Интеллектуальные системы

DOI 10.24412/2221-2574-2023-2-29-39 УДК 004.023

АНАЛИЗ ЭФФЕКТИВНОСТИ ИМИТАЦИОННЫХ МОДЕЛЕЙ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ С ИСПОЛЬЗОВАНИЕМ ЭЛЕМЕНТОВ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Гостев Иван Михайлович
доктор технических наук, ведущий научный сотрудник Центра распределённых вычислений, Институт проблем передачи информации им. А.А. Харкевича Российской академии наук (ИППИ РАН)[1].
E-mail: igostev@gmail.com

Голосов Павел Евгеньевич
кандидат технических наук, декан факультета информационных технологий и анализа данных Института ЭМИТ, Российская академия народного хозяйства и государственной службы при Президенте РФ (РАНХиГС)[2].
E-mail: golosov-pe@ranepa.ru

[1]*Адрес:* 127051, Российская Федерация, г. Москва, Большой Каретный переулок, д. 19, стр. 1.
[2]*Адрес:* 117571, Российская Федерация, г. Москва, проспект Вернадского, д. 82.

Аннотация: Исследуется имитационная модель распределённой (облачной) специализированной вычислительной системы, построенной в среде Simulink/SimEvent. Такую систему в теории массового обслуживания классифицируют как G/G/n/∞. Она характеризуется множеством входных потоков с бесконечной очередью, имеет две обратные связи, отражающие ситуации повторной обработки в случае отказа или отсутствия решения при первой попытке обработки. Архитектура системы ориентирована на параллельное выполнение определённого класса задач, разделяемых на независимые по данным подзадачи. Основным критерием функционирования такой системы является прохождение всех поступивших задач за директивный срок их выполнения. Поскольку аналитическое описание такой модели невозможно, то для определения её пропускной способности был разработан принципиально новый принцип оценки производительности. Основной целью настоящей работы является определение продуктивности и эффективности её работы при равномерно распределённых и экспоненциальных входных потоках задач.

Ключевые слова: имитационное моделирование, облачные и распределённые системы, планирование, эффективность выполнения, директивный срок выполнения, спорадическое управление.

Введение

Диспетчеризация ресурсов в облачной вычислительной системе существенно отличается от управления ресурсами единичного компьютера. Планировщики распределённых систем (облачные) обычно рассматривают такую систему как большой пул ресурсов, к которым они имеют полный доступ [1–3]. Фактически управление выполнением отдельного задания осуществляется независимо. При этом нагрузка всей системы, приоритеты, варианты расписания, не учитываются. Отсутствие координирующих механизмов приводит как к возникновению узких мест, так и к неполному использованию ресурсов. С другой стороны, конечные пользователи не имеют сведений о времени получения решения или сервисных возможностях, что ведёт к потере качества обслуживания.

Алгоритмы классических планировщиков, как правило, построены либо на минимизации времени отклика (системы реального времени), либо на максимизации общей загрузки ресурсов (разделения времени) [4–6]. А поскольку цель планирования заключается в улучшении состояния системы в целом, то требования, предъявляемые отдельными потребителями, практически не учитываются и время

выполнения задач пользователя не регламентировано.

Для повышения эффективности работы оборудования и минимизации времени обслуживания необходимо корректировать стратегию планирования, так, чтобы, с одной стороны, учесть интересы пользователей, приоритеты задач и время их выполнения, с другой иметь информацию о состоянии и распределении ресурсов системы для общей оптимизации её функционирования.

В настоящей работе рассматриваются принципы оценки эффективности специализированной вычислительной системы (СВС) при различных механизмах управления облачной высокопроизводительной, выполняющей ресурсоёмкие задачи, относящиеся к специальному классу методы, выполнения которых можно определить как случайный перебор с неизвестным исходом [7, 8]. Получение решения такой задачи основано на переборных алгоритмах и сводится к поиску фрагмента с наперёд заданными свойствами в большом массиве исходных данных. Такой массив, как правило, состоит из отдельных, неделимых, одинаковых по размеру, содержательно значимых фрагментов.

Каждая задача становится решённой, как только в некотором фрагменте данных удаётся выявить уникальный элемент. Задачи бывают разных видов. Это и поиск графического объекта на фрагментах карты для его распознавания, и поиск в Интернете некоторого текста по заданному фрагменту, и задачи шифрации и дешифрации.

Для таких задач существует только верхняя временная оценка для величины вычислительных затрат, определяемая на основе полного анализа (перебора) возможных значений. Однако в общем случае неизвестно, имеет ли решение поступающая задача или нет. Исходя из этого, в такой системе возникает неопределённость планирования, связанная с моментами и темпом поступления заявок, временем их выполнения, а также с потенциальной возможностью получения решения за некоторое время. И ещё большей проблемой является оценка производительности и эффективности использования имеющегося оборудования.

