DOI: 10.25728/avtprom.2021.09.01

Имитационное моделирование распределения вычислительной нагрузки между серверными станциями с использованием нечеткого логического вывода

Е.О. Викулов, О.В. Денисов (ОмГТУ), В.А. Мещеряков (СибАДИ), Л.А. Денисова (ОмГТУ)

Представлена имитационная модель системы балансировки вычислительной нагрузки между серверными станциями облачного ресурса, созданная программными средствами MATLAB /Simulink /SimEvents. Выполнены модельные исследования серверного комплекса как системы с дискретными состояниями на основе теории систем массового обслуживания. Предложен алгоритм распределения данных по серверам с учетом их состояния на основе нечеткого логического вывода. Показано преимущество этого алгоритма по сравнению с круговым распределением нагрузки, которое выражается в уменьшении длины очередей в системе и увеличении числа обслуженных заявок на обработку данных серверами.

Ключевые слова: высоконагруженные системы обработки информации, балансировка облачной нагрузки, производительность сети, нечеткий логический вывод.

Введение

Технологии облачных вычислений все больше внедряются в промышленность, особенно в отрасли, связанные с производством технически сложных изделий, например, машиностроение, где производственные линии генерируют большие объемы данных. В связи с тем, что объем информации, передаваемой через телекоммуникационные сети, постоянно возрастает, увеличиваются задержки при передаче данных по каналам связи и уменьшается скорость вычислений [1-3]. Скорость передачи данных важна для обработки информации в реальном времени также и в нефтегазовой и химической промышленности для обнаружения неисправностей и своевременного обслуживания оборудования. Для обеспечения возрастающих потребностей передачи и обработки данных клиент-серверных приложений создаются новые центры обработки данных (ЦОД) и прокладываются телекоммуникационные сети. Для повышения скорости доставки данных разработчики клиент-серверных приложений создают различные технологические решения, позволяющие вести распределенную обработку данных, предоставляя пользователю вычислительные ресурсы и мощности облачных сервисов [2-4].

Для распределения вычислительной нагрузки между узлами (облачными ресурсами или физическими серверными станциями) часто используется механизм балансировки [1-3]. Термин «балансировка нагрузки» (англ. *load balancing*) используется для обозначения как перенаправления данных, получаемых от клиентов, на определенный узел системы, так и распределения вычислительной нагрузки приложений и вычислительных систем. Самым распространенным методом балансировки является алгоритм кругового обслуживания Round Robin [3-5], реализующий перебор узлов по круговому циклу, что позволяет равномерно распределять данные между серверными станциями, снижать затраты, связанные с обработкой запросов от клиентов, и увеличивать доступность ресурсов. Поставщики услуг облачных вычислений предоставляют встроенные решения для балансировки нагрузки между вычислительными ресурсами, однако встроенные алгоритмы балансировки могут не учитывать некоторые параметры функционирования аппаратных ресурсов, выделяемых для решения той или иной вычислительной задачи, что приводит к увеличению времени ответа для конечного пользователя [1-3]. Например, при распределении вычислительной нагрузки задача пользователя может быть отправлена на самый географически близкий выделенный облачный ресурс (наиболее близкий ЦОД), хотя в этот момент ресурс может быть занят выполнением другой вычислительной задачи. Тем самым пользователь, отправивший запрос на выполнение задачи, будет ожидать дольше, чем в случае, если бы был выбран более удаленный, но менее загруженный облачный ресурс.

С увеличением числа пользователей и обрабатываемых задач в системе растет вычислительная нагрузка. В этом случае выбор подходящего для вычислений облачного ресурса целесообразно выполнять на основе таких данных, как географическое положение пользователей, расстояние от пользователя до серверной станции (облачного ресурса), а также состояние облачного ресурса (загруженность жесткого диска, использование оперативной памяти и другие параметры). Известны алгоритмы [3-6] распределения данных по дублирующим серверным станциям (облачным ресурсам), которые учитывают возможные местоположения дубликатов серверных станций (например, ЦОД), местонахождение конечных пользователей и данные о каналах связи. Для оценки эффективности связи между серверной станцией (дубликатом)

