УДК 519.687.1

М. В. Дубовик, В. В. Смелов, Г. Л. Тимонович

Белорусский государственный технологический университет

**ТОЧНЫЕ ГРАНИЦЫ ЭФФЕКТИВНОСТИ АЛГОРИТМОВ РАСПРЕДЕЛЕНИЯ
ЗАПРОСОВ МЕЖДУ СЕРВЕРАМИ**

Статья посвящена оценке эффективности алгоритмов диспетчеризации http-запросов между серверами. Для оценки эффективности предлагается найти ее теоретический максимум и минимум путем построения точной верхней и точной нижней границ эффективности алгоритмов распределения запросов между серверами. Главной проблемой является построение точной верхней границы. В целом вычисление границ можно свести к построению и анализу множества рациональных планов распределения между серверами поступающих от клиентов запросов, оценке эффективности алгоритмов распределения запросов и последующему нахождению функций экстремума эффективности в каждый момент времени. Предложенный метод вычисления точных границ эффективности предполагает, что известны конфигурация программной системы, последовательность запросов во времени и их трудоемкость. Алгоритм, используемый методом для вычисления границ, имеет экспоненциальную сложность от времени и является трудноразрешимым для реальных конфигураций программных систем. Таким образом, этот метод для оценки эффективности алгоритмов диспетчеризации http-запросов является непригодным для применения в инженерных задачах.

Ключевые слова: алгоритмы распределения нагрузки, балансировка нагрузки, оценка эффективности, точные границы эффективности, план распределения запросов.

Для цитирования: Дубовик М. В., Смелов В. В., Тимонович Г. Л. Точные границы эффективности алгоритмов распределения запросов между серверами // Труды БГТУ. Сер. 3, Физико-математические науки и информатика. 2021. № 2 (248). С. 80–86.

M. V. Dubovik, V. V. Smelov, G. L. Timonovich

Belarusian State Technological University

**EXACT BOUNDARIES FOR THE EFFICIENCY OF THE ALGORITHMS
FOR DISTRIBUTING REQUESTS BETWEEN SERVERS**

The article is devoted to evaluating the efficiency of algorithms for dispatching http requests between servers. To evaluate the efficiency, it is proposed to find the theoretical maximum and minimum efficiency by constructing the exact upper and exact lower bounds for the efficiency of algorithms for distributing requests between servers. The main problem is plotting an accurate upper bound. In general, the calculation of the boundaries can be reduced to the construction and analysis of a set of rational plans for the distribution of requests from clients between the servers, the evaluation of the efficiency of the algorithms for the distribution of requests, and then finding of the efficiency extremum functions at each moment of time. The proposed method for calculating the exact boundaries of efficiency assumes that the configuration of the software system, the sequence of requests in time and their complexity are known. The algorithm used by the method for calculating the boundaries has exponential time complexity and is difficult to solve for real configurations of software systems. Thus, this method for evaluating the efficiency of algorithms for dispatching http requests is unsuitable for use in engineering problems.

Key words: load balancing algorithms, load balancing, efficiency evaluation, precise efficiency boundaries, requests distribution plan.

For citation: Dubovik M. V., Smelov V. V., Timonovich G. L. Exact boundaries of the efficiency of the algorithms for distributing requests between servers. *Proceedings of BSTU, issue 3, Physics and Mathematics. Informatics*, 2021, no. 2 (248), pp. 80–86 (In Russian).

Введение. В статье [1] описывается математическая модель для анализа алгоритмов распределения запросов между серверами (далее ARD-алгоритмы, от англ. algorithms for requests distributing) в программной системе. Формально модель описывается четверкой:

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

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

Если кратко рассматривать модель, то каждый клиент $C \in \{C_j, j = \overline{1, \bar{j}}\}$ представляет собой 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.

Балансировщик, в свою очередь, представляет собой четверку:

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

где P – процедура распознавания категории запроса; I_B – процедура обработки запроса; A – алгоритм выбора сервера для обработки запроса; O_B – процедура обработки ответа.

Сервер $S \in \{S_i, i = \overline{1, \bar{i}}\}$ представляет собой пятерку:

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

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; O_S – процедура обработки ответа.

Критерием эффективности ε ARD-алгоритма на интервале времени $(0, t)$ в [1] принято отношение:

$$\varepsilon(t) = \frac{M(t)}{L(t)}, \quad (5)$$

где $L(t)$ – количество поступивших в балансировщик запросов; $M(t)$ – количество поступивших в балансировщик ответов к заданному моменту времени t . При этом предполагается, что чем это отношение ближе к 1, тем более эффективным считается алгоритм.

