淘宝网

@flyinweb
占超群[离哲]

NodeJs 应用
性能分析优化 & 分布式设计

提纲

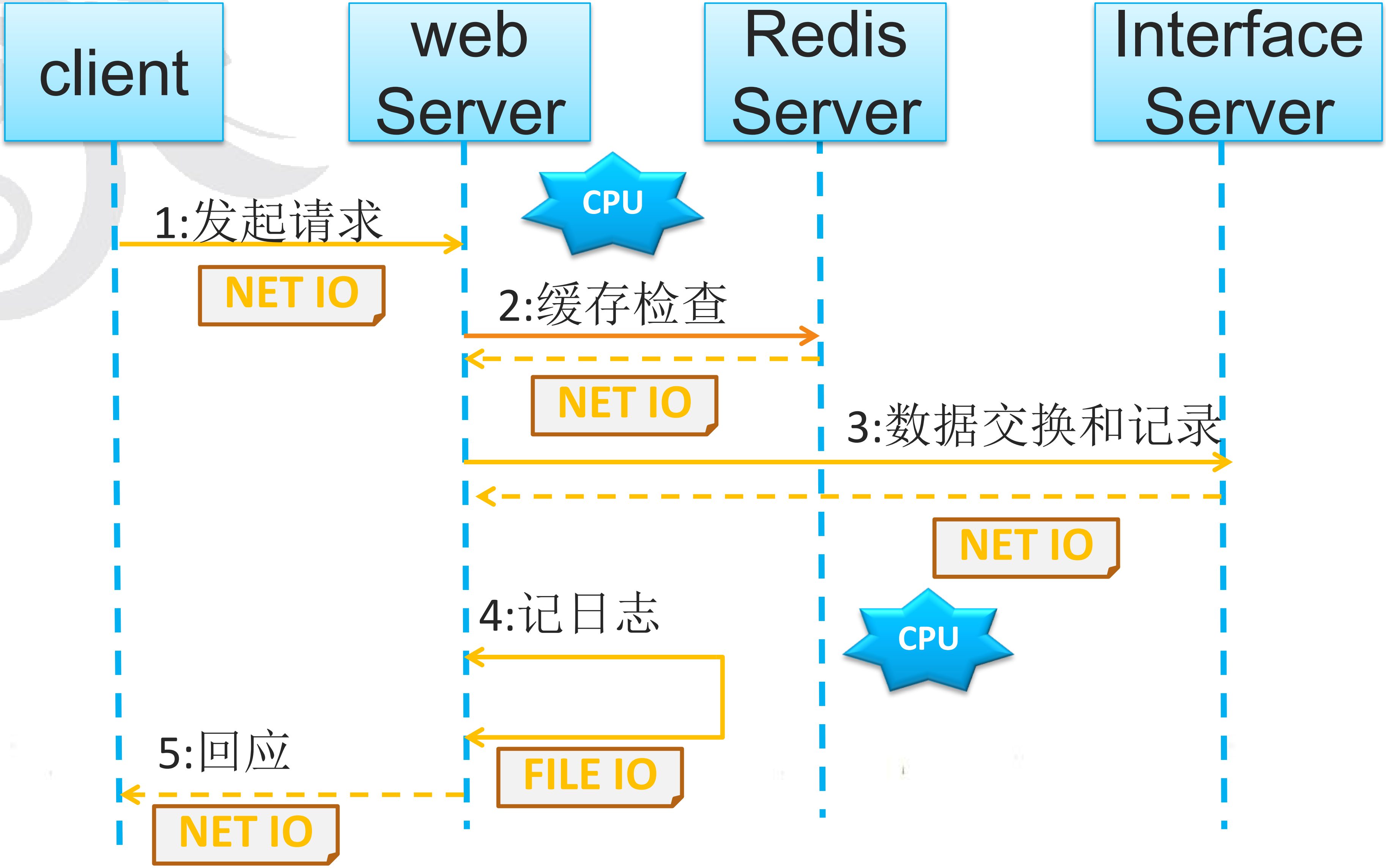
- 从实例开始
- 性能分析&优化方法
  - 资源占用分析
    - CPU、内存
    - 文件IO、网络IO
  - 慢代码分析
  - V8 掠影
    - 内存、堆栈、GC、预编译
- 分布式设计
  - 单机
  - 集群

1

性能分析和优化

性能分析流程、工具、方法、优化要点

从实例开始


client
web Server
Redis Server
Interface Server
CPU
1:发起请求
NET IO
2:缓存检查
NET IO
3:数据交换和记录
NET IO
CPU
4:记日志
5:回应
FILE IO
NET IO

工具

- Linux tools
  - pidstat/iostat/vmstat
  - sar/top/lsof
- node lib
  - v8-profiler
  - Benchmark.js
- V8 tools
  - node-v0.6.2/deps/v8/tools
    - linux-tick-processor
    - ll_prof.py
    - run-valgrind.py

