

УДК 004.75

DOI: [10.26102/2310-6018/2026.61.10.020](https://doi.org/10.26102/2310-6018/2026.61.10.020)

Поддержка принятия решений по управлению вычислительными ресурсами программных сервисов на основе временных рядов

Ю.А. Воронцов^{1✉}, А.В. Калач^{1,2}, Ю.Н. Золотарев²

¹МИРЭА – Российский технологический университет, Москва, Российская Федерация

²Воронежский институт ФСИН России, Воронеж, Российская Федерация

Резюме. В работе рассматривается задача оценки текущей потребности микросервиса в вычислительных ресурсах на основе данных мониторинга, представляемых в виде временных рядов, результаты которой затем могут быть использованы в процессе принятия решений по выделению вычислительных ресурсов. Под потребностью понимается требуемая интенсивность потребления наиболее востребованного вычислительного ресурса, обеспечивающая обработку поступающих задач при заданном окне дискретизации. Предлагается практический алгоритм, включающий: выбор информативных метрик с применением корреляционного анализа; синхронизацию, нормализацию и приведение метрик к единой дискретной шкале; очистку телеметрии от выбросов, квантования и пропусков с восстановлением значений; обнаружение точек разладки и сегментацию на режимы работы; формирование базовой линии потребления и восстановление ряда потребности $r(t)$. Для построения $r(t)$ рассматриваются два семейства методов. В бесклассовой постановке оценивается удельная ресурсоемкость запроса по расходу ресурса и счетчикам входящих/обработанных задач, вводится экспоненциальное сглаживание, а параметры сглаживания адаптируются при смене режимов. В классовой постановке вводится разбиение запросов по типам и решается регрессионная задача для оценивания стоимостей классов с использованием NNLS и онлайн-RLS с фактором забывания. Приводится методика экспериментальной проверки на воспроизводимом микросервисном стенде с контролируемыми сценариями нагрузки, сравнением точности относительно эталонного расхода и оценкой вычислительных затрат, что делает результаты применимыми для мониторинга и масштабирования.

Ключевые слова: оценка потребности в ресурсе, микросервисы, метрики мониторинга, временные ряды, обнаружение разладки, NNLS, RLS.

Для цитирования: Воронцов Ю.А., Калач А.В., Золотарев Ю.Н. Поддержка принятия решений по управлению вычислительными ресурсами программных сервисов на основе временных рядов. *Моделирование, оптимизация и информационные технологии.* 2026;14(10). URL: <https://moitvvt.ru/ru/journal/article?id=2633> DOI: 10.26102/2310-6018/2026.61.10.020

Support for making decisions on managing computational resources of software services based on time series

Y.A. Vorontsov^{1✉}, A.V. Kalach^{1,2}, Yu.N. Zolotarev²

¹MIREA – Russian Technological University, Moscow, the Russian Federation

²Voronezh Institute of the Federal Penitentiary Service of Russia, Voronezh,
the Russian Federation

Abstract. The paper considers the task of estimating the current demand for computing resources by a micro-service based on monitoring data represented as time series, the results of which can then be used in decision-making on the allocation of computing resources. The demand is understood as the required intensity of consumption of the most in-demand computing resource, which ensures the processing of

incoming tasks within a specified sampling window. A practical algorithm is proposed, which includes the following steps: selecting informative metrics using correlation analysis; synchronizing, normalizing, and converting metrics to a single discrete scale; cleaning telemetry from outliers, quantization, and missing values, and restoring them; detecting points of failure and segmenting them into operating modes; generating a baseline for consumption and restoring the demand series $r(t)$. Two families of methods are considered for building the demand series, $r(t)$. In the unsuper-vised setting, the specific resource consumption of a request is estimated based on the resource consumption and the counts of incoming/processed tasks, and exponential smoothing is introduced, with the smoothing parameters adapting to changes in modes. In the supervised setting, requests are divided into types/endpoints, and a regression problem is solved to estimate the costs of classes using NNLS and online RLS with a forgetting factor. The paper presents a methodology for experimental validation using a reproducible microservice testbed with controlled load scenarios, comparing the accuracy against the reference consumption, and estimating computational costs, making the results applicable for monitoring and scaling purposes.

Keywords: resource requirement estimation, micro-services, monitoring metrics, time series, outlier detection, NNLS, RLS.

For citation: Vorontsov Y.A., Kalach A.V., Zolotarev Yu.N. Support for making decisions on managing computational resources of software services based on time series. *Modeling, Optimization and Information Technology*. 2026;14(10). (In Russ.). URL: <https://moitvvt.ru/ru/journal/article?id=2633> DOI: 10.26102/2310-6018/2026.61.10.020

Введение

В последние годы интеллектуальные приложения достигли прорывных результатов благодаря быстрому развитию искусственного интеллекта и Интернета вещей (IoT), продемонстрировав мощные возможности в таких областях, как обработка естественного языка, распознавание изображений, интеллектуальные рекомендации и других [1].

В отличие от традиционных, такие задачи, генерируемые интеллектуальными приложениями, как правило, требуют больших вычислительных ресурсов и чувствительны к задержкам, в то время как ограниченные по ресурсам терминальные устройства и облачные серверы с высокой задержкой связи не могут себе этого позволить. Потенциальным решением являются граничные вычисления, которые переносят определенные сервисы из удаленного облака на периферию сети и обеспечивают параллельную обработку задач, тем самым повышая эффективность обработки [2]. Тем не менее, относительно ограниченные ресурсы периферии создают беспрецедентные проблемы, такие как таймаут обработки и конкуренция за ресурсы, что затрудняет обеспечение надежных интеллектуальных сервисов.

Распределение вычислительных ресурсов представляет собой актуальную прикладную задачу повышения качества обслуживания. Кроме того, в связи с быстрым развитием Интернета масштабы центров обработки данных, являющихся ключевой вычислительной инфраструктурой, поддерживающей цифровую экономику государства, значительно увеличились в последние годы, что привело к росту общих эксплуатационных расходов и энергопотребления. По оценкам, к 2030 году общее энергопотребление глобальных центров обработки данных достигнет около 848 ТВт·ч [3].