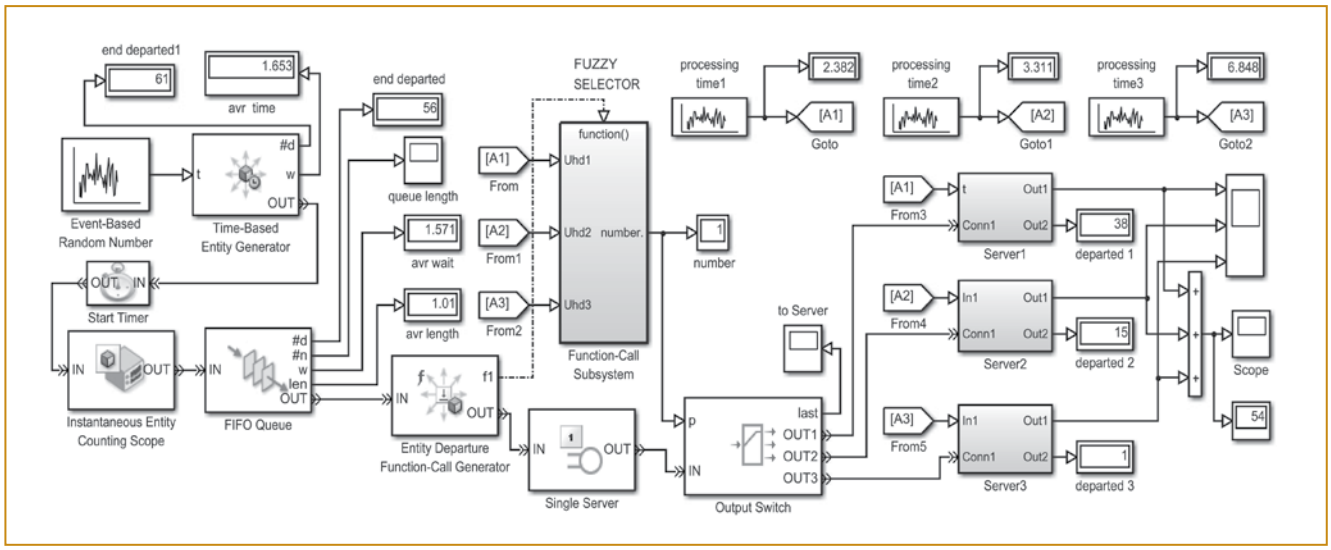

Рис. 1. Схема модели серверного комплекса

и конечным пользователем некоторые исследователи [3, 4] рассматривают время приема-передачи в сети, другие учитывают такие параметры, как доступную пропускную способность и качество связи [5, 6].

Для обоснования выбора сервера необходимо провести анализ больших объемов данных: исследовать параметры функционирования серверных станций облачного ресурса, а также данные о пользователях, чтобы определить, какие параметры имеют наибольшее влияние на скорость передачи данных пользователю. В работе представлен подход к выбору серверной станции для вычислительных задач и хранения данных, учитывающий доступную (для сервера-балансира) информацию о состоянии серверного комплекса облачного ресурса.

Процедуру выбора серверной станции на основе анализа параметров функционирования серверного комплекса предлагается реализовать в виде алгоритма балансировки нагрузки с применением нечеткого логического вывода [7]. В этом случае при оценивании состояний серверов и формировании решения о выборе серверной станции используется представление принятых показателей производительности сети в виде нечетких переменных. Такое представление параметров дает возможность приведения к единым относительным единицам измерения всех показателей, имеющих различный физический смысл и разные единицы измерения, а также использовать лингвистически сформулированные экспертные знания. Кроме того, в случае использования нечеткого логического вывода при принятии решения сервер-балансир сможет выбрать сервер для загрузки файлов в условиях лишь качественных представлений о зависимостях между параметрами системы и неполноты информации о состоянии серверов. Такая система балансировки даст возможность пользователям ускорить работу с серверами, так как позволит уменьшить временные затраты для загрузки и считывания файлов. Для исследования возможностей предлагаемой системы балансировки нагрузки между серверами необходимо разработать имитационную модель серверного комплекса облачного ресурса как системы с дискретными состояниями.

Применение теории систем массового обслуживания для исследования серверного комплекса

Для описания объектов, функционирующих в условиях действия случайных факторов, таких как серверный комплекс облачного ресурса, используется класс математических моделей, называемых системами массового обслуживания (СМО) [8]. В рассматриваемый серверный комплекс поступают заявки в случайные моменты времени и обслуживаются с помощью имеющихся каналов обслуживания – серверных станций, переключение между которыми выполняется с помощью сервера-балансира. Предполагается, что заявки на обслуживание образуют поток, то есть последовательность заявок с чередованием моментов их появления во времени. Понятие потока основано на предположении, что каждое событие будет происходить в заранее неизвестные случайные моменты времени. Имитационное моделирование систем массового обслуживания складывается из моделирования потока заявок и работы обслуживающих каналов.

Для описания серверного комплекса как СМО необходимо задать данные о входящем потоке заявок, которые поступают на обслуживание, порядок постановки заявки в очередь и выбора из нее, а также правило, по которому осуществляется распределение нагрузки (алгоритм работы сервера-балансира). Для описания входящего потока заявок необходимо описать моменты времени их поступления в серверный комплекс. Закон поступления может быть детерминированный или вероятностный. В нашем случае входящий поток заявок описывается распределением вероятностей интервалов времени между соседними заявками. Правила обслуживания заявок серверами характеризуются длительностью обслуживания (распределением времени обслуживания) и дисциплиной обслуживания. Для характеристики свойств каждой линии задается длительность обслуживания заявки

сервером или время занятости сервера как случайная величина с заданным законом распределения.

Рассмотрим обслуживающую систему, состоящую из трех каналов (в соответствии с числом серверов), способных одновременно и независимо друг от друга обслуживать заявки. В любой момент времени сервер находится в одном из двух состояний: свободном или занятом. Заявки принимаются к обслуживанию в порядке очереди, свободный канал приступает к обслуживанию той заявки, которая ранее других поступила в систему, то есть используется дисциплина обслуживания заявок, называемая в англоязычной литературе FIFO – First In - First Out (первым поступил – первым обслужился).

Выходными параметрами серверного комплекса являются величины, характеризующие качество его функционирования как СМО. Это такие параметры, как максимальная и средняя длина очередей заявок в информационной системе, время нахождения заявок в очередях и каналах обслуживания, а также число обслуженных заявок серверными станциями. Имитационная модель серверного комплекса позволит определять его характеристики и изменения состояния во времени при заданных потоках заявок, поступающих на входы системы, и обосновать алгоритм выбора сервера для загрузки данных пользователем.

Построение имитационной модели распределения данных

Для разработки и исследования системы балансировки нагрузки создана имитационная модель распределения данных между серверными станциями, реализованная с помощью инструментов моделирования пакета MATLAB / Simulink / SimEvents [9, 10]. Используемая среда Simulink содержит средства визуального моделирования, обладающие гибкостью, необходимой для построения сложных моделей, и удобством применения, так как позволяет строить модель из библиотечных блоков [9].

