УДК 004.777
DOI: 10.26102/2310-6018/2025.48.1.013

Разработка методов ограничения запросов к API в зависимости от классов потребителей

Р.М. Селезнёв✉

Новосибирский государственный технический университет, Новосибирск, Российская Федерация

Резюме. Ограничение запросов является важной частью управления доступностью и надежностью API. На сегодняшний день существует ряд подходов к реализации механизма ограничения запросов, каждый из которых основывается на определенном алгоритме или их комбинации. Однако существующие подходы зачастую рассматривают всех потребителей как однородную группу, что препятствует созданию гибких стратегий управления ресурсами в условиях современных распределенных архитектур. В данной статье автором предложены два новых метода ограничения запросов, основанные на алгоритме маркерной корзины. Первый метод предусматривает использование общей маркерной корзины с различными требованиями к минимальному уровню ее заполнения в зависимости от класса потребителей. Второй метод предполагает использование для каждого класса потребителей отдельных маркерных корзин со своими значениями параметров, но общим лимитом. Проведенное моделирование подтвердило, что оба метода позволяют обеспечить эффективное ограничение запросов к API, при этом были выявлены различия между методами в характере распределения ресурсов среди разных классов потребителей. Полученные результаты представляют практическую ценность для разработчиков информационных систем и сервисов, которым необходимо поддерживать высокий уровень доступности, одновременно обеспечивая гарантии доступа для разных категорий потребителей.

Ключевые слова: ограничение запросов, алгоритм маркерной корзины, программный интерфейс, класс потребителя, квота, порог, всплеск трафика.

Для цитирования: Селезнёв Р.М. Разработка методов ограничения запросов к API в зависимости от классов потребителей. *Моделирование, оптимизация и информационные технологии.* 2025;13(1). URL: https://moitvivt.ru/ru/journal/pdf?id=1803 DOI: 10.26102/2310-6018/2025.48.1.013

Development of API rate limiting methods based on consumer classes

R.M. Seleznev✉

Novosibirsk State Technical University, Novosibirsk, the Russian Federation

Abstract. Rate limiting is a crucial aspect of managing the availability and reliability of APIs. Today, there are several approaches to implementing rate limiting mechanisms, each based on specific algorithms or their combinations. However, existing methods often treat all consumers as a homogeneous group, hindering the creation of flexible resource management strategies in modern distributed architectures. In this article, the author proposes two new methods for rate limiting based on the token bucket algorithm. The first method involves using a shared token bucket with different minimum fill requirements depending on the consumer class. The second method suggests using separate token buckets for each consumer class with individual parameter values but a common limit. Simulation results confirmed that both methods enable efficient API request limitation, though disparities emerged regarding resource distribution patterns across diverse consumer classes. These findings have practical implications for developers of information systems and services who need to maintain high availability while ensuring access guarantees for various consumer categories.

© Селезнёв Р.М., 2025

Keywords: rate limiting, token bucket algorithm, software interface, consumer class, quota, threshold, burst traffic.

For citation: Seleznev R.M. Development of API rate limiting methods based on consumer classes. *Modeling, Optimization and Information Technology*. 2025;13(1). (In Russ.). URL: https://moitvivt.ru/ru/journal/pdf?id=1803 DOI: 10.26102/2310-6018/2025.48.1.013

Введение

Современные информационные системы и сервисы все чаще полагаются на программные интерфейсы приложений (API), чтобы предоставлять данные и функциональность другим системам [1]. Однако неограниченный доступ к таким ресурсам может привести к росту затрат, перегрузке серверов, снижению производительности и даже отказу системы в целом. Для предотвращения подобных негативных последствий применяется механизм ограничения запросов (rate limiting). Этот механизм позволяет контролировать количество запросов, поступающих к сервису за определенный промежуток времени, тем самым предотвращая злоупотребление ресурсами и защищая систему от атак типа DDoS [2–4].

