ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ТЕЛЕКОММУНИКАЦИИ

УДК 004.258 DOI: 10.46573/2658-5030-2026-2-103-109

УПРАВЛЕНИЕ РЕСУРСАМИ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ ДЛЯ ОПТИМИЗАЦИИ РАБОТЫ ФУНКЦИОНАЛЬНО СВЯЗАННЫХ КОМПОНЕНТОВ ПРОГРАММНОГО ПРИЛОЖЕНИЯ

М.А. ГЛЕБСКИЙ, магистрант, И.А. ЕГЕРЕВА, канд. техн. наук

Тверской государственный технический университет,
170026, Тверь, наб. Аф. Никитина, 22, e-mail: mglebskiy@mail.ru

В статье рассмотрена задача оптимального распределения ресурсов вычислительной системы – оперативной памяти и процессорного времени – между виртуальными контейнерами в составе сложных распределенных приложений. Проанализированы основные проблемы управления большими системами, такие как конфликты зависимостей, конкуренция за сетевые порты и несбалансированное потребление ресурсов. Показано, что традиционные подходы к распределению ресурсов, ориентированные на максимизацию суммарной производительности, оказываются неэффективными в условиях сильной функциональной взаимозависимости контейнеров, поскольку приводят к возникновению «узких мест». Предложена максиминная постановка задачи распределения ресурсов с учетом минимальных требований к пропускной способности каждого компонента, с тем чтобы обеспечить сбалансированную загрузку всех элементов системы и повысить ее эффективность.

Ключевые слова: виртуализация, виртуальный контейнер, Docker.

ВВЕДЕНИЕ

Инфраструктура современных приложений сложна. Как правило, каждое приложение состоит из нескольких частей. Так, самое простое современное web-приложение состоит из 3 частей: базы данных (например, postgresql [1]), серверного приложения, которое отвечает за бизнес логику, и http-сервера (например, nginx [2]), отвечающего за доставку пользователям клиентской части приложения. Для функционирования такой системы необходимо, чтобы все ее элементы находились в активном состоянии [3, 4].

Каждому из этих приложений нужны зависимости, порты для взаимодействия с другими приложениями и оперативная память для работы. Под «зависимостями» будем понимать библиотеки, модули, драйверы, а также другие приложения, необходимые для работы.

На данном этапе разработчики сталкиваются с рядом эксплуатационных проблем. Во-первых, возникают конфликты зависимостей: различные приложения требуют несовместимые версии одной и той же библиотеки, что затрудняет их совместное развертывание в единой среде. Во-вторых, происходит конфликт сетевых портов: несколько сервисов пытаются занять один и тот же порт, что приводит к сбоям

при запуске или недоступности отдельных компонентов. В-третьих, отдельные приложения демонстрируют избыточное потребление оперативной памяти. Это обычно обусловлено неоптимальной работой с кэшированием, утечками памяти в долгоживущих процессах или обработкой объемных данных без применения потоковой передачи. В результате такие приложения истощают системные ресурсы, снижая производительность платформы в целом и вызывая нестабильность работы соседних сервисов.

Если разработчик имеет дело с минимальной конфигурацией системы, задача разрешения конфликтов не является сложной. Однако по мере роста функциональности увеличивается количество приложений. Так, для обеспечения высоконагруженности и высокой скорости их выполнения добавляется списочная база данных для кэширования, например redis [5], и полнотекстовый поиск, например elastic search [6], а для интеграции между приложениями системы используется менеджер потоков событий, например kafka [7]. Получаем уже 6 приложений, которые не должны мешать друг другу функционировать, а некоторые из них должны стабильно взаимодействовать.

По мере роста нагрузки на систему добавляются дублирующие сервера и балансировщик нагрузки, файловое хранилище, аналитические системы, системы логирования и т.д. Как правило, управление таким количеством приложений становится весьма затруднительным.

