Гасанян Д.Ф. Автоматическое гибридное масштабирование микросервисов в KUBERNETES с учетом качества обслуживания на основе прогнозирования нагрузки и обучения с подкреплением// Международный журнал информационных технологий и энергоэффективности. – 2026. – Т. 11 № 6 (68) Ч.1 с. 17–29

Международный журнал информационных технологий и энергоэффективности

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/



УДК 004.75:004.85

АВТОМАТИЧЕСКОЕ ГИБРИДНОЕ МАСШТАБИРОВАНИЕ МИКРОСЕРВИСОВ В KUBERNETES С УЧЕТОМ КАЧЕСТВА ОБСЛУЖИВАНИЯ НА ОСНОВЕ ПРОГНОЗИРОВАНИЯ НАГРУЗКИ И ОБУЧЕНИЯ С ПОДКРЕПЛЕНИЕМ

Гасанян Д.Ф.
ФГБОУ ВО «МИРЭА - РОССИЙСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ», Москва, Россия (119454, г. Москва, пр-т Вернадского, д. 78, стр. 4), e-mail: diwork1721@gmail.com

Предложен метод автоматического гибридного масштабирования микросервисов в Kubernetes, ориентированный на качество обслуживания, а не только на загрузку процессора. Научный вклад состоит в совместной постановке трех компонентов: краткосрочного прогноза нагрузки с оценкой верхнего квантиля, прогнозного риска нарушения задержки 95-го процентиля (p95) и агента обучения с подкреплением, который выбирает изменение числа подов и ресурсного профиля пода. В качестве прогнозной модели используется линейное экспоненциальное сглаживание Хольта с квантильной поправкой по остаткам. Эта модель выбрана из-за низкой вычислительной стоимости, устойчивости на коротком горизонте и возможности явно оценить риск перегрузки. Для выбора действий применено табличное Q-обучение с маской безопасных действий. Маска запрещает шаги, нарушающие границы реплик и ресурсов, а при прогнозируемом превышении порога оставляет только действия, увеличивающие емкость узкого микросервиса. Метод проверен в воспроизводимой имитационной среде из трех связанных микросервисов. На 24 тестовых трассах предложенный контроллер снизил долю интервалов с нарушением целевого уровня обслуживания с 3,85% у реактивного горизонтального автомасштабировщика до 2,67%. По сравнению с вариантом без прогнозного риска доля нарушений уменьшилась с 7,53% до 2,67%, что показывает роль прогнозной части состояния.

Ключевые слова: Kubernetes, микросервисы, гибридное масштабирование, качество обслуживания, прогноз нагрузки, Q-обучение, горизонтальное масштабирование, ресурсный профиль.

AUTOMATIC HYBRID SCALE-UP OF MICROSERVICES IN KUBERNETS WITH QUALITY OF SERVICE BASED ON LOAD FORECASTING AND REINFORCEMENT LEARNING

Gasanyan D.F.
MIREA - RUSSIAN TECHNOLOGICAL UNIVERSITY, Moscow, Russia (119454, Moscow, avenue Vernadsky, 78, b. 4), e-mail: diwork1721@gmail.com

The paper proposes an automatic hybrid autoscaling method for Kubernetes microservices that targets service quality rather than processor utilization alone. The contribution combines three elements: short-term workload forecasting with an upper-quantile estimate, predictive risk of 95th-percentile latency violation, and a reinforcement learning (RL) agent that selects replica and per-pod resource-profile changes. The forecasting component uses Holt linear exponential smoothing with a residual-based quantile correction. This model is selected because it is inexpensive at inference time, stable on a short control horizon, and suitable for explicit overload-risk estimation. The decision component uses tabular Q-learning with a safe action mask. The mask blocks actions outside replica and resource bounds and, under predicted risk, leaves only actions that increase the capacity of the bottleneck microservice. The method is evaluated in a reproducible simulation of three interacting microservices. On 24 unseen workload traces, the proposed controller reduces service-level violation intervals from 3.85% under reactive HPA to 2.67%. Compared with the same learned policy without predictive risk, violations decrease from 7.53% to 2.67%, indicating the role of forecasting in the controller state.

Keywords: Kubernetes, microservices, hybrid autoscaling, quality of service, workload forecasting, Q-learning, horizontal scaling, resource profile.

Введение

Микросервисное приложение в Kubernetes редко перегружается как один неделимый процесс. Пользовательский запрос проходит через входной шлюз, сервис каталога, сервис заказа, базу данных и внешние зависимости. Поэтому рост задержки может возникнуть не в том поде, где выше всего загрузка процессора. Стандартный горизонтальный автомасштабировщик подов Kubernetes, далее HPA, масштабирует выбранный объект по отношению текущей метрики к целевой [1]. Это простая и надежная основа, но она остается реактивной.

