学校代码： 10246

学　　号：

硕 士 学 位 论 文

（专 业 学 位）

基于微服务架构的媒资分发系统设计与实现

A Media Asset Delivery System Based on the Microservice Architecture

院　　系：　软件学院

专　　业：　软件工程

姓　　名：

指 导 教 师：

完 成 日 期：　2016 年 2 月 13 日

目 录

摘 要

时代公司是一家著名的有线电视网络媒体公司，近年来开始制作自己的高品质节目，并将这些电视，电影节目供应给来自电视，网站，手机等多平台的用户。随着节目质量的上升，越来越多的用户订阅时代的媒体内容，也越来越多的合作伙伴将他们的媒体内容投放到时代公司的媒体平台上，随之而来的是大量的数字资产需要被存储，处理和分发。在这些数字资产中，以视频，音频，图像，文档资料等形式为主，这些资产一般需要进行重新组装后才能分发给用户或者合作伙伴，然而来自世界各地的合作伙伴拥有着不同格式的元数据格式与形式各异的资产标识，如何将这些大量的不同元数据格式的媒体资产进行转码编码并分发到各个平台成为了时代公司必须要攻克的难题。

首先讨论了时代公司在业务的扩大和增长的背景下，传统的单体应用架构在应对频繁的业务需求的增加和变化所体现出的不足，明确了设计了一种基于将业务逻辑细分成不同的逻辑职责单一的微服务架构的重要意义。在此基础上，讨论了媒资分发系统的主要功能模块包括：资产元数据整合模块，资产重组打包模块，资产分发模块等，并分析了资产分发状态监控流程，资产与元数据上传流程，视频资产转换流程。分析了基于微服务架构的媒资分发系统，对资产聚合子系统，资产编目子系统和资产投递子系统等核心子系统进行详细设计。其中重点讨论了通过资产聚合子系统将多种不同媒体资产进行整合并通过资产编目子系统对现有资产进行增加和修改。比较了单个微服务在快速开发和测试，部署和隔离系统错误上相比单体架构来的优势。最后，从扩展性，重用性，可用性，技术多样性，系统吞吐量，服务上线时间等维度对采用微服务架构和采用单体架构进行比较。

关键词 媒体资产；微服务架构；媒体资产分发系统；软件架构；新媒体

ABSTRACT

T Ltd.is a well-known cable television and media company in China, in recent years began to produce their own high-quality programs, and deliver programs and features to the end users from television, websites, mobile phones and other platforms. As the quality of the program increases, more and more subscribers subscribe to the media content of the T, besides, More than 40 business affiliates put their media content on the media platform of the T so that there is a large number of digital assets need to be stored, processed and distributed. The affiliates and business partners from all around the world have different formats of metadata and very different delivery requirement, how to deliver and manage the big amount final media assets become a problem.

Firstly, the current situation and problems of T are introduced, the shortage of the traditional monolithic architecture is identified and analyzed. Then the significance of building a new system based on microservice architecture is clarified. On this basis, the fundamental modules of the system including the final media asset delivery, final asset management and media asset transformation module are discussed and analyzed, the core processes of the media asset delivery monitor, final asset upload and modify and video transform are identified. Based on the requirement analysis of the media asset delivery system, the media asset ingestion, asset catalog and the media asset delivery are designed in detail, especially explain the asset aggregation service on handling ingestion request from different source with different formats. In the last of this paper, the system is compared with other systems from the availability, scalability and fault tolerance perspectives.

Keywords Media Asset; Microservice Architecture; Media Asset Delivery System; Software Architecture; New Media

第一章 引 言

随着科技的提高和人民生活水平的提升，越来越多的媒体公司开始更加关注人们的娱乐需求，大量的媒体内容如雨后春笋般的出现在各个媒体平台上，随着媒体内容的爆炸性增长，优质的媒体内容无疑成为了内容供应商的核心竞争力，然而从各个媒体平台的角度，快速的将大量来自不同内容供应上的优质内容投放到自己的媒体平台上至关重要。

1.1 媒资分发系统发展的现状

媒体对象是内容管理的基本元素并且是一个不可区分的对象，为了区分不同媒体对象便给这些媒体数据添加了属性信息和描述信息，这些属性信息便是元数据，利用元数据信息可以更加容易的找出相应的媒体对象，媒体对象加上媒体对象的元数据信息被称为媒体内容，媒体内容加上权限管理后就成为了媒体资产，简称媒资[1]。

传统的数字节目保存方式是通过保存物理录像带并将这些录像带模仿图书馆的管理模式，通过贴上标签存放在录像带库的架子上面，这样的保存方式有很多的不足，比如查询不便，信息的存储受到磁带质量的影响，对于这些资产重新利用的利用率比较低[2]，而媒资管理系统的出现很好的解决了传统方式的这种不足。媒资管理系统是一个对各种媒体资产进行统一管理的解决方案从而实现允许资产的拥有者对媒体资产的采集，存储，查找，编辑，发布的需要[3]。本文设计的媒资分发系统是结合时代公司业务需求面向媒体成品资产的分发和管理的一种媒体资产管理系统。

(1) 国外媒资管理系统的发展

媒资管理的概念起源于内容管理，随着20世纪90年代后期BBC和CNN这样的国际媒体企业把媒体资产管理的概念具体化才有了媒体资产的诸多定义，紧接着IBM,HP等IT界国际大公司加入，通过把IT技术与媒体资产管理相结合使得这些大公司的媒体资产管理系统成为了业界应用的典范[4]。

(2) 国内媒资管理系统的发展

国内的媒体资产管理系统在2000年由IBM公司引入，随后于2001年中国电影资料馆媒体资产管理系统的正式启用标志着我国的媒体资产管理从制播的媒体局部数字化时代迈向了实现全部数字化，紧接着2003年中央电视台音像资料馆媒体资产管理系统的建成促进了整个广电行业对媒体资产管理的数字化进程[5]。

最初的媒体资产管理系统的设计目标包括资产的存储以及帮助更多广电行

业人员更加便捷的维护业务，且早期资产存储的模式多数为封闭式，采用基于局域网的封闭网络访问媒体资源，局域网外部的其他人员是无法访问的，因此早期的资产管理系统多半是C/S架构的。

随着我国互联网发展的进步，广电行业内的资产也开始慢慢的互相流通起来，追求资源共享，于是一部分B/S架构的媒体资产管理系统[6]开始出现在行业内。

到了Web2.0时代，一些大的视频网络运营公司在以前的媒体资产管理系统之上开发了基于REST和工作流技术[7]以及基于面向服务架构（SOA）的媒体资产管理系统[8]，将媒资管理系统的各种业务场景进行抽象，提炼出各种服务，从而更好的服务广电行业用户。

(3) 时代媒资分发系统目前的情况

时代公司是一家著名的有线电视网络媒体公司，多年来凭借公司的电视网络和其他相关媒体产品在数字资产供应链媒体行业上树立了不错的口碑，公司凭借自己的行业地位和优势近年来开始独立制作自己节目并运营自己的媒体播放平台。伴随着节目质量的上升，越来越多的用户订阅时代的媒体内容，也吸引了更多合作伙伴将他们的媒体内容投放到时代的媒体平台上，随之而来的是大量的数字资产需要被存储，处理和分发并将这些电视，电影节目供应给来自时代公司的电视，网站，手机等多平台的用户。相比与国内外其他先进的资产分发管理系统，时代公司上一代的媒资产分发系统在面临这些新的挑战时表现出很大的差距，无法较好的保证资产的分发和管理，严重的拖慢了公司其他多媒体业务推进，不仅降低了公司同行业的竞争力，还影响了公司多年来积累的业界口碑。

1.2 时代媒资分发管理存在的问题

时代公司上一代的媒资分发系统是一个基于单体架构的媒资分发系统，随着时代公司业务的发展，公司发现系统频繁出现影响严重的问题，甚至影响公司在行业中的声誉，而且日常技术团队跟不上系统新的需求步伐，总是出现业务因为技术推迟等问题。为了解决这一系列问题，本文对时代公司上一代的媒资分发系统作了简要分析，明确了系统存在的主要问题：

(1) 新的资产分发业务功能上线迟缓

老系统基于单体架构已经开发并使用和维护了五年的时间，随着五年来开发团队不断的添加新的功能，老一代的系统的代码库代码库变得越来越庞大，同时随着一些员工的离职和新员工的加入，当年熟悉业务逻辑员工已经离职，导致新员工在维护已有的复杂业务代码上举步维艰，拖慢整体的项目进展。单体架构的代码耦合性高，资产分发的业务逻辑里面掺杂其他业务逻辑，且不同业务逻辑本

身仍然在同一个应用中，这样的高耦合系统导致很多其他问题，比如代码“牵一发而动全身”，为了保证其他功能不受影响，很小的资产分发业务的功能改动也需要大量的功能的回归测试，反复大量的回归测试同样也会拖慢项目进度从而影响业务。

(2) 资产分发系统的可用性差

另外随着时代公司数据量的增加，分发请求的不断增加，老一代系统经常出现内存溢出，分发任务的时间也变得越来越慢。公司不得不多次多费昂贵的价钱来不断升级硬件，但是性能问题仍然不断出现。系统由于性能瓶颈导致的服务失效后，没有任何其他自动恢复措施和策略。

当系统服务出现不可用的情况下，老版本的分发系统会导致系统数据的丢失，这对媒体资产来讲是一个重大的损失，目前公司不得不对重要资产在其他系统内备份以防止文件的丢失，对合作伙伴而言，丢失合作伙伴的资产数据虽然可以要求重发，但是这对影响其他合作伙伴对企业的信任。

(3) 无法有效监控资产分发状态

媒体资产的分发状态是指为了让业务人员更加快速和便捷的查询和获取当前的媒体资产分发的详细状态从而对业务系统进行支持，在时代上一代的媒资分发系统中并没有对有效监控资产分发状态形成重视，因此上一代的媒资分发系统没有对最终的产品资产进行保存和维护，对资产分发异常的排查也只有通过系统日志的低效方式排查，且当分发状态出现异常时，解决方案只能要求上游系统或合作伙伴机构重新发送资产分发请求。这种监控方式费时耗事，而且解决方式不优雅，容易受到其他合作伙伴的投诉。

目前基于时代公司的业务迫切需要以及通过对分析老一代的资产分发系统，时代公司管理层意识到开发一套全新的敏捷高效的媒体资产分发系统迫在眉睫，并意识到提高软件开发的生产效率和软件质量，从而能够更快速的迎合变动的市场，提高企业的整体行业竞争力。

1.3 本文的主要内容

本文首先介绍了媒体资产管理系统在国内外的发展现状，并介绍了时代公司的背景以及媒体资产分发系统和媒体资产管理系统的关系，并对时代公司在业务的扩大和增长的背景下传统的单体应用架构在应对频繁的业务需求的增加和变化所体现出的不足进行了明确的分析，讨论了构建一种基于将业务逻辑细分成不同的逻辑职责单一的微服务架构的重要意义。

在此基础上，对微服务架构相比单体架构在时代媒资分发系统的优势进行了介绍并给出时代媒体资产分发系统的整体的功能结构，讨论了媒资分发系统的主

要功能模块包括：资产元数据整合模块，资产重组打包模块，资产分发模块等，并分析了媒体资产上传流程，视频资产转换流程以及媒体资产分发状态监控流程三个核心系统流程并给出概要设计，并对系统的安全要求给出了需求分析。

完成上述需求分析后，本文给出了时代媒资分发系统的整体设计架构，对资产编目子系统，资产聚合子系统，资产投递子系统给出了详细设计，重点讨论了通过资产聚合微服务将多种不同媒体资产进行整合并通过资产编目服务对现有资产进行增加和修改，比较了单个微服务在快速开发和测试，部署和隔离系统错误上相比单体架构来的优势。最后，从扩展性，重用性，可用性，技术多样性，系统吞吐量，服务上线时间等维度对采用微服务架构和采用单体架构进行比较。

1.4 本文的篇章结构

本文共分为五个章节进行论述，首先介绍了本文的主要工作内容，介绍了微服务架构的基本情况并概要阐述和分析了时代媒资分发系统所面临的现状和存在的问题并基于时代公司的现实情况对时代媒资分发系统的业务需求进行梳理与分析，并围绕着微服务架构对系统进行设计和实现，最后讨论了系统存在的不足之处以及指明后续的工作重点。本文的详细篇章安排如下：

第一章绪论介绍了论文的选题背景。对目前时代公司媒资分发系存在的问题，结合实际情况作了分析和阐述，引出基于微服务架构的媒资分发系统对整合日益增长的媒资分发请求所具有的重要性，并介绍了本文的主要工作内容。

第二章介绍了微服务架构，对微服务的概念进行说明并于传统的单体架构进行比较和分析，明确了微服务架构是更为合适于媒资分发系统的架构，成为媒资分发系统的核心技术架构

第三章主要梳理了时代媒体资产分发系统的需求，对资产分发系统的主要功能模块以及核心业务流程进行了梳理，并提出了系统的安全要求，为详细设计奠定基础。

第四章根据媒体资产分发系统的需求分析，对基于微服务架构的媒体资产分发系统结构进行设计，介绍了核心的微服务划分，并具体的分析了主要微服务，详细的介绍了主要微服务之间的集成和交互。

第五章对基于微服务架构的媒资分发系统的特点和产生的效益进行总结，对系统的不足之处提出了改进措施和方案并指出系统下一步发展的要点以及对未来的展望。

第二章 微服务架构基础

目前，时代公司的媒资分发系统是基于单体架构，然而随着业务的增长，单体架构出现了越来越多的不足，开发效率和软件质量日益下降。当前环境下，为了适应媒体平台的快速发展，公司需要一种更加灵活的架构来适应业务上的频繁变动，从而提高软件开发的生产效率，提高软件质量，更快速的迎合变动的市场，从而提高企业的整体行业竞争力。

微服务架构是面向服务架构（SOA）的一种实现特例，可以被用于构建灵活的，可独立部署的软件系统,服务之间通过使用基于网络且技术无关的通信协议来实现彼此通信从而达到调用目标的过程[9]。

理解微服务的特征是使用微服务架构的一个重要前提。Mike Amundsen 与其合著者在 Microservice Architecture 一书中将 Martin Fowler 和 James Lewis 对微服务的描述总结了七个微服务的特性：规格小，消息型，自管理，限定业务上下文，可独立部署，去中心化，可通过自动化流程构建与发布[10]。

微服务的一个很重要的特征就是小，小不仅仅体现在代码量上，因为代码量的大小很难定量，最重要的是体现服务的职责上[11]。每个服务的职责应该是单一的，这个是对单一职责在服务级别的一种应用。

消息型是指微服务通信方式应该是一种轻量级的基于网络且技术无关的通信协议，通常是一种基于资源的 HTTP API[12]。

限定业务上下文的特性表明了微服务对业务驱动和领域驱动的拥护，微服务提倡围绕业务的上下文来定义微服务的接口，而不是围绕技术概念来定义，同时限定业务上下文可以在微服务数据建模上进行指导[13]。

自管理的特性是增加架构灵活性的非常重要的一种体现，每一个微服务开发的实现方式可以由开发团队自行决定，这样一种特性同样依赖于微服务的规格较小，在复杂度不高的情况下对一个微服务的设计决策不需要太多其他方面的知识，开发团队可以主导微服务的实现方式[14]。

可独立部署的特性被 Martin Fowler 认为是可以用来作为应用划分微服务的原则。通过微服务独立部署，业务需求的变动和错误的容错可以被隔离到微服务级别[15]。

去中心化特性描述了微服务采用 Unix 的思想，使得微服物端拥有更高的自治性和智能从而每一个微服务掌控自己相关的业务逻辑而不是统一交给业务逻辑中心，比如采用轻量级的消息中间件而不是向企业服务总线一样的智能中间件[16]。

可通过自动化流程构建与发布的特性是为了更好的利用微服务的灵活性和

可独立部署的特性，随着微服务数量的增加，手工的构建和发布是很恐怖的，利用自动化的工具将这个过程自动化，才能享受每一个微服务在可独立部署这个特性上面带来的优势。

本章内容首先介绍了什么是微服务架构，随后围绕着媒资分发系统对微服务的集成与协作进行详细介绍，随后对比了单体架构和微服务架构，明确了使用微服务构建时代公司下一代媒资分发系统。

2.1 微服务的集成

本小节将会在同步调用和异步调用的基础之上来讨论现有常用的微服务集成方式以及在媒资分发系统中采用的方式并介绍其理由。

（1） 共享数据库

共享数据库指的是多个不同的微服务同时分享一个数据库从而达到集成交互目的，这种方式有三个缺点因此没有在媒资分发系统中没有被采用。

①无法隐藏技术实现的细节，若要对数据库的结构发生变化，需要增加额外的沟通成本，违背了微服务高度自治的原则，导致代码的改动需要更多的回归测试。[17]

②共享数据库导致了技术选型的固定，例如一个微服务比较适合用MongoDB来做存储，则所有共享该数据库的微服务必须要选择MongoDB，然而可能另外一个微服务事实上更适合关系型数据库。

③要在所有共享的微服务之间都需要维护数据交互的逻辑。

共享数据库是不同微服务的一种紧耦合和低内聚的一种方式，所以本文没有采用这种方法进行微服务间之间集成，相反，对数据需要存储时，媒资分发系统中的每一个微服务都会将自身相关的数据存储到单独的数据库之中。

