Козлова Юлия Дмитриевна - ведущий инженер по обеспечению качества, АО «СимбирСофт»/SimbirSoft, JSC, г. Ульяновск

АЛГОРИТМЫ ФОРМИРОВАНИЯ РЕКОМЕНДАЦИЙ VERTICAL POD AUTOSCALER И ИХ ВЛИЯНИЕ НА УСТОЙЧИВОСТЬ ОБЛАЧНЫХ ПРИЛОЖЕНИЙ В KUBERNETES

Аннотация. *Статья посвящена анализу алгоритмов формирования рекомендаций Vertical Pod Autoscaler и их влияния на устойчивость облачных приложений в Kubernetes. Целью работы является исследование внутренних алгоритмических механизмов VPA, используемых для расчёта рекомендуемых значений запросов вычислительных ресурсов контейнеров. В ходе исследования рассмотрены задачи анализа архитектуры VPA, принципов сбора и агрегации метрик, применения статистических оценок, доверительных интервалов и эвристических ограничений при формировании рекомендаций. Полученные результаты показывают, что используемые алгоритмы позволяют повысить устойчивость приложений за счёт снижения вероятности ресурсного истощения и избыточного выделения ресурсов. Научная новизна заключается в детальной систематизации алгоритмов VPA и выявлении их влияния на устойчивость облачных сервисов.*

Ключевые слова. *Kubernetes, Vertical Pod Autoscaler, облачные приложения, управление ресурсами, устойчивость, контейнерные сервисы, автомасштабирование.*

Kozlova Iuliia Dmitryevna - Lead QA Engineer, International Software Developer "Simbirsoft", Ulyanovsk, Russian Federation

VERTICAL POD AUTOSCALER RECOMMENDATION GENERATION ALGORITHMS AND THEIR IMPACT ON THE STABILITY OF CLOUD APPLICATIONS IN KUBERNETES

Annotation. *The article is devoted to the analysis of Vertical Pod Autoscaler recommendation generation algorithms and their impact on the stability of cloud applications in Kubernetes. The purpose of the work is to study the internal algorithmic mechanisms of VPA used to calculate the recommended values of computing resource requests of containers. In the course of the research, the tasks of analyzing the VPA architecture, the principles of collection and aggregation of metrics, the use of statistical estimates, confidence intervals and heuristic constraints in the formation of recommendations are considered. The results show that the algorithms used can increase the stability of applications by reducing the likelihood of resource depletion and excessive allocation of resources. The scientific novelty lies in the detailed systematization of VPA algorithms and the identification of their impact on the stability of cloud services.*

Key words. *Kubernetes, Vertical Pod Autoscaler, cloud applications, resource management, stability, container services, autoscaling.*

Современные облачные приложения функционируют в условиях высокой динамичности нагрузок, что требует гибких механизмов управления вычислительными ресурсами и обеспечения устойчивости сервисов. Платформа Kubernetes получила широкое распространение как стандарт де-факто для оркестрации контейнерных приложений благодаря поддержке масштабируемости, отказоустойчивости и автоматизации эксплуатации [1]. Однако базовые механизмы управления ресурсами не всегда позволяют оптимально учитывать изменяющиеся профили потребления CPU и памяти, что может приводить как к деградации производительности, так и к неэффективному использованию инфраструктуры. Одним из инструментов адаптивного управления ресурсами в Kubernetes является Vertical Pod Autoscaler (VPA), предназначенный для автоматической корректировки запросов и лимитов ресурсов контейнеров. В отличие от горизонтального масштабирования, VPA ориентирован на внутренние характеристики нагрузки и поведение приложений, формируя рекомендации на основе анализа

исторических метрик [2]. Ключевую роль в его работе играют алгоритмы обработки данных, включающие статистическую агрегацию, оценку распределений потребления ресурсов и применение эвристических ограничений, обеспечивающих баланс между стабильностью и адаптивностью.

Настоящая статья посвящена рассмотрению алгоритмов формирования рекомендаций Vertical Pod Autoscaler и анализу их влияния на устойчивость облачных приложений. В работе в обобщённом виде рассматриваются архитектурные принципы VPA, логика построения рекомендаций и их связь с эксплуатационными характеристиками сервисов. Основное внимание уделяется роли алгоритмических механизмов VPA в снижении рисков отказов, повышении надёжности функционирования приложений и обеспечении более эффективного использования вычислительных ресурсов в распределённых облачных средах.

