Node EE方案

Rockerjs在微店的建设与发展

微店 杨力

目录

CONTENT

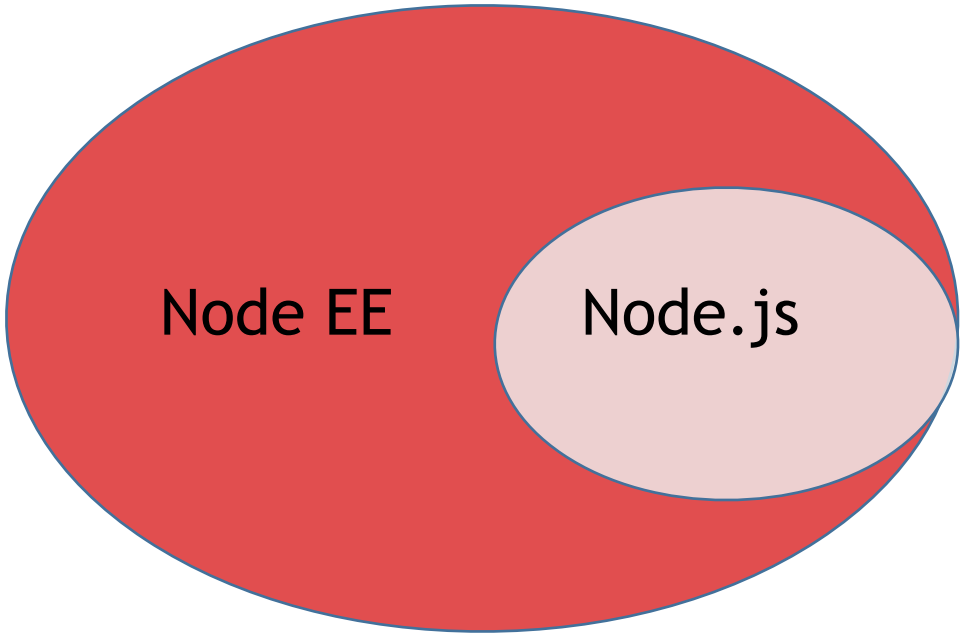
Part 1

Node EE的前世今生

什么是Node EE?

