рис. 5, кривая *3*), что свидетельствует о более эффективной работе всей системы.

Обсуждение и выводы

Анализ результатов моделирования предложенной имитационной модели ОВС показал следующее:

— при увеличении числа фрагментов задачи происходит существенное возрастание размера очереди, что не дает выигрыша в гарантированном выполнении задач за директивное время. Этот факт объясняется конкуренцией задач в очереди и практически 100 % загрузкой «процессоров». Увеличение значения приоритета для одного из типов задач приводит к несоблюдению директивного времени выполнения для других, что является неприемлемым;
— вариативность числа подзадач при небольших значениях (1–16) приводит только к увеличению очереди, повышению нагрузки системы и, как следствие, к увеличению директивных сроков выполнения одного или двух типов задач. При увеличении количества подзадач (более 16) происходит некоторое уменьшение времени выполнения всех задач, что свидетельствует о снижении общей нагрузки системы. Однако и в этом случае отсутствуют гарантии по срокам выполнения всех типов задач. При включении спорадического управления количеством подзадач очередь значительно уменьшается, но также отсутствует гарантия выполнения всех задач за директивное время;
— при включении спорадического управления приоритетами происходит некоторое «упорядочивание» расположения задач в очереди, что приводит, с одной стороны, к ее уменьшению, а с другой стороны обеспечивает гарантированное выполнение задач в директивное время;
— сравнение результатов моделирования (таблица) показывает, что, несмотря на тот факт, что экспоненциальное распределение дает меньшую нагрузку на систему, тем не менее наличие «тяжелых хвостов» иногда приводит к генерации нескольких одинаковых типов задач практически в небольшом отрезке времени. И, несмотря на небольшую нагрузку всей системы, этот факт приводит к нарушению гарантированного времени выполнения для одного из типов задач. При равномерном распределении такого эффекта не наблюдается;
— поскольку результаты экспериментов со спорадическим управлением показали, что средние времена ожидания для всех задач существенно меньше, чем директивные сроки, то в такой системе возможно волюнтаристически придавать некоторой задаче повышенный приоритет, для того чтобы минимизировать время ее выполнения.

Предложенная схема управления прохождением задач в облачных системах имеет ряд преимуществ по сравнению с существующими классическими схемами управления. Во-первых, в такой системе отсутствует планировщик, который при больших входных потоках и разного типа задачах будет требовать дополнительного времени на расчет оптимальной стратегии обслуживания. Во-вторых, как видно из таблицы и рис. 6, *d*, получается большой запас по гарантированному директивному времени обслуживания, что позволяет дополнительно нагрузить систему, и тем самым повысить ее эффективность. В-третьих, имеется возможность на основе спорадического управления внести «волюнтаризм» в управление системой, когда в силу каких-либо причин необходимо в кратчайшие сроки выполнить некоторую задачу, не нарушая работу всей системы.

Таким образом, предложенная имитационная модель позволяет провести исследования по наиболее эффективному управлению облачными сервисами с минимальными затратами. Разумеется, в реальном случае необходимо изучать типы поступающих задач, законы распределения входных потоков, производительность серверов и параметры времени на передачу информации между ними.

В предложенной модели это не было сделано для упрощения ее работы, так как такие задержки вводятся элементарно добавлением в тракт прохождения заданий обычного сервера из пакета SimEvent. При этом время обслуживания этого сервера будет определять задержки и может быть задано любой необходимой функцией.

Заключение

Рассмотрена имитационная модель, предназначенная для решения определенного круга задач, который постоянно расширяется. Модель может быть модернизирована введением элементов, характеризующих такие ресурсы, как оперативная и дисковая память. Возможны варианты с внесением различного типа задержек, обусловленных передачей данных в сетях и т. п.

