第一课：调用链系统概述与底层逻辑

主讲：鲁班

概要：

1. 分布式调用链系统概述
2. 调用链系统底层实现逻辑
3. 项目部署与启动

一、分布式调用链系统概述

讲个找BUG的故事：

客户打电话给客服说：“优惠券使用不了”。
客服告诉运营人员
运营打电话给技术负责人
技术负责人通知会员系统开发人员
会员找到营销系统开发人员
营销系统开发人员找到DBA
DBA找到运维人员
运维人员找到机房负责人
机房负责人找到一只老鼠 ， 因为就是它把网线咬断了。

分布式|微服务架构所带来的问题

定位一个问题怎么会如此复杂？竟然动用了公司一半以上的职能部门。但其实这只是当我系统变成分布式之后，当我们把服务进行细粒度的拆份之后的一小部分问题，更多问题在哪里？比如：

1. 开发成本会增加。
2. 测试成本增加。
3. 产品迭代周期将变长。
4. 运维成本增加。

产生这些问题的原因是什么

这是为什么？这跟我们的了解不一样呀，

在我们固有的认知里面，社会化分工越精细，专业化程度越高，产能就越高么？

一台汽车平均将近3万个零部件，来自全球各个供应商。你可能会说,汽车这种大物件太复杂了，一家公司搞不定，必须协作。那我们就说一个小的，一次性打火机不复杂吧，在浙江温

州 有一个打火机村，一个小小的打火机生产，是由20多个厂家协作完成，有的做打火机燃料有的做点火器。

好，我们在反观软件行业， 你见过哪家某个系统是由十几家企业协作完成的么？你觉得淘宝的电商系统可以让日本人去开发 购物车模块、让法国人实现评论模块、让印度人去实现下单功能、美国人实现商品模块，最后在由中国人拼装整合，可能嘛？

我想一下原因 其实就是在于三个字：“标准化”，在刚说的汽车3万个零件，每个都有其标准化规格，所以才能够顺利的拼装成品，但软件你能标准化至表格这么细么，我们连开发个接口都没有指定标准。接口命名、参数、结构等没有指定的标准，最多就是一个规范。

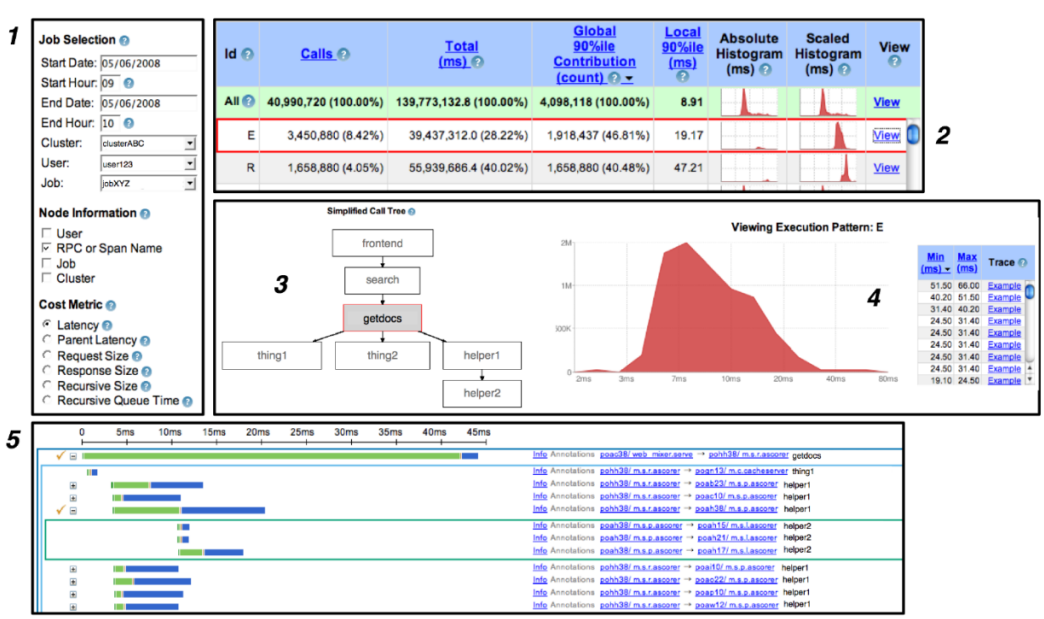
没有标准化，不能分工协作，那怎么实现软件的大规模生产呢？管理办法升级加工具升级，管理办法是类似于敏捷开发、工程师文化建设、开发形为准则。另外一个就是工具：像自动化构建、自动化部署、自动化运维、自动化扩容等、线上监控等。还有调用链追踪都属于工具

