УДК 004.75

В. Гайдамако, *dolpha@gmail.com*
Институт машиноведения и автоматики НАН КР, Бишкек

МОДЕЛИРОВАНИЕ ПРОИЗВОДИТЕЛЬНОСТИ МНОГОУРОВНЕВОГО ВЕБ-ПРИЛОЖЕНИЯ НА ОСНОВЕ ИСЧИСЛЕНИЯ РЕАЛЬНОГО ВРЕМЕНИ

Оценка производительности приложений и облачных сервисов крайне важна как во время проектирования приложения или сервиса – для разработчика, так и во время эксплуатации – для провайдера и клиента. В статье рассмотрен подход к оценке производительности трехуровневого веб-приложения с помощью аналитического моделирования на основе исчисления реального времени. Приведен пример построения ИРВ-модели, вычислены суммарная задержка на всех компонентах и задержка, получаемая с помощью свертки мин-плюс алгебры.

Ключевые слова: оценка производительности, аналитическое моделирование, модульный анализ производительности, исчисление реального времени.

Введение

Аналитическое моделирование является наименее затратным способом оценки производительности системы на ранних этапах проектирования. Наиболее распространенными подходами в аналитическом моделировании являются теория массового обслуживания, теория управления, сетевое исчисление (network calculus), стохастические сети с вознаграждениями [1,2]. В любом случае аналитическое моделирование должно обеспечить адекватность модели моделируемой системе и короткое время анализа так, чтобы этап моделирования стал частью процесса разработки системы.

Исчисление реального времени (ИРВ) используется для определения и предсказания соблюдения жестких ограничений времени исполнения в системах реального времени и встроенных системах. ИРВ основано на сетевом исчислении [3,4], которое было разработано для анализа прохожления потоков данных через сеть и часто используется для оптимизации параметров архитектуры систем обработки потоков пакетов. В работе [5] обосновано применение методов исчисления реального времени к оценке времени ответа приложений для облачных систем, а также возможности моделирования облачной среды и ее компонент – рабочей нагрузки, обработки задач, выделения ресурсов для виртуальных машин (ВМ), взаимного влияния (интерференции) ВМ на производительность, управления автономными ресурсами, консолидации серверов, а также стратегии масштабирования облачных вычислений (горизонтального и / или вертикального).

Моделируемое приложение. Рассмотрим пример трехуровневого веб-приложения, включающего уровень презентации (P) – веб-сервер, уровень приложения (B) – сервер приложений и уровень данных (D) – сервер баз данных (рис.1). В облачных системах все эти сервера существуют в нескольких экземплярах и являются

программными, так как работают на виртуальных машинах. Цель моделирования – предсказать время ответа для веб-приложения и определить, будет ли оно меньше дедлайна (для веб-приложений обычно 2 с) при известных характеристиках работы изолированного приложения на каждом уровне.

