西安交通大学

毕业设计（论文）

题　目 基于微信小程序及 SpringBoot 的短视频应用开发

电气工程 学院 电气工程及其自动化 专业 电气 512 班

学生姓名 林子牛

学　　号 2150400330

指导教师 甘永梅

设计所在单位 西安交通大学

2019 年 4 月

西 安 交 通 大 学

系（专 业） 电气工程及其自动化
系 (专业) 主任 _____
批 准 日 期 2019-03-05

毕业设计 (论文) 任务书

电气工程 学院 电气工程及其自动化 系 (专业) 电气 512 班 学生 林子牛

毕业设计 (论文) 课题 基于微信小程序及 SpringBoot 的短视频应用开发

毕业设计 (论文) 工作自 2019 年 2 月 26 日起至 2019 年 6 月 1 日止

毕业设计 (论文) 进行地点：西安交通大学

课题的背景、意义及培养目标

SpringBoot 是一个约定大配置的新型 Java Web 框架，其设计目的是简化 Spring 应用开发以及减少不必要的配置。SpringBoot 框架在开发微服务应用方面有着非常大的潜力。

微信小程序是由腾讯开发的新兴的应用分发形式。主要依托于微信，具有开发周期短、应用分发方便以及跨平台等优势。

本毕业设计拟利用 SpringBoot 和微信小程序开发短视频应用程序，其中 SpringBoot 为服务端，微信小程序为客户端。学生在通过此次毕业设计，学习和掌握互联网应用的开发流程、微服务的实践与部署以及了解互联网的体系架构，对于培养学生专业能力以及实践能力非常有帮助。

设计 (论文) 的原始数据与资料

1. SpringBoot Reference Guide
2. Bruce Eckel 著，Java 编程思想（第 4 版），北京：机械工业出版社，2007 年

课题的主要任务

本毕业设计主要是学习和开发基于 SpringBoot 的短视频应用。该学生具体的工作内容包括：

1) 了解和学习微信小程序以及 SpringBoot 框架的特点及相关技术。
2) 采用微信小程序开发应用客户端并上线运行。
3) 采用 SpringBoot 作为服务端框架，使用 FFmpeg 处理短视频，开发一个短视频应用服务端程序。
4) 进一步完成应用调试和测试，验证应用可以可靠地工作。
5) 撰写论文并完成约 3000 字翻译。

课题的基本要求 (工程设计类题应有技术经济分析要求)

1) 设计的软件功能完备，能够正常在智能手机中运行。
2) 论文条理清晰，表述清楚，格式规范。
3) 英文翻译正确。

完成任务后提交的书面材料要求 (图纸规格、数量，论文字数，外文翻译字数等)

1) 毕业设计论文一本（A4 纸, 1.5 万字以上）
2) 英文翻译（原文和译文，译文翻译字数在 3000 字以上）
3) 程序源码

主要参考文献

1) Craig Walls 著，Spring in Action（Fourth Edition），北京，人民邮电出版社，2016 年
2) 李华兴等著，Java Web 开发实战经典，北京：清华大学出版社，2010 年 8 月

指导教师：甘永梅

接受设计（论文）任务日期：2019-01-07

（注：由指导教师填写）　　学生签名：

1 绪论

本章阐述了此次毕业设计的背景，目前行业内的研究现状，此次毕业设计的研究问题以及此设计的主要工作与创新点等内容。

1.1 研究背景

近年来随着诸如 4G、高速宽带网络以及智能手机技术的飞速进步。人们的休闲娱乐重心逐渐从文字与图片相关内容转移到了短视频与直播应用上。市场上也涌现出了一系列的短视频应用软件。本毕业设计课题将完成短视频应用开发的整个流程。

目前短视频应用已经成为互联网产业链中一个重要的流量入口。截止 2018 年 12 月，在线视频行业月独立设备数达 10.17 亿，同比增长 1.7%，其中短视频行业月独立设备数达 7.34 亿台，同比增长率为 58.7%，用户短视频应用使用时长达到了总上网时长的 8.2%[中国互联网络信息中心 2018 第]。在智能手机已经普及的今天，短视频相关应用已经爆发出了强大的活力，具有相当高的市场价值。所以进行短视频应用开发对于我们认识短时频应用相关技术以及商业模式具有非常大的帮助作用。

短视频应用主要功能为接受用户上传的视频并对其进行一系列的处理，如：转码、压缩、合并背景音乐以及提取关键帧等操作。服务器在进行视频相关操作时所承受的压力较大，对于多用户同时上传视频的情景需要设计出一种合理的视频操作解决方案并进行一系列优化。如何合理地处理此种情景也是十分重要的。此外，视频的长度、清晰度、码率、上传的格式也会对服务器的视频操作压力产生影响，因此，本次毕业设计需在视频上传前对视频进行预处理。

当网络应用的用户数较大、同时访问人数较多时，应用系统的瓶颈通常位于服务器的 I/O 输入输出、数据库的操作以及服务器的带宽上，此时，单纯的增加服务器的数量往往不能有效地解决问题。因此，合理设计应用系统架构、合理地进行技术选型是非常有用的，有必要对各类型应用架构进行深入地调查、分析与研究。