调用链追踪工具的作用

分析业务实现
分析用例的性能瓶颈
决策支持
定位线上问题

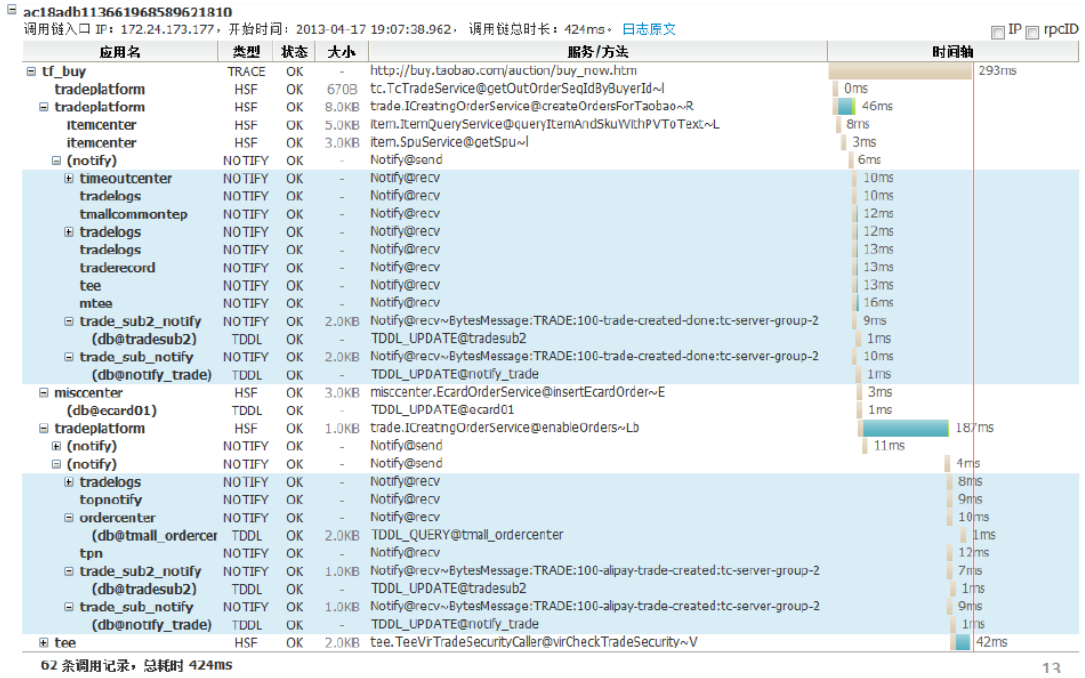
大厂的调用链系统发展史

- Dapper

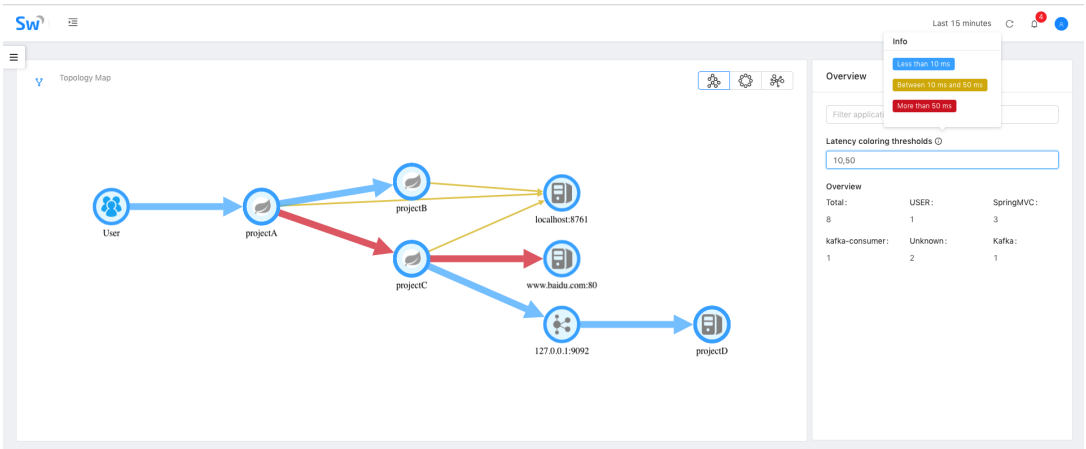


关于Dapper论文链接：https://www.jianshu.com/p/7c719a75df8c

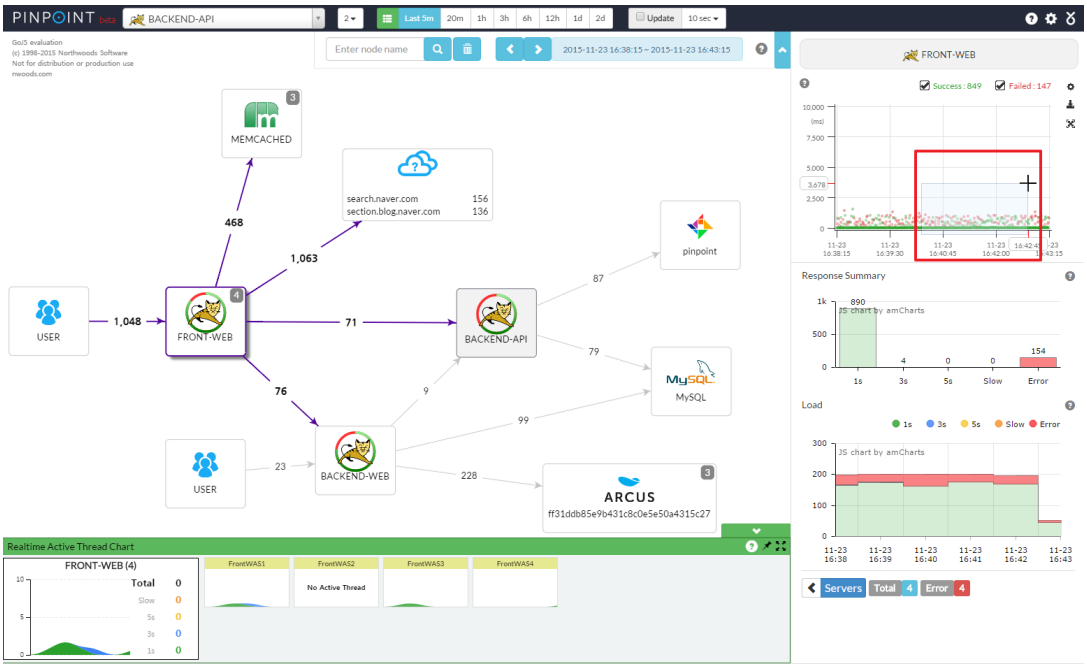
- 鹰眼



- skywalking



- pinpoint



图灵调用链项目演示：

