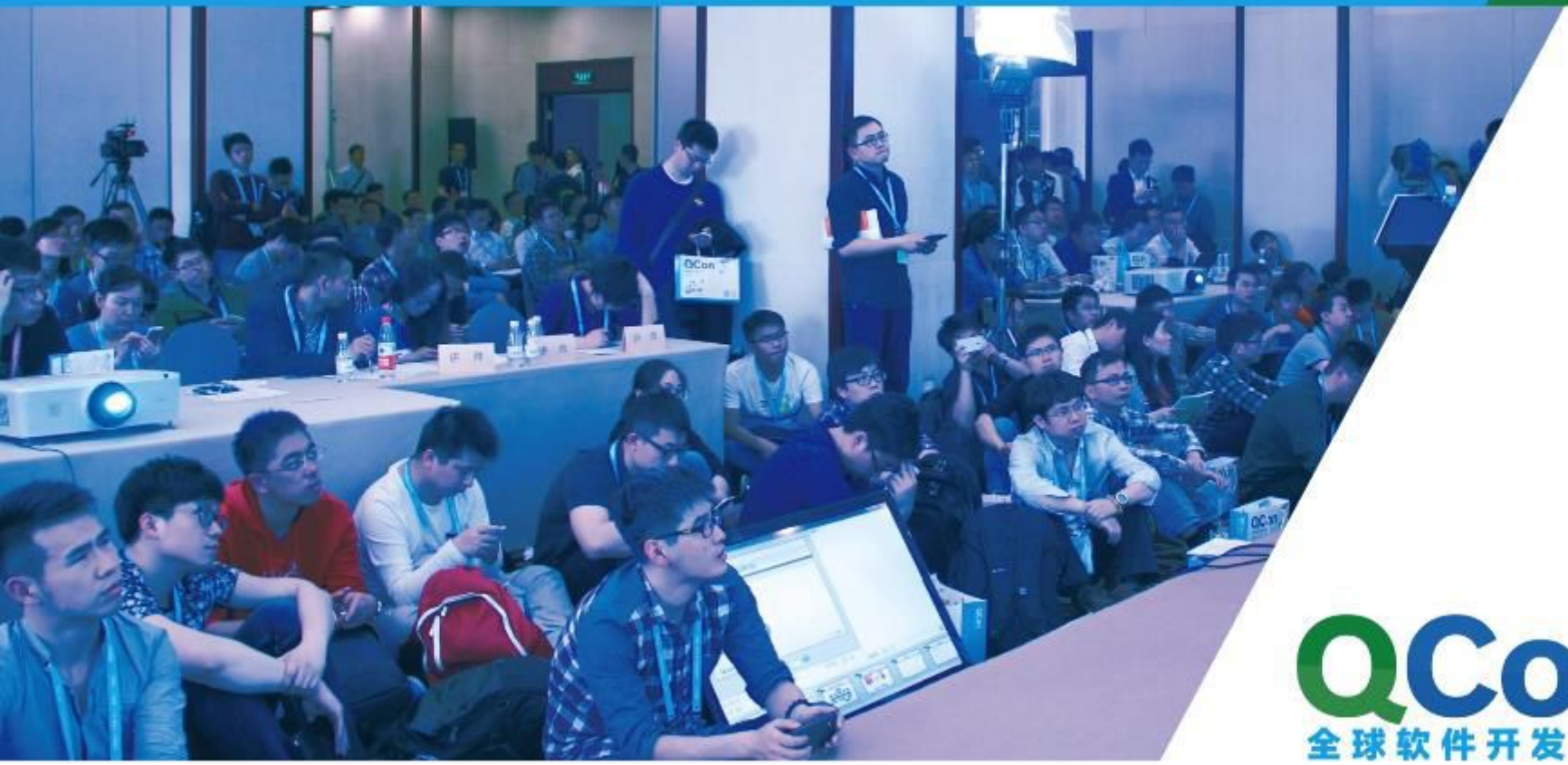
QCon全球软件开发大会

International Software Development Conference



QCon
全球软件开发大会

Nodejs与实时Baas云开发实践

野狗实时 谢乔

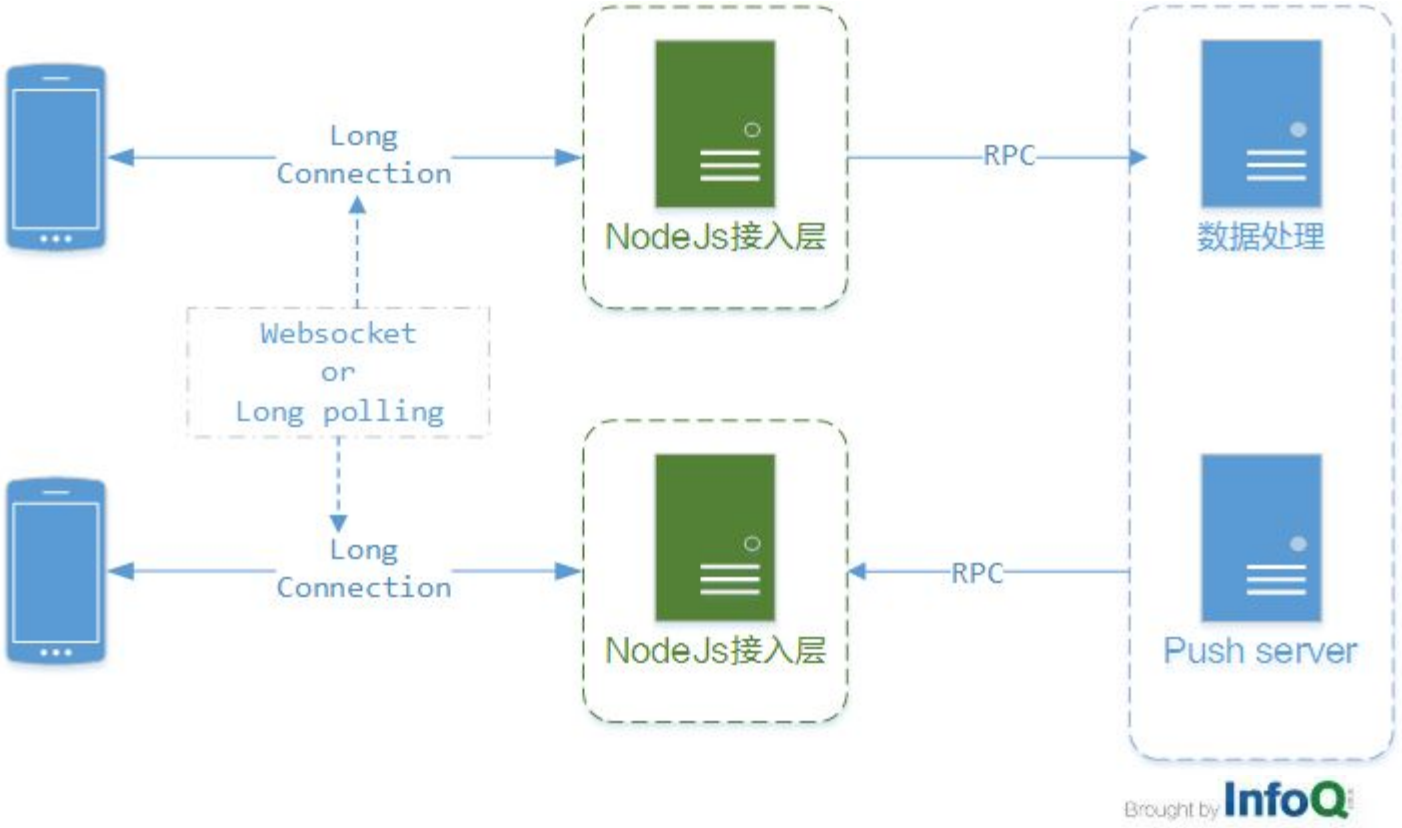
大纲

- NodeJs 开发实践
- 长连接接入层架构

NodeJs 开发实践

- 服务端
  - Websocket 长连接
  - Long polling
  - Rest API
- 网站前端
  - 前后端分离
  - 高效的上线发布
