API服务性能优化实践

杨前利

2017-08-26

2 自我介绍

2007.9

华中科技大学

本科

2011.9

华中科技大学

硕士

2014.3

美团网

核心业务系统部CRM

2015.10

新美大

酒旅度假业务研发部

门票API服务负责人[2017.4]

1.系统性能指标?

2.指标如何量化?

3.指标如何监控?

4.指标怎么优化?

大纲

5 美团性能指标体系

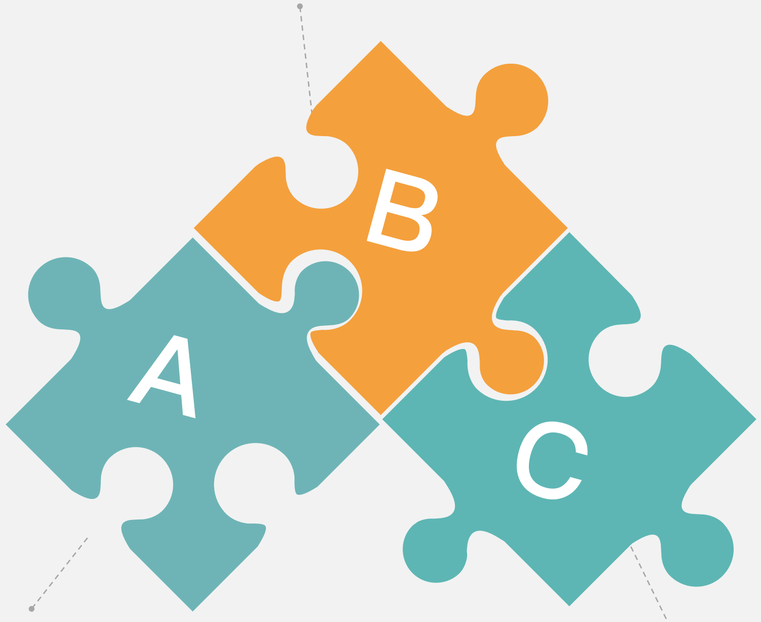
1.集群性能指标	QPS	响应时间	成功率	资源利用率	集群容量	
2.单机性能指标	QPS	响应时间	成功率	资源利用率	单机容量	线程池
3.JVM性能指标	GC	内存	线程	类加载		
4.机器性能指标	CPU	内存	磁盘	网络		
5.外部资源性能监控	数据库	缓存	消息队列	下游RPC调用指标		

6 服务性能指标

2.延迟

系统处理一个请求或者一个任务的延迟

平均时间、90%(TP90)、99%(TP99)等



1.吞吐量

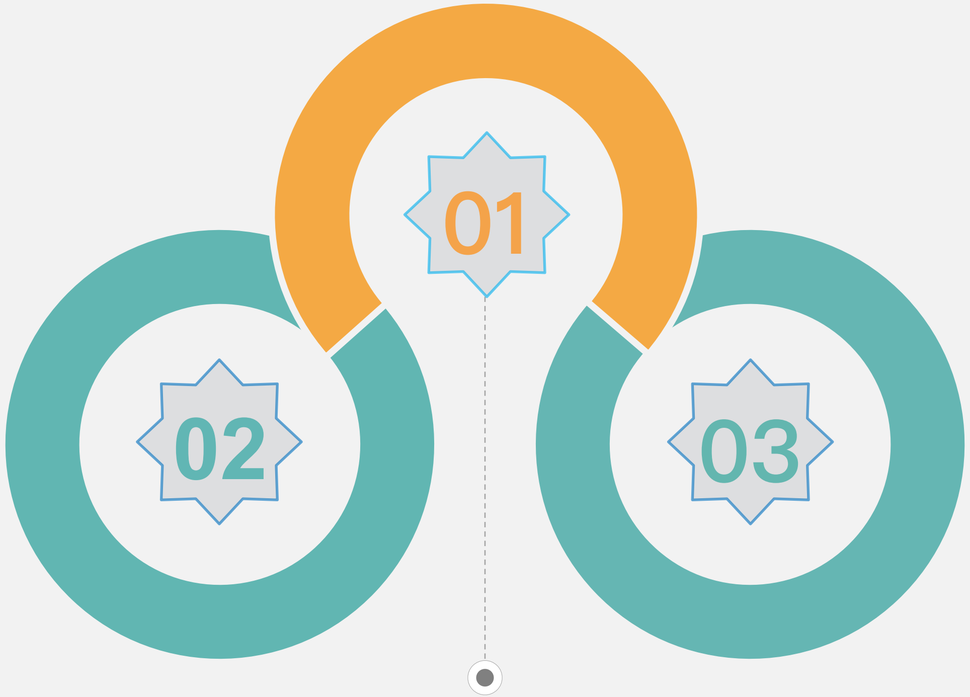
QPS，即单位时间内服务器可以处理的请求数或者任务数

3.成功率

高于一定比例才有意义

- 90%的响应时间：将一段时间内所有请求的响应时间中取一个值，使90%的请求响应时间均小于或等于它，此值即为90%请求覆盖的响应时间。

7 JVM性能指标



2.gc性能指标

jvm.gc.count
jvm.gc.time
jvm.fullgc.count
jvm.fullgc.time
jvm.younggc.count
jvm.younggc.time
常用命令：jstat –gc

3.thread性能指标

jvm.thread.count
jvm.thread.runnable.count
jvm.thread.blocked.count
jvm.thread.waiting.count
jvm.thread.time_waiting.count
常用命令：jstack

1.memory性能指标

jvm.memory.oldgen.used
jvm.memory.survivor.used
jvm.memory.eden.used
jvm.memory.perm.used
jvm.nio.directbuffer.used
常用命令：jmap

8 机器性能指标

内存

mem.memused.percent

mem.memfree.percent

mem.swapused.percent

常用命令：top、vmstat

磁盘

df.bytes.free.percent

df.inodes.free.percent

常用命令：df、du、ll等

CPU

load.1minPerCPU

cpu.switches

cpu.idle

cpu.iowait

常用命令：top、vmstat

网络

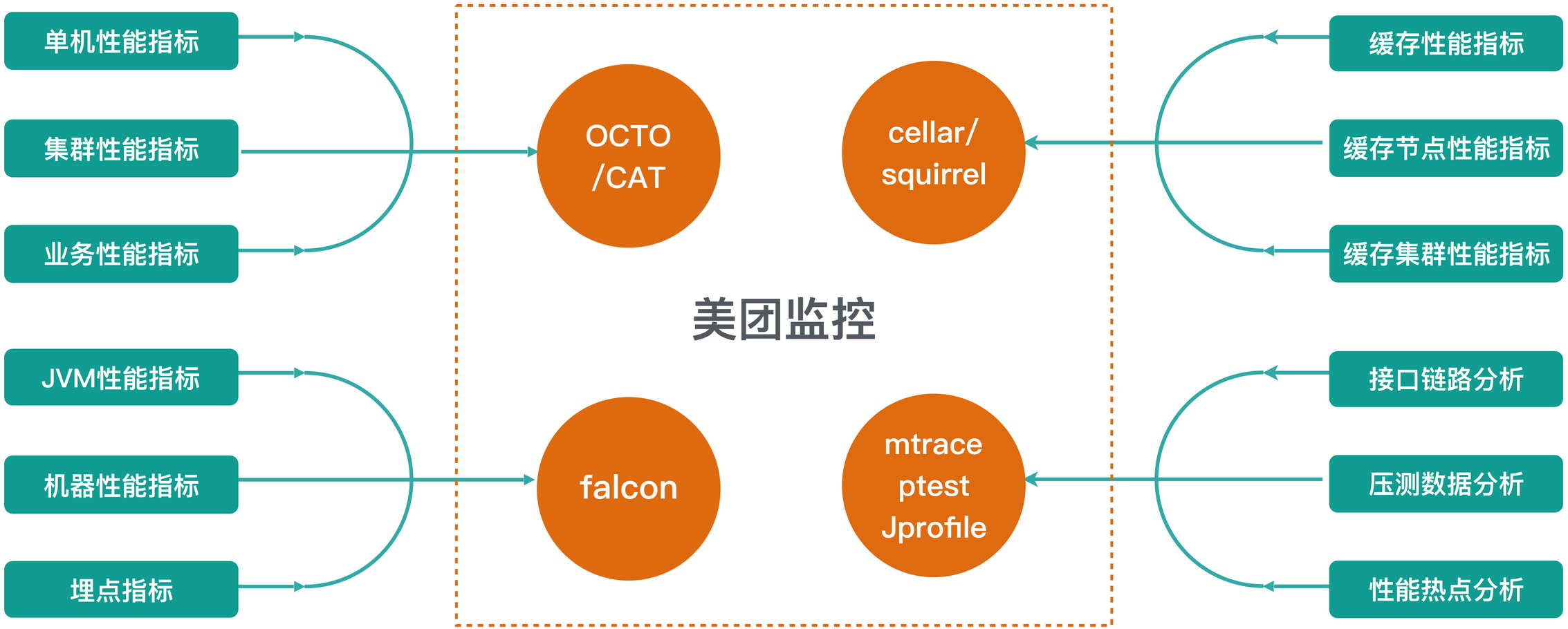
net.if.in.bytes

net.if.out.bytes

常用命令：netstat、ss

大纲

美团服务性能指标监控


单机性能指标
集群性能指标
业务性能指标
JVM性能指标
机器性能指标
埋点指标
OCTO
/CAT
falcon
美团监控
cellar/
squirrel
mtrace
ptest
Jprofile
缓存性能指标
缓存节点性能指标
缓存集群性能指标
接口链路分析
压测数据分析
性能热点分析

