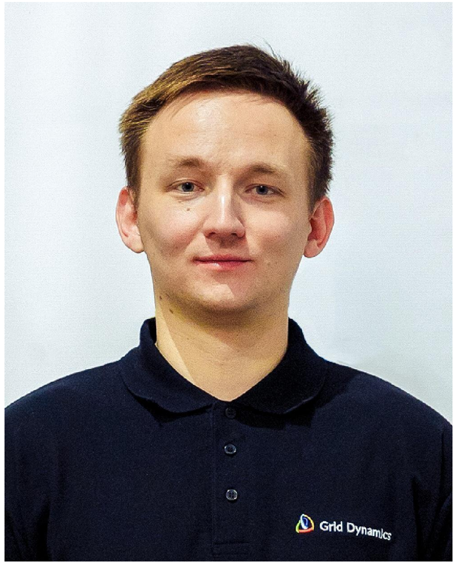
Node.js applications diagnostics under the hood

Nikolay Matvienko
2018

Author



Nikolay Matvienko

Full Stack JS developer at Grid Dynamics

twitter: @matvi3nko

facebook: matvienko.nikolay

mail: matvi3nko@gmail.com

github: @matvi3nko

Customers in retail domain

AMERICAN EAGLE OUTFITTERS

SEPHORA

macy's

KOHL'S

JCPenney

ToysRus®

Diagnostics

Node.js Diagnostics Working Group https://github.com/nodejs/diagnostics

Post Mortem Diagnostics Working Group https://github.com/nodejs/post-mortem

Work is divided into several domains:

1. Tracing

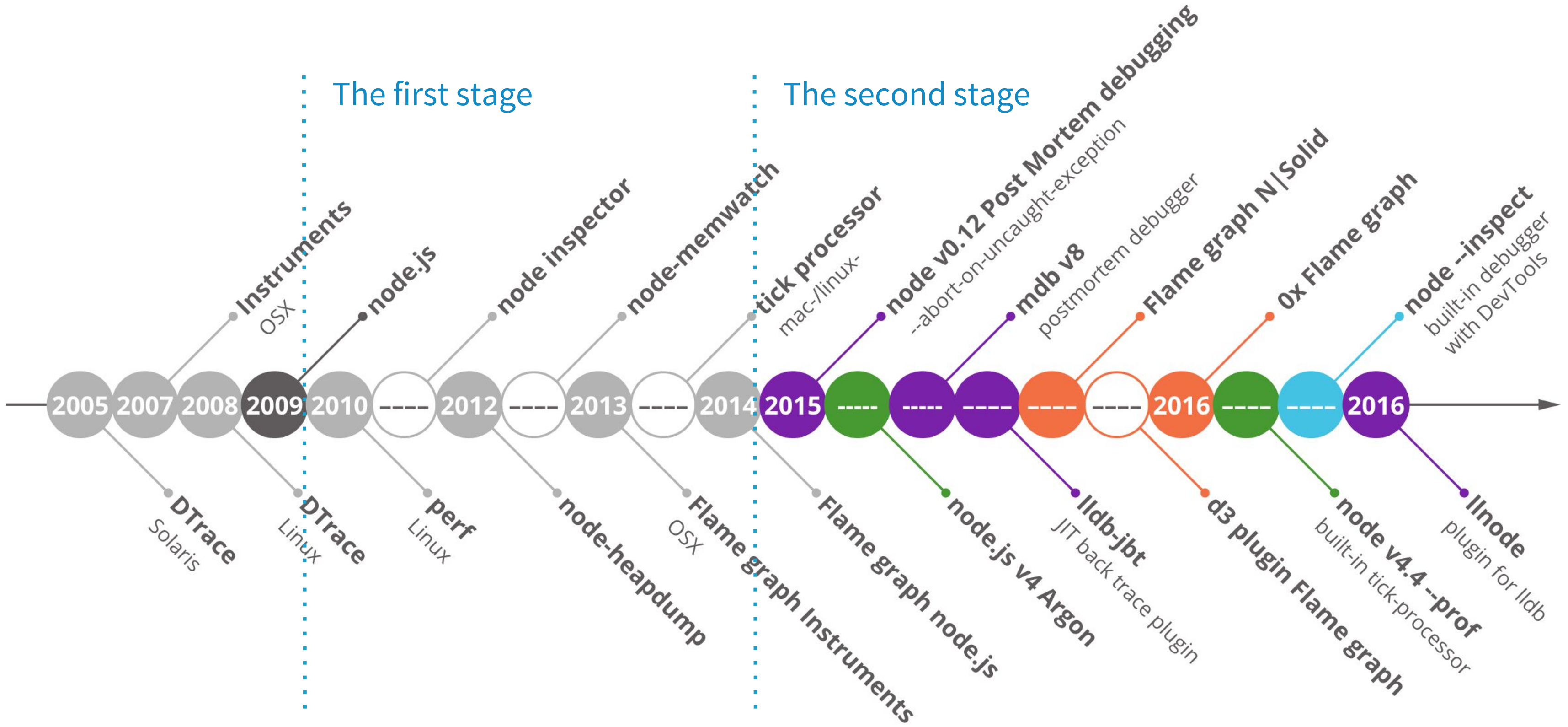
   Profiling

   Heap and Memory Analysis

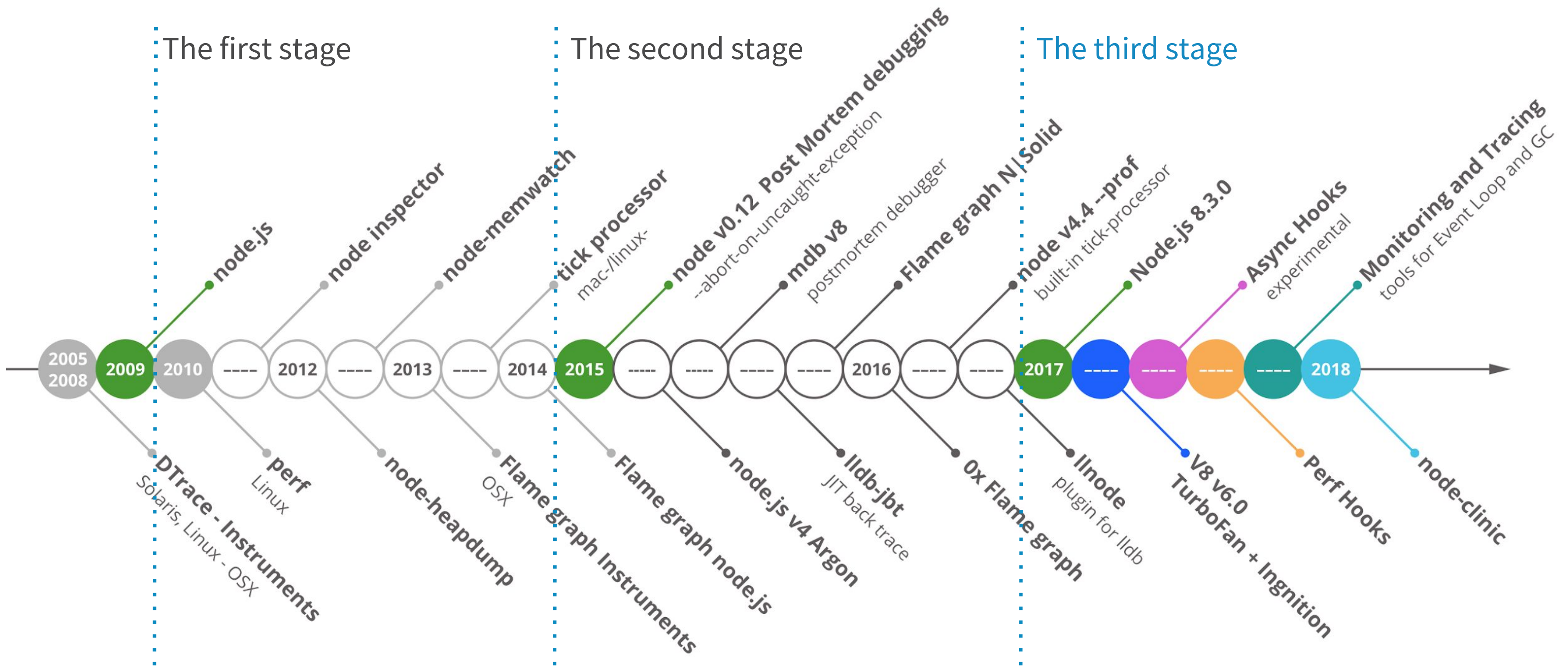
   Step Debugging

2. Postmortem debugging

Stages of Node.js diagnostics evolution before 2017



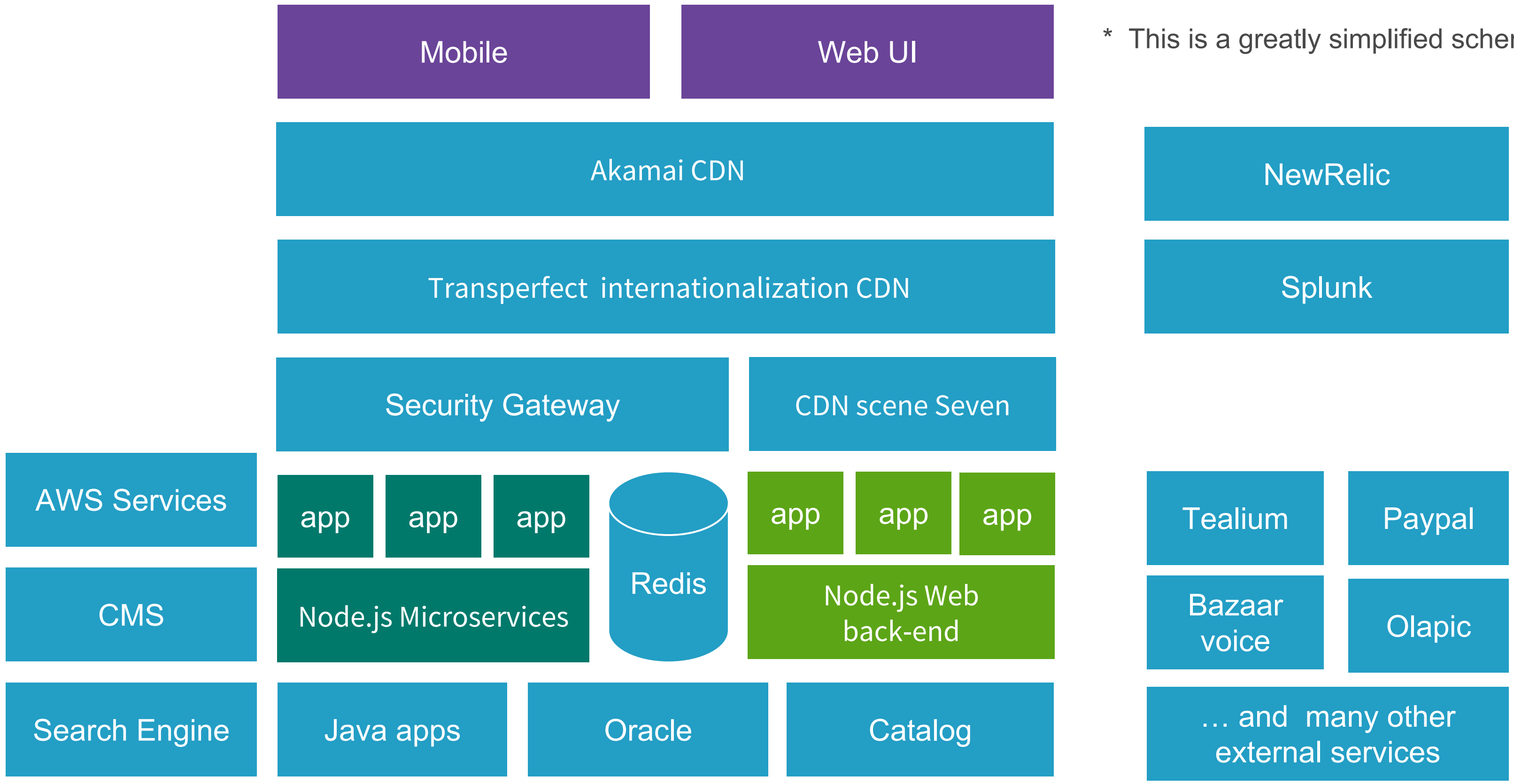
The third stage of Node.js diagnostics evolution



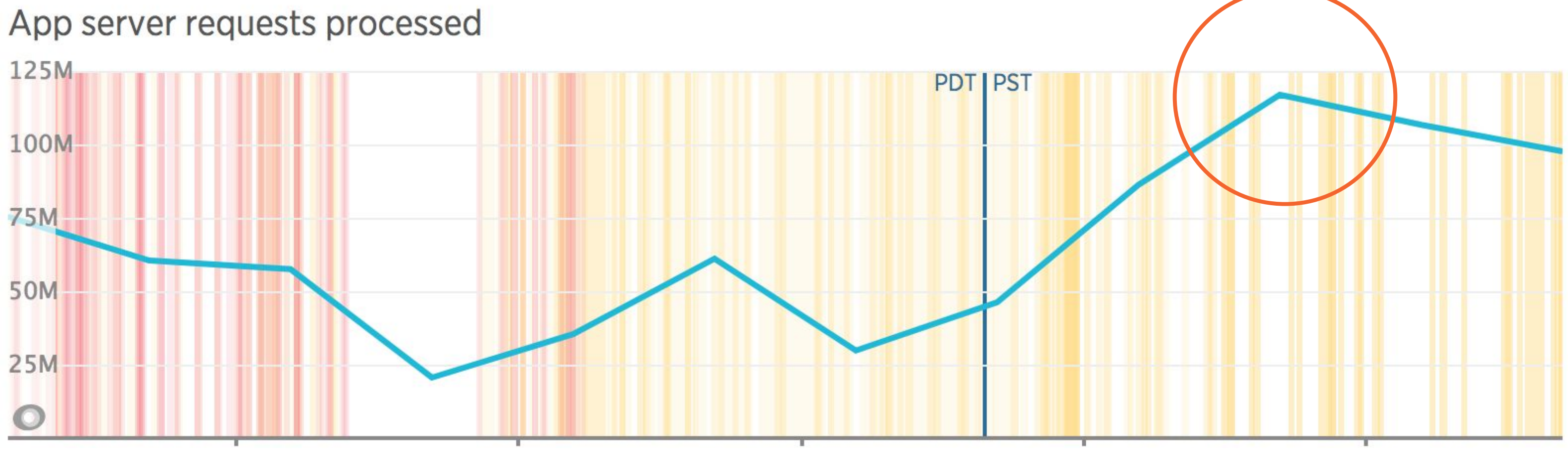
V8 engine update

1. Diagnostic tools appeared with a delay.
2. Most tools didn’t support the latest version of V8 (Node.js)
3. We had to choose: performance or tools
4. Or new tools appear, but without support for previous versions of Node. js

Node.js in Enterprise Architecture



Black Friday = ~60% of the annual income.