Node Enterprise Edition

Node EE的前世今生

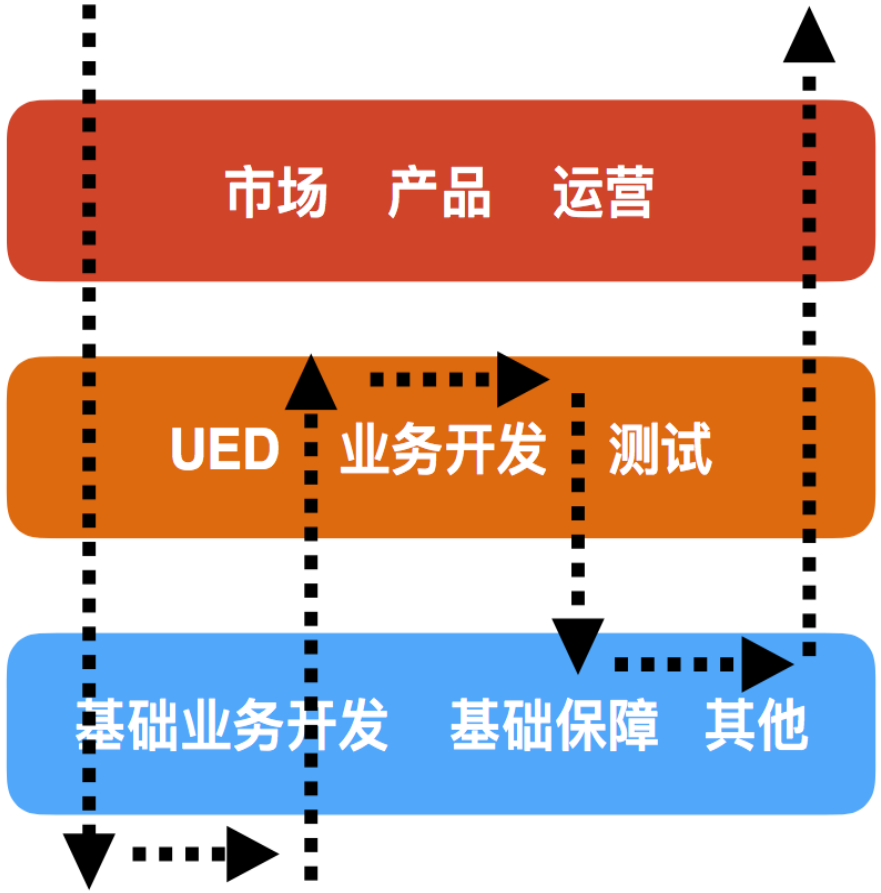


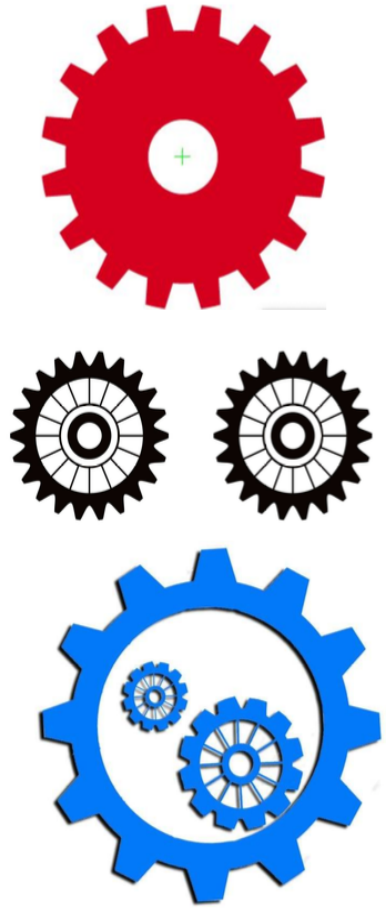
Node EE的前世今生

生产经营过程中的三种角色

Node EE的前世今生

生产经营过程中的效率与时间问题





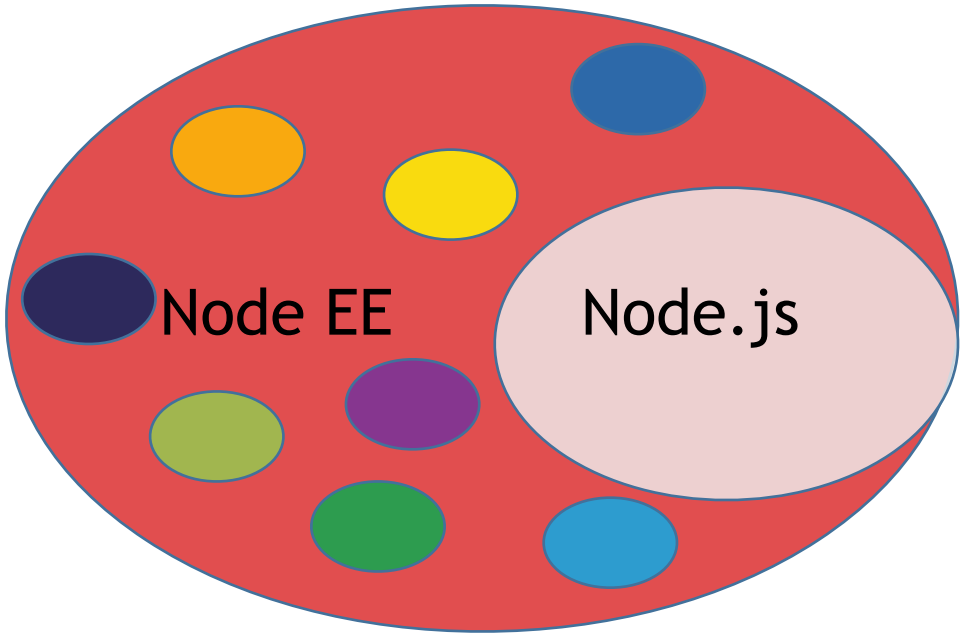
Node EE的前世今生

中台化

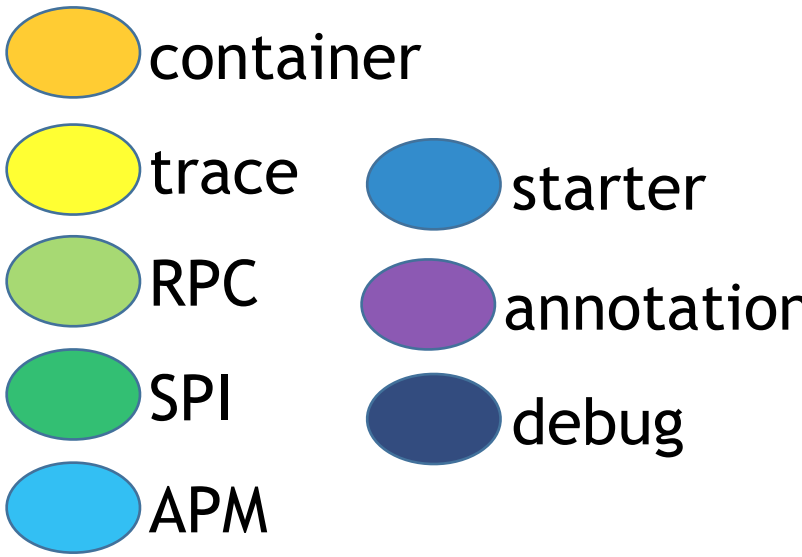
全栈化

Node EE

Node EE的前世今生



Node EE 是面向企业级应用开发场景，满足应用高可维护、可扩展，在无缝接入各级中间件同时，能追踪请求的各层链路、远程调试、在线实时监控与性能分析



什么是Node EE?

企业可以直接拿来“放心”使用的node.js解决方案

什么是Node EE?

