云函数

客户案例

腾讯云

文档目录

客户案例
电子商务行业案例

最近更新时间：2022-04-22 17:10:05

客户介绍

某电子商务企业，核心宗旨是购物与社区的相互结合，为更多消费者提供更有效的购物决策建议。

客户痛点

该企业上每天有几百万用户在线交流时尚、购物的话题，这些行为会产生大量数据，当这些数据源产生数据后，需要有一个组件获取数据源的数据，并将数据写到 Kafka。研发团队以往的解决办法，一是通过 Logstash、Filebeat 等开源的数据存储方案处理，二是研发团队开发代码实现这种逻辑。
随着业务的不断扩张，数据越来越大，为了保障可用性、可靠性以及性能相关的内容，需要大量的研发资源投入，因此，亟待新的解决方案支持。

Serverless + Ckafka 解决方案

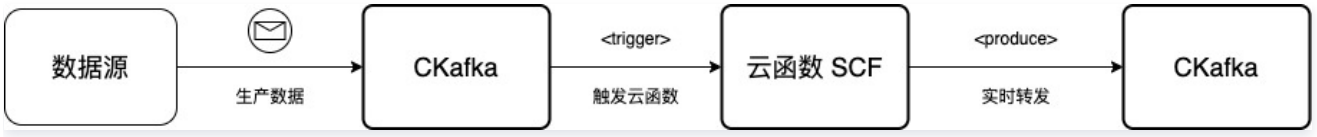
企业团队对比市场上的技术解决方案，从学习成本、扩缩容能力以及人工维护成本和稳定性方面考虑，腾讯云 Serverless 云函数具有天然的优势：

- 支持多语言。
- 学习成本低，无需学习开源方案和分布式调度。
- 无限的弹性扩容能力。
- 多重触发方式，事件触发、定时触发、主动触发。
- 几乎没有集群稳定性和可用性的维护成本。
- 按实际用量计费，1ms计费，费用较低。

> **说明**
> 如需了解更多 CKafka，请参见 消息队列 CKafka 。

腾讯云 Serverless 云函数 + Ckafka 提供自建的 UI 交互界面，可进行流量告警配置，同时控制台上可进行扩容配置且安全可靠。
腾讯云 Serverless 云函数团队为企业提供的业务解决方案，是通过云函数将一个实例中某个 Topic 的消息转储至另一个实例对应的 Topic上，对比原来的 Connector 方案，云函数能够通过腾讯云控制台进行管理，能控制触发阈值，触发开关等，可以很方便地对每个函数进行管理。简单来讲：

- **消息转储**：将 Topic 的消息同步至离线集群。
- **集群迁移**：在集群迁移合并的过程中起到双写作用。



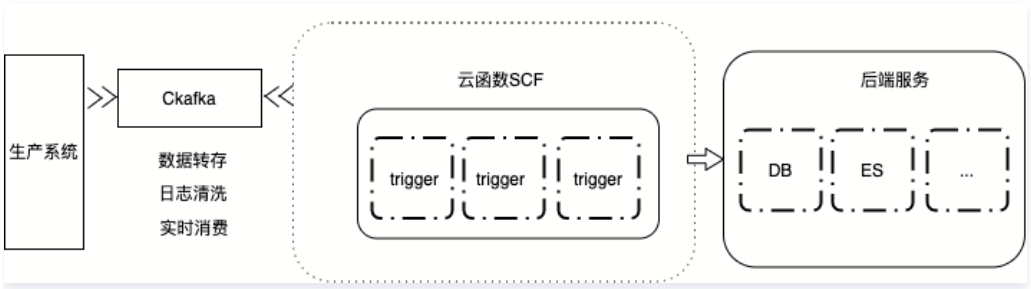
经过对比，腾讯云 Serverless 云函数 + Ckafka 是最优的解决方案，企业最终决定选择使用腾讯云 Serverless 云函数 + Ckafka 运用在消息同步业务上。

方案优势

Kafka 社区的繁荣，越来越多的电商用户开始使用 Kafka 进行日志收集、大数据分析、流式数据处理等。腾讯云 Ckafka 也借助了开源社区的力量，和云函数结合，推出了非常实用的功能，其优化点包括：

- 基于 ApacheKafka 的分布式、高可扩展、高吞吐。
- 100%兼容 Apache KafkaAPI（0.9及0.10）。
- 无需部署，直接使用 Kafka 所有功能。
- Ckafka 封装所有集群细节，无需用户运维。
- 支持动态升降实例配置，按照需求付费（开发中）。
- 对消息引擎优化，性能比社区最高提升50%。

如下图，云函数可以实时消费 Ckafka 中的消息，例如，数据转存、日志清洗、实时消费等。并且，像数据转存的功能已经集成到 Ckafka 控制台上，用户可以一键开启使用，大大降低了用户使用的复杂度。



对比使用云主机自建 Ckafka Consumer 的方式，云函数帮用户屏蔽掉了很多不必要的开销：

- 云函数控制台上可以一键开启 Ckafka 触发器，帮助用户自动创建 Consumer，并由云函数平台来维护组建的高可用。
- Ckafka 触发器自身支持很多实用配置，例如支持配置 offset 位置、支持配置1 – 1万消息聚合条数、支持配置1 – 1万次重试次数等。
- 基于云函数开发的业务逻辑，天然支持弹性伸缩，无需额外搭建和维护服务器集群等。

在线教育行业案例
音视频转码最佳实践

最近更新时间：2022-03-23 16:56:14

客户介绍

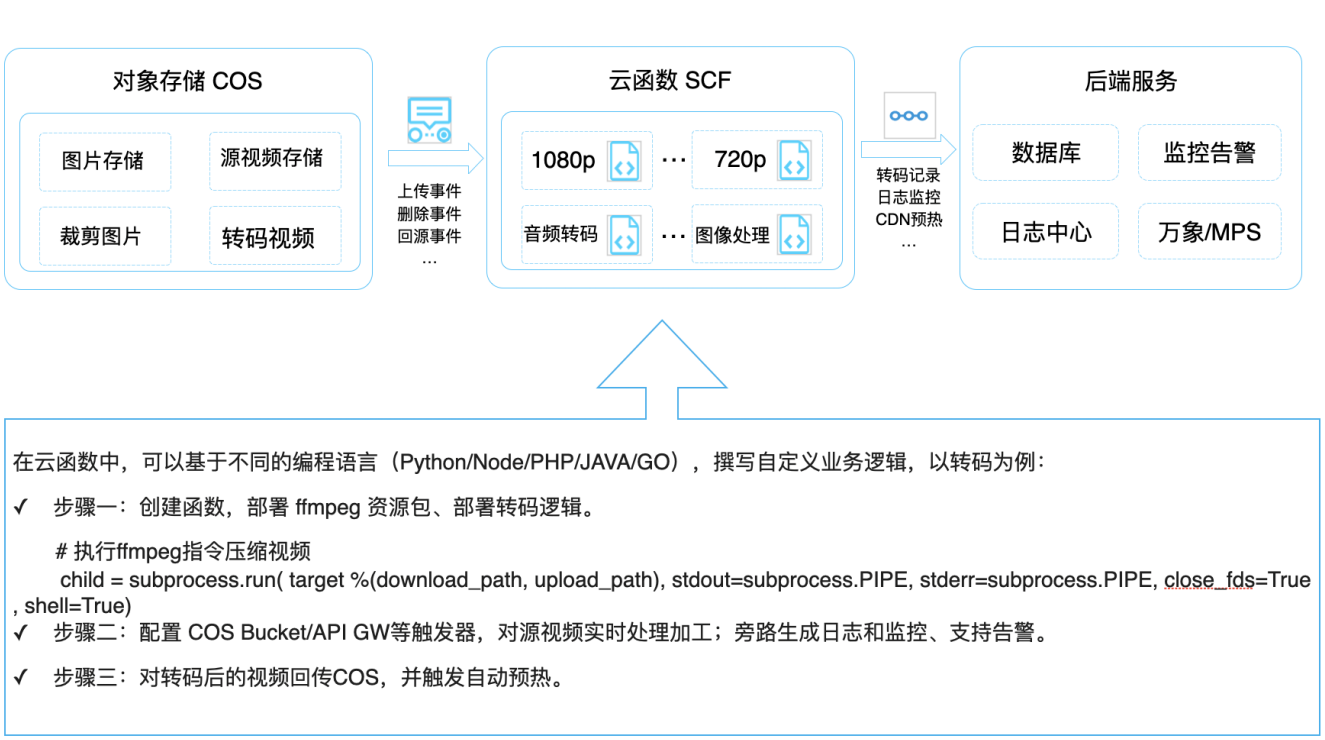
某国内在线教育企业，于2006年在美国纽约证券交易所上市，总部位于中国北京市海淀区中关村，是一家综合性教育集团，同时也是教育培训集团。该企业业务包括外语培训、中小学基础教育、学前教育、在线教育、出国咨询、图书出版等各个领域。旗下还有多家教育子品牌。

客户痛点

在每年暑期时，会有大量学生在企业平台上学习。在此之前都是在自建的机房里基于服务器和 NFS 来实现音视频课程的存储和转码逻辑。但由于暑期流量比较大，IDC 里的服务器不一定能满足计算需求，同时自建服务的硬件采购周期较长，于是期望寻找一种弹性方法，既能够支持快速业务部署，又能高效的完成转码功能。
在视频应用、社交应用等场景下，用户上传的图片、音视频的总量大、频率高，对处理系统的实时性和并发能力都有较高的要求。传统的容器服务，需要用户自己维护容器集群，弹性伸缩效率较低。

Serverless 解决方案

腾讯云 Serverless 云函数支持自定义转码函数，帮助企业快速搭建定制化任务处理能力，弥补当前单独云服务的功能盲点，将 ffmpeg 业务方便地从物理机、云主机或容器中移植到云函数。
使用云函数 + ffmpeg 和 COS 联动做音视频转码的运行原理如下图：



在云函数中，可以基于不同的编程语言（Python/Node/PHP/JAVA/GO），撰写自定义业务逻辑，以转码为例：

✓ 步骤一：创建函数，部署 ffmpeg 资源包、部署转码逻辑。

```
# 执行ffmpeg指令压缩视频
 child = subprocess.run( target %(download_path, upload_path), stdout=subprocess.PIPE, stderr=subprocess.PIPE, close_fds=True , shell=True)
```

✓ 步骤二：配置 COS Bucket/API GW等触发器，对源视频实时处理加工；旁路生成日志和监控、支持告警。

✓ 步骤三：对转码后的视频回传COS，并触发自动预热。

技术方案上，在云上采用云函数 + COS 的方式，可以支持弹性伸缩，即使将本地流量全部切到云上，也能全部承载。新的业务流程，会加入任务调度模块，当业务流量进入时，可以自动或者手动将流量分别导入自研服务和云上服务，同时在流程里加入了很多高可用技术，例如通过任务 TraceID 进行全链路追踪、云端计算失败本地自动重试等。新的方案里，云端服务开发简单，且无需投入太多运维精力，同时具有更优的成本优势。按量计费（用多少付多少）的云函数计费方式，降低了大量的资源成本。

Serverless 应用价值

使用腾讯云 Serverless 云函数实现音视频转码服务的优势：

- 云函数提供标准运行环境，并且保障资源的高可用和弹性伸缩，无需专人维护。
- 云函数基于实际业务消耗收费，不存在资源浪费。

- 云函数的开发调试流程效率会更加高效，依赖和业务解耦，可以分别单独更新，支持实时热更新。
- 运行环境隔离，单次请求失败不影响其他请求的正常执行。

当现有业务引入云函数时，需要注意以下两点：

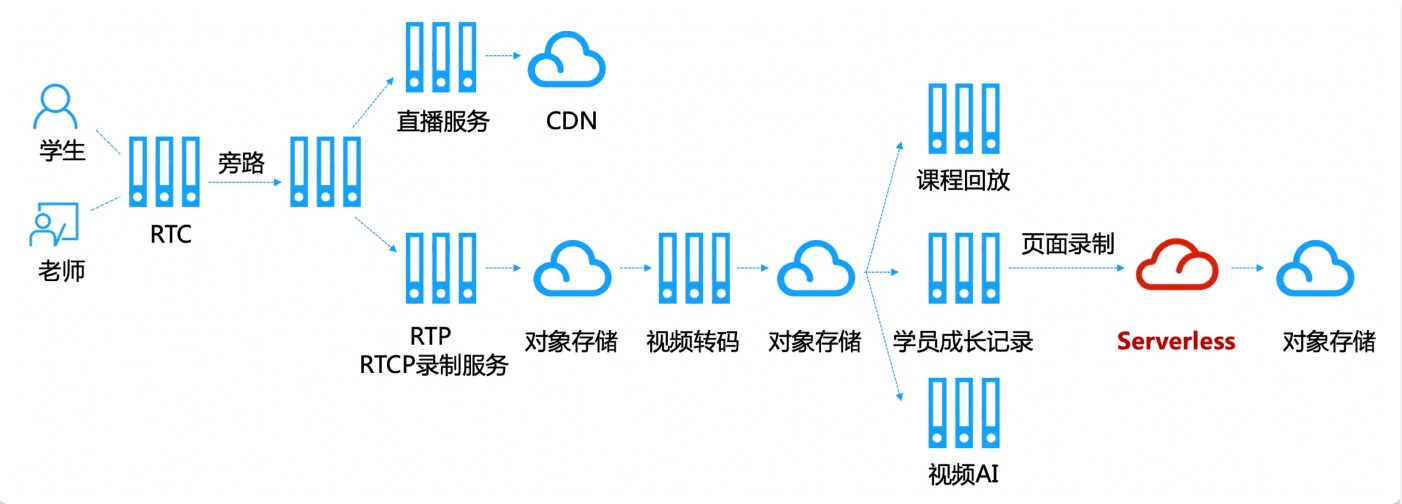
- 云函数的引入，需要对接现有 CI/CD 流程，开发方式上有一定的转变。
- 现有业务代码需要做一定的改造，主要改造是将 ffmpeg 集成到函数代码中，与代码文件一起部署到 SCF。

全景录制最佳实践

最近更新时间：2023-06-13 14:17:11

客户介绍

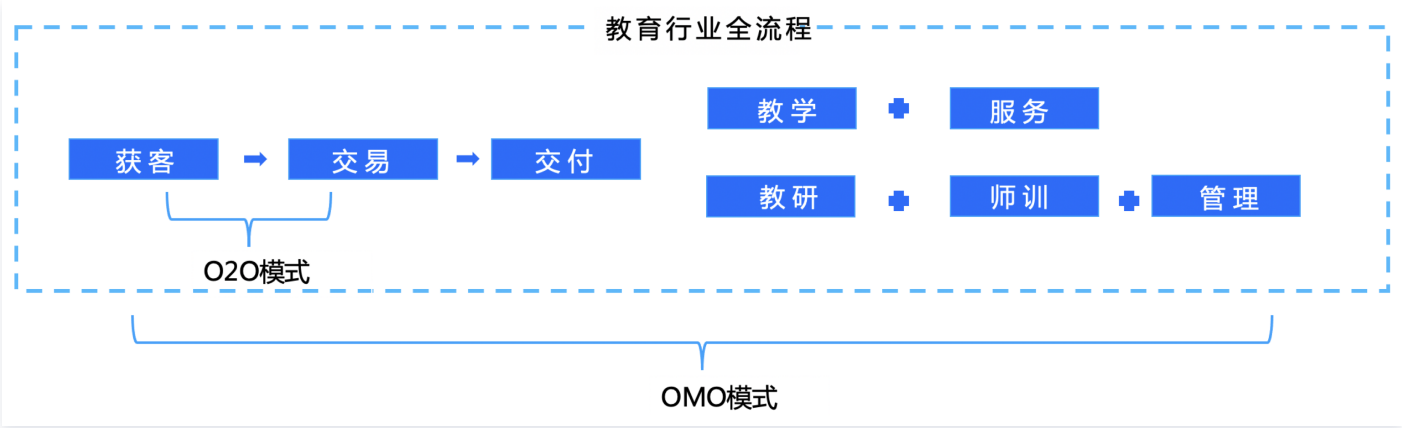
某国内在线英语教育企业作为国内最早实践 OMO 模式（Online-Merge-Offline）的在线教育机构之一，通过深耕行业累积的教研管理经验和真实场景教学数据，在自研的配套软件系统的基础上，支撑起整个教学环节。



业务需求

OMO 模式是指线上和线下相融合，利用技术发展打通线上线下，大幅提升市场效率的商业模式。当前，教育机构实现 OMO 模式有三种方式自研、外包服务，以及购买 SaaS 服务。但这三种方式存在以下劣势：

- **成本高：**OMO 模式对 AI、大数据、云计算、物联网等各领域技术要求较高，自研能力需要组建一支专业的技术团队和高昂的成本投入。
- **难匹配：**外包厂商对原有架构并不熟悉，在需求实现和开发权限上难以恰当匹配。
- **灵活度低：**购买 SaaS 服务虽然看似简单，但存在灵活性不佳、数据孤岛、无法满足定制化需求等问题。

