УДК 004.75

Яковлев Кирилл Андреевич, студент группы Б9122-09.03.04,
Дальневосточный федеральный университет, г. Владивосток
Березницкий Данил Сергеевич, студент группы Б9122-09.03.04,
Дальневосточный федеральный университет, г. Владивосток

СРАВНИТЕЛЬНЫЙ АНАЛИЗ АЛГОРИТМОВ БАЛАНСИРОВКИ НАГРУЗКИ В МНОГОПОЛЬЗОВАТЕЛЬСКИХ ОНЛАЙН-ИГРАХ

Аннотация

В статье рассматривается задача распределения нагрузки на серверы в многопользовательских онлайн-играх. Проведен разбор четырех алгоритмов балансировки нагрузки: Round-Robin, Least Connections, IP Hash и Weighted Round-Robin. Для каждого алгоритма описан принцип работы и показаны условия, при которых он дает хорошие результаты. Проведена серия вычислительных экспериментов на синтетических данных, которые имитируют поведение реальных игровых сессий. В роли основных метрик использованы среднее время отклика, стандартное отклонение нагрузки по серверам и доля потерянных соединений при пиковом трафике. Результаты показали, что алгоритм Least Connections в большинстве тестовых сценариев обеспечивает более равномерное распределение запросов, тогда как IP Hash выигрывает в задачах с хранением состояния сессии. Сделаны практические рекомендации по выбору алгоритма для разных типов игровых приложений.

Annotation

The article addresses the problem of load distribution across servers in multiplayer online games. Four load balancing algorithms are analyzed: Round-Robin, Least Connections, IP Hash and Weighted Round-Robin. The operating principles of each algorithm are described and conditions under which each performs well are shown. A series of computational experiments on synthetic data simulating the behavior of real gaming sessions is conducted. Average response time, standard deviation of server load and the rate of dropped connections under peak traffic are used as the main metrics. Results show that the Least Connections algorithm provides more even request distribution in most test scenarios while IP Hash performs better for stateful session applications. Practical recommendations for algorithm selection based on application type are provided.

Ключевые слова: балансировка нагрузки, алгоритмы распределения, многопользовательские игры, серверная архитектура, производительность

Keywords: load balancing, distribution algorithms, multiplayer games, server architecture, performance

Введение

Онлайн-игры за последние годы стали одной из самых быстрорастущих областей в индустрии программного обеспечения. При этом к серверной части игр предъявляются особые требования к задержке отклика: допустимый уровень задержки зависит от жанра и от требуемой точности действий игрока и для динамичных игр составляет порядка 100 миллисекунд, после чего результативность игрока и субъективное качество игрового опыта заметно падают [1].

Одна из главных технических проблем в этой области связана с тем, как равномерно распределить нагрузку между серверами при разном числе

активных игроков. В обычное время на серверах относительно мало соединений, но во время крупных игровых событий или релизов обновлений трафик может вырасти в несколько раз за считанные минуты. Если нагрузка распределена плохо, часть серверов уходит в перегрузку, а другая при этом простаивает. Это приводит к сбоям и отключениям пользователей.

Задача балансировки нагрузки решается с помощью специальных алгоритмов. Существует несколько классических подходов, каждый из которых имеет свои сильные и слабые стороны. Выбор конкретного алгоритма влияет на то, как система ведет себя при нормальной работе и при пиковых нагрузках. Тем не менее вопрос о том, какой алгоритм лучше всего подходит именно для игровых приложений с их спецификой, остается открытым.

Цель данной работы состоит в том, чтобы сравнить несколько широко используемых алгоритмов балансировки нагрузки применительно к сценариям, характерным для многопользовательских онлайн-игр. Для достижения этой цели были поставлены следующие задачи: описать принципы работы алгоритмов Round-Robin, Least Connections, IP Hash и Weighted Round-Robin; разработать модель нагрузки, имитирующую поведение игроков; провести вычислительные эксперименты и сравнить алгоритмы по выбранным метрикам производительности.