“做企业和开发者喜欢的Node EE方案”

— Rockerjs的目标

Part 2

Rockerjs的野蛮生长

Rockerjs是什么：

Rockerjs是微店对Node EE的一种探索和实现。它基于注解提供 IoC 和 AOP 的特性在简化模块依赖的同时让编码二维化，基于此衍生出来了MVC框架、RPC、Node Persistence of XML、ThreadLocal 、trace、SPI、分布式事务及容器监控等中间件，目前微店内部多个平台与外网服务基于此而生

Rockerjs的核心理念：

容器化与IoC

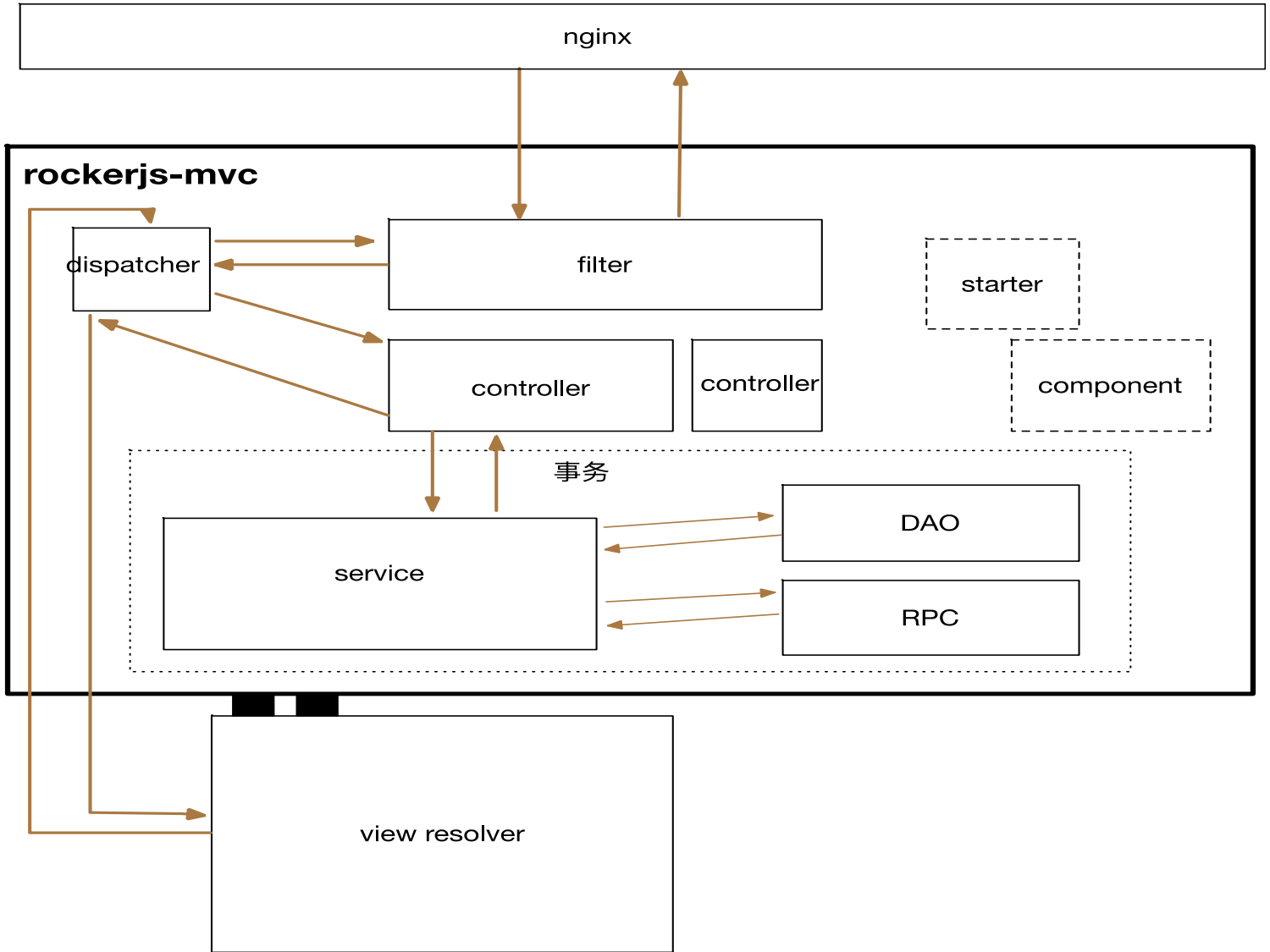
Rockerjs的容器化实现：

Rockerjs-core

Rockerjs的容器化的应用：

Rockerjs-MVC

Rockerjs-MVC



Rockerjs-MVC

- 配置大于一切
- 约定简化编码
- 元编程思想
- DI解耦
- TS强类型约束
- 面向对象、面向接口
- 熟悉的“main”