Agenda

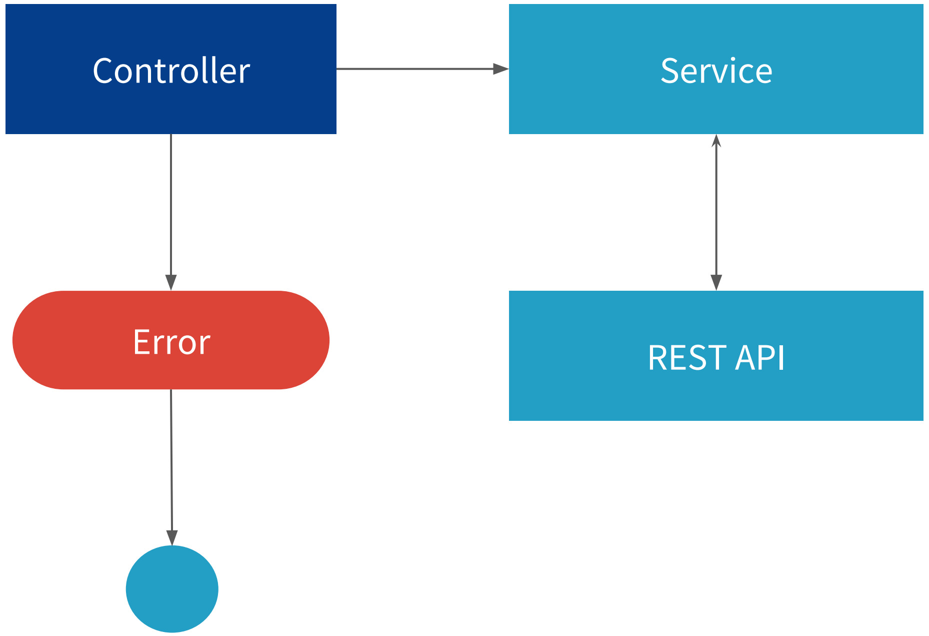
1. Debugging in production
2. Performance profiling
3. Search for memory leaks

1. Debugging in production

The reasons:

1. Uncaught Exception
2. Difficult reproducible errors
3. Production environment
4. ASAP

Example



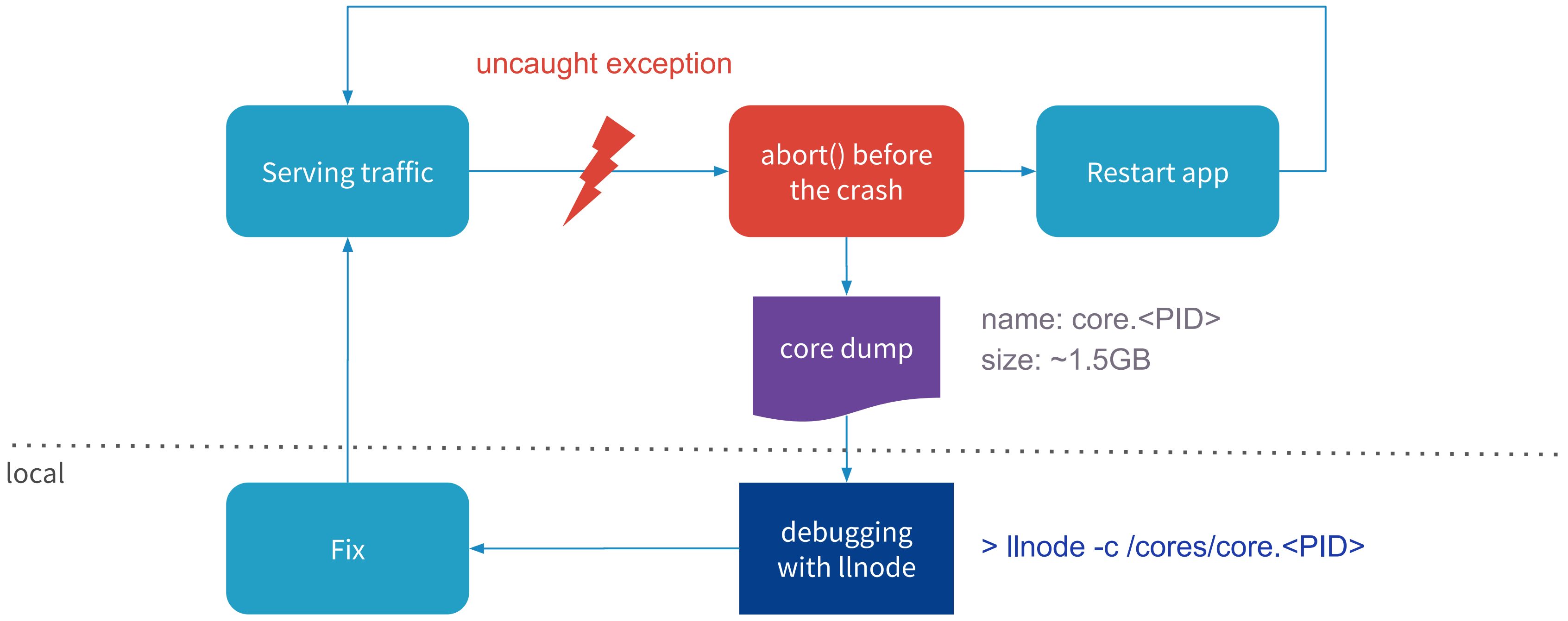
Case 1. Product reservation.

```
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;

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

Throws an Error when rewards is undefined

Core dump creation algorithm for postmortem debugging

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

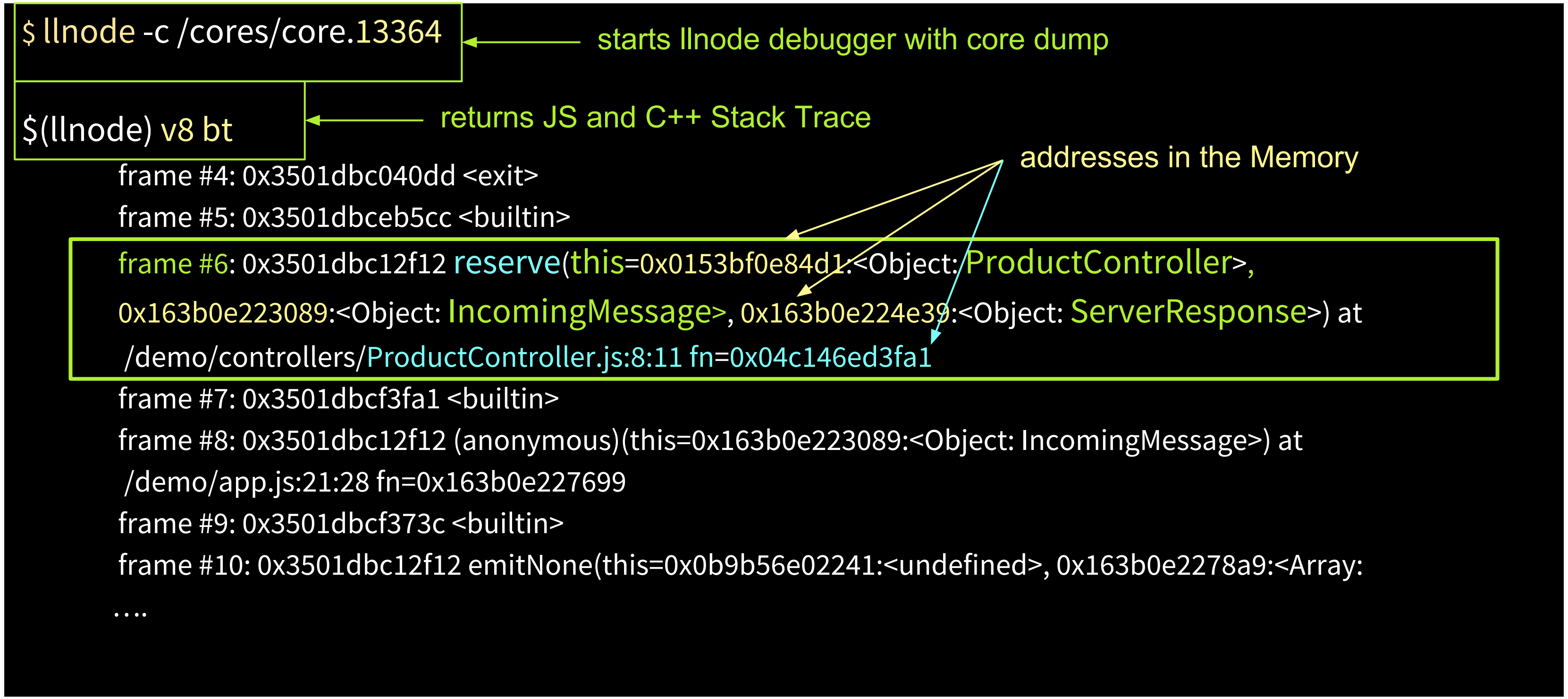


```
> llnode -c /cores/core.<PID>
```

Debugging steps

1. Get the Stack Trace and find the last JS function before the exit
2. Read the source code of the function
3. Get input parameters: request and response

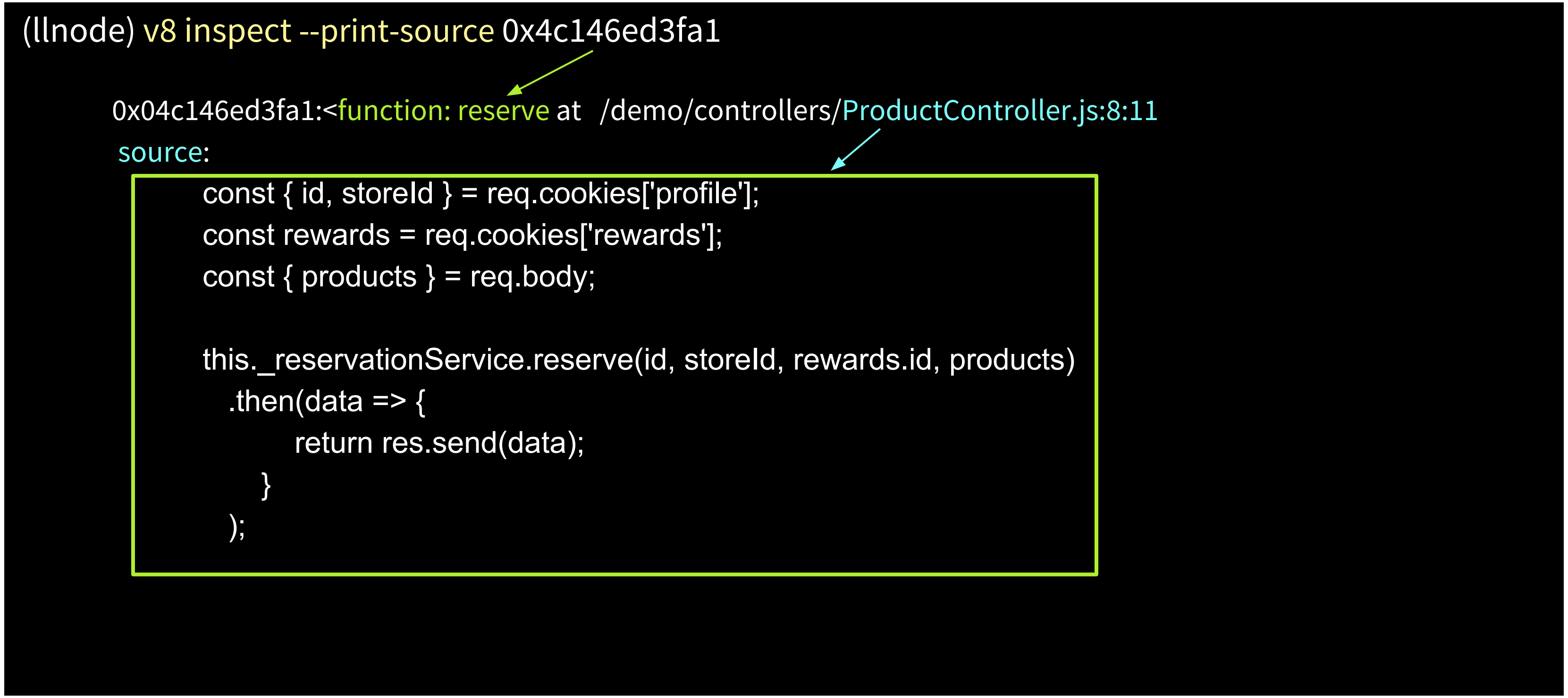
Step 1. Get the Stack Trace and find the last JS function



```
$ llnode -c /cores/core.13364
$(llnode) v8 bt
        frame #4: 0x3501dbc040dd <exit>
        frame #5: 0x3501dbceb5cc <builtin>
        frame #6: 0x3501dbc12f12 reserve(this=0x0153bf0e84d1:<Object: ProductController>,
        0x163b0e223089:<Object: IncomingMessage>, 0x163b0e224e39:<Object: ServerResponse>) at
         /demo/controllers/ProductController.js:8:11 fn=0x04c146ed3fa1
        frame #7: 0x3501dbcf3fa1 <builtin>
        frame #8: 0x3501dbc12f12 (anonymous)(this=0x163b0e223089:<Object: IncomingMessage>) at
         /demo/app.js:21:28 fn=0x163b0e227699
        frame #9: 0x3501dbcf373c <builtin>
        frame #10: 0x3501dbc12f12 emitNone(this=0x0b9b56e02241:<undefined>, 0x163b0e2278a9:<Array:
       ….
```

starts llnode debugger with core dump
returns JS and C++ Stack Trace
addresses in the Memory

Step 2. Read the source code of the function

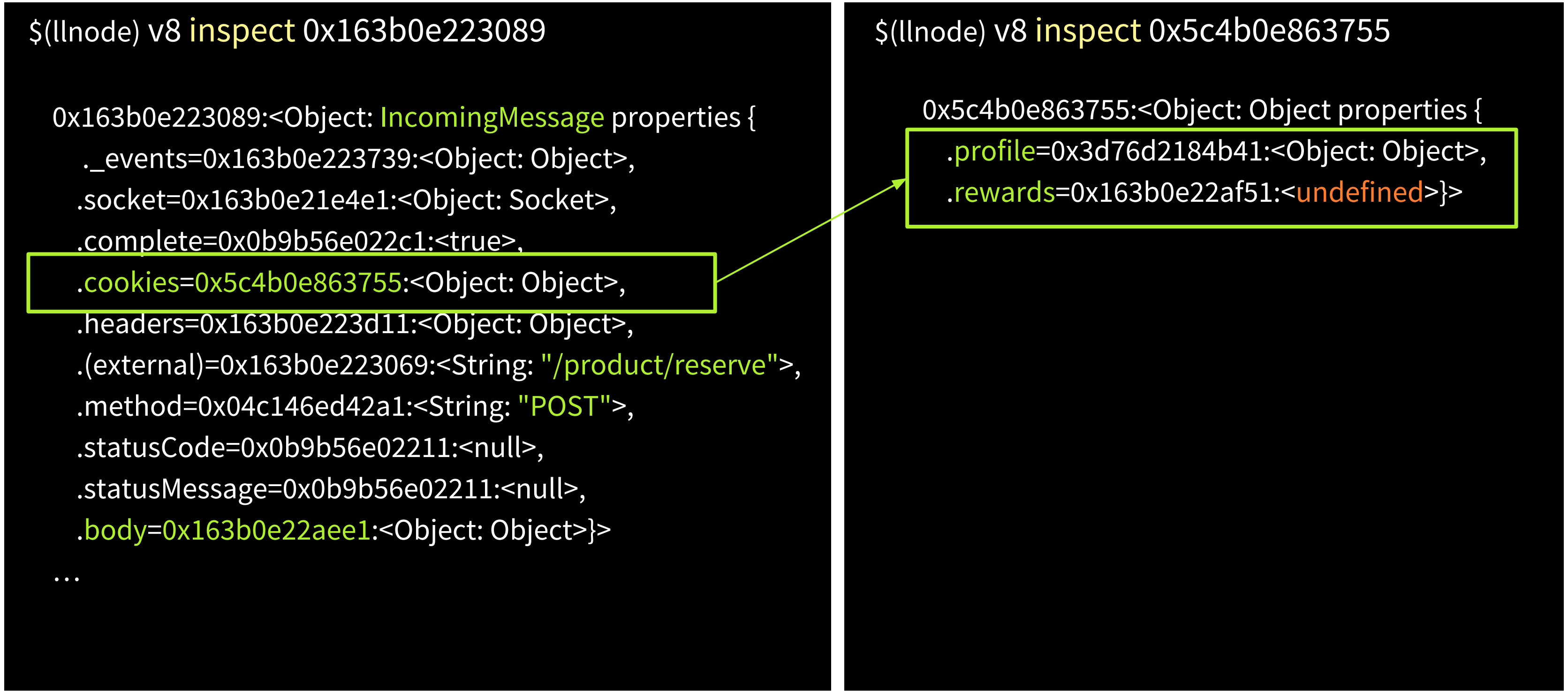