美团服务性能指标监控

监控工具：OCTO（服务治理平台 未开源）

服务治理平台 | Dashboard | 服务详情 | 服务运营 | 数据分析 | 服务报表 | 监控报警 | 服务视图 | 文档与帮助

选择服务：

数据总览 | 业务指标 | 性能指标 | 来源分析 | 去向分析 | 主机分析 | 上下游分析 | 调用链分析 | 标签治理

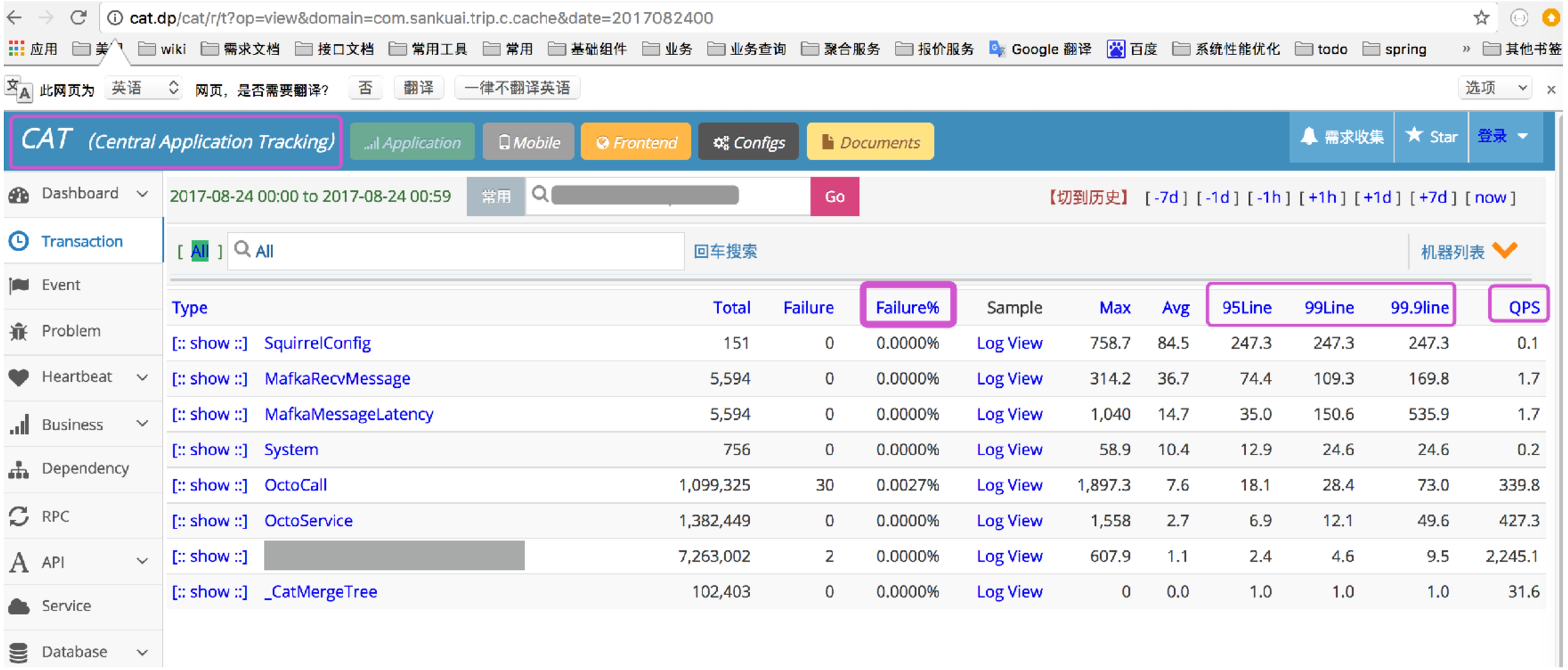
角色：来源 去向　环境：prod staging test　时间粒度：天 小时　时间：2017-08-12　查询　可用性指标　TP耗时数据　同比环比

说明：**1**，QPS，Tp99被表示为：[*数值，环比，同比*]，客户端统计量被表示为：[*调用量，失败数，失败数占比*]。**2**，耗时指标(eg. 平均耗时,TP数据)的单位为毫秒；**3**，可用性相关指标说明见上

接口	调用量	失败数/百分比	QPS(次/秒)	TP90	TP95	TP99	TP999	平均耗时	平均耗时占比	最大耗时
all	7792509	0, 0%	1138,-42%,-52%	47	69	112	238,-4%,-1%	0	0.00%	2231
接口A	1924308	0, 0%	281,-14%,-21%	43	48	67	144,2%,-31%	0	0.00%	441
接口B	409509	0, 0%	59,-63%,-68%	3	4	10	12,0%,-8%	0	0.00%	335
接口C	374466	0, 0%	54,-46%,-59%	14	16	19	53,1%,10%	1	3.65%	256
接口D	335724	0, 0%	49,-47%,-59%	91	106	165	512,0%,39%	8	26.18%	2036

12 美团服务性能指标监控

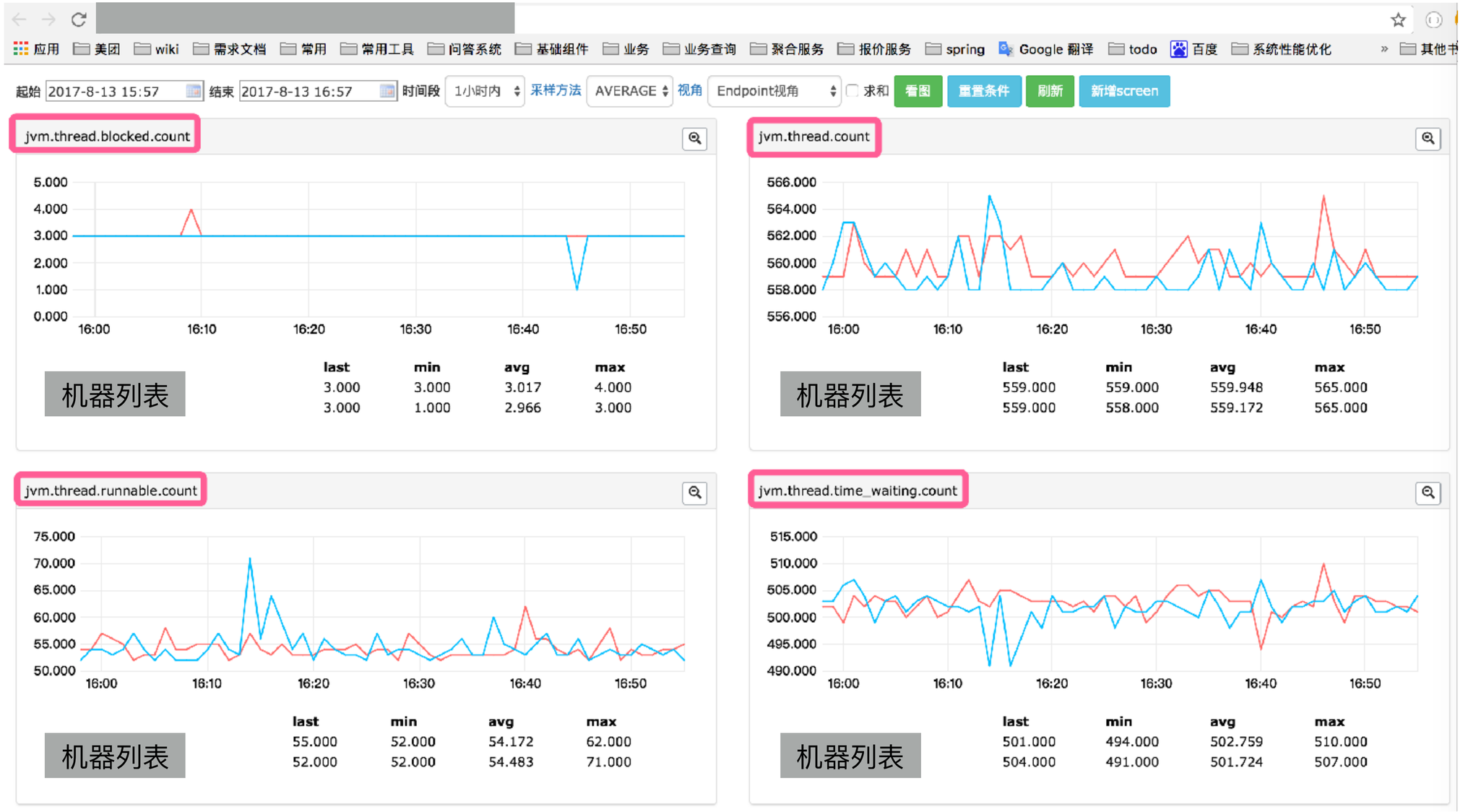
监控工具：CAT（Central Application Tracking 开源）



Type	Total	Failure	Failure%	Sample	Max	Avg	95Line	99Line	99.9line	QPS
[:: show ::] SquirrelConfig	151	0	0.0000%	Log View	758.7	84.5	247.3	247.3	247.3	0.1
[:: show ::] MafkaRecvMessage	5,594	0	0.0000%	Log View	314.2	36.7	74.4	109.3	169.8	1.7
[:: show ::] MafkaMessageLatency	5,594	0	0.0000%	Log View	1,040	14.7	35.0	150.6	535.9	1.7
[:: show ::] System	756	0	0.0000%	Log View	58.9	10.4	12.9	24.6	24.6	0.2
[:: show ::] OctoCall	1,099,325	30	0.0027%	Log View	1,897.3	7.6	18.1	28.4	73.0	339.8
[:: show ::] OctoService	1,382,449	0	0.0000%	Log View	1,558	2.7	6.9	12.1	49.6	427.3
[:: show ::] [redacted]	7,263,002	2	0.0000%	Log View	607.9	1.1	2.4	4.6	9.5	2,245.1
[:: show ::] _CatMergeTree	102,403	0	0.0000%	Log View	0	0.0	1.0	1.0	1.0	31.6