Исходя из возрастающих потребностей в количестве вычислительных ресурсов, актуальной становится задача оценки потребности в ресурсах для программных сервисов, разворачиваемых на вычислительной инфраструктуре. Результаты этой оценки могут быть использованы при принятии решений о планировании количества вычислительных ресурсов, позволяющего сохранить требуемый уровень надежности и

доступности сервиса для конечного пользователя. Дополнительную актуальность задача приобретает в микросервисных средах, которые активно используются для развертывания, где приложение состоит из множества связанных сервисов, динамически масштабируемых и развертываемых в контейнерах [4, 5]. В таких системах профиль запросов и, следовательно, потребность в ресурсе может быстро меняться как из-за поведения пользователей, так и из-за внутренних зависимостей [5, 6].

Целью работы является исследование возможности оценивать потребность микросервиса в вычислительном ресурсе по телеметрии, рассматривая ее как временной ряд, который можно очистить от артефактов, разделить на интервалы режимов и построить временной ряд потребности в ресурсе.

Материалы и методы

Под потребностью в ресурсе $r(t)$ будем понимать требуемую интенсивность потребления вычислительного ресурса (ядер центрального процессора (ЦП), МБ оперативной памяти, полосы пропускания системы ввода/вывода и т. д.), необходимую для обработки задач, поступивших в момент времени t . Наблюдаемые метрики формируют множество временных рядов $x(t)$.

На практике временные ряды телеметрии являются нестационарными и содержат выбросы, пропуски, а также могут содержать неверные данные из-за задержек доставки [7, 8]; кроме того, смена профиля запросов и окружения, связанная с релизами новых версий, переносом контейнеров сервиса между узлами (миграцией) и другими причинами, приводит к разным режимам работы [9, 10]. Поэтому перед построением $r(t)$ необходимо обработать временные ряды, чтобы избежать возможных проблем с данными.

Предлагаемый алгоритм включает: определение наиболее значимых метрик вычислительных ресурсов; синхронизацию и нормализацию для получения устойчивых временных рядов; нахождение точек разладки и разбиение данных на интервалы режимов для корректного учета смены нагрузки; построение ряда потребности.

Первым этапом является определение требуемых метрик для отслеживания микросервисов. На вход на данном этапе подается согласованный по времени набор рядов метрик. Выходом этапа является единая дискретная сетка наблюдений с шагом Δt и синхронизированные значения $x(t_k)$ [7, 11].

Поскольку набор доступных метрик, особенно в микросервисной инфраструктуре, может включать в себя десятки и сотни временных рядов, поэтому целесообразно выделить наиболее информативные показатели нагрузки и потребления ресурса [12, 13]. Минимально достаточный набор метрик можно сгруппировать по нескольким типам: нагрузка, производительность, вычислительные ресурсы и ограничения, качество. Метрики нагрузки могут быть представлены счетчиками входящих запросов/сообщений, интенсивностью поступления запросов/сообщений в очередь, длиной очереди, задержкой обработки/возрастом сообщений в очереди. Производительность представляется счетчиками завершенных задач, пропускной способностью или интенсивностью обработки сообщений в очереди, оценки времени обслуживания запроса/сообщения. Ресурсы и ограничения включают в себя метрики использования различных вычислительных ресурсов; а также лимиты/запросы на ресурсы для контейнеров, квоты для контейнеров и виртуальных машин и число реплик [11, 12]. Качество представляется $p95/p99$ (95/99 перцентиль) задержки, интенсивностью ошибок, таймаутами [14, 15]. Ключевые требования к рядам: синхронизация по времени, приведение к общей дискретной сетке, контроль пропусков и согласованные единицы измерения [7, 11].

При этом важным замечанием является то, что при обработке запросов/сообщений может использоваться разный набор ресурсов. Время обработки задачи может в первую очередь зависеть от ЦП, оперативной памяти, подсистемы ввода/вывода или сети. Это обуславливается типом поступающих в сервис задач, которые могут быть ориентированы на вычисления (CPU-bound) или же на получение данных через систему ввода/вывода (IO-bound). Чтобы избежать избыточного количества временных рядов метрик и сохранить интерпретируемость, на данном этапе целесообразно выбрать доминирующий ресурс/набор ресурсов для последующего построения $r(t)$ для этих ресурсов. Поскольку задача выглядит как поиск согласованности между метриками нагрузки/производительности/качества и вычислительных ресурсов, возможным способом решения является применение корреляционного анализа между метриками. Результатом является матрица корреляций, по которой исключаются дублирующие и слабосвязанные метрики, упрощая дальнейшую обработку [13].

На основании матрицы корреляций можно выделить ключевые параметры, в наибольшей степени связанные с потреблением ресурса, и тем самым сократить размерность входа для этапов очистки и моделирования. После отбора метрик фиксируется набор входных рядов $r_{fact}(t)$, для которых будет строиться ряд потребности $r(t)$. В дальнейшем корреляционный анализ можно использовать и для мониторинга смены режима: резкое изменение структуры корреляций часто указывает на изменение профиля работы [16].

Следующим этапом является очистка и восстановление временных рядов метрик. Метрики содержат артефакты измерений: кратковременные всплески, пропуски, задержки доставки и квантование [7, 8]. Помимо этого, такие метрики как латентность обработки запросов характеризуются распределением с тяжелыми хвостовыми значениями, поэтому некорректные точки могут существенно исказить оценку текущего состояния системы [15].

Практически применимая процедура очистки включает в себя несколько этапов обработки данных для восстановления временных рядов [7, 8]. В первую очередь необходимо привести все данные к временному шагу Δt и выровнять временные отметки. После этого необходимо заполнить пропуски с помощью ограниченного заполнения или интерполяции на малых разрывах. При этом большие разрывы должны маркироваться как недостоверные интервалы и не применяться при решении задач [7]. Следующим шагом необходимо избавиться от выбросов путем подавления аномальных пиков медианным или Хэмпеловским фильтром, либо усечением по перцентилям [7, 8]. Также при необходимости можно прибегнуть к экспоненциальному сглаживанию или сглаживанию при помощи скользящего среднего для устойчивой оценки метрик интенсивности [7].

Исходя из данных, полученных на предыдущем этапе, можно переходить к построению ряда потребностей в вычислительном ресурсе путем вычисления, сколько ресурсов израсходовано в каждый момент времени, исходя из ряда $r_{fact}(t)$, а также, сколько ресурсов требовалось на самом деле для обработки всех поступивших задач. Поскольку имеются взаимосвязанные ряды $\mu(t)$ и $r_{fact}(t)$ необходимо предложить такую модель, которая позволит объяснить изменение потребления ресурса в зависимости от количества обрабатываемых задач и, исходя из этого, получить ряд потребности в вычислительном ресурсе $r(t)$.

