ИНФОРМАТИКА И ТЕХНИЧЕСКИЕ НАУКИ
COMPUTER SCIENCE AND ENGINEERING SCIENCES

МОДЕЛИРОВАНИЕ ПРОЦЕССОВ И УПРАВЛЕНИЕ В ТЕХНИЧЕСКИХ СИСТЕМАХ
MODELLING OF PROCESSES AND MANAGEMENT IN TECHNICAL SYSTEMS

УДК 519.687.1

М. В. Дубовик, В. В. Смелов
Белорусский государственный технологический университет

МАТЕМАТИЧЕСКАЯ МОДЕЛЬ ДЛЯ АНАЛИЗА АЛГОРИТМОВ РАСПРЕДЕЛЕНИЯ ЗАПРОСОВ МЕЖДУ СЕРВЕРАМИ

Статья посвящена разработке математической модели программной системы с микросервисной архитектурой. Модель предназначена для анализа эффективности алгоритмов распределения входящих запросов между несколькими экземплярами автономных серверных компонент. С помощью имитационной реализации модели проведено экспериментальное исследование двух алгоритмов балансировки нагрузки: алгоритма Round Robin, который равномерно распределяет запросы по серверам, и алгоритма The Least Bandwidth Method, который позволяет распределить запросы по серверам на основе их загруженности. В отличие от алгоритма статического распределения нагрузки Round Robin, динамический алгоритм The Least Bandwidth Method учитывает текущую нагрузку каждого сервера в системе. При таком подходе можно динамически в зависимости от загруженности серверов распределять поступающие запросы, чтобы ускорить обработку. Такие алгоритмы дают хорошие результаты, особенно, когда время выполнения сильно варьируется от одной задачи к другой.

Ключевые слова: балансировка нагрузки, математическая модель, имитационная модель, Round Robin, The Least Bandwidth Method.

Для цитирования: Дубовик М. В., Смелов В. В. Математическая модель для анализа алгоритмов распределения запросов между серверами // Труды БГТУ. Сер. 3, Физико-математические науки и информатика. 2021. № 1 (242). С. 31–35.

M. V. Dubovik, V. V. Smelov
Belarusian State Technological University

MATHEMATICAL MODEL FOR ANALYZING ALGORITHMS FOR DISTRIBUTING REQUESTS BETWEEN SERVERS

The article is devoted to the development of a mathematical model of a software system with a microservice architecture. The model is designed to analyze the efficiency of algorithms for distributing incoming requests among several instances of stand-alone server components. Using a simulation implementation of the model, an experimental study of two load balancing algorithms was carried out: the Round Robin algorithm, which evenly distributes requests to servers, and the Least Bandwidth Method, which allows you to distribute requests across servers based on their load. Unlike the Round Robin static load balancing algorithm, The Least Bandwidth Method dynamic

algorithm takes into account the current load of each server in the system. With this approach, it is possible to dynamically distribute incoming requests depending on the server load in order to speed up processing. Such algorithms give good results, especially when the execution time varies greatly from one task to another.

Key words: load balancing, mathematical model, simulation model, Round Robin, The Least Bandwidth Method.

For citation: Dubovik M. V., Smelov V. V. Mathematical model for analyzing algorithms for distributing requests between servers. *Proceedings of BSTU, issue 3, Physics and Mathematics. Informatics*, 2021, no. 1 (242), pp. 31–35 (In Russian).

Введение. Основным трендом современного бизнеса является его цифровая трансформация – превращение предприятия в IT-предприятие. Наиболее отчетливо это проявляется в банковской и финансовой сферах, в оптовой и розничной торговле, транспортной логистике, рекламном бизнесе и средствах массовой информации. Современные предприятия предоставляют услуги клиентам с помощью интернет-сервисов, что позволяет им повысить эффективность своей деятельности и расширить аудиторию клиентов. При этом, как правило, сервисы имеют непрерывный цикл работы, предоставляющий возможность клиентам воспользоваться им в любой момент времени и из любой географической точки мира.