Вертикальное масштабирование дополняет горизонтальное, поскольку меняет запросы и ограничения ресурсов для уже существующих подов [2]. Однако в Kubernetes изменение ресурсного профиля может требовать пересоздания пода или отдельной поддержки изменения ресурсов без перезапуска. Для микросервисов с жестким ограничением задержки такая операция должна выполняться через слой безопасности, иначе выигрыш от увеличения ресурса может быть потерян на этапе обновления.

Исследования показывают несколько важных ограничений существующих подходов. Nguyen и соавторы экспериментально описали запаздывание HPA и чувствительность к частоте метрик [3]. Rattihalli и соавторы показали, что вертикальная корректировка ресурсов важна для снижения избыточного выделения [4]. Baresi и соавторы рассмотрели конфликт горизонтального и вертикального контуров управления [5]. Vu, Tran и Kim предложили прогнозное гибридное масштабирование, но их схема опирается на отдельный детектор всплесков и регрессионную модель производительности [6]. Работы Mondal и соавторов подтверждают, что качество прогноза зависит от типа нагрузки и окна наблюдения [7].

Обучение с подкреплением применяется в этой области потому, что действие масштабирования имеет отложенную цену. Добавление подов снижает риск задержки, но увеличивает стоимость. Уменьшение ресурсов экономит кластер, но может проявиться нарушением p95 через несколько интервалов. Rossi и соавторы, Abdel Khaleq и Ra, Xu и соавторы, а также Santos и соавторы показывают разные варианты такой постановки [8; 9; 10; 13]. При этом часть работ либо выбирает только метрику для HPA, либо переносит обучение на сложное пространство действий без явного защитного слоя.

В этой статье предлагается прогнозно-рисковый гибридный контроллер. Он не сводится к замене штатного автомасштабировщика обучаемой моделью. Контроллер вводит риск нарушения качества обслуживания как самостоятельный признак состояния, связывает прогноз нагрузки с графом микросервисов, применяет Q-обучение только внутри допустимой области действий и отделяет решение агента от слоя исполнения Kubernetes.

Цель

Цель исследования состоит в разработке архитектуры и алгоритма гибридного автомасштабирования микросервисов в Kubernetes, который заранее учитывает риск нарушения p95-задержки и выбирает между горизонтальным изменением числа подов и изменением ресурсного профиля пода.

Исследовательская часть работы выражена в четырех положениях. Во-первых, предложена формализация прогнозного риска для маршрута микросервисов, где риск зависит

от верхнего квантиля будущей нагрузки, емкости каждого сервиса и целевого порога задержки. Во-вторых, состояние агента обучения с подкреплением включает не только текущие метрики, но и прогнозный риск, направление нагрузки и идентификатор узкого микросервиса. В-третьих, действие агента ограничено маской безопасности, которая отделяет обучение от недопустимых операций Kubernetes. В-четвертых, функция награды объединяет нарушение качества обслуживания, ресурсную стоимость и устойчивость управления.

Гипотеза исследования формулируется так: если агент получает не только текущую загрузку, но и прогнозный риск нарушения p95 на маршруте микросервисов, то он выбирает более устойчивые гибридные действия, чем реактивное масштабирование по текущей метрике.

Материал и методы исследования

Материал исследования включает официальную документацию Kubernetes, статьи о горизонтальном и вертикальном масштабировании, работы о прогнозном масштабировании и публикации по применению обучения с подкреплением для управления микросервисами [1-13]. Для прогнозирования использована модель Хольта как базовый метод краткосрочного прогноза временных рядов с уровнем и трендом [14]. Математическая основа Q-обучения опирается на классическое описание Sutton и Barto, а также на исходную работу Watkins и Dayan [15; 16].

Базовая формула HPA задает желаемое число реплик через отношение текущего значения метрики к целевому. В обозначениях этой статьи она записана следующим образом.

$$N_{i,t+1}^{HPA} = \left\lceil N_{i,t} \frac{m_{i,t}}{m_i^*} \right\rceil \quad (1)$$

где $N_{i,t}$ — текущее число подов микросервиса i;

$m_{\mathrm{i,\,t}}$ — текущее значение метрики управления;

m_i^* — целевое значение метрики.

Формула (1) удобна для эксплуатации, но она не содержит будущей нагрузки, графа зависимостей и стоимости смены ресурсного профиля. Поэтому в предлагаемом методе HPA используется как базовая точка сравнения, а не как единственный контур управления.

Архитектура предлагаемого контроллера показана на рисунке 1. Система получает метрики из входного шлюза, Prometheus и программного интерфейса Kubernetes. В отдельном хранилище задается граф микросервисов: вершины соответствуют сервисам, ребра отражают маршрут запроса и коэффициент передачи нагрузки. Прогнозная модель оценивает нагрузку следующего интервала и верхний квантиль 0,90. Блок риска переводит прогноз в оценку p95-задержки по маршруту. Агент Q-обучения выбирает действие, но слой маски действий проверяет границы реплик, ресурсных профилей и паузу между изменениями.

