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. 调用链项目的使用

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

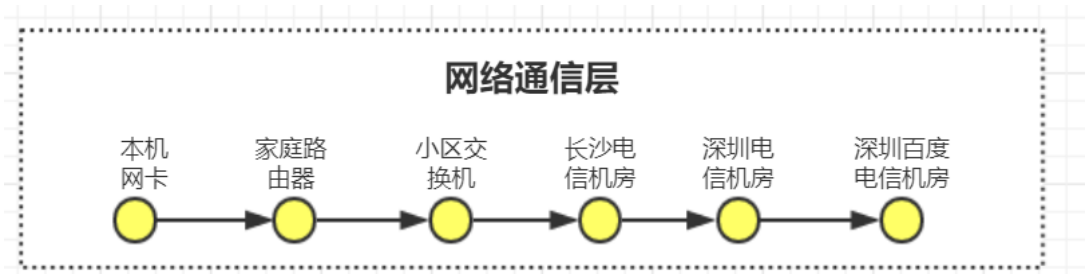
Dapper===
鹰眼
hydra
cat
zipkin
skywalking

二、调用链系统的底层实现逻辑

调用链系统的本质

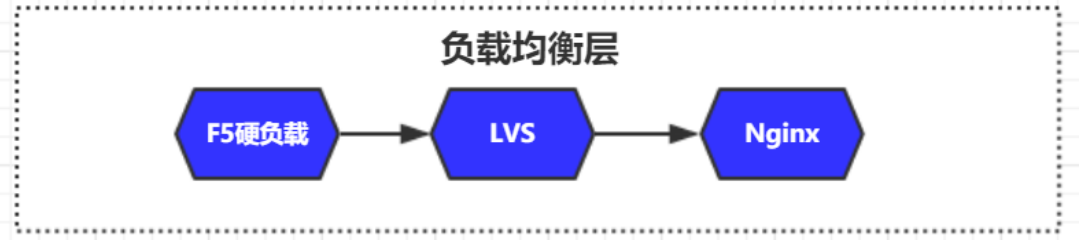
一张网页，要经历怎样的过程，才能抵达用户面前？

网络传输层

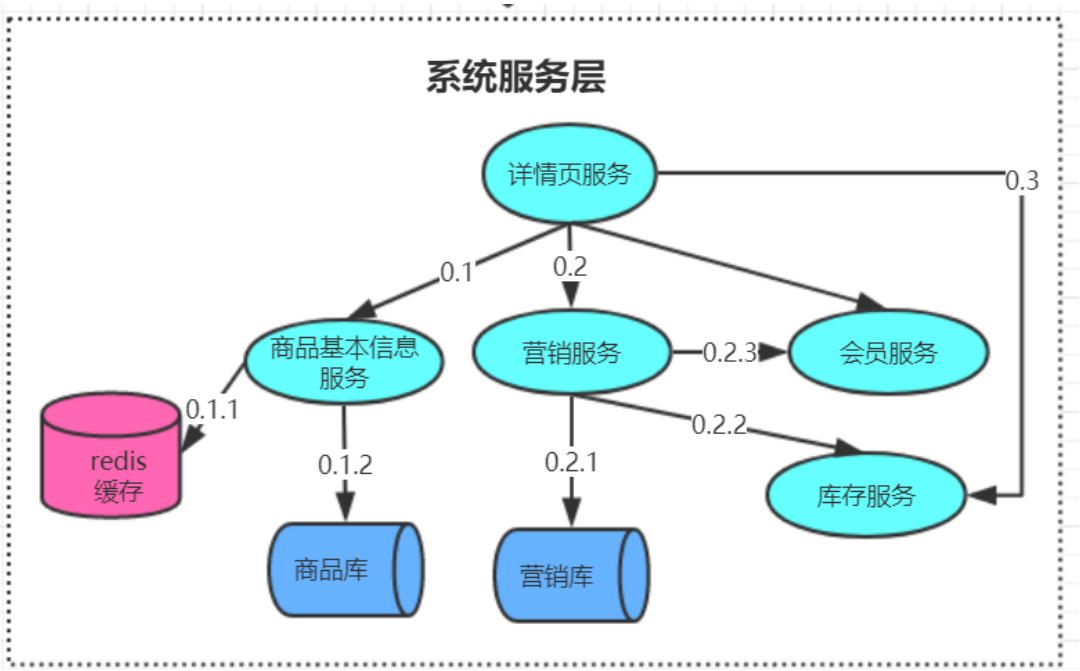



```
# 该命令可直接跟踪网络通信所经过的节点
tracert www.baidu.com
```

