微服务监控与优化

About me

刘地生

现任职于融数数据，负责微服务平台的设计、研发
曾就职于去哪儿、百度

对分布式系统、devops、机器学习有浓厚的兴趣

Agenda

微服务监控

- 微服务
- 为什么监控
- 怎么监控

Java栈监控机制

- CLI
- Log、SDK、AOP
- Instrument+JMX

实践及优化

- 微服务实践
- 监控实践
- java性能优化

微服务监控

微服务长什么样？

微服务架构本质

带自身特点的面向服务的

分布式架构模式

微服务架构特征

更细粒度服务边界

独立开发、测试、部署、扩展等等

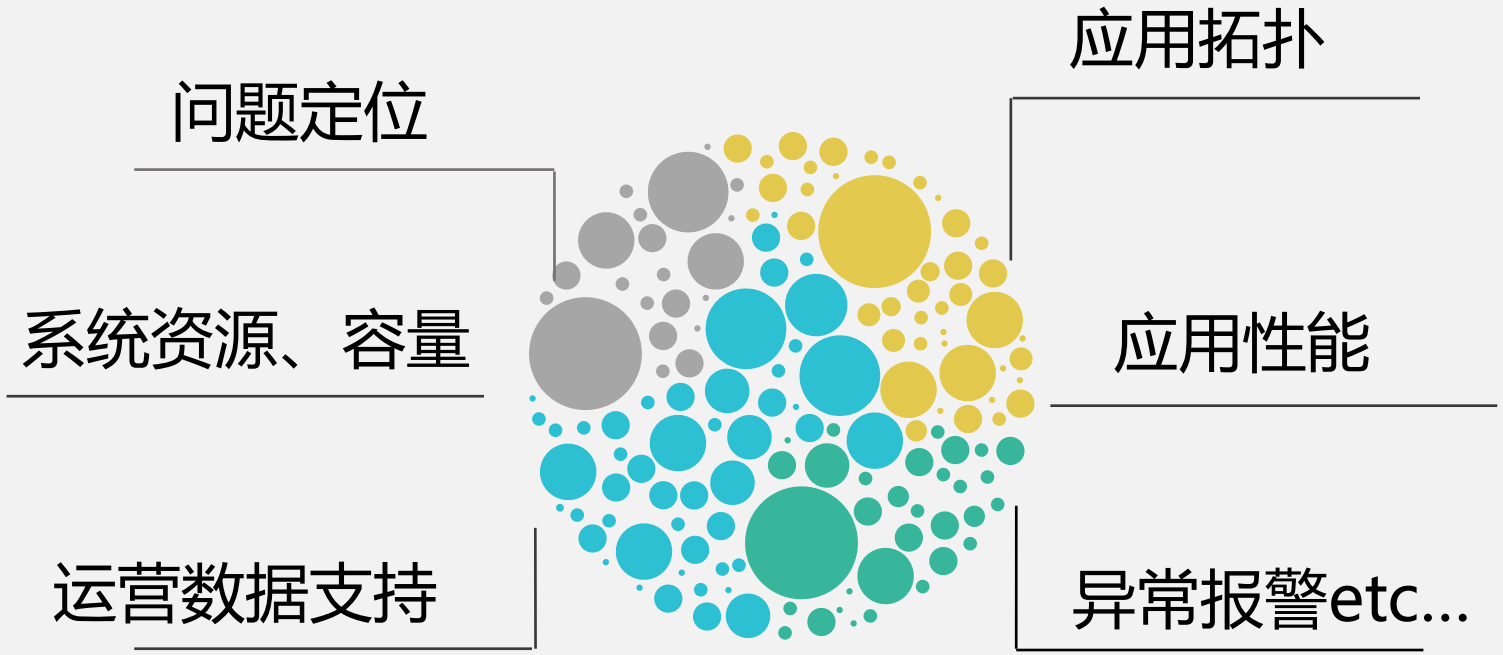
更细粒度带来的敏捷提升

分布式系统固有的复杂性

微服务监控

为什么需要监控？

Hold的住
不垮掉



微服务监控

怎么监控？数据驱动

微服务监控

常用监控手段

- Zabbix、ELK、Zipkin
- Prometheus、Pinpoint 、Newrelic
- 撸起袖子自己干

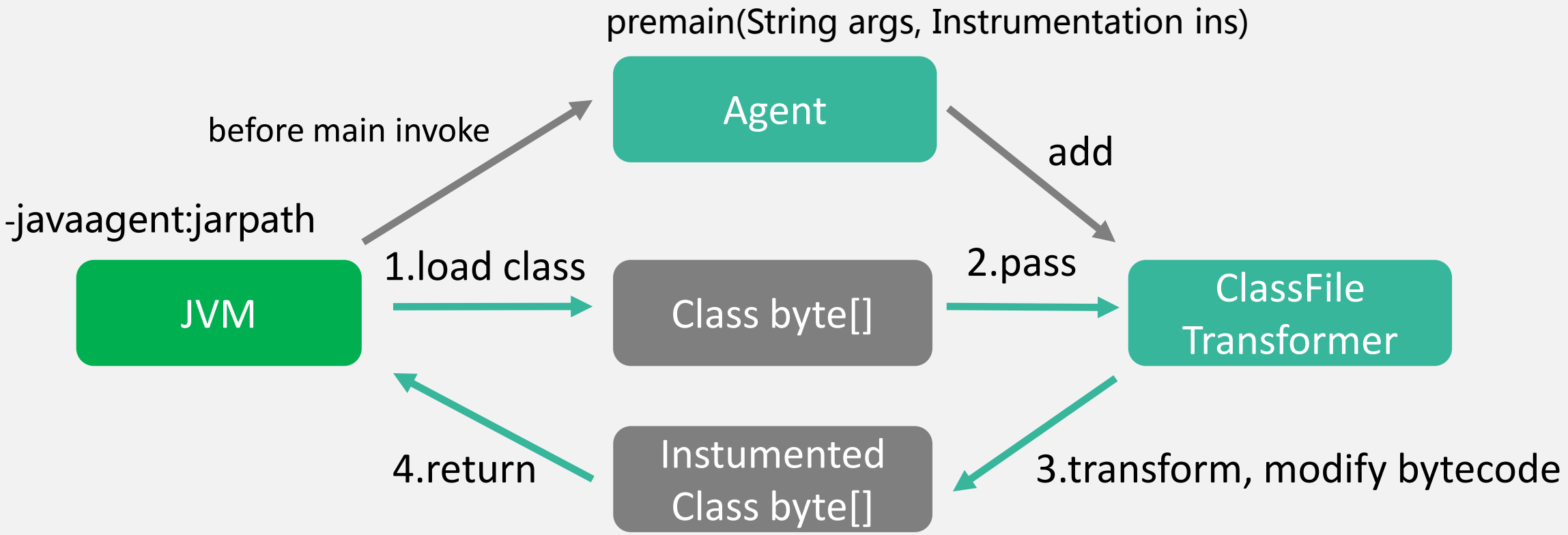
采集、接收、处理、展现

