Лекция 6

Node.js

Node.js как платформа

Что такое Node.js?

Node.js - среда выполнения javascript основанная на браузерном движке Google V8.

На что Node.js способен

- веб-приложения (бэкенд);
- десктопные приложения (Electron.js);
- микроконтроллеры (Espurino), ссылка;
- различные консольные утилиты;
- машинное обучение (TensorFlow.js), ссылка;
- бла-бла-бла…

На что Node.js способен

- **веб-приложения (бэкенд);**
- десктопные приложения (Electron.js);
- микроконтроллеры (Espurino), ссылка;
- различные консольные утилиты;
- машинное обучение (TensorFlow.js), ссылка;
- бла-бла-бла…


ЛИЧНО Я
Backend Node.js Developer

Преимущества Node.js

Скорость

Node.js - это платформа для разработки очень быстрого сервера. По производительности спокойно может сравниться с Java, Kotlin и Golang.

При этом все вышеперечисленные языки имею вложенные механизмы параллелизма (корутины, потоки и тд), а Node.js однопоточный.

Как так получается?

Асинхронное API

Движок V8 из коробки поддерживает исключительно асинхронное API, т.к. изначально был предназначен для работы в браузере на клиенте.
Это значит, что ни одна из read/write операций не блокирует основной поток исполнения (спасибо Event Loop).

Простота

Поднять простой веб-сервер на Node.js - это 15 строк кода (с натяжкой).

При этом, т.к. мы пишем на javascript, мы можем использовать различные парадигмы для написания кода.

Библиотеки

У Node.js огромное комьюнити, и с 2009 года существуют библиотеки для всего, что может теоретически понадобиться для серверной (и не только) разработки.

Количество javascript библиотек достигает такого огромного количества, что это уже даже достигло какого-то абсурдного уровня.

is-thirteen

2.0.0 • Public • Published 7 years ago

is-thirteen

Check if a number is equal to 13.

Installation

```
npm --save i is-thirteen
```

Usage

```
var isThirteen = require('is-thirteen');
```

Install

```
> npm i is-thirteen
```

Repository
github.com/jezen/is-thirteen

Homepage
github.com/jezen/is-thirteen#readme

Weekly Downloads
36

Version
2.0.0

License
WTFPL

JavaScript

Тут не нужно объяснений

Недостатки Node.js

16:54

У тебя было яблоко, тебе дали ещё одно

Сколько у тебя яблок?

16:55

Одно яблоко

Потому что яблоко + одно = одно яблоко

16:55

Твой родной язык это джаваскрипт что ли с[redacted]а ??

JavaScript

RunTime-ошибки

Это когда код запустился, но упал во время исполнения.

Такая возможность есть из-за того, что javascript интерпретируемый язык программирования.

```
1 const a = undefined;
2 console.log('шалом ');
3
4 a.func(); // RunTime Exception
5
6 console.log('братья');
7
```

Отсутствие типизации

Javascript - нестрого типизированный язык.

```
1 true + false
2 12 / "6"
3 "number" + 15 + 3
4 15 + 3 + "number"
5 [1] > null
6 "foo" + + "bar"
7 'true' == true
8 false == 'false'
9 null == ''
10 !!"false" == !!"true"
11 [‘x’] == ‘x’
12 [] + null + 1
13 0 || "0" && {}
14 [1,2,3] == [1,2,3]
15 {}+[]+{}+[1]
16 !+[]+[]+![]
17 new Date(0) - 0
18 new Date(0) + 0
19
```

Риск наговнокодить

Риск написать плохой код на JavaScript настолько высок, что сравнить его можно, наверно, только с Python. Обилие поддерживаемых парадигм, нестрогая типизация, сложная асинхронность, лексическое окружение, замыкания.

```
async registration(Form) {
    console.log('ajax post');
    return await ajax.post(backend.upload, new FormData(Form), true)
        .then(({status, responseObject}) => {
            let photo_name;
            if (status === 200 ) {
                photo_name = new Promise((resolve, reject) => {
                    resolve(JSON.stringify(responseObject));
                    this.linkImage.push(responseObject.replaceAll('"', ''));
                });
                console.log(photo_name);
                return ajax.post(backend.signup, this.Json());
            }

            if (status === 400) {
                throw new Error('Слишком большой размер фото, пожалуйста, загрузите фото меньшего размера');
            }

            if (status === 403) {
                throw new Error('Пожалуйста, загрузите фото с вашим лицом');
            }

            if (status === 500) {
                throw new Error('Неизвестная ошибка, пожалуйста, попробуйте позже');
            }
        });
}
```

Большая математика

До недавнего времени в javascript не поддерживались числа больше 9007199254740992. Недавно появился bigint и эта проблема частично решена.

```
1 const bigNumber = 9007199254740992;
2
3 console.log(bigNumber); // Infinity
4
```

А также в главных ролях

- отсутствие многопоточки;
- медленные вычисления;
- плохое ООП, несмотря на его наличие;
- скудные коллекции (по сравнению с java, но великолепные по-сравнению с Go)
- <подставьте сюда свою причину>

Особенности и использование

Как использовать Node.js

1. Создаем файл <название>.js
2. Пишем туда код на javascript
3. Запускаем командой `node <название>.js`
4. Profit!

Стандартная библиотека

- `events` - встроенный EventEmitter
- `fs` - работа с файловой системой
- `os` - работа с операционной системой
- `http` - основной модуль для создания http-серверов
- `net` - низкоуровневое сетевое соединение
- `path` - для работы с путями в директориях
- и еще много-много, почитать можно здесь

CommonJS модули

```
1  // file.js
2
3  let count = 0
4  const increase = () => ++count
5  const reset = () => {
6      count = 0
7      console.log('Счетчик сброшен.')
8  }
9
10 exports.increase = increase
11 exports.reset = reset
12
13 // или (эквивалентно)
14 module.exports = {
15     increase,
16     reset
17 }
18
```

```
1  // используем CommonJS модуль
2  const {
3      increase,
4      reset
5  } = require('./file')
6
7  increase()
8  reset()
9
10 // или
11 const file = require('./file')
12 file.increase()
13 file.reset()
14
```

global и globalThis

`global` - аналог `window` в среде Node.js.

Если нет уверенности, где будет исполняться ваш код, или использоваться написанная вами библиотека - используйте `globalThis`.