（2） 请求响应式：

目前比较常用的请求响应式的集成交互方法有RPC和REST，他们都是基于HTTP协议。在一个标准的HTTP请求由两部分组成：方法和资源，对于每一个方法而言都有其特殊的意义和特性，但大多数的RPC的一般只是使用了HTTP的GET和POST的方法。[18]

虽然REST并没有规定一定要使用HTTP协议，但是REST支持所有HTTP的方法，REST能够更简洁和统一的表明对资源的操作，又能够利用HTTP成熟的技术和生态圈使得上手REST不像RPC一样需要额外的学习，比如SOAP协议虽然是架设在HTTP之上但是摒弃了很多HTTP的特性，因为资产分发系统并没有对通信延迟有过高的要求，同时考略到上文考略到使用REST协议带来的便利，最终采用了基于HTTP 的REST作为了媒资分发系统的同步通信机制[19]。

（3） 基于事件的异步方式

对于基于事件的微服务通信而言，最重要的是去考虑如何去发出事件和接受通知事件，而这些都可以很好的被消息队列来实现，通过在微服务架构中引入消息队列，系统可以从以下几个方面受益：

①提供了异步处理机制

通过消息队列进行互动的微服务不需要对方立刻作出回应，这对时间比较长的任务比如视频编码需要比较长的时间，通过队列服务，客户端服务只需将需要编码的视频请求放入队列便可继续接受其他的请求[20]。

②提高系统的容错能力

对于特定队列的监听者微服务而言，如果微服务在出现宕机等失去服务能力的意外事件时，任务可以存储在消息队列中，在微服务恢复服务后可以继续消费队列，从而提高容错性[21]。

③为系统提供短时大量请求的缓冲和负载均衡

当大量的请求到来时会增加系统的负载，微服务的处理能力也将会受到大量负载的影响，短时间的大量请求会导致系统的崩溃，利用消息队列可以对短暂的大量请求进行缓存，处理请求的微服务可以根据自身的能力来适当的获取任务进行消费。

根据上文的描述，在基于微服务的媒资分发系统中，服务之间的主要集成分为了基于 HTTP 的 REST 来同步通信和基于事件的异步通信方式为各个微服务之间主要的集成与通信的方式。

2.2 微服务的协作

为了更好的支撑业务的流程，媒资分发系统中的各个微服务之间需要按照业务流程进行调度与协调，一般而言服务之间的协作有两种模式来实现这一目标：服务编制与服务编排[11]。

（1） 服务编制

服务编制的模式需要有一个中心的编制器来控制业务流程的运转，并且负责控制何时来调用相应的服务并组织调用结果以满足业务流程的需要。这种方式主要以一个编制器为中心来统一协调，配置和管理。

（2） 服务编排

服务编排的模式不需要一个统一的流程管理者，各个微服务间以一种去中心化的方式协同工作。其表现为各个微服务在运行期间根据其他微服务的行为来执行自己的行为，而不是依靠一种统一的管理者来指挥。

服务编排和服务编制在实现微服务的协作上并非是互斥的，意味着一个系统

的架构里既可以包含服务编排的模式也可以包含服务编制的模式，在本文设计媒资分发系统的第四章会具体介绍这两种模式的实现。

2.3 微服务架构与单体架构的比较

本节结合时代公司的具体背景从维护性，扩展性，重用性，可用性，技术多样性等不同纬度对时代上一代媒资分发系统的单体架构和微服务架构进行了对比。

（1） 重用性和可维护性

单体架构的整体代码库因为数量庞大，维护的成本变的越来越高， 原因是越来越大的应用程序会变得难以理解和修改。此外，因为没有硬模块边界，模块化随着时间的推移而模糊而导致的模块职责混乱，导致代码难以理解和正确的修改和重用。

微服务架构受益于规格小和高自治性可以使得微服务的代码量更少，开发者自己维护自己的代码使得代码质量相对可控。微服务的单一职责也使得微服务之间的重新组合来支持全新的业务流程变得更加的容易。

（2） 可扩展性和可用性

从应用程序的扩展性角度来看，单体架构的扩展方式是相对昂贵和单一的。单体架构的系统通过运行更多的应用程序副本和服务器实例来实现水平方向的扩展，对于某些系统的性能瓶颈仅仅是某一个模块，这种方式的扩展会造成很多的资源浪费。比如不同的应用程序组件具有不同的资源需求，一个可能是 I/O 密集型，而另一个可能是内存密集，使用单体架构，系统无法按照需求独立的扩展每个组件。

从可用性角度分析，单体架构对错误容忍相对较低，意味着一个功能的不可用有可能会级联并导致所有的其他功能都不可用。

微服务架构凭借自身的分布式优势和凭借单独部署的特性对有不同资源的需求服务进行单独扩展，另外在程序失败的时候，微服务架构能够更容易的将错误隔离从而保持其他服务的正常运营。

（3） 技术多样性

单体架构的应用程序的技术栈的可替换性相对较差，一般一个单体架构的应用程序在开发后更换的成本会越来越大，也就是说随着开发的时间越来越长，很难在去更换技术栈，甚至技术框架的版本升级都会变得非常的困难[14]。

（4）架构复杂性

相比于单体架构的分层结构，微服务架构作为分布式系统本身必然会有更高的复杂性。表现在分布式系统的因网络延迟导致的性能问题，异步实现机制导致

的开发复杂度以及为考虑实现成为而不得不采用最终数据一致性等。

通过上文的分析可以得出，在扩展型，可用性，维护性等纬度并结合时代公司的业务背景，微服务架构是时代公司新一代媒资分发系统更为合适的架构选择。

第三章 时代媒资分发系统需求分析

媒资分发系统以努力推进资产供应链的重新架构为出发点，具体而言，主要目标是实现一个现代化的，可扩展的，自助服务解决方案，汇总并提供时代公司的成品数字资产，最终服务于时代公司内部和外部的合作伙伴。通过对分发系统的需求进行分析可以得到三个核心功能模块：资产元数据整合模块，媒资分发模块以及媒资转换模块，以及三个核心业务流程：资产分发状态监控流程，资产与元数据上传流程，视频资产转换流程。本章对资产分发系统的三大功能模块和业务流程进行详细分析，为下一章子系统的设计与实现奠定基础。

3.1 媒资分发系统的主要功能

媒资分发系统的基本用户角色分为：系统管理员，企业内部用户，企业合作伙伴用户，企业业务支持用户。这些用户的需求构成了设计媒资分发系统的重点，也是系统核心功能的所在，图 3-1 描述了系统的功能结构。

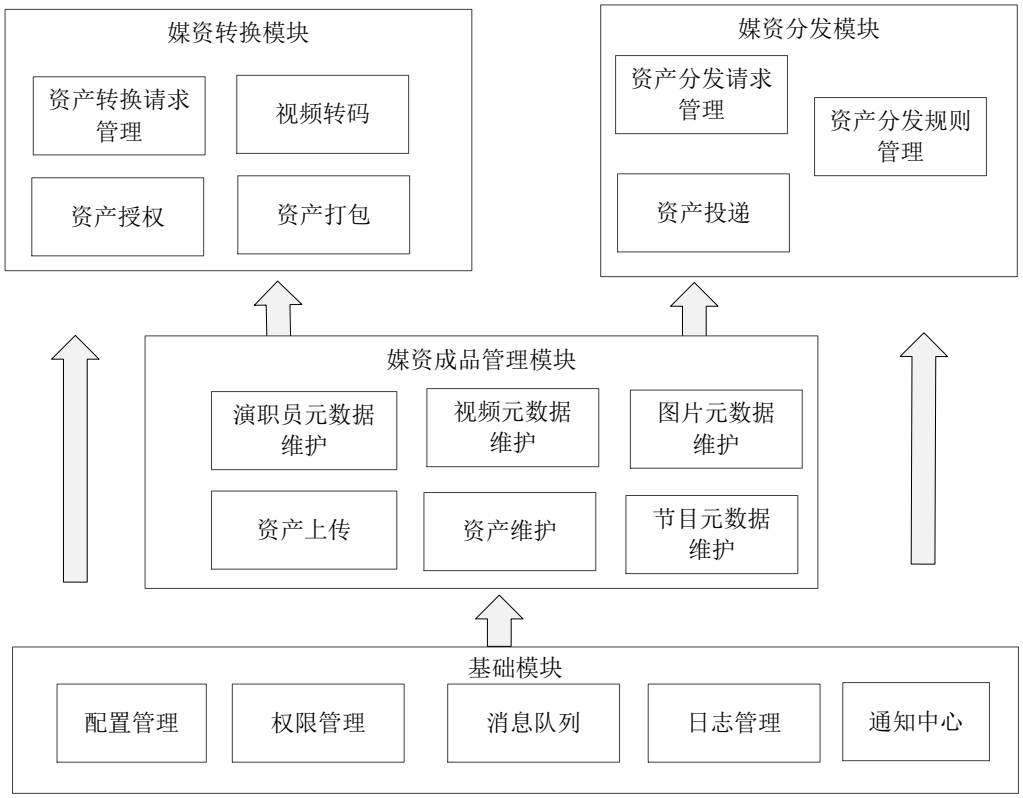

图 3-1 时代媒资分发系统功能图

3.1.1 资产元数据整合模块

资产元数据整合模块需要对多种媒体资产成品进行注入整合和管理，分为资产以及资产元数据的上传注入，各种媒体资产的维护和各种媒体资产元数据的维护三个部分，主要的用户包括的企业内部系统，企业内部的业务人员，以及合作伙伴的第三方系统等。资产元数据整合模块是分发系统的所有其他模块提供服务的前提，媒资分发模块和转换模块需要从资产元数据整合模块获取所需的媒体资产。资产元数据整合模块的功能用例图如图 3-2 所示。

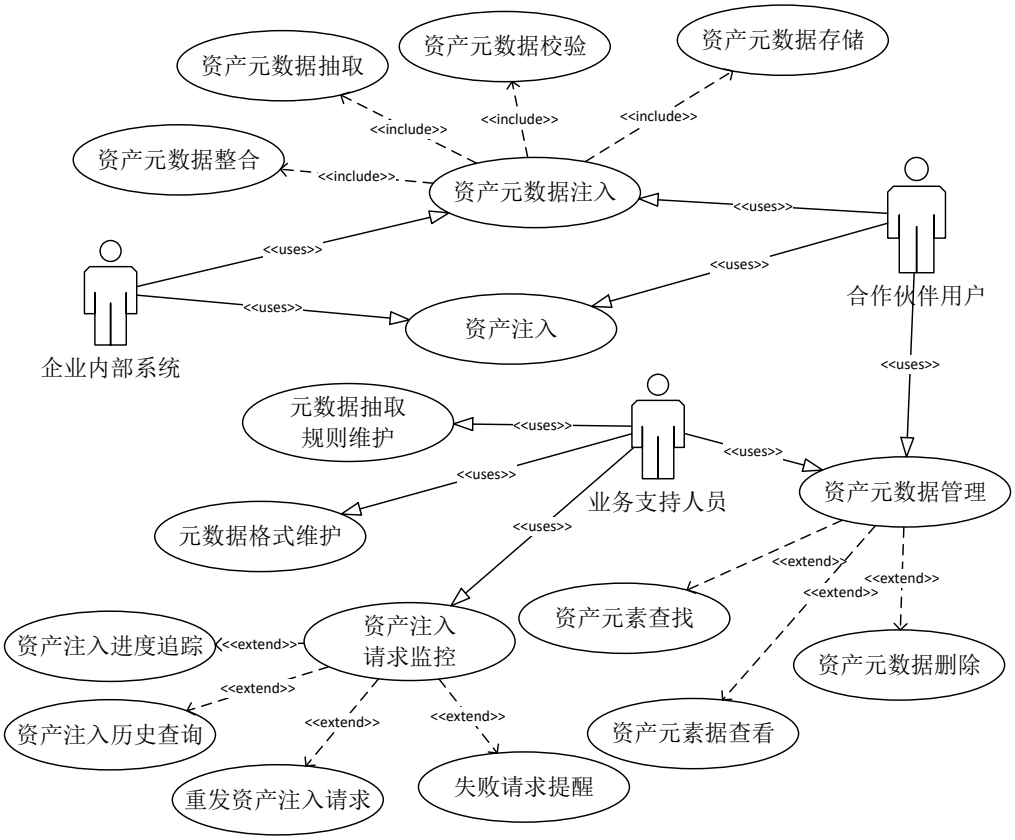

图 3-2 资产元数据整合模块的功能用例图

（1）资产与元数据注入

媒体资产注入的功能简单地定义了媒资分发系统从企业内外部获取媒体资产的过程。注入资产的请求来自企业内部系统同时系统也支持外部的合作伙伴的第三方服务或者用户向时代公司发送资产，从而进行相应的资产分发。作为整个媒资分发系统的一个入口模块，所有进入此系统的请求都应该进行认证和授权，确保用户被分配到正确的角色以获取相应的操作权限是其他操作的前提。资产注入的功能是媒资分发系统获得媒体资产的唯一途径，需要配合资产元数据的注入一同使用，当企业内部系统或者外部系统注入新的资产后并收到媒资分发系统的通知后，相应的用户可以进行资产元数据的注入。

资产元数据的注入功能作为资产注入的后续功能，内部系统或者合作伙伴将资产注入到媒资分发系统后需要对所注入的资产进行相应的描述，资产元数据注入便可以提供这样的功能，并且元数据注入后需要对注入的资产元数据进行验证用来检查是否符合元数据格式的基本要求，对格式不正确的元数据会给出相应的错误信息，其后是经过对资产元数据关键内容的抽取，元数据中可能包含大量的信息，但是与分发系统而言并非所有的信息是有意义的，抽取过后的元数据信息可以对元数据内容进行整合后转换为内部的数据模型格式，最终需要存储到系统内部的媒体成品管理模块以备后续使用。

（2）资产注入请求监控

资产注入请求的监控为业务支持人员提供了更便捷的方式来得知当前资产注入的情况，当大量的企业内部系统或者合作伙伴将资产注入到系统后，通过日志监控注入情况显然是低效的。资产注入请求监控的功能允许支持人员登录到系统上并通过系统的身份认证以及权限检测从而根据用户的权限来显示资产注入情况，资产注入请求监控允许业务支持人员对指定的资产注入请求状态跟踪，并可以历史的资产注入请求进行浏览，用户可以根据指定元数据的关键字对资产注入请求进行过滤从而快速的得到资产注入请求的状态信息，也可以按照多种方式组合查询排序来获取资产分发请求信息。

资产注入请求监控的核心价值体现在对异常状态下的资产注入请求处理上，当一个异常的资产注入请求发生时，业务支持人员会相应的收到系统定制的提醒邮件，业务支持用户可以通过查询历史注入请求得到失败的注入并作出相应的业务决策即是否需要对此请求重新发送，或通知相应的开发人员追查失败原因后再重新进行注入请求的尝试。

（3）资产元数据管理

资产元数据管理允许业务支持人员和合作伙伴查看媒资分发系统中存储的成品媒体资产。资产元数据的管理并不提供成品资产元数据的修改，因为理论上进入到媒资分发阶段的产品都已经是最终的成品，媒资分发系统提供资产转换的功能但也只是简单的转换并进行不加工与修改。

对于企业内部的成品资产分发，元数据管理功能使用的场景是通过业务支持人员来核查是否目前资产分发系统内存储了期望的媒体资产，同时业务支持人员有权利删除存储在系统内部的成品媒体资产，但没有权限删除合作伙伴的成品资产。

对于企业外部合作伙伴，元数据管理功能只允许其查看或者搜索自己分发的媒体成品资产而没有权限查看其他合作伙伴或者时代公司的成品媒体资产，有且仅有权利删除其自己上传的相关资产信息。元数据管理功能为合作伙伴和时代提供了一个基于多租户的维护成品资产的管理平台。

3.1.2 资产重组打包模块

通常意义上来讲，转换是将一种事物转化成为其他事物的过程，但是从媒资分发系统的角度来看，这通常仅仅是意味着媒体资产的格式转换。时代的媒资分发系统之所以能够接受多种不同媒体资产来源是因为它具备将一种格式的媒体内容可靠地转换成另一种媒体内容的能力。一种常见的媒体资产转换需要发生的是编码，授权和包装的资产的打包。对于资产重组打包模块而言，另外一个重要的功能就是对资产转换请求的管理和事件通知。