1.2 研究现状

本节引入了与本设计紧密相关的部分内容：Java Web 技术、数据库技术以及 FFmpeg 技术。

1.2.1 Java Web 技术

Java Web 技术即使用 Java 语言开发运行在 JVM(Java 虚拟机) 上的网络应用程序的技术。其主要任务是开发出具有合理、高效、低耦合、高内聚、较少错误和较好的可扩展性等优点的应用。Java Web 开发需基于 Java 官方给出的 Java EE 规范，且网络应用需运行在 Java 容器或 Java 网络服务器中。随着网络技术以及网络应用的发展，Sun 公司研发了 Servlet 技术。Servlet 是一个能够处理 HTTP 请求的 Java 程序。随着 Servlet 的几轮技术迭代，Sun 公司发布了基于 Java 2 平台的 Java 企业版。由此揭开了 Java 用于网络应用开发的大幕。现在，行业内一般使用某种 Java Web 框架作为基础进行开发，搭配 MVC 技术、ORM 技术以及 Web 服务器进行相关功能的实现。常用的框架有 EJB 框架以及 Spring 框架。

EJB 即 Enterprise Java Bean 是一个企业级的 Java 框架，内置了 JBOSS 服务器。图1-1 是 EJB 框架的架构图，EJB 框架提供了分布式应用功能，即你可以将一个应用分散到多个服务器上，由多个 Java 虚拟机一同运行。此外 EJB 还提供了组件化支持与持久化管理功能。EJB 技术可以有效地

提高 Java 企业级应用开发的效率，但与此同时，EJB 也产生了许多负面影响，如：EJB 使得应用系统更难测试、使应用系统更难部署、破坏了面向对象准则等。因此需要合理地抉择是否使用 EJB。需要使用 EJB 的情景有：应用的部分组件需要被远程访问的情景以及需要对应用进行分布式部署的情景。

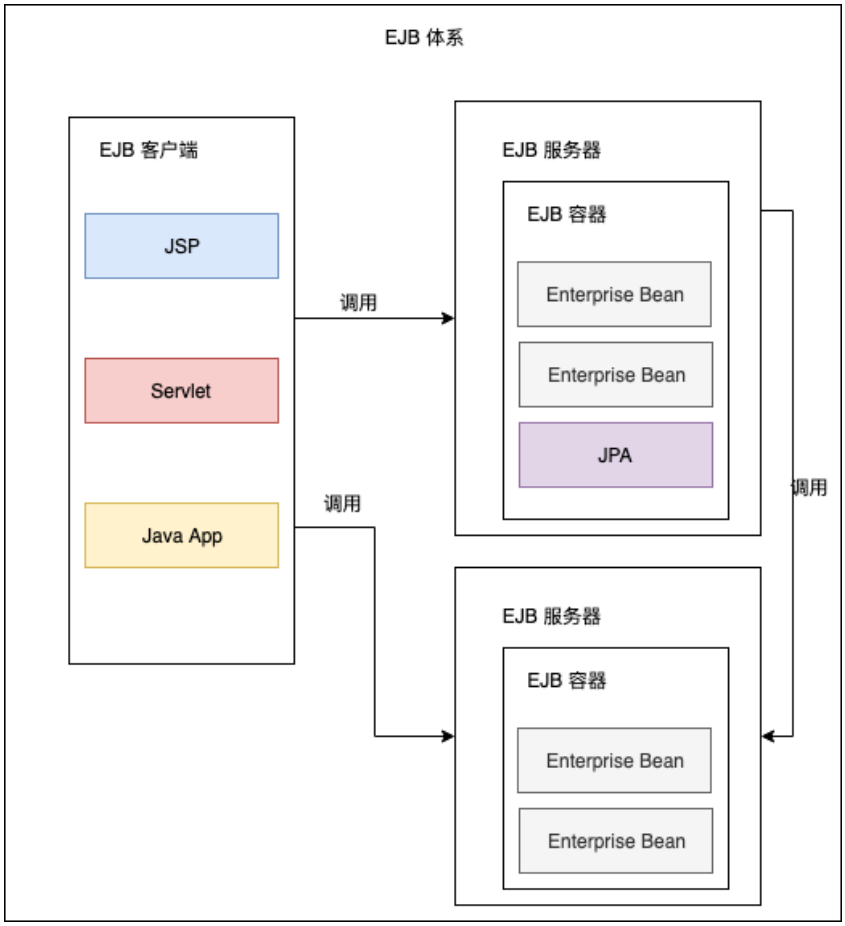


图 1-1　EJB 框架架构图

Spring 框架是目前非常流行的 Java 框架，图1-2 展示了 Spring 框架的各模块的组织情况。Spring 框架诞生之初就是为了取代类似 EJB 的重量级 Java 框架。Spring 框架通过扩展 POJO(Plain Old Java Object) 提供了一种轻量、方便的编程方式。Spring 框架提供了依赖注入与面向切面编程帮助用户创建低耦合、高内聚以及高可扩展性的应用程序。Spring 框架非常的灵活，用户可以自由选择应用中每一个模块的功能实现，同时也向用户提供了多种看问题的思路。Spring 框架的使用场景非常多，无论是传统网络应用开发还是新兴的微服务系统开发都可以使用 Spring 框架。

