Profiling Node.js Application in Production Environment

Node.js在线性能调优与故障排查

by @朴灵

自我介绍

- @朴灵
- 来自阿里云 [alinode] 团队
- JacksonTian@GitHub
- 《深入浅出Node.js》作者
- 目前从事Node.js APM产品开发

大概三件事

- CPU ~99%
- Memory leaks
- GC frequently(stop the world)

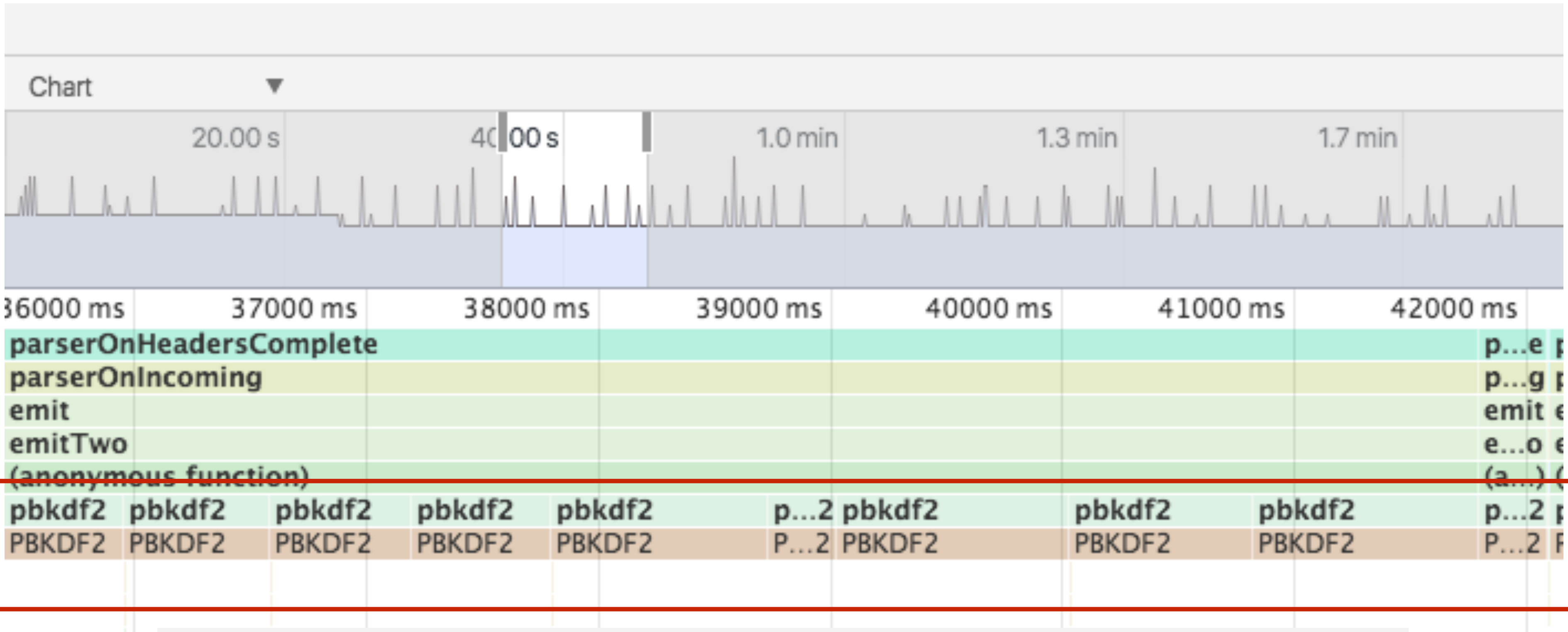
case 1: CPU issue

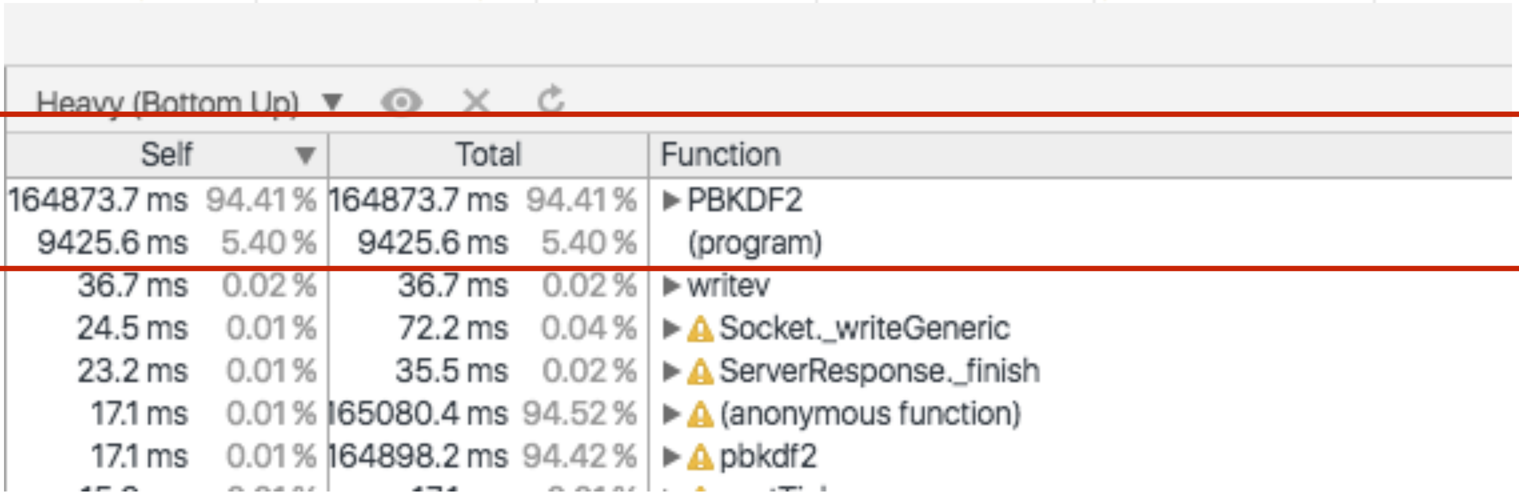
```
http.createServer(function (req, res) {

  var password = 'fjalsdfjas';
  var salt = crypto.randomBytes(128).toString('base64');
  var hash = crypto.pbkdf2Sync(password, salt, 10000, 512);

  res.writeHead(200);
  res.end('Hello world!\n');

}).listen(8989);
```

