ЭВОЛЮЦИЯ МЕТОДОВ БАЛАНСИРОВКИ НАГРУЗКИ В РАСПРЕДЕЛЕННЫХ СИСТЕМАХ ОТ АППАРАТНЫХ РЕШЕНИЙ К АДАПТИВНЫМ АЛГОРИТМАМ

Косяков М.М, студент

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Марков А.Н. – старший преподаватель

Аннотация. В работе представлен теоретический анализ эволюции методов балансировки нагрузки в распределенных системах – от аппаратных устройств к программно-определяемым и адаптивным алгоритмам с элементами машинного обучения. Рассмотрены предпосылки появления аппаратных балансировщиков, их ограничения при росте масштабов систем, а также переход к программным решениям в условиях виртуализации и облачных вычислений. Отдельное внимание уделено адаптивным подходам и перспективам применения машинного обучения для прогнозирования нагрузки. Показано, что эволюция методов приводит не к последовательной смене подходов, а к формированию гибридных и многокритериальных механизмов, учитывающих противоречивость показателей эффективности и контекстно-зависимый выбор стратегий.

Балансировка нагрузки является одним из ключевых механизмов обеспечения устойчивости и эффективности функционирования распределенных систем. В условиях увеличения числа пользователей, роста объемов сетевого трафика и усложнения архитектуры сервисов задача рационального распределения запросов между множеством вычислительных узлов приобретает критическое значение. От качества реализации механизмов балансировки напрямую зависят показатели доступности, отказоустойчивости, времени отклика и общего уровня производительности системы. В высоконагруженных информационных системах даже незначительное перераспределение трафика может существенно повлиять на стабильность работы сервисов.

Первоначально задачи балансировки нагрузки решались с использованием специализированных аппаратных устройств. Такие решения устанавливались в структуре сетевой инфраструктуры и выполняли функцию централизованного распределения входящих запросов. Аппаратный балансировщик анализировал сетевые пакеты и перенаправлял их на один из серверов, входящих в пул обработки. Он также осуществлял регулярную проверку доступности серверов и автоматически исключал из обработки узлы, не отвечающие на контрольные запросы. Подобный подход широко применялся в корпоративных центрах обработки данных и крупных интернет-порталах.

Аппаратные балансировщики обеспечивали высокую производительность благодаря специализированной архитектуре и оптимизированной обработке сетевых соединений. Они могли выполнять завершение защищенных соединений, снижая нагрузку на серверы приложений, а также поддерживали механизмы закрепления пользовательской сессии за конкретным сервером. Однако при росте числа пользователей и расширении сервисов возникали существенные ограничения. Масштабирование требовало модернизации оборудования или внедрения дополнительных устройств, что сопровождалось значительными финансовыми затратами. Кроме того, централизованный характер архитектуры увеличивал риск возникновения единой точки отказа, несмотря на использование резервирования.

С переходом к виртуализированным средам и облачным технологиям балансировка нагрузки стала реализовываться преимущественно программными средствами. Программные балансировщики развертываются на стандартных серверах или в виртуальных машинах и могут масштабироваться вместе с инфраструктурой. Примерами таких решений являются Nginx и HAProxy, которые широко используются в веб-инфраструктурах различного масштаба – от небольших сервисов до глобальных платформ.

В программных балансировщиках реализуются различные алгоритмы распределения запросов, которые условно можно разделить на статические и динамические. К статическим относятся методы, не учитывающие текущее состояние системы. Наиболее простым из них является циклический алгоритм, при котором каждый новый запрос направляется на следующий сервер по очереди. Данный подход эффективен при однородной конфигурации узлов и относительно стабильной нагрузке.

Развитием статических методов стали динамические алгоритмы, учитывающие отдельные параметры функционирования серверов. К ним относятся алгоритмы с весовыми коэффициентами, отражающими относительную производительность узлов, а также методы, ориентированные на текущее количество активных соединений. Учет числа соединений позволяет частично учитывать распределение нагрузки в реальном времени и снижать вероятность перегрузки отдельных серверов.

С развитием микросервисной архитектуры структура распределенных систем стала значительно сложнее. Вместо единого приложения система включает множество взаимосвязанных сервисов, каждый из которых может масштабироваться независимо. В такой архитектуре балансировка нагрузки требуется не только на внешнем уровне, но и внутри самой системы.

Оркестрационные платформы, такие как Kubernetes, автоматизировали процесс масштабирования сервисов в зависимости от показателей загрузки. При увеличении нагрузки система

может создавать дополнительные экземпляры приложения, после чего механизм балансировки автоматически включает их в обработку трафика. Это обеспечивает динамическое расширение инфраструктуры без вмешательства администратора. В таких условиях балансировка становится тесно связанной с мониторингом состояния узлов и механизмами автоматического управления ресурсами.

