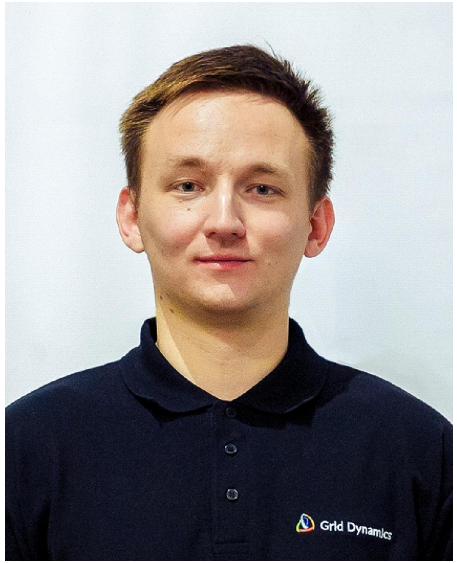
Поиск и устранение неисправностей Node.js приложений под капотом

Николай Матвиенко
2017г

Grid Dynamics

Автор



Матвиенко Николай

JS разработчик в Grid Dynamics

facebook: matvienko.nikolay

mail: matvi3nko@gmail.com

twitter: @matvi3nko

github: @nickkooper

Dan Shaw.
Основатель NODESOURCE и N|Solid
twitter.com/dshaw

Thomas Watson.
Node.js Diagnostics Working Group member
twitter.com/wa7son

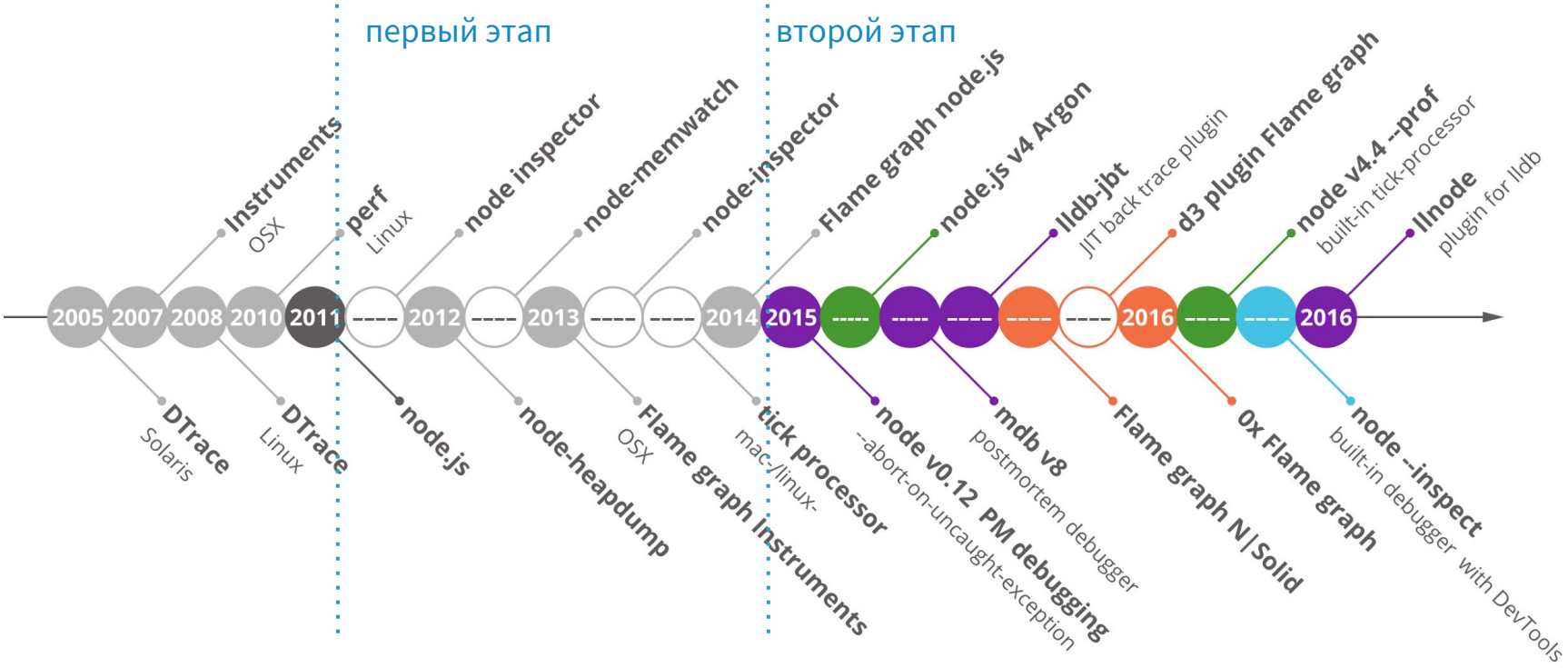
Node.js Diagnostics Working Group

https://github.com/nodejs/diagnostics

Работа делится на несколько доменов:

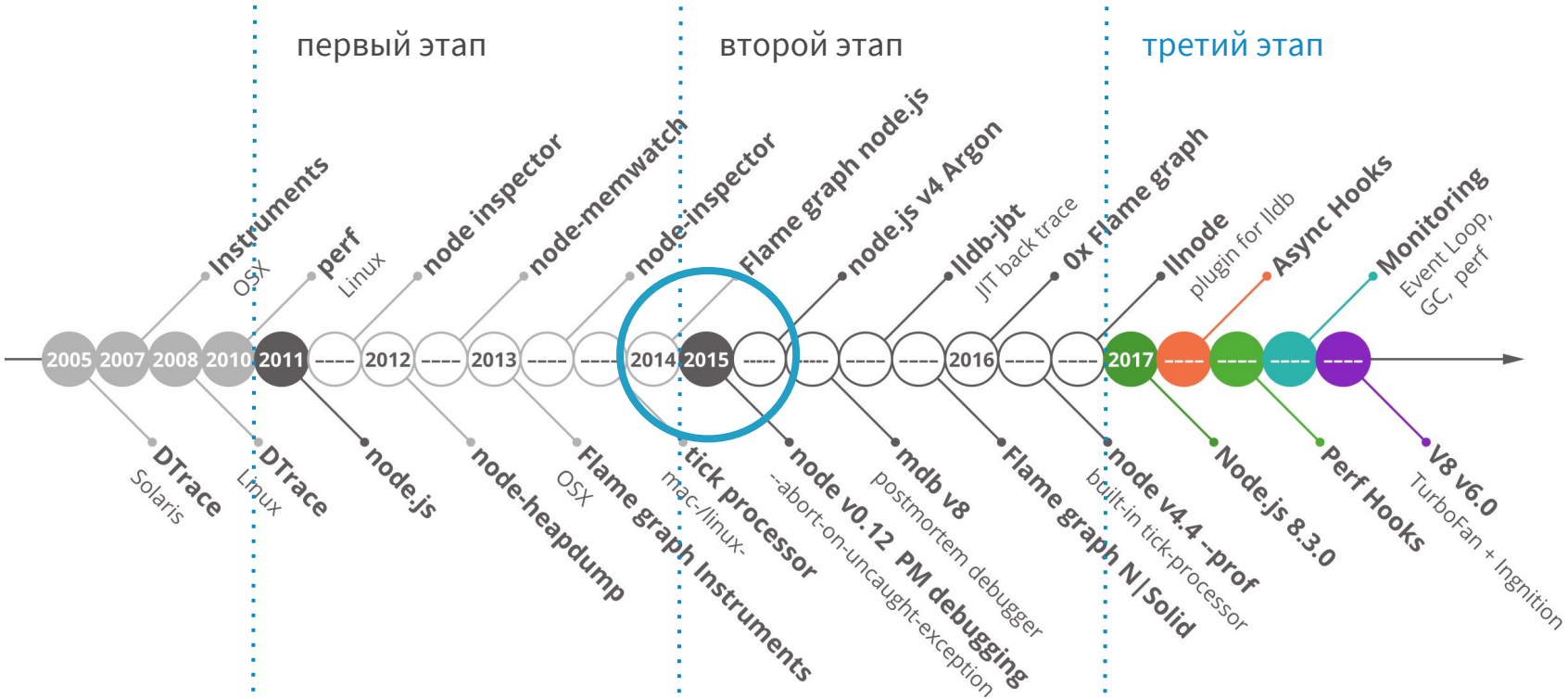
1. Tracing
2. Profiling
3. Heap and Memory Analysis
4. Step Debugging

Этапы развития Диагностики Node.js до 2017г



* Здесь отображено появление наиболее значимых инструментов.

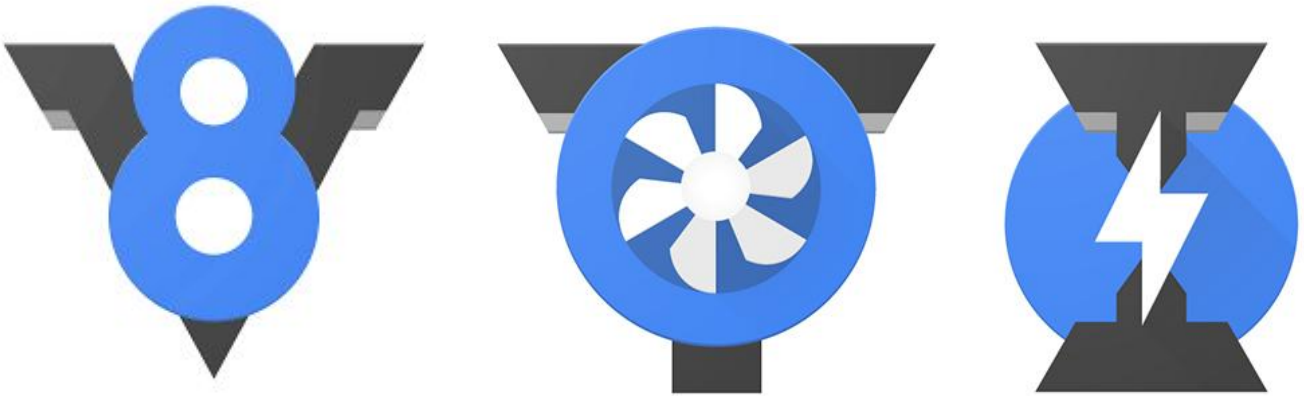
Третий этап в диагностике Node.js



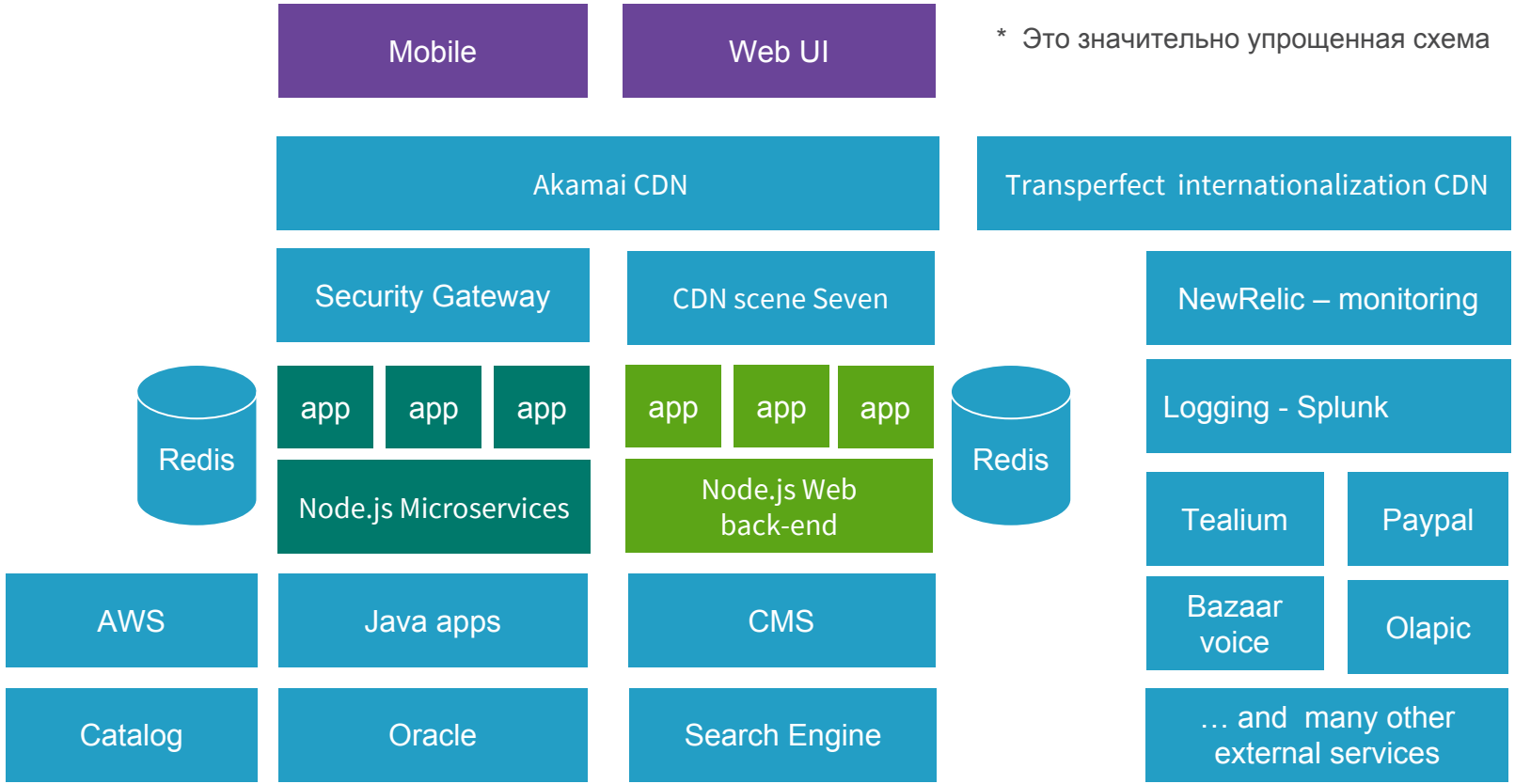
* Здесь отображено появление наиболее значимых инструментов.

Обновление движка V8

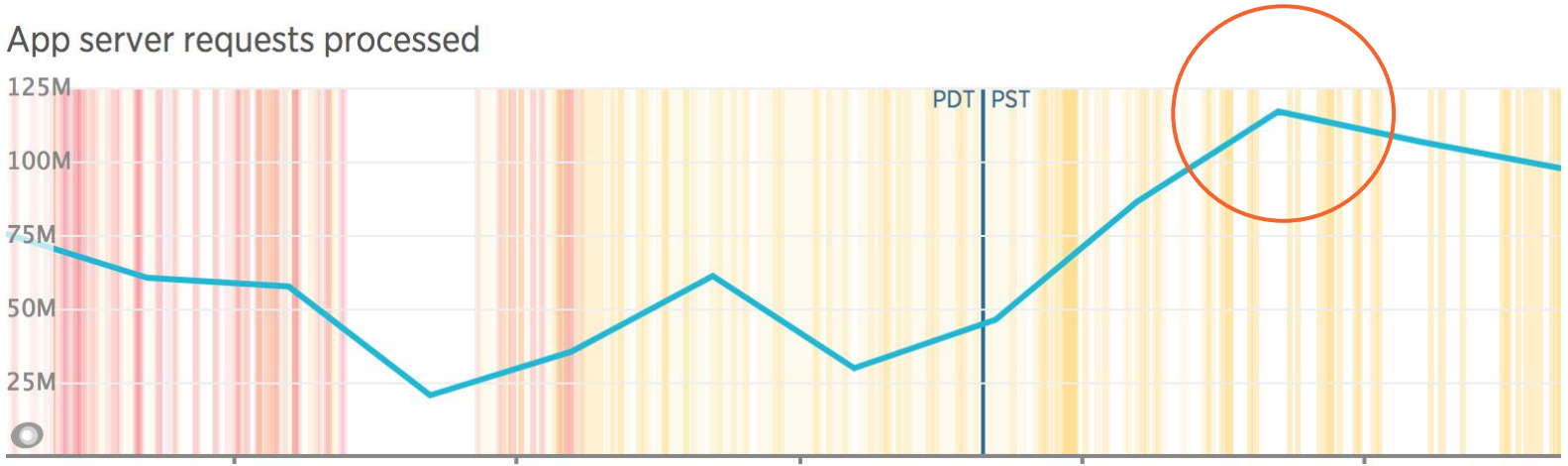
1. Большинство инструментов не поддерживают последние версии Node.js (V8)
2. Приходится выбирать: производительность или инструменты
3. Либо появляются новые инструменты, но без поддержки предыдущих версий Node.js



Node.js в Enterprise архитектуре



Количество запросов в день



War room

План

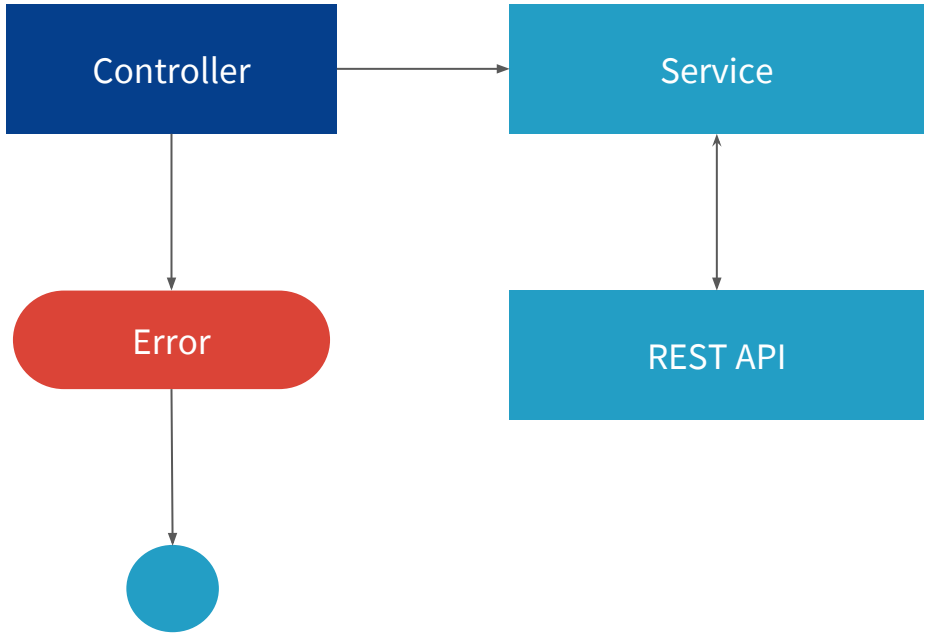
1. Production troubleshooting
2. Поиск и устранение утечек памяти
3. Профилирование и улучшение производительности приложения

1. Отладка приложений в production

Причины для отладки:

1. Uncaught Exception и Unhandled Promise Rejection
2. Трудно воспроизводимые ошибки
3. Production environment
4. ASAP

Пример



Пример 1. Резервирование продуктов

```
module.exports = class ProductController {
  constructor (reservationService) {
    this._reservationService = reservationService;
  }

  reserve (req, res) {
      const { id, storeId } = req.cookies['profile'];
      const rewards = req.cookies['rewards'];
      const { products } = req.body;

      //rewards is UNDEFINED
      this._reservationService.reserve(id, storeId, rewards.id, products)
        .then(data => { return res.send(data); });
  }};
```

Error

Пример 1. Резервирование продуктов

```
module.exports = class ProductController {
  constructor (reservationService) {
    this._reservationService = reservationService;
  }

  reserve (req, res) {
      const { id, storeId } = req.cookies['profile'];
      const rewards = req.cookies['rewards'];
      const { products } = req.body;

      %DebugTrace();

      //rewards is UNDEFINED
      this._reservationService.reserve(id, storeId, rewards.id, products)
        .then(data => { return res.send(data); });
  }};
```

node --allow-natives-syntax app.js

```
[2]: reserve [/Users/nmatvienko/projects/holyjs/controllers/ProductController.js:15] [bytecode=0x3d16f5c7d749
offset=97](this=0x3d165525d139 <ProductController map = 0x3d16c90f4ea9>#1#,
req=0x3d169d0982c1 <IncomingMessage map = 0x3d1695903b71>#2#,
res=0x3d169d099e99 <ServerResponse map = 0x3d1695903541>#3#) {
 // stack-allocated locals
 var /* anonymous */ = 0x3d169d0a0ca1 <Object map = 0x3d1695903a11>#26#
 var id = 1
 var storeId = 1
 var rewards = 0x3d1600e02311 <undefined>
 var products = 0x3d169d0a0df1 <JSArray[3]>#28#
 // heap-allocated locals
 var res = 0x3d169d099e99 <ServerResponse map = 0x3d1695903541>#3#

--------- s o u r c e   c o d e ---------
function reserve(req, res) {\x0a      const { id, storeId } = req.cookies['profile'];\x0a      const rewards =
req.cookies['rewards'];\x0a      const { products } = req.body;\x0a\x0a      %DebugTrace();\x0a\x0a      //rewards is
UNDEFINED\x0a      this._reservationService.reserve(id, storeId, rewards.id, products)\x0a    ...
```

node --allow-natives-syntax app.js

```
[2]: reserve [/Users/nmatvienko/projects/holyjs/controllers/ProductController.js:15] [bytecode=0x3d16f5c7d749
offset=97] this=0x3d165525d139 <ProductController map = 0x3d16c90f4ea9>#1#,
req=0x3d169d0982c1 <IncomingMessage map = 0x3d1695903b71>#2#,
res=0x3d169d099e99 <ServerResponse map = 0x3d1695903541>#3#) {
 // stack-allocated locals
 var /* anonymous */ = 0x3d169d0a0ca1 <Object map = 0x3d1695903a11>#26#
 var id = 1
 var storeId = 1
 var rewards = 0x3d1600e02311 <undefined>
 var products = 0x3d169d0a0df1 <JSArray[3]>#28#
 // heap-allocated locals
 var res = 0x3d169d099e99 <ServerResponse map = 0x3d1695903541>#3#

--------- s o u r c e   c o d e ---------
function reserve(req, res) {\x0a      const { id, storeId } = req.cookies['profile'];\x0a      const rewards =
req.cookies['rewards'];\x0a      const { products } = req.body;\x0a\x0a      %DebugTrace();\x0a\x0a      //rewards is
UNDEFINED\x0a      this._reservationService.reserve(id, storeId, rewards.id, products)\x0a    ...
```

node --allow-natives-syntax app.js

[2]: reserve [/Users/nmatvienko/projects/holyjs/controllers/ProductController.js:15] [bytecode=0x3d16f5c7d749 offset=97](this=0x3d165525d139 <ProductController map = 0x3d16c90f4ea9>#1#,
req=0x3d169d0982c1 <IncomingMessage map = 0x3d1695903b71>#2#,
res=0x3d169d099e99 <ServerResponse map = 0x3d1695903541>#3#) {

```
// stack-allocated locals
var /* anonymous */ = 0x3d169d0a0ca1 <Object map = 0x3d1695903a11>#26#
var id = 1
var storeId = 1
var rewards = 0x3d1600e02311 <undefined>
var products = 0x3d169d0a0df1 <JSArray[3]>#28#
// heap-allocated locals
var res = 0x3d169d099e99 <ServerResponse map = 0x3d1695903541>#3#
```

--------- s o u r c e c o d e ---------
function reserve(req, res) {\x0a const { id, storeId } = req.cookies['profile'];\x0a const rewards = req.cookies['rewards'];\x0a const { products } = req.body;\x0a\x0a %DebugTrace();\x0a\x0a //rewards is UNDEFINED\x0a this._reservationService.reserve(id, storeId, rewards.id, products)\x0a ...

