

# Странный случай с node.js-метриками

Андрей Мелихов

В один прекрасный день  
случился релиз...

# Стадии релиза

- › Собирается версионированный билд
- › Проходит интеграционное и нагрузочное тестирование
- › Раскатывается на один сервер
- › Смотрим логи и метрики, раскатываем на всех

i

# NodeJS active requests total



Что за **active requests**?

Вызываем Event Handler



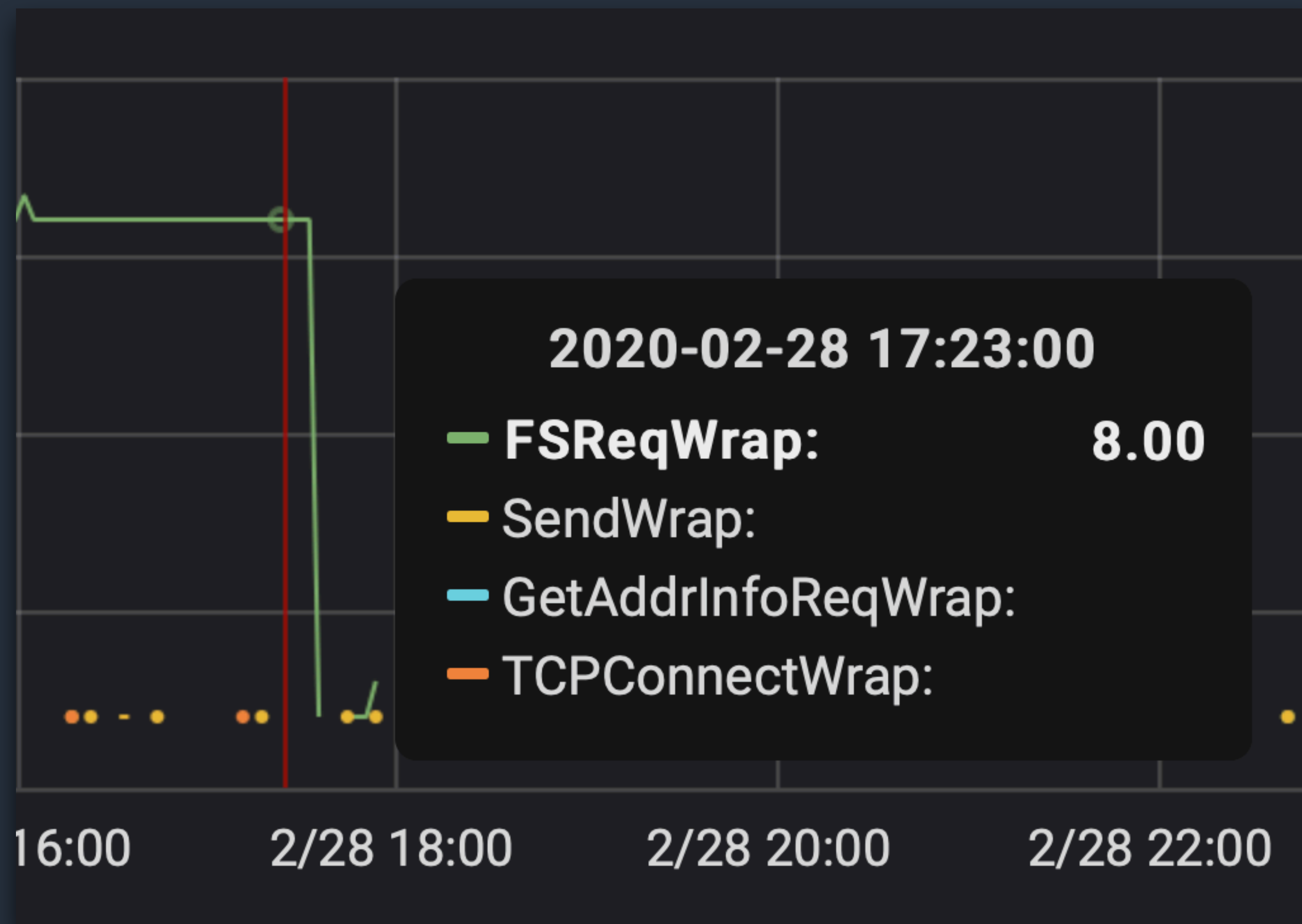
Регистрируем I/O задачу

```
340 static int uv__loop_alive(const uv_loop_t* loop) {  
341     return uv__has_active_handles(loop) ||  
342         uv__has_active_reqs(loop) ||  
343         loop->closing_handles != NULL;  
344 }
```