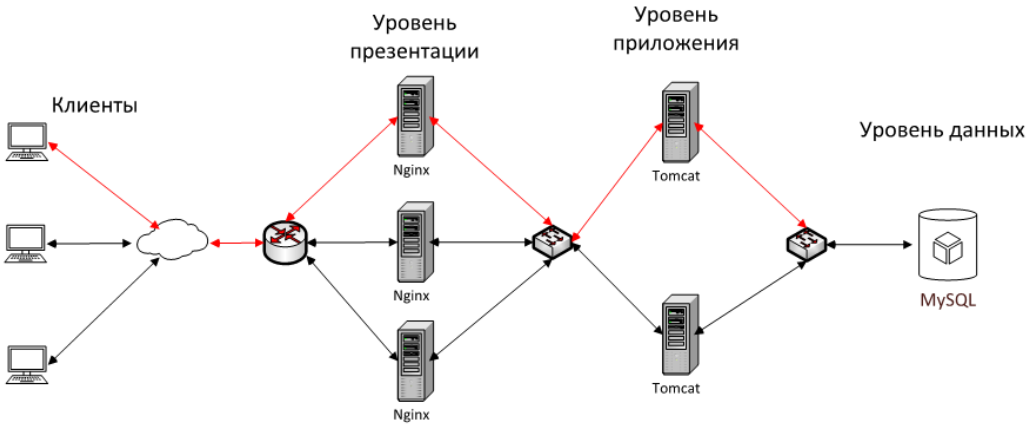

Рисунок 1 – Трехуровневое веб-приложение

При обработке запроса выполняются следующие задачи: 1) запрос пользователя с веб-сервера пересылается на сервер приложений, 2) сервер приложений посылает запрос к серверу баз данных, 3) сервер баз данных выполняет запрос или серверную процедуру и пересылает результат серверу приложений, 4) сервер приложений получает результат запроса, формирует ответ и пересылает его на веб-сервер для 5) презентации пользователю. Так как веб-сервер работает с созданием канала (statefull), запросы от пользователя и презентация результатов производятся одной виртуальной машиной (ВМ). Запрос пользователя проходит через виртуальные машины ВМ1, ВМ2, ВМ3, затем обратно к ВМ2, ВМ1, пользователь видит результат в своем браузере (рис.2) [5].

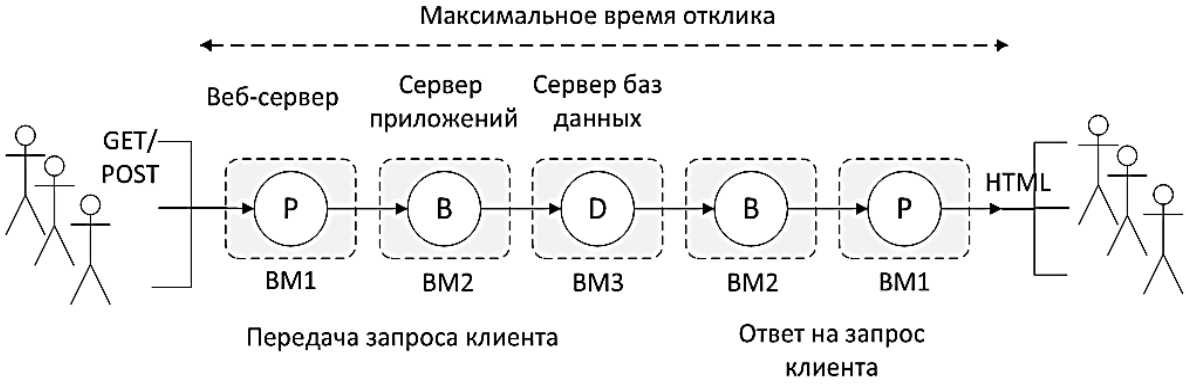

Рисунок 2 – Обработка запроса пользователя веб-приложением

Модульный анализ производительности с использованием исчисления реального времени

Модульный анализ производительности представлен в [4-7]. Для проведения модульного анализа производительности с ИРВ (МА-ИРВ) необходимо построить абстрактную модель производительности, включающую модель нагрузки и модель обслуживания. Абстрактная модель должна включать сведения о доступных вычислительных и коммуникационных ресурсах, о приложениях (или выделенных программных/аппаратных компонентах), а также об архитектуре самой системы.

Реальная система представляется в виде абстрактных аналитических компонент (ИРВ-компоненты), поведение которых может быть детерминированным или недерминированным [4-7]. ИРВ принадлежит к классу так называемых детерминированных теорий массового обслуживания. Оно детерминировано в том смысле, что всегда можно найти жесткие верхние и нижние границы показателей производительности (задержки). Это отличает его от класса методов недетерминированного, или стохастического анализа, таких как теория массового обслуживания (ТМО) и теория управления, для которых жесткие гарантии в целом предоставлены быть не могут.

Детерминированные теории массового обслуживания, в частности, МА-ИРВ, позволяют определять жесткие границы для показателей производительности, оценить наихудший сценарий, то есть максимальную задержку как для отдельного компонента, так и для всех компонентов, через которые проходит потокок событий [5,6].