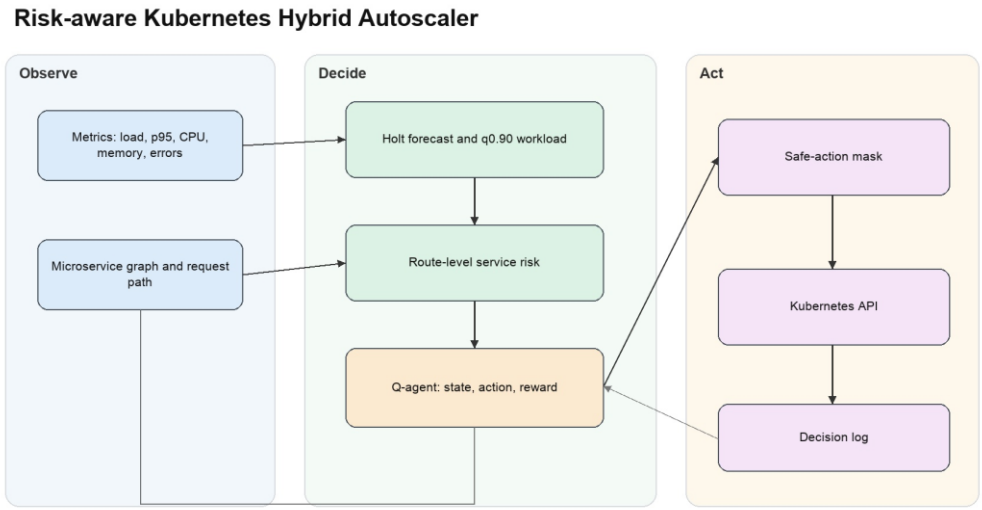


Рисунок 1 — Архитектура прогнозно-рискового гибридного автомасштабировщика

В качестве прогнозной модели используется линейное экспоненциальное сглаживание Хольта. Эта модель не требует большой обучающей выборки, имеет малую стоимость выполнения в контроллере и подходит для короткого горизонта, где соседние значения нагрузки часто информативны. При этом модель сохраняет тренд, что важно перед плавным ростом трафика. Уравнения обновления уровня и тренда приведены в формулах (2) и (3).

$$\ell_t = \alpha\lambda_t + (1-\alpha)(\ell_{t-1} + b_{t-1})$$
$$b_t = \beta(\ell_t - \ell_{t-1}) + (1-\beta)b_{t-1} \quad (3)$$

где λ_t — наблюдаемая входная нагрузка в момент t;

ℓ_t — сглаженный уровень ряда;

b_t — оценка локального тренда;

α, β — коэффициенты сглаживания.

$$\hat{\lambda}_{t+h} = \ell_t + h\, b_t \quad (4)$$
$$\hat{\lambda}^{0.90}_{t+h} = \hat{\lambda}_{t+h} + z_{0.90}\hat{\sigma}_t,\ z_{0.90} = 1.2816 \quad (5)$$

где $\hat{\lambda}_{t+h}$ — точечный прогноз нагрузки на горизонт h;

$\hat{\lambda}^{0.90}_{t+h}$ — прогноз верхнего квантиля;

$z_{0.90}$ — квантиль стандартного нормального распределения для уровня 0.90;

$\hat{\sigma}_t$ — экспоненциально сглаженная ошибка прогноза.

Точечный прогноз и верхний квантиль нагрузки задаются формулами (4) и (5). Верхний квантиль используется не как попытка угадать среднюю нагрузку, а как инженерная оценка риска. Контроллер должен готовиться к правому хвосту нагрузки, потому что именно он чаще приводит к нарушению p95.

Для микросервиса i емкость задается числом подов и ресурсным профилем пода. Если $r_{i,t}$ обозначает выделенный ресурс на под, а $\mu_i(r_{i,t})$ означает обслуживающую способность одного пода, то прогнозная утилизация определяется формулой (6).

$$\rho_{i,t+h} = \frac{\omega_i\, \hat{\lambda}^{0.90}_{t+h}}{(N_{i,t}\, \mu_i(r_{i,t}) + \varepsilon)} \quad (6)$$

где $\rho_{i,t+h}$ — прогнозная утилизация микросервиса i;

ω_i — коэффициент передачи входной нагрузки в микросервис i;

ε — малое число для численной устойчивости.

$$\hat{L}^{95}_{i,t+h} = B_i + \frac{K_i \rho^2_{i,t+h}}{1.03 - \rho_{i,t+h}} + P_i\left[\rho_{i,t+h} - 1\right]_+ \quad (7)$$

где $\hat{L}^{95}_{i,t+h}$ — прогноз p95-задержки микросервиса;

B_i, K_i, P_i — параметры профиля микросервиса;

$[x]_+ = \max(x, 0)$ — положительная часть величины.