Таким образом, существует проблема управления системой, состоящей из большого количества приложений, в том числе управления зависимостями, сетевым взаимодействием и распределением ресурсов вычислительной системы.

ЗАДАЧА РАСПРЕДЕЛЕНИЯ РЕСУРСОВ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

Одним из способов решения проблемы развертывания крупных систем является ввод дополнительного слоя абстракции – виртуальных контейнеров. Наиболее часто используемым решением является Docker [8] или его альтернатива с открытым исходным кодом – Podman [9]. Docker является платформой для упаковки, доставки и запуска приложений в изолированных контейнерах. Виртуальный контейнер отличается от виртуальной машины. Docker оперирует именно контейнерами (рисунок).

Сравним основные особенности работы виртуальной машины и виртуального контейнера:

виртуальная машина виртуализирует аппаратное обеспечение. Каждая виртуальная машина содержит свою собственную полноценную гостевую операционную систему (ОС), которая работает поверх хостовой ОС через слой гипервизора[2]. Это требует много ресурсов (ядре процессора, оперативной памяти, постоянной памяти) и увеличивает время запуска, но создает полную изоляцию;

Docker-контейнер виртуализирует ОС. Все контейнеры на одной машине используют ядро хостовой ОС, но работают в изолированных пространствах процесса. Это делает их невероятно легковесными, быстрыми в запуске и требующими значительно меньше ресурсов.

Хотя виртуальные контейнеры более легковесные, чем виртуальные машины, и позволяют куда проще дублировать элементы, все еще остается проблема рационального распределения ресурсов ОС для получения максимальной производительности.

[2] Гипервизор (монитор виртуальных машин) — «программа, создающая среду функционирования других программ (в том числе других операционных систем) за счет виртуализации аппаратных ресурсов вычислительной системы» [10].

Сравнение виртуальной машины и виртуального контейнера

МАТЕМАТИЧЕСКАЯ ПОСТАНОВКА ЗАДАЧИ

Решению аналогичных задач посвящено достаточно большое количество работ специалистов в области программной инженерии и оптимального управления [10–14]. В данной работе приводится подход к решению задачи распределения ресурсов, основанный на применении виртуальных контейнеров для изоляции и управления системой и максиминного подхода к распределению ресурсов.

Определим задачу: распределить ресурсы вычислительной системы между контейнерами для получения максимальной производительности. Под производительностью контейнера будем иметь в виду производительность компонента системы, заключенного в контейнер.

Приведем входные параметры, используемые при решении задачи распределения ресурсов:

- объем оперативной памяти;
- количество ядер процессора;
- количество контейнеров;
- производительность контейнеров.

Обозначим переменные:

- m_i – объем памяти, выделенный i-му контейнеру;
- c_i – объем ядер, выделенный i-му контейнеру;
- n – количество контейнеров.

Тогда задача распределения ресурсов эквивалентна следующей задаче:

$$\sum_{i=1}^{n} F_i(m_i, c_i) \rightarrow \max, \quad (1)$$

где $F_i(m_i, c_i)$ – функция производительности для i-го контейнера при выделенных m_i и c_i при ограничениях.

Общая память, выделенная контейнерам, не должна превышать общий объем доступной оперативной памяти M:

$$\sum_{i=1}^{n} m_i \leq M.$$

Общее количество ядер, выделенное контейнерам, не должно превышать общий объем доступных ядер процессора C:

$$\sum_{i=1}^{n} c_i \leq C.$$

Каждый контейнер должен получать не менее минимально допустимого объема памяти l_i и ядер d_i:

$$m_i \geq l_i, \quad i = 1, \ldots, n,$$
$$c_i \geq d_i, \quad i = 1, \ldots, n.$$

Однако при такой формулировке задачи возникает проблема «узкого места»[3], связанная с производительностью наименее эффективного элемента, так как контейнеры являются частью единой системы и влияют на работу друг друга и функционирование всей системы.