通过上文对资产重组打包模块需求的分析，可以得到资产元数据整合模块的用例图如图 3-3 所示。

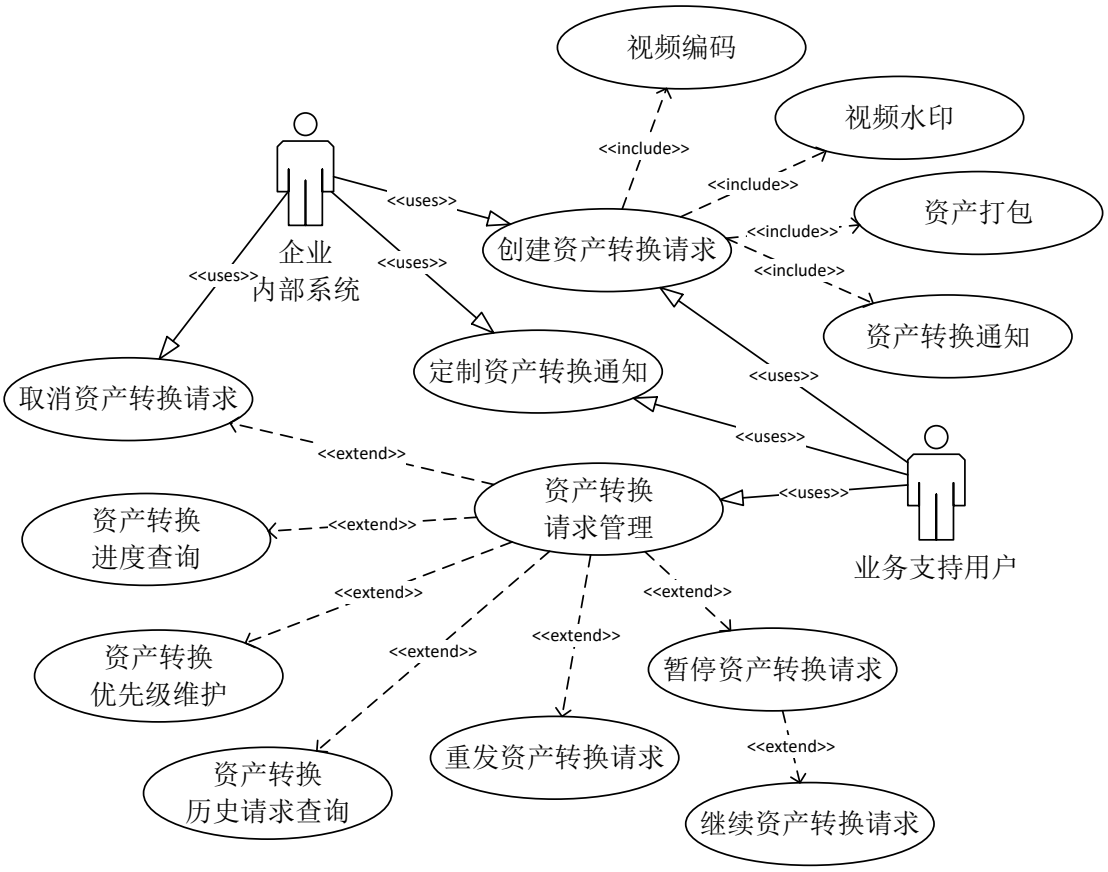

图 3-3 资产重组打包模块功能用例图

（1） 创建资产转换请求

资产转换请求可以由系统根据资产元数据的生成而进行判断是否需要进行创建资产转换请求，通过解析视频资产的元数据对视频资产作出相应的编码以及其他处理。与此同时，资产重组打包模块允许其他企业内部的系统和业务支持人员对资产重组打包模块发出请求从而进行资产转换操作。

一个完整的资产转换请求包括对视频资产按照规格要求进行编码处理，对有

授权要求的视频资产创建水印以及集成相应的字幕信息等操作，在完成这些操作后资产重组打包模块会对转换的资产进行打包操作，按照请求中指定的存放位置对打包后的资产进行存储。在资产转换完成后可以根据用户或系统指定的通知格式与方式发出相应的资产转换通知，告知用户或系统资产转换已经完成。

（2） 定制资产分发通知

资产转换是一个需要耗费时间较长的过程，对于业务支持人员和其他系统而言，知道资产转换的结果是很重要的，通过定制资产分发通知，业务人员可以制定接受自己感兴趣的资产转换从而接受系统的提醒信息已作出相应的下一步决策。对于系统而言，可以在接受到转换通知后进行下一步的业务处理。

（3） 资产转换请求管理

为了更好的帮助业务支持人员管理和维护媒资分发系统的转换请求，资产重组打包模块需要提供转换请求管理的功能。对于业务支持人员而言，对资产转换过程的掌控和干预是必要的。当大量资产的转换请求堆积时，业务支持人员需要通过制定和调整优先级的策略来使得需要优先处理的请求能够被更早的处理，这就意味着用户同时必须具备对所有资产转换请求进行查找和搜索的功能，进度查询功能可以帮助用户知道当前资产转换具体进行到哪一步，业务支持人员可以通过在系统上对过程必要时进行暂停以及继续。

为了更还的提高媒资分系统分发资产的维护性，对出现异常状态的资产转换请求应当可以被重新处理，从系统的角度而言，资产转换请求的管理更大的程度上是在帮助业务支持人员更好的为紧急业务和异常业务而服务。

3.1.3 媒资分发模块

分发的简单定义是发送或拷贝某些媒体内容或媒体资产到指定的目的地的过程。媒体内容可以是媒体相关小文本文件如资产的字幕信息，也可以是非常大的二进制对象如视频信息。从媒资分发系统的角度来看，分发的对象可以是媒体资产或者媒体元数据。

作为一个分发系统，至关重要的是其能够从不同的的粒度来追踪哪些资产已经被其所分发，哪些资产的分发是在计划之中的,哪些资产在分发的过程中失败了等，而也正是媒资分发系统的首要职责。目前资产分发模块的使用用户有企业其他的内部系统，业务支持人员，业务主管人员。

从功能的角度来看资产分发模块需要对资产分发的规则进行维护，按照内部业务人员制定的分发规则并确定是否，何时，何地进行资产的分发。资产分发请求的管理，最后分发模块需要将这些媒体内容通过指定的途径分发到目标位置。通过上文对资产分发模块需求的分析，可以得到资产元数据整合模块的用例图如

图 3-4 所示。

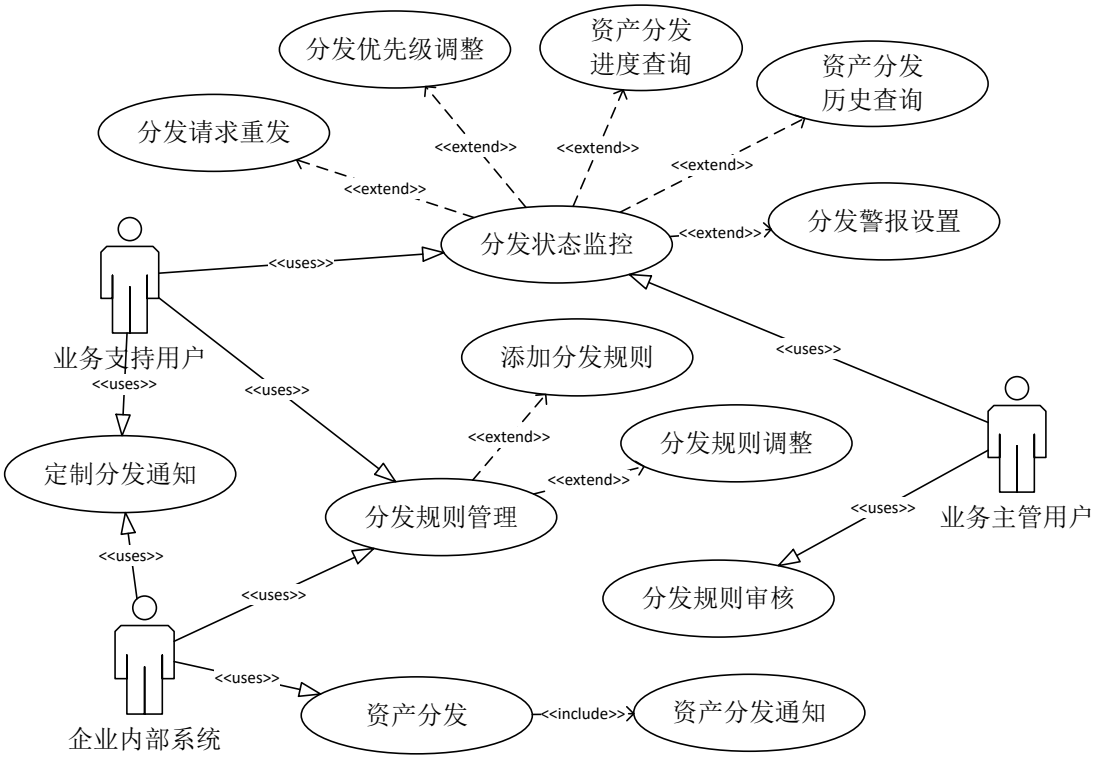

图 3-4 资产分发模块用例图

（1） 资产分发规则管理

分发业务模块作为时代媒资分发系统的核心业务是唯一一个需要业务主管用户参与的模块。资产分发模块的核心功能是业务规则的管理，用户对分发规则的管理可以直接影响到分发的结果。业务支持人员可以创建一个新的资产分发规则，随后业务规则会被搁置为待审核状态，业务主管人员可以对所有待审核的分发规则作出通过审核和拒绝通过审核。

被决绝通过的审核需要业务支持人员对业务规则作出调整后再次提交给业务主管进行审核，未通过审核的分发规则不会生效即不会创建任何与其相关的资产分发请求所以不会对媒资分发的流程造成任何影响。

被业务主管审核通过的分发规则会去计算分发所需要的媒体资产，若具备所有分发所需要的成品资产，则相应的资产分发请求会被创建。不具备的资产会等待需要的成品资产存入资产元数据整合模块后在进行创建分发请求。

（2） 分发状态监控

分发状态监控允许业务支持人员和业务主管对分发请求记录进行查找从而对分发的状态进行跟踪并在业务需要的情况下来修改资产分发请求的优先级，优先级高的分发请求可以被分发模块优先的处理。当异常的分发请求出现时，业务

支持人员和业务主管可以重新发送指定的分发请求。

（3） 资产分发

资产分发功能是对分发请求处理的过程，一般而言资产的分发是由系统根据对分发请求处理而自动发生的过程，所以业务支持人员无法直接干预一个资产分发的开始，但对于内部系统而言，资产分发模块应该开放功能接口并允许通过认证的内部系统直接调用资产分发，从而起到更高效率的利用分发模块的效果。

3.2 媒资分发系统核心流程分析

通过对媒资分发系统的主要功能的分析，发现若要从业务角度出发，更好的划分服务和功能职责，需要对媒资分发系统的核心流程进行分析，经过时代公司业务的整理可以得出资产分发的如下核心流程：视频转换流程，成品媒体资产的上传流程以及媒体资产状态分发监控流程。这些核心流程是时代媒体资产分发系统需要处理的主要工作。视频资产作为时代公司最为重要的媒体资产之一，视频转换的过程是指视频编码，水印处理和视频打包操作的总称，视频转换的流程是支持资产的衍生和授权处理的重要过程。成品媒体资产的上传是时代媒资分发系统唯一的成品资产如口，上传流程是资产顺利分发的基本，只有资产顺利被上传到媒体分发系统才能做资产转换和分发等。媒体资产状态监控流程是对业务人员对媒体资产分发过程的监控，在媒体资产的分发出现异常或故障时，业务人员能够根据业务要求进行处理。

3.2.1 媒体资产上传流程

外部合作伙伴通过调用媒资分发系统提供的成品资产管理模块来上传媒体资产和对应的媒体资产元数据，本节对资产与元数据的上传流程加资产与元数据上传流程以详细的解释与说明。

首先，作为合作伙伴在时代媒资分发系统上传成品资产的前提条件，业务支持人员需要在媒资分发系统提供的授权服务内为外部合作伙伴配置相应资源的访问权限，并获取到合作伙伴提供的元数据使用规格，在对新的元数据的验证规则进行设置，随后需要配置元数据的抽取规则，同时注入模块需要整合元数据数据并按照符合时代公司内部的数据模型将元数据信息存入。

准备好这些配置信息后，合作伙伴便可以将媒体资产上传到媒资分发系统上，在资产上传成功后，将媒体资产的元数据传递给媒资分发系统上来，媒资分发系统收到后需要对元数据进行验证，只有媒体元数据通过验证以后才可以进行元数据内容抽取环节。作为一个媒姿分发系统，合作伙伴提够的资产元数据并不

是所有信息都是需要的，分发系统只需要关注其中的一部分用来将元数据信息转化为系统内部的统一的数据模型的关键信息，有了这些必要的信息，它便可以按照一定的规则将这些数据转换成本文定义的数据模型存入，并完成资产的上传流程。图 3-7 描述了上述分发流程：

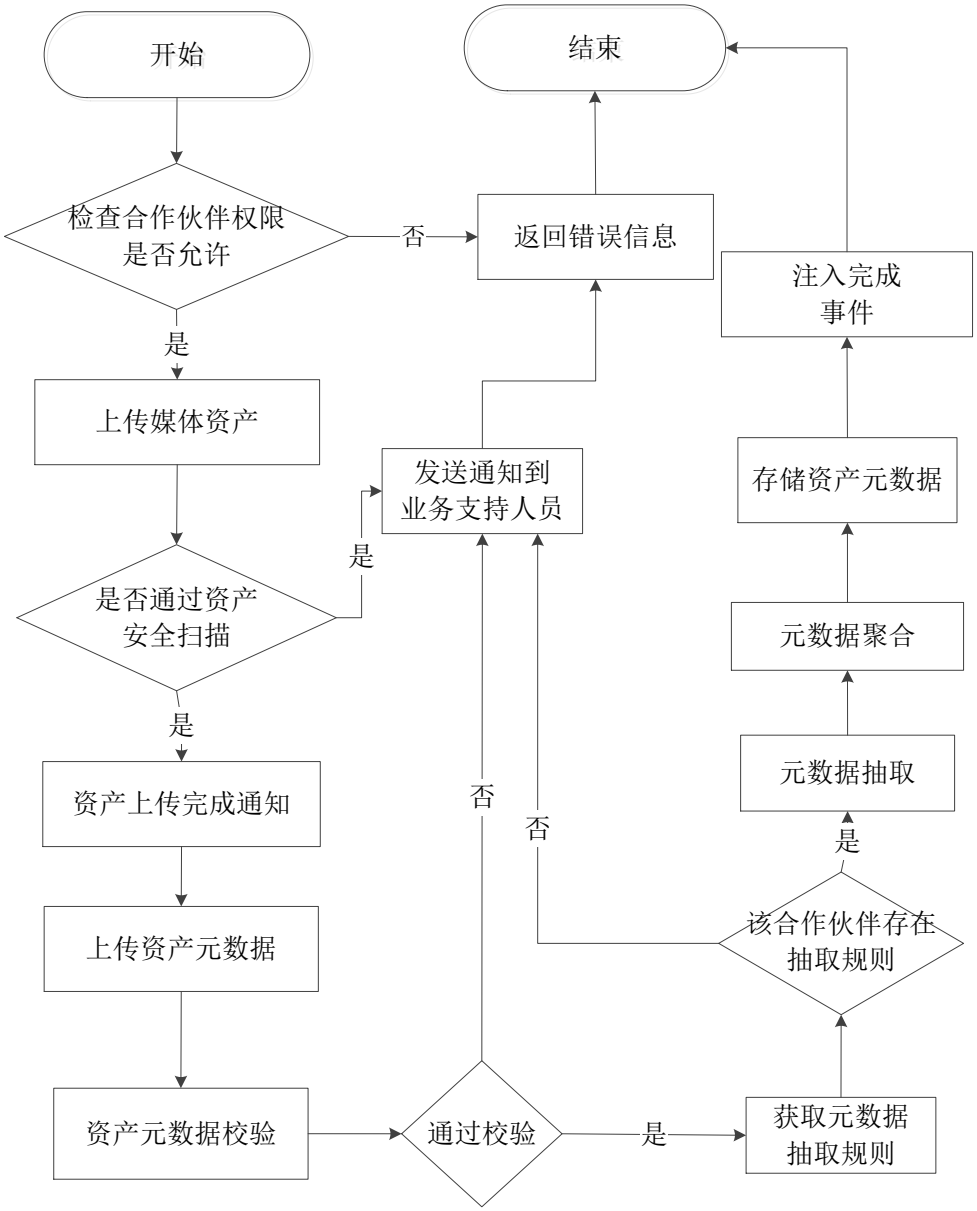

图 3-7 资产与元数据上传流程

3.2.2 视频资产转换流程

当一个新的媒体资产被成功上传后，一个来自资产元数据整合模块的通知会使得资产重组打包模块触发资产转换的检查，通过判断资产的类型是否为视频元数据信息来决定是否需要开始视频转换流程，若资产检查是视频元数据资产，则根据视频资产的元数据信息去查找是否能找到其对应的视频文件，若可以找到视

频文件，则开始进入转换的过程，若没有找到视频文件就记录当前视频资产元数据信息，等待一段时间再去尝试，若超过最大尝试次数，则通过资产转换报告发送异常报告。

若能找到对应的视频资产，则开始对视频资产发到对应转码服务器上进行转码操作，一个资产转码中的状态会被资产转换事件发给定制的人。视频转码完成后会发送更新事件，更新此前的状态为资产转换完成，并检查视频元数据中是否指明需要水印授权，若有水印授权的需求，需要资产进行水印处理。

进行水印处理需要获取授权信息，获得授权信息后方可相应的生产水印并处理视频。并发送水印处理的通知事件，若视频元数据信息中未标明需要进行水印处理则可跳过此步骤，进行资产打包操作。经过资产打包操作后，资产转换的流程就已经完成，此时会生成资产转换完成的通知事件并将转换后的资产和资产元数据进行存储。图 3-5 描述了此视频转换流程。