Rockerjs-MVC demo

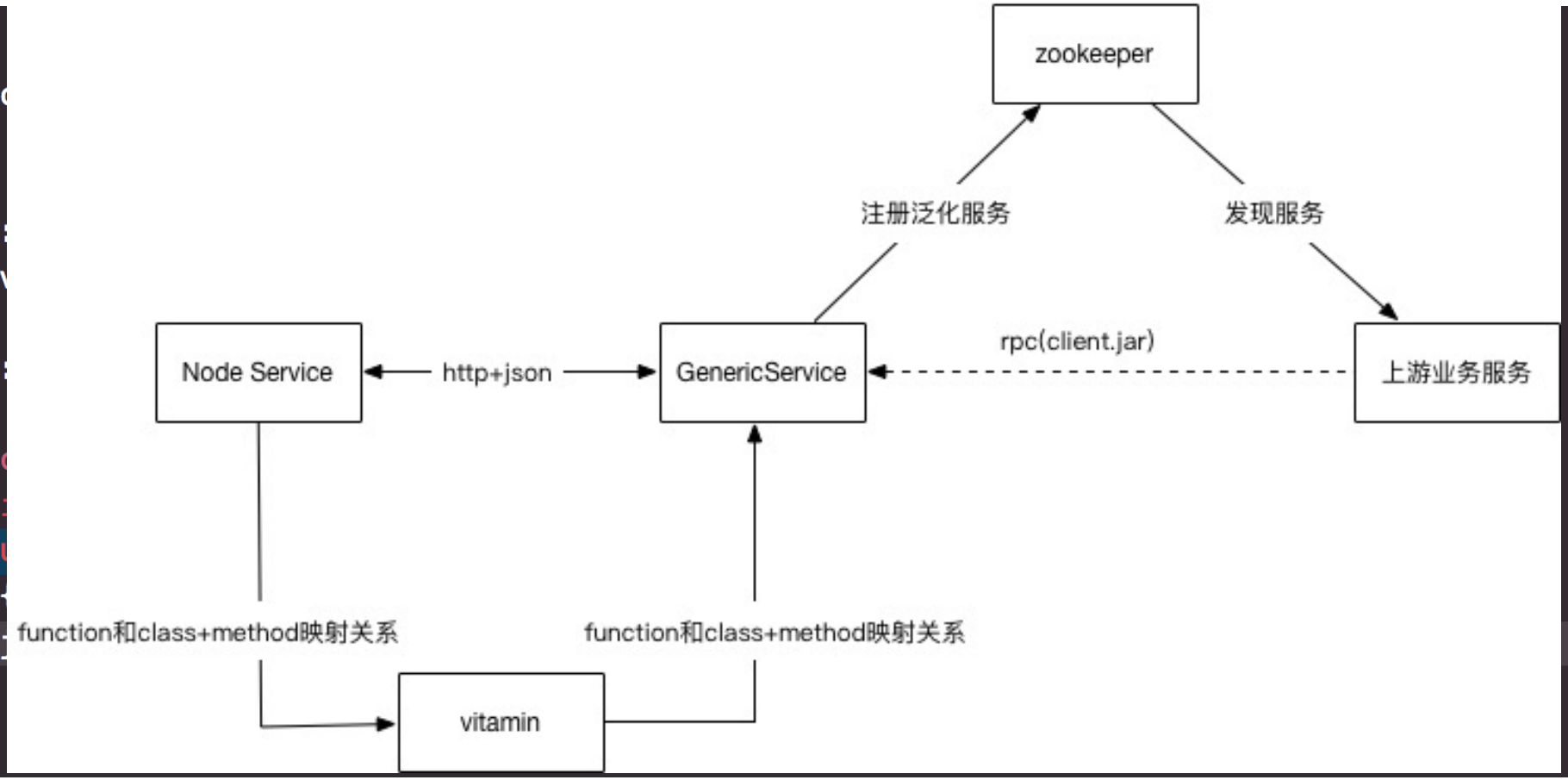
```
import { Application, AbstractApplication } from "@rockerjs/mvc";

@Application
class App extends AbstractApplication{

    public static async main(args: string[]) {
        console.log('main bussiness', args);
    }
}
```

RPC--Dubbo

```
@Dubbo({
    interfa
    method:
    params:
    version:
    nodeServ
})
@Post({ url:
@AutoWrap
public asyn
    const {
    let resu
    return
        pro
    }
}
```




```
context: object) {
```

ORM

基于XML模板的ORM工具

DO

```
import { Mysql } from "@rockerjs/mysql-starter";
let {Column, DOBase, Mapping, Transaction} = Mysql.Module;
import * as moment from 'moment';

export class AppInfo {
    @Column
    public id;

    @Column
    public appid;

    @Column
    public secrete;

    @Column
    public username;

    @Column
    public appname;

    @Column('gmt_create')
    public createTime(createTime) {
        try {
            return moment(createTime).format('YYYY-MM-DD HH:mm:ss')
        } catch (e) {
            return createTime;
        }
    }
}
```

DAO

```
import { Mysql } from "@rockerjs/mysql-starter";
const { DOBase, Mapping } = Mysql.Module; // 使用 typeof 强转类型
import { AppInfo } from "../dto/App_info";

export class AppInfoDao extends DOBase {
    public async add({appid,secrete,username,appname}) {
        const fId = await this.exe('appInfo:add',{
            appid,secrete,username,appname
        });
        return fId;
    }

    @Mapping(AppInfo)
    public async queryAll() {
        const ary = await this.exe('appInfo:queryAll', {});
        return ary;
    }

    @Mapping(AppInfo)
    public async queryByName(name) {
        const ary = await this.exe('appInfo:queryByName', {name});
        return ary;
    }

    @Mapping(AppInfo)
    public async queryByNameAndAppname(name,appname) {
        const ary = await this.exe('appInfo:queryByNameAndAppname', {
            username: name,
            appname
        });
        return ary;
    }
```

```
<select id="queryByNameAndAppname" resultType="../dto/App_Info">
    select * from app_info where username=#{username} and appname=#{appname}
</select>

<insert id="add">
    insert into app_info ( appid,secrete,username,appname )
    <trim prefix="values (" suffix=")" suffixOverrides=",">
        #{appid},#{secrete},#{username},#{appname}
    </trim>
</insert>

<update id="editBatchWithCondition">
    update app_info
    <set>
        <foreach collection="Object.keys(data)" item="key" index="index" >
            <if test="index <= (Object.keys(data).length -2)">
                ${key} = #{data[key]},
            </if>
            <if test="index > (Object.keys(data).length -2)">
                ${key} = #{data[key]}
            </if>
        </foreach>
    </set>
    <where>
        <foreach collection="Object.keys(info)" item="key" index="j" >
            <if test="j <= (Object.keys(info).length -2)">
                ${key} = #{info[key]} and
            </if>
            <if test="j > (Object.keys(info).length -2)">
                ${key} = #{info[key]}
            </if>
        </foreach>
    </where>
</update>
```