```
(llnode) v8 inspect --print-source 0x4c146ed3fa1

      0x04c146ed3fa1:<function: reserve at  /demo/controllers/ProductController.js:8:11
       source:
            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);
                 }
              );
```

Step 3. Get input parameter – request

```
$(llnode) v8 bt 20

        frame #4: 0x3501dbc040dd <exit>
        frame #5: 0x3501dbceb5cc <builtin>
        frame #6: 0x3501dbc12f12 reserve(this=0x0153bf0e84d1:<Object: ProductController>,
        0x163b0e223089:<Object: IncomingMessage>, 0x163b0e224e39:<Object: ServerResponse>) at
         /demo/controllers/ProductController.js:8:11 fn=0x04c146ed3fa1
        frame #7: 0x3501dbcf3fa1 <builtin>
        frame #8: 0x3501dbc12f12 (anonymous)(this=0x163b0e223089:<Object: IncomingMessage>) at
         /demo/app.js:21:28 fn=0x163b0e227699
        frame #9: 0x3501dbcf373c <builtin>
        frame #10: 0x3501dbc12f12 emitNone(this=0x0b9b56e02241:<undefined>, 0x163b0e2278a9:<Array:
        length=2>,0x0b9b56e022f1:<false>, 0x163b0e223089:<Object: IncomingMessage>)
        at events.js:103:18 fn=0x0f2606a87391
        ….
        frame #20: ...
```

Step 3 result. Request object



```
$(llnode) v8 inspect 0x163b0e223089

  0x163b0e223089:<Object: IncomingMessage properties {
     ._events=0x163b0e223739:<Object: Object>,
     .socket=0x163b0e21e4e1:<Object: Socket>,
     .complete=0x0b9b56e022c1:<true>,
     .cookies=0x5c4b0e863755:<Object: Object>,
     .headers=0x163b0e223d11:<Object: Object>,
     .(external)=0x163b0e223069:<String: "/product/reserve">,
     .method=0x04c146ed42a1:<String: "POST">,
     .statusCode=0x0b9b56e02211:<null>,
     .statusMessage=0x0b9b56e02211:<null>,
     .body=0x163b0e22aee1:<Object: Object>}>
  ...
```

```
$(llnode) v8 inspect 0x5c4b0e863755

    0x5c4b0e863755:<Object: Object properties {
      .profile=0x3d76d2184b41:<Object: Object>,
      .rewards=0x163b0e22af51:<undefined>}>
```

Case 1. Informative Stack Trace

```
$(llnode) v8 bt

        frame #4: 0x3501dbc040dd <exit>
        frame #5: 0x3501dbceb5cc <builtin>
        frame #6: 0x3501dbc12f12 reserve(this=0x0153bf0e84d1:<Object: ProductController>,
        0x163b0e223089:<Object: IncomingMessage>, 0x163b0e224e39:<Object: ServerResponse>) at
         /demo/controllers/ProductController.js:8:11 fn=0x04c146ed3fa1
        frame #7: 0x3501dbcf3fa1 <builtin>
        frame #8: 0x3501dbc12f12 (anonymous)(this=0x163b0e223089:<Object: IncomingMessage>) at
         /demo/app.js:21:28 fn=0x163b0e227699
        frame #9: 0x3501dbcf373c <builtin>
        frame #10: 0x3501dbc12f12 emitNone(this=0x0b9b56e02241:<undefined>, 0x163b0e2278a9:<Array:
       length=2>,0x0b9b56e022f1:<false>, 0x163b0e223089:<Object: IncomingMessage>)
        at events.js:103:18 fn=0x0f2606a87391
       ….
```

Case 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, products)
        .then(data => {
            return res.send(data);
        });
        // NO CATCH block is here and next() is not used
}
```

reserve() throws an Error
sinse expect the rewards.id

Case 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, products)
        .then(data => {
            return res.send(data);
        });
        // NO CATCH block is here and next() is not used
}
```

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

Case 2. Uninformative Stack Trace

```
$(llnode) v8 bt

        frame #8: 0x2262d63840dd <exit>
        frame #9: 0x2262d64735e9 <builtin>
        frame #10: 0x2262d6392f12 process.on(this=0x18901b589ec1:<Object: process>,
         0x27a40a5f15d9:<unknown>, 0x27a40a5f11a9:<unknown>) at /demo/app.js:40:34
         fn=0x0af20adefc91
         frame #11: 0x2262d6472cc8 <builtin>
         frame #12: 0x2262d6392f12 emitTwo(this=0x3e5911182241:<undefined>,
          0x0af20adefc91:<function: process.on at /demo/app.js:40:34>, 0x3e59111822c1:<true>,
          0x18901b589ec1:<Object: process>, 0x27a40a5f15d9:<unknown>, 0x27a40a5f11a9:<unknown>)
          at events.js:123:17 fn=0x368a12a07421
         frame #13: 0x2262d6473a25 <builtin>
         frame #14: 0x2262d6392f12 emit(this=0x18901b589ec1:<Object: process>,
          0x368a12a373e1:<String: "unhandledRejecti...">) at events.js:155:44 fn=0x3ef321f07b19
```

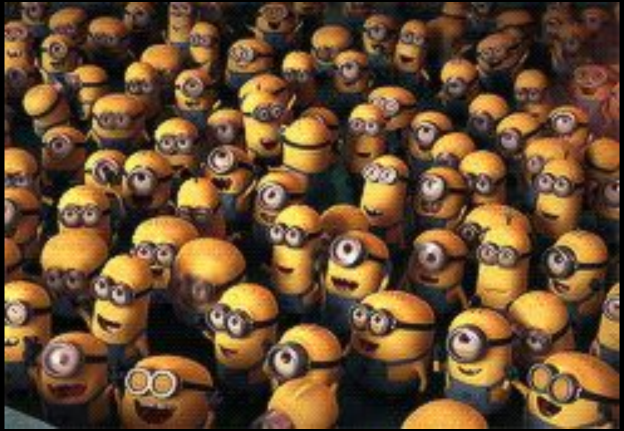
Case 2. Finding all req object instances

```
$(llnode) v8 findjsinstances IncomingMessage
          ...
          0x0af20adff4a1:<Object: IncomingMessage>
          0x25c630d8fe69:<Object: IncomingMessage>
          0x25c630d97861:<Object: IncomingMessage>
          0x25c630d9d811:<Object: IncomingMessage>
          0x25c630da9729:<Object: IncomingMessage>
          0x25c630daf8d9:<Object: IncomingMessage>
          0x25c630dbb669:<Object: IncomingMessage>
          0x25c630dc1591:<Object: IncomingMessage>
          0x25c630dd6c31:<Object: IncomingMessage>
          0x25c630ddcb59:<Object: IncomingMessage>
          0x25c630de2ed1:<Object: IncomingMessage>
          0x25c630de8d01:<Object: IncomingMessage>
          0x25c630deec29:<Object: IncomingMessage>
          ...
```

N

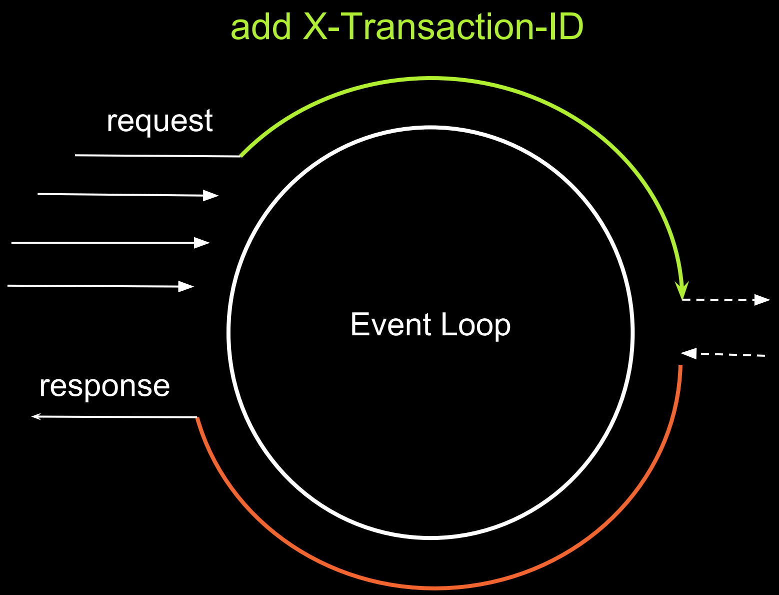
Case 2. Which request crashed the process

```
$(llnode) v8 findjsinstances IncomingMessage
        ...
        0x0af20adff4a1:<Object: IncomingMessage>
        0x25c630d8fe69:<Object: IncomingMessage>
        0x25c630d97861:<Object: IncomingMessage>
        0x25c630d9d811:<Object: IncomingMessage>
        0x25c630da9729:<Object: IncomingMessage>
        0x25c630daf8d9:<Object: IncomingMessage>
        0x25c630dbb669:<Object: IncomingMessage>
        0x25c630dc1591:<Object: IncomingMessage>
        0x25c630dd6c31:<Object: IncomingMessage>
        0x25c630ddcb59:<Object: IncomingMessage>
        0x25c630de2ed1:<Object: IncomingMessage>
        0x25c630de8d01:<Object: IncomingMessage>
        0x25c630deec29:<Object: IncomingMessage>
        ...
```

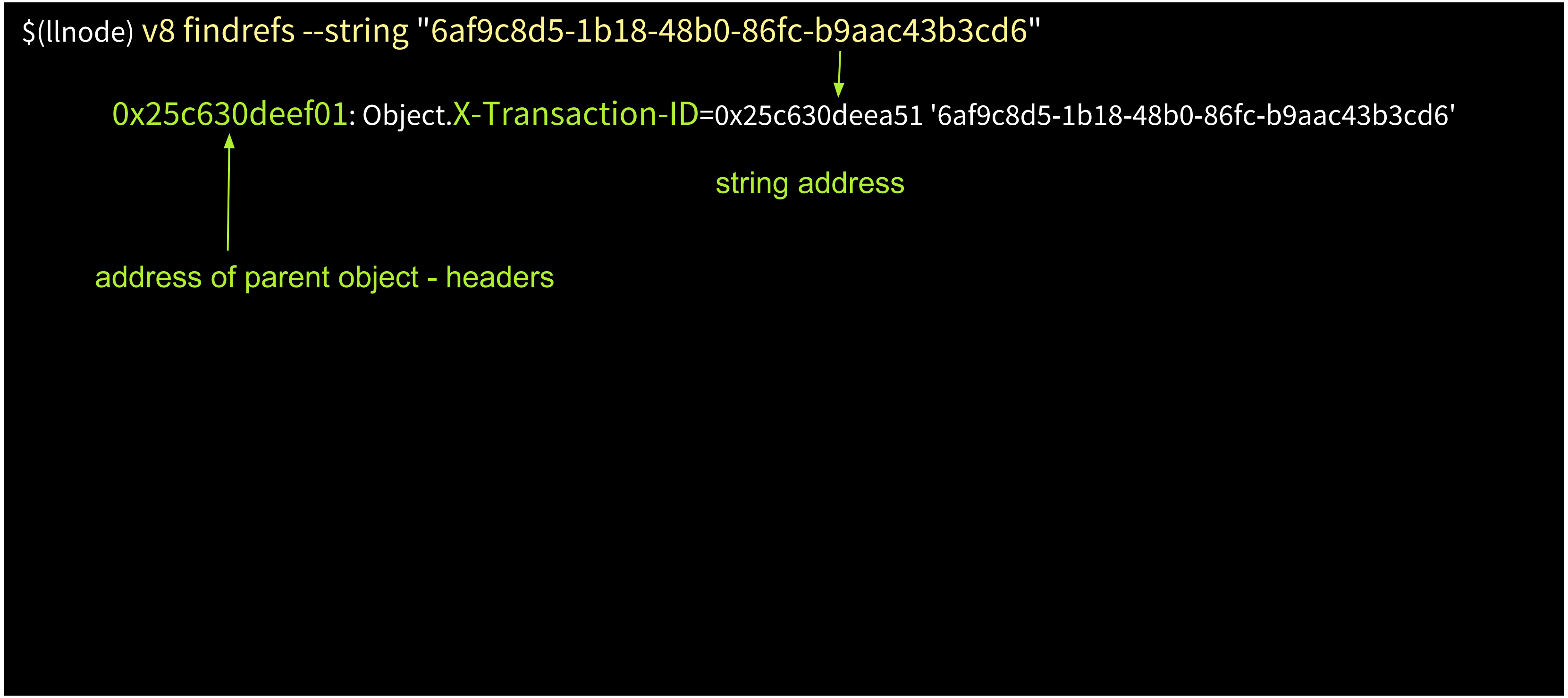


Case 2. List of the killed requests in current process

```
$(llnode) v8 findjsinstances IncomingMessage
          ...
          0x0af20adff4a1:<Object: IncomingMessage>
          0x25c630d8fe69:<Object: IncomingMessage>
          0x25c630d97861:<Object: IncomingMessage>
          0x25c630d9d811:<Object: IncomingMessage>
          0x25c630da9729:<Object: IncomingMessage>
          0x25c630daf8d9:<Object: IncomingMessage>
          0x25c630dbb669:<Object: IncomingMessage>
          0x25c630dc1591:<Object: IncomingMessage>
          0x25c630dd6c31:<Object: IncomingMessage>
          0x25c630ddcb59:<Object: IncomingMessage>
          0x25c630de2ed1:<Object: IncomingMessage>
          0x25c630de8d01:<Object: IncomingMessage>
          0x25c630deec29:<Object: IncomingMessage>
          ...
```



Case 2. Search headers by the value of request ID



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

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

Case 2. Search request by address of headers object

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

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

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

the request object that we need

Search for local variables