- [ ] 基本功能
- [ ] 细节功能
- [ ] 搜索功能

二、调用链系统的底层实现逻辑

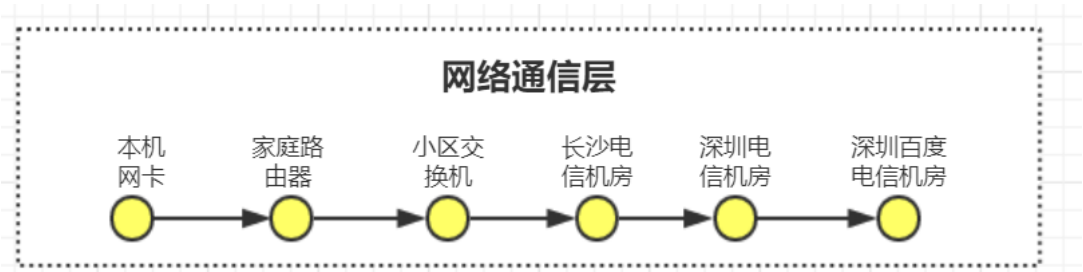
1、核心概念：事件

一次网络请求的完整过程：

一张网页，要经历怎样的过程，才能抵达用户面前？其可分为三个层面

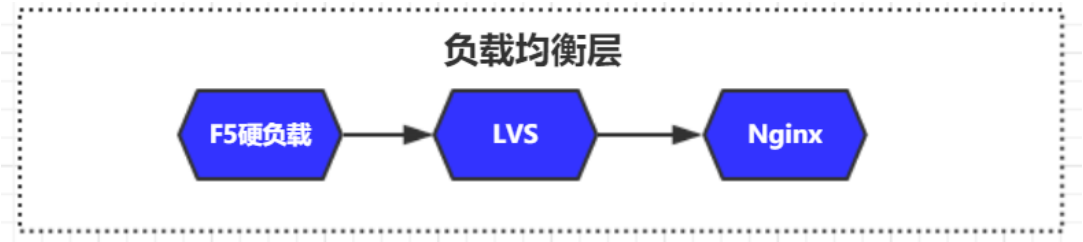
1. 网络通信传输层
2. 负载均衡层
3. 系统服务层

- 网络传输层

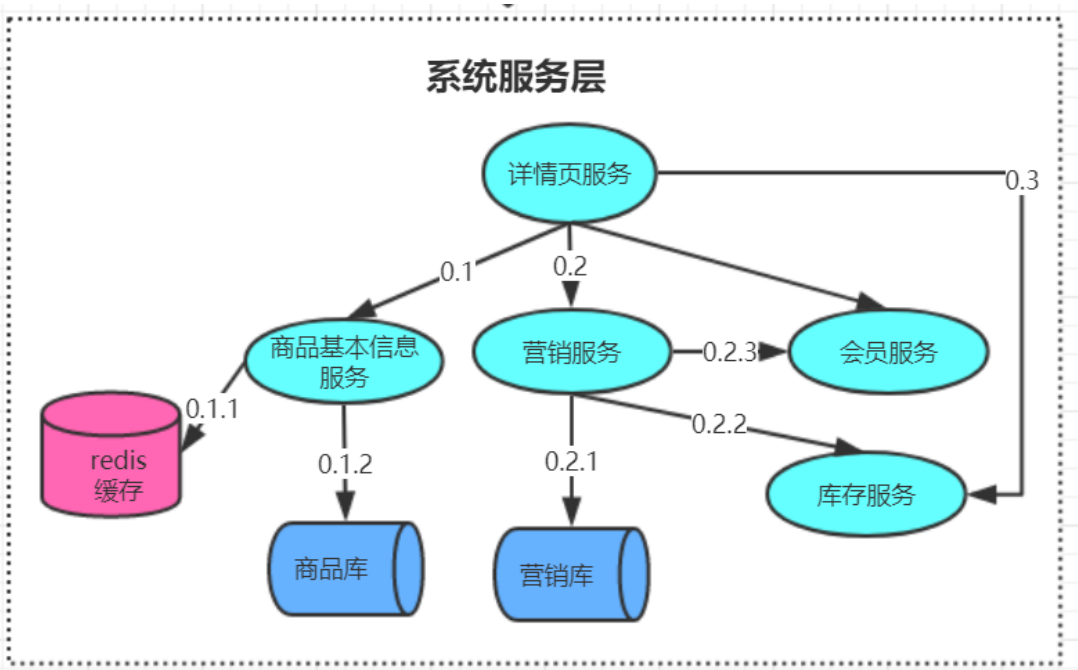



```
# 该命令可直接跟踪网络通信所经过的节点
tracert www.baidu.com