Применение приложения SimEvents дает возможность осуществлять моделирование системы с дискретными состояниями на основе теории очередей и СМО [8, 10]. Библиотека SimEvents позволяет создавать имитационные модели, зависящие как от времени, так и от дискретных состояний. Кроме того, такая модель позволяет учитывать переменные значения величин задержки в каналах обработки и передачи данных и определять изменение суммарного времени обработки заявок серверным комплексом в зависимости от загруженности серверов и времени обработки файлов.

При дискретно-событийном моделировании принято использовать понятие сущности (англ. entity). Сущности могут перемешаться через сети очередей и серверов, управляемых дискретными событиями. Под событием (англ. event) в модели, построенной средствами SimEvents, понимается мгновенное дискретное явление, которое изменяет переменную состояния, выход блока модели или является причиной появления других событий. Событием является, например, перемещение сущности от одного блока к другому, а также завершение обслуживания сущности в сервере.

Рассмотрим схему математической модели серверного комплекса при распределении нагрузки с помощью сервера-балансира, которая приведена на рис. 1. Модель собрана из библиотечных блоков SimEvents и Simulink, для настройки свойств и параметров которых вызываются диалоговые окна [9, 10].

Источником заявок (сущностей) в модели служит блок Time-Based Entity Generator. Закон распределения и среднее время между возникновением заявок заданы в параметрах блока генерации случайных чисел Event-Based Random Number. Для учета времени обслуживания заявок в модель введены таймерные блоки. Для инициации вычисления временного интервала пребывания заявки в системе служит блок Start Timer. Сгенерированные заявки поступают в блок очереди FIFO Queue и далее должны распределяться по серверным станциям для обслуживания.

Рассмотрим возможности распределения данных по серверам с помощью кругового распределения нагрузки Round robin и алгоритма на основе нечеткого логического вывода, учитывающего данные о состоянии серверов.

Для реализации алгоритма балансировки нагрузки и выбора сервера (для загрузки файла пользователем) на основе нечеткого логического вывода в модели предназначена подсистема FUZZY SELECTOR. С целью синхронизации этой подсистемы (ее тип Function-Call Subsystem) с событийно-управляемой имитационной моделью использованы блоки Entity Departure Function-Call Generator (генерирует управляющий сигнал в Function-Call Subsystem при каждом убытии заявки из блока) и Single Server (вспомогательный блок, применяется совместно с блоком Entity Departure Function-Call Generator). Затем заявки поступают в блок автоматического переключателя Output Switch, на управляющий вход которого поступает (от подсистемы FUZZY SELECTOR) сигнал о номере выбранного сервера. Следует отметить, что в случае моделирования режима кругового распределения нагрузки Round robin, возможность реализации которого