Перед тем, как рассматривать саму модель, необходимо определить разницу между ресурсами, которые могут быть использованы при обработке. Основное отличие между ресурсами заключается в способе их разделения. Принято разбивать

вычислительные ресурсы на разделяемые по времени или потоковые, и разделяемые по пространству или емкостные. Такое разбиение напрямую влияет на то, как необходимо оценивать потребность в ресурсе при обработке задач.

Потоковые вычислительные ресурсы определяют скорость выполнения работы. Подобные ресурсы естественным образом измеряются как «единицы ресурса в секунду» и связаны с тем, сколько работы система должна выполнить за интервал времени. К потоковым ресурсам обычно относят ЦП в виде времени ЦП/с, пропускной способности сети в виде байт/с, некоторые характеристики дисковой подсистемы, в частности, пропускной способности в виде байт/с или же OPS/с.

Емкостные вычислительные ресурсы характеризуют объем данных или состояния, который система должна удерживать одновременно. Они измеряются в абсолютных единицах объема и зависят не от количества поступивших запросов, а от количества одновременно обрабатываемых запросов, размеров обрабатываемых наборов данных и состояния сервиса. К емкостным ресурсам обычно относят оперативную память в байтах, различные логические емкости: размер кэша, число соединений в пуле, лимиты очередей и т. п. Такие вычислительные ресурсы требуют дополнительного шага вычислений при определении требуемого объема для обработки поступающих задач.

В текущей работе предлагается рассматривать несколько подходов, которые основываются на определении среднего потребления вычислительного ресурса на выполнение одной задачи, а также на построение регрессионной модели на основе зависимости между $\mu(t)$ и $r_{fact}(t)$.

Первым предлагаемым способом построения ряда $r(t)$ является получение в каждый момент времени удельной ресурсоемкости одной задачи в единицах ресурсо-времени. В общем виде она определяется как отношение наблюдаемого расхода ресурса к числу обработанных задач за тот же интервал, с защитой от деления на ноль:

$$r_{task}^{raw}(t) = \frac{r_{fact}(t) - b(t)}{\max(N_{out}(t), \varepsilon)}, \quad (1)$$

где ε предотвращает деление на ноль, $b(t)$ – базовая потребность, описывающая ресурс на обслуживание сервиса и накладные расходы, не зависящие напрямую от числа задач. Такая постановка позволяет рассматривать ресурсоемкость задачи как некоторое среднее значение для всех типов задач, выполняемых отдельным сервисом. В качестве базового приближения можно считать, что сервис выполняет ограниченный набор схожих операций, поэтому на первом шаге допустимо пренебречь различиями между типами задач.

Такая постановка характерна для разделяемых по времени ресурсов, однако для разделяемых по пространству ресурсов необходимо вести дополнительную величину $N_{process}(t) = \max(0, N_{process}(t-1) + N_{in} - N_{out}(t))$, которая показывает количество запросов, находящихся в обработке. Тогда для ресурсов, разделяемых по пространству, емкость одной задачи:

$$r_{task}^{raw}(t) = \frac{r_{fact}(t) - b(t)}{\max(N_{process}(t), \varepsilon)}. \quad (2)$$

На основании этого потребность для разделяемых по времени ресурсов представляется как сумма базовой составляющей и ресурса на выполнение поступивших задач:

$$r(t) = N_{in}(t) \cdot r_{task}(t) + b(t). \quad (3)$$

Важным уточнением при постановке задачи является то, как рассматривать характеристику поступления и обработки запросов сервисом. В изначальной постановке

выше указаны ряды $N_{in}(t)$, $N_{out}(t)$, которые формируются как приросты счетчиков запросов за окно Δt .

Пусть $N_{in}(t)$ и $N_{out}(t)$ – число задач, поступивших/обработанных за окно Δt , тогда $r_{fact}(t)$ также необходимо рассматривать как расход ресурса за окно, например, ЦП-секунды/окно. Тогда:

$$r_{task}^{raw}(t) = \frac{\max(r_{fact}(t) - b_t, 0)}{\max(N_{out}(t), \varepsilon)}, \quad (4)$$

$$r^{raw}(t) = b_t + N_{in}(t) \cdot r_{task}^{raw}(t). \quad (5)$$

Для повышения устойчивости в стационарных режимах вводится экспоненциально сглаженная (EWMA) оценка ресурсоемкости:

$$r_{task}^{ewma}(t) = (1 - \eta) \cdot r_{task}^{ewma}(t - 1) + \eta \cdot r_{task}^{raw}(t), \quad (6)$$

и соответствующий сглаженный ряд потребности:

$$r^{ewma}(t) = b_t + N_{in}(t) \cdot r_{task}^{ewma}(t). \quad (7)$$

Таким образом, далее можно использовать две параллельные оценки: $r^{raw}(t)$ и $r^{ewma}(t)$.

Для более гибкого сглаживания и аппроксимации временных рядов потребности внутри режима могут применяться регрессионные сплайны и В-сплайны со штрафами [17].

Метрики потока входящих и обработанных задач могут быть приведены к интенсивностям – $\lambda_{in}(t)$ и $\lambda_{out}(t)$ в запрос/с, тогда необходимо рассматривать $u_{fact}(t)$ в ядрах с тем же самым окном дискретизации, а вычисления будут выполняться в нормированных единицах. При этом связь с исходным вариантом задается соотношениями: $N_{in}(t) = \lambda_{in}(t)\Delta t$, $N_{out}(t) = \lambda_{out}(t)\Delta t$, $r_{fact}(t) = u_{fact}(t)\Delta t$, $b_t = \bar{b}(t)\Delta t$.

Тогда мгновенная удельная ресурсоемкость остается в тех же единицах ресурсо-времени на задачу:

$$r_{task}^{raw}(t) = \frac{\max(u_{fact}(t) - \bar{b}(t), 0)}{\max(\lambda_{out}(t), \varepsilon)}, \quad (8)$$

а потребность может быть выражена сразу в ядрах:

$$C^{raw}(t) = \bar{b}(t) + \lambda_{in}(t) \cdot r_{task}^{raw}(t), \quad (9)$$

$$C^{ewma}(t) = \bar{b}(t) + \lambda_{in}(t) \cdot r_{task}^{ewma}(t), \quad (10)$$

где $r_{task}^{ewma}(t)$ вычисляется той же формулой EWMA:

$$r_{task}^{ewma}(t) = (1 - \eta) \cdot r_{task}^{ewma}(t - 1) + \eta \cdot r_{task}^{raw}(t). \quad (11)$$

При необходимости переход обратно к ресурсо-времени на окно выполняется как $r(t) = C(t)\Delta t$.