1. Обзор алгоритмов балансировки нагрузки

Балансировка нагрузки в распределенных системах изучается уже несколько десятилетий. Классические алгоритмы распределения запросов между узлами кластера — Round-Robin, Least Connections, IP Hash и Weighted Round-Robin — систематизированы в литературе по серверной балансировке, где описаны принципы их работы, способы реализации в

программных и аппаратных балансировщиках и типовые сценарии применения [2].

Round-Robin является самым простым из рассматриваемых алгоритмов. Входящие запросы распределяются по серверам по кругу в фиксированной очередности. Каждый сервер получает ровно одинаковое число запросов. Главный плюс этого подхода состоит в простоте реализации и отсутствии накладных расходов на сбор информации о текущем состоянии серверов. Минус в том, что алгоритм не учитывает реальную нагрузку: если один из запросов потребует долгой обработки, сервер, его получивший, окажется перегружен, пока остальные простаивают.

Least Connections принимает во внимание текущее число активных соединений на каждом сервере. Новый запрос всегда отправляется на тот сервер, у которого в данный момент меньше всего открытых соединений. Это позволяет выровнять нагрузку в тех случаях, когда запросы имеют разное время обработки. Для игровых приложений это особенно важно, потому что игровые сессии могут длиться от нескольких минут до нескольких часов.

IP Hash вычисляет хеш от IP-адреса клиента и на основании этого значения закрепляет клиента за конкретным сервером. Пока IP-адрес не меняется, все запросы от одного пользователя попадают на один и тот же сервер. Это свойство называется сессионной привязкой и крайне полезно для игровых приложений, которые хранят состояние игры на стороне сервера. Недостаток подхода состоит в том, что при неравномерном распределении IP-адресов нагрузка может сосредоточиться на нескольких серверах.

Weighted Round-Robin является расширением базового алгоритма Round-Robin. Каждому серверу присваивается числовой вес, который отражает его вычислительную мощность. Серверы с большим весом

получают больше запросов пропорционально своим возможностям. Это полезно в гетерогенных кластерах, где серверы имеют разные характеристики. Сложность состоит в корректном назначении весов и их своевременном обновлении при изменении конфигурации кластера.

2. Особенности нагрузки в онлайн-играх

Серверная нагрузка в многопользовательских играх отличается от типичной веб-нагрузки по нескольким параметрам. Во-первых, игровые приложения, как правило, сохраняют состояние (stateful): сервер хранит информацию о позиции игрока, его инвентаре и состоянии игрового мира. Это накладывает ограничения на возможность свободного перераспределения запросов между серверами в ходе сессии.

Во-вторых, игровые сессии имеют длительный характер. Типичное веб-соединение длится секунды или доли секунды, тогда как игровая сессия может продолжаться часами. За это время активность игрока меняется: в бою он генерирует много коротких пакетов, а при изучении меню активность снижается. Алгоритм балансировки должен справляться с такими изменениями интенсивности внутри одного соединения. Измерения трафика реальных игровых серверов подтверждают эту специфику: поток образован преимущественно короткими пакетами, отправляемыми с высокой периодичностью, а длительность пользовательских сессий распределена крайне неравномерно [3].

В-третьих, в онлайн-играх наблюдаются ярко выраженные пиковые нагрузки. Они возникают в момент выхода обновлений, проведения внутриигровых событий и в вечерние часы, когда большинство пользователей свободны. В такие периоды число подключений может вырасти в 5-10 раз по сравнению со средними значениями. Система

балансировки должна справляться с этими пиками без заметного роста задержки.

Требования к времени отклика в играх жестче, чем в большинстве других сервисов. Задержка свыше 100 миллисекунд начинает ощущаться игроком как дискомфорт, а при значениях свыше 200 миллисекунд нормальный игровой процесс становится невозможным. Это требование фактически запрещает балансировщику перенаправлять запросы на серверы, расположенные географически далеко от клиента.

3. Методика проведения эксперимента

Для сравнительного анализа алгоритмов была разработана программная модель на языке Python. Модель включала три компонента: генератор нагрузки, набор виртуальных серверов и реализации четырех алгоритмов балансировки.