根据目前互联网业界的经验以及本次毕业设计的需求来看，EJB 框架过于臃肿复杂，不适合本次应用的开发。本次应用开发将使用 Spring 框架，遵循软件开发过程中的增量模型，即软件开发过程经过若干次迭代，每次迭代都添加部分新功能。Spring 框架的开放性、轻量性与灵活性高度适合互联网应用的开发。因此本次应用设计将基于 Spring 框架进行。

1.2.2　数据库技术

数据库是指长期储存在计算机系统内部的、有组织的、可共享的大量数据的集合，数据库中数据按一定数据模型储存，具有有组织、可共享、冗余度低、独立性高以及可扩展性强等特点。数据库系统使用数据模型来对数据进行抽象、组织与管理。数据模型根据其功能可以分为两大类。第

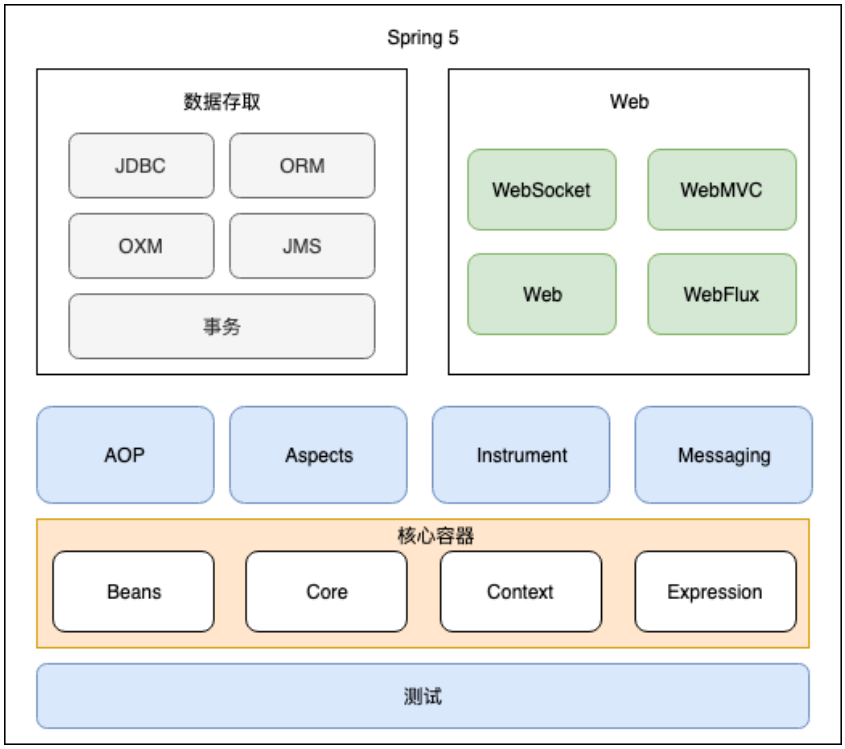


图 1-2　Spring 模块组织图

一类是概念数据模型，第二类是逻辑数据模型与物理数据模型。概念数据模型主要用于设计数据库，逻辑数据模型和物理数据模型主要用于数据库的实现。其中，逻辑数据模型根据数据结构的不同还可以分为：层次模型、网状模型、关系模型、面向对象数据模型、对象关系模型以及半结构化数据模型。现在一般使用关系模型作为数据库的了逻辑数据模型。1970 年 E.F.Codd 提出了关系数据模型。关系数据模型建立在严格的数学基础之上，每一个关系都是一张二维表。关系模型具有许多优点：关系模型的概念比较单一，无论是数据库中抽象出的实体还是这些实体之间的关系都可以用关系表来抽象表示，此外，关系模型中的数据操作时集合操作，其存取路径对于用户来说是隐蔽的，用户在操作时只需指定做什么而不需要告知数据库怎么做，极大地方便了用户使用数据库。

随着互联网的发展，关系型数据库逐渐暴露出了诸多弊端：如关系数据库无法有效地处理大量数据的写入、数据更新时改变表的结构以及简单查询迅速返回结果。为了解决这些弊端，开发者们提出了非关系型数据库。非关系型数据库只用于特定领域，几乎不做复杂运算，它具有易于分散数据、方便提升性能的优点。非关系型数据库根据储存原理的不同又可分为：键值型数据库、文档型数据库和列储存数据库等。键值型数据库原理类似哈希表，一键值对为单元存取数据，其存取速度较快，但存取方式单一，Redis 为键值型数据库最流行的实现。文档型数据库类似一个 JSON 文件，即使不定义表的结构也可像定义了表结构那样使用。MongoDB 是文档型数据库最常用的实现。列储存数据库以列作为储存单元，可以对大量行少数列进行读取，对某一列所有行同时更新，具有高度的可扩展性。HBase 是列储存数据库最常用的实现。