Как правило, сервисы представляют собой web-приложения – программные системы, имеющие клиент-серверную архитектуру и применяющие для взаимодействия между своими компонентами протокол HTTP. Современный подход к разработке web-приложения предполагает разработку легковесных серверных компонент – микросервисов, одним из основных свойств которых является их автономность. Автономность позволяет строить программные системы с несколькими одновременно работающими независимо друг от друга экземплярами микросервисов.

Применение автономных серверных компонент (микросервисов) представляет возможность разработчикам создавать программные системы с несколькими одновременно работающими экземплярами одной и той же серверной компоненты. Такое решение позволяет, с одной стороны, наращивать производительность программной системы, с другой – повысить ее надежность.

Ниже рассматривается математическая модель программной системы с микросервисной архитектурой. Модель предназначена для анализа эффективности алгоритмов распределения входящих запросов между несколькими экземплярами автономных серверных компонент.

Основная часть. Рассмотрим программную систему W, представленную на рис. 1 и включающую три типа компонентов: клиенты (C_i, $i = \overline{1,i}$), серверы (S_j, $j = \overline{1,j}$) и балансировщик нагрузки (B).

Клиенты – программная компонента, обменивающаяся сообщениями с другими программными компонентами (серверами). Сообщения, перемещающиеся от клиента к серверу, будем называть запросами (на рисунке обозначены символом d), а от сервера к клиенту, – ответами (символ r).

Балансировщик нагрузки (далее просто – балансировщик) – программная компонента, которая является промежуточным звеном между клиентами и серверами: он принимает запросы от клиентов и перенаправляет их серверам и, наоборот, принимает ответы от серверов и отправляет соответствующим клиентам. Запросы, отправляемые балансировщиком серверам, формируются на основе запросов, полученных от клиентов: содержат все исходные данные запроса и, кроме того, дополнительную информацию (на рисунке обозначена символом α), которая может быть использована для управления поведением сервера. Ответы, направляемые серверами балансировщику, тоже могут содержать дополнительную информацию (β), которая может быть использована для управления поведением балансировщика.

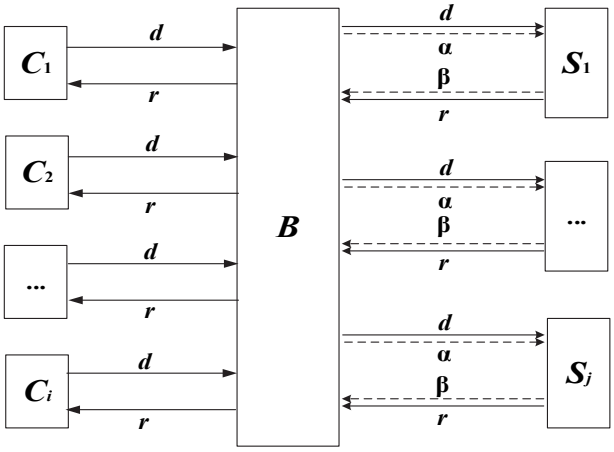


Рис. 1. Схема программной системы W

Будем далее предполагать, что процесс обмена сообщениями между клиентами и серверами обладает следующими свойствами:

1) каждому запросу соответствует один и только один ответ;

2) запрос может быть обработан любым сервером;

3) если клиент отправляет серию запросов, то эти запросы в общем случае могут обрабатываться разными серверами;

4) ответ на запрос всегда получает тот клиент, который отправил запрос;

5) сервер, на который направляется клиентский запрос, определяется только алгоритмом работы балансировщика.

Представим описанную выше программную систему W как четверку:

$$W = \left\langle \left\{ C_j, j = \overline{1,\bar{j}} \right\}, \left\{ K_l, l = \overline{1,\bar{l}} \right\}, B, \left\{ S_i, i = \overline{1,\bar{i}} \right\} \right\rangle, \quad (1)$$

где C_j, $j = \overline{1,\bar{j}}$ – множество клиентов; K_l, $l = \overline{1,\bar{l}}$ – множество категорий запросов; B – балансировщик; S_i, $i = \overline{1,\bar{i}}$ – множество серверов.