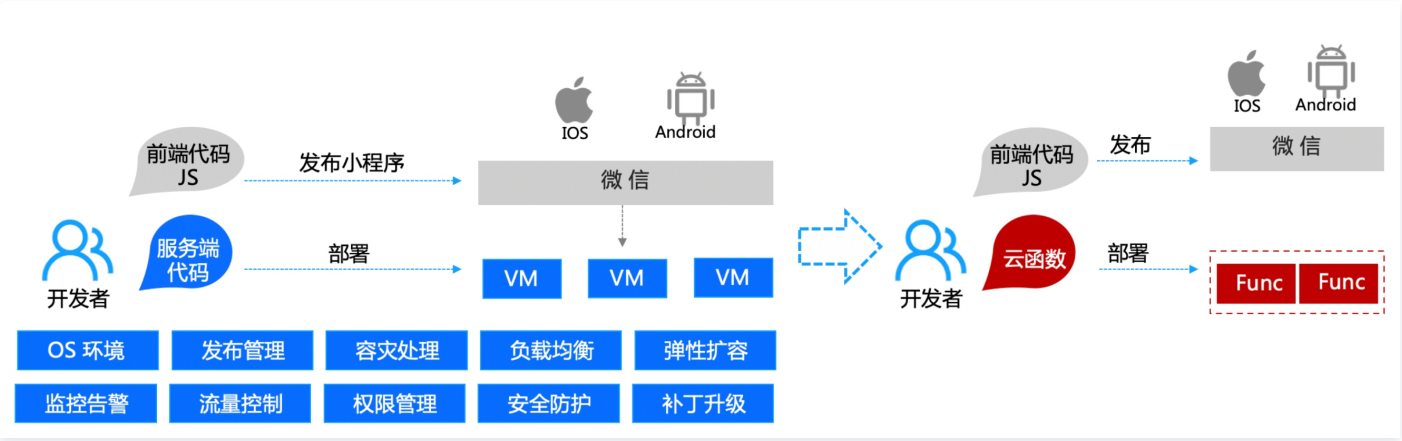


小步快跑，别让教育行业困在代码中

在线教育行业的 OMO 模式涉及复杂流程和新技术，对运营成本、技术能力、盈利能力提出更高的要求和挑战。而影响在线教育行业营收的关键是用户转化率、续费率和转介绍率，本质上提升转化率的基础是教学体验和学习成果。这需要教育行业用最小的成本，快速试探用户的反馈，不断验证市场反应。基于对用户行为的理解，业务侧需要快速迭代新功能：支持回放、收藏、下载和转发学生在课堂中的精彩影像片段，留下学生在成长进步路上的精彩瞬间。让专业的人做专业的事，让教育行业只需专注业务逻辑。

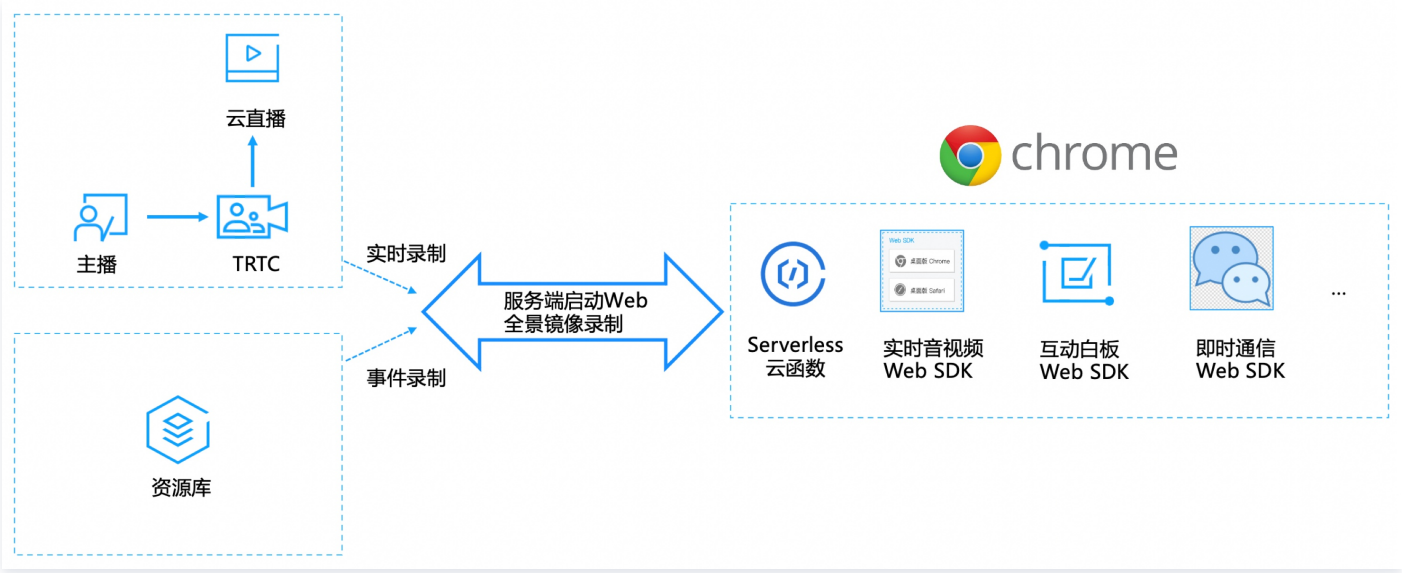
Serverless 解决方案

使用 Serverless，客户无需维护跟业务无关的底层基础设施，能够专注于客户自己的业务，缩短研发周期，真正实现**小步迭代、试错快跑**的敏捷开发。在 Serverless 的架构中，用户操作的是服务化的组件，例如存储服务、授权服务等，缩短了开发周期，降低了开发难度，且避免了由基础设施产生的延迟。



「全景录制」实时音视频 TRTC + 云函数 SCF 解决方案

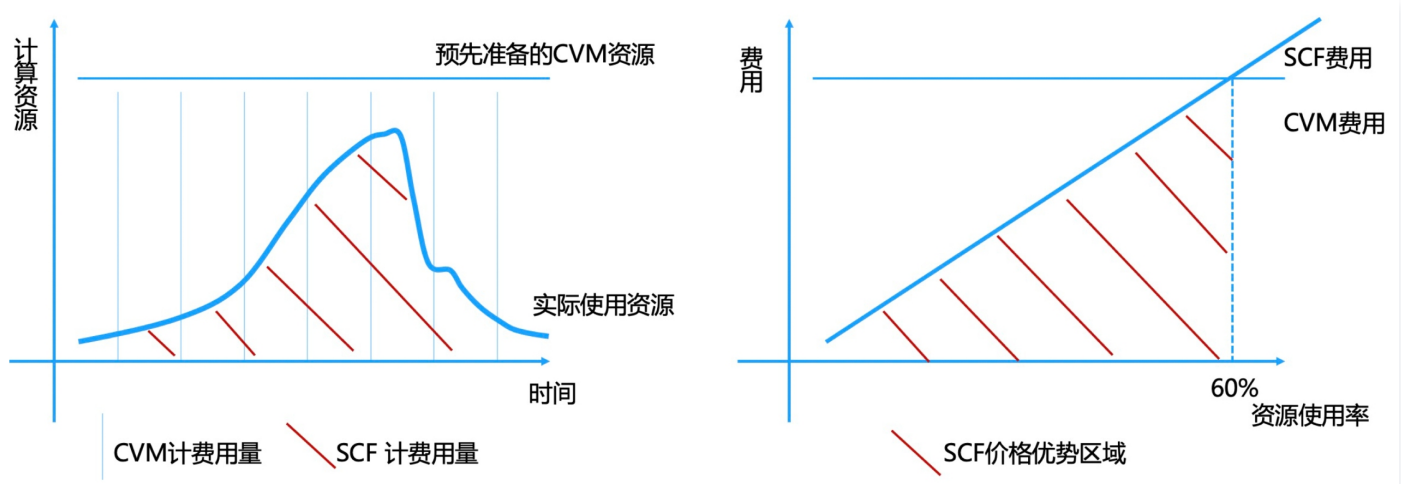
1. 利用云函数 SCF 实时录制直播内容，生成 ts 文件存在指定位置。
2. 精彩片段触发之后，读取触发点前几个片段，调用函数资源池，快速生成精彩片段。
   全景录制流程图如下：



- 云函数 SCF 一键触发，实时弹性启动，服务端执行浏览器全景镜像录制。
- 浏览器多路解码、一路编码，降低算力消耗。
- 浏览器实现多路直播流、信令、白板等同步集成，简单直观。
- 录制过程灵活调整布局，切换主播、观众视角。
- 实时音视频 TRTC 与 云函数 SCF 内网推拉流，极大提升实时性，降低网络流量成本。

降本增效，技术进步的最佳体现

在线教育的技术投入往往需要投入不少成本和人力，以线上课堂的业务系统举例，需要 iOS 开发、Android 开发、PC 开发、后台 Web 开发等，即使搭建最简单的教育系统，通常至少需要10人左右的研发团队，还不包括后期运维、服务器和时间成本的投入。同时，在线教育行业的用户流量波峰波谷明显，通常会面临意料不到的流量突增。传统 IDC 服务器不一定可以满足计算需求，自建服务的硬件采购周期较长，因此亟待找到最小成本的可行性方案。
在 Serverless 解决方案上，即使将本地流量全部切到云上，也可以全部承载，支持弹性伸缩。云函数大大节省了运维成本和服务器开销，1ms粒度的按用量计费模式，成本可降低70%（具体收益结合业务场景和使用案例预估）。云函数按用量计费模式如下图所示：



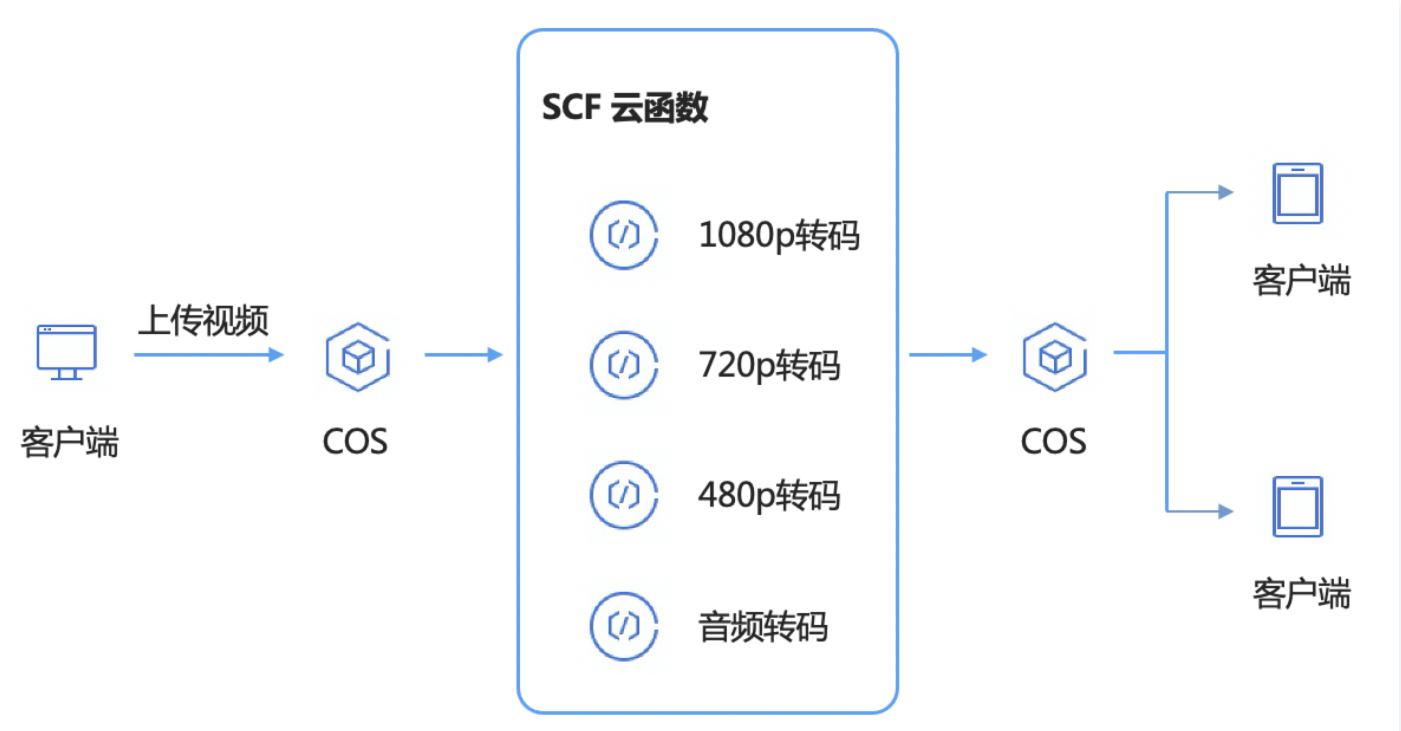
腾讯云 Serverless 团队曾服务过多家业界知名的教育客户，Serverless 作为下一代计算资源的使用范式，真正意义上实现了 IT 资源的按需使用。结合腾讯云全球互联的数据中心，面对教育行业的区域分布广、延迟敏感强、区间并发高等场景特性，提供了针对性的解决方案，高并发场景下资源快速拉起，低谷时进行资源快速回收，满足用户需求的同时降低资源的使用成本。

腾讯云 Serverless 教育解决方案全面升级

聚焦音视频和多媒体处理

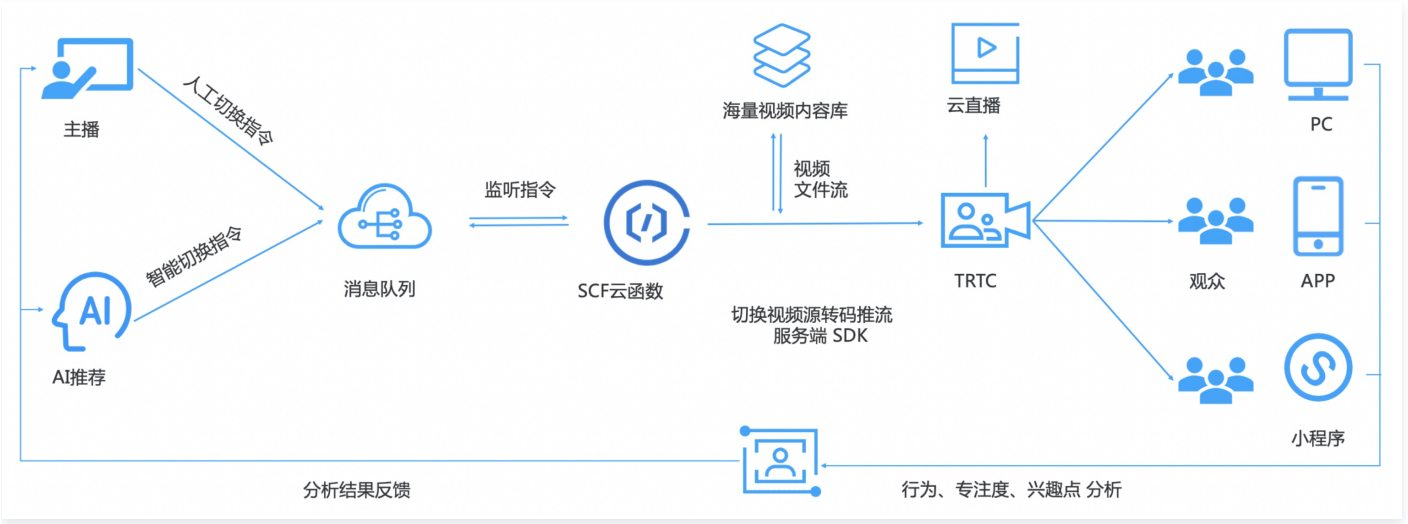
1V1 课堂、小班课、互动大班课和双师课堂是在线教育的四大基础场景，对技术的需求是满足高质量的音视频处理和高并发的稳定性。
Serverless 音视频转码、推流、直播和图片处理等方案，支持灵活、自定义的转码方式，快速搭建 RTC、RTM、互动白板、实时录制等各种产品组合的定制化任务处理能力，补足当前单独云服务的功能盲点；采用分布式架构，能够承受住海量高并发场景，99.99%高可用，满足千人大班课场景需求。

- **高效整合**
  通过云函数 SCF 联动 Faas + Baas 服务，将视频上传、视频处理、图片处理、存储场景、数据处理有机地整合为一体。
- **长时运行**
  利用云函数的长时运行机制，支持12h – 24h的运行时长，可覆盖大文件耗时较长的转码场景。
- **平滑迁移**
  支持用户自定义配置 FFmpeg 命令参数、以及部署自建 FFmpeg，转码方式灵活。
- **成本低廉**
  所有组件交互均走内网，无需额外流量费，1毫秒粒度按用量计费，拥有显著的成本优势。