1.2.3　FFmpeg 技术

FFmpeg 是一套可以用来记录、转换数字音频、视频，并能将其转化为流的开源计算机程序。FFmpeg 具有视频格式转换、码率压制以及视频剪接与合并等功能。FFmpeg 基础库主要包含：AVFormat 库、AVDevice 库、AVUtil 库、AVCodec 库以及 AVFilter 库等。AVFormat 负责音视频的封装与解封装。AVCodec 负责音视频的编解码。AVFilter 库负责添加与删除滤镜。FFmpeg 的工作

流程较为简单：解封装、解码、编码、封装，最后即可输出目标视频。FFmpeg 自身支持一部分编码同时也可以通过挂载外部库的方式对其进行定制。

1.3 论文的主要工作

本节介绍了本次毕业设计中的各项具体工作。

首先，本文详细阐述了此次毕业设计使用的各种技术的原理以及整个短视频应用系统架构的设计。在应用系统开发完成后在云服务器平台进行部署与测试。

在云端完成部署后开始压力测试，找出应用系统设计问题与性能瓶颈。此应用的性能瓶颈是 FFmpeg 处理视频时的性能压力较大和数据库访问量过高时数据库响应过慢等问题。本文将详细论述这些问题的解决过程如：数据库的优化方法与效果、FFmpeg 性能提升方法等。实验证明这些优化措施在云服务器上可以起到比较好的作用，对于系统性能的提升也有一定的作用。

服务端应用系统开发完毕后，开始微信小程序的开发。开发完成后与服务端进行联合调试，找出问题并解决。完成整个应用系统的开发。

1.4 论文的组织结构

2　设计原理

本节介绍本设计的理论框架，包括 Spring 框架、数据库设计与优化技术以及视频压制优化技术等。

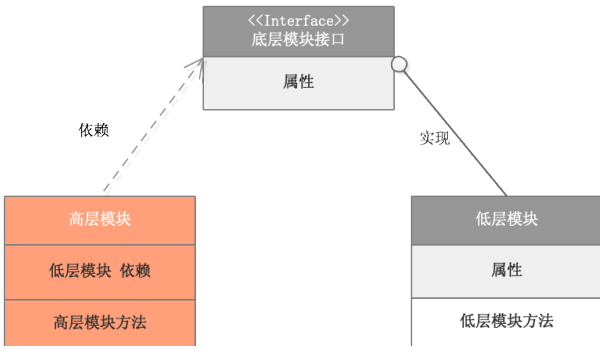
2.1　Spring 框架

Spring 框架是近年来非常流行的轻量级 Java Web 开发框架，它的开发目的主要是减小 Java 应用开发的复杂度。Spring 框架构建在核心模块之上，Spring 在它的核心模块张总为我们提供了一个控制反转容器以及各种工具类。在核心模块之上是 Spring AOP 模块，这为开发者提供了一系列面向切面编程支持。在 AOP 模块之上是数据访问与事务管理功能。下文将详细介绍这三个 Spring 框架的基础功能。

2.1.1　Spring 框架的控制反转容器

控制反转 (Inversion of Control, IoC) 概念来源于设计模式中的依赖倒置原则。依赖倒置原则即高层模块不应该依赖低层模块，二者都应该依赖其抽象；抽象不应该依赖细节；细节应该依赖抽象。简单地说，高层模块与低层模块不应该直接产生耦合，高层与低层模块应通过一个接口耦合在一起，即高层模块依赖于该接口，低层模块实现该接口。如图 2-1 所示，这样对于低层模块的修改就不会对高层模块产生较大的影响同时也减小了系统各模块的耦合。

(a) 使用 IoC 前



(b) 使用 IoC 后

图 2-1　使用 IoC 前后 UML 图

Spring 的控制反转容器是通过依赖注入 (Dependency Injection, DI) 实现的。如图 2-2 所示，依赖注入即高层模块在使用低层依赖时，不是自己讲低层模块实例化而是向控制反转容器申请一个已创建好的依赖对象。这样高层模块与低层模块间的耦合被降低了，对于低层模块的改变对高层模块产生的影响也变小了。同时，由于低层模块由容器创建，我们就可以轻松地实现单例模式等设计模式，进一步简化了开发也提高了系统的性能。

Spring 框架通过 BeanFactory 实现了最基本的控制反转功能，并且在 BeanFactory 之上提供了

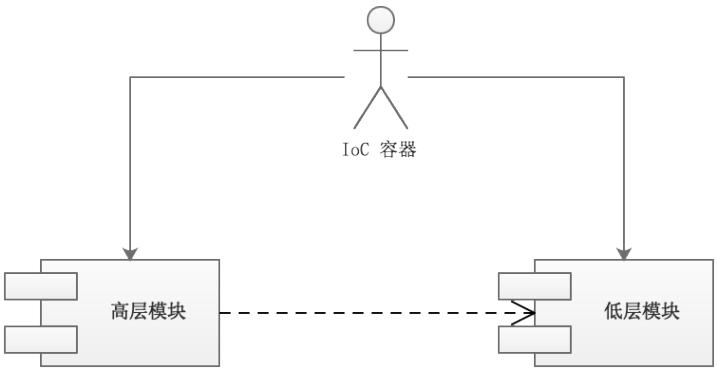


图 2-2　Spring 依赖注入作用

更先进的控制反转容器实现，即 ApplicationContext。BeanFactory 是基础类型的 IoC 容器，提供完整的 IoC 服务支持。ApplicationContext 是高级的 IoC 容器，提供了诸如 Bean 生命周期管理、统一资源处理以及 AOP 支持等功能。通过使用 Spring 框架提供的 IoC 容器，我们可以轻松地使用控制反转构建应用系统。

