Сгладить углы

Задача изучения применения очередей как способ справляться с пиковой нагрузкой и не терять SLA(а значит и клиентов 🙂) в случае недоступности.

Навыки:

- Performance Testing (k6)
- Go (Rate Limit, Queue Clients, Worker Pool, Metrics)
- Monitoring (Promtheus/Grafana)
- Queue (Nats)

Уровень: Middle

Материалы:

https://github.com/veggiemonk/compose-nats

https://github.com/grafana/k6

https://go.dev/tour/

https://prometheus.io/docs/prometheus/latest/querying/basics/

https://gobyexample.com/worker-pools

Часть 1. Пиковая нагрузка.

1. Сервис

Написать Golang сервис, принимающий некоторые запросы заказов для маркетплейса.

```
{
   "product_id": "33",   // От 0 до 100
   "count": "3", // От 0 до 100
   "username": "maksim.konovalov"
}
```

Веб сервер должен принимать заказы и отдавать, что заказ принят в формате JSON

HTTP 200

```
{
  "result": "created"
}
```

Или если product_id и count не валидны отдавать

HTTP 400

```
{
  "result": "invalid"
}
```

Если RPS превышает 50(если вы не понимаете как это сделать - загуглите реализацию рейт лимитов, по сути вы делаете тоже самое просто со статусом падения, а не отбиваете запросы):

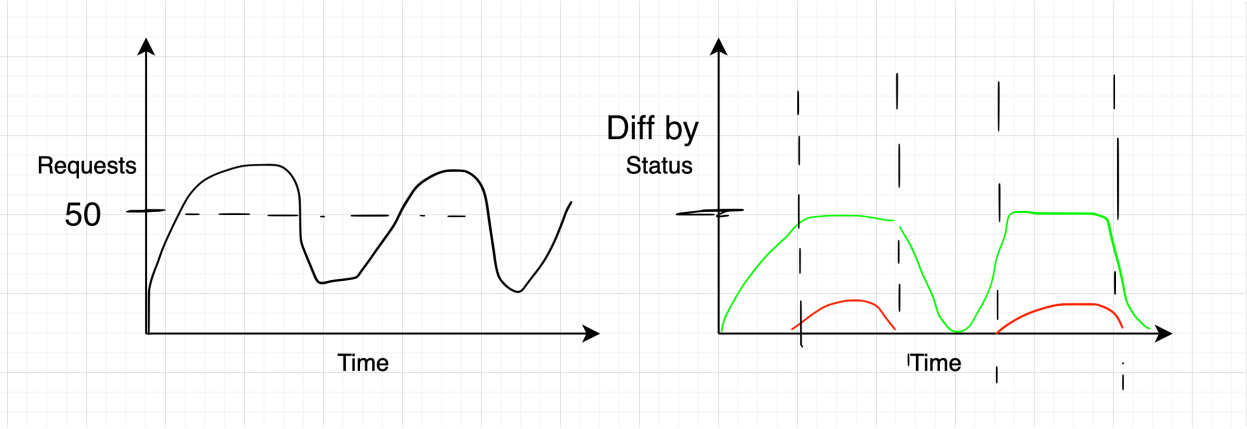
HTTP 500

```
{
   "result": "server-kaput"
```

```
}
```

2. Покрываем метриками.

Нам необходимо вывести графики нагрузки нашего сервиса. На графиках мы должны наблюдать число статусов ответа. В конечном результате требуется получить следующий график:



3. Сделайте counter метрику Work Done с лейблом status (done/error), которая будет пополняться с каждым успешным запросм. В данном случае она должна совпадать с графиком Diff By Status.
3. Создать нагрузку аналогичную что показана на графиках с помощью k6

Загрузка должна быть синусоидного вида от 0 до 100 по RPS. В моментах когда нагрузка пробивает 50 RPS мы эмулируем критические точки для нашего маркетпелйса, когда заказы активно покупают. Моменты просадки RPS - ночь, когда наше железо простаивает.

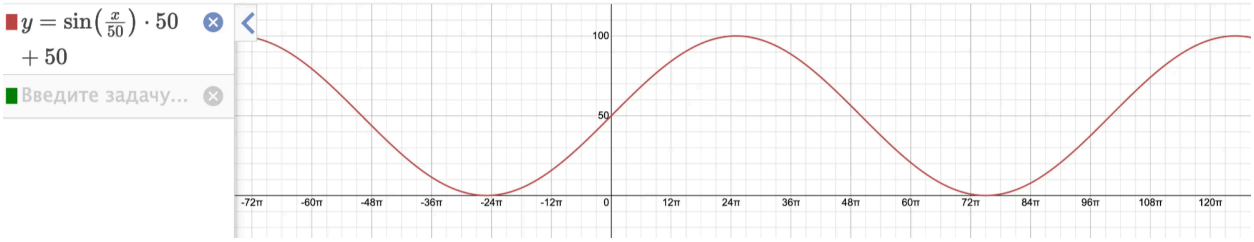
Почему именно k6?

https://www.youtube.com/watch?v=iu6X4QNgFig

С чего начать его писать?

https://grafana.com/docs/k6/latest/get-started/

Какую функцию нужно реализовать в нагрузке?

