

DOI: 10.32517/0234-0453-2025-40-2-79-87



АНАЛИЗ СОВРЕМЕННЫХ ПОДХОДОВ К БАЛАНСИРОВКЕ НАГРУЗКИ СЕРВЕРНЫХ КЛАСТЕРОВ В ОБРАЗОВАТЕЛЬНЫХ ИНФОРМАЦИОННЫХ СИСТЕМАХ И СООТВЕТСТВУЮЩИХ АЛГОРИТМОВ

Р. В. Сущенко^{1,2} ✉, В. Е. Петров¹¹ Московский государственный технологический университет «Станкин», г. Москва, Россия² Фирма «1С», г. Москва, Россия

✉ newromeones@gmail.com

Аннотация

Системы управления обучением (*англ.* Learning Management System, LMS) играют ключевую роль в образовательном процессе, предоставляя студентам разнообразные виды образовательного контента, а преподавателям — инструменты для контроля знаний обучающихся и оценки прогресса. Однако с увеличением числа студентов возрастает и нагрузка на каналы передачи данных, что может привести к снижению качества обучения. Чтобы решить эту проблему, требуется комплексно подойти к проблеме балансировки нагрузки кластера серверов образовательных платформ.

В статье рассмотрены основные существующие алгоритмы балансировки, даны их комплексное сравнение и оценка применимости к образовательным платформам. Особое внимание уделено сетям доставки контента (*англ.* Content Delivery Network, CDN) и их важной роли в этом процессе. Проанализированы современные тенденции интеграции балансировщика с облачными технологиями, выделена особая важность интеллектуального управления кешированием и оркестрации контейнеров как одного из основных способов увеличения скорости работы LMS и их отказоустойчивости. Предложена архитектура кластеров серверов «1С:Предприятие 8» с практическими схемами построения масштабируемых инфраструктур для онлайн-обучения.

Ключевые слова: кластеры серверов, балансировка нагрузки, LMS, CDN, алгоритмы распределения, отказоустойчивость образовательных систем, облачная оркестрация, «1С:Предприятие 8».

Для цитирования:

Сущенко Р. В., Петров В. Е. Анализ современных подходов к балансировке нагрузки серверных кластеров в образовательных информационных системах и соответствующих алгоритмов. *Информатика и образование*. 2025;40(2):79–87. DOI: 10.32517/0234-0453-2025-40-2-79-87.

ANALYSIS OF MODERN APPROACHES TO LOAD BALANCING OF SERVER CLUSTERS IN EDUCATIONAL INFORMATION SYSTEMS AND CORRESPONDING ALGORITHMS

R. V. Suschenya^{1,2} ✉, V. E. Petrov¹¹ Moscow State Technological University "STANKIN", Moscow, Russia² 1C Company, Moscow, Russia

✉ newromeones@gmail.com

Abstract

Learning Management Systems (LMS) play a critical role in the educational process, providing students with a variety of educational content, and teachers with tools for monitoring students' knowledge and assessing progress. However, as the number of students increases, the load on data transmission channels also increases, potentially leading to a decline in the quality of education. To solve this problem, a comprehensive approach to load balancing within server clusters of educational platforms is required.

The article reviews the main existing load balancing algorithms, provides a comparative analysis, and assesses their applicability to educational platforms. Particular attention is paid to Content Delivery Networks (CDN) and their critical role in the process. Modern trends in integrating load balancers with cloud technologies are analyzed, emphasizing the importance of intelligent cache management and container orchestration as key methods for increasing the performance and fault tolerance of LMS platforms. A server cluster architecture based on 1C:Enterprise 8 is proposed, along with practical schemes for building scalable infrastructures for online learning.

Keywords: server clusters, load balancing, LMS, CDN, distribution algorithms, fault tolerance of educational systems, cloud orchestration, IC:Enterprise 8.

For citation:

Suschenya R. V., Petrov V. E. Analysis of modern approaches to load balancing of server clusters in educational information systems and corresponding algorithms. *Informatics and Education*. 2025;40(2):79–87. (In Russian.) DOI: 10.32517/0234-0453-2025-40-2-79-87.

1. Введение

Современные образовательные информационные системы крайне требовательны к грамотному и эффективному распределению ресурсов и нагрузки для обеспечения высокой производительности, а также отказоустойчивости и надежности. Подобные системы управления обучением одновременно обслуживают тысячи пользователей, количество которых постоянно растет. Постоянный рост объемов обрабатываемых данных, увеличение количества входящих запросов и усложнение решаемых задач часто приводят к ситуациям, когда часть серверов перегружена, а другая, наоборот, используется недостаточно. При этом можно наблюдать общее падение производительности, которое может выражаться как в виде задержек в обработке запросов, так и в виде сбоев системы в часы пиковой нагрузки из-за того, что какой-то один сервер из всего кластера стал «узким горлышком».

Данная проблема может быть решена простым увеличением мощностей кластера серверов, но на определенном этапе это станет нерентабельно. **Балансировка нагрузки** — это критически важный процесс, который отвечает за то, на каком сервере из кластера будет обрабатываться запрос или решаться входящая задача. Этот механизм позволяет минимизировать время отклика системы, избежать нежелательных перегрузок серверов и с максимальной эффективностью использовать вычислительные ресурсы. Таким образом, грамотная балансировка нагрузки позволяет как повысить качество обслуживания, минимизируя время простоя системы, так и снизить затраты на поддержку инфраструктуры [1, 2].