2.1.2　Spring AOP 框架

在一个应用系统中存在着一些许多模块公用的功能，如：日志记录、安全检查以及事务功能等。这些公用功能如果处置不当就会造成系统的代码冗余度增加和耦合性上升，例如：要在系统中增加一次日志记录，我们就必须在所有的模块中加入相同的代码，着无疑是非常低效的。对此，我们可以使用面向切面编程 (Aspect Oriented Programming) 来解决这一问题。

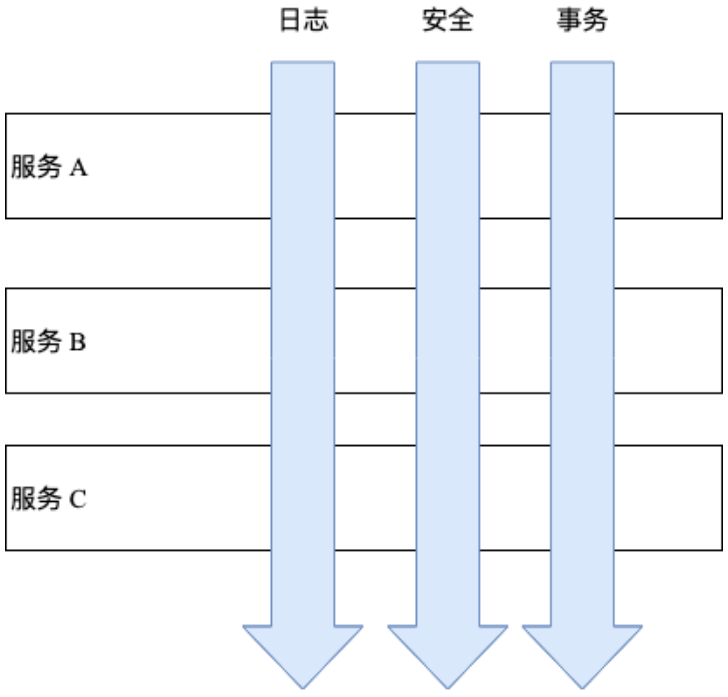


图 2-3　面向切面编程中的横切关注点

如图 2-3 所示，面向切面编程可以将系统中的公共功能抽象成为许多称之为横切关注点的功能模块，例如：日志记录功能就是一个横切关注点。系统中需要使用这些公共功能的地方称之为连接点 (Joinpoint) 。在系统运行时面向切面编程框架会将横切关注点织入连接点中，这样系统就可以使用所有的公共功能，减少了系统的冗余代码与耦合性。

Spring 框架中的面向切面编程功能由动态代理实现，源于设计模式中的代理模式。如图 2-4 所示，代理模式的作用是为其他对象提供一种代理以控制对这个对象的访问。代理处于访问者与被访问者之间，隔离两者的直接交互。在代理将访问者的访问请求传递给被访问者时，代理可以添加一部分额外的功能，如安全检测。因此，代理模式可以有效地增强系统的灵活性与安全性。动态代理的实现还要依靠 Java 提供的反射功能。反射功能雨荨我们在 Java 程序的运行期间动态载入类、创建对象以及生成代理。通过实现代理模式、使用反射机制，我们就可以创建出动态代理系统，实现面向接口编程。

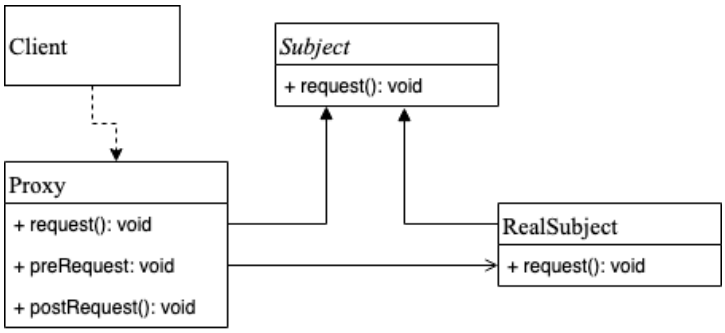


图 2-4　代理模式 UML 图

Spring AOP 的设计遵循了 8/2 原则，即通过 20% 的代码实现 AOP 框架 80% 的功能。因此 Spring AOP 支持的 AOP 功能是不完全的，例如 Spring AOP 不支持属性级别的拦截等。系统运行时，我们在系统中定义所有的切面、连接点都会被 Spring AOP 拼装起来，这个操作被称为织入。织入时，Spring 会自动生成切面的代理包装类，通过代理包装类将切面织入代码中。Spring AOP 框架为我们提供了 AOP 的大部分功能，同时我们也可以整合使用 AspectJ 等框架来获取 Spring AOP 不支持的功能。