Формула (7) не претендует на универсальную модель очереди. Она нужна как воспроизводимая среда для обучения и проверки алгоритма: при приближении утилизации к насыщению задержка растет нелинейно, а при перегрузке добавляется отдельный штраф.

$$g_t = max_{\pi \in \Pi}\left(\frac{\sum_{i \in \pi} \hat{L}^{95}_{i,t+1}}{\tau_\pi}\right)$$

где g_t — прогнозный риск нарушения качества обслуживания;

Π — множество пользовательских маршрутов в графе микросервисов;

τ_π — целевой порог p95-задержки для маршрута π.

Итоговый риск маршрута определяется формулой (8). Состояние агента обучения с подкреплением строится из дискретизированных признаков (9). Дискретизация выбрана намеренно: она уменьшает размерность задачи и делает поведение политики проверяемым. В данной работе применяется табличное Q-обучение, а не глубокая нейронная модель. Такой выбор оправдан тем, что исследуется архитектурная постановка и действие имеет конечный набор вариантов.

$$s_t = \left(g_t, \Delta\hat{\lambda}_t,\ i_t^{max}, \frac{L_t^{95}}{\tau}, C_t,\ r_{i,t}^{max}\right) \quad (9)$$

где $\Delta\hat{\lambda}_t$ — направление прогнозируемого изменения нагрузки;

i_t^{max}=arg $max_i \rho_{i,t+1}$ — прогнозируемый узкий микросервис;

$C_t = \sum_i N_{i,t} r_{i,t}$ — текущая ресурсная стоимость.

$$a_t = (i_t, \Delta N_{i,t}, \Delta r_{i,t}),$$
$$\Delta N_{i,t} \in \{-1,0,1,2\}, \qquad \Delta r_{i,t} \in \{-1,0,1\} \quad (10)$$

где a_t — действие агента;

i_t — выбранный микросервис;

$\Delta N_{i,t}$ — изменение числа подов;

$\Delta r_{i,t}$ — переход между ресурсными профилями.

Действие агента задается формулой (10). Перед исполнением действие проходит через маску безопасности. Если прогнозный риск не превышает порог, маска оставляет все допустимые действия в пределах минимального и максимального числа подов. Если риск высок, маска оставляет только действия, которые увеличивают емкость прогнозируемого узкого микросервиса. Ограничения безопасности задаются в формуле (11).

$$A_{safe}(s_t) = \{a \in A: N^{min}{}_i \le N_i + \Delta N_i \le N^{max}{}_i, \quad r^{min}{}_i \le r_i + \Delta r_i \le r^{max}{}_i\} \tag{11}$$

где $A_{safe}(s_t)$ — множество действий, разрешенных слоем безопасности в состоянии s_t;

A — исходное множество действий агента.

Алгоритм одного шага контроллера имеет следующую логику. Сначала собираются метрики текущего интервала и обновляется прогноз нагрузки. Затем вычисляется прогнозная задержка каждого сервиса и риск маршрута. После этого формируется множество безопасных действий, агент выбирает действие с максимальным Q-значением внутри этого множества, а слой исполнения применяет изменение числа подов или ресурсного профиля через программный интерфейс Kubernetes. Журнал решений возвращается в контур наблюдения и используется при следующем шаге.

Функция награды имеет вид (12).

$$R_t = -w_q \left[\frac{L^{95}{}_t}{\tau} - 1\right]_+ - w_c C_t - w_s S_t - w_u U_t + B_t \tag{12}$$

где R_t — награда агента;

(w_q, w_c, w_s, w_u) — неотрицательные веса штрафов;

S_t — цена изменения действия и ресурсного профиля;

U_t — штраф за перегрузку или слишком низкую утилизацию;

B_t — малый бонус за работу без нарушения качества в допустимой зоне утилизации.

Обновление Q-значения выполняется по формуле (13).

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_q \left(R_t + \gamma \max_{a' \in A_{safe}(s_{t+1})} Q(s_{t+1}, a') - Q(s_t, a_t) \right) \tag{13}$$

где $Q(s_t, a_t)$ — оценка полезности действия в состоянии;

α_q — скорость обучения;

a' — возможное действие на следующем шаге;

$A_{safe}(s_{t+1})$ — множество безопасных действий в следующем состоянии;

γ — коэффициент учета будущей награды.

Обучение проводилось в имитационной среде. Среда содержит три микросервиса: входной шлюз, каталог и сервис заказа. Для каждого сервиса заданы коэффициент передачи нагрузки, базовая задержка, коэффициент очереди, обслуживающая способность одного ядра процессора, минимальное и максимальное число подов. Трассы нагрузки включали синусоидальные компоненты, тренд, нормальный шум и четыре всплеска разной ширины. Это не замена производственного стенда, а контролируемый способ проверить саму постановку алгоритма до внедрения в реальный кластер.