Так как его используют? Этот ваш Node.js

Как уже повторяли для серверной разработки. Например:

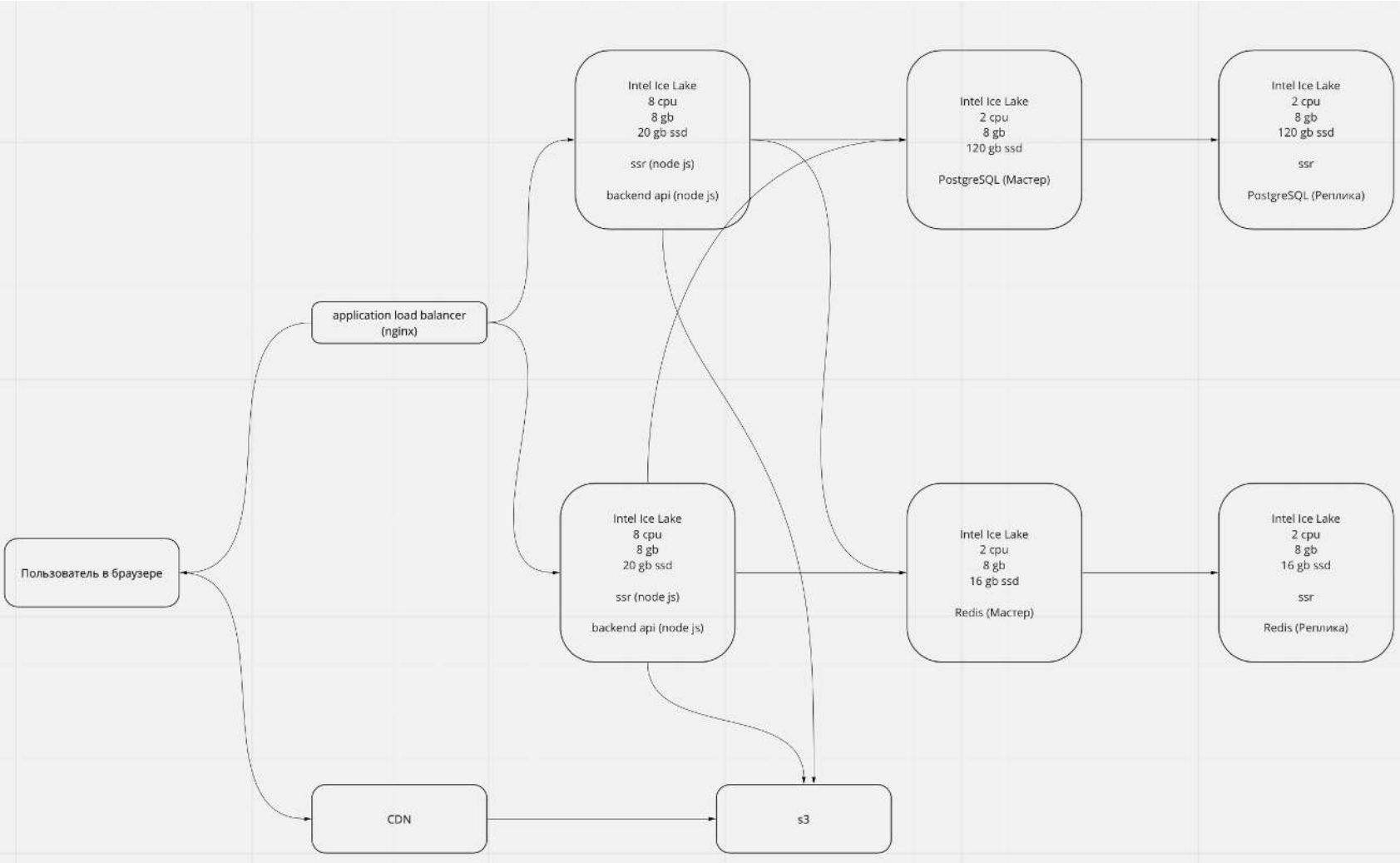
- SSR - серверный рендеринг для SPA
- BFF - бэкенд-прокси для агрегации данных
- Full-Backend API - полноценный бэкенд
- различные микросервисы со специфической логикой

Поглядим примерчики

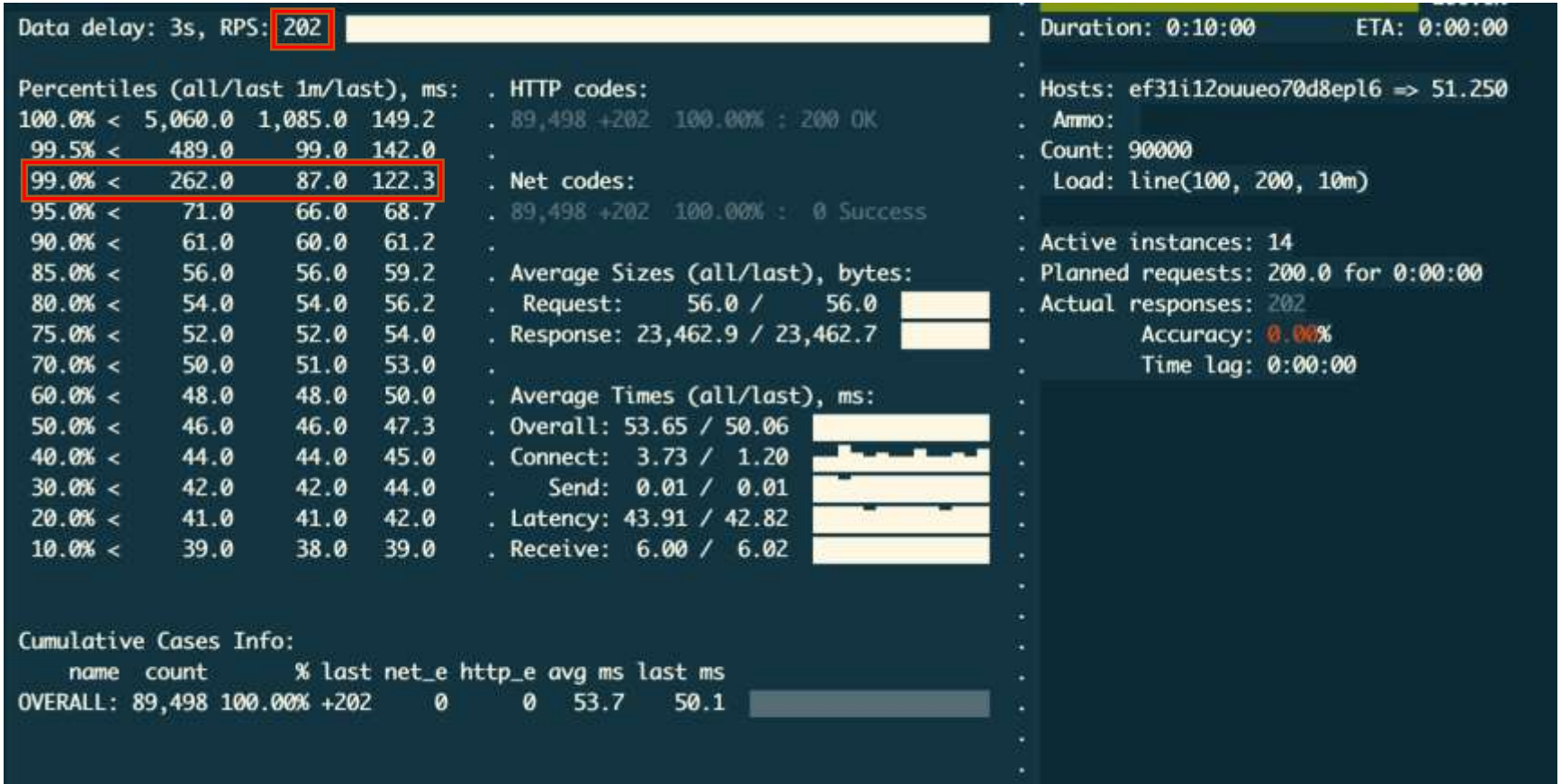
А что там все таки по скорости?
Давайте посмотрим пример

RPS

RPS - это метрика, которая определяет сколько запросов в секунду выдерживает сервис. Необходим для оценки нагрузки от пользователей и зачастую используется в расчетах.