Клиента $C \in \{C_j, j = \overline{1,\bar{j}}\}$ представим как двойку:

$$C = \langle K, \lambda(t) \rangle, \quad (2)$$

где $K \in \{K_l, l = \overline{1,\bar{l}}\}$ – категория запросов, генерируемых клиентом; $\lambda(t)$ – интенсивность запросов, генерируемых клиентом (в общем случае является функцией времени).

Категория $K \in \{K_l, l = \overline{1,\bar{l}}\}$ запроса является целочисленной положительной величиной, характеризующей относительную трудоемкость обработки запроса. При этом значение данной величины не зависит от сервера, обрабатывающего этот запрос. Например, если $K_1 = 3$ и $K_2 = 6$, то это означает, что при равных условиях обработки запросы первой категории имеют трудоемкость в 2 раза меньше трудоемкости обработки запросов второй категории.

Балансировщик формально представим в виде четверки:

$$B = \langle P, I_B, A, O_B \rangle, \quad (3)$$

где P – процедура распознавания категории запроса; I_B – процедура обработки запроса (принимает запрос от клиента и отправляет его серверу, использует алгоритм A, а также формирует для сервера дополнительную информацию α, помещаемую в запрос); A – алгоритм выбора сервера для обработки запроса (использует результат работы процедуры P); O_B – процедура обработки ответа (принимает ответ сервера и отправляет его клиенту).

Сервер $S \in \{S_i, i = \overline{1,\bar{i}}\}$ представим в виде пятерки:

$$S = \langle h, k, F, I_S, O_S \rangle, \quad (4)$$

где h – максимальное количество запросов, которые могут одновременно обрабатываться сервером; k – коэффициент, который при умножении на значение категории запроса дает время, затрачиваемое данным сервером на обработку этого запроса; F – процедура вычисления величины дополнительного времени обработки запроса из-за загруженности сервера; I_S – процедура обработки запроса (обрабатывает управляющую информацию от балансировщика α); O_S – процедура обработки ответа (дополняет ответ сервера управляющей информацией β для балансировщика).

Следует обратить внимание на то, что формальное описание программной системы W не учитывает временные задержки, связанные с пересылкой сообщений в компьютерной сети и вычислениями балансировщика, а также не ограничивает интенсивности потоков сообщений. Это связано с тем, что задачей исследования является исключительно алгоритм управления балансировщиком, работа которого сводится к правильному выбору сервера, для обработки очередного поступившего от клиента запроса. При этом информацию, которую использует балансировщик в своих вычислениях, он извлекает только из поступающих запросов и ответов, а критерий его эффективности на интервале времени $(0, t)$ описывается целевой функцией (5).

$$\frac{M(t)}{L(t)} \to 1, \quad (5)$$

где $L(t)$ – количество поступивших в балансировщик запросов; $M(t)$ – количество поступивших в балансировщик ответов к заданному моменту времени t. Другими словами, если W_1 и W_2 – две программные системы, отличающиеся только входящими в их состав балансировщиками B_1 и B_2, то алгоритм балансировщика B_1 является более эффективным, чем алгоритм балансировщика B_2 на интервале времени $(0, t)$, если верным является неравенство:

$$\frac{M_1(t)}{L_1(t)} > \frac{M_2(t)}{L_2(t)}, \quad (6)$$

где $L_1(t)$, $M_1(t)$ и $L_2(t)$, $M_2(t)$ – количество поступивших запросов и ответов соответственно в первый и второй балансировщики к моменту t. Очевидно, что временные задержки, связанные с пересылкой сообщений по компьютерной сети и интенсивности входных потоков, являются общими для сравниваемых алгоритмов и не влияют на результат сравнения. Учет сложности алгоритмов балансировщика (продолжительности вычисления) не предусматривается моделью и предполагается, что время, потраченное балансировщиками на вычисления, является постоянным и не влияет на результат сравнения.

На рис. 2 представлен протокол работы имитационной модели программной системы W с параметрами, описанными в таблице ниже.