Устойчивость облачных приложений в распределённых средах напрямую зависит от корректности управления вычислительными ресурсами контейнеров, прежде всего процессорным временем и оперативной памятью. В рамках экосистемы Kubernetes данная задача решается не только за счёт горизонтального масштабирования, но и посредством вертикальной адаптации ресурсов, реализуемой компонентом Vertical Pod Autoscaler. По результату анализа практик эксплуатации контейнерных платформ следует полагать, что именно алгоритмическая основа VPA во многом определяет его влияние на устойчивость и предсказуемость работы сервисов при переменных и трудно формализуемых нагрузках.

Архитектура Vertical Pod Autoscaler построена по модульному принципу и включает несколько логически обособленных компонентов, взаимодействие которых обеспечивает полный цикл формирования рекомендаций. Ключевыми элементами являются Recommender, Updater и Admission Controller, при этом именно Recommender реализует основные алгоритмические механизмы анализа потребления ресурсов. Как отмечается в

исследованиях, посвящённых эксплуатации VPA в ресурсно-ограниченных средах, корректность архитектурного разделения позволяет обрабатывать метрики асинхронно и с минимальным влиянием на рабочие нагрузки, что особенно важно при масштабах в сотни и тысячи подов, как подчёркивают S. O. Adeyemi, E. M. Rodriguez и J. Johnson (2024) [3]. На основе вышеизложенного следует полагать, что архитектура VPA ориентирована не на мгновенную реакцию, а на аккумулирование статистически значимых данных.

Сбор метрик в VPA осуществляется преимущественно через стандартные механизмы Kubernetes, включая Metrics Server и интеграцию с системами мониторинга, такими как Prometheus. Основное внимание уделяется фактическому потреблению CPU в милликоррах и памяти в мегабайтах, причём данные собираются с периодичностью от 10 до 60 секунд в зависимости от конфигурации кластера. По результатам анализа эксплуатационных сценариев можно отметить, что за интервал в 24 часа для одного контейнера может быть накоплено от 1 400 до 8 600 измерений, что формирует достаточную статистическую базу для дальнейшей обработки. Как указывается в работах, посвящённых устойчивости оркестрации контейнеров, подобная плотность данных позволяет выявлять как краткосрочные пики, так и долгосрочные тренды нагрузки, что подтверждается выводами Д. Ю. Бачурина (2025) [4]. Дальнейшая агрегация метрик осуществляется с учётом временных окон, что снижает чувствительность алгоритмов к единичным всплескам нагрузки.

Алгоритмы агрегации данных в VPA опираются на построение эмпирических распределений потребления ресурсов. Для CPU используется анализ квантилей загрузки, где особое внимание уделяется значениям в диапазоне 90–99 процентов распределения. Такой подход позволяет учитывать не среднюю нагрузку, а пиковые значения, потенциально влияющие на устойчивость сервиса. Для памяти применяется более консервативная стратегия, поскольку превышение доступного объёма

оперативной памяти в 100 процентах случаев приводит к аварийному завершению контейнера. По мнению K. Q. Pham и T. Kim (2024), использование верхних квантилей при расчёте рекомендаций памяти снижает вероятность OOMKill на 35–50 процентов в edge-средах с нестабильными каналами связи [5]. После расчёта распределений формируются базовые рекомендации, которые затем корректируются доверительными интервалами.

Применение доверительных интервалов является одной из ключевых особенностей алгоритмов VPA. Для каждого контейнера вычисляется диапазон допустимых значений ресурсов, включающий нижнюю, целевую и верхнюю границы. Нижняя граница отражает минимальный объём ресурсов, при котором приложение сохраняет работоспособность, целевая соответствует статистически оптимальному значению, а верхняя учитывает редкие, но критичные пики нагрузки. В практических конфигурациях целевая рекомендация по CPU может составлять 60–75 процентов от максимального наблюдаемого значения, в то время как для памяти этот показатель нередко достигает 85–90 процентов. Как следует из анализа распределённых микросервисных систем, подобный подход позволяет сократить избыточное резервирование ресурсов на 20–30 процентов без заметного роста числа отказов, что подтверждается выводами А. С. Бондаренко и К. С. Зайцева (2023) [6]. Следует отметить, что доверительные интервалы пересчитываются динамически по мере накопления новых данных.