2.1.3　Spring 事务管理功能

事务管理的作用是保证应用系统在操作数据库时不会对数据的正确性产生破坏，具体的事务定义将在下文介绍。在 Spring 框架中，我们通常使用声明式的方式使用事务，事务也是一个常用的 AOP 中的横切关注点。通常情况下使用事务，我们需在每次访问数据库前声明事务开始，在事务正常结束后进行结束事务操作，在事务异常结束后进行事务的回滚。Spring 框架使用 AOP 来处理事务，通过在运行时将事务代码织入应用中来简化开发。

我们在需要使用事务的地方使用 @Transactional 注解即可开启声明式事务功能，开启声明式事务功能后 Spring 并不直接管理事务，而是提供多种事务管理器，由它们将事务的具体管理委派给具体使用的各种持久化框架。我们也可以通过这些事务管理器进行事务属性的配置。事务属性主要包括：隔离级别、传播行为、回滚规则、是否只读以及超时时间等。通过注解使用声明式事务对于简化数据访问层的开发具有较大的意义。

2.2　数据库的原理与设计

本次设计采用 MySQL 作为系统主数据库、Redis 作为缓存数据库。

2.2.1　MySQL 原理

MySQL 是一个开源的关系型数据库管理系统。MySQL 非常的灵活，可以适应多种运行场景，例如：MySQL 既可以嵌入在程序中运行也可以支持数据仓库、在线处理系统等应用。

1) 关系型数据库

关系型数据库是支持关系模型的数据库系统。关系模型非常的简单，只包含单一的数据结构—关系。一个关系是一张二维表，可以由多个域的笛卡尔积表示。域是一组具有相同数据类型的值得集合。如一个班中所有学生的年龄的集合就是一个域。

笛卡尔积是一种域上的集合运算，其定义为：

$$D_1 \times D_2 \times \cdots \times D_n = (d_1, d_2, \cdots, d_n) | d_i \in D_i, i = 1, 2, \cdots, n \tag{2-1}$$

其中，每一个元素 $(d_1, d_2, \cdots, d_n)$ 为一个元组。笛卡尔积可以表示一个二维表，表中的一行对应笛卡尔积中的一个元组。$D_1 \times D_2 \times \cdots \times D_n$ 的子集为在域 $D_1, D_2, \cdots, D_n$ 上的关系。

每一个关系中都存在一些特殊的属性。若关系中某一属性组可以唯一标志该元组，而其任意子集均无此性质，则该属性组为这个关系的候选码。任意包含在至少一个候选码中的属性为主属性，不包含在所有候选码中的属性为非主属性。在任意关系表中不可能存在具有相同候选码的两个或多个元组。

关系数据库中，关系模式是关系的类型，关系是关系模式的实例。关系模式中主要包含关系中有哪些属性、关系中有哪些域以及域与属性的映射关系。关系模式可以表示为 $R(U, D, DOM, F)$，其中 R 为关系名，U 为属性名的集合，D 为域的集合，DOM 为属性与域的映射，F 为属性间的依赖关系。

关系数据模型中也具有许多对于关系的操作，这些操作可以分为两类：集合操作和关系专用的操作。集合操作包括：并 (union, $\cup$) 、差 (except, $-$) 、交 (intersection, $\cap$) 、除 (divide,)□□□□□($DescartesProduct$,$\times$)。关系专用操作包括：选择 (select, σ) 、投影 (project, Π) 以及连接 (join,)。其中，选择、投影、并、差、笛卡尔积是基本操作。其他的操作与基本操作关系如下：

2.3 监控摄像头部署方案的评价指标

监控摄像头的部署方案包括摄像头的数量以及部署位置。摄像头的部署位置会影响监控场景的完整性、监控画面的光线质量以及监控目标的呈现角度。而监控摄像头数量受成本预算的限制，不可能无限增加，因此在预算有限的约束下，如何设计监控摄像头的部署位置，使得监控效果最优，便成为一个值得研究的问题。在研究优化问题之前，需要定义监控摄像头部署方案的评价指标 $\mathcal{E}$。

监控效果的优劣可以定义为在当前的监控方案下，跨摄像头追踪特定行人的能力。当前在多摄像头多行人追踪（Multi-Target Multi-Camera Tracking，MTMC Tracking）领域主流的评价指标有多目标跟踪准确度（$MOTA$）[ristani2016MTMC] 和识别 F 值（IDF_1）[ristani2016MTMC]。

2.3.1 多目标跟踪准确度（*MOTA*）

多目标跟踪准确度（Multiple Object Tracking Accuracy，*MOTA*）是衡量多目标追踪效果的常见指标。对于一段视频，其画面帧的总数为 T，那么该段视频的多目标追踪准确度（*MOTA*）的定义为：

$$MOTA = 1 - \frac{FP + FN + \Phi}{T} \tag{2-2}$$

其中 *FN* 是视频所有帧中将负样本预测为正的总数，*FP* 是视频所有帧中将正样本预测为负的总数，Φ 是预测序列中标签跳变的次数。*MOTA* 指标的值域为 $(-\infty, 1]$，越接近 1 代表跟踪的效果越好。