Оба варианта эквивалентны при фиксированном Δt , так как интенсивность является нормированным приростом. На практике приросты счетчиков удобнее использовать, когда расчет ведется за окно определенной длины, а интенсивности предпочтительнее, когда нужно сопоставлять ряды при разных Δt и при этом сравнивать сервисы в одних единицах. Далее в работе допускается использование любого из двух представлений.

Однако такая постановка может приводить к некорректной оценке потребности в ресурсе, если не учитывать нестационарность. Даже после очистки ряды метрик

остаются нестационарными: смена профиля трафика, релизы, перемещение подов и борьба за ресурс на узле приводят к разным режимам работы. Для корректного моделирования $r(t)$ целесообразно выделять интервалы режимов, внутри которых статистические свойства ряда устойчивы [9, 18].

Границы режимов определяются детекторами разладок. Для практического применения подходят CUSUM [19], а также вероятностные методы с online-обновлением, например, ВОСРД [20]. Результатом определения точек разладки является разбиение временной оси на интервалы $\tau_i \leq t < \tau_{i+1}$. Поскольку сглаживание может негативно влиять на моментальные изменения вблизи разладки, при переходе на новый интервал необходимо увеличивать влияние новых наблюдений в оценке $r_{task}(t)$. Это реализуется одним из двух механизмов:

1. Сброс сглаженной оценки на границе режима: $r_{task}^{ewma}(\tau_i) \leftarrow r_{task}^{raw}(\tau_i)$.
2. Ускоренная адаптация после разладки:

$$\eta(t) = \begin{cases} \eta_{high}, & \tau_i \leq t < \tau_i + L \\ \eta_{low}, & \text{иначе} \end{cases}, \eta_{high} > \eta_{low}. \quad (12)$$

На каждом интервале $[\tau_i, \tau_{i+1})$ расчеты выполняются с параметрами сглаживания, согласованными с соответствующим режимом.

Отдельно в данной постановке стоит рассматривать обновление потребления ресурса на выполнение служебных задач окружения, в котором развертывается микросервис, закладываемое в $b(t)$. Поскольку служебные задачи расходуют ресурс вне зависимости от поступающих пользовательских запросов, можно проводить оценку на отрезках, где количество пользовательских запросов мало или равно нулю. Если запросов мало $N_t \leq N_{base}$, то $b(t)$ обновляется при помощи EWMA:

$$b_t = (1 - \eta_b)b_{t-1} + \eta_b R_t. \quad (13)$$

Если $N_t > N_{base}$, $b(t)$ фиксируется:

$$b_t = b_{t-1}. \quad (14)$$

Поскольку даже в рамках микросервисной архитектуры присутствует несколько вариантов задач, которые обрабатывает один сервис, необходимо также рассмотреть постановку задачи с учетом классификации запросов. При такой модели необходимо учитывать разные типы задач и их вклад в общую нагрузку на вычислительные ресурсы. Введение классов задач позволяет учитывать неоднородность запросов и устойчиво оценивать вклад каждого класса в суммарное потребление ресурса на основе телеметрии. Дополнительно потребность в классификации запросов выражается в том, что изменение профиля нагрузки приводит к смещению оценки требуемого ресурса. Для устранения этого эффекта в задачах оценки потребности в ресурсах применяется декомпозиция потребления ресурса по классам запросов на основе регрессионных моделей; при этом распространенным выбором является МНК-регрессия с ограничением неотрицательности коэффициентов (NNLS), обеспечивающим осмысленные оценки ресурсоемкости классов [5, 10].

В современных самоадаптивных системах оценка потребности выполняется непрерывно и должна сохранять качество в динамической среде, что отражено в подходах к непрерывной/самоадаптивной оценке потребности в вычислительных ресурсах [10].

Пусть сервис имеет K классов задач, а также, помимо общего числа запросов $N_{out}(t)$, на каждом окне Δt наблюдается число обработанных задач класса k в окне $N_{out,k}(t)$.

Тогда модель наблюдаемого расхода ресурса задается как:

$$r_{fact}(t) \approx b(t) + \sum_{k=1}^K c_k N_{out,k}(t) + \xi_t, \quad (15)$$

где $c_k \geq 0$ – удельная ресурсоемкость задач класса k , ξ_t – ошибка аппроксимации.

После оценки c_k ряд требуемого ресурса для выполнения всех поступивших задач строится по входным счетчикам (или интенсивностям):

$$r(t) = b(t) + \sum_{k=1}^K c_k N_{in,k}(t). \quad (16)$$

Такой подход соответствует стандартной постановке оценки потребности в ресурсах по агрегированным наблюдениям в задачах оценки потребности в ресурсах [5, 10].

Соберем данные на интервале из W окон $t \in t_1, \dots, t_W$. Определим:

- вектор наблюдений $y \in R^W$, $y_i = r_{fact}(t_i)$;
- матрицу признаков $X \in R^{W \times (K+1)}$: $X_i = [1, N_{out,1}(t_i), \dots, N_{out,K}(t_i)]$;
- вектор параметров $\theta = [b, c_1, \dots, c_K]^T$.

Тогда $y \approx X\theta$.

Поскольку базовая линия потребления и ресурсоемкости классов физически неотрицательны, используется ограничение $\theta \geq 0$:

$$\theta_{NNLS} = \arg \min_{\theta \geq 0} \|y - X\theta\|_2^2. \quad (17)$$

Использование NNLS-регрессии в задачах оценки потребности в ресурсах по мониторинговым данным является типичным приемом, поскольку ограничение неотрицательности устраняет нефизические решения и повышает интерпретируемость результатов.