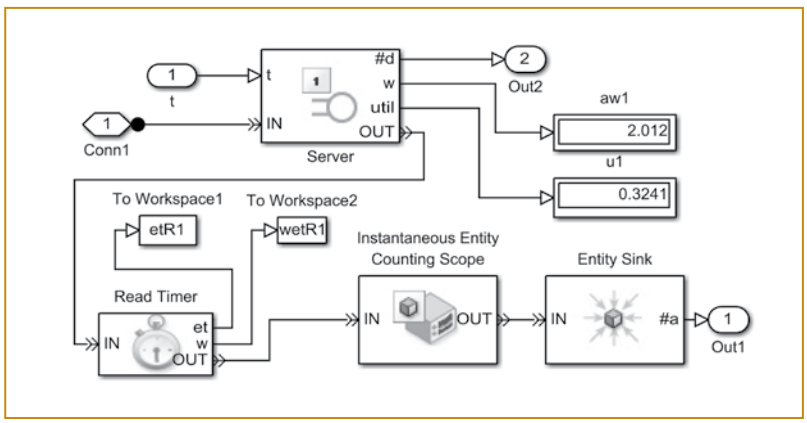

Рис. 2. Подсистема модели сервера Server1

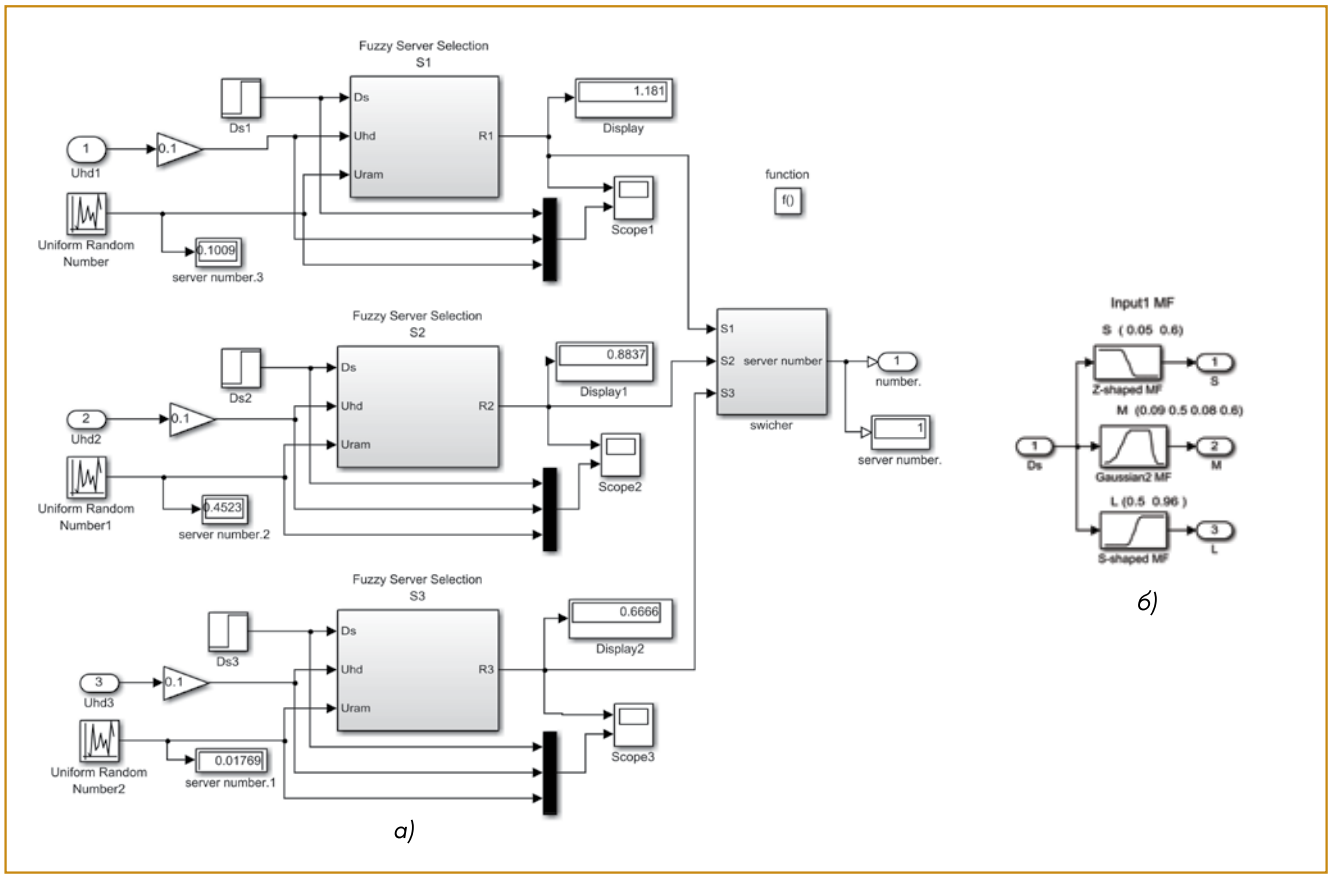

Рис. 3. Подсистемы: а – нечеткого выбора сервера FUZZY SELECTOR; б – фаззификации Input1 MF

предусмотрена в самом блоке Output Switch, этот входной сигнал не используется. Далее от блока автоматического переключателя Output Switch заявки перемещаются к одной из трех серверных станций, которые представлены подсистемами Server1... Server3.

На рис. 2 показана структура подсистемы Server1 – модели серверной станции. Основным элементом этой модели является канал обслуживания – собственно сервер (Server), который принимает на вход заявку и задерживает ее в течение времени обслуживания. Сигнал, соответствующий величине задержки, поступает на вход сервера от генератора случайной величины времени обработки заявки Processing time1 (тип блока – Event-Based Random Number).

Для учета времени пребывания заявки в системе использован блок Read Timer, который останавливает таймер, запущенный при генерации заявки (см. рис.1, блок Start Timer) и считывает его показания. Для завершения обслуживания заявки служит блок Entity Sink, на выходе которого формируется сигнал о количестве поглощенных им заявок. Подсистемы Server2 и Server3 имеют идентичную структуру.

Требуется распределить данные по серверным станциям таким образом, чтобы по возможности уменьшить время, затраченное на обработку данных. Для создания системы распределения данных, учитывающей состояние серверного комплекса, необходимо установить показатели производительности сети, от которых зависит время обработки данных.

Рассмотрим в качестве входной информации для принятия решения о выборе сервера такие его параметры, как удаленность от пользователя и доступные ресурсы. Введем следующие обозначения: D_s – расстояние от пользователя (клиента), загружающего данные, до серверной станции, относит. ед.; U_{hd}, который означает аппаратный ресурс – загруженность жесткого диска серверной станции, относит. ед.; U_{ram} соответствует аппаратному ресурсу – загруженности оперативной памяти серверной станции, относит. ед. Это входные сигналы для принятия решения о выборе серверной станции S_i (i=1,2,3). Введем обозначения для выходного параметра: R – решение о выборе сервера для загрузки файла. Этот параметр может иметь значения: P – позитивное; Z; –нейтральное; N – негативное.

Предлагается осуществлять балансировку нагрузки с помощью переключателя серверных станций, реализованного с использованием математического аппарата нечеткой логики [8]. Применение нечеткой логики позволяет принимать решение (в данном случае о выборе сервера) в условиях, когда данные о параметрах объекта являются недостаточно определенными. Схема модели переключателя – подсистемы FUZZY SELECTOR, которая осуществляет нечеткий выбор серверной станции при распределении нагрузки с помощью сервера-балансира, приведена на рис. 3а. Подсистема FUZZY SELECTOR, в свою очередь, имеет в своем составе три подсистемы селекторов Fuzzy Server Selection S1 – S3 (для каждого из трех серверов), осуществляющих

