DOI 10.58351/2949-2041.2026.35.6.034

Княжев Алексей Викторович, студент
Московский государственный технический университет
им. Н. Э. Баумана

Романова Татьяна Николаевна
доцент кафедры ИУ-7, кандидат физико-математических наук
Московский государственный технический университет
им. Н. Э. Баумана

МЕТОД РАСПРЕДЕЛЕНИЯ РЕСУРСОВ НА ОСНОВЕ ДОМИНАНТНО-ОСТАТОЧНОГО МЕТОДА И МЕТОДА ПЕРЕРАСПРЕДЕЛЕНИЯ НА ПЛАТФОРМЕ KUBERNETES

Аннотация. Предложен подход для распределения ресурсов на платформе Kubernetes на основе доминантного метода с учетом остаточного критерия и методе перераспределения. Произведен подбор параметров метода, обеспечивающих наибольшее количество размещенных экземпляров приложений. Произведено сравнение предложенного метода со стандартным планировщиком Kubernetes.

Ключевые слова: Планировщик, Kubernetes, распределенная система, кластер, оркестрация, контейнеризация.

Введение. На данный момент многие организации выбирают микросервисный подход к разработке информационных систем. Для запуска приложений используется технология контейнеризации [1]. Системы оркестрации контейнеров, такие как Kubernetes, который фактически стал стандартом индустрии, упрощают развертывание контейнеризированных приложений. Ключевой задачей оркестратора является распределение физических ресурсов вычислительного кластера между контейнерами [2].

В общем случае, задачу распределения ресурсов в вычислительном кластере можно описать так: необходимо выбрать, какой рабочий узел будет выполнять условное задание. В основном, под рабочим узлом подразумевается физический сервер, а под заданием – приложение [3].

Задача распределения ресурсов в вычислительном кластере является NP-трудной [4, 5] и не может быть решена за полиномиальное время [6].

Для задачи планирования традиционным решением являются эвристические алгоритмы [6]. Эвристические алгоритмы нацелены на универсальность, а также легко понимаются и реализуются. Примеры таких алгоритмов включают «Round-Robin» [7], «First Fit» [8] и «Max-Min» [9], которые хорошо работают при определённых нагрузках. В рамках данной статьи рассматривается комбинация методов распределения по доминантному ресурсу с учетом остаточного критерия и метода перераспределения.

Методы. Для дальнейшего описания метода необходимо ввести следующие обозначения.

- $A_{n\times q}^{(0)}$ – матрица доступных ресурсов на узлах перед размещением экземпляров приложений за исключением уже занятых системными компонентами ресурсов. $A_{i,k}^{(0)}$ – величина ресурса k, доступного на узле i, $A_{i,k}^{(0)} \in N$.
- t – номер размещаемого экземпляра приложения, $1 \leq t \leq \tau$. При размещении экземпляра t не известны характеристики экземпляров $t+1$, $t+2$ и так далее.
- r_t – вектор запросов на ресурсы экземпляра t, размерностью q. $r_{t,k}$ – запрос на ресурс k экземпляром t, $r_{t,k} \in N \cup \{0\}$. Является расчетной величиной планировщика, значение поля resources.requests для каждого из контейнеров в рамках экземпляра приложения.

- $A_{n\times q}^{(t)}$ – матрица доступных ресурсов на узлах после размещения экземпляра приложения t. $A_{i,k}^{(t)}$ – величина ресурса k, доступного на узле i, $A_{i,k}^{(t)} \in N \cup \{0\}$.
- $c(t,i) \in \{0,1\}$ – предикат, задающий нересурсные ограничения Kubernetes. Значение $c(t,i) = 1$ означает, что экземпляр t разрешено разместить на рабочем узле i с точки зрения меток, ограничений и допусков, требований к томам и иных нересурсных правил. Значение $c(t,i) = 0$ означает, что узел i для данного экземпляра недопустим независимо от свободных ресурсов.
- I_t – вектор рабочих узлов, допустимых для размещения экземпляра t. Размерность вектора n. $I_{t,i} \in \{0,1\}$, где i – номер узла.