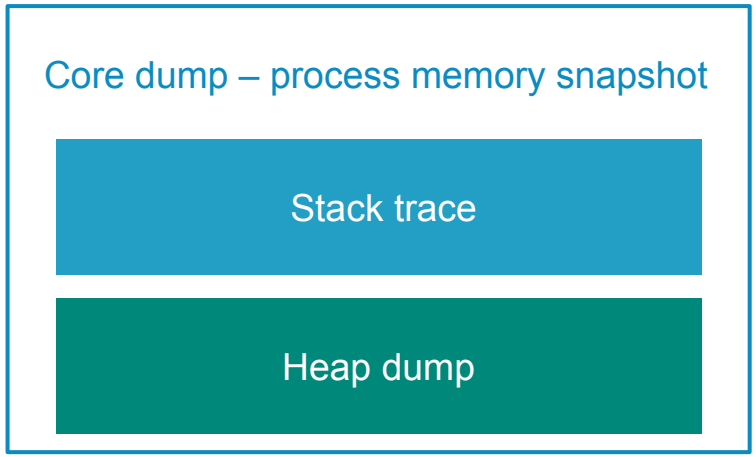
node --allow-natives-syntax app.js

```
[2]: reserve [/Users/nmatvienko/projects/holyjs/controllers/ProductController.js:15] [bytecode=0x3d16f5c7d749
offset=97](this=0x3d165525d139 <ProductController map = 0x3d16c90f4ea9>#1#,
req=0x3d169d0982c1 <IncomingMessage map = 0x3d1695903b71>#2#,
res=0x3d169d099e99 <ServerResponse map = 0x3d1695903541>#3#) {
 // stack-allocated locals
 var /* anonymous */ = 0x3d169d0a0ca1 <Object map = 0x3d1695903a11>#26#
 var id = 1
 var storeId = 1
 var rewards = 0x3d1600e02311 <undefined>
 var products = 0x3d169d0a0df1 <JSArray[3]>#28#
 // heap-allocated locals
 var res = 0x3d169d099e99 <ServerResponse map = 0x3d1695903541>#3#
```

```
--------- s o u r c e   c o d e ---------
function reserve(req, res) {\x0a      const { id, storeId } = req.cookies['profile'];\x0a      const rewards =
req.cookies['rewards'];\x0a      const { products } = req.body;\x0a\x0a      %DebugTrace();\x0a\x0a      //rewards is
UNDEFINED\x0a      this._reservationService.reserve(id, storeId, rewards.id, products)\x0a    ...
```

Node.js v0.12 Post-mortem debugging

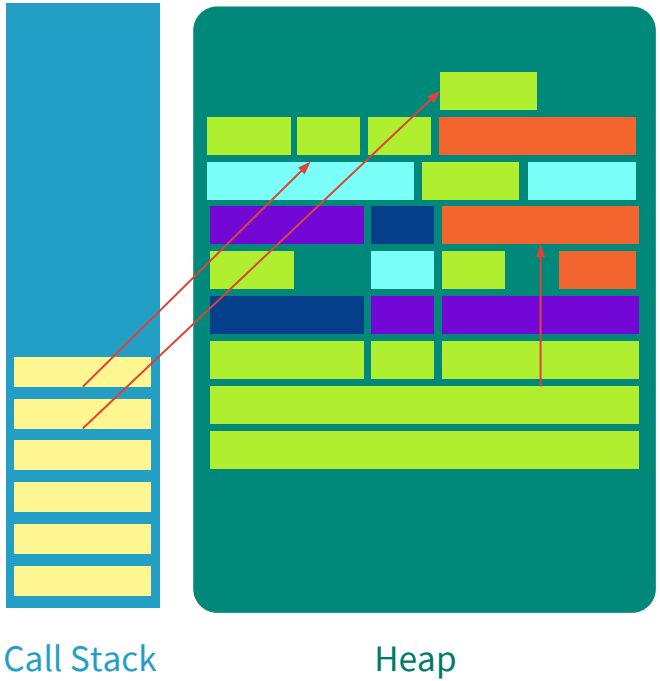
node **--abort-on-uncaught-exception** app.js



1. Прочитать core dump отладчиком
2. Прочитать из памяти Call Stack – список последних вызовов в Mean Thread
3. Получить состояние объектов из Heap

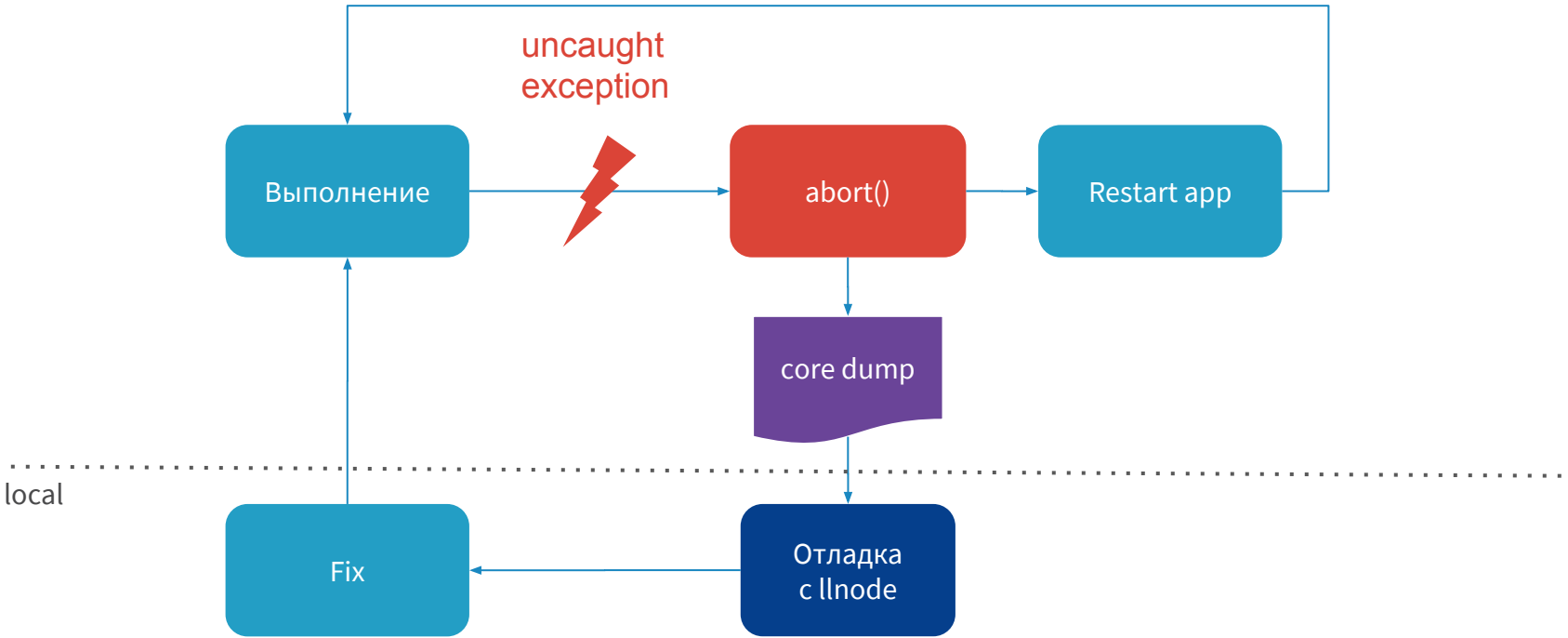
Структура V8 Call Stack и Heap

- 984 KB для 64bit,
- 492 KB для 32 bit систем
- Статическое выделение/освобождение памяти
- Делится на frames
- Хранит только SMI immediate small integers 31bit



- Большой размер памяти
- Динамическое выделение памяти
- Делится на пространства
- Хранит ссылочные типы (объекты), замыкания, строки, глобальные объекты

Алгоритм создания core dump



Post-mortem debuggers

Debuggers

1. mdb
2. gdb
3. lldb

Plugins

1. mdb_v8
2. gdb JIT Interface in V8
3. lldb-v8
4. lldb-jbt
5. llnode for lldb

Пример 1. Резервирование продуктов

```
module.exports = class ProductController {
  constructor (reservationService) {
    this._reservationService = reservationService;
  }

  reserve (req, res) {
      const { id, storeId } = req.cookies['profile'];
      const rewards = req.cookies['rewards'];
      const { products } = req.body;

      //rewards is UNDEFINED
      this._reservationService.reserve(id, storeId, rewards.id, products)
        .then(data => {
            return res.send(data);
        });
  }};
```

Error

Пример 1. Запуск приложения

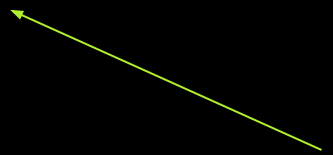
```
GD:holyjs nmatvienko$ ulimit -c unlimited
GD:holyjs nmatvienko$ NODE_ENV=production  node --abort-on-uncaught-exception app.js
This process pid is 13364
```

Пример 1. Uncaught exception

```
This process pid is 13364

Uncaught TypeError: Cannot read property 'id' of undefined

FROM
ProductController.reserve ( /holyjs/controllers/ProductController.js:1:1)
IncomingMessage.<anonymous> ( /holyjs/app.js:1:1)
emitNone (events.js:1:1)
_combinedTickCallback (internal/process/next_tick.js:1:1)
process._tickCallback (internal/process/next_tick.js:1:1)

Illegal instruction: 4 (core dumped)
```

Пример 1. Проверка создания core dump

```
GD:holyjs nmatvienko$ ls -lrth /cores/core.13364

-r--------  1 nmatvienko  admin   1.3G Oct  1 21:13 /cores/core.13364
```



core dump

Пример 1. Чтение dump памяти процесса

```
GD:holyjs nmatvienko$ ls -lrth /cores/core.13364

-r--------  1 nmatvienko  admin   1.3G Oct  1 21:13 /cores/core.13364

$ llnode -c /cores/core.13364

(lldb) target create --core "/cores/core.13364"
Core file '/cores/core.13364' (x86_64) was loaded.
(lldb) plugin load
/Users/nmatvienko/.nvm/versions/node/v8.4.0/bin/../lib/node_modules/llnode/llnode.dylib
(lldb) settings set prompt '(llnode) '
```

Пример 1. Трассировка

```
$(llnode) bt

        * thread #1, stop reason = signal SIGSTOP
          * frame #0: 0x0000000100be6b82 node`v8::base::OS::Abort() + 18
            frame #1: 0x00000001006da0b3 node`v8::internal::Isolate::Throw(v8::internal::Object*)
            frame #2: 0x0000000100697d14 node`v8::internal::LoadIC::Load(v8::internal::Handle<v8::internal::Object>,
            v8::internal::Handle<v8::internal::Name>) + 1012
            frame #3: 0x000000010069fc92 node`v8::internal::Runtime_LoadIC_Miss(int, v8::internal::Object**)
            frame #4: 0x00003501dbc040dd
            frame #5: 0x00003501dbceb5cc
            frame #6: 0x00003501dbc12f12
            frame #7: 0x00003501dbcf3fa1
            frame #8: 0x00003501dbc12f12
            frame #9: 0x00003501dbcf373c
            frame #10: 0x00003501dbc12f12
        ...
```

Пример 1. Фреймы 4 – 20 пусты

```
frame #3: 0x000000010069fc92 node`v8::internal::Runtime_LoadIC_Miss(int, , v8::internal::Isolate*) + 530
frame #4: 0x00003501dbc040dd
frame #5: 0x00003501dbceb5cc
frame #6: 0x00003501dbc12f12
frame #7: 0x00003501dbcf3fa1

...

frame #15: 0x00003501dbcf4242
frame #18: 0x00003501dbc12f12
frame #19: 0x00003501dbc10650
frame #20: 0x00003501dbce4ead
frame #32: 0x0000000100a4c308 node`node::Start(int, char**) + 333
frame #33: 0x0000000100001334 node`start + 52
```

javascript stack trace

Пример 1. Поиск метода сбросившего ошибку

```
$(llnode) v8 bt 20

        frame #4: 0x00003501dbc040dd <exit>
        frame #5: 0x00003501dbceb5cc <builtin>
        frame #6: 0x00003501dbc12f12 reserve(this=0x00000153bf0e84d1:<Object: ProductController>,
        0x0000163b0e223089:<Object: IncomingMessage>, 0x0000163b0e224e39:<Object: ServerResponse>) at
         /holyjs/controllers/ProductController.js:8:11 fn=0x000004c146ed3fa1
        frame #7: 0x00003501dbcf3fa1 <builtin>
        frame #8: 0x00003501dbc12f12 (anonymous)(this=0x0000163b0e223089:<Object: IncomingMessage>) at
         /holyjs/app.js:21:28 fn=0x0000163b0e227699
        frame #9: 0x00003501dbcf373c <builtin>
        frame #10: 0x00003501dbc12f12 emitNone(this=0x00000b9b56e02241:<undefined>,
0x0000163b0e2278a9:<Array:
       length=2>,0x00000b9b56e022f1:<false>, 0x0000163b0e223089:<Object: IncomingMessage>)
        at events.js:103:18 fn=0x00000f2606a87391
       ....
```

Пример 1. Исходный код метода reserve(req,res)

```
(llnode) v8 inspect --print-source 0x4c146ed3fa1

0x000004c146ed3fa1:<function: reserve at   /holyjs/controllers/ProductController.js:8:11
 source:
      const { id, storeId } = req.cookies['profile'];
      const rewards = req.cookies['rewards'];
      const { products } = req.body;

      //rewards is UNDEFINED
      this._reservationService.reserve(id, storeId, rewards.id, products)
        .then(data => {
              return res.send(data);
          }
        );
```

Пример 1. Метод ProductController.reserve(req,res)

```
$(llnode) v8 bt 20

        frame #4: 0x00003501dbc040dd <exit>
        frame #5: 0x00003501dbceb5cc <builtin>
        frame #6: 0x00003501dbc12f12 reserve(this=0x00000153bf0e84d1:<Object: ProductController>,
        0x0000163b0e223089:<Object: IncomingMessage>, 0x0000163b0e224e39:<Object: ServerResponse>) at
         /holyjs/controllers/ProductController.js:8:11 fn=0x000004c146ed3fa1
        frame #7: 0x00003501dbcf3fa1 <builtin>
        frame #8: 0x00003501dbc12f12 (anonymous)(this=0x0000163b0e223089:<Object: IncomingMessage>) at
         /holyjs/app.js:21:28 fn=0x0000163b0e227699
        frame #9: 0x00003501dbcf373c <builtin>
        frame #10: 0x00003501dbc12f12 emitNone(this=0x00000b9b56e02241:<undefined>,
0x0000163b0e2278a9:<Array:
       length=2>,0x00000b9b56e022f1:<false>, 0x0000163b0e223089:<Object: IncomingMessage>)
        at events.js:103:18 fn=0x00000f2606a87391
       ....
```

Пример 1. Исследуем объект req

```
$(llnode) v8 inspect 0x163b0e223089

        0x0000163b0e223089:<Object: IncomingMessage properties {
          ._readableState=0x0000163b0e223179:<Object: ReadableState>,
          ._events=0x0000163b0e223739:<Object: Object>,
          .socket=0x0000163b0e21e4e1:<Object: Socket>,
          .complete=0x00000b9b56e022c1:<true>,
          .cookies=0x00005c4b0e863755:<Object: Object>,
          .headers=0x0000163b0e223d11:<Object: Object>,
          .(external)=0x0000163b0e223069:<String: "/product/reserve">,
          .method=0x000004c146ed42a1:<String: "POST">,
          .statusCode=0x00000b9b56e02211:<null>,
          .statusMessage=0x00000b9b56e02211:<null>,
          .body=0x0000163b0e22aee1:<Object: Object>}>
        …
```

Пример 1. Читаем состояние объекта req.body

```
$(llnode) v8 inspect 0x0000163b0e22aee1

        0x0000163b0e22aee1:<Object: Object properties {
          .products=0x00003d76d2184e41<Array: length=3 >}>
```

Пример 1. Получаем список переданных продуктов

```
$(llnode) v8 inspect 0x0000163b0e22aee1

        0x0000163b0e22aee1:<Object: Object properties {
          .products=0x00003d76d2184e41<Array: length=3 >}>


$(llnode) v8 inspect 0x00003d76d2184e41

        0x00003d76d2184e41:<Array: length=3 {
          [0]=0x00003d76d218c2a1:<String: "1234">,
          [1]=0x0000435cd635c665:<String: "3456">,
          [2]=0x00003d76d21993b9:<String: "5678">}>
```

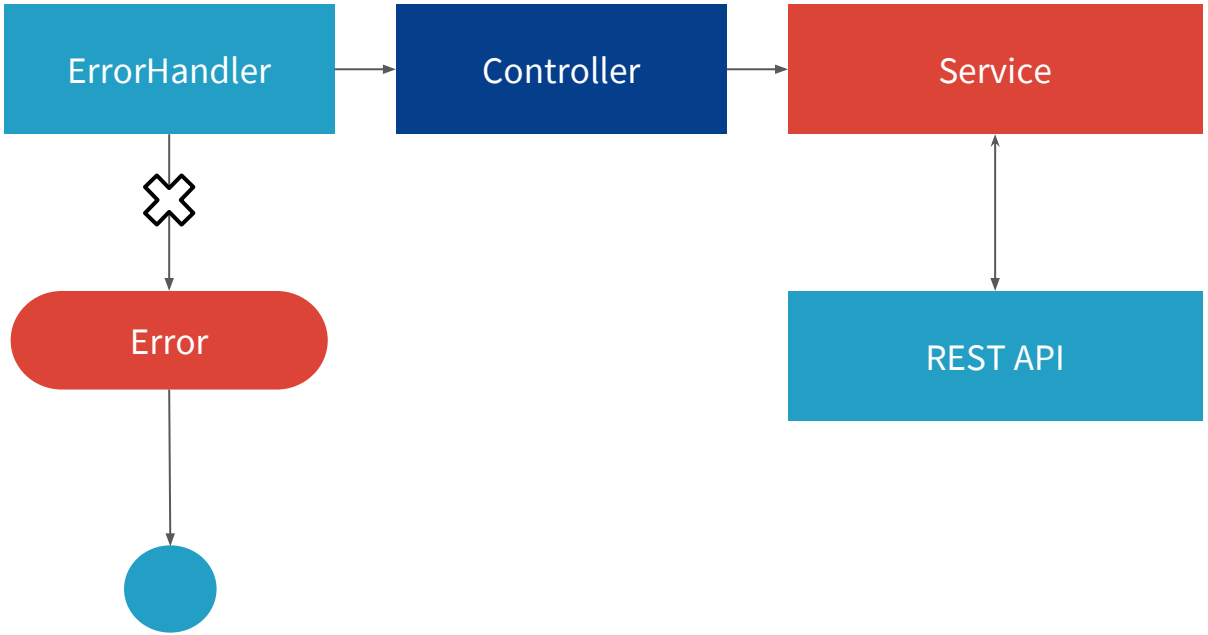
Пример 1. Находим свойство объекта req.cookies

```
$(llnode) v8 inspect 0x00005c4b0e863755

        0x00005c4b0e863755:<Object: Object properties {
          .profile=0x00003d76d2184b41:<Object: Object>,
          .rewards=0x0000163b0e22af51:<undefined>}>
```

Пример 1. Получаем user profile из cookie

```
$(llnode) v8 inspect 0x00005c4b0e863755

    0x00005c4b0e863755:<Object: Object properties {
      .profile=0x00003d76d2184b41:<Object: Object>,
      .rewards=0x0000163b0e22af51:<undefined>}>

$(llnode) v8 inspect 0x00003d76d2184b41

    0x00005c4b0e863755:<Object: Object properties {
      .id=0x00003d76d21993b9:<String: "10">,
      .storeId=0x00003d76d21993b9:<String: "99">}>
```

Отладка Express приложения на production



Пример 2. Unhandled rejection

```
reserve (req, res, next) {
    const { id, storeId } = req.cookies['profile'];
    const rewards = req.cookies['rewards'];
    const products = req.body;

    this._reservationService.reserve(id, storeId, rewards.id, products)
        .then(data => {
            return res.send(data);
        });
        // NO CATCH block is here
}
```