Протокол представляет собой таблицу, состоящую из 8 столбцов (слева на право): текущее модельное время (значение 50000); алгоритм балансировщика (*RR*); содержимое строки (*L*, *R*, *M*, *L-R-M*, *M*/*L* или *F*/*L*); четыре столбца, соответствующие категориям запросов, суммарное значение для всех категорий. Строки, помеченные символом *L*, содержат количество запросов, поступивших в балансировщик; символом *R* – количество запросов,

отвергнутых сервером ввиду превышения значения h; M – количество ответов, поступивших в балансировщик; L-R-M – количество запросов, которые обрабатываются с системе в текущий момент времени; M/L – отношение полученных ответов к общему количеству полученных запросов; F/L – отношение дополнительного времени (результат работы процедуры F балансировщика) к общему количеству поступивших запросов.

```
- 50000: RR     L:   5046   5047   5047   5046   = 20186
- 50000: RR     L:   5047   5046   5047   5047   = 20187
- 50000: RR     L:   5047   5047   5046   5047   = 20187
- 50000: RR     L:                               == 60560
- 50000: ----------------------------------------
- 50000: RR     R:      0      0      0      0   =     0
- 50000: RR     R:   1470   1609   1894   2800   =  7773
- 50000: RR     R:   2769   2851   3186   4231   = 13037
- 50000: RR     R:                               == 20810
- 50000: ----------------------------------------
- 50000: RR     M:   5046   5046   5047   5046   = 20185
- 50000: RR     M:   3550   3410   3128   2227   = 12315
- 50000: RR     M:   2241   2171   1837    803   =  7052
- 50000: RR     M:                               == 39552
- 50000 ----------------------------------------
- 50000: RR L-R-M:      0      1      0      0   =     1
- 50000: RR L-R-M:     27     27     25     20   =    99
- 50000: RR L-R-M:     37     25     23     13   =    98
- 50000: RR L-R-M:                               ==   198
- 50000 ----------------------------------------
- 50000: RR   M/L:      1      1      1      1   =     1
- 50000: RR   M/L:  0,703  0,676   0,62  0,441   =  0,61
- 50000: RR   M/L:  0,444   0,43  0,364  0,159   = 0,349
- 50000: RR   M/L:                               == 0,653
- 50000: ----------------------------------------
- 50000: RR   F/L:      1      3      4      5   =     3
- 50000: RR   F/L:    276    268    248    178   =   242
- 50000: RR   F/L:    307    300    256    112   =   244
- 50000: RR   F/L:                               ==   163
```

Рис. 2. Результат имитационного моделирования при $A = RR$

Для каждого типа в протоколе записано по четыре строки: первые три соответствуют трем серверам в моделируемой системе, а последняя содержит суммарное значение (для L, R, M, L-R-M) или среднее арифметическое (для M/L, F/L).

Кратко поясним некоторые числовые значения, отраженные в протоколе.

1. Первый столбец: все значения в протоколе соответствуют состоянию системы на момент 50 000 единиц модельного времени.

2. Второй столбец: все значения в протоколе соответствуют алгоритму балансировщика RR, циклически (по одному) распределяющий запросы серверам.

3. Суммарная строка M/L: из общего количества поступивших запросов программная система сумела обработать примерно 65% запросов.

4. Суммарная строка L: общее количество запросов, поступивших в систему 60 560.

5. Последний столбец трех верхних строк L: количество запросов, поступивших на каждый сервер. Следует обратить внимание, что эти значения не отличаются более чем на 1. Это результат работы алгоритма RR, равномерно распределяющий запросы по серверам.

6. Суммарная строка R: общее количество запросов, отвергнутых серверами из-за перегрузки 20 810.

7. Суммарная строка M: общее количество обработанных запросов системой 39 552.

8. Суммарная строка L-R-M: общее количество запросов, обрабатываемых системой в текущий момент времени 198.

9. Суммарная строка F/L: в среднем каждый запрос, поступивший в систему, задерживается на обработке в сервере на 163 единицы времени из-за загруженности серверов.