Традиционные методы ограничения запросов к API основаны на использовании фиксированных лимитов для всех категорий пользователей, что предполагает одинаковый подход ко всем потребителям независимо от их ожиданий относительно качества предоставляемых услуг. Хотя некоторые исследования предлагают внедрение адаптивных механизмов, учитывающих такие факторы, как критичность вызываемого API, типы выполняемых операций и их вычислительная сложность [5–7], основное внимание в этих подходах уделяется обеспечению устойчивости системы при высоких нагрузках. При этом не принимается во внимание неоднородность потребителей и различия в их требованиях к уровню обслуживания.

Методы ограничения запросов, основанные исключительно на фильтрации трафика по IP-адресам, типам устройств или временным интервалам [8], также не могут адекватно учесть приоритетность различных групп потребителей. Эти методы больше ориентированы на защиту от внешних угроз, чем на обеспечение распределения ресурсов между авторизованными потребителями.

Целью данного исследования является разработка и оценка новых методов, направленных на повышение эффективности управления запросами к API путем учета различий в приоритетах потребителей. Эти методы должны позволить минимизировать риски истощения ресурсов низкоприоритетными потребителями, сохраняя при этом высокий уровень доступности для высокоприоритетных.

Материалы и методы

Алгоритм маркерной корзины

Для решения задачи ограничения запросов могут применяться различные алгоритмы. Среди них наиболее известными являются [9]:

– алгоритм маркерной корзины (token bucket);
– алгоритм дырявого ведра (leaking bucket);
– счетчик фиксированных интервалов (fixed window counter);
– журнал скользящих интервалов (sliding window log);
– счетчик скользящих интервалов (sliding window counter).

Каждый алгоритм обладает определенными достоинствами и недостатками. Так, к преимуществам метода с фиксированными интервалами можно отнести его простоту и эффективное использование памяти, но в то же время ключевым недостатком данного

метода является его склонность к превышению установленных квот при высоких уровнях трафика вблизи границ интервалов [9]. Алгоритмы же, основанные на использовании очередей сообщений и динамическом расчете скользящих интервалов, при присущей им большей точности ограничения трафика характеризуются значительным потреблением памяти [10] и могут демонстрировать низкую эффективность в условиях резких колебаний трафика.

Алгоритм маркерной корзины лишен этих недостатков и с учетом простоты его реализации стал одним из наиболее популярных [5, 11, 12]. Его успешно применяют в проектах с открытым исходным кодом (open source) и облачных решениях крупных технологических компаний[1]. Остановимся на нем более подробно.

Основные компоненты алгоритма (Рисунок 1):

– Маркерная корзина. Это контейнер с заранее заданным размером, который регулярно наполняют маркерами. В случае полного заполнения корзины новые маркеры в нее не добавляются.

– Механизм проверки и уменьшения маркеров. При поступлении каждого запроса к API выполняется проверка, достаточно ли маркеров в корзине: если маркеров достаточно (в корзине содержится хотя бы один маркер), из корзины извлекается один маркер и запрос обрабатывается; если маркеров недостаточно, то запрос отклоняется.

Основные параметры алгоритма:

– Размер корзины (емкость). Это максимальное количество маркеров, которое может находиться в корзине одновременно;

– Скорость пополнения. Это скорость, с которой маркеры добавляются в корзину (число маркеров в единицу времени).