  - Grunt Bower

- 服务模型
  - 统一接入层的实现
  - 双向通信长连接
  - Long-polling，需要Session维持不稳定的长连接



为什么使用Node.js

- 开发效率高、易维护
- 单进程高并发
- 线程安全，无锁
- 适合IO密集
- 减少CS，节省栈空间
- 适合请求分发、透传、轻业务逻辑

还不错的特性

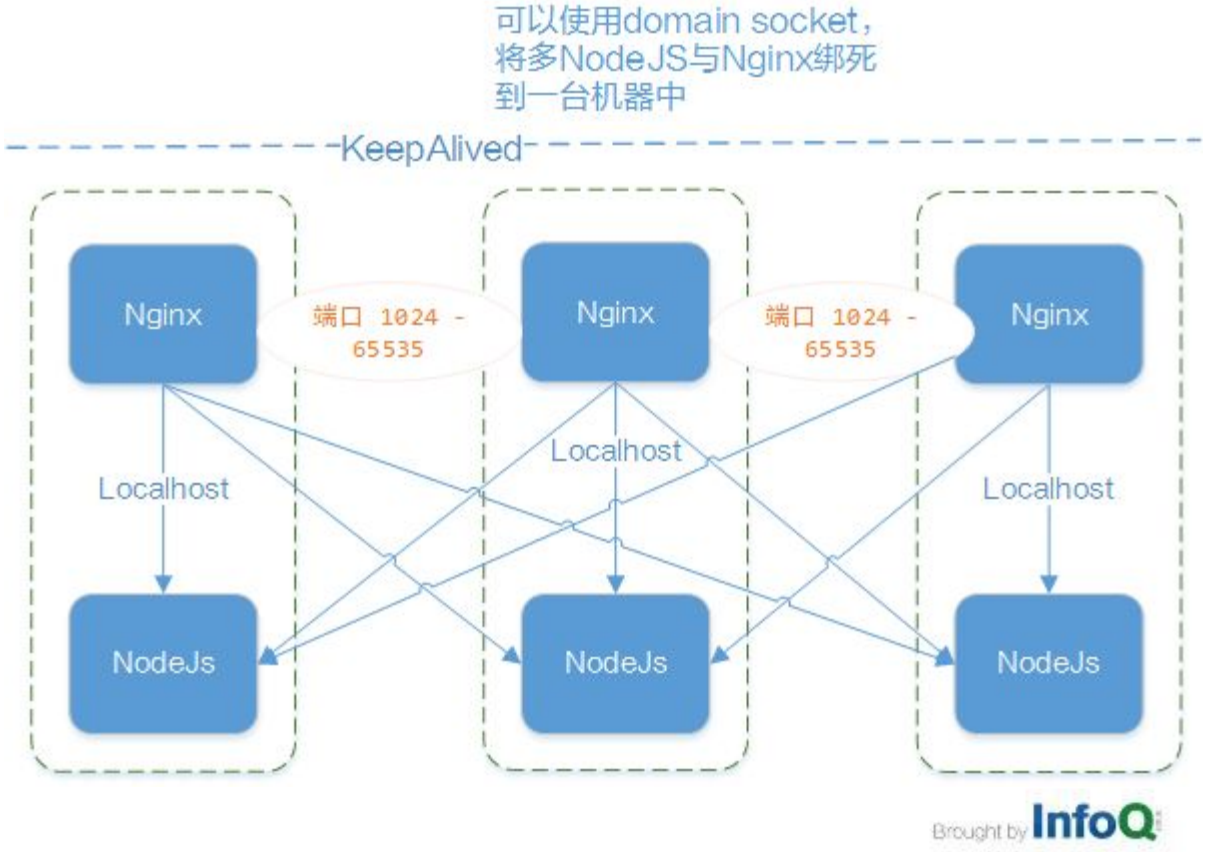
- 异步编程的习惯
- V8的垃圾回收
- Incremental Marking 增量标记
- Openssl AES-NI
- Debug，chrome profiles