```
module.exports = class ReservationService {
  ...
  reserve (id, storeId, rewardsId, products) { // id = 'ff104cde3452332e0cc6'
     return Promise.all([
        this._rest.GET('user', id),
        this._db.getStore(storeId, rewardsId)
     ]).then(([user, rewardsInStore]) => {

        const data = { user, products, rewardsInStore };

        return this._rest.POST('reserve', data);
     });
}};
```

v8 findrefs --string "'ff104cde3452332e0cc6'"

v8 findrefs --name "rewardsInStore"

v8 findrefs --string "jeans-xxl"

$$\begin{tikzcd}
\text{Middleware} \arrow[r] & \text{Controller} \arrow[r] & \boxed{\text{Service}} \arrow[r, leftrightarrow] & \text{Storage}
\end{tikzcd}$$

node-report

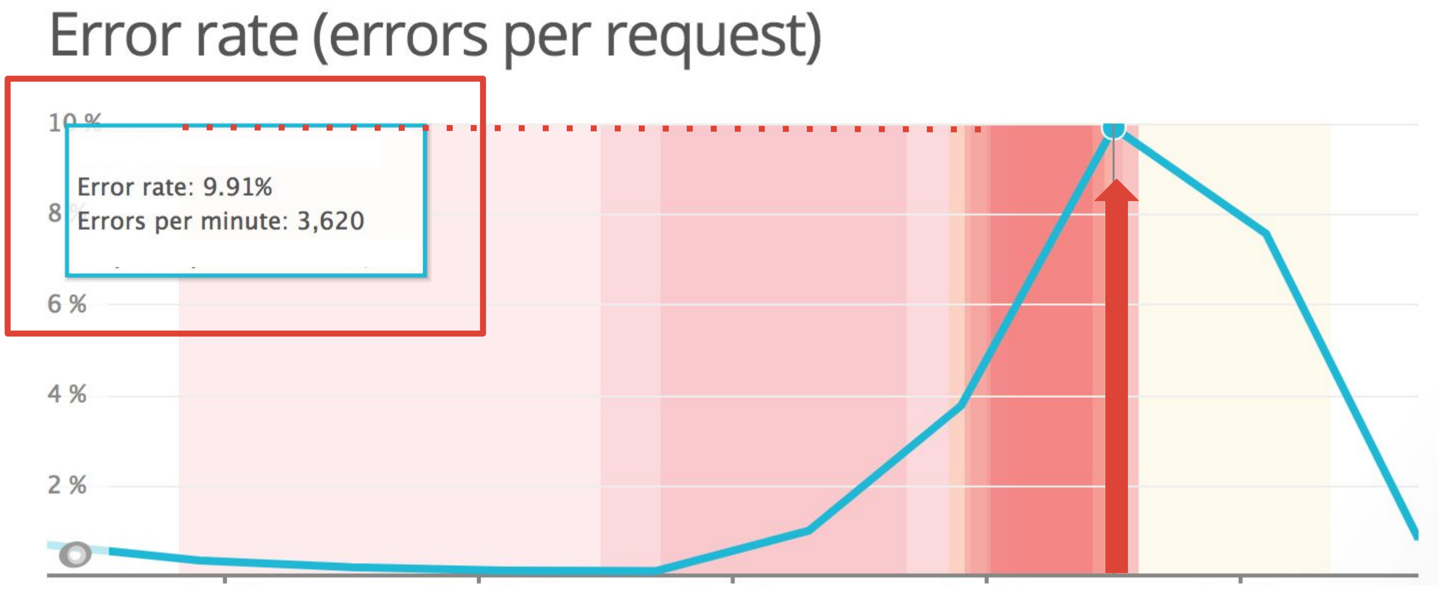
```
Event: exception, location: "OnUncaughtException"
Dump event time:  2017/09/27 00:57:26
Process ID: 35866
==== JavaScript Stack Trace ====================================
Object.fs.readFileSync (fs.js:1:1)
ProductController.reserve (/demo/controllers/ProductController.js:1:1)
==== Native Stack Trace ========================================
 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
```

What?

Where?

When?

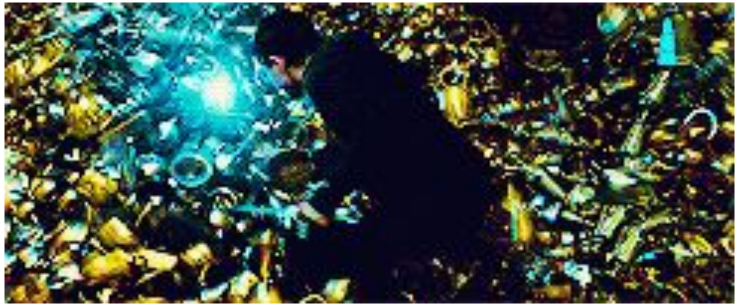
Errors in production



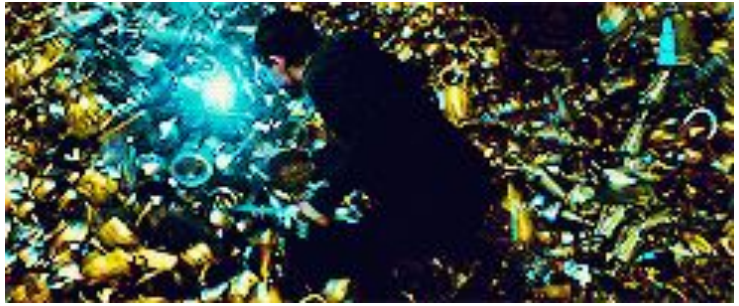
Error rate: 9.91%
Errors per minute: 3,620

I use core dump!

I use core dump!


Let’s the debugging begin!

I use core dump!

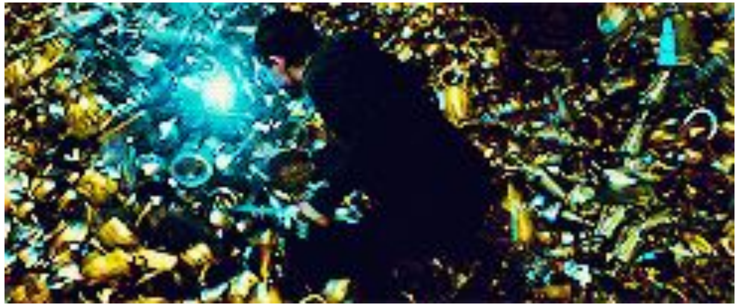


Let’s the debugging begin!

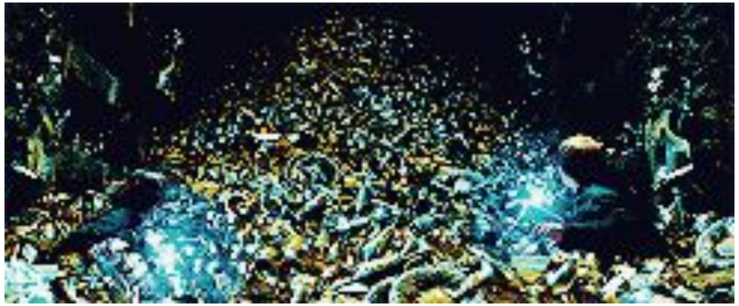
What is the difference between all of them?

Core dump flood

I use core dump!


Let’s the debugging begin!

What is the difference between all of them?


Boss: How much time do you need to solve the issue?

Error handler with Error Registry and gencore

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

npm install --save-dev gencore

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

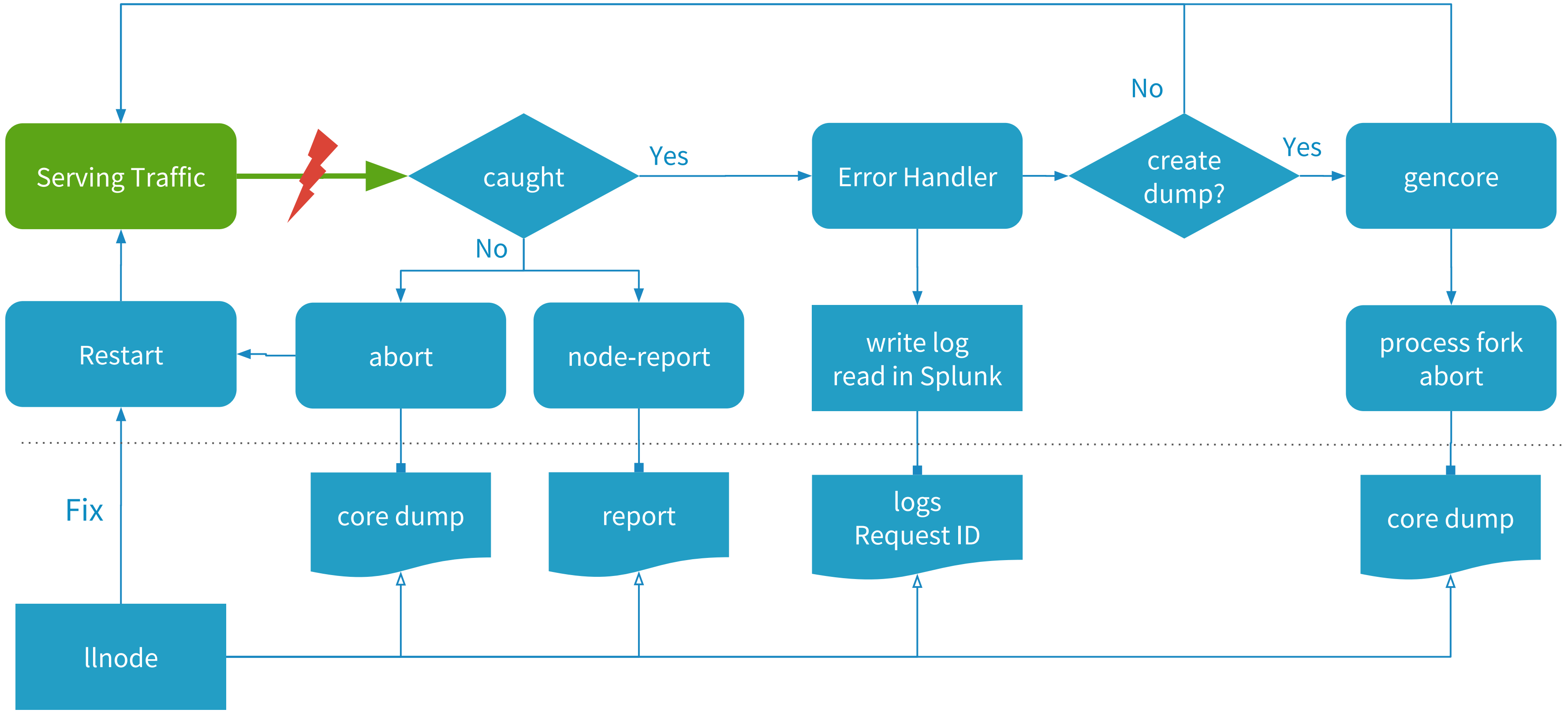
error selection logic

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

            gencore.collectCore((error, name) => {
                  logger.info(`Core dump created at: ${name}
                  for request: ${req.headers['X-Transaction-ID']}`);
            });
         }
            // … send 500 status
      })
      .catch(err => {  … // handleError(err, req, res, next);  })
});
```

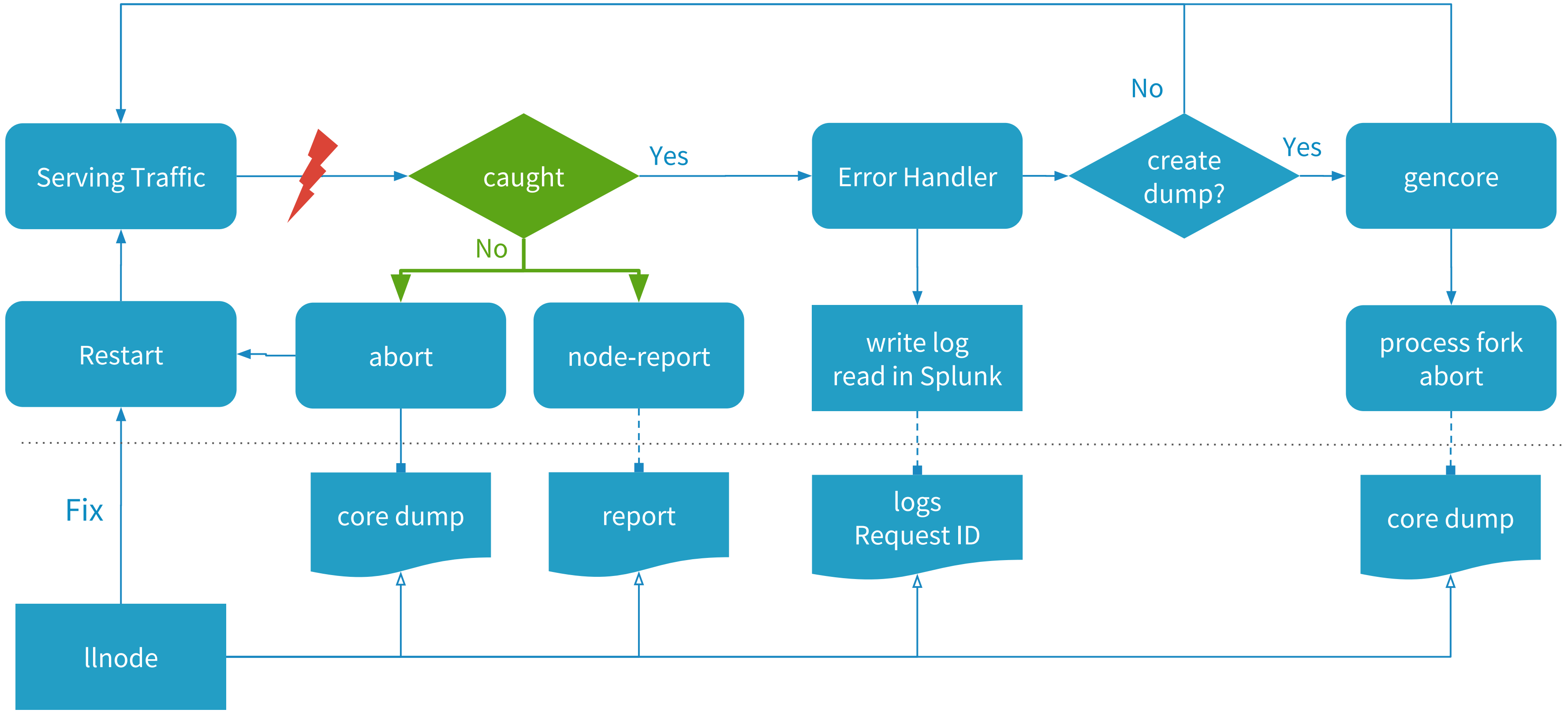
Makes process fork and creates core dump

* This is the simplified example

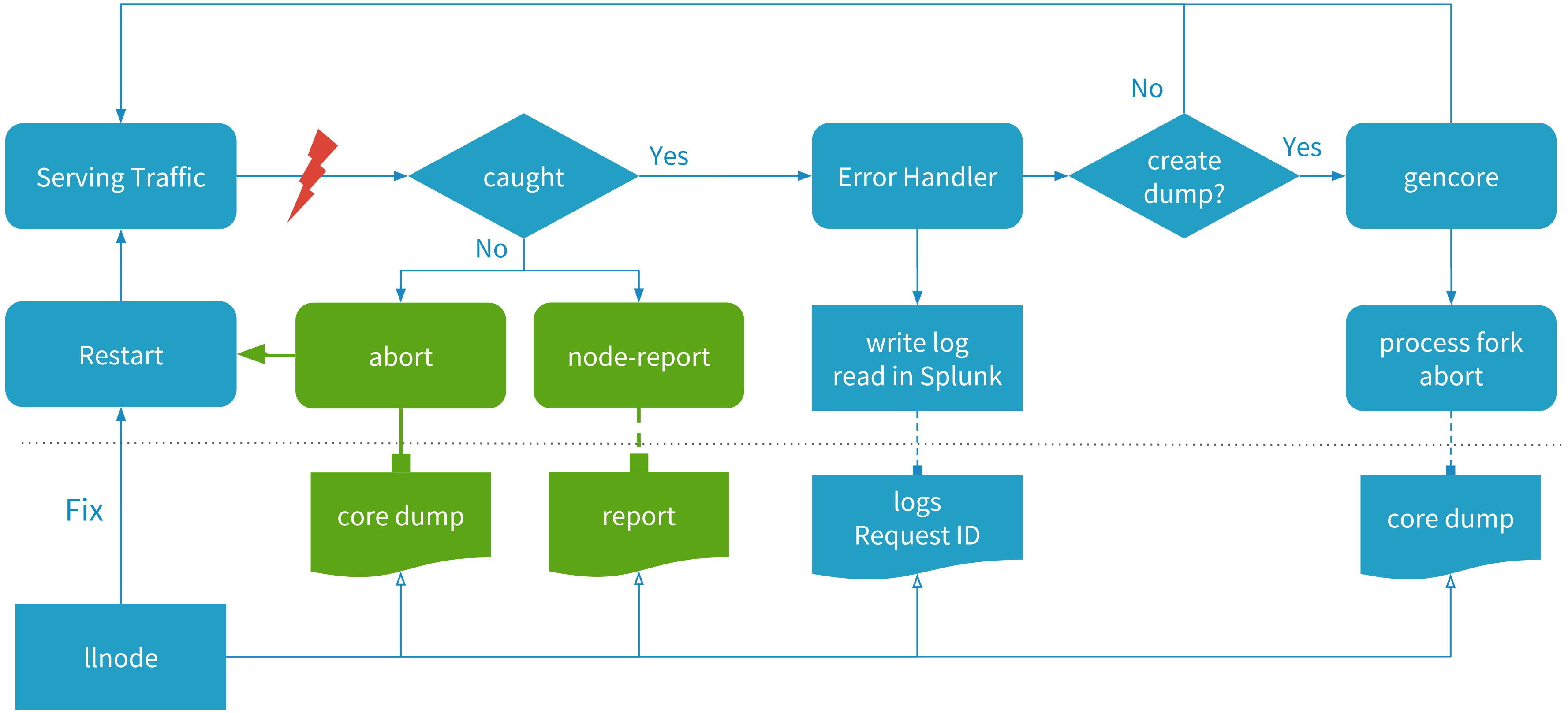
Core dump creation algorithm in production