CPU 占用资源分析

利用率：

用户进程/内核/中断/IO等待/空闲

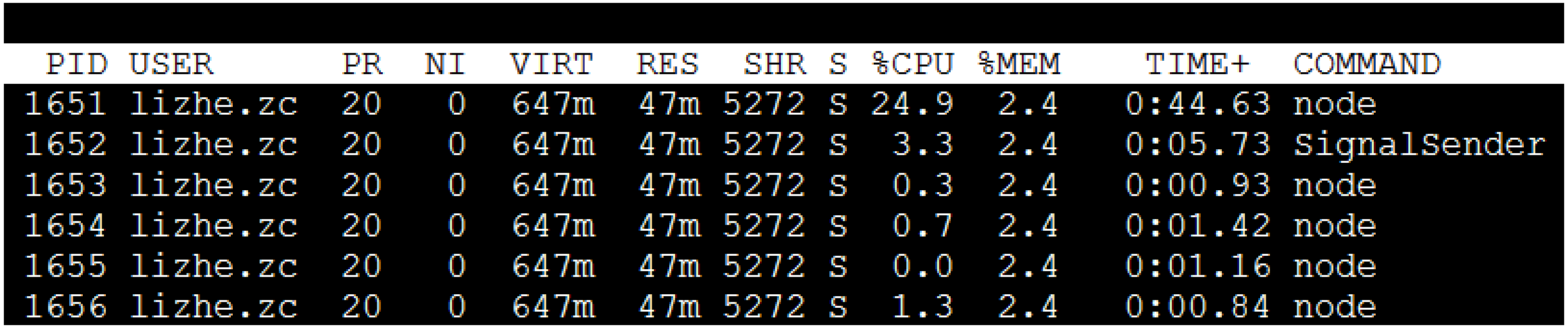
us / sy/(hi/si)/wa /id →top

建议值：

usr/sys:65%-75% / 30%-35%

分析:

top (1->shift+h)



```
  PID USER       PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1651 lizhe.zc   20   0  647m  47m 5272 S 24.9  2.4   0:44.63 node
 1652 lizhe.zc   20   0  647m  47m 5272 S  3.3  2.4   0:05.73 SignalSender
 1653 lizhe.zc   20   0  647m  47m 5272 S  0.3  2.4   0:00.93 node
 1654 lizhe.zc   20   0  647m  47m 5272 S  0.7  2.4   0:01.42 node
 1655 lizhe.zc   20   0  647m  47m 5272 S  0.0  2.4   0:01.16 node
 1656 lizhe.zc   20   0  647m  47m 5272 S  1.3  2.4   0:00.84 node
```

CPU 占用资源分析

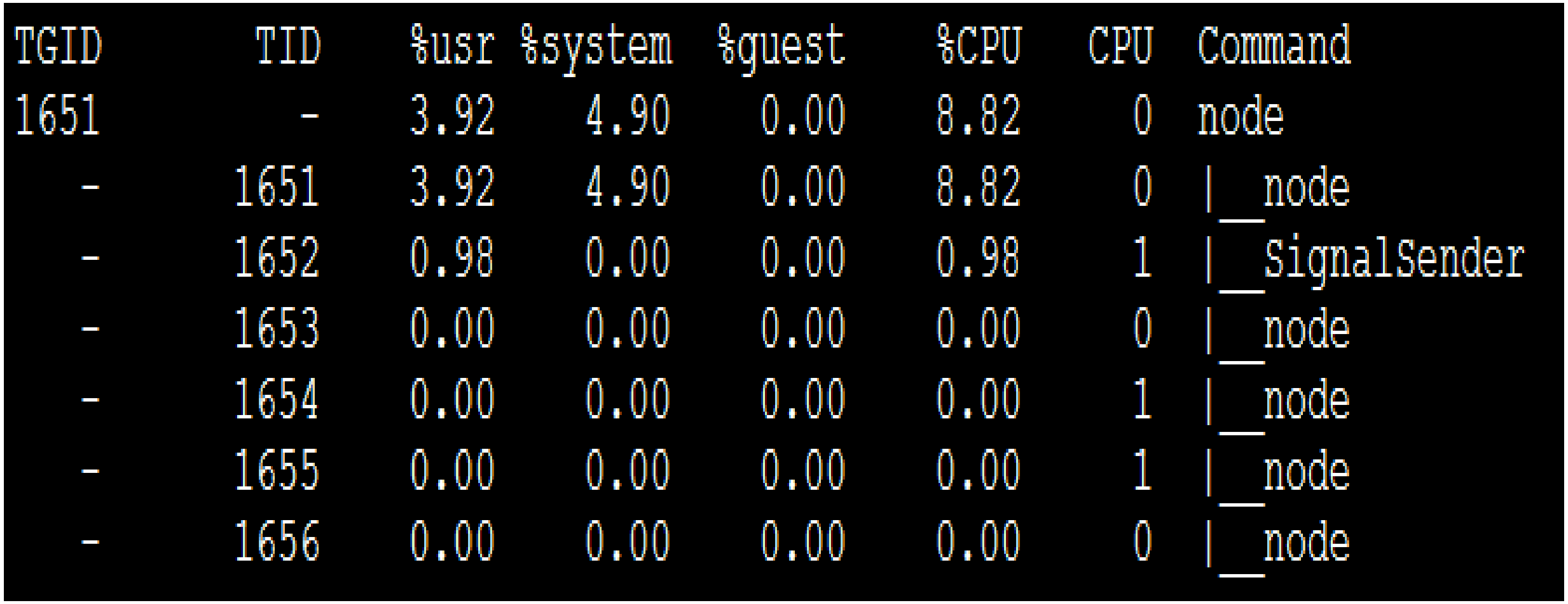
- pidstat -p 1651 -t 1 100

  node 应用

  |-node 主线程

  |-SignalSender线程 profile sampling

  |-4个libuv线程 iowatcher etc



```
TGID       TID    %usr %system  %guest    %CPU   CPU  Command
1651         -    3.92    4.90    0.00    8.82     0  node
   -      1651    3.92    4.90    0.00    8.82     0  |__node
   -      1652    0.98    0.00    0.00    0.98     1  |__SignalSender
   -      1653    0.00    0.00    0.00    0.00     0  |__node
   -      1654    0.00    0.00    0.00    0.00     1  |__node
   -      1655    0.00    0.00    0.00    0.00     1  |__node
   -      1656    0.00    0.00    0.00    0.00     0  |__node
```