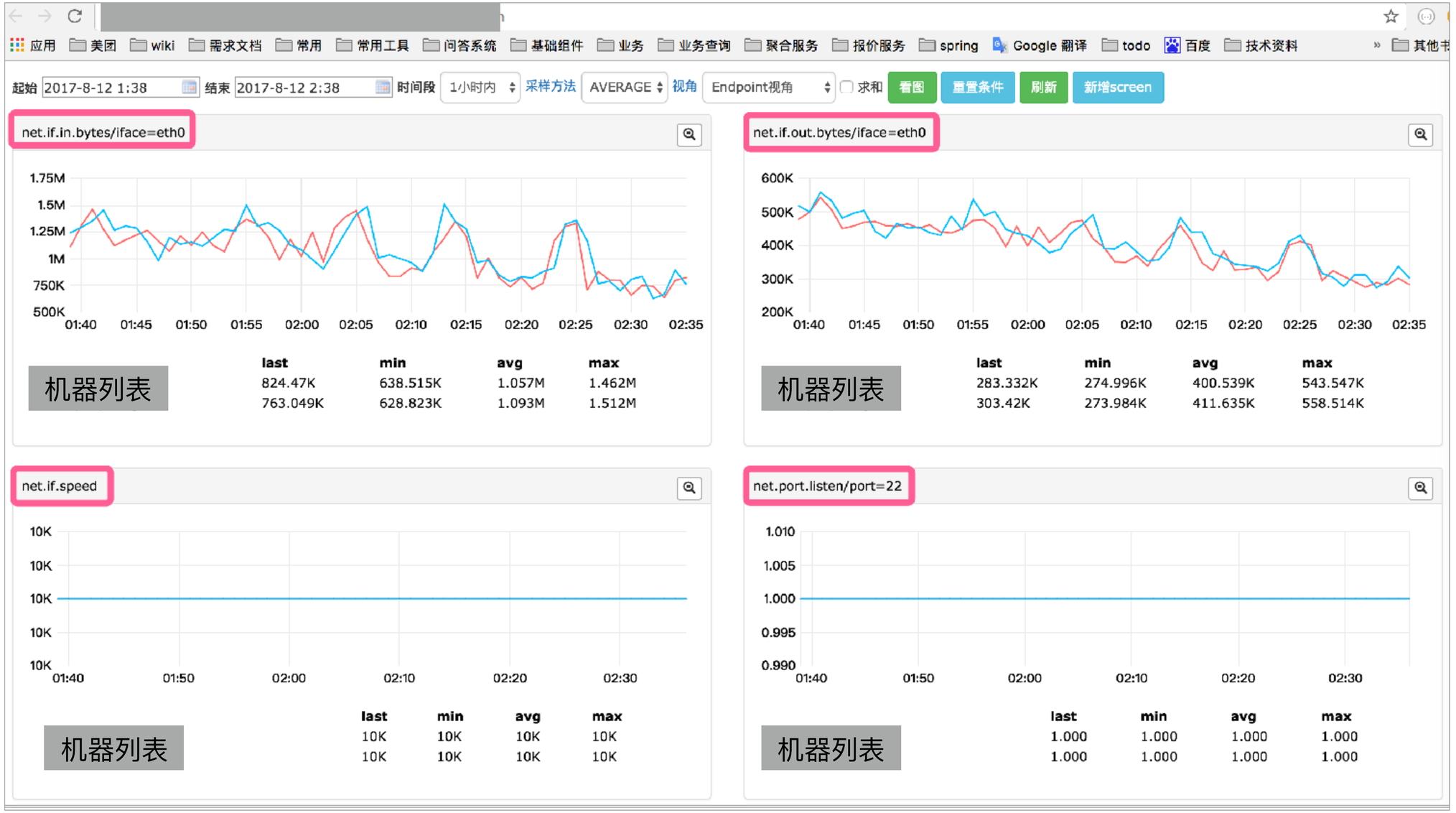
13 美团JVM性能指标监控

监控工具：falcon平台（小米开源的open-falcon）



美团机器性能指标监控

监控工具：falcon平台　开源



服务接口调用链路分析

监控工具：mtrace（分布式会话跟踪系统 未开源）

美团压测工具

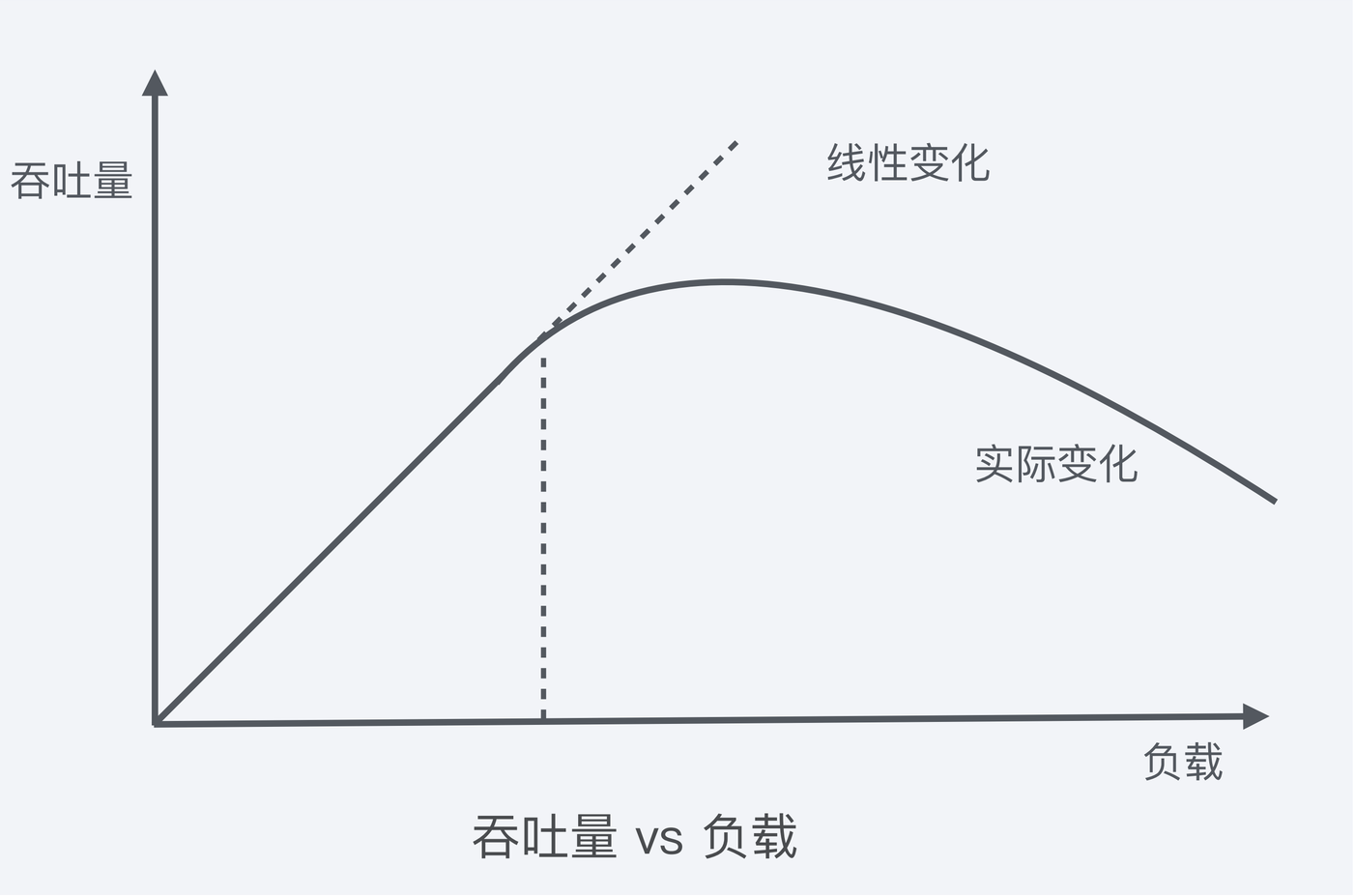
* 美团侧性能测试工具：ptest 未开源
* 点评侧性能测试工具：jmeter 开源
* 十个免费的WEB压力测试工具 http://coolshell.cn/articles/2589.html

吞吐量 vs 负载

1. 在一定阶段扩展性是线性变化的

2. 拐点—资源的竞争开始影响性能

3. 内聚性导致系统完成的工作变少进而吞吐量也减小了



吞吐量 vs 负载

美团压测工具

压力测试工具：ptest

美团压测工具

压力测试工具：ptest

1 请求 | 2 配置 | 3 测试报告与监控

○ 简单模式 ◉ 递增模式 ○ octo权重调整模式 ○ pigeon权重调整模式

初始用户数 5 递增用户数 5 结束用户数 30 *QPS=并发用户数*1000/响应时间(ms)

运行时长(s) 150

单施压机最大并发数 5 *默认50,取值范围[4,100]

施压机数 6 *施压机数=(并发用户数-1)/单施压机最大并发数+1

压力工具所在机房 [redacted] *北京服务:[redacted],上海服务:[redacted]