```

- 负载均衡层



- 系统服务层



传统的APM(应用性能管理系统)系统关注于前两个层面，而现在的调用链追踪系统则测重于对系统服务层面性能及可用性进行进行追踪监控。其本质上就是用来回答以下问题:

1. 什么时间?
2. 在什么节点上?
3. 发生了什么事件?
4. 这个事件由谁触发?

所以调用链系统的核心概念即是事件。关于事件及其属性说明如下：

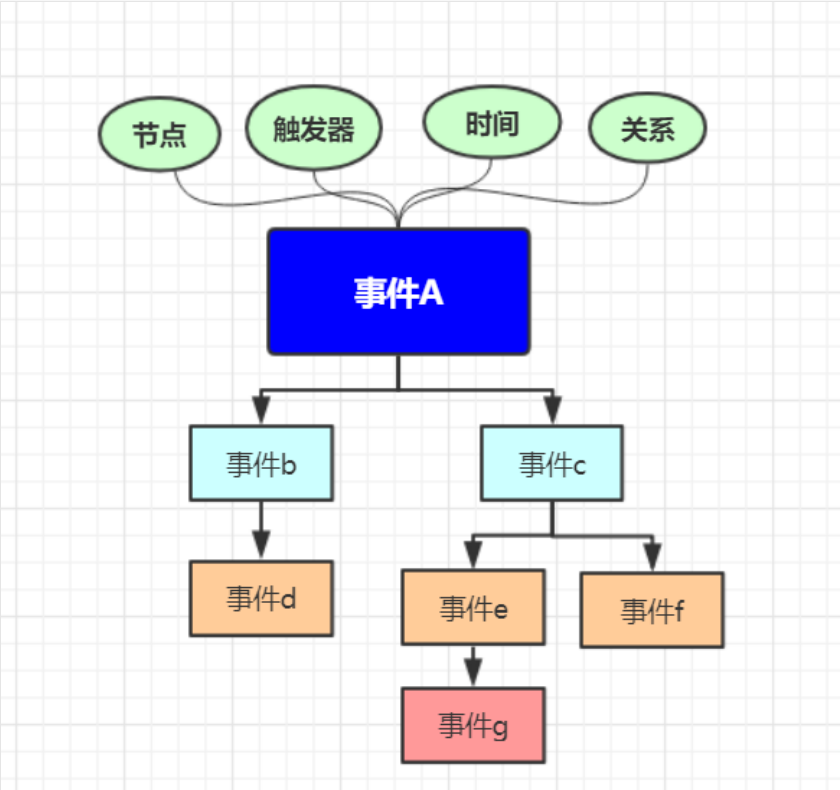
事件：请求处理过程当中的具体动作如：WEB 请求、方法调用、SQL执行、远程调用、消息发送。