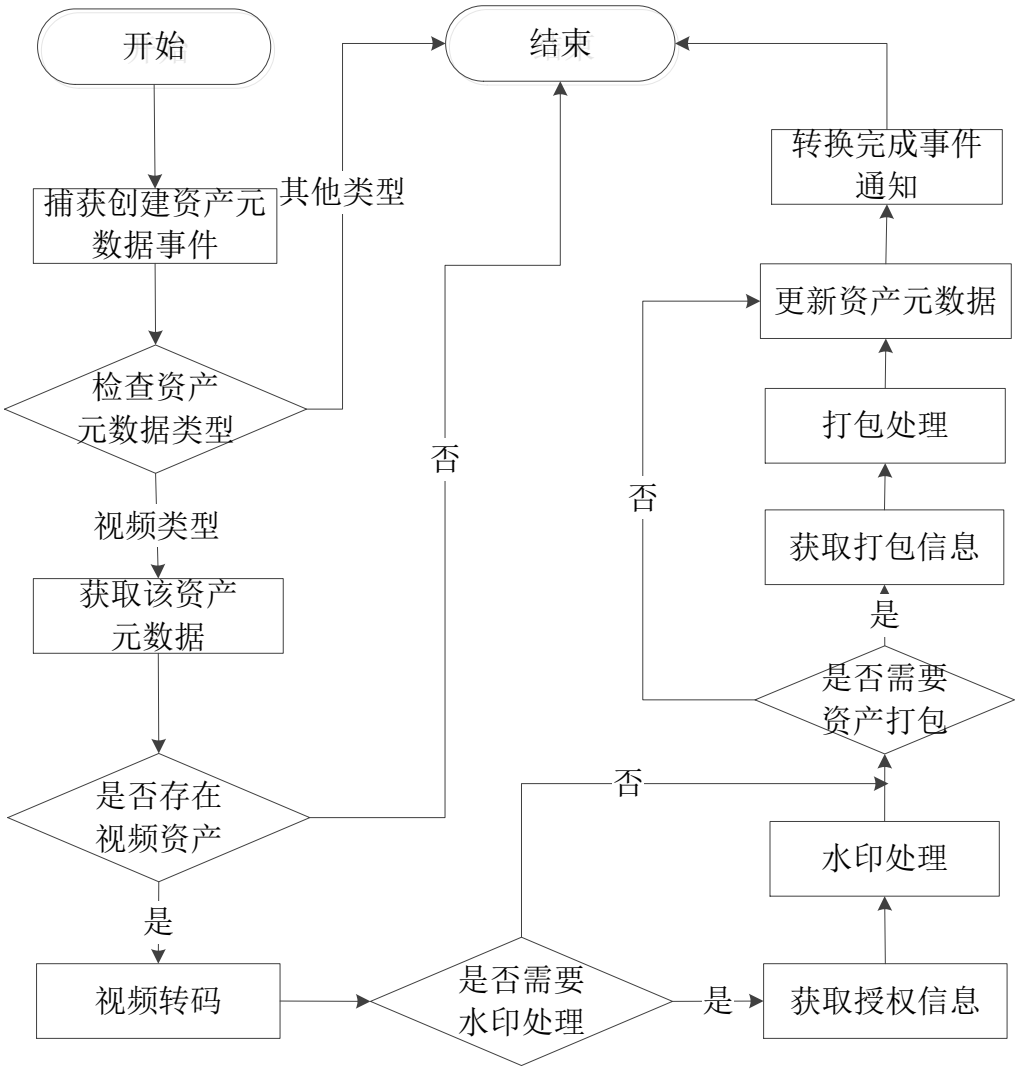

图 3-5 视频资产转换流程图

3.2.3 媒体资产分发状态监控流程

在媒资分发系统中，一个重要的功能就是对媒体资产分发状态监控，无论是合作伙伴资产抑或是自营媒体资产的分发，都有可能会出现由于各种原因导致的分发错误，从而导致媒体资产没有按照规定的时间送达到下游系统或者合作伙伴所指定的其他系统。媒资分发系统具备的功能是能够根据错误的情况通知分发系统的业务支持人员从而在必要的时候引入人工干预，对一个媒资分发请求进行重发。

一个媒体资产分发请求出现后需要根据错误信息进行甄别属于哪一阶段分发错误，从而有针对性的进行重发，资产分发监控的具体流程如图 3-6 所示。

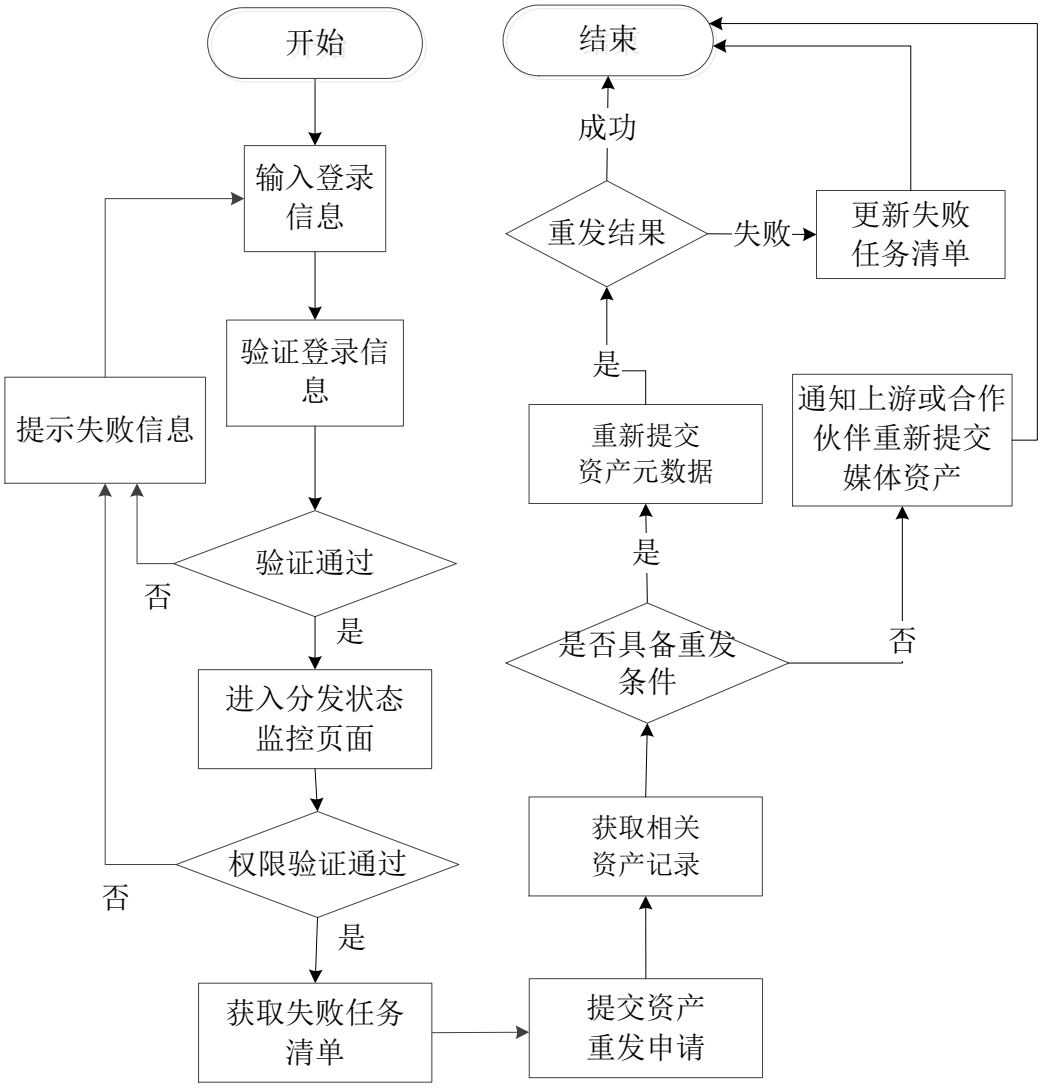

图 3-6 媒体资产状态监控流程

时代媒资分发系统的业务人员需要使用自己的身份信息登录到媒资分发系统的控制台页面，通过权限认证后业务支持人员可以进入分发状态监控页面对

对当前系统的分发记录进行追踪和监控，通过获取失败任务清单，业务支持人员可以得到所有分发失败的请求，并可在系统页面上得到具体失败的原因，通过提交资产重发申请可以查看该分发请求是否具备了分发请求重发的条件，在条件具备的情况下，业务支持人员可以提交分发记录的重发请求，重发请求被触发后会被排在队列中等待处理，此时业务人员可以根据业务实际情况对当前请求的优先级作出调整，从而加快分发请求被执行。资产分发重发请求被处理后，如果任务仍然失败，则该请求仍然会在失败任务清单中，此时便需要业务支持人员联系技术支持人员查看分发具体失败的原因。分发请求监控可以帮助业务支持人员更快，更方便的定位问题，并对大多数问题提供了相应的解决方案。

3.3 媒资分发系统的安全需求

时代媒体资产分发系统内存储这大量的来自企业内部和合作伙伴机构的成品媒体资产，包括电影，电视剧，宣传片等等，这些都是媒体行业重要的资产。意外丢失或者被盗取都不仅仅意味着财产的损失可能会体现外泄未上映的资产导致企业的口碑和声望收到严重的影响。由于媒体资产分发系统面向多个企业的合作伙伴，所以需要对外网开发端口，于是就需要更多的考虑网络安全问题，如如何应对受到恶意网络攻击等。为了确保时代公司和其合作伙伴的成品媒体安全，媒资分发系统需要考虑以下几点安全需求：

（1）与合作伙通信的安全需求

对与媒体资产分发系统运行的基础设施层面而言，需要考虑的是网络层连接的安全设计，对与资产分发系统与企业内部系统之间的调用是基于公司的私有网络，可以收益于公司内网的安全策略。由于与合作伙伴的通信需要通过公网，需要使用客户端证书机制来确保请求主体的身份，并对合作伙伴对资源的访问权限进行设置。

（2）页面控制台的安全需求

页面控制台的使用者是基于浏览器的普通用户，控制台页面与用户之间的连接采用 HTTPS 的方式进行数据之间的传输从而确保用户的登录名密码等身份信息不会在 HTTP 传输的过程中泄漏。页面控制台用户的身份信息需要通过用户管理服务和认证服务进行认证以确保控制台页面有权利访问相应的业务相关的服务。对于拥有业务决定性权利和拥有特权的用户需要通过 MFA 来对身份进行认证，提高安全系数。

第四章 时代媒资分发系统设计

根据对媒资分发系统的需求分析结果，并满足不同场景下的分发请求，本文采用了微服务架构对整体的系统进行设计，利用了服务编排来进行不同微服务之间的通信，分别采用了基于非关系型数据库的 MongoDB 数据库对数据进行持久，为了最大化发挥微服务优势，本系统将大部分服务部署在云端，并尽可能的利用云服务来实现大量的自动化任务，从而进一步达到节约成本，缩短开发周期等目的。

在本章中，首先给出了一个基于微服务架构的媒资分发系统的总体架构设计，按照功能和业务的需求对各个核心模块给出了一种微服务的划分方式，并对各种微服务之间的协作方式进行介绍，对本系统采用的协作方式进行解释说明。随后对系统三个核心子系统进行详细设计，最后将本系统与其他类似系统架构在业务运营，成本，安全性等层面上进行对比，并对系统的可用性，持久性，可维护性，扩展性，安全性等方面对整体架构进行了对比说明。

4.1 媒资分发系统总体架构设计

媒资分发系统的逻辑架构通过分层的手段将不同的功能划分在不同逻辑层次便于对功能的梳理和维护，总体上媒资分发系统的逻辑架构分为四个层次：接入层，业务微服务层，基础业务微服务层以及数据层。

数据层通过数据库将媒体资产和用户数据等重要数据保存，基础业务微服务层通过数据访问层对数据进行使用。基础业务微服务不直接服务于接入层而是服务与业务微服务层。基础业务微服务包括系统通用服务服务以及对业务领域数据封装的微服务。业务微服务层通过使用消息队列或者是 REST 与基础业务微服务进行通信。业务微服务层可以对接入层提供服务，接入层如控制台页面以及合作伙伴的系统可以通过 REST 方式与业务微服务进行通信。

软件架构方面本文采用微服务架构来构建媒资分发系统。相比其他架构，微服务有很多优势,首先，系统内部的各种职责被分成小的服务，比如内部的资产元数据整合模块就可以简单的根据面向的对象不同划分成不同的服务，如内部资产注入适配微服务，如外部资产注入适配微服务。

职责清晰并且代码量不大可以很大程度上来降低维护成本，缩短开发新版本和更新旧版本的时间。其次，微服务架构中拥有多个微服务组成，可以使用技术无关的轻量级的通信协议，在本文的系统设计中，采用 REST 和消息队列来进行通信，这一特征在时代的媒资分发系统中是非常重要的，因为受益于各个不同服

务间面向的技术无关通信协议，可以采用不同的语言技术来实现各个微服务，从而充分的发挥一些语言的优势。

图 4-1 资产分发系统逻辑架构图

例如在分发模块里，有的合作伙伴需要时代公司的最终资产投递到 AWS S3 的桶里面，则可以利用 Python 的脚本语言的优势，通过几十行代码来实现一个 S3 投递服务。服务架构的可独立部署的特性是保障业务需求变动的前提，通过微服务独立部署，业务需求的变动和错误的容错可以被隔离到服务级别。

在媒资分发系统中，若一个合作伙伴改变了其接收数据资产的格式，分发系统可以通过对投递规则服务进行相应的修改，然后将修改后的服务替换掉产品线上的服务而不影响其他业务的正常运行。

利用微服务架构可以更加容易的实现具有针对性的垂直和水平扩展，在媒体

分发系统中，注入模块需要加载大量的元数据文件进行定制化的解析，属于内存密集型的模块，单体架构中必须为了这个模块而升级服务器，因为所有模块都部署在一起。相反，在微服务架构上，只需要对一个服务的模块进行升级即可。

在媒资分发系统的架构设计中既采用了服务编制的模式，也使用到了服务的编排模式。从整体的系统架构层面系统采用了编排模式。媒体资产的注入，转换和分发并没有一个流程管理者在控制系统的流程，而是各个模块在运行时动态的相互协作完成了媒体资产的注入，转换和分发的流程。媒体资产的元数据在注入到系统中后会生成相应的事件。资产转换模块会监听自己感兴趣的事件来确定是否需要触发视频编码等操作，资产分发模块同样会根据事件来决定是否触发资产的分发。图 4.2 从宏观的角度描述了几个子系统中的服务编排模式。

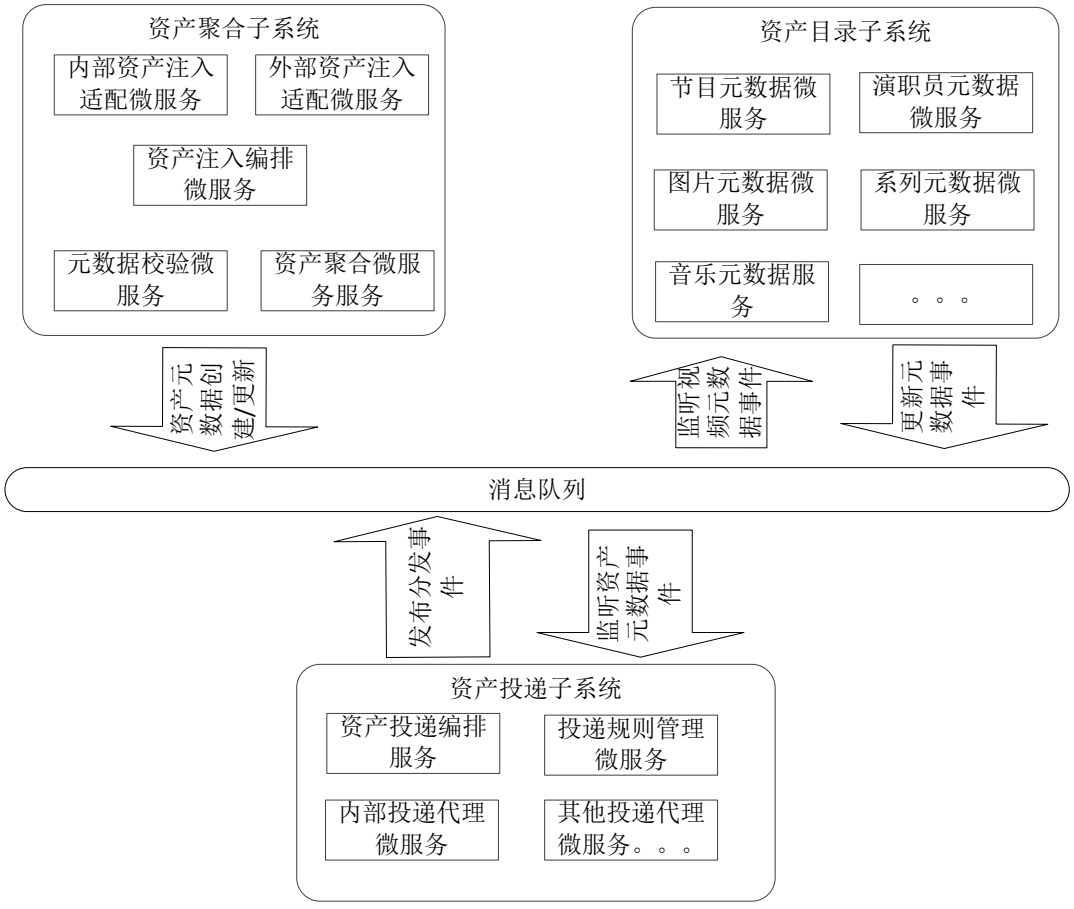

图 4-2 核心子系统交互图

从微观的角度来看，每一个模块子系统中都是一种服务编制模式，因为每一个模块子系统中都需要一个管理者的角色来协调其他子系统中的微服务以实现该业务子系统的业务流程。以分发模块为例，分发请求总管服务便充当了服务中的编制器角色，他需要协调控制何时触发资产的分发，并控制逻辑何时调用正确的分发代理，并控制分发的优先级。图 4-2 描述了三个核心子系统的交互方式。