Solution

- 分析资料：*.cpuprofile
- 生成工具：
  - v8-profiler/node-inspector
  - alinode
- 分析工具：chrome dev tools

分析表现



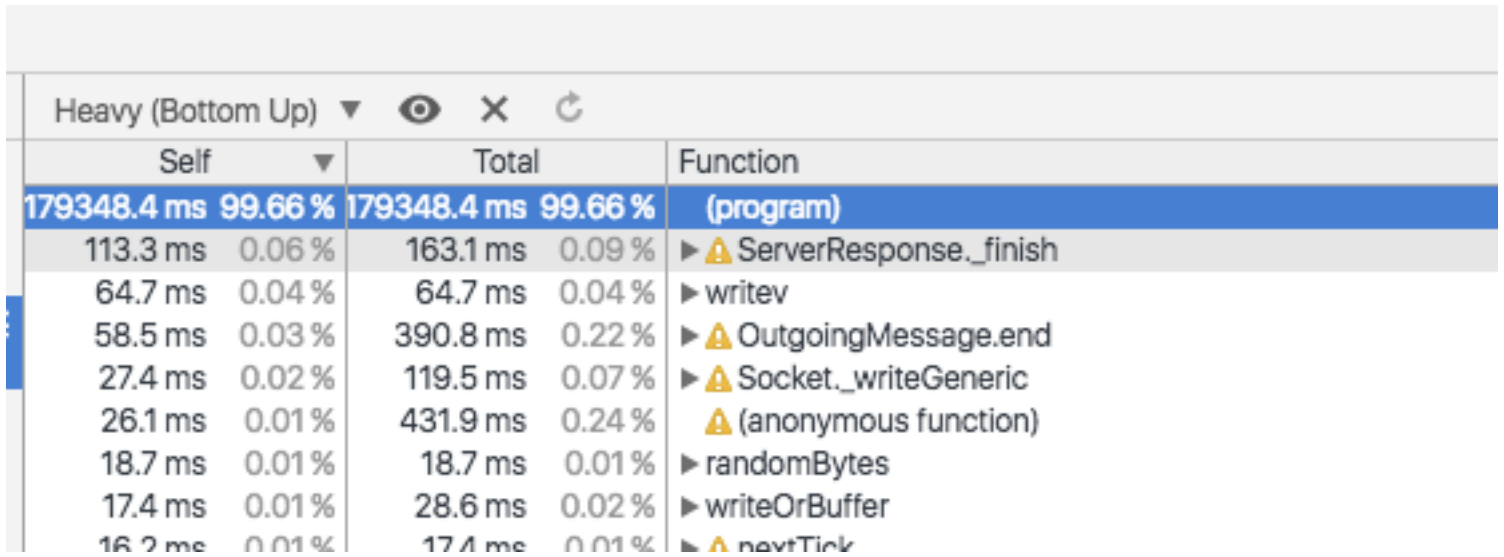

Heavy (Bottom Up)

Self		Total		Function
164873.7 ms	94.41 %	164873.7 ms	94.41 %	▶ PBKDF2
9425.6 ms	5.40 %	9425.6 ms	5.40 %	(program)
36.7 ms	0.02 %	36.7 ms	0.02 %	▶ writev
24.5 ms	0.01 %	72.2 ms	0.04 %	▶ ⚠ Socket._writeGeneric
23.2 ms	0.01 %	35.5 ms	0.02 %	▶ ⚠ ServerResponse._finish
17.1 ms	0.01 %	165080.4 ms	94.52 %	▶ ⚠ (anonymous function)
17.1 ms	0.01 %	164898.2 ms	94.42 %	▶ ⚠ pbkdf2

case 1: CPU issue

```
http.createServer(function (req, res) {

  var password = 'fjalsdfjas';
  var salt = crypto.randomBytes(128).toString('base64');

  crypto.pbkdf2(password, salt, 10000, 512, function(err, hash) {
    res.writeHead(200);
    res.end('Hello world!\n');
  });

}).listen(8989);
```

分析表现



Heavy (Bottom Up)