В настоящее время существует множество самых разных алгоритмов, которые отвечают решению задачи балансировки. Это как традиционные методы, которые включают статические и динамические алгоритмы, так и современные адаптивные подходы с использованием нейронных сетей и машинного обучения.

Целью статьи является проведение комплексного сравнительного анализа различных существующих подходов к балансировке нагрузки в серверных кластерах образовательных платформ, а также определение возможностей оптимизации этих подходов и оценка перспектив их применения с использованием отечественных программных продуктов.

2. Классический подход к балансировке нагрузки

Следует понимать, что балансировка нагрузки может осуществляться на разных уровнях — как на сетевом, так и на транспортном, и даже на при-

кладном. При работе с образовательными системами основным является именно способ программной балансировки на прикладном уровне, так как это позволяет максимально точно проанализировать запросы пользователей и грамотно перераспределить их между узлами кластера, в зависимости от размера, сложности и запроса определенных ресурсов.

Традиционные алгоритмы балансировки отличаются крайне простой реализацией и малыми расходами на поддержание алгоритма, но, к сожалению, они не учитывают изменчивую нагрузку серверов, сложность запроса и другие важные параметры.

Наиболее простыми являются **статические алгоритмы**, а их основным представителем является **алгоритм циклического распределения Round Robin** [3], суть которого в том, что все входящие запросы, независимо от их «тяжести», циклически направляются на все серверы кластера, т. е. по очереди. И это очень часто приводит к ситуациям, когда на сервер, занятый очень сложной вычислительной задачей, продолжают падать новые запросы, которые он не может обработать. Существует улучшенная версия алгоритма с названием **«взвешенный Round Robin»** (Weighted Round Robin) [4], который присваивает каждому серверу свой «вес» в зависимости от мощности, и такие серверы получают больше запросов относительно более слабых. Но проблема, связанная с отсутствием реакции у таких алгоритмов на изменчивость нагрузки в реальном времени, все еще остается.

Также к статическим алгоритмам относятся:

- **Sticky Sessions**, в котором клиент закрепляется за определенным сервером на время сеанса;
- **Destination Hash Scheduling** — сервер для обработки запроса выбирается по IP-адресу получателя;
- **Source Hash Scheduling** — работает аналогично предыдущему, но только сервер выбирает по IP-адресу отправителя;
- **Hash/IP Hash** — когда клиент навсегда привязывается к серверу по IP-адресу [5].

Итоговое сравнение статических алгоритмов представлено в таблице 1.

Проблема, связанная с отсутствием реакции у статических алгоритмов на изменчивость нагрузки в реальном времени, решается благодаря **динамическим алгоритмам**, которые в режиме реального времени отслеживают состояние всех серверов в кластере и распределяют входящий трафик запросов с учетом текущей загрузки.

Одним из основных примеров таких алгоритмов является алгоритм наименьшего числа соединений

Таблица 1 / Table 1

Сравнительная таблица статических методов балансировки

Comparative table of static balancing methods

№ п/п	Метод	Описание	Достоинства	Недостатки
1	Round Robin	Циклическое распределение запросов	Простота и равномерность нагрузки	Нет учета состояния сервера
2	Weighted Round Robin	Циклическое распределение запросов с учетом веса узлов	Учет разной мощности серверов	Нет мониторинга нагрузки серверов
3	Sticky Sessions	Один клиент — один сервер на время сессии	Позволяет сохранить состояние сессии	Риск перегрузки одного сервера
4	Destination Hash Scheduling	Выбор сервера по хешу значения	Предсказуемость	Вероятность неравномерной нагрузки
5	Source Hash Scheduling	Выбор сервера по хешу источника	Постоянство соединения	Вероятность неравномерной нагрузки
6	Hash/IP Hash	Статическое распределение запросов на основе IP-адреса	Стабильность соединения, простота реализации	Вероятность неравномерной нагрузки

Least Connections [3]. Механизм очень простой — алгоритм отслеживает количество активных соединений у каждого сервера и направляет запрос тому, у кого их меньше всего. Предполагается, что чем меньше у сервера соединений, тем быстрее он сможет обработать новое. И такой подход работает куда лучше, чем Round Robin, но только с одним условием — среда должна быть однородной, т. е. все запросы по сложности должны быть примерно одинаковыми. Как только начинают появляться запросы с разной «тяжестью», алгоритм может сработать так, что несколько наиболее ресурсозатратных задач будут отправлены на один и тот же сервер. По аналогии с Weighted Round Robin существует и **Weighted Least Connections**, который добавляет каждому серверу коэффициент его мощности, который учитывается при распределении.