Для онлайн-обновления параметров по мере поступления новых окон также часто применяется рекурсивный метод наименьших квадратов (RLS) с коэффициентом забывания $\lambda \in (0,1]$, что позволяет адаптировать параметры к изменению режима/профиля нагрузки без пересчета на всей истории. Для окна t вводятся $x_t = [1, N_{out,1}(t), \dots, N_{out,K}(t)]^T$, $y_t = r_{fact}(t)$. Стандартные рекуррентные обновления имеют вид:

$$K_t = \frac{P_{t-1}x_t}{\lambda + x_t^T P_{t-1} x_t}, \quad (18)$$

$$\hat{\theta}_t = \hat{\theta}_{t-1} + K_t(y_t - x_t^T \hat{\theta}_{t-1}), \quad (19)$$

$$P_t = \frac{1}{\lambda} (P_{t-1} - K_t x_t^T P_{t-1}), \quad (20)$$

где P_t – матрица доверия (например, $P_0 = \delta I$) [21].

Применимость RLS в вычислительных системах обоснована тем, что многие практические задачи требуют непрерывного обновления моделей взаимосвязи между нагрузкой/производительностью и ресурсом и реакцией на изменяющиеся условия выполнения задач. В работах по управлению ресурсами и энергоэффективностью модели производительности/мощности обновляются онлайн с использованием RLS. Аналогично, RLS применяется и в задачах точного выделения ЦП, где параметры модели обновляются по наблюдаемым метрикам в режиме выполнения [22]. В контексте оценки потребности в динамических средах задача непрерывной адаптации оценок и выбора/настройки методов подчеркивается в современных работах по самоадаптивной оценке потребности в ресурсе [10].

Неотрицательность в RLS базовой формы не гарантируется; для сохранения физического смысла используют проекцию после шага:

$$\hat{c}_k \leftarrow \max(\hat{c}_k, 0), \hat{b} \leftarrow \max(\hat{b}, 0). \quad (21)$$

В текущей версии работы многоклассовая постановка ограничивается классами входных задач сервиса.

Для экспериментальной проверки предложенного подхода использовался стенд микросервисного приложения, включающий генератор нагрузки, тестируемое приложение и подсистему мониторинга. Архитектура стенда приведена на Рисунке 1.

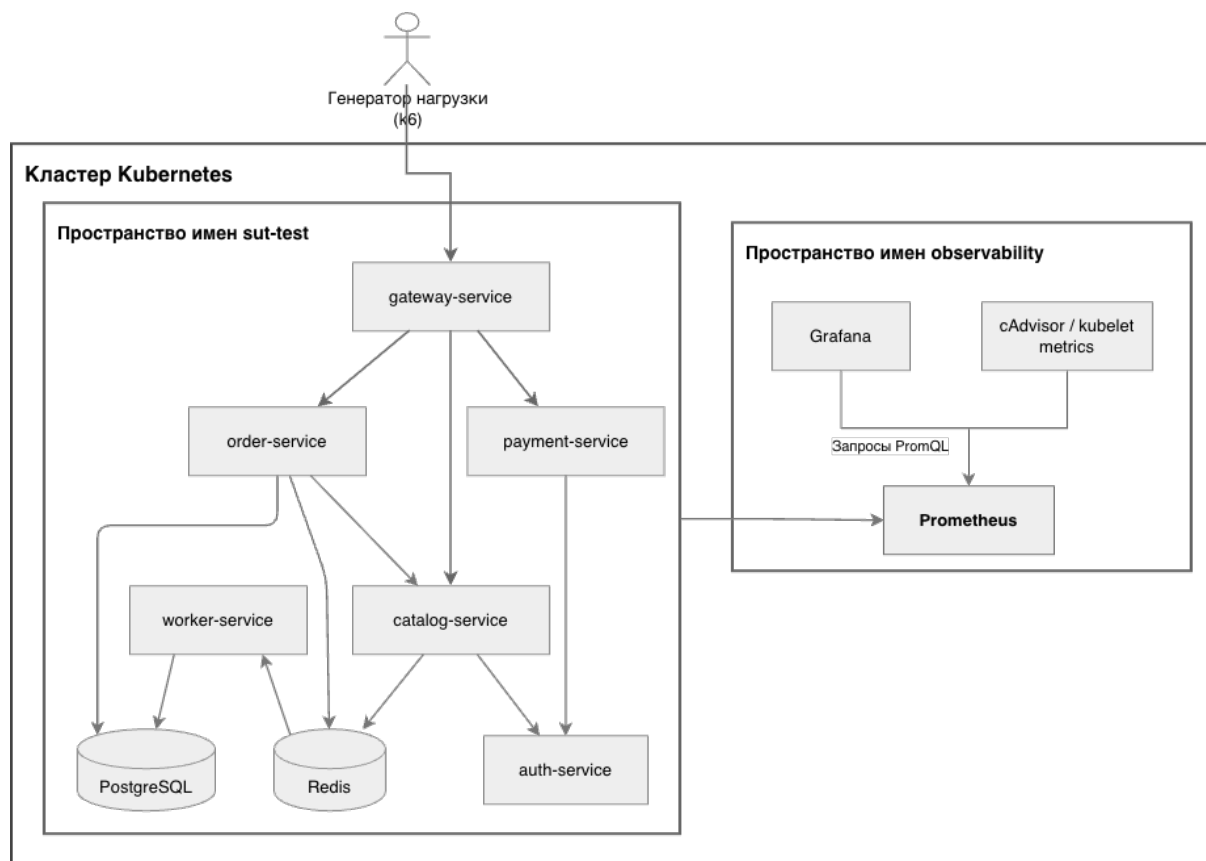


Рисунок 1 – Архитектура экспериментального стенда
Figure 1 – Architecture of the experimental stand

Пользовательская нагрузка формируется внешним генератором запросов и поступает через шлюз микросервисного приложения. Далее запросы распределяются между прикладными сервисами, часть из которых инициирует дополнительные межсервисные обращения и обращения к инфраструктурным компонентам хранения и обработки данных. Система мониторинга собирает метрики с сервисов и контейнеров, после чего эти метрики используются в модуле онлайн-оценки потребности в вычислительном ресурсе.

Результаты

В качестве примера рассматривался синхронный сервис обработки запросов catalog-service. В первую очередь был сформирован согласованный набор временных рядов метрик, включающий интенсивность входящих запросов, интенсивность обработанных запросов, число запросов в обработке, загрузку вычислительных ресурсов. Затем определялась зависимость между метриками вычислительных ресурсов и метриками запросов при помощи матрицы корреляции на основе метода Пирсона.

Тепловая карта корреляций представлена на Рисунке 2, а итоговый набор наиболее значащих метрик вычислительных ресурсов представлен в Таблице 1.