CPU 优化/利用点

- 代码适应 V8
- 减少GC
- 多进程
- 原生代码
- 模板选型
- 复杂计算业务逻辑转移
  - Java/C app
  - Gearman类 任务分发（异步化）
  - MQ
- 语言层面
  - eval
  - setInterval/setTimeout (arg)
  - Primitive operations
  - Regexp
  - async+parallel
  - Object constructor
  - Array Pre-allocate
  - 精简解析（httpparser）

文件IO 占用资源分析

- pidstat -d -p 1651 -t 1 100
  - kB_rd/s   kB_wr/s kB_ccwr/s
- iostat -x **vda2** 3 5
  - %util
  - await
- sar –b
  - rtps / wtps / (bread/s) / (bwrtn/s)

- 优化点
  - IO分散
  - stream读取大文件
  - async
  - unwatcher

网络IO占用分析

- sar –n DEV 1 10
  - IFACE   rxpck/s   txpck/s   rxkB/s
- sar –n SOCK 1 10
  - tcpsck   udpsck
- sar –n TCP 1 10
  - iseg/s   oseg/s

- 优化点：
  - maxsockets
  - timeout
  - response header（expire .. ）
  - request （no cookies.. ）
  - pool
  - sync/async （ getaddrinfo/ gethostbyname / ares_gethostbyname ）
  - 分段读取
  - 压缩传输（msgpack/bin/gzip）

内存占用

- free/vmstat
  - Cached/buffered/swpd
- sar –B 1 5
  - (pgpgin/s) / (pgpgout/s) / (pgscank/s)
  - (pgscand/s) /(pgsteal/s) /(%vmeff)
- sar -r 1 5
  - Kbmemfree+kbbuffers+kbcached
- pidstat –r –p 1813 1 10
  - minflt/s majflt/s   VSZ   RSS
- pidstat –s –p 1813 1 10
  - minflt/s majflt/s   VSZ   RSS

内存占用

- 优化点：
  - 整体
    - 加入 Buffer(堆外内存)
    - 加大最大内存设置
      - --max_old_space_size =1900 (64bit os)
      - --stack_size=4096
      - --max_new_space_size=10000
      - --use_big_map_space （**慎用**）
  - 语言层面
    - 局部变量
    - Try {bigset}catch（ ）
      - > try {fn}
      - http://jsperf.com/try-catch-performance-overhead
    - TypedArray
    - Cache
    - With
    - 对象转换、copy
    - String concat …

开始说说 代码性能+V8

- Benchmark 测试
  - 单元测试不仅仅只验证**正确性**

```
var suite = new Benchmark.Suite;
// add tests
suite.add(‘RegExp#test’, function() {
        /o/.test(‘Hello World!’);
})
.add(‘String#match’, function() {
        !! ‘Hello World!’.match(/o/);
})
// add listeners
.on(‘cycle’, function(event, bench) {
        ….
})
.on(‘complete’, function() {
        …
})
.run({ ‘async’: true });
```

Sample

data length	\u0000	data bytes

6	\u0000	T	A	O	b	A	O

Step 1

```
Parser.prototype.parse1 = function (s) {
  var l = '';
  for (var i = 0; i < s.length; i++) {
    if (s[i] == '\u0000') {
      l = Number(l);
      this.emit('data', s.substr(i + 1, l));
      return this.parse1(s.substr(i + 1 + l));
    } else {
      l += s[i];
    }
  }
  return s;
};
```

Step 1-Stress

```
var p = new Parser();
var NOF_RUNS = 1000;
var start = Date.now();
for (var j = 0; j < RUN_NUMBERS; j++) {
  p.parse3(fakeInput);
}
var end = Date.now();

var timeSpent = end - start;
console.log(timeSpent + ' ms');
```

400 ms

Step 1—key profile

```
[JavaScript]:
   ticks  total  nonlib   name
    38   15.8%   21.0%  Stub: SubStringStub
     2    0.8%    1.1%  Stub: StringAddStub
     2    0.8%    1.1%  LazyCompile: *Parser.parse1 /work/project/stress/src/BinFile.js:10
     1    0.4%    0.6%  Stub: StringAddStub {1}
     1    0.4%    0.6%  LazyCompile: *substr native string.js:698
[GC]:
   ticks  total  nonlib   name
   151   62.9%
-----------------
pause=9 mutator=7 gc=s external=0 mark=0 sweep=0 sweepns=0 compact=0
      total_size_before=22049776 total_size_after=22001000 holes_size_before=335256
      holes_size_after=335256 allocated=16776520 promoted=7174952
Memory allocator,   used: 84180992, available: 1450934272
New space,          used:  9551576, available:  7225640
Old pointers,       used:   605384, available:  1710936, waste:      160
Old data space,     used:   203208, available:   300712, waste:       16
Code space,         used:   361472, available:   126208, waste:        0
Map space,          used:    39704, available:   207624, waste:     4640
Cell space,         used:     8128, available:   251968, waste:        0
Large object space, used: 13025280, available: 1450926016
```

GC成本随长时间存活对象的个数线性上涨

Step 2

```
Parser.prototype.parse1 = function (s) {
  var l = '';
  for (var i = 0; i < s.length; i++) {
    if (s[i] == '\u0000') {
      l = Number(l);
      this.emit('data', s.substr(i + 1, l));
      return this.parse1(s.substr(i + 1 + l));
    } else {
      l += s[i];
    }
  }
  return s;
};
```