В общем случае эффективность ARD-алгоритма зависит от конфигурации программной системы (количество серверов и их производительность, интенсивность клиентских запросов и их трудоемкость и т. д.) и может изменяться в течение времени. Очевидно, что оценка эффективности алгоритма имеет смысл для заданной конфигурации и на данный момент времени: сравнение эффективности двух ARD-алгоритмов для разных конфигураций программной системы и/или в разные моменты времени, может дать различные результаты.

Для оценки $E_A(t)$ эффективности ARD-алгоритма A удобно иметь оценки нижней $\underline{E}(t)$ и верхней $\bar{E}(t)$ границ эффективности ARD-алгоритма, вычисленные при заданных условиях:

$$P = \left\langle \left\{ C_j, j = \overline{1, \bar{j}} \right\}, \left\{ K_l, l = \overline{1, \bar{l}} \right\}, \left\{ S_i, i = \overline{1, \bar{i}} \right\} \right\rangle. \quad (6)$$

Графически верхняя и нижняя границы эффективности ARD-алгоритма представлены на рис. 1.

Рис. 1. Границы эффективности ARD-алгоритмов

Очевидно, что в любой момент времени t эффективность любого ARD-алгоритма $E_A(t) \leq 1$ и в качестве верхней границы можно было бы выбрать функцию $\bar{E}(t) \equiv 1$, но эта оценка не дает возможности ответить на вопрос: достижима ли в принципе (теоретически) при заданных условиях P заданная эффективность ε . Нижняя граница $\underline{E}(t) \equiv 0$ тоже малоинформативная, поэтому в качестве оценки нижней границы целесообразно выбрать эффективность какого-нибудь известного и просто реализуемого алгоритма, например Round Robin [2].

Интерес представляет величина, отражающая теоретический максимум эффективности ARD-алгоритмов. Далее такую оценку $\bar{E}(t)$ будем называть точной верхней границей эффективности ARD-алгоритмов. При этом следует отметить еще раз, что точная оценка верхней границы зависит от заданных условий P , поэтому, чтобы подчеркнуть это, будем использовать обозначение $\bar{E}_P(t)$.

Ниже рассматривается математическая постановка задачи вычисления точной верхней границы $\bar{E}_P(t)$ эффективности ARD-алгоритмов и предложен метод ее вычисления.

Основная часть. Очевидно, что теоретический максимум $\bar{E}_P(t)$ эффективности на заданном участке времени $[0, t_e]$ может быть достигнут алгоритмом A в том случае, если он может предсказывать моменты времени и трудоемкости поступающих от клиентов запросов, а также производительности серверов и моменты времени завершения обработки серверами запросов. В этом случае, задача вычисления функции $\bar{E}_P(t)$ сводится к задачам поиска и анализа планов распределения между серверами поступающих от клиентов запросов (далее для краткости – ARD-планов).

На рис. 2 представлен пример построения ARD-плана на отрезке дискретного времени $[0, 10]$. План описывает распределение между двумя серверами запросов, поступающих от двух клиентов.

---t---	C00	C01	S00[k=01,h=01]	S01[k=02,h=02]	L	M	M/L	h[0]	h[1]	
-- 0:	1	3	0	0	2	0	0	00	01	--- 0: C00->S00 C01->S01
-- 1:	0	3	1	0	3	1	0,33	00	01	--- 1: C01->S00
-- 2:	1	0	0	0	4	1	0,25	00	00	--- 2: C00->S01
-- 3:	0	3	0	0	5	1	0,2	00	00	--- 3: C01->X
-- 4:	0	3	1	1	6	3	0,5	00	01	--- 4: C01->S00
-- 5:	1	0	0	0	7	3	0,43	00	00	--- 5: C00->S01
-- 6:	1	3	0	1	9	4	0,44	00	01	--- 6: C00->X C01->X
-- 7:	0	3	1	1	10	6	0,6	00	02	--- 7: C01->S00
-- 8:	1	3	0	0	12	6	0,5	00	01	--- 8: C00->S01 C01->X
-- 9:	1	0	0	0	13	6	0,46	00	01	--- 9: C00->X
-- 10:	1	0	1	1	14	8	0,57	00	02	--- 10: C00->S00

Рис. 2. Пример ARD-плана

Собственно, план на рис. 2 представлен в крайнем правом столбце, содержащем символическое описание распределения запросов в каждый момент времени $t \in [0, 10]$. При этом моменты времени t обозначены в плане первым слева столбцом и продублированы в последнем. Например, запись «0: C00 → S00 C01 → S01» обозначает, что в момент времени $t = 0$ запрос клиента 0 направлен на обработку серверу 0 и запрос клиента 1 направлен серверу 1, запись «4: C01 → S00» соответствует отправке запроса клиента 1 серверу 0 в момент времени $t = 4$.

