北京理工大学

本科生毕业设计（论文）开题报告

**学 院：** 计算机学院

**专 业：** 软件工程

**班 级：** 08111503

**姓 名：** 常佳辉

**指导教师：** 李鹏

**校外指导教师：**

二○一九年一月三日

毕业设计（论文）开题报告评审表

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 姓名 | |  | | 学号 |  | 班级 |  | 专业 | | |  |
| 导师 | |  | | 校外导师（职称） | |  | | 校外导师单位 | |  | |
| 论文选题 | 题目名称 | | |  | | | | | | | |
| 题目性质 | | | 软件开发（ ） 理论研究（ ）  工程设计（ ） 技术科学研究与工程技术研究（ ） | | | | | | | |
| 题目来源 | | | 结合科研 （ ） 结合生产实际（ ）  结合实验室建设（ ） 自拟题目 （ ） | | | | | | | |
| 评审组成员 | 姓 名 | | | 职 称 | | 工作单位及职务 | | | 签 字 | | |
|  | | |  | |  | | |  | | |
|  | | |  | |  | | |  | | |
|  | | |  | |  | | |  | | |
| 评  审  意  见 | （含：选题意义；选题是否满足毕业要求；技术方案是否可行；进度安排是否合理等） | | | | | | | | | | |
| 成 绩 | | |  | | | | | | | | |
| 评审组长签字： 年 月 日 | | | | | | | | | | | |

注：成绩以“合格”“不合格”记；评审组长为高级职称人员。

# 毕业设计（论文）选题的内容

毕业设计（论文）题目为“面向微服务调用的无侵入式全链路跟踪系统的设计与实现”。本课题主要研究实现面向微服务调用的无侵入式全链路跟踪系统，帮助分析系统整体调用链，快速定位性能瓶颈和业务问题；该系统支持RPC、数据库、缓存、异步消息通信，并且对业务代码无侵入，根据不同的采集需求可设置全采集、采样采集等方式；采集结果支持多样化输出，主要包括文件日志、共享内存、消息队列。最终将以可视化界面展示全链路跟踪结果，提供按照traceId字段等查询具体的某一次调用的调用链结果，输出每一次调用的具体耗时，调用关系等数据。

# 研究方案

## 2.1.本选题的主要任务

本选题的主要任务是设计实现一套公用工具，可以用于跟踪并记录微服务系统架构运行时各服务之间的调用流程，并且对业务代码无侵入式，能够在保证已在运行的系统稳定的情况下实现跟踪系统的部署。这样能够减小跟踪系统部署的人力、资源成本，但提高了系统的设计难度。跟踪系统将采集各微服务间的调用关系以及一些附加信息，这样在复现调用流程，定位问题关键会起到很大帮助。采集的信息应当能够有一个可视化界面，有利于问题的分析。因此任务项还应包括一个前端系统的设计。

## 2.2.技术方案的分析、选择

微服务架构的各模块之间调用有多种方式，比如RPC、HTTP等，因此掌握调用的协议是一项必不可少的任务。同时各服务之间采用的语言架构可能不尽相同，使跟踪系统能够拥有较大的覆盖率需要考虑全面。

### 2.2.1 项目背景

如今互联网产品迭代更新十分迅速，以往的单一应用程序的开发模式在快速的变更的需求之下逐渐显得臃肿，效率低下。在这样的大环境下出现了一种模块化的开发方式，即微服务架构开发方式。它将应用分成几个功能独立的模块，能够各自开发管理，甚至能用不同的语言编写。他们相互之间合作，共同服务一个或多个请求。这种松耦合的设计非常方便迭代开发和二次部署。然而在简易了开发流程的同时，它也面临着一些新的问题。非本地调用的特性使得它本能上就更容易受到外部的影响，因此追踪各模块的性能是不可或缺的，在对问题的追踪并解决上会使系统运行的更加稳定。目前较成熟的有ZeroC IceGrid、Spring Cloud、基于消息队列与Docker Swarm四种微服务架构方案。