Другие динамические алгоритмы:

- **алгоритм минимального времени ответа (Least Response Time)**: распределение запросов между серверами происходит по следующему правилу: запрос направляется на тот сервер, который в текущий момент времени имеет минимальное время ответа [6];
- **случайный выбор сервера (Random Two Choices)**: улучшенная версия алгоритма Random Choice, который почти не используется в своем простейшем виде. Механизм работы заключается в выборе двух случайных серверов, после чего запрос направляется на менее загруженный;
- **дрессельный алгоритм**: ограничивает число соединений с целью предотвратить перегрузку. Работает благодаря таблице индексов виртуальных машин (или серверов), в которой фиксирует их состояние;
- **модифицированный дрессельный алгоритм**: отличается от предыдущего метода

тем, что имеет динамический порог нагрузки, который автоматически меняется в зависимости от состояния сервера;

- **ESCE (Execution Scheduling Current Execution)**: соотносит размер запроса с приоритетом и отдает задание самой свободной машине. Является более сложным в поддержании алгоритмом, особенно с учетом накладных расходов на поддержание таблицы индексов серверов.

Отличием динамических алгоритмов от статических является то, что динамические методы минимально, но все же учитывают состояние сервера (табл. 2).

3. Современные адаптивные алгоритмы балансировки нагрузки

С развитием облачных технологий, микросервисов, сложных образовательных систем возникла необходимость в «умных» балансировщиках, которые способны подстраивать распределение трафика под текущую ситуацию, учитывая множество параметров кластера серверов, а также сложность запросов. В зависимости от сложности запроса, будь это просмотр лекции, прохождение теста или выгрузка больших данных, система должна реагировать поразному и перенаправлять запрос на наиболее подходящий по множеству параметров сервер. Таким образом, перед распределением запроса происходит его классификация по «тяжести» для сервера и запросу присваивается вес. Далее алгоритм балансировки комбинирует информацию о весе запроса с показателями загруженности серверов и перенаправляет его на самый свободный. Данный подход реализован на прикладном уровне, что позволяет контент-ориентированным алгоритмам, которые работают в системах с различными видами данных и активностей,

Таблица 2 / Table 2

Сравнительная таблица динамических методов балансировки

Comparative table of dynamic balancing methods

№ п/п	Метод	Описание	Достоинства	Недостатки
1	Least Connections	Распределение на основе минимального количества соединений	Учитывает текущую нагрузку	Не учитывает сложность запросов
2	Weighted Least Connections	Least Connections с учетом мощности серверов	Более точное понимание загруженности серверов	Сложность реализации
3	Least Response Time	Распределение на основе минимального времени ответа	Минимальная задержка ответа	Требуется постоянный мониторинг времени отклика
4	Random Two Choices	Выбор наименее загруженного из двух серверов	Простой и эффективный	Чем меньше узлов, тем меньше эффективность
5	Дроссельный алгоритм	Ограничивает число соединений для предотвращения перегрузки	Защита от перегрузки	Возможна потеря части запросов
6	Модифицированный дроссельный алгоритм	Динамическое ограничение нагрузки	Адаптивность, меньшие потери	Сложность реализации
7	Execution Scheduling Current Execution	Балансировка по текущему исполнению задач	Равномерное распределение нагрузки с учетом сложности задач	Требуется сложная система мониторинга

например на платформах онлайн-обучения, грамотно распределять запросы и не терять в производительности даже в часы пиковой нагрузки.

К современным алгоритмам можно отнести **био-вдохновленные алгоритмы**, а также **алгоритмы на основе машинного обучения, контент-ориентированные и гибридные алгоритмы**.

Наиболее интересной с точки зрения происхождения является категория **«природных» алгоритмов**, которые были вдохновлены естественными процессами. Такие алгоритмы используются в больших распределенных сетях, облаках и сложных вычислительных задачах.

Наиболее известными являются **генетические алгоритмы** (англ. Genetic Algorithms) [7], метод балансировки в которых основан на принципах естественного отбора. В результате работы алгоритма серверы оцениваются на основе определенных критериев, самые успешные из которых потом наследуются, а самые слабые, наоборот, больше не участвуют в процессе балансировки.

Также в эту категорию относят **алгоритм оптимизации роем частиц** (англ. Particle Swarm Optimization) [8–10] и **алгоритм муравьиных колоний** (англ. Ant Colony Optimization) [11, 12]. Первый метод вдохновлен коллективным поведением рыб и птиц: каждая частица (агент) ищет оптимальное решение (путь) как на основе своего лучшего опыта, так и на основе опыта своих соседей. Благодаря такому подходу для каждой задачи находится оптимальный сервер. Второй основан на поведении колоний муравьев: условный «муравей» исследует все доступные ему пути, перемещаясь по серверам; чем лучше найденный маршрут, тем больше на

нем оставлено «феромонов» (пометок). Такие пути чаще используются для распределения запросов, но если сервер стал менее эффективен, то «феромоны» испаряются, благодаря чему и обеспечивается динамическая адаптация. Как было сказано выше, подобные алгоритмы применяются скорее для задач распределения в облаках или крупных системах, нежели на обучающих онлайн-платформах, в связи с их высокой сложностью.

