从 0 构建大规模 nodejs 应用

来自京东无线页面渲染系统的实践

分享人：曾通

目录

背景介绍

- 一切都从需求开始

对象：活动及频道页面

特点：

数量大

变化快

流量大

无线端

对策：

可视化平台

组内闭环

扛

首屏优先

背景介绍

- 这系统是干什么的？

目标：频道页、活动页的渲染

支持渠道：APP，M站，微信手Q

背景介绍


818手机超级品类日，领券最高立减888元！
登录
爱/生/活
玩3C
iPhone 超低价
更多优惠GO！
京东家电
领券最高减1500
815周年庆
京东超市
每满99减50
爱家
厨具满200-60
199-188清仓
爱宝宝
爆款低至5折
爱美丽
美妆0元试用
爱吃
低至9.9包邮
爱穿搭
七夕免息购
三星S8芭比粉
预订赠桃花酒
享/品/质
全球尖货
进口品质
买3免1
京东精选
精选优质生活
品质推荐
推荐
分类
发现
购物车
我的


京东超市
轻松逛超市
京东超市
零食
满99减50
超市快讯
外出
当美酒邂逅佐酒小食，像爱情般美妙
超市抢购
HOT
超市尖货轮番抢
8月18日 12:00
8月18日 16:00
8月18日 20:00
8月19日 00:00
精选好货轮番抢
距结束 02时25分28秒
必抢


爱吃
每日必抢
N元N件
优选促销
精选大牌
搜索您想要的商品
热搜
牛奶
零食
坚果
美食排行榜
面包
两件七折
四川美食季
一起任性抢
冰爽美食节
两件七折
快速查看
每日必抢
更多
8月18日 12:00
8月18日 16:00
8月18日 20:00
8月19日 00:00
8月19日 08:00
正在抢购，先下单先得哦
距结束 02时23分56秒

背景介绍


8.18手机超级品类日
8.10-8.13 摄影主题期
8.14-8.17 手游主题期
8.18 超级品类日
8.19-8.22 冠军专场
手机
超级品类日
6期白条免息
优惠券
iPhone专区
爆款力荐
新品放量
更多优惠券 GO
0点 2点 10点 14点 18点 20点限量发放，限部分京东自营手机使用
¥888 满5818元使用
限时领取
¥600 满4818元使用
限时领取
¥300 满2818元使用
¥180 满1818元使用
¥80 满818元使用


京东超市活动
精选推荐
食品饮料
纸品清洁
京东生鲜
居家生活
中外名酒
挚爱之选
满满的自然力量
正品低价立即购
Nutrilon
Aptamil
【京东超市】爱他美 Aptamil 幼儿配方...
¥1110
【京东超市】爱他美 白金版Aptamil 儿...
¥344
【京东超市】诺优能 Nutrilon 儿童配方...
¥150
品质牛奶
缤纷好心情 好奶享不停
低至第2件半价