需要克服的缺陷

- 单进程，进程崩溃，game over
- 不能充分利用多核
- 用户单线程执行，同步代码影响性能
- 可靠性差

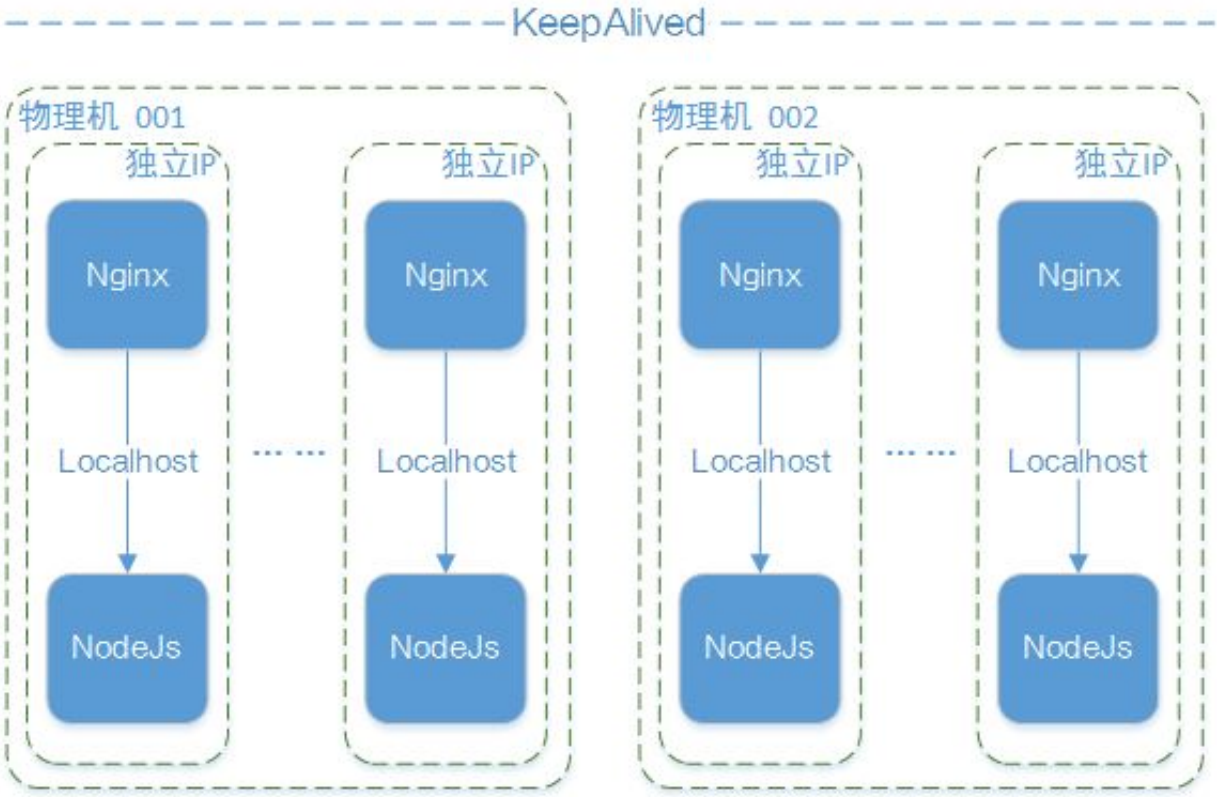
方案一　Nginx + NodeJs

- 优势
  - 并发建连能力强
  - TLS 加解密优势
  - 负载均衡
  - 权重控制
  - 健康检查
  - IP Hash
- 劣势
  - Port限制
  - 长连接开销 x 2



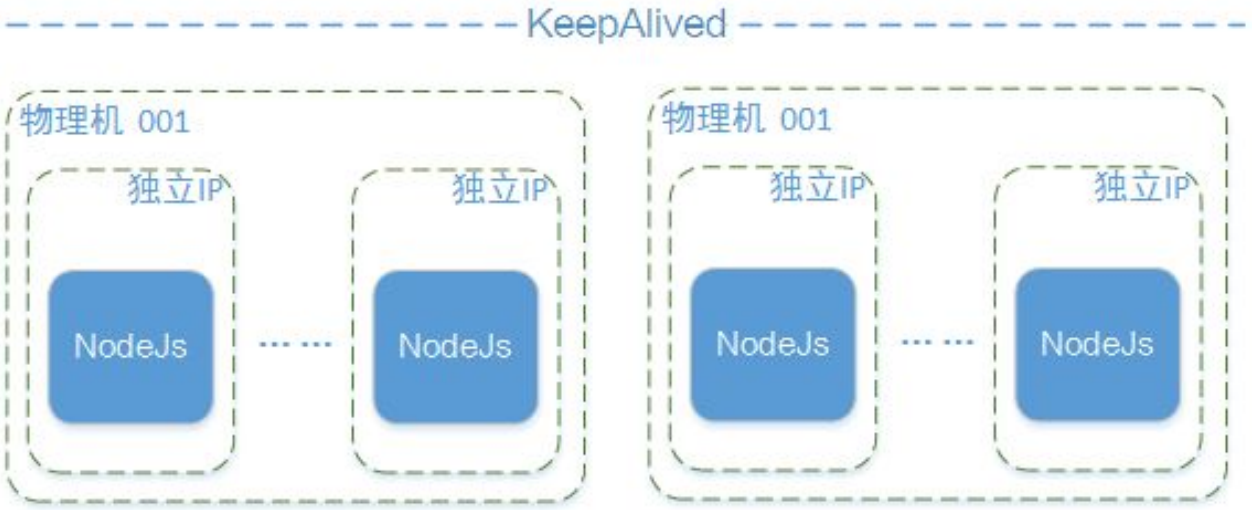
方案二 虚拟机 + Nginx + NodeJs

- 优势
  - 并发建连能力强
  - TLS 加解密优势
  - 充分利用资源
- 劣势
  - 单点故障
  - 运维成本高
  - 没有负载均衡
  - 长连接开销 x 2