Но ИРВ не позволяет моделировать среднее время отклика. Для этой цели лучше использовать стохастические подходы, например, ТМО. Вероятностный анализ на основе ИРВ, описанный в [5], может быть полезен для получения гарантий мягкого реального времени в облачной инфраструктуре. В то же время, в отличие от ИРВ, средняя величина задержки, используемая в анализе на основе ТМО, не позволяет получить никаких гарантий времени отклика.

Таким образом, трехуровневое веб-приложение можно смоделировать в виде пяти ИРВ-компонент (очередей). Входной поток событий представляется конечным числом типов событий – это HTTP-запросы браузера клиента к веб-серверу. Далее на веб-сервере входной поток HTTP-запросов преобразуется в поток запросов от веб-сервера к серверу приложений. Сервер приложений преобразует поток запросов от веб-сервера в поток запросов к базам данных. Сервер баз данных преобразует поток запросов от сервера приложений в поток ответа на запрос. Входной поток представляется кривыми поступления, причем входной поток на ресурсе (ВМ) преобразуется в выходной поток, который, в свою очередь, будет являться входным потоком для следующей ВМ [5].

Для инфраструктуры рис.2 исчисление реального времени может быть применено для анлитического определения максимального времени отклика приложения и максимальной длины очереди на обслуживание с учетом дисциплины обслуживания на различных ресурсах обработки. Среднее время обработки с помощью ИРВ не определить.

Функции и кривые поступления и обслуживания

Определение 1. (Функции поступления и обслуживания). Поток событий может быть описан функцией поступления R, где R(t) означает количество событий, которые произошли (поступили) в интервал времени [0, t).

Вычислительные или коммуникационные ресурсы могут быть описаны функцией обслуживания C, гле C(t) означает количество событий, которые были обслужены в интервал времени [0, t) [5].

Определение 2. (Кривые поступления и обслуживания), Верхняя и нижняя кривые $\alpha^u(\Delta)$, $\alpha^l(\Delta)$ ϵ $\mathbf{R}\geq 0$ (**R** – множество действительных чисел) поступления функции поступления R(t) удовлетворяют следующему неравенству:

$$\alpha^l(t-s) \leq R(t) - R(s) \leq \alpha^u(t-s),\ \forall s, t: 0 \leq t$$

Верхняя и нижняя кривые обслуживания

$$\beta^u(\Delta),\ \beta^l(\Delta)\ \epsilon\ \mathbf{R}\geq 0$$

функции обслуживания C(t) удовлетворяют неравенству:

$$\beta^l(t-s) \leq C(t) - C(s) \leq \beta^u(t-s),\ \forall s, t: 0 \leq t$$ [5].

Верхняя и нижняя кривые поступления обобщают и описывают все возможные трассировки потока событий (спорадические, периодические, периодические с джиттером), их можно получить из трассировок реальных серверов или имитационных моделей на временных отрезках конечной длины, например, методом скользящего окна. Кривые поступления также могут рассматриваться как количество запросов ресурса в интервал времени. Кривые обслуживания представляют минимальное и максимальное количество ресурсов, доступных в интервале времени, и могут быть получены из характеристик ресурса, аналитически или путем проведения измерений.

Конкретные и абстрактные компоненты системы

Поток событий в системах реального времени, как и в нашем примере, обычно проходит через цепочку ресурсов или выполняет цепь задач, а каждая задача выполняется на отдельном ресурсе. Входной поток событий R(t) трансформируется конкретным компонентом, доступность которого описывается функцией обслуживания C(t), в выходной поток R'(t), а доступные ресурсы теперь могут быть описаны функцией C'(t). Абстрактный компонент ИРВ трансформирует входные кривые поступления α^u и обслуживания β^l в выходные кривые α'^u и β'^l(рис.3).