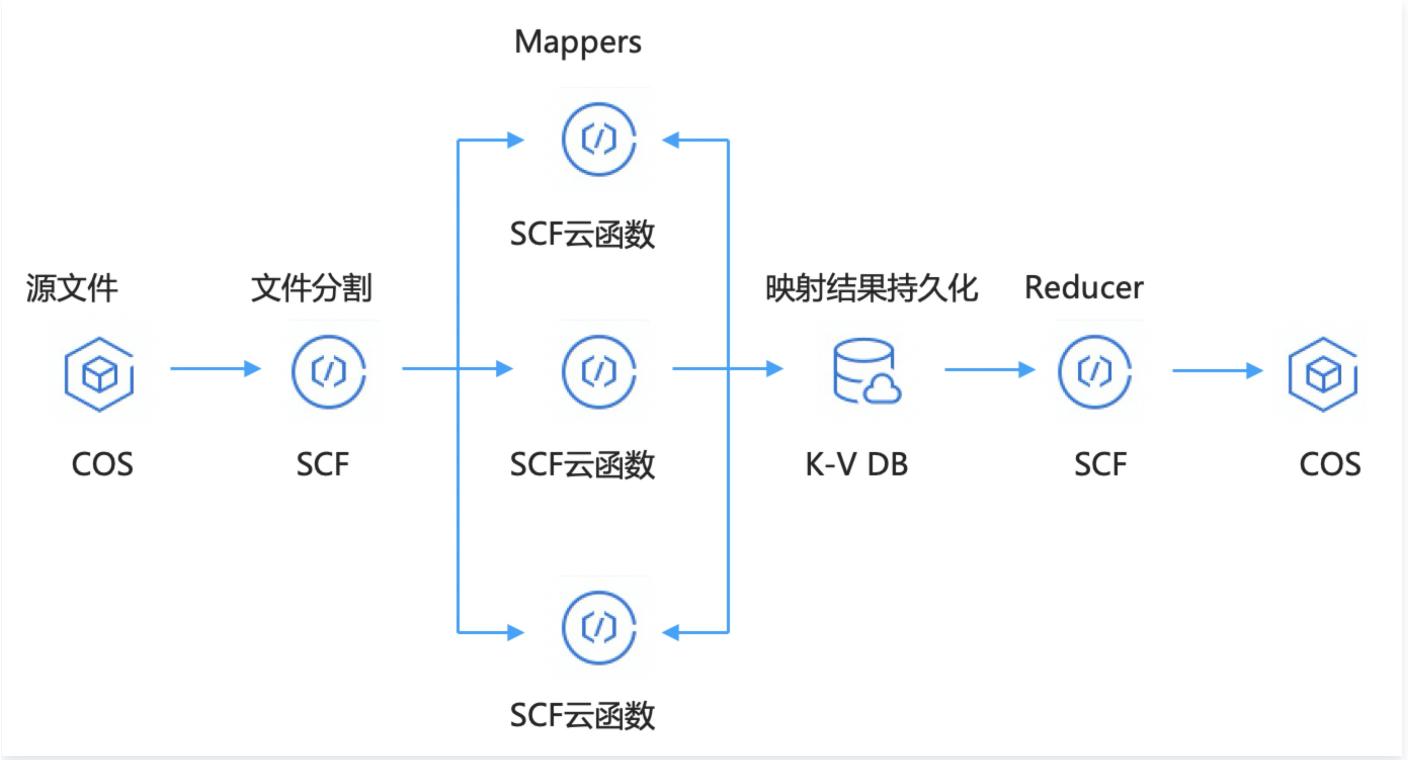
AI 互动和内容监测与审核

在线教育行业发展中势头迅猛的 AI 互动课堂，可以根据学生的学习进度提供个性化的教学方案和游戏化的互动体验，成为用户、在线教育机构和资本关注的焦点。
腾讯云 Serverless 结合 AI 及音视频技术，提供视频智能化编排处理解决方案，包括隐藏式数字水印、AI 智能审核涉黄内容，辅助视频内容分析和生产，进行实时个性化推荐等。以下为智能推荐流程图：



数据 ETL 处理

在教学过程中，学生画像、学情分析和课堂质量分析等产生的大数据需要进行沉淀和分析，帮助学生、家长和老师用科学和量化的方法把握教学效果，及时调整课堂内容和教学进度。
腾讯云 Serverless ETL 解决方案可以轻松地进行大容量数据计算，例如对源数据并发执行多个 mapper 函数，并通过 reducer 函数汇总执行结果。腾讯云 Serverless 的技术优势将持续拓展到「备、教、练、考、评、管」的教学全流程中。



音视频实时互动 Serverless 系列解决方案

- 云函数和 Headless Chrome 进行实时渲染录制合流
- 使用 Serverless 云函数为 TRTC 输入在线媒体流
- 使用 Serverless 云函数实现 TRTC 单流 / 混流录制

移动互联网行业案例

最近更新时间：2022-10-18 15:57:28

客户介绍

某移动互联网企业，致力于“在人机共存的世界里，用科技让生活更美好”。2014年在纽交所正式上市后，在保持原有业务优势基础上，逐渐从移动互联网向以AI 驱动的产业互联网进行战略升级，并构建了垂直一体化的 AI 能力，包括自研芯片算力、算法能力、系统能力、应用能力、商业大脑。

客户痛点

为追求速度体验和极致的 SEO 效果，越来越多的技术管理者和架构师倾向于采用 SSR (Server Side Rendering) 技术来构建前端项目，以支持同构代码的服务器端渲染。企业团队在2016年时开始使用 React，2017年开始研究并尝试 React Server Render，同期 Facebook 网站已经采用 Isomorphic 技术实现，性能相对较好。为满足公司业务需求和技术传承，客户自研了前端技术框架 Koot.js，目前已成为客户前端的主要技术方案。
HTML、CSS、JS 以及其他资源都能够做到按需加载，但 SSR 更能够 Isomorphic（前后端同构），结合 SSR (Server Side Rendering) 和 CSR (Client Side Render) 的优点，让用户浏览网页时不管是首屏还是随后操作的其他页面都能更快的展示响应。
SSR 项目落地的时候通常不是很顺畅，项目部署的时候需要具备服务器技术能力才能和运维顺畅沟通，因此项目落地不仅需要前端同学掌握后端开发能力还需要对运维技术、并发等问题多方面考虑，这对前端技术同学的技术能力有较高要求。

Serverless 解决方案

Serverless 技术可以降低前端对服务端和运维的技术能力要求，更适合大部分需要使用 SSR 的前端团队。腾讯云 Serverless 提供了比较全面的组件支持，与 Serverless Cloud Framework 基本无缝结合，周边社区和生态支持也比较到位，使用过程中能避免很多问题。在选定平台之后接下来的部署将较为顺畅，因为 Serverless Cloud Framework 提供了很多标准化的接口，在封装 Koot.js Serveless 组件的过程中能够更便捷省心。
客户早期就进行了前端分离的开发，前后端完全使用 API 对接，协作改变不大。由于使用了 Isomorphic，因此对 API 的要求变高，用户的请求不止来源于 Node 服务器，还有来自浏览器的请求，对安全性要求也会更高。

SSR 落地建议

未来的 SSR 都应该是 Isomorphic 模式，带来的好处是减少传输成本，分摊渲染压力，用户体验也会有所提升。体验的提升其实非常小，网络情况好时，用户几乎感知不到，但小小的提升在技术开发中却做出了非常多的工作，因此我们会越来越完善技术框架，让业务开发同学能够快速开发出需求，同时又享有 Isomorphic 带来的技术体验。
对于还未开始使用 SSR 的团队，如果是 ToC 产品业务，建议尝试 SSR，能够让用户尽快看见页面内容。
前端的 SSR 一定会考虑是否需要 Isomorphic，如果小团队建议先尝试比较流行的框架，例如 Next.js、Nuxt.js 等，也推荐体验 Koot.js。Next.js、Nuxt.js 这类框架在腾讯云 Serverless Cloud Framework 都有现成的组件支持。无论是 SSR 还是 Serverless，最好都是基于现有框架，如果没有足够业务支持建议无需浪费精力自己研发框架，学会一个框架的成本要远小于维护一个框架的成本。

在线效率工具行业案例

最近更新时间：2023-07-31 15:00:33

客户介绍

某企业提供多功能笔记类应用，是知识管理领域中影响力较大的工具之一。它不仅是管理个人信息的智能助手，也是提高团队效率的企业工具，同时还是一个内容平台，富集高价值信息，以及一个优秀的知识和信息相关智能硬件的生产商。
该企业的服务对象主要是高教育程度、高收入的知识人群、广大知识工作者和学生，服务用户已过亿。产品采用业界领先的 Freemium（免费增值）商业模式，保持了非常高的用户活跃度，付费转化和续费率。同时，该企业积极拓展用户的使用场景，自主研发了多款智能硬件，有效支持手写、扫描、语音等多种输入场景。

客户痛点

客户自2018年完成资本重组以来，产品功能频繁迭代，业务增长迅速。技术团队频繁需要应对一部分短期需快速上线的功能或项目，沿用之前传统服务或微服务开发交付模式，都无法满足工程排期需求。因此开始寻找解决方案。
企业分析了其自身业务特点，推出的新功能相对独立，自身业务逻辑清晰，与其他模块耦合度低。同时并发处理量与用户实际行为有关，前期无法准确预估资源使用情况。

Serverless 解决方案

使用腾讯云 Serverless 技术后，在开发体验上有了明显的提升：

- 开发速度明显加快，Serverless 或云函数 SCF 都提供丰富的预置工程模板，且与其他腾讯云服务例如 CMQ，COS 等紧密集成。
- 部署方便，无需前期复杂的任务编排和资源配置步骤。
- 业务上线后便于维护，运维人员不再需要考虑压力和扩容问题。
- 利用完备的日志和统计功能，可以即时、便捷的掌握功能服务的运行状态。

目前，企业逐步在一些内部业务数据处理和用户异步通知功能中尝试利用腾讯云 Serverless 等框架进行快速开发并交付上线。
未来，企业还将进一步拓展尝试使用腾讯云 Serverless 技术，主要从以下几个方向进行：

- 小程序服务端功能。
- SEO/SSR 相关。
- 用户异步交互，如定期提醒，账户状态相关通知等。

最后，给还在考虑使用 Serverless 技术的团队一些建议：
如果产品团队尝试为产品迭代或者开发产品新功能，可以考虑使用腾讯云 Serverless 技术，Serverless 与其他模块耦合度低，且无需担心用户使用量，Serverless 拥有无限弹性扩容的能力，几乎无集群稳定性和可用性的维护成本。
同时，如果团队初期要进行一些新项目的验证，需要进行数据拉取、数据分析，可以考虑使用 Serverless ETL ，进行数据抽取（Extract）、数据转换（Transform）、数据加载（Load），Serverless 在这方面的优势在于灵活，不影响已经有项目的数据处理流程，可单独运行并满足数据验证需求，同时具有较低的学习成本和费用成本优势。

餐饮行业案例

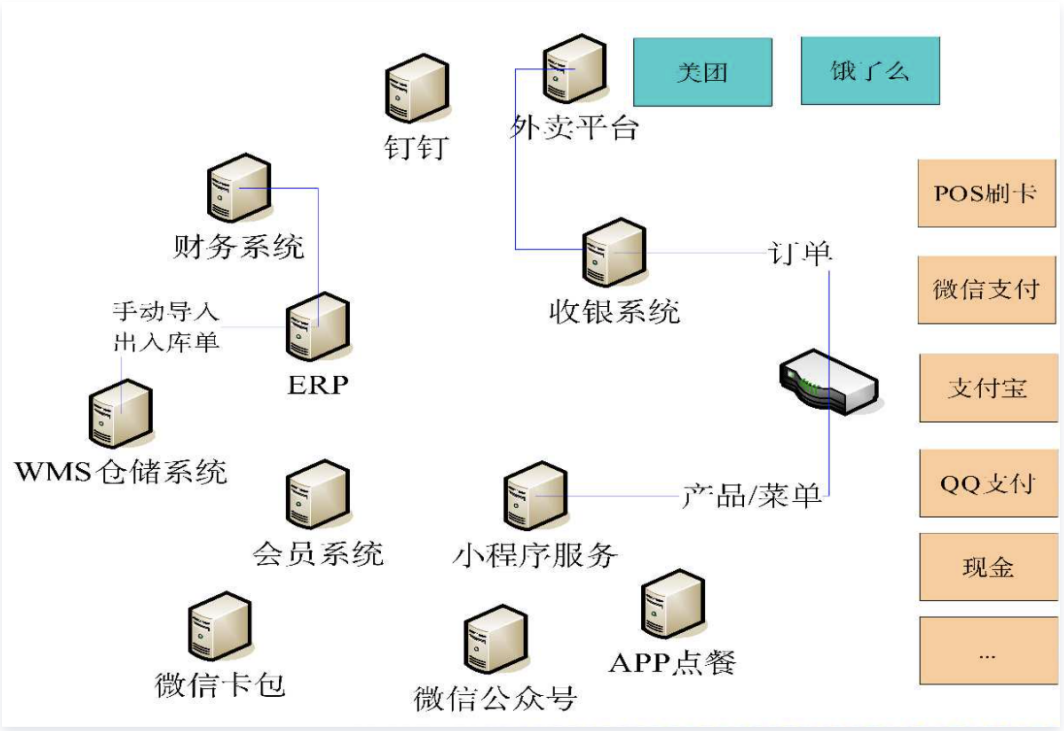
最近更新时间：2024-12-12 15:04:42

客户介绍

国内某连锁餐饮企业，目前在深圳、广州、上海、苏州、佛山、惠州、东莞、昆明、重庆等地拥有140多家直营门店。同时也是红杉资本成员企业，是红杉资本在中国投资的餐饮企业之一。

客户痛点

- 2017年，企业业务系统信息孤岛严重，有二次开发困难，大量用户用的系统语言和接口都不一样。
- 2017年底，团队自研推出了小程序点餐系统，和后台业务系统打通，亟待解决多个系统之间的互联互通问题。另外业务系统耦合度高、业务拆分困难、系统稳定性差，各种紧急的业务需求和活动无法及时满足。



新餐饮介绍

在上述痛点下，企业开始对数字化新餐饮进行调研，新餐饮本身也是人货场的重构，主要为实现日常经营管理所有的数据能够全流程在线化，主要分为以下四块：

- 点餐、下单、支付、制作、出品实现实时在线化。
- 用餐评价，会员管理可以在线化，支持数字化营销。
- 供应链订货、收获、盘点、损耗等流程在线，数据打通。
- 人事、考勤、成本核算等在线化、工具化、高效化。新餐饮转型需要进行全方面的数字化改造。

经过一系列调研分析，企业决定向云上进行业务迁移，并采用腾讯云 Serverless 云函数承载业务。

腾讯云 Serverless 解决方案

使用 Serverless 云函数可以实现的业务功能非常多，例如：

- 微信小程序的服务应用，云函数提供计算支持。
- 公众号消息推送服务。
- 实时通讯服务。
- 不同的业务系统数据同步，例如，金蝶的 ERP 系统以及 WMS 系统的对接和数据同步，SOS 餐饮系统和金蝶 ERP 数据接口的同步。
- 统一的支付服务、订单的付款、订单的支付，定时异常数据邮件提醒。
- 第三方外卖平台，第三方系统的数据抽取及处理入库。需要临时上线的功能需求或接口对接。

使用 Serverless 的优势

- **Serverless 云函数是天然微服务的架构**
  一个函数可以看作一个服务。微服务本身能解决很多数据耦合度的问题。
- **适合处理大量零散的定时任务**
  例如，可以极大的减少部署独立的定时器服务，解决集群服务的定时任务管理问题，尤其是并发的集群定制任务。
- **更低的开发难度，更高的开发效率**
  资源可以直接使用，针对不同的业务场景，可以借助不同语言特性，结合研发团队的能力，更快速的实现对应的需求。利用研发人员熟悉的语言去做业务场景，而非需要团队使用统一的技术架构，团队可以更灵活快速推进需求。
- **更低的运维成本**
  云函数的计费方式为按量计费，且未运行时不产生费用，同时支持自动扩缩容。大大降低了资源和人力运维成本。

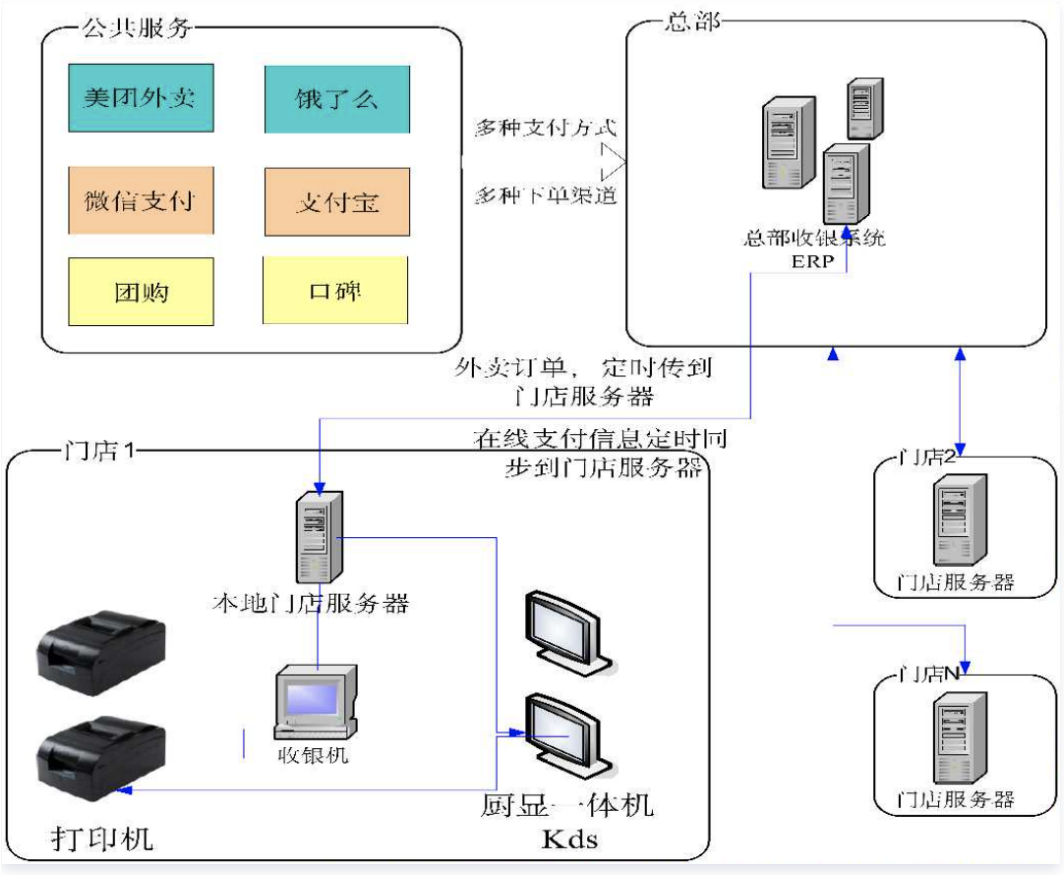
案例一：云打印服务

传统架构

用云函数 SCF 实现云打印服务。云打印是一个小型的打印系统，用来解决用户在门店下单之后，打印机打印小票的业务流程。
在传统架构环境中，每一个门店都需要准备以下硬件：

- 部署一个本地服务器，这个本地服务器提供一个本地服务。
- 一个交银机
- 一个一体机
- 一个打印机

这些硬件通过本地服务器进行管控。总部也需要有服务进行数据的整合。下文以美团订单为例进行简单介绍：
1. 美团的订单会先将订单下发到总部的服务器上。
2. 总部服务器会将订单数据推送到门店。
3. 门店服务器接到数据之后，通知到打印机进行打印小票。
目前，大量的订单皆为线上下单，无论是小程序订单或是其他平台订单，都主要是通过网上下单，传统架构信息传递链路长，非常依赖网络链路的稳定性。传统架构如下图所示：