Self		Total		Function
179348.4 ms	99.66 %	179348.4 ms	99.66 %	(program)
113.3 ms	0.06 %	163.1 ms	0.09 %	▶ ⚠ ServerResponse._finish
64.7 ms	0.04 %	64.7 ms	0.04 %	▶ writev
58.5 ms	0.03 %	390.8 ms	0.22 %	▶ ⚠ OutgoingMessage.end
27.4 ms	0.02 %	119.5 ms	0.07 %	▶ ⚠ Socket._writeGeneric
26.1 ms	0.01 %	431.9 ms	0.24 %	⚠ (anonymous function)
18.7 ms	0.01 %	18.7 ms	0.01 %	▶ randomBytes
17.4 ms	0.01 %	28.6 ms	0.02 %	▶ writeOrBuffer
16.2 ms	0.01 %	17.4 ms	0.01 %	▶ ⚠ nextTick

case 2: Memory leaks

```
function LeakingClass() {}

var leaks = [];

var http = require('http');

http.createServer(function (req, res) {
  leaks.push(new LeakingClass);

  res.writeHead(200);
  res.end('Hello world!\n');
}).listen(8989);
```

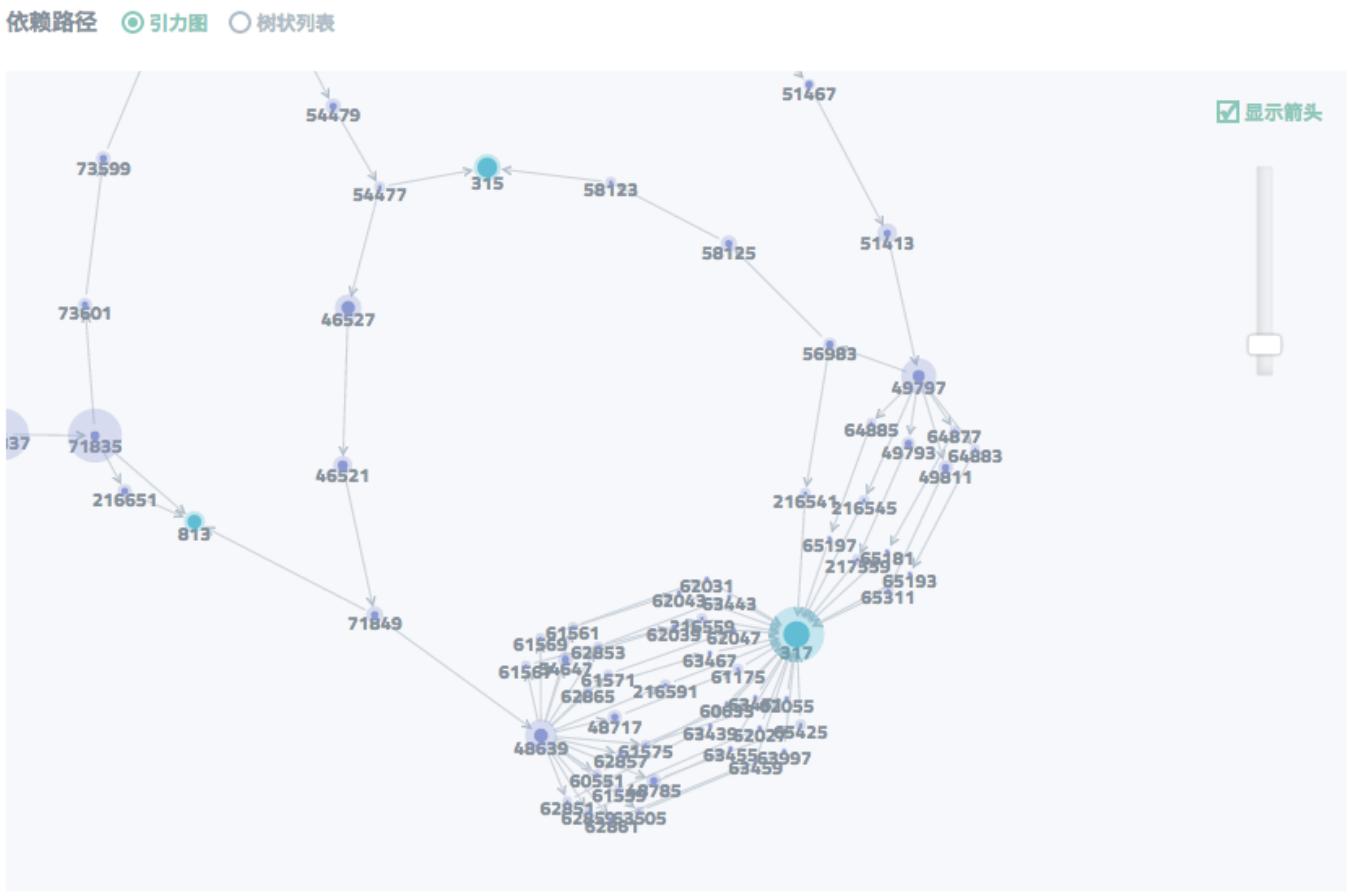
case 2: Memory leaks

```
var leaks = [];

var http = require('http');

http.createServer(function (req, res) {
  leaks.push({id: i});

  res.writeHead(200);
  res.end('Hello world!\n');
}).listen(8989);
```