В работах [8, 9] такой специализированный класс задач именуется как citu-задания, citu-задачи; от английского термина: computation – intensive – task – under – uncertainty. Следует отметить целый ряд существенных особенностей, присущих данному классу: выполнение всех citu-задач должно происходить в режиме реального времени. Они допускают распараллеливание по данным и являются ресурсоёмкими. Их решения должны быть получены как можно быстрее и в отдельности, и в совокупности. Там же [9] вводится понятие директивный срок окончания (ДСО) — это установленный диспетчером и согласованный с пользователем момент календарного времени, до наступления которого citu-задача должна быть решена. В настоящей имитационной модели диспетчер и пользователь исключены из участия в процессе вычислений. Поэтому мы будем использовать термин «директивное время вычислений (ДВВ)», которое назначается как один из параметров задачи и означает, что она должна быть выполнена за время, не превышающее значение этого параметра с момента её генерации.

В отечественной практике существуют примеры успешных реализаций таких систем управления ресурсоёмкими вычислениями, например, программный комплекс «Пирамида» [10, 11], который имеет иерархическую древовидную архитектуру и обеспечивает высокую надежность вычислений за счёт дублирования. Дублирование может составлять половину от общего количества работ, тем самым в два раза снижая эффективность вычислений. Многолетняя практика показывает, что такие массово-параллельные системы диспетчеризации, построенные с использованием архитектуры с интерфейсом mpi [12–14], становятся малопригодными для крупных облачных систем, а особенно для СВС.

Рассмотренный в [8] комплекс CORSAR позволяет избежать потери производительно-

сти при увеличении числа и мощности вычислительных ресурсов. В нём ожидается практически линейный рост кривой «эффективность — быстродействие». В комплексе CORSAR, согласно концептуальной схеме, планирование работ осуществляется на основании результатов, полученных с помощью математических моделей. Весь процесс управления СВС разбивается контрольными точками на отдельные этапы — плановые периоды или операционные окна, а распараллеливание и распределение работ происходит с учётом текущей производительности ресурсов. Однако участие в таком комплексе оператора, согласовывающего свои действия с пользователем, приводит к существенному ухудшению эффективности его работы.

Постановка задачи

Управление выполнением задач в облачной СВС представляет собой сложную многокритериальную задачу со слабо формализуемыми начальными условиями и связанную с необходимостью распределения задач по вычислительным узлам. При этом необходимо выполнение ряда формальных условий:

- все задачи должны выполняться в рамках заданного ДВВ для каждой задачи;
- необходима возможность выполнения одной задачи параллельно на нескольких вычислителях (например, подзадач, независимых по данным);
- функционирование системы при множестве входных потоков подзадач с различными законами их поступления (равномерный, экспоненциальный, пуассоновский и т. д.);
- загрузка такого комплекса по возможности должна быть максимальной. То есть желательно, чтобы все вычислители были загружены на 100%;
- для повышения эффективности работы системы, в случае нахождения решения задачи в одной из подзадач, выполнение всех остальные должно быть прекращено;
- в случае отсутствия решения некоторой задачи при заданных условиях она должна быть автоматически перезапущена с изменёнными начальными условиями;
- при отказе одного из вычислителей при решении некоторой подзадачи, она должна быть выполнена повторно.

Таким образом, в терминах теории массового обслуживания имеем многоканальную сеть массового обслуживания с обратными связями с:

- множеством входных потоков задач с произвольными законами поступления;
- множеством обслуживающих приборов (серверов);
- дисциплина обслуживания постоянна для каждого типа задач, то есть время обслуживания каждой задачи определено при её генерации;
- все задачи после генерации сразу разделяются на некоторое произвольное число подзадач равной длины (длина определяется типом задачи), то есть данную систему можно отнести к пакетной [15].
- при разбиении задачи на подзадачи подразумевается, что решение существует в одной из подзадач (случайно выбранной), в остальных время выполнения фиксировано и равно T/n, где T — максимальное время выполнения задачи, а n — число фрагментов (ветвей) деления задачи;
- время обслуживания успешных подзадач определяется по равновероятному закону от 0 до T/n.