Трудоемкость запросов указывается в столбцах с заголовками C00 и C01. Например, в момент времени $t = 0$ поступило два запроса: от клиента 0 с трудоемкостью 1 и от клиента 1 с трудоемкостью 3. В момент времени $t = 4$ поступает один запрос от клиента 1 с трудоемкостью 3.

Моменты времени завершения обработки серверами запросов описываются в столбцах с заголовками S00 и S01. Каждый сервер характеризуется h – максимальным количеством подключений в один момент времени и k – коэффициентом трудоемкости выполнения запросов. Сервер 0 в каждый момент времени может обслуживать только одного клиента ($h = 1$), а продолжительность выполнения им запросов совпадает с их трудоемкостью ($k = 1$). Сервер 1 может одновременно обслуживать двух клиентов ($h = 2$), при этом продолжительность выполнения запросов в два раза больше их трудоемкости ($k = 2$). Например, если в момент времени $t = 0$ поступил запрос от клиента 0 с трудоемкостью 1 и этот запрос распределен серверу 0, то сервер может его исполнять только в том случае, если он в этот момент не обслуживает другого клиента, и выполнять его он будет 1 единицу времени. Если от клиента 1 в этот же момент времени $t = 0$ поступит запрос с трудоемкостью 3 и он будет направлен для обработки серверу 1, то сервер примет его к исполнению (так как он не занят обработкой других запросов) и будет исполнять его 6 единиц времени. В столбцах S00 и S01 указывается количество обработанных запросов (отправленных

ответов) в этот момент времени. Например, сервер 0 в момент времени $t = 1$ завершит обработку одного запроса, поступившего в момент времени $t = 0$, а сервер 1 в момент времени $t = 6$ завершит обработку запроса трудоемкости 3, поступившего на обработку в момент времени $t = 0$.

Столбцы с заголовками $h[0]$ и $h[1]$ описывают количество неиспользованных серверами подключений в каждый момент времени. Например, в момент времени $t = 4$ сервер 0 больше не может принимать на обработку запросы (значение равно 0), а сервер 1 может принять на обработку еще один запрос.

Для каждого момента времени подсчитывается с нарастающим итогом количество поступивших от клиентов запросов (столбец L), количество обработанных запросов (M) и эффективность (M/L). Например, на момент завершения времени $t = 4$ в систему поступило 6 запросов, обработано из них 3, эффективность системы $3/6 = 0,5$.

Обратим внимание, в плане (рис. 2) на момент времени $t = 3$: запись «3: C01 → X» обозначает, что в момент времени $t = 3$ алгоритм распределения не смог по каким-то причинам (будет пояснение ниже) выбрать сервер для обработки поступившего от клиента 1 запроса. Этот случай соответствует отказу в обслуживании и потере системой запроса, что приводит к снижению производительности. Заметим, что отказы есть и в другие моменты времени $t = 6, 8, 9$.

Во всех случаях имеется возможность назначить сервер 1 ($h[1] > 1$) для обработки запроса. Объясняется это свойством применяемого алгоритма планирования, который распределяет запросы между серверами строго равномерно, не учитывая предельную их загруженность. Стоит отметить, что при распределении запросов выбираемые в плане серверы чередуются.

На рис. 3 приведены примеры еще двух ARD-планов, в основе которых задача равномерного распределения запросов.