Вторая категория — это **алгоритмы на основе машинного обучения** — новая веха в развитии балансировщиков нагрузки. Основная их идея — это прогнозирование состояния системы на определенный момент времени и проактивное распределение трафика. Подобные алгоритмы анализируют историю трафика, нагрузку, количество запросов и еще множество параметров и с помощью обученных моделей оценивают, какой сервер лучше всего подойдет для текущего запроса. Основное преимущество подобных балансировщиков — это предиктивная балансировка, а также то, что система сама находит скрытые зависимости. И самое важное, что со временем такие алгоритмы будут все точнее и лучше предоставлять решения проблем, дообучаясь в процессе работы. Но у подобных алгоритмов и много минусов. Во-первых, это время на обучение и сбор всех необходимых качественных данных. Во-вторых, сложность разработки и поддержки, так как требуются специалисты из области машинного обучения и ресурсы на периодическое переобучение. И в-третьих, это накладные расходы на поддержку CPU/GPU-модулей. Поэтому имеет смысл внедрять такие алгоритмы только в крупных системах, где выигрыш от оптимального распределения окупит затраты [6, 13].

Последняя категория из рассматриваемых адаптивных алгоритмов — это **гибридные алгоритмы**. Они комбинируют в себе несколько подходов, получая лучшее из них. В гибридных алгоритмах один алгоритм перекрывает минусы второго, например, соединяя алгоритм Least Connections с алгоритмами машинного обучения, которые будут прогнозировать рост трафика. Недостатком подобных методов является их усложненная архитектура и высокие затраты на поддержку. К таким алгоритмам можно отнести HSLJF [14], DCLASB, WAMLB [15] и др.

4. Балансировка нагрузки в образовательных онлайн-системах

Если обратиться к образовательным средам онлайн-обучения (LMS — Learning Management Systems, MOOK-платформам — Massive Open Online Courses), то они должны работать бесперебойно под любой нагрузкой, причем активно меняющейся. Поэтому для них выбор алгоритма балансировки является крайне важной, но в то же время сложной задачей.

Для начала следует отметить, что для крупных облачных платформ применяются в том числе и не рассмотренные ранее методы. Один из ключевых —

это **DNS-балансировка**, когда запросы клиентов распределяются по географически разнесенным центрам обработки данных (ЦОД) [16, 17]. Данный механизм использует, например, Coursera, для того чтобы обеспечить себе горизонтальное масштабирование по регионам.

Помимо этого применяются **аппаратные балансировщики нагрузки**, которые представляют собой физический прибор, предназначенный для распределения входящего трафика. Он обеспечивает лучшую производительность за счет специализированного аппаратного обеспечения, предназначенного исключительно для балансировки нагрузки. Но к минусам подобного метода следует отнести высокую цену и сложность изменения алгоритма балансировки, который уже находится внутри.

Если говорить о более стандартных методах балансировки, то следует отметить следующую тенденцию взаимодействия классов алгоритмов с сетями обучения (табл. 3).

Для более точной балансировки в LMS и MOOK используется механизм **сетей доставки контента** — **CDN** (англ. Content Delivery Network). Это система распределения контента по серверам разных регионов, позволяющая ускорить загрузку веб-сайтов

Таблица 3 / Table 3

Соотношение классов алгоритмов балансировки с образовательными онлайн-системами
Relationship of classes of balancing algorithms with online educational systems

№ п/п	Класс алгоритмов	Используются ли в LMS	Для чего используются	Почему используются или редко используются	Примеры применения
1	Статические алгоритмы (Round Robin, Weighted RR, IP Hash)	Очень часто	Простая балансировка для малых и средних образовательных систем	Простота реализации и настройки	Локальные LMS вузов и школ
2	Динамические алгоритмы (Least Connections, Weighted Least Connections, Least Response Time, ESCE)	Очень часто	Эффективная адаптация к нагрузке и минимальное время отклика	Баланс между сложностью и эффективностью	Coursera, edX, Udemy, Stepik, Moodle
3	DNS-балансировка	Часто	Географическое распределение нагрузки	Необходима для крупных международных платформ	Coursera, edX, Udemy, Khan Academy, Open edX
4	Контент-ориентированные алгоритмы балансировки (LBLC [18], Destination Hash, CDN-oriented)	Очень часто	Быстрый доступ к видеоконтенту, кеширование статических ресурсов, более быстрый доступ к материалам	Простота реализации и высокая скорость доставки материалов	Coursera, Udemy, edX, Stepik, Skillbox, GeekBrains, Khan Academy
5	Алгоритмы балансировки с машинным обучением	Очень редко	Теоретически — адаптивное прогнозирование нагрузки	Слишком высокая стоимость и требуемая вычислительная мощность	Практических внедрений нет, только экспериментальные исследования
6	Био-вдохновленные алгоритмы	Никогда	Теоретически — адаптивное распределение нагрузки	Избыточная сложность реализации	Практических внедрений нет в открытом доступе
7	Гибридные алгоритмы балансировки	Очень редко	Адаптивное распределение массовых мероприятий	Слишком высокая стоимость и затраты на инфраструктуру	Практических внедрений нет в открытом доступе

и уменьшить нагрузку на исходный сервер. Принцип работы CDN состоит в том, что содержимое веб-сайта (текст, изображения, видео и другие ресурсы) хранится (кешируется) на узлах CDN, которые находятся ближе к конечному пользователю. Таким образом, когда пользователь запрашивает содержимое веб-сайта, оно будет загружено с ближайшего к нему узла CDN, что ускоряет работу сайта. Каждый узел CDN поддерживает множество соединений и обеспечивает параллельную передачу данных, что значительно уменьшает время загрузки контента. Кроме того, CDN уменьшает нагрузку на исходный сервер, что позволяет выдерживать большой трафик и улучшает производительность всего сайта [19, 20].