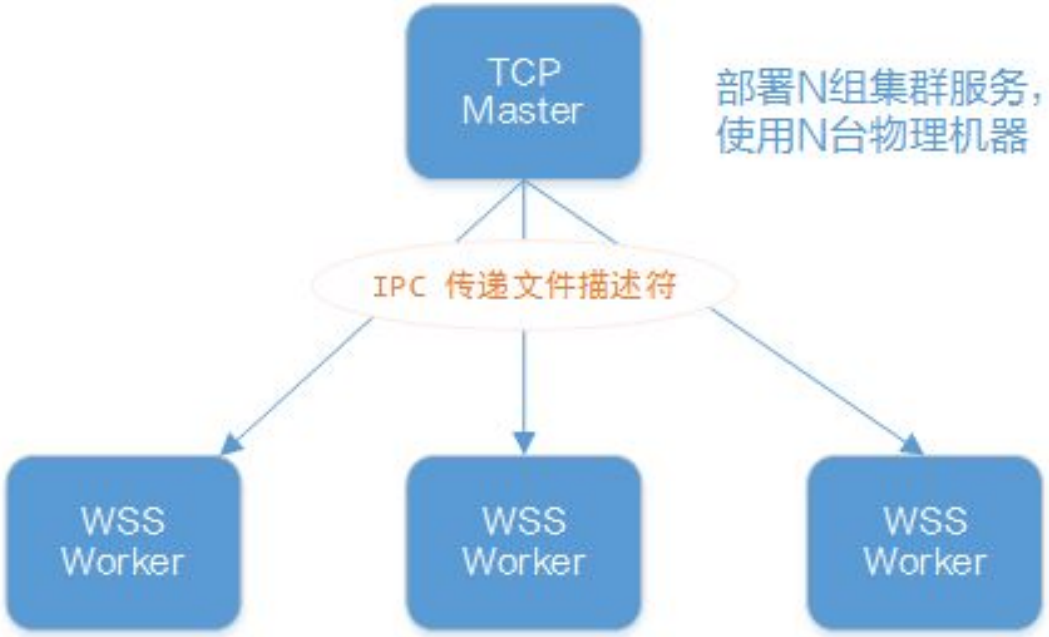
方案三　虚拟机 + NodeJs

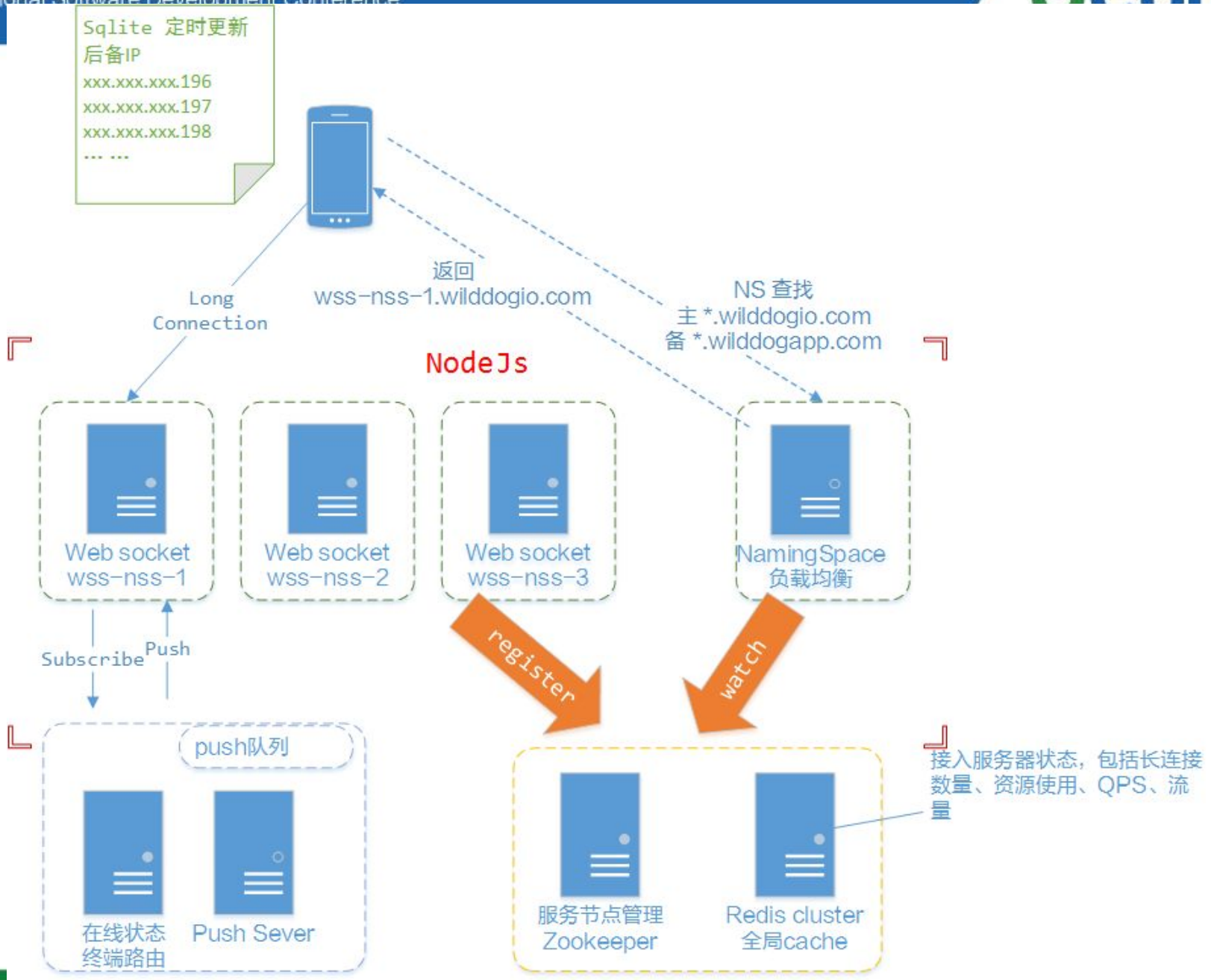
- 直接建立连接
- TLS建连实际表现不俗
- 问题依旧存在



方案四　NodeJs Master - Workers

- 优势
  - 简单，运维成本低
  - TLS建连实际表现不俗
- 劣势
  - 没有负载均衡
  - Master单点故障
- 解决方案
  - Client-Server负载均衡
  - Client 重连机制
  - Server可用性与伸缩性