Параметры имитационной модели программной системы W

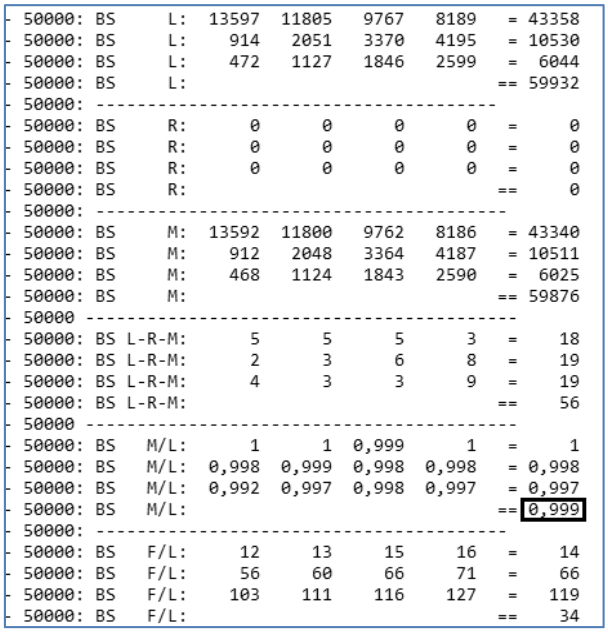
Общие параметры	
Количество клиентов	4
Количество серверов	3
Количество категорий запросов	4
Время моделирования, ед. модельного времени	50 000
Клиенты $C = < K, \lambda(t) >$	
1	$K = 1$, $\lambda(t) = 0{,}3$
2	$K = 2$, $\lambda(t) = 0{,}3$
3	$K = 3$, $\lambda(t) = 0{,}3$
4	$K = 4$, $\lambda(t) = 0{,}3$
Категории	
1	$K_1 = 1$
2	$K_2 = 2$
3	$K_3 = 3$
4	$K_4 = 4$
Серверы $S = < h, k, F, I_S, O_S >$	
1	$h = 100$, $k = 1$ $F(t) = k(L(t) - R(t) - M(t))$ I_S, O_S – не используется
2	$h = 100$, $k = 4$ $F(t) = k(L(t) - R(t) - M(t))$ I_S, O_S – не используется
3	$h = 100$, $k = 7$ $F(t) = k(L(t) - R(t) - M(t))$ I_S, O_S – не используется
Балансировщик $B = < P, I_B, A, O_B >$	
	P, I_B, O_B – не используется $A = RR$

Протокол на рис. 3 отражает результат моделирования программной системы W с параметрами, описанными в таблице, но с другим алгоритмом балансировщика ($A = BS$) выбора сервера.

Поясним результат моделирования, отраженный в протоколе на рис. 3.

1. Алгоритм BS отслеживает количество запросов, обрабатываемых каждым сервером, и направляет текущий запрос серверу, имеющему наименьшее количество обрабатываемых в данный момент запросов.

2. Последний столбец трех верхних строк L-R-M имеют примерно одинаковое значение. Это результат работы алгоритма BS, выравнивающий количество одновременно обрабатываемых серверами запросов.



```
- 50000: BS     L:  13597  11805   9767   8189   = 43358
- 50000: BS     L:    914   2051   3370   4195   = 10530
- 50000: BS     L:    472   1127   1846   2599   =  6044
- 50000: BS     L:                               == 59932
- 50000: ----------------------------------------
- 50000: BS     R:      0      0      0      0   =     0
- 50000: BS     R:      0      0      0      0   =     0
- 50000: BS     R:      0      0      0      0   =     0
- 50000: BS     R:                               ==     0
- 50000: ----------------------------------------
- 50000: BS     M:  13592  11800   9762   8186   = 43340
- 50000: BS     M:    912   2048   3364   4187   = 10511
- 50000: BS     M:    468   1124   1843   2590   =  6025
- 50000: BS     M:                               == 59876
- 50000 ----------------------------------------
- 50000: BS L-R-M:      5      5      5      3   =    18
- 50000: BS L-R-M:      2      3      6      8   =    19
- 50000: BS L-R-M:      4      3      3      9   =    19
- 50000: BS L-R-M:                               ==    56
- 50000 ----------------------------------------
- 50000: BS   M/L:      1      1  0,999      1   =     1
- 50000: BS   M/L:  0,998  0,999  0,998  0,998   = 0,998
- 50000: BS   M/L:  0,992  0,997  0,998  0,997   = 0,997
- 50000: BS   M/L:                               == 0,999
- 50000: ----------------------------------------
- 50000: BS   F/L:     12     13     15     16   =    14
- 50000: BS   F/L:     56     60     66     71   =    66
- 50000: BS   F/L:    103    111    116    127   =   119
- 50000: BS   F/L:                               ==    34
```