Данная сеть имеет две обратные связи:

- первая обратная связь заключается в повторном поступлении ***всей*** задачи на выполнение в случае отсутствия решения при некоторых входных параметрах (то есть поступление её в систему как новую задачу того же типа);
- вторая обратная связь возникает при отказе сервера (отказы задаются на основе равновероятного закона). В этом случае (именно) подзадача поступает на выполнение в общую

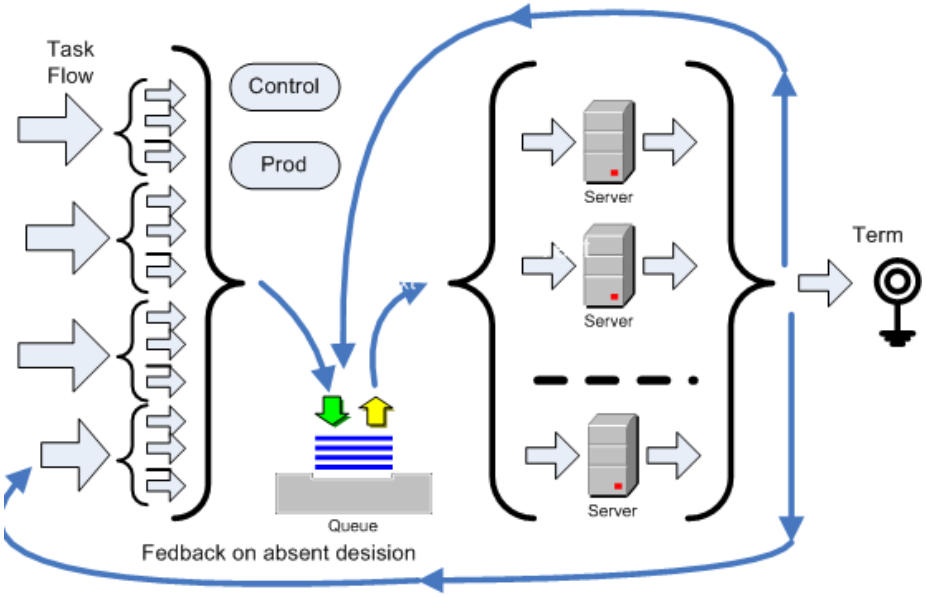

Рис. 1. Общая блок схема имитационной модели

очередь подзадач. Эта обратная связь существует для всех серверов.

В случае выполнения одной из подзадач — все остальные подзадачи (этой задачи) снимаются с выполнения (выполнение подзадачи прерывается на любом сервере и на любом этапе выполнения). Количество подзадач в каждой новой сгенерированной задаче может варьироваться в зависимости от длины очереди.

Дисциплина обслуживания задач в очереди может быть выбрана любая из следующих: FIFO, LIFO, по приоритетам (определённым по некоторым параметрам самих задач). Причём приоритеты могут изменяться в зависимости от изменения состояний системы.

Основная цель построения такой сети массового обслуживания — решение особых классов задач в облачной (распределенной) СВС с максимальной эффективностью использования оборудования и гарантированным временем выполнения задач каждого типа.

Исходя из того факта, что поступление подзадач в обратных связях не подчиняется экспоненциальному закону, а имеет равновероятное распределение, данную систему можно отнести к классу G/G/n/∞. Кроме того, прерывание работающих подзадач в случае успешного получения решения в одной из них можно расценивать, как внесение в систему еще одной обратной связи с отрицательными заявками [16–18]. Такие заявки появляются в системе, когда в одной из подзадач найдено решение, а другие подзадачи еще выполняются или ждут в очереди. Сама отрицательная заявка не выполняется, но снимает с выполнения некоторое количество других заявок.

Все вышеприведённые условия приводят к пониманию, что построение аналитического описания такой системы не представляется возможным. А, следовательно, невозможно построить аналитическое описание производительности и эффективности работы таких систем. На основании этого необходимо разработать принцип оценки производительности и эффективности работы таких систем.

Архитектура системы

Структура и алгоритм функционирования такой имитационной модели облачной системы уже были неоднократно рассмотрены в [16–18]. Она была разработана специально для решения задач вышеописанного типа и представлена на рис. 1. Кратко она включает:

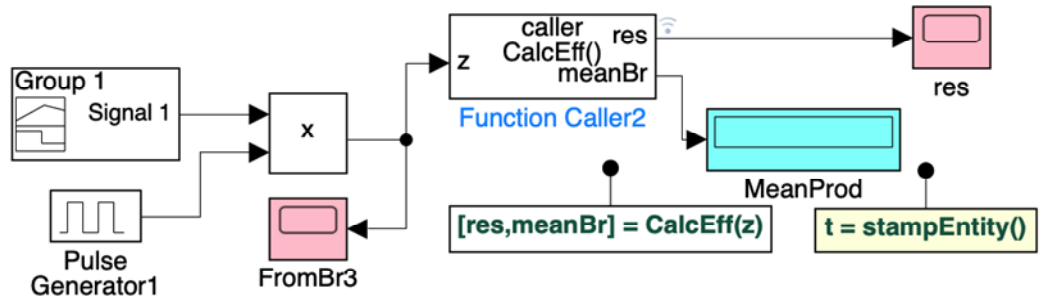

Рис. 2. Схема блока расчета продуктивности

- поток задач (Task Flow), каждая из которых динамически разделяется на некоторое число подзадач;
- очередь задач — Queue;
- 32 сервера, выполняющих роль исполнительных систем, основанные на автомате конечных состояний;
- подсистему управления (Control) количеством подзадач и их приоритетами для оптимизации работы всей системы;
- блок Prod, предназначенный для вычисления продуктивности и эффективности работы модели (рассмотрена ниже);
- множество дисплеев, индикаторов и других вспомогательных блоков, необходимых для контроля работы модели и позволяющие производить различные оценки её работы.

Некоторые пояснения

В настоящей работе необходимо ввести некоторые определения, без которых невозможно объяснить и проанализировать результаты моделирования. Под понятием продуктивности мы будем понимать сумму ДВВ всех задач, выполнившихся за некоторый промежуток времени. Для оценки качества работы модели предположим, что имеется N серверов. Пусть мы имеем на входе системы поток задач, каждая из которых имеет единичное время обслуживания. Тогда в единицу времени можно обслужить N задач, а за промежуток времени t — количество решённых задач будет $N{\cdot}t$. На основании этого будем определять среднюю ***продуктивность*** данной имитационной модели, как сумму ДВВ решенных задач за некоторый промежуток времени. В работе использовался промежуток времени, равный 300 единицам условного времени (далее будем считать их секундами). Тогда, например, для рассматриваемой модели теоретическая средняя продуктивность при 32 серверах за 300 секунд будет составлять 9600 условных единиц.

С другой стороны, эффективность работы системы с жестко заданными (вышеприведёнными) требованиями будет определяться отношением числа всех невыполненных задач к общему числу выполненных за ДВВ на всём интервале моделирования. Будем рассчитывать такую эффективность в процентах. Выполнение полностью всех задач в ДВВ за некоторый период моделирования будем считать 100%-ой эффективностью работы. Если, например, за этот период из 100 задач 10 не уложились в ДВВ, то эффективность работы такой системы будет только 90%. Необходимо ещё раз заметить, что в настоящей работе временной интервал моделирования увеличен более чем в три раза. Это было сделано для измерения вышеприведённых параметров в некотором установившемся равновесии — при отсутствии возрастания задач в очереди.

Блок оценки эффективности Prod

Для определения вышеприведённых характеристик работы модели был разработан блок Prod, представленный на рис. 2. Он состоит из генератора Group1, который определяет временной интервал, в пределах которого будет измеряться продуктивностью и эффектив-

ность. Импульсный генератор PulseGenerator вырабатывает сигналы, по которым происходят измерения. Блока вызова функции Function Caller, который по сигналу генератора вызывает функцию CalcEff(), которая и вычисляет производительность системы (более подробно механизм изложен ниже). Дисплей Res показывает, начиная с 300-й секунды, величину рассчитанных значений продуктивности на смещаемом по временной оси интервале времени в 300 секунд. Вычисление продуктивности осуществляется через промежуток в 10 секунд функцией CalcEff(), как сумму длин ДВВ всех выполнившихся задач на этом интервале. Индикатор MeanProd отражает итоговое среднее значение продуктивности на всём интервале моделирования.

Механизм функционирования системы

Генерируемые задачи в данной модели могут иметь произвольный закон следования. Мы использовали два варианта закона генерации — равномерный и экспоненциальный. После генерации задача разделяется на подзадачи, одна из которых случайным образом отмечается как содержащая успешное решение. Количество подзадач определяется динамически подсистемой Control в зависимости от различных факторов, рассмотренных в [18]. Далее все задачи поступают в очередь Queue на выполнение. К ним добавляются подзадачи с серверов, на которых произошел отказ. Из очереди подзадачи поступают на сервера для выполнения и далее на терминатор для окончания работы. Перед выполнением каждой подзадачи производится проверка наличия решения в другой ветви этой задачи. В положительном случае подзадача снимается с выполнения. В процессе выполнения могут возникать следующие ситуации.

- Подзадача выполнилась до конца и решение не найдено. В этом случае она уничтожается.
- Подзадача в процессе выполнения получает решение. В этом случае происходит запись о её успешном выполнении.
- Подзадача в процессе выполнения обнаружила, что для этой задачи уже получено решение. Подзадача немедленно уничтожается.
- Все подзадачи выполнены, но решение не найдено. В этом случае происходит повторный запуск задачи с другим набором.