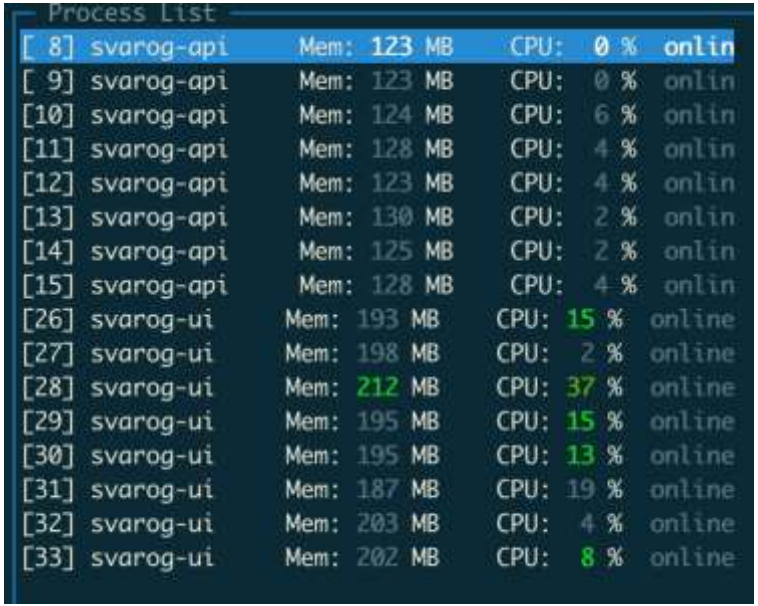

Intel Ice Lake
8 cpu
8 gb
20 gb ssd
ssr (node js)
backend api (node js)
Intel Ice Lake
2 cpu
8 gb
120 gb ssd
PostgreSQL (Мастер)
Intel Ice Lake
2 cpu
8 gb
120 gb ssd
ssr
PostgreSQL (Реплика)
application load balancer
(nginx)
Пользователь в браузере
Intel Ice Lake
8 cpu
8 gb
20 gb ssd
ssr (node js)
backend api (node js)
Intel Ice Lake
2 cpu
8 gb
16 gb ssd
Redis (Мастер)
Intel Ice Lake
2 cpu
8 gb
16 gb ssd
ssr
Redis (Реплика)
CDN
s3

