

摘要【300-320，不要超过330】：

2023年11月份，我所在公司承担了某银行“智能终端管理系统”的开发工作，我作为该项目的系统架构设计师，负责该系统的需求分析与架构设计工作。该系统**主要提升银行对智能收银终端软硬件的管理及维护能力**，业务模块包括应用APP管理、设备管理、远程协助以及屏媒广告等多个模块。本文以该平台为例，论述云原生架构在该项目中的具体应用。首先，遵循服务化的设计原则，将平台拆分为多个小的、自治的服务，每个服务独立运行；其次，利用容器的弹性特性，平台能够自动适应负载变化，解决高并发问题；最后，利用云原生架构的可观测性和自动化，引入了自动化测试和代码审查来确保代码的质量。整个项目历时11个月开发完成并上线，到目前运行稳定。实践证明，云原生架构的使用极大提升了开发效率，减少了开发成本，也为后续的持续优化奠定了基础。

正文【2200-2500】：

随着互联网技术的飞速发展，传统的收银设备操作系统简单，功能单一，无法远程升级（OTA），设备的状态也无从得知，所以智能设备替换传统收银设备是一个趋势。为了增加商户的粘性，提升服务质量，各大银行开始加大智能收银终端的投放。不同于传统的收银设备，智能设备联网能力强，操作系统强大，能够实现更多更复杂的业务。为了充分发挥智能设备的优势，**并进一步提高银行对设备的监管**，行方提出建设一个智能终端管理系统的需求，实时监控设备状态，并对设备的软硬件生命周期进行有效的管理。

2023年11月份，我司承担了某银行的智能终端管理系统的开发工作，我作为该项目的系统架构设计师，参与整个开发过程。该系统的主要业务模块包括应用APP管理、设备管理、远程协助以及屏媒广告等。建设该**系统旨在加大银行对设备的管理，提升对商户设备的监管能力，以及对设备的维护效率**。应用APP管理模块，主要包括应用上架，应用管理等功能。银行或商户可定制专属应用市场，上传和发布自己的APP。银行作为管理方，还可以针对不同业态的商户配置应用的可见性。设备管理模块，包括设备总览、定制设备以及远程更新功能。银行通过平台的可视化界面，能够清晰、实时的关注到设备当前的软件使用情况和硬件状态。另外，银行可通过平台配置装机初始化任务，设备首次激活时，会联网获取装机任务，包括定制开机动画、应用预装以及系统配置等。远程协助模块，主要包括文件传送接送，录屏以及远程控制等功能。针对软件常见的BUG问题，在商家授权后，通过该平台的远程协助功能，可远程操控设备，获取设备日志，便于软件开发人员排查问题。屏媒广告模块，主要包括媒体资源管理、广告数据统计等功能。银行或商户可通过平台上传广告资源，如优惠活动宣传等，并分析广告播放的数据统计。

云原生架构近年来渐渐成为各行各业数字化转型的新宠儿，云原生技术作为云计算服务的新模式，逐渐成为企业持续发展的主流选择。它基于一系列架构原则和设计模式，旨在将云应用中的非业务代码部分进行最大化剥离，让云设施接管应用中原有的大量非功能特性，如服务化、弹性、可观测性以及自动化原则等。首先，服务化原则是云原生架构中的核心设计原则之一，它强调将复杂的应用程序拆分为一组小型、独立且可复用的服务，每个服务都专注于执行单一的业务功能，并通过定义良好的接口与其他服务进行通信。同时服务化架构以面向接口编程，服务内部的功能高度内聚，模块间通过公共功能模块的提取增加软件的复用程度；其次，弹性原则是云原生架构中另一非常重要的设计原则，它要求应用程序具备根据负载变化自动调整资源使用的能力，**以有效应对突发的流量高峰或系统故障**。传统系统依赖于上线前对业务量进行估算，准备相应的软硬件资源，这一步骤往往周期较长。而弹性完美解决了该问题，它可以随着业务量的变化而自动伸缩，无需事先规划软硬件资源。再次，可观测性原则强调应用程序应设计并部署足够的监控和日志数据收集机制，以确保开发团队能够实时、全面地了解系统的运行状态和性能表现。例如，程序在处理高并发的用户访问时，系统必须能够提供详尽的日志数据，如错误消息、调试信息等。最后，自动化原则是指通过持续集成（CI）、持续部署（CD）、自动化监控和告警等技术手段来实现对应用程序的自动化管理。自动化运维减少了人工干预，缩短了开发和部署的周期。