**process.\_getActiveHandles()**

**process.\_getActiveRequests()**





Что за **FSReqWrap**?

```
... 279 function readFile(path, options, callback) {
280     callback = maybeCallback(callback || options);
281     options = getOptions(options, { flag: 'r' });
282     if (!ReadFileContext)
283         ReadFileContext = require('internal/fs/read_file_context');
284     const context = new ReadFileContext(callback, options.encoding);
285     context.isUserFd = isFd(path); // file descriptor ownership
286
287     const req = new FSReqCallback();
288     req.context = context;
289     req.oncomplete = readFileAfterOpen;
290
291     if (context.isUserFd) {
292         process.nextTick(function tick() {
293             req.oncomplete(null, path);
294         });
295         return;
296     }
```

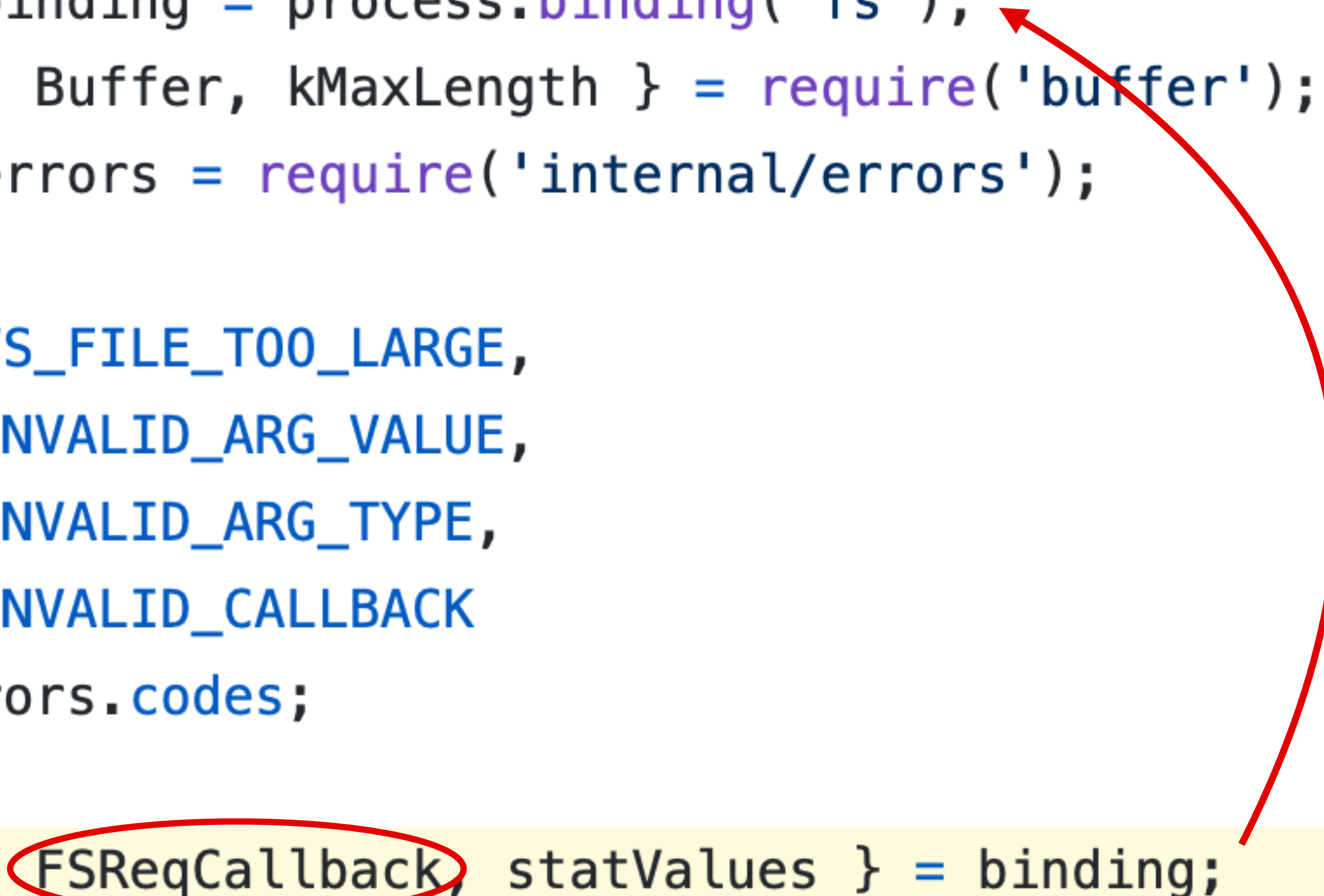
Node.js 10

⌵ ⌶		@@ -284,7 +284,7 @@ function readFile(path, options, callback) {
284	284	const context = new ReadFileContext(callback, options.encoding);
285	285	context.isUserFd = isFd(path); // file descriptor ownership
286	286	
287	-	const req = new FSReqWrap();
	287	+ const req = new FSReqCallback();
288	288	req.context = context;
289	289	req.oncomplete = readFileAfterOpen;
290	290	