There is an Error inside service logic

Пример 2. Перехват события “unhandledRejection”

```
reserve (req, res, next) {
    const { id, storeId } = req.cookies['profile'];
    const rewards = req.cookies['rewards'];
    const products = req.body;

    this._reservationService.reserve(id, storeId, rewards.id, products)
        .then(data => {
            return res.send(data);
        });
         // NO CATCH block is here
}
```

```
process.on('unhandledRejection', (reason, p) => {
      logger.error('Unhandled Rejection at:', p, 'reason:', reason);
      process.abort();
});
```

Пример 2. Поиск функции сбросившей ошибку

```
$(llnode) v8 bt

        frame #8: 0x00002262d63840dd <exit>
        frame #9: 0x00002262d64735e9 <builtin>
        frame #10: 0x00002262d6392f12 process.on(this=0x000018901b589ec1:<Object: process>,
         0x000027a40a5f15d9:<unknown>, 0x000027a40a5f11a9:<unknown>) at /holyjs/app.js:40:34
         fn=0x00000af20adefc91
        frame #11: 0x00002262d6472cc8 <builtin>
        frame #12: 0x00002262d6392f12 emitTwo(this=0x00003e5911182241:<undefined>,
         0x00000af20adefc91:<function: process.on at /holyjs/app.js:40:34>, 0x00003e59111822c1:<true>,
         0x000018901b589ec1:<Object: process>, 0x000027a40a5f15d9:<unknown>, 0x000027a40a5f11a9:<unknown>)
         at events.js:123:17 fn=0x0000368a12a07421
        frame #13: 0x00002262d6473a25 <builtin>
        frame #14: 0x00002262d6392f12 emit(this=0x000018901b589ec1:<Object: process>,
         0x0000368a12a373e1:<String: "unhandledRejecti...">) at events.js:155:44 fn=0x00003ef321f07b19
```

Пример 2. Поиск экземпляров объекта req

```
$(llnode) v8 findjsinstances IncomingMessage
        ...
        0x00000af20adff4a1:<Object: IncomingMessage>
        0x000025c630d8fe69:<Object: IncomingMessage>
        0x000025c630d97861:<Object: IncomingMessage>
        0x000025c630d9d811:<Object: IncomingMessage>
        0x000025c630da9729:<Object: IncomingMessage>
        0x000025c630daf8d9:<Object: IncomingMessage>
        0x000025c630dbb669:<Object: IncomingMessage>
        0x000025c630dc1591:<Object: IncomingMessage>
        0x000025c630dd6c31:<Object: IncomingMessage>
        0x000025c630ddcb59:<Object: IncomingMessage>
        0x000025c630de2ed1:<Object: IncomingMessage>
        0x000025c630de8d01:<Object: IncomingMessage>
        0x000025c630deec29:<Object: IncomingMessage>
        …
```

N

Пример 2. Поиск headers по X-Transaction-ID

```
$(llnode) v8 findrefs --string "6af9c8d5-1b18-48b0-86fc-b9aac43b3cd6"

        0x25c630deef01: Object.X-Transaction-ID=0x25c630deea51 '6af9c8d5-1b18-48b0-86fc-b9aac43b3cd6'
```

Пример 2. Проверяем объект headers

```
$(llnode) v8 findrefs --string "6af9c8d5-1b18-48b0-86fc-b9aac43b3cd6"

        0x25c630deef01: Object.X-Transaction-ID=0x25c630deea51 '6af9c8d5-1b18-48b0-86fc-b9aac43b3cd6'

$(llnode)  v8 inspect 0x25c630deef01

        .X-Transaction-ID=0x25c630deea51:<String: "6af9c8d5-1b18-48b0-86fc-b9aac43b3cd6">,
        .host=0x000027a40a5bbb11:<String: "localhost:3000/product/reserve">,
        .user-agent=0x000027a40a5bbb61:<String: "ApacheBench/2.3">,
        .accept=0x000027a40a5bbba9:<String: "*/*">}>
```

Пример 2. Поиск объекта req, ссылающегося на headers

```
$(llnode) v8 findrefs --string "6af9c8d5-1b18-48b0-86fc-b9aac43b3cd6"

        0x25c630deef01: Object.X-Transaction-ID=0x25c630deea51 '6af9c8d5-1b18-48b0-86fc-b9aac43b3cd6'

$(llnode)  v8 inspect 0x25c630deef01

        .X-Transaction-ID=0x25c630deea51:<String: "6af9c8d5-1b18-48b0-86fc-b9aac43b3cd6">,
        .host=0x000027a40a5bbb11:<String: "localhost:3000">,
        .user-agent=0x000027a40a5bbb61:<String: "ApacheBench/2.3">,
        .accept=0x000027a40a5bbba9:<String: "*/*">}>

$(llnode) v8 findrefs --value 0x25c630deef01
        0x27a40a5bbce9: IncomingMessage.headers=0x25c630deef01
```

Поиск локальных переменных

```
module.exports = class ReservationService {
    reserve (id, storeId, rewardsId, products) {

      let user = this._restService.getUser(id);

      let store = this._db.getStore(storeId);

      let someLocalValue = {
          id: 'ff104cde3452332e0cc6',
          userAccount: user,
          preferredStore: store
      };

      return this._restService.reserve(someLocalValue, products);
}};
```

v8 findrefs --string "'ff104cde3452332e0cc6'"

v8 findrefs --name "userAccount"

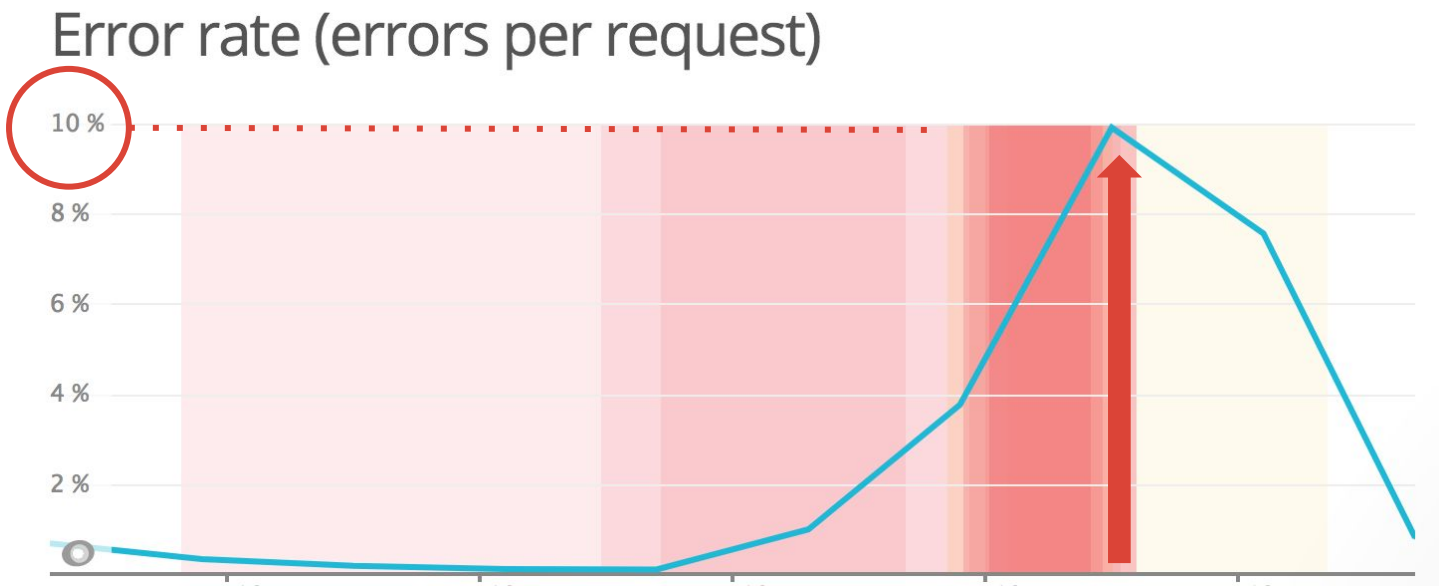
v8 findrefs --name "preferredStore"

Поиск объектов в памяти по:

- адресу
- типу
- по имени свойсв
- по значению

Ошибки в production

Ошибки в staging/production.



Core dump?

Core dump?

Production debugging

Core dump?

Production debugging

Руководство

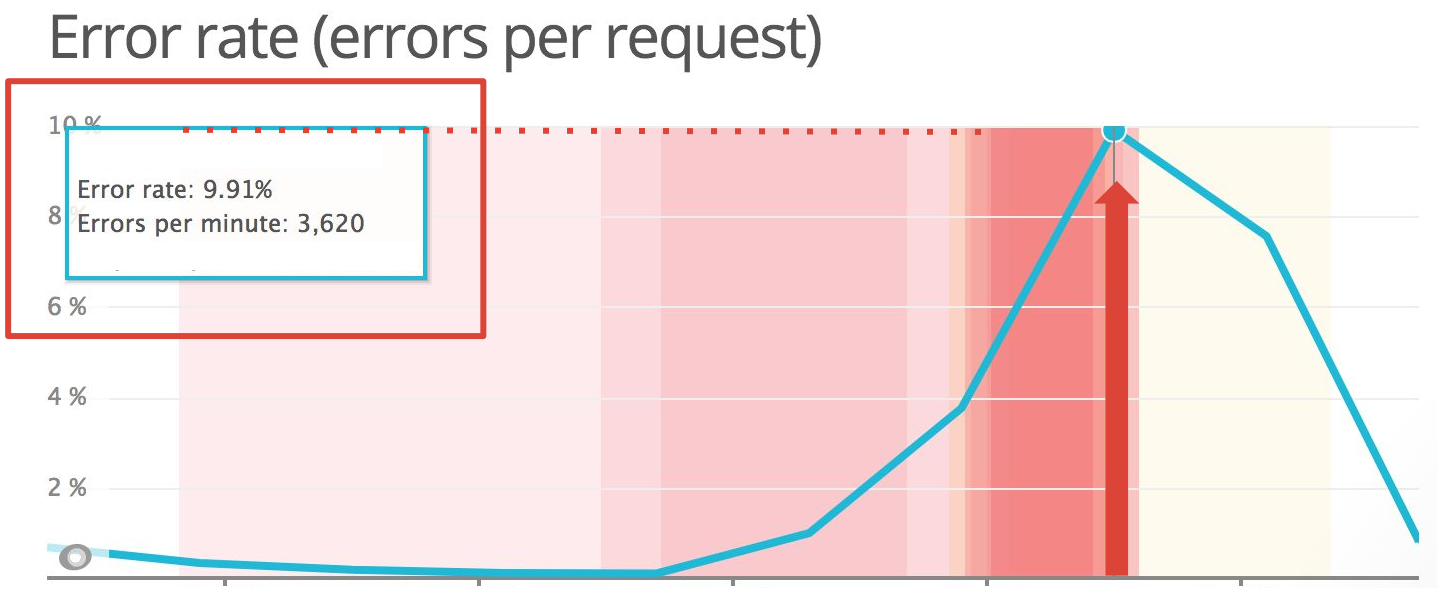
Core dump?

Production debugging

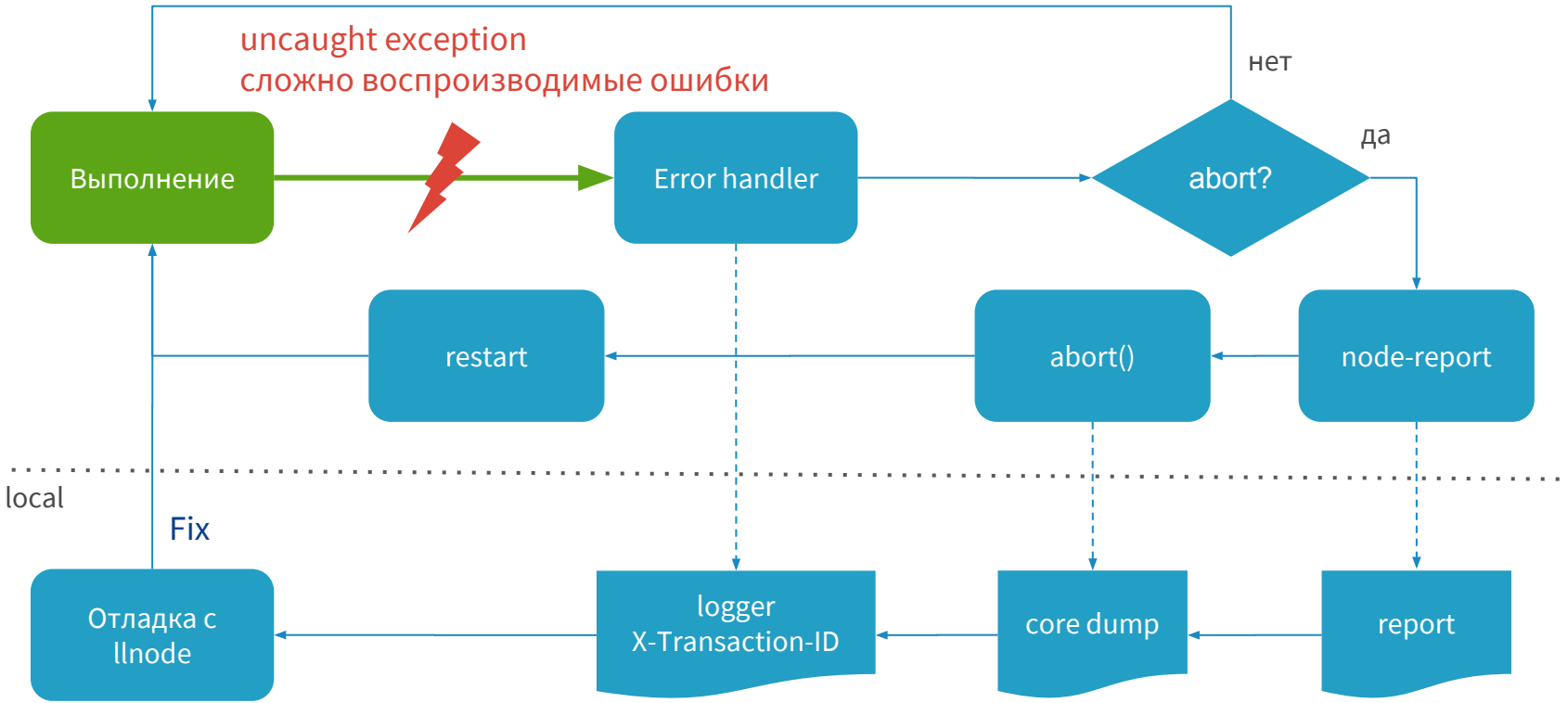
Руководство

Отключить генерацию core dump!
ulimit -c 0

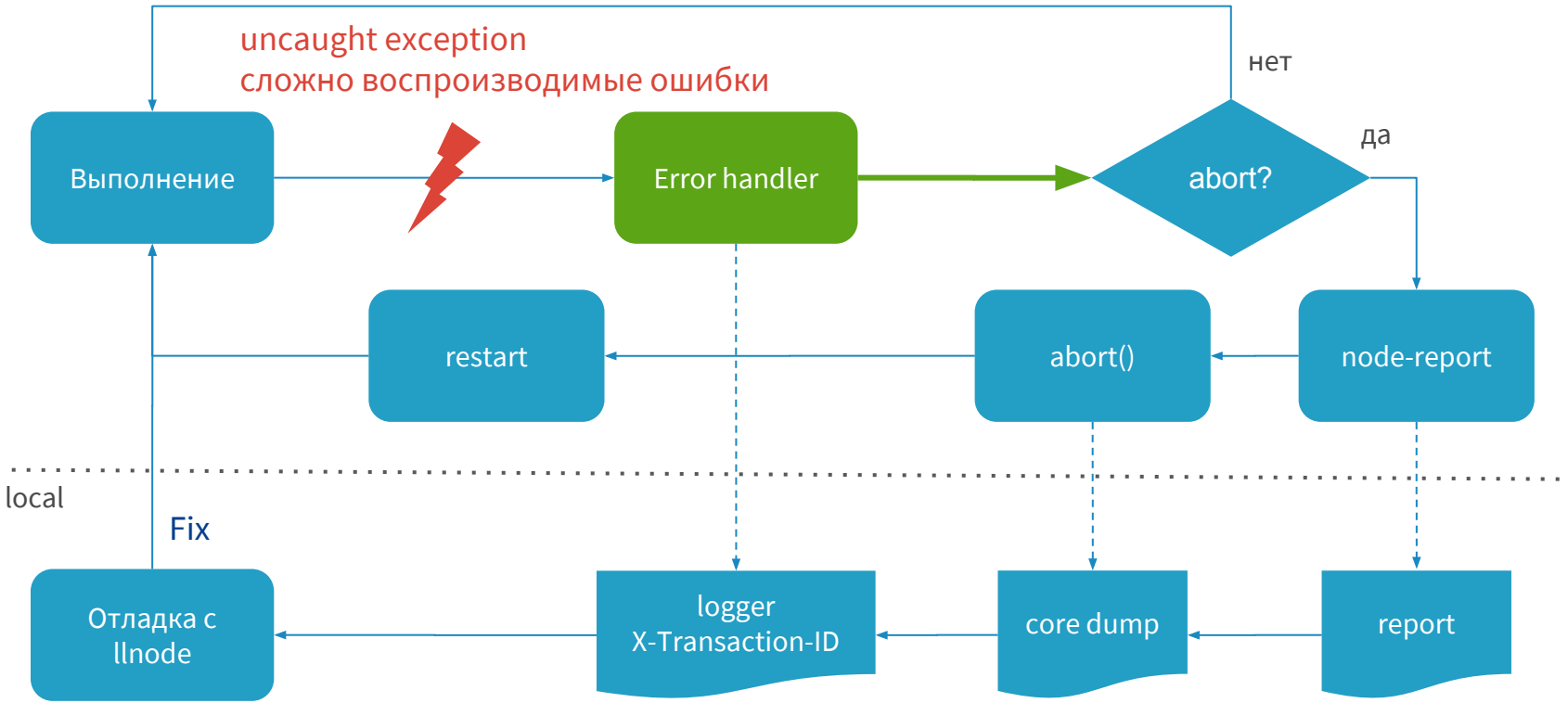
Частота ошибок в production



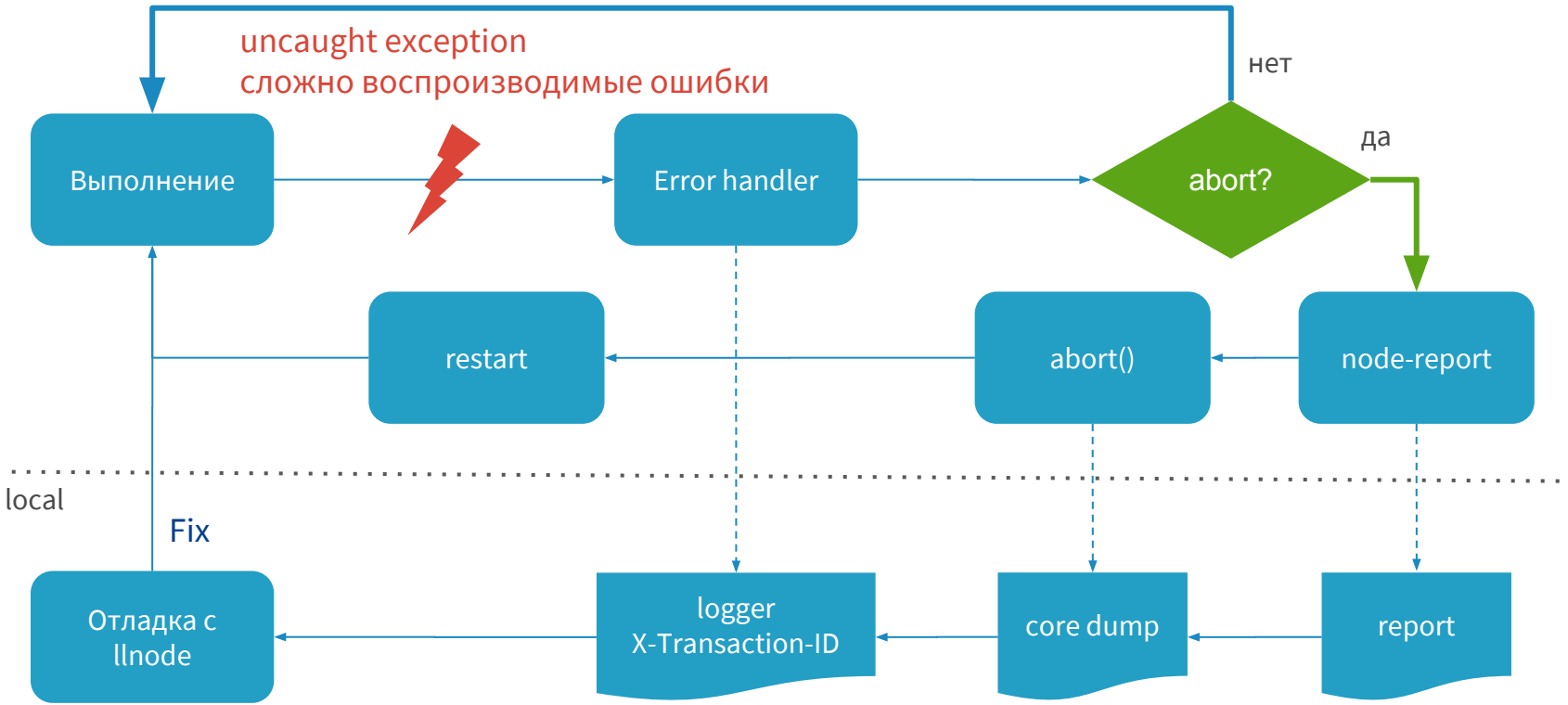
Алгоритм отладки в staging/production среде