背景介绍

- 现在系统运行得怎么样呢？

在线频道/活动页面数量：5000+

日PV单位：亿

峰值单位：万/秒

集群CPU核数单位：千

框架选型

为什么使用框架？

- 效率
- 安全
- 成本

框架选型

express

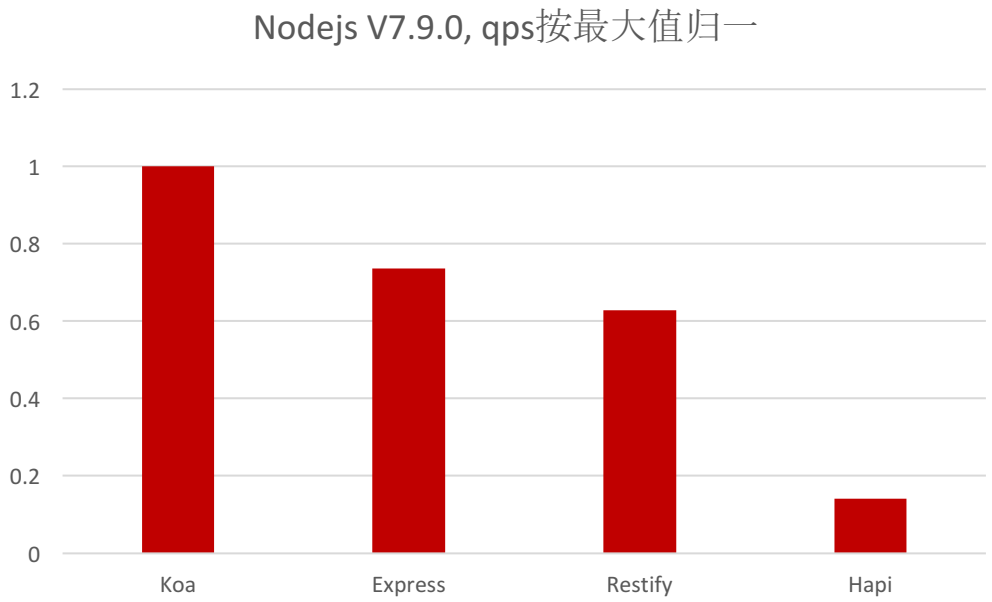
koajs

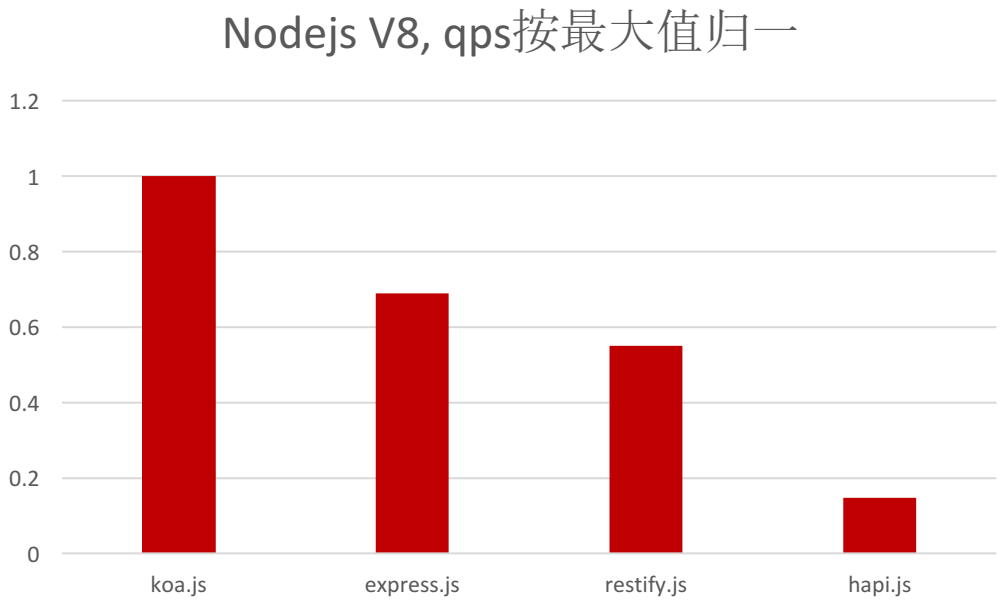
restify

hapi

egg.js

框架选型





Ref: https://github.com/hbakhtiyor/node-frameworks-benchmark

框架选型

TIPS:

流行程度/社区规模

团队水平

有人能摆平

开发效率

运行效率

文档

升级维护成本

选一个先跑起来

适合自己的才是最好的！

基础设施

- 私有npm库

```
var toString = {}.toString;

module.exports = Array.isArray || function (arr) {
  return toString.call(arr) == '[object Array]';
};
```

```
module.exports = leftpad;
function leftpad (str, len, ch) {
  str = String(str);
  var i = -1;
  if (!ch && ch !== 0) ch = ' ';
  len = len - str.length;
  while (++i < len) {
    str = ch + str;
  }
  return str;
}
```

外部：

- 网络环境
- 依赖安全
- 安装速度

内部：

- 网络权限
- 内部模块
- 稳定性

基础设施

jnpm: Private npm registry for npm.m.jd.com

术语

- jnpm 指我们部署的私有 npm 库，即 npm.m.jd.com 背后的服务，下文会称其为 jnpm 库；
- jnpm 同时也指代 jnpm 命令，是用于私有库相关的操作的客户端，下文会称其为 jnpm 客户端；
- @Scope，指私有包所属的 Scope，具体请见文档 npm-scope，下文以 @jd 做为示例；

Scope

- `jnpm 库` 默认的公共 `Scope` 为 `@jd`
- 你可以把对公司内部开放的包发布在 `@jd` 下
- 如果你需要专有的 `Scope` 请联系无线运维xxx（xxx@jd.com）同学
- @jmfe 为无线前端开发组使用；

安全提示

在发布私有包时请务必谨慎，不要把私有包发布到外部 npm 库中：

- 确保你的私有包名字带 `@jd` 前缀，例如 test 项目的 package.js 中，`"name"` 字段的值为 `@jd/test`；
- 推荐使用 jnpm 工具来替换 npm 命令，特别是发布私有包时，请使用 jnpm publish；

外部包

- 我们库中的外部包: registry.m.jd.com, 同步自 https://registry.npm.taobao.org

基础设施

- Cnpm部署改造

单机 

主备部署

共享数据库

备机关闭自动同步

Rsync同步文件

基础设施

TIPS:

基础设施

- 内部系统打通

  jmfe-node-jss 文件存储服务

  jmfe-node-logbook 日志系统

  jmfe-node-redis redis缓存

  ……

  jmfe-node-ump 统一监控系统

应用部署

- 怎么发到服务器上呢？

手动部署

1. 上传代码到服务器
2. 修改配置
3. 安装npm包
4. 重启服务

手动部署问题

1. 这100台机器咋办呢?
2. 新代码有问题，快回滚！
3. 手一抖……
4. 运维：“XXX，把YYY搞挂了！”