Java栈监控机制

- Command line tools
- Log、SDK、AOP
- Instrument + JMX

Java栈监控机制

Instrument机制

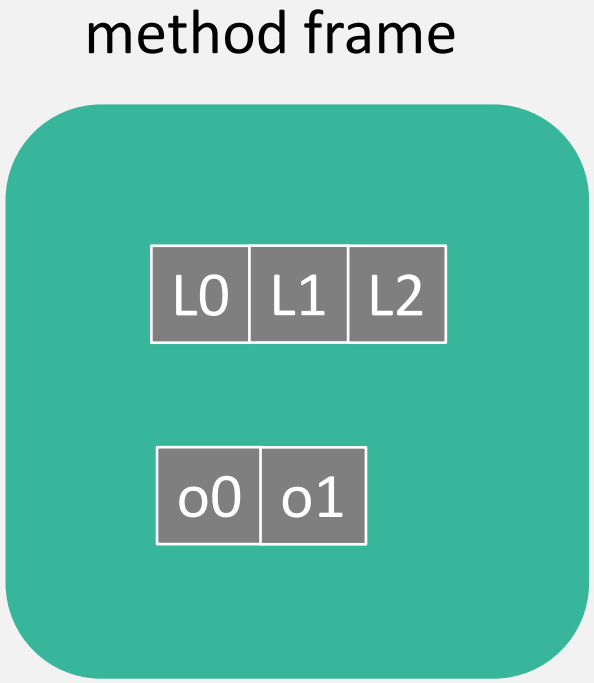


Java栈监控机制

bytecode获取方法执行时的数据

获取方法返回值示例

```
ALOAD  0
ACONST_NULL
ASTORE varindex
……
DUP
ASTORE varindex
ARETURN
```



Java栈监控机制

JMX机制获取JVM、OS相关数据

ManagementFactory.getXXXMXBean();

OperatingSystemMXBean
RuntimeMXBean
MemoryMXBean
ThreadMXBean
Collection<GarbageCollectorMXBean>

实践及优化

微服务实践

思路：围绕微服务的开发、部署、调用、通信、业务处理过程

开发部署	工具链 + 插件代码生成
调用	环境透明初始化、开箱即用
通信	数据压缩 + 长连接+事件异步
使用	框架透明、减少或消除依赖
运维监控	接入整合监控平台

