ИУ7-54Б, 16_KOZ, Козлитин

1 Аналитическая часть

1.1 Weighted Response Time

Weighted Response Time - алгоритм, при котором время ответа серверов определяет, какому серверу будет направлен следующий запрос. Сервер, отвечающий на запрос быстрее всех, получает следующий запрос. [1]

Пусть имеется N запросов и M узлов. Алгоритм состоит из следующих шагов.

1. Сформировать массив $nodes$, содержащий узлы;
2. Для каждого запроса из N:
   - Установить переменную $target$ на первый доступный узел - $nodes[0]$.
   - Создать переменную $i = 1$.
   - Пока $i < M$:
     - если время ответа узла $target$ меньше времени ответа узла $nodes[i]$, то установить $target$ на текущий узел - $nodes[i]$.
     - увеличить значение i;
   - Отправить запрос на узел $target$.

1.2 Random 2 (N) choices

Random 2 (N) choices - алгоритм, при котором определяется нагрузка 2 (N) серверов, выбранных случайным образом, и распределяется на наименее загруженный из них. В случае N=2 максимальная нагрузка на n серверов с высокой вероятностью составит $\Theta(\log \log n)$.[2]

Данный метод может быть использован, когда запрос требуется отправить на наименее загруженный сервер. Однако, полная информация о загрузке всех серверов может оказаться дорогостоящей для получения. Например, для получения загрузки на сервер может потребоваться отправка сообщения и ожидание ответа, обработка прерывания сервером. [2]

Альтернативный подход при котором информация о загрузке серверов не требуется, заключается в том, чтобы распределить запрос на слуайный сервер.

В таком случае максимальная нагрузка на n серверов с высокой вероятностью составит $\Theta(\log n / \log \log n)$.[2]

Пусть имеется K запросов и M узлов, при этом $2 <= N <= M$. Алгоритм состоит из следующих шагов.

1. Сформировать массив *nodes*, содержащий узлы;
2. Для каждого запроса из K:
   - Сфомировать массив *randoms*, содержащий N узлов, выбранных случайным образом из массива *nodes*;
   - Установить переменную *target* на первый доступный узел - *randoms*[0].
   - Создать переменную $i = 1$.
   - Пока $i < M$:
     - если нагрузка узла *randoms*[i] меньше нагрузки узла *target*, то установить *target* на узел *randoms*[i].
     - увеличить значение i;
   - Отправить запрос на узел *target*.

1.3 Resource based algorithm

Resource based algorithm - алгоритм, при котором трафик распределяется балансировщиком нагрузки, в зависимости от текущей нагрузки на сервер. [3]

Специализированное программное обеспечение, называемое агентом, запускается на каждом сервере и рассчитывает использование ресурсов сервера, таких как его вычислительная мощность и память. Затем агент проверяется балансировщиком нагрузки на наличие достаточного количества свободных ресурсов перед распределением трафика на данный сервер. [3]

Пусть имеется N запросов и M узлов. Алгоритм состоит из следующих шагов.

1. Сформировать массив *nodes*, содержащий узлы;
2. Для каждого запроса из N:

- Сформировать массив $resources$, содержащий информацию об использовании ресурсов, соответствующим узлом;
- Установить переменную $target$ на первый доступный узел - $resources[0]$.
- Создать переменную $i = 0$.
- Пока $i < M$:
  - Узел $resources[i]$ обладает достаточным количеством свободных ресурсов для выполнения запроса:
    - установить $target$ на узел $resources[i]$;
    - прекратить выполнение цикла;
  - увеличить значение i;
- Отправить запрос на узел $target$.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Данешманд Б.*, *Ту Л. А.* ИССЛЕДОВАНИЕ И ОБЗОР МЕХАНИЗМОВ БАЛАНСИРОВКИ НАГРУЗКИ НА ОСНОВЕ SDN В 5G/IMT-2020 // Вестник Воронежского государственного технического университета. — 2022. — Т. 18, № 1.
2. The Power of Two Random Choices: A Survey of Techniques and Results. — Режим доступа: `https://www.eecs.harvard.edu/~michaelm/postscripts/handbook2001.pdf` (дата обращения: 07.10.2023).
3. What Is Load Balancing? — Режим доступа: `https://aws.amazon.com/what-is/load-balancing/` (дата обращения: 07.10.2023).