应用部署

- 怎么发到服务器上呢？

自动部署

1. 权限控制
2. 规范流程
3. 策略灵活
4. 查询状态
5. 自动切流
6. 多机并发
7. 智能检测
8. 快速回滚

一次自动部署的主要环节：

1. 文件拉取
2. 环境设置
3. 执行发布前置脚本
4. 文件分发
5. 执行发布后置脚本

应用部署

自动部署

1. Web操作
2. Git 管理代码
3. Rsync同步文件

应用部署

- 怎么发到服务器上呢？

IP	状态	结果
10.187.210.249	成功	NODE_ENV=production NODE_PORT=[redacted] APP_INSTANCE_NUM=[redacted] APP_NAME=[redacted] [redacted]: stop Nginx ... [redacted] Nginx is not running Please waiting pro.m.jd.com to stop ... stop pm2 application [redacted] ... stop pm2 application [redacted] successed ... installing npm dependencies ... Starting pro.m.jd.com ... [PM2] [[redacted]]([redacted]) ✓ [PM2] [[redacted]]([redacted]) ✓ [redacted]: start Nginx ... [redacted] Nginx start successed, execution return message: pm2 start nodejs successed. PID=[redacted]

应用部署

Tips:

分组部署

灰度发布

控制发布次数

发布完检测

周五不发布

发布要谨慎!