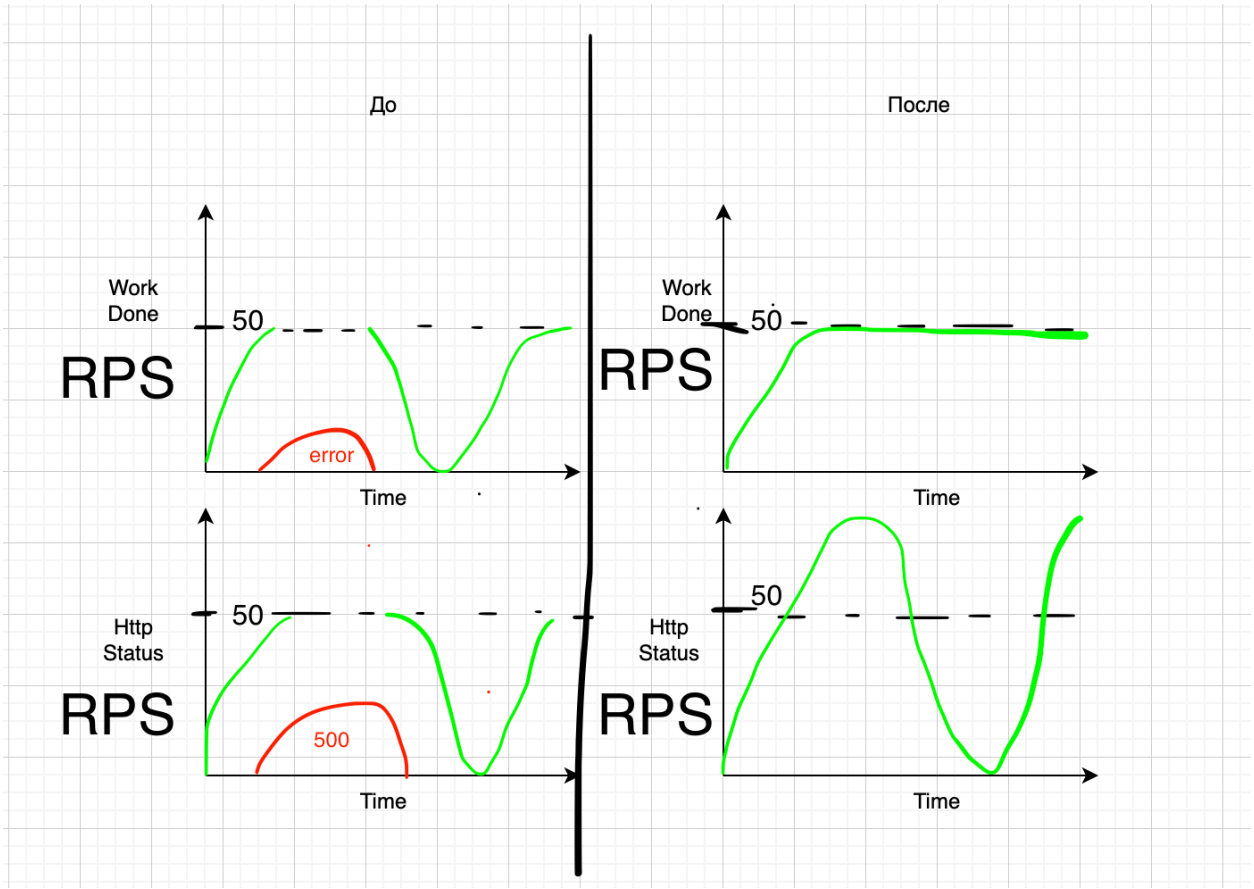


Только в данном случае вместо Pi мы используем время 🙂, подумайте как использовать его вместо X, кажется это просто сделать?

Часть 2. Сгладить углы

Давайте модифицируем наш сервер таким образом, чтобы мы не отбивали наш излишний трафик, а сохраняли устойчивость в случае резки скачков.

Наша задача добиться того чтобы мы не отбивали излишние запрос и график статусов пришел к следующему виду



Предположим, что наш сервер обработает обращения к нашему маркетплейсу ночью, вместо того чтобы терять заказы от пользователей.

Давайте теперь вместо того чтобы сразу обрабатывать наш заказ и делать некоторую работу отправлять наши задачи в очередь Nats, а забирать оттуда задачи, которые мы заводим.

Чтобы поднять Nats с готовыми графиками в Grafana воспользуйтесь:

https://github.com/veggiemonk/compose-nats

Для обработки задач - парралельно с помощью горутин воспользуйтесь механиков Worker Pool, чтобы в фоне разгребать наши задачи и делать некоторую обработку.

- Изменилось ли latency ответа http сервера?
- Можем ли мы гарантировать пользователю обработку его заказа? Почему? Что может пойти не так?
- Какие еще метрики можно собрать с нашей очереди и сервиса?