Система CDN является крайне мощным инструментом в контент-ориентированной балансировке нагрузки. Она уменьшает число запросов на основной сервер, забирая основную нагрузку на себя, позволяя студентам и преподавателям получать контент в максимально короткий срок.

Если проводить сравнительный анализ LMS, то можно прийти к выводу, что они используют общие принципы балансировки, а именно равномерное распределение нагрузки и устранение «горячих точек». Но конкретная реализация у разных LMS немного отличается. Кто-то используется фронтальные веб-серверы с балансировкой (Nginx или Apache mod_cluster), например система Moodle. Коммерческие MOOK-платформы, такие как edX или Coursera, изначально создавались с расчетом на массовость, поэтому они опираются на мощности глобальных провайдеров, а также интегрируют балансировку с авто-масштабированием, за счет чего обеспечивают почти бесконечную масштабируемость. Система CDN позволяет получать контент любому пользователю этих платформ в кратчайшие сроки.

5. Влияние балансировки на образовательный процесс

Очевидным фактом является то, что надежность и быстродействие онлайн-платформ напрямую влияют на качество образовательного процесса. И **эффективная балансировка нагрузки позволяет избежать множества негативных сценариев:**

- **Простой системы.** Если образовательный портал «падает» во время экзамена или иного важного события, это вызывает стресс у обучающихся и преподавателей. Резервирование мощностей и кластеризация с распределением запросов позволяют этого избежать.
- **Долгое время отклика.** При решении тестирований на время или при работе с интерактивной средой время отклика системы является крайне важным показателем. Поэтому балансировка с учетом минимального времени отклика и текущей загруженности серверов является очень важной. Если все будет работать быстро, то студенты будут тратить время на обучение, а не на ожидание загрузки.

- **Невозможность запуска новых форматов обучения.** Онлайн-формат обучения продолжает развиваться, и появляются новые виды обучения, например, автоматическая проверка заданий студентов в изолированной среде. Подобный метод требует большого количества ресурсов и создает весовую нагрузку на систему. Без грамотной балансировки подобные сервисы или не работали бы совсем, или работали бы очень медленно.
- **Отток учащихся.** Процент слушателей, которые прошли обучение до конца, является очень важным показателем как для курса, так и для образовательной платформы в целом. И если решить проблему наполнения балансировка не в силах, то качество технической составляющей, количество технических проблем и т. п. лежат на плечах алгоритмов балансировки. Ведь чем лучше работает образовательная система, тем у студента больше вовлеченности и уверенности в том, что он все сможет сдать вовремя и в любой момент сможет ознакомиться с любым доступным ему материалом.

Таким образом, можно сделать вывод, что балансировка нагрузки является очень незаметным со стороны, но крайне важным фактором успешности обучения слушателей. Никто не задумывается о том, как устроены кластеры серверов, как происходит распределение трафика и откуда мы получаем данные, но все ощущают результат этой работы: стабильность и скорость.

6. Перспективы развития балансировки

Говоря о перспективах развития балансировки, следует отметить **несколько основных направлений:**

- **Улучшение локализации и кэширования данных.** Основные запросы к MOOK-системам — это чтение данных лекций, тестов, статей и т. д. В настоящее время уже используется система CDN, которая сокращает количество нагрузки на центральный сервер. Но если использовать механизм улучшенной локализации, то при получении данных будут учитываться не только географическое положение и наличие данных в CDN, но и еще факторы по загрузке локального сервера, а также другие, более комплексные настройки.
- **Автономные адаптивные алгоритмы.** Внедрение более современных и опирающихся на машинное обучение и нейронные сети алгоритмов, которые будут более автономными и смогут работать предиктивно, предсказывая нагрузку и выделяя под нее ресурсы.
- **Интеграция с облачной оркестрацией.** Более масштабное внедрение в образовательные онлайн-системы механизма контейнеров, а также систем управления ими, благодаря чему получается качественно повысить гибкость и масштабируемость учебных платформ.

7. Структура построения серверного кластера образовательной системы с применением платформы «1С:Предприятие 8»

При развертывании образовательной системы с использованием кластера серверов 1С необходимо определить следующее:

- требуемый алгоритм балансировки «поверх» балансировки самого кластера;
- пороговое значение коллективной нагрузки серверов кластера;
- собираемые с сервера параметры.

Следует отметить, что реализованный в самом кластере алгоритм балансировки является доработанной версией Least Response Time с некоторыми доработками для более точного определения загрузки одного сервера, но он не позволяет при необходимости добавлять в кластер дополнительные мощности или разворачивать дополнительные кластеры.

Рассмотрим следующий пример. Имеется отдельный кластер серверов 1С, на котором расположена образовательная система. В кластере находятся два сервера «А» и «Б» неравной мощности. В базовой реализации кластер на вход получает поток заявок, причем их «тяжесть», назначение и другие параметры не учитываются при балансировке. Кластер на основе доработанного алгоритма Least Response Time распределяет заявки между серверами. Если в кластере есть сервер «В», который по умолчанию выключен и не принимает заявки, автоматический балансировщик не может его включить (рис. 1).