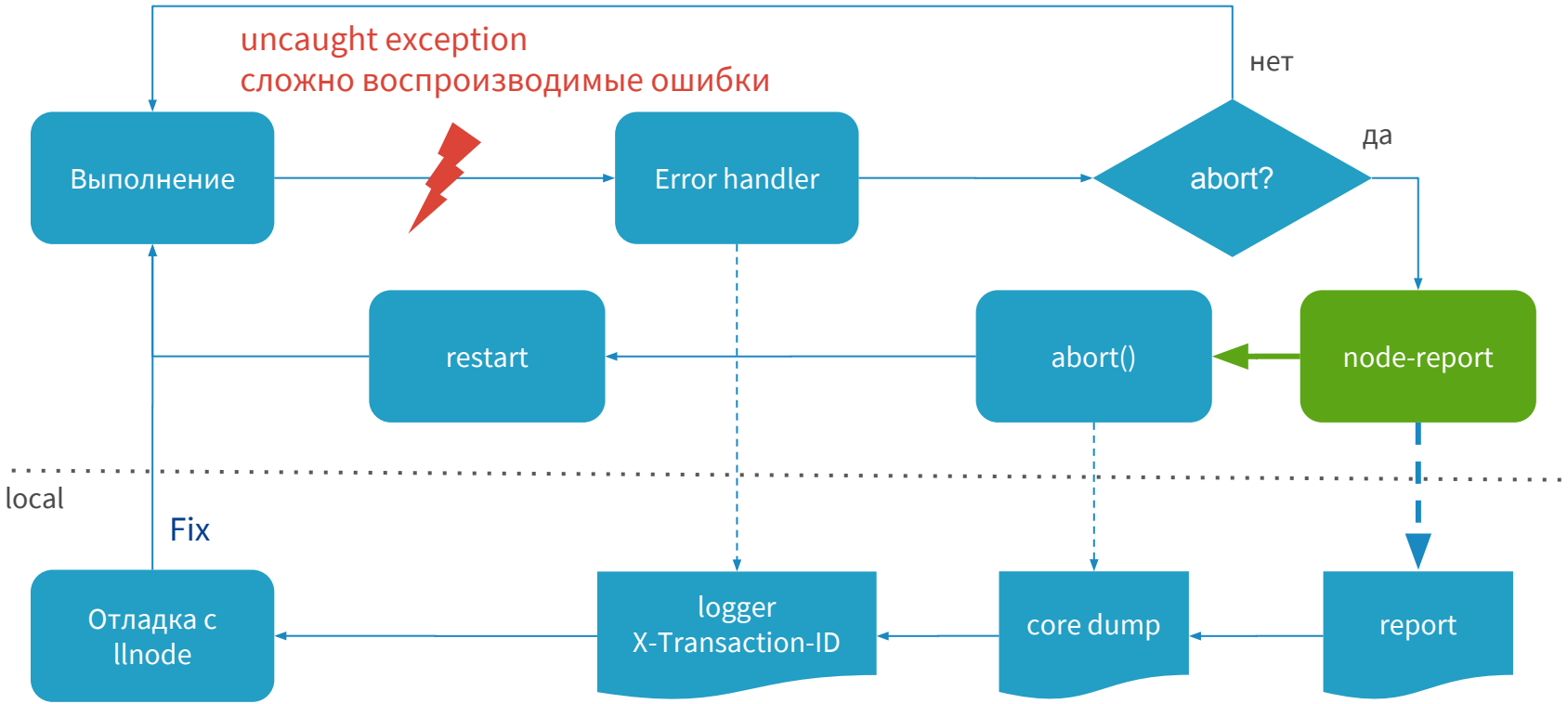
Алгоритм отладки в staging/production среде



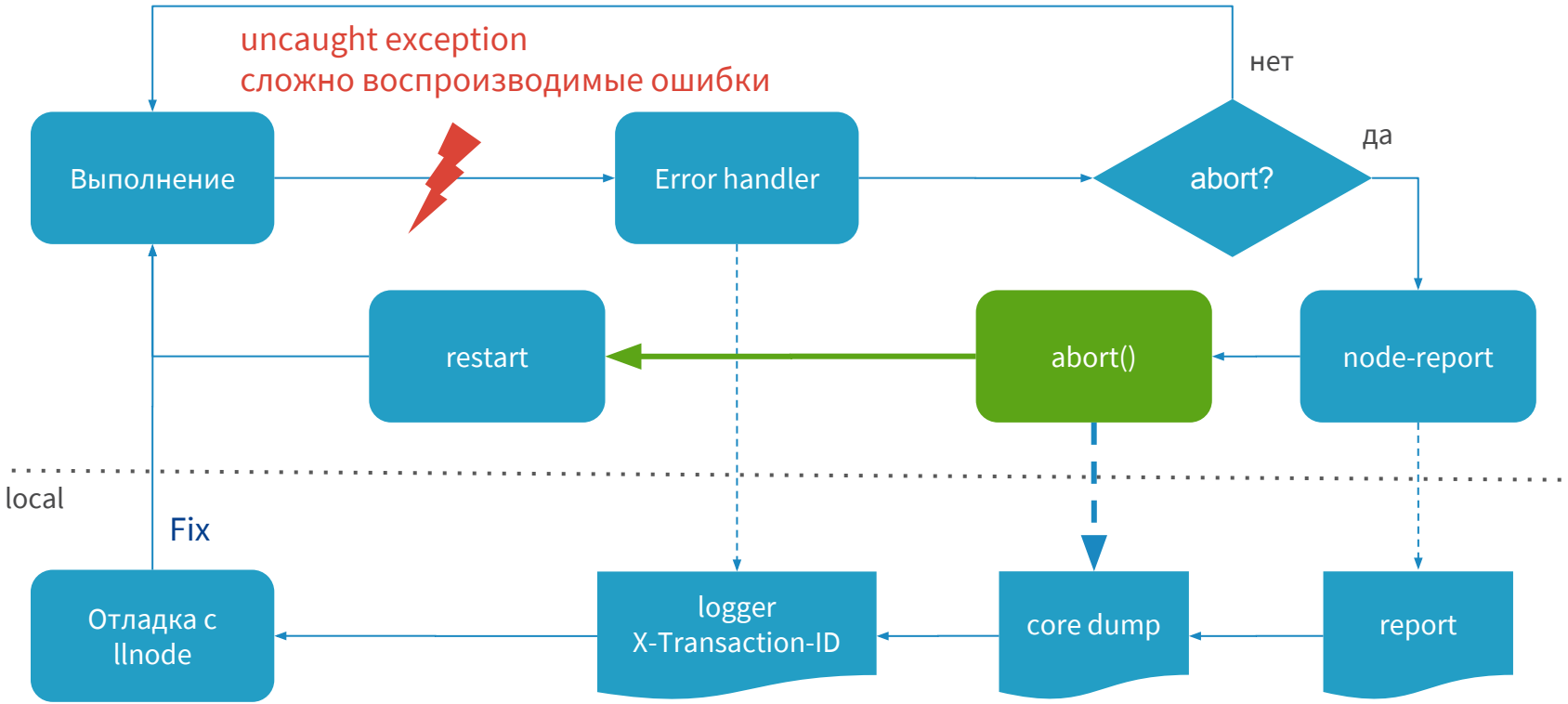
Алгоритм отладки в staging/production среде



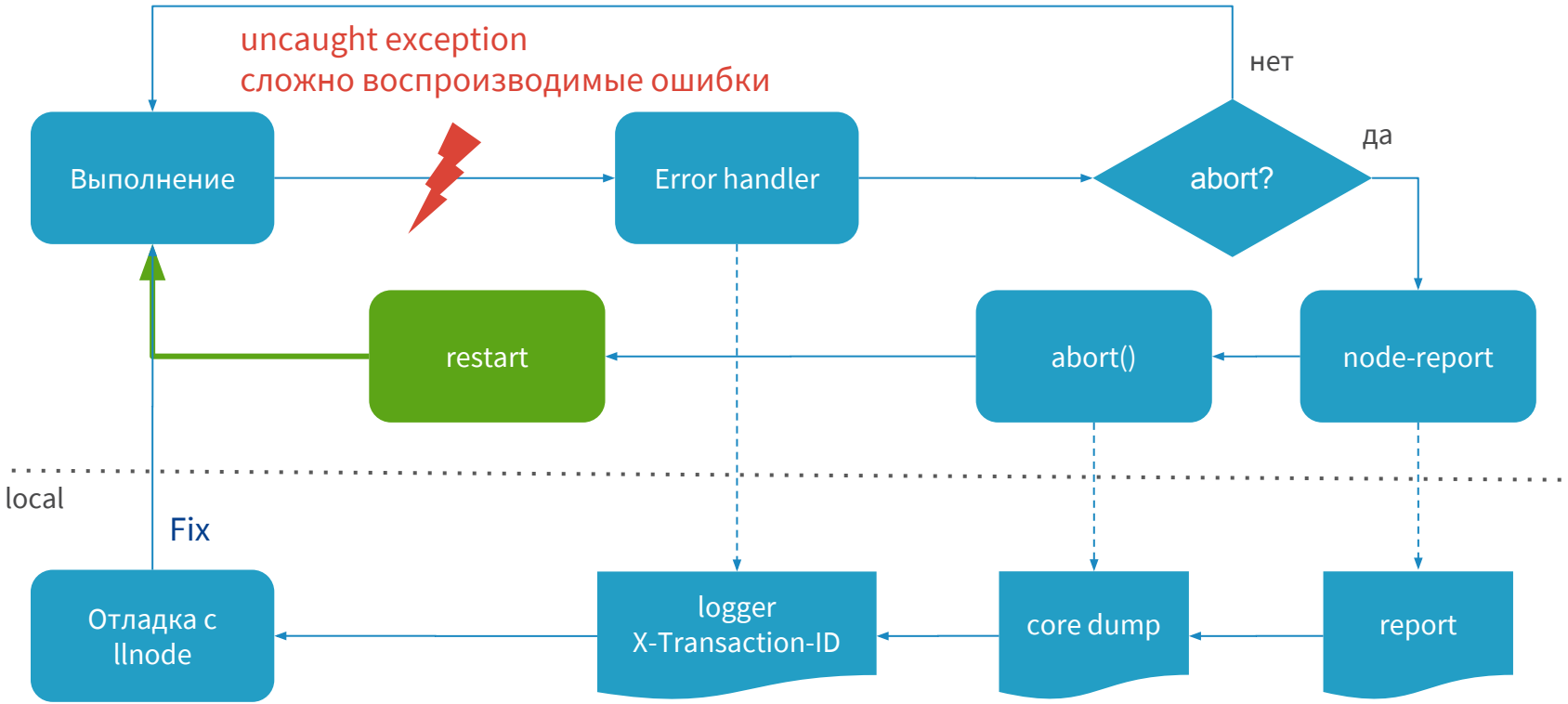
Алгоритм отладки в staging/production среде



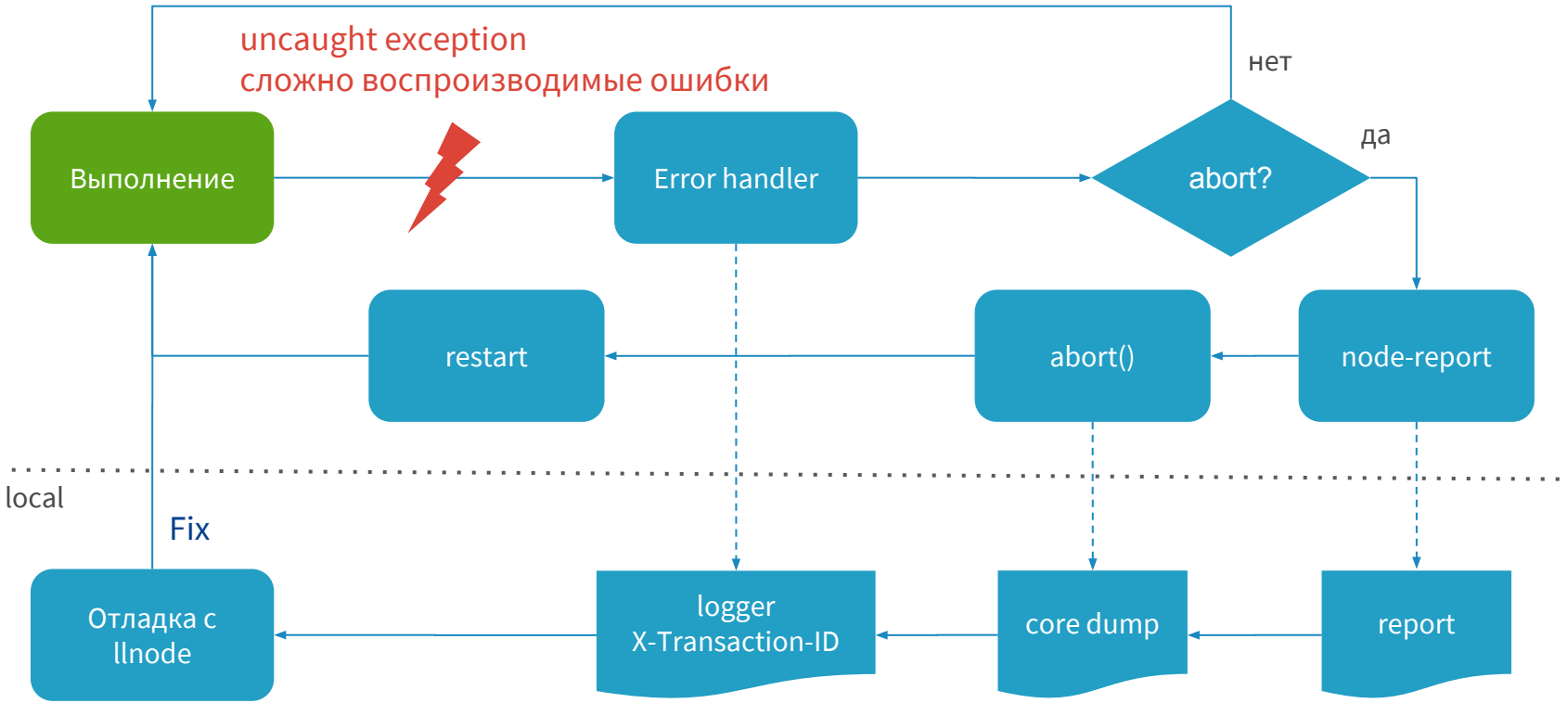
Алгоритм отладки в staging/production среде



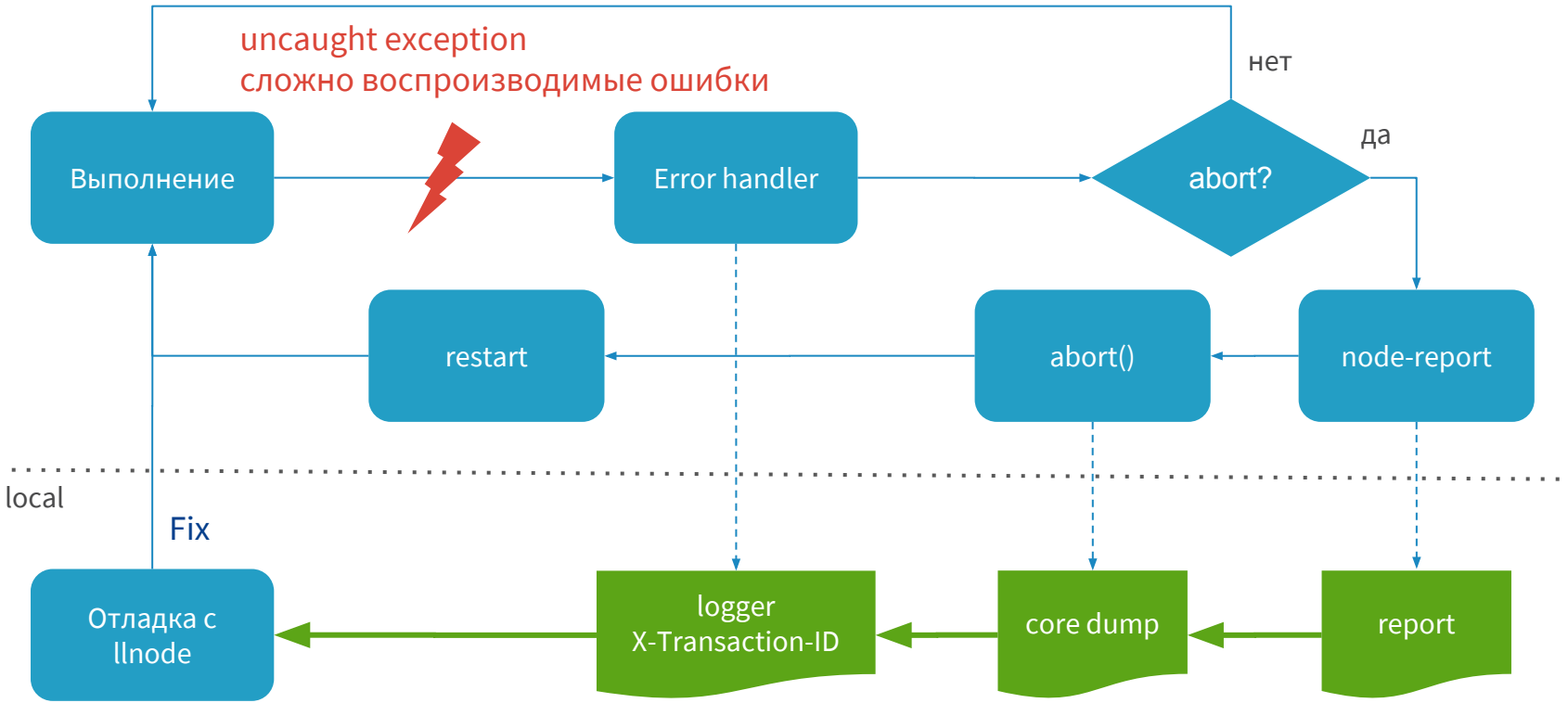
Алгоритм отладки в staging/production среде



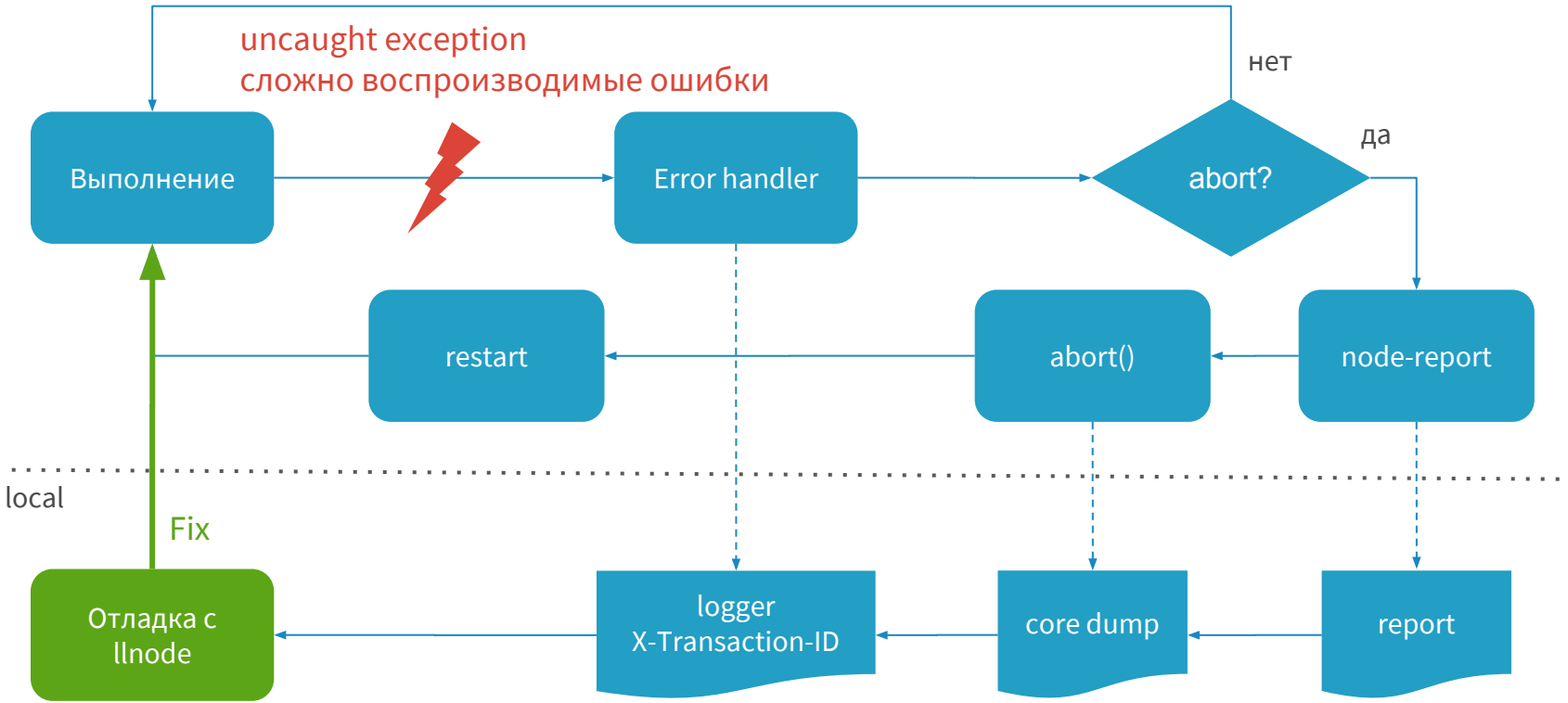
Алгоритм отладки в staging/production среде



Алгоритм отладки в staging/production среде



Алгоритм отладки в staging/production среде



Реестр ошибок. Ограничение создания core dump

```
const errorRegistry = new ErrorRegistry(redisClient);

app.use( function errorHandlerMiddleware (err, req, res, next) {
  errorRegistry.add(err, req)
      .then(result => {
        if (result) { // was error added to registry

          nodeReport.triggerReport(err);
          process.abort();

        } else { // already exists or hadn't to
          logger.error(err);
          res.status(500).send('Something broke!');
          next();
        }
      })
      .catch(err => {  … // handleError(err, req, res, next);  })
});
```

node-report

```
Event: exception, location: "OnUncaughtException"
Dump event time:  2017/09/27 00:57:26
Process ID: 35866
==== JavaScript Stack Trace ========================================
Object.fs.readFileSync (fs.js:1:1)
ProductController.reserve (/holyjs/controllers/ProductController.js:1:1)
==== Native Stack Trace ============================================
 0: [pc=0x1023eb7b1] nodereport::OnUncaughtException(v8::Isolate*)
 1: [pc=0x1006da003] v8::internal::Isolate::Throw(v8::internal::Object*)
==== JavaScript Heap and Garbage Collector =========================
Heap space name: new_space
    Memory size: 4,194,304 bytes, committed memory: 2,116,048 bytes
    Capacity: 2,062,336 bytes, used: 824,392 bytes, available: 1,237,944 byte
Heap memory limit: 1,501,560,832
```

Что?