负载均衡层



系统服务层



调用链基本元素:

1. **事件：**请求处理过程当中的具体动作
2. **节点：**请求所经过的系统节点，即事件的空间属性。
3. **时间：**事件的开始和结束时间
4. **关系：**事件与上一个事件关系

调用链系统本质上就是用来回答这几问题:

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

事件捕捉

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

事件串联

事件串联的目的:

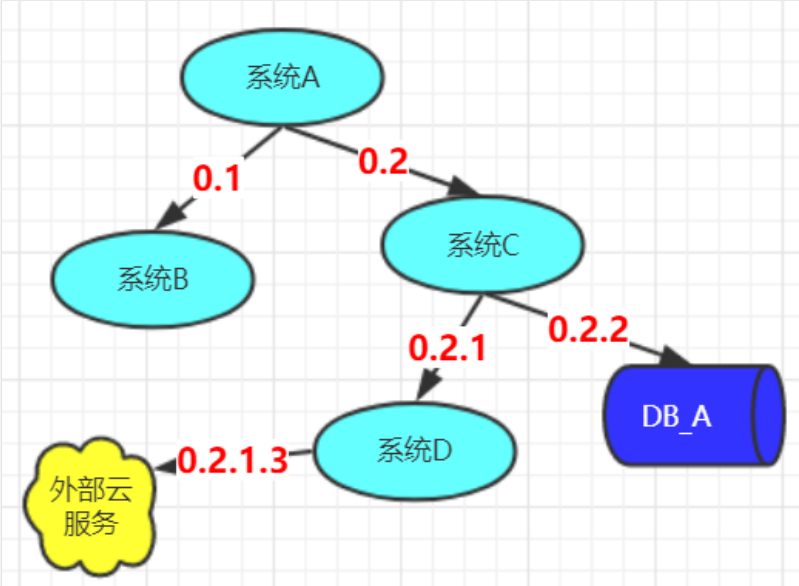
1. 所有事件都关联到同一个调用

2. 各个事件之间是有层级关系的

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：

trackID:在整个系统中唯一，该值相同的事件表示同一次调用。
eventID:在一次调用中唯一、并展出事件的层级关系

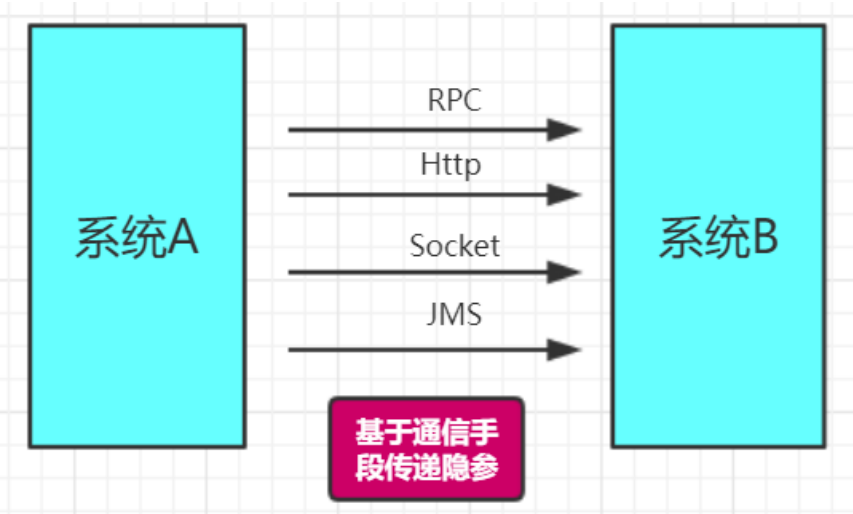
1、怎么生成TrackID
2、怎么传递参数
3、怎么并发情况下不允响传递的结果