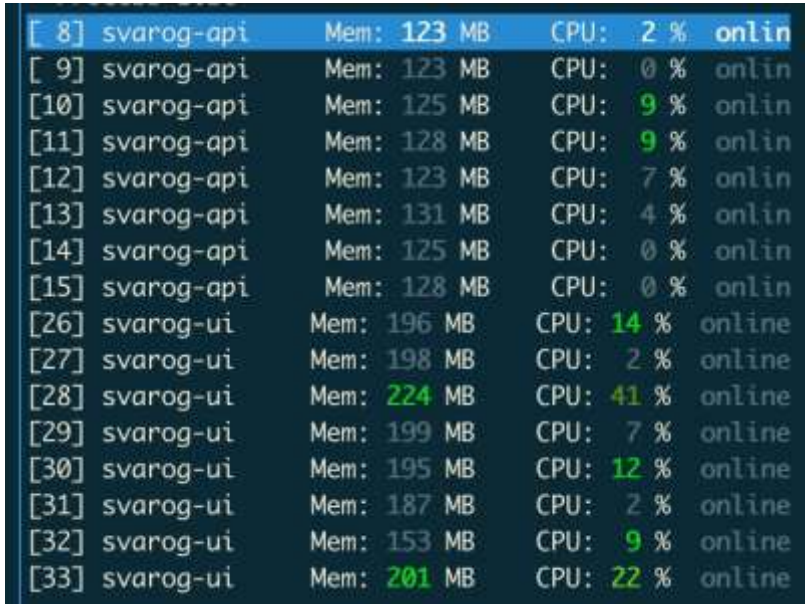
Высоконагруженность 200 rps



Высоконагруженность 200 rps

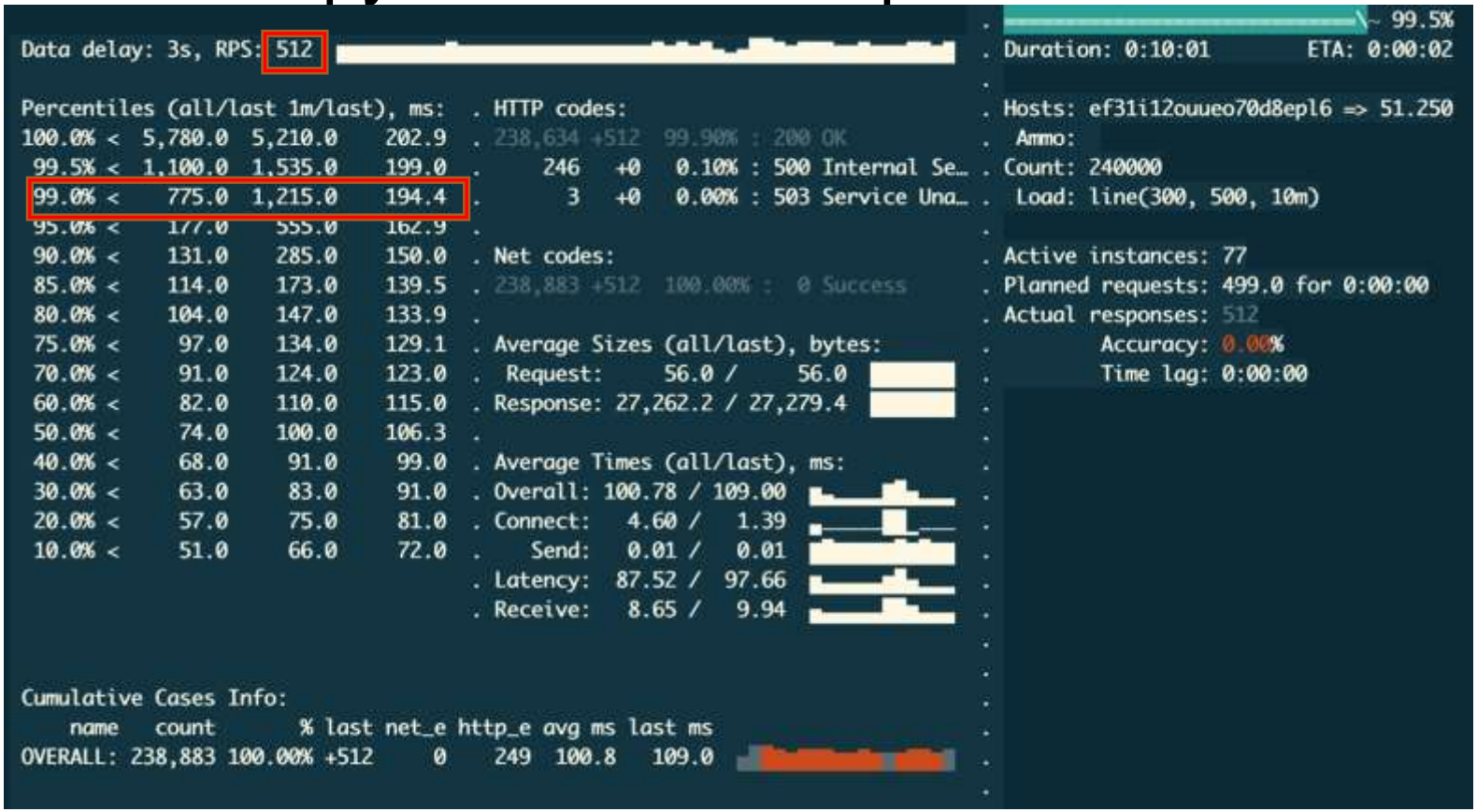
Высоконагруженность 200 rps



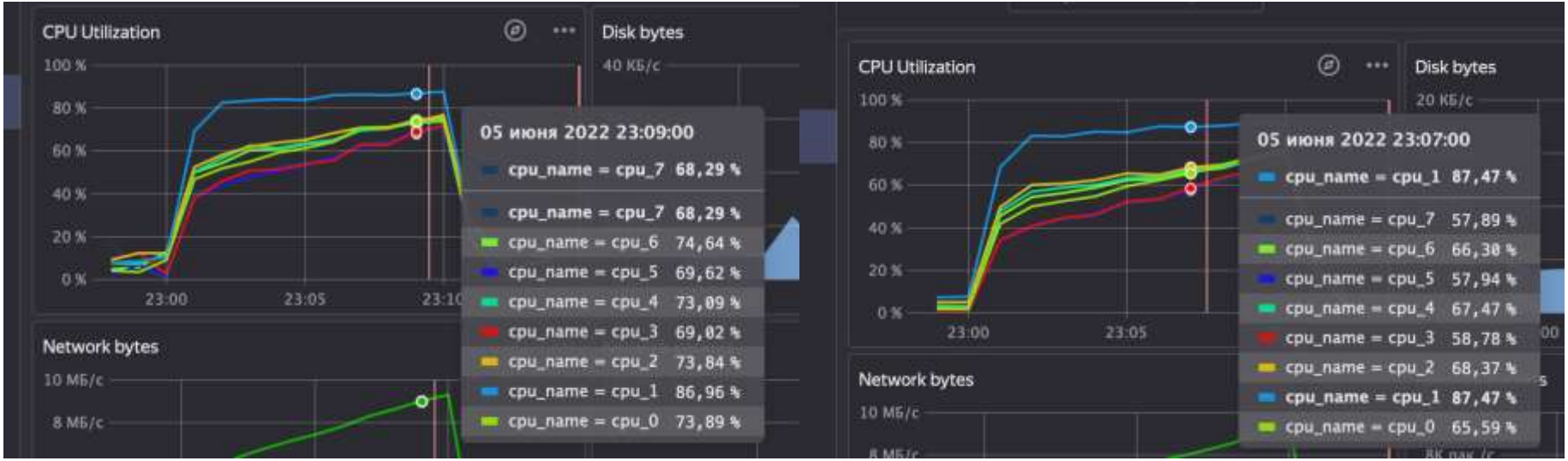


Высоконагруженность 200 rps

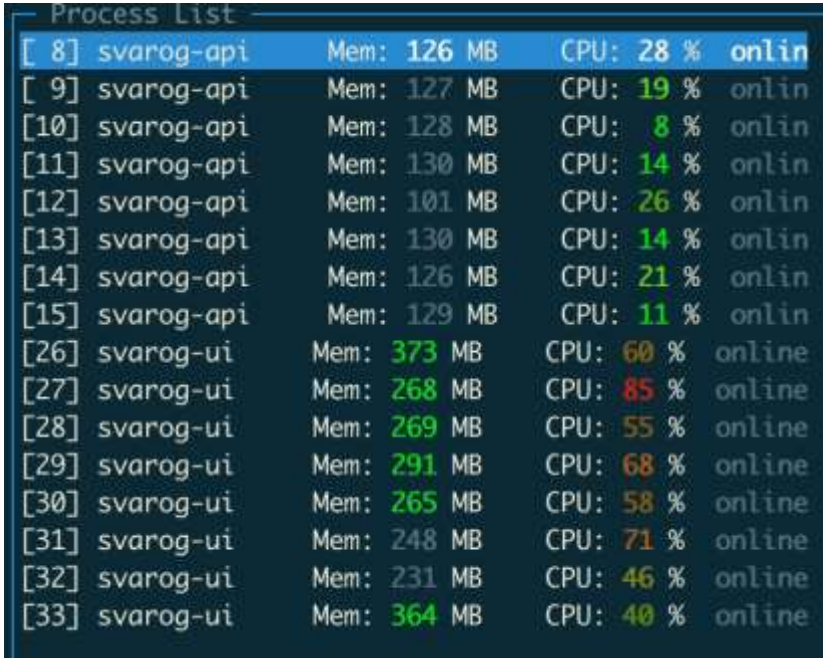
Высоконагруженность 500 rps