Step 2

```
Parser.prototype.parse1 = function (s) {
  var l = '';
  for (var i = 0; i < s.length; i++) {
     if (s[i] == '\u0000') {
      l = Number(l);
      this.emit('data', s.substr(i + 1, l));
      s = s.substr(i + 1 + l);
      i = 0;
      l = '';
     } else {
      l += s[i];
    }
  }
  return s;
};
```

170 ms

Step2 -profile

```
[JavaScript]:
   ticks  total  nonlib   name
     42   18.8%   44.2%  Stub: SubStringStub
      3    1.3%    3.2%  Stub: StringAddStub
      2    0.9%    2.1%  LazyCompile: *Parser.parse2 /mnt/share/stress/src/BinFile.js:25
      1    0.4%    1.1%  LazyCompile: b native v8natives.js:1264
[GC]:
   ticks  total  nonlib   name
     36   16.1%
---------
pause=0 mutator=1 gc=s external=0 mark=0 sweep=0 sweepns=0 compact=0
      total_size_before=7550080 total_size_after=3394272 holes_size_before=69824
      holes_size_after=69824 allocated=4148080 promoted=0
Memory allocator,   used: 71888896, available: 1463226368
New space,          used:    22560, available:  4171744
Old pointers,       used:  2060512, available:   245784, waste:     2056
Old data space,     used:   252568, available:   259256, waste:      240
Code space,         used:   415616, available:    88320, waste:        0
Map space,          used:    39704, available:   215752, waste:     4640
Cell space,         used:     8128, available:   251968, waste:        0
Large object space, used:   724992, available: 1463218112
```

Step 3

```
Parser.prototype.parse3 = function (s) {
  var l = '';
  //方法3
  var j = 0;

  for (var i = 0; i < s.length; i++) {
    if (s[i] == '\u0000') {
      l = Number(l);
      this.emit('data', s.substr(i + 1, l));
      i += l;
      j = i + 1;
    } else {
      l += s[i];
    }
  }
  return s.substr(j);
};
```

11 ms

Step3-profile

```
[JavaScript]:
   ticks  total  nonlib   name
      1    0.7%    3.4%  Stub: CallFunctionStub
[GC]:
   ticks  total  nonlib   name
     20   13.8%
-------------------------
pause=1 mutator=2 gc=s external=0 mark=0 sweep=0 sweepns=0
    compact=0 total_size_before=2880944 total_size_after=2766424
    holes_size_before=18528 holes_size_after=30208 allocated=790456
    promoted=671920
Memory allocator,   used: 70520832, available: 1464594432
New space,          used:   262136, available:   786440
Old pointers,       used:  1232664, available:    34592, waste:     712
Old data space,     used:   231136, available:    28936, waste:      24
Code space,         used:   434112, available:    53568, waste:       0
Map space,          used:    54208, available:   193120, waste:    4640
Cell space,         used:     8624, available:   243344, waste:       0
Large object space, used:   667648, available: 1464586176
```

Step4

```
Parser.prototype.parse4 = function (s) {
var l = 0, i = 0;
  while (i < s.length) {
    var ch = s.charCodeAt(i);
    if (ch === 0) {
    this.emit('data', s.substr(i + 1, l));
          i += l + 1;
          l = 0;
    } else {
      l = l * 10 + ch;
      i ++;
    }
  }
};
```

50X

8 ms

Step4-profile

```
[JavaScript]:
  ticks  total  nonlib   name

[GC]:
  ticks  total  nonlib   name
    17   12.4%
----------------------------
pause=3 mutator=1 gc=s external=0 mark=0 sweep=0 sweepns=0
    compact=0 total_size_before=2889880 total_size_after=2769744
    holes_size_before=20472 holes_size_after=25392 allocated=786184
    promoted=666304
Memory allocator,   used: 70520832, available: 1464594432
New space,          used:   262136, available:   786440
Old pointers,       used:  1231248, available:    36048, waste:     672
Old data space,     used:   231080, available:    20880, waste:       8
Code space,         used:   438976, available:    64960, waste:       0
Map space,          used:    54152, available:   201304, waste:    4640
Cell space,         used:     8608, available:   243360, waste:       0
Large object space, used:   667648, available: 1464586176
```

预编译和v8代码优化 日志

```
[optimizing: Queue.push / 25d70710ba79 - took 0.064 ms]
Compiled: 33 functions with 37333 byte source size in 31.198000ms.
[marking NonStringToString 0xc69df07d020 for recompilation]
Bailout in HGraphBuilder: @"NonStringToString": call to a JavaScript runtime function
[disabled optimization for: NonStringToString / c69df07d021]
[marking Buffer.write 0x143784371b80 for recompilation]
Bailout in HGraphBuilder: @"Buffer.write": SwitchStatement: non-literal switch label
```

Nodejs prof分析方法

1.Linux perf + node deep prof

```
perf record -R -e cycles -c 10000 -f  node ../script.js --ll-prof
ll_prof.py --disasm-top=10
```

2.Node parameter

Optimization:

--trace_opt (trace lazy optimization)

--trace_opt_stats (trace lazy optimization statistics)

--trace_deopt (trace deoptimization)

--trace_bailout (print reasons for falling back to using the classic V8 backend)

GC:

--trace_gc (print one trace line following each garbage collection)

--trace_gc_nvp (print one detailed trace line in name=value format after each garbage collection)

--print_cumulative_gc_stat (print cumulative GC statistics in name=value format on exit)

--trace_gc_verbose (print more details following each garbage collection)