Таблица 1. Функций принадлежности лингвистических термов

Для внутренних термов M (функция *gauss2mf*)	Для крайних термов S (функция *zmf*)	Для крайних термов L (функция *smf*)
$\mu(x)=\begin{cases} e^{-\frac{(x-c_1)^2}{2\sigma_1^2}}, & x<c_1; \\ 1, & c_1\le x\le c_2; \\ e^{-\frac{(x-c_2)^2}{2\sigma_2^2}}, & x>c_2. \end{cases}$ (1)	$\mu(x)=\begin{cases} 1, & x<a; \\ 1-2\left(\frac{x-a}{b-a}\right)^2, & a\le x\le\frac{a+b}{2}; \\ 2\left(\frac{x-b}{b-a}\right)^2, & \frac{a+b}{2}<x\le b; \\ 0, & x>b\ . \end{cases}$ (2)	$\mu(x)=\begin{cases} 0, & x<a; \\ 2\left(\frac{x-a}{b-a}\right)^2, & a\le x\le\frac{a+b}{2}; \\ 1-2\left(\frac{x-b}{b-a}\right)^2, & \frac{a+b}{2}<x\le b; \\ 1, & x>b\ . \end{cases}$ (3)

нечеткий логический вывод о пригодности серверов для загрузки файлов. Входные сигналы на каждую из подсистем Fuzzy Server Selection S1 – S3 в них переводятся в значения нечетких переменных. Эта процедура осуществляется в подсистемах фаззификации Input MF. Схема одной из подсистем Input1 MF (для параметра D_s сервера S_1) представлена на рис. 3б. Фаззификация остальных входных параметров (U_{hd} и U_{ram}) для каждого из серверов производится аналогично. Диапазоны изменения всех входных переменных представляются термами значений: *S* – малое, *M*–среднее, *L* –большое.

В качестве функций принадлежности для внутренних лингвистических термов M (англ. *mean value* – среднее значение) входных переменных селекторов принята двухсторонняя гауссова функция принадлежности (*gauss2mf*), которая является комбинацией двух гауссовских функций принадлежности. Функция формируется в соответствии с выражением (1) табл. 1. Здесь параметры c_1 и c_2 упорядочены соотношением $c_1 < c_2$ и задают минимальное (c_1) и максимальное (c_2) значения ядра нечеткого множества. Параметры σ_1 и σ_2 задают коэффициенты концентрации левой и правой частей функции принадлежности, соответственно. Вызов функции имеет синтаксис *y=gauss2mf*(*x*,[*sig1 c1 sig2 c2*]) и возвращает выходной аргумент *y*, содержащий степень принадлежности входного аргумента *x* лингвистическому терму *M*.

Таблица 2. База нечетких правил для выбора сервера

Показатель U_{hd}	Показатель U_{ram}		
	S	*M*	*L*
При показателе $D_s = S$			
S	*P*	*P*	*P*
M	*P*	*P*	*P*
L	*Z*	*N*	*N*
При показателе $D_s = M$			
S	*P*	*P*	*P*
M	*P*	*Z*	*N*
L	*N*	*N*	*N*
При показателе $D_s = L$			
S	*N*	*N*	*N*
M	*N*	*N*	*N*
L	*N*	*N*	*N*

Для задания крайних лингвистических термов *S* (англ. *small value*) используются *Z*-образные функции принадлежности (*zmf*), зависящие от двух параметров *a* и *b*, которые определяют крайние точки наклонной части функции принадлежности. Эта функция принадлежности является невозрастающей, ее параметры определяют интервал, внутри которого функция нелинейно убывает от 0 до 1. Вызов функции имеет синтаксис *y= zmf*(*x*,[*a b*]) и возвращает выходной аргумент y, содержащий степень принадлежности входного аргумента x терму *S*. Функция формируется в соответствии с выражением (2) табл. 1.

Таким же образом для задания крайних лингвистических термов *L* (англ. *large value*) используются *S*-образные функции принадлежности (*smf*), зависящие от двух параметров *a* и *b*, которые определяют крайние точки наклонной части функции. Вызов функции имеет синтаксис *y= smf*(*x*,[*a b*]) и возвращает выходной аргумент *y*, содержащий степень принадлежности входного аргумента *x* терму *L*. Функция формируется в соответствии с выражением (3) табл. 1.

Такие функции принадлежности для термов входных переменных выбраны в связи с тем, что имеют аналитическое представление в виде простых формул с малым числом параметров, что упрощает числовые расчеты и сокращает временные затраты при моделировании. Четкое число *R*, задающее заключение каждого правила, рассматривается как одноэлементное нечеткое множество с точечной или синглтонной (от англ. *single* – единственный) функцией принадлежности. Полученная база знаний соответствует базе знаний Сугено нулевого порядка, в которой посылки (входные параметры) заданы нечеткими множествами, а заключения правил (решения о пригодности выбора сервера) – четкими числами.

Операция обратного преобразования нечетких переменных в четкие (дефаззификация) формирует значения выходной переменной. При этом четкий вывод о пригодности каждого сервера для выбора осуществляется путем нахождения взвешенного среднего:

$$R = \sum_{j=1}^{m} \mu(R_j) R_j \Big/ \sum_{j=1}^{m} \mu(R_j),$$

где *R* – четкое значение выходной переменной; R_j – значение выходной переменной для *j*-го терма с единичным значением степени принадлежности;

$\mu(R_j)$ – степень принадлежности к этому терму; m=3 – число термов. Принималось, что число R_j, задающее заключение каждого правила, может иметь следующие значения: N=0; Z=1; P=2.

Таким образом, значение степени пригодности сервера для выбора определяется с помощью нечеткого логического вывода на основе использования совокупности нечетких правил и переменных. База нечетких продукционных правил для выбора сервера S_1 (такая же, как и для серверов S_2 и S_3), которая реализуется в подсистеме Fuzzy Server Selection S_1, представлена в табл. 2.