Предлагаемое решение предполагает создание балансировщика «поверх» кластера. Благодаря такому решению получится решить следующие проблемы стандартного механизма:

- внешний по отношению к кластеру балансировщик сможет отслеживать общее состояние

всего кластера на основе кастомизируемого алгоритма, который можно изменять при необходимости;

- в балансировщик можно включить механизм адресации, который позволит распределять разные типы запросов, а также разные по «тяжести» запросы между различными кластерами серверов;
- появится возможность внешнего управления всей сетью кластеров, автоматического включения новых серверов в кластер или добавления новых кластеров;
- добавляется возможность оркестрации с помощью Kubernetes¹ с целью разворачивания новых контейнеров, содержащих свои собственные кластеры серверов, что обеспечивает высокую скорость масштабирования;
- появится возможность учитывать все необходимые метрики, которые не включены в стандартный механизм.

За счет этих преимуществ внешнего балансировщика получится построить систему, более устойчивую к нагрузкам и способную более грамотно оперировать запросами и их перенаправлением на наиболее подходящие серверы (рис. 2).

Таким образом, внедрение внешнего балансировщика в систему кластера серверов дистанционного обучения может стать ключевым фактором в обеспечении стабильного и надежного процесса онлайн-обучения, что, в свою очередь, повысит качество образовательного процесса и удовлетворенность студентов и преподавателей.

¹ Kubernetes — это платформа с открытым исходным кодом для автоматизации развертывания, масштабирования и управления приложениями в контейнерах (изолированных средах).