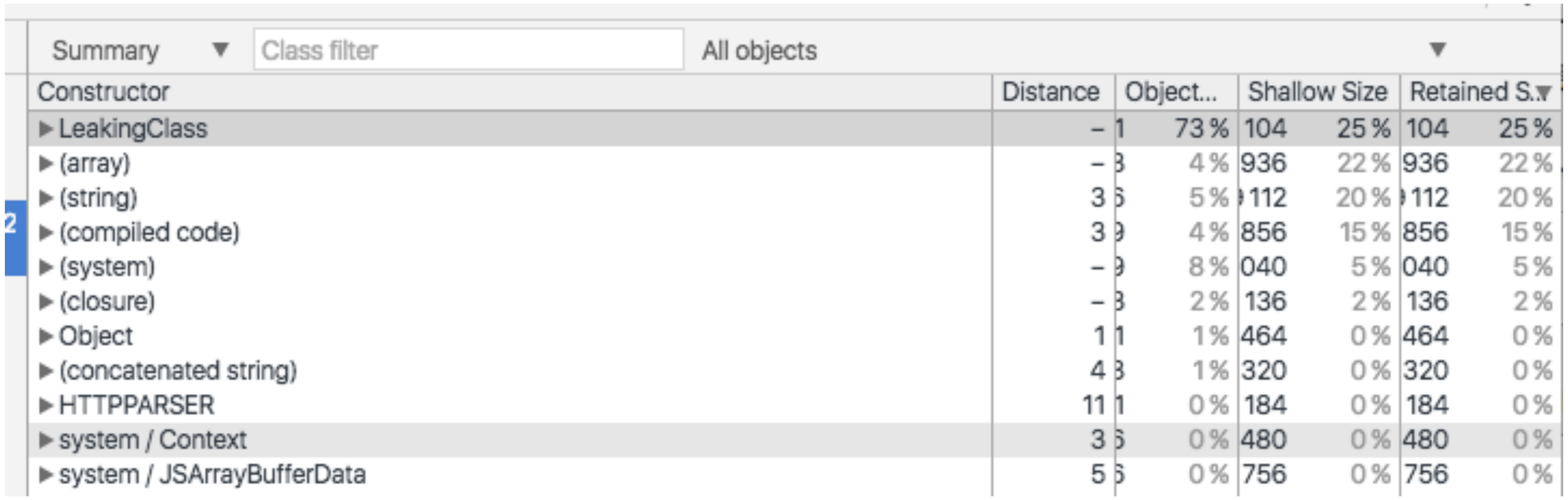
Solution

- 分析资料：*.heapsnapshot
- 生成工具：
  - v8-profiler/node-inspector
  - alinode
- 分析工具：
  - chrome dev tools
  - heapdump analyze service by @alinode

分析表现


依赖路径
[✓ 引力图] 树状列表
☑ 显示箭头
315
813
317
71835
71849
48639
49797
56983
58123
58125
54477
54479
46527
46521
73599
73601
216651
51467
51413

分析表现



Summary ▼ Class filter All objects ▼

Constructor	Distance	Object...		Shallow Size		Retained S.	
▶ LeakingClass	–	1	73 %	104	25 %	104	25 %
▶ (array)	–	3	4 %	936	22 %	936	22 %
▶ (string)	3	6	5 %	112	20 %	112	20 %
▶ (compiled code)	3	9	4 %	856	15 %	856	15 %
▶ (system)	–	9	8 %	040	5 %	040	5 %
▶ (closure)	–	3	2 %	136	2 %	136	2 %
▶ Object	1	1	1 %	464	0 %	464	0 %
▶ (concatenated string)	4	3	1 %	320	0 %	320	0 %
▶ HTTPPARSER	11	1	0 %	184	0 %	184	0 %
▶ system / Context	3	6	0 %	480	0 %	480	0 %
▶ system / JSArrayBufferData	5	6	0 %	756	0 %	756	0 %

chrome dev tools

- 无法处理非常大的heapsnapshot文件
  - 想象一下，用浏览器下载1GB的heapsnapshot文件，然后进行分析
- 无法精准给出内存泄漏的根源
  - 依赖开发者给出准确的class name

