性能优化指北:通用流程和方法

陈熙旻@Rokid

1. 简介

性能优化二三事

2. 案例

YODAOS 核心语音链路性能优化

3. 流程

性能问题处理模板

4. 方法

性能问题的解决方式

1. 简介

- 首屏速度/渲染效率/...
- WHAT QPS/响应时间/...
 - •

WHY

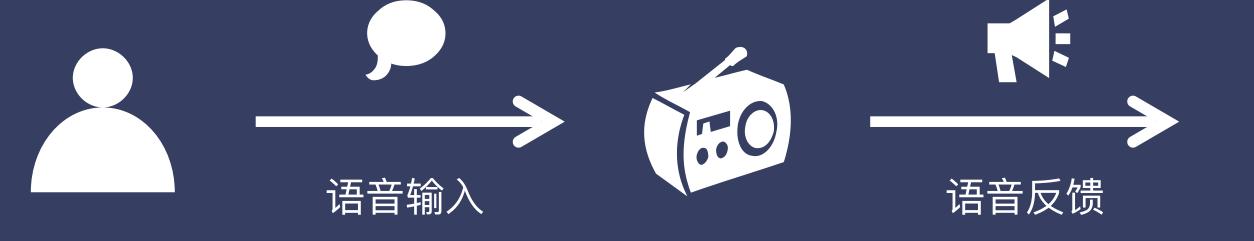
- 提升体验
- 成本控制

WHEN · 持久战

终端:智能音箱

CPU: 单核 900M

业务: 唤醒



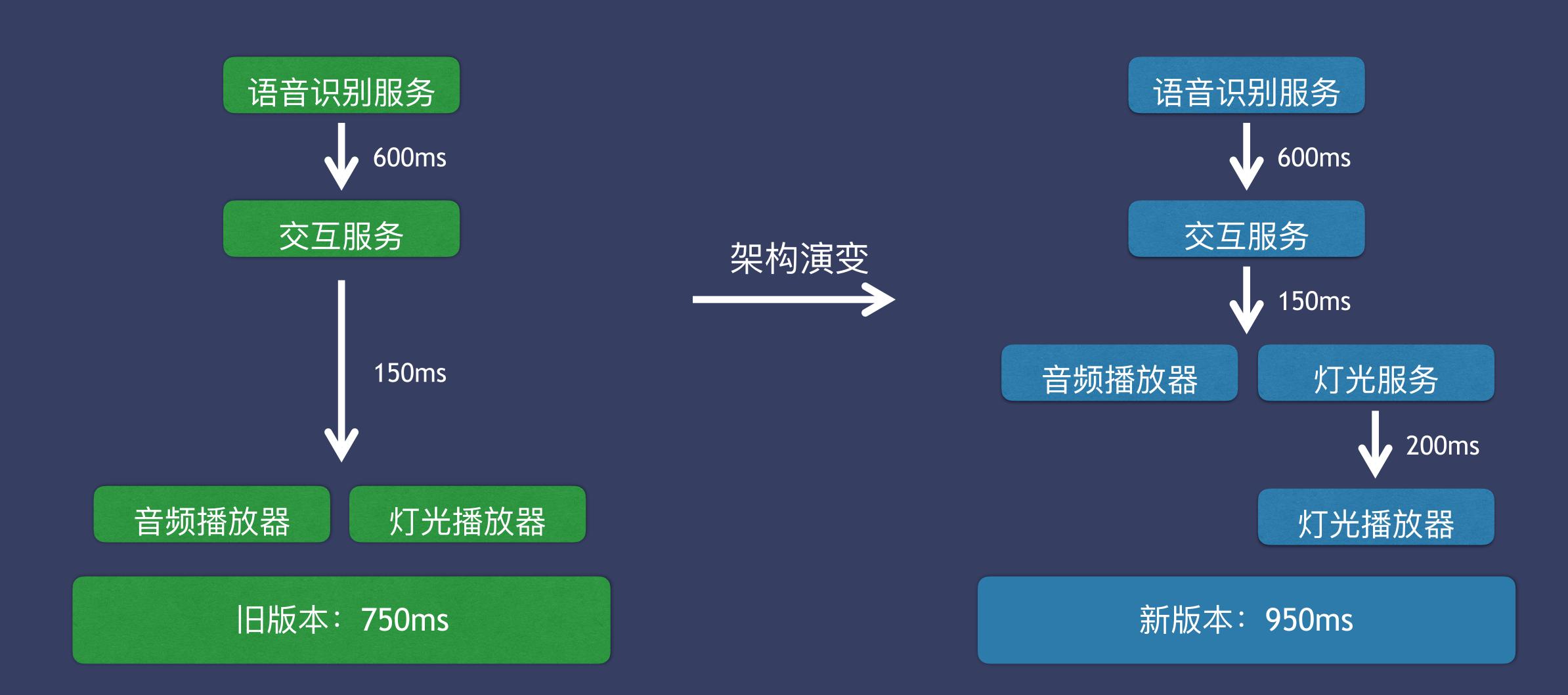
『有什么可以帮你的』

问题

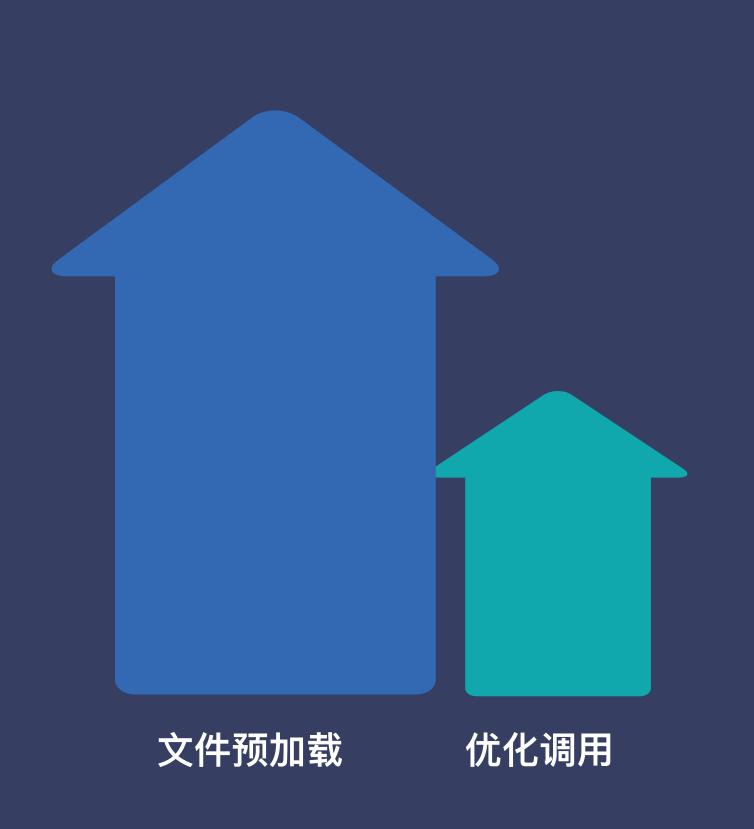
目标

唤醒速度变慢

恢复到原有水平



2.4.1 案例 - 排查





系统 CPU

交互进程 CPU

GC?

Percent of CPU Usage

```
void print_cpu_usage () {
   sleep(1000)
   total_cpu = usr + sys + idle + iowait + ...
   per = (total_cpu - idle - iowait) / total_cpu
   pcpu = pusr + psys + pidle + piowait + ...
   per = ( pcpu / sum(pcpu) ) * per
   print("total: %f\%, process: %f\%", per, pper)
}
```