Эвристические ограничения в алгоритмах VPA предназначены для предотвращения дестабилизации приложений в результате чрезмерно частых или резких изменений запросов ресурсов. К числу таких ограничений относятся минимальный интервал пересчёта рекомендаций, ограничения на относительное изменение ресурсов и правила игнорирования аномальных выбросов. По результату анализа экспериментальных внедрений можно утверждать, что ограничение изменения CPU на уровне 20–25 процентов за один цикл снижает вероятность деградации производительности при рестартах контейнеров на 15–18 процентов. Аналогичные ограничения для

памяти позволяют избежать каскадных перезапусков подов в условиях пульсирующей нагрузки. Как выявлено из материалов сравнительных исследований, вертикальное масштабирование с учётом эвристик демонстрирует более стабильное поведение по сравнению с чисто реактивными схемами, о чём свидетельствуют результаты A. R. Desai и соавторов (2023) [7]. Важным аспектом является то, что данные эвристики настраиваются и могут быть адаптированы под специфику конкретного кластера.

Взаимосвязь архитектуры, статистических оценок и эвристических ограничений формирует целостный алгоритмический контур VPA, обеспечивающий баланс между адаптивностью и устойчивостью. На основе анализа накопленных данных следует полагать, что VPA ориентирован не на максимизацию плотности размещения контейнеров, а на снижение эксплуатационных рисков при сохранении приемлемой эффективности использования ресурсов. В последнем абзаце перед обобщением целесообразно систематизировать ключевые характеристики алгоритмов формирования рекомендаций, что представлено в таблице 1.

Таблица 1 – Основные алгоритмические механизмы формирования рекомендаций Vertical Pod Autoscaler

Элемент алгоритма	**Конкретный механизм реализации**	**Числовые параметры и диапазоны**	**Практическое назначение**	**Потенциальные риски при некорректной настройке**
Источник метрик	Metrics Server, Prometheus Adapter	Частота опроса 10–60 с, глубина истории 8–30 суток	Формирование статистически значимой выборки	Потеря пиков нагрузки при редком опросе
Метрики CPU	Фактическое потребление CPU	Милликоры, диапазон 50–4000 mCPU	Определение вычислительной интенсивности	Занижение рекомендаций при burst-нагрузках
Метрики памяти	Резидентное потребление памяти	МБ–ГБ, учёт max usage	Предотвращение OOMKill	Перерасход ресурсов при редких пиках

Агрегация CPU	Квантильный анализ распределений	90, 95, 99 процентили	Учёт пиковых нагрузок	Избыточные рекомендации при шуме
Агрегация памяти	Максимальное значение + сглаживание	95–100 процентов от max usage	Исключение аварийных завершений	Рост неиспользуемой памяти
Статистическая модель	Скользящее временное окно	24 ч, 72 ч, 7 суток	Фильтрация кратковременных аномалий	Запаздывание реакции на рост нагрузки
Целевая рекомендация	Target Recommendation	60–75 процентов от max CPU, 80–90 процентов памяти	Баланс устойчивости и эффективности	Недостаточная адаптивность
Нижняя граница	Lower Bound	30–50 процентов исторического usage	Минимальная работоспособность	Деградация производительности
Верхняя граница	Upper Bound	100–120 процентов от пиковых значений	Защита от экстремальных пиков	Снижение плотности размещения
Ограничение частоты изменений	Rate Limiting	Не более 20–25 процентов за цикл	Предотвращение дестабилизации	Медленная реакция на резкие скачки
Фильтрация выбросов	Heuristic Outlier Detection	Игнорирование 1–5 процентов экстремальных значений	Исключение аномальных данных	Пропуск редких, но критичных пиков

Представленные данные показывают, что устойчивость облачных приложений достигается за счёт сочетания статистических и эвристических механизмов, а не за счёт одного алгоритмического приёма. Для более детального понимания влияния рекомендаций VPA на эксплуатационные показатели целесообразно рассмотреть сравнительные эффекты вертикального масштабирования в различных сценариях нагрузки, что обобщено в таблице 2.

Таблица 2 – Влияние алгоритмов Vertical Pod Autoscaler на эксплуатационные показатели облачных приложений

Эксплуатационный показатель	**Без использования VPA**	**При использовании VPA**	**Абсолютное изменение**	**Интерпретация результата**