传统架构存在以下痛点：

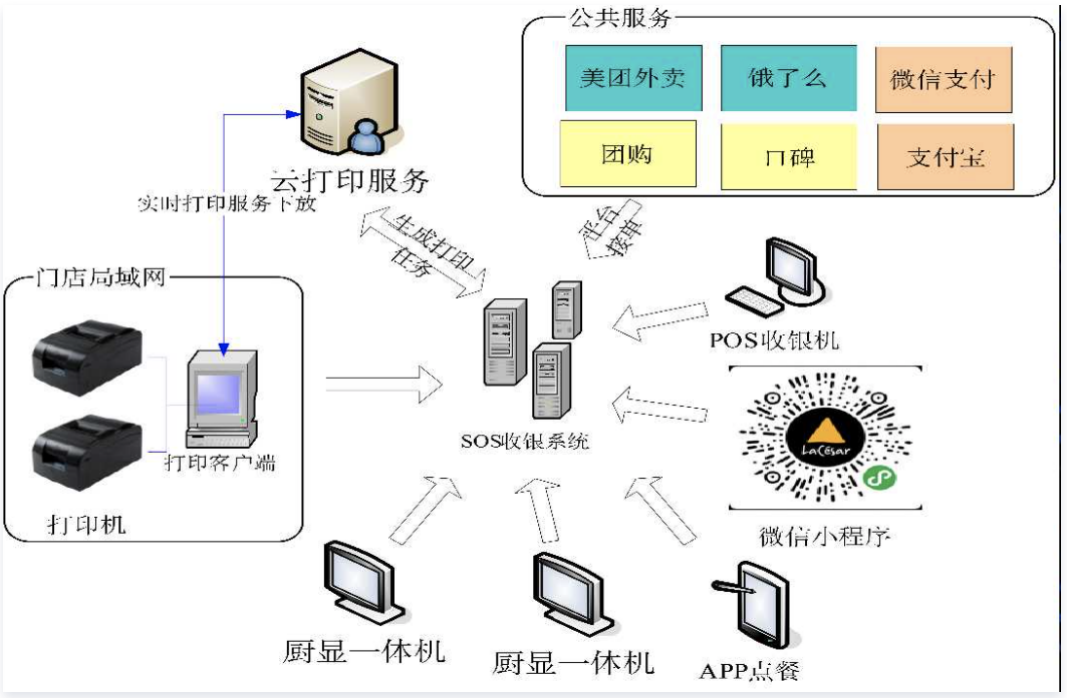
- 硬件及维护成本非常高。门店需要部署独立的本地服务器。
- 总部对于门店的服务情况无法快速掌控，由于数据每隔一段时间才会通讯，很难做到数据的实时同步。

- 收银电脑普遍配置极低，订单数据无法实时上传汇总，数据同步存在时间差。

新的 Serverless 云服务架构

新的 Serverless 服务架构，取消了本地服务器，转为在原本配置比较低的收银电脑或 SDK 上部署轻量级客户端，通过 SCF 为客户端提供计算资源。

- 打开浏览器，即可实时传回所有订单数据，企业可以追踪所有的服务状态。
- 打印机部署在门店的局域网内部，所以单独在云端的服务就是 SOS 收银系统。
- 通过云打印服务连接本地的收银机客户端，通过该客户端来定制的打印机可实现实时通讯，随时知道门店的打印机是否能正常功能。

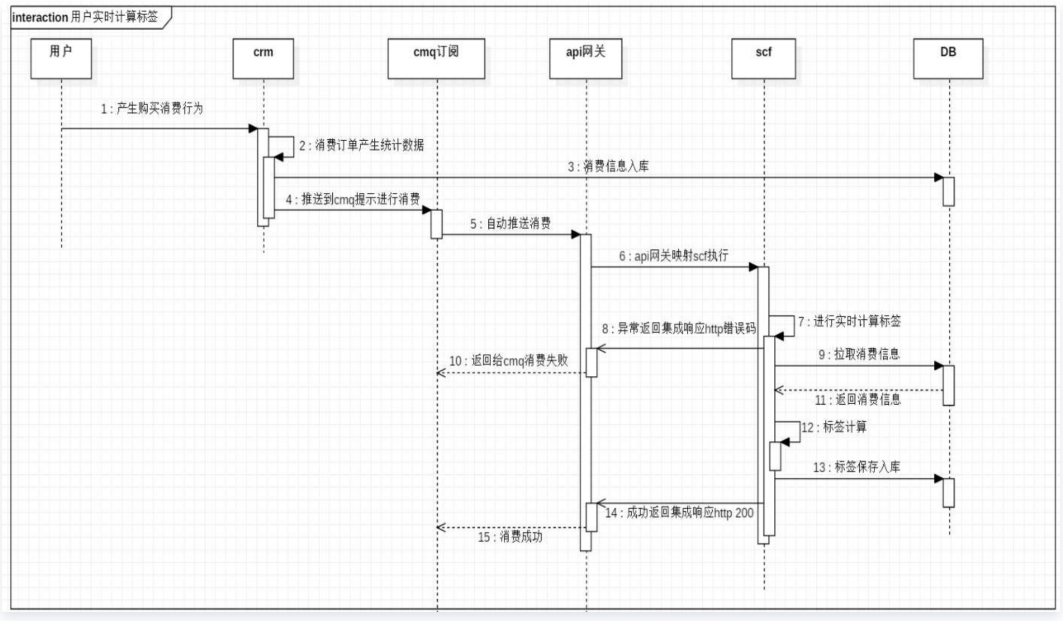


案例二：会员画像标签运算系统

用云函数结合 CMQ 、API 网关，完成会员画像标签运算系统。
会员系统比较重要的一项数据即会员画像。要基于会员消费行为消费轨迹生成用户会员画像，包括会员爱好、消费频率、消费档次等。企业希望能够快速实时的跟踪这些数据，通过 SCF 可以用极低的成本完成整个过程。
这个过程如何实现？以下为您简单介绍整个会员画像实时计算系统的运行逻辑：

1. 用户产生的购买消费行为数据推送到柜员系统，通过消费订单产生统计数据。
2. 数据统计完之后将写入数据库，同时数据信息会推送到消息队列 CMQ 中，CMQ 调用 API 网关触发用户画像功能的云函数执行。
3. 每一个用户画像功能的云函数会对应一类标签，包括消费频次、消费偏好等，这类云函数对其进行标签的实时运算：
   - 如果消费过程中产生错误，会将这个消息反馈给 CMQ，CMQ 反馈消费失败。

- 如果消费正常，会将计算完成的标签保存入库，最终生成用户画像。

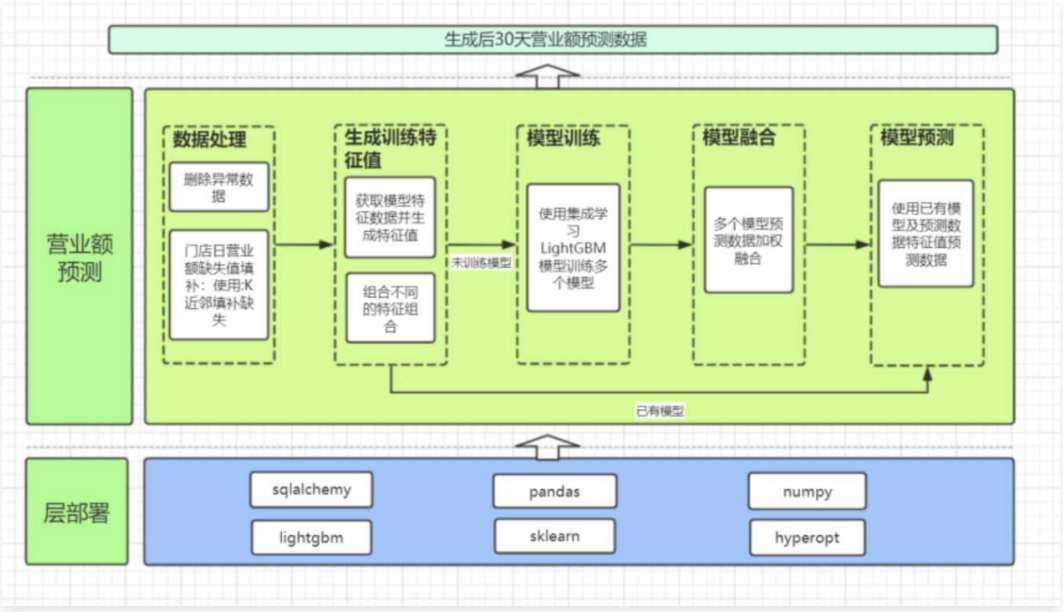


> **说明**
> 目前会员画像实时计算系统从2019年上线至今已完成3000多万会员用户画像。

案例三：营业额智能预测

通过云函数 SCF + 机器学习完成营业额智能预测。拉取近两年门店的营业额，最长两年，最少是一到两个月，每天滚动生成每一家门店接下来30天的营业额预测。
通过 SCF 完成如下：

- **抽取数据处理**：对营业额的数据做抽取，做相关的异常处理。
- **生成训练特征**：为数据组合特征。
- **定期训练模型**：使用 LightGBM 模型训练多个模型，多个模型预测数据加权融合，使用已有模型集预测数据特征值预测数据。通常两周到一个月做一次训练，到目前运行下来有接近半年的时间，目前营业额单店预测平均准确度高达87%。



腾讯云 Serverless 云函数的价值

上述三个应用案例中云函数主要价值点体现如下：

云打印服务

- 通过使用腾讯云提供的 Websocket 服务，减少了底层框架的开发难度，研发人员只需关注业务开发即可。
- 一位研发大概2周多时间就可完成，人力和时间成本节省至少50%。

会员画像标签运算系统

- 通过使用 CMQ，API 网关和云函数快速搭建起来一个高质量稳定的计算服务架构。目前从上线运行到现在0故障，运行稳定，且无需考虑服务器资源问题。
- 一位中级研发工程师三年的经验，从研究使用到搭建框架到开发完成不到2周时间。

智能营业额预测

- 通过使用函数的分层管理，减少了代码管理和调试难度。
- 用较低的成本和更高的效率实现了机器学习的智能应用。

整体来看，服务运维及成本、服务稳定性和性能都有较好的保障，而且费用投入远低于自建服务的方式，至少节省了**3台4核8G内存的服务器**。

开发环境	传统开发环境	云函数开发环境
开发周期	100%	55%
研发难度	100%	45%
成本	100%	30%

在线视频行业案例

最近更新时间：2024-11-28 15:20:43

客户介绍

国内某在线视频媒体平台是以视听互动为核心，集网络特色与电视特色于一体，实现 “多屏合一”，独播、跨屏、自制的新媒体视听综合传播服务平台，也是国内 A 股中最早国有控股的视频平台之一。

业务需求

需求1：音视频编解码业务

音视频团队承担的主要业务有：

- 自有内容音视频输出：
  - 每天海量时长的源视频内容生产。
  - 240P 到 4K 等不同清晰度，超50多种格式的转码及快速适配上线。
- 转码效率/算法探索：
  - 码率超高压缩算法。
  - 老片修复、视频超分、插帧。

在自有内容的音视频播出里，每天有海量视频的生产，需要消耗很大的计算资源。每天处理的量从240P到4K等不同清晰度、超50余种格式，包含 AVC、HEVC、MPEG 等多种编码格式。
而在 UGC 方面，需要快速将创作者的内容呈现给用户，给创造者带来收益。

需求2：主观感兴趣视频编码的研究

主观感兴趣视频编码，基于视觉冗余原理和对编码器引擎深度优化，相对原生 X264、X265 编码软件同等主观画质能够降40%以上低码率。

- **内容感知自适应编码**
  进行前期的预处理，通过分析这个视频动态复杂度、场景、镜头等，自适应匹配到 RDO 编码曲线最佳性价比码率，从而实现在不降低主观画质的体验上，降低30%以上码率。
- **主观感兴趣的区域编码**
  如下图所示，人眼聚焦的点在小汽车上面，可以看到一个人开着车的动作，利用人眼视觉感兴趣区域关注特点，通过基于 AI 的主观感兴趣区域预测模型，指导编码器在不同区域的编码质量权重分配，实现在同等主观体验下，能够降低15%以上码率。
- **视频编码图像增强技术**
  企业自主研发的系列编码图像优化技术，在不增加码率前提下，达到超越源片画质的体验提升。

需求3：音视频转码平台产品迭代

- 第一代，于2015年开始，基于 Hadoop 的 MapReduce 计算实现，处理量相对较少，随着业务的快速发展，后期该架构扩展将比较困难。
- 第二代，基于 mesos 的分布式资源管理框架，此时的业务已有所增加，日处理量得到提升，尤其是2016年起提出独播的战略，音视频内容需要快速生产上线，在此基础上实现了视频分段转码。
- 第三代，在2019年引进 AI 技术和腾讯云图像优化技术，采用 K8S 来实现资源的调度，自研调度及工作流编排。疫情期间，视频量增长非常快速，尤其在 UGC，超过以前十倍以上，引入了腾讯云 Serverless 后，能够快速实现集群扩容、提供所需要的计算资源。

为什么要使用腾讯云 Serverless?

从下图可以看到，从左到右云端计算的发展，云计算技术一直在提升。左边是早期物理机托管，到云主机到容器的出现，再到现在 Serverless 的出现，已经得到非常快速的发展。

早期的物理机和云主机和容器的特点，决定了30%的时间是处于低负载的情况。而早期视频的转码，大概在本地 IDC 机房有上百台服务器。白天资源严重不足，但到凌晨则处于低负载，服务器资源无法合理使用，而如今采用腾讯云 Serverless 以后，能够保证资源得到合理利用。

Serverless 能够带来哪些价值?

- 快速部署、弹性伸缩灵活的按量使用，降低业务使用瓶颈。
- 按使用场景实现任务的多地域调度能力，根据用户来源去调用每个区域资源，保证合理使用。
- 云上备份容灾机制，实现业务的不间断运行。
- 在确保性能的基础上降低资源成本和人力成本。

Serverless 落地实践

音视频转码

在云端利用 Serverless 实现音视频转码， 只需执行以下简单的三个步骤即可实现：

1. 创建函数，部署自研编码器资源包、部署转码逻辑。
2. 配置 COS Bucket 触发器，对源视频实时处理加工，旁路生成日志和监控、支持告警。
3. 将转码后的视频回传 COS，并分发到自建 CDN 或腾讯 CDN 节点。

核心优势主要是在于凭借云函数强大的联动能力，能够将视频上传以及视频处理和加工，以及视频提取、存储场景有机结合为一体。

支持灵活处理

能支持自定义转码函数，自身编码器能够快速进行部署，弥补单独服务，云服务的盲点。

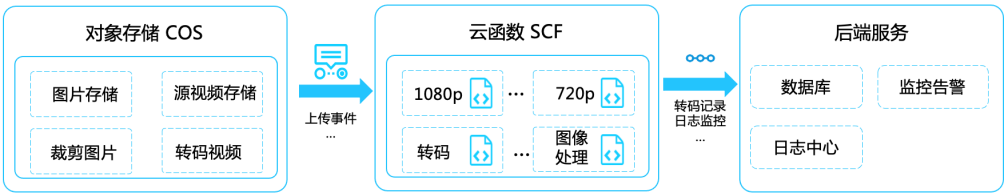
平滑迁移转码系统

线上 UPGC 内容采用的云厂家的点播服务，编码器处于不可控的状态。而采用 Serverless 后能够平滑迁移转码系统，将自研的编码器迁移到 Serverless，可自由调节所需参数，达到优化视频质量的目的。

降本增效

使用云函数 Serverless 最大的优势之一，能够大幅降低成本，在上线 Serverless 后，计算资源的**成本降低了45%以上**。

云函数转码原理及优势



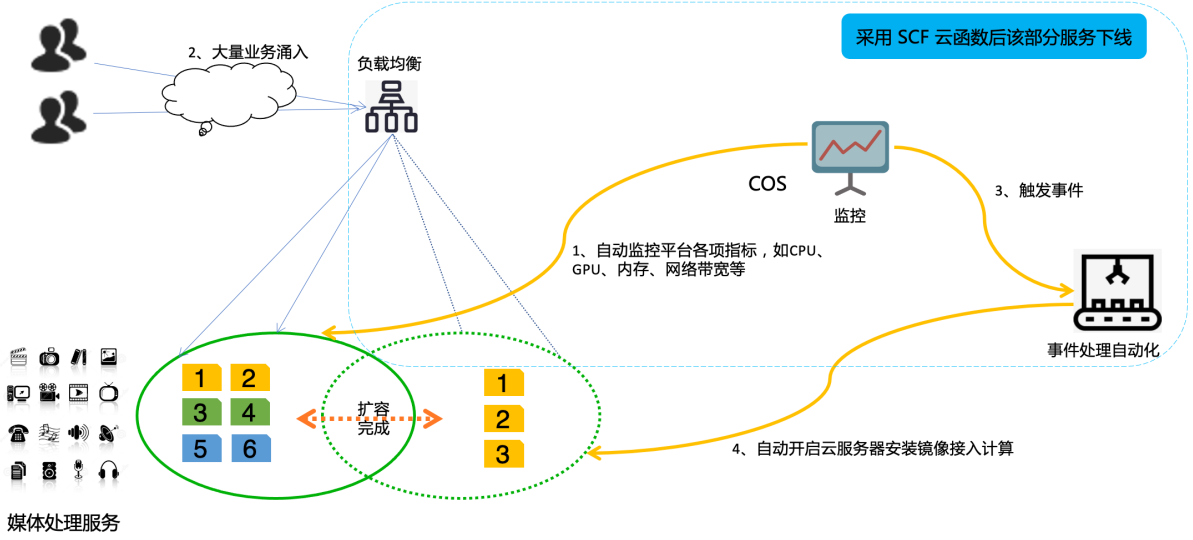
核心优势

1. **高效整合**：凭借云函数(SCF)的强大联动能力，将视频上传、视频处理、图片处理、存储场景有机地整合为一体。
2. **灵活处理**：支持自定义转码函数，快速搭建定制化任务处理能力，弥补当前单独云服务的功能盲点。
3. **平滑迁移**：将自研编码器及分布式转码系统从物理机、云主机移植到云函数。
4. **成本低廉**：云函数可选择实例规格，按需付费，成本较低。