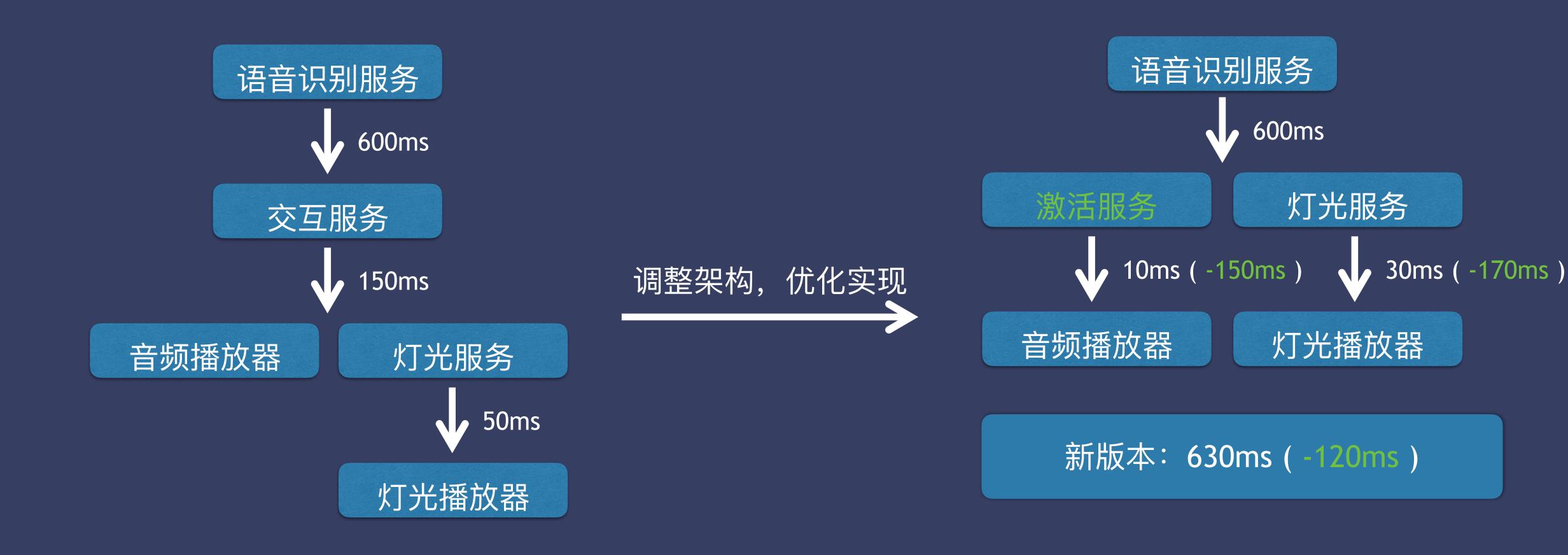
25% Timeline



Test \$ nice -n -20 node ./app.js



2.5 案例 - 解决方案



分解问题

场景

链路

问题发生在什么场景?

这个场景的链路是什么样的?

3.2 流程 - 明确目标

定义指标

需要关注哪些指标?这些指标需要谁来定义?

预期结果

优化后最终希望达到什么样的目标?

安排周期

是否需要分阶段进行?每个阶段需要完成什么目标?



3.5 流程 - 完成

解决

皆大欢喜

规避

查明原因但无法解决,通过某些方式避免

兜底

查明或未查明原因时都无法解决,出现问题时自动订正

搁置

可承受范围内,暂时先不管

评估指标

规划目标

批量

缓存

惰性&预加载

延迟

并发

```
1 console.time('cache-test')
2 function handleArray() { }
3 var arr = new Array(100000)
4 for (var i = 0; i < arr.length; ++i) {
    handleArray(arr[i], function handler() {})
7 console.timeEnd('cache-test')
1 console.time('cache-test')
2 function handleArray() {}
3 \text{ var arr} = \text{new Array}(100000)
4 var length = arr.length
5 var handler = function() {}
6 for (var i = 0; i < length; ++i) {
    handleArray(arr[i], handler)
9 console.timeEnd('cache-test')
```

750ms

80ms

4.2.1 方法 - 思路 - 延迟

```
1 function sendStatistic() {
2   /*
3     * send statistic data
4     */
5 }
6
7 function onRequest() {
8     /*
9     * event logic
10     */
11 sendStatistic()
12 }
```

```
1 function sendStatistic() {
2   /*
3     * send data
4    */
5 }
6
7 function onRequest() {
8   /*
9     * event logic
10     */
11 delay(sendStatistic, 1000)
12 }
```

4.2.1 方法 - 思路 - 批量

```
1 var sendQueue = []
 3 function sendStatistic() {
     sendQueue.push(...)
     if (sendQueue.length >= MAX QUEUE SIZE) {
        * send data
       sendQueue = []
11 }
  function onRequest() {
14
     * event logic
     setImmediate (sendStatistic)
18 }
```