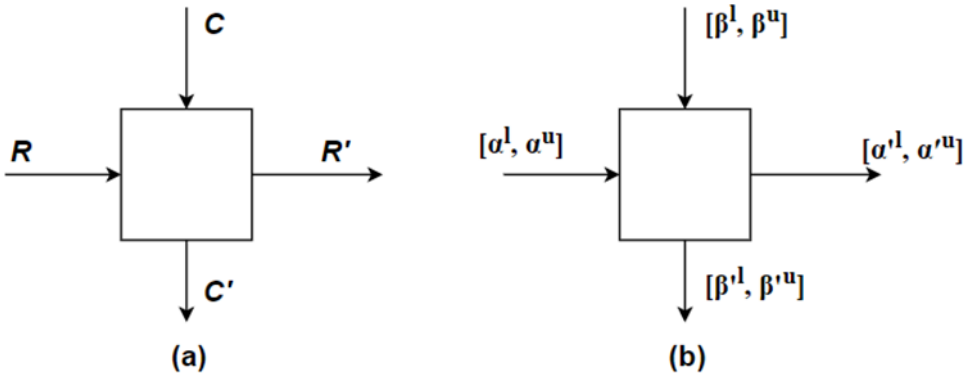

Рисунок 3 – Трансформация входных функций в выходные; (а) Конкретные функции поступления R(t) и обслуживания C(t) в конкретном обрабатывающем ресурсе трансформируются в выходные функции R' и C'; (б) Абстрактные кривые поступления α^u и обслуживания β^l абстрактным компонентом (ресурс ИРВ) трансформируются в выходные кривые $\alpha^{u'}$ и $\beta^{l'}$

Соотношение между входными и выходными кривыми определяется характеристиками компонента (семантикой обработки). Например, конкретный компонент, который при поступлении события для его обработки запускает полностью вытесняемую задачу, дисциплина планирования активных задач – очередь (FIFO), используется «жадная» стратегия исполнения, может быть описан уравнениями min-плюс и max-плюс алгебры [6]:

$$\alpha'^u = \min\{(\alpha^u \otimes \beta^u) \oslash \beta^l, \beta^u\},$$

$$\alpha'^l = \min\{(\alpha^l \oslash \beta^u) \otimes \beta^l, \beta^l\},$$

$$\beta'^u = (\beta^u - \alpha^l) \overline{\oslash} 0,$$

$$\beta'^l = (\beta^l - \alpha^u) \overline{\otimes} 0,$$ где операторы $\otimes$, $\oslash$, означают min-plus свертку (convolution) и обратную min-plus свертку (deconvolution) [4-8].

,

$$(f \otimes g)(\Delta) = \inf_{0 \le \lambda \le \Delta} \{f(\Delta - \lambda) + g(\lambda)\}$$
$$(f \oslash g)(\Delta) = \sup_{\lambda \ge 0} \{f(\Delta + \lambda) - g(\lambda)\} \quad .$$

Символы свертки и обратной свертки max-plus алгебры дополняются чертой сверху [4–8].

$$\left(f \overline{\otimes} g\right)(\Delta) = \sup_{0 \le \lambda \le \Delta} \{f(\Delta - \lambda) + g(\lambda)\} \ ,$$
$$\left(f \overline{\oslash} g\right)(\Delta) = \inf_{\lambda \ge 0} \{f(\Delta + \lambda) - g(\lambda)\} \ .$$

Описанное поведение в системах реального времени и встроенных системах называют компонентом с фиксированными приоритетами. Будем считать, что поведение веб-серверов, серверов приложений и серверов баз данных вполне соответствует описанному. Для компонент с другим поведением потребуются другие уравнения [6].

Если поток событий проходит через множественные ресурсы с минимальными кривыми обслуживания β^l_1, β^l_2, β^l_3…, β^l_n с использованием дисциплины очереди (FIFO), минимальная кривая обслуживания этого потока событий может быть вычислена с помощью итеративной свертки [5,6]:

$$\beta^{l\prime} = (((\beta^l_1 \otimes \beta^l_2) \otimes \beta^l_3) \otimes \ldots) \otimes \beta^l_n.$$

Таким образом, при моделировании веб-приложения необходимо задать максимальную кривую запросов и минимальные кривые обслуживания для каждой из ИРВ-компонент и вычислить суммарную кривую обслуживания. Максимальная задержка будет выражена максимальным горизонтальным расстоянием между максимальной кривой поступления и минимальной кривой обслуживания, а максимальное количество запросов в очереди – максимальным вертикальным расстоянием.