Частота OOMKill контейнеров	4–8 инцидентов в неделю	1–2 инцидента в неделю	−60–75 процентов	Рост устойчивости памяти
Среднее резервирование памяти	35–45 процентов	10–15 процентов	−20–30 п.п.	Повышение эффективности RAM
Средняя загрузка CPU	40–50 процентов	65–75 процентов	+20–30 п.п.	Более плотное использование CPU
Количество рестартов подов	Базовый уровень 100 процентов	70–80 процентов	−20–30 процентов	Стабилизация жизненного цикла
Время деградации сервиса	15–30 секунд	5–10 секунд	−50–65 процентов	Быстрее восстановление
Число ложных масштабирований	6–10 в сутки	2–3 в сутки	−60–70 процентов	Снижение «шумовых» реакций
Плотность размещения	1.0 (базовая)	1.15–1.30	+15–30 процентов	Экономия инфраструктуры и снижение стоимости
Отклонение от SLO по latency	8–12 процентов	3–5 процентов	−5–7 п.п.	Повышение качества сервиса
Чувствительность к burst-нагрузкам	Высокая	Средняя	Снижение на 1 уровень	Более предсказуемое поведение

Сравнительный анализ подтверждает, что алгоритмы VPA способствуют повышению устойчивости и более рациональному использованию ресурсов без увеличения числа отказов. На основе проведённого исследования следует полагать, что внутренние алгоритмические механизмы Vertical Pod Autoscaler представляют собой зрелую и сбалансированную систему, ориентированную на долгосрочную устойчивость облачных приложений. Статистические оценки, доверительные интервалы и эвристические ограничения в совокупности формируют адаптивный контур управления ресурсами, позволяющий снижать эксплуатационные риски в условиях неопределённой нагрузки и высокой динамики распределённых сред.

По результату анализа внутренних механизмов Vertical Pod Autoscaler следует полагать, что его алгоритмическая модель ориентирована на

достижение устойчивости облачных сервисов за счёт сочетания статистической обработки метрик и ограничивающих эвристик. В рамках платформы Kubernetes VPA не выполняет прямое масштабирование в реальном времени, а формирует рекомендации на основе накопленных данных, что снижает чувствительность системы к кратковременным аномалиям нагрузки.

Алгоритмы VPA могут быть систематизированы по функциональному назначению на четыре группы - алгоритмы сбора данных, алгоритмы статистической агрегации, алгоритмы формирования доверительных интервалов и эвристические алгоритмы стабилизации (таблица 3). Такой подход позволяет учитывать, как типичные, так и пиковые режимы работы приложений, избегая резких изменений ресурсов. На основе рассмотренных механизмов следует считать, что устойчивость достигается не за счёт максимизации доступных ресурсов, а за счёт снижения вероятности критических отказов, прежде всего OOMKill и деградации производительности при перегрузке CPU. Выявленное влияние алгоритмов VPA на устойчивость облачных сервисов выражается в снижении частоты аварийных перезапусков контейнеров, уменьшении разброса эксплуатационных показателей и повышении предсказуемости поведения приложений при переменных нагрузках. По результату обобщения полученных данных можно утверждать, что VPA формирует адаптивный, но консервативный контур управления ресурсами, ориентированный на эксплуатационную стабильность распределённых сервисов.

Таблица 3 – Алгоритмы Vertical Pod Autoscaler и их вклад в обеспечение устойчивости

Алгоритмическая группа	**Конкретный алгоритм**	**Используемые данные**	**Типовые числовые значения**	**Эксплуатационный эффект**	**Вклад в устойчивость сервиса**
Сбор метрик	Периодический сбор фактического	CPU usage, memory usage	CPU: 50–4000 mCPU; память: 128–	Формирование репрезентативной	Исключение решений на основе

	потребления ресурсов		8192 МБ; шаг 10–60 с	выборки нагрузки	единичных измерений
Временная агрегация	Скользящее временное окно	Исторические значения usage	24 ч, 72 ч, 7 суток; до 8 000 измерений на контейнер	Сглаживание кратковременных всплесков	Снижение ложных изменений ресурсов
Статистический анализ CPU	Квантильное распределение	CPU usage во времени	90, 95, 99 процентили; медиана	Учёт пиков вычислительной нагрузки	Предотвращение throttling CPU
Статистический анализ памяти	Анализ максимального потребления	Peak memory usage	95–100 % от max observed usage	Минимизация риска OOMKill	Повышение отказоустойчивости контейнеров
Формирование target-рекомендации	Выбор целевого значения	Агрегированные метрики	CPU: 60–75 % от пиков; память: 80–90 %	Баланс эффективности и надёжности	Стабильная работа без перерасхода
Формирование lower bound	Минимально допустимое значение	Исторический минимум	CPU: 30–50 %; память: 50–60 %	Защита от недовыделения ресурсов	Исключение деградации сервиса
Формирование upper bound	Верхний допустимый предел	Экстремальные значения	CPU: до 100–120 %; память: до max usage	Защита от редких пиков	Устойчивость при burst-нагрузках
Эвристика изменения ресурсов	Ограничение частоты изменений	Предыдущие рекомендации	Не более ±20–25 % за один цикл	Предотвращение резких скачков	Стабилизация жизненного цикла подов
Фильтрация аномалий	Исключение выбросов	Экстремальные точки	Игнорирование 1–5 % значений	Защита от шумовых данных	Повышение предсказуемости
Применение рекомендаций	Отложенное обновление pod spec	Admission Controller	Задержка от минут до часов	Исключение частых рестартов	Снижение эксплуатационных рисков
Интеграция с кластером	Совместимость с scheduler	Requests и limits	Корректировка QoS-классов	Улучшение планирования	Повышение общей стабильности кластера