4.2 资产编目子系统设计

媒资编目子系统是一个用来维护所有媒资分发系统最终资产的元数据记录的服务，除了提供数据的存储服务功能，编目服务需要提供灵活的查询接口以便其他服务可以更加通过灵活的方式获取到想要的资产元数据。

提供一个编目服务意味着该服务需要维护一套数据模型来表述各种各样的媒体资产元数据，然而一套统一的数据模型在今后时代公司的分发工作中是是至关重要的，目前的情况是资产元数据可能来自时代公司内部但也可能来自与其他的合作伙伴，他们之间可能会存在不一样的数据格式以及存在着不一样的数据项，不同的数据项同需要被记录因为在资产分发的环节，本文需要按照分发规范将这些数据项进行拼装和建模。

4.2.1 资产的建模与微服务划分

对于媒体资产元数据的存储需要前首先要做的事情是对媒体资产的元数据进行建模，对时代公司的媒体资产类型进行划分，图 4-3 是按照时代公司的业务对媒体资产元数据进行的建模（部分数据项已省略）。

根据 E-R 图所示，时代公司的媒体资产大致围绕着节目进行组织，因为时代的全球业务原因，需要对不同的国家和地区进行支持，一个节目就需要有不同的版本，同时由于节目需要在多种不同的播放平台进行播放，对视频的编码标准也有不一样的要求，因此会有不同的节目格式。

此外资产仍然包括一些文本信息，比如演职员表信息以及这些演职员的基本个人信息，节目的标题和大纲描述需要支持多国语言。媒资编目服务需要提供接口供其他服务获取这些资产信息，就需要一些灵活的查询和修改的接口。

微服务架构的设计中对微服务的粒度划分至关重要，过小的粒度会导致服务数量的膨胀，而过大的微服务粒度又不能充分发挥微服务架构的优势，本节将根据业务的需求和各个模块的用例图对资产编目子系统的各个业务模型从微服务粒度进行划分。

对资产编目子系统的微服务划分可以有两种方式:第一种是直接将所有的资产类型的元数据全部划分为一个微服务，第二种是根据业务模型来细分各个不同的微服务。这种划分方式优点是可以统一管理所有的资产元数据，在编程角度来讲可以节省重复的 DAO 层代码逻辑，但是从时代公司的业务角度出发来看，第二种划分方式更满足时代公司业务不断变动，且需要存储来源不同的合作伙伴数据，所以本节将根据 E-R 图建模结果进行对微服务做更细粒度的划分。

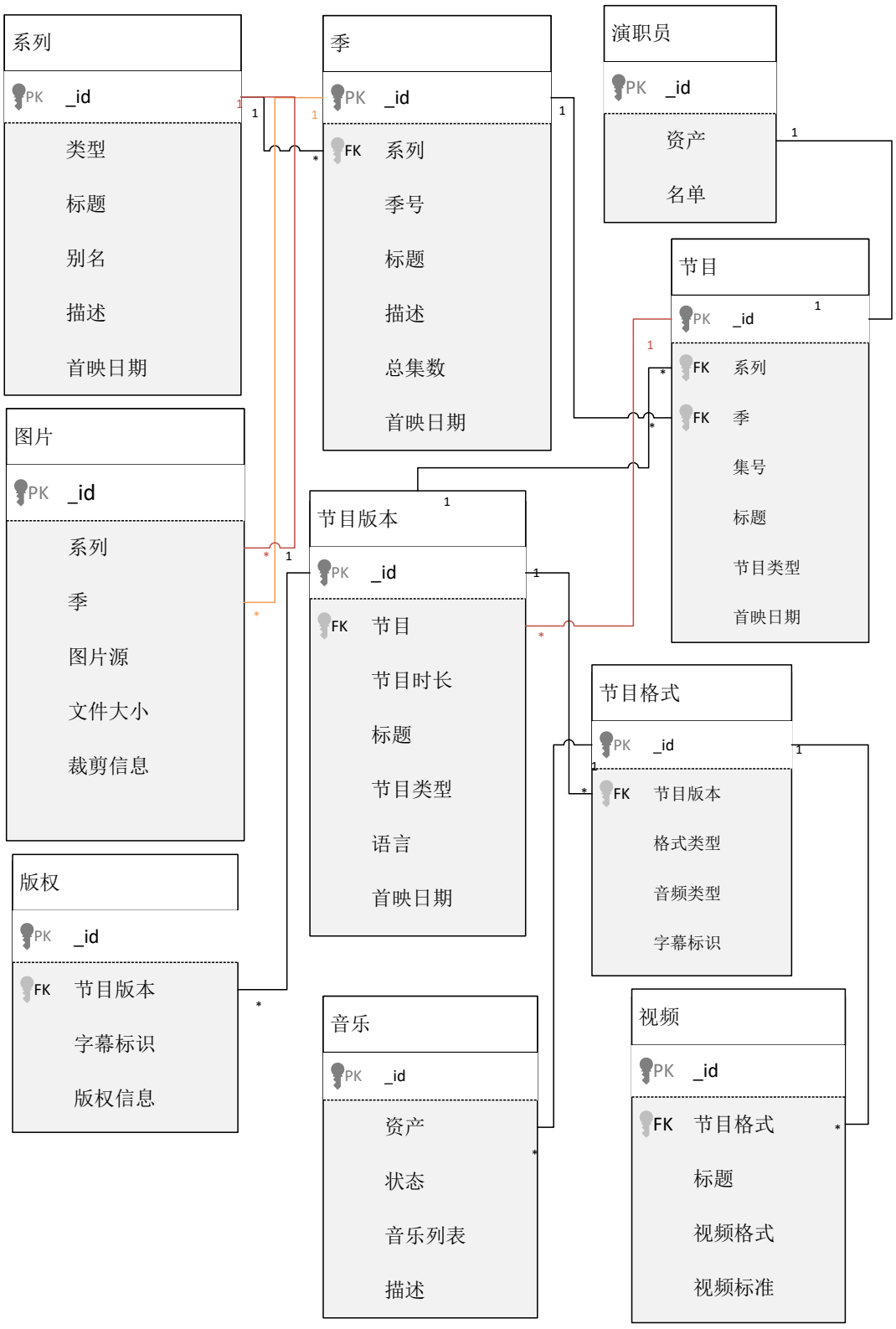

图 4-3 媒体资产元数据 E-R 图

根据图 4-3 元数据信息 E-R 图进行微服务的划分，将会对每一种模型单独进行管理，考虑到时代公司对模型的修改往往都是单独进行不需要关联的，因此可以为编目子系统得到以下微服务的划分：节目元数据微服务，节目版本元数据微服务，节目格式元数据微服务，系列元数据微服务，季元数据微服务，演职

员元数据微服务,图片元数据微服务，音乐元数据微服务，视频元数据微服务。

由于时代媒资分发系统存储的是成品资产，大部分资产在写入后绝大多数下都是只读的，且数据并非实时使用，意味着对媒资分发系统对数据的写入不需要强一致性，最终一致性完全可以达到系统需求，所以这也是可以对资产单独建立微服务的一个重要的原因。下一节以节目元数据为例，详细介绍元数据的数据项定义与数据存储。

4.2.2 节目元数据微服务数据存储

节目元数据模型用于描述节目的全局定义（也称为节目的本质）。一个节目虽然可能有许多版本和格式的标题，但其“本质”保持不变。但作为一个媒体资产分发平台，其主要职责围绕着需要分发给各个合作伙伴的一个节目的一些衍生格式，表 4-1 为节目元数据的各个数据项定义。

微服务架构带给媒资分发系统的优势之一便是更加灵活的技术选型，在资产编目服务只需要按照其资产元数据模型来选择合适的数据库技术而不需要考虑此数据库选型对其他微服务的影响，因为所有对资产元数据模型的操作都会被资产编目服务进行封装并经过资产编目服务暴露出来，所有直接对资产目录数据库进行直接的操作都是非法的请求，在网络层次就会拒绝。

节目元数据的数据由于在不同的命名空间可以是不同的，表 4-1 也是定义了一个半结构化的模型，使用 MongoDB 这样的文档型数据库进行存储就可以满足节目元数据对存取的需求,并可以在数据量大时提供分片[22]。

表 4-1 节目元数据数据项定义

数据项	数据类型	数据项描述	可选/必需
id	String	全局的节目资产元数据标识符	必需
系列	String	引用自全局的系列资产的元数据标识符	可选
季	String	引用自全局的季资产的元数据标识符	可选
节目类型	String	字符串表明当前节目的类型	必需
集号	Map	Map 类型数据指定在当前系列的集号和当前季的集号	可选
短标题	String	节目的短标题	可选
标题	String	节目标题	可选
标签	Array		可选

别名	Array		可选
描述	String	对节目内容的描述作为内部引用	可选
高清	Boolean	该节目是否为高清	必需
首映日期	Date	日期类型	可选
发布日期	Date	日期类型	可选
生产年份	Date	日期类型	可选
原产地	String	字符串描述节目的产地	可选
命名空间	String	命名空间用来区分不同来源的媒体节目资产	必需
命名空间标识	String	用来标示该命名空间内资产	必需
审计项	Map	审计信息，包括记录最后一位修改的人和时间	必需

4.2.3 节目元数据微服务的交互方式

在设计微服务的交互方式遵循的原则是避免微服务之间的交互技术来决定系统设计者可以使用什么技术栈来实现微服务，这样便可以实现微服务之间的技术不可知，从而有更大的自由性在技术选型上。

(1) 节目元数据微服务的交互方式选取

目前，常用的微服务交互方式有两种：同步方式和异步方式。

同步方式是指微服务的客户端发出请求后需要阻断其他任何活动的运行知道微服务端操作完成后给予回复。异步方式指的是微服务的客户端发出请求后不需要等待微服务端的操作答复，而是可以继续其他操作，客户端本身甚至可以不考虑微服务端方最终的结果。对于元数据而言所有的调用时间都不长，可以直接使用同步方式进行调用。

在一个标准的 HTTP 请求由两部分组成：方法和资源，对于每一个方法而言都有其特殊的意义和特性，表 4-2 展现了 REST 中常用的 HTTP 方法以及其方法意义和特性。在本文 2.1 对两种方法作出了比较，这里只强调 REST 对 HTTP 的支持，在节目元数据微服务中交互里，我们使用 REST 方法进行与其他微服务交互。

表 4-2REST 和 RPC 用到的 HTTP 方法

方法	意义	幂等	安全	可缓存

GET	获取一个资源	是	是	是
POST	创建一个资源或触发资源处理流程	否	否	仅当请求的响应中包含显式的刷新信息
PUT	全量更新一个资源或者是创建一个资源	是	否	否
PATCH	部分更新一个资源	否	否	仅当请求的响应中包含显式的刷新信息
DELETE	删除一个资源	是	否	否

(2) 节目元数据目录微服务交互接口描述

下面对节目编目服务提供的元数据的操作和相应的说明，在编目子系统中的其他资产元数据微服务也都有类似的操作接口来实现与各个元数据编目服务进行交互。

存储一个节目资产元数据信息:该接口是通过使用 REST POST 方法用来创建一个节目资产元数据记录,不需要提供任何的其他查询参数,请求的正文是任何符合 4.2.2 指定的数据项的 JSON 格式的节目信息,在没有异常发生的情况下,该接口会返回一个用来描述节目资产元数据的自动生成的全局标识符。API 中有两个参数一个是用来指出当前节目的版本号，另一个是描述当时节目的命名空间，调用的 HTTP API 为：/program/{version}/{namespace}。

查询给定的节目资产标识：该接口通过使用 REST GET 方法来在指定的命名空间中利用节目资产的标识符来查询给定节目的接口，该接口允许用户通过 projects 查询参数来指定返回节目资产的具体数据项，若不填写则默认以 JSON 数据格式返回一个节目资产的信息。API 中有三个路径参数，一个用来指定当前版本号另外一个用来指定命名空间和标识符，调用的 HTTP API 为：/program/{version}/{namespace}/{id}。

根据全局标识符查找节目资产：该接口通过 HTTP GET 方法来利用节目资产的全局标识符来获取唯一个的节目资产元数据信息，该接口也允许用户通过 projects 查询参数来指定返回节目资产的具体数据项，若可以正常查找到相应的节目资产，则以 JSON 数据格式返回符合查询要求的节目信息，否则返回 404 的

HTTP 状态码表明没有查到相应的资源。该 API 中只包含两个个路径参数，版本号以及用来指明节目资产的全局标识符，调用的 HTTP API 为：program/{version}/{id}。

根据发布日期查找节目资产：该接口通过 HTTP GET 方法来利用节目资产的发布日期来获取唯一个的节目资产元数据信息，该接口也允许用户通过 projects 查询参数来指定返回节目资产的具体数据项，若可以正常查找到相应的节目资产，则以 JSON 数据格式返回符合查询要求的节目信息，否则返回 404 的 HTTP 状态码表明没有查到相应的资源。该 API 中只包含两个个路径参数，版本号以及节目的发布日期，调用的 HTTP API 为：program/{version}/{publishDate}。

根据系列资产的全局资产标识和节目在此系列中的集号获取节目资产：该接口使用两个节目资产元数据中的属性来查询指定的节目资产，若找到以 JSON 数据格式返回符合查询要求的节目信息，若没有查到返回 404 HTTP 状态码，若是指定的参数有误返回 400 的 HTTP 状态码。该接口允许用户通过 projects 查询参数来指定返回节目资产的具体数据项，API 中的路径参数分别为版本号，命名空间，系列标识，资产类型以及该节目在此系列中的集号。调用的 HTTP API 为 :/program/{version}/{namespace}/getbyseries/{seriesurn}/{type}/{episodenoinseries}。

根据系列资产的全局资产标识和节目的首映日期来获取节目资产：该接口使用 HTTP GE 方法进行查找，由于该接口返回的节目信息可能是多个，若该 API 可以找到数据记录则以 JSON 的 Array 格式返回所有符合查询要求的节目信息，该接口允许用户通过 projects 查询参数来指定返回节目资产的具体数据项：/program/{version}/{namespace}/series/{seriesURN}/premieredate/{premieredate}
另外一个 API 在这个基础之上扩展成为了根据首映日期的区间来查找匹配项：/program/{version}/{namespace}/series/{seriesURN}/premieredate/{premiereDateStart}/{premiereDateEnd}。

除了上文描述查找接口外，节目元数据还应提供相应的修改接口。表 4-3 对这一类接口进行了描述：

表 4-3 节目元数据目录微服务修改信息的 REST API

方法	API	描述	请求正文
PUT	/program/{version}/{namespace}/{id}	用请求体中包含的节目资产元数据更新替换编目服务中已经存在的节目信息，这个操作时幂等操作	符合 4.2.1 指定的数据项的 JSON 格式的节目信息
PUT	/program/{versio	用请求体中包含的节目资	符合 4.2.1 指定的数

	n}/{id}	产元数据更新替换编目服务中已经存在的节目信息，这个操作时幂等操作	据项的 JSON 格式的节目信息
PATCH	/program/{version}/{namespace}/{id}	用请求体中包含的节目资产元数据补充更新编目服务中已经存在的节目信息	符合 4.2.1 指定的数据项的 JSON 格式的节目信息的一部分
PATCH	/program/{version}/{namespace}/{id}/tag	向节目资产元数据记录中添加一个标签信息，创建一个新的数组如果标签不存在	一个字符串类型的数据代表一个标签信息
DELETE	/program/{version}/{namespace}/{id}	根据标识删除一个节目资产元数据记录	无

(3) 接口的实现

通过使用框架来实现微服务的接口可以加快软件开发的速度，下面是利用 Spring 和 CXF 框架对 REST 接口进行定义，如下为 spring 配置文件节目微服务接口的代码片段：

```
<bean id="programService"
        class="com.catalog.program.impl.ProgramCatalogServiceImpl" >
        <property name="programDao" ref=" programDao"/>
        <property name="programQueryService" ref="programQueryService"/>
        <property name="programValidationService" ref="programValidationService"/>
        <property name="programSchemaService" ref="programSchemaService"/>
    </bean>

    <bean id="objectMapper" class="com.fasterxml.jackson.databind.ObjectMapper">
        <property name="serializationInclusion" value="NON_NULL"/>
    </bean>

    <jaxrs:server id="programServiceRS" address="/program/v1">
        <jaxrs:providers>
            <bean class="com.fasterxml.jackson.jaxrs.json.JacksonJsonProvider" />
```

```
        </jaxrs:providers>
        <jaxrs:serviceBeans>
            <ref bean="programService" />
        </jaxrs:serviceBeans>
        <jaxrs:extensionMappings>
            <entry key="json" value="application/json" />
            <entry key="xml" value="application/json" />
        </jaxrs:extensionMappings>
    </jaxrs:server>
```