节点：请求所经过的系统节点，即事件的空间属性。如：应用系统、数据库实例、缓存节点、外部服务等

时间：事件的开始和结束时间

关系：事件与上一个事件关系

触发:可以是用户触发、定时器触发、远程调用触发



思考题：一次Http请求事件 包含哪些具体属性及子事件。

2、事件捕捉

我们如何采集这些事件的数据呢？采集数据的方式即为**事件捕捉**，有已下四种方式：

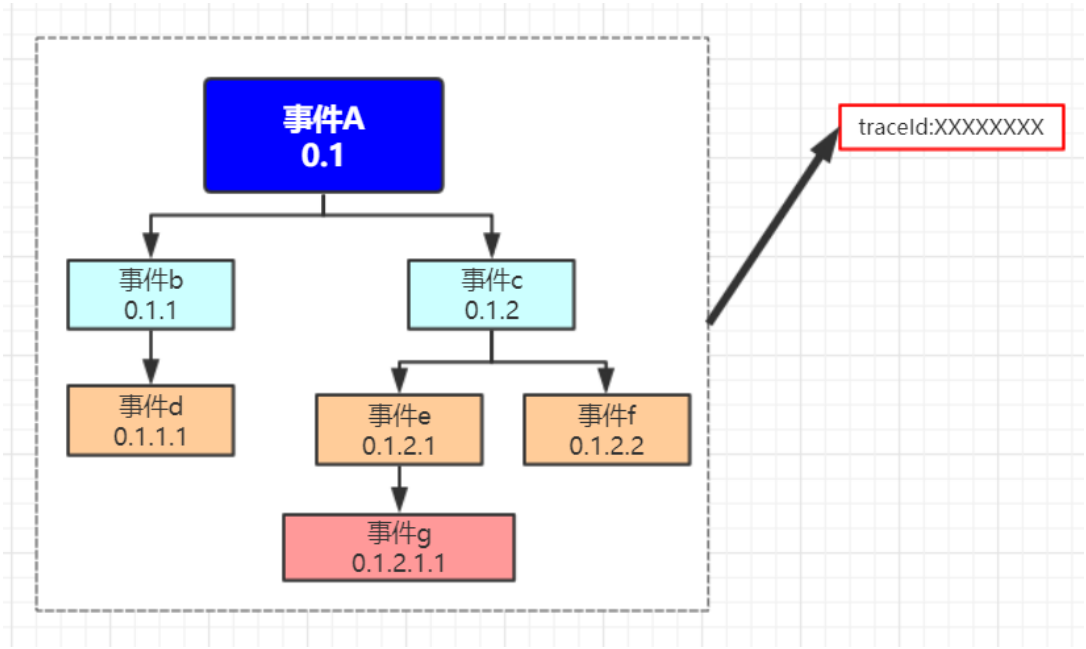
1. 硬编码埋点捕捉
2. AOP埋点捕捉
3. 公共组件埋点捕捉
4. 字节码插桩捕捉

3、事件串联

如果只给你一断SQL 而不给出这段SQL的调用上下文，基于此你做出的判断是非常有限的，无法得知SQL是哪个用例触发的？什么时候触发的。避免单个事件独立存在，就需要我们把事物进行串联:

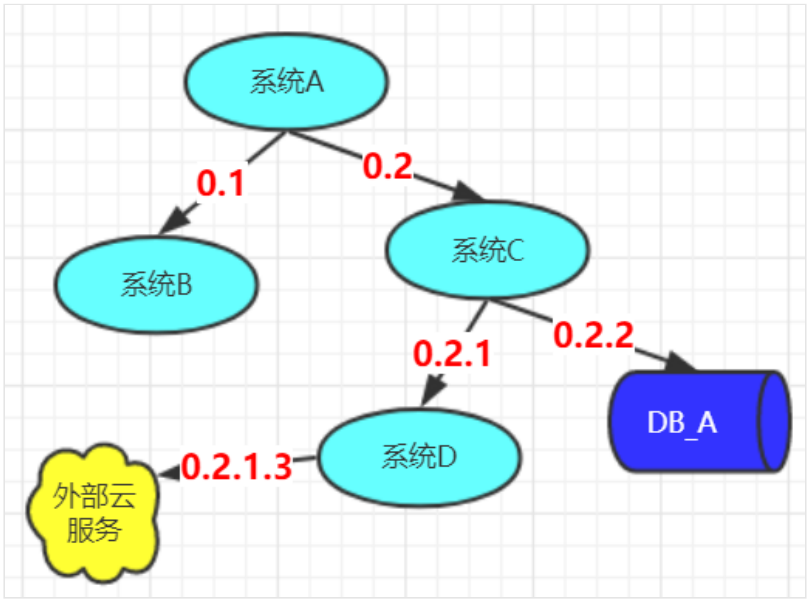
1. 所有事件都关联到同一个调用。
2. 事件与子事件基于层级关展现。

为了到达这两个目的地，几乎所有的调用链系统都会有以下两个属性：
traceID:在整个系统中唯一，该值相同的事件表示同一次调用。
event ID:在一次调用中唯一、并展出事件的层级关系

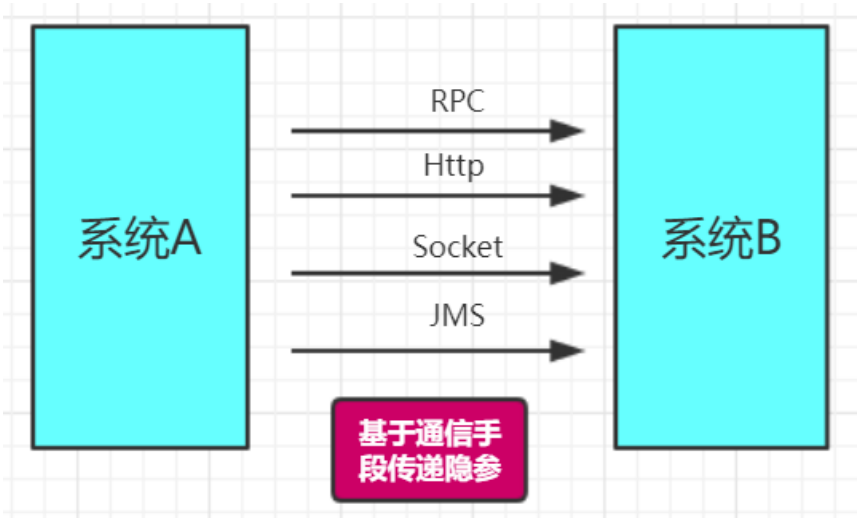


其串联的实现过程如下：

1. 在调用的初始事件生成 TraceId 与EventId 并保存于ThreadLocal
2. 当下一个事件发生的时候 直接从ThredLocal取出 TraceId 与EventId进行自增并保存。
3. 有时事件是跨节点的，这时就需要在远程调用的时候一并将TraceID 与EventId 一并传送过去并保存自ThreadLocal。



trackId与eventId 基于远程调用的传递



4、事件的开始与结束

我们知道一个事件是一个时间段内系统执行的若干动作，所以对于事件捕捉必须包含开启监听和结束监听两个动作？如果一个事件在一个方法内完成的，这个问题是比较好解决的，我们只要在方法的开始创建一个Event对象，在方法结束时调用该对像的close 方法即可。

```
public void addUser(){
  // 方法的开始处，开启一个监听
  Event event=new Event();
    //业务代码执行
    .....

    .....
  // 方法的结束处，关闭一个监听
  event.close();
}
```

但如果一个事件的开始和结束触发分布在多个对象或方法当中，情况就会变得异常复杂。比如一个JDBC执行事件，应该是在构建 Statement 时开始，在Statement 关闭时结束。怎样把这两个触发动作对应到同一个事件当中去呢（即传递Event对象）？在这里的解决办法是对返回结果进行动态代理，把Event放置到代理对象的属性当中，以达到付递的目标。当这个方法只是适应JDBC这一个场景，其它场景需要重新设计Event 传递路径，目前还没有通用的解决办法。

```
// JDBC事件开始
Connection.prepareStatement(String sql);
//JDBC 事件结束
PreparedStatement.close();
```