监控与告警

背景：

- 流量是电商业务的宝贵资源
- 巨大的流量
- 应用的任何故障都会造成直接经济损失

要求：

- 掌握服务器状态
- 及时发送告警
- 数据可视化
- 故障定位
- A/B Test

监控与告警

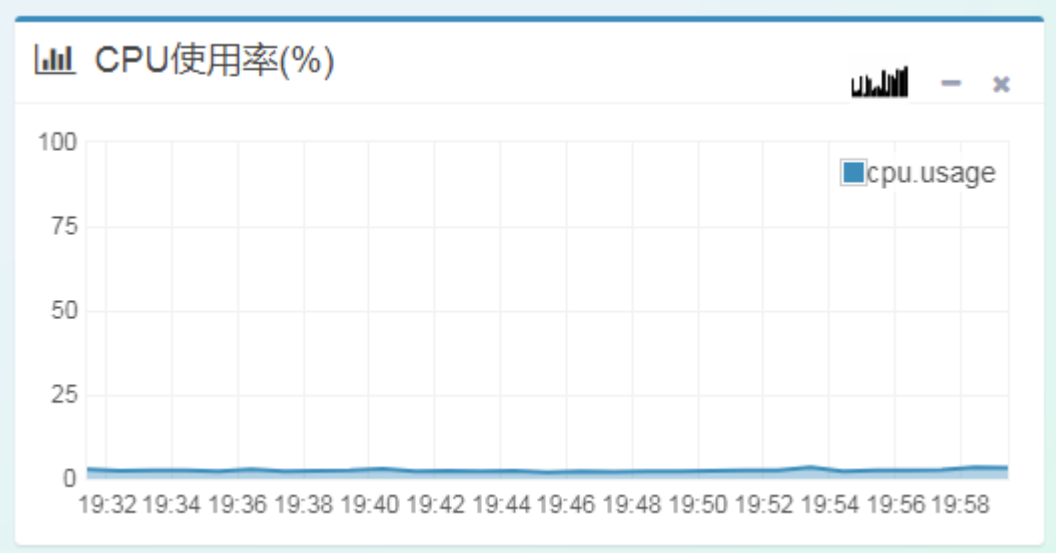
➢ 监控系统简介

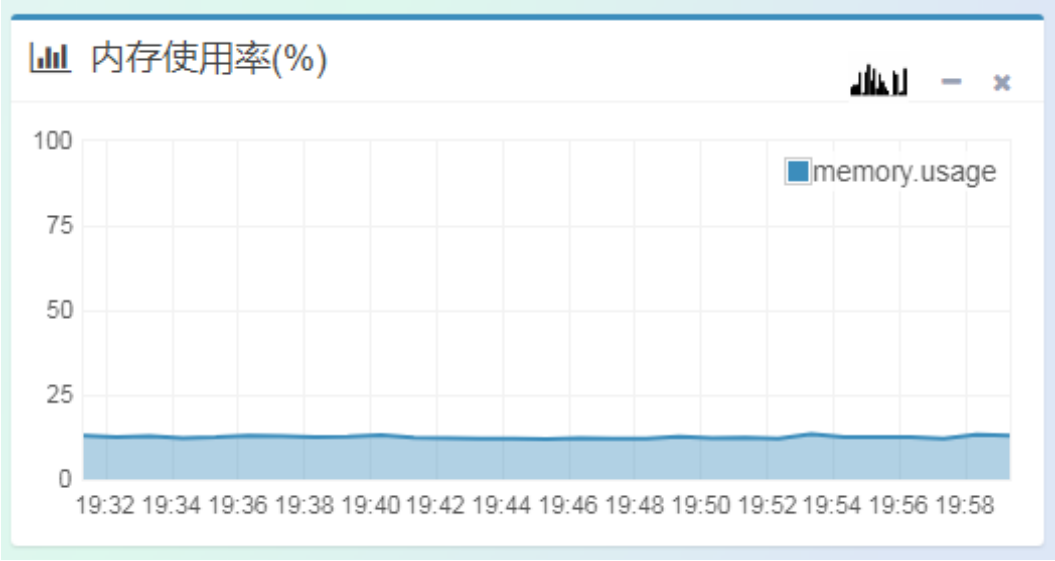
- 数据收集：服务器状态以及应用日志
- 数据传输：agent将数据发送至接收模块，对日志进行分类、格式处理后进行存储
- 数据存储：redis、mysql、Hbase
- 数据分析：数据处理，分析判断，报警
- 前台展示：数据展示、设置告警规则、查询历史数据等

监控与告警

服务器状态监控：

CPU使用率、内存使用率、负载、磁盘使用率、磁盘读写速度、网络入流量、网络出流量、TCP连接数、TCP重传数等





监控与告警

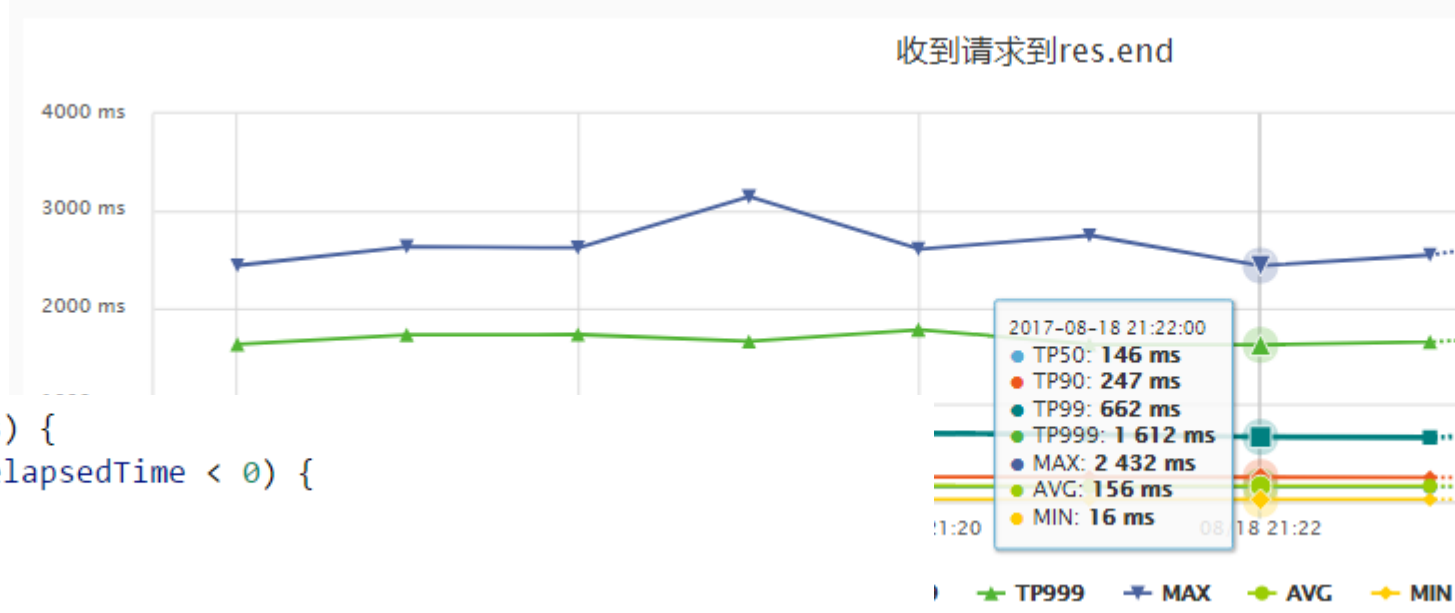
系统存活监控：

```
function _addHeartBeats(aliveLogger, key, ipAddress, heartBeatInterval) {
    var handler = setInterval(()=> {
        var result = `"key":"${key}","hostname":"${ipAddress}","time":"${formatCurTime()}"`;
        aliveLogger.info(result);
    }, heartBeatInterval);
    return handler;
}
```