Node.js 12

<https://github.com/nodejs/node/commit/172b4d7cebaa3ee047dd80ea908f06dd031c39f2#diff-9a205ef7ee967ee32efee02e58b3482dL287>

```
45  const binding = process.binding('fs');
46  const { Buffer, kMaxLength } = require('buffer');
47  const errors = require('internal/errors');
48  const {
49    ERR_FS_FILE_TOO_LARGE,
50    ERR_INVALID_ARG_VALUE,
51    ERR_INVALID_ARG_TYPE,
52    ERR_INVALID_CALLBACK
53  } = errors.codes;
54
55  const { FSReqCallback, statValues } = binding;
```





Вызываем Event Handler



**JS** land

**Single** thread



**C** land

**Multiple**  
threads

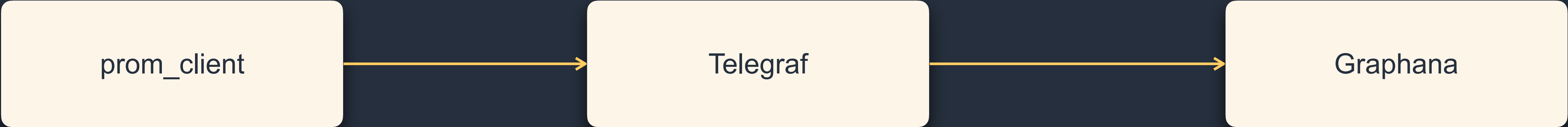
Регистрируем I/O задачу



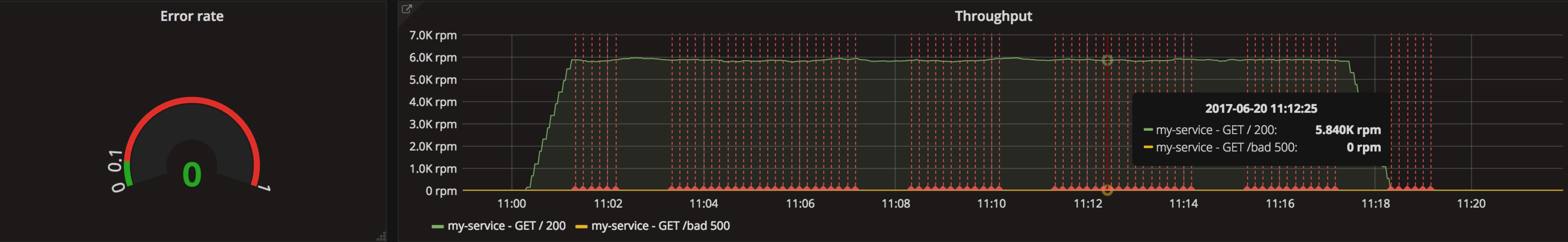
Кто постоянно читал файл?

И почему перестал?