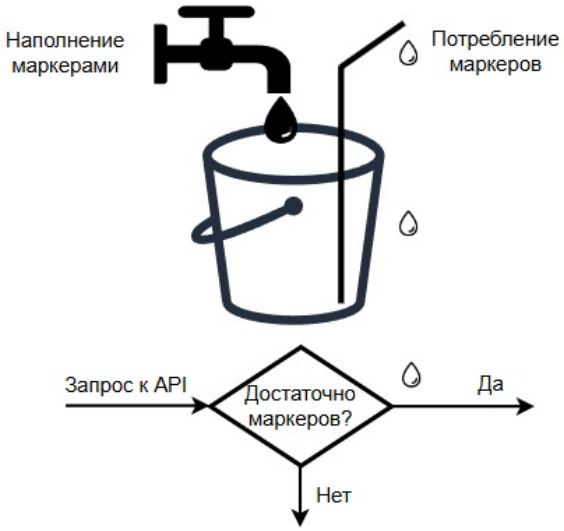

Рисунок 1 – Алгоритм маркерной корзины
Figure 1 – Token bucket algorithm

Основной проблемой классического алгоритма маркерной корзины, равно как и других рассмотренных ранее альтернатив, является отсутствие дифференцированного подхода к различным категориям потребителей. В результате потребители с низким приоритетом могут исчерпать доступные маркеры, препятствуя доступу со стороны более приоритетных потребителей. Так, например, незарегистрированные пользователи,

[1] Примеры использования алгоритма маркерной корзины представлены в технической документации: https://www.envoyproxy.io/docs/envoy/latest/configuration/http/http_filters/local_rate_limit_filter и https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-request-throttling.html

а также пользователи, которым услуги предоставляются бесплатно [13], могут использовать всю квоту, лишая доступа тех, кто имеет платную подписку.

Создание индивидуальных корзин для каждого отдельно взятого потребителя частично устраняет эту проблему, однако это не всегда применимо, значительно повышает потребление памяти и, при этом, сохраняются риски злоупотребления со стороны большого числа низкоприоритетных потребителей или потенциальных злоумышленников. Реализация отдельных конечных точек (endpoints) для различных категорий потребителей способствует снижению вероятности блокировки запросов от высокоприоритетных потребителей, однако этот подход ограничивает гибкость архитектурного решения, усложняя дальнейшую модификацию количества поддерживаемых классов потребителей.

Для преодоления этих недостатков предлагается разработать новые методы, сохраняющие достоинства алгоритма маркерной корзины и устраняющие его ограничения. Ниже рассматриваются две предлагаемые модификации данного алгоритма, отвечающие названным требованиям.

Метод 1: общая корзина с секциями

Первый метод предполагает использование одной общей маркерной корзины для всех классов потребителей. Для успешной обработки запроса к API необходимо наличие определенного количества маркеров в корзине, причем требование к количеству маркеров зависит от класса потребителя.

Основные компоненты алгоритма (Рисунок 2):

– Секционная маркерная корзина. Это контейнер с заранее заданным размером, который регулярно наполняют маркерами. В случае полного заполнения корзины новые маркеры в нее не добавляются. Особенность данной корзины состоит в том, что по мере ее наполнения появляется возможность извлекать маркеры из все более высоких секций, закрепленных за разными классами потребителей, от более приоритетного класса к менее приоритетному.

– Механизм проверки и уменьшения маркеров. При поступлении каждого запроса к API определяется класс потребителя, после чего выполняется проверка, достаточно ли маркеров в корзине для данного класса: если маркеров достаточно (в корзине маркеров не меньше, чем установленное пороговое значение), из корзины извлекается один маркер и запрос обрабатывается; если маркеров недостаточно, то запрос отклоняется.

В данном методе в дополнение к стандартным параметрам алгоритма маркерной корзины (размеру корзины и скорости пополнения) вводится параметр «пороговое значение», назначаемый каждому классу потребителей. Принцип: чем более приоритетный класс потребителей, тем ниже пороговое значение. Это соотношение может быть выражено следующей формулой:

$$\forall i \in \{1,2,\dots,N-1\}: P_{i+1} < P_i, T_{i+1} > T_i, \quad (1)$$

где i – порядковый номер класса потребителя, N – количество классов потребителей (общее число секций в корзине), P_i – приоритет для i-го класса, T_i – пороговое значение для i-го класса.

Например, для первого класса может быть достаточно наличия хотя бы одного маркера в корзине[2], для второго класса требуется заполнение корзины минимум на 24 %, а для третьего класса – минимум на 62 %.