Рассмотрим архитектуру, состоящую из двух контейнеров: первый (шина данных) отвечает за прием входных запросов и передачу результатов клиенту, второй (вычислительный сервис) осуществляет непосредственную обработку данных. При распределении вычислительных ресурсов пропорционально трудоемкости операций (например, 80 % ресурсов выделяется сервису обработки, 20 % – шине данных) теоретически достигается баланс с точки зрения вычислительной задачи. Однако на практике подобное распределение ведет к дисбалансу производительности компонентов: шина данных, обладая ограниченными ресурсами, не способна обеспечить пропускную способность, достаточную для загрузки вычислительного сервиса. В результате сервис значительную часть времени находится в простое, ожидая поступления данных, что снижает общую производительность системы и делает изначально «оптимальное» распределение ресурсов неэффективным в реальных условиях эксплуатации.

Предлагаемые способы решения проблемы «узкого места»:

1. Ввести в задачу весовой коэффициент w_i, отображающий значимость элемента в системе со стороны выделяемых ресурсов. Значения весовых коэффициентов индивидуальны для каждой системы и зависят от ее составляющих, подбор значений осуществляется экспертом.

Тогда задача о распределении ресурсов примет вид

$$\sum_{i=1}^{n} w_i \, F_i(m_i, c_i) \rightarrow \max,$$

где w_i – значимость контейнера при распределении ресурсов.

2. Перейти к максиминной задаче. В этом случае будем считать, что производительность системы определяется контейнером с наименьшей производительностью, так как последний является «узким местом» и тормозит работу всей системы. Тогда задача о распределении ресурсов примет вид

$$F_i(m_i, c_i) \rightarrow \max \min,$$

т.е. необходимо распределить ресурсы таким образом, чтобы контейнер с минимальной производительностью принял свое максимальное значение.

3. Ввести дополнительное ограничение, которое будет отображать реальную работоспособность контейнера в соответствии с его функциональной спецификацией.

[3] Узкое место – это компонент, процесс или этап, ограничивающий общую производительность системы или рабочую скорость. В соответствующем англоязычном термине «bottleneck» (бутылочное горлышко) прослеживается аналогия с горловиной бутылки, узость которой не позволяет вылить или высыпать все ее содержимое сразу, даже если ее перевернуть.

Для этого введем еще один входной параметр – r_i, т.е. количество запросов в секунду, которые должен обрабатывать контейнер, а функция производительности отображает реальное количество запросов, которое выполняет элемент при заданных m_i и c_i. Тогда

$$F_i(m_i, c_i) \geq r_i ,$$
$$\sum_{i=1}^{n} F_i(m_i, c_i) \rightarrow \max.$$

В таком случае каждый элемент системы гарантированно будет выполнять свой функционал. Однако это не отменяет того, что при выполнении ограничений оставшиеся ресурсы распределятся между контейнерами с лучшей степенью производительности, т.е. проблема «узкого места» остается.

Решение задачи заключается в объединении способов 1 и 3, т.е. необходимо ввести дополнительный входной параметр и соответствующее ему ограничение по желаемой производительности контейнера и представить задачу в максиминной форме:

$$F_i(m_i, c_i) \rightarrow \max \min$$

при ограничениях

$$m_i \geq l_i,\ i = 1, \dots, n,\ \sum_{i=1}^{n} m_i \leq M;$$
$$c_i \geq d_i,\ i = 1, \dots, n, \sum_{i=1}^{n} c_i \leq C;$$
$$F_i(m_i, c_i) \geq r_i , i = 1, \dots, n,$$

где m_i – объем памяти, выделенный i-му контейнеру; c_i – объем ядер, выделенный i-му контейнеру, n – количество контейнеров; l_i – минимальный объем памяти, необходимый i-му контейнеру; d_i – минимальный объем ядер, необходимый i-му контейнеру; r_i – количество запросов в секунду, которые должен обработать контейнер; $F_i(m_i, c_i)$ – функция производительности контейнера, отображающая пропускную способность (запросов в секунду) от выделенных оперативной памяти и ядер; M – общий объем оперативной памяти; C – общее количество ядер процессора.