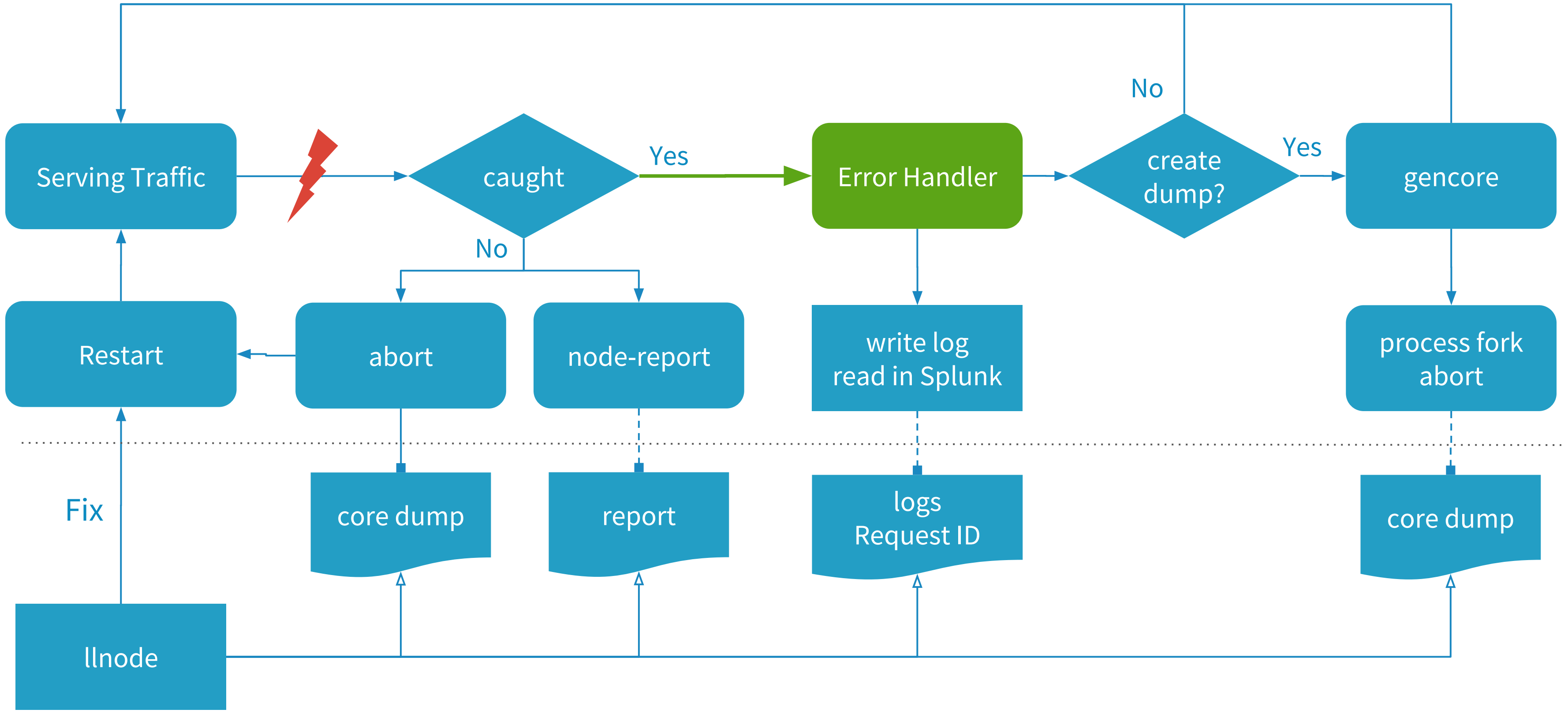
Core dump creation algorithm in production



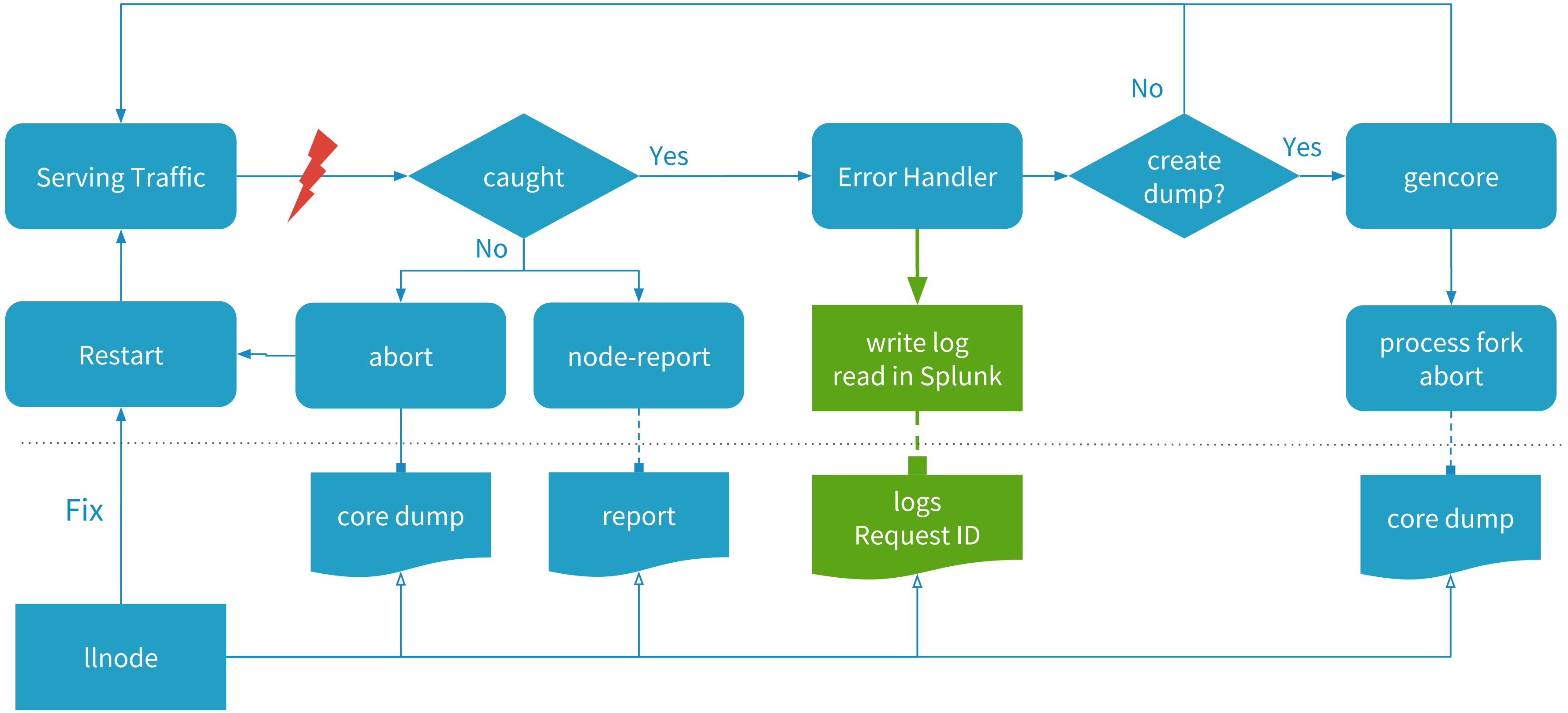
Core dump creation algorithm in production



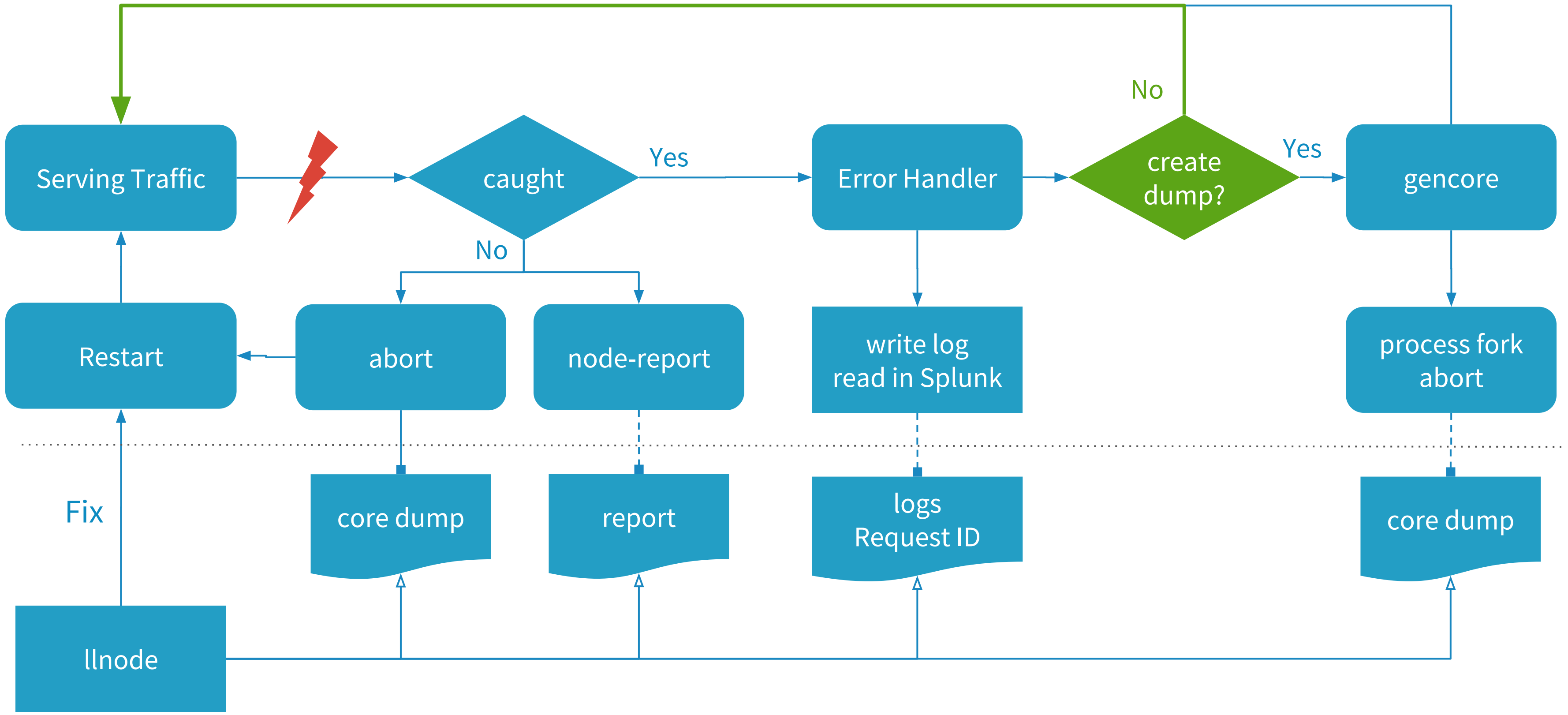
Core dump creation algorithm in production



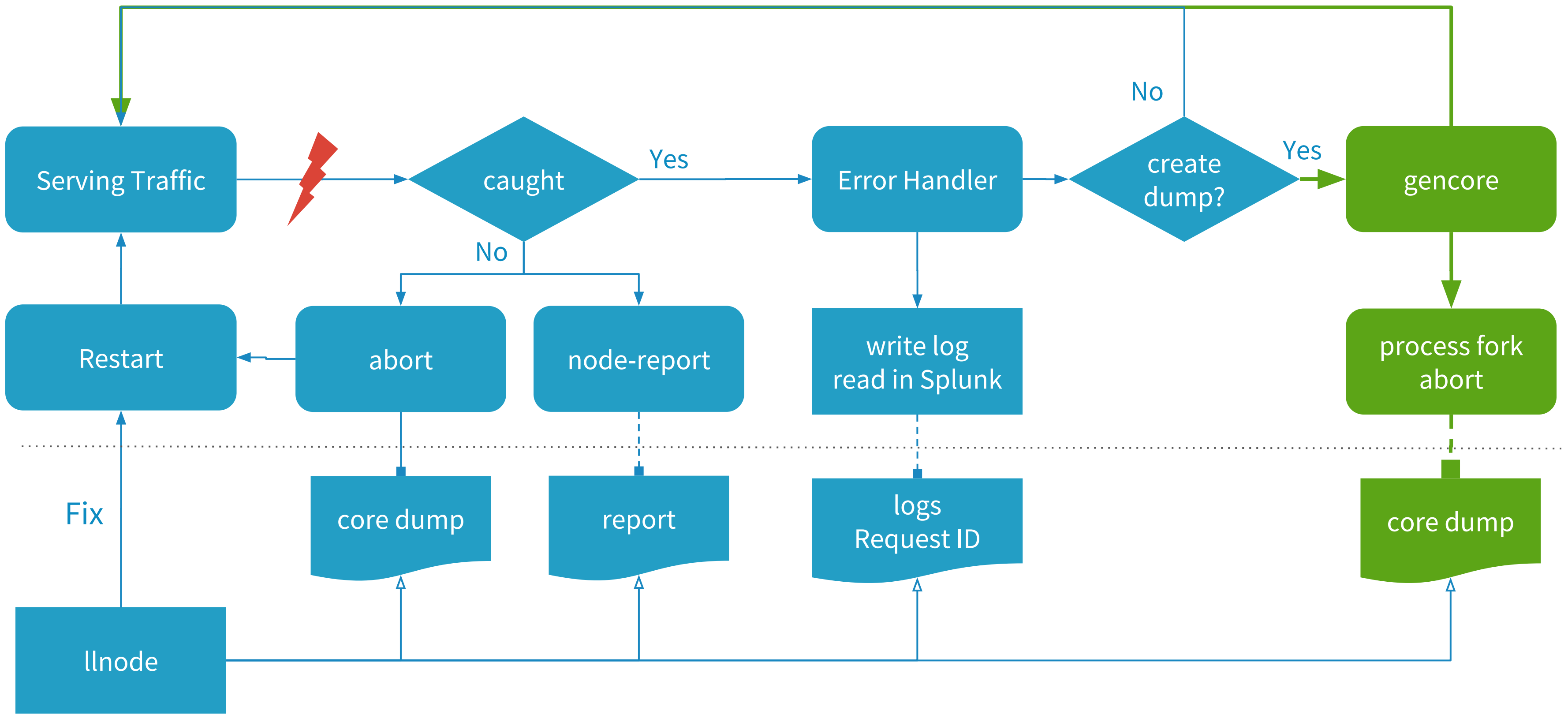
Core dump creation algorithm in production



