Node.js携程落地和最佳实践

潘斐斐

携程高级研发经理

- Node.js框架平台整体构建
- 产品性能优化
- 创新型项目研发

Contents

面临的问题

面临的问题

面临的问题

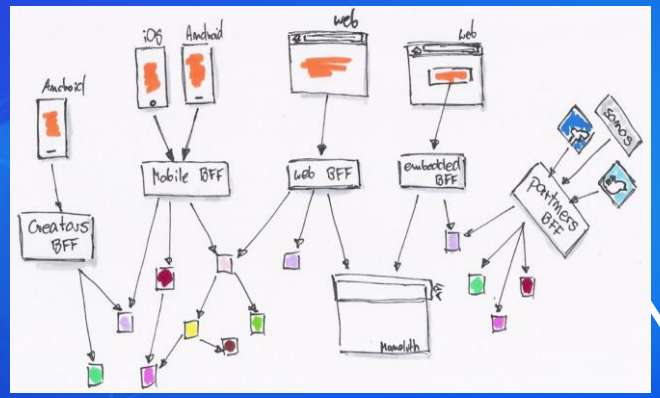
经过 2 年，应用数实现 8x 增长，覆盖公司 33 个业务部门

携程Node.js工程化

Node.js的场景

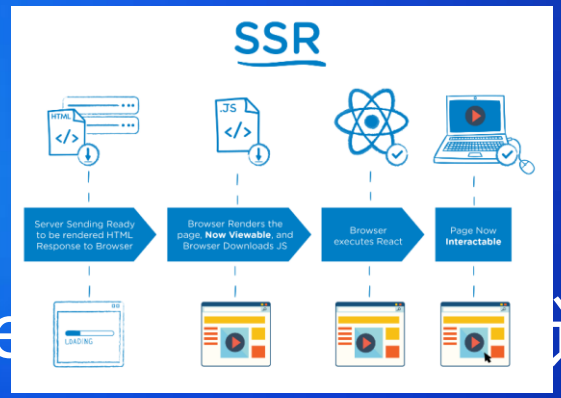
DA

Data Aggregation



SSR

Server Side Rendering



Tools

Desktop Tools

Node 述的三

Node.js工程化

开发 → 构建(CI) → 测试 → 发布 → 运维

- 根据业务需求，提供适用于各场景的**脚手架**

- ✓ Web Application(SSR)
- ✓ DA Service
- ✓ Desktop Application

```
~/ctrip_work
$ mkdir my-first-ctrip-node

~/ctrip_work
$ cd my-first-ctrip-node

~/ctrip_work/my-first-ctrip-node
$ npx @ctrip/node-starter-cli init
npx: 10 安装成功，用时 2.695 秒
Project Name (my-first-ctrip-node):
AppID (Check it from http:[redacted] For example:10003344)): 10018233
Local Port (8888): 8080
```

Node.js工程化

开发 → 构建(CI) → 测试 → 发布 → 运维

- 提供核心中间件
  -- 20+个中间件
- 数据Mock平台
  -- 聚合数据
- Docker化开发
  -- 开发和构建环境保持一致

Node.js工程化

开发 | 构建(CI) | 测试 | 发布 | 运维

- 制作Docker镜像
  -- 维护2个Node.js固定LTS版本
- 安装依赖包、编译扩展包
  -- node_modules缓存，C++模块预编译包
- 检查依赖包版本
  -- 框架核心中间件的统一升级
- 构建目标文件
  -- webpack / Babel / TSC / 虚拟文件系统VFS

Node.js工程化

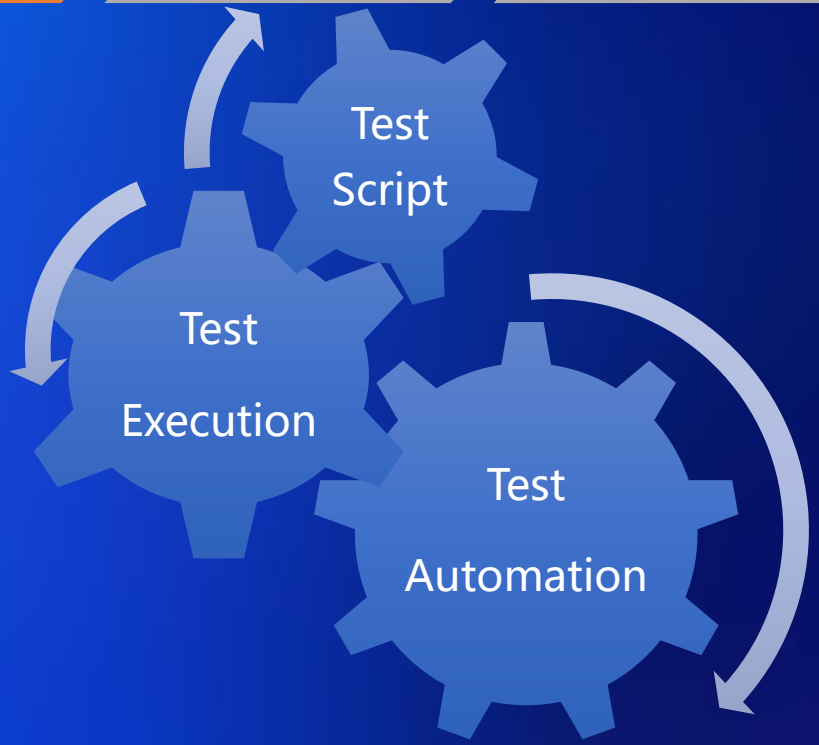
开发 | 构建(CI) | 测试 | 发布 | 运维

- 单元测试、代码覆盖率
  - -- 国际化检查
  - -- 项目质量检查
- 安全扫描
  - -- 扫描特定有隐患的第三方模块，设置为黑名单

Node.js工程化

开发 → 构建(CI) → 测试 → 发布 → 运维

- 自动化测试
- 集成测试
- 预发布灰度测试
- 压力测试



Node.js工程化

开发 → 构建(CI) → 测试 → 发布 → 运维

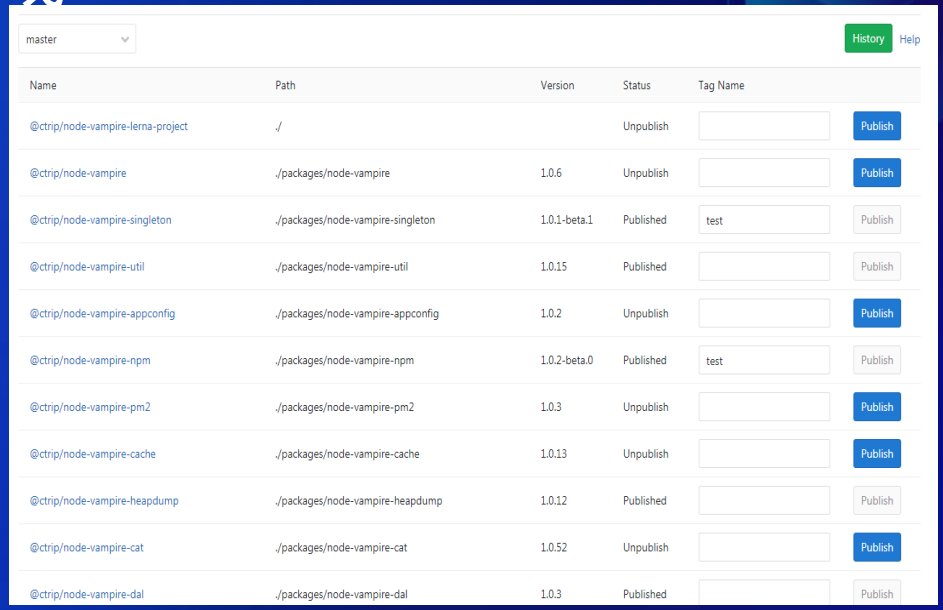
- 选择携程云或者公有云
  -- 基础设施和核心中间件将云之间的差异抹平
- 灰度与回滚
  -- 快速回滚机制
- Node.js应用一体化发布
  -- 应用/资源/react native bundle