Построение абстрактной модели ИРВ. На рис.2 приведена декомпозиция процедуры обслуживания запроса пользователя, представим упрощенную абстрактную модель ИРВ в виде последовательности задач, выполняющихся на виртуальных серверах (рис. 4).

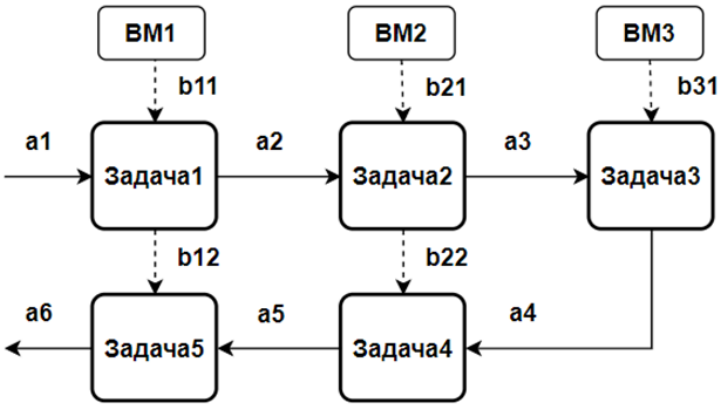

Рисунок 4 – ИВЛ-модель трехуровневого веб-приложения

Каждый компонент имеет свои входные и выходные кривые поступления и обслуживания. Потоки данных в модели ИРВ изображают горизонтально, использование ресурсов – вертикально. Для простоты в рассматриваемой модели

предполагается, что задачи имеют одинаковый приоритет. Моделирование проводится в MATLAB с помощью пакета RTC Toolbox [8].

Модульный анализ производительности в RTC Toolbox

Для проведения анализа модели, представленной на рис. 4, прежде всего нужно построить кривые поступления и обслуживания [8]. На данном этапе не будем рассматривать линии связи и устройства коммутации, будем считать, что запросы и ответы от серверов отправляются и принимаются мгновенно независимо от размера сообщения, хотя, разумеется, на практике это не так, и нужно модулировать линии коммутации.

Рассмотрим поведение системы на интервале 100 мс.

Кривые обслуживания. Строим кривые обслуживания с помощью функции `rtcfs`.

```
b11 = rtcfs(1);
b21 = rtcfs(2);
b31 = rtcfs(2);
```

Функция `rtcfs(B)` создает набор кривых обслуживания, соответствующих полностью доступному ресурсу с полосой пропускания B (для процессора это скорость обработки) [8]. Предполагаем, что для диспетчеризации используется дисциплина FIFO, задача поступает на процессор и освобождает его после завершения. Кривые, которые строятся – это всегда пара, массив из двух кривых – верхней (индекс 1) и нижней (индекс 2). Функции `rtcfsu` и `rtcfsl` создают верхнюю и нижнюю кривые обслуживания для такого ресурса. Например, b11(1) – верхняя кривая обслуживания, то есть максимально доступные ресурсы сервера BM1, b11(2) – минимальные доступные ресурсы.

Кривые поступления. Построим график функции и кривые поступления (прибытия). Допустим, из логов реального сервера на окне 100 мс получено, что запросы от пользователей поступили в 0, 5, 8, 11, 15, 18, 21, 25, 27, 31, 33, 35, 39, 40, 45, 48, 54, 57, 61, 66, 69, 72, 75, 78, 83, 86, 88, 90, 94, 96, 100. Задаем график функции прибытия:

```
R=rtccurve([[0,1,0];[5,2,0];[8,3,0];[11,4,0];[15,5,0];[18,6,0];[
21,7,0];[25,8,0];[27,9,0];[31,10,0];[33,11,0];[35,12,0];[39,13,0
];[40,14,0];[45,15,0];[48,16,0];[54,17,0];[57,18,0];[61,19,0];[6
6,20,0];[69,21,0];[72,22,0];[75,23,0];[78,24,0];[83,25,0];[86,26
,0];[88,27,0];[90,28,0];[94,29,0];[96,30,0];[100,31,0];]);
```