Preload

```
1 var data = []
2 /**
3 * init all data
4 */
5
6 function onRequest(i) {
7 return data[i]
8 }
```

Lazy Load

```
1 var data = []
3 function onEvent(i) {
 if (!data[i]) {
      * init data[i]
    return data[i]
```

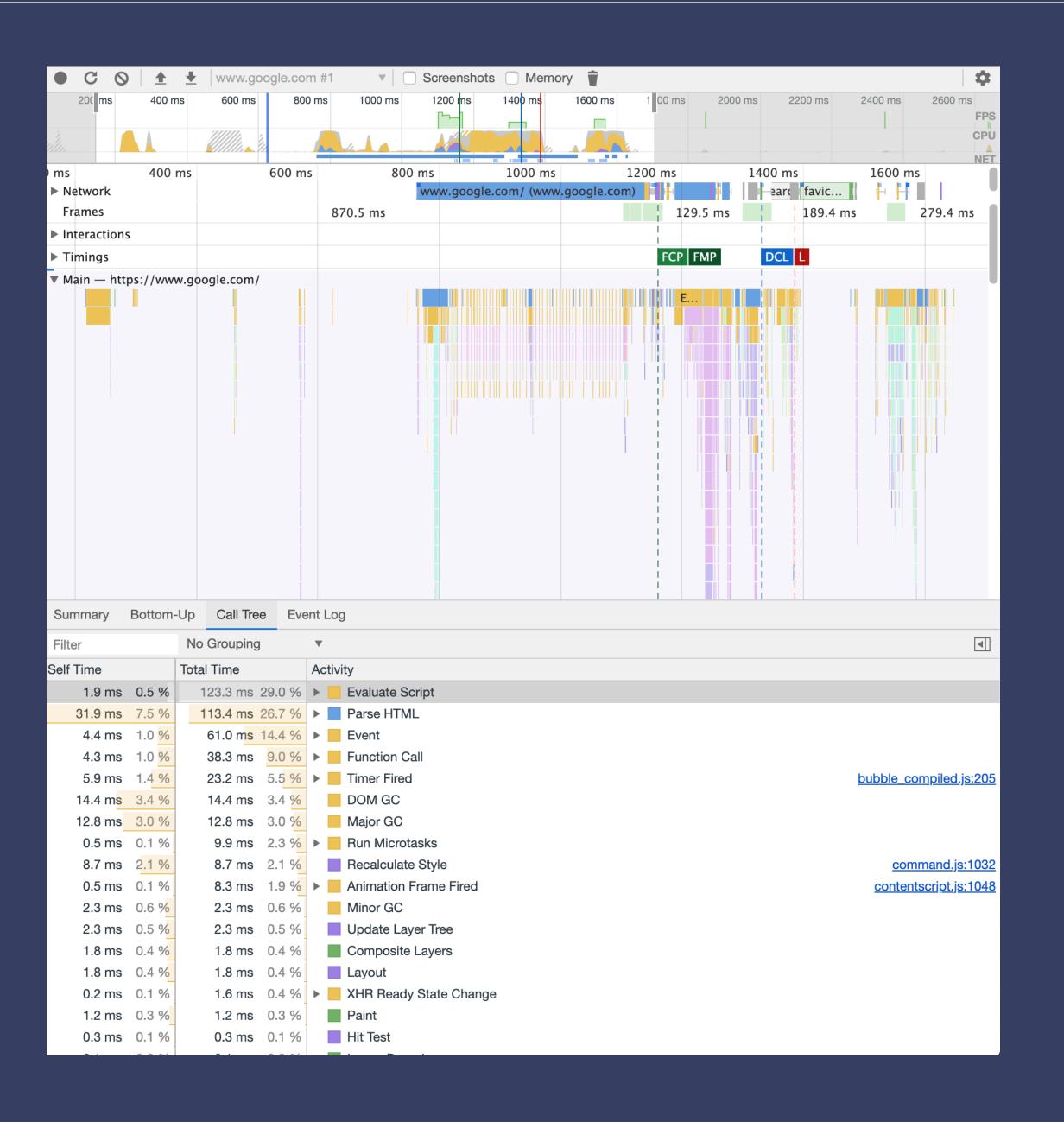
4.2.1 方法 - 思路 - 并发

```
/************ master **********/
2 var worker = new Worker('./worker file', {
   workerData: data
 worker.on('message', function (message) {
   console.log('worker message: ', message)
 /************ worker **********/
2 var result = handleWork(workerData)
3 masterPort.postMessage (result)
```