Таким образом, находим максимальное значение функции при минимальных ресурсах и заданных ограничениях по минимальной производительности и максимальному объему ресурсов, т.е. решение, при котором производительность самого слабого контейнера будет максимальной.

Логическое развитие решения приведенной задачи связано с дополнительными условиями применения горизонтального масштабирования[4], в случае если вертикальное масштабирование[5] становится неэффективным.

РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЯ

В ходе исследования была сформулирована задача оптимального распределения ресурсов вычислительной системы, таких как оперативная память и процессорные ядра, между виртуальными контейнерами с целью достижения максимальной производительности всей системы. Учтены базовые ограничения: ограниченность общих ресурсов, минимальные требования к ресурсам каждого контейнера, а также

[4] Горизонтальное масштабирование – это подход к увеличению производительности и пропускной способности системы путем добавления дополнительных узлов (серверов, экземпляров) в кластер, при этом нагрузка распределяется между ними с помощью балансировщика [13].

[5] Вертикальное масштабирование – это подход к повышению производительности системы путем наращивания аппаратных ресурсов существующего узла: увеличения количества процессоров (ядер), объема оперативной памяти, производительности дисковой подсистемы или сетевых интерфейсов [13].

необходимость обеспечения заданной пропускной способности (в запросах в секунду) для каждого компонента системы. Предложена максиминная постановка задачи, сочетающая введение нижних границ производительности с максимизацией наихудшего показателя в системе. Такой подход позволяет избежать эффекта «узкого места», при котором перераспределение ресурсов в пользу наиболее эффективных контейнеров снижает общую производительность из-за дисбаланса нагрузки. Сформулированная постановка задачи распределения ресурсов вычислительной системы может быть использована как основа для разработки алгоритмов динамического управления ресурсами в системах управления контейнерами, таких как Docker или Podman, или системах оркестрации, таких как Kubernetes.

ЗАКЛЮЧЕНИЕ

Виртуальные контейнеры значительно упрощают развертывание сложных распределенных систем и управление ими, однако требуют продуманного подхода к распределению вычислительных ресурсов. В работе показано, что классическая оптимизация по суммарной производительности не всегда применима в условиях взаимозависимости компонентов системы. Предложенная постановка задачи распределения ресурсов вычислительной системы с учетом функциональных требований к каждому контейнеру позволяет достичь сбалансированного распределения ресурсов и повысить общую эффективность работы системы.

Дальнейшие исследования могут быть направлены на применение предложенного подхода при реализации сценариев горизонтального масштабирования, а также на его интеграцию с системами управления и оркестрации контейнеров.

ЛИТЕРАТУРА

1. Postgresql. URL: https://www.postgresql.org/ (дата обращения: 07.02.2026).
2. Nginx. URL: https://nginx.org/ (дата обращения: 07.02.2026).
3. Карпов В.Е., Коньков К.А. Операционные системы, среды и оболочки. URL: https://scinetwork.ru/disk/file/1555 (дата обращения: 03.02.2026).
4. Таненбаум Э.С., Вудхалл А.С. Операционные системы. Разработка и реализация / пер. с англ., 3-е изд. СПб.: Питер, 2007. 704 с.
5. Redis. URL: https://redis.io/ (дата обращения: 07.02.2026).
6. Elasticsearch. URL: https://www.elastic.co/elasticsearch (дата обращения: 07.02.2026).
7. Kafka. URL: https://nginx.org/ (дата обращения: 07.02.2026).
8. Docker. URL: https://www.docker.com/ (дата обращения: 07.02.2026).
9. Podman. URL: https://podman.io/ (дата обращения: 07.02.2026).
10. Кравченко Ю.А., Курситыс И.О. Комбинированный подход к решению задачи распределения ресурсов // *Известия Южного федерального университета. Технические науки.* 2017. № 7 (192). С. 111–122.
11. Босов А.В. Обобщенная задача распределения ресурсов программной системы // *Информатика и ее применения*. 2014. Т. 8. № 2. С. 39–47.
12. Баранов А.В., Николаев Д.С. Использование контейнерной виртуализации в организации высокопроизводительных вычислений // *Программные системы: теория и приложения*. 2016. Т. 7. № 1 (28). С. 117–134.
13. Бондаренко А.С., Зайцев К.С. Управление контейнерами при построении распределенных систем с микросервисной архитектурой // *International Journal of Open Information Technologies*. 2023. Т. 11. № 8. С. 17–23.
14. Коновалов О.А., Коновальчук Е.В., Сербулов Ю.С. Решение задачи равномерного распределения ресурсов методом динамического программирования // *Лесотехнический журнал*. 2016. Т. 6. № 3 (23). С. 248–254.