2.3.2　识别 F 值（IDF_1）

相比与 *MOTA* 中更多地关注目标人群的召回率以及目标追踪的稳定性，识别 F 值（Identification F-Score，IDF_1）更关注多摄像头多行人追踪过程中，行人标签的准确率。对于一段总帧数为 T 的视频，其识别 F 值（IDF_1）定义为：

$$IDF_1 = \frac{2 \times IDTP}{2 \times IDTP + IDFP + IDFN} \tag{2-3}$$

其中 *IDTP* 是视频所有帧中将人物标签预测准确的总和，*IDFP* 是视频所有帧中将正样本的人物标签预测错误的总和，*IDFN* 是视频所有帧中将负样本的人物标签预测错误的总和。

2.3.3　评价指标与当前的数据库结合

结合本项目的实际情况，以及第一次预拍摄收集到的数据集的特点，将原始的 17 个摄像头中的第 1 个作为行人重识别算法的图库图片来源，其余 16 个摄像头按照物理位置和拍摄画面分为 $G = 5$ 组，每组摄像头个数为 2 至 4 个不等，定义 $N = 16$ 为待分配的摄像头总数，第 i 组的摄像头个数为 C_i，同一组内摄像头大致拍摄到同一个物理位置，代表人物监控追踪场景中重点关注的位置。第 i 组的第 j 个摄像头定义为 $c_{i,j}$，摄像头 $c_{i,j}$ 的画面帧集合为 $\mathcal{I}_{c_{i,j}}$。假设在预算有限的前提下，每个位置（每个分组）只能选择 1 个摄像头，如何从组内选择合适的摄像头，使得监控效果最佳，便是需要解决的问题。该优化问题可以用数学语言形式化表示为：

$$\max \left\{ \mathcal{E}\left[\mathcal{F}\left(\sum_{i=1}^{G} \mathcal{I}_{c_{i,j}} \right) \right] \middle| 1 \le i \le G, 1 \le j \le C_i \right\} \tag{2-4}$$

其中 $\mathcal{I}_{c_x} + \mathcal{I}_{c_y}$ 表示摄像头 c_x 的视频数据与摄像头 c_y 的视频数据在时序上依次拼接。$\mathcal{E}[\mathcal{F}(\mathcal{I})]$ 表示对视频数据 $\mathcal{I}$ 做多目标跟踪准确度（*MOTA*）或识别 F 值（IDF_1）评估。

2.4　强化学习模型

强化学习（Reinforcement Learning）是近年来十分流行的人工智能算法，相比于监督学习（Supervised Learning），强化学习不需要整个环境（Environment）所有情况的监督信息，只需要环境在某种特定的情况下给出相应的反馈（Reward）。在很多现实问题当中，优化空间的状态个数可能是个非常大的数字，且很证明是否收集到足够多样本，可以用来近似代表位置环境状态的分布。同时，在围棋问题[silver2016mastering] 中，各状态空间很难用监督的方法给每个状态评估价值，而强化学习只需要环境在每次动作（Action）之后给出相应的反馈，即可逐渐向更优的方向前进。强化学习也不属于无监督学习（Unsupervised Learning），无监督学习对于出现在测试集却不在训练集的样本没有处理能力，只能错误地分到已有的类中，而强化学习可以应对没有遇到的情况。

强化学习模型中的一般形式是一个智能体（Agent）在一个客观的环境（Environment）中，每一个时刻处于一个状态（State），当前状态存在一个短期价值和长期价值，短期价值可以是采取某种动作（Action）之后得到的反馈（Reward），长期价值（Value）则表示当前状态到最终状态能够得到的所有反馈总和的最大值。智能体根据当前状态的短期或长期价值和某种策略（Policy）采取某种动作，可立刻得到得到环境的反馈（Reward），并据此按照状态转移规则转移到下一个状态。在本项目中，根据要解决的具体问题，将上述概念相应定义如下：

当前的状态的集合 $S = \{\boldsymbol{s} \,|\, \boldsymbol{s} \in \mathbb{R}^N, s_i \in \{0,1\}\}$，其中 $s_i = 1$ 表示选择了第 i 个摄像头。智能体可以采取的动作集合为 $A = \{\boldsymbol{a} \,|\, \boldsymbol{a} \in \mathbb{R}^N, a_i \in \{0,1\}\}$，按照策略 P 来选择动作，1 表示选择

该摄像头，0 表示不选该摄像头。在本项目中，进行了动作之后，状态是确定的，不存在一个动作可能导致几个不同的后续状态的情况，即状态转移概率 $\pi\left(\boldsymbol{s}^{t} \mid \boldsymbol{s}^{t-1}, \boldsymbol{a}\right) \equiv 1$。执行动作之后得到的反馈 $r=\mathcal{E}\left[\mathcal{F}\left(\mathcal{I}^{(t+1)}\right)\right]-\mathcal{E}\left[\mathcal{F}\left(\mathcal{I}^{(t)}\right)\right]$，其中 $\mathcal{I}^{(t)}$ 表示 t 时刻的视频数据，由 t 时刻的状态 $\boldsymbol{s}^{(t)}$ 决定。当前状态 S 的长期价值 $V: S \mapsto \mathbb{R}^{N}$，是一个可学习的变量，代表智能体对于当前环境的认识程度。智能体应对当前状态的策略 $P: V \mapsto A$，是长期价值 V 到动作 A 的映射，可以简单地用贪心的策略，即选择概率最高的 5 个摄像头。也可以用 Policy Network ，即深度神经网络来实现。在本项目中采用简单的贪心策略实现，如算法 1 所示。