---t---:C00	C01	S00[k=01,h=01]	S01[k=02,h=02]	L	M	M/L	h[0]	h[1]	
0:	1	3	0	0	2	0	00	01	--- 0: C00->S00 C01->S01
1:	0	3	1	0	3	1	00	01	--- 1: C01->S00
2:	1	0	0	0	4	1	00	00	--- 2: C00->S01
3:	0	3	0	0	5	1	00	00	--- 3: C01->X
4:	0	3	1	1	6	3	00	01	--- 4: C01->S00
5:	1	0	0	0	7	3	00	00	--- 5: C00->S01
6:	1	3	0	1	9	4	00	00	--- 6: C00->S01 C01->X
7:	0	3	1	1	10	6	00	01	--- 7: C01->S00
8:	1	3	0	1	12	7	00	00	--- 8: C00->S01 C01->S01
9:	1	0	0	0	13	7	00	00	--- 9: C00->X
10:	1	0	1	1	14	9	00	01	---10: C00->S00
11:	0	0	1	0	14	10	01	01	---11:
12:	0	0	0	0	14	10	01	01	---12:
13:	0	0	0	0	14	10	01	01	---13:
14:	0	0	0	1	14	11	01	02	---14:

a

---t---:C00	C01	S00[k=01,h=01]	S01[k=02,h=02]	L	M	M/L	h[0]	h[1]	
0:	1	3	0	0	2	0	00	01	--- 0: C00->S01 C01->S00
1:	0	3	0	0	3	0	00	00	--- 1: C01->S01
2:	1	0	0	1	4	1	00	00	--- 2: C00->S01
3:	0	3	1	0	5	2	00	00	--- 3: C01->S00
4:	0	3	0	1	6	3	00	00	--- 4: C01->S01
5:	1	0	0	0	7	3	00	00	--- 5: C00->X
6:	1	3	1	0	9	4	00	00	--- 6: C00->S00 C01->X
7:	0	3	1	1	10	6	01	00	--- 7: C01->S01
8:	1	3	0	0	12	6	00	00	--- 8: C00->S00 C01->X
9:	1	0	1	0	13	7	00	00	--- 9: C00->S00
10:	1	0	1	1	14	9	01	00	---10: C00->S01
11:	0	0	0	0	14	9	01	00	---11:
12:	0	0	0	1	14	10	01	01	---12:
13:	0	0	0	1	14	11	01	02	---13:
14:	0	0	0	0	14	11	01	02	---14:

б

Рис. 3. Примеры рациональных ARD-планов:

а – распределение запросов по серверам начинается с сервера 0;

б – распределение запросов по серверам начинается с сервера 1

Но в отличие от плана, приведенного на рис. 2, на рис. 3 в случае предельной загрузки выбранного сервера осуществляется попытка направить запрос на следующий в последовательности сервер. И лишь в том случае, когда все перебранные серверы не могут принять запрос, происходит отказ от обслуживания запроса.

Такие планы будем в дальнейшем называть рациональными ARD-планами, алгоритмы их построения – рациональными алгоритмами планирования и ARD-алгоритмы, которые распределяют запросы в соответствии с рациональными ARD-планами, – рациональными ARD-алгоритмами.

Два плана на рис. 3 построены на основе рационального алгоритма планирования и отличаются выбором сервера в момент времени $t = 0$. При этом в конечный момент времени $t = 10$ в обоих случаях производительность системы совпала ($\varepsilon_0(10) = \varepsilon_1(10) = M(10) / L(10) = 0,64$), но в другие моменты времени она разная.

Разницу в производительностях системы $\varepsilon_0(t)$, $\varepsilon_1(t)$ на каждый момент времени при распределении запросов по планам, представленным на рис. 3, можно наблюдать на рис. 4.

Рис. 4. Изменение производительности программной системы при распределении запросов по планам, представленным на рис. 3

По всей видимости, функции $\underline{E}(t) = \min(\varepsilon_0(t), \varepsilon_1(t))$ и $\overline{E}(t) = \max(\varepsilon_0(t), \varepsilon_1(t))$, где $t = 1, 10$, $\varepsilon_0(t)$, $\varepsilon_1(t)$ – значения в столбцах M/L планов (рис. 3), их можно рассматривать как точные (нижнюю и верхнюю) границы рационального ARD-алгоритма при заданных условиях (рис. 5).

Пусть $A = \{\alpha_1, \alpha_2, \dots, \alpha_k\}$ множество всех рациональных ARD-алгоритмов, а $\Pi = \{\pi_1, \pi_2, \dots, \pi_k\}$ множество всех рациональных ARD-планов при заданных условиях P . Очевидно, что любой алгоритм $\alpha \in A$ реализует один из планов $\pi \in \Pi$.