Теперь необходимо задать кривые поступления в виде периодической функции `rtcpjd(p,j,d)`, где p – период, j – джиттер, d – минимальная временная дистанция между событиями. Чем выше джиттер, тем больше расстояние между верхней и нижей кривыми поступления:

```
a=rtcpjd(3, 10, 0);
```

Для проверки соответствия смоделированных кривых поступления функции поступления выполняется отбратная мин-свертка кривой, описывающей поток запросов, кривая свертки нигде не должна быть выше верхней кривой поступления.

```
S = rtcmindeconv(R, R);
```

Кривые можно вывести с помощью команды

```
>> rtcplot(R, 'r',a,'r--', b11,'g', b21, 'g:', b31,'g--', 100);
```

Рассчитаем выходной поток запросов, кривую обслуживания для ресурсов, не занятых обработкой запросов, максимальную задержку и размер буфера с помощью

функции `rtcgpc` для компонентов GPC (General Processing Component, Компонент общей обработки) (рис.5).

Сначала рассчитаем задержки для каждой задачи в цепочке и суммируем их.

Задача 1. Веб-сервер принимает от пользователя запрос и пересылает его серверу приложений.

```
e1=2;  (здесь указывается худшее ожидаемое время выполнения)
[a2 b12 del1 buf1] = rtcgpc(a, b11, e1);
disp(['delay = ',num2str(del1), ';  buffer = ',num2str(buf1)]);
```

Получаем `delay = 4;  buffer = 4.`

Если выполнение задачи будет занимать 2 мс

```
delay = 8;  buffer = 4
```

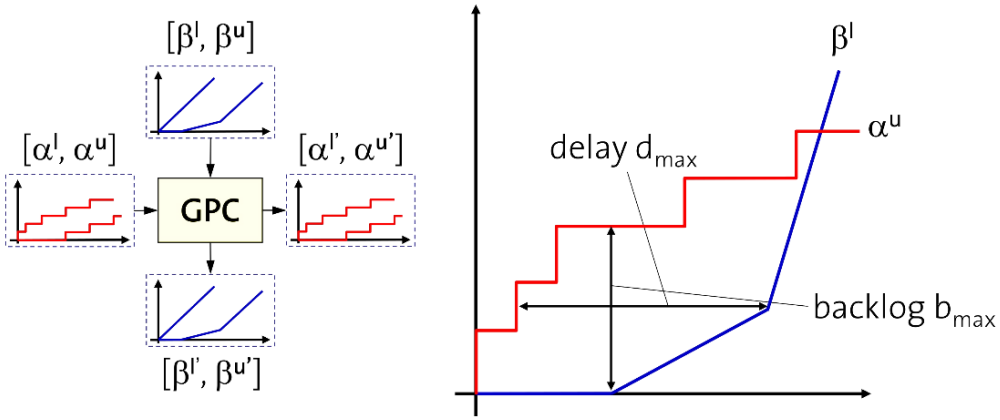

Рисунок 5 – Компонент общей обработки и графическое представление максимальной задержки (delay) и максимальной очереди (backlog) [8]

Задача 2. Сервер приложений формирует запросы к базе данных и пересылает их серверу, `e2=3;`

```
[a3 b22 del2 buf2] = rtcgpc(a2, b21, e2);
delay = 1.5;  buffer = 1
```

Задача 3. Сервер баз данных формирует ответ на запрос сервера приложений.

```
e3=4;
[a4 b32 del3 buf3] = rtcgpc(a3, b31, e3);
disp(['delay = ',num2str(del3), ';  buffer = ',num2str(buf3)]);
delay = 2;  buffer = 1
```

Задача 4. Сервер приложений получает ответ от сервера баз данных и пересылает его веб-серверу.

```
e4=2;
[a5 b23 del4 buf4] = rtcgpc(a4, b22, e4);
delay = 13;  buffer = 6
```

Задача 5. Веб-сервер получает ответ от сервера приложений и пересылает результат на браузер клиента.

```
e5=1;
[a6 b13 del5 buf5] = rtcgpc(a5, b12, e5);
disp(['delay = ',num2str(del5), ';  buffer = ',num2str(buf5)]);
delay = 40;  buffer = 14
```

Суммарная задержка:

```
del=del1+del2+del3+del4+del5= 64.5 мс
```

Заметим, что если время выполнения задачи 5 будет равно 2, получим

```
delay = Inf;  buffer = Inf
```

Это означает, что веб-сервер не сможет обработать все приходящие запросы, и они будут накапливаться «бесконечно», то есть нужно подключать второй сервер. Результат «бесконечность» на любом из серверов означает, что необходимо увеличение количества серверов данного типа.