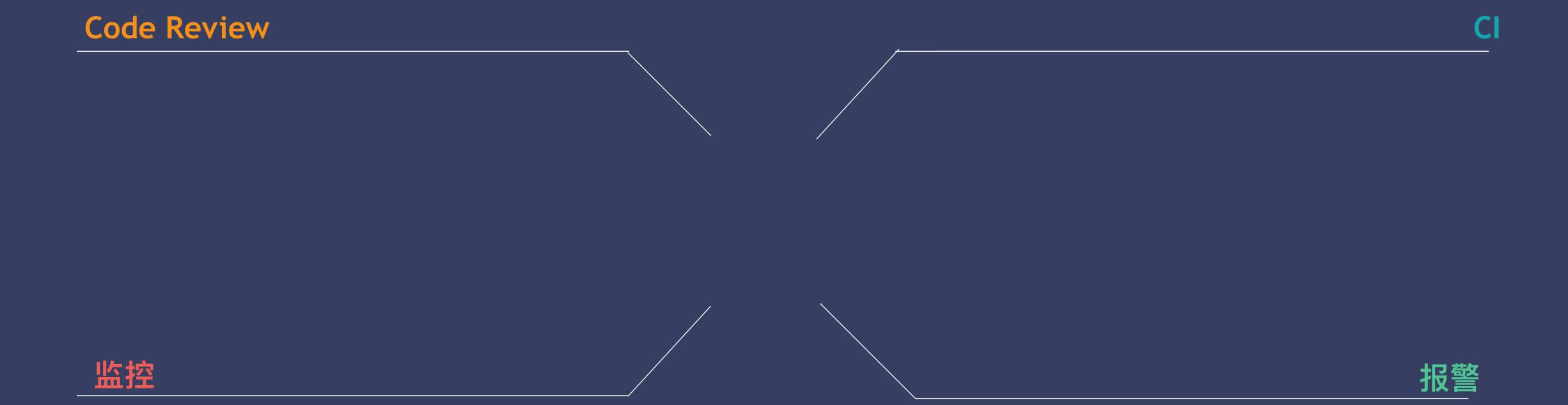
4.3.1 方法 - 工具 - Heap Snapshot

• 0	Summary ▼ Class filter All objects			▼	
Profiles	Constructor	Distance	Shall	low Size	Retained Size
	▶ (system) ×152902		2	12 647 552 52 %	15 164 896 62 %
HEAP SNAPSHOTS	▶ Window / ×17		1	952 0 %	13 923 600 57 %
Snapshot 1 Save	▶ Object ×22952		2	1 282 728 5 %	9 208 928 38 %
	▶ (closure) ×38143		2	2 271 008 9 %	6 280 192 26 %
Snapshot 2 23.4 MB	▶ (array) ×12071		2	3 692 720 15 %	5 609 424 23 %
	► Array ×9032		2	289 024 1 %	4 949 512 20 %
	►Window ×122		1	5 192 0 %	3 174 536 13 %
	▶ (string) ×40277		2	3 112 984 13 %	3 113 064 13 %
	▶ (compiled code) ×7806		3	580 040 2 %	2 514 240 10 %
	▶ system / Context ×2569		3	214 912 1 %	740 744 3 %
	▶ Window / chrome-extension://ihlenndgcmojhcghmfjfneahoeklbjjh		1	56 0 %	589 104 2 %
	▶Trie ×42		2	4 368 0 %	516 336 2 %
	►TrieNode ×3004		3	144 368 1 %	414 072 2 %
	▶ Document ×22		3	704 0 %	205 432 1 %
	► HTMLElement ×110		3	5 632 0 %	201 904 1 %
	Retainers				
	Object		ance	Shallow Size	Retained Size
	▶properties in Window @4673 □		1	56 0	% 588 424 2 %

4.3.2 方法 - 工具 - Profiler



4.5 方法 - 持久战



QA