Node.js工程化

开发 → 构建(CI) → 测试 → 发布 → 运维

内部npm包开发发布流程与GIT高度集成

- 通过git仓库发布模块
- npm包的修改有历史可查
- 可做限制（权限控制、unpublish操作）



Node.js工程化

开发 → 构建(CI) → 测试 → 发布 → 运维

日志监控

Tracing(Request scoped)

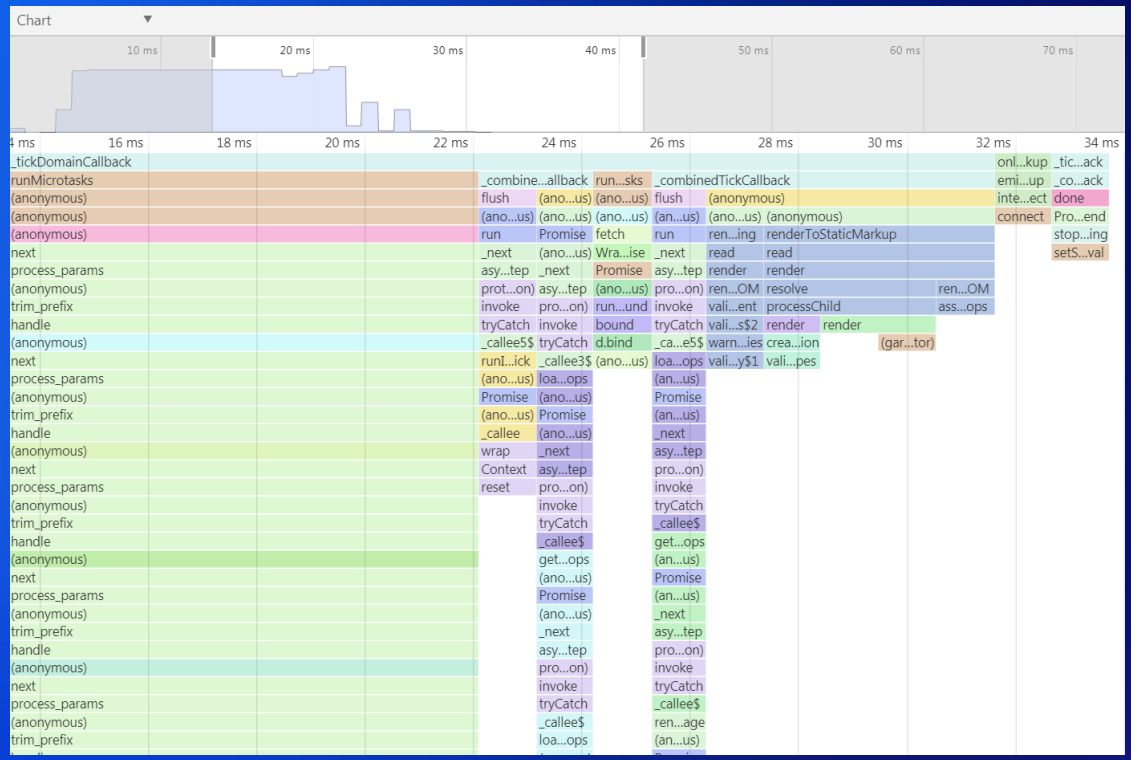
Logging(Events)

Metrics (Aggregatable)