При этом каждый план $\pi_i \in \Pi$, $i = \overline{1, k}$ характеризуется функцией эффективности $\varepsilon_i(t) = \frac{M_i(t)}{L_i(t)}$ на отрезке времени $[0, t_e]$. Тогда функции $\underline{E}_P(t) = \min_i \varepsilon_i(t)$ и $\overline{E}_P(t) = \max_i \varepsilon_i(t)$ являются точными (нижней и верхней) границами эффективности для множества алгоритмов A на отрезке времени $[0, t_e]$ при заданных условиях P .

Рис. 5. Точные границы эффективности для рационального ARD-алгоритма: a – нижняя граница; $б$ – верхняя граница

Рассмотрим пример вычисления точных границ эффективности для всех рациональных ARD-алгоритмов в программной системе (условия P) с конфигурацией, описанной в табл. 1.

Вычисление точных границ $\underline{E}_P(t)$ и $\overline{E}_P(t)$ при заданных условиях P (табл. 1) на отрезке времени $[0, 10]$ сводится к двум задачам:

– построение множества $\Pi = \{\pi_1, \pi_2, \dots, \pi_k\}$ всех рациональных ARD-планов;

– анализ функций эффективности $\varepsilon_i(t) = \frac{M_i(t)}{L_i(t)}$, $i = \overline{1, k}$, $t \in [0, t_e]$ планов и построение

функций $\underline{E}_P(t) = \min_i \varepsilon_i(t)$, $\overline{E}_P(t) = \max_i \varepsilon_i(t)$ точных границ.

Построение множества Π по сути является задачей рекурсивного перебора и осуществляется пошагово для каждого момента $t = \overline{1, 10}$ дискретного времени: $\Pi_0, \Pi_1, \dots, \Pi_{10}$, где

$\Pi_t = \{\pi_{t,1}, \pi_{t,2}, \dots, \pi_{t,k_t}\}$ – множество планов, построенное на шаге t . При этом $\Pi_{t+1} = \Phi(\Pi_t)$, где Φ – процедура построения множества всевозможных планов на основе множества планов, полученных на предыдущем шаге. Несложно вычислить, что максимально возможное количество планов, которое может быть получено на шаге $t = 10$ при крайних условиях (в каждый момент времени 2 запроса, серверы обеспечивают неограниченное количество подключений), составляет 4^{11} и растет экспоненциально по закону $(c \cdot s)^{t+1}$, где c – количество клиентов, s – количество серверов, t – момент дискретного времени.

Таблица 1

Конфигурация программной системы

Общие параметры	
Количество клиентов	2
Количество серверов	2
Количество категорий запросов	2
Время моделирования, ед. модельного времени	10
Категории	
1	$K_1 = 1$
2	$K_3 = 3$
Клиенты $C = \langle K, \lambda(t) \rangle$	
1	$K = K_1$, последовательность запросов (рис. 2)
2	$K = K_3$, последовательность запросов (рис. 2)
Серверы $S = \langle h, k, F, I_S, O_S \rangle$	
1	$h = 1, k = 1, F(t) \equiv 0, I_S, O_S$ – не используются
2	$h = 2, k = 2, F(t) \equiv 0, I_S, O_S$ – не используются
Балансировщик $B = \langle P, I_B, A, O_B \rangle$	
Любой рациональный ARD-алгоритм	

В нашем случае количество разных планов составляет $|\Pi_{10}| = 47$. Другими словами, существует 47 различных способов рационального распределения запросов (рис. 2) на отрезке времени $[0, 10]$. Все полученные планы можно разбить на пять групп (табл. 2) по величине $\varepsilon(10)$ – значению эффективности в конечный момент времени $t = 10$.

Таблица 2

Группы планов по значению эффективности

$\varepsilon(10)$	Количество планов	$\max_{0 \leq t \leq 10} \varepsilon(t)$
0,43	3	0,60
0,50	4	0,60
0,57	20	0,60
0,64	16	0,64
0,71	4	0,71

На рис. 6, $a, б, в, г, д$ представлены графики 47 функций, сгруппированных по семействам (табл. 2).

График на рис. 6, $е$ отображает точную нижнюю $\underline{E}_P(t) = \min_i \varepsilon_i(t)$ и точную верхнюю $\overline{E}_P(t) = \max_i \varepsilon_i(t)$ границы эффективности рациональных ARD-алгоритмов при заданных условиях P (табл. 1).