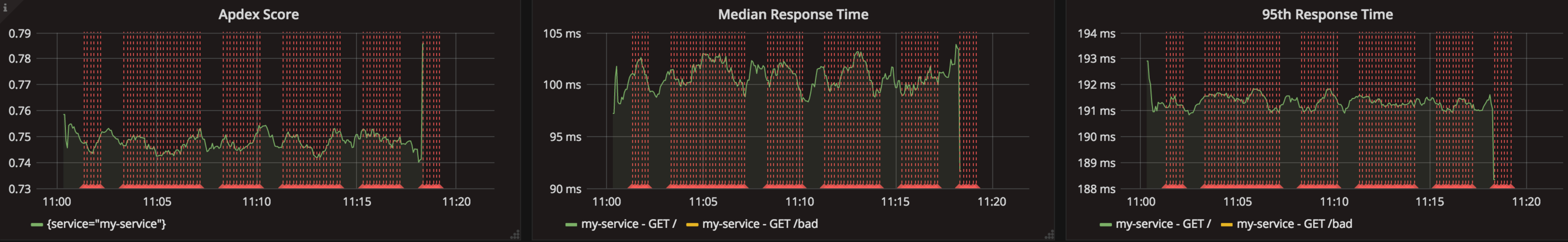
# Пайплайн мониторинга



Throughput



Response time



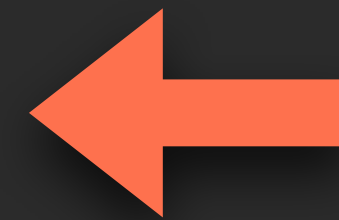


# Подключаем prom-client

```
const promClient = require('prom-client');
```

```
const promRegister = promClient.register;
```

```
promClient.collectDefaultMetrics({timeout});
```



```
setInterval(() => {
```

```
  const metrics = promRegister.getMetricsAsJSON();
```

```
  metrics.forEach((metric) => {
```

```
    metric.values.forEach((metricValue) => {
```

```
      monitoringService.byPerfomance({/.../});
```

```
    });
```

```
  });
```

```
}, timeout);
```

# Иследуя причины

- › Кейс проявился на обновлении prom\_client
- › Не воспроизводится на MacOS

Branch: master ▾


prom-client / lib / metrics /

Create new file

Upload files














Find file

History

 **amel-true** fix: remove useless setting timestamp after #333 (#343)

✓ Latest commit 236a767 20 days ago

..

 <a href="#">helpers</a>	fix: collect metrics on scrape, not timeout (#329)	last month
 <a href="#">eventLoopLag.js</a>	fix: collect metrics on scrape, not timeout (#329)	last month
 <a href="#">gc.js</a>	feat: implement GC metrics collection without native(C++) modules (#274)	2 months ago
 <a href="#">heapSizeAndUsed.js</a>	feature: remove all timestamp support (#333)	last month
 <a href="#">heapSpacesSizeAndUsed.js</a>	fix: remove useless setting timestamp after #333 (#343)	20 days ago
 <a href="#">osMemoryHeap.js</a>	fix: remove useless setting timestamp after #333 (#343)	20 days ago
 <a href="#">osMemoryHeapLinux.js</a>	fix: collect metrics on scrape, not timeout (#329)	last month
 <a href="#">processCpuTotal.js</a>	fix: remove useless setting timestamp after #333 (#343)	20 days ago
 <a href="#">processHandles.js</a>	fix: collect metrics on scrape, not timeout (#329)	last month
 <a href="#">processMaxFileDescriptors.js</a>	fix: collect metrics on scrape, not timeout (#329)	last month
 <a href="#">processOpenFileDescriptors.js</a>	fix: collect metrics on scrape, not timeout (#329)	last month
 <a href="#">processRequests.js</a>	fix: remove useless setting timestamp after #333 (#343)	20 days ago
 <a href="#">processStartTime.js</a>	fix: collect metrics on scrape, not timeout (#329)	last month

<https://github.com/siimon/prom-client/tree/master/lib/metrics>

```

54 60      return () => {
55      -      fs.readFile('/proc/self/status', 'utf8', (err, status) => {
56      -          if (err) {
57      -              return;
58      -          }
59      -          const now = Date.now();
60      -          const structuredOutput = structureOutput(status);
61      +      try {
62      +          const stat = fs.readFileSync('/proc/self/status', 'utf8');
63      +          const structuredOutput = structureOutput(stat);
61 64
62      -      residentMemGauge.set(structuredOutput.VmRSS, now);
63      -      virtualMemGauge.set(structuredOutput.VmSize, now);
64      -      heapSizeMemGauge.set(structuredOutput.VmData, now);
65      -      });
65      +      residentMemGauge.set(structuredOutput.VmRSS);
66      +      virtualMemGauge.set(structuredOutput.VmSize);
67      +      heapSizeMemGauge.set(structuredOutput.VmData);
68      +      } catch (er) {
69      +          return;
70      +      }
66 71      };
67 72      };

```

# Причина

ReadFile() // async



process.\_getActiveRequests()



ReadFile callback

# Проблема научного эксперимента

**ReadFile >> ReadFileSync**



```
// Sync I/O is often problematic, but /proc isn't really I/O, it a  
// virtual filesystem that maps directly to in-kernel data structures  
// and never blocks.  
//  
// Node.js/libuv do this already for process.memoryUsage()
```



# Уроки, которые мы вынесли

- Мониторинг внутреннего состояния позволяет много узнать о приложении
- Но он не должен влиять на состояние приложения
- Используемые инструменты нужно изучить досконально
- Как первичные, так и вторичные

**Спасибо!**