Где?

Когда?

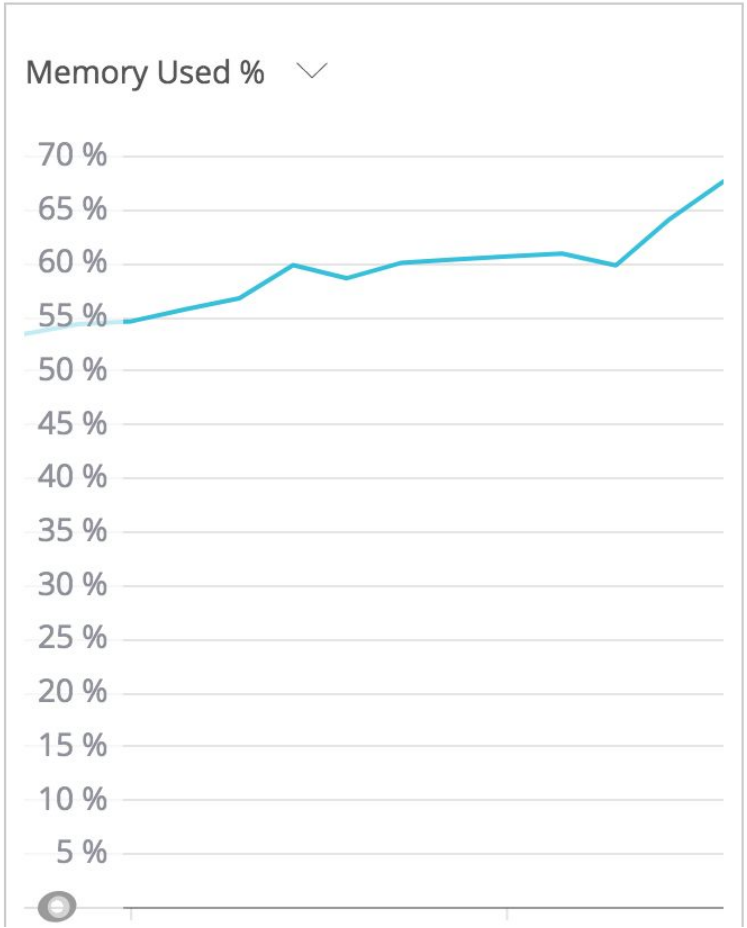
Прогресс

1. Отладка неисправностей приложений в production ✔
2. Поиск утечек памяти
3. Профилирование производительности приложения

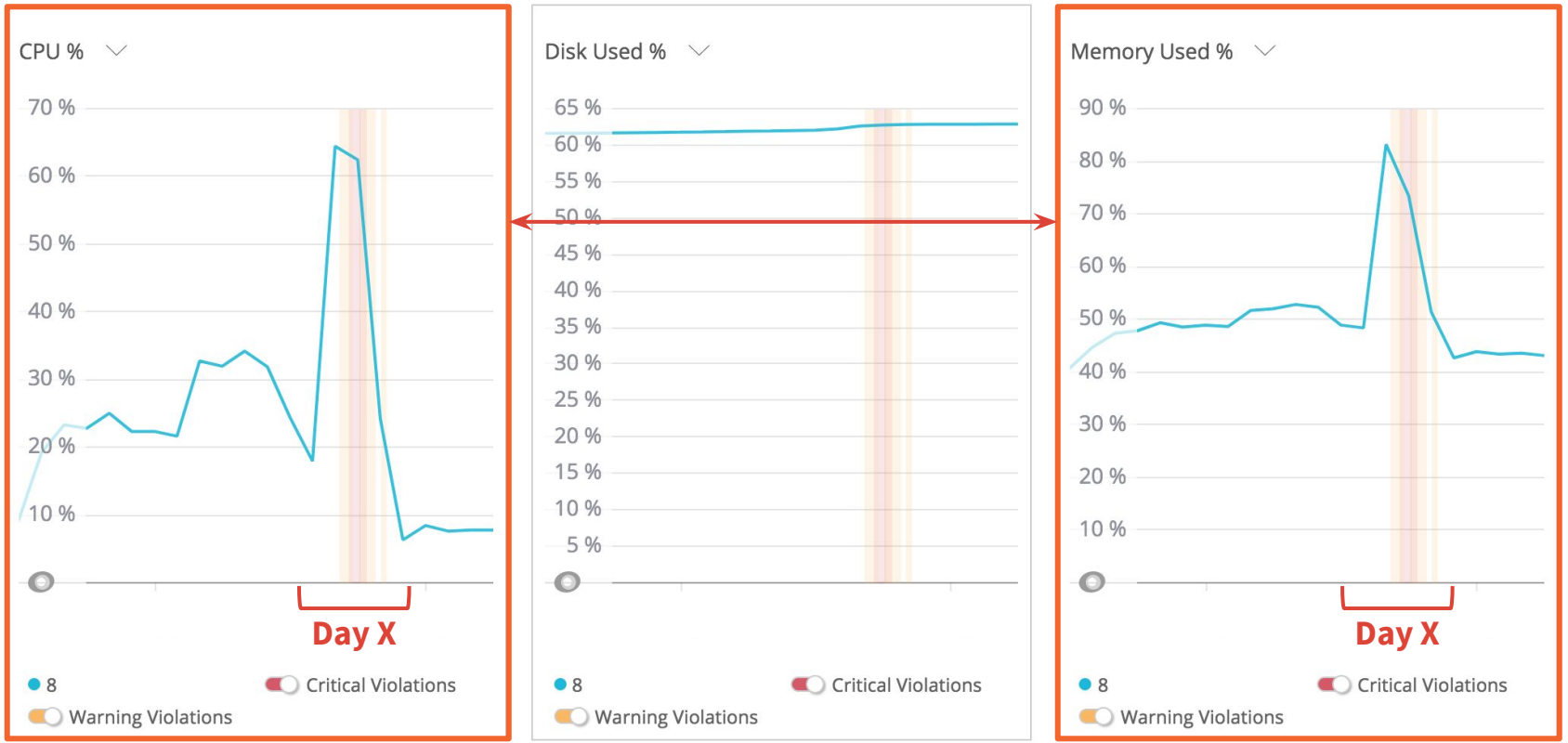
2. Утечки памяти

Способы выявления:

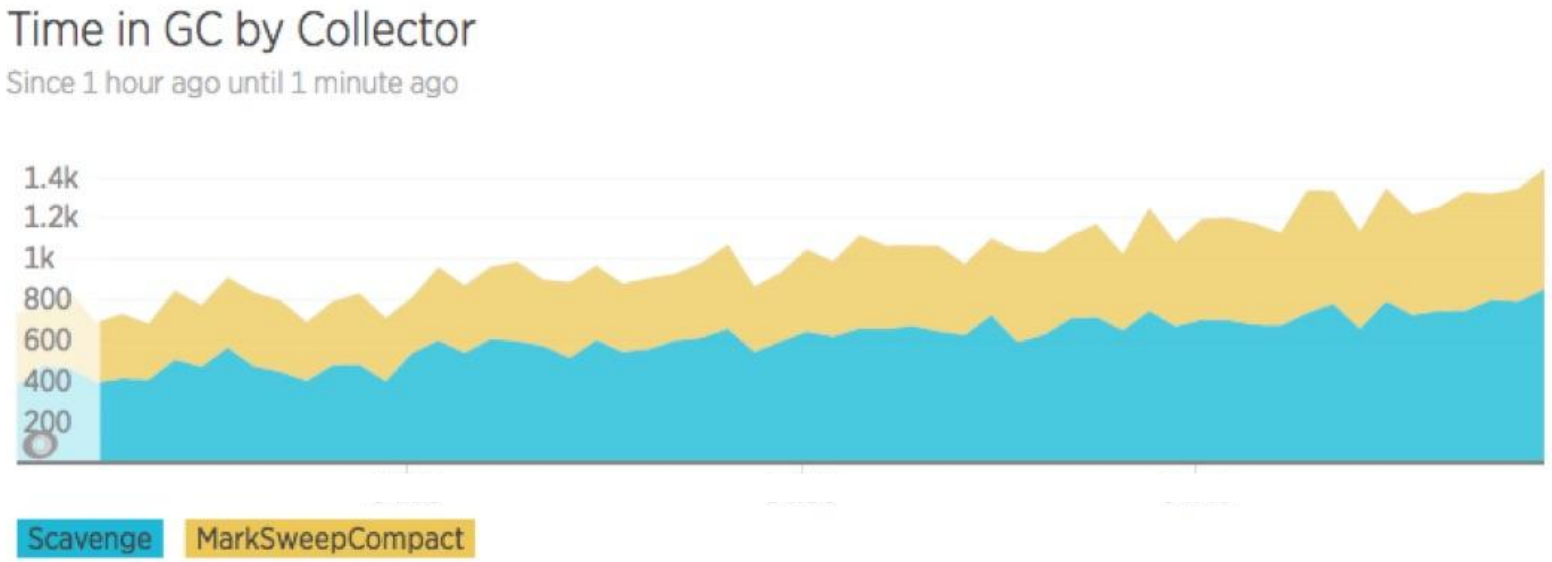
- Системы мониторинга приложений
- DTrace, Instruments
- node-memwatch
- Трассировка GC
- Valgrind для C++ модулей
- Fatal error
- и другие



Взаимосвязь CPU и Memory



Количество сборок мусора: Scavenge и MarkSweepCompact GC



Трассировка событий GC

```
node --trace_gc --trace_gc_verbose app.js > gc-trace.log
```

пространства памяти

```
[48933:0x102800c00]   291220 ms: Scavenge 317.4 (354.8) -> 302.7 (355.8) MB, 2.3 / 0.0 ms [allocation failure].
[48933:0x102800c00] Memory allocator,   used: 364368 KB, available: 1102000 KB
[48933:0x102800c00] New space,          used:    580 KB, available:  15539 KB, committed:  32768 KB
[48933:0x102800c00] Old space,          used: 146356 KB, available:   3772 KB, committed: 153556 KB
[48933:0x102800c00] Code space,         used:   3764 KB, available:   2482 KB, committed:   7168 KB
[48933:0x102800c00] Map space,          used:   2965 KB, available:      0KB, committed:   9296 KB
[48933:0x102800c00] Large object space, used: 156288 KB, available: 1100959 KB, committed: 158508 KB
[48933:0x102800c00] All spaces,         used: 309955 KB, available: 1122754 KB, committed: 361296 KB
[48933:0x102800c00] External memory reported:    795 KB
[48933:0x102800c00] Total time spent in GC  : 10568.3 ms
```

Популярные причины утечек памяти

Частые сборки Scavenge GC в New Space нагружают CPU

- Дорогие операции клонирования и слияния объектов
- Частое создание новых объектов

Old Space

- Замыкания
- Забытые таймеры

Large Object Space

- JSON сериализация/десериализация

Трассировка Scavenge GC. New Space.

172047 ms: Scavenge 546.9 (**592.6**) -> 538.4 (**592.6**) MB, **1.5** / 0.0 ms [allocation failure].

172093 ms: Scavenge 547.8 (**593.6**) -> 534.0 (**593.6**) MB, **2.8** / 0.0 ms [allocation failure].

172136 ms: Scavenge 549.1 (**594.6**) -> 535.0 (**595.6**) MB, **2.2** / 0.0 ms [allocation failure].

172202 ms: Scavenge 550.4 (**595.6**) -> 536.3 (**596.6**) MB, **3.3** / 0.0 ms (+ 14.3 ms in 206 steps since last GC) [allocation failure].

172269 ms: Scavenge 551.6 (**596.6**) -> 537.1 (**597.6**) MB, **3.9** / 0.0 ms (+ 15.5 ms in 238 steps since last GC) [allocation failure].

172335 ms: Scavenge 552.9 (**598.6**) -> 538.6 (**598.6**) MB, **3.9** / 0.0 ms (+ 14.2 ms in 243 steps since last GC) [allocation failure].

Трассировка Scavenge GC. New Space.

256201 ms: Scavenge 824.5 (**1014.4**) -> 811.4 (**1019.2**) MB, **16.9** / 0.0 ms [allocation failure].

256245 ms: Scavenge 838.8 (**1033.3**) -> 827.7 (**1039.7**) MB, **17.6** / 0.0 ms [allocation failure].

256296 ms: Scavenge 852.8 (**1075.9**) -> 842.9 (**1086.5**) MB, **11.0** / 0.0 ms [allocation failure].

256557 ms: Scavenge 869.8 (**1201.6**) -> 858.1 (**1209.6**) MB, **43.3** / 0.0 ms (**+ 86.0 ms** in 176 steps since last GC) [allocation failure].

256625 ms: Scavenge 884.1 (**1232.8**) -> 871.9 (**1239.2**) MB, **36.5** / 0.0 ms (**+ 56.7 ms** in 168 steps since last GC) [allocation failure].

Трассировка Mark-sweep GC

168372 ms: Mark-sweep 493.7 (**558.6**) -> 151.4 (**352.1**) MB, **90.3 ms** (+ 150.3 ms in 731 steps, biggest step 18.1 ms)

134838 ms: Mark-sweep 525.3 (**680.7**) -> 169.7 (**366.8**) MB, **91.2 ms** (+ 157.0 ms in 790 steps, biggest step 14.2 ms)

...

300609 ms: Mark-sweep 664.8 (**825.1**) -> 192.0 (**387.1**) MB, **110.4 ms** (+ 193.8 ms in 982 steps, biggest step 19.5 ms)

308164 ms: Mark-sweep 683.8 (**886.1**) -> 199.0 (**447.1**) MB, **120.6 ms** (+ 218.7 ms in 996 steps, biggest step 20.7 ms)

...

956609 ms: Mark-sweep 1164.8 (**1325.1**) -> 792.0 (**1007.1**) MB, **214.2 ms** (+ 296.8 ms in 1482 steps, biggest step 19.5 ms)

990164 ms: Mark-sweep 1183.8 (**1405.1**) -> 799.0 (**1124.1**) MB, **216.7 ms** (+ 307.7 ms in 1496 steps, biggest step 20.7 ms)

Трассировка Mark-sweep GC

168372 ms: Mark-sweep 493.7 (558.6) -> 151.4 (352.1) MB, 90.3 ms (**+ 150.3 ms** in **731 steps,** biggest step 18.1 ms)

134838 ms: Mark-sweep 525.3 (680.7) -> 169.7 (366.8) MB, 91.2 ms (**+ 157.0 ms** in **790 steps**, biggest step 14.2 ms)

300609 ms: Mark-sweep 664.8 (825.1) -> 192.0 (387.1) MB, 110.4 ms (**+ 193.8 ms** in **982 steps**, biggest step 19.5 ms)

308164 ms: Mark-sweep 683.8 (825.1) -> 199.0 (387.1) MB, 120.6 ms (**+ 218.7 ms** in **996 steps**, biggest step 20.7 ms)

956609 ms: Mark-sweep 1164.8 (1325.1) -> 792.0 (1007.1) MB, 214.2 ms (**+ 296.8 ms** in **1482 steps**, biggest step 19.5 ms)

990164 ms: Mark-sweep 1183.8 (1325.1) -> 799.0 (1016.1) MB, 216.7 ms (**+ 307.7 ms** in **1496 steps**, biggest step 20.7 ms)

Пример 3. profillingMiddleware

```
const { performance } = require('perf_hooks');

app.use(function profillingMiddleware (req, res, next) {
    performance.mark(`${id}-end`);

    res.on('finish', () => {
        performance.mark(`${id}-end`);
        performance.measure(`${id}--duration`, `${id}-end`, `${id}-end`);
        const measure = performance.getEntriesByName('req-duration')[0];

        someProcessOperation(req, measure);
    });
    next();
});
```

Пример 3. Объект req удержан после отправки res. Замыкание

```
const { performance } = require('perf_hooks');

app.use(function profillingMiddleware (req, res, next) {
    performance.mark(`${id}-end`);

    res.on('finish', () => {
        performance.mark(`${id}-end`);
        performance.measure(`${id}--duration`, `${id}-end`, `${id}-end`);
        const measure = performance.getEntriesByName('req-duration')[0];

        someProcessOperation(req, measure);
    });
    next();
});
```

Пример 3. Исправление

```
const { performance } = require('perf_hooks');

app.use(function profillingMiddleware (req, res, next) {
    const id = req.get('X-Transaction-ID');
    const url = req.url;
    performance.mark(`${id}-end`);
    res.on('finish', () => {
        performance.mark(`${id}-end`);
        performance.measure(`${id}--duration`, `${id}-end`, `${id}-end`);
        const measure = performance.getEntriesByName('req-duration')[0];

        someProcessOperation(id, url, measure);
    });
    next();
});
```

Поиск утечек памяти с lldb(llnode)

Как создать core dump процесса для замеров по требованию?

- Спровоцировать process.abort()?
- Использовать gcore?

Поиск утечек памяти с lldb(llnode)

Как создать core dump процесса для замеров по требованию?

- Спровоцировать process.abort()?
- Использовать gcore?

Ответ: Post Mortem Diagnostics Working Group работает над созданием данного API.

Альтернативное решение:

- Использовать gen-core

Создание dump по требованию с gencore

```
const gencore = require('gencore');

module.exports = class DiagnosticController {
  constructor () {  this._inProcess = false;  this._name = null; }
  getDump (req, res) {
    if (this._inProcess) {
      this._inProcess = true;
      res.status(200).send(`Core dump: in progress.\n`);
    } else {
      gencore.collectCore((error, name) => {
        console.log(`Core dump created at: ${name}`);
        this._name = name;
        this._inProcess = false;
      });
      res.status(201).send(`Core dump: ${this._name}\n`);  } } }
```

Анализ Heap с llnode.
Запрашиваем список всех js объектов

```
$(llnode) v8 findjsobjects

Instances  Total Size Name
---------- ---------- ----
       251      88216 ServerResponse
       251      58440 IncomingMessage
        40       9920 Socket
        42       9408 WritableState
        93      34016 ReadableState
        94      15528 BufferList
        86       6880 Module
       194      12504 TickObject
      3688     118016 (Array)
        90       7920 Timeout
     31043     369864 (String)
  …
```

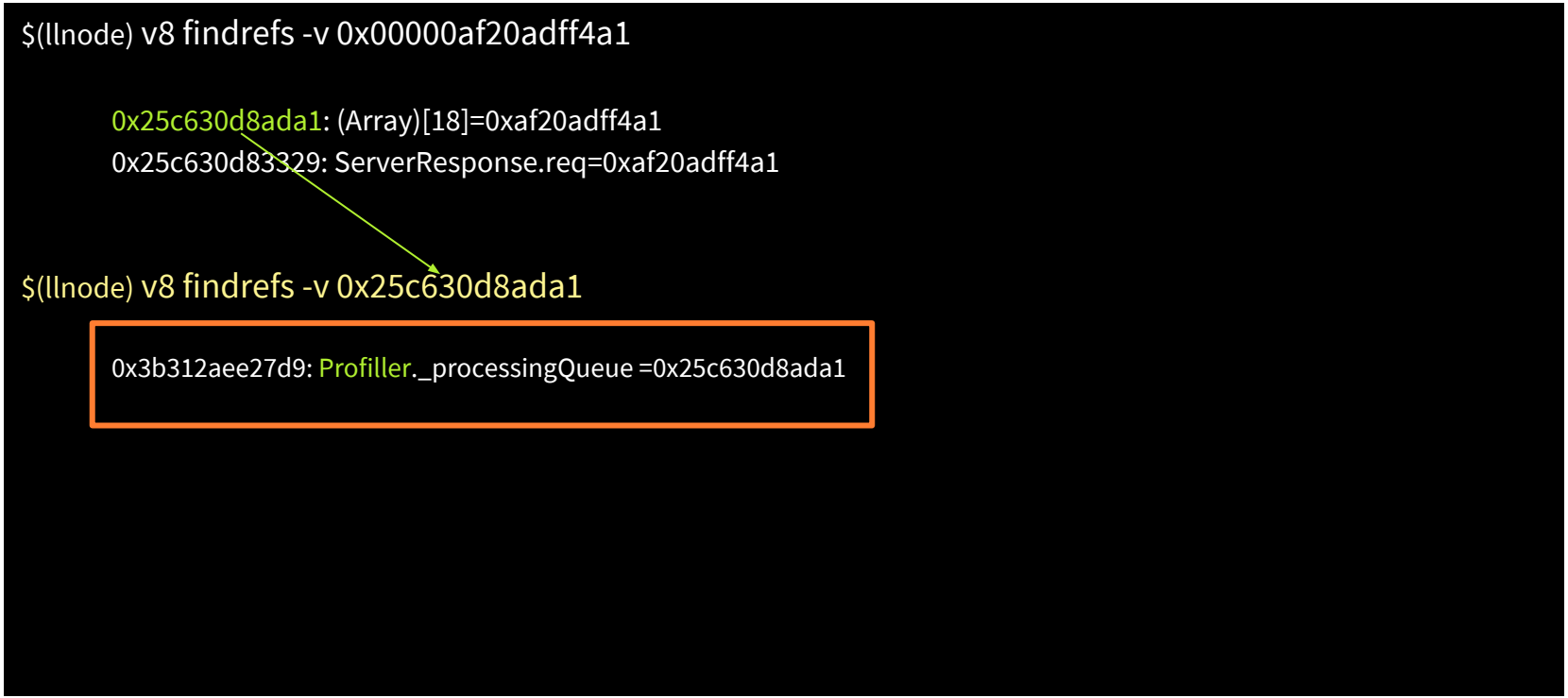
Поиск всех экземпляров IncommingMessage

```
$(llnode) v8 findjsinstances IncomingMessage

        0x00000af20adff4a1:<Object: IncomingMessage>
        0x000025c630d8fe69:<Object: IncomingMessage>
        0x000025c630d97861:<Object: IncomingMessage>
        0x000025c630d9d811:<Object: IncomingMessage>
        0x000025c630da37c9:<Object: IncomingMessage>
        0x000025c630daf8d9:<Object: IncomingMessage>
        0x000025c630db5741:<Object: IncomingMessage>
        0x000025c630dbb669:<Object: IncomingMessage>
        0x000025c630dc1591:<Object: IncomingMessage>
        0x000025c630dd6c31:<Object: IncomingMessage>
        0x000027a40a588a11:<Object: IncomingMessage>
        0x000027a40a58e841:<Object: IncomingMessage>
        0x000027a40a5a38f1:<Object: IncomingMessage>
        0x000027a40a5a9819:<Object: IncomingMessage>
```