3.Manual

--noprof-auto

profiler.startProfiling('startup'); - start/resume collection of data

profiler.stopProfiling - pause collection of data

v8-profiler

-
  ```
  var profiler = require('v8-profiler');
  profiler.startProfiling('startup');
  slowStartupFoo();
  profiler.stopProfiling('startup');
  profiler.takeSnapshot('beforeLeak');
  leakyFoo();
  profiler.takeSnapshot('afterLeak');
  ```

Node App 应用层面运维建议

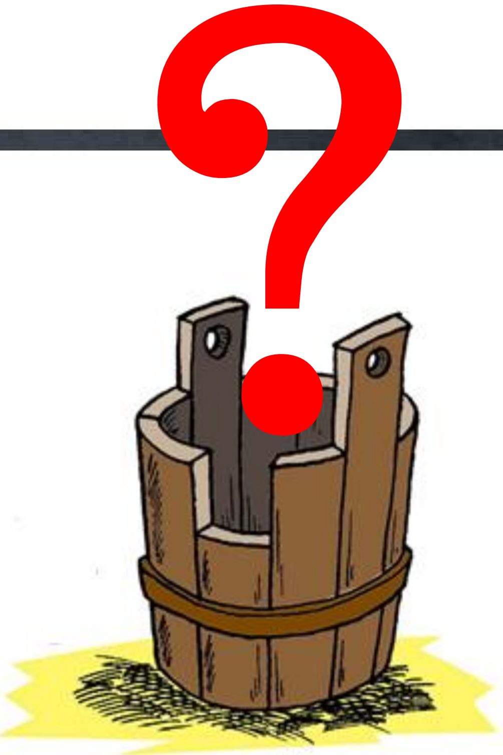
- 定期收集运行信息（建议秒级别）
  - process.memoryUsage()
    - { rss, heapTotal, heapUsed}
  - process.uvCounters()
    - eio_init、 req_init、handle_init,
    - stream_init、tcp_init、udp_init,
    - prepare_init、check_init
    - idle_init、async_init
    - timer_init:、process_init、fs_event_init
- 定期开启profiler
  - 收集关键函数调用时间
  - 收集堆栈信息
- 其它IO收集
  - 请求数、响应时间
  - 内部系统交互响应时间等

2

分布式设计探讨

分布式设计（探讨）

- 单机：
  - 多进程（domain socket）
    - cluster
    - multi-node
- 集群
  - 节点无交互
    - Proxy（nginx proxy..）
    - LVS..
  - 节点有交互
    - RPC（缺点？）
      - thrift、rest、web services
- 高并发系统特性
  - 消息交互
  - 无状态
  - 异步？



Nodejs集群（复杂计算逻辑 +异构系统）

- ZEROMQ
  - 跨多种传输协议和方式
    - 进程内通讯
    - **IPC**
    - **TCP**
    - 广播
  - 多连接模型
    - REQ/REP
    - PUB/SUB
    - PUSH/PULL
  - 全局拓扑
    - 智能感知路由
  - 无锁
  - 异步消息交互
  - 低延迟高并发
  - 接口高度一致