Таблица 1 - Параметры обучения и проверки

Параметр	Значение
Прогноз нагрузки	линейное экспоненциальное сглаживание Хольта, горизонт 1 минута, квантиль 0,90 по остаткам
Алгоритм обучения	табличное Q-обучение с маской безопасных действий
Обучение	1200 эпизодов по 360 минут
Тестовая проверка	24 новые трассы по 500 минут
Микросервисы	шлюз, каталог, заказ
Ресурсные профили	0,50, 0,75 и 1,00 ядра процессора на под
Целевой уровень	p95-задержка маршрута не выше 380 мс
Сравниваемые контроллеры	реактивный HPA, прогнозный HPA, предложенный контроллер, абляция без прогнозного риска

Результаты исследования и обсуждение

Рисунок 2 показывает динамику обучения агента. В начале обучения политика часто выбирает действия, которые либо не успевают убрать риск, либо создают лишнюю стоимость. По мере накопления Q-значений средняя награда растет, а доля нарушений снижается. Важный момент состоит не в самой форме кривой, а в том, что обучение происходит в ограниченном пространстве безопасных действий. Агент не получает возможности обучаться на заведомо недопустимых операциях Kubernetes.



Рисунок 2 — Динамика обучения агента Q-обучения

Итоговые значения по 24 тестовым трассам представлены в таблице 2. Реактивный HPA имел 3,85% интервалов с нарушением целевого уровня обслуживания. Прогнозный HPA снизил этот показатель до 2,82%, поскольку использовал верхний квантиль нагрузки при расчете числа подов. Предложенный контроллер дал 2,67% нарушений. Его преимущество над прогнозным HPA по доле нарушений невелико, но контроллер менял не только число подов, но и ресурсный профиль узкого микросервиса.

Таблица 2 - Сводные результаты на 24 тестовых трассах

Контроллер	**Нарушения, %**	**p95, мс**	**Средние ядра**	**Действия**
Реактивный HPA	3,85	366,7	17,61	633,6
Прогнозный HPA	2,82	348,9	18,87	279,5
Предложенный контроллер	2,67	354,4	19,92	473,5
Без прогнозного риска	7,53	411,2	19,29	423,9

Абляция без прогнозного риска важнее прямого сравнения с HPA. При той же структуре действия и той же обученной политике исключение прогнозного риска из состояния увеличило долю нарушений до 7,53% и p95 до 411,2 мс. Это означает, что прогнозный признак не является декоративным входом модели. Он меняет класс состояний, в которых агент выбирает защитное действие заранее.