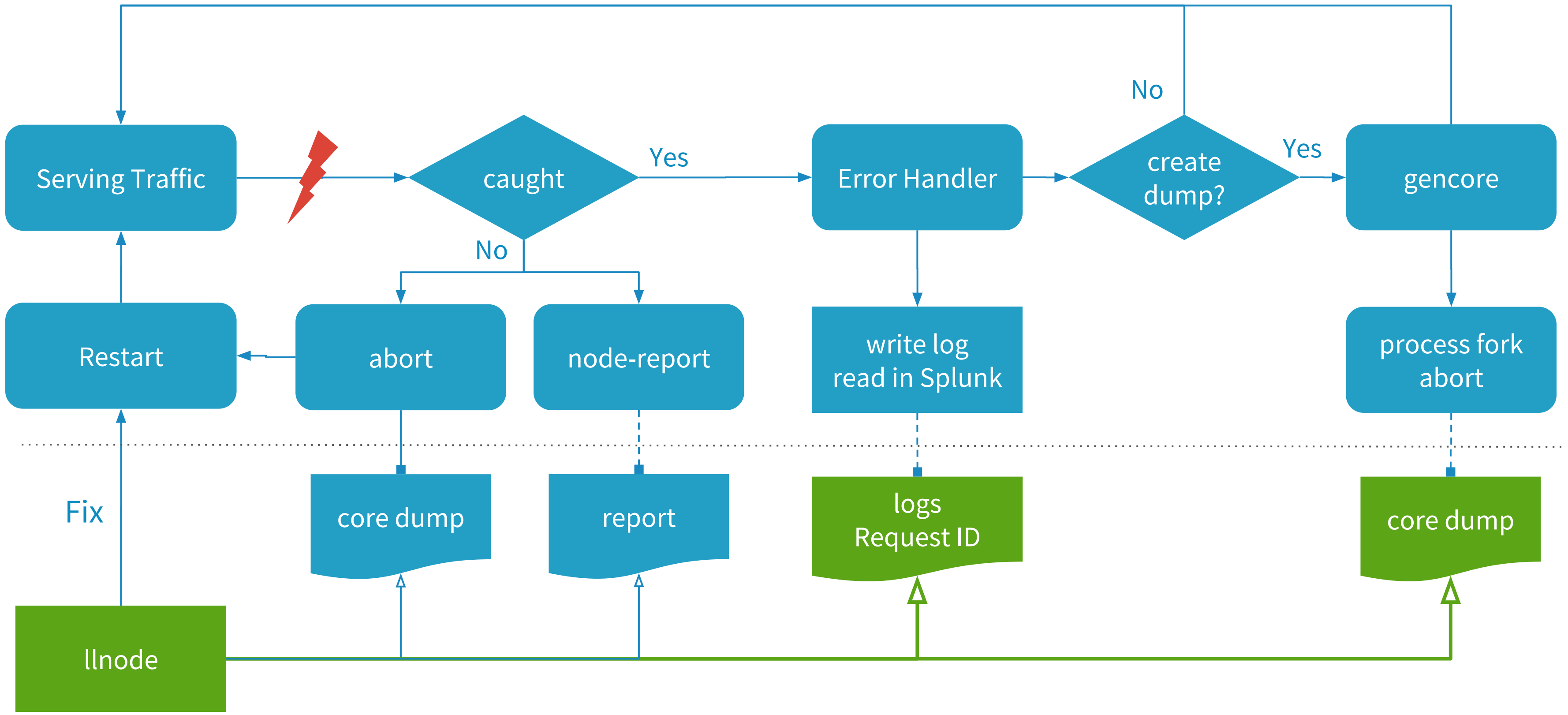
Core dump creation algorithm in production



Core dump creation algorithm in production



Core dump creation algorithm in production



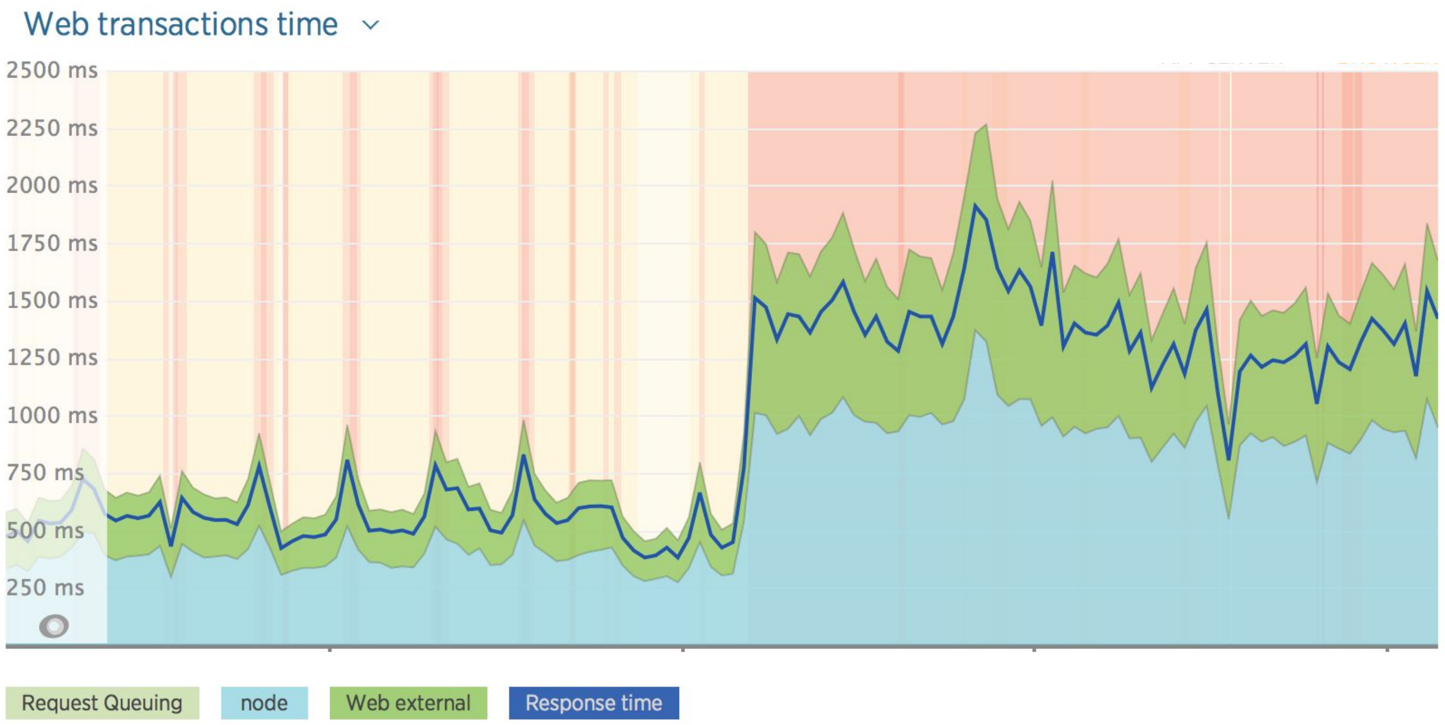
Tips

1. Follow to Error handling best practices
2. Call abort() only in centralized error handler
3. Use gencore
4. Use node-report to generate a report
5. Track requests using id – X-Transaction-ID or X-Request-ID (NGINX) header (or Zipkin headers)
6. Avoid core dump flood

Progress

1. Debugging in production ✔
2. Performance profiling
3. Search for memory leaks

2. The response of the Node.js web server has grown from ~300 to ~1500 ms

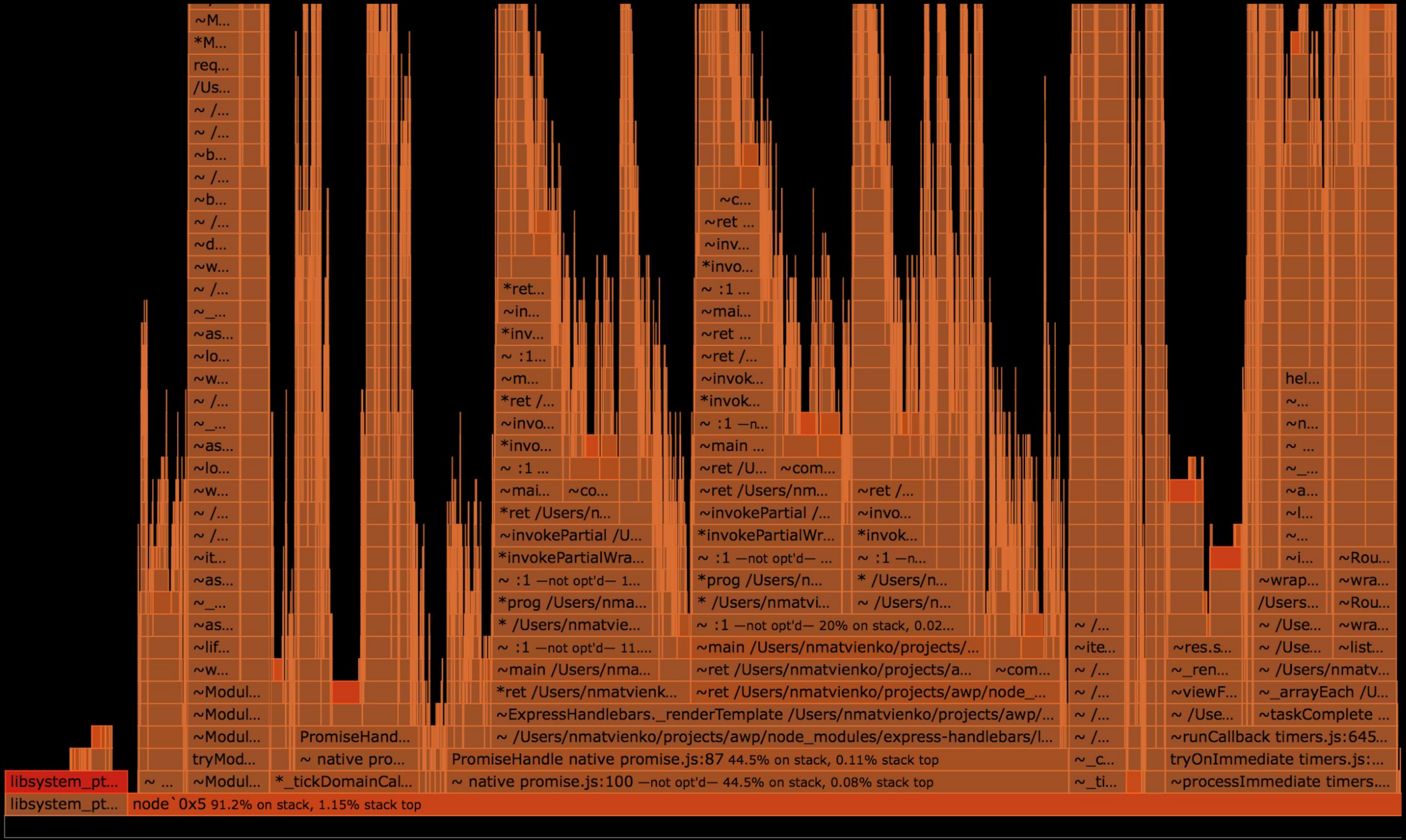


Profiling: node --prof(iler) + apache bench/JMeter

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

```
 [Summary]:
    ticks  total  nonlib   name
    2710   18.9%  19.0%   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&)
...
```

0x app.js

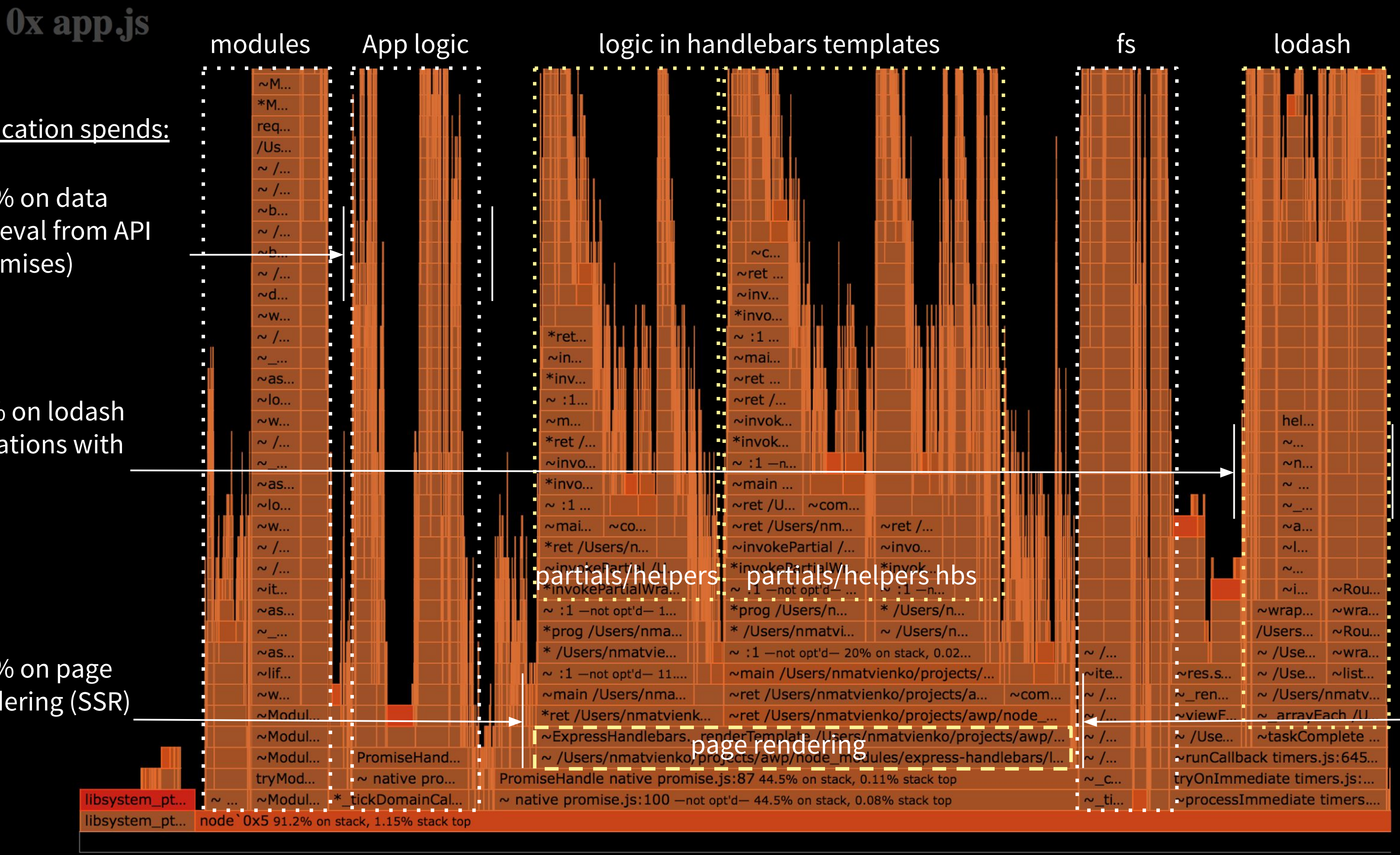

libsystem_pt...
node`0x5 91.2% on stack, 1.15% stack top
~ native promise.js:100 —not opt'd— 44.5% on stack, 0.08% stack top
PromiseHandle native promise.js:87 44.5% on stack, 0.11% stack top
~ /Users/nmatvienko/projects/awp/node_modules/express-handlebars/l...
~ExpressHandlebars._renderTemplate /Users/nmatvienko/projects/awp/...
~ret /Users/nmatvienko/projects/awp/node_...
~main /Users/nmatvienko/projects/...
~ :1 —not opt'd— 20% on stack, 0.02...
*_tickDomainCal...
PromiseHand...
~processImmediate timers....
tryOnImmediate timers.js:...
~runCallback timers.js:645...
~taskComplete ...
~_arrayEach /U...