监控与告警

方法性能监控：

性能数据、可用率、调用次数等




```
sendRegisterInfo(key, elapsedTime, isSuccess) {
    if (typeof elapsedTime !== "number" || elapsedTime < 0) {
        return;
    }
    if (isSuccess === false) {
        isSuccess = "1"
    } else {
        isSuccess = "0"
    }
    var result = `"time":"${formatCurTime()}","key":"${key}","appName":"${this.appName}",
    "hostname":"${this.ipAddress}","processState":"${isSuccess}","elapsedTime":"${elapsedTime}
    "`
    this.tpLogger.info(result);
}
```

监控与告警

自定义监控：值累计、次数累计等

```
valueAccumulate(key, bValue) {
    var result = '';
    if (!((Object.prototype.toString.call(key) === '[object String]') && Number.isFinite
    (bValue))) {
        return;
    }
    result = `"bTime":"${formatCurTime()}","logtype":"BIZ","bKey":"${key}","bHost":"$
    {this.ipAddress}","type":"1","bValue":"${bValue}"`;
    this.bizLogger.info(result);
}
countAccumulate(key) {
    var result = '';
    if (!(Object.prototype.toString.call(key) === '[object String]')) {
        return;
    }
    result = `"bTime":"${formatCurTime()}","logtype":"BIZ","bKey":"${key}","bHost":"$
    {this.ipAddress}","type":"2","bCount":"1"`;
    this.bizLogger.info(result);
}
```

监控与告警

自动告警

设置性能报警规则

TP99 [ ] 毫秒　　TP999 [ ] 毫秒

设置可用率报警规则

当方法的可用率连续[ ]次小于[ ]%则报警，且在[ ]分钟内只报一次警

设置调用次数报警规则

当连续[ ]次，方法调用次数在[ ]*5分钟内[请选择]▼[ ]次进行报警，且在[ ]分钟内只报一次警

告警记录

	接入Key	Key说明	报警时间	报警内容	操作
1	[redacted]	[redacted]	2017-08-18 21:55:04	方法调用次数报警!KEY【[redacted]	明细

监控与告警

手动告警

```
var phones = ['13600000000', '13700000000', '13800000000'].join(';');
var emails = ['xxx@jd.com', 'yyy@jd.com', 'zzz@jd.com'].join(';');
function sendAlarm(param) {
    var formatPara = qs.escape(JSON.stringify(param));
    var options = {
        hostname: 'xxx.m.jd.com',
        path: '/alarm?req=' + formatPara,
        method: 'GET'
    };
    var req = http.request(options, function (res) {
        logger.warn('警告信息发送成功: ' + qs.unescape(formatPara));
    });
    req.on("error", function (e) {
        logger.error('告警信息发送失败: ' + qs.unescape(formatPara));
    });
    req.end();
}
```

监控与告警

终端页面性能监控：

Performance.Timing API

```
if (that.isTimingApiAvailable) {
    that.performance.timing.firstPaint = firstPaintTimeValue;
    var timing = that.performance.timing;
    timingData.domainLookup = timing.domainLookupEnd - timing.domainLookupStart; // point5
    timingData.redirection = timing.fetchStart - timing.navigationStart;// point6
    timingData.serverConnection = timing.requestStart - timing.connectStart;// point7
    timingData.request2ResponseStart = timing.responseStart - timing.requestStart;// point8
    timingData.responseStart2responseEnd = timing.responseEnd - timing.responseStart;// point9
    timingData.request2ResponseEnd = timing.responseEnd - timing.requestStart;// point10
    timingData.domInteractive = timing.domInteractive - timing.navigationStart;// point11
    timingData.domContentLoaded = timing.domContentLoadedEventStart - timing.navigationStart;// point12
    timingData.firstPaint = timing.firstPaint - timing.navigationStart;// point13
    timingData.pageLoad = timing.loadEventStart - timing.navigationStart;// point14
    timingData.backEnd = timing.responseEnd - timing.navigationStart;// point15
    timingData.frontEnd = timing.loadEventStart - timing.responseEnd;// point16
```

15	point15	responseEnd-navigationStart	0	否	2016-10-21 14:31:41	2016-10-21 14:31:41	查看数据 编辑 删除
16	point16	onloadEventStart - responseEnd	0	否	2016-10-21 14:31:41	2016-10-21 14:31:41	查看数据 编辑 删除
20	point20	首屏第一张图片本身下载耗时	0	否	2016-10-21 14:31:41	2016-10-21 14:31:41	查看数据 编辑 删除
21	point21	navigationStart到首屏第一张图片下载完成	0	是	2016-10-21 14:31:41	2016-11-28 17:02:26	查看数据 编辑 删除

监控与告警

依赖的接口与系统监控

在上游接口调用时添加监控（便于了解上游接口性能、状态、查错）

数据库、缓存服务等系统中添加监控，并密切关注

监控与告警

TIPS:

美好的一天，从看监控开始

熟悉应用的负载（日常负载数据、负载随流量增长的变化趋势）

掌握应用的临界值（通过压测等）

流量峰谷的规律（峰、谷的时间、流量；形状；运营规律；预测）

区分日志与告警（告警是急迫的、未知的）

对告警保持敏感（告警及时处理、定期检查告警记录）

高可用与容灾

1. 增强代码健壮性

输入校验

添加 uncaughtException，unhandledRejection 处理

Try…Catch…

及时清理代码

单元测试

高可用与容灾

2. HAProxy+Nginx+pm2+cluster

部署 HAProxy 做负载均衡

Nginx 在80端口，反向代理至node应用端口

Nginx 处理gzip，http头设置

Pm2自动重启进程

Pm2 使用 cluster 模式

高可用与容灾

3. 多机房部署（异地机房）

4. 单机故障发现与剔除（返回状态码监控，摘vip）

5. 业务节点的无状态化（stateless，便于横向扩展，业务中的状态数据集中放到缓存服务器中）

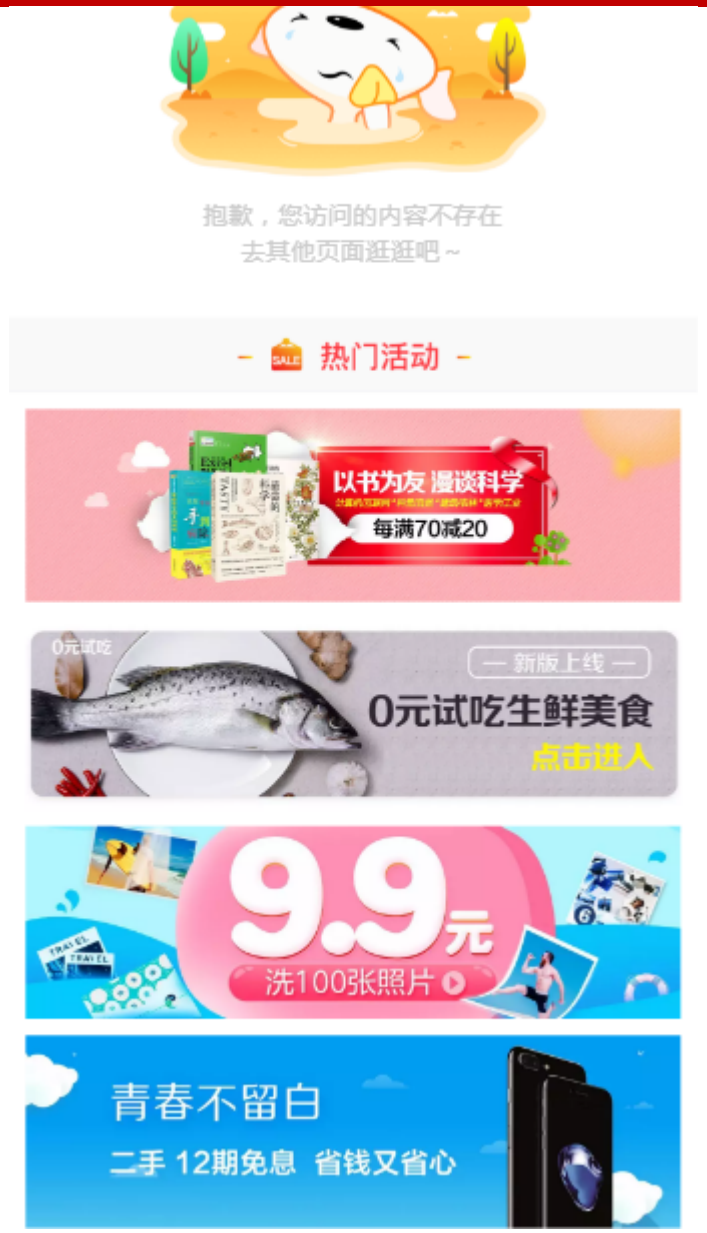
6. 业务节点不存在单点依赖（消除单点故障隐患）

7. 快速扩容（有相应的基础设施与流程）

高可用与容灾

8. 统一错误页

- 友好提示，避免出现404, 500等错误码
- 充分利用流量



高可用与容灾

8. 本地缓存与Redis缓存

将非个性化页面在本地与redis中进行缓存

本地内存有限，基于LRU实现缓存，占用空间小

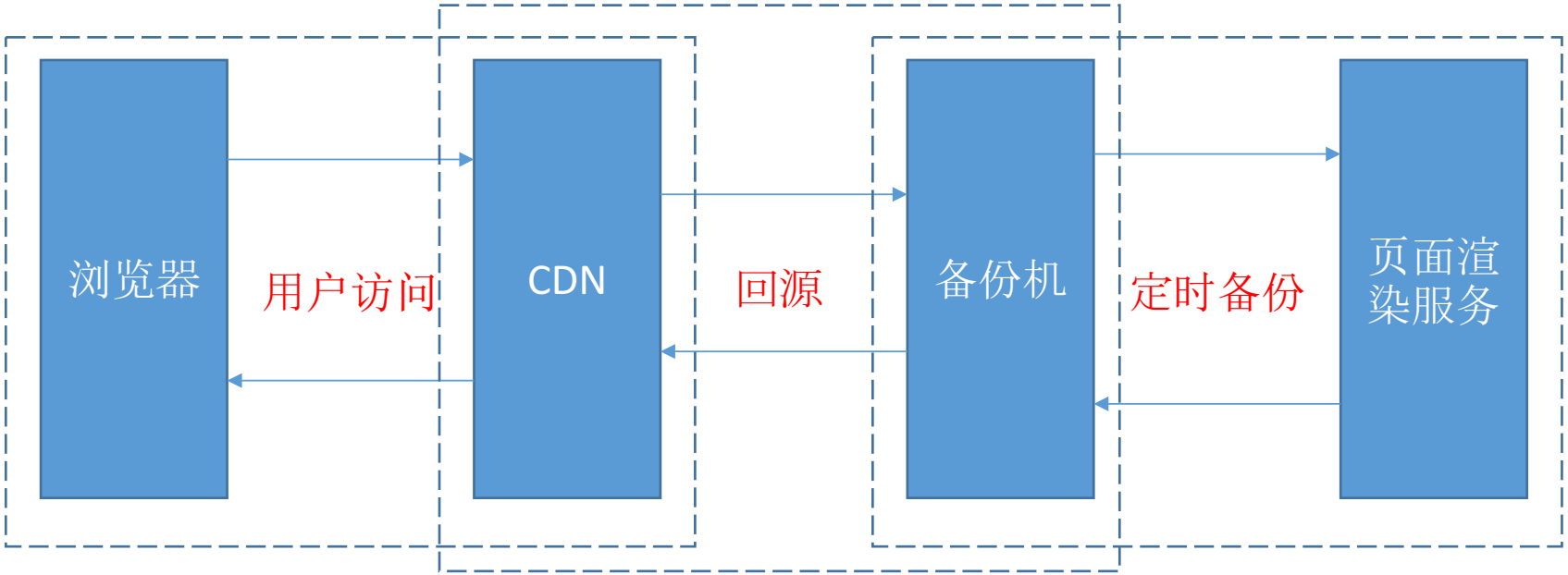
本地缓存不能命中的，去 Redis中取

Redis不能命中，则走完整渲染逻辑

高可用与容灾

9. CDN兜底

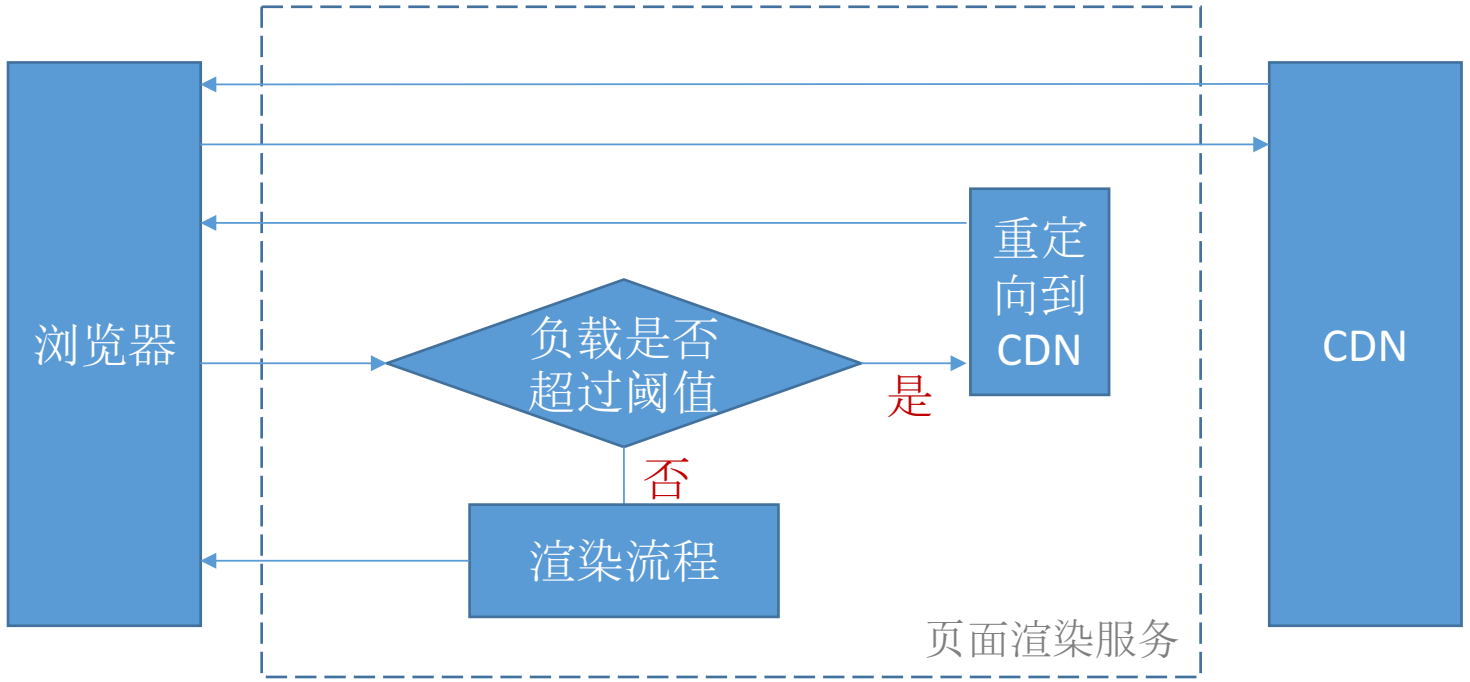
- 由备份机定期请求一遍所有活动，并保存
- CDN从备份机回源
- 618，11.11等大促时，在零点前预先备份零点后页面