### 2.2.2研究理念

本课题主要强调适用性，因此研究学习的过程需要紧贴行业领域现状。了解全链路跟踪相关应用领域背景知识，了解国内外行业规则、规范和技术发展趋势，理解其对环境以及社会可持续发展的影响，理解相关行业的政策和法律法规。

在老师、前辈的指导下自学相关的技术知识，阅读国内外优秀的文献以及观看有价值的引导视频，了解实现项目的原理，并在这个过程中不断尝试动手搭建项目、测试以达到成功的目的。

### 2.2.3技术路线

首先阅读谷歌发布的一篇名为《Dapper, a Large-Scale Distributed Systems Tracing Infrastructure》的论文，这篇论文讲述了谷歌公司在面临大规模分布式系统跟踪问题时采用的策略。目前各公司的跟踪系统的设计理念多取与此，在这篇论文的基础上衍生出了许多版本的跟踪系统，如twitter的zipkin、naver的pinpoint、以及阿里的鹰眼系统等。其中twitter将zipkin系统开源，可以作为本选题的学习资料之一。

通过阅读上述的论文还有众多的项目引导文件，了解微服务跟踪系统的设计结构和基础技术方案。论文提出系统的设计是能够达到应用层透明的，它采用的方法是开发出一组包含能广泛应用的线程，控制流和RPC库，通过使用这些库可以将一个项目标准化，也就是能够采集调用信息。跟踪系统的一个重要目标是能广泛的采集信息，我们需要开发出适合尽可能多的语言和框架能够使用的库，zipkin还有一些社区提供了部分已实现的库。这一部分的学习可以分为三步，首先尝试在本地使用这些工具库，了解他们的部署和使用方法。然后阅读理解它们的原理和设计方法，之后再尝试创新构建自己的工具库，覆盖到远程过程调用、数据库调用、缓存、异步消息通信。

为了降低跟踪系统造成的负载，调用消息不能掺入太多跟踪所需的数据，对调用的侵入仅加入traceid标志，用于将采集的数据串联起来，恢复整个调用链。而其他相关信息如事件时间、类型等，保存在本地。为了保证信息的统一，分布的采集系统需要一个共同的策略来上传数据，使得程序员查询时会显示一次完整的调用链。上传采用unix的socket通信上报收集到的日志。为了降低跟踪系统的负载，同样需要对采集信息的频率加以限制，这里在库开发的过程加入一个采样频率限制的机制，可以调节采样的程度。

采样收集中心主机运行一个后台进程，与分布式系统的各主机进行通信，将采样的结果放入数据库中，由于产生的数据量较大，需要对数据库进行索引优化，根据性能表现选择合适的数据库类型。由于laravel框架模块间耦合度低，适合快速开发，用它开发出一个方便程序员查询的图形界面。

### 2.2.4初步方案

1. 前提知识：微服务框架常用的调用协议，RPC协议具体内容。了解追踪系统所需要跟踪的信息。
2. 确定采集的信息的数据结构，一个信息单元是一个span，它拥有一个traceId，spanId, parentId 三个基本信息。TraceId在request中传播，利用一个traceId能够恢复一个完整的调用链。初次之外还包含start和end两个timestamp来确定一个处理占用的时间，以及一些其他的信息。
3. 编写工具库文件，拦截收到的请求，并生成一个span记录。对于顺序调用的方式可以简单记录时长，对于异步通信采用构造回调函数放到一个线程或其他执行顺序中并将trace的上下文放入其中，当返回来临时，调用回调函数会将这次的调用记录下来。收集完信息后统计一定的记录使用socket通信上报给collector插入数据库中。
4. collector收集进程在后台运行，它收到每个主机发送来的数据，并且插入数据库中。其中可以用traceId和spanId，parentSpanId来回现一个调用链。
5. 使用PHP编写的laravel web框架编写一个供程序员查询的页面。