Тогда

$$\mathrm{I_{t,i}} = \begin{cases} 1, \text{если } \mathrm{c(t,i)} = 1 \text{ и } \mathrm{r_{t,k}} \le \mathrm{A_{i,k}^{(t-1)}}, 1 \le \mathrm{k} \le \mathrm{q}, \\ 0, \text{иначе.} \end{cases}$$

$$I\_t\hat{}\{\} = \} = \{\mathrm{i} \in \mathrm{N}\colon 1 \le \mathrm{i} \le \mathrm{n}, \mathrm{I\_\{t,i\}} = 1\}$$

описывает множество рабочих узлов, допустимых для размещения экземпляра t.

Если $\mathrm{I_t'} = \emptyset$, то в рассматриваемом состоянии экземпляр t не может быть размещен и остается в ожидании.

Для каждого допустимого рабочего узла $i \in I_t'$ можно рассчитать показатель относительной занятости каждого ресурса k, который будет достигнут, если разместить экземпляр t на узле i.

Для краткости дальнейшего изложения данный показатель будет называться потенциальной нормированной загрузкой ресурса k после размещения экземпляра j.

$$\rho_{\mathrm{i,k}}(\mathrm{t}) = \frac{\mathrm{A_{i,k}^{(0)}} - \mathrm{A_{i,k}^{(t-1)}} + \mathrm{r_{t,k}}}{\mathrm{A_{i,k}^{(0)}}}.$$

Из ресурсной допустимости текущего состояния и определения множества I_t' следует, что для каждого $i \in I_t'$ и $1 \le k \le q$ выполнено $0 \le \rho_{i,k}(t) \le 1$.

Поскольку будущие экземпляры неизвестны в момент принятия решения, в оперативном режиме используются последовательные эвристические правила выбора рабочего узла [10]. Данное правило в дальнейшем будет обозначаться как $i_*(t)$ и получает номер узла, наиболее подходящего для запуска экземпляра приложения t.

Доминантно-остаточный метод. Для учета будущих типовых запросов введем множество типовых профилей ресурсных запросов B.

Типовые профили соответствуют повторяющимся классам экземпляров баз данных или обслуживающих компонентов, для которых заранее известны характерные запросы процессорного ресурса, памяти и иных учитываемых ресурсов. Множество типовых профилей B формируется из заранее заданного каталога ресурсных конфигураций рассматриваемых приложений. Для всех $b \in B$ и $1 \leqslant k \leqslant q$ предполагается $b_k \geqslant 0$.

Для определения показателя остаточной пригодности для вектора свободных ресурсов $a \in (\mathbb{N} \cup \{0\})^q$, введем показатель кратности размещения типового профиля b в остатке ресурсов a:

$$m(b,a) = \min_{1 \leqslant k \leqslant q:\ b_k > 0} \frac{a_k}{b_k}.$$

Если $m(b,a) < 1$, то профиль b не помещается в остатке даже в одном экземпляре. Если $m(b,a) \geqslant 1$, то величина $m(b,a)$ показывает, сколько экземпляров профиля b потенциально может быть размещено в данном остатке по наиболее дефицитному ресурсу.

Пусть $C \geqslant 1$ – параметр насыщения, ограничивающий вклад очень больших остатков.

Если для типовых профилей известны частоты или вероятностные веса, можно ввести коэффициенты

$$\omega_b > 0, b \in B.$$

Тогда, показатель остаточной пригодности с оценкой кратности размещения типовых профилей имеет вид:

$$H\ (a) = \frac{\sum_{b \in B} \omega_b \min\ (\frac{m\ (b,a)}{C}, 1)}{\sum_{b \in B} \omega_b}.$$

Показатель $H\ (a)$ не решает задачу упаковки будущих заявок; он служит однократной суррогатной оценкой того, какие типовые профили еще не исключены данным остатком ресурсов.

Если экземпляр t разместить на рабочем узле i, то остаток ресурсов этого узла станет равен

$$\hat{A}_i^{(t)} = A_i^{(t-1)} - r_t.$$

Если $i \in I_t^{'}$, то $\hat{A}_i^{(t)} \in\ (\mathbb{N} \cup \{0\})^q$, то есть является вектором неотрицательных чисел из q элементов. Потеря остаточной пригодности при размещении экземпляра t на рабочем узле i задается формулой

$$\delta_i\ (t) = H\ (A_i^{(t-1)}) - H\ (\hat{A}_i^{(t)}), i \in I_t^{'}.$$

Чем больше $\delta_i\ (t)$, тем сильнее размещение экземпляра t ухудшает пригодность остатка ресурсов узла i для будущих типовых профилей.