[2] Такое значение позволит минимизировать число отклонённых запросов, поступивших от потребителей первого класса.

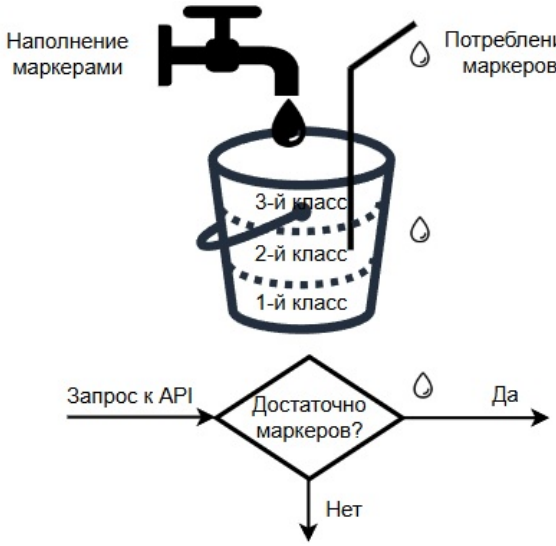

Рисунок 2 – Алгоритм маркерной корзины с общей корзиной (на примере поступления запроса от потребителя второго класса)
Figure 2 – Token bucket algorithm with a shared bucket (example of a request from a second-class consumer)

Метод 2: отдельные корзины с общим лимитом

Второй метод предусматривает создание отдельных маркерных корзин для каждого класса потребителей. Для успешной обработки запроса к API необходимо наличие маркеров в корзине, относящейся к соответствующему классу потребителя.

Основные компоненты алгоритма (Рисунок 3):

– Набор маркерных корзин. Это контейнеры с заранее заданными размерами, которые регулярно и независимо друг от друга наполняют маркерами. В случае полного заполнения какой-либо корзины новые маркеры в нее не добавляются. Каждому классу потребителей назначается своя корзина. Каждая корзина из набора в общем случае имеет свой собственный размер и скорость пополнения, что позволяет независимо регулировать доступность сервисов для различных типов потребителей.

– Механизм проверки и уменьшения маркеров. При поступлении каждого запроса к API определяется класс потребителя, после чего выполняется проверка, достаточно ли маркеров в корзине, назначенной для данного класса: если маркеров достаточно (в корзине содержится хотя бы один маркер), из данной корзины извлекается один маркер и запрос обрабатывается; если маркеров недостаточно, то запрос отклоняется.

В данном методе в дополнение к стандартным параметрам алгоритма маркерной корзины (размеру корзины и скорости пополнения) вводится параметр «общий лимит», позволяющий исключить возможность превышения допустимого для API лимита. Общий лимит ограничивает сверху суммарный размер всех корзин, что можно описать формулой:

$$\sum_{i=1}^{N} S_i \leq M, \quad (2)$$

где i – порядковый номер корзины, N – количество классов потребителей (общее число корзин), S_i – размер i-й корзины, M – общий лимит для API.

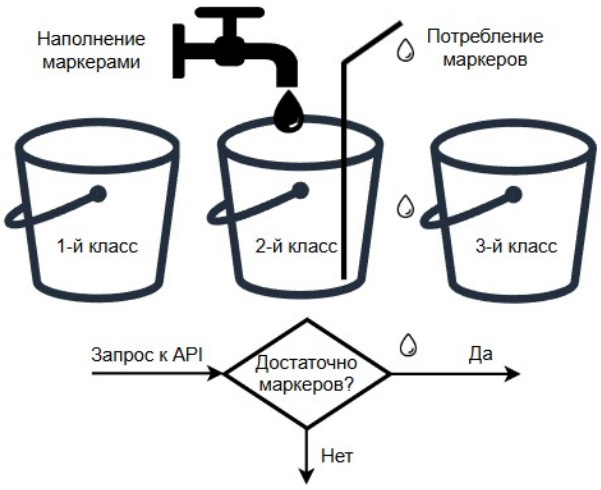

Рисунок 3 – Алгоритм маркерной корзины с отдельными корзинами (на примере поступления запроса от потребителя второго класса)
Figure 3 – Token bucket algorithm with separate buckets (example of a request from a second-class consumer)