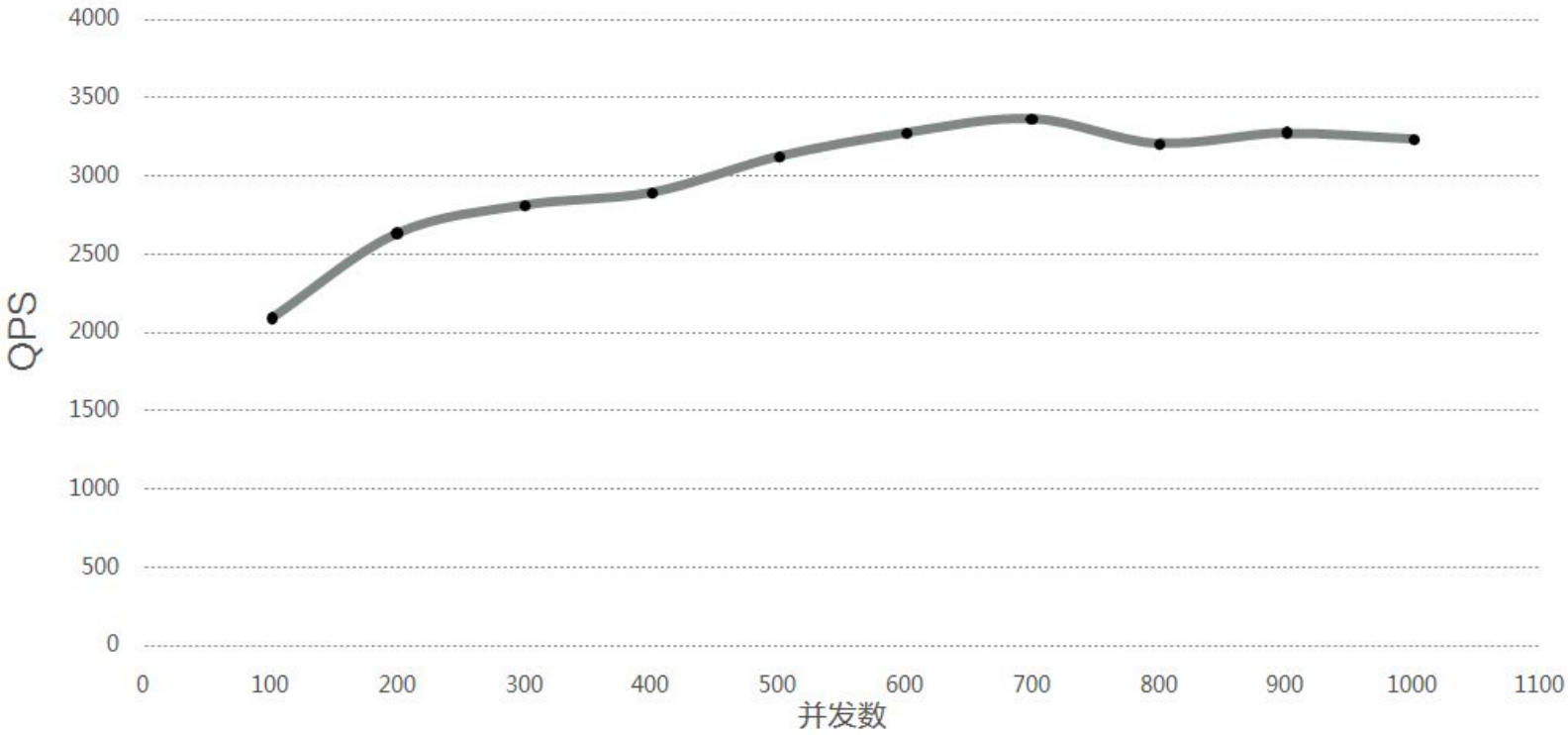

Sqlite 定时更新
后备IP
xxx.xxx.xxx.196
xxx.xxx.xxx.197
xxx.xxx.xxx.198
... ...
Long Connection
返回
wss-nss-1.wilddogio.com
NS 查找
主 *.wilddogio.com
备 *.wilddogapp.com
NodeJs
Web socket
wss-nss-1
Web socket
wss-nss-2
Web socket
wss-nss-3
NamingSpace
负载均衡
Subscribe
Push
register
watch
push队列
在线状态
终端路由
Push Sever
服务节点管理
Zookeeper
Redis cluster
全局cache
接入服务器状态，包括长连接数量、资源使用、QPS、流量

Nodejs 集群

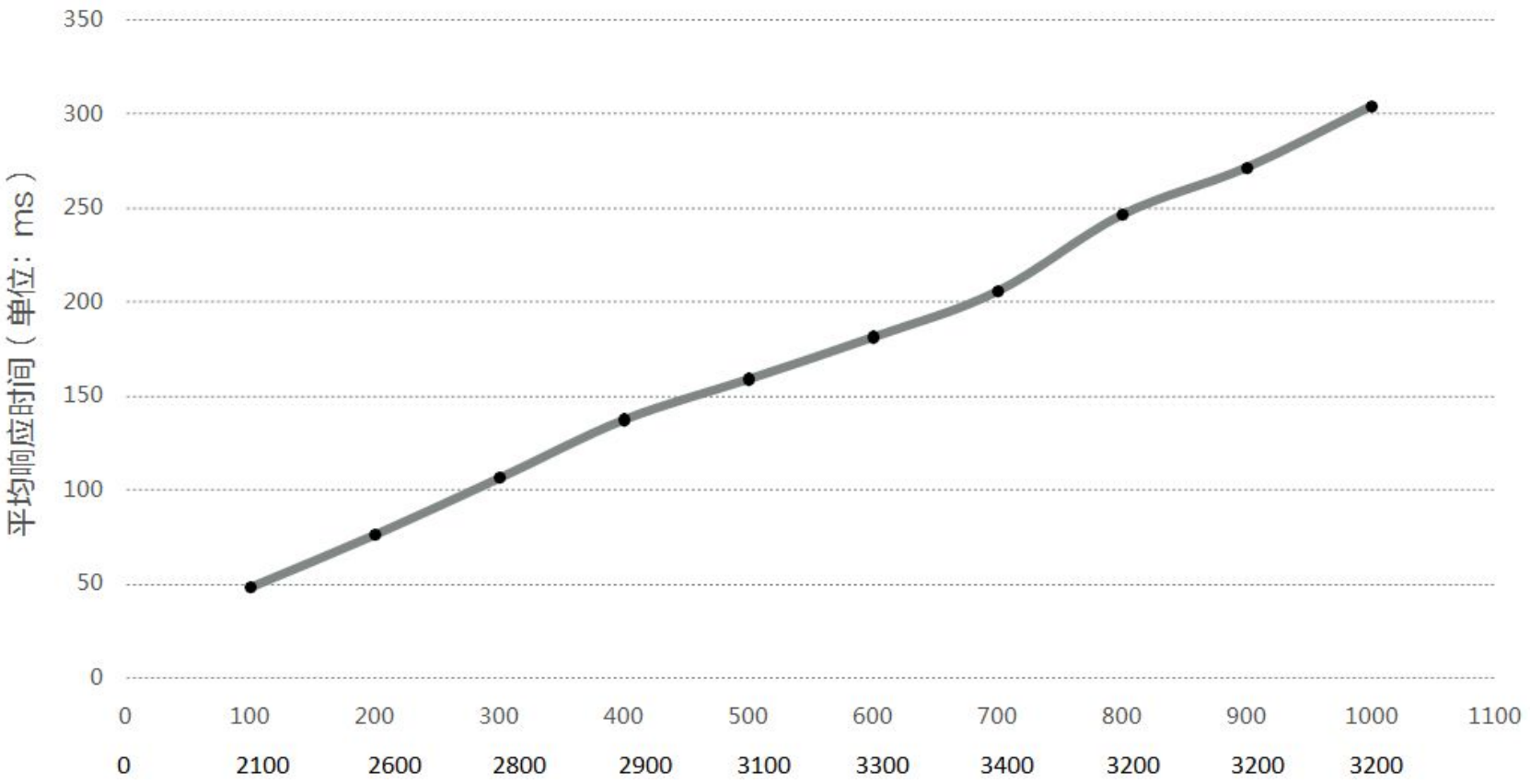
- 规模
  - 12核24线程，16woker
  - 单进程50K连接
- 集群内负载均衡
  - IP Hash
  - Round-Robin + 外部全局cache

