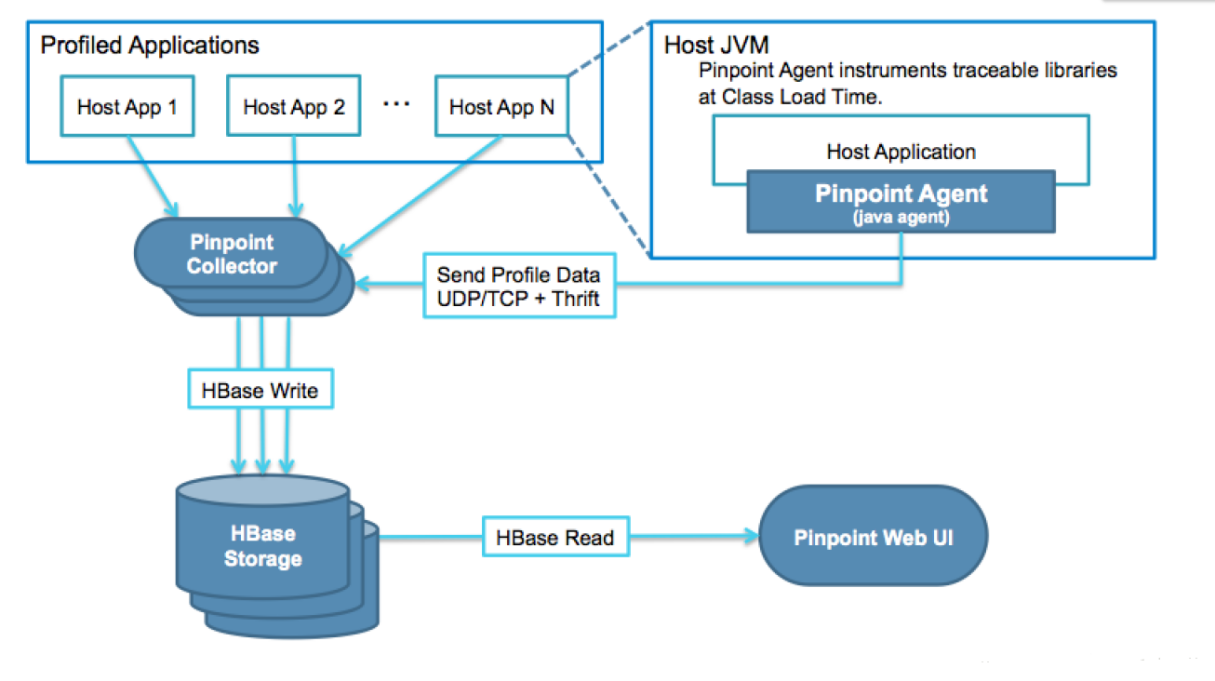
分布式跟踪工具Pinpoint技术

Pinpoint是对Java编写的大规模分布式系统APM(Application Performance Management)工具，是基于Dapper的分布式跟踪系统。前台向后台发起一个查询请求，后台服务可能会调用多个服务，每个服务可能又会调用其他服务，最终将结果返回，汇总到页面上。

Pinpoint是完整的性能监控解决方案，从探针、收集器、存储到Web界面的全套体系。而Zipkin只侧重收集器和存储服务，虽然也有用户界面，但其功能与Pinpoint不可同日而语。在Pinpoint中提供Java Agent探针，通过字节码注入的方式实现调用拦截和数据收集，可以做到真正的代码无侵入，只需要在启动服务器的时候添加一些参数，就可以完成探针的部署，而Zipkin的Java接口实现Brave，只提供基本的操作API，如需要与框架或者项目集成的话，就需要手动添加配置文件或者增加代码。Pinpoint系统架构图如下所示：



主要有以下几个组件：

* Collector，收集Agent端发送的数据
* Agent，部署在应用中，其通过Java Agent向节点应用注入字节码，而不用直接修改源代码
* Storage，默认使用HBase
* Web UI，从HBase中读取数据并展示给用户

Pinpoint消息的数据结构包含三种类型：Span、Trace和TraceId

1. Span，最基本的调用追踪单元

当远程调用到达的时候，Span指代理该调用的作业，并且携带追踪数据。为了实现代码级别的可见性，Span下面还包含一层SpanEvent的数据结构。每个Span都包含一个SpanId。

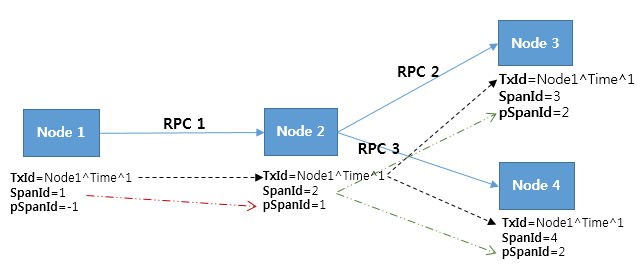
1. Trace，相互关联的Span集合

同一个Trace下的Span共享TransactionId，而且会按照SpanId和ParentSpanId排列成一棵有层级关系的树形结构。

1. TraceId，是TransactionId、SpanId和ParentSpanId的组合，TxId是一个交易下横跨整个分布式系统收发消息的ID，其必须在整个服务器组件中全局唯一。

* TransactionId，识别整个调用链
* SpanId，处理远程调用作业的ID，当调用达到一个节点的时候随即产生
* ParentSpanId，产生当前Span的调用方Span的ID，最初发起方其ParentSpanId是-1，以标志是整个交易的根Span

下图能够比较直观的说明这些ID结构之间的关系：



# PinPoint安装和使用

PinPoint使用HBase作为Span数据存储器，其安装不再介绍，从Github上下载Agent、Collector及Web安装文件。

## 安装Pinpoint Collector

Pinpoint Collector用于接收Agent收到的Span数据，并将其写入到存储系统中，其安装与配置步骤如下：

1. 下载安装文件

*wget <https://github.com/naver/pinpoint/releases/download/1.8.0/pinpoint-collector-1.8.0.war>*

1. 准备安装目录
2. 配置
3. 启动

# 源码分析

Java Agent

https://blog.csdn.net/mingyu1016/article/details/53925611

https://github.com/naver/pinpoint

http://www.tangrui.net/2016/pinpoint-plugin-development.html

http://www.tangrui.net/2016/implement-distributed-tracking-system-with-zipkin-and-brave.html