线上压测为了保证安全性,系统限制当前场景最大并发用户数为100 *系统默认为2,如需修改请联系管理员

更多配置

日志打印比例(%) 1 *默认为1,成功的请求按比例打印,失败的请求全部打印

上一步 下一步 保存

美团压测工具

压力测试工具：ptest

1 请求 | 2 配置 | 3 测试报告与监控

+ 添加测试报告项

监控数据获取失败立即熔断

测试报告项	添加项	操作
响应时间topX	90	删除

+ falcon监控

1.通过 “添加监控” 设置压测监控机器

2.通过 “监控阈值” 设置压测终止条件（可为空）

是否需要与高峰时段进行对比

分组名	主机名(Falcon endpoint)	falcon环境	监控阈值			操作
被压机器	[redacted]	线上	load.1min	>	16	删除

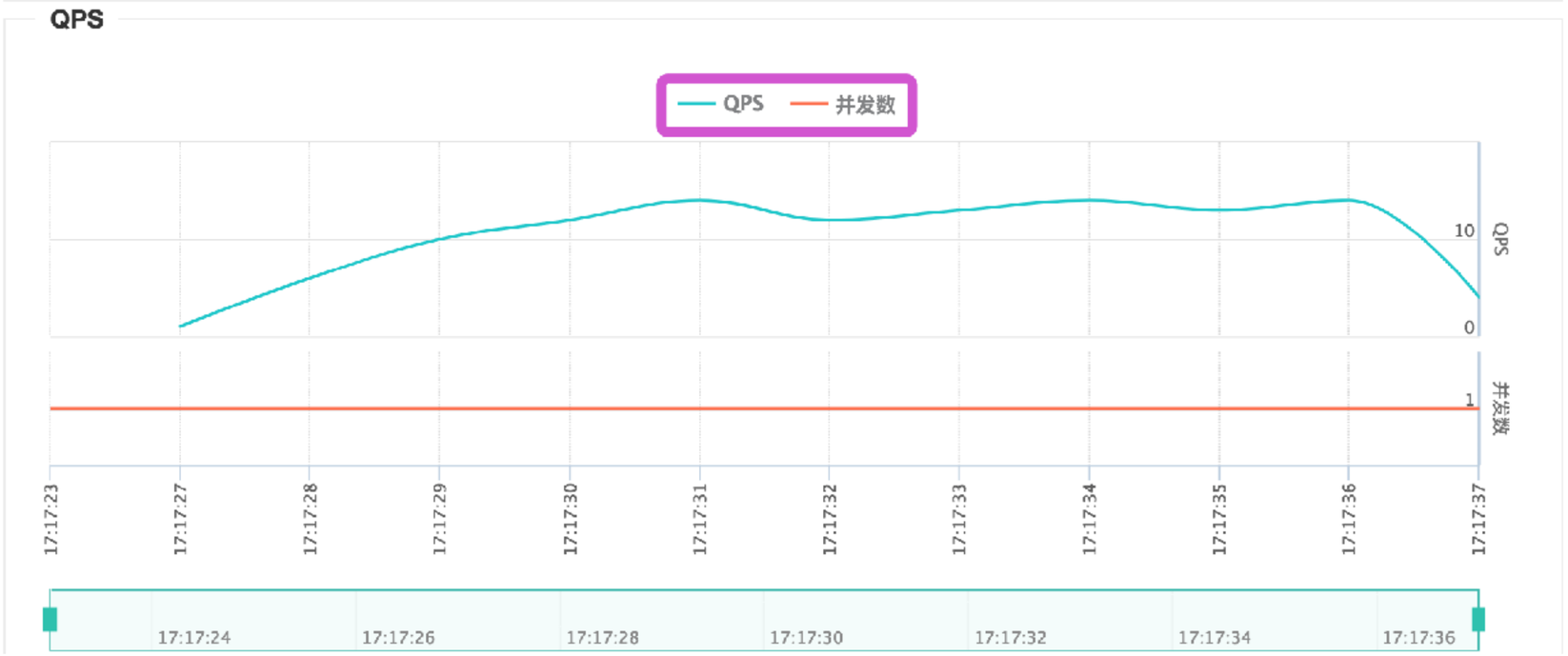
美团压测工具

压力测试工具：ptest

Console输出 | 测试报告 | Falcon监控数据 | Octo监控数据 | Cat监控数据 | 对比报告 | 测试配置

测试报告 查看指标说明

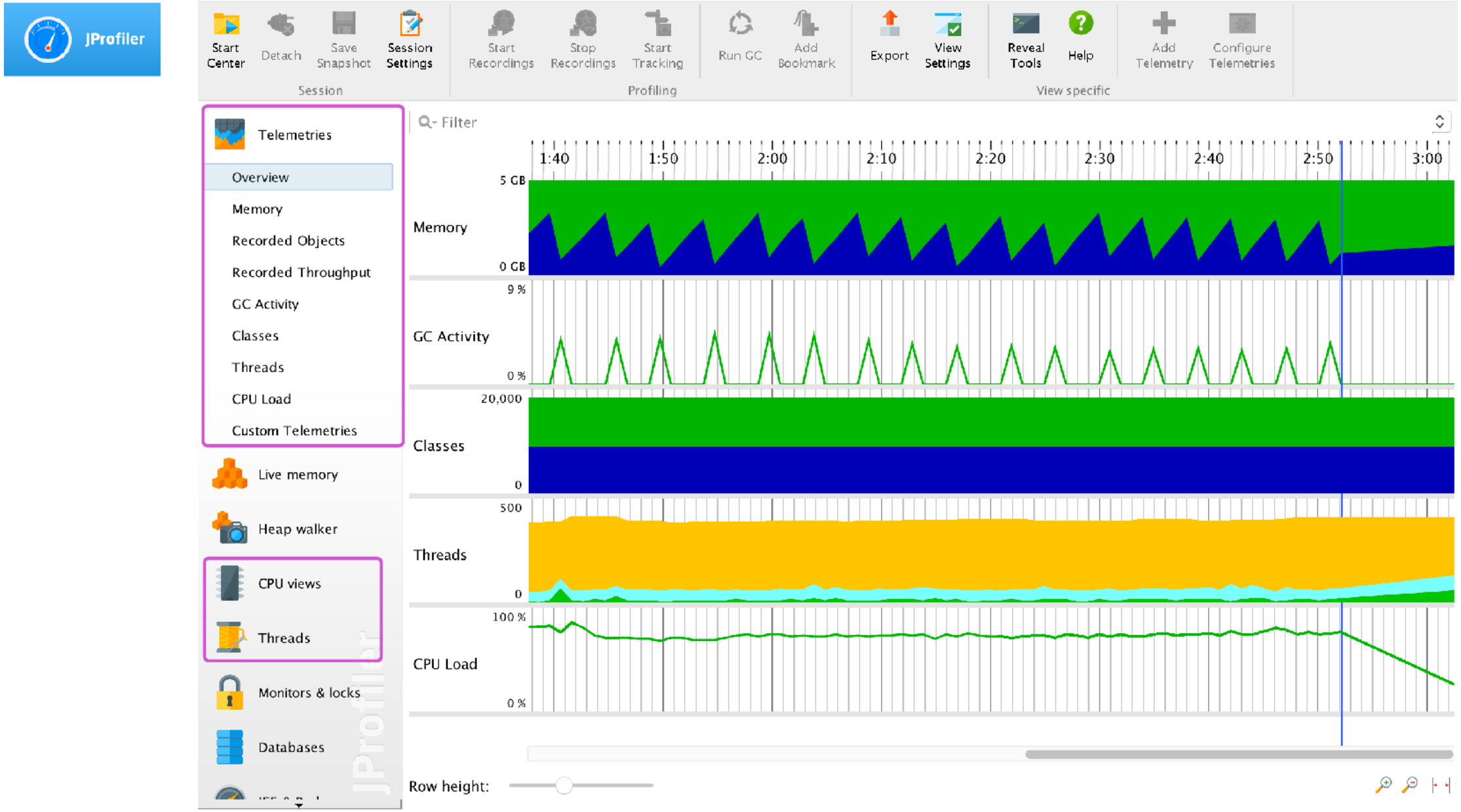
路径	QPS	响应时间					错误率	请求量
		平均	top50	top90	top95	top99		
/api/[redacted]	10	88ms	176ms	186ms	192ms	199ms	0%	113
总计	10	88ms	176ms	186ms	192ms	199ms	0%	113



热点分析

- 热点分析工具：Java的JProfiler
- 分析重点：运行时间长、调用次数多
- 收益较大

热点分析

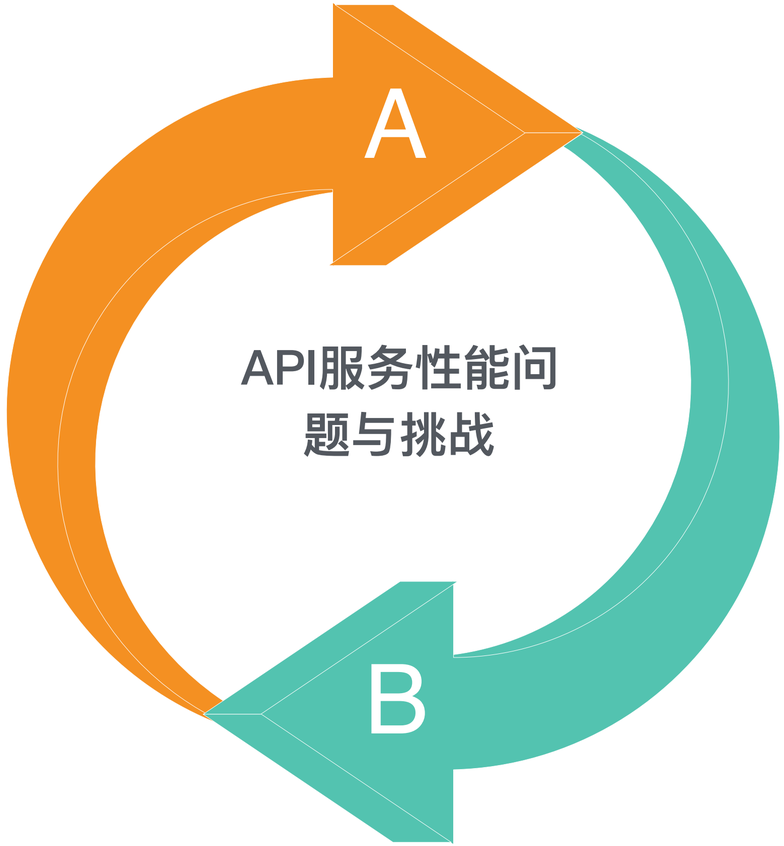

JProfiler
Start Center
Detach
Save Snapshot
Session Settings
Session
Start Recordings
Stop Recordings
Start Tracking
Profiling
Run GC
Add Bookmark
Export
View Settings
Reveal Tools
Help
View specific
Add Telemetry
Configure Telemetries
Telemetries
Overview
Memory
Recorded Objects
Recorded Throughput
GC Activity
Classes
Threads
CPU Load
Custom Telemetries
Live memory
Heap walker
CPU views
Threads
Monitors & locks
Databases
Filter
1:40
1:50
2:00
2:10
2:20
2:30
2:40
2:50
3:00
5 GB
Memory
0 GB
9 %
GC Activity
0 %
20,000
Classes
0
500
Threads
0
100 %
CPU Load
0 %
Row height:

大纲

API服务性能问题与挑战

A 技术方面

- 核心接口的TP90较高，整体服务TP较高
- 服务中线程池使用较多较乱
- 单机并发能力较差
- 服务依赖的下游不稳定



B 业务方面

- 业务增长迅速：服务的QPS峰值已经大于5000
- 日调用量峰值接近3亿，还处于增长中
- 承接业务的端较多，并且兼容30多个版本

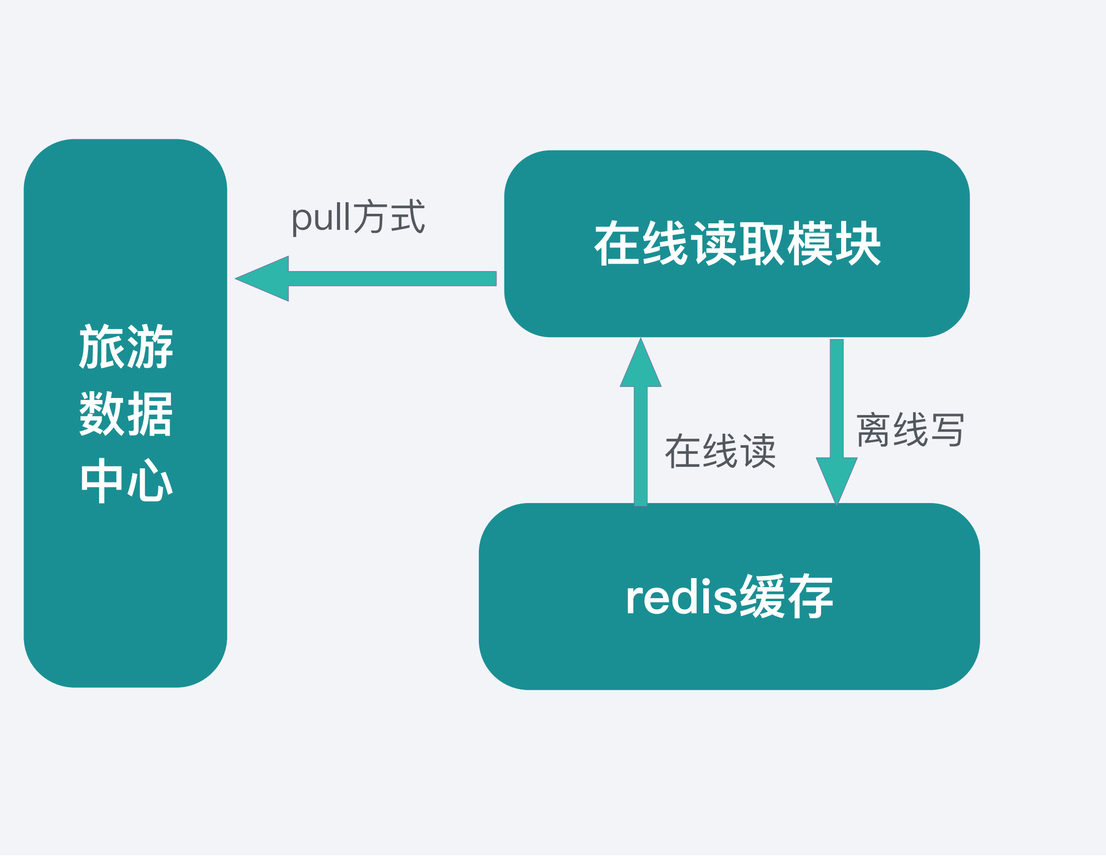
常用性能优化策略

常用性能优化策略

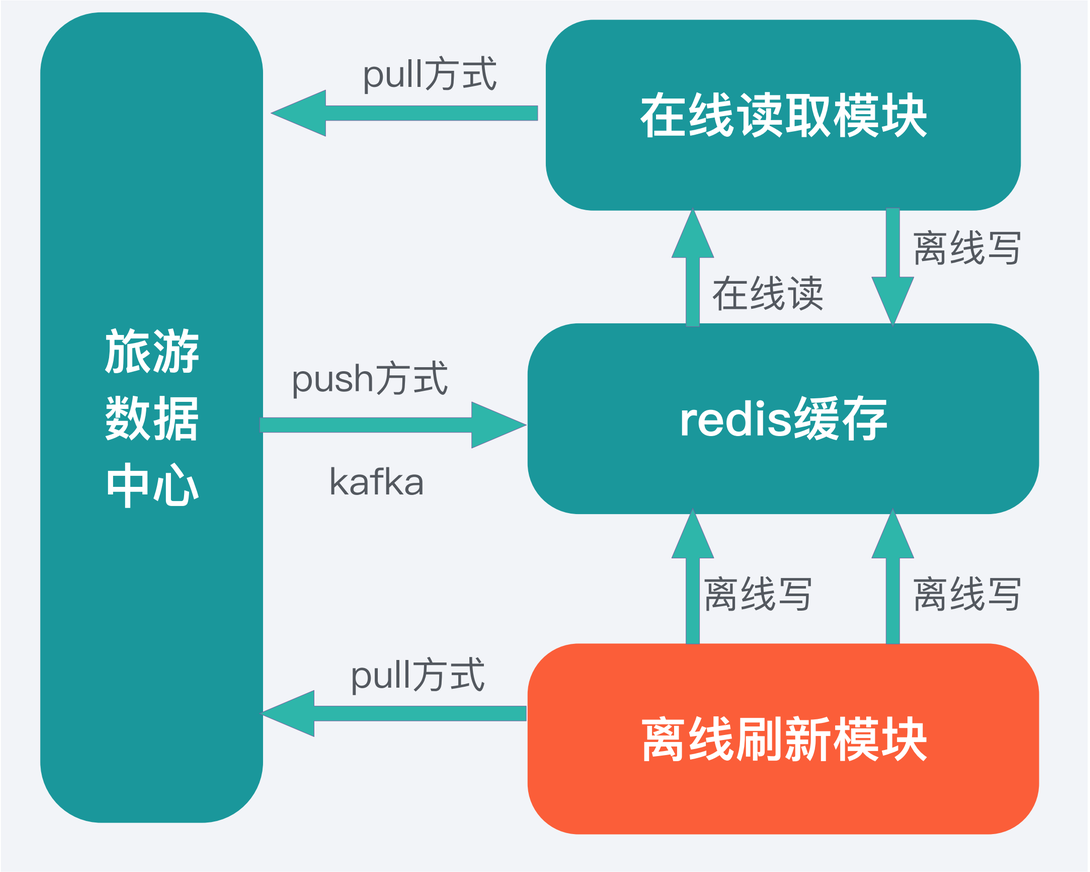
优化方向	性能瓶颈
并行	服务响应延迟高、CPU资源利率用低
异步	服务响应延迟高、CPU资源利率用低
NIO异步化	线程总数高、服务吞吐量低、CPU Load高
JVM调优	CPU Load高、GC次数频繁、GC time高、thread数目异常
热点分析优化	CPU idl低的问题、CPU Load高、服务吞吐量低
缓存优化	服务响应延迟高、网卡流量高
数据库调优	服务响应延迟高 ..
业务优化	服务响应延迟高 ..
组件优化	..