Генератор нагрузки создавал пуассоновский поток запросов. Такое распределение хорошо описывает потоки событий, в которых моменты поступления запросов статистически независимы друг от друга; пуассоновский входной поток при нескольких параллельных обслуживающих узлах является стандартной моделью и в аналитических исследованиях алгоритмов балансировки [4]. Время обработки каждого запроса задавалось случайным образом из равномерного распределения на интервале от 0.5 до 5 секунд, что моделировало разницу между короткими пакетами состояния и длинными игровыми сессиями.

Каждый виртуальный сервер характеризовался вместимостью: максимальным числом соединений, которые он способен обслуживать одновременно. При превышении этого порога новые запросы считались потерянными. Всего в модели использовалось пять серверов с одинаковой вместимостью в 100 соединений. Следует отметить, что при однородных

серверах алгоритм Weighted Round-Robin вырождается в базовый Round-Robin, поскольку различие весов не проявляется; его оценка в гетерогенном кластере выходит за рамки настоящей работы и требует отдельного исследования.

Тестирование проводилось в трех сценариях. Первый сценарий имитировал равномерную нагрузку с интенсивностью 50 запросов в секунду, что составляет около 50 процентов от максимальной пропускной способности системы. Второй сценарий моделировал пиковую нагрузку с интенсивностью 120 запросов в секунду, превышающей расчетную пропускную способность. Третий сценарий воспроизводил нарастающую нагрузку с постепенным ростом от 20 до 150 запросов в секунду на протяжении 10 минут модельного времени. Каждый сценарий воспроизводился 30 раз с различными начальными значениями генератора псевдослучайных чисел; далее приводятся усредненные по прогонам значения метрик.

В качестве метрик сравнения были выбраны три показателя. Первый показатель представлял собой среднее время отклика на запрос. Второй показатель отражал стандартное отклонение числа активных соединений по серверам в каждый момент времени и характеризовал равномерность распределения нагрузки. Третий показатель фиксировал долю запросов, потерянных из-за перегрузки серверов.

4. Результаты и анализ

В сценарии с равномерной нагрузкой все четыре алгоритма показали близкие результаты по среднему времени отклика. Разница между лучшим и худшим результатом составила около 8 процентов. Наиболее равномерное распределение нагрузки между серверами обеспечил алгоритм Least Connections: стандартное отклонение числа соединений по серверам

оказалось примерно в 1.4 раза ниже, чем у Round-Robin. IP Hash в этом сценарии показал заметный дисбаланс, потому что хеш-функция при небольшом числе запросов распределяла клиентов неравномерно.

При пиковой нагрузке различия между алгоритмами стали более выраженными. Round-Robin потерял около 18 процентов запросов, потому что в случае перегрузки одного из серверов продолжал направлять к нему новые запросы наравне с остальными. Least Connections сократил потери до 11 процентов за счет того, что своевременно переключал новые запросы на менее нагруженные серверы. Weighted Round-Robin дал результаты, близкие к базовому Round-Robin, поскольку серверы в модели имели одинаковые характеристики и разница в весах не применялась.

IP Hash в пиковом сценарии показал самые высокие потери соединений среди всех алгоритмов: около 22 процентов. Это связано с тем, что при жесткой привязке клиентов к серверам перераспределить нагрузку в момент пика практически невозможно. Серверы, к которым оказалось привязано много клиентов, уходили в перегрузку, не имея возможности разгрузиться за счет соседних серверов.

В сценарии с нарастающей нагрузкой алгоритм Least Connections сохранял стабильное распределение вплоть до момента, когда суммарная нагрузка превышала 80 процентов от пропускной способности системы. После этого порога поведение всех алгоритмов ухудшалось примерно в одинаковой мере. Это говорит о том, что при сильной перегрузке выбор алгоритма отходит на второй план, а определяющим фактором становится общая вычислительная мощность кластера.