Поиск держателя объекта req

```
$(llnode) v8 findrefs -v 0xaf20adff4a1

    0x25c630d8ada1: (Array)[18]=0xaf20adff4a1
    0x25c630d83329: ServerResponse.req=0xaf20adff4a1
```

Массив удерживающий память



Анализ Heap. Поиск Timers (Timeout)

```
$(llnode) v8 findjsobjects

 Instances  Total Size Name
 ---------- ---------- ----
        31       4216 ServerResponse
       251      58440 IncomingMessage
        40       9920 Socket
        42       9408 WritableState
        62       3968 NativeModule
        93      34016 ReadableState
        94      15528 BufferList
        86       6880 Module
       180       4320 CallSite
       194      12504 TickObject
      3688     118016 (Array)
        90       7920 Timeout
     31043     369864 (String)
     ...
```

Список всех экземпляров Timeout

```
$(llnode) v8 findjsinstances Timeout

        0x00003b312aedca29:<Object: Timeout>
        0x00003b312aeddb59:<Object: Timeout>
        0x00003b312aeddc79:<Object: Timeout>
        0x00003b312aeddd99:<Object: Timeout>
        0x00003b312aeddeb9:<Object: Timeout>
        0x00003b312aeddfd9:<Object: Timeout>
        0x00003b312aede1d9:<Object: Timeout>
        0x00003b312aede2b9:<Object: Timeout>
        0x00003b312aede399:<Object: Timeout>
        0x00003b312aede479:<Object: Timeout>
        0x00003b312aede559:<Object: Timeout>
        0x00003b312aede639:<Object: Timeout>
        0x00003b312aede719:<Object: Timeout>
        …
```

Список всех экземпляров Timeout

```
$(llnode) v8 i 0x00003b312aedca29

        0x00003b312aedca29:<Object: Timeout properties {
          ._called=0x000028d1ac0043c1:<true>,
          ._idleTimeout=<Smi: 1>,
          ._idlePrev=0x000028d1ac004201:<null>,
          ._idleNext=0x000028d1ac004201:<null>,
          ._idleStart=<Smi: 54>,
          ._onTimeout=0x00003b312aedc8c9:<function: setInterval at /projects/holyjs/tick.js:8:20>,
          ._timerArgs=0x000028d1ac004381:<undefined>,
          ._repeat=0x000028d1ac004201:<null>}>
```

Пример 4. Выявление утечек памяти на этапе проектирования с V8/D8 – Debug8

```
function foo() {
  let x = { bar: "bar"};

  %DebugTrackRetainingPath(x);

  return () => { return x; }
}

let closure = foo();

gc();
```

Путь до root объекта в Heap

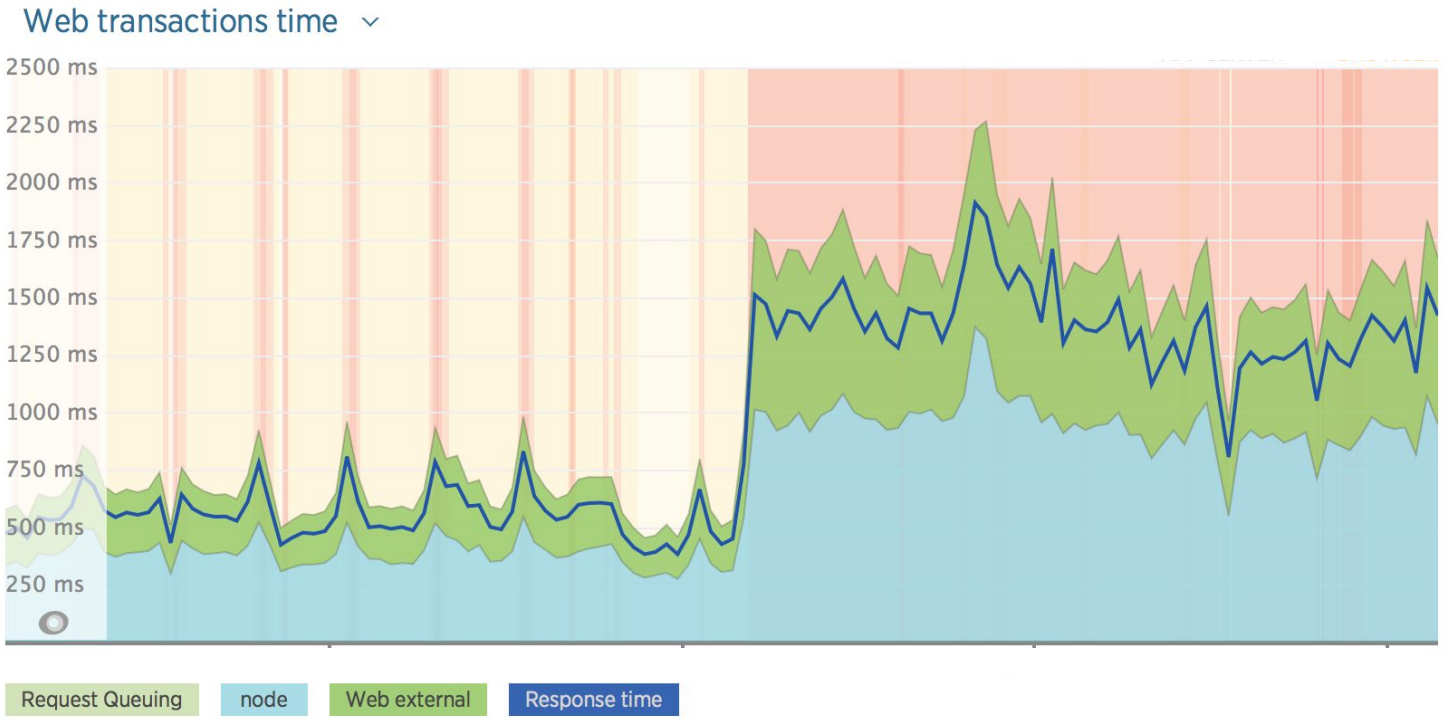
$ out/x64.release/**d8 --allow-natives-syntax --track-retaining-path --expose-gc** test.js

```
Retaining path for 0x245c59f0c1a1:

Distance from root 6: 0x245c59f0c1a1 <Object map = 0x2d919f0d729>

Distance from root 5: 0x245c59f0c169 <FixedArray[5]>

Distance from root 4: 0x245c59f0c219 <JSFunction (sfi = 0x1fbb02e2d7f1)>

Distance from root 3: 0x1fbb02e2d679 <FixedArray[5]

Distance from root 2: 0x245c59f0c139 <FixedArray[4]>

Distance from root 1: 0x1fbb02e03d91 <FixedArray[279]>

Root: (Isolate)
```

Прогресс

1. Отладка неисправностей приложений на production ✔
2. Поиск утечек памяти в приложении ✔
3. Профилирование производительности приложения

3. Ответ веб сервера Node.js
после релиза вырос с ~300 до ~1500 ms



Алгоритм

While: Не устраивает работоспособность приложения?

Do:

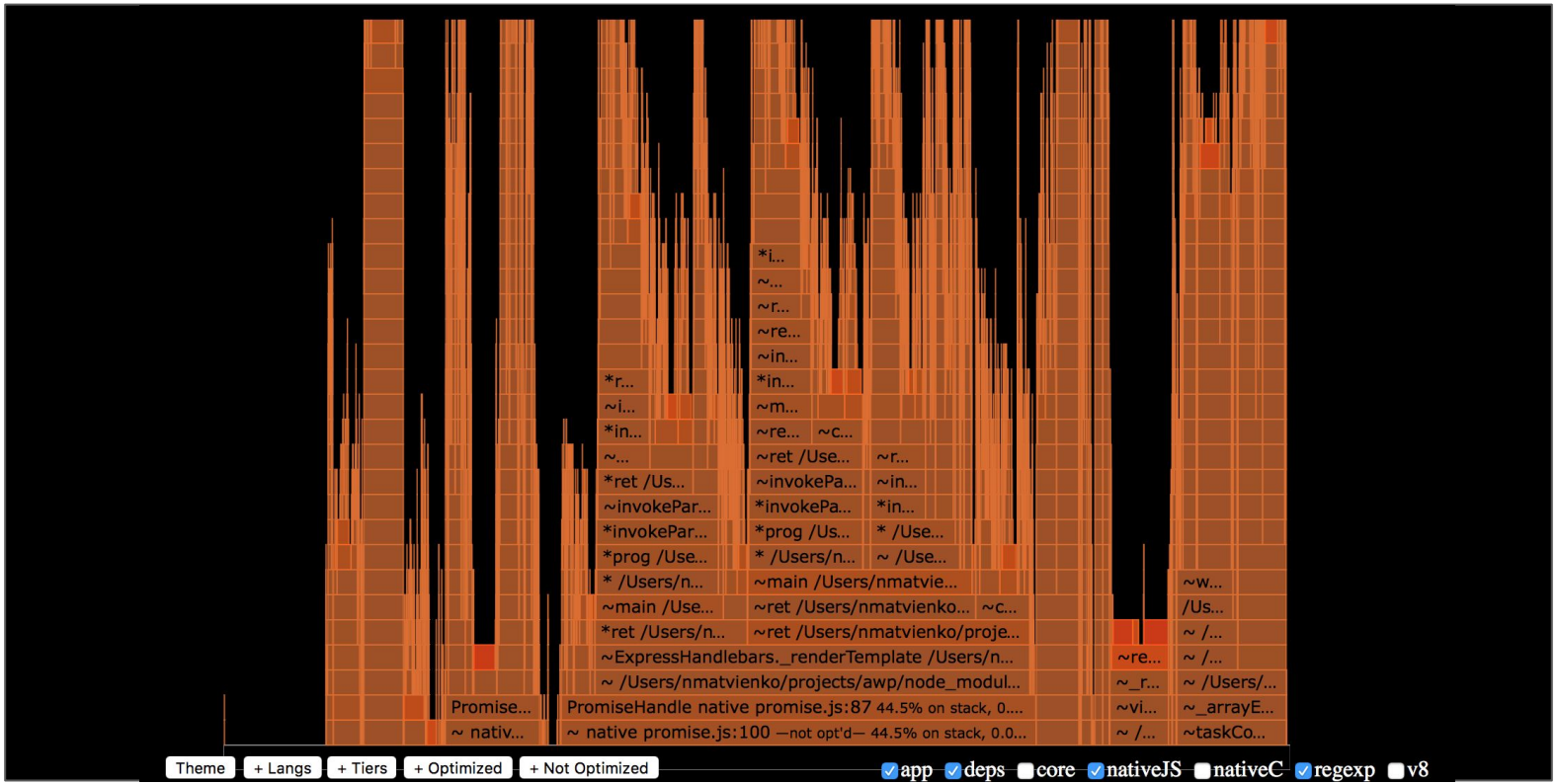
1. Определить “природу” проблемы
2. Подобрать инструменты под проблему
3. Произвести замер
4. Определить местоположение проблемы
5. Исправить (улучшить)
6. Произвести повторный замер

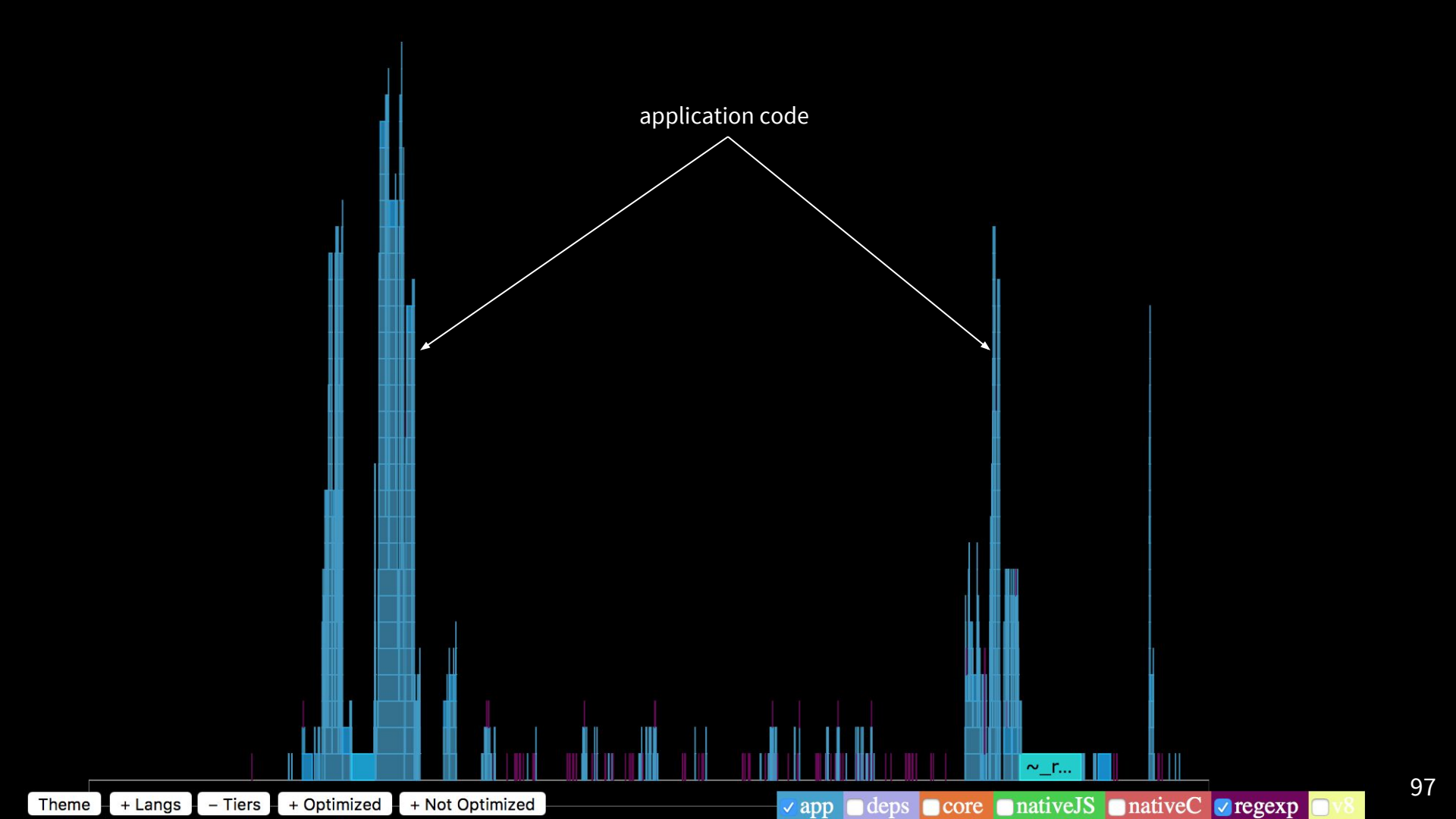
Замер: node --prof(iler) + apache bench/JMeter

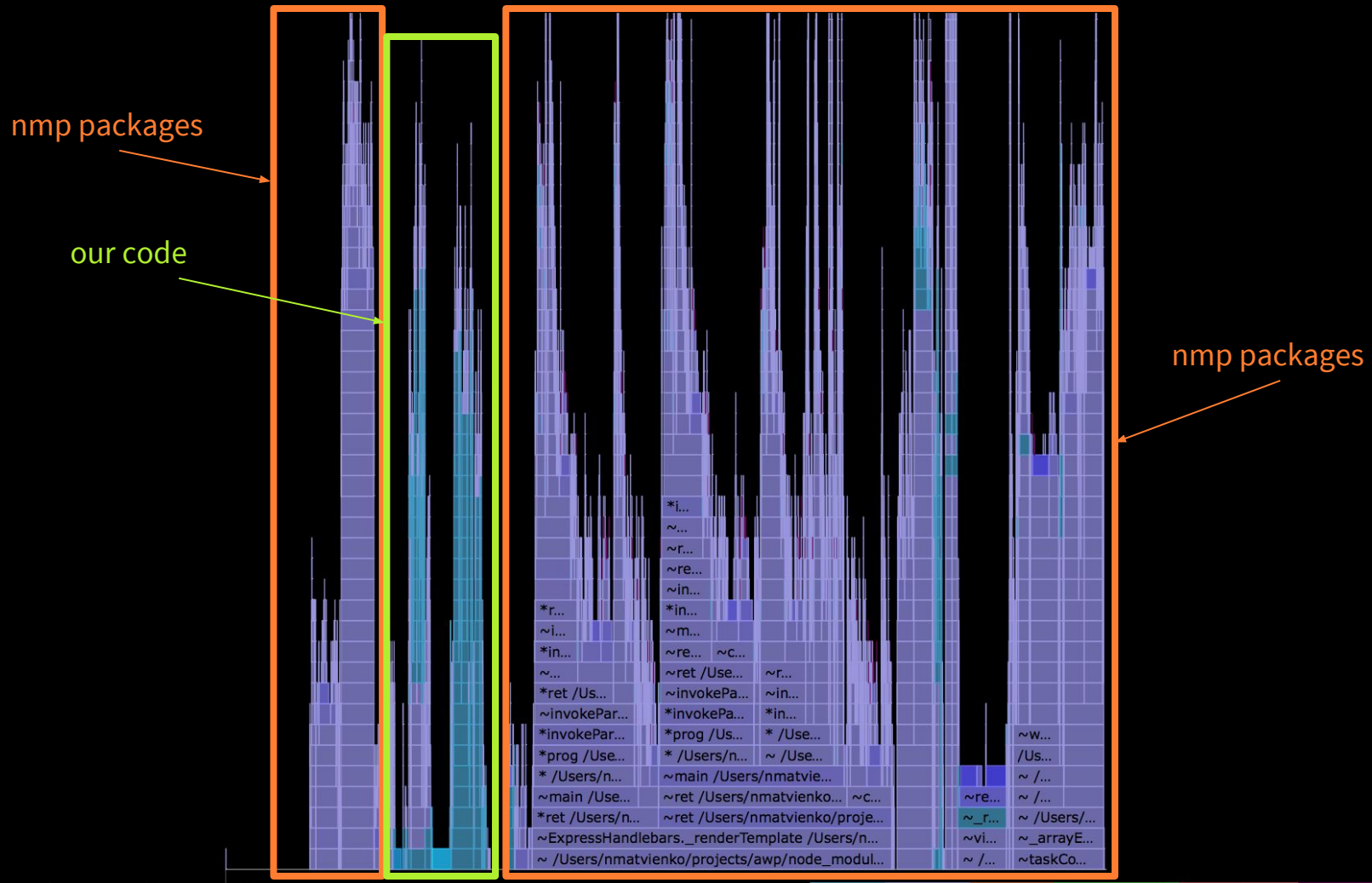
node --prof app.js и node --prof-process isolate-0x-v8.log > profile.txt