Суммарную задержку можно посчитать и другим способом:

```
delay_add=rtcdel(a,b11,e1) + rtcdel(a2,b21,e2) +
rtcdel(a3,b31,e3)+ rtcdel(a4,b22,e4) + rtcdel(a5,b12,e5);
```

Результат – тот же, 64.5.

Теперь проведем расчет через свертку кривых обслуживания.

```
delay_opt=rtcdel(a,b11,e1,b21,e2,b31,e3, b22,e4,b12,e5);
delay = 46.5
```

Результаты показывают, что суммирование задержек на компонентах дает более пессимистичную оценку.

Зададим кривые обслуживания для ресурса с разделением времени – TDMA (Time Division Multiple Access) для задач той же продолжительности. Функция `rtctdma(S,C,B)` создает кривые обслуживания для ресурсной модели TDMA со слотом длины S, TDMA циклом C и полосой пропускания B. С ресурсами процессоров, как в первом случае, при обслуживании в порядке поступления с полностью доступным процессором (1, 2, 4) сразу получаем «бесконечность»:

```
cpu1 = 2;
cpu2 = 4;
cpu3 = 4;

b11 = rtctdma(2,3,cpu1);
b21 = rtctdma(2,3,cpu2);
b31 = rtctdma(4,6,cpu3);

delay1 = 6;  buffer = 4
delay2 = 1.75;  buffer = 2
delay3 = 4;  buffer = 3
delay4 = 9;  buffer = 6
delay5 = 30.5;  buffer = 11
Суммарная задержка (сумма задержек на компонентах) : 51.25
Суммарная задержка (оптимизированная задержка) : 33.75
```

Расчеты для обслуживания в порядке поступления с полностью доступным процессором и скоростями процессоров (2, 4, 4) дают результат:

```
delay1 = 4;  buffer = 4
delay2 = 0.75;  buffer = 1
delay3 = 1;  buffer = 1
delay4 = 4;  buffer = 4
delay5 = 11;  buffer = 7
Суммарная задержка (сумма задержек на компонентах): 20.75
Суммарная задержка (оптимизированная, результат свертки): 16.75
```

Задание кривых обслуживания для TDMA с длиной слота, равной периоду, моделирует систему, аналогичную системе с полностью доступным процессором

```
b11 = rtctdma(2,2,cpu1);
b21 = rtctdma(2,2,cpu2);
b31 = rtctdma(4,4,cpu3);
```

Результат аналогичен предыдущему.

Для компонентов с разными моделями обсуживания:

```
b11 = rtcfs(cpu1);
b21 = rtctdma(2,3,cpu2);
b31 = rtctdma(4,6,cpu3);
delay1 = 4;  buffer = 4
delay2 = 2.5;  buffer = 3
delay3 = 5.25;  buffer = 4
delay4 = 9.25;  buffer = 7
delay5 = 13.5;  buffer = 10
Суммарная задержка (сумма задержек на компонентах): 34.5
Суммарная задержка (оптимизированная, результат свертки): 26
```

В более детальных моделях необходимо учитывать длину и количество передаваемых сообщений, характеристики линий передачи и сетевого оборудования. Для моделирования процесса предоставления ВМ предусматриваются периоды простоя. Чтобы смоделировать взаимное влияние ВМ, работающих на одном физическом сервере, на производительность (интерференция ВМ), в ИРВ-модель можно добавить дополнительный логический компонент производительности (недетерминированный компонент ИРВ). В частности, кривая обслуживания этого компонента позволит моделировать недетерминированный доступ к общим ресурсам в виртуализированных средах. Для анализа производительности этот абстрактный компонент должен быть правильно откалиброван. Должно быть выяснено количество ВМ на физическом сервере, при котором взаимное влияние является приемлемым. Это может быть определено аналитически с использованием, например, теории массового обслуживания или измерено на реальных серверах [5].