实践及优化

监控采集端实践

思路：在有限资源内实现高效

无侵入 Instrument + JMX

可配置 Config reload

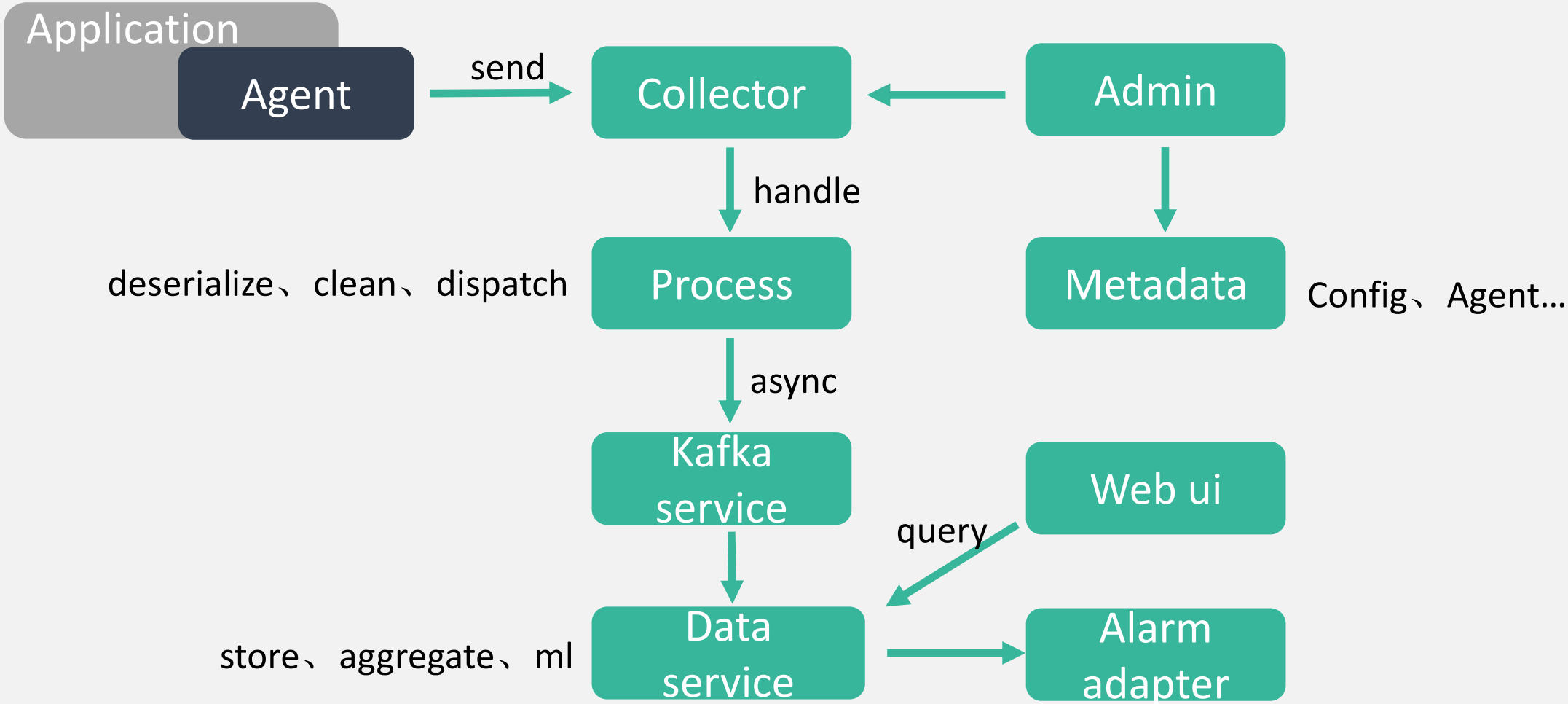
微内核 Core + plugin

插件可拔插 Classloader + config

高性能、异步 Limit CPU、Mem + thread + sample

实践及优化

监控整体架构



实践及优化

换个视角看性能优化

比优化更困难的是发现问题

没有条件或目标的优化都是耍流氓

追求对资源的高效利用

实践及优化

Java性能优化：针对特定问题的常见代码优化

- 文件io操作，IO操作使用buffer

  动机：减少内核级调用、减少IO操作、可能减少CPU指令

- 多线程环境，并行、减少锁竞争

  动机：解放单线程限制、获取CPU核数带来的计算能力的扩展

- 使用jdk collection，数据结构指定大小

  动机：减少扩容带来的内存占用、及其复制和老数据回收带来的CPU指令

- 调整算法

  动机：在执行每一次任务时，减少或优化CPU指令

实践及优化

Java性能优化：jvm调优，时间空间运维的权衡

- 调整heap大小

  动机：应用稳定状态下新生代、老年代、方法区大小。大小的调整进而影响到gc的行为

- 更换GC

  动机：面向响应时间、吞吐量，终极目标针对CPU

- 启停其他特定参数

  动机：针对特定场景，如对验尸、逃逸分析、JIT支持

实践及优化

Java性能优化：jvm调优步骤

- 根据gc日志计算出应用长期存活对象(老年代、永久代)的大小
- 建立heap基准大小

  参考建议：基于长期存活对象大小，整堆如果限定为其大小的3-4x。

  那么新生代为其1~1.5x，老年代为其2-3x，永久代为自身大小的1.5x
- 按目标是针对吞处理、响应时间来调整GC行为

  启用相应GC收集器，调整相应新生代、老年代、永久代大小
- 按额外需求起停其他参数

饿了么 | IT大咖说 知识分享平台

THANKS