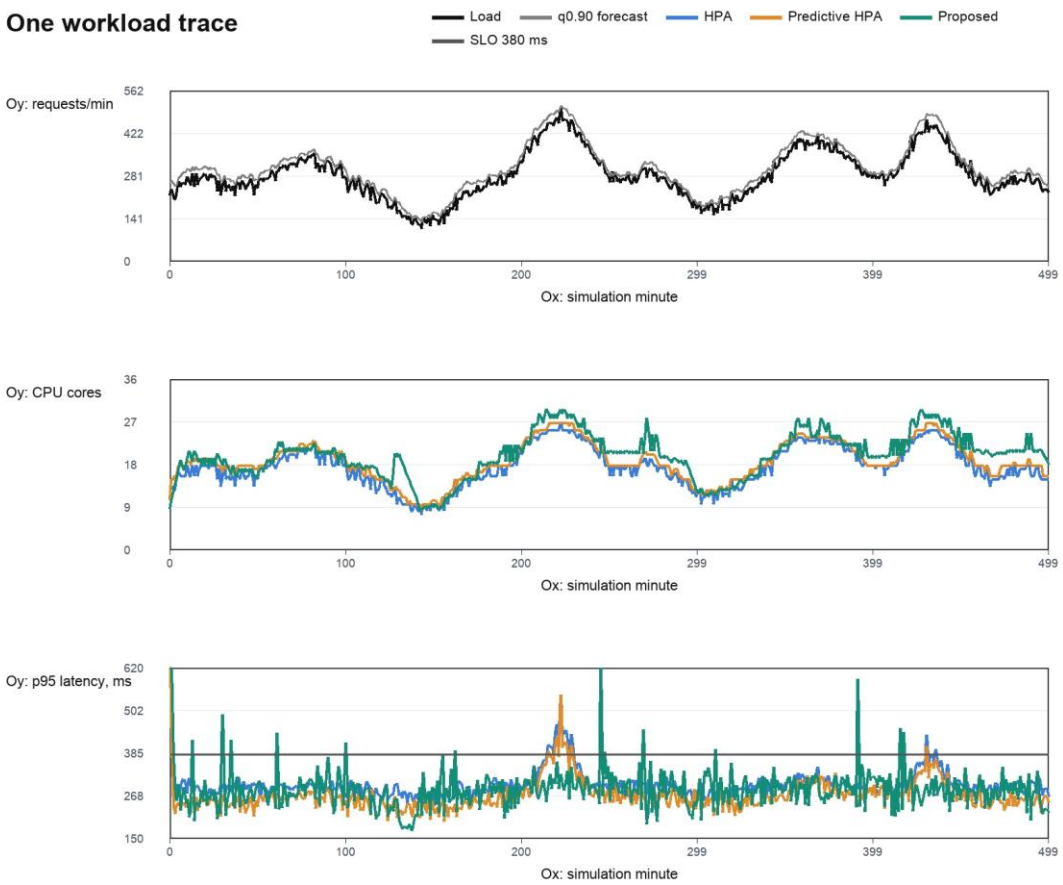


Рисунок 3 — Поведение контроллеров на одной тестовой трассе

На одиночной трассе видно различие контуров управления. Реактивный HPA часто совершает много малых действий, потому что реагирует на текущую утилизацию. Прогнозный HPA действует спокойнее, но меняет только число подов. Предложенный контроллер добавляет ресурсный профиль как вторую степень свободы: при риске он усиливает узкий сервис, а после ухода пика может снизить избыточную емкость.

Ресурсная цена предложенного контроллера выше, чем у реактивного HPA: 19,92 среднего ядра против 17,61. Это ожидаемая цена ориентации на p95. При этом число действий ниже, чем у HPA: 473,5 против 633,6. По сравнению с прогнозным HPA предложенный контроллер делает больше действий, потому что управляет двумя осями масштабирования. Практический выбор между этими вариантами зависит от того, насколько оператору важнее снизить долю нарушений или уменьшить число операций изменения состояния.

Полученные результаты согласуются с литературой. Vu, Tran и Kim показывают пользу прогнозного гибридного масштабирования для контейнерных приложений [6]. Gwydion демонстрирует, что агент обучения с подкреплением должен видеть состояние нескольких развертываний, а не отдельный сервис [13]. Предложенный метод объединяет эти идеи через прогнозный риск маршрута и маску безопасных действий. В отличие от подходов, где агент выбирает только метрику для HPA [9], здесь действие меняет емкость микросервиса напрямую.

Ограничения исследования

Ограничения существенны. Проверка выполнена в имитационной среде, а не в производственном Kubernetes-кластере. Модель задержки не учитывает сетевые повторные передачи, дисковую подсистему, базу данных и холодный запуск конкретных контейнерных образов. Прогноз Хольта выбран из-за низкой стоимости и воспроизводимости, но для нагрузок с сильными календарными режимами может потребоваться сезонная модель или модель с внешними признаками. Табличное Q-обучение удобно для анализа, но при десятках микросервисов пространство состояний станет слишком большим. В этом случае потребуется аппроксимация функции Q или разбиение политики по подсистемам приложения.

Также нельзя переносить численные проценты из этой работы на любой кластер. Они относятся к заданной имитационной среде и служат проверкой алгоритмической постановки. Для промышленного вывода нужны эксперименты в реальном кластере, несколько типов приложений, трассы разной природы и анализ чувствительности к порогу p95, ширине квантиля прогноза и стоимости изменения ресурсного профиля.

Выводы

1. Предложена архитектура прогнозно-рискового гибридного автомасштабировщика для Kubernetes, где прогноз нагрузки, оценка риска p95 и обучаемая политика выбора действия работают как единый контур.
2. В качестве прогнозной модели выбран метод Хольта с квантильной поправкой по остаткам. Он достаточно прост для встраивания в контроллер и дает верхнюю оценку нагрузки, пригодную для расчета риска.
3. В качестве алгоритма обучения с подкреплением использовано табличное Q-обучение с маской безопасных действий. Маска отделяет обучение от недопустимых операций и заставляет контроллер увеличивать емкость узкого сервиса при высоком прогнозном риске.
4. На 24 тестовых трассах предложенный контроллер снизил долю нарушений целевого уровня обслуживания с 3,85% у реактивного HPA до 2,67%. Абляция без прогнозного риска дала 7,53%, что подтверждает роль прогнозного состояния.
5. Улучшение качества обслуживания достигается ценой большего среднего выделения процессора. Поэтому метод целесообразен для сервисов, где p95-задержка важнее минимальной ресурсной стоимости.

Заключение

В ходе исследования была предложена архитектура автоматического гибридного масштабирования микросервисов в Kubernetes, ориентированная на соблюдение качества обслуживания при изменяющейся нагрузке. В отличие от реактивного масштабирования, предложенный подход использует прогноз нагрузки и оценку риска нарушения задержки p95, а затем передает это состояние агенту обучения с подкреплением. Агент выбирает не только изменение числа подов, но и изменение ресурсного профиля узкого микросервиса, что позволяет рассматривать горизонтальное и вертикальное масштабирование как единый контур управления.