Результаты

Для оценки эффективности предложенных методов было выполнено имитационное моделирование с использованием симуляции сетевого трафика. В рамках эксперимента рассматривались три класса потребителей с разной интенсивностью запросов и отличающимися требованиями к доступности ресурсов. В силу отсутствия влияния на работу алгоритмов конкурентный доступ дополнительно не имитировался.

Сценарий 1: равномерная нагрузка

В данном сценарии интенсивность запросов оставалась постоянной для всех трех классов потребителей. Оба метода продемонстрировали высокую эффективность при обработке запросов, однако первый из них показал смещение доли успешных ответов в пользу более приоритетных классов. При стабильно высокой нагрузке и истощении маркеров в системе первым начал испытывать трудности в доступе к ресурсам третий класс, за ним последовал второй. Второй метод обеспечил более сбалансированное распределение ресурсов среди всех классов потребителей, предотвращая чрезмерную концентрацию доступа у одного из них. Полученные результаты представлены в левой части Рисунка 4.

Сценарий 2: неравномерная нагрузка

Во втором сценарии проводились три независимых эксперимента, каждый из которых имитировал резкий рост трафика от отдельно взятого класса потребителей. С целью обеспечения сопоставимости полученных данных общее количество обращений к API и продолжительность симуляции были аналогичны значениям, использованным в первом сценарии.

Оба метода справились с неравномерной нагрузкой, подтвердив заметное влияние всплесков трафика на долю успешных запросов соответствующего класса. Тем не менее, характер реакции на пиковые нагрузки зависел от используемого метода (Рисунок 4).

Всплески трафика от более приоритетных классов в случае применения первого метода значительно ограничивали доступ к ресурсу для менее приоритетных потребителей. В то же время второй метод сохранял квоты на ресурсы для всех классов,

обеспечивая их доступность даже в условиях повышенных требований со стороны высокоприоритетных клиентов.

Всплеск трафика от менее приоритетных классов также увеличивал долю успешных запросов этих классов. Однако в случае первого метода выраженность этого увеличения снижается по мере падения приоритета. Второй метод продемонстрировал, при прочих равных параметрах, примерно одинаковое влияние всплесков на увеличение числа успешных запросов вне зависимости от приоритетности класса.