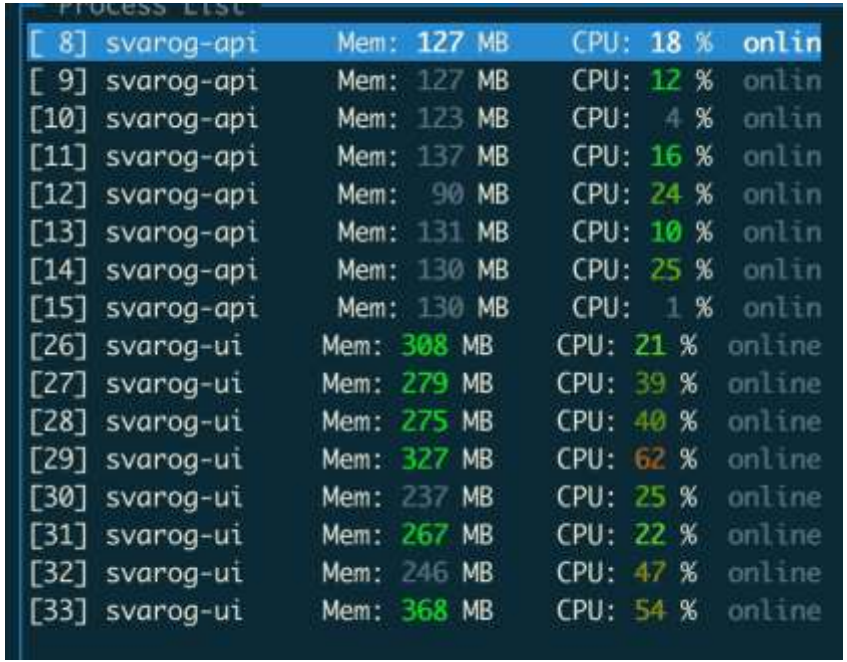


Высоконагруженность 500 rps

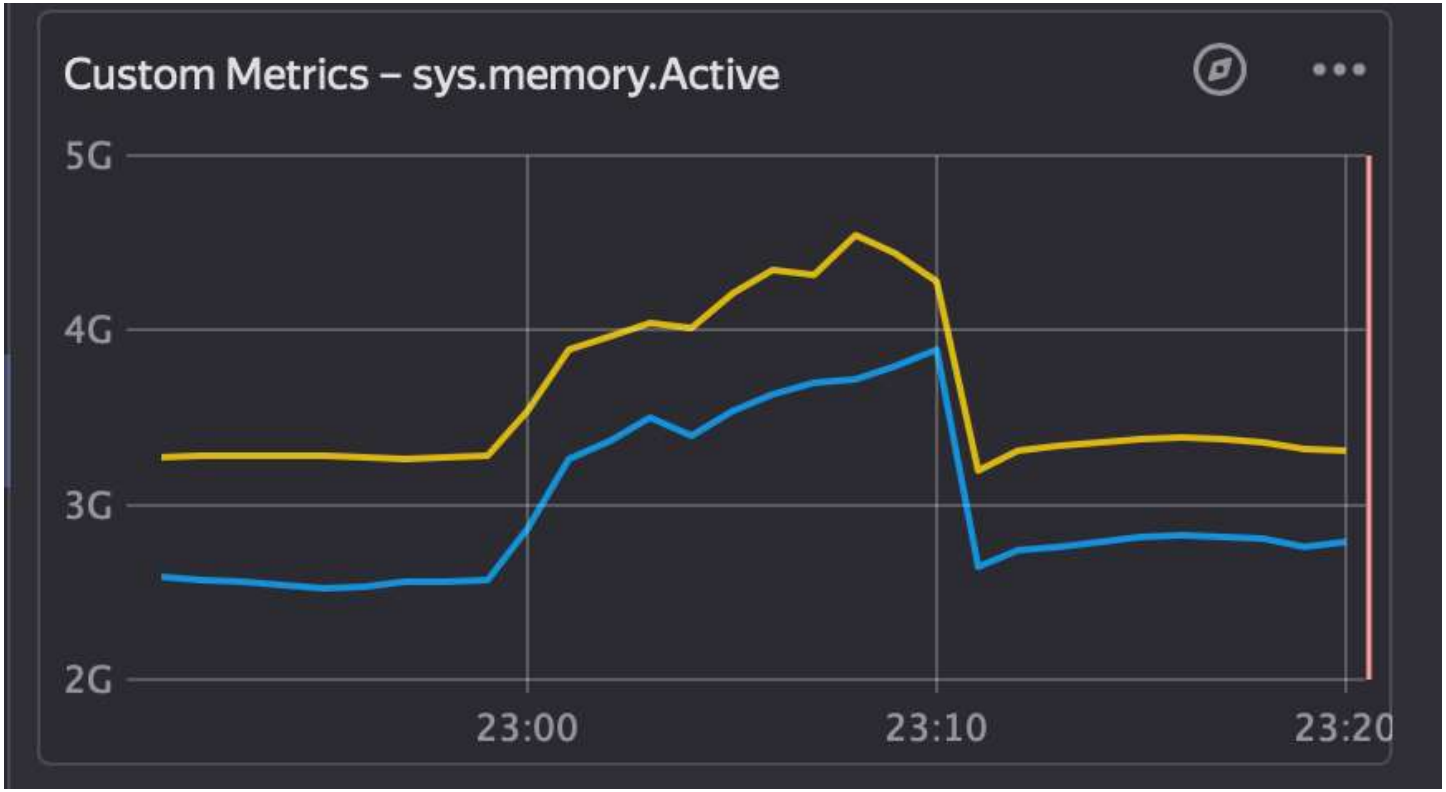


Высоконагруженность 500 rps

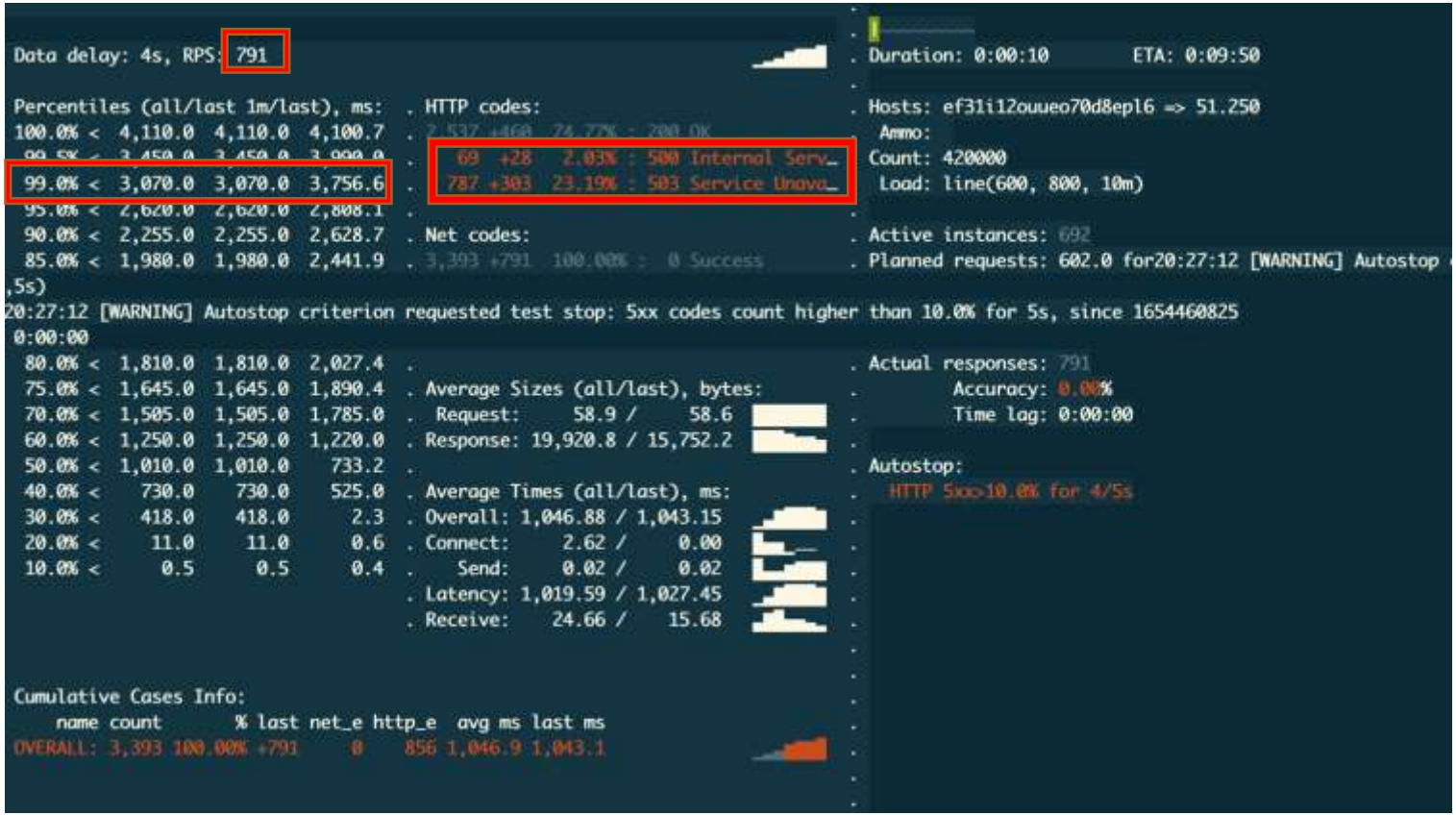




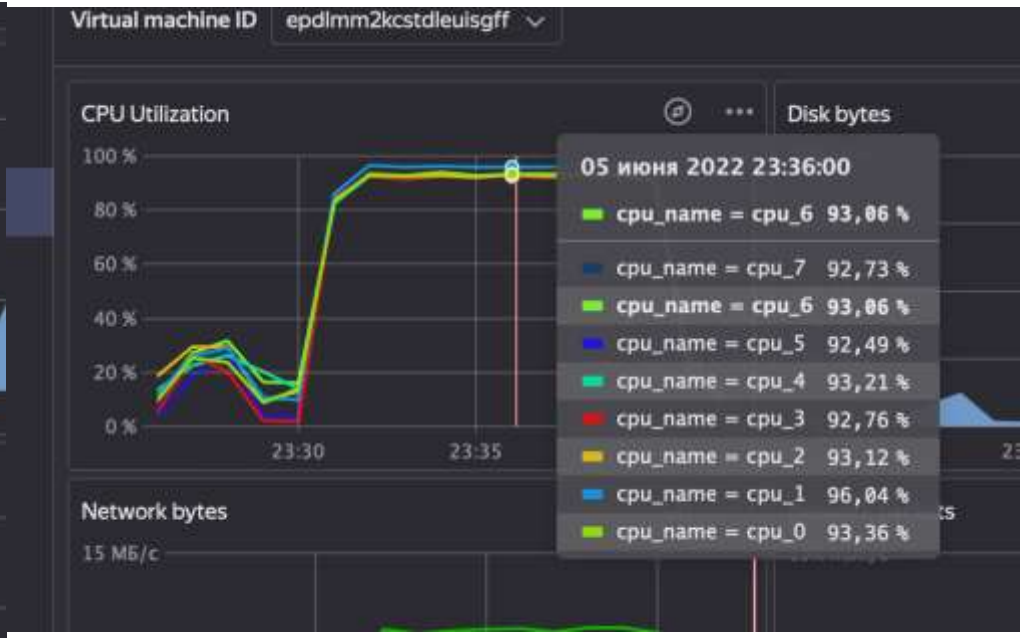
Высоконагруженность 500 rps



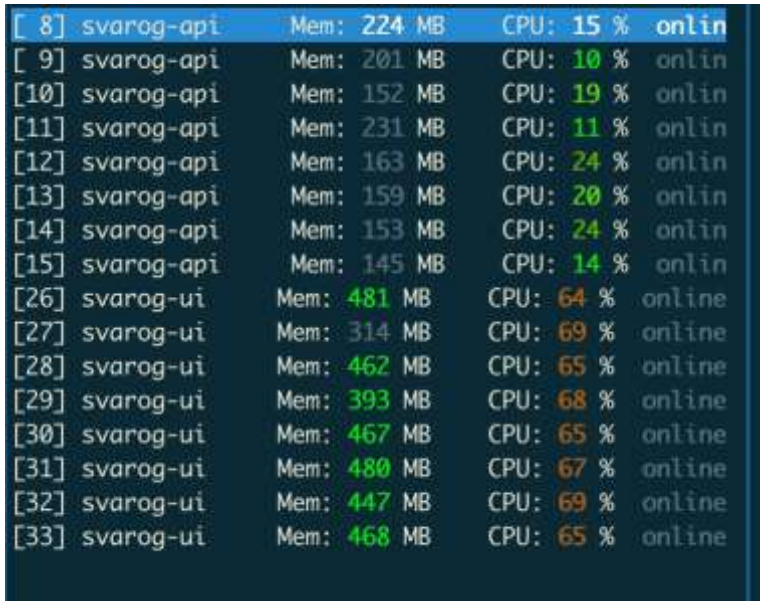