Поскольку $\hat{A}_i^{(t)} \leqslant A_i^{(t-1)}$ покомпонентно, из определения $H\ (a)$ следует, что $H\ (\hat{A}_i^{(t)}) \leqslant H\ (A_i^{(t-1)})$. Следовательно,

$$0 \leqslant \delta_i\ (t) \leqslant 1.$$

Величина $\varphi_i\ (t)$ также лежит в отрезке $\lceil 0,1 \rceil$ для всех $i \in I_t^{'}$. Поэтому оба слагаемых критерия являются безразмерными нормированными показателями.

Доминантно-остаточный метод выбора рабочего узла задается функцией

$$L_i\ (t) = \lambda \varphi_i\ (t) +\ (1 - \lambda)\delta_i\ (t), i \in I_t^{'}, 0 \leqslant \lambda \leqslant 1.$$

Параметр λ задает относительную важность доминантной загрузки. При $\lambda = 1$ критерий совпадает с чистым методом доминантного ресурса; при $\lambda < 1$ дополнительно учитывается потеря остаточной пригодности.

Параметр λ рассматривается как заранее заданный внешний параметр метода. Его выбор не входит в процедуру планирования отдельного экземпляра.

Рабочий узел выбирается по правилу

$$i_*\ (t) \in \arg\min_{i \in I_t^{'}}\ L_i\ (t).$$

При равенстве значений $L_i\ (t)$ выбор выполняется по заранее заданному детерминированному порядку рабочих узлов.

После выбора рабочего узла $i_*\ (t)$ расчетное состояние обновляется: занятые ресурсы выбранного узла увеличиваются на r_t, а свободный остаток уменьшается на r_t. Для остальных рабочих узлов расчетные остатки не изменяются. Следующий экземпляр приложения рассматривается уже в обновленном состоянии.

Метод перераспределения. Если объект Pod остается в состоянии Pending после неуспешной попытки планирования из-за отсутствия допустимого рабочего узла, то соответствующий экземпляр приложения далее называется экземпляром, ожидающим размещения. Наличие таких экземпляров служит условием запуска проверки возможности ограниченного перераспределения.

Множество уже размещенных экземпляров определяется как

$$P^{(t)} = \{j \in \{1, \dots, t\}: \sum_{i=1}^{n} S_{j,i}^{(t)} = 1\}.$$

Множество экземпляров, ожидающих размещения, определяется как

$$W^{(t)} = \{j \in \{1, \dots, t\}: \sum_{i=1}^{n} S_{j,i}^{(t)} = 0\}.$$

Если $W^{(t)} = \emptyset$, то ограниченное перераспределение не запускается, поскольку все рассмотренные экземпляры уже размещены.

Если $W^{(t)} \neq \emptyset$, то каждый экземпляр $\ell \in P^{(t)}$ может быть рассмотрен как кандидат на принудительное вытеснение.

Для кандидата $\ell \in P^{(t)}$ используется предикат безопасности

$$Q(\ell) \in \{0,1\}.$$

Значение $Q(\ell) = 1$ означает, что внешняя проверка безопасности разрешает принудительное вытеснение экземпляра ℓ в текущем снимке состояния кластера. Если проверка безопасности недоступна или возвращает неопределенный результат, принимается $Q(\ell) = 0$.

В рамках данной работы предполагается, что предикат безопасности реализуется администратором кластера Kubernetes в соответствии с особенностями запускаемых приложений, поэтому формализовываться не будет.

Корректность предиката безопасности рассматривается как внешнее предположение модели: если $Q(\ell) = 1$, то вытеснение экземпляра ℓ не нарушает заданные эксплуатационные ограничения в текущем снимке состояния кластера. В минимальном варианте предикат должен запрещать вытеснение системных и критичных экземпляров, экземпляров без управляющего объекта, а также экземпляров, вытеснение которых может нарушить доступность приложения, кворум реплик или иные заданные пользователем ограничения.

Для кандидата $\ell \in P^{(t)}$ в расчетном состоянии освобождаются ресурсы, занятые этим экземпляром на его текущем рабочем узле. После этого в фиксированном порядке выполняется пробное повторное планирование экземпляров из множества $W^{(t)} \cup \{\ell\}$ по локальному критерию L.