应用排障

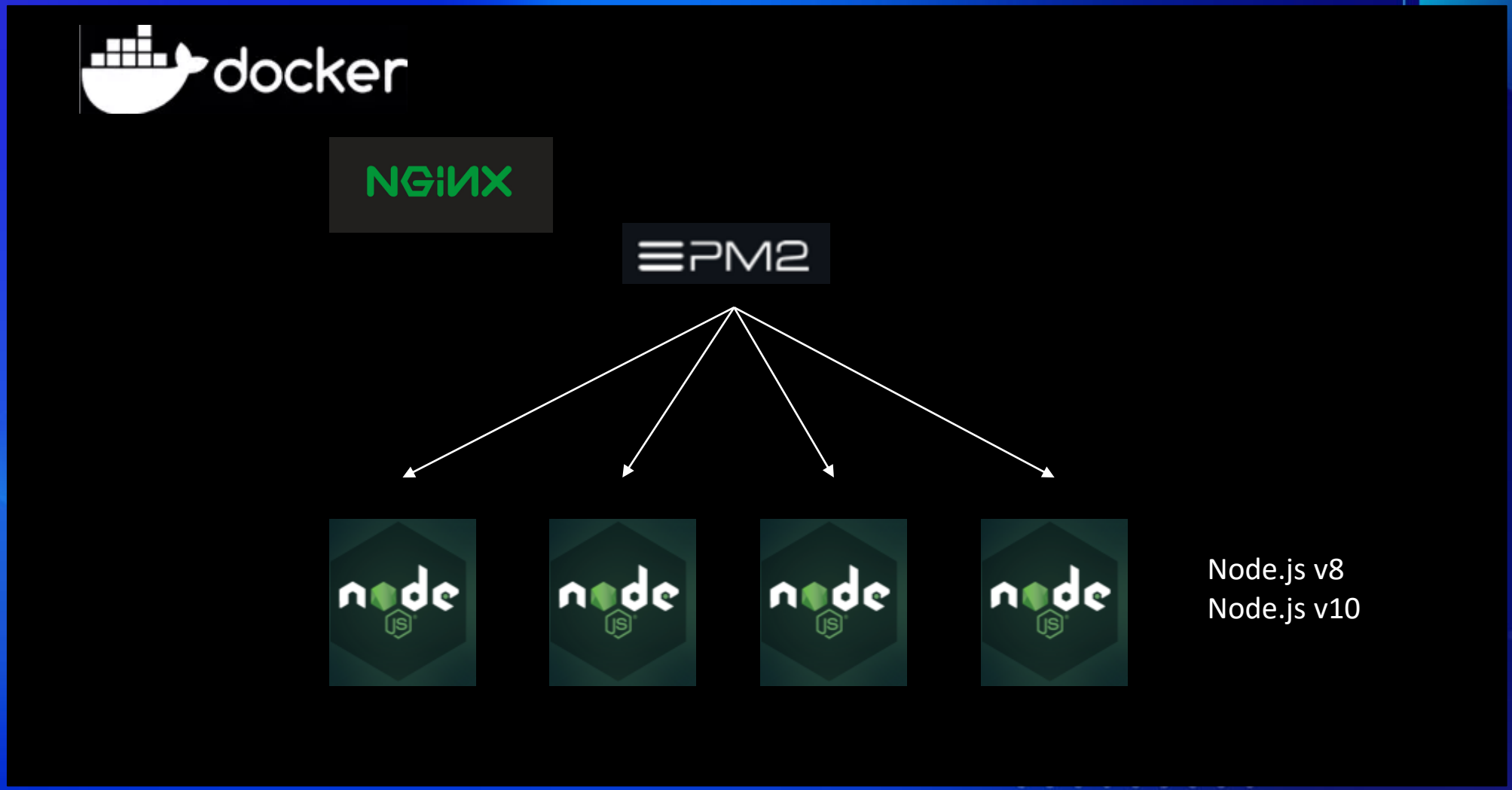
访问异常

内存泄漏

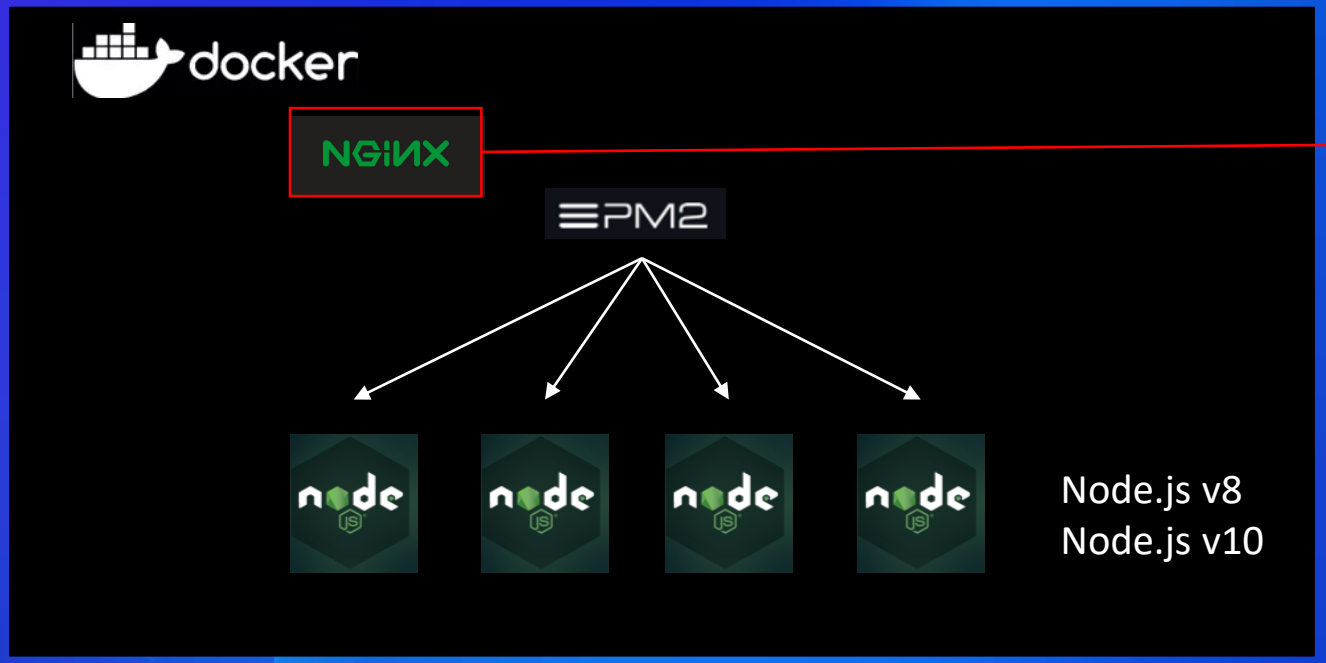


携程Node.js最佳实践

Node.js部署模型

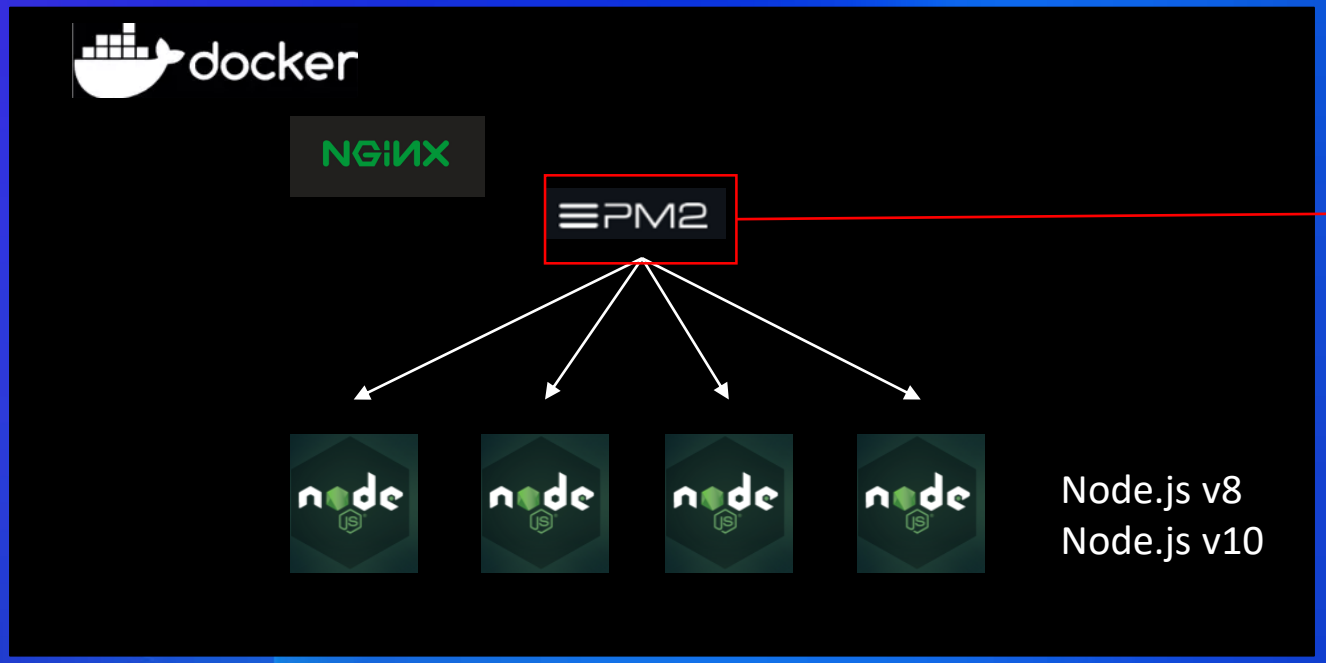


Node.js部署模型



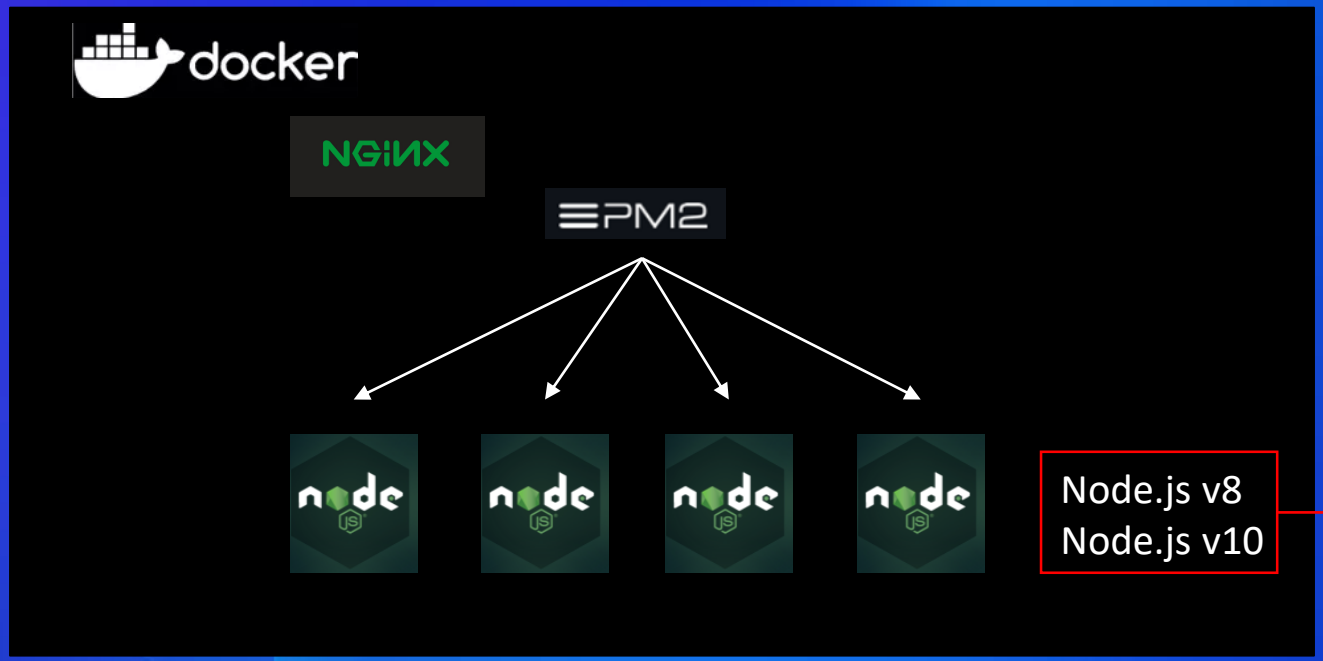
1. access.log标准化
2. 反向代理、重定向、gzip等操作尽量放在nginx上完成，保证稳定性
3. 限流降级机制。当node应用有问题时，nginx不会pending请求，立刻响应请求，防止雪崩。

Node.js部署模型



1. 稳定性：守护进程、内部负载均衡
2. 方便排障：监控指标输出，导入到监控系统

Node.js部署模型



1. 扩展包提供预编译版本
2. 降低开发和维护成本

Node.js核心中间件

存储服务

Ceph Client

调用服务

SOA Service

SOA Client

监控服务

Tracing(CAT)

Logging(Clogging)

Metrics(Dashboard)