Рис. 6. Графики функции:
 $a - \varepsilon(t), t = \overline{0,10}, \varepsilon(10) = 0,43$; $б - \varepsilon(t), t = \overline{0,10}, \varepsilon(10) = 0,5$; $в - \varepsilon(t), t = \overline{0,10}, \varepsilon(10) = 0,57$;
 $г - \varepsilon(t), t = \overline{0,10}, \varepsilon(10) = 0,64$; $д - \varepsilon(t), t = \overline{0,10}, \varepsilon(10) = 0,71$; $е - \underline{E}_P(t) = \min_i \varepsilon_i(t), \overline{E}_P(t) = \max_i \varepsilon_i(t), t = \overline{0,10}$

На рис. 6, е обратим внимание на следующее: значения функций $\underline{E}_P(2) = \underline{E}_P(2)$ – совпадают точке $t = 2$; максимальное значение эффективности на отрезке времени $[0, 10]$ достигается в точке $t = 10$ ($\overline{E}_P(10) = 0,71$); на отрезке времени $[6, 10]$ эффективность не ниже значения $0,42$ ($\overline{E}_P(t) \geq 0,42, t \in [6, 10]$). Другими словами, на отрезке времени $[0, 10]$ при условиях P любой рациональный ARD-алгоритм в точке $t = 2$ имеет эффективность, равную $0,25$; достигает максимальной эффективности в точке $t = 10$; имеет эффективность не менее $0,42$ на отрезке времени $[6, 10]$. На рис. 7 совмещены рис. 6, е и 4.

Рис. 7. Точные границы рационального алгоритма равномерного распределения запросов (рис. 3), не выходя за пределы функций $\overline{E}_P(t)$ и $\underline{E}_P(t)$ (рис. 6, е) на отрезке времени $t = \overline{0,10}$

Стоит отметить, что графики функций точных границ рационального алгоритма равномерного распределения запросов соответствуют всем описанным здесь свойствам.

Заключение. Предложенный метод вычисления точных границ эффективности ARD-алгоритмов предполагает, что помимо конфигурации принимающей и обрабатывающей клиентские запросы программной системы, известна последовательность (во времени) запросов и их трудоемкость. В реальных условиях может быть известна только конфигурация и, в лучшем случае, интенсивности запросов по категориям трудоемкости их обслуживания. Алгоритм для вычисления границ имеет экспоненциальную сложность от времени t и является трудноразрешимым для реальных конфигураций программных систем (включающих более 10 серверов и несколько десятков одновременно поступающих запросов). Кроме того, как правило, интерес представляет эффективность программной системы в установившемся режиме, стабилизация работы которой может занимать достаточно продолжительный интервал времени. Все это делает непригодным применение предложенного метода для оценки эффективности ARD-алгоритмов в инженерных задачах, когда требуется оценить эффективность существующей программной системы. Оценка эффективности в таких случаях, как правило, осуществляется с помощью нагрузочного тестирования, которое позволяет на основе собранной статистики получить обоснованную оценку.

Интерес представляют адаптивные ARD-алгоритмы, которые «умеют приспосабливаться» к изменениям конфигурации программной системы, интенсивности и структуры потока

запросов. В этом случае важен процесс адаптации, который, по сути, является переходным процессом и в результате которого система минимизирует потерю эффективности (в случае неблагоприятных изменений) или, наоборот, максимизирует эффективность при благоприятных изменениях. Переходной процесс

может быть описан как множество сценариев (планов) адаптации системы на ограниченном (относительно небольшом) отрезке времени. При этом можно вычислить точные границы эффективности, которые позволят оценить «коридор возможностей» адаптационного алгоритма.

Список литературы

1. 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.
2. 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.

References

1. Dubovik M. V., Smelov V. V. Mathematical model for analyzing algorithms for distributing requests between servers. *Trudy BGTU* [Proceedings of BSTU], issue 3, Physics and Mathematics. Informatics, 2021, no. 1 (242), pp. 31–35 (In Russian).
2. 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.

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 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

Тимонович Георгий Леонидович – кандидат технических наук, доцент кафедры информационных систем и технологий. Белорусский государственный технологический университет (220006, г. Минск, ул. Свердлова, 13а, Республика Беларусь). E-mail: tgl@hmti.ac.by

Information about the authors

Dubovik Marina Vladimirovna – Master's degree student, the Department of Information Systems and Technologies. 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

Timonovich Georgy Leonidovich – PhD (Engineering), Associate Professor, the Department of Information Systems and Technologies. Belarusian State Technological University (13a, Sverdlova str., 220006, Minsk, Republic of Belarus). E-mail: tgl@hmti.ac.by

Поступила после доработки 14.09.2021