REQ/REP模型

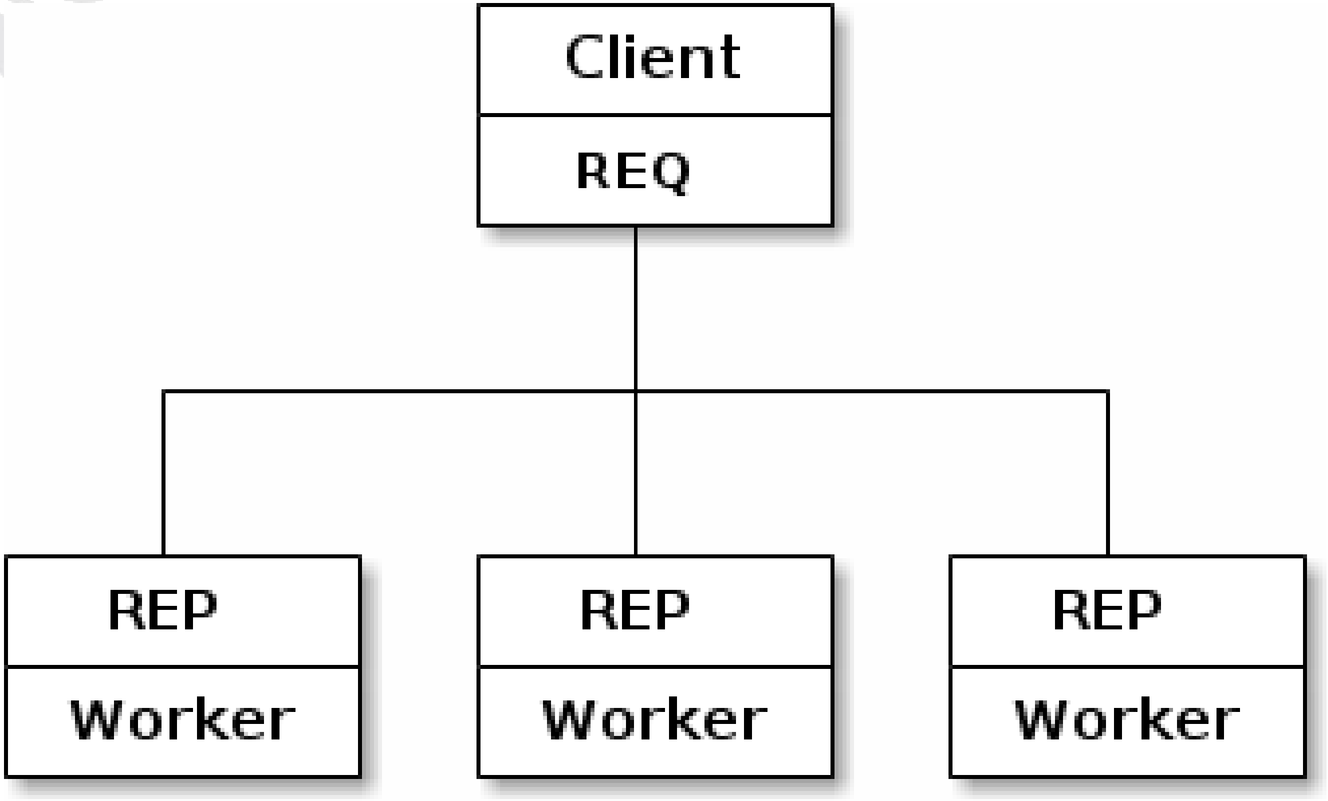


Figure 39 – Basic request-reply

RE

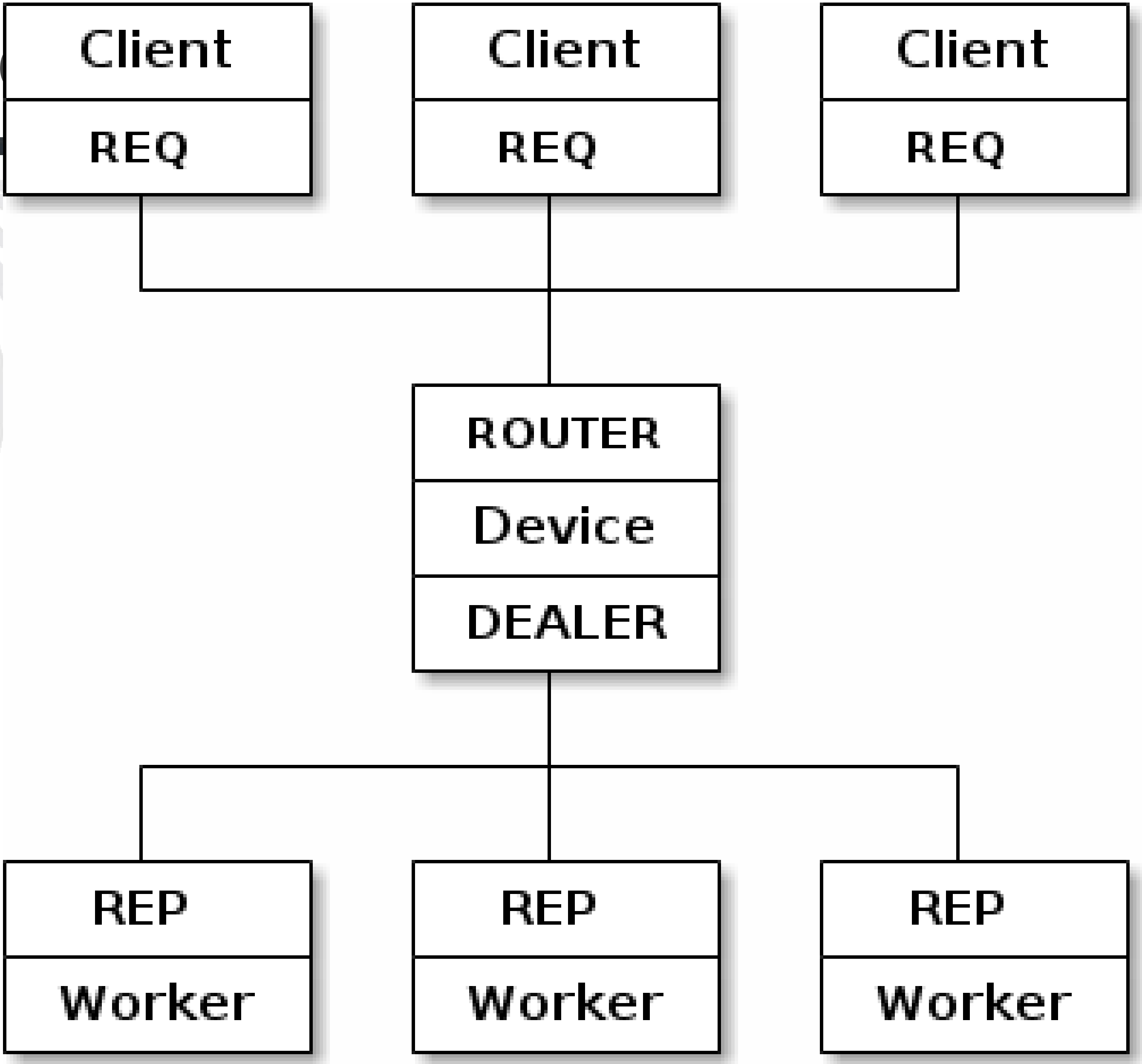


Figure 40 — Stretched request-reply

REQ/REP模型

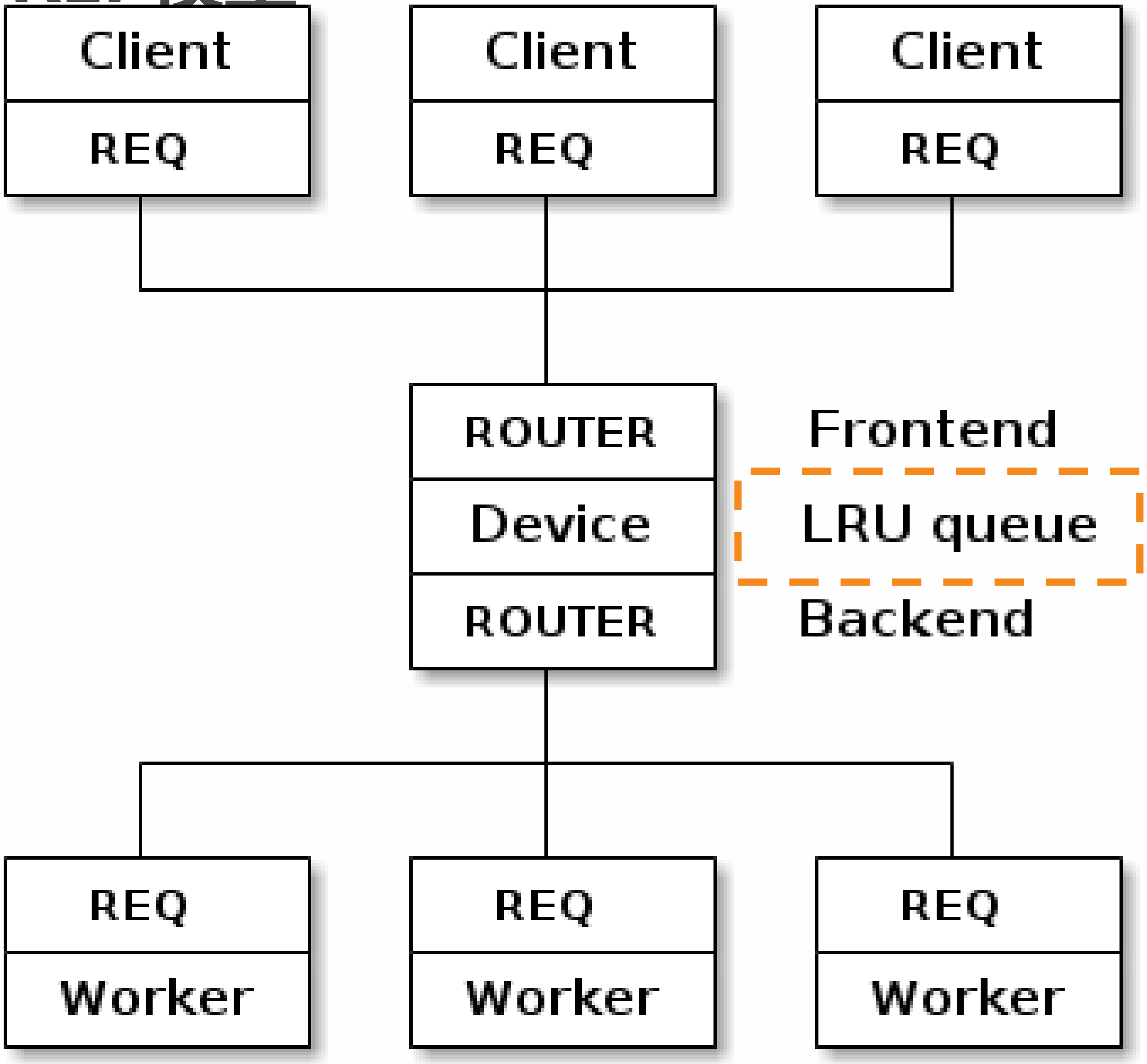


Figure 41 – Stretched request-reply with LRU

REQ/REP模型