缓存优化实践

经典缓存设计



离线刷新缓存设计



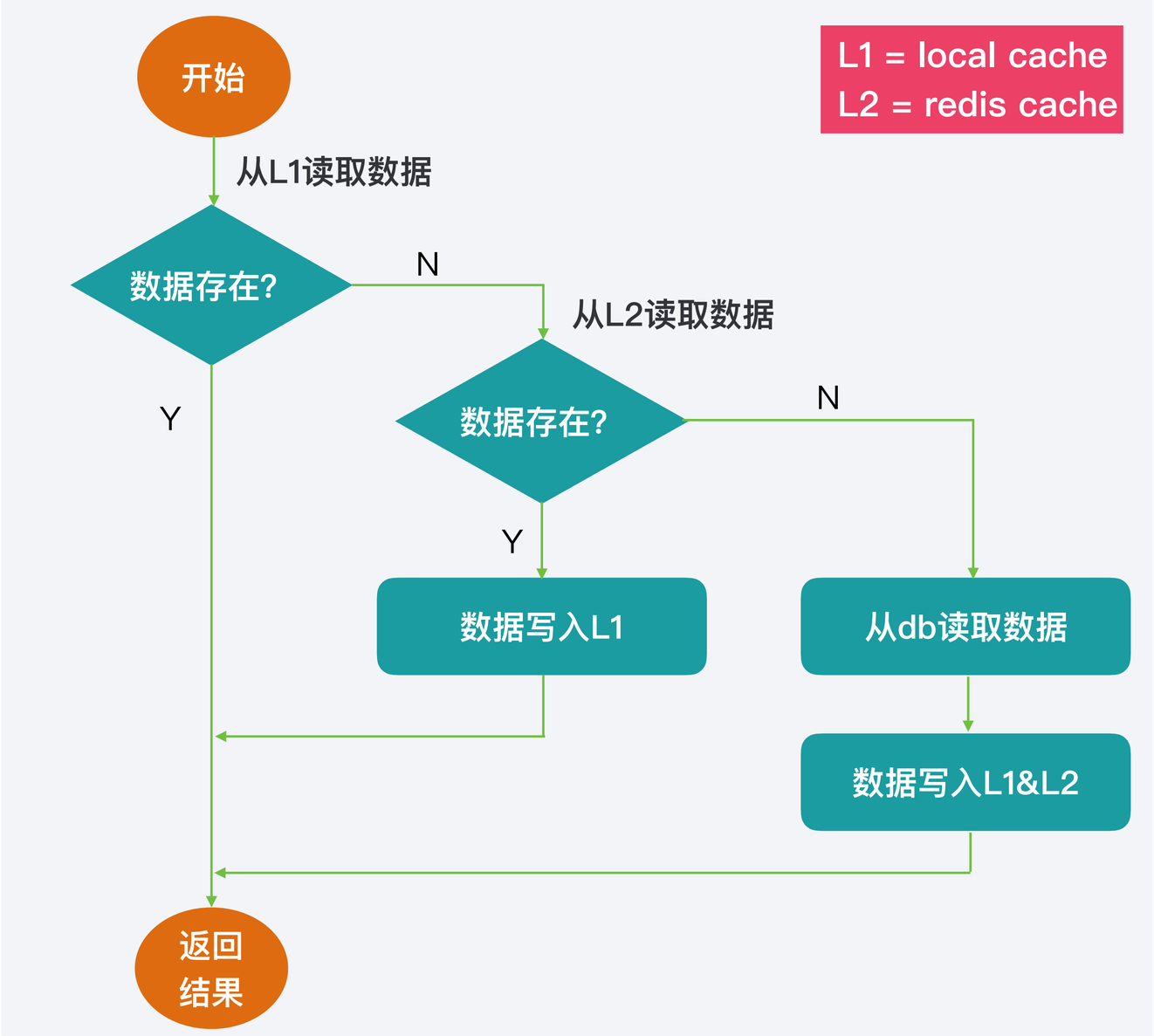
缓存命中率：85%->99%　　服务PT90：53ms->37ms

缓存优化实践

经典二级缓存设计

缓存Key	缓存value	缓存时间
k1	local v1	1h
k1	redis v1	6h

- 本地内存大小
- 本地内存时效性
- 多节点缓存不一致
- 重启缓存失效

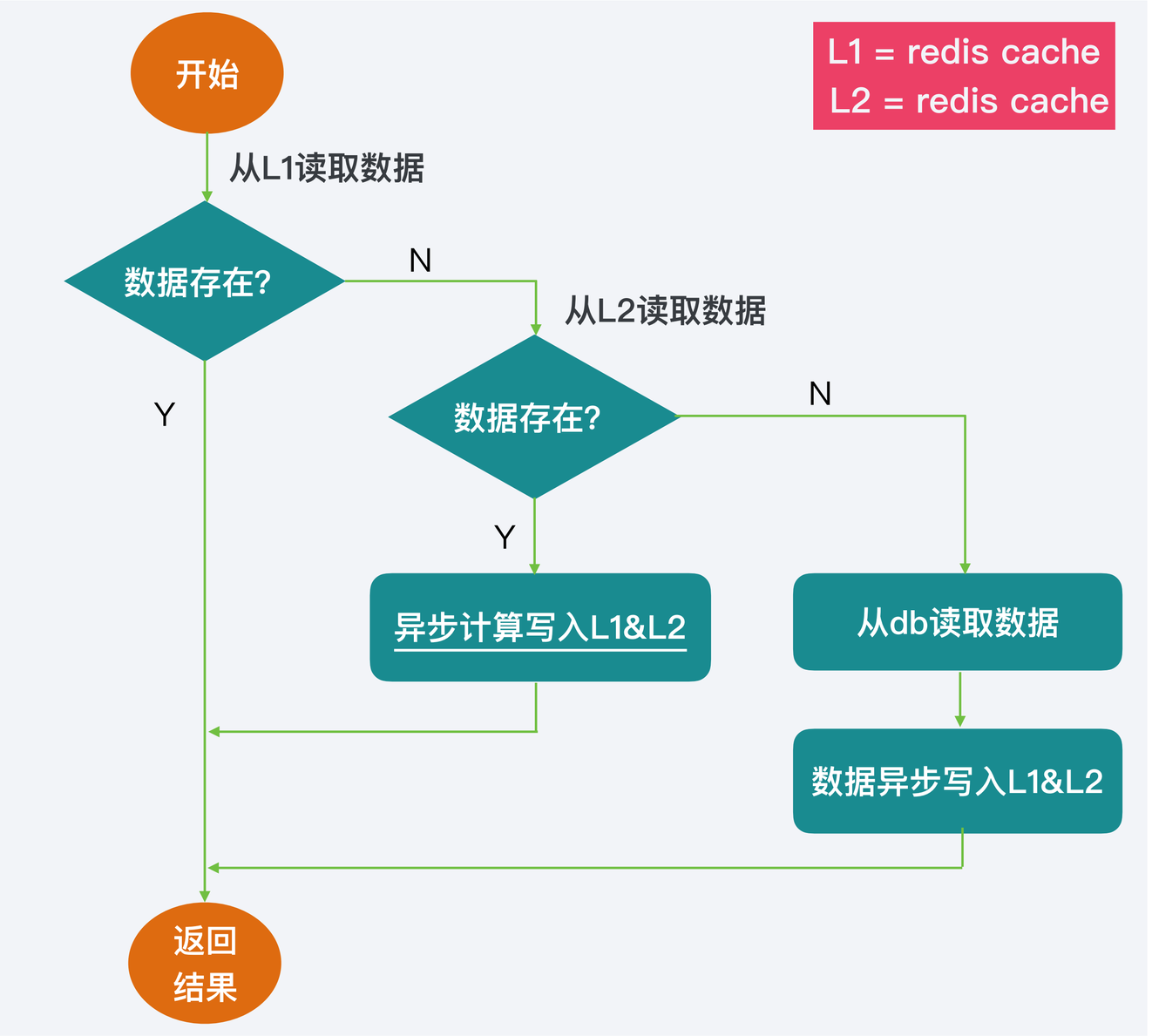


缓存优化实践

改进型二级缓存设计 1

缓存Key	缓存value	缓存时间
k1	redis v1	1h
k1_back	redis v1	6h

- 实践于API服务的poi列表页标签模块
- 缓存命中率: 91% -> 99%
- poi列表页 TP90：320ms ->250ms
- 缺点：一级缓存过期会读到二级缓存，二级缓存不保证数据最新

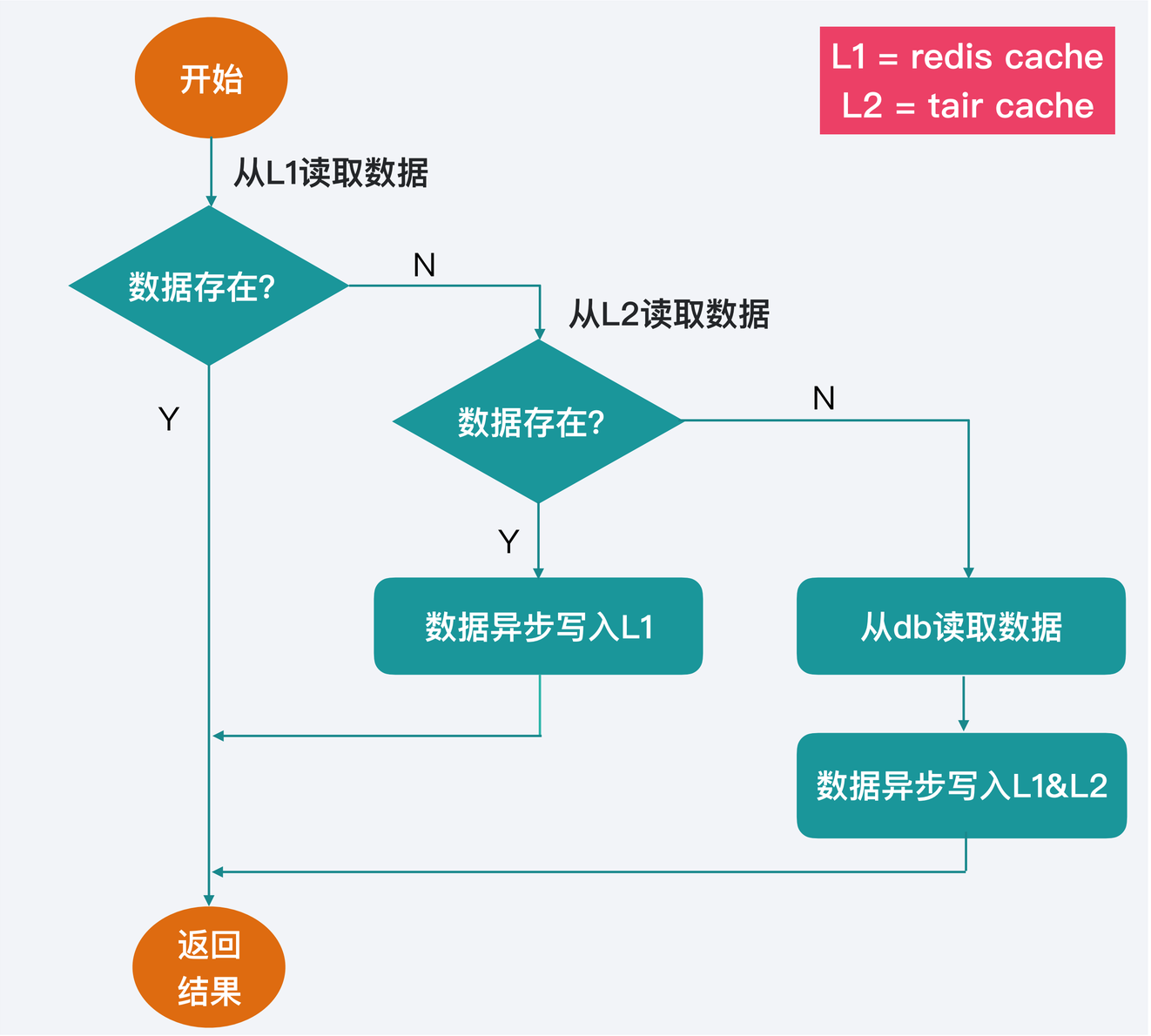


缓存优化实践

改进型二级缓存设计 2

缓存Key	缓存value	缓存时间
k1	redis v1	1h
k1	tair v1	– –

- 一级缓存使用 redis 集群
- 二级缓存使用 tair 集群
- 保证在一级缓存挂的时候 二级缓存可用
- 正在实践中
- 成本较高 也考虑转化为双缓存 一键切换