Rockerjs的野蛮生长

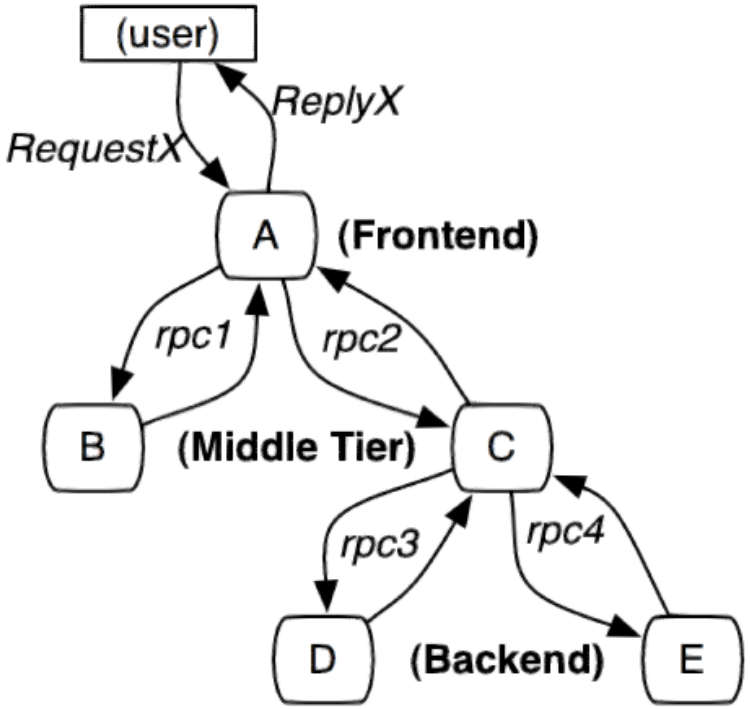
trace

Vtrace 入口大盘 链路大盘 异常链路查询 业务自定义链路查询 服务分析 180e0000016a0a122c330a20c2c85ab8

0.1	thor-server-inner	- =>	HTTP	OK	/detail/innerGetItemInfo/1.0	5150	36/36ms	200
0.1.1	detail-h5	=>	DUBBO	OK	com.vdian.thor.common.detailService.inner	5279	34/34ms	OK
0.1.1.1	vitemcenter-L0	=>	DUBBO	OK	com.vdian.vitemcenter.client.ItemCoreRead	5225	4/3ms	OK
0.1.1.1._.1.1	vitemcenter-L0	=>	REDIS	OK	wdsvr-item_ALL_mget	0	1/1ms	ok
0.1.1.1._.2.1	vitemcenter-L0	=>	REDIS	OK	wdsvr-item_ALL_mget	0	1/1ms	ok
0.1.1.1._.3.1	vitemcenter-L0	=>	REDIS	OK	wdsvr-item_ALL_mget	0	1/1ms	ok
0.1.1.1._.4.1	stockcenter	=>	DUBBO	OK	com.vdian.stockcenter.client.service.sku.Sk	1200	1/1ms	OK
0.1.1.2	detail-h5	=>	DUBBO	OK	com.weidian.vc.jupiter.vcserver.RiskCheckS	0	2/2ms	OK
0.1.1.3	wd-shop	=>	DUBBO	OK	com.weidian.wdp.wdshop.api.ShopService.	3694	5/4ms	OK
0.1.1.4	vexpressfee	=>	DUBBO	OK	com.vdian.vexpressfee.client.service.ItemE	233	2/2ms	OK
0.1.1.5	vmpcoupon	=>	DUBBO	OK	com.koudai.vmp.service.BuyerCouponReac	192	6/5ms	OK
0.1.1.6	intranet-business-jupiter3	=>	DUBBO	OK	com.weidian.vc.jupiter.api.RiskCheckServic	186	4/3ms	OK
0.1.1.7	vmp	=>	DUBBO	OK	com.vdian.vmp.client.service.detail.DetailPr	1401	11/9ms	OK
0.1.1.8	wd-collect	=>	DUBBO	OK	com.weidian.wdp.collect.api.item.CollectIte	311	2/0ms	OK
0.1.1.9	wd-collect	=>	DUBBO	OK	com.weidian.wdp.collect.api.item.CollectIte	306	2/0ms	OK
0.1.1.9.1	wd-collect	=>	REDIS	OK	wd_collect_ALL_hmget	0	0/0ms	ok
0.1.1.10	wd-collect	=>	DUBBO	OK	com.weidian.wdp.collect.api.shop.CollectSI	228	2/1ms	OK
0.1.1.11	wd-collect	=>	DUBBO	OK	com.weidian.wdp.collect.api.shop.CollectSI	213	1/1ms	OK
0.1.1.12	internet-web-limitcenter	=>	DUBBO	OK	com.vdian.limitcenter.client.service.arealimi	229	1/1ms	OK
0.1.1.13	detail-h5	=>	REDIS	OK	detail-h5_ALL_get	0	0/0ms	ok
0.1.1.14	detail-h5	=>	DUBBO	OK	com.weidian.media.remote.api.media.Medi	0	2/2ms	OK
0.1.1.15	wd-shop	=>	DUBBO	OK	com.weidian.wdp.wdshop.api.ShopUdcApi	137	1/1ms	OK
0.1.1.16	udc-core	=>	DUBBO	OK	com.vdian.udc.core.api.UserInfoExtendApi.	139	2/1ms	OK
0.1.1.17	pay-cashier	=>	DUBBO	OK	com.weidian.cashier.client.facade.PrefixPay	297	13/12ms	OK
0.2	thor-server-inner	- =>	HTTP	OK	/detail/innerGetLoginUserData/1.0	1141	16/16ms	200
0.3	thor-server	- =>	HTTP	OK	/poseidon/exhibit.spaceJson/1.0	1811	5/5ms	200