Для цитирования: Глебский М.А., Егерева И.А. Управление ресурсами вычислительной системы для оптимизации работы функционально связанных компонентов программного приложения // Вестник Тверского государственного технического университета. Серия «Технические науки». 2026. № 2 (30). С. 103–109.

RESOURCE MANAGEMENT IN COMPUTING SYSTEMS FOR OPTIMIZING THE PERFORMANCE OF FUNCTIONALLY RELATED SOFTWARE APPLICATION COMPONENTS

M.A. GLEBSKY, magister, I.A. EGEREVA, Cand. Sc.

Tver State Technical University
22, Af. Nikitin emb., Tver, 170026, e-mail: mglebskiy@mail.ru

The article addresses the problem of optimal allocation of computational resources - specifically RAM and CPU time – among virtual containers within complex distributed applications. It analyzes key challenges in managing systems composed of numerous interconnected components, including dependency conflicts, competition for network ports, and unbalanced resource consumption. The study demonstrates that traditional resource allocation approaches, which focus on maximizing aggregate performance, prove ineffective under conditions of strong functional interdependence among containers, as they tend to create system bottlenecks. To address this issue, a mathematical model for resource allocation is proposed that incorporates minimum throughput requirements for each component. This approach aims to ensure balanced loading across all system elements and thereby enhance overall system efficiency.

Keywords: virtualization, virtual containers, Docker.

Поступила в редакцию/received: 24.02.2026; после рецензирования/revised: 26.02.2026; принята/accepted: 02.03.2026

УДК 681.5 DOI: 10.46573/2658-5030-2026-2-109-119

СИСТЕМНЫЙ ПОДХОД К ДАЛЬНЕМУ ПРОГНОЗИРОВАНИЮ РАЗВИТИЯ ХОЛОДИЛЬНЫХ КОМПРЕССОРОВ С ИСПОЛЬЗОВАНИЕМ АНАЛИЗА КЛЮЧЕВОГО ПРОТИВОРЕЧИЯ

С.Л. ГОРОБЧЕНКО[1], канд. техн. наук, С.А. МЕШКОВ[2], канд. техн. наук,
Н.Н. ВЕРНЕР[3], канд. техн. наук, С.А. ВОЙНАШ[4], мл. научн. сотр.,
В.А. СОКОЛОВА[1], канд. техн. наук

[1]Санкт-Петербургский государственный университет промышленных технологий и дизайна, 191186, Санкт-Петербург, ул. Большая Морская, 18,
e-mail: sgorobchenko@yandex.ru
[2]Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова, 190005, Санкт-Петербург, ул. 1-я Красноармейская, 1,
e-mail: meshkovcergey@mail.ru
[3]Санкт-Петербургский государственный лесотехнический университет им. С.М. Кирова, 194021, Санкт-Петербург, Институтский пер., 5,
e-mail: wernern@mail.ru