Разработка высоконагруженных и надежных систем

Андрей Смирнов, 2015

Практическое задание №6

Эффект от использования “обычного” и exponential backoff на стороне клиента.

Сценарий: обычная работа ⇛ сервер “упал” ⇛ сервер поднялся.

1. Сервер

- Отвечает на запросы за `minDelay` (100 мс), пока количество одновременных запросов меньше `concurrencyLimit` (30)

- После превышения порога время ответа: $\texttt{minDelay} * \texttt{factor}^{\texttt{(concurrency-concurrencyLimit)/K}}$

2. Клиент

- N потоков, каждый из которых по экспоненциальному распределению с параметром `lambda` делает запрос

- В случае ошибки либо повтор через delay (простое повторение), либо exponential backoff

```
$ docker run -t -i smira/hl-tasks:exponential
root@0aa60f85d73f:/home/hl-tasks# ./server
Apr 21 14:16:10.233: concurrency:   58, last delay: 114.638379ms
Apr 21 14:16:11.233: concurrency:   51, last delay: 111.331089ms
Apr 21 14:16:12.234: concurrency:   54, last delay: 111.331089ms
^Ac
root@0aa60f85d73f:/home/hl-tasks# ./client -exponential-backoff
OK: 395.60 req/sec, errors: 0.00 req/sec, timedout: 0.00 req/sec
OK: 382.80 req/sec, errors: 0.00 req/sec, timedout: 0.00 req/sec
OK: 378.20 req/sec, errors: 0.00 req/sec, timedout: 0.00 req/sec
root@0aa60f85d73f:/home/hl-tasks#
^A0
^Z
root@0aa60f85d73f:/home/hl-tasks# fg
…
```

Сценарий

- Сервер запущен (нет нагрузки)
- Запущен клиент (стабильная нагрузка)
- Сервер остановлен (^Z)
- Клиент обнаруживает отказ, повторяет запросы
- Сервер возобновил работу (fg)
- Восстанавливается стабильная нагрузка ИЛИ сервер “падает”

Как отличается поведение при simple и exponential backoff стратегиях?

Как зависит результат от параметров `lambda`, `clients`?