Порядок пробного повторного планирования задается заранее и не зависит от результатов рассматриваемого кандидата: сначала рассматриваются экземпляры из $W^{(t)}$ в порядке времени поступления в состояние Pending, затем рассматривается вытесняемый экземпляр ℓ. При равенстве времени поступления выбирается минимальный объект в заранее заданном порядке экземпляров приложения, то есть лексикографически по паре «пространство имен, техническое имя объекта».

На каждом шаге используются ресурсные и нересурсные проверки Kubernetes в текущем расчетном состоянии. Если очередной экземпляр нельзя разместить ни на одном рабочем узле, он остается неразмещенным в рамках данной пробной процедуры, а расчетное состояние не изменяется.

Пусть $R_{\ell}^{(t)}$ – множество экземпляров, оставшихся неразмещенными после пробного вытеснения ℓ и повторного размещения множества $W^{(t)} \cup \{\ell\}$.

Показатель полезности вытеснения задается формулой

$$G_{\ell}^{(t)} = |W^{(t)}| - |R_{\ell}^{(t)}|.$$

Показатель $G_{\ell}^{(t)}$ задает изменение числа неразмещенных экземпляров.

Условие $G_{\ell}^{(t)} > 0$ означает, что пробное перераспределение уменьшает число экземпляров, остающихся без размещения, даже с учетом необходимости повторно разместить сам вытесняемый экземпляр. Таким образом, если $G_{\ell}^{(t)} > 0$, при удалении данного экземпляра приложения количество ожидающих размещения экземпляров будет меньше.

Множество допустимых кандидатов на вытеснение имеет вид

$$E^{(t)} = \{\ \ell \in P^{(t)} : Q(\ell) = 1,\ G_{\ell}^{(t)} > 0,\ \ell \notin R_{\ell}^{(t)}\ \}.$$

Условие $\ell \notin R_{\ell}^{(t)}$ означает, что сам вытесняемый экземпляр должен быть повторно размещен в пробной процедуре. Поэтому перераспределение не допускает превращения уже размещенного экземпляра в новый ожидающий экземпляр ради формального улучшения числа размещенных новых заявок.

Если $E^{(t)} = \emptyset$, перераспределение не выполняется. Иначе кандидат на вытеснение выбирается по правилу

$$\ell_* \in \arg\max_{\ell \in E^{(t)}} G_\ell^{(t)}.$$

Если максимум $G_\ell^{(t)}$ достигается на нескольких кандидатах, выбор между ними выполняется по заранее заданному детерминированному порядку экземпляров. Это обеспечивает воспроизводимость решения и не изменяет определение показателя полезности вытеснения.

Исследование. Было проведено исследование методом запуска экземпляров приложений из тестовой выборки, имитирующей запуск баз данных в вычислительном кластере Kubernetes.

Характеристики оборудования, на которых производились замеры, следующие.

- Процессор: Apple M1 Pro.
- Объем оперативной памяти: 16 ГБ

Выборка данных была сгенерирована на основе данных из открытых источников, характер данных имеет следующие особенности.

- Типовые виды экземпляров приложений – экземпляры СУБД Redis и MongoDB.
- Частота экземпляров СУБД Redis – 75%, СУБД MongoDB – 25% [11].
- Типовые профили для СУБД Redis представлены двух видов – с использованием постоянного хранения данных, и в режиме кеширования. Частота экземпляров с постоянным хранением – 15%, с кеширующим – 85%.
- Для каждой из трех категорий СУБД выделяется по 4 типовых профиля: профиль стендовой СУБД, профиль промышленной версии СУБД небольшого размера, рабочей промышленной СУБД среднего размера, промышленной версии СУБД большого размера. Предполагается, что частота типовых профилей 50%, 25%, 12,5%, 12,5% соответственно.

При генерации для дисковых характеристик рабочих узлов используется детерминированное отклонение от типового значения до 8%, чтобы учесть неоднородность дисковой подсистемы.