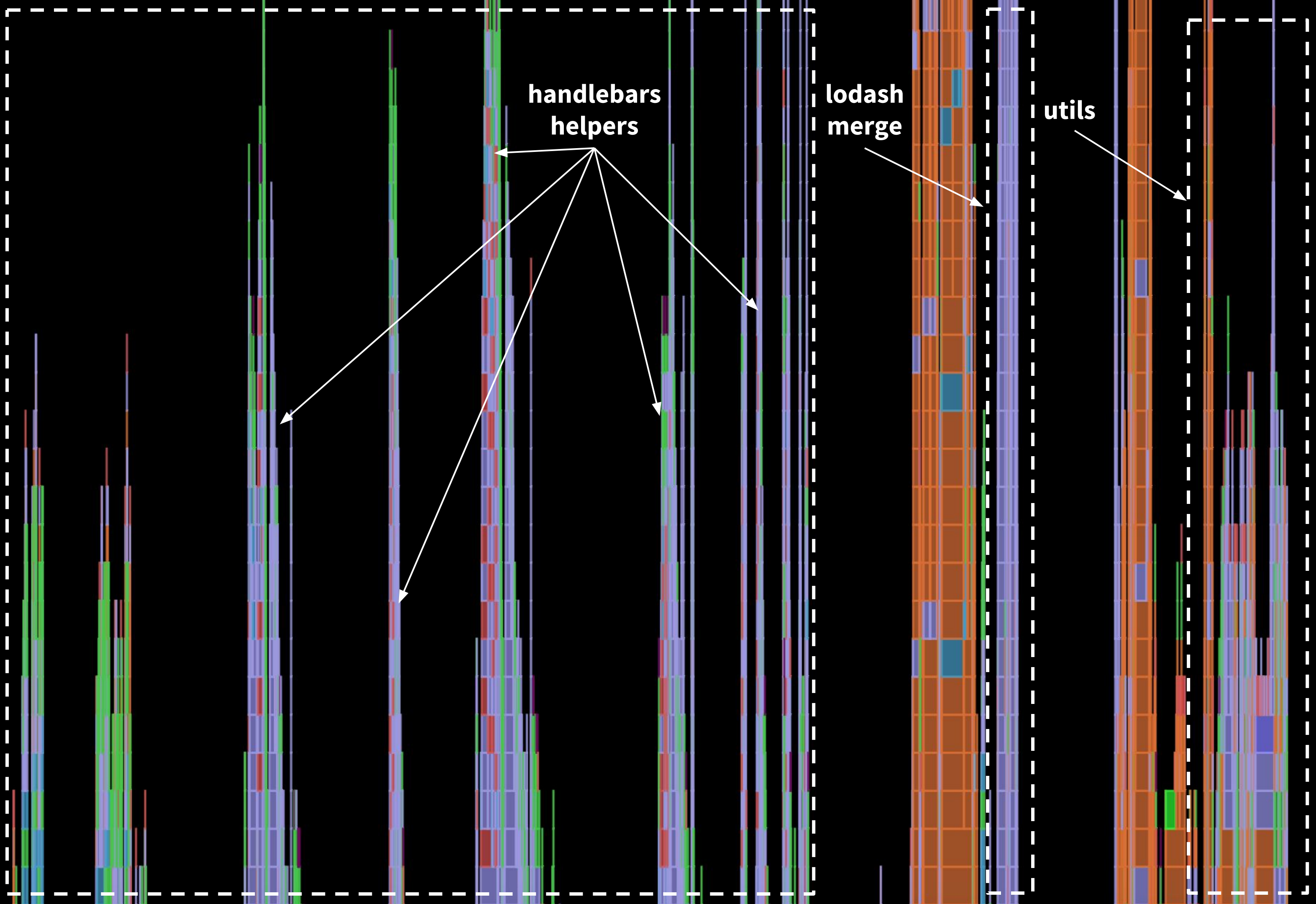
Application spends:

~10% on data retrieval from API (Promises)

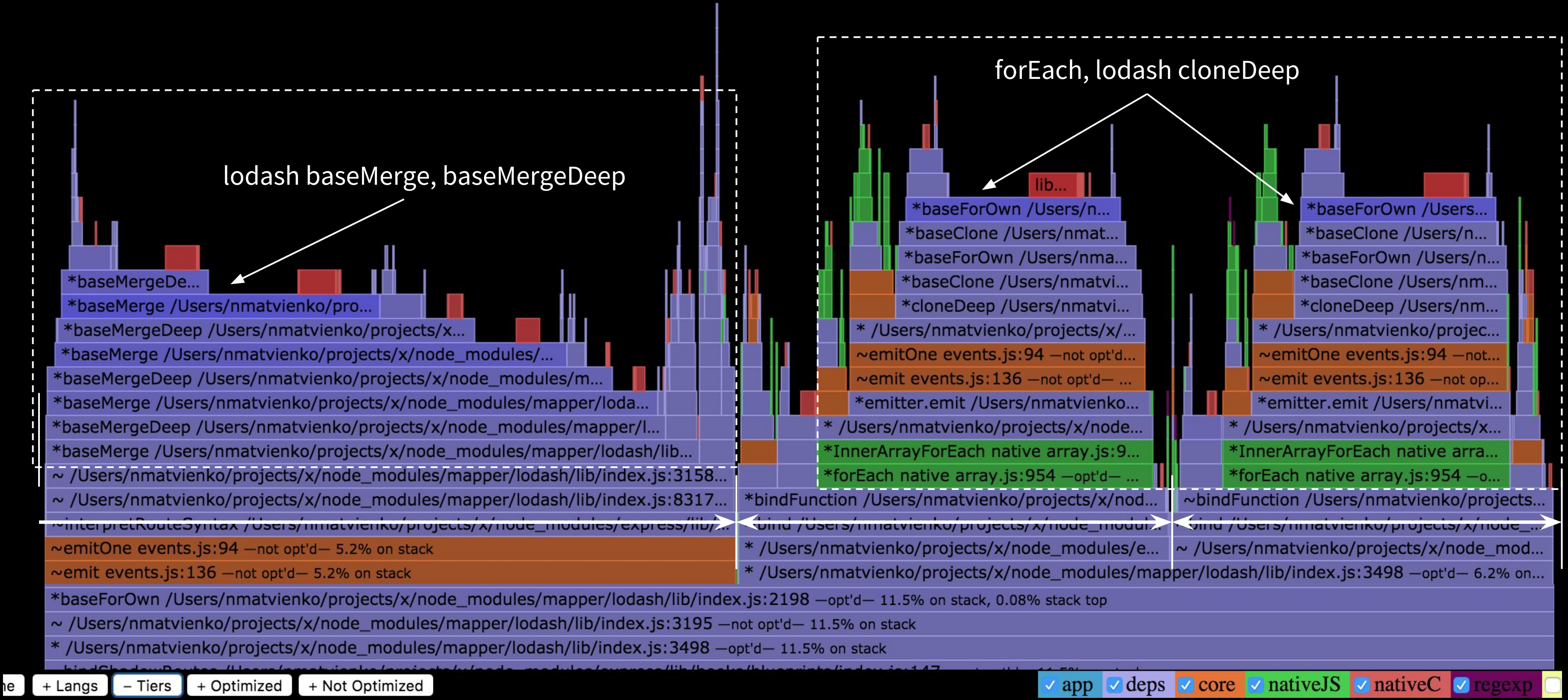
~20% on lodash operations with data

~65% on page rendering (SSR)




handlebars helpers
lodash merge
utils

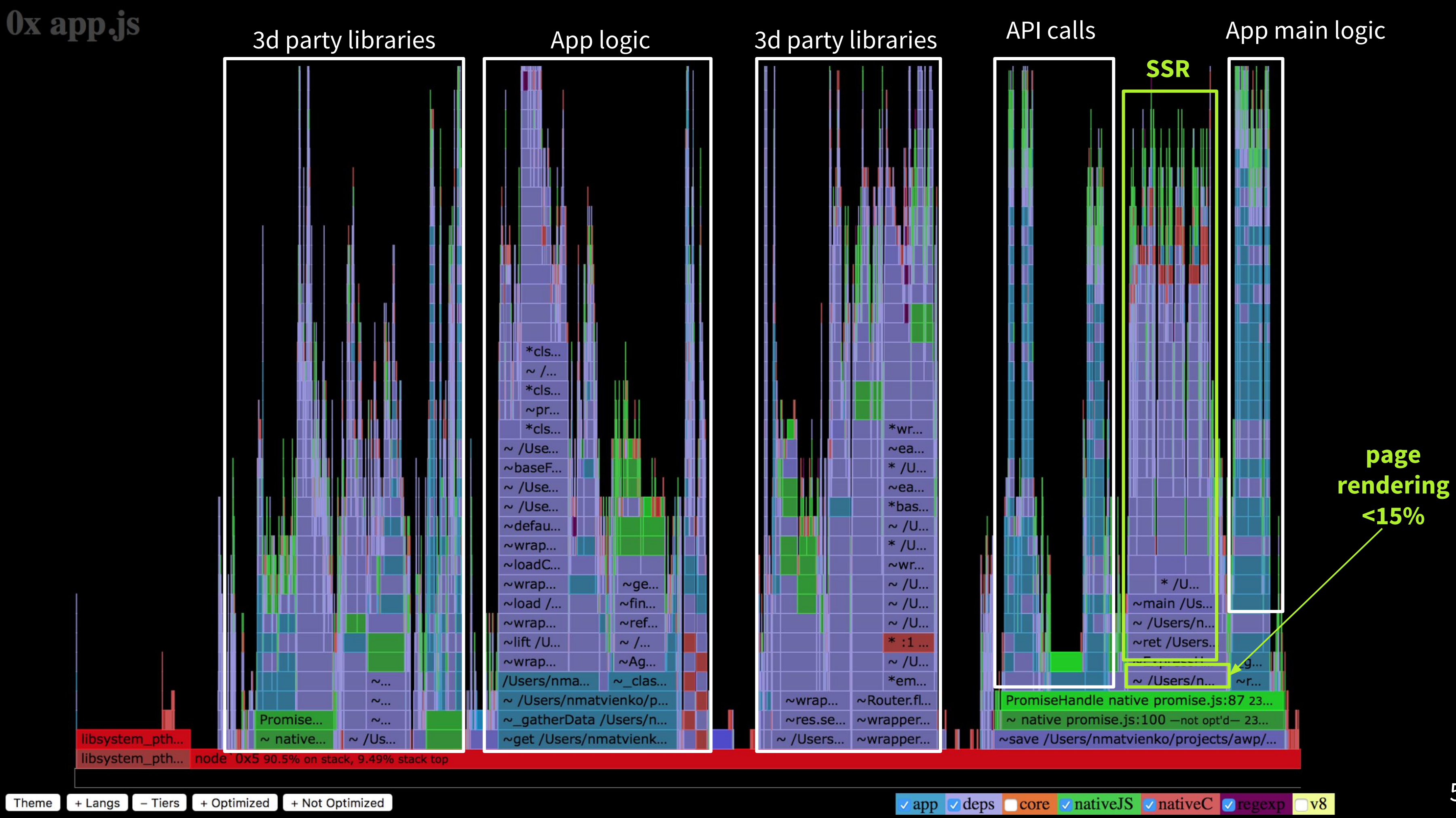
Data processing



Performance improvements

After performance profiling with perf, DTrace, 0x and bcc utils we:

- Reduced CPU intensive operations:
    a. lodash objects merge, deep clone
    b. long cycles
    c. JSON parse
- Removed logic from view templates and helpers
- Reviewed and updated npm dependencies
- Gradually implemented React SSR


0x app.js
3d party libraries
App logic
3d party libraries
API calls
App main logic
SSR
page rendering <15%
*cls...
~ /...
*cls...
~pr...
*cls...
~ /Use...
~baseF...
~ /Use...
~ /Use...
~defau...
~wrap...
~loadC...
~wrap...
~load /...
~wrap...
~lift /U...
~wrap...
/Users/nma...
~ /Users/nmatvienko/p...
~_gatherData /Users/n...
~get /Users/nmatvienk...
~ge...
~fin...
~ref...
~ /...
~Ag...
~_clas...
Promise...
~ native...
~ /Us...
*wr...
~ea...
* /U...
~ea...
*bas...
~ /U...
* /U...
~wr...
~ /U...
~ /U...
~ /U...
* :1 ...
~ /U...
*em...
~wrap...
~res.se...
~ /Users...
~Router.fl...
~wrapper...
~wrapper...
* /U...
~main /Us...
~ /Users/n...
~ret /Users...
~ /Users/n...
~r...
PromiseHandle native promise.js:87 23...
~ native promise.js:100 —not opt'd— 23...
~save /Users/nmatvienko/projects/awp/...
libsystem_pth...
libsystem_pth...
node`0x5 90.5% on stack, 9.49% stack top
Theme
+ Langs
– Tiers
+ Optimized
+ Not Optimized
✓ app
✓ deps
core
✓ nativeJS
✓ nativeC
✓ regexp
v8

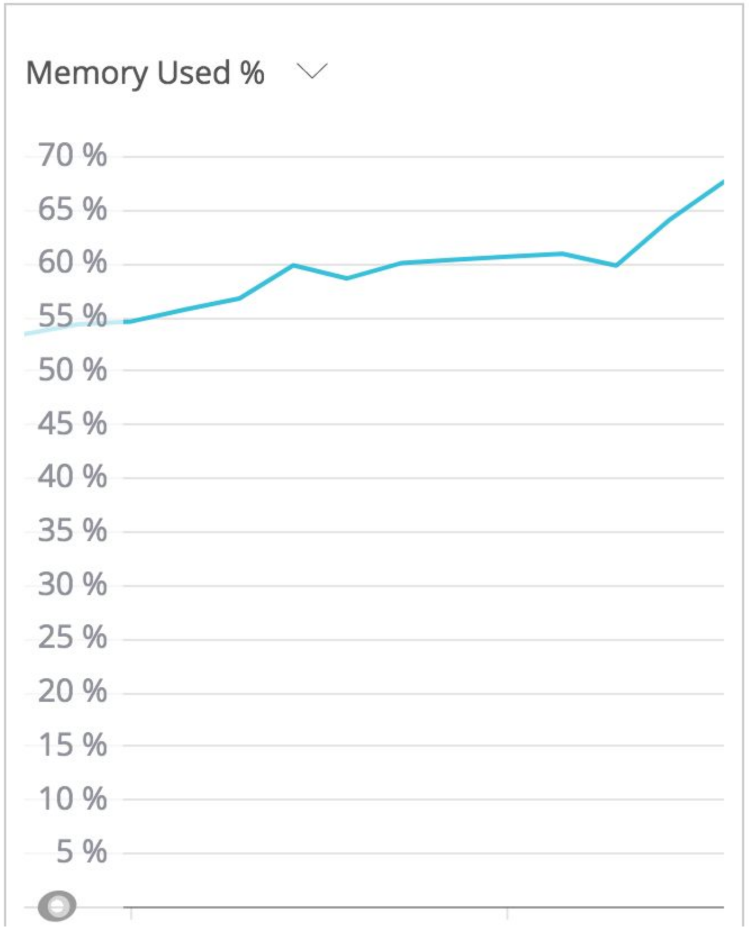
Tips

1. Performance should be a part of requirements.
2. Measure performance before embedding third-party libraries and after
3. Collect measurement results archive
4. Profile on staging/pre-production environments
5. Monitor diagnostic tools performance impact