Высоконагруженность 800 rps

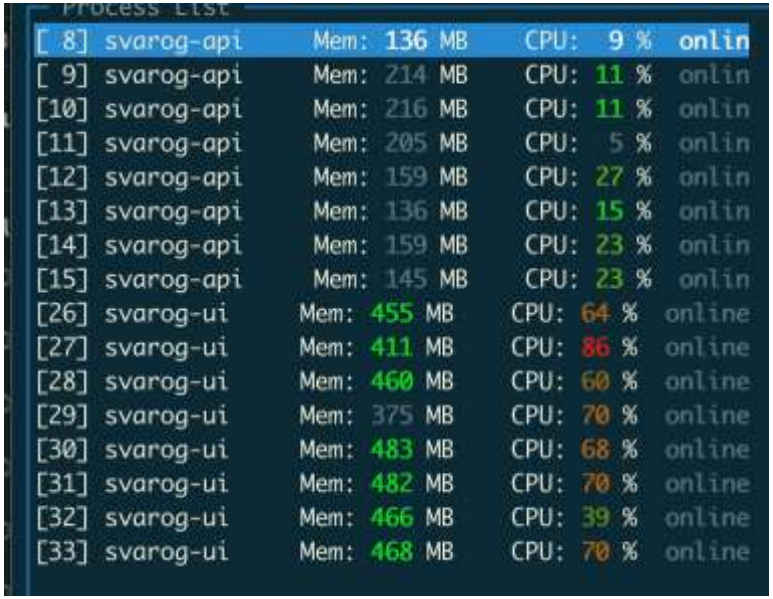


Высоконагруженность 800 rps

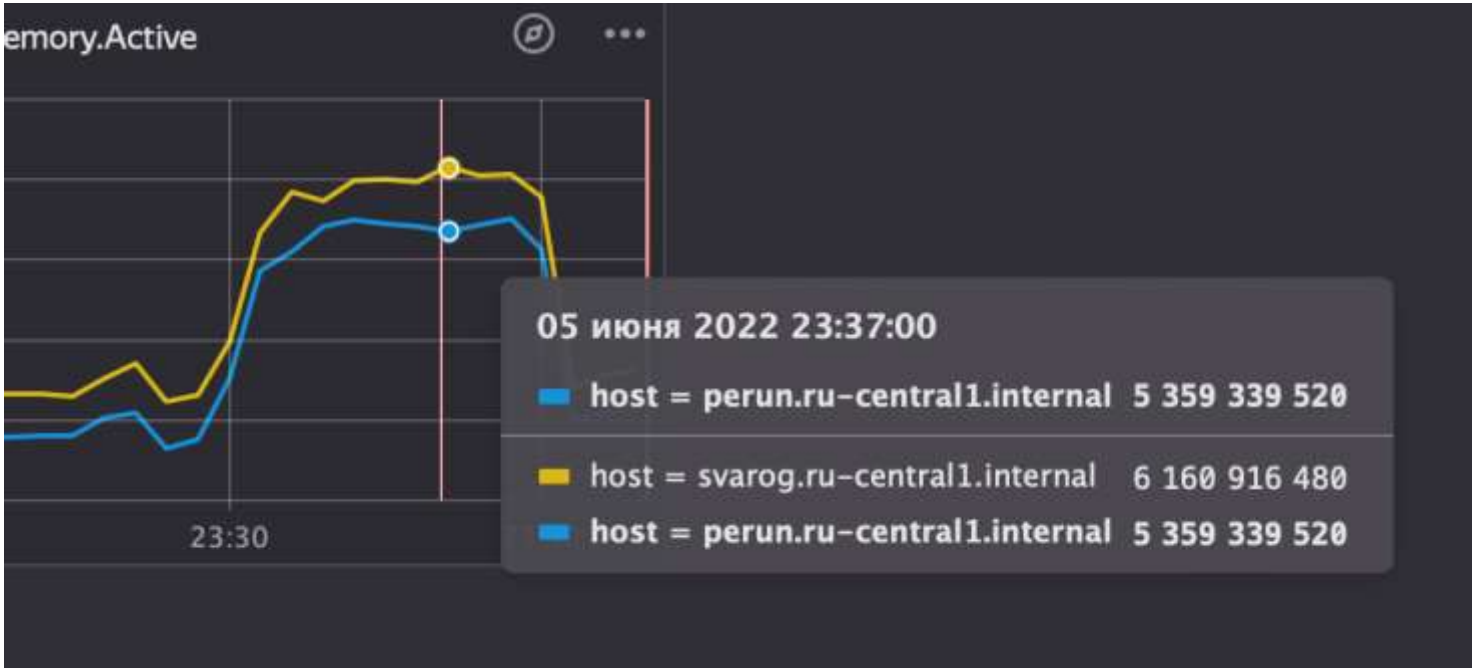


Высоконагруженность 800 rps

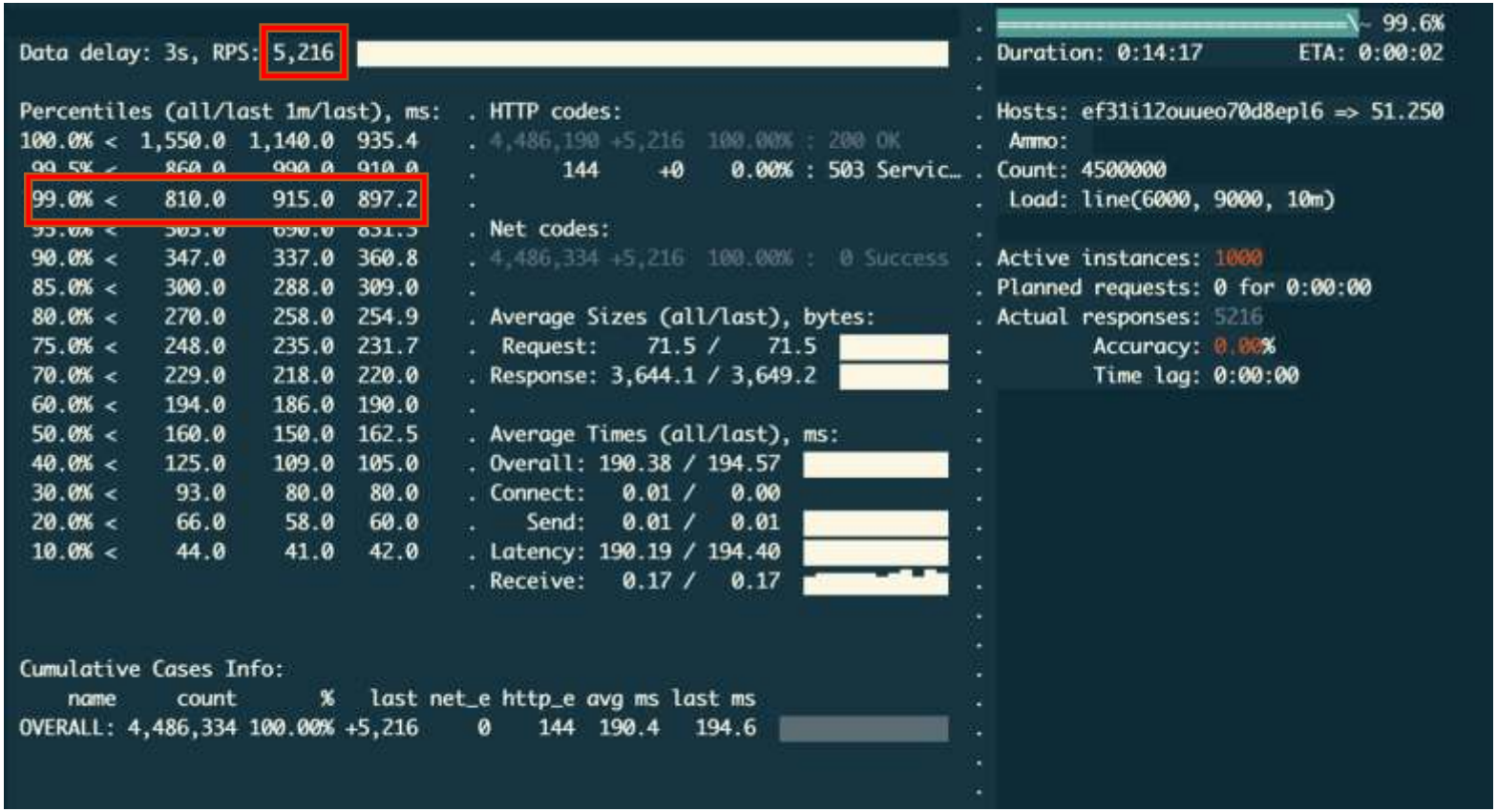




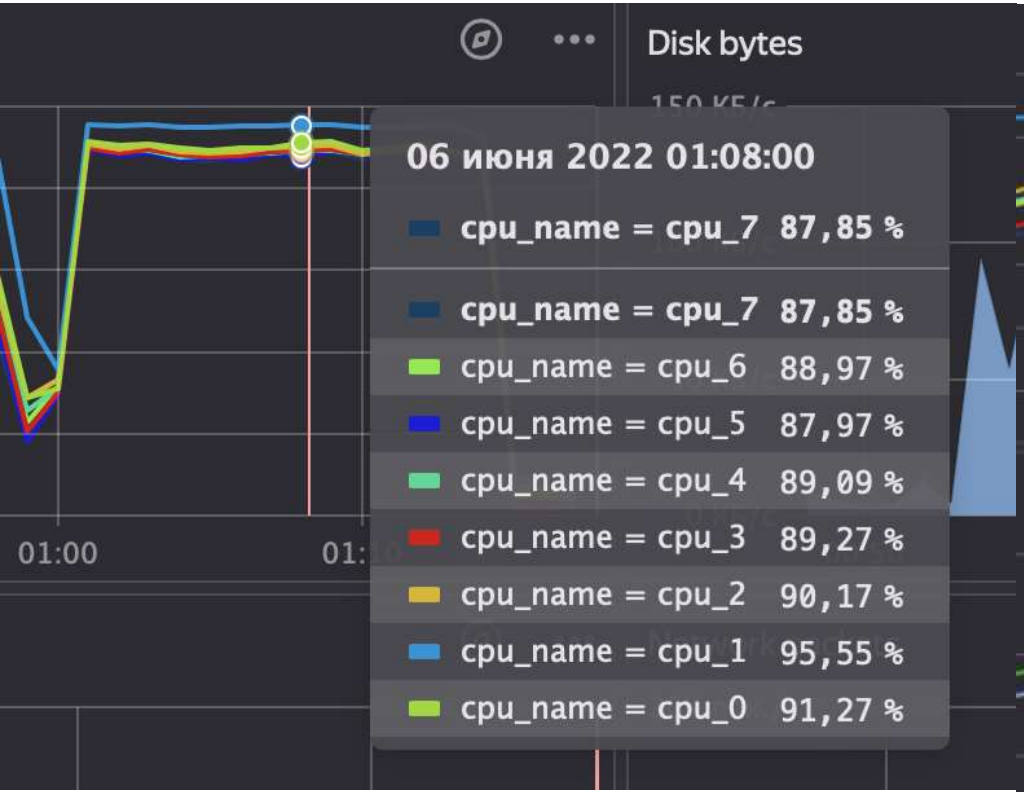
Высоконагруженность 800 rps



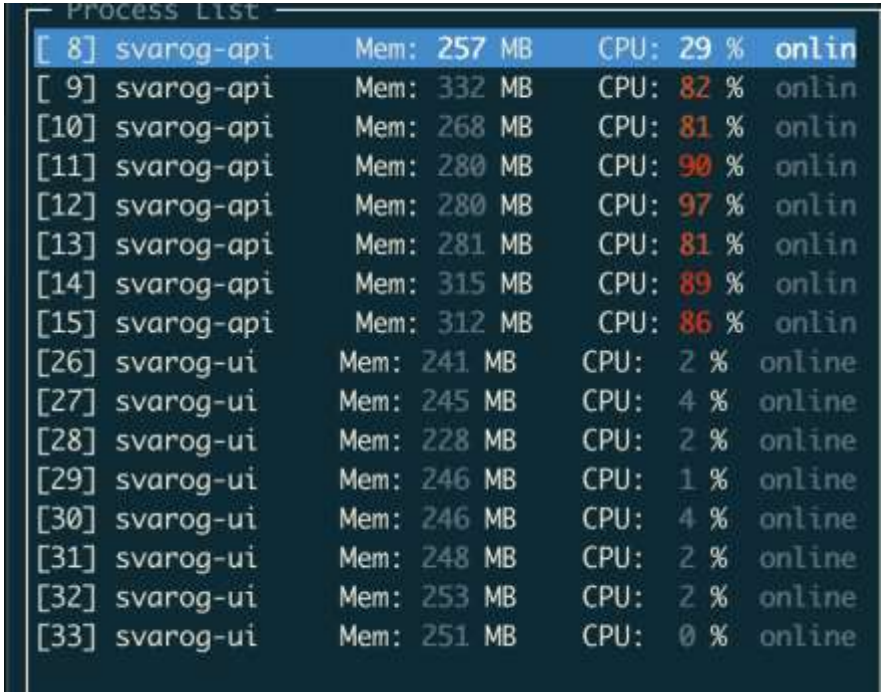