На основе обработки входных параметров для каждого сервера получены результаты в виде степени принадлежности к множеству положительных решений о пригодности серверов для загрузки файлов. После этого принятие окончательного решения о выборе номера сервера осуществляется подсистемой Switcher. Выбирается тот сервер, для которого значение степени принадлежности к множеству положительных решений о пригодности для загрузки файлов является наибольшим.

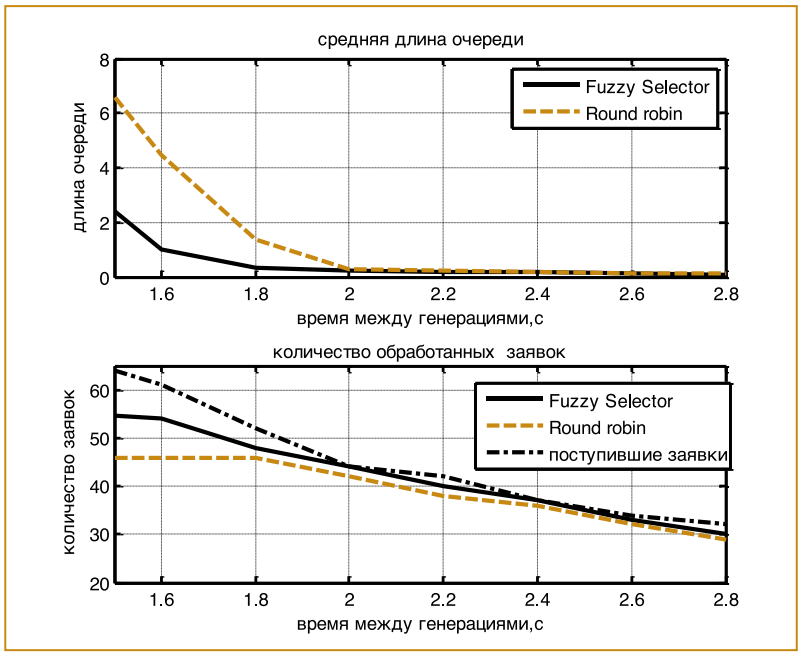

Рис. 4. Средняя длина очереди и число обработанных заявок в зависимости от интервала их поступления

Результаты экспериментальных исследований

С целью получения исходных данных, необходимых для построения имитационной модели серверного комплекса, был проведен натурный эксперимент по распределению заявок на загрузку файлов на реальные серверные станции [4]. Исследовались случайные процессы поступления заявок в системе, определялись такие случайные величины, как интервалы между моментами поступления заявок пользователей в сети, и оценивались показатели состояния серверов. Получено, что поток заявок на загрузку файлов в систему имеет экспоненциальный закон распределения, при котором плотность вероятности длительности пауз между генерациями заявок $p(t) = \lambda e^{-\lambda t}$, где λ – интенсивность потока (средняя длительность паузы между генерациями заявок $t_{ig} = \lambda^{-1}$). Определено, что длительность паузы между поступлением заявок t_{ig} изменялась в диапазоне 1...4 с. Принятые для моделирования исходные данные о диапазонах изменения показателей состояния серверов приведены в табл. 3. В качестве основного параметра, влияющего на время обработки заявки сервером, принималась загруженность жесткого диска, которая изменялась в принятых диапазонах по равномерному закону распределения (с плотностью вероятности $p(t) = (U_{hd\max} - U_{hd\min})^{-1}$).

С использованием данных, полученных от реальных серверных станций, проведено имитационное моделирование системы распределения нагрузки на основе нечеткого логического вывода и (для сравнения) балансировки с помощью метода кругового распределения нагрузки Round Robin. В результате моделирования получены такие характеристики системы, как средняя длина очереди в серверном комплексе и число обработанных заявок (рис. 4).

Расчеты проводились для различных значений интервалов поступления заявок в систему, и оценивались характеристики работы системы балансировки при круговом распределении и с помощью нечеткого селектора. Видно, что в обоих случаях средняя длина очереди уменьшается при увеличении длительности пауз между генерациями заявок до тех пор, пока эти длительности не будут больше времени обработки задач серверами. При этих же условиях число обработанных заявок становится равным числу сгенерированных заявок. Получено, что среднее значение длины очереди заявок при использовании

Таблица 3. Исходные данные для проведения моделирования

Сервер	Показатели состояния сервера				
	Загруженность жесткого диска		Загруженность оперативной памяти		Расстояние от пользователя
	$U_{hd\ min}$	$U_{hd\ max}$	$U_{ram\ min}$	$U_{ram\ max}$	D_s
S1	0,2...0,3	0,8...0,9	0,3...0,4	0,6...0,7	0,7
S2	0,1...0,2	0,4...0,5	0,6...0,7	0,9...1	0,75
S3	0,1...0,2	0,99...1	0,2...0,3	0,9...1	0,8