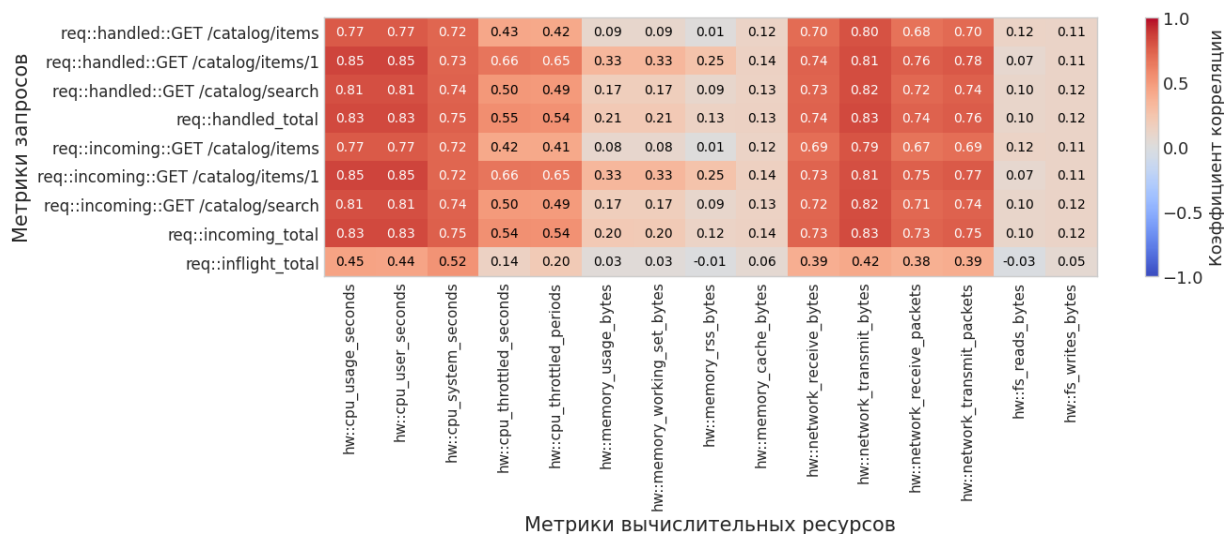


Рисунок 2 – Тепловая карта корреляции метрик
Figure 2 – Metric correlation heat map

Таблица 1 – Итоговый набор метрик вычислительных ресурсов
Table 1 – The final set of metrics for computing resources

Метрика	Назначение
cpu_usage_seconds	Суммарная интенсивность использования ЦП контейнером сервиса
network transmit bytes	Интенсивность исходящего сетевого трафика

Оценка устойчивости и точности методов выполнялась при помощи различных сценариев нагрузки, которые охватывают стационарные и переходные режимы. Эксперимент проводился с применением нескольких методов для сравнения их точности и применимости для построения ряда потребности в ресурсе в сравнении с действительным использованием ресурса. Были применены следующие методы: на основе средней стоимости одной задачи $r_{task}^{raw}(t)$, на основе сглаженной средней стоимости задачи $r_{task}^{ewma}(t)$ без учета точек разладки временного ряда, а также с учетом их при помощи ВОСРД, рекурсивный метод наименьших квадратов с фактором забывания (FFRLS), а также метод неотрицательных наименьших квадратов (NNLS). В дополнении к этому рассматривался поиск стоимости одной задачи без учета $b(t)$, путем простого деления $r_{fact}(t)$ на $\mu(t)$. Методы тестировались на данных с временным окном в 5 секунд, а также с разными значениями Δt между точками временного ряда. Поскольку временные ряды хранились в TSDB Prometheus, первичная обработка временного ряда выполнялась при помощи встроенных в него операций. На Рисунке 3 приведен фрагмент временного ряда потребления ЦП с результатами применения методов для воссоздания ряда.

В Таблице 2 приведены метрики точности MAE, RMSE и NMAE для восстановления ряда потребности в ресурсе ЦП.

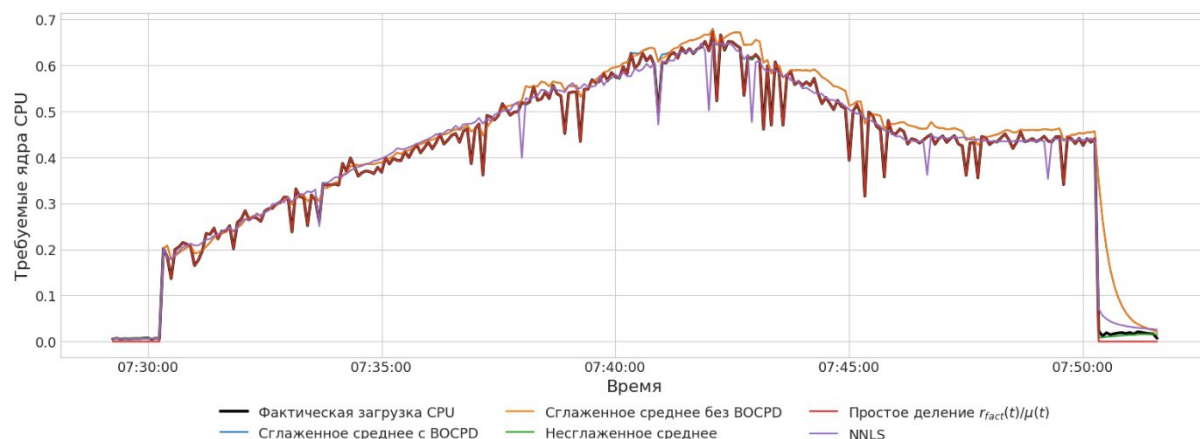


Рисунок 3 – Результаты применения методов
Figure 3 – Results of applying the methods

Таблица 2 – Метрики точности алгоритмов
Table 2 – Metrics of algorithm accuracy

Алгоритм	MAE	RMSE	NMAE
Несглаженное среднее	0,0006	0,0012	0,0852
Простое деление	0,0082	0,0239	0,2234
NNLS	0,0094	0,0265	0,7478
Сглаженное среднее без BOCPD	0,0182	0,5834	0,3300
Сглаженное среднее с BOCPD	0,0182	0,5834	0,3301
RLS	5,3449	19,9854	61,0465

Поскольку в рамках исследования важна не только точность, но и практическая применимость методов, особенно в режиме реального времени, отдельное внимание должно быть уделено вычислительной эффективности. В Таблице 3 представлены результаты сравнения методов по времени вычисления и дополнительным затратам CPU и памяти при обработке одной метрики с окном в 5 секунд и для 6486 точек.