В данной работе для управления работой всей системы используется два механизма. Первый механизм основан на интеллектуальном агенте, базирующемся на механизме спорадического контроля. Во втором механизме второй интеллектуальный агент управляет поведением системы на основе варьирования приоритетов некоторых параметров исполняющихся задач. Оба агента динамически регулируют количество задач в очереди Queue и очерёдность их прохождения через неё. Исходя из этого, использование термина «интеллектуальный агент» в настоящей работе можно считать обоснованным [19].

В данной модели два интеллектуальных агента получают статистическую информацию о состоянии модели и управляют состояниями имитационной модели для обеспечения её эффективной работой. Более подробно механизмы функционирования этих агентов были рассмотрены в [18]. Таким образом, не имея никакой априорной информации о том, как выполнять поставленные перед моделью требования, система в процессе своего функционирования сама, без какой либо целевой функции, определяет некоторый оптимальный режим своей работы. В дополнении к предыдущим работам целью настоящей работы является изложение методики оценки производительности имитационной модели и её эффективности работы и результатов моделирования.

Результаты моделирования

Для моделирования были выбраны четыре типа задач со временем выполнения, равным 160, 100, 64, 32 секундам. В качестве периода моделирования был использован отрезок в 0–1500 секунд. Такой интервал был выбран,

Таблица 1. Экспоненциальное распределение

N ser	M	S	Eff	InQ	LenQ	DepQ	QWait	Tm
1	10135	137 (0,0,56,81)	82,64	0	285,2	11856	35,79	1497
2	10369	123 (0,0,48,75)	84,41	0	201,8	15627	19,16	1498
3	11789.9	0	100	0	26,68	23115	1,617	1407

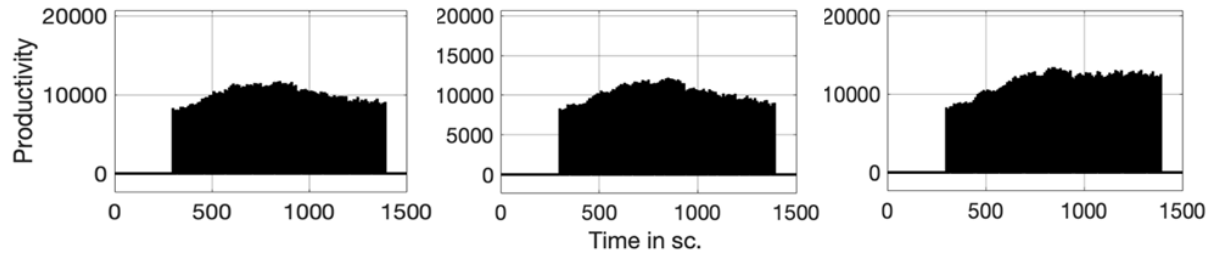

Рис. 3. Динамика изменения продуктивности во времени. Слева направо: без управления; со спорадическим управлением; спорадическое и приоритетное управление.

исходя из того, что очередь, начиная с 1200 секунды перестаёт возрастать, и наступает некоторый «равновесный» режим. При этом все генераторы работали только в промежутке времени 0–1400. Были проведены две серии экспериментов: с отсутствием всякого управления (только очередь FIFO); с применением спорадического управления; со спорадическим контролем и управлением на основе приоритетного обслуживания по типу «самые короткие вперёд». В первой серии использовался экспоненциальный закон генерации задач, во второй — равновероятный закон. И в первой, и второй серии рассчитывалась продуктивность работы модели и её эффективность.

В первой серии (при экспоненциальном законе распределения) средние интервалы времени при генерации задач были λ = 14, 10, 7, 4. То есть задачи с большим временем выполнения генерировались реже, чем короткие. Во всех вариантах этой серии были сгенерированы 91, 132, 203, 358 задач по типам, соответственно. Результаты моделирования сведены в таблицу 1, где: M — общая средняя продуктивность системы, рассчитанная на интервале 300–1400 секунд; S — количество задач, не выполнившихся за ДВВ; Eff — эффективность работы системы в процентах; InQ — количество задач, оставшихся в очереди после окончания времени моделирования (1500); LenQ — средняя длина очереди в подзадачах; DepQ — общее количество обслуженных подзадач; QWait — среднее время ожидания подзадачи в очереди; Tm — время окончания выполнения всех сгенерированных задач.

На графике на рис. 3 показаны изменения значений продуктивности. Эти значения вычислялись на скользящем интервале времени длиной 300 сек., начинаются со значения 300 сек. с интервалом в 10 сек. Для первых двух экспериментов значения продуктивности достигают своего максимального значения в районе 800-й секунды, а затем начинают плавно уменьшаться пропорционально росту очереди. В третьем случае, когда включены все управляющие воздействия, из рисунка видно, что количество подзадач в очереди не увеличивается и продуктивность системы остаётся практически постоянной.