✓ 在云函数中，采用go语言，撰写自定义业务逻辑，实现UGC转码业务：
步骤一：创建函数，部署自研编码器资源包、部署转码逻辑。
步骤二：配置 COS Bucket 触发器，对源视频实时处理加工；旁路生成日志和监控、支持告警。
步骤三：对转码后的视频回传COS，并分发到自建CDN或腾讯CND节点。

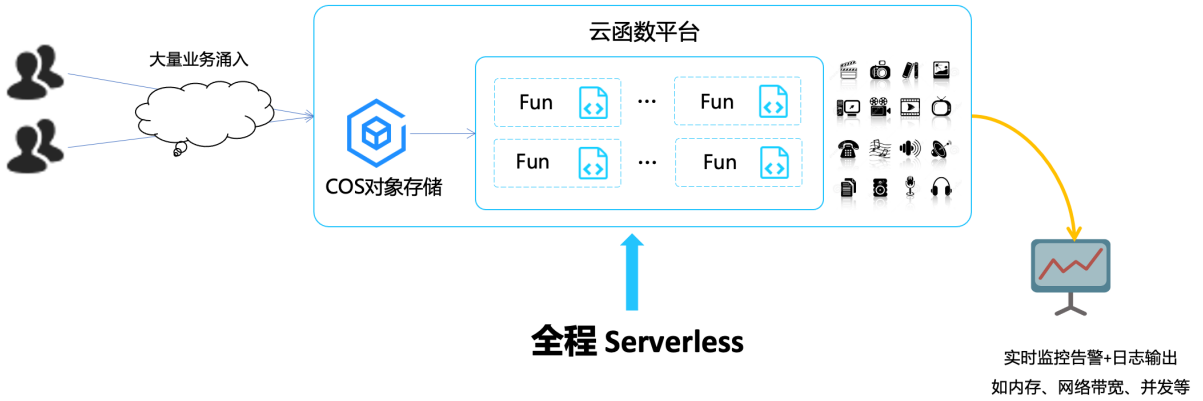
原有的音视频转码架构，需要自己监控各项指标，例如 CPU 、内存、网络带宽等。通过触发器去调动云服务器的安装镜像去接入计算平台，还存在一个延时比较高的问题。如下图所示：

原有音视频转码架构



在启动后无法动态回收资源，造成大量计算资源的浪费。采用云函数 Serverless 后，可以依赖云函数自动扩容的方式去应对大量用户请求，同时使用腾讯云实时监控功能，实时观察内存、并发，网络带宽的使用情况。如下图所示：

架构升级：自动弹性伸缩——应对自如



业务迁移腾讯云 Serverless 云函数之后，可以实现：

- 高易用性，只需实现业务代码的逻辑，无需关心非功能开发以外的问题，免运维。
- 稳定性，腾讯云 Serverless 通过多地容灾等方式提供高稳定性保障。
- 快迭代，支持版本、API 流量自由分配，快速实现灰度方案。
- 快启动能力，每次在一秒钟之内实现，20秒以内完成，满足业务需求。

腾讯互娱国际（IEGG）

最近更新时间：2024-08-28 16:44:51

客户介绍

腾讯互动娱乐事业群（IEG）旗下的腾讯游戏主打游戏开发和运营、网络游戏社区的机构。在游戏上云的道路上，腾讯互娱一直在不断探索、不断突破。2021年03月，腾讯互娱针对国际业务推出了在线游戏开发平台 PGOS，PGOS 提供：

- **后端服务平台：**PGOS（Proxima Game Online Service）是一种游戏在线服务解决方案，旨在降低游戏后端开发和维护难度，同时降低成本，从而使开发者专注于游戏玩法与核心逻辑开发。
- **全面托管服务：**借助完整的后端解决方案，消除了大规模构建，管理和运行服务器的挑战。即时自动扩缩容的专用服务器，为实时游戏提供低延迟和高可靠性。
- **一站式控制面板：**开发者及运维人员可以在 PGOS Web 门户上检索玩家信息及查看日志流水，监控实时数据并编辑相关服务配置。
- **跨平台 SDK：**提供开箱即用的 C++ SDK 和 UE4 插件，方便开发者在其游戏客户端和专有服务器中使用 PGOS 服务。
- **灵活的匹配规则：**为玩家提供在实时战斗中快速准确地与其他玩家进行匹配的能力。
- **扩展性和灵活性：**整个系统采用了微服务架构模式，而非具有不同层级的整体结构，使得开发人员可以添加和修改相应服务之间的交互。

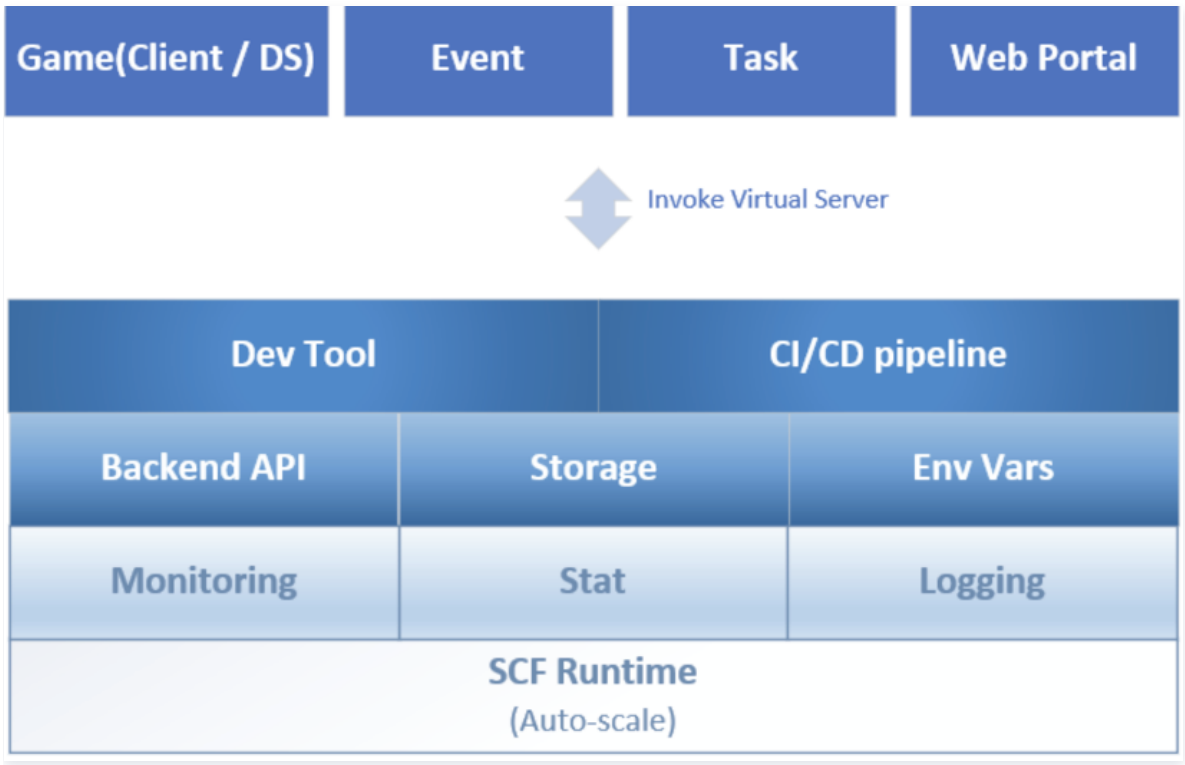
Serverless 解决方案

腾讯云 Serverless 天然支持上述 PGOS 提供的能力特性，PGOS 选择使用腾讯云 Serverless 技术提供底层计算支持，能够更好助力团队快速上云。

- **开箱即用：**用户无需额外购买、搭建和配置服务器，可完全专注于业务代码。这种架构方式不仅加快游戏发行和迭代速度，同时可大大降低运维成本，用户无需关注底层资源，腾讯云 Serverless 保障业务的稳定、安全和资源的可用。
- **动态扩缩容：**Serverless 的另一大特点是自动扩缩，轻松应对流量洪峰。在访问量突增时，自动扩容保障业务的正常运行。在流量低谷，自动缩容以节约成本。
- **实时监控：**腾讯云 Serverless 提供实时日志、监控面板，研发人员、管理人员可以实时监控业务运行状态，并且对接腾讯云可观测平台，提供运行时间、状态异常等多维度告警能力，使得问题可以在最短时间内被捕捉并且通知到用户。
- **扩展性和灵活性：**FaaS 的原子特性，天然支持业务灵活扩展。不同的云函数可支持独立的功能，既可支持函数间的相互调用又可独立更新和部署。同时支持函数代码在线编辑功能，从业务开发到部署再到监控，腾讯云 Serverless 提供了一站式的解决方案。
- **多种事件触发：**腾讯云 Serverless 支持约10种事件触发方式，包括定时触发器、API 网关触发器、对象存储触发器等等，满足用户多种触发场景的需求。

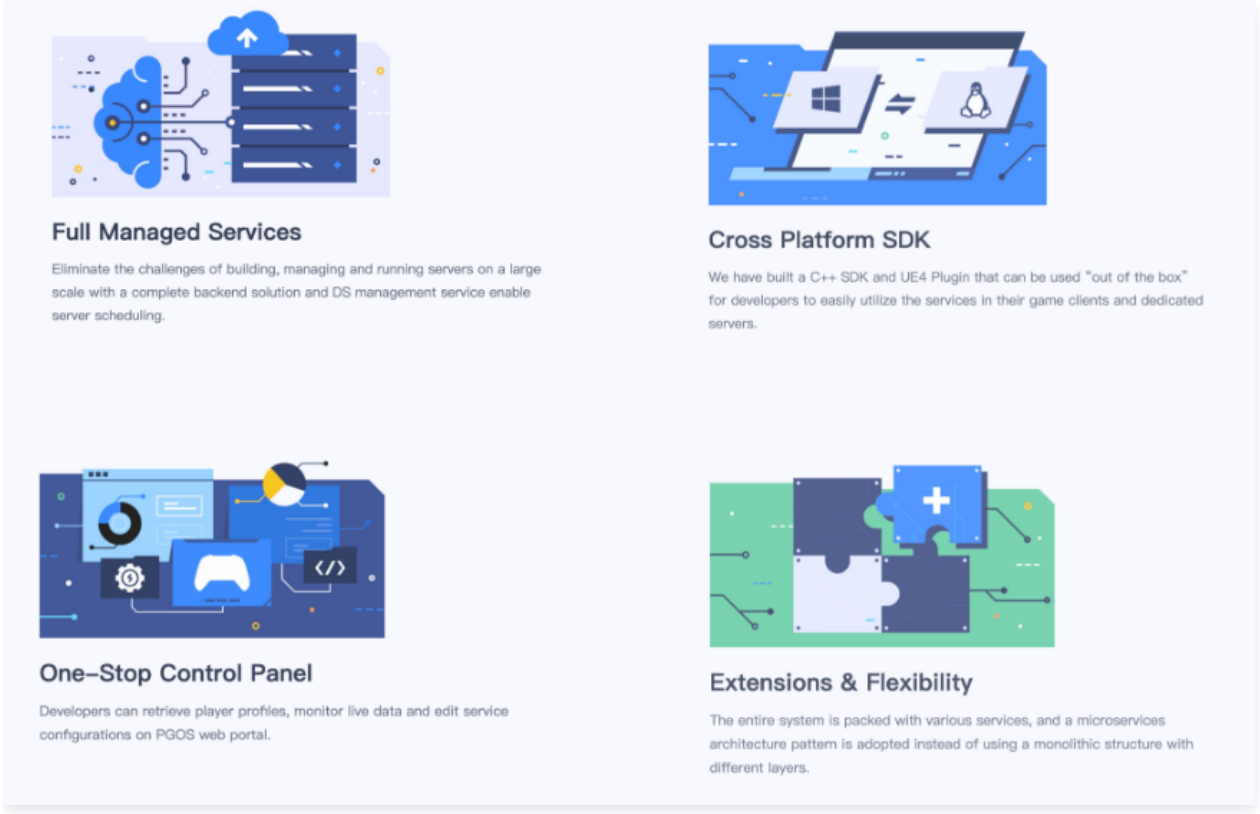
Serverless 为游戏上云提供算力支持的技术原理

腾讯云 Serverless 可以为国际业务 PGOS 提供底层运算支持，一个虚拟服务器（Virtual Server）对应一个或多个云函数，用户创建 Virtual Server 并编写对应的业务逻辑。PGOS 依赖 Serverless 提供完善的监控、日志能力，并对接后端服务，PGOS 更进一步封装 DevOps 工具，为用户提供全托管、自动构建和部署的功能。



PGOS 提供多种驱动方式，底层对应不同的函数触发器来实现触发 Virtual Server 中业务的运行。

- **定时器驱动**：游戏可以在 Web Portal 上配置一个定时任务，定时触发 Virtual Server 的指定接口。
- **事件驱动**：Virtual Server 可以监听特定事件，当事件发生时自动触发 Virtual Server。
- **游戏驱动**：Game Client 或 DS 可以主动调用 Extension Interface，通过 Gateway 触发 Virtual Server 的指定接口。
- **手动驱动**：通过 Web Portal 手动运行/触发 Virtual Server 的指定接口。



微保 WeSure

最近更新时间：2024-11-28 15:20:43

本文将具体介绍微保前端的架构演进过程，以及最终选择使用腾讯云 Serverless 技术支撑前端架构的原因。

客户介绍

微医保 · 百万医疗险是腾讯控股的保险代理平台微保 WeSure，携手国内知名保险公司为用户提供的一项保险服务，不受疾病种类的限制，可以报销无论是因意外、或因疾病的治疗费用，保额最高达600万。

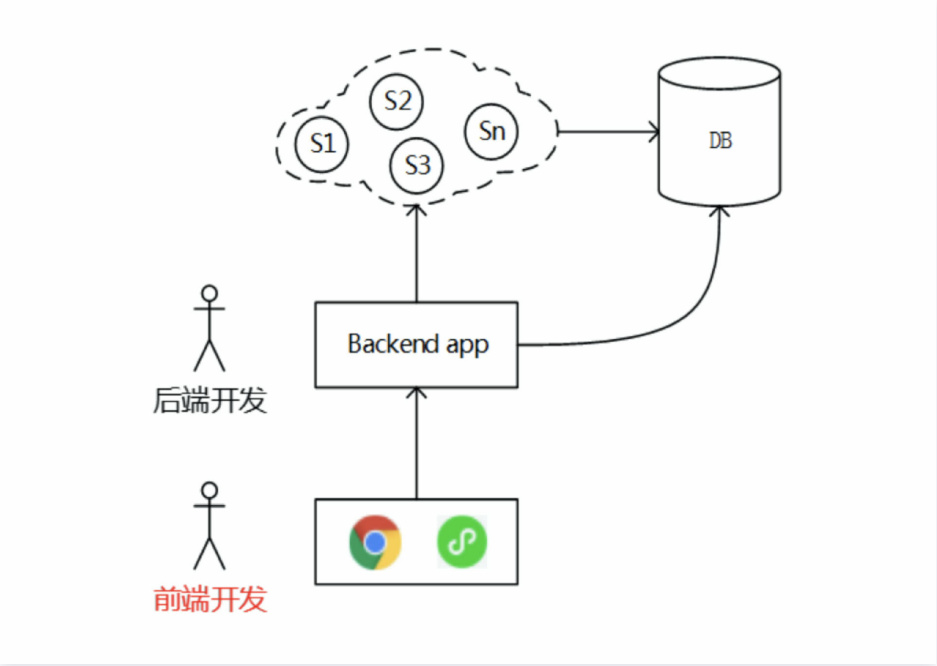
Serverless 解决方案

微保前端架构在业务发展中，根据业务、团队、开发等实际情况，不断进化调整，微保团队使用 Serverless 技术的主要应用场景：

- 前端开发同学，应用在 BFF 层，目前接入的有小程序，H5 页面。
- 数据组同学，面向风控和推荐算法应用，做计算使用。

微保架构 v1

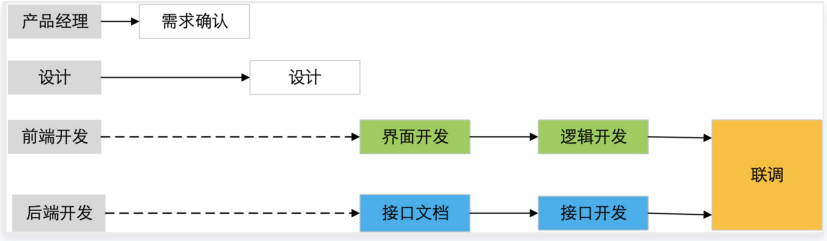
早期，微保团队使用经典的前后分离架构，前端开发与后端开发通过接口进行合作。合作流程如下图所示：



优势

前后端分离的架构有以下比较显著的优势：

- **前后端开发解耦**
  - 前端与后端开发并行，缩短需求整体开发周期。



  - 角色分工明确，线上问题定位与修复更加清晰。

    前后端分别设计与实现自己的错误监控和告警系统，前端对页面脚本错误进行捕获，上报至日志平台，经过日志处理工具，设置告警机制，将错误信息推送至相应的开发人员。
  - 利于前端组件化与后端微服务化架构。