高可用与容灾

10. 高负载自动保护

页面渲染服务的主要消耗在模板引擎渲染，由于渲染是CPU密集型任务，访问量大的时候，容易使得系统负载（CPU, 网络流量）升高，导致响应延迟，性能下降；严重时可能使得请求累积，内存飙升；为应对大促时的高QPS，保持系统健康和快速响应，我们对负载超过安全阈值时，自动将请求重定向到CDN，从而对系统自动保护。



高可用与容灾

11. 手动按比例切走流量

由于在CDN上已经有兜底的静态页面，在负载超过阈值时，也可以按比例手工将流量切到CDN页面。

12. 偶发性错误时（请求超时等），自动跳到CDN兜底页面，充分利用流量；

高可用与容灾

13. Nginx转发请求

在极端情况下，页面渲染node应用不可用时，可以按照 URL 规则，把请求从当前机器重定向到CDN上的静态兜底页面。

14. 域名切换

在特别极端情况下，整个集群不可用时，在域名解析层，通过域名转发把整个应用的所有流量都切换到CDN的静态兜底页面上。

15. 应急响应预案

在高压下，人的行为可能有所偏差；提前制定预案能规范操作，缩小影响范围，减少系统恢复时间。

测试

1. 单元测试
2. 回归测试
3. 压力测试

- 掌握系统的极限
- 流量生成尽可能模拟真实用户（测试页面的总数、不同页面的比例、峰值出现的速度等，流量的来源）
- 压测环境尽可能与生产环境一致（机器配置、所在机房、）
- 全链路压测（应用不是孤立的，带着上下游一起压测才能更好暴露问题）
- 优化方向往往隐藏在压测数据中
- 避免污染线上数据、避免影响用户访问

安全

1. HTTP 安全头部设置

- Strict-Transport-Security
- Content-Security-Policy
- X-XSS-Protection
- X-Content-Type-Options

```
var express = require('express')
var helmet = require('helmet')

var app = express()

app.use(helmet())
```

安全

2. 注意敏感信息外泄

不在生成的页面（前端）中留下无关信息

不在后端代码中留下服务器账户信息、系统管理账户等

不将内部 API 公开

经常 code review

3. 谨慎操作 cookie

- 配置属性Secure、HttpOnly
- 写入 cookie 时填写完整的 Domain，path，expires 信息
- 读写时分别进行 decode 与 encode

安全

4. 防XSS攻击

使用xss扫描器

对所有用户输入进行特殊字符转义

对所有会写进页面的数据进行特殊字符转义

Require(‘xss’)，使用 xss 等模块

5. 确保依赖库安全

- 慎重增加新依赖，优先选用用户数量大的模块
- 私有npm库
- npm shrinkwrap

安全

6. 隐藏应用相关信息

- X-Powered-By
- 防止直接输出错误堆栈信息

7. 信赖专业团队

- 涉及登陆、用户信息等敏感数据，寻求专业团队帮助
- 邀请安全团队做审计

辅助工具

开发及打包工具

- 自动生成CSS
- 自动生成预编译模板
- 自动重启开发服务器
- 自动上传静态资源到CDN
- 自动生成待发布文件

辅助工具

系统参数配置工具

辅助工具

运维相关工具

域名; 不填默认为 xxx.m.jd.com

请填入端口号; 不填默认为 80

活动ID; 不填默认为 xxxxxxxxxxxxxxxxxxxxxxxxxxxx

host; 不填默认为 pro.m.jd.com:80

页面成功判断依据

请输入 IP 地址列表，不填默认检查所有服务器上保存的默认机器列表;

开始检查

辅助工具

TIPS:

增强意识：重复的工作自动化

善于使用开源工具

重在积累

工具与主应用分类

性能优化

TIPS:

先抗住再优化

能加机器解决的先加

用数据说话

在需求迭代中持续优化

注意缓存的坑

to: suhaoxin@jd.com

cc: cengtong@jd.com

Thank You