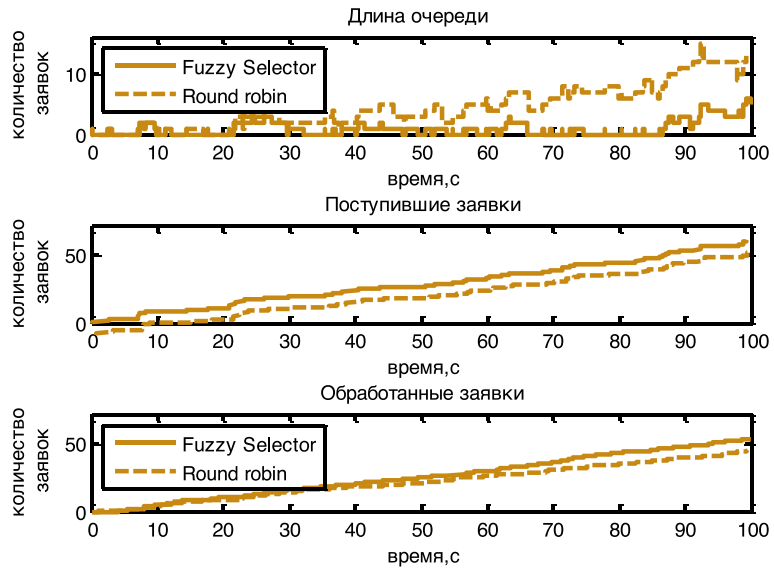

Рис. 5. Длина очереди и количество поступивших и обработанных заявок (при интенсивности входного потока со средним t_{ig} =1,6 с)

с помощью кругового метода Round robin. При использовании нечеткого селектора длина очереди намного меньше – 6 заявок. Общее число поступивших заявок – 60 ед., обработана системой с нечетким распределением нагрузки 51 заявка. В случае распределения нагрузки с помощью кругового метода число обработанных заявок меньше – 46 ед.

На рис. 6 представлены результаты распределения заявок по серверным станциям. В случае использования метода кругового распределения нагрузки Round Robin заявки распределены по серверам примерно одинаково. Сервером S_1 обработано 16 заявок, также как и сервером S_2, а сервером S_3 – 14 заявок. Всего обработано серверным комплексом 46 заявок из 60 поступивших.

Система с нечетким распределением нагрузки на основе данных о состоянии серверов работает лучше. Всего серверным комплексом обработана 51 заявка, причем основное число заявок было распределено на сервер S_1, имеющий наибольшую скорость обработки данных (35 заявок), на менее производительный сервер S_2 направлено 15 заявок, и всего одна заявка обработана третьим сервером S_3. Длительности пребывания заявок в системе в случае учета показателей работы серверов существенно уменьшились (⩽10 с, а при круговом распределении превышают 20 с). Среднее время пребывания заявок в системе сократилось на 20...60% для среднего времени между поступлением заявок t_{ig} от 1 с до 1,8 с. Таким образом, при распределении нагрузки с помощью нечеткого селектора существенно улучшены временные характеристики обслуживания заявок серверным комплексом.

нечеткого селектора, определяющего номер сервера для загрузки данных на основе информации о текущем состоянии серверов комплекса, существенно меньше, чем в случае, когда заявка отправляется на сервера прос-то по очереди (по кругу). Также и число обработанных заявок (при высокой частоте их поступления) больше, если учитывать данные о показателях работы серверов, то есть производительность серверного комплекса увеличивается.

Динамику процесса можно проследить по графикам на рис. 5, соответствующим временным диаграммам работы серверов для случая высокой интенсивности поступления заявок (t_{ig}=1.6 с), превышающей интенсивности потоков обслуживание заявок серверами. На протяжении 100 с длина очереди возрастает и достигает 14 заявок, когда распределение нагрузки осуществляется

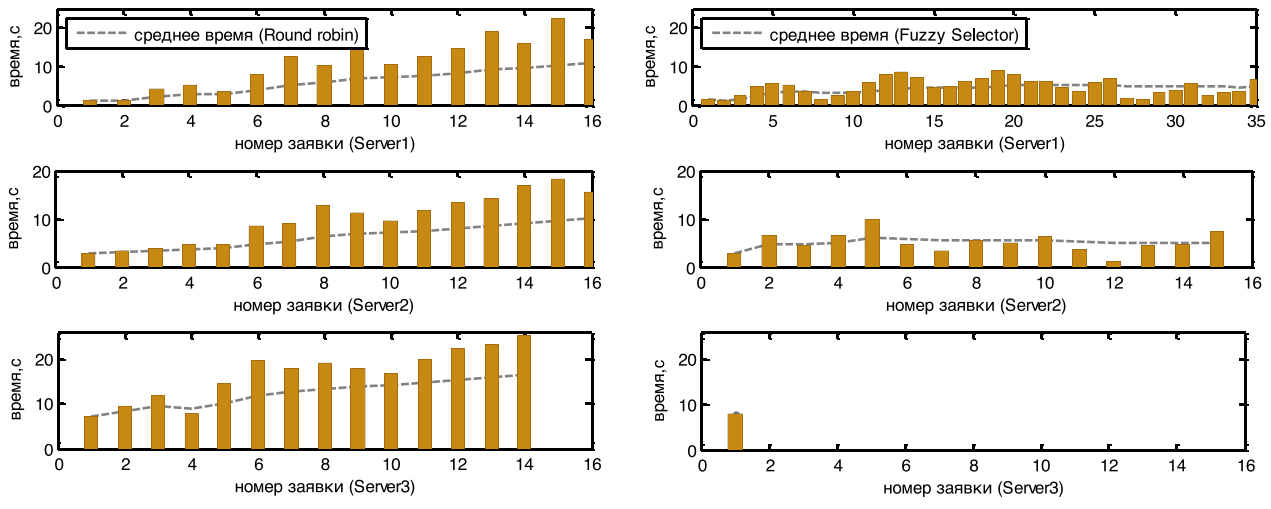

Рис. 6. Длительности пребывания заявок в системе

Заключение

На основании проведенного исследования сделан вывод, что выбранные программные средства MATLAB/Simulink/SimEvents/Stateflow пригодны для моделирования работы серверного комплекса облачного ресурса как системы с дискретными состояниями на основе теории систем массового обслуживания. Выполненные численные эксперименты по распределению данных по серверам позволили оценить преимущество подхода к выбору сервера на основе нечеткого логического вывода, которое выражается в уменьшении длины очередей, возникающих в серверном комплексе, и сокращению времени обслуживания пользователей.