Высоконагруженность 5000 rps

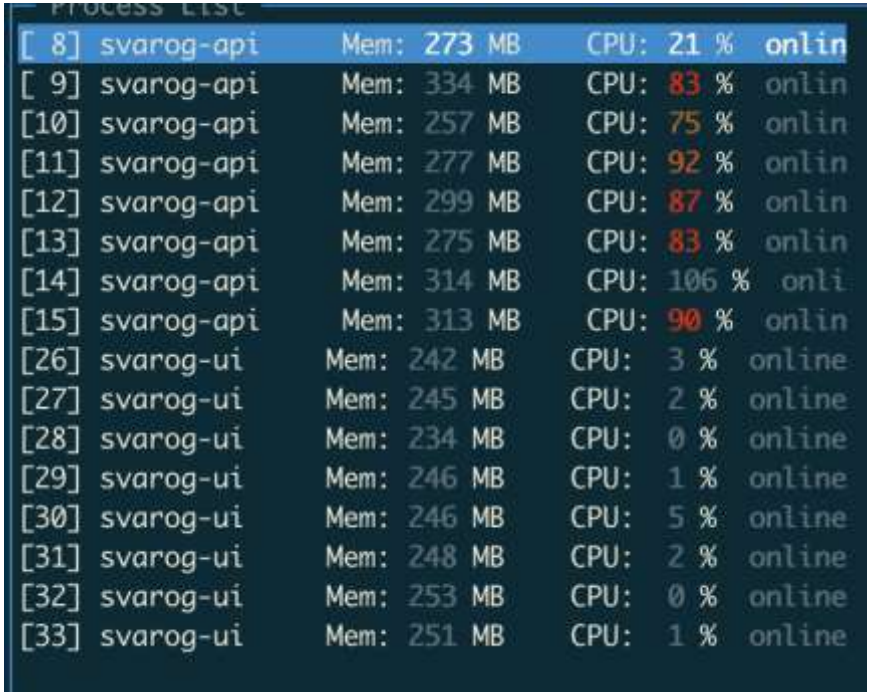


Высоконагруженность 5000 rps

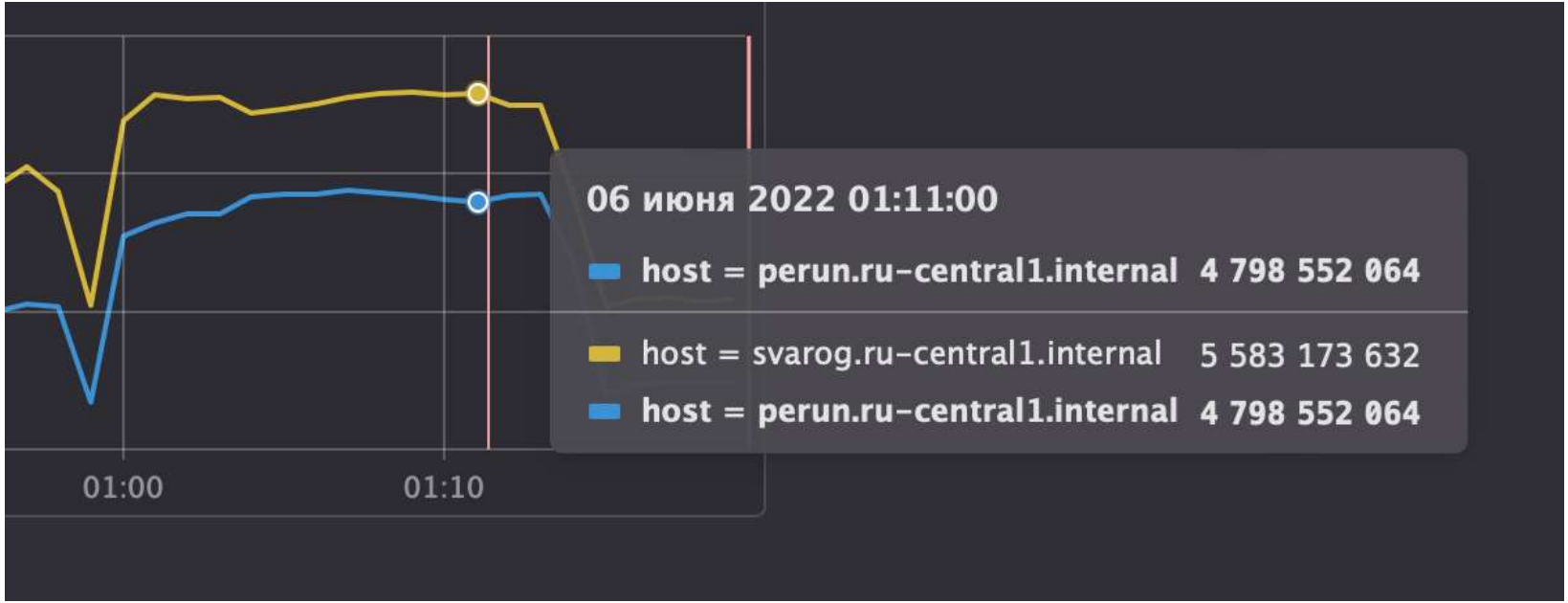


Высоконагруженность 5000 rps





Высоконагруженность 5000 rps



Высоконагруженность 5000 rps

А можно ли не городить свои костыли?

Можно! Давайте посмотрим на Nest.js

Nest.js

Nest.js - это фреймворк для создания эффективных, расширяемых серверных приложений.

Какие преимущества дает Nest.js

- готовую архитектуру;
- готовые решения для middleware;
- удобные механизмы валидации;
- возможность grpc почти из коробки;
- dependency injection
- java-подобное приложение.