串联的过程：

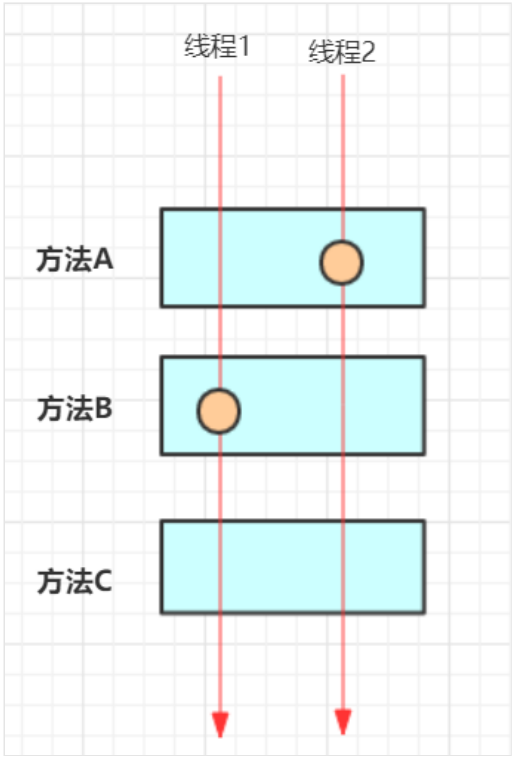
1. 由跟踪的起点生成一个TrackId， 一直传递至所有节点，并保存在事件属性值当中。
2. 由跟踪的起点生成初始EventId，每捕捉一个事件ID加1，每传递一次，层级加1.

trackId与eventId 的传递



eventId 自增生成方式

我们的埋点是埋在具体某个实现方法类，当多线程调用该方法时如何保证自增正确性?



解决办法是每个跟踪请求创建一个互相独立的会话，EventId的自增都基于该会话实现。通常会话对象的存储基于ThreadLocal实现。

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();
```

上传

上传有两种方式

1. 基于Http请求直接上传
2. 打印日志，然后在基于Flume或Logstash采集上传。

第一种相对简单，直接把数据发送服务进行持久化，但如果系统流量较大的情况下，会影响系统本身的性能，造成压力。第二种相对复杂，但可以应对大流量，通常情况下会采用第二种解决办法

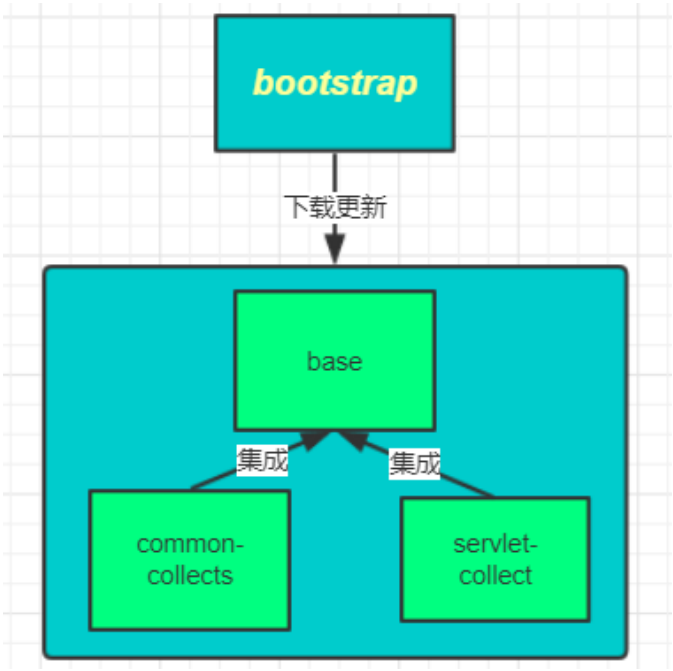
三、项目部署

调用链Agent 如何部署:

1. 从课程目录网盘下载 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

部署原理:

JAR包构成:



自动更新机制:

作业安排：

1. 将一套系统集成调用链Agent，并在调用链提供的后台观察系统的形为，找出你认为不合理的地方。

环境要求：

a. 部署项目：任意WEB项目，最好用Tomcat来启动
b. 环境要求：linux\windows\mac
c. JVM要求：JVM1.7 及以上

部署步骤：

1. 从课程网盘下载Agent.zip 并解压
2. 在你启动项目的JVM参数里添加 -javaagent:cbt-agent-bootstrap-1.0-SNAPSHOT.jar 完整路径
3. 重启你的应用，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!
```

4、登陆 调用链管理WEB页面 http://client.cbtu.pro:9978/trace/requests

2. 设计并绘制监听器的采集过程图，图中必须表现调用关系是如何整合成链条的。

Tommy

湖南 长沙