Результаты имитационного эксперимента показали, что учет прогнозного риска снижает долю нарушений целевого уровня обслуживания по сравнению с реактивным и абляционным

вариантами. При этом улучшение качества обслуживания достигается ценой большего среднего выделения ресурсов, поэтому практическое применение метода требует настройки весов функции награды под требования конкретной системы. Дальнейшее развитие работы целесообразно связать с проверкой метода в реальном Kubernetes-кластере, расширением числа микросервисов и переходом от табличного Q-обучения к методам аппроксимации функции полезности.

Список литературы

1. Кубернетес. Горизонтальное автомасштабирование модулей. URL: https://kubernetes.io/docs/concepts/workloads/autoscaling/horizontal-pod-autoscale/ (дата обращения: 01.05.2026).
2. Кубернетес. Вертикальное автомасштабирование модулей. URL: https://kubernetes.io/docs/concepts/workloads/autoscaling/vertical-pod-autoscale/ (дата обращения: 01.05.2026).
3. Нгуен Т.-Т., Йем Й.-Дж., Ким Т., Парк Д.-Х., Ким С. Горизонтальное автомасштабирование подов в Kubernetes для эластичной оркестрации контейнеров // Датчики. 2020. Том. 20, № 16. Статья 4621. DOI: 10.3390/s20164621.
4. Rattihalli G., Govindaraju M., Lu H., Tiwari D. Изучение потенциала неразрушающего вертикального автоматического масштабирования и оценки ресурсов в Kubernetes // 2019 IEEE 12-я Международная конференция по облачным вычислениям. 2019. С. 33-40. DOI: 10.1109/CLOUD.2019.00018.
5. Baresi L., Hu D. Y. X., Quattrocchi G., Terracciano L. KOSMOS: Вертикальное и горизонтальное автоматическое масштабирование ресурсов для Kubernetes // Сервисно-ориентированные вычисления. Lecture Notes in Computer Science. 2021. С. 821-829. DOI: 10.1007/978-3-030-91431-8_59.
6. Vu D.-D., Tran M.-N., Kim Y. Predictive Hybrid Autoscaling for Containerized Applications // IEEE Access. 2022. Vol. 10. P. 109768-109778. DOI: 10.1109/ACCESS.2022.3214985.
7. Mondal S. K., Wu X., Kabir H. M. D., Dai H.-N., Ni K., Yuan H., Wang T. Toward Optimal Load Prediction and Customizable Autoscaling Scheme for Kubernetes // Mathematics. 2023. Vol. 11, № 12. Статья 2675. DOI: 10.3390/math11122675.
8. Rossi F., Nardelli M., Cardellini V. Горизонтальное и вертикальное масштабирование приложений на основе контейнеров с использованием обучения с подкреплением // 2019 IEEE 12-я Международная конференция по облачным вычислениям. 2019. С. 329-338. DOI: 10.1109/CLOUD.2019.00061.
9. Abdel Khaleq A., Ra I. Разработка агентов с учетом качества обслуживания с использованием обучения с подкреплением для автоматического масштабирования микросервисов в облаке // 2021 IEEE Международная конференция по автономным вычислениям и самоорганизующимся системам. 2021. DOI: 10.1109/ACSOS-C52956.2021.00025.
10. Xu M., Song C., Ilager S., Gill S. S., Ye K., Xu C.-Z. CoScal: Многогранное масштабирование микросервисов с помощью обучения с подкреплением // IEEE Transactions on Network and Service Management. 2022. DOI: 10.1109/TNSM.2022.3210211.

11. Hossen M. R., Islam M. A., Ahmed K. Практическое эффективное автомасштабирование микросервисов с обеспечением качества обслуживания // Труды 31-го Международного симпозиума по высокопроизводительным параллельным и распределенным вычислениям. 2022. С. 240-252. DOI: 10.1145/3502181.3531460.
12. Spatharakis D., Dimolitsas I., Vlahakis E. E., Dechouniotis D., Athanasopoulos N., Papavassiliou S. Распределенное автомасштабирование ресурсов в кластерах Kubernetes Edge // 18-я Международная конференция по управлению сетями и сервисами. 2022. С. 163-169. DOI: 10.23919/CNSM55787.2022.9965056.
13. Santos J., Reppas E., Wauters T., Volckaert B., De Turck F. Gwydion: Эффективное автомасштабирование сложных контейнеризированных приложений в Kubernetes с помощью обучения с подкреплением // Журнал сетевых и компьютерных приложений. 2025. Том. 234. Статья 104067. DOI: 10.1016/j.jnca.2024.104067.
14. Гайндман Р. Дж., Атанасопулос Г. Прогнозирование: принципы и практика. 3-е изд. Мельбурн: OTexts, 2021. URL: https://otexts.com/fpp3/ (дата обращения: 06.05.2026).
15. Саттон Р.С., Барто А.Г. Обучение с подкреплением: Введение. 2-е изд. Кембридж: MIT Press, 2018.
16. Уоткинс С.Дж.Ч., Даян П. Q-обучение // Машинное обучение. 1992. Том. 8. С. 279-292. DOI: 10.1007/BF00992698.

