Повышение производительности алгоритмов распределенного кэширования

Илья Усов
научный руководитель С.Ю. Сартасов

31 мая 2017 г.

Цель работы

- Сделать обзор существующих решений
- Реализовать свой алгоритм в рамках библиотеки с открытым исходным кодом Redis
- Сравнить с существующими решениями

Существующие решения

- Redis
- Memcached
- Hazelcast

Redis

- Ведущие и ведомые узлы
- Ключи распределены между ведущими узлами
- Решение о вытеснении принимается на основе LRU (или LFU с версии 4.0)
- Используется аппроксимированный LRU (или LFU с версии 4.0)

Redis

Существуют следующие правила при переполнении кэша:

- noeviction
- allkeys-lru
- volatile-lru
- allkeys-lfu
- volatile-lfu
- allkeys-random
- volatile-random
- volatile-ttl

Недостатки

- Решение о вытеснении принимается без учета статистики на ведомых устройствах

Первый подход

- Ведущий отправляет своим подчиненным свой пул кандидатов на выселение
- Ведомые обновляют полученный пул и отправляют обратно
- Ведомые отправляет свой пул ведущему
- Ведущий на основе полученных данных обновляет пул

Второй подход

- Ведомые отправляют свои списки кандидатов на выселение ведущему
- Ведущий сохраняет полученные данные в буфер
- Пул обновляется по мере необходимости кандидатами из буфера

Особенности реализации

- Реализация написана на языке C
- Передача данных осуществляется с помощью Redis API
- Буфер при заполнении начинает заполняться сначала
- Буфер содержит копии ключей
- Если буфер пуст берутся случайные ключи

Апробация

Алгоритм	70/20	75/25	80/35
Redis LRU	69.39%	69.93%	62.12%
Первый	70.47%	70.13%	61.75%
Второй	69.79%	69.85%	61.22%

Таблица: Точность различных алгоритмов кэширования на основе LRU.

Алгоритм	70/20	75/25	80/35
Redis LFU	71.24%	68.92%	60.78%
Первый	69.25%	70.08%	61.38%
Второй	71.30%	69.31%	61.34%

Таблица: Точность различных алгоритмов кэширования на основе LFU.

Апробация

Алгоритм	Ведущий	Ведомый 1	Ведомый 2
Redis LRU	14191	13016	12988
Первый	12616	10975	11016
Второй	14025	12825	12958

Таблица: Количество запросов в секунду к узлу при различных алгоритмах кэширования

Результаты

- Сделан обзор существующих решений
- Реализован алгоритм вытеснения с учетом статистики ведомых узлов в рамках библиотеки с открытым исходным кодом Redis
- Поддержаны следующие политики вытеснения:
  - allkeys-lru
  - volatile-lru
  - allkeys-lfu
  - volatile-lfu
- Сравнена эффективность алгоритмов на различных данных