Таблица 2. Равномерное распределение

N ser	M	S	Eff	InQ	LenQ	DepQ	QWait	Tm
1	7281	607 (63,111,189,244)	50,45	6177	3299	12226	241,1	1500
2	6547	623 (68,116,188,251)	49,14	3503	1923	7527	225,6	1500
3	18973,8	0	100	0	53,53	35602	2,107	1407

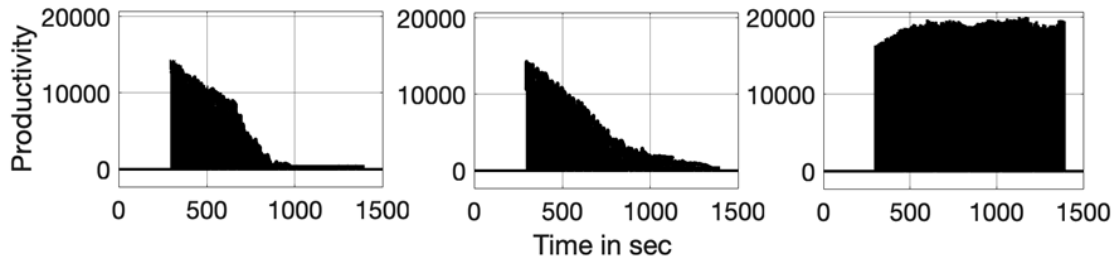

Рис. 4. Динамика изменения продуктивности во времени для равномерного закона следования генерируемых задач. Слева направо: без управления; со спорадическим управлением; спорадическое и приоритетное управление.

Во второй серии экспериментов использованы те же длительности всех типов задач с равновероятным законом распределения входных потоков на интервалах: [0–18], [0–12], [0–8], [0–6], соответственно. Получаем еще более значительные результаты, сведённые в таблицу 2. Для первых двух экспериментов по окончании времени моделирования в очереди остались 6177 и 3503 подзадач, при этом размер очереди достигал около 7000 и 5000 подзадач, соответственно, которые система полностью не смогла выполнить. Количество отказов (невыполнение ДВВ) составило 607 и 623 задачи.

При таких размерах очереди продуктивность падает практически до 0. То есть, при количестве подзадач в очереди больше 1000 система перестаёт выполнять свои функции. В третьем эксперименте продуктивность плавно возрастает практически до значений 20000 и остается практически постоянной. Очередь подзадач в этом случае остается в пределах 200–300 подзадач. То есть у системы ещё имеется некоторый резерв в вычислительной мощности.

Обсуждение

Проведённые серии экспериментов в дополнении к [17] ещё раз показали высокую эффективность управлением прохождения задач в СВС. Использованная методика оценки производительности и эффективности показала реальные возможности данной модели, которые невозможно определить другими способами.

Здесь необходимо ещё внести дополнительные пояснения относительно того факта, почему в экспериментах практическая продуктивность существенно превышает теоретическую (20000 по сравнению с 9600). Для объяснения этого необходимо вспомнить особенности механизма функционирования серверов и обработки прерываний в них. Задачи после генерации разделяется на подзадачи, число которых задается динамически на основании работы системы спорадического контроля. При этом понятно, что положительное решение задачи находится в одной из подзадач (независимость по данным). Также необходимо напомнить, что

все неуспешные подзадачи будут выполняться за время T/n, где T — время выполнения всей задачи (ДВВ), а n — количество подзадач. Успешная подзадача будет выполняться с равной вероятностью в интервале [0–T/n]. Как только такой фрагмент задачи будет выполнен, все остальные подзадачи снимаются с вычислений на серверах. Если же они стоят в очереди, то они, как только приходят на сервер, без вычислений заканчивают работу. Таким образом, по сравнению с теоретической продуктивностью, практическая должна возрастать, исходя из того, что время выполнения успешной подзадачи в среднем будет T/2n. В то время как теоретическая продуктивность будет вычисляться в среднем как T/n, что можно наблюдать на представленных результатах.

Как в первой серии экспериментов, так и во второй, показано, что количество обработанных подзадач при включении управления практически удваивается. При этом количество отказов остается равным 0. Мы не стали задавать параметры входного потока, по которым система с данными типами задач подошла бы к своему пределу производительности, поскольку это не является целью настоящей работы. Однако представленные результаты свидетельствуют о высокой эффективности разработанного управления.