Литература

1. Erl T., Puttini R., Mahmood Z. Cloud Computing: Concepts, Technology & Architecture. Pearson, 2013. 528 p.
2. Sudheer M.S., Reddy K.G., Sree R.K., Raju V.P. An effective analysis on various scheduling algorithms in cloud computing // Proc. of the International Conference on Inventive Computing and Informatics (ICICI). 2017. P. 931–936. https://doi.org/10.1109/ICICI.2017.8365274
3. Hu J., Gu J., Sun G., Zhao T. A scheduling strategy on load balancing of virtual machine resources in cloud computing environment // Proc. of the 3rd International Symposium on Parallel Architectures, Algorithms and Programming (PAAP). 2010. P. 89–96. https://doi.org/10.1109/PAAP.2010.65
4. Brandberg C., Di Natale M. A SimEvents model for the analysis of scheduling and memory access delays in multicores // Proc. of the 13th International Symposium on Industrial Embedded Systems (SIES). 2018. P. 8442094. https://doi.org/10.1109/SIES.2018.8442094
5. Gomathi B., Krishnasamy K. Task scheduling algorithm based on Hybrid Particle Swarm Optimization in cloud computing environment // Journal of Theoretical and Applied Information Technology. 2013. V. 55. N 1. P. 33–38.
6. Rossi F.D., Calheiros R.N., De Rose C.A.F. A terminology to classify artefacts for cloud infrastructure // Research Advances in Cloud Computing. Springer, 2017. P. 75–91. https://doi.org/10.1007/978-981-10-5026-8_4
7. Tomar A., Pant B., Tripathi V., Verma K.K., Mishra S. Improving QoS of cloudlet scheduling via effective particle swarm model // Lecture Notes in Electrical Engineering. 2022. V. 768. P. 137–150. https://doi.org/10.1007/978-981-16-2354-7_13
8. Golosov P.E., Kozlov M.V., Malashenko Yu.E., Nazarova I.A., Ronzhin A.F. Analysis of computer job control under uncertainty // Journal of Computer and Systems Sciences International. 2012. V. 51. N 1. P. 49–64. https://doi.org/10.1134/S1064230711060062
9. Цымблер М.Л. Параллельный алгоритм поиска диссонансов временного ряда для многоядерных ускорителей // Вычислительные методы и программирование. 2019. Т. 20. № 3. С. 211–223. https://doi.org/10.26089/NumMet.v20r320
10. Голосов П.Е., Гостев И.М. Об имитационном моделировании функционирования операционной системы с вытесняющим планированием // Телекоммуникации. 2021. № 8. С. 2–22. https://doi.org/10.31044/1684-2588-2021-0-8-2-22
11. Li W., Mani R., Mosterman P.J. Extensible discrete-event simulation framework in SimEvents // Proc. of the 2016 Winter Simulation Conference (WSC). 2016. P. 943–954. https://doi.org/10.1109/WSC.2016.7822155
12. Tareghian S., Bornaee Z. A new approach for scheduling jobs in cloud computing environment // Cumhuriyet University Faculty of Science Science Journal (CSJ). 2015. V. 36. N 3. P. 2499–2506.
13. Kecskemeti G., Hajji W., Tso F.P. Modelling low power compute clusters for cloud simulation // Proc. of the 25th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP). 2017. P. 39–45. https://doi.org/10.1109/PDP.2017.33
14. MathWorks. 2016. Simulink® User's Guide. MathWorks®, Release R2016a, Natick, MA.
15. MathWorks. 2016. SimEvents®, User's Guide. MathWorks®, Release R2016a, Natick, MA.
16. 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, Conference Proceedings. 2021. P. 9416100. https://doi.org/10.1109/IEEECONF51389.2021.9416100
17. Голосов П.Е., Гостев И.М. Имитационное моделирование серверов с прерываниями в больших многопроцессорных системах // Известия вузов. Приборостроение. 2021. Т. 64. № 11. С. 879–886. https://doi.org/10.17586/0021-3454-2021-64-11-879-886