缓存优化实践

分布式缓存使用小技巧

- 合并多次get请求为一次mGet请求
- 通过异步方式提高吞吐量
- 缓存value不宜过大
- 预防缓存被击穿——分布式锁使用

缓存优化实践

场景

1.很热的某个key过期了

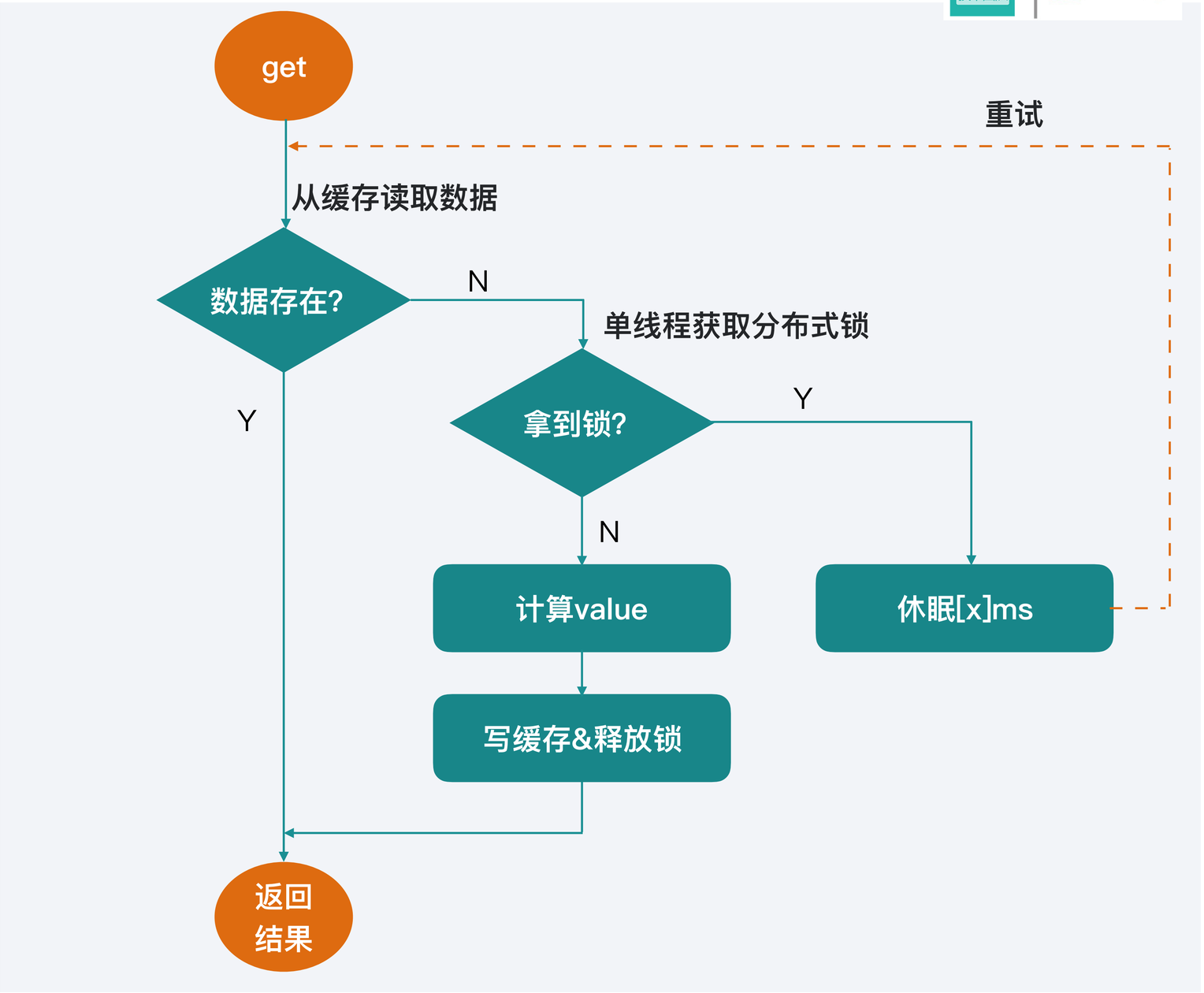
2.恰好这个点大量的并发请求 这个key

3.大量并发计算key可能会打垮后端DB

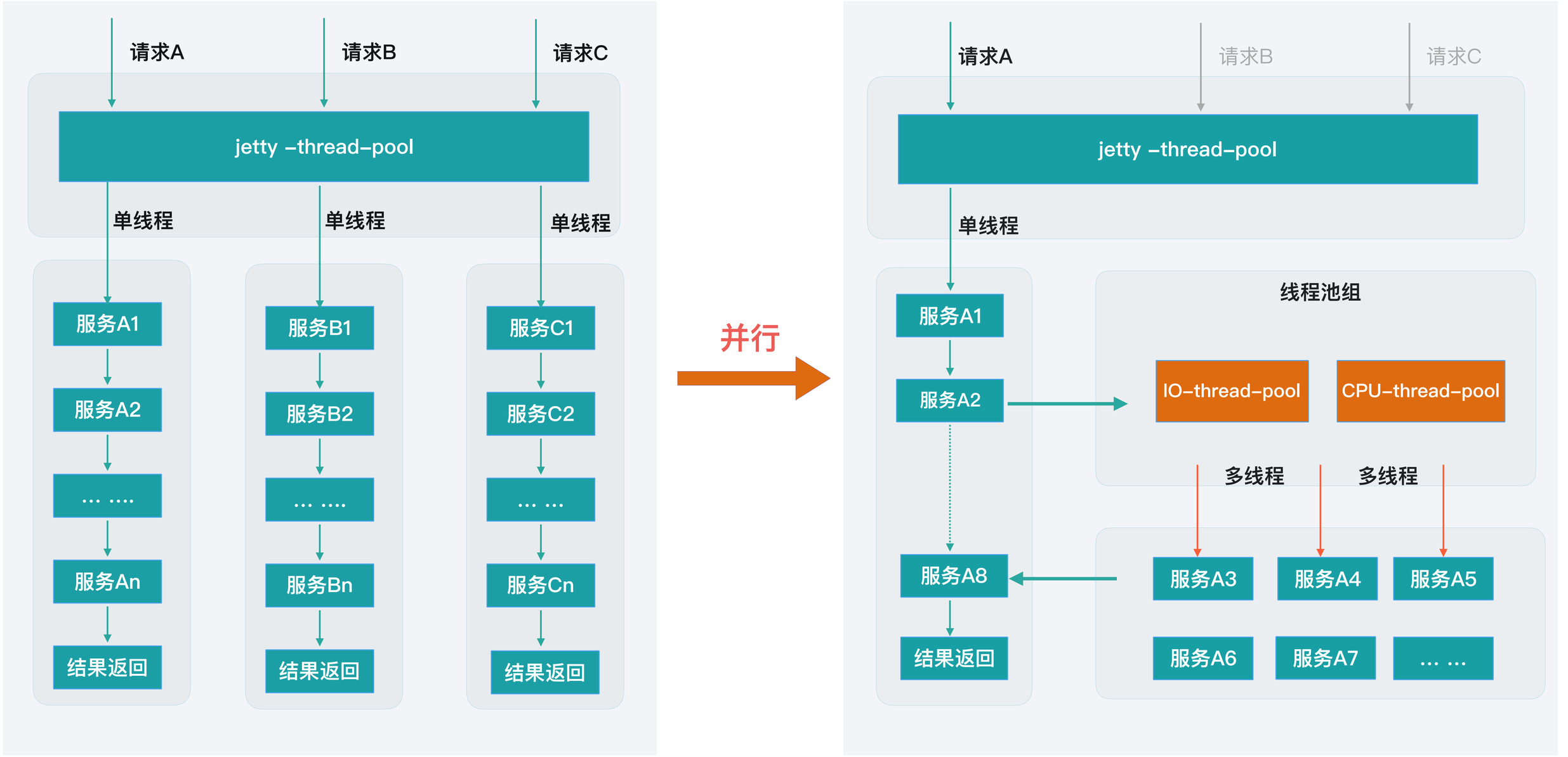
解决方案

1.分布式锁方案

2.Redis的SETNX



并行与异步优化实践


请求A
请求B
请求C
jetty –thread–pool
单线程
单线程
单线程
服务A1
服务A2
… ….
服务An
结果返回
服务B1
服务B2
… ….
服务Bn
结果返回
服务C1
服务C2
… …
服务Cn
结果返回
并行
请求A
请求B
请求C
jetty –thread–pool
单线程
服务A1
服务A2
服务A8
结果返回
线程池组
IO–thread–pool
CPU–thread–pool
多线程
多线程
服务A3
服务A4
服务A5
服务A6
服务A7
… …

并行与异步优化实践

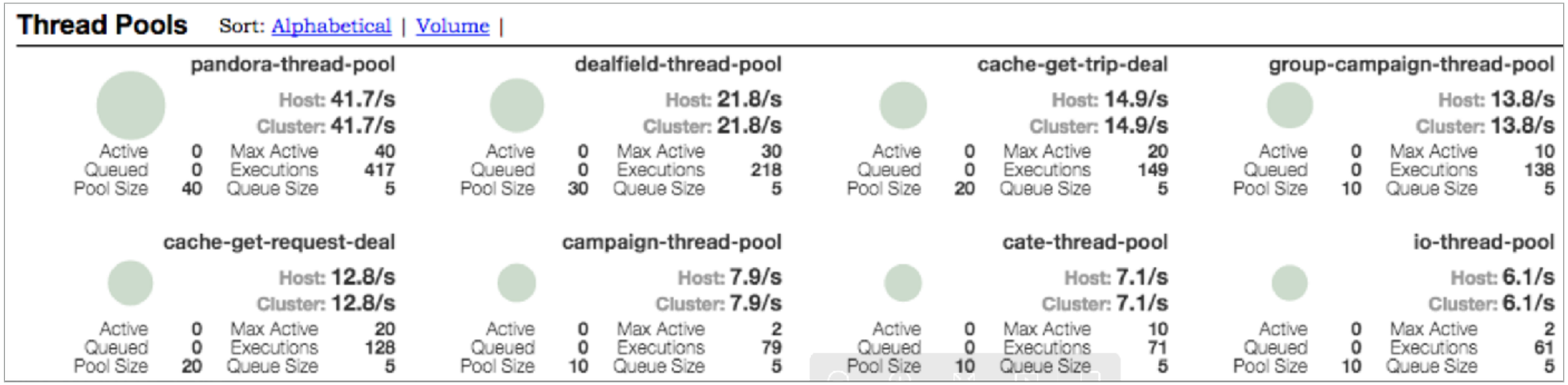
线程池优势

- ✦ 系统响应延迟得到很大的改善
- ✦ 资源利用率得到了提升
- ✦ 节省了不少线程创建和销毁的开销

线程池弊端

- ❖ 线程池大小 队列长度 隔离程度 不易控制
- ❖ 过多上下文切换耗CPU
- ❖ 线程池数量高 耗内存
- ❖ 线程池的漏斗效应
- ❖ 单机并发能力仍然受限

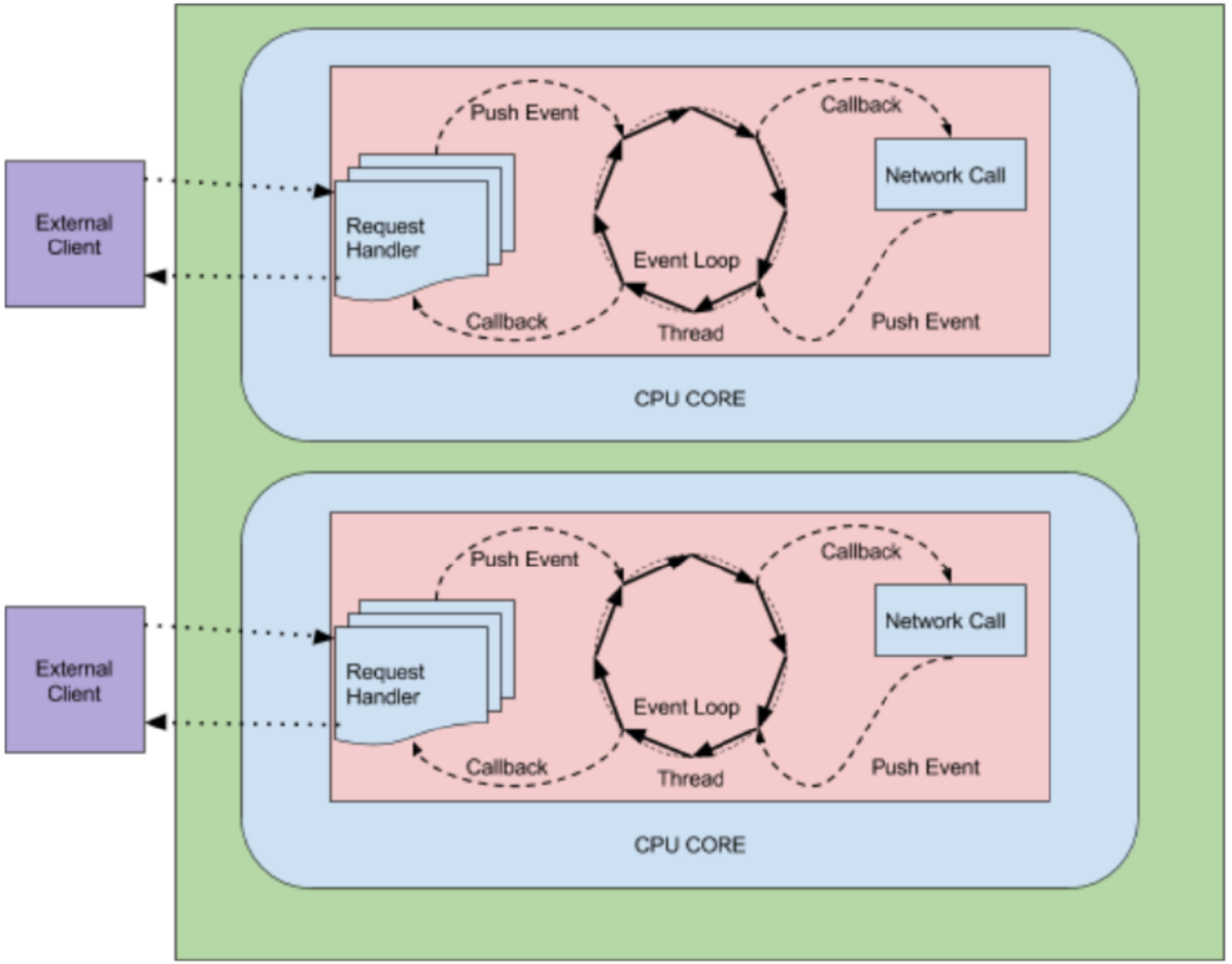
图来源：弹性应用工作平台（https://resilience.meituan.com/）