В результатах второй серии экспериментов при использовании равномерного распределения при генерации задач входного потока видно, что система при отсутствии управления существенно перегружена и не отвечает заявленным требованиям. Включение подсистем управления показывает, что в этом режиме её пропускная способность возрастает многократно, а продуктивность увеличивается примерно в два раза относительно теоретической.

Заключение

На основании проведённого исследования можно сделать следующие выводы:

- разработанный принцип определения производительности СВС позволяет ввести некоторую «меру», на основании которой можно проводить сравнение производительности и эффективности работы таких систем;
- численные измерения производительности показали, что данный тип управления на основании интеллектуальных агентов является высокоэффективным;
- поскольку данная имитационная модель построена на основе сетей массового обслуживания, то её архитектуру и механизмы управления заданиями можно рекомендовать в качестве основы моделей в других направлениях исследований, таких, как коммуникационные и телекоммуникационные сети, системы управления производством, логистические цепи и многие другие направления.

Литература

1. *Wang L., Ranjan R., Chen J., Benatalla B.* Cloud Computing: Methodology, Systems, and Applications. Boca-Raton, USA: CRC Press, 2017. 844 p.

2. *Kavis M.* Architecting the Cloud: Design Decisions for Cloud Computing Service Models (SaaS, PaaS, and IaaS). Hoboken, USA: John Wiley & Sons, Inc., 2014. 224 p.

3. *Jackson K., Goessling S.* Architecting Cloud Computing Solutions. Birmingham, UK: Packt, 2018. 378 p.

4. *Stallings W.* Operating Systems: Internals and Design Principles. 9th ed. Upper Saddle River, USA: Pearson, 2017. 768 p.

5. *Silberschatz A., Galvin P.B., Gagne G.* Operating System Concepts, 10th ed. Hoboken, USA: John Wiley & Sons, Inc., 2018. 944 p.

6. *Brucker P.* Scheduling Algorithms, 5th ed. Berlin, Germany: Springer-Verlag, 2007. 378 p.

7. *Малашенко Ю.Е., Назарова И.А.* Модель управления разнородными вычислительными заданиями на основе гарантированных оценок времени выполнения // Изв. РАН. ТиСУ. 2012. № 4. С. 29–38.

8. *Купалов-Ярополк И.К., Малашенко Ю.Е., Назарова И.А.* и др. Модели и программы для системы управления ресурсоемкими вычислениями. М.: ВЦ РАН, 2013. 72 с.

9. *Malashenko Yu.E., Nazarova I.A.* Controlling Computationally Intensive Heterogeneous Computational Tasks with Directive Deadlines // Journal of Computer and Systems Sciences International. 2012. Vol. 51. No. 5. Pp. 628–635.

10. *Баранов А.В., Киселев А.В., Корнеев В.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.

11. *Забродин А.В., Левин В.К., Корнеев В.В.* Массово-параллельные системы МВС-100 и МВС-1000 // Сб. тр. научн. сессии МИФИ, 2000. М.: МИФИ, 2000. Т. 2. С. 194–195.

12. *Антонов А.С.* Параллельное программирование с использованием технологии MPI: Учебное пособие. М. Изд-во МГУ, 2004. 71с.

13. *Стронгин Р.Г.* (ред.) Высокопроизводительные параллельные вычисления на кластерных системах // Сборник материалов Седьмой Международной конференции-семинара. Изд-во Нижегородского гос. университета, 2007. 443 стр.

14. Воеводин В.В. Параллельные вычисления БХВ-Петербург, 2002. 608 с.

15. *Кудрявцева Е.Н., Росляков А.В.* Базовые принципы и перспективы использования теории сетевого исчисления (network calculus) // Инфокоммуникационные технологии. 2013. Т. 11. №3. Pp. 34–39.

16. *Голосов П.Е., Гостев И.М.* Имитационное моделирование серверов с прерываниями в больших многопроцессорных системах // Известия вузов. Приборостроение. 2021. Т. 64. No 11. С. 879–886.

17. *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.

18. *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.

19. *Рассел С., Норвинг П.* Искусственный интеллект: современный подход. М.: Вильямс, 2006. 1408 с.

Поступила 17 апреля 2023 г.

English

PERFORMANCE ANALYSIS OF SIMULATION MODELS FOR CLOUD COMPUTING USING ARTIFICIAL INTELLIGENCE ELEMENTS

Ivan Mikhailovich Gostev — Grand Dr. in Engineering, Leading Researcher, Institute for Information Transmission Problems of the Russian Academy of Sciences (Kharkevich Institute)[1].
E-mail: igostev@gmail.com

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

[1]*Address:* 127051, Russian Federation, Moscow, Bolshoy Karetny Lane, 19/1.
[2]*Address:* 117571, Russian Federation, Moscow, Vernandsky Ave., 82.