Таблица 3 – Метрики затрат времени и ресурсов
Table 3 – Metrics of time and resource expenditure

Алгоритм	Время на ряд, с	Время на одну точку, мс	ЦП-время на 1 точку, мс	Пиковая память, МБ (средняя)
Простое деление	72,4248	11,1663	3,2468	3,9728
Несглаженное среднее	74,0358	11,4147	3,2056	3,9733
RLS	76,4034	11,7797	3,3687	3,9816
Сглаженное среднее без BOCPD	76,9779	11,8683	3,2406	3,9915
NNLS	100,886	15,5544	4,1138	4,0205
Сглаженное среднее с BOCPD	181,534	27,9887	4,3277	4,0023

Обсуждение

На основании результатов эксперимента можно говорить о подтверждении возможности восстановления устойчивой оценки потребности сервиса в ресурсе на основе агрегированных метрик мониторинга. Бесклассовая модель, основанная на вычислении средней удельной ресурсоемкости запроса, обеспечивает простую и вычислительно эффективную оценку потребности сервиса. Такой подход хорошо подходит для систем с однородным профилем запросов и позволяет получать достаточно точные оценки при минимальных вычислительных затратах. Однако при наличии значительной неоднородности типов запросов точность данного подхода может снижаться. В случае же рассмотрения классовой постановки применение методов регрессионной оценки, таких как NNLS и RLS с фактором забывания, позволяет корректно восстанавливать вклад различных классов запросов в суммарное потребление ресурса. При этом метод NNLS обеспечивает более устойчивые оценки и дает меньшую ошибку. Также важно отметить, что при внедрении рассматриваемых методов в контур мониторинга необходимо обеспечить возможность оценки потребности в вычислительном ресурсе за время, значительно меньшее, чем время, отводимое на принятие решения об изменении количества выделяемого вычислительного ресурса. Также важной характеристикой является размах колебаний значения потребности в вычислительном ресурсе, при больших колебаниях возможны слишком частые изменения выделяемого количества ресурсов, что негативно сказывается на работе сервиса и вычислительной инфраструктуры.

Заключение

В работе предложен и формализован практический алгоритм восстановления временного ряда потребности микросервиса в вычислительном ресурсе на основе метрик мониторинга. Алгоритм включает выбор наиболее информативных метрик, приведение данных к единой временной сетке, очистку временных рядов от выбросов и пропусков, а также сегментацию на режимы работы по точкам разладки. В рамках потоковых ресурсов потребность трактуется как требуемая интенсивность расхода ресурса в окне дискретизации, что позволяет напрямую сопоставлять оценку $r(t)$ с доступной мощностью и использовать ее для планирования. Для оценки потребности рассмотрены как бесклассовая постановка через удельную ресурсоемкость запроса со сглаживанием, так и многоклассовая постановка, где ресурс распределяется между устойчивыми типами операций сервиса. Для многоклассового случая проанализированы регрессионные методы NNLS и RLS, обеспечивающие интерпретируемые оценки ресурсоемкости классов и возможность онлайн-адаптации при изменении профиля нагрузки.

Предложенная экспериментальная методика на воспроизводимом микросервисном стенде позволяет сравнивать подходы по точности восстановления $r(t)$, устойчивости к нестационарности и вычислительным затратам, что важно для применения в реальном времени. Полученный ряд потребности может служить входом при принятии решений об автомасштабировании сервисов и в задачах планирования емкости вычислительных ресурсов, так как отделяет полезную нагрузку от базовой составляющей и снижает влияние шумов телеметрии. В качестве дальнейшего развития целесообразно рассмотреть аппроксимацию и краткосрочный прогноз $r(t)$ с учетом смены режимов, а также вычисление дефицита и профицита ресурса относительно выделенной мощности и заданного уровня риска. Дополнительным направлением является расширение модели на цепочки вызовов к нижестоящим сервисам и согласование оценок потребности между связанными сервисами.

СПИСОК ИСТОЧНИКОВ / REFERENCES

1. Chen L., Yang X., Sun H., Yu X., Zhi K. Dependent task offloading in multi-access edge computing: A GCN augmented deep reinforcement learning approach. *Computer Networks*. 2025;269:111445. <https://doi.org/10.1016/j.comnet.2025.111445>
2. Ni Zh., Chen H., Xue H., Li Zh., Chen N., Yu J. Joint optimization of service placement, task offloading and resource allocation for dependent subtasks in hierarchical edge computing systems. *Computer Networks*. 2025;275:111896. <https://doi.org/10.1016/j.comnet.2025.111896>
3. Mytton D., Ashtine M.I. Sources of data center energy estimates: A comprehensive review. *Joule*. 2022;6(9):2032–2056. <https://doi.org/10.1016/j.joule.2022.07.011>
4. Ahmad H., Treude Ch., Wagner M., Szabo C. Towards resource-efficient reactive and proactive auto-scaling for microservice architectures. *Journal of Systems and Software*. 2025;225(9):112390. <https://doi.org/10.1016/j.jss.2025.112390>
5. Turin G., Borgarelli A., Donetti S., Damiani F., Johnsen E.B., Tapia Tarifa S.L. Predicting resource consumption of Kubernetes container systems using resource models. *Journal of Systems and Software*. 2023;203(4):111750. <https://doi.org/10.1016/j.jss.2023.111750>
6. Park J., Choi B., Lee Ch., Han D. Graph Neural Network-Based SLO-Aware Proactive Resource Autoscaling Framework for Microservices. *IEEE/ACM Transactions on Networking*. 2024;32(4):3331–3346. <https://doi.org/10.1109/TNET.2024.3393427>
7. Tawakuli A., Havers B., Gulisano V., Kaiser D., Engel Th. Survey: Time-series data preprocessing: A survey and an empirical analysis. *Journal of Engineering Research*. 2025;13(2):674–711. <https://doi.org/10.1016/j.jer.2024.02.018>
8. Blázquez-García A., Conde A., Mori U., Lozano J.A. A Review on Outlier/Anomaly Detection in Time Series Data. *ACM Computing Surveys*. 2021;54(3):56. <https://doi.org/10.1145/3444690>
9. Gupta M., Wadhvani R., Rasool A. Comprehensive analysis of change-point dynamics detection in time series data: A review. *Expert Systems with Applications*. 2024;248(2):123342. <https://doi.org/10.1016/j.eswa.2024.123342>
10. Grohmann J., Eismann S., Bauer A., et al. SARDE: A Framework for Continuous and Self-Adaptive Resource Demand Estimation. *ACM Transactions on Autonomous and Adaptive Systems*. 2021;15(2):6. <https://doi.org/10.1145/3463369>
11. Tzanettis I., Androna Ch.-M., Zafeiropoulos A., Fotopoulou E., Papavassiliou S. Data Fusion of Observability Signals for Assisting Orchestration of Distributed Applications. *Sensors*. 2022;22(5):2061. <https://doi.org/10.3390/s22052061>
12. Li B., Peng X., Xiang Q., et al. Enjoy your observability: An industrial survey of microservice tracing and analysis. *Empirical Software Engineering*. 2022;27(1):25. <https://doi.org/10.1007/s10664-021-10063-9>
13. Воронцов Ю.А. Сокращение размерности множества метрик мониторинга для оценки состояния вычислительных систем. *Научно-технический вестник Поволжья*. 2025;(5):218–223.
Vorontsov Yu.A. Dimensional reduction of the set of monitoring metrics for assessing the state of computing systems. *Scientific and Technical Volga region Bulletin*. 2025;(5):218–223. (In Russ.).
14. Usman M., Ferlin S., Brunstrom A., Taheri J. A Survey on Observability of Distributed Edge & Container-Based Microservices. *IEEE Access*. 2022;10:86904–86919. <https://doi.org/10.1109/ACCESS.2022.3193102>