case 3: GC frequently

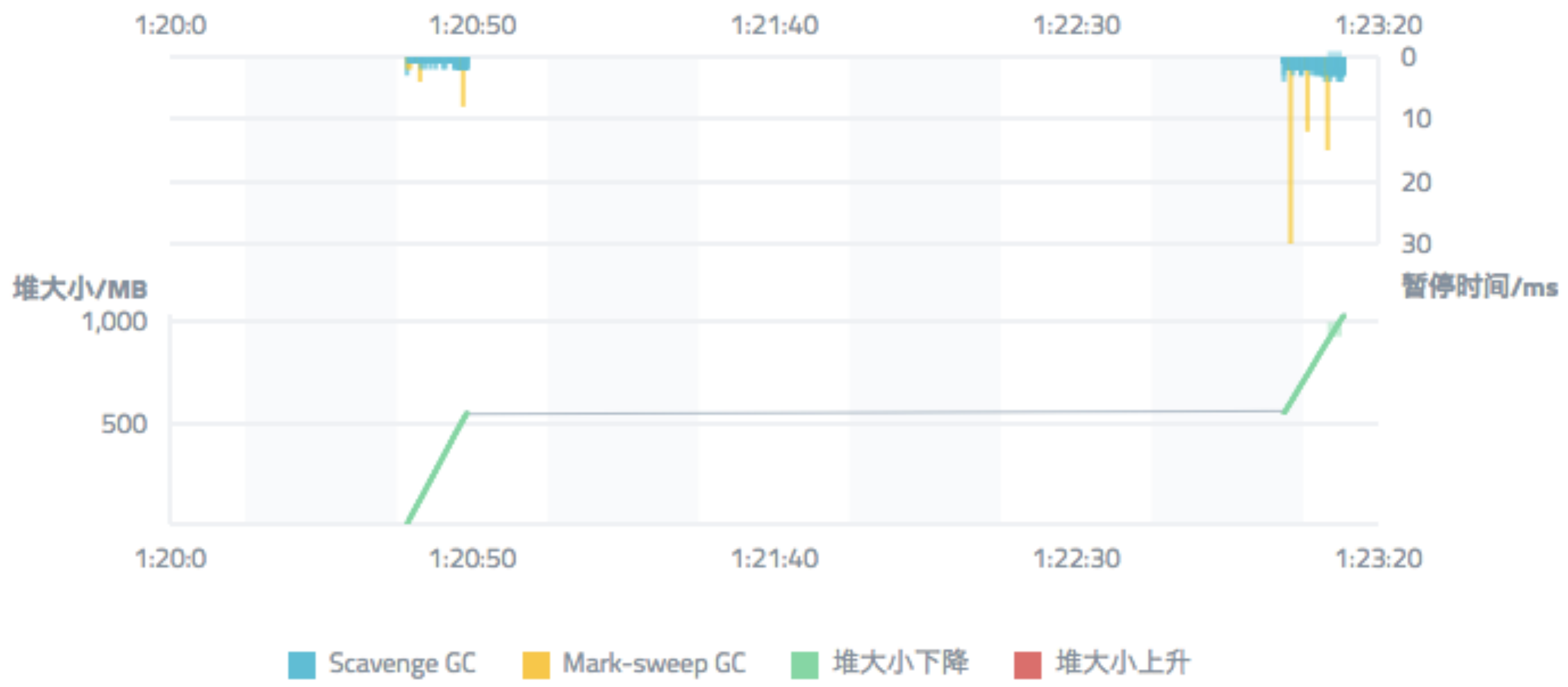
```
var http = require('http');

var GIANT;

function leak() {
  var HUGE = GIANT;

  function unusedClosure() {
    HUGE.slice(1);
  }

  GIANT = {
    willBeLeaked: new Array(1e5).join('.'),
    notAClosure: function notAClosure() {
      return 1;
    }
  }
}
```

```
http.createServer(function(req, res) {
  leak();

  res.writeHead(200);
  res.end('Hello World!\n');

}).listen(3000);
```

Solution

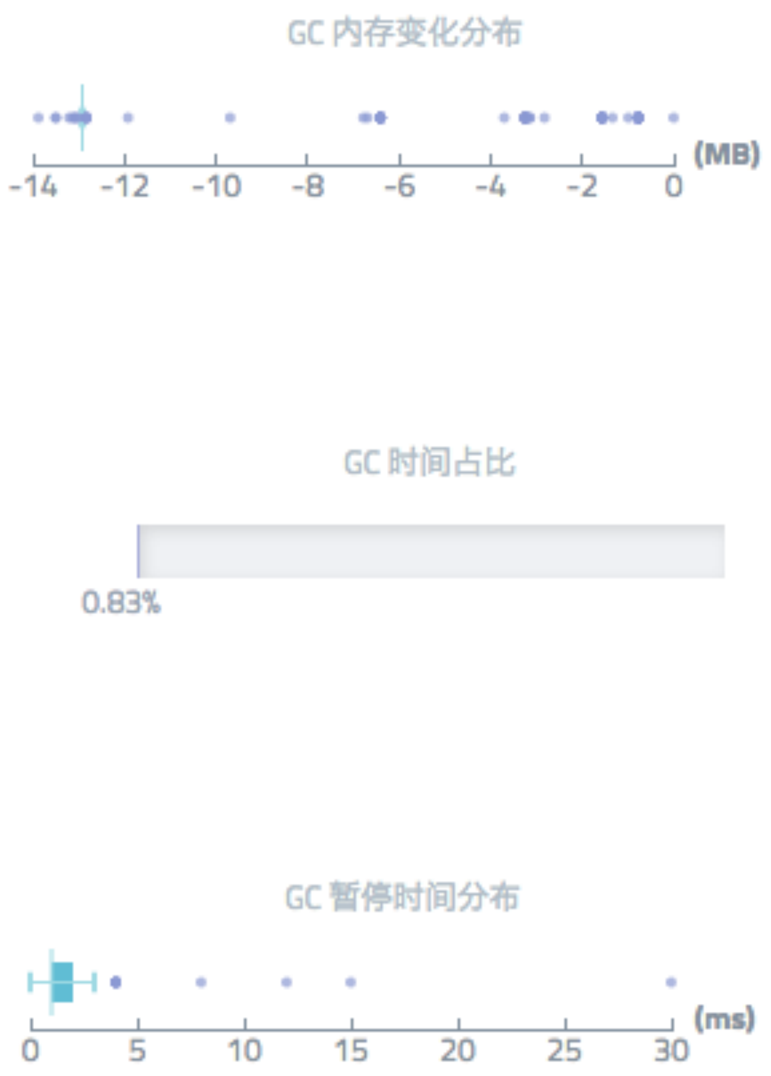
- 分析资料：GC trace log
- 生成工具：
  - node **--trace_gc --trace_gc_verbose** app.js
  - alinode
- 分析工具：
  - GC trace log analyze service by @alinode