Nodejs 性能

使用WRK Connection:close



单进程Nginx的Https为6100Req/S



注意：延迟是包含TCP+TLS握手

Nodejs 垃圾回收

- --min_semi_space_size=128 --max_semi_space_size=256

```
New space,          used:   4476 KB, available: 124515 KB, committed: 257984 KB
Old space,          used:  16926 KB, available:    542 KB, committed:  18139 KB
Code space,         used:   3663 KB, available:    858 KB, committed:   4980 KB
Map space,          used:   1171 KB, available:    776 KB, committed:   2015 KB
Large object space, used:      0 KB, available: 2591727 KB, committed:      0 KB
All spaces,         used:  26237 KB, available: 2718420 KB, committed: 283119 KB
```

```
Scavenge 774.8 (1869.0) -> 767.3 (1869.0) MB, 84.3 / 0 ms [allocation failure].
Scavenge 823.5 (1869.0) -> 817.8 (1869.0) MB, 80.7 / 0 ms [allocation failure].
Scavenge 896.4 (1869.0) -> 889.6 (1869.0) MB, 86.4 / 0 ms [allocation failure].
Scavenge 946.9 (1869.0) -> 941.7 (1869.0) MB, 82.7 / 0 ms [allocation failure].
Scavenge 1018.7 (1869.0) -> 1012.5 (1869.0) MB, 82.9 / 0 ms [allocation failure].
Scavenge 1070.8 (1869.0) -> 1065.6 (1869.0) MB, 86.0 / 0 ms [allocation failure].
Scavenge 1141.6 (1869.0) -> 1135.2 (1869.0) MB, 83.6 / 0 ms [allocation failure].
Scavenge 1195.4 (1869.0) -> 1190.5 (1869.0) MB, 84.4 / 0 ms [allocation failure].
Scavenge 1265.0 (1869.0) -> 1258.5 (1869.0) MB, 80.2 / 0 ms [allocation failure].
Scavenge 1319.6 (1869.0) -> 1314.5 (1869.0) MB, 91.2 / 0 ms [allocation failure].
Scavenge 1390.7 (1869.0) -> 1383.7 (1886.0) MB, 83.2 / 0 ms (+ 110.6 ms in 1080 steps since
```

- Old space

```
--incremental_marking (use incremental marking)
        type: bool  default: true
--incremental_marking_steps (do incremental marking steps)
        type: bool  default: true
```

```
Mark-sweep 533.8 (944.0) -> 206.4 (778.0) MB,
105.4 / 0 ms (+ 205.0 ms in 1729 steps since start of marking, biggest step 0.6 ms)

Mark-sweep 1115.6 (1571.0) -> 463.1 (1515.0) MB,
95.1 / 0 ms (+ 385.8 ms in 2528 steps since start of marking, biggest step 1.2 ms)

Mark-sweep 1313.8 (1789.0) -> 443.8 (1696.0) MB,
108.8 / 0 ms (+ 447.7 ms in 1910 steps since start of marking, biggest step 1.7 ms)

Mark-sweep 1558.3 (2012.0) -> 1193.2 (1982.0) MB,
115.0 / 0 ms (+ 1033.8 ms in 3849 steps since start of marking, biggest step 3.4 ms)
```

- gc trace

```
int external_time = static_cast<int>(current_.scopes[Scope::EXTERNAL]);
double duration = current_.end_time - current_.start_time;
Output("%.1f / %d ms", duration, external_time);

if (current_.type == Event::SCAVENGER) {
  if (current_.incremental_marking_steps > 0) {
    Output(" (+ %.1f ms in %d steps since last GC)",
           current_.incremental_marking_duration,
           current_.incremental_marking_steps);
  }
} else {
  if (current_.incremental_marking_steps > 0) {
    Output(
        " (+ %.1f ms in %d steps since start of marking, "
        "biggest step %.1f ms)",
        current_.incremental_marking_duration,
        current_.incremental_marking_steps,
        current_.longest_incremental_marking_step);
  }
}
```

Nodejs Master

- accept tcp socket
- pauseOnConnect
- 处理IPC失败，关闭socket

```
server = net.createServer(function (socket) {

    socket.pause();

    var worker = getWorker(socket);
```

```
server = net.createServer({pauseOnConnect: true}, function (socket) {

    var worker = getWorker(socket);

    worker.send('master-tcp-socket', socket, function (error, arg) {
        if (error) {
            ...
            socket.destroy();
            return ;
        }
    });

});
```