结合行业特性和用户需求，我们最终确定了采用云原生架构作为系统设计的核心理念。我作为架构师，紧密围绕服务化、弹性、可观测性以及自动化核心原则，成功构建了灵活且高度可扩展的管理系统。下面我将详细阐述云原生架构在该系统的应用过程。

首先，在服务化设计方面，我们摒弃了传统的单体应用架构，将业务功能拆分为多个独立的服务，包括用户管理服务、应用管理服务、设备管理服务、远程协助服务、屏媒广告管理服务等等。这样，当需要对某个功能进行升级维护时，可以仅对该功能对应的服务进行操作，而不会影响到其他功能的正常运行。同时，这种拆分还带来了配置和部署的灵活性。以本系统的设备管理服务为例，该模块需要频繁的对数据进行解析和转换，因此会占用更多的CPU资源，因此我们特意为该模块分配更多的CPU资源，以保证服务的高效运行。另外，服务间的通讯依赖于服务网格技术，它是一个透明且统一的服务间通信管理层，具备服务发现，负债均衡以及流量管理等功能。服务网格的引入极大的简化了微服务间的交互流程。

其次，在弹性设计方面，考虑到智能终端管理平台必须能够应对**高并发访问和突发流量的情况**，我们充分利用了容器化技术和Kubernetes平台，容器化确保了服务的一致性和可移植性，而Kubernetes则为我们提供了强大的服务自动伸缩和资源动态分配能力。面对某些服务在突发流量下的性能瓶颈，我们深入进行了性能优化工作，并引入了更细粒度的资源调度策略，如基于机器学习的资源预测和自动扩容算法，以确保系统在高负载下依然能够保持稳定的运行状态。相反，当服务资源占用缩减到我们设置的阈值时，系统会自动减少资源数量，避免资源的浪费，使得我们系统能够从容的应对各种流量变化。

最后，在可观测方面，我们对日志进行等级划分并评审，通过日志记录了每个请求之间的参数传递，接口耗时等。在自动化设计方面，为了提高开发效率和代码质量，我们实施了持续集成/持续部署（CI/CD）流程，面对初期阶段由于流程配置复杂导致的部署失败率较高的问题，我们对CI/CD流程进行了全面的优化，引入了更多的自动化测试和代码审查环节来确保代码的质量；同时利用容器化和镜像技术来简化部署过程；还采用了蓝绿部署和金丝雀发布等策略来降低部署风险并提高系统的可用性。通过这些改进措施的实施，我们成功地降低了部署失败率并提高了代码的稳定性。

整个项目历时11个月开发完成，并于2024年10月顺利上线，到目前运行稳定，市场反馈良好。尽管该架构带来了许多优势，但实际应用中仍然存在一些不足之处。比如在设备使用高峰期，设备服务流量激增，导致访问性能降低，针对该情况，我们深入进行了性能优化工作，设计更细粒度的资源调度策略。该策略会根据CPU使用率、内存使用情况以及请求量来进行自动扩容操作。当负载减少时，系统又会自动回收多余的实例以节省资源。

实践证明，通过基于云原生设计的智能终端管理平台，不仅降低了开发成本，提高了团队开发效率，同时，系统在大规模设备管理中展现出了良好的稳定性。通过各服务间的分工和协作，实现了系统的高效、灵活和安全，有效提升了银行智能终端管理效率，减少了设备维护成本，从项目完工后的使用效果来看，达到了预期目的，也获得了行方的一致好评。

最后，经过这次参与云原生架构的设计实施后，我也看到了自己身上的不足之处，在未来还会不断地更新知识，完善本系统在各方面的设计，我们相信能够更好地满足不断变化的业务需求，并提升系统的整体性能和稳定性。在为公司创造价值的同时，也能够为客户开发出更稳定可靠的系统。