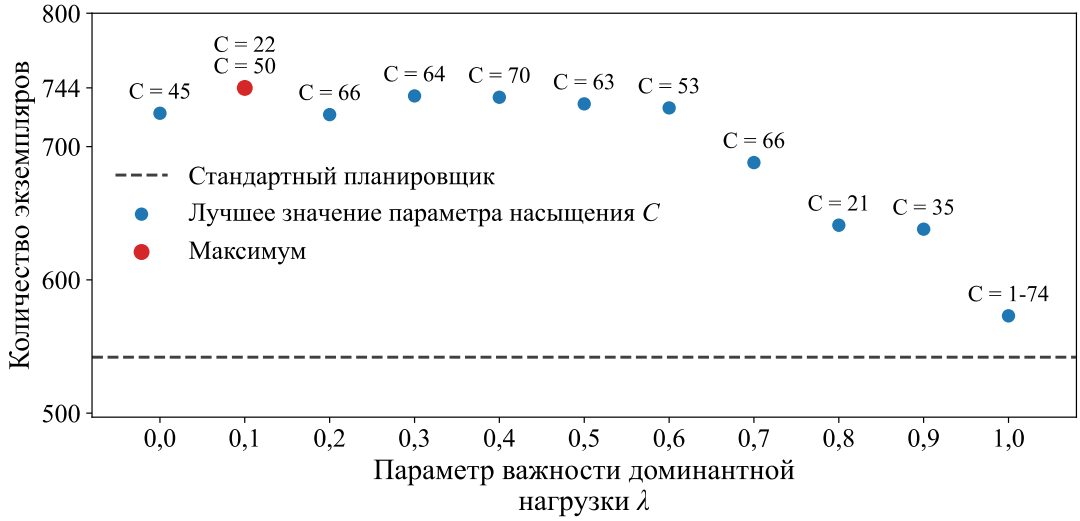
На основе вышеуказанных правил было сгенерировано 17 рабочих узлов и 1000 экземпляров приложений.

Для локального запуска исследования использовался набор инструментов KWOK [12]. Этот набор позволяет добавлять в Kubernetes-кластер рабочие узлы без запущенного демона kubelet. Таким образом, можно имитировать работу большого количества узлов и экземпляров приложений в кластере.

Результаты. На рисунке 1 представлен график зависимости количества размещенных экземпляров приложений от параметра λ. Для одного λ выбран наилучший результат по всем C.

Рисунок 1

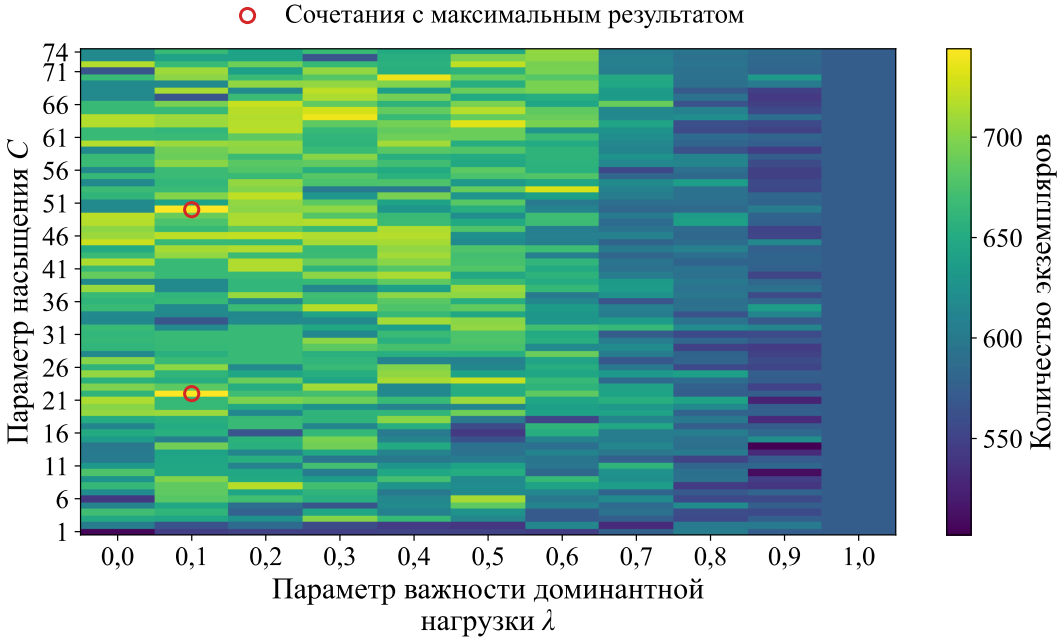
Результаты тестирования реализованного метода при различных значениях параметров C и λ



На рисунке 2 представлена тепловая карта количества размещенных экземпляров приложений от параметров λ и C.