公共服务

Configuration

ABTest

DAL

缓存服务

Redis Client

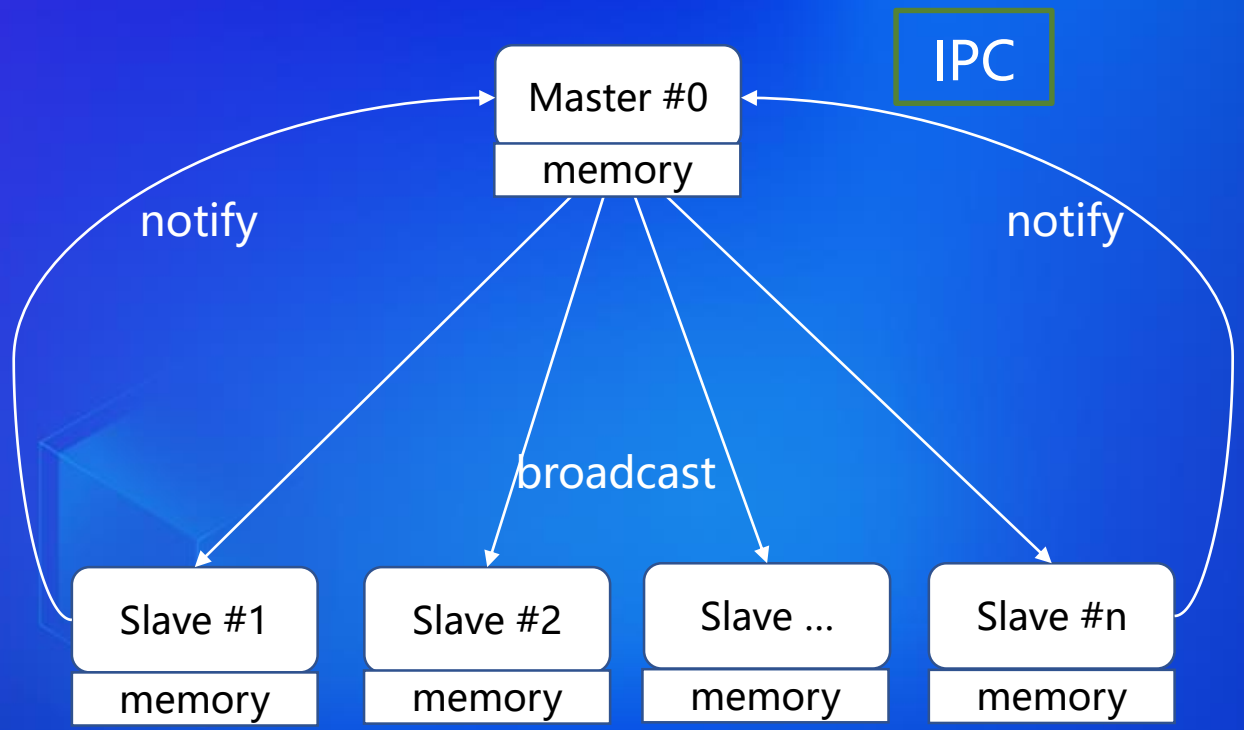
Shared Memory

消息队列

Kafka Producer

问题1 - 多进程通信

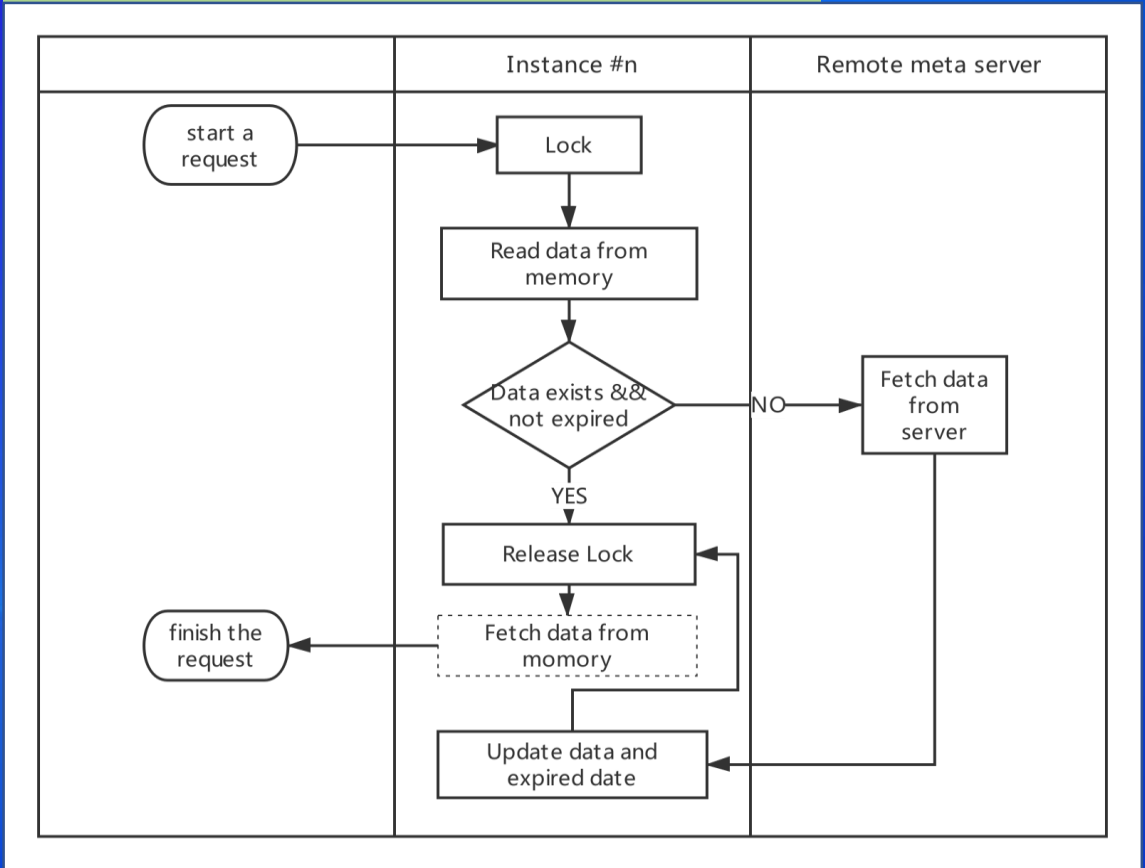
v1.0 进程间的ipc管道



1. 数据量不能太大
2. 数据有延迟
3. master要时刻在线

问题1 - 多进程通信

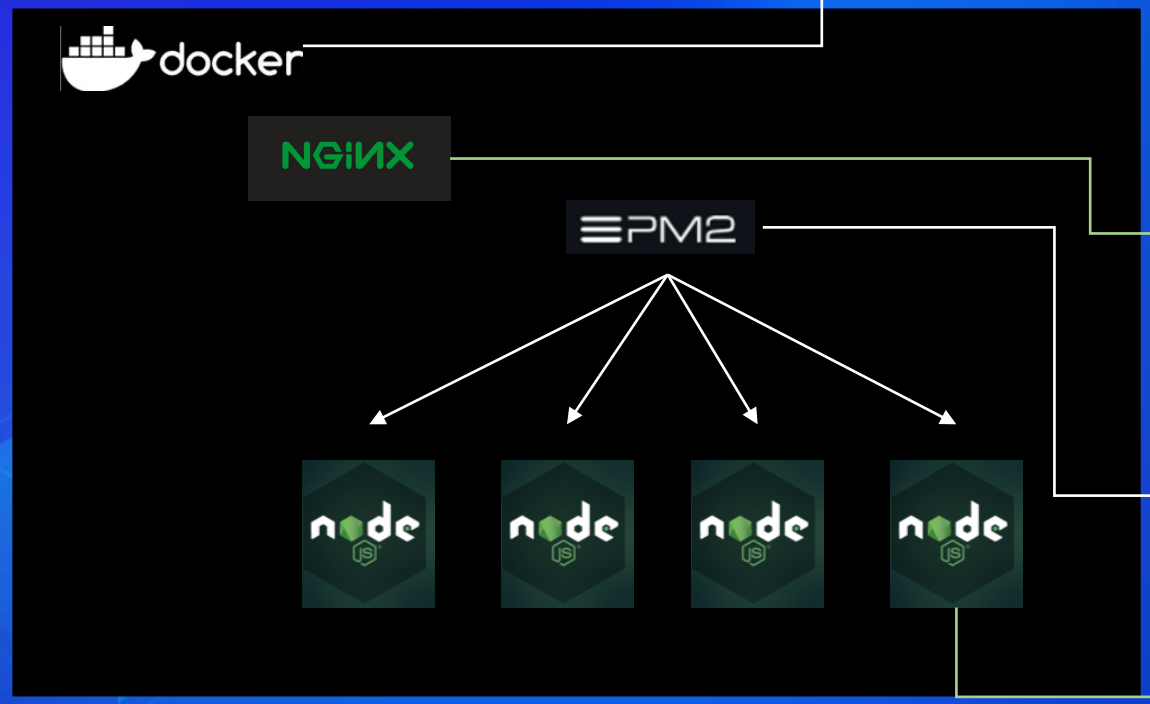
v2.0 共享内存(shared memory)



1. 用flock锁来保证读写不冲突

2. 共享内存提高读写效率

问题2 - 监控什么内容



CPU usage
CPU Throttle time
CPU throttle count
Memory usage
Available file handle

HTTP incoming
HTTP outgoing
Connection reset

Event loop
Restart

Heartbeat
Tracing(request)
Logging(error/warning…)
Metrics(performance data)