References

1. Kubernetes. Horizontal Pod Autoscaling. URL: https://kubernetes.io/docs/concepts/workloads/autoscaling/horizontal-pod-autoscale/ (дата обращения: 01.05.2026).
2. Kubernetes. Vertical Pod Autoscaling. URL: https://kubernetes.io/docs/concepts/workloads/autoscaling/vertical-pod-autoscale/ (дата обращения: 01.05.2026).
3. Nguyen T.-T., Yeom Y.-J., Kim T., Park D.-H., Kim S. Horizontal Pod Autoscaling in Kubernetes for Elastic Container Orchestration // Sensors. 2020. Vol. 20, No. 16. Article 4621. DOI: 10.3390/s20164621.
4. Rattihalli G., Govindaraju M., Lu H., Tiwari D. Exploring Potential for Non-Disruptive Vertical Auto Scaling and Resource Estimation in Kubernetes // 2019 IEEE 12th International Conference on Cloud Computing. 2019. pp. 33-40. DOI: 10.1109/CLOUD.2019.00018.
5. Baresi L., Hu D. Y. X., Quattrocchi G., Terracciano L. KOSMOS: Vertical and Horizontal Resource Autoscaling for Kubernetes // Service-Oriented Computing. Lecture Notes in Computer Science. 2021. pp. 821-829. DOI: 10.1007/978-3-030-91431-8_59.
6. Vu D.-D., Tran M.-N., Kim Y. Predictive Hybrid Autoscaling for Containerized Applications // IEEE Access. 2022. Vol. 10. P. 109768-109778. DOI: 10.1109/ACCESS.2022.3214985.
7. Mondal S. K., Wu X., Kabir H. M. D., Dai H.-N., Ni K., Yuan H., Wang T. Toward Optimal Load Prediction and Customizable Autoscaling Scheme for Kubernetes // Mathematics. 2023. Vol. 11, No. 12. Article 2675. DOI: 10.3390/math11122675.
8. Rossi F., Nardelli M., Cardellini V. Horizontal and Vertical Scaling of Container-Based Applications Using Reinforcement Learning // 2019 IEEE 12th International Conference on Cloud Computing. 2019. P. 329-338. DOI: 10.1109/CLOUD.2019.00061.

9. Abdel Khaleq A., Ra I. Development of QoS-aware agents with reinforcement learning for autoscaling of microservices on the cloud // 2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion. 2021. DOI: 10.1109/ACSOS-C52956.2021.00025.
10. Xu M., Song C., Ilager S., Gill S. S., Ye K., Xu C.-Z. CoScal: Multifaceted Scaling of Microservices With Reinforcement Learning // IEEE Transactions on Network and Service Management. 2022. DOI: 10.1109/TNSM.2022.3210211.
11. Hossen M. R., Islam M. A., Ahmed K. Practical Efficient Microservice Autoscaling with QoS Assurance // Proceedings of the 31st International Symposium on High-Performance Parallel and Distributed Computing. 2022. P. 240-252. DOI: 10.1145/3502181.3531460.
12. Spatharakis D., Dimolitsas I., Vlahakis E. E., Dechouniotis D., Athanasopoulos N., Papavassiliou S. Distributed Resource Autoscaling in Kubernetes Edge Clusters // 2022 18th International Conference on Network and Service Management. 2022. pp. 163-169. DOI: 10.23919/CNSM55787.2022.9965056.
13. Santos J., Reppas E., Wauters T., Volckaert B., De Turck F. Gwydion: Efficient auto-scaling for complex containerized applications in Kubernetes through Reinforcement Learning // Journal of Network and Computer Applications. 2025. Vol. 234. Article 104067. DOI: 10.1016/j.jnca.2024.104067.
14. Hyndman R. J., Athanasopoulos G. Forecasting: Principles and Practice. 3rd ed. Melbourne: OTexts, 2021. URL: https://otexts.com/fpp3/ (дата обращения: 06.05.2026).
15. Sutton R. S., Barto A. G. Reinforcement Learning: An Introduction. 2nd ed. Cambridge: MIT Press, 2018.
16. Watkins C. J. C. H., Dayan P. Q-learning // Machine Learning. 1992. Vol. 8. pp. 279-292. DOI: 10.1007/BF00992698.