算法 1 Q-Learning 算法求当前状态的长期价值

输入： 状态集合 S、动作集合 A、生命周期数 N、学习率 α、远见性 γ

输出： 长期价值矩阵 Q

function QLearning(S, A, N, α, γ)
  随机初始化长期价值矩阵 Q
  for $i=0 \rightarrow N$ **do**
    $\boldsymbol{s}^{(0)} \leftarrow \boldsymbol{s}^{\star}$，$\boldsymbol{s}^{\star}$ 为从状态集合 S 中随机初始化的智能体的状态
    $t \leftarrow 1$
    repeat
      $\boldsymbol{a}^{(t)} \leftarrow \boldsymbol{a}^{\star}$，$\boldsymbol{a}^{\star}$ 为从动作集合 A 中随机选取的动作
      $\boldsymbol{s}^{(t+1)} \leftarrow \pi\left(\boldsymbol{s}^{(t)}, \boldsymbol{a}^{(t)}\right)$
      $r^{(t)} \leftarrow R\left(\boldsymbol{s}^{(t+1)}, \boldsymbol{s}^{(t)}\right)$
      $Q\left(\boldsymbol{s}^{(t)}, \boldsymbol{a}^{(t)}\right) \leftarrow (1-\alpha) \times Q\left(\boldsymbol{s}^{(t)}, \boldsymbol{a}^{(t)}\right)+\alpha \times\left[r^{(t)}+\gamma \times \max_{\boldsymbol{a}'} Q\left(\boldsymbol{s}^{(t+1)}, \boldsymbol{a}'\right)\right]$
      $\boldsymbol{s}^{(t)} \leftarrow \boldsymbol{s}^{(t+1)}$
      $t \leftarrow t+1$
    until $\boldsymbol{s}^{(t)}=\boldsymbol{s}^{(0)}$
  end for
  return Q
end function

按照以上定义，智能体的一个动作就是选择一个合法的摄像头部署方案。对于一个动作，它的反馈就是下一个状态的性能指标 $\mathcal{E}$（*MOTA* 或 IDF_1）减去当前状态的性能指标。在反馈已知的前提下，适合用 Q-Learning[watkins1989learning] 算法。在 Q-Learning 算法中，智能体关于当前状态的长期价值表示为一个价值矩阵 Q。Q-Learning 算法首先随机初始化各个状态的长期价值，让智能体在环境中随机游走，每走一步会得到一个反馈，根据反馈更新当前状态的长期价值，直到收敛。智能体从而可学习出一个对于该环境的认知。在算法 1 中最关键的算法在于如何更新长期价值 Q，更新的公式中包含两个参数学习率 α 和远见性 γ，学习率 α 表示 Q 的更新速度，远见性 γ 的取值范围为 $(0,1)$，表示智能体对于当前反馈与长远价值的重视程度，γ 越大，代表越重视长远价值。

2.5　面向 CPU 集群的分布式深度学习训练框架

计算机集群是一种计算机组织的物理形态，它是由一组彼此连接的计算机组成的，这些计算机一般协同完成同一项计算任务。在同一集群中每台计算机的内部结构可以不同，特别地，若集群中每台计算机主要的算力提供者是 CPU，那么称该集群为 CPU 集群。分布式计算是计算的一种工作方式，它将一个计算任务划分成多个子任务，每一个子任务与其它子任务相对独立，可以在时间上并行计算，以缩短计算时间。CPU 集群提供了一组在物理上相对独立的计算机，所以可以很自然地考虑将分布式计算中的各个子任务部署到 CPU 集群中，充分利用 CPU 集群的计算资源。

深度神经网络模型的训练过程一般可分为前馈计算（Forward）、误差反向传播（Loss Backpropagation）和参数更新。其中前向计算和反向传播的计算量很大，而且可以针对不同的训练集进行同步计算，因此深度神经网络模型的训练过程在结构上很适合进行分布式训练。在本项目中，集群的数量 $n = 5$，每个节点属于天河二号的 GPU 分区，具备高性能的 CPU 和 GPU 计算资源，节点之间通过千兆网络连接，避免各节点的通信速度成为分布式计算的性能瓶颈。本项目中分布式训练架构如图 2-5 所示。

图 2-5　分布式神经网络训练架构图

如图 2-5 所示，在训练过程中将训练数据通过随机采样的方式平均分成 n 份，分别输入集群中的各计算机。每一台计算机内存中包含一个独立的深度神经网络模型，进行该份训练数据的前馈计算和反向传播计算，得到该批次训练数据在当前模型下的各参数梯度。参数服务器的计算任务是收集集群中各计算机回传的梯度，更新模型参数，并将新参数分发给各计算机，各计算机得到新参数后进行下一批次训练数据的计算。更新模型参数的方法为：

$$\Delta W = \frac{1}{n}\sum_{i=1}^{n}\Delta W_i \tag{2-5}$$

$$W = W - \eta\Delta W \tag{2-6}$$

其中 η 是参数的学习率。