共**118**条调用记录,总耗时**0**+ms

trace



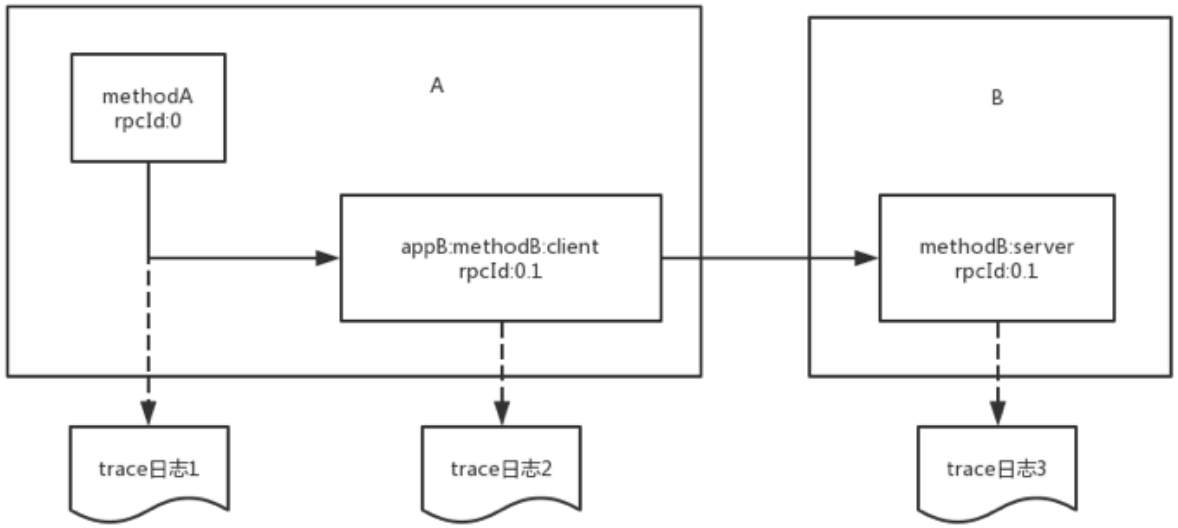
核心：

调用链，每次请求都生成一个全局的ID（TraceID），通过该ID将不同系统，位于不同机器上的日志（Trace日志）串在一起，重组成调用链，使其价值达到1+1>2的效果