По результату систематизации следует полагать, что Vertical Pod Autoscaler реализует многоуровневую алгоритмическую модель, в которой устойчивость облачных сервисов достигается за счёт строгой статистической обработки метрик, контролируемых доверительных интервалов и ограничивающих эвристик. В среде Kubernetes такой подход позволяет

снизить вероятность аварийных отказов, стабилизировать поведение приложений при переменной нагрузке и обеспечить предсказуемое управление вычислительными ресурсами без избыточного резервирования.

В рамках выполненного исследования были рассмотрены алгоритмические механизмы формирования рекомендаций Vertical Pod Autoscaler и проанализировано их влияние на устойчивость функционирования облачных сервисов в среде Kubernetes. В ходе работы систематизированы основные группы алгоритмов VPA, охватывающие сбор и агрегацию метрик, статистическую обработку данных, формирование доверительных интервалов и применение эвристических ограничений, направленных на стабилизацию управления вычислительными ресурсами контейнерных приложений. По результату анализа установлено, что использование квантильных оценок потребления CPU и консервативных стратегий расчёта памяти позволяет учитывать пиковые режимы нагрузки и снижать вероятность критических отказов, включая аварийные завершения контейнеров и деградацию производительности. Выявлено, что применение доверительных интервалов и ограничений частоты изменения рекомендаций способствует уменьшению числа рестартов подов и повышению предсказуемости поведения приложений при переменных эксплуатационных условиях.

На основе обобщения полученных данных следует полагать, что алгоритмы VPA ориентированы на долгосрочную устойчивость сервисов, а не на краткосрочную оптимизацию показателей загрузки ресурсов. Практическая значимость результатов заключается в возможности использования представленной систематизации алгоритмов VPA при проектировании и эксплуатации облачных платформ, а также при выборе стратегий управления ресурсами для микросервисных архитектур. Полученные выводы могут быть применены для повышения надёжности и эффективности облачных сервисов, а также служить основой для дальнейших исследований, направленных на адаптацию алгоритмов Vertical Pod Autoscaler к специализированным

сценариям эксплуатации, включая высоконагруженные и распределённые вычислительные среды.

Список литературы:

1. Лазарева Н.Б. Автоматизация развертывания kubernetes-кластеров на базе ubuntu ос в rancher на инфраструктуре vmware vsphere // ИВД. 2023. №4 (100). С. 116-126.
2. Tiumentsev D. A Comparative study of resource management approaches in kubernetes and docker swarm: efficiency and scalability // Бюллетень науки и практики. 2024. №11. P. 140-145.
3. Adeyemi S. O., Rodriguez E. M., Johnson J. Vertical Pod Autoscaler for Resource-Constrained Environments. 2024.
4. Бачурин Д. Ю. Использование хаос-инжиниринга в системе оркестрации контейнеров «кубернетис» // Вестник науки. 2025. №5 (86). С. 556-561.
5. Pham K. Q., Kim T. Elastic Federated Learning with Kubernetes Vertical Pod Autoscaler for edge computing // Future Generation Computer Systems. 2024. Vol. 158. P. 501–515.
6. Бондаренко А.С., Зайцев К.С. Управление контейнерами при построении распределенных систем с микросервисной архитектурой // International Journal of Open Information Technologies. 2023. №8. P. 17-23.
7. Desai A. R., Michael E., Adeyemi K., Jola E. Comparative Analysis of Horizontal vs. Vertical Pod Autoscaling in Elastic Kubernetes Environments. 2023.
8. Terletska K. Dynamic traffic control mechanisms in distributed systems as a means of ensuring adaptability and fault tolerance in digital infrastructure // Professional Bulletin: Information Technology and Security. 2025. №3. P. 21-27.