Отметим, что разработанная имитационная модель позволяет исследовать работу сервера-балансира при распределении нагрузки между серверными станциями, учитывая случайный характер моментов поступления заявок на обработку, а также переменную длительность задержек при поступлении задач. То есть такая модель представляет возможность испытывать программное обеспечение сервера-балансира для серверных станций в условиях, приближенных к реальным. Кроме того, представляется перспективным использование предлагаемого подхода к разработке имитационных моделей для проведения испытаний и анализа функциональной корректности, а также надежности распределения нагрузки серверных станций облачных ресурсов в различных режимах работы.

Список литературы

1. *Afzal* and *Kavitha*. Load balancing in cloud computing – A hierarchical taxonomical classification / Journal of Cloud Computing: Advances, Systems and Application. 2019.
2. *Викулов Е.О., Леонов Е.А., Денисова Л.А.* Автоматизированное распределение больших объемов данных высоконагруженных систем. Динамика систем, механизмов и машин. 2014. № 3. С.146-149.
3. *Ghomi E.J., Rahmani A.M., Qader N.N.* Load-balancing algorithms in cloud computing: A survey / Journal of Network and Computer Applications. 2017.
4. *Vikulov E. O., Denisov O. V. and Denisova L. A.* Data distribution system preparation of server stations data. Journal of Physics: Conference Series Cep. "Mechanical Science and Technology Update, MSTU 2018" 2018. С. 012097.
5. *Sanders P., Mehlhorn K., Dietzfelbinger M., Dementiev R.* Sequential and Parallel Algorithms and Data Structures. 2019.
6. *AlKhatib A., Sawalha T., AlZu'bi S.* Load Balancing Techniques in Software-Defined Cloud Computing: an overview. 2020.
7. *Штовба С.Д.* Проектирование нечетких систем средствами MATLAB. – М.: Горячая линия –Телеком, 2007. – 288 с.
8. *Мещеряков В.А., Денисов В.П., Денисова Л.А.* Обучение студентов имитационному моделированию систем массового обслуживания в MATLAB // Сборник материалов II Торгового форума Сибири. Омск, 2013. С. 179-181.
9. Simulink: software for numerical simulation of continuous processes. Available: http://www.mathworks.com/products/simulink.
10. MATLAB SimEvents User's Guide [электронный ресурс] – Режим доступа. - URL: http://www.mathworks.com.

Викулов Егор Олегович *– ст. преподаватель,*
Денисов Олег Владимирович *– аспирант Омского государственного технического университета (ОмГТУ),*
Мещеряков Виталий Александрович *– д-р техн. наук, проректор по информационным технологиям Сибирского государственного автомобильно-дорожного университета (СибАДИ),*
Денисова Людмила Альбертовна *– д-р техн. наук, профессор ОмГТУ.*
Контактный телефон (381-2) 65-20-84.
E-mail: vikuloveo@gmail.com, olegdenisov95@yandex.ru, meshcheryakov_va@sibadi.org, denisova@asoiu.com

Петербургский метрополитен развернул единое информационное пространство для обмена данными

Группа компаний «РАМАКС» совместно с партнерами DATAREON и Docsvision завершила второй этап проекта по формированию единого информационного пространства в ГУП «Петербургский метрополитен» для обеспечения бесшовной интеграции эксплуатируемых прикладных систем, целостности НСИ и эффективного взаимодействия с внешними информационными системами.

Целью проекта является создание взаимосвязанных систем (комплекса) для бесшовной интеграции эксплуатируемых прикладных систем метрополитена, накопления и управления документами, используемыми в прикладных системах, обеспечения целостности, актуальности и достоверности нормативно-справочной информации (НСИ), гармонизации взаимодействия с внешними информационными системами.

В интеграционный контур проекта вошли системы SAP, СЭД Docsvision, «1С:MDM» и «Босс-кадровик», которые были подключены к централизованной системе управления НСИ. Инструментом интеграции между ними выступила корпоративная сервисная шина данных DATAREON ESB.

Команде разработчиков RAMAX предстояло решить вопрос не только хранения документов во внешнем хранилище, но и обеспечить незаметность происходящих изменений в системе с точки зрения бизнес-процессов для ее пользователей. Для этого решение максимально интегрировано в стандарт SAP по работе с Knowledge Provider (KPro). Масштабность проекта подчеркивается не только объемом миграции данных и обеспечением возможности хранения большого числа файлов, но и тщательной разработкой алгоритмов передачи и создания сложных связей бизнес-объектов и связанных с ними документов, включая систему НСИ.

Внедренное комплексное решение позволило предприятию оптимизировать работу сразу нескольких подразделений, существенно сократив количество итераций на получение той или иной информации, и тем самым повысить эффективность работы в целом.

В русле директив по импортозамещению необходимая заказчику функциональность перенесена на российскую платформу Docsvision. Теперь общий объем хранения в архиве на базе Docsvision составляет более 2 млн. карточек, что эквивалентно 1 Тб занятого пространства. И это только начало, ведь целевой объем хранения может достичь 30 Тб. Архив будет развиваться.

https://www.ramax.ru