3. Memory grows

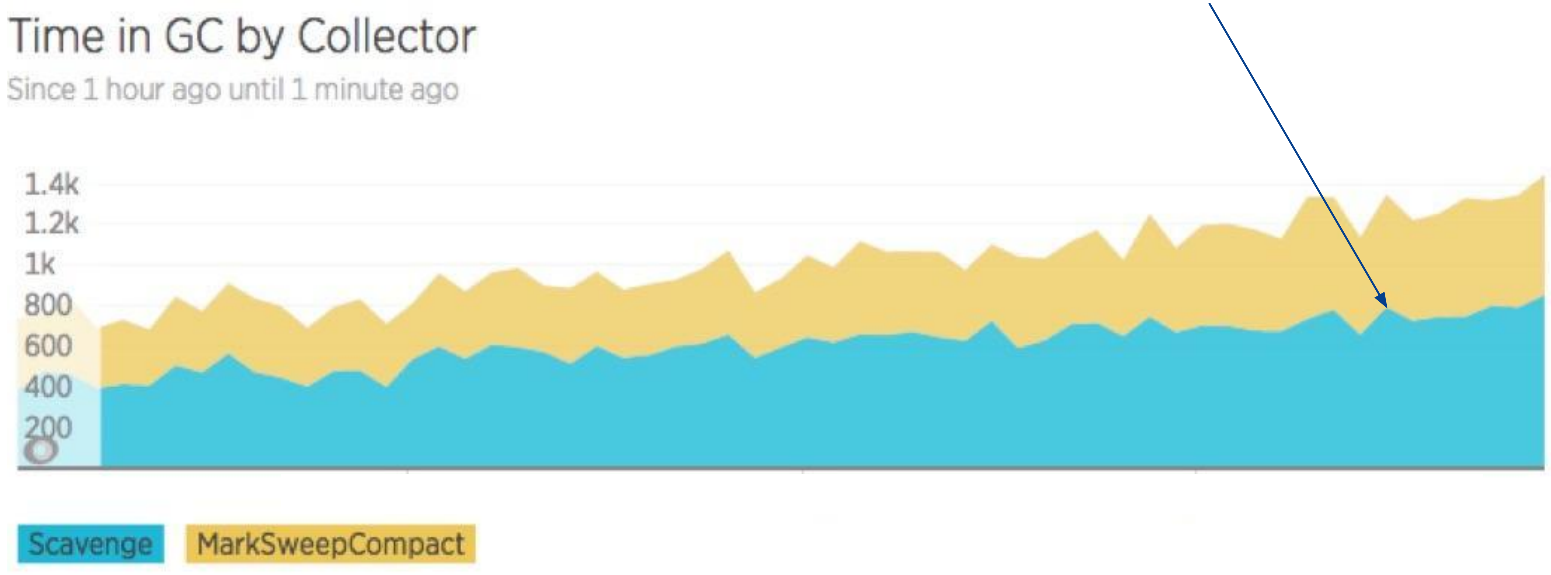
Ways to identify:

- Application Monitoring tools (N|Solid, NewRelic)
- DTrace, Instruments, perf
- node-memwatch
- GC Tracing
- Valgrind for C++ modules
- heapdump
- Fatal error
- and other...



Case: Excessive garbage collection. The creation a lot of new objects

GC rate grows in New Space (Scavenge GC)



Case: Scavenge GC in New Space.

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

Memory grows

Long Garbage collection

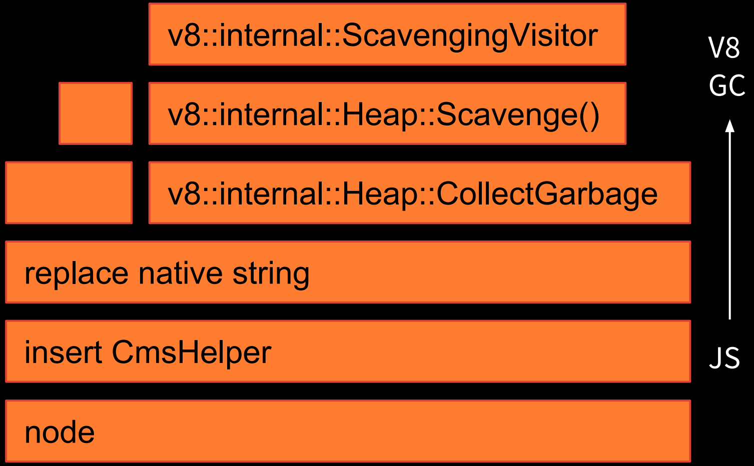
Interval

```
256201 ms: Scavenge 824.5 (1014.4) -> 811.4 (1016.2) MB, 16.9 / 0.0 ms [allocation failure].

256218 ms: Scavenge 838.8 (1019.3) -> 827.7 (1024.7) MB, 17.6 / 0.0 ms [allocation failure].

256236 ms: Scavenge 852.8 (1024.7) -> 842.9 (1026.5) MB, 11.0 / 0.0 ms [allocation failure].


256557 ms: Scavenge 869.8 (1201.6) -> 858.1 (1209.6) MB, 33.3 / 0.0 ms (+ 86.0 ms in 176 steps since
last GC) [allocation failure].

256925 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].
```

Profiling

Visual representation

Flame Graph


v8::internal::ScavengingVisitor
v8::internal::Heap::Scavenge()
v8::internal::Heap::CollectGarbage
replace native string
insert CmsHelper
node
V8 GC
JS

Text representation

Operation that requested memory allocation

```
{
 "name": "LazyCompile:~insert
/Users/nmatvienko/projects/x/src/cms/cmsHelper.js:806",
 "children":
 {
  "name": "LazyCompile:*replace native string.js:132",
  "children": [
  {
   "name": "Stub:CEntryStub",
   "children": [
   {
    "name": "v8::internal::Scavenger::ScavengeObjectSlow"
   }
```

Garbage collection

```
{
"name": "LazyCompile:~ /Users/nmatvienko/projects/x/api/domain/product/mappers/ProductMapper.js:11",
 "value": 1240,
 "top": 2,
 "children": [
…                           {
                            "name": "LazyCompile:*map native array.js:994",
                            "value": 1025,
                            "top": 20,
                            "children": [
                            ...
                                                {
                                                 "name": "Stub:CEntryStub",
                                                 "value": 41,
                                                 "top": 32,
                                                 "children": [
                                                   {
                                                    "name": "node`v8::internal::Scavenger::ScavengeObject",
                                                    "value": 2,
                                                    "top": 2
                                                   },
                                                   {
                                                    "name": "node`v8::internal::Space::AllocationStep",
                                                    "value": 1,
                                                    "top":  }
                                                 ]
```

Operation that requested allocation of memory

Garbage collection in New Space

GC tracing in New Space after improvements

Memory doesn’t grow

Garbage collection time

```
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].
```

Case: Many references, missed closures and timers. GC in Old Space

Long garbage collection

```
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)


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)
```

Get all JS objects from the Heap using llnode

```
$(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)
        250     21920      Timeout
      31043   369864     (String)
      ...
```

Find all IncomingMessage instances

```
$(llnode) v8 findjsinstances IncomingMessage

        0xaf20adff4a1:<Object: IncomingMessage>
        0x25c630d8fe69:<Object: IncomingMessage>
        0x25c630d97861:<Object: IncomingMessage>
        0x25c630d9d811:<Object: IncomingMessage>
        0x25c630da37c9:<Object: IncomingMessage>
        0x25c630daf8d9:<Object: IncomingMessage>
        0x25c630db5741:<Object: IncomingMessage>
        0x25c630dbb669:<Object: IncomingMessage>
        0x25c630dc1591:<Object: IncomingMessage>
        0x25c630dd6c31:<Object: IncomingMessage>
        0x27a40a588a11:<Object: IncomingMessage>
        0x27a40a58e841:<Object: IncomingMessage>
        0x27a40a5a38f1:<Object: IncomingMessage>
        0x27a40a5a9819:<Object: IncomingMessage>
```

Find req object holders

```
$(llnode) v8 findjsinstances IncomingMessage

        0xaf20adff4a1:<Object: IncomingMessage>
        0x25c630d8fe69:<Object: IncomingMessage>
        0x25c630d97861:<Object: IncomingMessage>
        0x25c630d9d811:<Object: IncomingMessage>
        0x25c630da37c9:<Object: IncomingMessage>
        0x25c630daf8d9:<Object: IncomingMessage>
        0x25c630db5741:<Object: IncomingMessage>
        0x25c630dbb669:<Object: IncomingMessage>
        0x25c630dc1591:<Object: IncomingMessage>
        0x25c630dd6c31:<Object: IncomingMessage>
        0x27a40a588a11:<Object: IncomingMessage>
        0x27a40a58e841:<Object: IncomingMessage>
        0x27a40a5a38f1:<Object: IncomingMessage>
        0x27a40a5a9819:<Object: IncomingMessage>
```

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

        0x25c630d8ada1: (Array)[250]=0xaf20adff4a1
        0x25c630d83329: ServerResponse.req=0xaf20adff4a1
```

Find all IncomingMessage instances

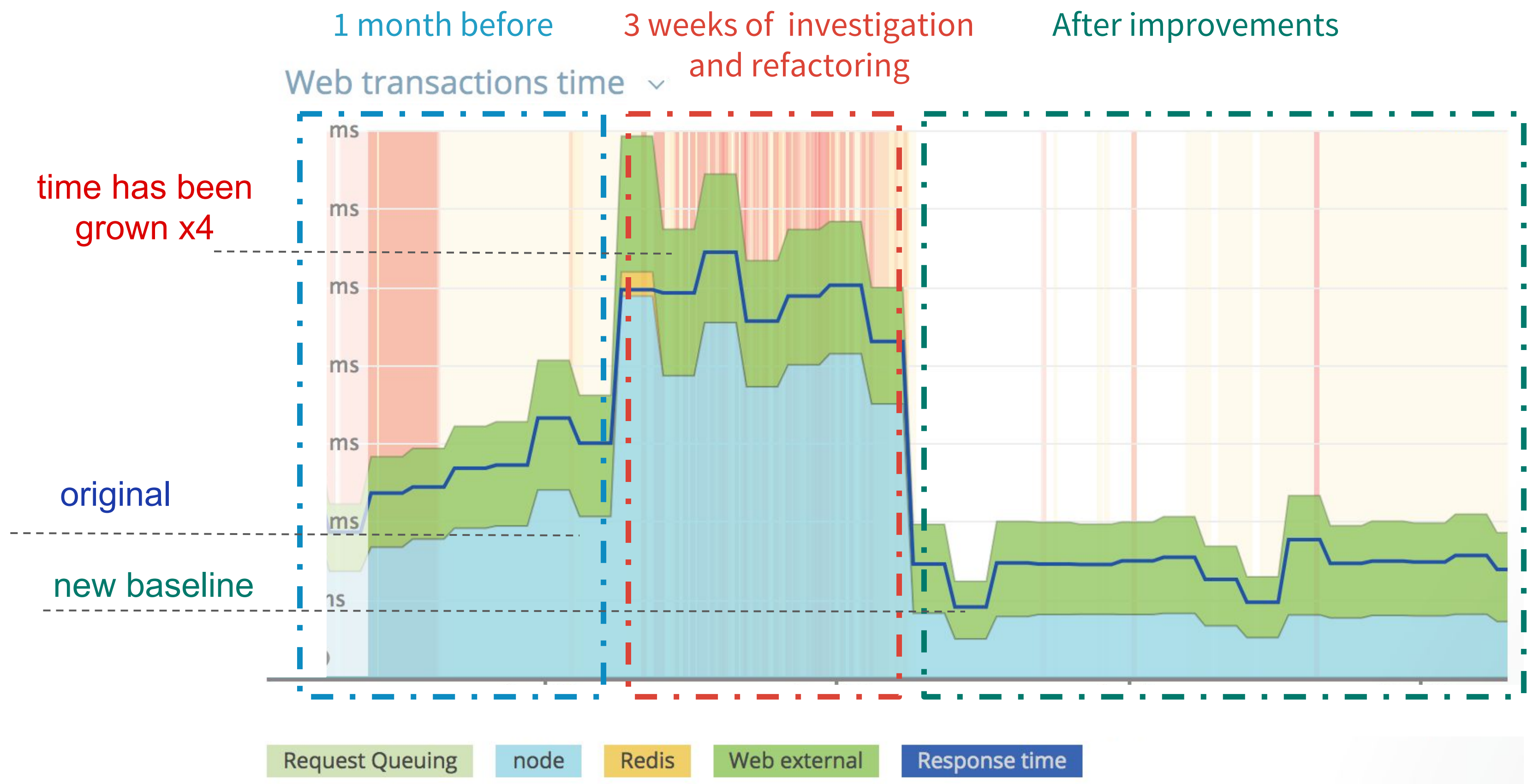
```
$(llnode) v8 findjsinstances IncomingMessage

        0xaf20adff4a1:<Object: IncomingMessage>
        0x25c630d8fe69:<Object: IncomingMessage>
        0x25c630d97861:<Object: IncomingMessage>
        0x25c630d9d811:<Object: IncomingMessage>
        0x25c630da37c9:<Object: IncomingMessage>
        0x25c630daf8d9:<Object: IncomingMessage>
        0x25c630db5741:<Object: IncomingMessage>
        0x25c630dbb669:<Object: IncomingMessage>
        0x25c630dc1591:<Object: IncomingMessage>
        0x25c630dd6c31:<Object: IncomingMessage>
        0x27a40a588a11:<Object: IncomingMessage>
        0x27a40a58e841:<Object: IncomingMessage>
        0x27a40a5a38f1:<Object: IncomingMessage>
        0x27a40a5a9819:<Object: IncomingMessage>
```

```
$(llnode) v8 findrefs -v 0x0af20adff4a1

        0x25c630d8ada1: (Array)[250]=0xaf20adff4a1
        0x25c630d83329: ServerResponse.req=0xaf20adff4a1

$(llnode) v8 findrefs -v 0x25c630d8ada1

0x3b312aee27d9 ProfileMiddleware._requests =0x25c630d8ada1
```

Tips

1. Control the Life Cycle of app Objects and Sessions
2. Trace Scavenge GC in Profiler Stacks to find allocation cause (bcc/tools/stackcount.py)
3. Use D8 to check Javascript behavior in V8 (%DebugTrackRetainingPath(x))
4. Call GC manually to check memory allocation
5. Write memory usage tests for code and libraries with leakage
6. Create heapdump/ coredump for deep investigation

Result


1 month before
3 weeks of investigation and refactoring
After improvements
Web transactions time
ms
ms
ms
ms
ms
ms
ms
time has been grown x4
original
new baseline
Request Queuing
node
Redis
Web external
Response time

Resources

Nikolay Matvienko

mail: matvi3nko@gmail.com

twitter: @matvi3nko

github: @nickkooper

Linked-in nikolaymatvienko

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