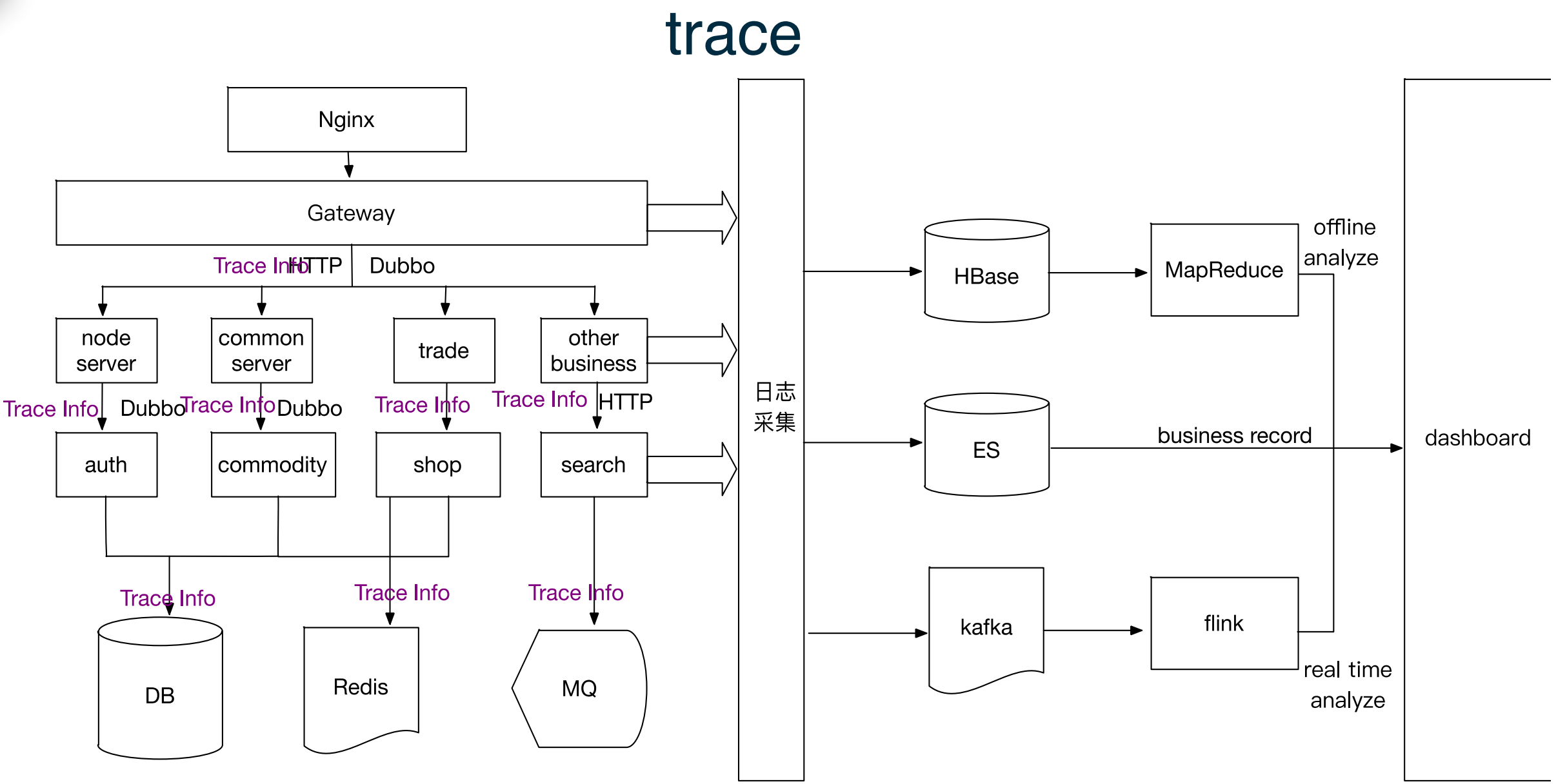
RpcID代表了Trace链路上的一次调用，它用来标记调用链的层级，是追踪树的扁平化体现

trace

RpcID代表了Trace链路上的一次调用，它用来标记调用链的层级，是追踪树的扁平化体现



Rockerjs的野蛮生长



埋点和日志生成

- 链通过中间件创建调用上下文，生成埋点
  - ✓ traceID，rpcId，isSample（是否采用），等等
- 调用上下文放到本地ThreadLoacal，对应用透明
- 调用上下文在整个链路透传：
  - ✓ dubbo通过dubbo的attachment机制
  - ✓ common－http通过header透传