Подводя итог сравнению, можно сказать, что для игровых приложений без жесткого требования к сессионной привязке наилучшим выбором является алгоритм Least Connections. Он обеспечивает хорошую равномерность распределения нагрузки и устойчивость при пиковых

нагрузках. Для игр, в которых сервер хранит состояние сессии и переключение между серверами недопустимо, предпочтительным остается IP Hash, несмотря на его слабость при перегрузках. В гетерогенных кластерах с серверами разной мощности стоит рассмотреть Weighted Round-Robin с правильно настроенными весами.

Заключение

В данной работе был проведен сравнительный анализ четырех алгоритмов балансировки нагрузки применительно к условиям эксплуатации многопользовательских онлайн-игр. На основании вычислительных экспериментов с синтетическими данными установлено, что алгоритм Least Connections обеспечивает наиболее равномерное распределение нагрузки при нормальных и пиковых условиях работы. Алгоритм IP Hash уступает другим по эффективности распределения, но остается незаменимым в приложениях, требующих сессионной привязки.

Результаты показывают, что выбор алгоритма балансировки оказывает заметное влияние на качество обслуживания пользователей при пиковой нагрузке. Разница в доле потерянных запросов между алгоритмами может достигать 11 процентных пунктов при нагрузке, превышающей расчетную мощность системы.

Дальнейшее развитие данного направления может быть связано с применением методов машинного обучения для адаптивной балансировки нагрузки. Такие системы способны предсказывать пиковые нагрузки на основе истории трафика и заблаговременно перераспределять ресурсы; в обзорах современных методов балансировки динамические и адаптивные алгоритмы выделяются как основное направление развития по сравнению со статическими схемами [5]. Кроме того, перспективным направлением

является совместное использование нескольких алгоритмов с динамическим переключением между ними в зависимости от текущего состояния системы.

Литература

1. Claypool M., Claypool K. Latency and Player Actions in Online Games // Communications of the ACM. 2006. Vol. 49. No. 11. P. 40–45. DOI: 10.1145/1167838.1167860.
2. Bourke T. Server Load Balancing. Sebastopol: O'Reilly Media, 2001. 192 p.
3. Feng W.-C., Chang F., Feng W.-C., Walpole J. A Traffic Characterization of Popular On-Line Games // IEEE/ACM Transactions on Networking. 2005. Vol. 13. No. 3. P. 488–500. DOI: 10.1109/TNET.2005.850221.
4. Mitzenmacher M. The Power of Two Choices in Randomized Load Balancing // IEEE Transactions on Parallel and Distributed Systems. 2001. Vol. 12. No. 10. P. 1094–1104. DOI: 10.1109/71.963420.
5. Shafiq D.A., Jhanjhi N.Z., Abdullah A. Load Balancing Techniques in Cloud Computing Environment: A Review // Journal of King Saud University – Computer and Information Sciences. 2022. Vol. 34. No. 7. P. 3910–3933. DOI: 10.1016/j.jksuci.2021.02.007.

Literature

1. Claypool M., Claypool K. Latency and Player Actions in Online Games // Communications of the ACM. 2006. Vol. 49. No. 11. P. 40–45. DOI: 10.1145/1167838.1167860.

2. Bourke T. Server Load Balancing. Sebastopol: O'Reilly Media, 2001. 192 p.
3. Feng W.-C., Chang F., Feng W.-C., Walpole J. A Traffic Characterization of Popular On-Line Games // IEEE/ACM Transactions on Networking. 2005. Vol. 13. No. 3. P. 488–500. DOI: 10.1109/TNET.2005.850221.
4. Mitzenmacher M. The Power of Two Choices in Randomized Load Balancing // IEEE Transactions on Parallel and Distributed Systems. 2001. Vol. 12. No. 10. P. 1094–1104. DOI: 10.1109/71.963420.
5. Shafiq D.A., Jhanjhi N.Z., Abdullah A. Load Balancing Techniques in Cloud Computing Environment: A Review // Journal of King Saud University – Computer and Information Sciences. 2022. Vol. 34. No. 7. P. 3910–3933. DOI: 10.1016/j.jksuci.2021.02.007.