Nodejs Worker

- 没有实际的listener
- 手动emit connection

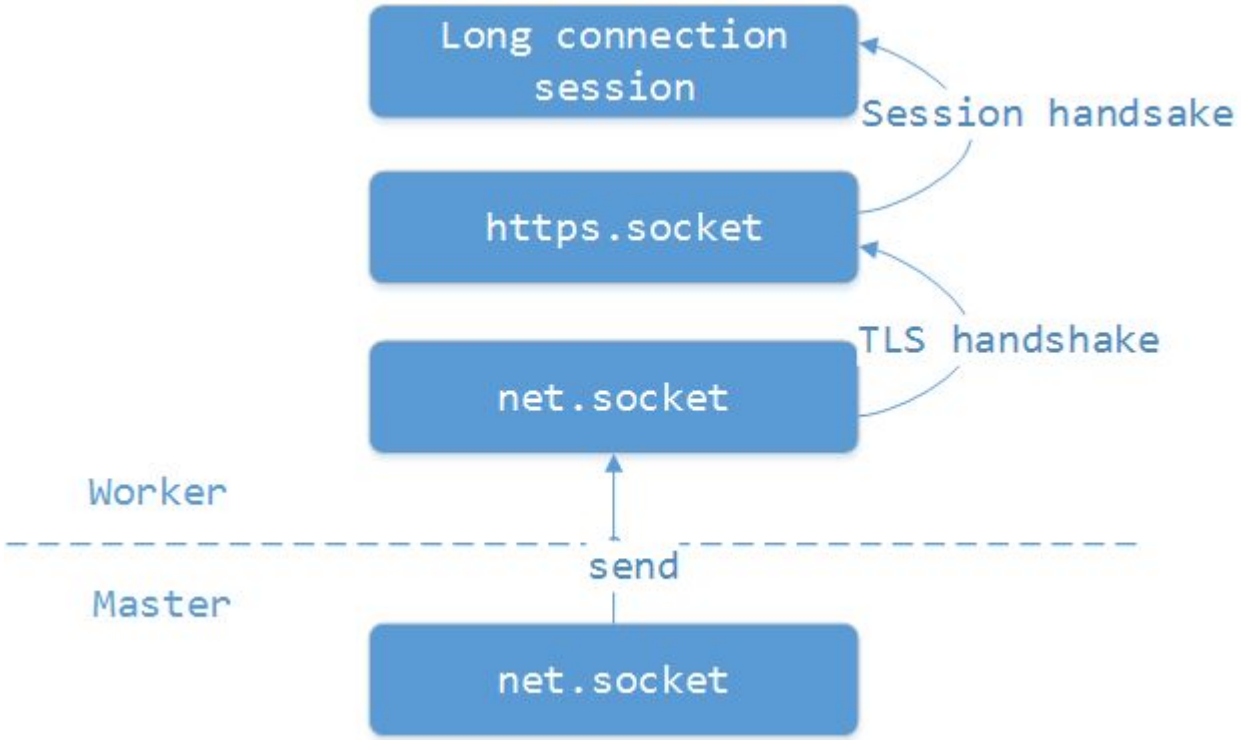
```
process.on('message', function (msg, socket) {
    if (msg !== 'master-tcp-socket') {
        return;
    }

    socket.setTimeout(120 * 1000, function() {
        socket.destroy();
    });
    // server = https.createServer
    server.emit('connection', socket);
    // resume socket read
    socket.resume();
});
```

Nodejs 健壮性保障

- Master
  - 外部watchdog
  - 检测Master进程是否存活
  - 检测Master的端口是否存活
- Worker
  - Master监听woker事件
  - 内部watchdog
  - 向指定worker建立连接，模拟请求测试是否存活
  - 注意有坑：遇到大量建立连接时，可能checkAlive的建连超时，误判为worker已死。

