Кеширането днес е неразделна част от всеки уеб проект.

За да генерираме една страница от доста сложен ресурс, може да се наложи да направим десетки обаждания, от които се формира отговорът към потребителя. Много от тях ще бъдат бързи, но винаги има малък брой заявки, чието изчисление може да бъде за секунди или минути. Ако сумираме цялото време, което прекарваме в очакване на резултатите от заявките, получаваме незадоволително време за отговор.

Решението на този проблем е кеширането: поставяме резултата от изчисленията в някакво хранилище (например memcached), което има отлични характеристики на времето за достъп до информация. Сега, вместо да осъществяваме достъп до бавните, сложни и тежки backend, просто трябва да изпълним заявката към бърз кеш.

Memcached

Memcached беше реализиран от Брад Фицпатрик като част от проекта на LiveJournal.

Това е огромна хеш-таблица в RAM, достъпна чрез мрежовия протокол. Memcached предоставя услуга за съхраняване на стойности, свързани с ключове. Клиент може да бъде програма, написана на произволен език за програмиране (C/C ++, PHP, Perl, Java и други).

Най-простите операции:

1. Взимане на стойността на посочения ключ (get)
2. Задаване на стойността на ключа (set)
3. Изтриване на ключ (del)

Архитектура

Memcached е проектиран така, че всичките му операции имат алгоритмична сложност O(1), т.е. времето за изпълнение на операция не зависи от броя на запаметените ключове. Това означава, че някои операции ще отсъстват в него, ако тяхното изпълнение изисква само линейно време. В memcached няма възможност за групиране на ключове и няма групови операции върху ключове (или техните стойности).

Основните оптимизирани операции са избиране/освобождаване на блокове с памет за съхранение на ключове и определяне на политиката на най-неизползваните ключове (LRU) за изчистване на кеша, когато няма достатъчно памет. Търсенето на ключове става чрез хеширане и следователно има сложност O(1).

Всъщност можем да кажем, че времето за реакция на memcached се определя само от мрежовите разходи и е почти равно на времето, необходимо за изпращане на пакета от frontend на сървъра с memcached(RTT). Такива характеристики позволяват да се използва memcached във високо заредени уеб проекти за решаване на различни проблеми, включително кеширане на данни.

Особености на имплементацията

- ❖ Кластеризация memcached

Вместо единичен запомнен сървър се използва клъстер от такива сървъри за разпределяне на натоварването и постигане на отказоустойчивост. Сървърите, включени в клъстера, могат да бъдат конфигурирани с различни количества памет, а общият размер на кеша ще бъде равен на сумата от размерите на кеша на всички memcached на клъстера. Когато работите с клъстер, ключовете се разпределят между сървърите, тоест всеки сървър обработва част от общия масив ключове на проекта. В случай на отказ на един от сървърите, ключовете ще бъдат преразпределени към останалите сървъри на клъстера.

- ❖ Статистика на работа memcached

Необходимо е постоянно да наблюдаваме клъстера от memcached сървъри, за да сме сигурни, че сме постигнали оптимална производителност.

Най-простата команда stats ви позволява да получите основни статистически данни: време за работа на сървъра (uptime), количество използвана памет, брой get заявки и брой хитове (hits). Тяхното съотношение ни позволява да преценим ефективността на кеширането като цяло.

Употреба на технологията

В момента Memcached е достъпен само на Unix/Linux платформата.

Наистина ли memcached е по-ефективен за видео и аудио, за разлика от текстовите данни?

Memcached работи еднакво добре за всички видове данни. Всяка стойност, която съхранявате, е само поток от данни. Максималният размер на обект, който можете да съхранявате в памет, е 1 MB. Ако планирате да използвате memcached с аудио и видео съдържание, вероятно ще искате да увеличите максималния размер на обекта.

Не забравяйте също, че memcached се използва за кеширане на информация за четене. Той не трябва да се използва за писане, освен ако информацията за кеша не е актуализирана.