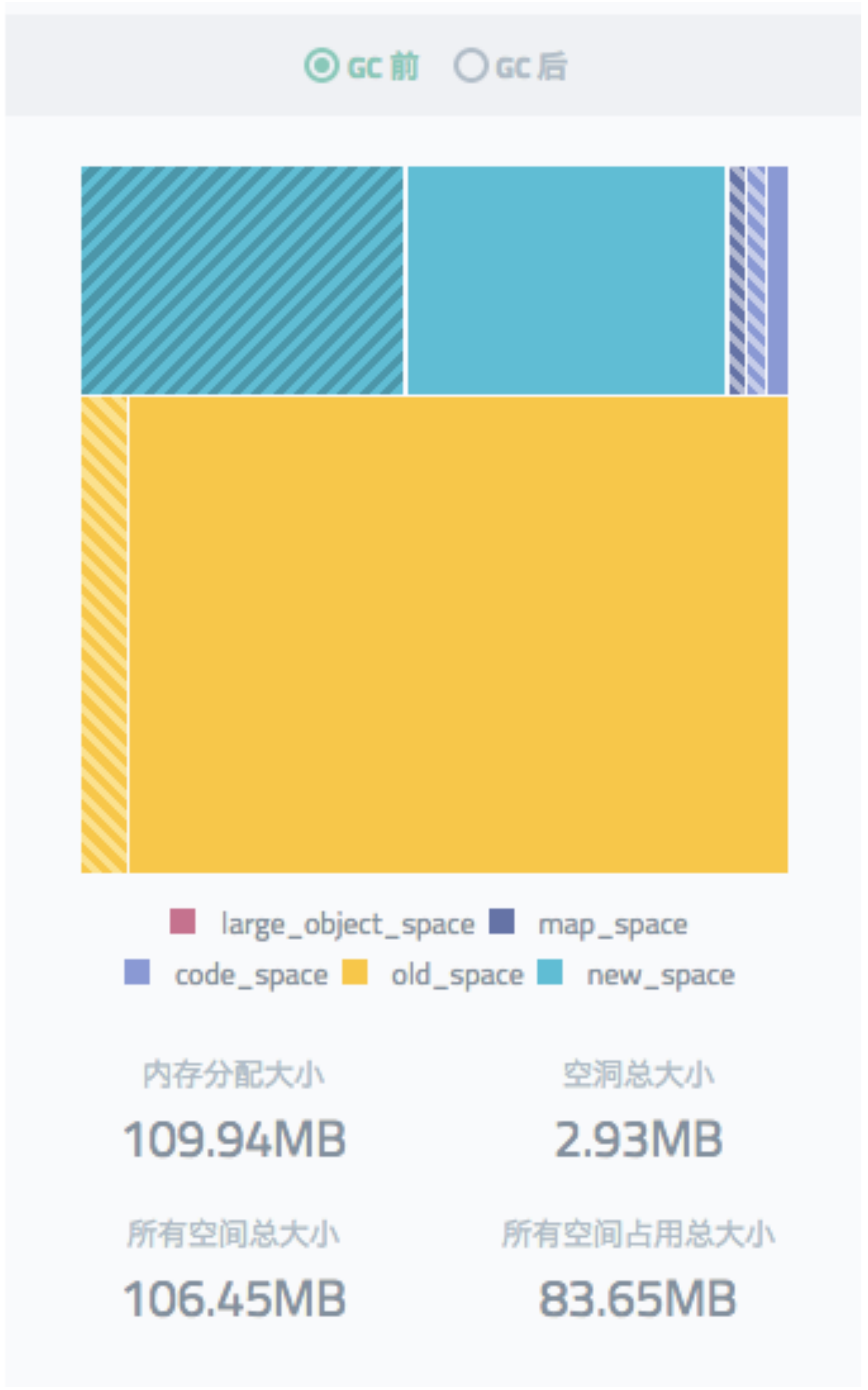
分析表现



1:20:0
1:20:50
1:21:40
1:22:30
1:23:20
0
10
20
30
暂停时间/ms
堆大小/MB
1,000
500
1:20:0
1:20:50
1:21:40
1:22:30
1:23:20
Scavenge GC
Mark-sweep GC
堆大小下降
堆大小上升