References

1. Erl T., Puttini R., Mahmood Z. *Cloud Computing: Concepts, Technology & Architecture*. Pearson, 2013, 528 p.
2. Sudheer M.S., Reddy K.G., Sree R.K., Raju V.P. An effective analysis on various scheduling algorithms in cloud computing. *Proc. of the International Conference on Inventive Computing and Informatics (ICICI)*, 2017, pp. 931–936. https://doi.org/10.1109/ICICI.2017.8365274
3. Hu J., Gu J., Sun G., Zhao T. A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. *Proc. of the 3rd International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*, 2010, pp. 89–96. https://doi.org/10.1109/PAAP.2010.65
4. Brandberg C., Di Natale M. A SimEvents model for the analysis of scheduling and memory access delays in multicores. *Proc. of the 13th International Symposium on Industrial Embedded Systems (SIES)*, 2018, pp. 8442094. https://doi.org/10.1109/SIES.2018.8442094
5. Gomathi B., Krishnasamy K. Task scheduling algorithm based on Hybrid Particle Swarm Optimization in cloud computing environment. *Journal of Theoretical and Applied Information Technology*, 2013, vol. 55, no. 1, pp. 33–38.
6. Rossi F.D., Calheiros R.N., De Rose C.A.F. A terminology to classify artefacts for cloud infrastructure. *Research Advances in Cloud Computing*. Springer, 2017, pp. 75–91. https://doi.org/10.1007/978-981-10-5026-8_4
7. Tomar A., Pant B., Tripathi V., Verma K.K., Mishra S. Improving QoS of cloudlet scheduling via effective particle swarm model. *Lecture Notes in Electrical Engineering*, 2022, vol. 768, pp. 137–150. https://doi.org/10.1007/978-981-16-2354-7_13
8. Golosov P.E., Kozlov M.V., Malashenko Yu.E., Nazarova I.A., Ronzhin A.F. Analysis of computer job control under uncertainty. *Journal of Computer and Systems Sciences International*, 2012, vol. 51, no. 1, pp. 49–64. https://doi.org/10.1134/S1064230711060062
9. Zymbler M.L. A parallel discord discovery algorithm for time series on many-core accelerators. *Numerical Methods and Programming*, 2019, vol. 20, no. 3, pp. 211–223. (in Russian). https://doi.org/10.26089/NumMet.v20r320
10. Golosov P.E., Gostev I.M. On some simulation models of operating system functioning with preemptive scheduling. *Telecommunications*, 2021, no. 8, pp. 2–22. (in Russian). https://doi.org/10.31044/1684-2588-2021-0-8-2-22
11. Li W., Mani R., Mosterman P.J. Extensible discrete-event simulation framework in SimEvents. *Proc. of the 2016 Winter Simulation Conference (WSC)*, 2016, pp. 943–954. https://doi.org/10.1109/WSC.2016.7822155
12. Tareghian S., Bornaee Z. A new approach for scheduling jobs in cloud computing environment. *Cumhuriyet University Faculty of Science Science Journal (CSJ)*, 2015, vol. 36, no. 3, pp. 2499–2506.
13. Kecskemeti G., Hajji W., Tso F.P. Modelling low power compute clusters for cloud simulation. *Proc. of the 25th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, 2017, pp. 39–45. https://doi.org/10.1109/PDP.2017.33
14. *MathWorks. 2016. Simulink® User's Guide*. MathWorks®, Release R2016a, Natick, MA.
15. *MathWorks. 2016. SimEvents®, User's Guide*. MathWorks®, Release R2016a, Natick, MA.
16. 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, Conference Proceedings*, 2021, pp. 9416100. https://doi.org/10.1109/IEEECONF51389.2021.9416100
17. Golosov P.E., Gostev I.M. Simulation of servers with interrupts in large multiprocessor systems. *Journal of Instrument Engineering*, 2021, vol. 64, no. 11, pp. 879–886. (in Russian). https://doi.org/10.17586/0021-3454-2021-64-11-879-886

Авторы

Голосов Павел Евгеньевич — кандидат технических наук, декан, Российская академия народного хозяйства и государственной службы при Президенте Российской Федерации (РАНГХиГС), Москва, 119571, Российская Федерация, SC 55035594300, https://orcid.org/0000-0003-4313-0887, pgolosov@gmail.com

Authors

Pavel E. Golosov — PhD, Dean, The Russian Presidential Academy of National Economy and Public Administration (RANEPA), Moscow, 119571, Russian Federation, SC 55035594300, https://orcid.org/0000-0003-4313-0887, pgolosov@gmail.com

Гостев Иван Михайлович — доктор технических наук, ведущий научный сотрудник, Институт проблем передачи информации имени А.А. Харкевича РАН, Москва, 127051, Российская Федерация, SC 6602252369, https://orcid.org/0000-0003-4121-1894, igostev@gmail.com

Ivan M. Gostev — D.Sc., Leading Researcher, Institute for Information Transmission Problems of the Russian Academy of Sciences (Kharkevich Institute), Moscow, 127051, Russian Federation, SC 6602252369, https://orcid.org/0000-0003-4121-1894, igostev@gmail.com

Статья поступила в редакцию 30.11.2021
Одобрена после рецензирования 03.02.2022
Принята к печати 17.03.2022

Received 30.11.2021
Approved after reviewing 03.02.2022
Accepted 17.03.2022