## 2.3. 实施技术方案所需的条件

* 1. 操作系统：linux/Ubuntu 16.04、windows 10
  2. 开发环境：VMware虚拟机、PhpStorm、Visual Studio 2015 community
  3. 开发语言：C/C++、php

## 2.4. 存在的主要问题和技术关键

* 1. 异步通信的跟踪

对于同步通信原进程会阻塞直到消息的到来，这样我们按照执行顺序记录下各事件的时间即可，但有时调用是异步的，如何保证异步通信的情况下也能很好的记录是个问题。

* 1. 消息的同步

本地的采集者如果在采集完信息的下一步就将数据发送给collector，将会增加网络负担。并且如果恰好此时处于调用流程下游的进程还没完成记录，而程序员却查看调用链，那么系统就会显示出一个不完整的调用链，控制信息传递的同步是个需要解决的问题。

* 1. 工具库的编写

跟踪系统要求对原系统无侵入式跟踪，并且需要支持多语言多框架，所以具有一定的设计难度。

* 1. 查询直观度

一次调用链可能包含多次不同层级的调用，如何设计一个交互方式能够让程序员直观的看出一次调用的流程。

## 2.5. 预期能够达到的研究目标

能够熟悉掌握微服务系统跟踪系统的设计理念和流行方式。实现一个能应用到多语言多框架的分布式跟踪系统，方便程序员查看每次调用链详情并定位系统瓶颈。

# 课题计划进度表

* + 1. 学习并掌握软件构件技术基础理论。（第1周-第2周）
    2. 研究谷歌的链路跟踪方案Dapper及推特开源的zipkin的具体实现，确定全链路跟踪中的数据采集需求，定义数据结构（第3周-第4周）
    3. 学习Unix网络编程相关技术，研究实现无侵入式日志采集上报（第5周-第12周）
    4. 开发图形界面用于查询调用链路信息（第13周-第14周）
    5. 完成毕业论文，提交软件及相关文档。（第14周-第15周）
    6. 完成本科生毕业设计（论文）外文翻译；（第1周-第15周）
    7. 完成本科生毕业设计（论文）答辩；（第1周-第15周）

**参考文献**

1. Sigelman B H, Barroso L A, Burrows M, et al. Dapper, a large-scale distributed systems tracing infrastructure[R]. Technical report, Google, Inc, 2010.  
   Chow M, Meisner D, Flinn J, et al. The Mystery Machine: End-to-end Performance Analysis of Large-scale Internet Services[C]//OSDI. 2014: 217-231.
2. Dragoni N, Giallorenzo S, Lafuente A L, et al. Microservices: yesterday, today, and tomorrow[M]//Present and Ulterior Software Engineering. Springer, Cham, 2017: 195-216.
3. 郑邦峰.分布式系统服务链追踪与监控[J].工业技术创新,2018,05(02):56-60.
4. 辛园园,钮俊,谢志军,张开乐,毛昕怡.微服务体系结构实现框架综述[J].计算机工程与应用,2018,54(19):10-17.
5. Grpc Team.GRPC[EB/OL].[2018-07-25].https://grpc.io/.
6. Nguyen T.Benchmarking performance of data serialization and RPC frameworks in microservices architecture: Grpc vs. Apache Thrift vs. Apache Avro[Z].2016.
7. 杨帆.基于zipkin协议的分布式调用跟踪方案[J].福建电脑,2018,34(01):124+152.
8. “Open Zipkin A distributed tracing system”2013, <https://github.com/openzipkin/zipkin>.
9. Enck W, Gilbert P, Han S, et al. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones[J]. ACM Transactions on Computer Systems (TOCS), 2014, 32(2): 5.
10. Myalapalli V K, Shiva M B. An appraisal to optimize SQL queries[C]//Pervasive Computing (ICPC), 2015 International Conference on. IEEE, 2015: 1-6.