前后端分离后，前端可以使用更为便捷的框架以及基于这些框架的基础 UI 组件，大大提升开发效率。另外，前端开发也会基于业务特点，提取业务专属的公共组件，所有组件化的沉淀，都是对生产效率的提升。

- **部署解耦**
  - 前端静态文件单独部署 CDN。

    前端项目中有大量的静态文件，包括 HTML、CSS、JS、图片、视频等，将这些文件部署在 CDN 上，充分利用现有云服务的 CDN 能力，既能提升资源访问的速度又能保证资源访问的稳定性，尤其是在高并发场景下。
  - 更加快捷的 CI/CD ，前端的编译过程可以非常简单地接入 CI/CD。

    前后端的统一部署相互依赖，分开部署后，可以针对前端项目以 Gitlab 的 repo 级别来做相应的 CI/CD。

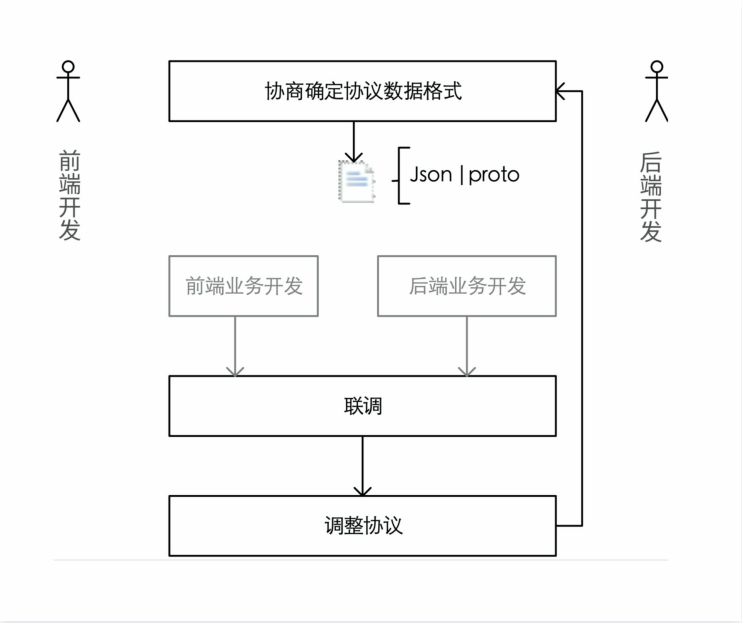
前后分离的架构对于业务早期的快速发展非常有效，而且在团队规模较小时，前后端开发人员合作固定，彼此对于对方的开发习惯、性格较熟，因此跨角色沟通的问题并不突出。但随着团队规模和业务规模持续扩大，这种架构模式将会慢慢给团队带来一些副作用。在实践中，遇到了几个以下比较明显的问题：

劣势

- **前后端协作耦合成为开发效率提升的瓶颈**

  团队规模与业务规模的扩大，意味着合作的人员变多、接口的复杂度也会相应增加。

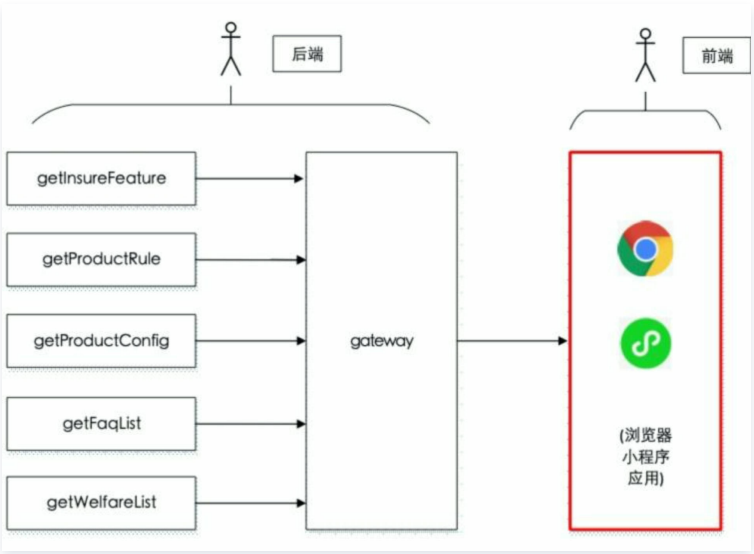
  早期专人专项，团队成员彼此熟悉其他人的开发习惯和一些业务，因此业务接口参数的调整沟通成本较低。但随着业务规模和团队成员扩充，在各种跨业务合作时往往会遇到一些同事不习惯阅读 proto 文档，或有些复杂业务需要花费大量时间阅读 proto 文档，或前后端反复沟通接口调用时参数的具体含义等问题。如下图所示：



- **页面渲染效率较差**

  以产品详情页为例，页面的渲染至少需要请求5个后端接口，再对数据进行组装和处理。不仅增加小程序端的代码体积，同时也降低了页面渲染速度。

  即使在前端页面对接口进行并行访问，但数据的整合逻辑依然会非常复杂。小程序作为微保主要的产品承载形态，代码量巨大，几近达到微信规定的代码上限，这种对于代码的增加随着业务增长也是一个隐形的风险。如下图所示：

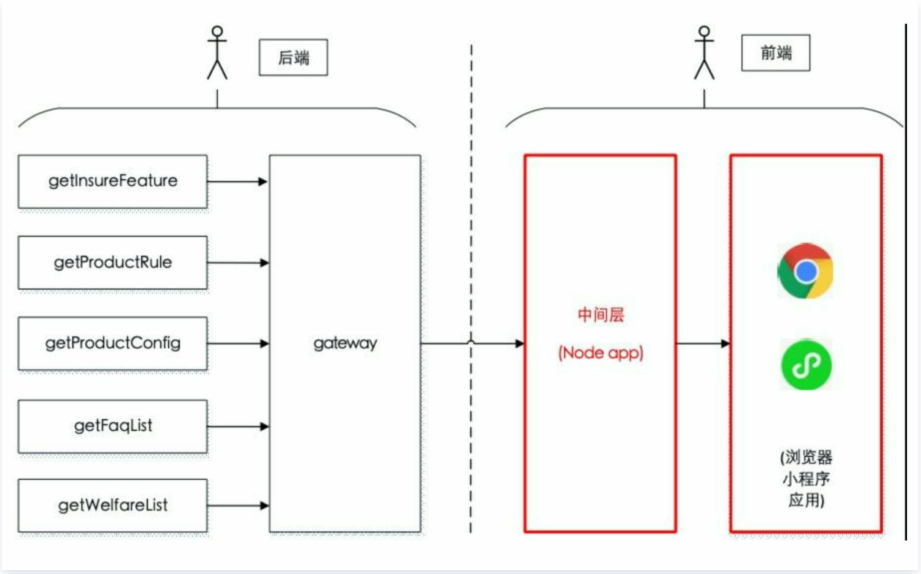


微保架构 v2

鉴于上述微保架构 v1 阶段前后端合作模式中的痛点，团队对架构再次进行优化，原则是业务“前”移、核心下沉。在前期的各种业务支撑中，团队积累了一些业务中台的经验，例如投保服务、续保服务、保单服务等。
中间层的引入让团队的开发效率进一步得到提升，前端对于业务的把控力及页面性能优化的操作空间也大大加强。不管是从团队的敏捷性、还是应用的体验，都有不错的改善，主要体现在以下几个方面：

优势

- **前后端流程上的耦合大大减小，角色责任的专一性逐步形成**
  基于一部分后端服务能力的积累，例如保单相关的需求，在需求评审及开发过程中，只需前端开发同学参加即可。前端开发同学与业务产品沟通业务逻辑，在 API 市场或服务文档查询相应的服务能力，完成业务开发。同时对于团队逐步开展业务中台化、前端组件化大有助益，整个架构对于丰富多变的业务需求的响应更敏捷。
- **渲染层对后端的服务进行聚合，减少页面请求**
  无论是 H5 网页还是小程序页面，均只需跟中间层进行数据交互，前端开发人员根据业务诉求，自行对接口进行聚合，端上只需1个请求就可开始渲染页面。接口聚合之前，产品详情页面的显示需要请求5个接口，平均的接口请求耗时在120ms左右，聚合后，通过中间层来请求5个内网接口，避免端与服务的多次连接耗时。



- **中间层对数据进行加工，大大减少小程序端的逻辑代码量**
  之前在小程序端的数据整合代码，有些复杂的逻辑，可以交给中间层处理，这些代码的节省对于业务持续增长时会越来越体现出价值。以年金产品详情页为例，数据在中间层聚合能够节省10KB左右的体积。

中间层的引入是对生产力的进一步解放，但基于一个巨型 App 的 Node 中间层，在后期运维中也暴露出一些问题。中间层的应用部署在2台云服务器 CVM 上，有其先天的一些不足，主要体现在以下几个方面：

劣势

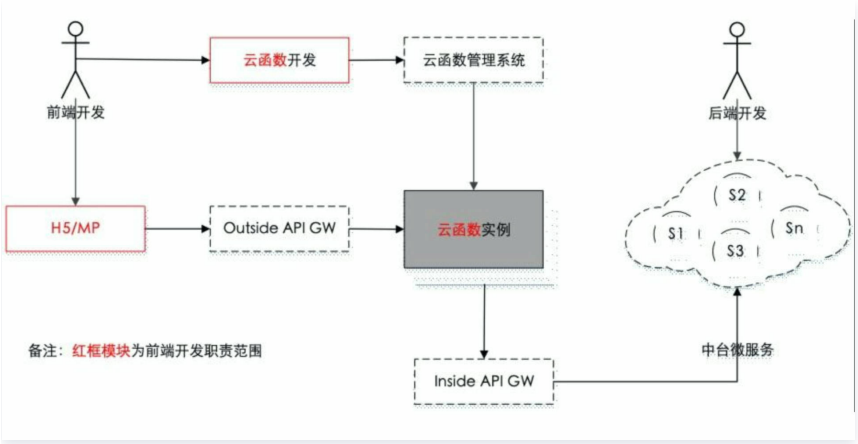
- **应对尖峰流量的冲击能力差**
  微保经常会有一些运营和投放需求，这些事件会导致瞬间的大流量进入，而云服务器 CVM 的扩容速度相对滞后。
- **App 级别的部署与发布**
  中间层不断积累业务代码，整个应用线性增长，每次部署与发布都是巨石应用的发布，部署效率低、风险高。
- **使用门槛提高**
  前端开发人员在开发、测试环境中需要自己在机器上查阅日志和服务操作，提高了普及的门槛。

微保架构 v3

基于上述微保架构 v2 阶段的一些限制，微保开始关注新技术，加州大学伯克利分校计算机科学 Riselab 团队的实验室研究室提出 Serverless 是云计算的下一个浪潮。国内各大厂商也都开始布局 Serverless ，腾讯云 Serverless 团队是国内较早在这方面进行部署的团队，技术已经非常成熟，在数百家企业已有成功落地实践。
微保通过调研了解到腾讯云 Serverless 云函数的优势：

- 强大的扩缩容能力，特别适合应对流量洪峰，且性能稳定。
- 每个函数单独运行、单独部署、单独伸缩，用户上传代码后即可自动部署，提升了独立开发和迭代的速度。
- 云函数提供精细的日志记录，可方便地查看函数的运行状况，并对代码进行调试、测试和审计。支持相关的监控指标上报，能快速了解函数的整体运行概况，也可自定义云函数的监控指标。
- 精确到1ms计费规则，只对正在运行的函数计费。

综上，基本解决了微保架构 v2 中面临的问题。经过团队整体评估，微保决定使用腾讯云 Serverless 云函数进行架构的进一步调整。调整后的角色合作流程示意图，如下图所示：

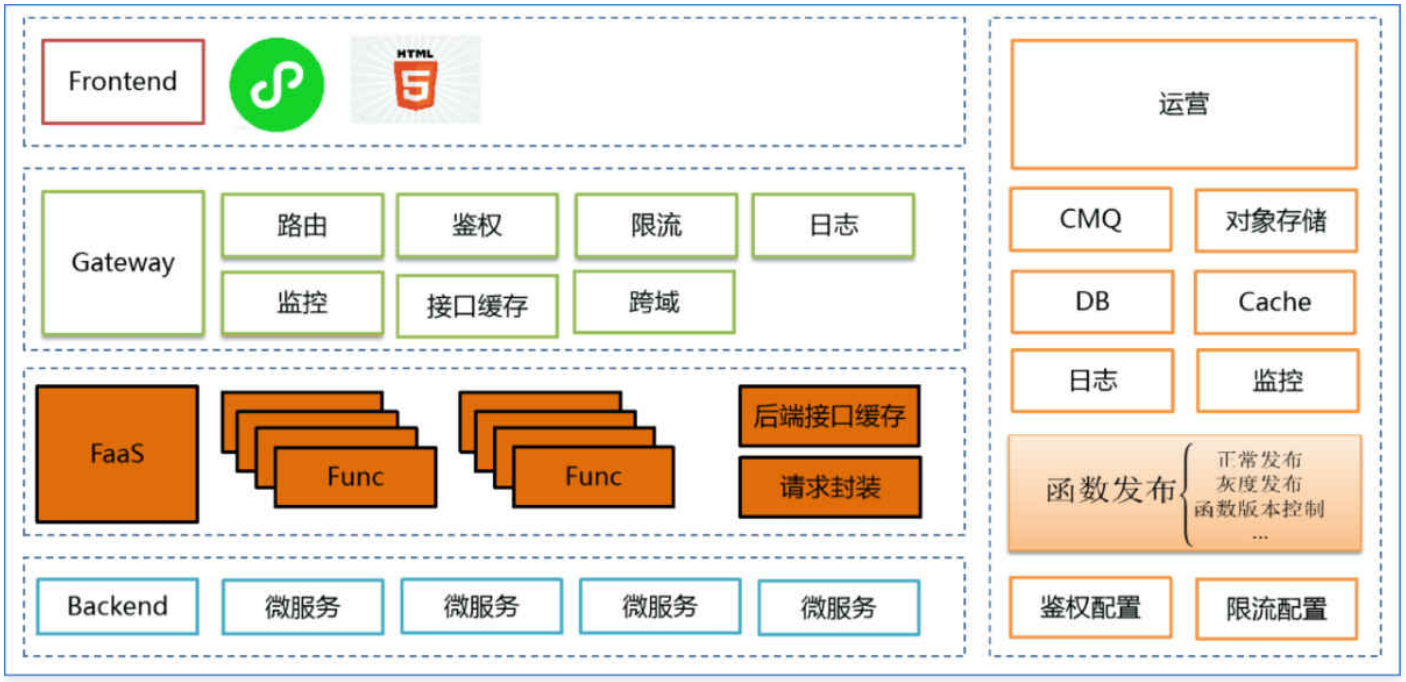


C 端的请求发至云函数 API 网关，网关转发请求至相应的云函数实例，云函数再向后请求服务的网关。整个链条上最大的变化是将云函数取代了 Node App，成为中间层的技术形态。
使用云函数替换掉 Node App，背后考虑有以下几方面，也基本是针对 Node App 实践中遇到的一些问题去加以解决：

- **自动扩缩容**
  开发者无需专门去配置，云函数可以根据请求量在函数层级水平扩展，正常情况下，一个空的云函数（运行时间50ms），300个并发，压测可以达到6000+的 QPS，基本能够应对日常的高并发需求。
- **函数级别的开发与部署**
  一个云函数对应一个 Gitlab 的项目，函数开发与发布都是围绕单个项目进行 CI/CD，高效、安全。
- **按需收费**
  对于金融模式下的流量特点，大部分情况下请求量较少，云函数的使用可以避免稳定的资源投入，空闲情况下费用大大减少。
- **简单的运维管理**
  开发者无需在服务器上自己维护服务和查阅日志，通过云函数的配套工具轻松管理函数、查阅日志，也可以根据自己的诉求设置告警机制。

Serverless 优化业务架构

微保每一次架构的调整，都致力于让各种研发角色的职责更为单一、内聚，角色间更加解耦。但这种调整也需要有配套的工具，其中的 trade-off 需要根据短期成本和长期利益来衡量。腾讯云 Serverless 云函数很好的支持了本次架构的重新调整。如下图所示：


Frontend
HTML
Gateway
路由
鉴权
限流
日志
监控
接口缓存
跨域
FaaS
Func
Func
后端接口缓存
请求封装
Backend
微服务
微服务
微服务
微服务
运营
CMQ
对象存储
DB
Cache
日志
监控
函数发布
正常发布
灰度发布
函数版本控制
...
鉴权配置
限流配置