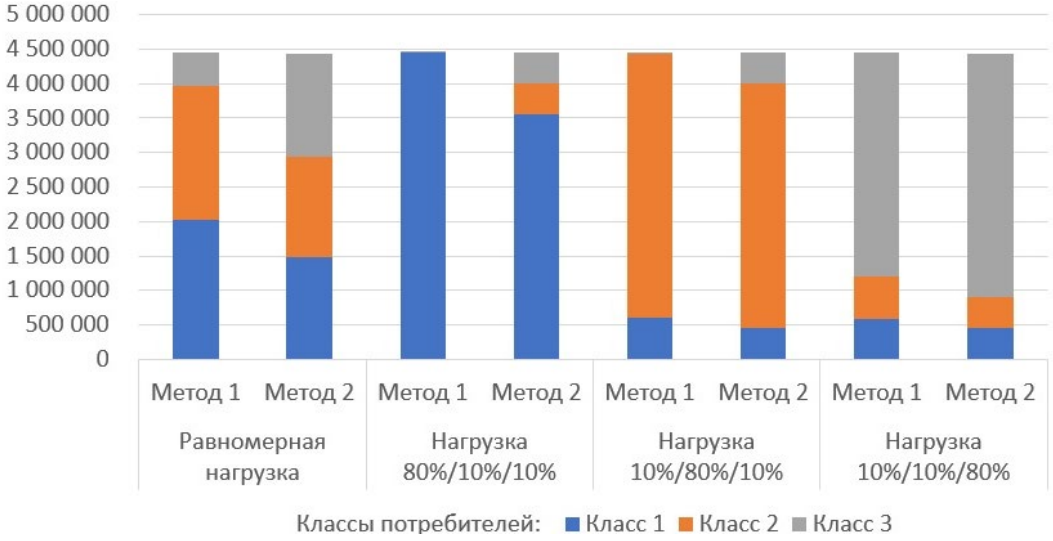

Рисунок 4 – Результаты моделирования предлагаемых методов в условиях различной нагрузки
Figure 4 – Simulation results of the proposed methods under different load conditions

Обсуждение

Проведенное исследование показало, что оба предложенных метода демонстрируют высокую эффективность в управлении трафиком к API в различных сценариях нагрузки. Однако выбор оптимального метода зависит от специфических условий эксплуатации системы.

Первый метод выглядит более предпочтительным в ситуациях, где необходимо обеспечить максимальную защиту критически важных сервисов путем выделения большего объема ресурсов наиболее приоритетным потребителям. Это может быть особенно полезно в системах, где доступность и надежность ключевых компонентов имеют первостепенную важность, а высокоприоритетные потребители должны иметь гарантии доступа даже при аномальном увеличении нагрузки со стороны остальных классов потребителей.

Второй метод продемонстрировал свою ценность в контексте обеспечения справедливого распределения ресурсов между всеми классами потребителей, когда недопустимо создание условий, при которых одни потребители получают существенно больше ресурсов, чем другие. Это делает его оптимальным выбором для систем, ориентированных на предсказуемый доступ к сервисам, а также для систем, предоставляющих своим потребителям гарантии доступа независимо от уровня их приоритета.

Заключение

В данной работе были предложены два новых метода ограничения запросов на основе алгоритма маркерной корзины. Они позволяют дифференцированно подходить к

управлению доступом к ресурсам в зависимости от важности потребителей, демонстрируя тем самым имеющийся потенциал для улучшения существующих подходов. Однако следует отметить, что использование предложенных методов требует тщательной настройки параметров, таких как количество и размер корзин, скорость пополнения и пороговые значения. Это может вызывать определенные сложности для адаптации под конкретные условия эксплуатации.

Дальнейшие исследования могут быть направлены на устранение этих ограничений и изучение возможностей интеграции этих методов с другими механизмами управления трафиком, такими как автоматическое масштабирование ресурсов. Кроме того, перспективным направлением является разработка гибридных решений, объединяющих сильные стороны обоих представленных методов.