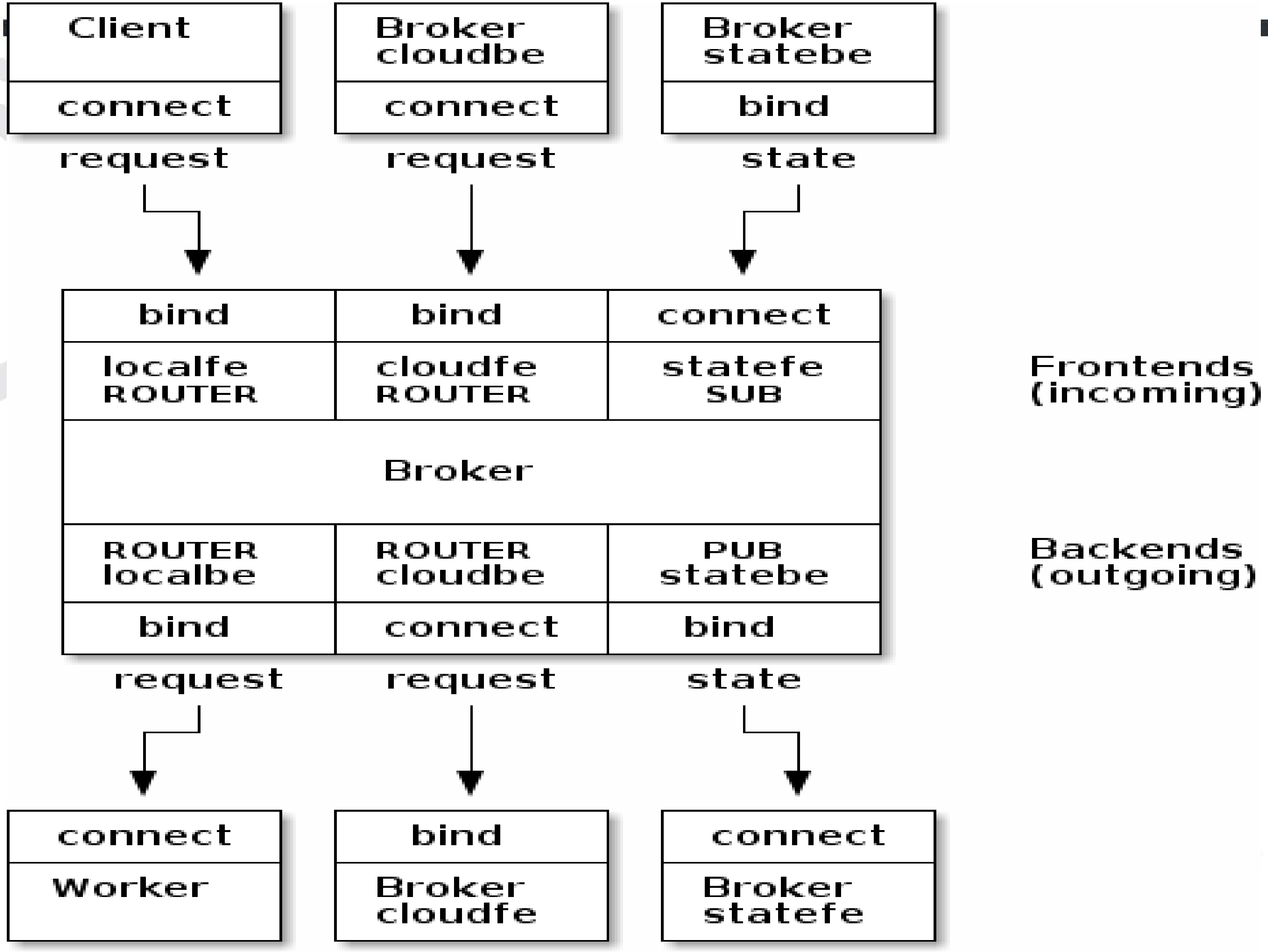


Figure 53 — Broker socket arrangement

REQ/REP模型

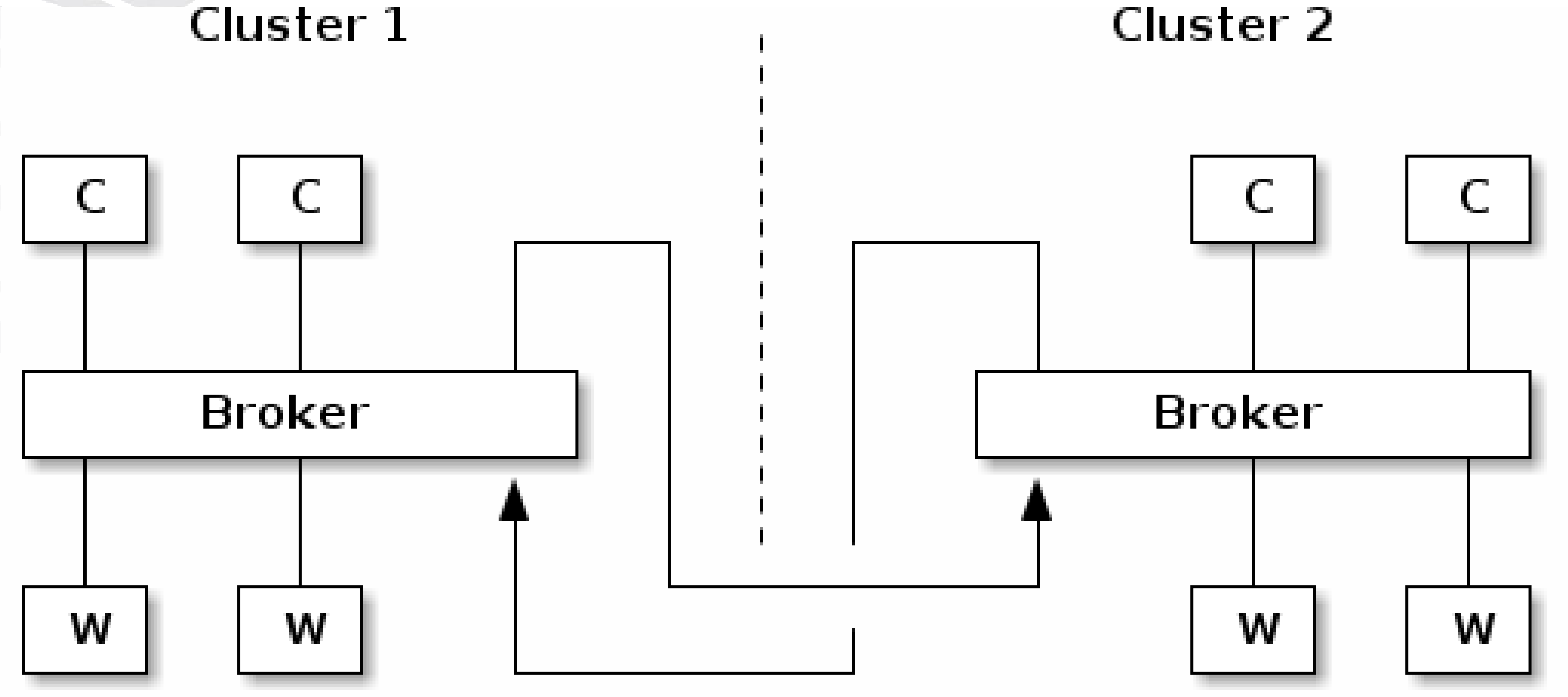


Figure 52 — Cross connected brokers in federation model

Sample

```
var zmq = require('zmq')
  , sock = zmq.socket('rep');
var i = 0;
sock.bindSync(url);
sock.on('message', function(msg){
});
```

代码不用变

url：

‘ipc:///tmp/zmq‘ ——进程间通讯

‘tcp://*:23456‘ ——网络

推荐：

- 编程规范：
  - http://cnodejs.org/blog/?p=4739
  - https://github.com/windyrobin/iFrame/
- Blazing fast node.js: 10 performance tips from LinkedIn Mobile
- Efficient JavaScript
- JavaScript performance playground

Q & A