利用 Java 代码编写具体的服务接口一节具体的实现类从而实现具体的操作接口，具体的 ProgramCatalogService.java 的实现代码片段如下：

```
        /**
         * Get program from the database with the projects client wants
         * @param id mongoid of the program
         * @param projections the projects client wants the api to return
         * @return the serialized version of the program with the properties that client
wants
         * @throws programCatalogServiceException
         *          ERROR CODE 404 Not Found: cannot find the schema or cannot find the
id in the database
         *          ERROR CODE 500 Internal Server Error: internal server error happened
during retrieving the program
         */
        @GET
        @Path("/api/{id}")
        public Program getProgramById(@PathParam("id") String id, @QueryParam("")
UpdateOption updateOption) throws programCatalogServiceException;
```

具体实现服务的实现类 ProgramCatalogServiceImpl.java 代码片段如下：

```
      ….
      public Program saveProgram(Program program, UpdateOption updateOption)
throws EspCatalogServiceException {
            // … 省略输入检查 ，获取更新选项
             List<String> crudOptions = generateOption(updateOption);
             Program existProgram = null;
            try {
```

```
                existProgram =
programDao.getProgramById(CURRENT_SCHEMA_VERSION, program.getId(),
PROJECTES_FOR_All);
                    if(existProgram == null){
                        programDao.saveProgram(CURRENT_SCHEMA_VERSION,
program, crudOptions);
                    } else {
                        if(!program.equals(existProgram)){

program.setAudit(CatalogServiceUtils.updateAuditModifiedTime(existProgram));
                                program.setId(null);

this.programDao.updateProgramById(CURRENT_SCHEMA_VERSION, id,program,
Arrays.asList("SET"));
                            }
                    }
            }catch (WebApplicationException e) {
    … // 省略异常处理}
```

4.3 资产聚合子系统设计

在资产分发系统中，为了提高分发请求的效率以及日后对分发的资产追踪，资产分发系统内部会维护最终媒体资产的元数据信息和最终资产，这样对于重复的分发请求和类似的分发请求（如仅仅是资产内容一样的不同终端的分发请求，或者是视频一样但是不同的字幕等）可以更加有效率的处理。因此，对于内部系统，若资产信息发生了变动，资产聚合子系统需要对变动的资产信息重新注入并更新更改掉的部分，资产聚合子系统按照职责上下文划分了以下五个微服务：内部资产注入适配微服务，外部资产注入适配微服务，元数据校验微服务，资产注入编排微微服务，资产聚合微服务。

内部资产注入适配微服务：主要负责通过监听消息队列中的数据修改事件来获取内部系统的交互信息，并将数据变更请求转化为相应的资产注入请求，并调用元数据校验微服务对请求中相应的资产元数据进行校验后传递给资产注入编排微服务。

外部资产注入适配微服务： 直接对外提供资产注入的 REST API，主要面向的是外部的合作伙伴，在调用此服务之前需要对合作伙伴的身份和权限进行验

证。在接受到新的资产元数据后，也需要调用元数据校验微服务对元数据进行验证，校验后便可以将请求传递给资产注入编排微服务从而创建一个资产注入请求。

元数据校验微服务：元数据校验微服务的功能如同它的名字一样，它仅仅负责对元数据的完整性进行检查，并通过调用配置服务对元数据的合法性进行验证。

资产注入编排微服务：资产注入编排微服务负责资产注入请求创建和管理，资产注入编排微服务在接收到资产元数据信息后，通过对资产元数据抽取去对关键信息提取并转换成内部数据模型格式，在调用资产聚合微服务决定去更新数据模型并维护着资产转入的状态。控制台服务可以通过调用资产注入编排微服务的 REST API 来展现或追踪一个资产注入请求的状态。

资产聚合微服务：资产聚合微服务维护着资产分发系统所有的资产模型，并维护着资产的命名空间以及各个模型的更新和创建的策略。资产聚合微服务提供获取和更改资产的 REST API，并当资产发生修改时发出模型更改事件到消息队列服务，关心资产变化的其他服务可以做出相应的响应

4.3.1 资产聚合微服务的集成与交互

本节从企业内部资产注入和外部合作伙伴资产注入的角度对资产聚合微服务的集成和交互作出说明，两个不同的业务流程在微服务架构中只需要在流程不同的地方添加新的服务来支持即可。

（1）内部资产的注入的微服务协作

对于内部资产的注入流程来讲，内部资产注入适配服务的功能相对比较简单一些，它负责创建一个资产注入请求，然后将这个请求以任务的形式扔到资产注入的消息队列里即可，资产注入编排微服务会去队列里面取得注入请求的任务，然后协调其他几个微服务完成媒体资产的注入过程，并将资产注入的状态维护在数据库中。在内部系统的注入过程中，注入环节只需要关注媒体资产的元数据，真正的媒体资产存储在企业内部的其他系统中，元数据中表明了他们资产的位置。

从图 4-6 中可以得出整个注入流程是以资产注入编排微服务为中心的一种服务编制模式。资产注入编排微服务通过借用消息队列来实现对注入任务的异步处理到，在这里介绍采用这种编制模式配合 AWS 提供的基础措施具体可以为资产元数据整合模块带来的最主要的三个优势：弹性的处理资产注入，为服务的故障恢复提供缓冲时间，顺序和优先级的保障[23]。

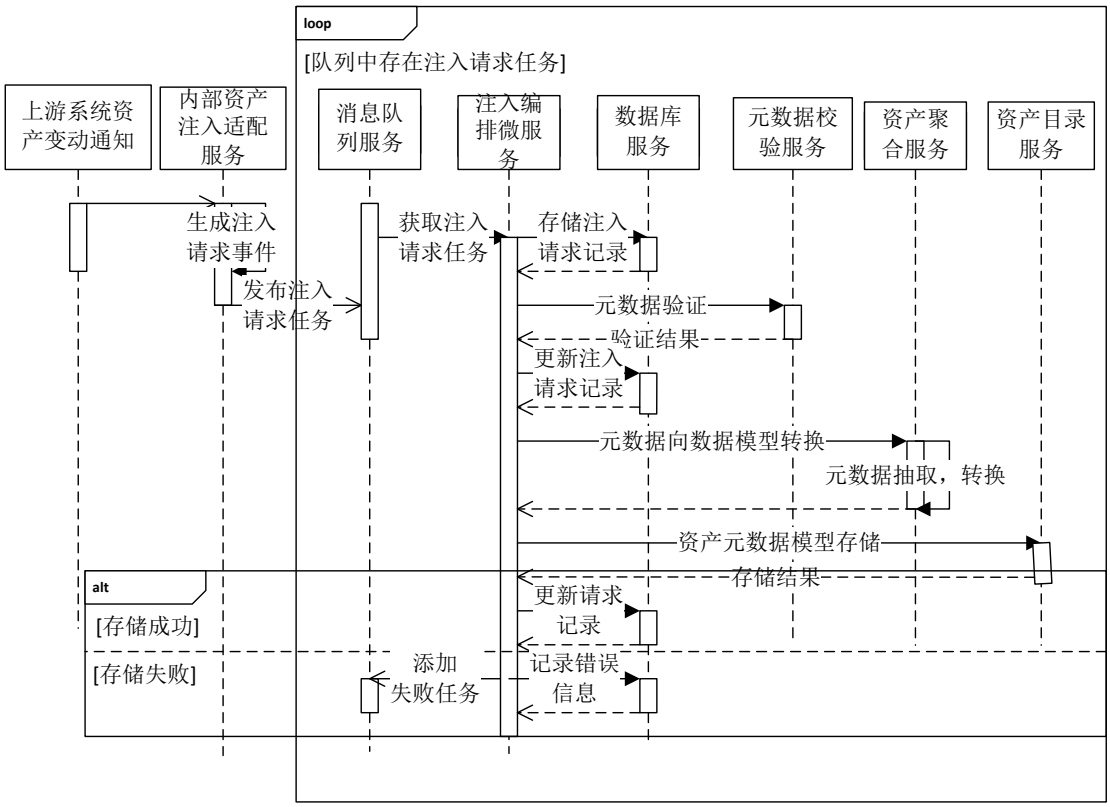

图 4-4 内部资产的注入的微服务协作时序图

弹性的处理资产注入，当请求任务当消息队列里面的任务量突然间比较高时，消息队列起到了缓冲的作用，这时只是资产注入编排微服务仍然可以按照自己的节奏消费任务，但是达到一定的数量处于一定的时间时，可以利用 AWS 的 Auto scaling 对资产注入编排微服务进行自动水平扩展，通过启动新的资产注入编排微服务实例来分担当前系统的负载，当消息队列里面的任务少于某一个特定值达到一段时间后，可以在相应的停掉当前一台资产注入编排实例。

为服务的故障恢复提供缓冲时间当资产注入编排微服务短暂性的出现故障时，在新的实例在生产环境被部署上去之前会有一个比较短暂的服务不可用时间，然而资产的注入消息队列可以将所有的资产注入请求暂时阻塞在消息队列中，当新的资产注入编排微服务可用时，可以继续提供服务。

为资产注入请求提够队列顺序和优先级的保障，对于默认情况下的资产注入消息，媒资注入系统应该是以队列的顺序，先到先服务，但是在媒资分发系统中经常出现的情况是有一些比较紧急的分发任务，他们所分发的资产临近了上线时间，这个时候就需要媒资分发系统提高这种注入请求的优先级，当大量的请求等候在队列中，资产注入编排微服务可以根据优先级从而优先消费高优先级的任务。

（2）合作伙伴资产注入的微服务协作

外部合作伙伴的资产注入流程与企业内部资产注入最大的区别在于，企业内部的资产信息维护在企业的内网系统中，而外部的合作伙伴需要将真正的资产传递给媒资分发系统上，从而进行进一步的处理。

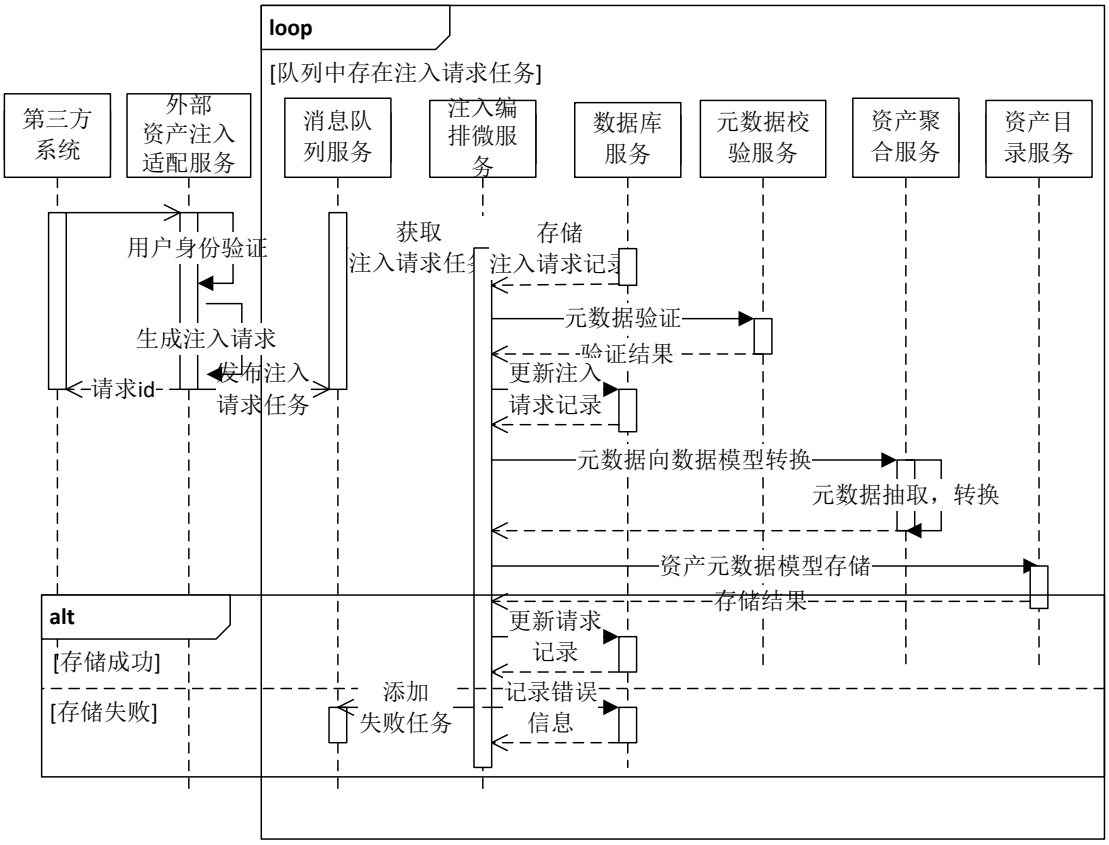

图 4-5 合作伙伴资产注入的微服务交互时序图

合作伙伴用户可以通过控制台页面来上传媒体资产，控制台支持在页面前台通过 multi-part 的方式将媒体资产上传到 AWS S3 上。第三方的应用程序可以直接通过调用 AWS S3 的 API 对媒体资产进行上传，同时也可以支持加密上传的方式来保护媒体资产在传输过程中的安全。

外部的合作伙伴资产元数据的注入流程与内部的十分相似，但是仍然有一些细节上的不同，比如需要在与第三方系统连接时多添加客户端身份认证的过程，同时因为第三方系统可能之后需要对系统注入的进度和状态查询，在外部注入适配服务中需要给资产注入请求生成一个请求 Id，这个过程可以通过调用 GUID 服务来实现（图 4-6 中省略了此步骤），这样第三方系统便可以根据此唯一的 ID 值对注入请求进行追踪。

4.3.2 资产注入编排微服务的协作管理

资产注入请求状态的一个完整生命周期是：新建，已开始，已验证，已转换，

已注入，完成。当一个资产注入请求被创建并加入到消息队列中，此时资产注入消息请求的状态为新建，代表着仅仅是创建了一个新的消息但并没有被任何消费者消费。

当资产注入编排微服务开始消费资产注入消息时，会将其状态更新为已开始，代表对此任务的处理已经开始。接下来，总管服务会调用元数据校验微服务对元数据进行校验，如果验证正确，则将此注入请求标识为已验证，代表收到的资产元数据是合法的，可以进行后续的处理。

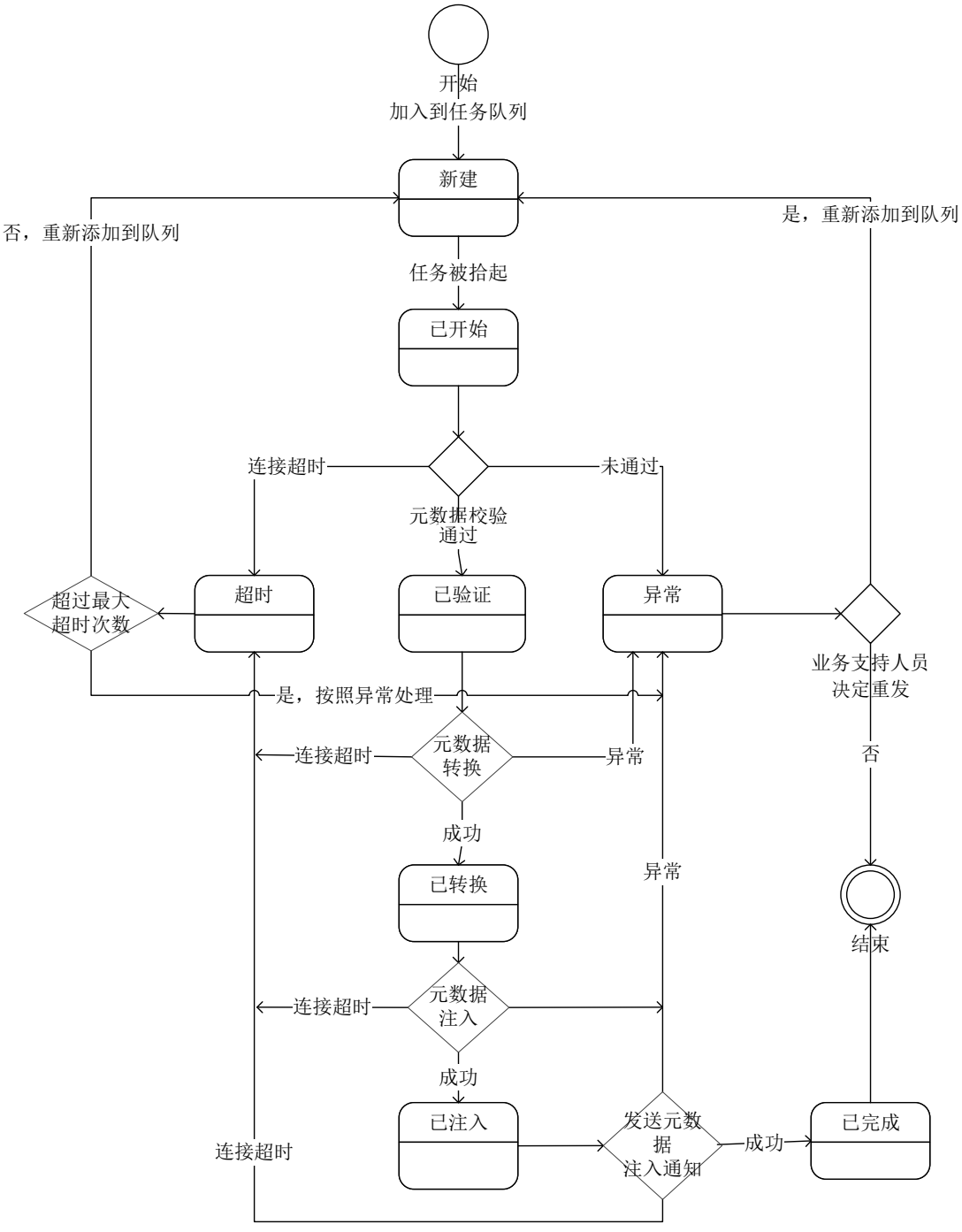

图 4-6 资产注入请求状态机图

资产注入编排微服务接下来会调用资产聚合微服务来对资产元数据向内部数据库模型进行转换，若没有在转换过程中发生任何异常，此时的状态将会被标识为已转换。已转换的资产注入请求意味着注入的元数据信息可以被存储到媒资分发系统的数据库中进行存储，接下来资产总管服务便会调用编目服务将数据模型进行存储，这个过程的成功会会将状态更新为已注入。