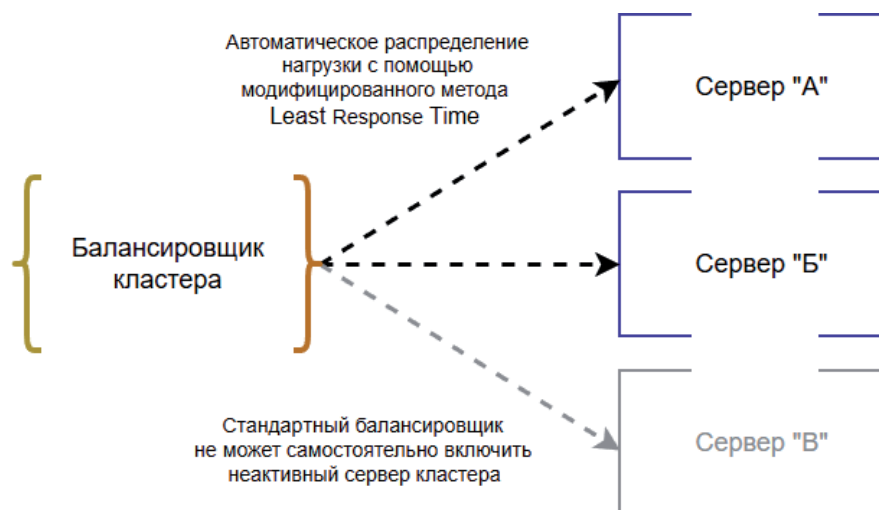


Рис. 1. Схематическое описание работы стандартного балансировщика

Fig. 1. Schematic description of the operation of a standard balancer

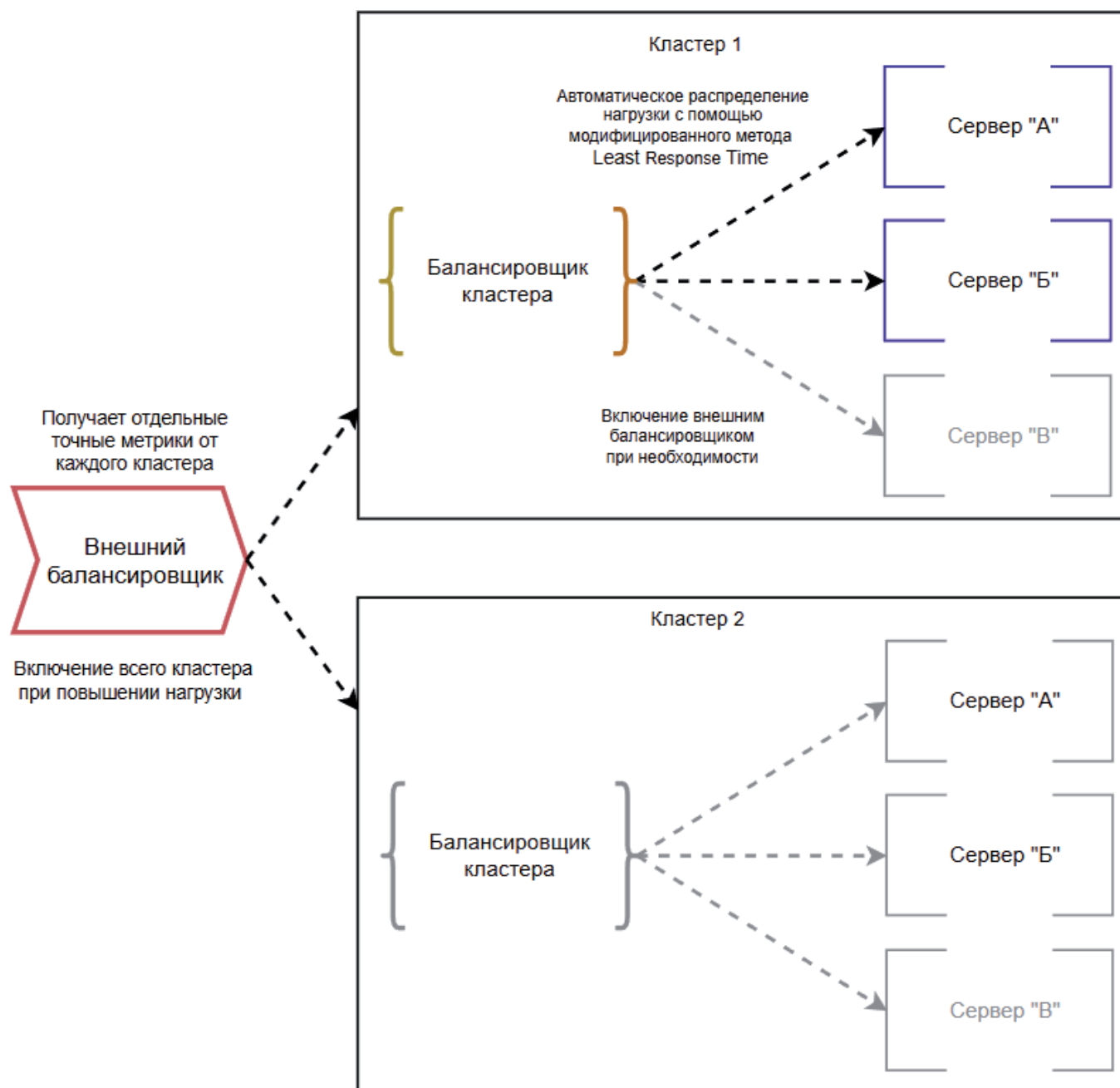


Рис. 2. Схематическое описание работы внешнего балансировщика

Fig. 2. Schematic description of the operation of an external balancer

8. Заключение

Балансировка нагрузки в серверных кластерах образовательных систем является одним из важнейших факторов, который определяет производительность, устойчивость, а также масштабируемость онлайн-платформ обучения. В статье были рассмотрены как классические (традиционные) методы распределения нагрузки, так и более современные алгоритмы балансировки, которые учитывают множество параметров и способны подстраиваться под текущие условия. Был проведен анализ применения классов балансировочных алгоритмов в рамках LMS

и MOOK-платформ, показаны возможные направления развития балансировки в виде более плотной интеграции с CDN, облачной оркестрации и механизма интеллектуального кэширования.

Особое внимание было уделено построению масштабируемой и отказоустойчивой системы онлайн-обучения на базе серверного кластера «1С:Предприятие 8» с применением внешнего балансировщика. Представленное решение позволяет существенно повысить качество образовательного процесса, снизить риски отказов оборудования и сократить время отклика системы за счет перераспределения запросов на оптимальные серверы.