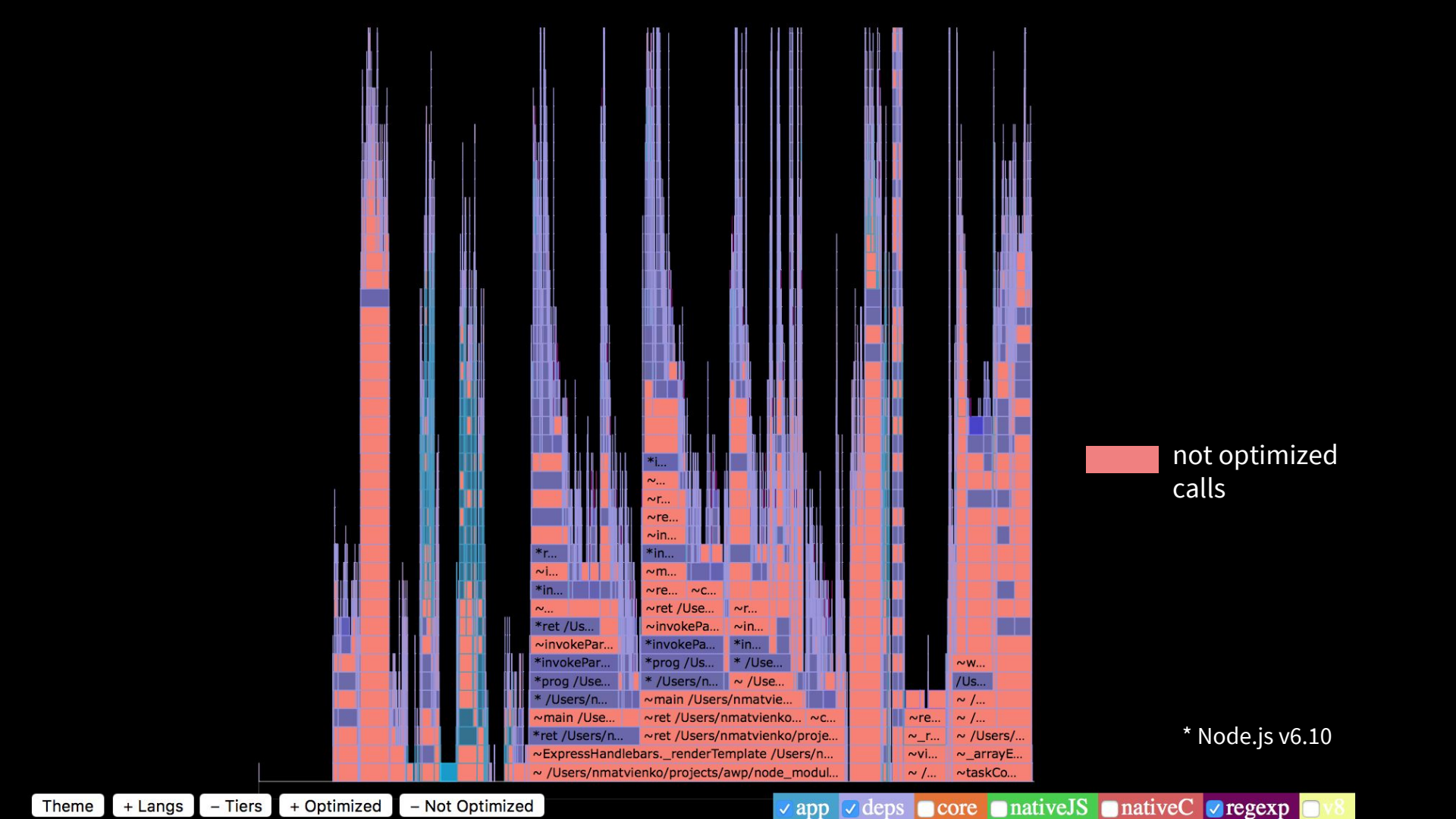
```
[Summary]:
   ticks  total  nonlib   name
   2710   11.9%  12.0%   JavaScript
   13348  58.7%  58.9%   C++
   561     2.5%   2.5%    GC
[JavaScript]:
    112   0.5%   0.5%  Builtin: CallFunction_ReceiverIsNotNullOrUndefined
     4    0.0%   0.0%  LazyCompile: *emit events.js:136:44

    42  15.2%   15.2% presentation/handlebars/helpers/urlBuilder.helper.js:51:37
     2   0.0%   0.0%  LazyCompile: ~substr native string.js:324:22
 [C++]:
   1292   5.7%   5.7%  node::ContextifyScript::New(v8::FunctionCallbackInfo<v8::Value> const&)
```

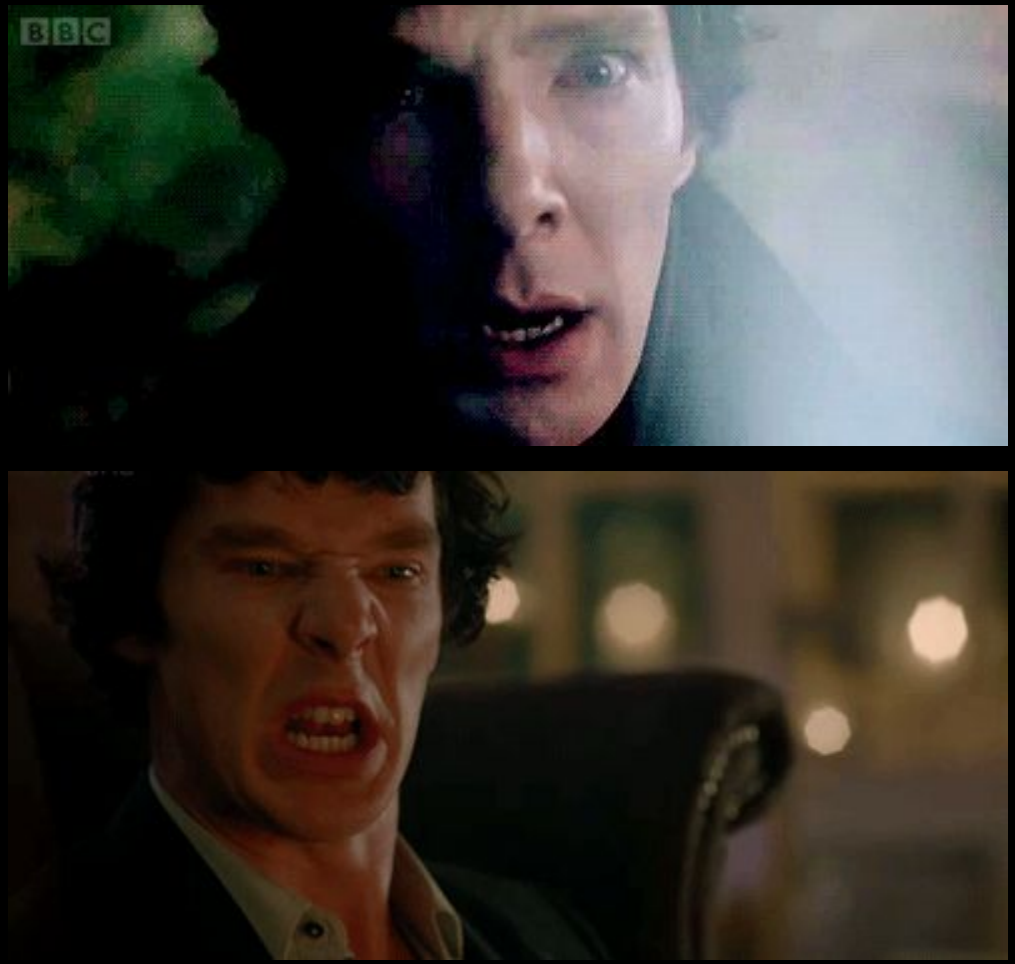
Пример 5. 0x flame graph приложения.

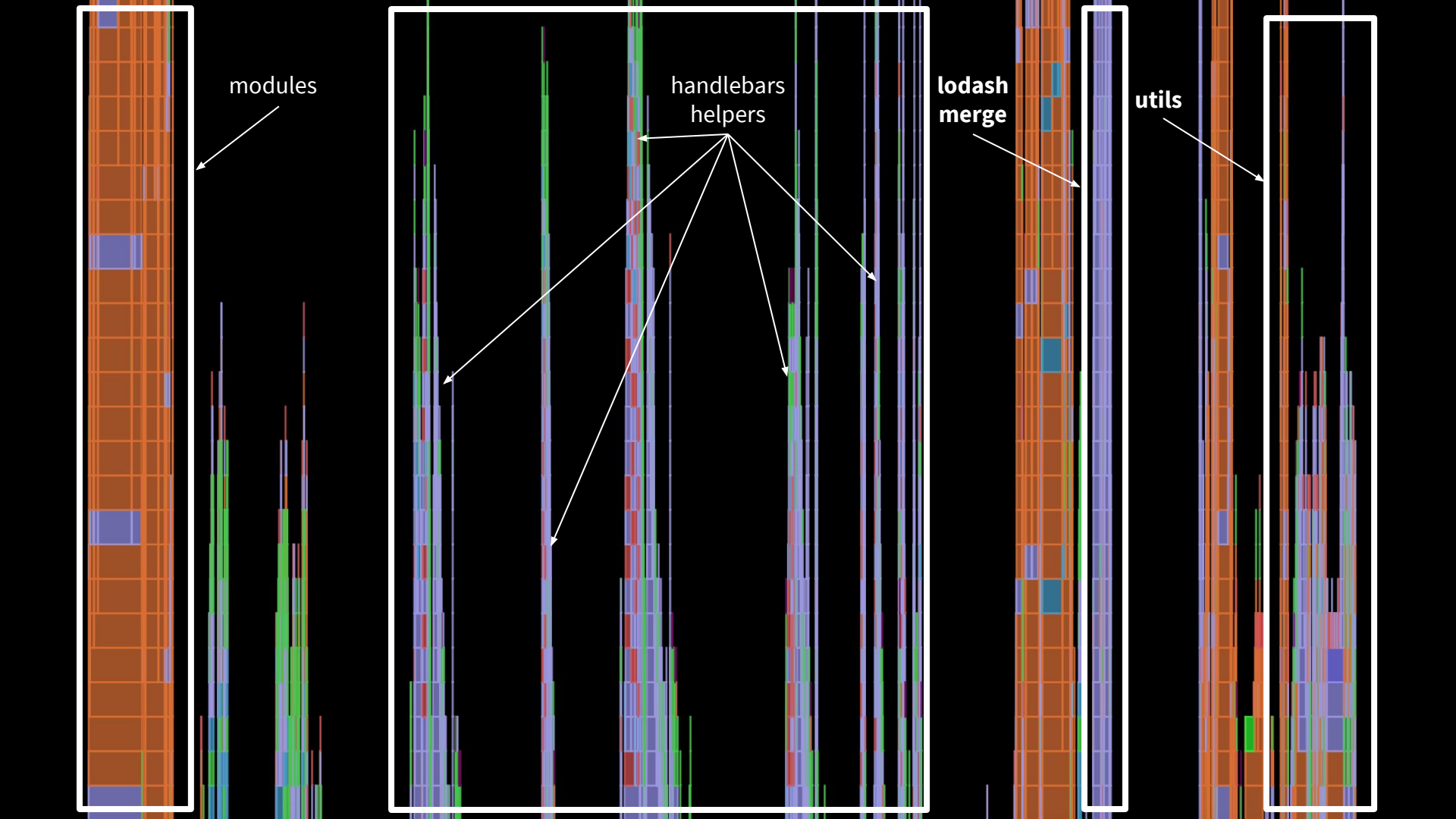


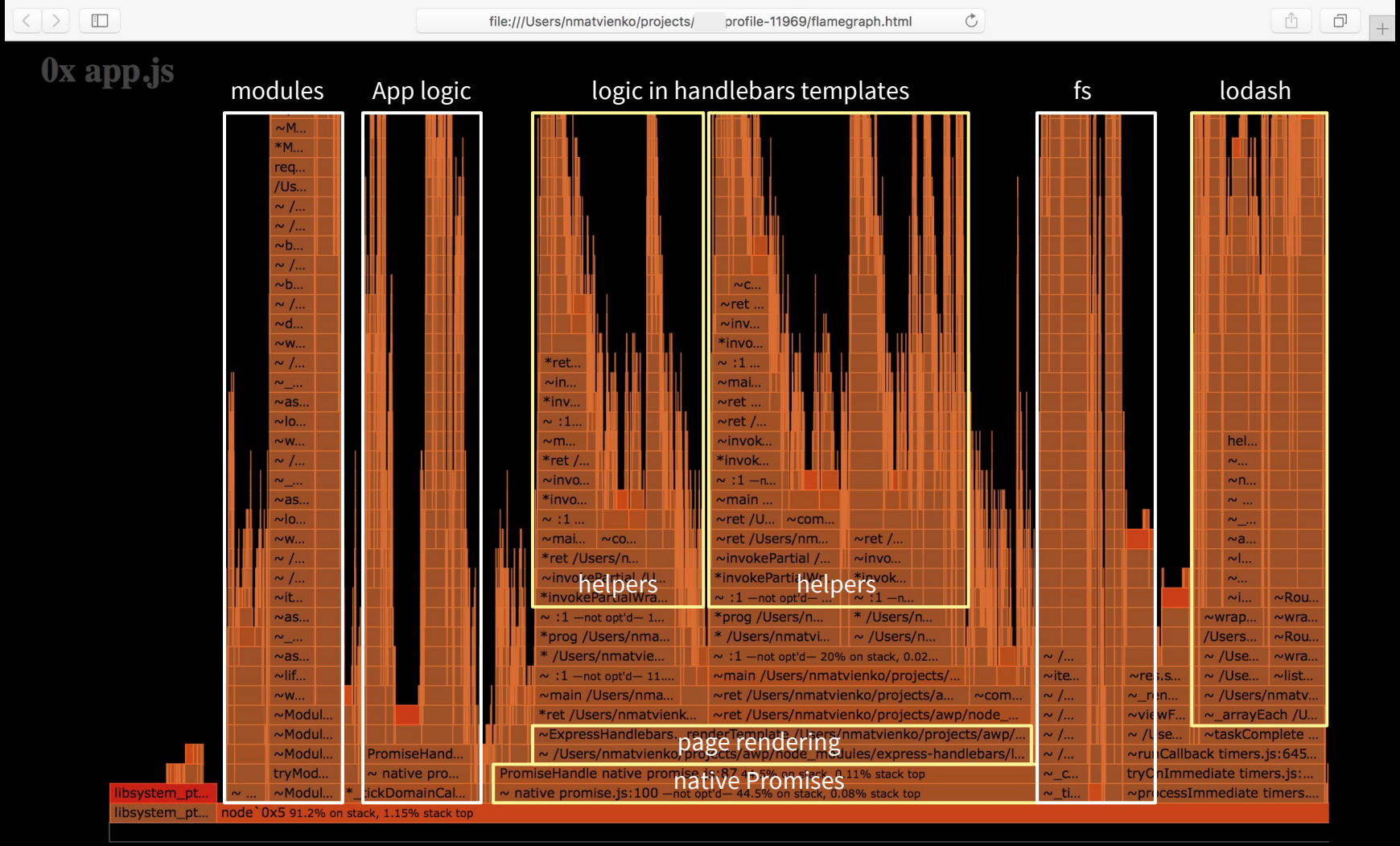

application code
~_r...
Theme
+ Langs
– Tiers
+ Optimized
+ Not Optimized
app
deps
core
nativeJS
nativeC
regexp
v8


nmp packages
our code
nmp packages
*i...
~...
~r...
~re...
~in...
*r...
*in...
~i...
~m...
*in...
~re...
~c...
~...
~ret /Use...
~r...
*ret /Us...
~invokePa...
~in...
~invokePar...
*invokePa...
*in...
*invokePar...
*prog /Us...
* /Use...
~w...
*prog /Use...
* /Users/n...
~ /Use...
/Us...
* /Users/n...
~main /Users/nmatvie...
~ /...
~main /Use...
~ret /Users/nmatvienko...
~c...
~re...
~ /...
*ret /Users/n...
~ret /Users/nmatvienko/proje...
~_r...
~ /Users/...
~ExpressHandlebars._renderTemplate /Users/n...
~vi...
~_arrayE...
~ /Users/nmatvienko/projects/awp/node_modul...
~ /...
~taskCo...
Theme
+ Langs
– Tiers
+ Optimized
+ Not Optimized
app
deps
core
nativeJS
nativeC
regexp
v8


not optimized calls
* Node.js v6.10
Theme
+ Langs
– Tiers
+ Optimized
– Not Optimized
app
deps
core
nativeJS
nativeC
regexp
v8

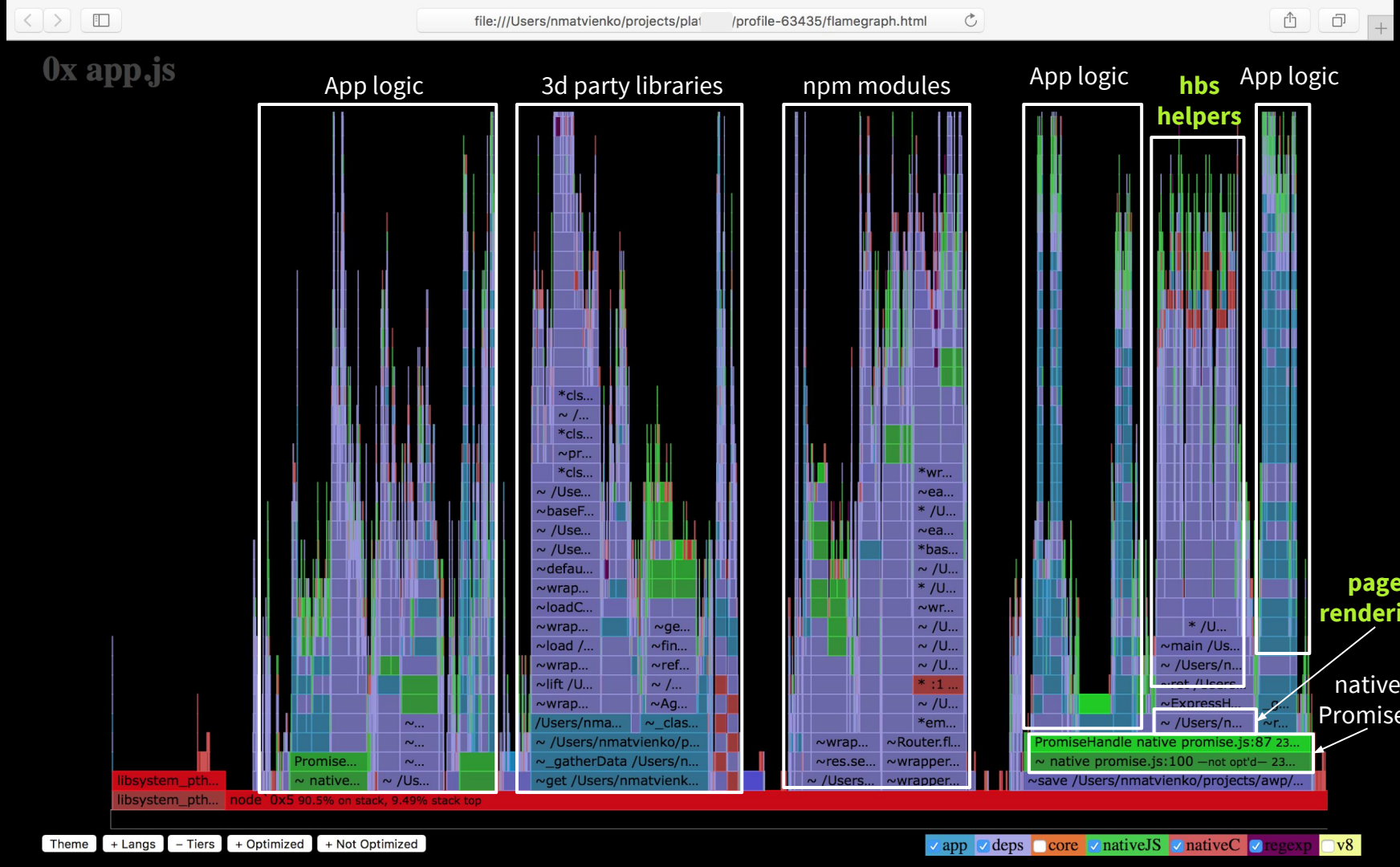

BBC


modules
handlebars
helpers
lodash
merge
utils

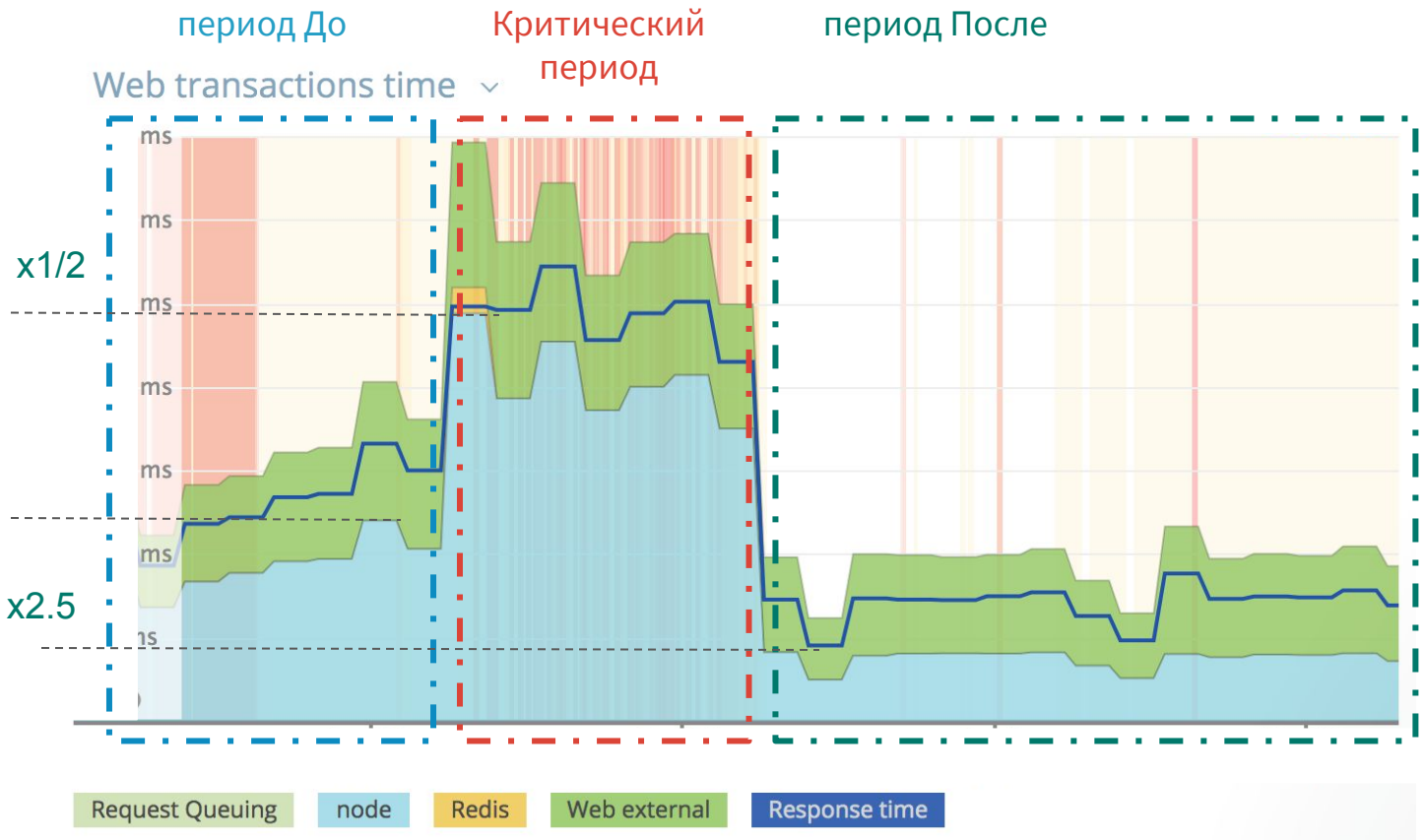

file:///Users/nmatvienko/projects/ profile-11969/flamegraph.html
0x app.js
modules
App logic
logic in handlebars templates
fs
lodash
helpers
helpers
page rendering
native Promises
node`0x5 91.2% on stack, 1.15% stack top