已注入的状态意味着整个资产注入的环节已经成功的完成，但是资产注入编排仍然需要发出一个注入事件，将注入的模型信息告知其他关心注入环节的服务，当所有更改的模型通知事件都发出去后，整个注入请求的状态会被更新为完成，图 4-7 阐述了各个状态之间的关系。

在整个生命周期的任意一个环节都有可能出现故障而导致请求进入异常状态，在异常状态下，资产注入请求需要标明出错的具体环节以及出错的具体信息并且会向失败任务的消息队列添加失败任务，一个失败的任务被加入到异常队列后会通过通知服务触发一条邮件通知信息发送到业务支持团队，业务支持团队可以通过控制台决定如何处置异常任务：忽略或者触发重发。

另外一种异常的注入请求是请求超时，因为资产注入编排需要同步调用一些服务从而完成资产的注入，而在同步的 HTTP 调用时容易产生连接超时的现象。对于超时的请求记录会被以最低的优先级添加回到消息处理队列，并在注入请求记录加入当前尝试次数，如果当前尝试此处已经超过系统设置的最大值，则会将超时请求上升为异常状态，然后将此任务作为失败任务添加到异常队列。

对于本文设计的微服务架构来讲，由于不同的微服务之间采用基于网络的 HTTP 协议且微服务之间的网络连接是很有可能出现故障的，所以这里需要特殊处理对于超时请求的处理，本文采用了一种超时自动尝试的机制来应对这种失败。下段 Spring 中对是超时重试的配置信息：

```
<bean id="retryTemplate" class="org.springframework.retry.support.RetryTemplate">
        <property name="retryPolicy" ref="retryPolicy" />
        <property name="backOffPolicy" ref="backOffPolicy" />
    </bean>

    <bean id="retryPolicy" class="org.springframework.retry.policy.SimpleRetryPolicy">
        <property name="maxAttempts" value="${retry.attempts}" />
    </bean>

    <bean id="backOffPolicy"
class="org.springframework.retry.backoff.ExponentialBackOffPolicy">
        <property name="initialInterval" value="5000" />      <!-- 5 seconds -->
```

```
        <property name="multiplier" value="2" />        <!-- double the wait each time -->
        <property name="maxInterval" value="30000" />    <!-- maximum of 30 seconds
-->
    </bean>
```

4.3.3 资产聚合微服务的数据抽取与转换

来自不同的合作伙伴的资产元数据可能拥有不同的数据格式，然而如何在这些形式不同的元数据中获取转换成上述数据模型是一个重要的挑战。在本文设计的媒资分发系统的设计中，数据聚合服务提供了数据的抽取和转换的功能从而实现将不同的媒体资产元数据存储到资产编目服务中。

(1) 模板引擎在数据抽取和转换上的应用：

目前资产元数据类型从数据格式上来区分大致分为两种：基于 XML 格式的资产元数据和基于 JSON 格式的资产元数据，当这些资产元数据发送到资产注入编排微服务后需要对元数据进行元数据转换，通过引入模板引擎技术，资产元数据的转换可以仅仅通过添加数据模板的方式，从而避免大量重复的业务逻辑代码。

资产聚合微服务基于 Java 语言进行编写，采用了 Apache FreeMarker 和 XLTS 两种模板技术分别来对资产元数据 JSON 格式和 XML 格式进行抽取和转换。下面分别是采用 Freemarker 模板进行元数据转换的模板样例：

```
<#if data["type"] == "EPISODE">
            "_guid::series::series": {
                "seriesid":"${data.series}",
                "lockId":"${data.series}"
            },
            "_guid::season::season": {
                "seasonid":"${data.season}",
                "lockId":"${data.season}"
            },
            <#if data["episodeNo"]??>"episodeNo":
                { "inSeries" : ${data.episodeNo["inSeries"]},
                  "inSeason" : ${data.episodeNo["inSeason"]}
                },
            </#if>
        </#if>
```

```
        "type": "${data["type"]}",
        <#if data["short-title"]??>
    "shortTitle": "${data["short-title"]?json_string}",
</#if>
        <#if data["title"]??>
        "title": "${data["title"]?json_string}",
        </#if>
    …
```

当所有的业务逻辑都由模板引擎来负责转换后，聚合服务中的逻辑代码就变得更加的通用化，主要负责处理引擎载入和模板的渲染，这样当业务转换规则发生任何变动时，不需对代码逻辑进行修改。通过这种方式可以更好的支持时代公司快速变动的业务逻辑，核心代码如下：

```
@Override
    public List<Canonical> process(Request request, List<TransformationRule>
transformationRules, String content) throws Exception {
        try {
            List<Canonical> canonicalList = new LinkedList<>();
            Map<String, Object> parameters = buildParameters(request);

            for (TransformationRule rule : transformationRules) {
                logger.debug("Applying transformation rule {} on request {}",
rule.getId(), request.getId());
                canonicalList.addAll(applyTransformationRule(request.getId(), content,
rule, new HashMap<>(parameters)));
            }
            return canonicalList;
        } catch (MaitredException e) {
            throw e;
        } catch (Exception e) {
            throw new Exception(e, "Failed to transform request " + request.getId());
        }
    }
```

2. 利用预定义的转换规则快速转换新的资产元数据

当使用模板对资产元数据进行转换时，需要考虑当使用模板来进行数据转换

时，何时使用新的模板以及何时应该重新组织和使用已有的模板。通过制定和业务规则相关联的资产元数据转换模板可以帮助资产聚合微服务来作出决定。下面是一个资产元数据的转换规则样例，资产聚合微服务采用了 JSON 的格式来表示这些规则。

```
{
    "contentType" : "Application/JSON",
    "messageType" : "CASTCREW",
    "service" : "serviceA",
    "sourceSystem" : "ServiceB",
    "templateEngine" : "Freemarker",
    "templateName" : [
        "/serivceA/serviceA _castcrew.ftl",
        "/serviceA /serivceA _people.ftl",
    ]
}
```

通过使用模板引擎和预定义的资产元数据转换规则，媒资分发系统可以快速的应对新的资产注入请求和已有的资产元数据变动，最终实现快速的业务上线时间已最大化的为公司争取更多的盈利和商业机会。

4.4 资产投递子系统设计

资产投递子系统的主要功能是根据用户定义的分发规则计算最小分发单元并相应的触发资产投递请求，资产投递请求按照资产分发规范调用相应资产投递代理来实现最终的资产和资产元数据的投递。按照资产分发的业务需求，将资产投递子系统划分为如下微服务：

投递规则管理微服务：投递规则管理服务提供 Rest API 来对媒体资产的分发规则进行添加和调整，同时该服务需要根据分发规则获取计算出最小的分发资产组成，并监听资产聚合模块发出的资产事件，当所需的分发的资产全部存在聚合服务内之后，便会触发分发请求，递交给分发请求管理服务。

投递请求编排微服务：投递请求编排微服务主要负责创建和维护资产的投递请求按照资产分发目的地的类型来决定使用哪种分发代理，并对外提供接口从而控制台服务可以获取到资产分发请求的进度，通过提供 Rest API 对资产分发请求的优先级进行维护，并提供 Rest API 对历史的资产分发请求进行查询。

内网投递代理微服务：内网投递代理微服务的主要职责是负责面向内网的企

业内部的平台，按照分发规范进行资产的投递并生成投递记录，投递请求编排微服务汇报的进度和状态。

REST 投递代理微服务：REST 投递代理微服务负责对指定 URL 为分发目标的投递请求按照分发规范进行资产的投递并生成投递记录，投递完成后向投递请求编排微服务汇报。

AWS S3 投递代理微服务：AWS S3 投递代理微服务面向的投递是亚马逊云平台上的 S3 云存储，投递代理按照分发规范进行资产的投递并生成投递记录，也需要向投递请求编排微服务汇报。

CDN 投递代理微服务：CDN 投递代理微服务主要负责定向将资产投递到企业的内容分发网络，需要按照分发规范进行投递并向投递请求编排微服务汇报。

对于媒资产分发系统而言，目前需要考虑分发的资产大概有四种：资产元数据，图片，视频/字幕，分发通知信息。不同资产的分发方式和资产的获取方式也有少许的不一样，取决与具体要分发到的下游服务，表 4-4 以时代公司的网站平台为例，比较说明了四种不同资产的分发。

表 4-4 时代网站平台资产分发表

资产实体	是否需要存储	资产获取自	资产分发至	方法
资产元数据	是	存储服务	指定 URL	REST 投递代理微服务
图片	否	存储服务	AWS S3	S3 投递代理微服务
视频/字幕	否	公司内网	AWS S3	内网投递代理微服务
分发通知信息	是	存储服务	指定 URL	REST 投递代理微服务

一般而言，不同的合作伙伴对资产分发方式的要求很可能都是不一样的，有时尽管是同是内部平台，可能需要对资产分发的方式也是不一样的，比如对时代公司的 OTT 平台而言，所有的分发途径都要求是 CDN 网络，则需要相应的 CDN 分发代理来实现分发。对于媒资分发子系统而言，最大的挑战是如何快速的接入新的合作伙伴平台。本节会从资产分发流程到分发记录的状态管理角度介绍基于微服务的分发子系统，并说明该设计如何快速接入新的合作伙伴的挑战。

4.4.1 资产投递代理的微服务协作

在投递子系统中，主要的业务流程是围绕着具体的投递代理，需要在不同的

场景下正确的调用合适的资产投递代理来处理投递请求。如何与投递代理更好的协作完成一个资产投递是本节的重点。

图 4-9 是资产投递过程的时序图，描述了投递相关各个微服务之间的集成与交互。投递规则管理服务会按照业务用户创建的分发规则并监听各个来自编目服务的资产变更事件，当资产目录的微服务中具备资产分发规则中所要求的资产时，投递规则管理服务需要创建一个资产分发请求，然后将这个请求以任务的形式扔到资产分发的消息队列里。

分发请求管理服务会去队列里面取得资产分发任务，然后协调其他几个分发代理微服务完成媒体资产的分发过程，并将资产分发记录的状态信息维护在数据库中。对于资产元数据信息和资产分发通知信息来讲，他们共同的特点是他们均为文本信息，可以被存储到存储服务内作为后续检查和使用。

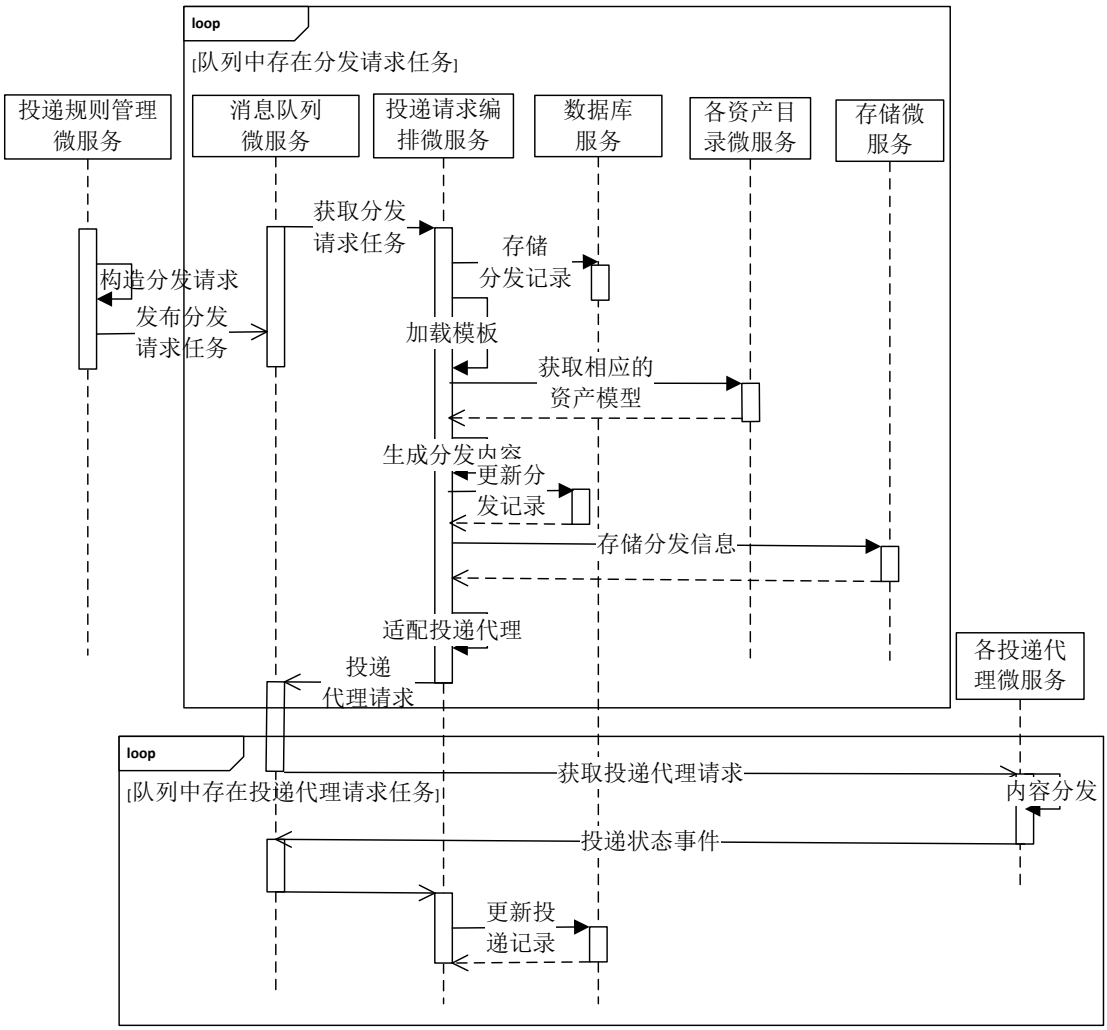

图 4-7 资产投递微服务的集成与交互

待存储好相应的信息后，投递请求编排微服务便可以匹配适合投递请求来最终处理资产的投递，并将投递的代理请求通过消息队列微服务上传到消息队列中等待投递代理的进一步处理。各投递代理微服务在队列中取出各自的任务进行分

发并将最终的投递结果返回到消息队列中。

媒体资产的分发流程是以分发请求管理服务为中心的一种服务编制模式。分发请求管理服务通过借用消息队列来实现对分发任务的异步处理，同时各个分发代理仍然使用了消息队列的方式来向分发请求管理服务汇报分发结果。

采用这样的服务协作模式可以弹性的处理资产大量的资产分发请求任务，当消息队列里面的任务量突然间比较高时，消息队列起到了缓冲的作用，这一点和资产聚合子系统中有着异曲同工的作用。

资产分发管理服务可以按照逐个消费任务，当队列里堆积的任务达到一定的时间时，可以利用 AWS 的 Auto scaling 对资产分发服务进行自动水平扩展，通过启动新的资产分发实例来分担当前系统的负载，当消息队列里面的任务少于某一个特定值达到一段时间后，可以在相应的停掉当前一台资产分发服务实例。

这种协作模式可以为服务的故障恢复提供缓冲时间，当投递请求编排微服务短暂性的出现故障时，在新的实例在生产环境被部署上去之前会有一个比较短暂的服务不可用时间，然而资产分发请求消息队列可以将所有的分发任务暂时阻塞在消息队列中，当投递请求编排微服务可用时，可以继续提供服务。

目前需要支持的分发代理有，AWS S3 分发代理服务，内网分发代理服务，REST 分发代理服务等，如果业务需要，可以快速增加新的的分发代理。OCP（Open Close Principle）原则是面向对象编程中的一个原则，指的是软件应该保持对修改关闭但是对添加开放，代理微服务的设计符合这个原则，具体体现为，每当业务需求导致一种新的分发方式出现时，全新的业务规则需要添加到配置服务中，此时业务分发规则服务可能需要做一些模板的修改来应对新的服务，而对分发请求管理服务而言，不需要代码层面的改动,创建一种新的分发代理来处理相应的分发逻辑，而其他流程仍应保持不变。