Список источников / References

1. Nagelli A., Yadav N. K. Efficiency unveiled: Comparative analysis of load balancing algorithms in cloud environments. *International Journal of Information Technology and Management*. 2024;18(2):142–147. DOI: 10.29070/pb64mp50.
2. Пономаренко Д. Н. Обзор алгоритмов балансировки нагрузки в облачных вычислительных системах. *Молодой ученый*. 2024;(12(511)):21–30. EDN: WGCGDY.
3. [Ponomarenko D. N. Review of load balancing algorithms in cloud computing systems. *Molodoy uchenyy*. 2024;(12(511)):21–30. (In Russian.) EDN: WGCGDY.]
3. Harjanti T. W., Setiyani H., Trianto J. Load balancing analysis using round-robin and least-connection algorithms for server service response time. *Applied Technology and Computing Science Journal*. 2022;5(2):40–49. DOI: 10.33086/atcsj.v5i2.3743.
4. Katangur A., Akkaladevi S., Vivekanandhan S. Priority weighted round robin algorithm for load balancing in the cloud. *2022 IEEE 7th Int. Conf. on Smart Cloud (SmartCloud)*. IEEE; 2022:230–235. DOI: 10.1109/SmartCloud55982.2022.00044.
5. Baihaqi M. R., Negara R. M., Tulloh R. Analysis of load balancing performance using round robin and IP hash algorithm on P4. *2022 5th Int. Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*. IEEE; 2022:93–98. DOI:10.1109/ISRITI56927.2022.10052975.
6. Kashani M. H., Mahdipour E. Load balancing algorithms in fog computing. *IEEE Transactions on Services Computing*. 2022;16(2):1505–1521. DOI: 10.1109/TSC.2022.3174475.
7. Malathi K., Anandan R., Vijay J. F. Cloud environment task scheduling optimization of modified genetic algorithm. *Journal of Internet Services and Information Security (JISIS)*. 2023;13(1):34–43. DOI: 10.58346/JISIS.2023.11.004.
8. Gad A. G. Particle swarm optimization algorithm and its applications: a systematic review. *Archives of computational methods in engineering*. 2022;29(5):2531–2561. DOI:10.1007/s11831-021-09694-4.
9. Zhang Y., Kong X. A particle swarm optimization algorithm with empirical balance strategy. *Chaos, Solitons & Fractals*. X. 2023;10(11):100089. DOI:10.1016/j.csf.2022.100089.
10. Abualigah L., Sheikhan A., Ikotun A. M., Zitar R. A., Alsoud A. R., Al-Shourbaji I., Hussien A. G., Jia H. Particle swarm optimization algorithm: Review and applications. Elsevier BV; 2024:1–14. DOI: 10.1016/b978-0-443-13925-3.00019-4.
11. Zheng H., Guo J., Zhou Q., Chen Y. Application of improved ant colony algorithm in load balancing of software-defined networks. *The Journal of Supercomputing*. 2023;79(7):7438–7460. DOI: 10.1007/s11227-022-04957-8.
12. Deng X., Zeng S., Chang le, Wang Y., Wu X., Liang J., Ou J., Fan C. An ant colony optimization-based routing algorithm for load balancing in LEO satellite networks. *Wireless Communications and Mobile Computing*. 2022;2022(1):3032997. DOI:10.1155/2022/3032997.
13. Gures E., Shaye A. I., Ergen M., Bin Azmi M., El-Saleh A. Machine learning-based load balancing algorithms in future heterogeneous networks: A survey. *IEEE Access*. 2022;10(2):37689–37717. DOI:10.1109/ACCESS.2022.3161511.
14. Taghizadeh M., Ahmadi M. A heuristic task scheduling algorithm for vehicular fog computing. *2024 8th Int. Conf. on Smart Cities, Internet of Things and Applications (SCIoT)*. IEEE; 2024:168–173. DOI:10.1109/SCIoT62588.2024.10570144.
15. Zeedan M., Attiya G., El-Fishawy N. Enhanced load balancing based on hybrid artificial bee colony with enhanced -hill climbing in cloud. *Mansoura Engineering Journal*. 2023;48(3):11. DOI:10.58491/2735-4202.3098.
16. Rahman F., Titouna C., Naït-Abdesselam F. Scalable blockchain system for load balancing DNS servers using heuristic sharding. *2023 Fifth Int. Conf. on Blockchain Computing and Applications (BCCA)*. IEEE; 2023:420–425. DOI:10.1109/BCCA58897.2023.10338848.
17. Singh I. T., Singh T. R., Sinam T. Server load balancing with round robin technique in SDN. *2022 Int. Conf. on Decision Aid Sciences and Applications (DASA)*. IEEE; 2022:503–505. DOI:10.1109/DASA54658.2022.9765287.
18. Zhang P., Li J., Tang Y., Wang H., Cao T. A novel, efficient, green and real-time load balancing algorithm for 5G network measurement report collecting clusters. *Journal of Circuits, Systems and Computers*. 2022;31(18):2250313. DOI:10.1142/S0218126622503133.
19. Papagianni C., Leivadreas A., Papavassiliou S. A cloud-oriented content delivery network paradigm: Modeling and assessment. *IEEE Transactions on Dependable and Secure Computing*. 2013;10(5):287–300. DOI: 10.1109/TDSC.2013.12.
20. Triukose S., Wen Zh., Rabinovich M. Measurement of the commercial content delivery network. *Proc. 20th Int. Conf. on World Wide Web (WWW'11)*. New York, NY, USA, Association for Computing Machinery; 2011:467–476. DOI: 10.1145/1963405.1963472.

Информация об авторах

Сущеня Роман Валерьевич, аспирант кафедры управления информацией в технических системах, факультет информационных технологий, Московский государственный технологический университет «Станкин», г. Москва, Россия; разработчик, фирма «1С», г. Москва, Россия; ORCID: <https://orcid.org/0009-0003-7946-9302>; e-mail: newromeones@gmail.com

Петров Валерий Евгеньевич, канд. тех. наук, доцент, доцент кафедры управления информацией в технических системах, факультет информационных технологий, Московский государственный технологический университет «Станкин», г. Москва, Россия; ORCID: <https://orcid.org/0009-0006-9112-8182>; e-mail: v.petrov@stankin.ru

Information about the authors

Roman V. Suschenya, a postgraduate student at the Department of Information Control in Technical Systems, Faculty of Information Technologies, Moscow State Technological University "STANKIN", Moscow, Russia; Software Developer, 1C Company, Moscow, Russia; ORCID: <https://orcid.org/0009-0003-7946-9302>; e-mail: newromeones@gmail.com

Valery E. Petrov, Candidate of Sciences (Engineering), Doct, Associate Professor at the Department of Information Control in Technical Systems, Faculty of Information Technologies, Moscow State Technological University "STANKIN", Moscow, Russia; ORCID: <https://orcid.org/0009-0006-9112-8182>; e-mail: v.petrov@stankin.ru

Поступила в редакцию / Received: 24.03.25.

Поступила после рецензирования / Revised: 07.04.25.

Принята к печати / Accepted: 08.04.25.