问题2 - 监控什么内容

- **CPU usage**

  CPU总的使用率

- **CPU throttle count&time**

  CPU被限制的次数和时间

这两个指标上升一般表示应用有CPU密集型操作，
需检查是否有大量的计算等操作

问题2 - 监控什么内容

HTTP incoming&outgoing

http request的数量变化趋势。

如果有错误响应或者超过了告警的阈值，则会在趋势图中显示

Connection reset

这个指标如果上升，表示应用出现了大量的拒绝请求，例如是服务器的并发数超过了原本的承载量等原因

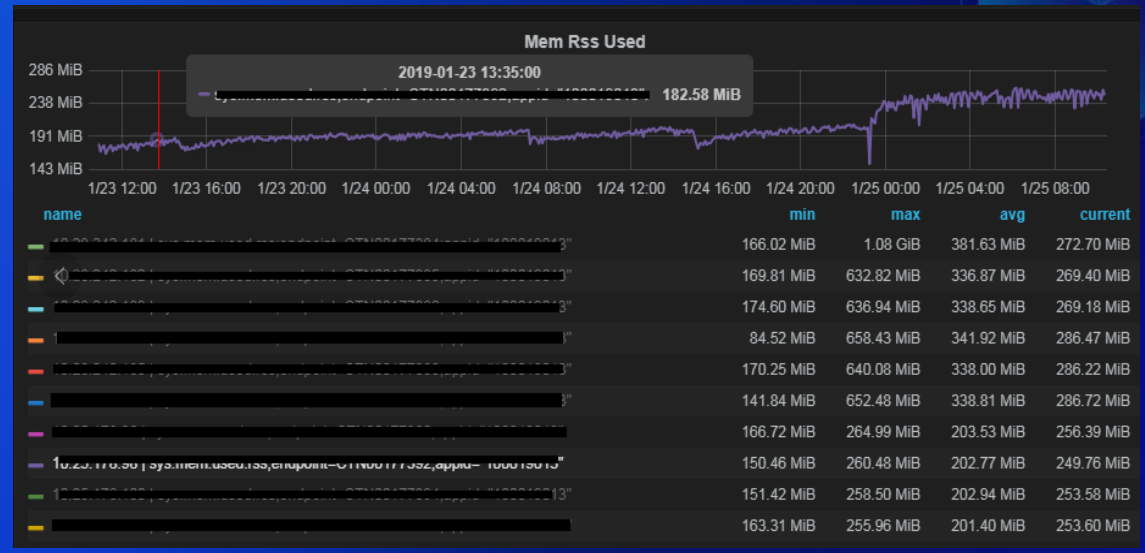
问题2 - 监控什么内容

- **Mem RSS used**

  这个指标上升一般显示应用内存泄漏的问题

- **Available Memory**

  可用物理内存



问题2 - 监控什么内容

Error&Warning

应用逻辑出错，例如JSON数据出错

HTTP请求出错，记录状态码、请求地址、返回内容

应用中使用了不同版本的同一个包

Detailed log

针对应用的逻辑埋点。故障发生时，复盘的辅助手段

问题2 - 监控什么内容

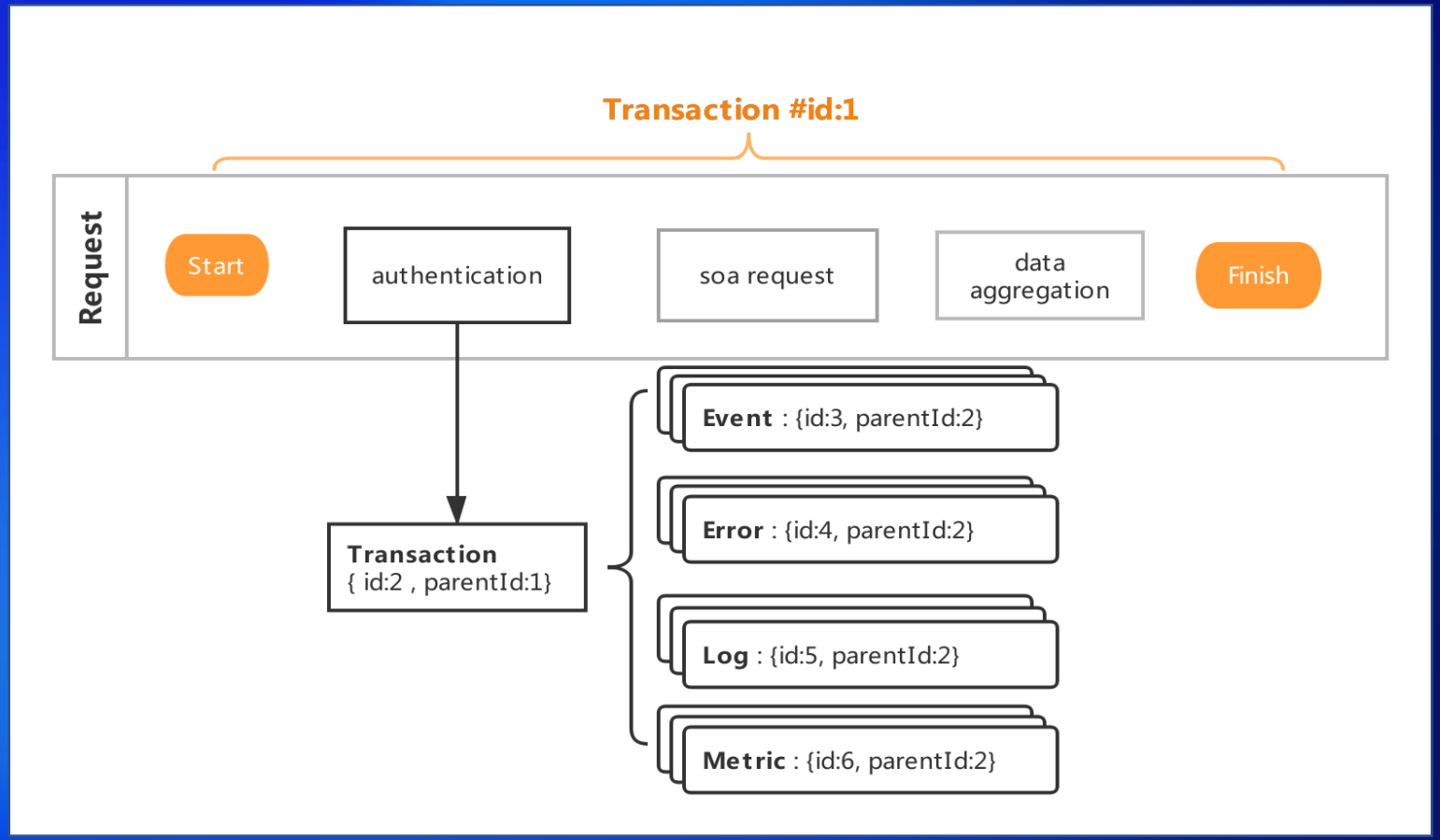
Performance

每个响应的请求耗时 每个Transaction的耗时 跨应用调用的请求耗时

Type	Name 全部	Time 全部 0 ms
SOA2Client	... getexperiment	17:49:45:823 20.711ms _domain=100019040 _threadGroupName=node _ipAddre _messageId
SOA2Client.reqSize	... getexperiment	17:49:45:823 31
【:: 外部调用 ::】		
SOA2Client.resCode	Success	17:49:45:843
SOA2Client.serviceApp	100019083	17:49:45:843
SOA2Client.serviceIP	10.5.98.150	17:49:45:843
SOA2Client.responseStatusAckValue	Success	17:49:45:843