15. Cai B., Wang B., Yang M., Guo Q. AutoMan: Resource-efficient provisioning with tail latency guarantees for microservices. *Future Generation Computer Systems*. 2023;143(3):61–75. <https://doi.org/10.1016/j.future.2023.01.014>
16. Cabrieto J., Meers K., Schat E., et al. kcpRS: An R Package for Performing Kernel Change Point Detection on the Running Statistics of Multivariate Time Series. *Behavior Research Methods*. 2022;54(3):1092–1113. <https://doi.org/10.3758/s13428-021-01603-8>
17. Li Zh., Cao J. Automatic search intervals for the smoothing parameter in penalized splines. *Statistics and Computing*. 2023;33(1):1. <https://doi.org/10.1007/s11222-022-10178-z>
18. Филаретов Г.Ф., Бучаала З. Разработка алгоритмов обнаружения разладки временных рядов на основе непараметрических критериев проверки гипотез. *Вестник МЭИ*. 2021;(3):67–77. <https://doi.org/10.24160/1993-6982-2021-3-67-77>
Filaretov G.F., Bouchaala Z. Development of Time Series Disorder Detection Algorithms Based on Nonparametric Hypothesis Testing Criteria. *Bulletin of MPEI*. 2021;(3):67–77. (In Russ.). <https://doi.org/10.24160/1993-6982-2021-3-67-77>
19. Aue A., Kirch C. The state of cumulative sum sequential changepoint testing 70 years after Page. *Biometrika*. 2024;111(2):367–391. <https://doi.org/10.1093/biomet/asad079>
20. Sholihat S.S., Indratno S.W., Mukhaiyar U. The role of parameters in Bayesian Online ChangePoint Detection: detecting early warning of mount Merapi eruptions. *Heliyon*. 2021;7(7):e07482. <https://doi.org/10.1016/j.heliyon.2021.e07482>
21. Loughman M., Barton S., Farrell R., Dooley J. Early Stopping Criterion for Recursive Least Squares Training of Behavioural Models. *Wireless Personal Communications*. 2022;126(2):1633–1648. <https://doi.org/10.1007/s11277-022-09813-9>
22. Zhu K., Yu Ch., Wan Y. Recursive Least Squares Identification With Variable-Direction Forgetting via Oblique Projection Decomposition. *IEEE/CAA Journal of Automatica Sinica*. 2022;9(3):547–555. <https://doi.org/10.1109/JAS.2021.1004362>

ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

Воронцов Юрий Алексеевич, старший преподаватель, МИРЭА – Российский технологический университет, Москва, Российская Федерация.
e-mail: voroncov_yu@mirea.ru
ORCID: [0000-0002-1732-6002](https://orcid.org/0000-0002-1732-6002)

Yuriy A. Vorontsov, Senior Lecturer, MIREA – Russian Technological University, Moscow, the Russian Federation.

Калач Андрей Владимирович, заслуженный работник высшей школы Российской Федерации, доктор химических наук, профессор, профессор кафедры безопасности информации и защиты сведений, составляющих государственную тайну, Воронежский институт ФСИН России, Воронеж; профессор кафедры прикладной математики, МИРЭА – Российский технологический университет, Москва, Российская Федерация.
e-mail: a_kalach@mail.ru
ORCID: [0000-0002-8926-3151](https://orcid.org/0000-0002-8926-3151)

Andrew V. Kalach, Honored Worker of Higher Education of the Russian Federation, Doctor of Chemical Sciences, Professor, Professor at the Department of Information Security and Protection of Information Constituting a State Secret, Voronezh Institute of the Federal Penitentiary Service of Russia, Voronezh; Professor at the Department of Applied Mathematics, MIREA – Russian Technological University, Moscow, the Russian Federation.

Золотарев Юрий Николаевич, доктор технических наук, доцент, профессор кафедры безопасности информации и защиты сведений, составляющих государственную тайну, Воронежский институт ФСИН России, Воронеж, Российская Федерация.

e-mail: vifsin@36.fsin.gov.ru

ORCID: [0000-0003-0582-3009](https://orcid.org/0000-0003-0582-3009)

Yuri N. Zolotaryov, Doctor of Engineering Sciences, Docent, Professor at the Department of Information Security and Protection of Information Constituting a State Secret, Voronezh Institute of the Federal Penitentiary Service of Russia, Voronezh, the Russian Federation.

Статья поступила в редакцию 31.08.2026; одобрена после рецензирования 21.10.2026; принята к публикации 29.10.2026.

The article was submitted 31.08.2026; approved after reviewing 21.10.2026; accepted for publication 29.10.2026.