分析表现





内存分配大小 109.94MB

空洞总大小 2.93MB

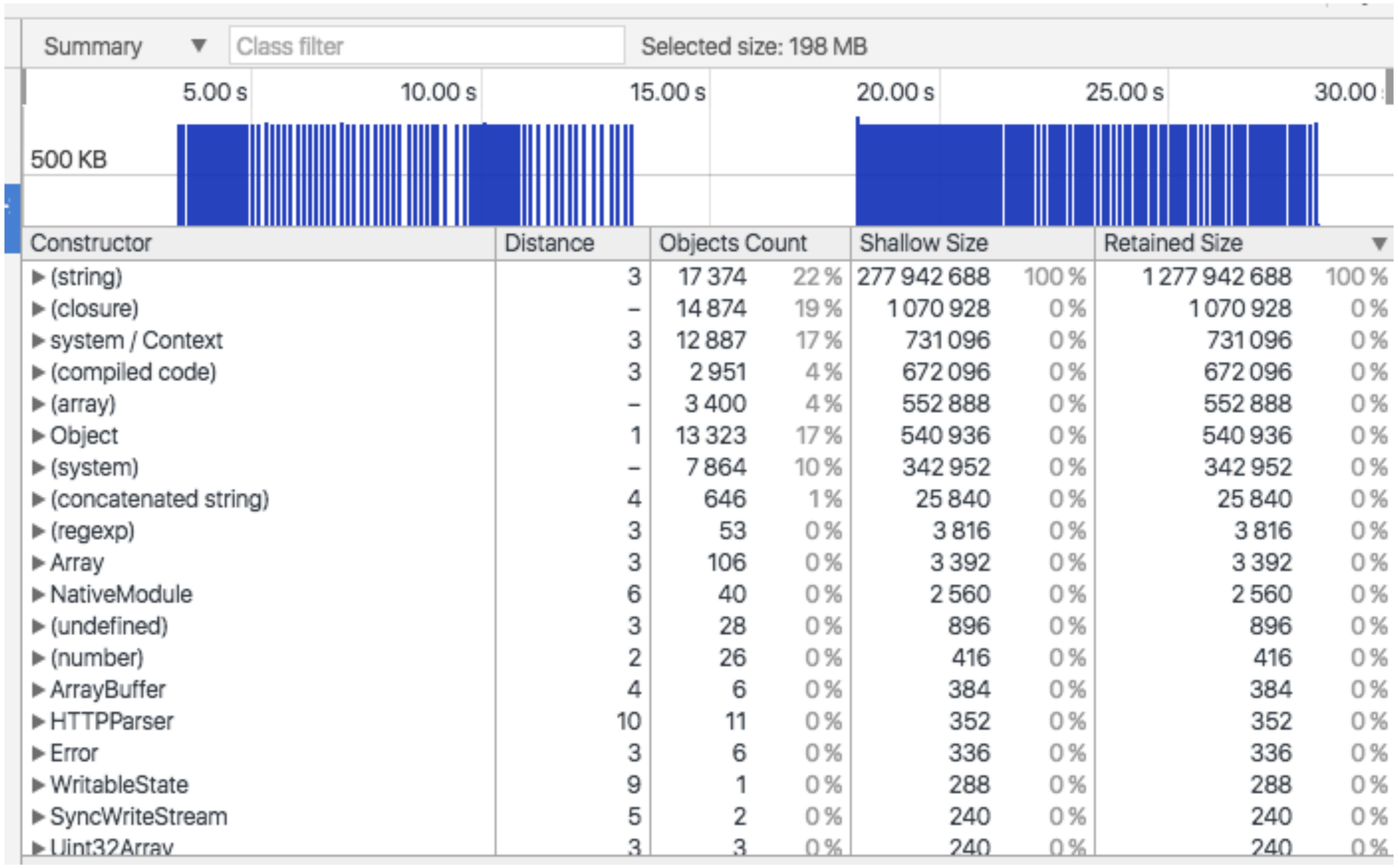
所有空间总大小 106.45MB

所有空间占用总大小 83.65MB

Solution

- 分析资料：*.heaptimeline
- 生成工具：
    - alinode
- 分析工具：
    - chrome dev tools

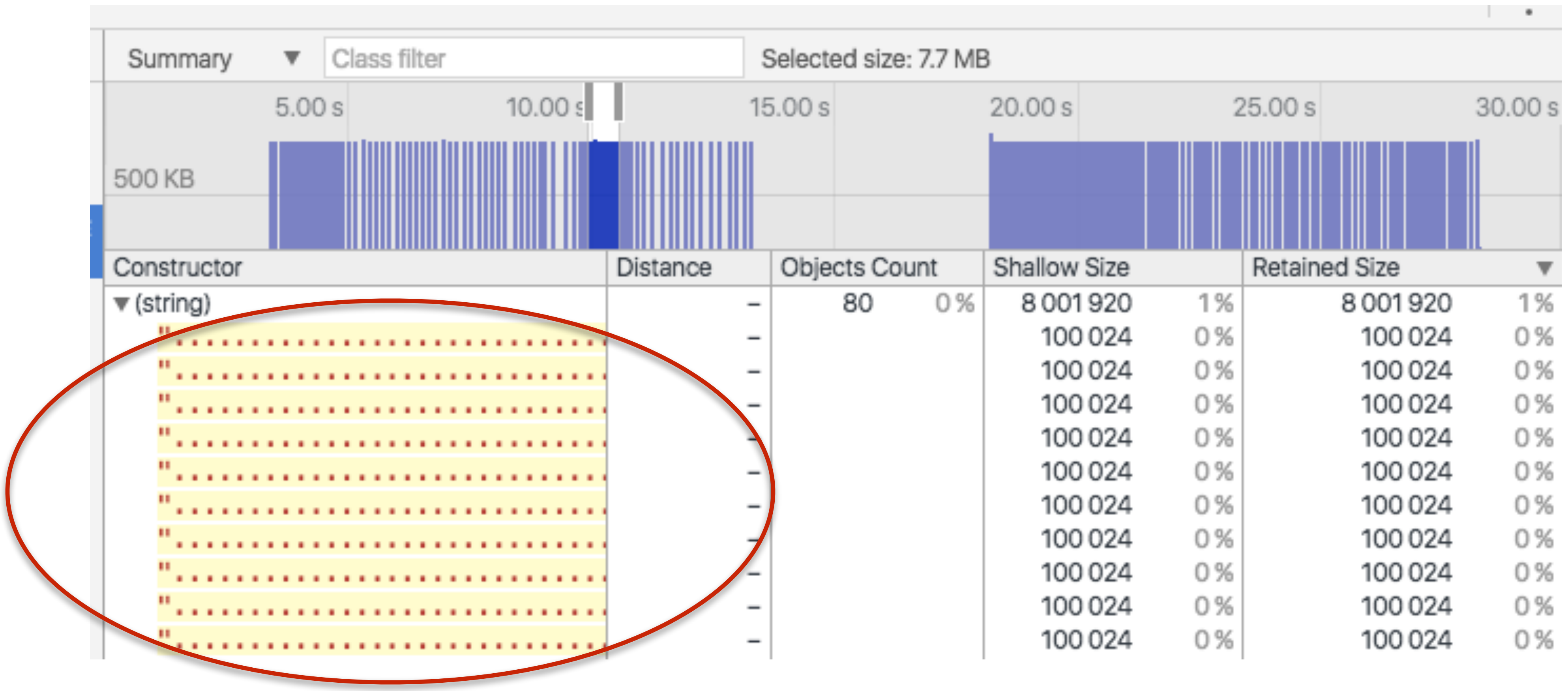
分析表现



Selected size: 198 MB

Constructor	Distance	Objects Count		Shallow Size		Retained Size	
▶ (string)	3	17 374	22 %	277 942 688	100 %	1 277 942 688	100 %
▶ (closure)	–	14 874	19 %	1 070 928	0 %	1 070 928	0 %
▶ system / Context	3	12 887	17 %	731 096	0 %	731 096	0 %
▶ (compiled code)	3	2 951	4 %	672 096	0 %	672 096	0 %
▶ (array)	–	3 400	4 %	552 888	0 %	552 888	0 %
▶ Object	1	13 323	17 %	540 936	0 %	540 936	0 %
▶ (system)	–	7 864	10 %	342 952	0 %	342 952	0 %
▶ (concatenated string)	4	646	1 %	25 840	0 %	25 840	0 %
▶ (regexp)	3	53	0 %	3 816	0 %	3 816	0 %
▶ Array	3	106	0 %	3 392	0 %	3 392	0 %
▶ NativeModule	6	40	0 %	2 560	0 %	2 560	0 %
▶ (undefined)	3	28	0 %	896	0 %	896	0 %
▶ (number)	2	26	0 %	416	0 %	416	0 %
▶ ArrayBuffer	4	6	0 %	384	0 %	384	0 %
▶ HTTPParser	10	11	0 %	352	0 %	352	0 %
▶ Error	3	6	0 %	336	0 %	336	0 %
▶ WritableState	9	1	0 %	288	0 %	288	0 %