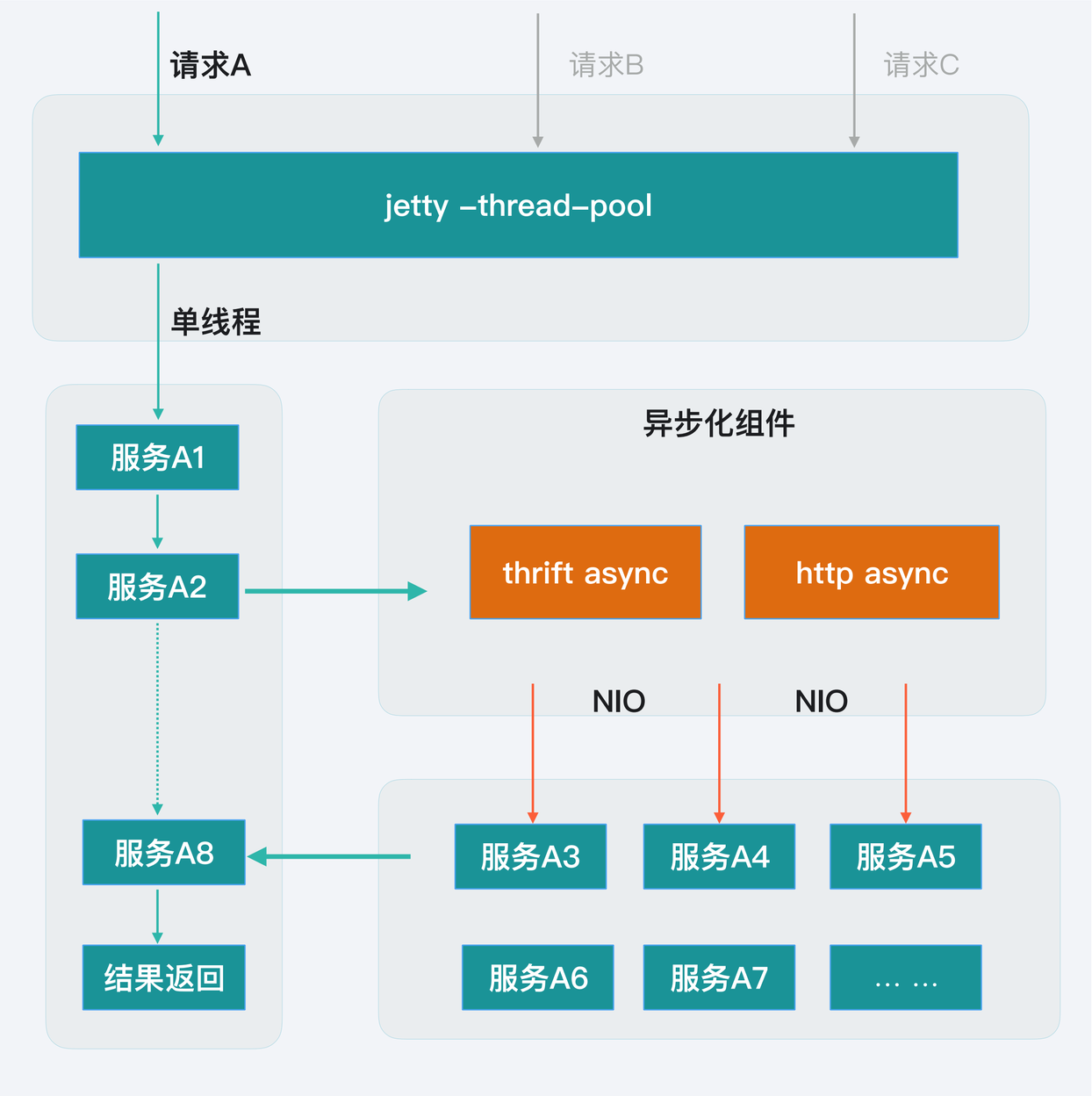
优化前线程池达20多个

并行与异步优化实践

事件驱动



并行与异步优化实践



NIO异步化实现并行

- 事件驱动模型
- 单线程处理多任务
- 线程数下降
- 单机吞吐量提升

优化后线程池达仅有4个

Thread Pools Sort: Alphabetical | Volume |

io-thread-pool				cache-requestMode/poiTag			
Host: 19.5/s				Host: 15.7/s			
Cluster: 19.5/s				Cluster: 15.7/s			
Active	1	Max Active	6	Active	0	Max Active	14
Queued	0	Executions	195	Queued	0	Executions	157
Pool Size	80	Queue Size	5	Pool Size	30	Queue Size	5

cache-trip-poi/deal-model				cpu-thread-pool			
Host: 11.6/s				Host: 6.4/s			
Cluster: 11.6/s				Cluster: 6.4/s			
Active	0	Max Active	9	Active	0	Max Active	10
Queued	0	Executions	116	Queued	0	Executions	64
Pool Size	50	Queue Size	5	Pool Size	10	Queue Size	5

并行与异步优化实践

基础架构组件

* RPC框架：mtthrift异步化 （未开源）
* HTTP调用：Apache HttpAsyncClient （开源）
* Cache: Cellar、Squirrel异步接口

异步化编程

* command模式
* guava ListenableFuture

并行与异步优化实践

优化前后单机容量的对比

Console输出 | 测试报告 | Falcon监控数据 | Octo监控数据 | 对比报告 | 测试配置

基线构建编号 119

指标	#119	#122
并发数	40	60↗
单施压机最大并发数	5	5
QPS	229	435↗
平均响应时间	98 ms	78 ms↘
top90响应时间	186 ms	142 ms↘
错误率	0.00%	0.00%
请求量	12596	78915↗

业务优化实践

主要方式

* 优化业务逻辑
* 精简业务对象
* 预加载
* 灵活使用小技巧 具体问题具体分析

API性能优化–业务优化实践

poi详情页业务优化实例

- 货架接口返回的deal商品数量多
- 单个deal商品的数据量大
- 网络传输慢
- 用户等待时间较长

API性能优化–业务优化实践

poi详情页业务优化实例

- **接口拆分** = simple接口 + full接口
- simple接口：只返回手机首屏数据，一般会是2~3个deal商品
- full接口：返回全量deal商品数据
- 客户端APP 并发请求两个接口，优先渲染simple接口，减少用户等待时间
- 网络加载–22%，cell绘制时间–77%

API性能优化–业务优化实践

精简业务对象

门票模型

基本数据 [10%]

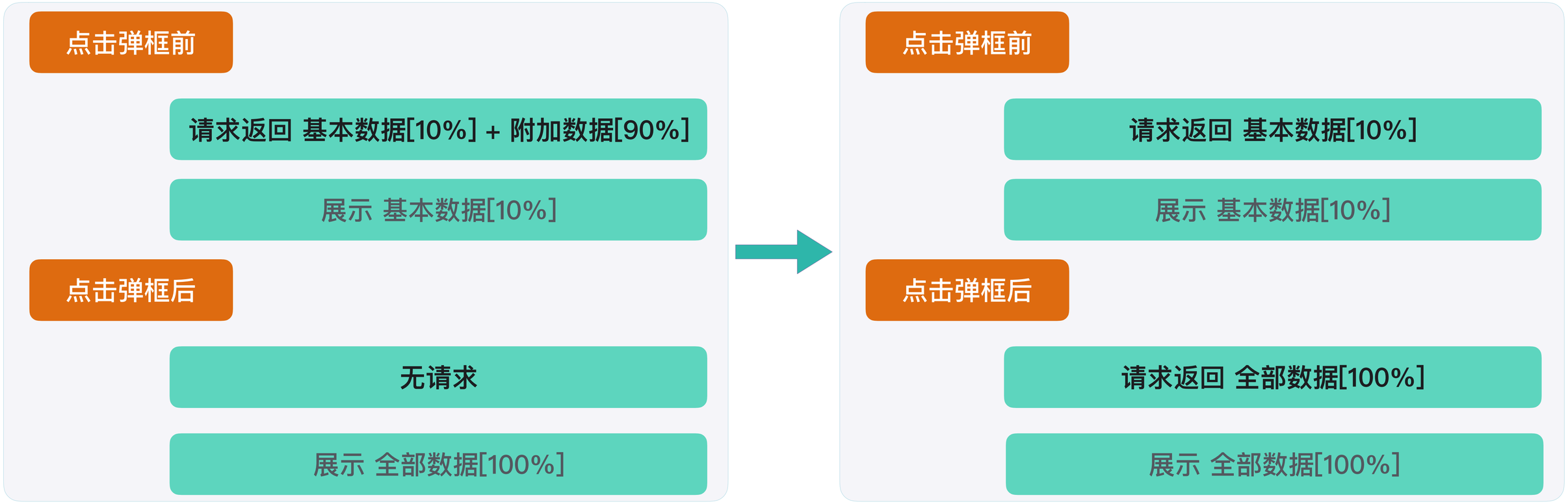
- 标题
- 价格
- 标签
- 销量
- 促销
- 票种

附加数据 [90%]

- 评价
- 关联poi
- 供应商
- 购买须知
- 安全须知
- 服务保障
- 分享
- 收藏
- 客服

点击弹出

API性能优化–业务优化实践



效果：点击弹框前的数据包返回减小了 80%左右

API性能优化–业务优化实践

预加载

API性能优化–热点优化实践

常见热点问题

* JSON解析
* 循环调用
* 签名加密
* 日志输出
* 外部jar包（容易忽视的地方）

API性能优化–热点优化实践

一个JSON问题的热点分析

解决方案：底层数据返回model类型，避免使用JSONArray.getJSONObject，或者换用别的json组件

API性能优化–热点优化实践

关键指标	优化前	优化后	总结
CPU耗时	53.7%	2.3%	降低96%
QPS	177	258	增长46%
平均耗时(ms)	138	111	降低20%

API性能优化–优化效果

优化前后响应时间的对比

角色：来源 去向　环境：prod staging test　时间粒度：天 小时　时间：2016-08-13　查询

说明：**1**，QPS，Tp99被表示为：[*数值，环比，同比*]，客户端统计量被表示为：[*调用量，失败数，失败数占比*]。**2**，耗时指标(eg. 平均耗时,TP

接口	调用量	调用量(客户端)	可用性	QPS(次/秒)	TP90	TP95	TP99	TP999
all	73391527	0, 0, 0%	0%	849,18%,1%	107	144	273	655,-3%,8%
接口	6926709	0, 0, 0%	0%	80,21%,2%	5	7	71	192,-5%,-5%
	5221965	0, 0, 0%	0%	60,17%,3%	175	247	569	928,-10%,1%

角色：来源 去向　环境：prod staging test　时间粒度：天 小时　时间：2017-08-12　查询

说明：**1**，QPS，Tp99被表示为：[*数值，环比，同比*]，客户端统计量被表示为：[*调用量，失败数，失败数占比*]。**2**，耗时指标(eg. 平均耗时,TP

接口	调用量	调用量(客户端)	可用性	QPS(次/秒)	TP90	TP95	TP99	TP999
all	192941995	192941995, 0, 0%	100%	2233,NaN,-6%	53	76	123	234,,-3%
接口	31284294	31284294, 0, 0%	100%	362,NaN,1%	50	57	83	175,,-16%
	15072181	15072181, 0, 0%	100%	174,NaN,-5%	3	4	10	12,,-8%

总结

性能优化是一门令人激动的、富于变化同时又充满挑战的学科。

—《性能之巅》

Q&A