Современный этап развития характеризуется появлением и распространением адаптивных алгоритмов, способным учитывать совокупность параметров состояния системы. Учет одного показателя не всегда позволяет адекватно оценить реальное состояние вычислительного узла. Помимо числа соединений анализируются загрузка процессора, использование памяти, сетевые задержки, частота ошибок и время отклика. На основе этих показателей формируется интегральная оценка состояния узла, позволяющая гибко регулировать объем направляемого на него трафика. Если один из серверов начинает демонстрировать увеличение времени отклика, система может временно снизить его долю в общем распределении, тем самым предотвращая перегрузку.

Однако переход к многопараметрической адаптации порождает фундаментальную проблему: отсутствие единого интегрального критерия эффективности. Адаптивный алгоритм вынужден решать задачу многокритериальной оптимизации, где требования минимизации времени отклика, максимизации утилизации ресурсов, обеспечения отказоустойчивости и энергоэффективности могут вступать в противоречие. Более того, в распределенных системах различной природы приоритеты меняются: для базы данных критична задержка, для потоковой обработки – пропускная способность, для edge-вычислений – энергопотребление узлов. Именно поэтому динамические и статические алгоритмы сохраняют свою область применения там, где предсказуемость и детерминизм важнее адаптивности. Ключевая тенденция современного этапа – не замена одних методов другими, а формирование гибридных схем, позволяющих переключаться между стратегиями в зависимости от текущего профиля нагрузки и заданных бизнес-приоритетов.

Дополнительным направлением развития является внедрение методов машинного обучения для прогнозирования нагрузки и повышения эффективности управления ресурсами. В распределенных системах часто наблюдаются повторяющиеся временные закономерности: увеличение активности в вечерние часы, в выходные дни или в период проведения рекламных кампаний. Анализ исторических данных позволяет построить модель, способную предсказывать будущие пики нагрузки. Это дает возможность заранее масштабировать инфраструктуру и корректировать параметры балансировки до возникновения критической ситуации. Таким образом, система переходит от реактивного к проактивному управлению.

Эволюция методов балансировки нагрузки выступает отражением более широкой трансформации распределенных вычислительных архитектур – от жестко централизованных аппаратных решений к гибким, программно управляемым и интеллектуально адаптирующимся механизмам. Балансировка нагрузки перестала быть исключительно сетевой функцией и стала неотъемлемым элементом комплексного управления распределенной вычислительной инфраструктурой.

Выводы. Проведенный теоретический анализ показал, что развитие методов балансировки нагрузки в распределенных системах обусловлено изменением архитектурных подходов и ростом требований к масштабируемости и устойчивости сервисов. Аппаратные решения заложили основу для построения высокопроизводительных инфраструктур, однако их ограничения способствовали переходу к программным средствам. Развитие облачных технологий и микросервисной архитектуры привело к интеграции балансировки с системами автоматического масштабирования и мониторинга. Ключевой тенденцией развития является не последовательная смена методов, а формирование гибридных и многокритериальных механизмов балансировки. Адаптивные алгоритмы требуют компромисса между противоречивыми показателями эффективности, что предполагает переход к многокритериальным моделям принятия решений. Перспективы дальнейших исследований связаны с разработкой методов динамического назначения весов критериям в гибридных балансировщиках и оценкой устойчивости таких систем при резких изменениях профиля нагрузки.

Список использованных источников:

1. Tanenbaum, A. S. Distributed Systems: Principles and Paradigms / A. S. Tanenbaum, M. Van Steen. – 2nd ed. – Upper Saddle River : Pearson Prentice Hall, 2007. – 686 p.

2. Harchol-Balter, M. Performance Modeling and Design of Computer Systems: Queueing Theory in Action / M. Harchol-Balter. – Cambridge : Cambridge University Press, 2013. – 576 p.

3. Cardellini, V. Dynamic Load Balancing on Web-Server Systems / V. Cardellini, M. Colajanni, P. S. Yu // IEEE Internet Computing. – 1999. – Vol. 3, No. 3. – P. 28–39.

4. Марков, А. Н. Проблема балансировки нагрузки облачных вычислений / А. Н. Марков // Компьютерные системы и сети: материалы 55-й юбилейной научной конференции аспирантов, магистрантов и студентов, Минск, 22-26 апреля 2019 г. – Минск : БГУИР, 2019. – С. 127–128.

5. Марков, А. Н. Анализ работы балансировщика нагрузки сервиса видео-конференц-связи / А. Н. Марков, А. И. Парамонов // Цифровая трансформация. – 2022. – Т. 28, № 2. – С. 43–51.

6. Документация Nginx [Электронный ресурс]. – Режим доступа: https://nginx.org. – Дата доступа: 02.04.2026.

7. Документация Kubernetes [Электронный ресурс]. – Режим доступа: https://kubernetes.io. – Дата доступа: 02.04.2026.