▶ SyncWriteStream	5	2	0 %	240	0 %	240	0 %
▶ Uint32Array	3	3	0 %	240	0 %	240	0 %

分析表现



Selected size: 7.7 MB

Constructor	Distance	Objects Count		Shallow Size		Retained Size	
▼ (string)	–	80	0 %	8 001 920	1 %	8 001 920	1 %
"..	–			100 024	0 %	100 024	0 %
"..	–			100 024	0 %	100 024	0 %
"..	–			100 024	0 %	100 024	0 %
"..	–			100 024	0 %	100 024	0 %
"..	–			100 024	0 %	100 024	0 %
"..	–			100 024	0 %	100 024	0 %
"..	–			100 024	0 %	100 024	0 %
"..	–			100 024	0 %	100 024	0 %
"..	–			100 024	0 %	100 024	0 %
"..	–			100 024	0 %	100 024	0 %

case 3: GC frequently

```
var http = require('http');

var GIANT;

function leak() {
  var HUGE = GIANT;

  function unusedClosure() {
    HUGE.slice(1);
  }

  GIANT = {
    willBeLeaked: new Array(1e5).join('.'),
    notAClosure: function notAClosure() {
      return 1;
    }
  }

  HUGE = null;  /* not used anymore! */
}
```

```
http.createServer(function(req, res) {
  leak();

  res.writeHead(200);
  res.end('Hello World!\n');

}).listen(3000);
```

“就是三件事情，很惭愧，就做了一点微小的工作，谢谢大家。”

alinode

- https://alinode.aliyun.com/
- “一键”解决线上Node.js问题