Параметры компонент могут определяться в процессе мониторинга облачной системы [1,2,9,10] и использоваться для управления инфраструктурой облачной системы и автобалансировки на основе ИРВ. Изменение длины очереди или задержки, вычисленных на базе данных мониторинга, будет сигналом о необходимости изменения количества экземпляров серверов, миграции ВМ на другие физические сервера, в другие центры обработки данных (ЦОД) или принятия других решений по горизонтальному или вертикальному масштабированию.

Заключение

Приведеные примеры моделирования трехуровневого веб-приложения, показывают, что ИРВ применимо для аналитической оценки производительности приложений, причем к компонентам построенной ИРВ-модели могут применяться разные значения параметров, отражающих как характеристики виртуальных серверов, так и дисциплину обслуживания. Расчеты проводятся относительно быстро, что позволяет применять ИРВ как при проектировании инфраструктуры приложения или облачного сервиса, так и для мониторинга и автобалансировки. Рассмотренная модель приложения является упрощенной, в ней не учтено количество и размеры сообщений, передаваемых между серверами, а также характеристики линий связи, в более подробных рабочих моделях это должно учитываться. Также при использовании

нескольких экземпляров серверов каждого типа должны учитываться накладные расходы на перенаправление входных потоков событий на выбранный сервер. ИРВ-моделирование также применимо к облачным системам, предоставляющим «как сервис» датчики, актуаторы или другие устройства. Многие процессы в таких системах являются строго периодическими с дедлайном, взаимодействие с устройствами производится в реальном времени и при проектировании инфраструктуры и выборе оборудования должны учитываться требования жесткого реального времени.

Литература

1. QiangDuan, Cloud service performance evaluation: status, challenges, and opportunities – a survey from the system modeling perspective. // Digital Communications and Networks, Volume 3, Issue 2, May 2017, Pages 101-111. URL: https://www.sciencedirect.com/science/article/pii/S2352864816301456, (дата обращения 25.10.2020)
2. Гайдамако В.В. Обзор методов мониторинга и оценки производительности компонент облачной информационно-измерительной системы // Проблемы автоматики и управления, №2 (37), 2019, с. 26–38
3. Росляков А. В., Лысиков А. А. Применение теории стохастических сетевых исчислений к анализу характеристик VPN. // T-Comm. 2013, №7, с.106-109
4. Philipp Rümmer. A Short Introduction to Real-time Calculus. // Uppsala University 2014. URL: http://www.it.uu.se/edu/course/homepage/realtid/ht14/schedule/workload-models-2.pdf / (дата обращения 25.10.2020)
5. Гарай Г.Р., Черных А., Дроздов А.Ю. Сравнительный анализ методов оценки производительности многоуровневых облачных приложений. Труды ИСП РАН, том 27, вып. 6, 2015 г., стр. 199-224. DOI: 10.15514/ISPRAS-2015-27(6)-14.
6. E. Wandeler, L.Thiele, M.Verhoef. P.Lieverse. System architecture evaluation using modular performance analysis: a case study // Computer Science International Journal on Software Tools for Technology Transfer, 2006, Pages 649–667.
7. W. Haid, S. Perathoner, N. Stoimenov, L.Thiele. Modular Performance Analysis with Real-Time Calculus. // PhD Course on Automated Formal Methods for Embedded Systems DTU – Lyngby, Denmark – June 11, 2007. URL: http://www2.imm.dtu.dk/courses/02917/MPA_RTC_DTU07.pdf / (дата обращения 25.10.2020).
8. RTC Toolbox Tutorial. URL:https://www.mpa.ethz.ch/static/Tutorial.html / (дата обращения 25.10.2020).
9. G. Aceto, A. Botta, W. De Donato, and A. Pescapè, "Cloud monitoring: A survey" // Computer Networks, vol. 57, pp. 2093-2115, 2013.
10. Cloud monitoring guide. / URL:https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/manage/monitor/(дата обращения 25.10.2020).