ThreadLocal in Node.js === AsyncContext bound

实现

- Zone.js (Node.js与ThreadLocal)
- Async hooks

容器级监控与分析

NPS看板

申请应用

vstudio

yangli

2018-05-17 16:05:53

诊断 监控

chenjunbin

2018-05-30 14:18:32

诊断 监控

zhouwei

2018-07-04 17:27:26

诊断 监控

yangyuliang

2018-08-15 16:26:36

诊断 监控

shuaihaijun

2018-10-22 14:51:01

诊断 监控

gaoxiaoqian

2018-12-28 14:53:31

诊断 监控

kangzhe

2018-12-28 15:07:57

诊断 监控

chenguocheng

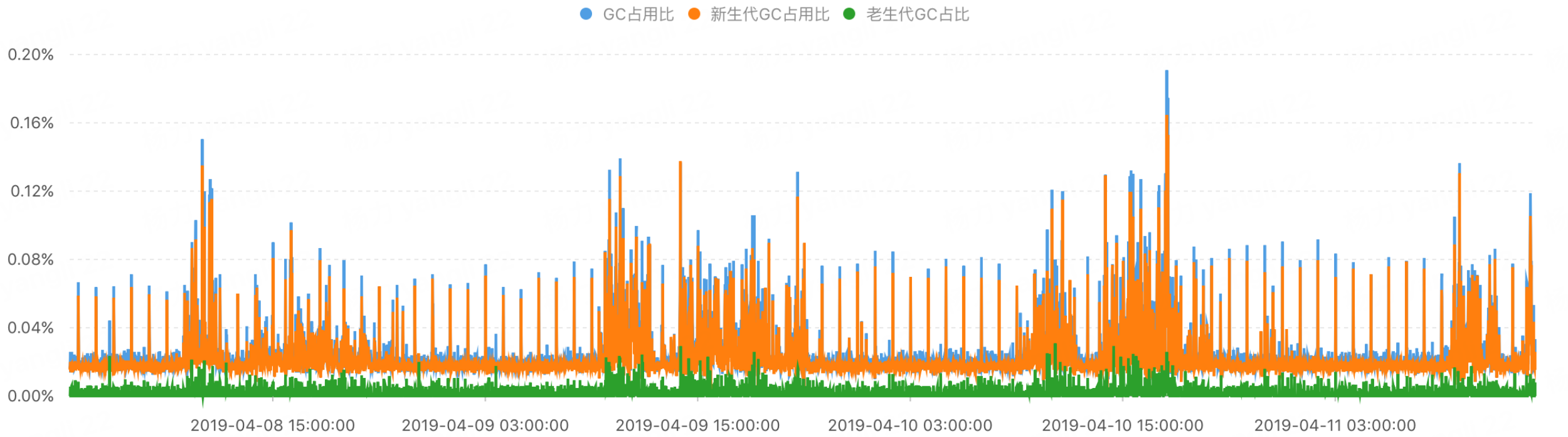
2018-05-31 17:51:56

诊断 监控

容器级监控与分析

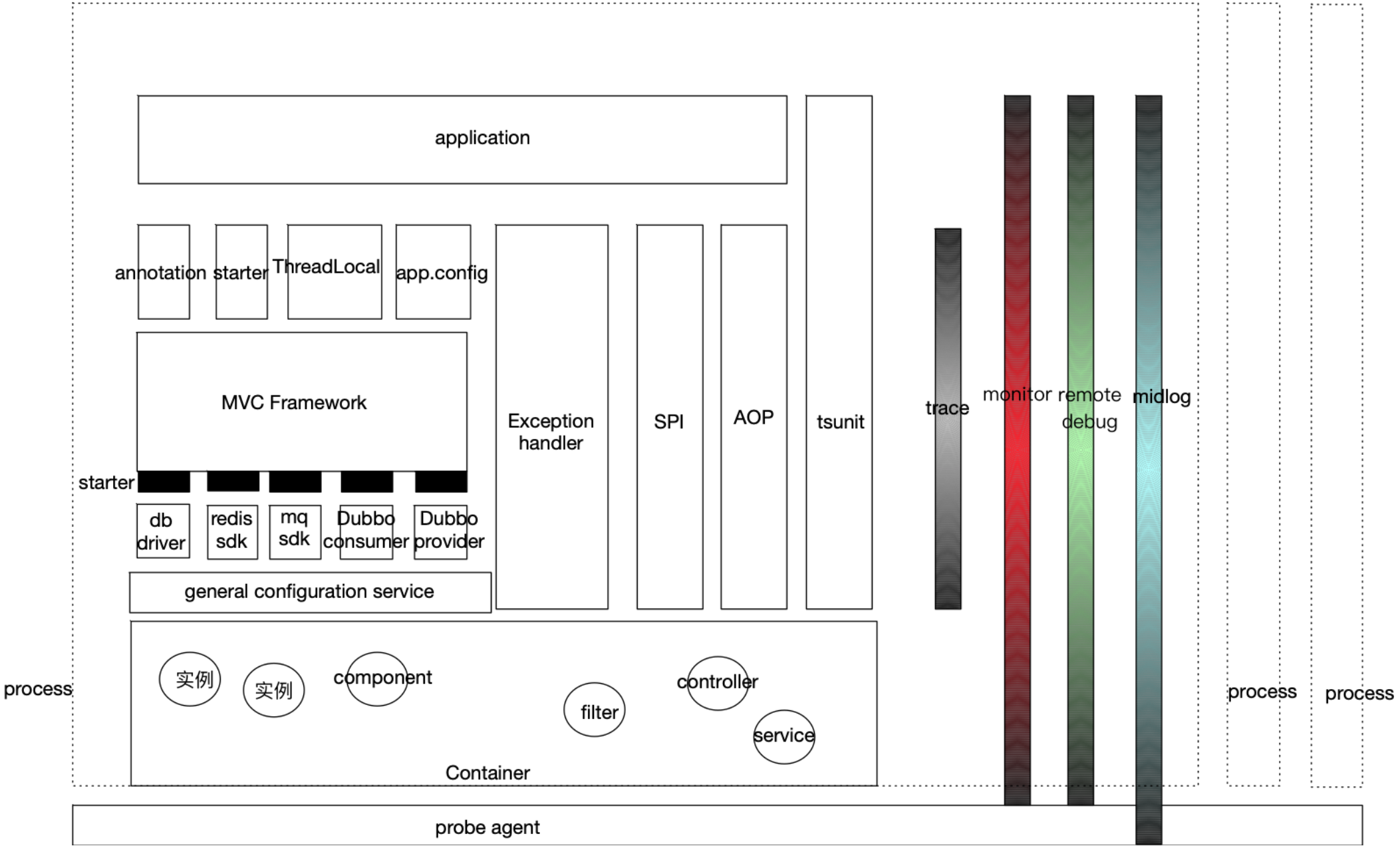
node通用数据

nodeGC



Rockerjs的野蛮生长

Node EE


application
annotation
starter
ThreadLocal
app.config
MVC Framework
starter
db driver
redis sdk
mq sdk
Dubbo consumer
Dubbo provider
general configuration service
Exception handler
SPI
AOP
tsunit
trace
monitor
remote debug
midlog
process
实例
实例
component
filter
controller
service
Container
process
process
probe agent

Part 3

未来的挑战

未来的挑战

- 生态建设
  - WebSocket
  - xxxStarter
  - Cli
  - Plugins(graphql、restful)
  - …
- 框架周边建设
  - Rockerjs/mvc
  - Rockerjs/dao
  - Rockerjs/tracer
  - Rockerjs/viewer
  - Rockerjs/Tcc
- 文档

JOIN US

感谢观看