СПИСОК ИСТОЧНИКОВ / REFERENCES

1. Firmani D., Leotta F., Mecella M. On Computing Throttling Rate Limits in Web APIs through Statistical Inference. In: *2019 IEEE International Conference on Web Services (ICWS), 08–13 July 2019, Milan, Italy*. IEEE; 2019. pp. 418–425. https://doi.org/10.1109/ICWS.2019.00075
2. Alharbi S.J., Moulahi T. API Security Testing: The Challenges of Security Testing for Restful APIs. *International Journal of Innovative Science and Research Technology*. 2023;8(5):1485–1499. https://doi.org/10.5281/zenodo.7988410
3. Serbout S., Malki A.E., Pautasso C., Zdun U. API Rate Limit Adoption – A pattern collection. In: *EuroPLoP '23: Proceedings of the 28th European Conference on Pattern Languages of Programs*, *05–09 July 2023, Irsee, Germany*. New York: Association for Computing Machinery; 2024. https://doi.org/10.1145/3628034.3628039
4. Wanda P., Hiswati M.E. Belief-DDoS: stepping up DDoS attack detection model using DBN algorithm. *International Journal of Information Technology*. 2024;16(1):271–278. https://doi.org/10.1007/s41870-023-01631-x
5. Padma Latha V.L., Sudhakar Reddy N., Suresh Babu A. On optimizing scalability and availability of cloud based software services using scale rate limiting algorithm. *International Journal of Nonlinear Analysis and Applications*. 2022;13(2):1893–1905. https://doi.org/10.22075/ijnaa.2022.27403.3588
6. Park J., Park J., Jung Y., Lim H., Yeo H., Han D. TopFull: An Adaptive Top-Down Overload Control for SLO-Oriented Microservices. In: *ACM SIGCOMM '24: Proceedings of the ACM SIGCOMM 2024 Conference, 04–08 August, 2024, Sydney, Australia*. New York: Association for Computing Machinery; 2024. pp. 876–890. https://doi.org/10.1145/3651890.3672253
7. Zhou H., Chen M., Lin Q., Wang Y., She X., Liu S., Gu R., Ooi B.C., Yang J. Overload Control for Scaling WeChat Microservices. In: *SoCC '18: Proceedings of the ACM Symposium on Cloud Computing, 11–13 October 2018, Carlsbad, USA*. New York: Association for Computing Machinery; 2018. pp. 149–161. https://doi.org/10.1145/3267809.3267823
8. El Malki A., Zdun U., Pautasso C. Impact of API Rate Limit on Reliability of Microservices-Based Architectures. In: *2022 IEEE International Conference on Service-Oriented System Engineering (SOSE), 15–18 August 2022, Newark, USA*. IEEE; 2022. pp. 19–28. https://doi.org/10.1109/SOSE55356.2022.00009
9. Сюй А. *System Design. Подготовка к сложному интервью*. Санкт-Петербург: Питер; 2024. 304 с.
   Xu A. *System Design Interview – An Insider's Guide.* Saint Petersburg: Piter; 2024. 304 p. (In Russ.).

10. Bass L., Clements P., Kazman R. *Software Architecture in Practice*. *4th Edition.* Boston: Addison-Wesley Professional; 2021. 464 p.
11. Barnhart B., Brooker M., Chinenkov D., Hooper T., Im J., Jha P.C., Kraska T., Kurakula A., Kuznetsov A., McAlister G., Muthukrishnan A., Narayanan A., Terry D., Urgaonkar B., Yan J. Resource Management in Aurora Serverless. *Proceedings of the VLDB Endowment*. 2024;17(12):4038–4050. https://doi.org/10.14778/3685800.3685825
12. Kaldor J., Mace J., Bejda M., Gao E., Kuropatwa W., O'Neill J., Ong K.W., Schaller B., Shan P., Viscomi B., Venkataraman V., Veeraraghavan K., Song Y.J. Canopy: An End-to-End Performance Tracing and Analysis System. In: *SOSP '17: Proceedings of the 26th Symposium on Operating Systems Principles, 28 October 2017, Shanghai, China*. New York: Association for Computing Machinery; 2017. pp. 34–50. https://doi.org/10.1145/3132747.3132749
13. Lin Y.-W., Lin T.-X., Farn C.-K. The Free-of-Charge Phenomena in the Network Economy – A Multi-Party Value Exchange Model. *Journal of Theoretical and Applied Electronic Commerce Research*. 2021;16(7):2981–3002. https://doi.org/10.3390/jtaer16070163

ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

Селезнёв Роман Михайлович, магистр техники и технологии, Новосибирский государственный технический университет, Новосибирск, Российская Федерация.
e-mail: r.m.seleznev@gmail.com
ORCID: 0009-0006-0036-5976

Roman M. Seleznev, Master of Science, Novosibirsk State Technical University, Novosibirsk, the Russian Federation.

Статья поступила в редакцию 16.01.2025; одобрена после рецензирования 27.01.2025; принята к публикации 01.02.2025.

The article was submitted 16.01.2025; approved after reviewing 27.01.2025; accepted for publication 01.02.2025.