Рис. 3. Результат имитационного моделирования при $A = BS$

3. Серверы не были перегружены и не отвергли ни одного запроса.

4. В среднем каждый запрос, поступивший в систему, задерживается на обработке в сервере на 34 единицы времени из-за загруженности серверов.

5. Значение $M/L = 1$ – критерий эффективности выполнен.

Заключение. В данной статье предложена математическая модель программной системы с микросервисной архитектурой, состоящая из трех типов компонент: клиенты, серверы и балансировщик нагрузки.

В отличие от ранее существовавших моделей балансировки нагрузки, предложенная в статье математическая модель учитывает категории поступающих запросов, характеризующие трудоемкость обработки запроса. Представленная модель программной системы позволяет проанализировать эффективность алгоритма, распределяющего входящие запросы между несколькими экземплярами серверных компонент.

Было воспроизведено поведение программной системы с микросервисной архитектурой и проведено экспериментальное исследование алгоритмов балансировки нагрузки Round Robin (равномерное распределение запросов) [1] и The Least Bandwidth Method (по минимальному времени реакции) [2]. Полученные в ходе исследования результаты эксперимента позволили отметить преимущества динамических алгоритмов балансировки серверной нагрузки перед статическими. Алгоритм The Least Bandwidth Method учитывает текущую нагрузку каждого сервера в системе и динамически в зависимости от загруженности серверов распределяет поступающие запросы, обеспечивая высокий критерий эффективности и снижение количества отвергнутых запросов из-за перегрузки.

Список литературы

1. Stuti D., Prashant M. Utilizing Round Robin Concept for Load Balancing Algorithm at Virtual Machine Level in Cloud Computing // International Journal of Computer Applications. 2014. Vol. 94. No. 4. P. 23–29.

2. Вартанян С. О., Сокол В. В., Лесная Н. С. Методы балансировки нагрузки в Cloud системах // Кібернетика та системний аналіз. 2012. № 3. С. 101–103.

References

1. Stuti D., Prashant M. Utilizing Round Robin Concept for Load Balancing Algorithm at Virtual Machine Level in Cloud Computing. *International Journal of Computer Applications*. 2014, vol. 94, no. 4, pp. 23–29.

2. Vartanyan S. O., Sokol V. V., Lesnaya N. S. Methods of loading balancing in Cloud systems. *Kibernetika ta sistemniy analiz* [Cybernetics and System Analysis], 2012, no. 3, pp. 101–103 (In Russian).

Информация об авторах

Дубовик Марина Владимировна – магистрант кафедры информационных систем и технологий. Белорусский государственный технологический университет (220006, г. Минск, ул. Свердлова, 13а, Республика Беларусь). E-mail: dubovik@belstu.by

Смелов Владимир Владиславович – кандидат технических наук, доцент, заведующий кафедрой информационных систем и технологий. Белорусский государственный технологический университет (220006, г. Минск, ул. Свердлова, 13а, Республика Беларусь). E-mail: smw@belstu.by

Information about the authors

Dubovik Marina Vladimirovna – Master's degree student. Belarusian State Technological University (13a, Sverdlova str., 220006, Minsk, Republic of Belarus). E-mail: dubovik@belstu.by

Smelov Vladimir Vladislavovich – PhD (Engineering), Associate Professor, Head of the Department of Information Systems and Technologies. Belarusian State Technological University (13a, Sverdlova str., 220006, Minsk, Republic of Belarus). E-mail: smw@belstu.by

Поступила после доработки 08.02.2021