媒资分发系统与分发系统类似，以队列的顺序来服务所有的资产分发请求，先到先服务，对于比较紧急的分发任务，例如他们所分发的资产临近了上线时间，这个时候就需要提高分发请求的优先级从而保证资产分发服务消费高优先级的任务。

4.4.2 资产分发请求记录状态管理

资产分发请求管理服务另外一个重要的功能就是维护资产分发请求，每一个资产分发请求，资产分发请求管理服务都会相应的生成一条分发请求记录。一条资产元数据或者分发通知数据的分发请求记录完整的生命周期为：新建，已开始，已生成，已存储，分发中，已分发，对于图片资产和视频字幕资产的分发记录完整的生命周期中没有生成和存储，即：新建，已开始，分发中，已分发。几

个异常状态为重复，不可用和失败。

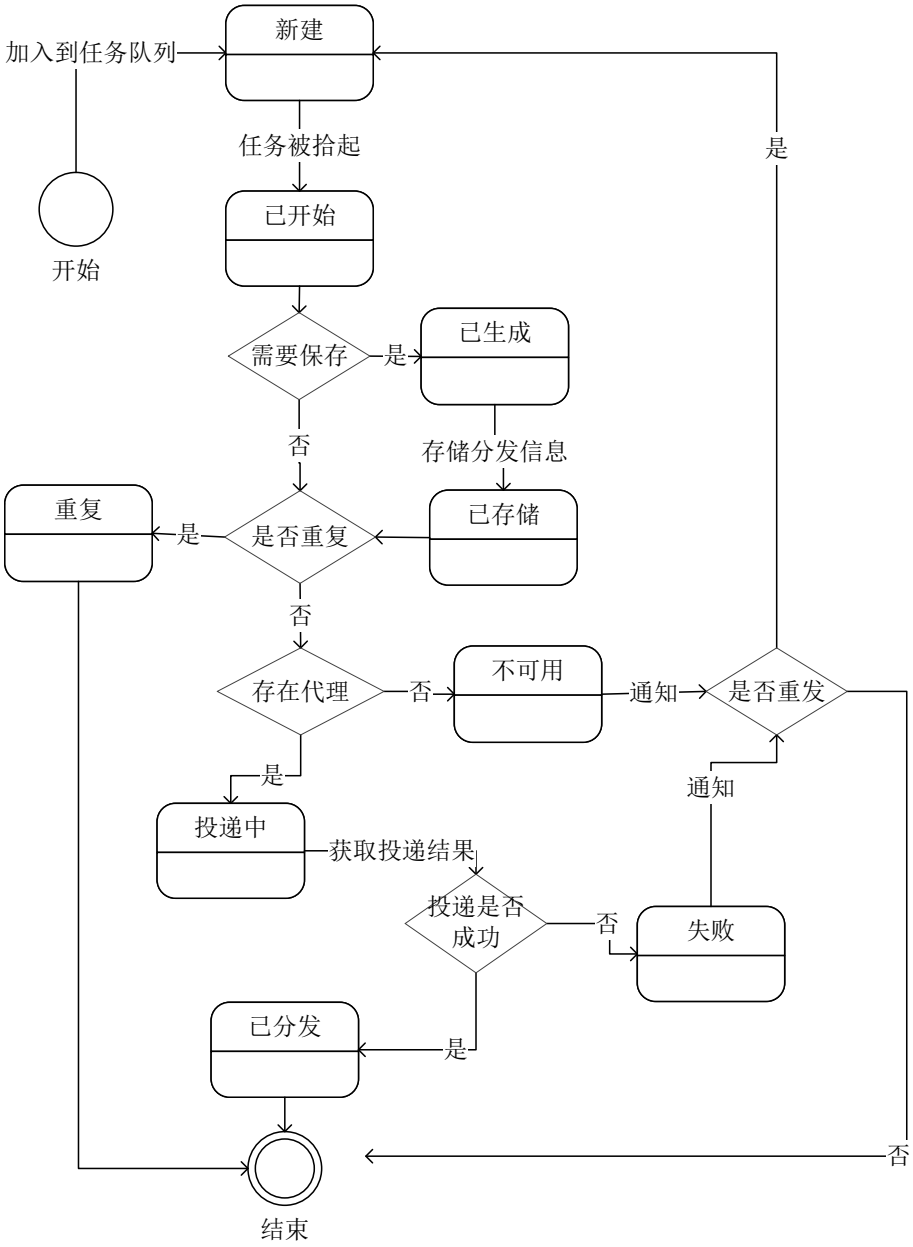

图 4-8 投递请求状态机图

重复指的是不是带有重发标识却是已经分发过资产分发信息，这一类会被显式的标识为重复，并停止资产分发流程的后续操作。不可用状态特指根据当前的配置信息无法匹配到合适分发代理，则将该资产分发请求标识为不可用。当资产分发代理因各种原因导致的分发失败都将会导致这条分发请求的最终失败，另外整个流程中发生的异常时，分发记录同样也会成为失败状态，图 4-10 描述了各个状态之间关系和转移（状态中的异常捕获已省略）。

在资产分发记录状态的整个生命周期中，在任意一个环节都有可能出现故障而导致分发请求进入失败状态，在失败状态下，资产分发请求同样也会标明发生

故障或出现异常的具体环节以及具体的详细信息，同时一条任务失败的消息事件将会被加入到消息队列中，进而通过通知服务触发一条邮件的通知信息或短信息发送到业务支持团队，业务支持团队可以通过控制台决定如何处置失败任务：忽略或者触发重发。忽略会清除消息队列中的失败任务但数据库中失败记录仍然会被保存，重发会使得当前的资产分发请求经历整个生命周期。

4.5 与同类媒资分发系统的比较

时代媒资分发系统基于微服务架构，在支持媒资产分发系统的核心业务基础之作出架构上的创新。时代媒资分发系统从设计和实现上与同类系统比较后主要具有以下优势。

（1）高内聚，低耦合

微服务的架构对各个微服务的细粒度划分借鉴了面向对象的 SRP（Single Responsibility Principle）单一职责原则，而这一原则使得每一个微服务的复杂度降低并且大幅度的提高了功能的内聚性，同时职责的单一使得各个服务之间的耦合性减低，时代媒资分发系统的各个微服务之间通过不同组织方式协同工作，但每一个微服务又保持功能内聚的特性，这一点是其他同类系统很难做到的。高内聚和低耦合的特性可以为系统整体带来鲁棒性，从而大幅度提高软件质量和降低软件的维护成本。

（2）高可用性

时代媒资产分发系统通过采用服务编排和引入消息队列等方式来提高微服务的可用性。当服务出现不可用现象时，时代媒资产分发系统的架构从设计上可以最大程度的降低不可用的影响并对错误进行缓冲时间，从而确保业务的正常运行。大部分同类系统因为没有相应的防范机制，当出现某一个功能组件的不可用时会升级为系统不可用，从而造成系统瘫痪甚至影响到业务的正常进行。

（5）灵活的扩展性

时代媒资产分发系统可以根据请求数量对相应的微服务进行微服务级别的水平扩展，从而保障更高质量和的完成分发任务，并且可以利用队列来缓冲短时间的大量请求，当请求数量减少后可以相应的减少微服务的实例数量从而减少运营成本。相比与时代媒资分发系统的微服务级别扩展，同类系统的扩展方式为系统整体扩展或通过提高服务器物理能力的垂直扩展，其成本开销比较大且扩展效果不明显。

4.6 时代媒资分发系统应用效果

通过采用微服务架构，不仅仅时代媒资分发系统的系统性能得到提升，软件的质量也随着组织结构的变化而变得更加优秀。凭借着微服务架构的各方面优势，时代媒资分发系统应用后取得了如下效果：

（1） 更优秀的服务质量

媒体资产分发系统借助微服务架构的性能优势和质量保证，自上线以来用户订阅数量增长了十万人，全球总订阅人数超过七千万。依靠微服务架构提供的高质量保证，自上线以来，未出现一次因系统不可用而收到的用户投诉或因资产分发未按制定时间标准而受到业务伙伴投诉，而在上一代的时代资产分发系统中，平均一个月要收到约十条来自业务伙伴关于资产未按时抵达的投诉。

（2） 更广泛的业务涉猎

更多的订阅用户和合作伙伴给予了时代公司更多的机会去尝试新的业务领域，例如直播，互动性视频等。相比与上一代的时代媒资分发系统，因为系统没有能力去良好扩展业务使得计划被一直被搁置，高效稳定的媒体资产分发平台通过微服务架构为业务的拓展做好了充分的准备，其通过快速创建新的微服务来应对新的业务类型从而更小的影响现有的业务流程。在系统上线一个月后，时代公司便在媒资分发系统的支持下，开展了直播视频业务。

（3） 更快的市场上线时间

新的业务服务上线时间平均从一个半月减少到三天，得到媒体行业普遍认可，并且为时代公司创造了许多商机。媒体资产分发平台作为时代公司构建一个数字化媒体网络供应链平台的重要前提，其成功上线并快速大规模的向合作伙伴分发媒体资产获得了来自媒体行业对时代公司数字化媒体资产供应平台的认可和更多业务合作往来意愿，自上线以来，合作伙伴数量增长大约 20 个百分点。

第五章 结 论

本文在时代公司的媒体资产分发和管理业务的日益增长的前提下，对处理大量的不同格式的元数据格式与形媒体资产进行转码编码以及分发到各个平台的迫切需求提出了基于微服务架构的分发系统解决方案，相比与其他架构，使得系统在扩展性，可用性，重用性以及系统吞吐量层面上可以更好的应对不同元数据格式的媒体资产分发请求，为时代公司解决了技术无法支持业务的局面，并帮助公司的媒体资产开拓了了更多的市场，提高了整体业务运行水平。

5.1 时代媒资分发系统的特点

基于微服务架构的媒资分发系统在设计上秉承着微服务架构的核心理念，对不同的业务领域内模块进行了微服务拆分，从而降低了每一个核心业务功能的维护成本，提供了更灵活的媒体资产类型配置的资产分发管理。相比与其他同类系统，时代媒资分发系统具有以下特点。

（1） 保证资产分发的可靠性

时代媒资分发系统的设计中，基于微服务的编排以及采用事件的响应模式可以更好的应对故障发生。当任务实例恢复可用时，时代的媒资系统可以继续进行，其他服务仍然可以正常提供服务而不受到干扰。

从可用性出发，时代媒资分发系统可以帮助时代公司有底气和能力去应对更大和更多的资产业务，同时保证业务期间不会出现丢失分发数据的情况，大大提高了公司在行业内的口碑。基于微服务架构的架构优势可以更好的给媒资系统提供容错功能，细粒度的微服务隔离了从根本上隔离了错误，解决了过去一块逻辑错了整个应用程序都不可以使用的问题，从而提高了资产分发的可靠性。

（2） 支持多源异构的成品媒体资产

时代媒资分发系统支持通过隔离出一个资产目录的微服务，并根据多源异构的特点采用了 MongoDB 对数据进行储存和维护，摒弃了传统的关系型数据库的强制预先定义数据格式的策略，而应用了半结构化的预定义数据格式，从而更好的支持多源异构的资产元数据的存储。

通过采用微服务架构的方式来集成资产目录微服务，时代公司的业务人员可以频繁的更改或者添加不同来源的媒体资产的元数据格式，资产目录微服务通过升级 API 的版本号的方式来支持新的元数据类型，从而在最小的程度上对其他的微服务造成影响。

（3） 灵活的资产分发规则管理

时代媒资分发系统通过引入微服务架构对不断变换的媒体资产分发规则进行单独管理,从而更加贴切的迎合了分发规则频繁更改的业务场景。解决了其他同类系统分发规则的更改牵连整体系统的更改状况，同时缩短了软件测试的时间，加快了新的媒体资产推进市场的时间，从而帮助企业抢得市场先机。

5.2 不足与展望

文本从架构方向作为切入点，讨论了基于微服务架构的媒资分发系统的设计与实现，由于在项目初期公司人员对微服务架构的实践经验不充足，从设计到实施还是有考虑不周全的地方，需要成为今后开发和改进工作的重点。下一步的工作可以分为：

（1）本文设计的微服务架构从宏观的角度是基于服务编排，因此业务整体的流程没有一个管理者统筹管理，而是利用监听者的模式收集各个服务的状态，而大量收集各个服务的状态仍然是一件开销比较大任务，今后的工作方向可以改善对业务流程监控的性能。

（2）云计算时代的到来的使得越来越多的企业将自己的基础设施和基础服务搬到云上从而享受云计算带来的更低成本的基础设施解决方案。采用基于微服务架构的应用系统是一个很适合利用云平台的策略，在各个服务级别的开发和维护，以及采用云平台既有的 SaaS 服务都可以很大程度的消减构建媒资分发系统应用的开销。虽然本文设计的媒资分发系统已经采用了部分的云计算服务，但是随着云计算平台功能的不断完善，越来越多的服务可以托管给云，从而减少不要的开发和维护。

参考文献

[1] Austerberry D. Digital Asset Management, Second Edition[M]. Focal Press, 2006.

[2] Krogh, Peter. The DAM Book[M], Second Edition. United States of America: O'Reilly Media, 2009.

[3] Yongquan Lu; Chu Qiu; Pengdong Gao; Jintao Wang; Rui Lv.Research on Application of Object Technologies in Media Assets Management: High Performance Comput[C].Beijing:Management and Service Science, 2009(9)

[4] 宋培义. 电视媒体数字资产管理模式研究[D].北京交通大学, 2007

[5] 张增琦. 媒体资产管理在中国[J]. 视听界,2006,(04):29-30.

[6] 张洋. 电视台全台网建设中媒体资产管理系统应用的研究[D].北京邮电大学, 2009.

[7] 周轶. 基于 REST 和工作流的数字媒体资产管理系统构建[D].复旦大学,2010.

[8] 倪燕燕. 媒体资产管理系统的设计与实现[D].华东师范大学, 2010.

[9] Atasso, Cesare. Microservices in Practice, Part 1: Reality Check and Service Design.IEEE Software[J]. 2017.34 (1): 91 - 98.

[10] Irakli Nadareishvili, Ronnie Mitra, Matt McLarty, Mike Amundsen. Microservice Architecture [M]. United States of America：O'Reilly，2016：105-118.

[11] S. Newman, Building Microservices: Designing Fine-Grained Systems, O'Reilly, 2015.

[12] Zimmermann O. Microservices tenets[J]. Computer Science - Research and Development, 2016:1-10.

[13] Evans. Domain-Driven Design: Tacking Complexity In the Heart of Software[M]. Addison-Wesley, 2004.

[14] David Gonzalez. Developing Microservices with Node.js [M].UK：Packt Publishing Ltd.，2016：59-69.

[15] Dmitry Namiot, Manfred Sneps-Sneppe. On Micro-services Architecture[J]. International Journal of Open Information Technologies，2014，2(9):24-27.

[16] Nicolai M.Josuttis. SOA in Practice[M]. United States of America: o'Reilly Media, Inc, 2007, 32-34

[17] 王福强. SpringBoot 揭秘:快速构建微服务体系[M]. 北京：机械工业出版社, 2016.

[18] Fielding R T. Architectural styles and the design of network-based software architectures[C]// University of California, Irvine, 2000:303

[19] J. Webber , S. Parastatidis , and I. Robinson , REST in Practice: Hypermedia and Systems Architecture , United States of America: O'Reilly , 2010

[20] 王磊. 微服务架构与实践[M]. 北京：电子工业出版社, 2016.

[21] 黄勇. 轻量级微服务架构（上册）[M]. 北京：电子工业出版社, 2016.

[22] Kristina Chodorow，Michael Dirolf.MongoDB 权威指南[M].北京：人民邮电出版社，2011：12-20

[23] Amazon Web Services, Architecting for the Cloud: AWS Best Practices[EB/OL/].https://d0.awsstatic.com/whitepapers/AWS_Cloud_Best_Practices.pdf, 2016

致 谢

转眼间复旦大学两年的学习过程就要过去了，首先要感谢 XXX 导师孜孜不倦的教诲，尤其记忆深刻的是论文的开题和初稿期间，导师不厌烦的屡次提出修改建议，并时刻叮嘱我注意论文的进度，指点我学习和改进的方法。其次要感谢复旦大学所有授课老师以及给予我帮助的各位同学。

感谢父母和妻子平日里默默的支持以及对工作之余仍忙于论文的理解，感谢同事和领导的帮助。最后，感谢复旦大学提供给我这次学习的机会，定当铭记和践行学校校训：博学而笃志,切问而近思。

论文独创性声明

本论文是我个人在导师指导下进行的研究工作及取得的研究成果。论文中除了特别加以标注和致谢的地方外，不包含其他人或其它机构已经发表或撰写过的研究成果。其他同志对本研究的启发和所做的贡献均已在论文中作了明确的声明并表示了谢意。

作者签名：_____ 日期：_____

论文使用授权声明

本人完全了解复旦大学有关保留、使用学位论文的规定，即：学校有权保留送交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其它复制手段保存论文。保密的论文在解密后遵守此规定。

作者签名：_____ 导师签名：_____ 日期：_____