腾讯相册

最近更新时间：2020-01-02 11:07:30

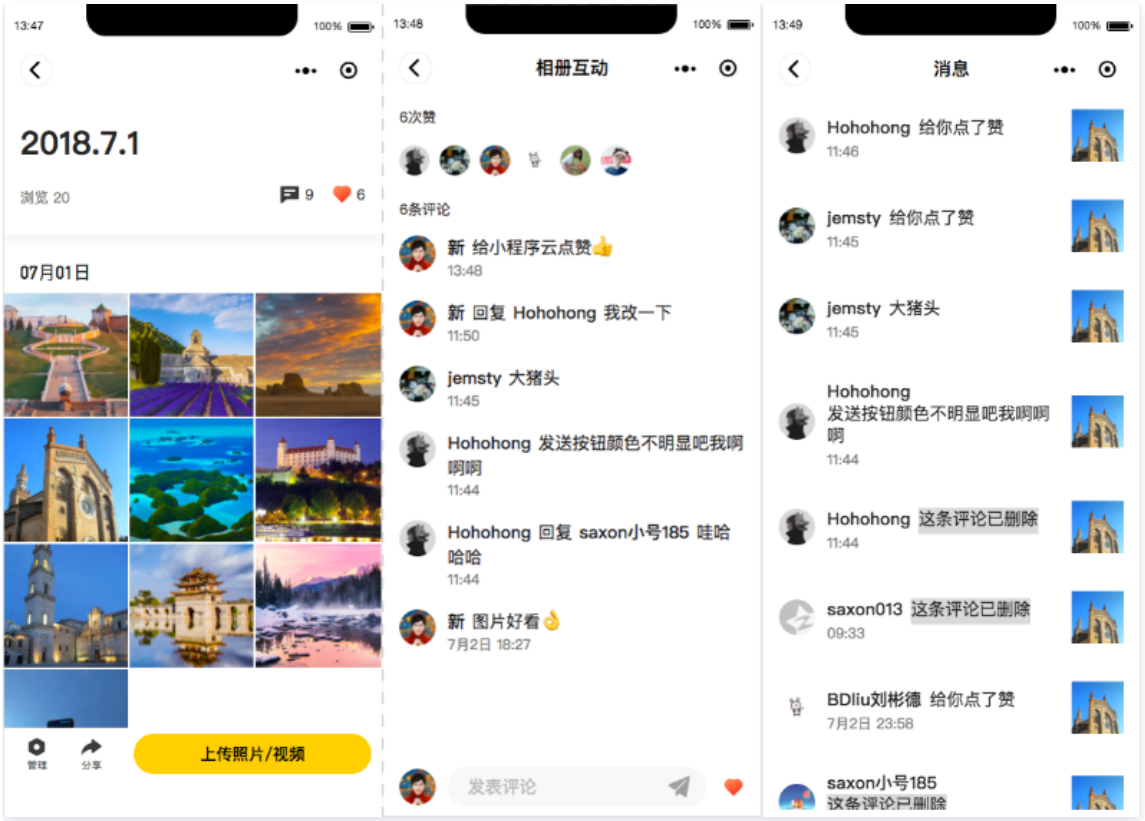
客户简介

腾讯相册小程序，是腾讯官方的相册小程序，为用户免费保存原图和视频，且支持长视频发朋友圈，并为用户一键生成更优雅的音乐相册或主题相册。

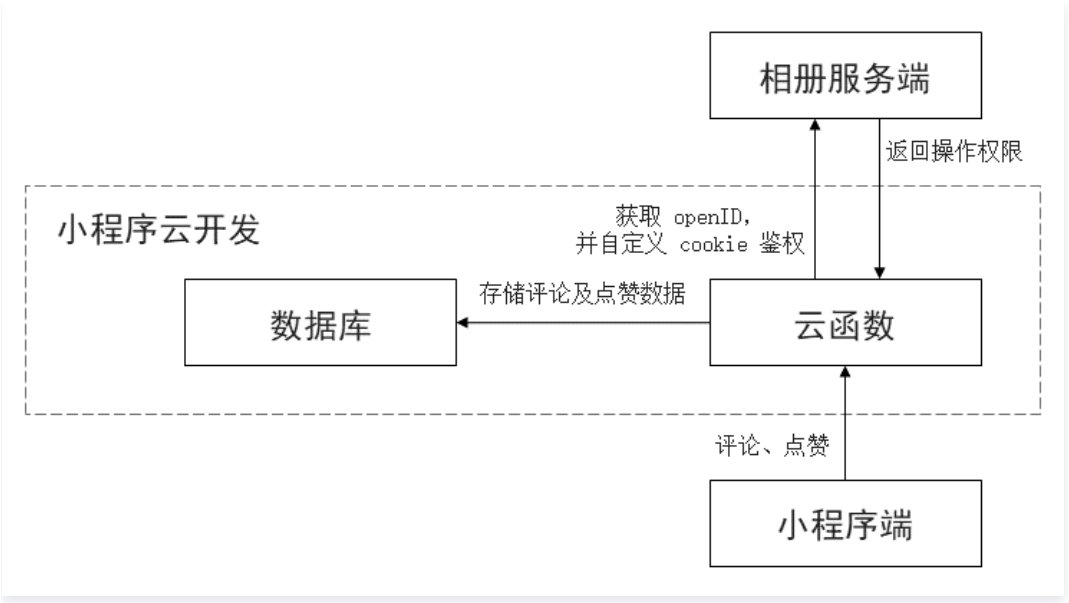
客户方案

腾讯相册通过小程序和空间相册打通，实现了在小程序端的照片上传、下载、分享好友、点赞、评论、生成小程序码等功能。本文档以点赞评论功能和优化小程序分享二维码为例。

“基于小程序云开发的点赞评论功能” 案例



基于小程序云开发的点赞评论功能的操作逻辑如下图所示：

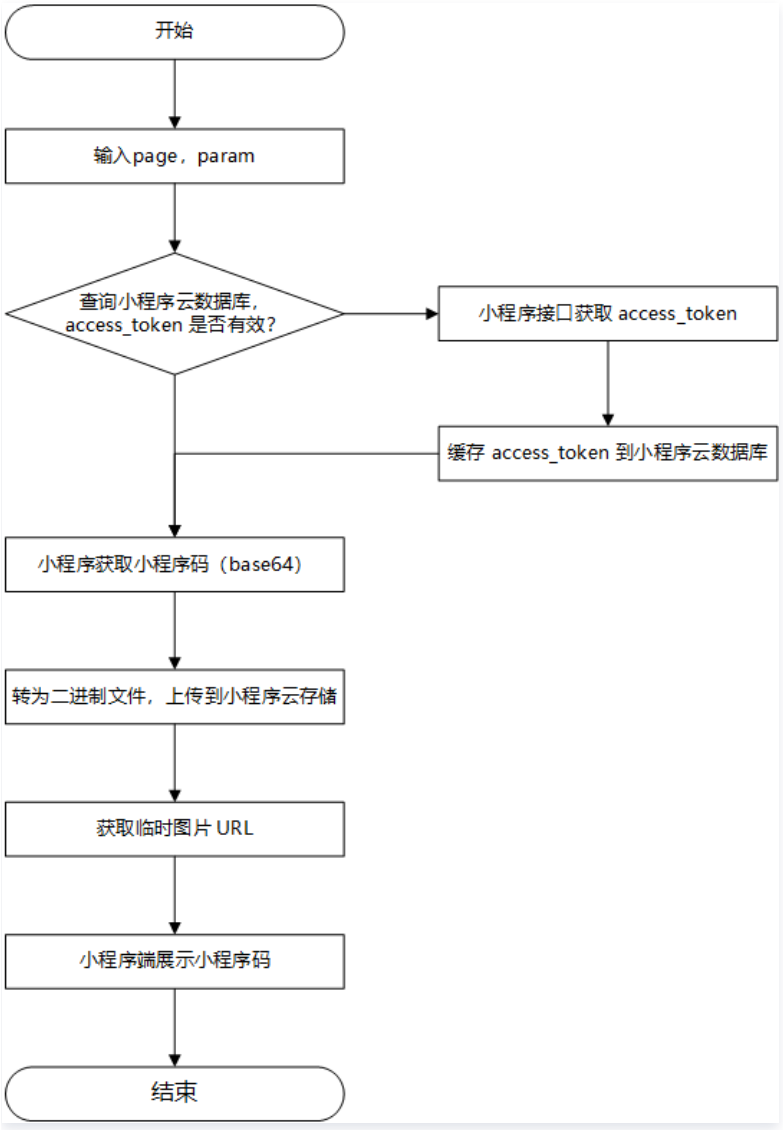


在云函数中，实现以下功能：

- 对小程序端评论、点赞等数据进行拉取和存储的操作。
- 当小程序端进行评论、点赞等操作时，通过云函数的路由功能，在原有的相册服务端获取用户的鉴权信息。

“优化小程序分享二维码” 案例

优化小程序分享二维码的云函数处理逻辑流程如下图所示：



在云函数中，实现以下功能：

- **调用微信的生成小程序码的接口。**
- **将需要生成二维码的图片或视频存储至云开发的文件存储。**
- **获取图片的临时 URL 并返回前端。**

客户价值

- 新开发的小程序后端与原有的后端服务互不冲突。
- 原有的后端服务架构复杂，新增功能困难。
- 使用小程序云开发可节省排期和联调的时间，提升开发效率。
- 在云函数中完成鉴权，即可打通原有的数据资源。

腾讯在线教育

最近更新时间：2023-05-22 16:57:53

本文分享了腾讯在线教育使用云函数的真实案例。腾讯在线教育团队是：

- IMWeb 团队隶属腾讯公司，是国内领先的专业前端团队之一。
- 专注前端领域多年，负责过 QQ 资料、QQ 注册、QQ 群等亿级业务。
- 目前聚焦于在线教育领域，精心打磨腾讯课堂、腾讯企鹅辅导及 ABCmouse 三大产品。

技术方案的尝试

腾讯在线教育团队在传统的 Web 应用方向其实有众多技术方面的尝试，包括传统离线包、PWA 离线应用等，但每个技术栈都有其优点及缺点。目前团队技术方案的多维度对比如下图所示：

方案名称	首屏时间	开发难度	维护成本	迭代成本	依赖网络	依赖客户端	用户体验	SEO友好度	日志系统
异步渲染	1.2s ±	低	低	低	是	否	差	一般	Badjs / Sentry
离线包 (异步渲染)	900ms ±	低	低	低	否	是	好	不涉及	Badjs / Sentry
直出	700ms ±	中高	中高	中高	是	否	中	好	Monitor / 机器log

通过对比可发现 SSR 在首屏渲染以及 SEO 等方面都较为突出。基于此结果，团队较倾向于 SSR 技术。在选择 SSR 技术方案时，可从以下两个方面进行考虑：

SSR 应用的性能

我们重点关注以下性能优化问题：

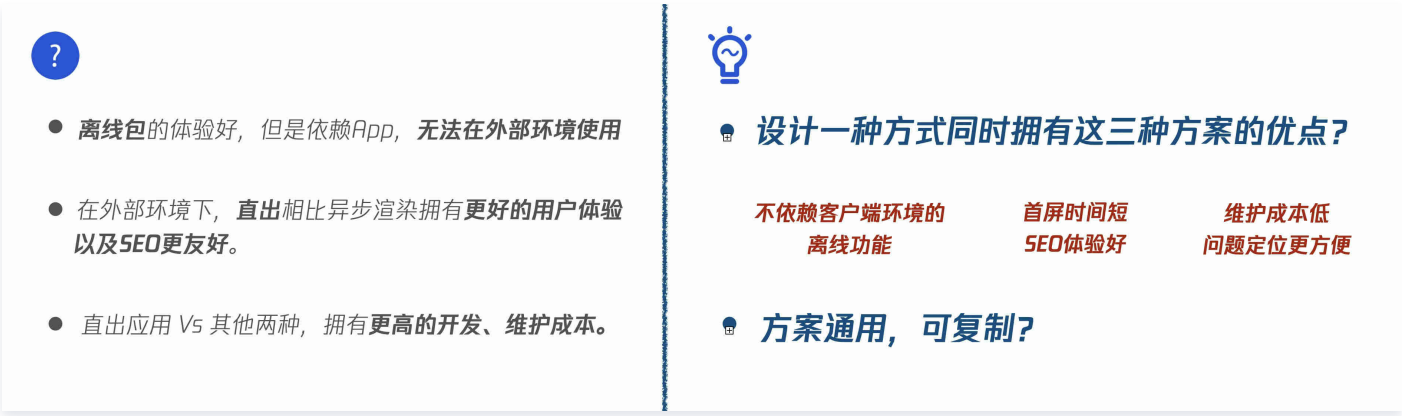
- **处理大量 CPU 密集型计算**：类 React 的应用的 SSR 本质为：在服务端调用 React 的 renderToString 方法，将 React 组件渲染为 HTML 字符串。对于复杂的 SSR 应用来说，此过程可能存在大量的 CPU 密集型计算，且这不属于 Node 擅长的领域。
- **提高首屏性能**：由于离线包的环境依赖性（依赖 App），在传统的 Web 环境内是否可以有一套完整的解决方案来缓存相关的页面，从而提高首屏的性能。

SSR 的运维成本

服务的可用性：大部分前端工程师可能不擅长服务运维方面的工作，服务的可用性问题也成为了技术选型时的重要关注点之一。
根据以上两个方面，可将考虑因素总结为下图中的两点：

- 如何设计一种方式同时拥有这三种方案的优点?
- 不依赖客户端环境的离线功能。
- 首屏时间短，SEO 体验好。
- 维护成本低，问题定位更方便。

- 方案是否通用且可复制?



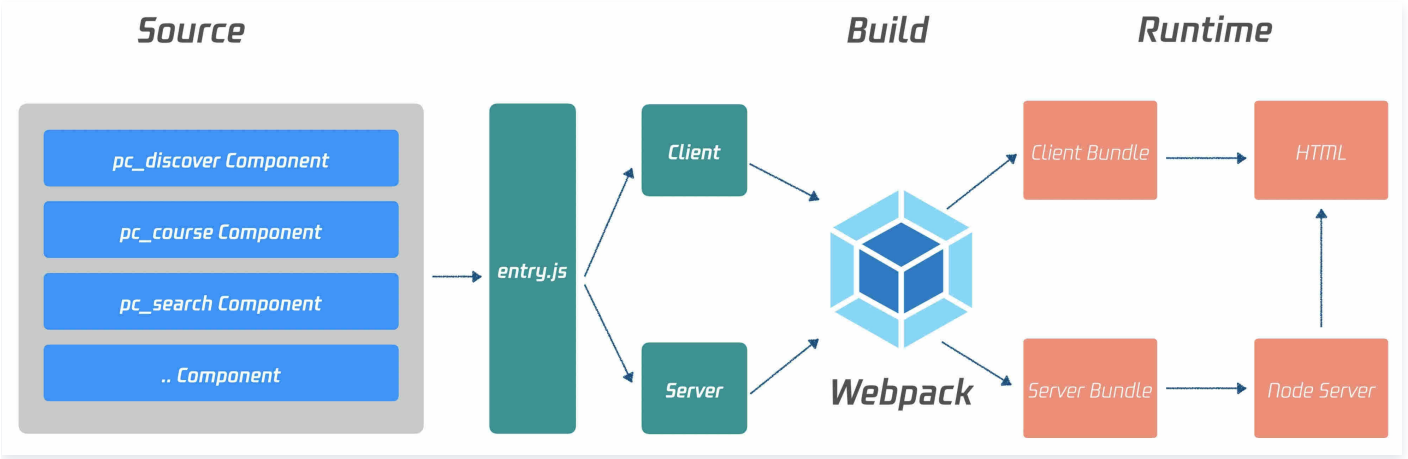
腾讯在线教育团队 SSR 架构方案介绍

团队使用的 SSR 技术架构图如下所示:

接下来我们从代码组织、性能优化、运行上下文三个方面来详细讲解团队现有方案。

代码组织

在 PC/H5 项目中均采用了同构的模式来构建 SSR 应用。如下图所示：



在同构的模式下，业务开发者更关注业务的功能本身，而不用太过关心运行时的问题，但同时也要注意以下几点：

- 传统浏览器中的常量使用，例如 `window` 、 `document` 等。
- HTTP 数据请求库必须同时支持服务端和客户端。
- 合理使用 React 应用的生命周期。
- 通过注入环境变量来区分当前运行时环境。

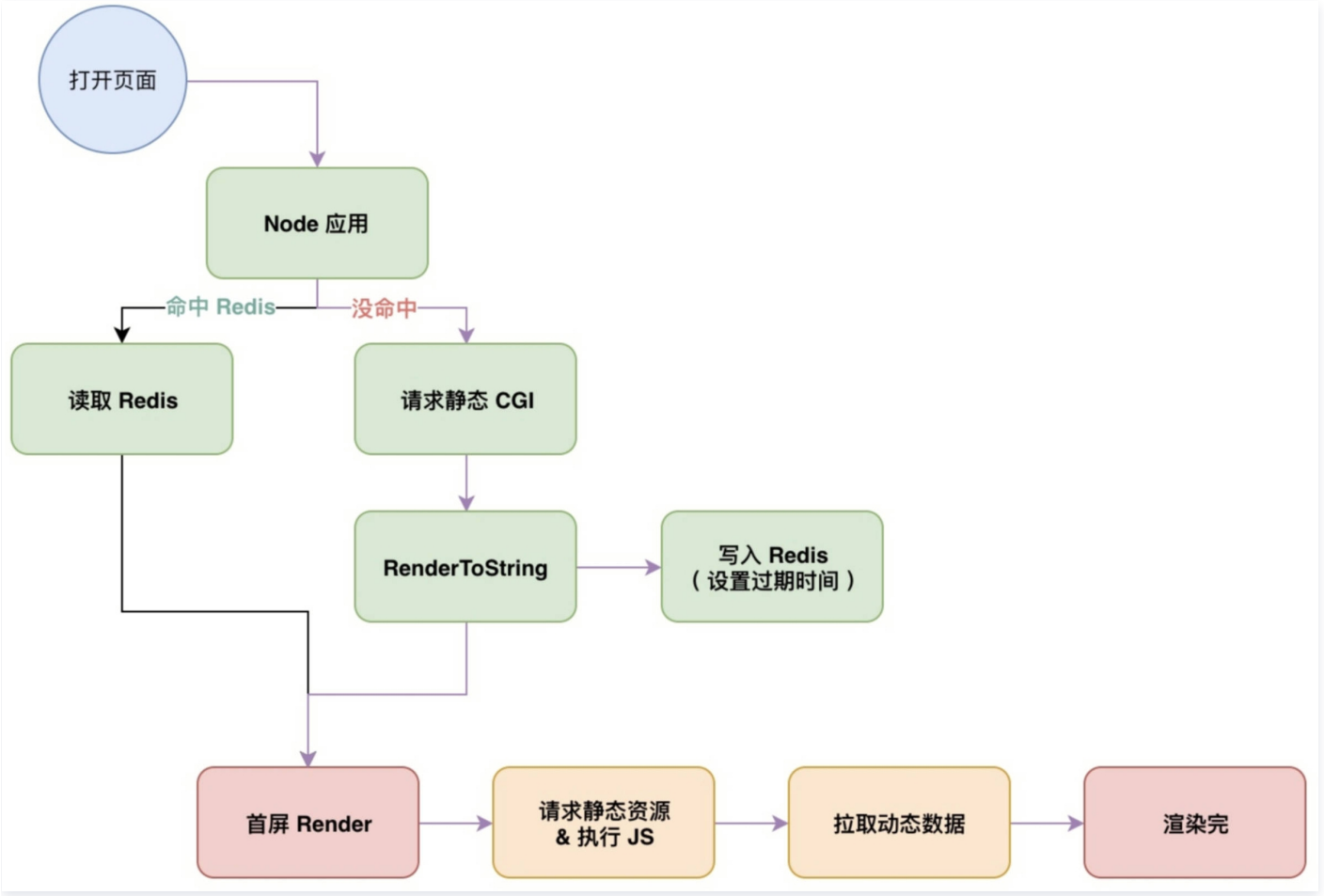
性能优化

接口动静分离

页面的渲染通常依赖后端的相关数据，而数据可以拆分为动态数据与静态数据两个部分：

- 静态数据：页面中不经常变更的数据。例如，企鹅辅导产品的课程标题、课程描述等。
- 动态数据：页面中与用户登录态相关的数据。例如，课程是否已经购买、当前课程的折扣等。