Исправления

- Уменьшение CPU затратных операций: merge, JSON и string parse
- Удаление логики из view templates
- Ревью npm зависимостей и обновление пакетов.
- Обновление Node.js (обход V8 optimization killers) – в процессе


0x app.js
App logic
3d party libraries
npm modules
App logic
hbs helpers
App logic
page rendering
native Promises
Promise...
~ native...
libsystem_pth...
libsystem_pth...
node 0x5 90.5% on stack, 9.49% stack top
PromiseHandle native promise.js:87 23...
~ native promise.js:100 —not opt'd— 23...
~save /Users/nmatvienko/projects/awp/...
Theme
+ Langs
– Tiers
+ Optimized
+ Not Optimized
app
deps
core
nativeJS
nativeC
regexp
v8

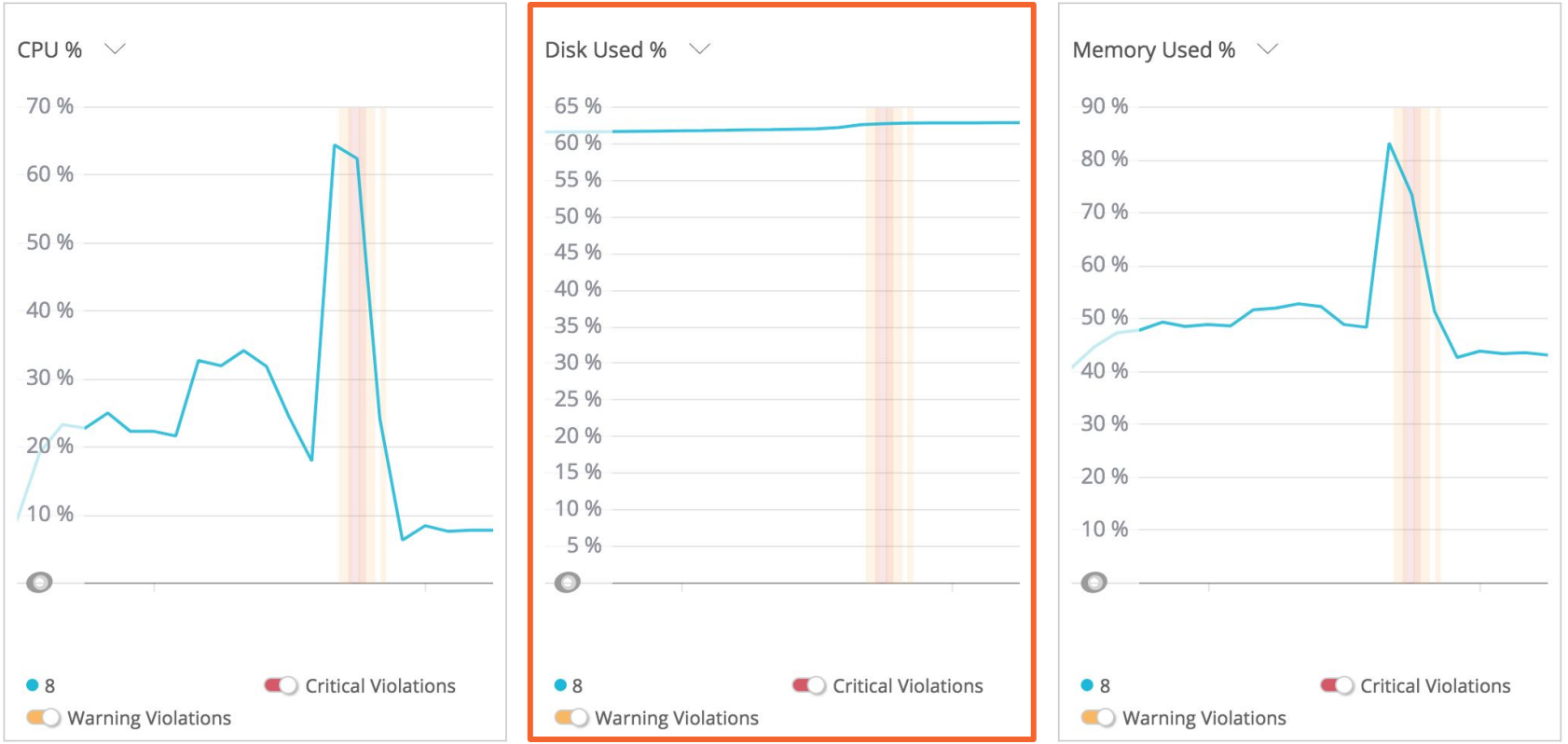
Результат


период До
Критический период
период После
Web transactions time
ms
ms
ms
ms
ms
ms
x1/2
x2.5
Request Queuing
node
Redis
Web external
Response time

Вывод

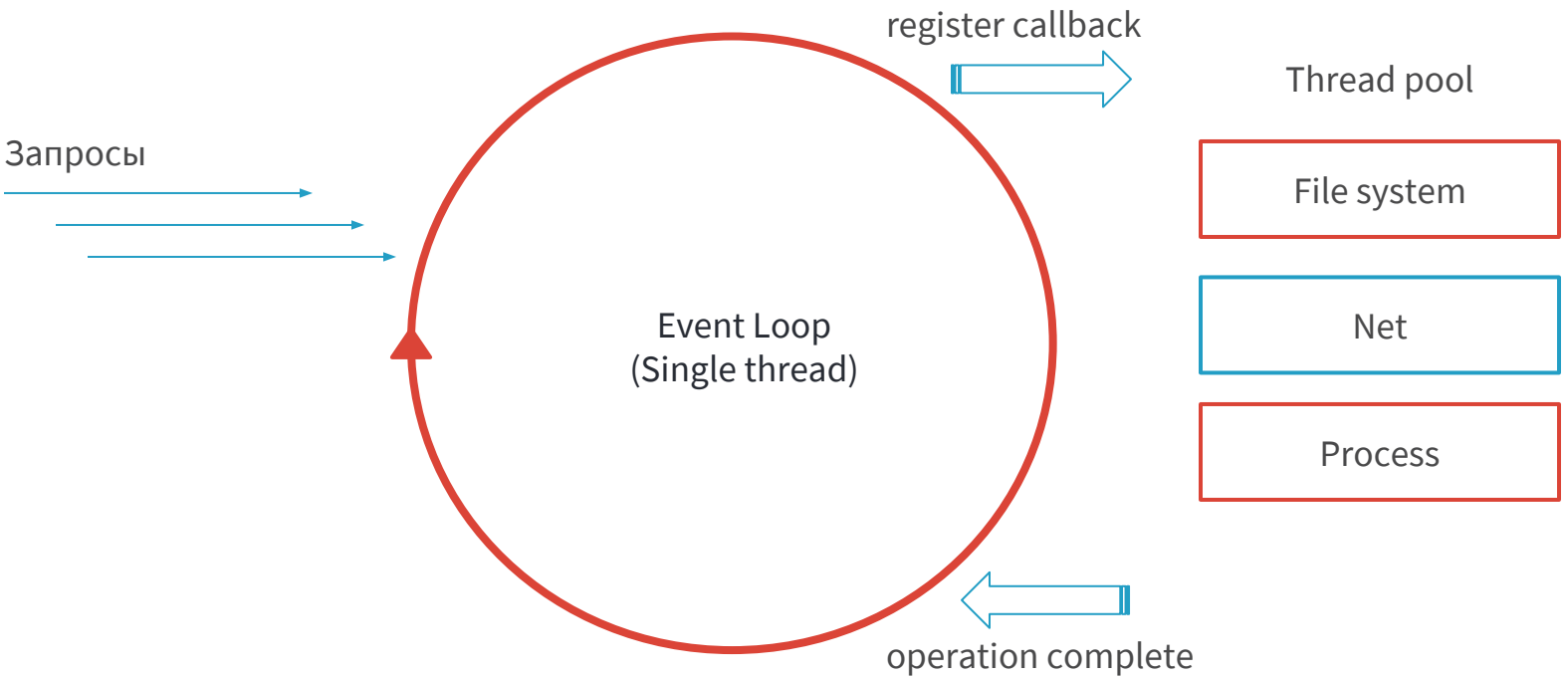
1. Производительность должна быть частью требований.
2. Производить замеры при внедрении сторонних библиотек
3. Замер производить на staging/pre-production environments
4. Собирать архив результатов замеров

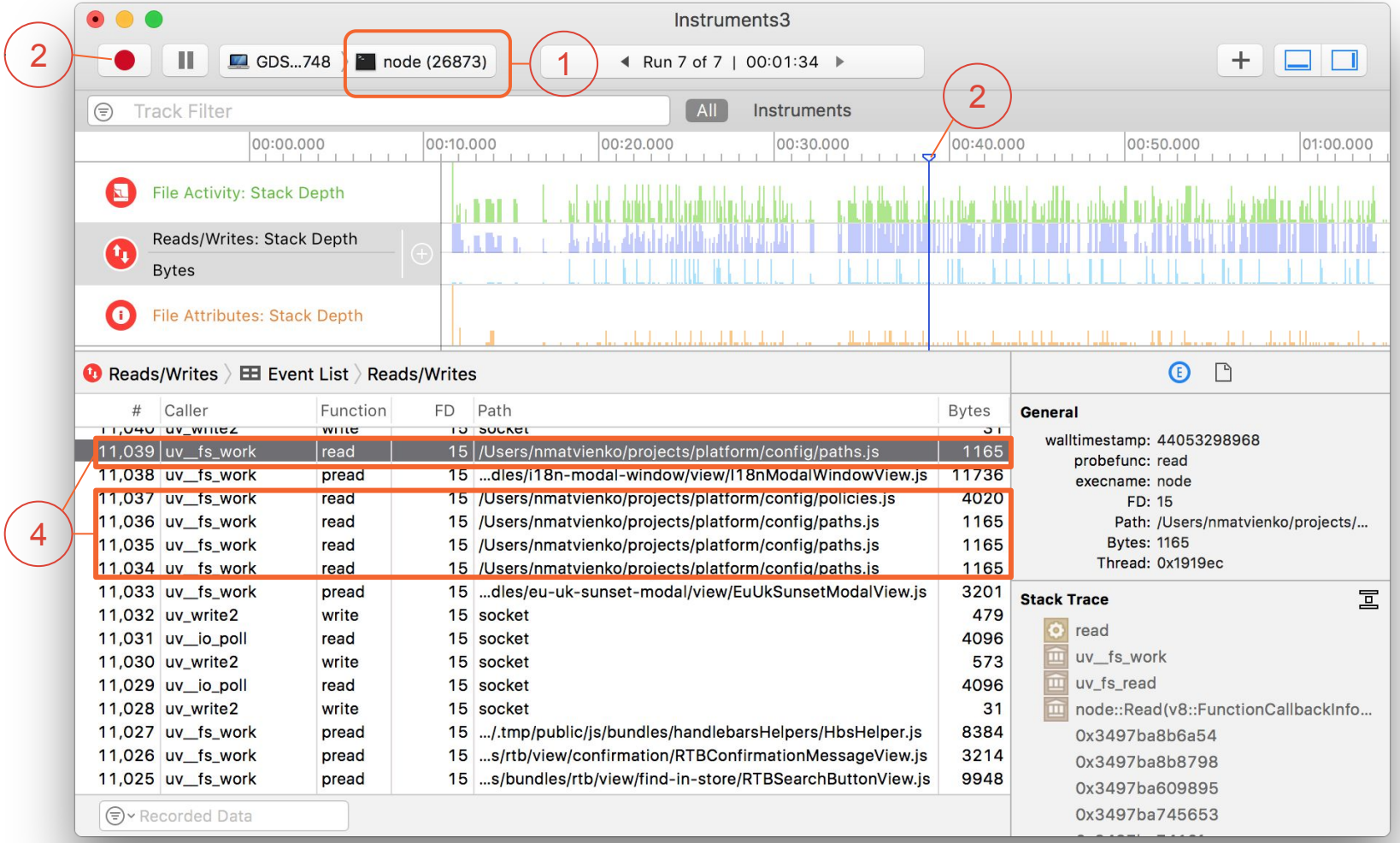
Disk used


CPU %
70 %
60 %
50 %
40 %
30 %
20 %
10 %
8
Critical Violations
Warning Violations
Disk Used %
65 %
60 %
55 %
50 %
45 %
40 %
35 %
30 %
25 %
20 %
15 %
10 %
5 %
8
Critical Violations
Warning Violations
Memory Used %
90 %
80 %
70 %
60 %
50 %
40 %
30 %
20 %
10 %
8
Critical Violations
Warning Violations

Event Loop

Увеличить количество потоков UV_THREADPOOL_SIZE 4, 8, 16, ... 128.

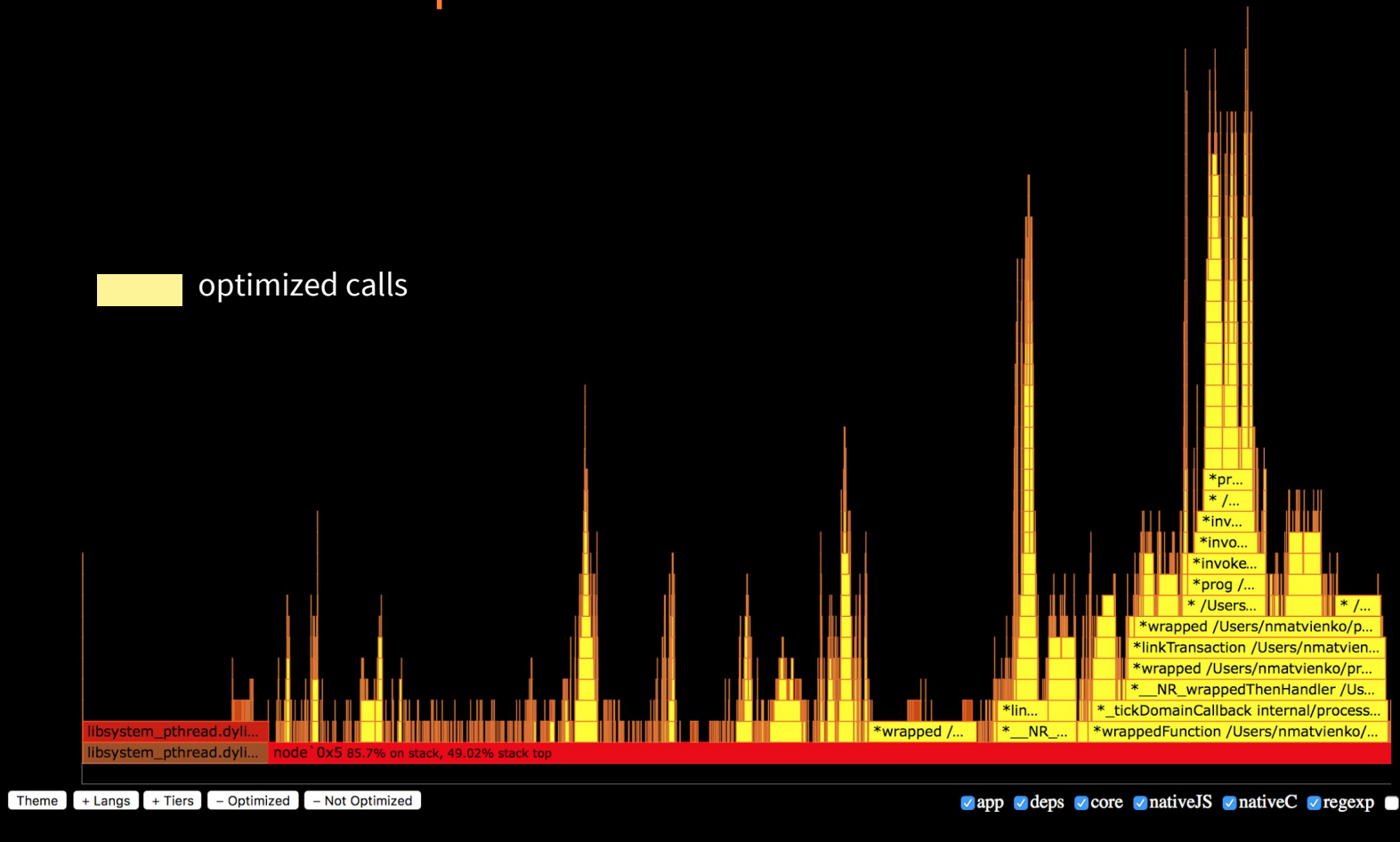



Instruments3
2
GDS...748
node (26873)
1
Run 7 of 7 | 00:01:34
Track Filter
All
Instruments
2
00:00.000
00:10.000
00:20.000
00:30.000
00:40.000
00:50.000
01:00.000
File Activity: Stack Depth
Reads/Writes: Stack Depth
Bytes
File Attributes: Stack Depth
Reads/Writes
Event List
Reads/Writes
Caller Function FD Path Bytes
11,039 uv__fs_work read 15 /Users/nmatvienko/projects/platform/config/paths.js 1165
11,038 uv__fs_work pread 15 ...dles/i18n-modal-window/view/I18nModalWindowView.js 11736
11,037 uv__fs_work read 15 /Users/nmatvienko/projects/platform/config/policies.js 4020
11,036 uv__fs_work read 15 /Users/nmatvienko/projects/platform/config/paths.js 1165
11,035 uv__fs_work read 15 /Users/nmatvienko/projects/platform/config/paths.js 1165
11,034 uv__fs_work read 15 /Users/nmatvienko/projects/platform/config/paths.js 1165
11,033 uv__fs_work pread 15 ...dles/eu-uk-sunset-modal/view/EuUkSunsetModalView.js 3201
11,032 uv_write2 write 15 socket 479
11,031 uv__io_poll read 15 socket 4096
11,030 uv_write2 write 15 socket 573
11,029 uv__io_poll read 15 socket 4096
11,028 uv_write2 write 15 socket 31
11,027 uv__fs_work pread 15 .../.tmp/public/js/bundles/handlebarsHelpers/HbsHelper.js 8384
11,026 uv__fs_work pread 15 ...s/rtb/view/confirmation/RTBConfirmationMessageView.js 3214
11,025 uv__fs_work pread 15 ...s/bundles/rtb/view/find-in-store/RTBSearchButtonView.js 9948
Recorded Data
4
General
walltimestamp: 44053298968
probefunc: read
execname: node
FD: 15
Path: /Users/nmatvienko/projects/...
Bytes: 1165
Thread: 0x1919ec
Stack Trace
read
uv__fs_work
uv_fs_read
node::Read(v8::FunctionCallbackInfo...
0x3497ba8b6a54
0x3497ba8b8798
0x3497ba609895
0x3497ba745653

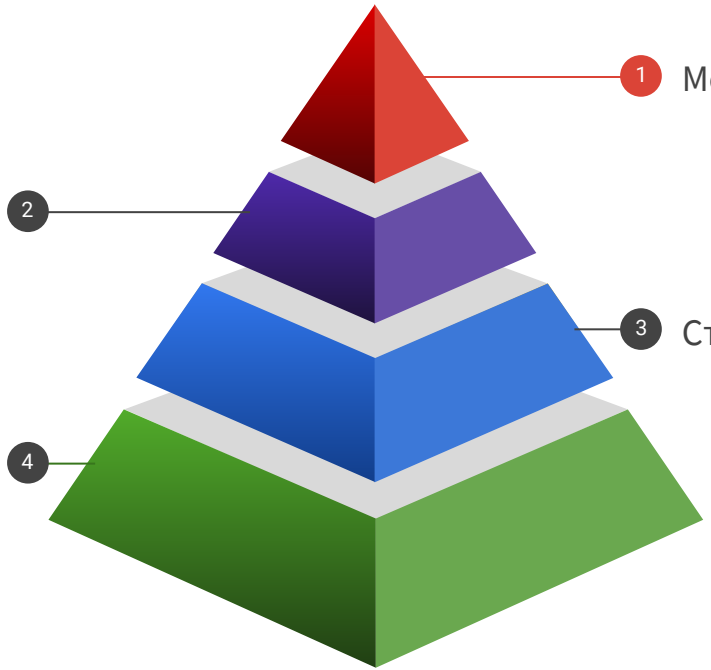
Пример 6. llnode. Установка break point на libuv

```
$ llnode -- /Users/nmatvienko/.nvm/versions/node/v8.9.1/bin/node app.js

        * thread #1, queue = 'com.apple.main-thread', stop reason = signal SIGSTOP
          frame #0: 0x00007fff8e76dd96 libsystem_kernel.dylib`kevent + 10
        libsystem_kernel.dylib`kevent:
        ->  0x7fff8e76dd96 <+10>: jae    0x7fff8e76dda0          ; <+20>
            0x7fff8e76dd98 <+12>: movq   %rax, %rdi
            0x7fff8e76dd9b <+15>: jmp    0x7fff8e765caf          ; cerror_nocancel
            0x7fff8e76dda0 <+20>: retq


$(llnode) b -n uv_fs_read -c cb==NULL
        Breakpoint 1: where = node`uv_fs_read, address = 0x100bd16fe
```

Turbofan code optimizations



Диагностика



Ресурсы

https://github.com/nickkooper/nodejs-diagnostics-resources

Пишите вопросы

Матвиенко Николай

mail: matvi3nko@gmail.com

twitter: @matvi3nko

github: @nickkooper

Linked-in nikolaymatvienko