Рисунок 2

Тепловая карта количества размещенных экземпляров приложений от параметров C и λ



Обсуждение. Разработанное программное обеспечение применимо в кластерах Kubernetes, в которых рабочие узлы имеют неоднородные ресурсные характеристики, а запускаемые экземпляры приложений образуют повторяющиеся типовые ресурсные профили. В таких условиях учет остаточной пригодности позволяет выбирать рабочий узел не только по текущему уровню заполнения ресурсов, но и по тому, насколько оставшийся набор ресурсов будет пригоден для размещения последующих экземпляров приложений.

Результаты тестирования показывают, что на исследуемой выборке разработанный метод во всех рассмотренных сочетаниях параметров разместил больше экземпляров приложений, чем стандартный планировщик Kubernetes. Максимальный результат составил 744 экземпляра приложений при $C = 22$, $\lambda = 0{,}1$ и $C = 50$, $\lambda = 0{,}1$, тогда как стандартный планировщик разместил 542 экземпляра приложений. Разработанный метод разместил на 37,3% больше экземпляров приложений, чем стандартный планировщик. Следовательно, применение разработанного программного обеспечения целесообразно в ресурсно ограниченных кластерах, где основным показателем качества является увеличение числа размещенных экземпляров приложений без нарушения ресурсных и нересурсных ограничений Kubernetes.

Заключение. В статье представлено описание нового подхода на основе доминантного метода распределения ресурсов с учетом остаточной пригодности и метода перераспределения для платформы Kubernetes. Представленный метод на тестовой выборке разместил на 37,3% больше экземпляров приложений, чем стандартный планировщик Kubernetes

Список литературы:

1. Кочер П. С. Микросервисы и контейнеры Docker. – Litres, 2022.

2. Carri´on C. Kubernetes scheduling: Taxonomy, ongoing issues and challenges // ACM Computing Surveys. – 2022. – Т. 55, № 7. – С. 1 – 37.

3. Scheduling in distributed systems: A cloud computing perspective / L. F. Bittencourt [и др.] // Computer science review. – 2018. – Т. 30. – С. 31 – 54.

4. Mor B., Shabtay D., Yedidsion L. Heuristic algorithms for solving a set of NP-hard single-machine scheduling problems with resource-dependent processing times // Computers & Industrial Engineering. – 2021. – Т. 153. – С. 107024.

5. Cloud resource scheduling with deep reinforcement learning and imitation learning / W. Guo [и др.] // IEEE Internet of Things Journal. – 2020. – Т. 8, № 5. – С. 3576 – 3586.

6. Marrouche W. Unlocking the Potential of Metaheuristics for NP-Hard Problems: дис.... канд. / Marrouche Wissam. – University of Portsmouth, 2024.

7. An ameliorated Round Robin algorithm in the cloud computing for task scheduling / N. Ghazy [и др.] // Bulletin of Electrical Engineering and Informatics. – 2023. – Т. 12, № 2. – С. 1103 – 1114.

8. Keshri R., Vidyarthi D. P. Communication-aware, energy-efficient VM placement in cloud data center using ant colony optimization // International Journal of Information Technology. – 2023. – Т. 15, № 8. – С. 4529 – 4535.

9. Advanced cost-aware Max–Min workflow tasks allocation and scheduling in cloud computing systems / M. Raeisi-Varzaneh [и др.] // Cluster computing. – 2024. – С. 1 – 13.

10. Borodin A., El-Yaniv R. Online computation and competitive analysis. – cambridge university press, 2005.

11. Княжев А. Доклад «Как мы снизили до нуля операционку по приемке Kubernetes-кластеров для баз данных» на Positive Hack Days [Электронный ресурс]. – URL: https://vkvideo.ru/video-28022322_456241335 (дата обращения: 15.04.2026).

12. Официальный сайт KWOK [Электронный ресурс]. – URL: https://kwok.sigs.k8s.io/ (дата обращения: 15.04.2026)