- Socket 泄漏
  - 每一层的socket都需要setTimeout

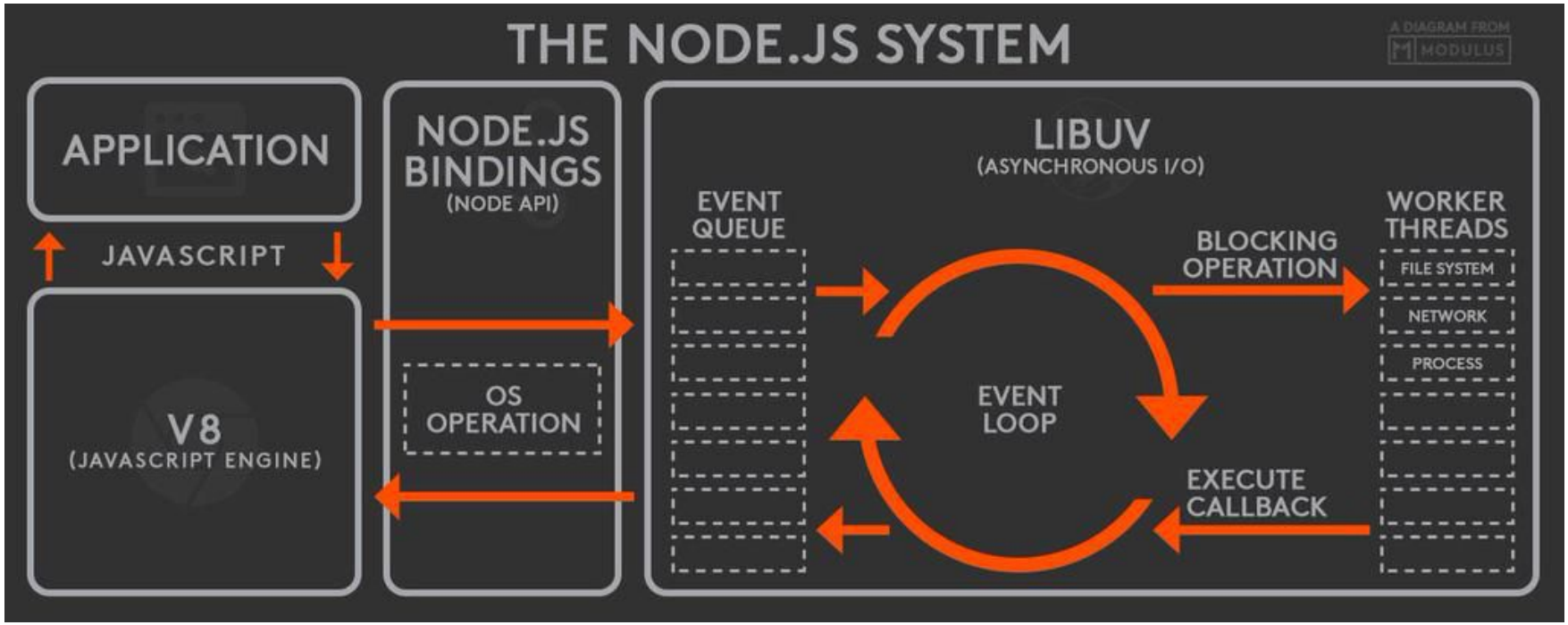


开发 Tips

- Master 只接收TCP socket
  - TLS handshake 与 crypt 交给worker
- 将阻塞代码交给backend server
  - payload内容的加解密
  - 大数据的Json解析

- Session resume
  - 减少Long polling的消耗
  - 每个worker进程内缓存
  - 刷新缓存超时
  - 定时回收长时间不使用的
- OCSP stapling

- Asynchronous threads



本图出自
[https://twitter.com/BusyRich/status/494959181871316992]

- defualt 4

```
static uv_thread_t default_threads[4];
```

- Max 128

```
nthreads = ARRAY_SIZE(default_threads);
val = getenv("UV_THREADPOOL_SIZE");
if (val != NULL)
  nthreads = atoi(val);
if (nthreads == 0)
  nthreads = 1;
if (nthreads > MAX_THREADPOOL_SIZE)
  nthreads = MAX_THREADPOOL_SIZE;

threads = default_threads;
if (nthreads > ARRAY_SIZE(default_threads)) {
  threads = uv__malloc(nthreads * sizeof(threads[0]));
  if (threads == NULL) {
    nthreads = ARRAY_SIZE(default_threads);
    threads = default_threads;
  }
}
```

- UV_THREADPOOL_SIZE

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
61097	**root**	**20**	**0**	**18.484g**	**0.012t**	**7552**	**R**	**95.7**	**83.1**	**0:07.27**	**node**
61108	root	20	0	18.484g	0.012t	7552	S	50.8	83.1	0:04.33	node
61105	root	20	0	18.484g	0.012t	7552	S	50.2	83.1	0:04.03	node
61106	root	20	0	18.484g	0.012t	7552	S	48.8	83.1	0:04.14	node
61107	root	20	0	18.484g	0.012t	7552	S	48.2	83.1	0:03.83	node
61099	root	20	0	18.484g	0.012t	7552	S	1.0	83.1	0:00.05	V8 WorkerThread
61100	root	20	0	18.484g	0.012t	7552	S	1.0	83.1	0:00.04	V8 WorkerThread
61098	root	20	0	18.484g	0.012t	7552	S	0.7	83.1	0:00.05	V8 WorkerThread
61101	root	20	0	18.484g	0.012t	7552	S	0.7	83.1	0:00.04	V8 WorkerThread

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
61210	**root**	**20**	**0**	**11.592g**	**7.757g**	**7560**	**R**	**93.7**	**50.4**	**0:04.18**	**node**
61219	root	20	0	11.592g	7.757g	7560	S	28.9	50.4	0:01.24	node
61221	root	20	0	11.592g	7.757g	7560	S	28.9	50.4	0:01.22	node
61224	root	20	0	11.592g	7.757g	7560	S	28.9	50.4	0:01.24	node
61225	root	20	0	11.592g	7.757g	7560	S	28.2	50.4	0:01.22	node
61227	**root**	**20**	**0**	**11.592g**	**7.757g**	**7560**	**R**	**27.9**	**50.4**	**0:01.22**	**node**
61218	root	20	0	11.592g	7.757g	7560	S	27.6	50.4	0:01.20	node
61220	**root**	**20**	**0**	**11.592g**	**7.757g**	**7560**	**R**	**27.6**	**50.4**	**0:01.26**	**node**
61222	**root**	**20**	**0**	**11.592g**	**7.757g**	**7560**	**R**	**27.6**	**50.4**	**0:01.19**	**node**
61223	root	20	0	11.592g	7.757g	7560	S	27.6	50.4	0:01.20	node
61226	root	20	0	11.592g	7.757g	7560	S	26.9	50.4	0:01.18	node
61212	root	20	0	11.592g	7.757g	7560	S	0.7	50.4	0:00.02	V8 WorkerThread
61213	root	20	0	11.592g	7.757g	7560	S	0.7	50.4	0:00.02	V8 WorkerThread
61214	root	20	0	11.592g	7.757g	7560	S	0.7	50.4	0:00.02	V8 WorkerThread
61211	root	20	0	11.592g	7.757g	7560	S	0.3	50.4	0:00.01	V8 WorkerThread

- 使用uv_queue_worker
  - node_zlib
  - node_crypto
  - AddOn

```
struct AsyncTask {
    // required
    uv_work_t request;                    // libuv
    Persistent<Function> callback;        // js callback
    // other ...
};
```

```
Handle<Value> asyncExecute(const Arguments &args) {
    AsyncTask *task = new AsyncTask;
    // callback argument
    Handle<Function> cb = Handle<Function>::Cast(args[1]);
    // attach task to uv work request
    task->request.data = task;

    // args[0] ...

    task->callback = Persistent<Function>::New(cb);
    // uv_default_loop is the node.js event loop
    uv_queue_work(uv_default_loop(), &task->request, execute, executeAfter);

    return Undefined();
}
```

- 使用实例池
  - lib/internal/
  - freelist

```
'use strict';

// This is a free list to avoid creating so many of the same object.
exports.FreeList = function(name, max, constructor) {
  this.name = name;
  this.constructor = constructor;
  this.max = max;
  this.list = [];
};


exports.FreeList.prototype.alloc = function() {
  return this.list.length ? this.list.shift() :
                            this.constructor.apply(this, arguments);
};


exports.FreeList.prototype.free = function(obj) {
  if (this.list.length < this.max) {
    this.list.push(obj);
    return true;
  }
  return false;
};
```

野狗科技官方订阅号

野狗实时BaaS交流QQ群

THANKS