Abstract: The simulation model of distributed (cloud) special-purpose computing system built in Simulink/SimEvent environment is investigated. Such system is classified as G/G/n/∞ in queuing theory. It features multiple input streams with an infinite queue, has two feedbacks involving situations of re-processing in case of failure or lack of solution at the first processing try. The system architecture is focused on parallel execution of certain class tasks divided into data-independent subtasks. The main criterion for such system functioning is tasking of all received tasks within the scheduled time of their execution. Since an analytical description of such a model is impossible then a fundamentally new principle to estimate performance was developed to define its throughput. The main purpose of this research work is to define producing capacity and efficiency of its operation with evenly distributed and exponential task inputs. Measurements were performed in steady state unlike in preceding work when the queue stops increasing. Producing capacity variations of such system were examined under various options for sporadic control of both the number of part tasks incoming in continuous stream and priority control of each part task. The research work investigates efficiency assessment principle of special-purpose computing systems (SCS) of this type for various options for system control.

Keywords: simulation modeling, cloud and distributed systems, planning, execution efficiency, scheduled execution time, sporadic control.

References

1. *Wang L., Ranjan R., Chen J., Benatalla B.* Cloud Computing: Methodology, Systems, and Applications. Boca-Raton, USA: CRC Press, 2017. 844 p.
2. *Kavis M.* Architecting the Cloud: Design Decisions for Cloud Computing Service Models (SaaS, PaaS, and IaaS). Hoboken, USA: John Wiley & Sons, Inc., 2014. 224 p.
3. *Jackson K., Goessling S.* Architecting Cloud Computing Solutions. Birmingham, UK: Packt, 2018. 378 p.
4. *Stallings W.* Operating Systems: Internals and Design Principles. 9th ed. Upper Saddle River, USA: Pearson, 2017. 768 p.
5. *Silberschatz A., Galvin P.B., Gagne G.* Operating System Concepts, 10th ed. Hoboken, USA: John Wiley & Sons, Inc., 2018. 944 p.
6. *Brucker P.* Scheduling Algorithms, 5th ed. Berlin, Germany: Springer-Verlag, 2007. 378 p.
7. *Malashenko Yu.E., Nazarova I.A.* Management model of heterogeneous computational tasks based on guaranteed estimates of execution time. Izvestiya RAN. TiSU. 2012. No. 4. Pp. 29–38.
8. *Kupalov-Yaropolk I.K., Malashenko Yu.E., Nazarova I.A.,* etc. Models and programs for the resource-intensive computing management system. Moscow: VC RAS, 2013. 72 p.
9. *Malashenko Yu.E., Nazarova I.A.* Controlling Computationally Intensive Heterogeneous Computational Tasks with Directive Deadlines. Journal of Computer and Systems Sciences International. 2012. Vol. 51. No. 5. Pp. 628–635.
10. *Baranov A.V., Kiselev A.V., Korneev V.V., etc.* The Pyramid software package for organizing parallel computing with data parallelization. International. Supercomputer Conference and Conference of young scientists " Scientific Services & Internet: Supercomputing Centers and Applications ". Moscow: MSU, 2010. Pp. 299–302.
11. *Zabrodin A.V., Levin V.K., Korneev V.V.* Massively parallel systems MVS-100 and MVS-1000. Iss. of MEPhI, 2000. M.: MEPhI, 2000. Vol. 2. Pp. 194–195.
12. *Antonov A.S.* Parallel programming using MPI technology: Textbook. Moscow: Izdatel'stvo MGU, 2004. 71 p.
13. *Strongin R.G. (ed.)* High-performance parallel computing on cluster systems. Proceedings of the Seventh International Conference-Seminar. Publishing House of NNSU, 2007. 443 p.
14. *Voevodin V.V.* Parallel computing BHV-Peterburg, 2002. 608 p.
15. *Kudryavtseva E.N., Roslyakov A.V.* Basic principles and prospects of using the theory of network calculus (network calculus). Infocommunication technologies. 2013. Vol. 11. No. 3. Pp. 34–39.
16. *Golosov P.E., Gostev I.M.* Simulation modeling of servers with interrupts in large multiprocessor systems. Izvestiya vuzov. Instrumentation. 2021. Vol. 64. No. 11. Pp. 879–886.
17. *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.
18. *Golosov P.E., Gostev I.M.* Simulation model of cloud computing with a sporadic control mechanism for parallel problem solving. Scientific and Technical Journal of Information Technologies, Mechanics and Optics. 2022. Vol. 22. No. 2. Pp. 269–278.
19. *Russell S., Norving P.* Artificial Intelligence: a Modern Approach. Moscow: Williams, 2006. 1408 p.