三、项目部署与启动

概要：

1. 项目整体架构介绍
2. 项目工程结构介绍

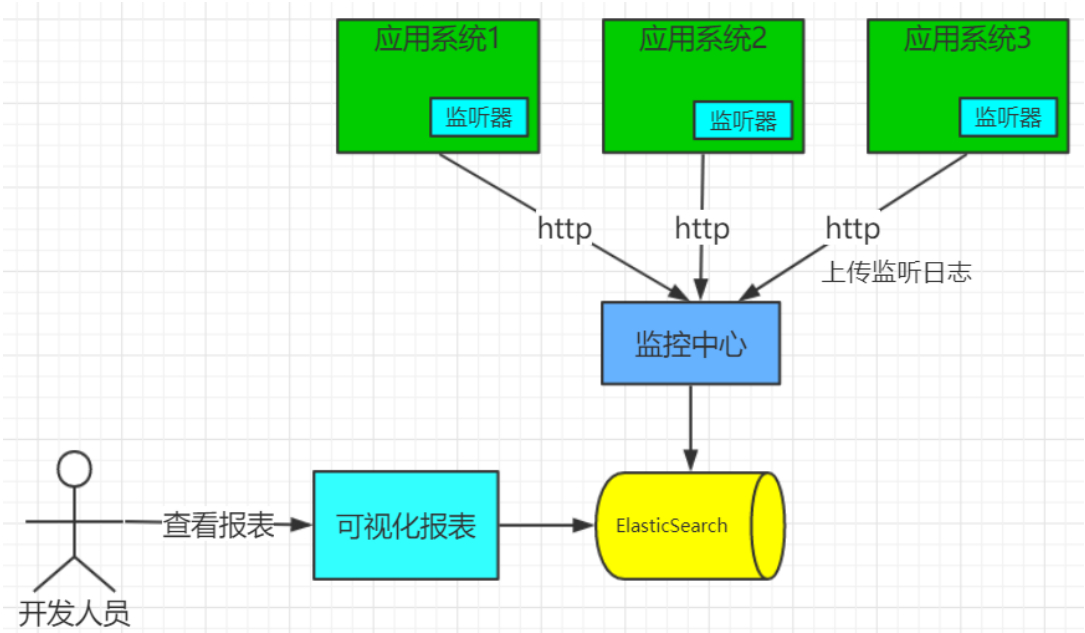
3. 项目部署启动演示

1. 项目整体架构介绍

架构目标：

1. 基本功能：实现监控数据采集上报与展示
2. 可扩展性：能够非常轻松灵活的添加新的采集器
3. 可运维：Agent自动更新、异常日志输出

基本功能

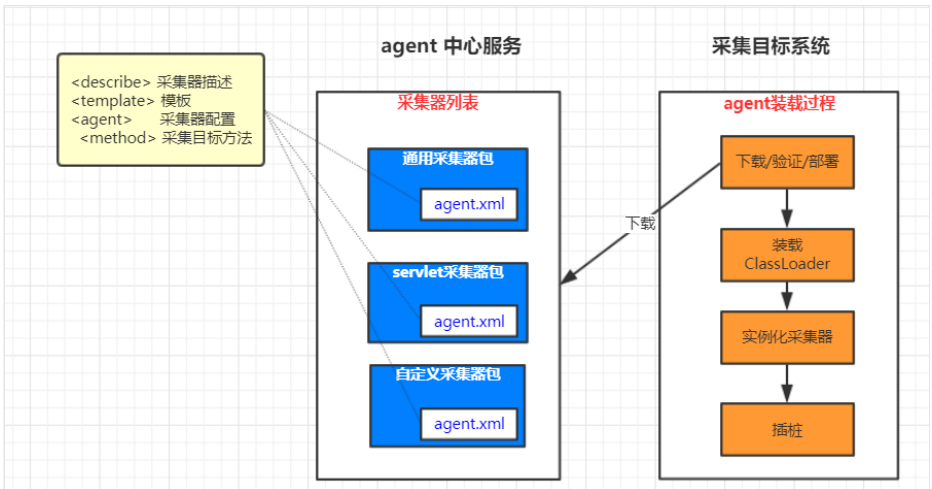


全局架构说明：

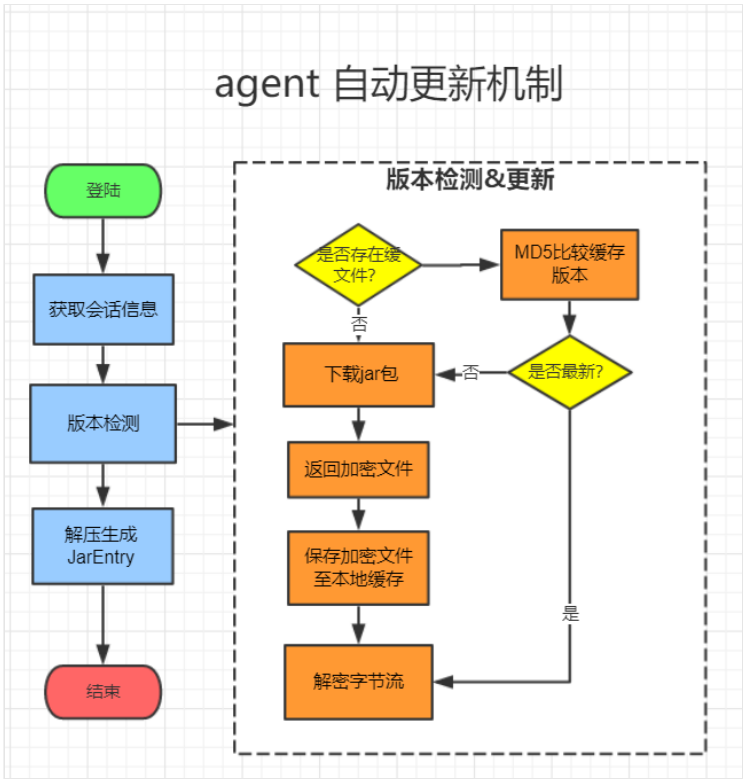
1. 监听器基于埋点采集，接口、SQL、Servlet等响应信息
2. 采集的数据基于后台线程 以Http的方式发送至监控中心
3. 监控中心实时上传至ElasticSearch
4. 可视化展示调用链详情

可扩展性：

插件机制、灵活自定义扩展，其过程见下图。



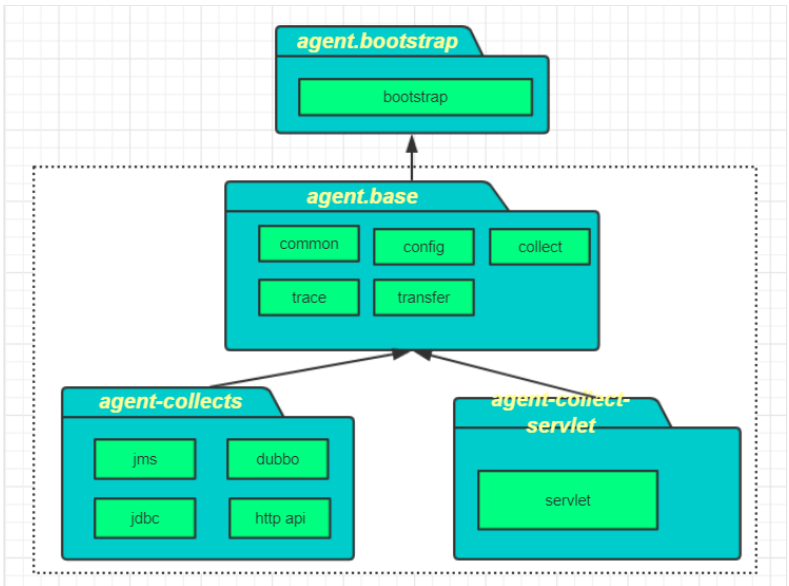
可运维:
插件自动更新.



2、项目工程结构介绍

项目整体分为两个工程:
1、Agent 工程:
git地址:http://git.tulingxueyuan.cn/java-vip/tuling-trace-agent
2、server 工程:
git地址:http://git.tulingxueyuan.cn/java-vip/tuling-trace-service
项目依懒:mysql、ElasticSearch

Agent项目工程结构:



3.项目部署启动演示

- [ ] 启动服务端
- [ ] 启动测试客户端

调用链Agent 如何部署:

1. 从源码中心http://git.tulingxueyuan.cn/java-vip/tuling-trace-agent/src/master/bin/agent.zip下载agent.zip 至应用系统
2. 解压缩 agent.zip
3. 添加jvm 参数 -javaagent:<cbt-agent-bootstrap-1.0-SNAPSHOT.jar 路径>
4. 重启应用,观察你的应用日志，如果发现以下日志代表启动成功了

```
[2018-06-12:10:32:42]加载藏宝图配置文件来自G:\git\cbt-agent\out\conf\cbt.properties
[2018-06-12:10:32:43]藏宝图服务登陆成功!
[2018-06-12:10:32:43]藏宝图服务启动成功!
```