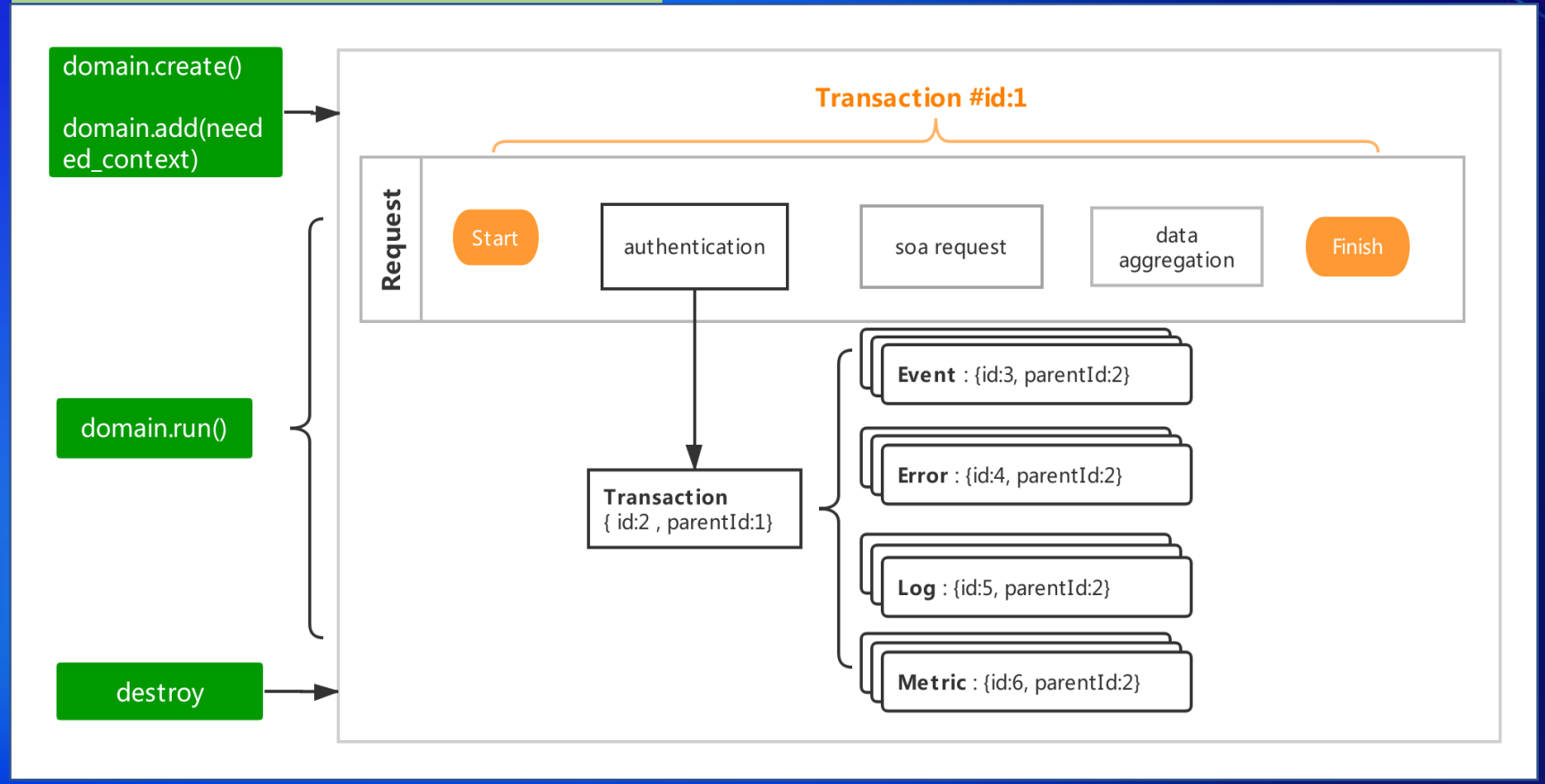
问题3 – 全链路监控

Tracing模型



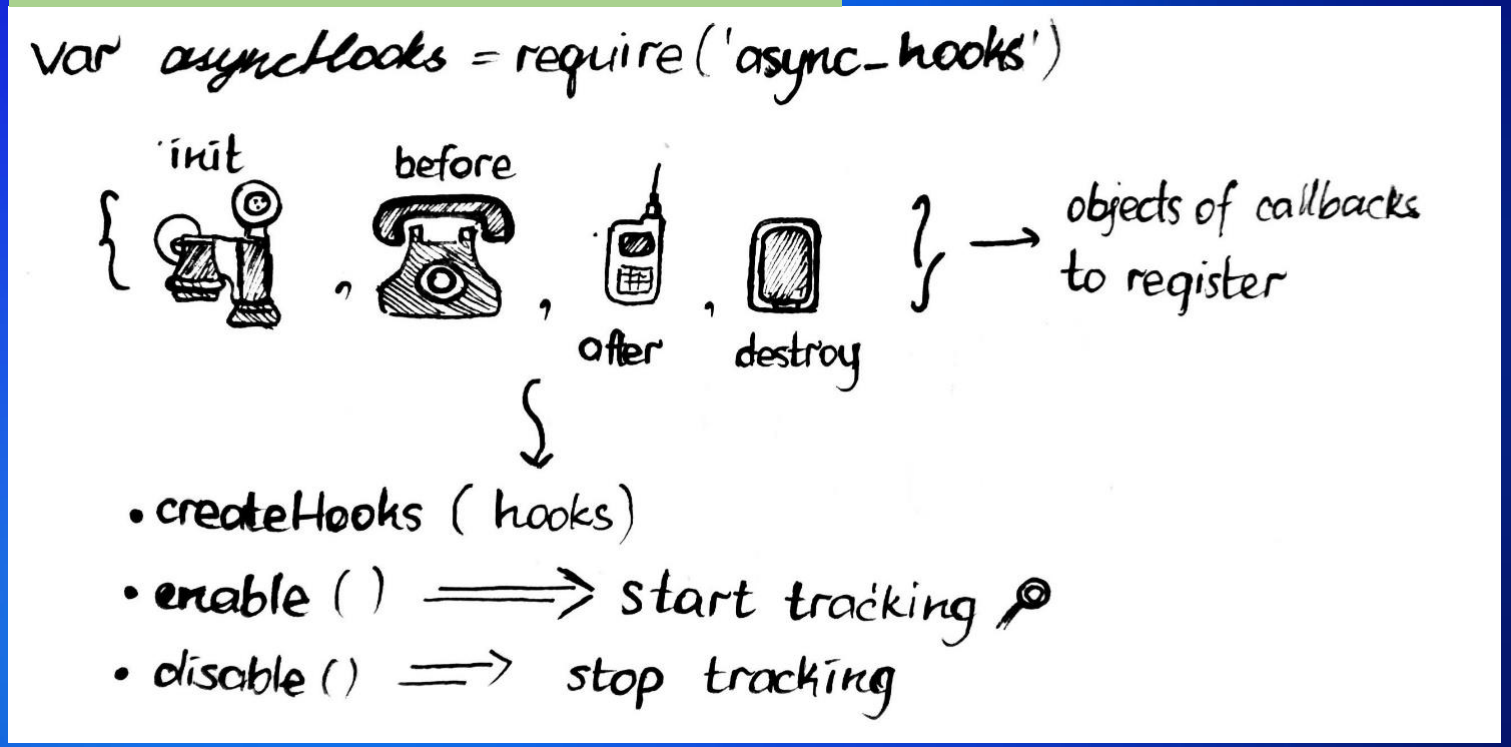
问题3 – 全链路监控

v1.0 use domain module



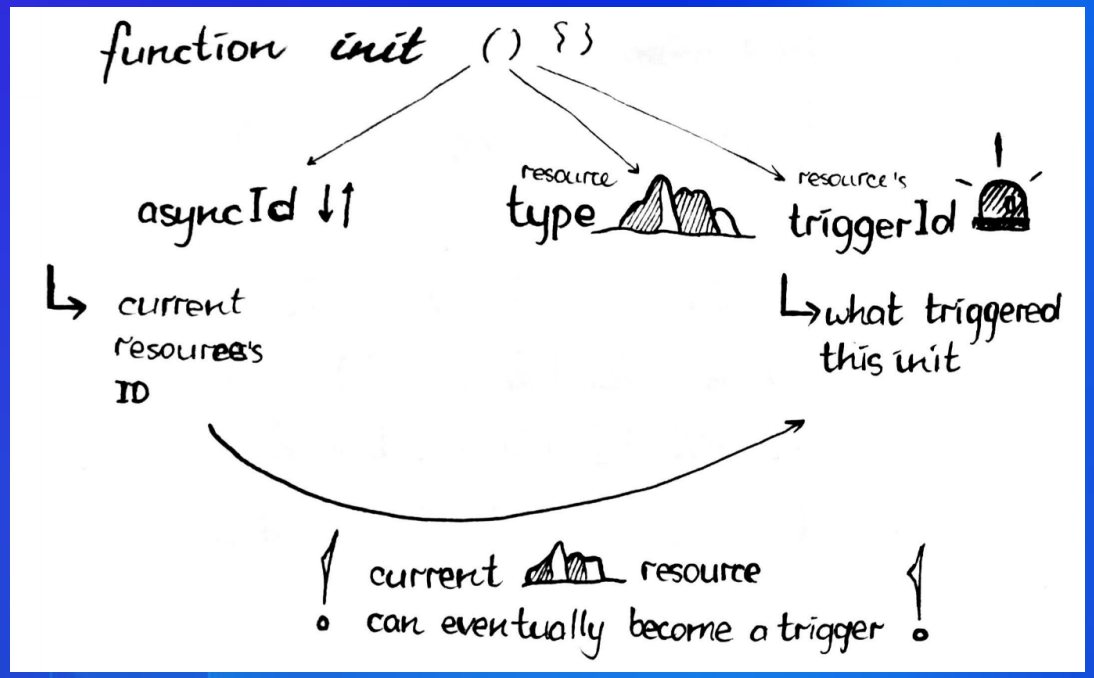
问题3 – 全链路监控

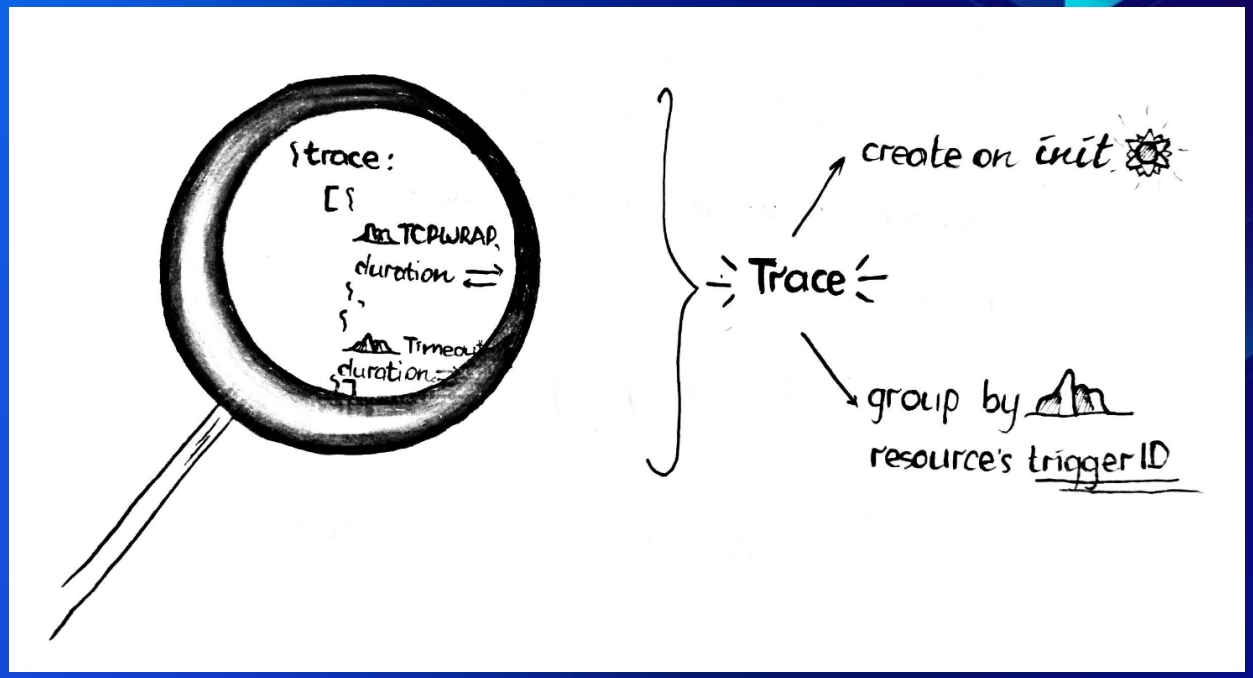
v2.0 use async_hooks module



https://medium.com/the-node-js-collection/async-hooks-in-node-js-illustrated-b7ce1344111f

问题3 – 全链路监控

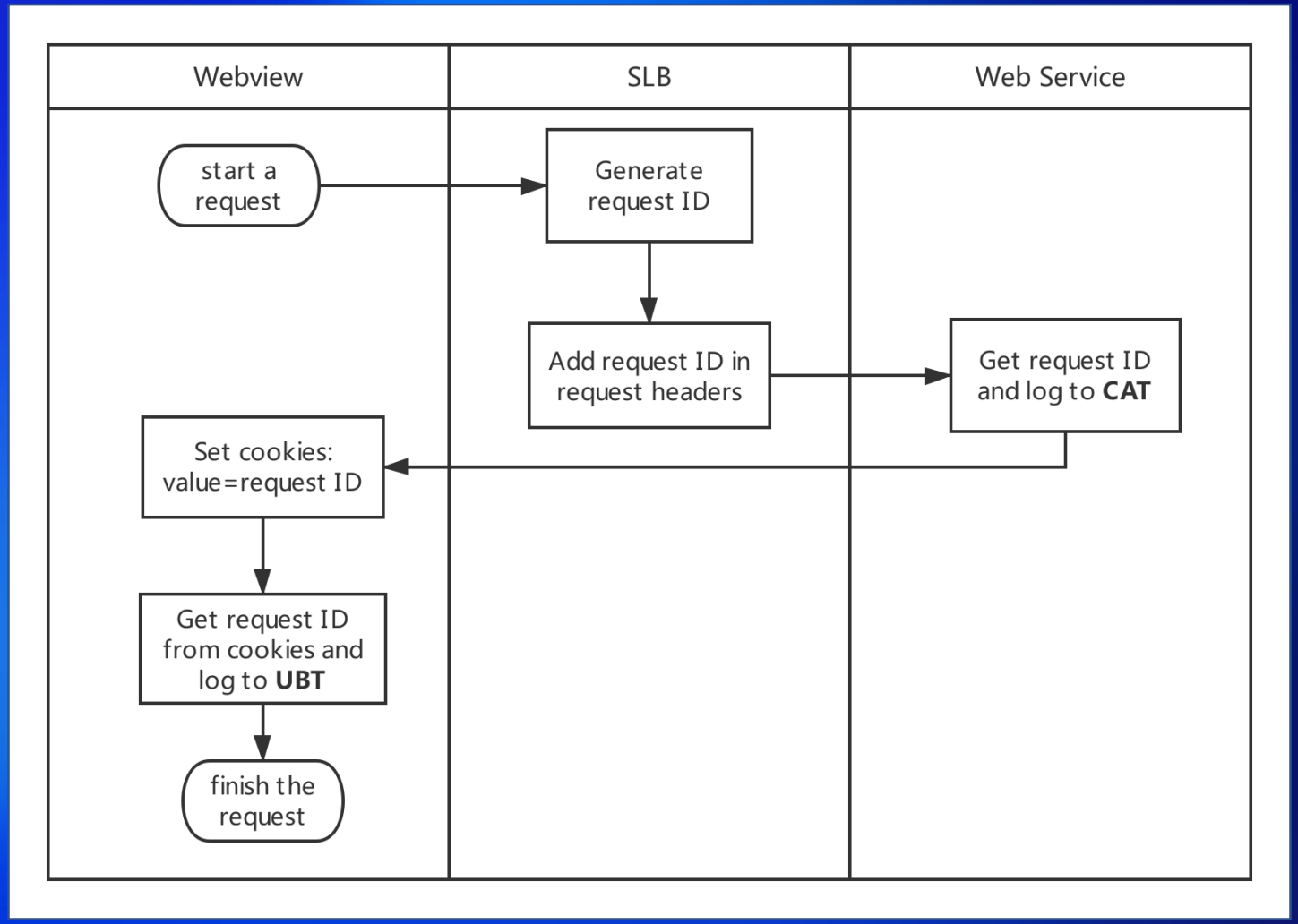




https://medium.com/the-node-js-collection/async-hooks-in-node-js-illustrated-b7ce1344111f

问题3 – 全链路监控

页面请求模型



问题3 – 全链路监控

Pageview | Monitor 29 | Metric 12 | Trace 7 | Action 13 | Error 0

时间	埋点	值	服务号	网络	其他
2019-11-07 16:12:49	o_task_success		TP转TCP专用服务号 / 平台研发中心	4G	(-200)
2019-11-07 16:12:49	o_http_success			4G	/restapi/soa CAT CLOG

问题3 – 全链路监控

Type	Name 全部	Time 全部 0 m
TCP		15:52:08:077 _domain Tree.Begin=15731131280 _threadName=pc 5 _threa
Gateway.	TcpToH5	15:52:08:077
【:: 外部调用 ::】		
AccessService	... getfina on?__gw_os=IOS&__gw_pla	15:52:08:077 CallType=sync
Gateway.	application/json;charset=utf-8	15:52:08:114
sbu	null	15:52:08:114
Cat.Index.Event	clientId	15:52:08:114 12001066310020182318
Cat.Index.Event	messageNumber	15:52:08:114 15731143691550055
【:: 异步调用 ::】		

Recap

治理能力

部署模型

监控模型

核心中间件

开发效率

工程化推进

技术栈差异

排障能力

用户体验

选择合适场景

提高产品质量

本PPT来自2019携程技术峰会

更多信息请关注“携程技术中心”微信公众号~