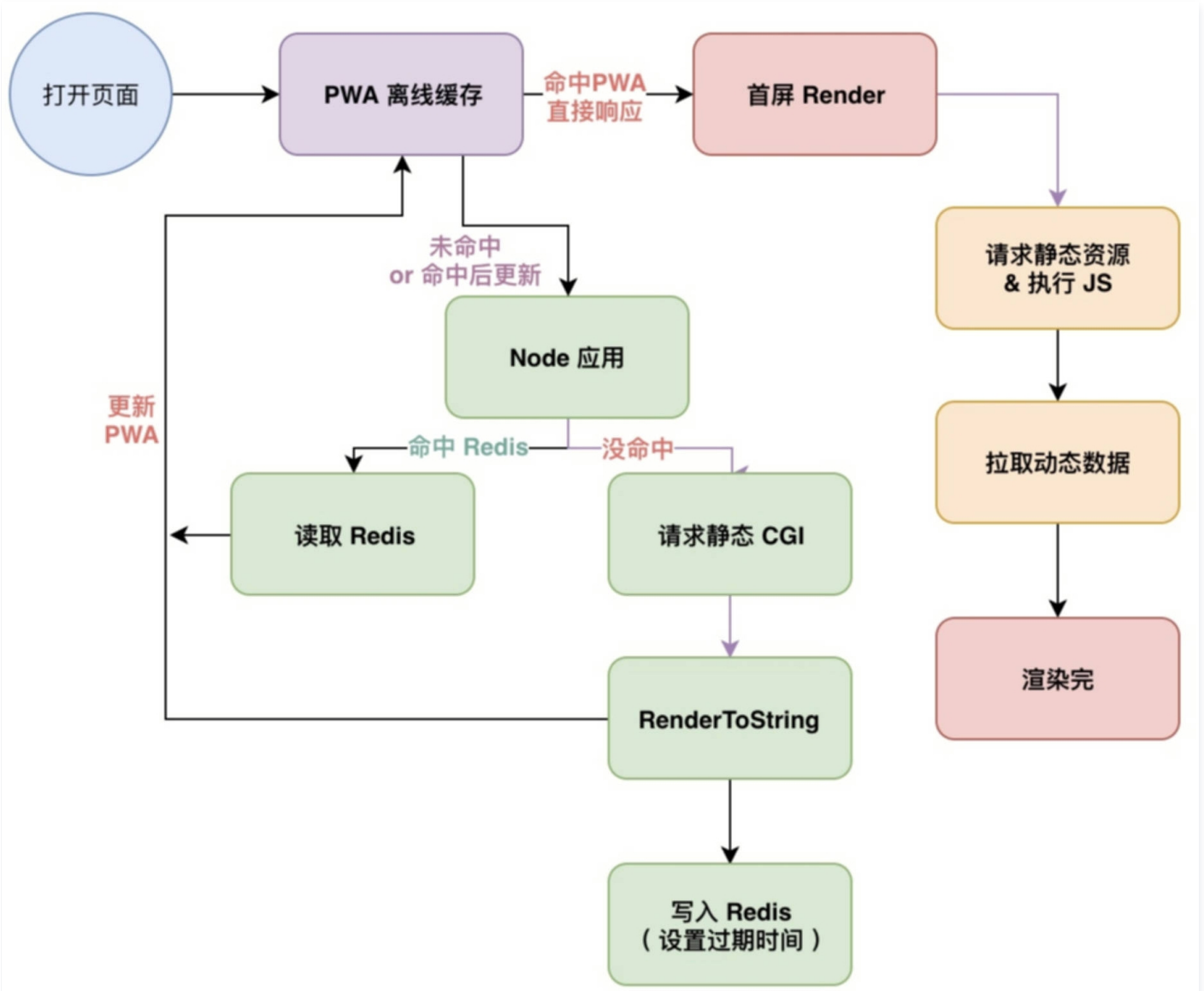
对接口做动静分离的意义在于，我们可以利用静态数据的时延性敏感度低的特性做缓存，在服务端利用静态数据渲染页面，之后在服务端利用动态数据做二次渲染。主要逻辑如下图所示：



我们利用 Redis 对静态数据渲染出来的页面做缓存，不仅可以加快 SSR 的渲染时间，同时可以提高单机的 QPS（`renderToString` 在一定意义上为 CPU 密集型操作）。

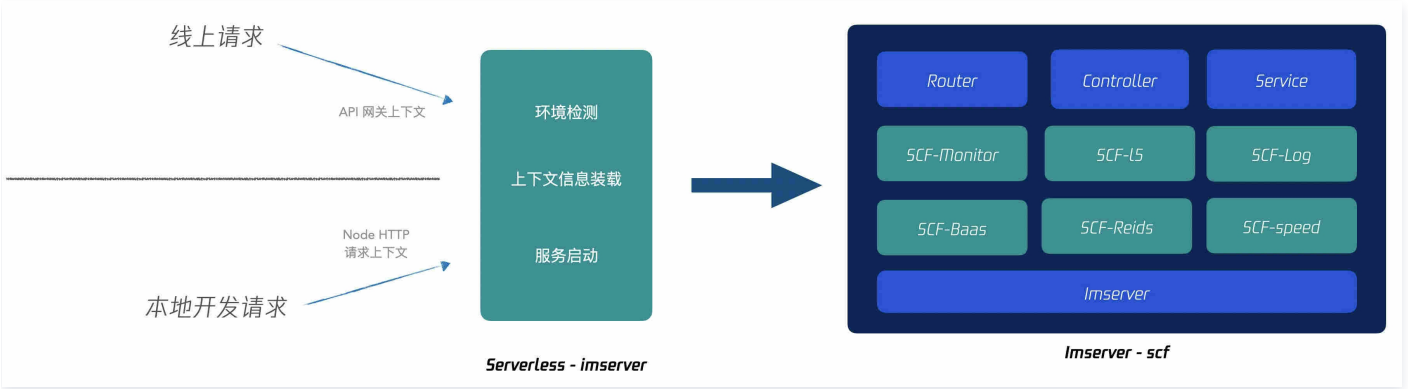
浏览器中利用 PWA 做离线缓存

在客户端中，我们可以利用 PWA 来做离线缓存，缓存静态数据直出的 HTML 页面，从而进一步的提高了直出页面的首屏性能。主要逻辑如下图所示：



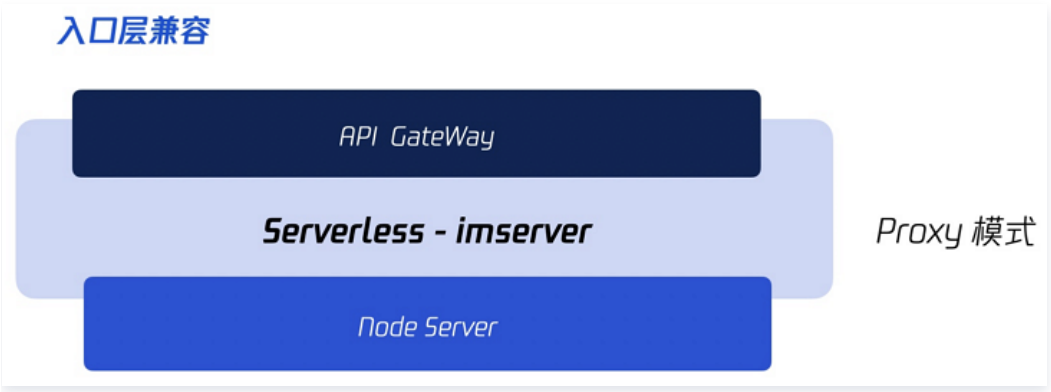
运行上下文

由于后端应用的运维复杂性、维护成本较高等问题，这里我们使用了 Serverless（腾讯云云函数 SCF）来做直出应用的部署。得益于 Serverless 架构模式的天然优势，不用再关心服务的运维、服务的扩容等问题。



如上图所示，SSR 的应用本质为一个 Node 应用，SCF 的调用本质为一个 Event 事件。针对此问题我们采用了如下方法兼容这两种模式：
借助腾讯云 Serverless Cloud Framework 提供的很多标准化接口，给自研 Node 框架（imserver）增加了一层 Serverless 的封装。同时还在入口增加

了 Event 到 Koa Request Context 的兼容。如下图所示：



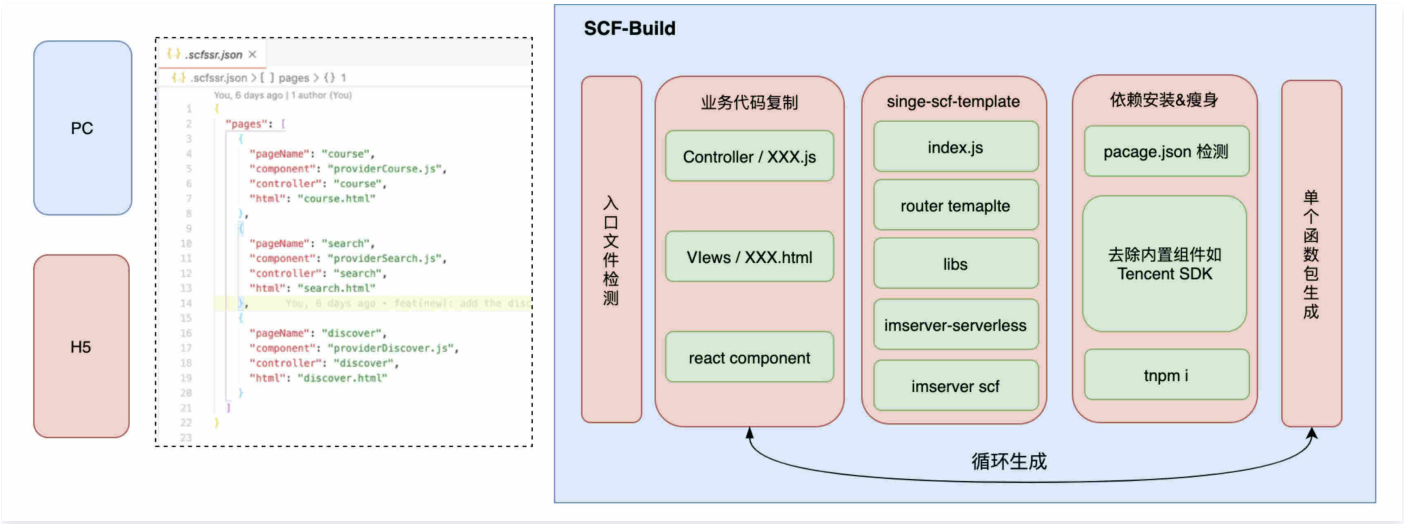
SSR 的技术方案落地过程中的问题

问题1：云函数拆分

业务中有多个页面是通过 SSR 实现的，在采用了 SCF 实现 SSR 后，需确定是合并至一个云函数中（业务级），还是拆分为多个云函数（页面级）。推荐选择拆分为多个云函数（页面级），优点如下：

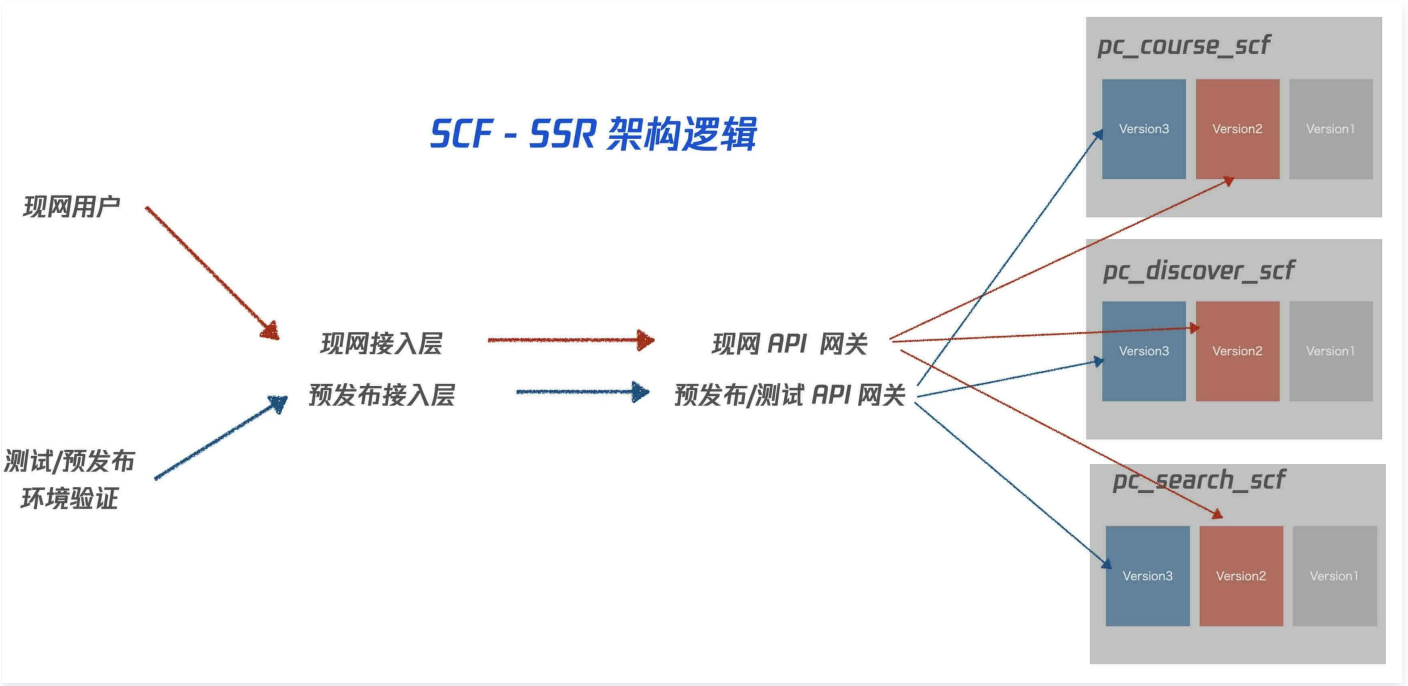
- 云函数互相独立。假设页面 A 云函数调用失败，并不会影响到业务 B 的云函数。
- 云函数包的大小会降低。由于云函数的冷启动过程，代码的包的大小对函数的冷启动时间也有一定的影响。

此处基于当前项目实现了云函数的自动化构建，通过 `.scfssr.json` 的配置文件自动生成相应的云函数，对现有的开发工作无任何影响，仅在构建时生成多个云函数，降低了应用的维护成本及开发成本。
同时基于云函数的构建过程，可使单个云函数代码更简练。通过分析 `package.json` 中的依赖，移除云函数容器中已经内置的工具包，并对云函数所依赖的第三方包做相应的引入分析，去除冗余。如下图所示：



问题2：云函数发布优化

下图为我们设计的基于 SCF 的多云函数直出方案逻辑，可查看当有版本更新时，发布流程及步骤是较为复杂的。



配置过程：初始化进行一次

在函数中创建 release、prohub 别名，可预先指向 `$LATEST` 版本。

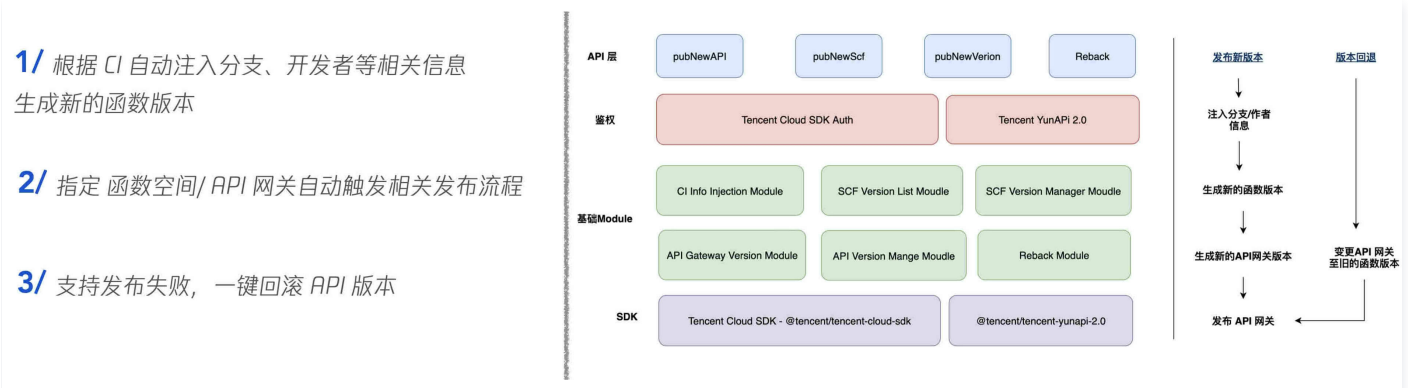
- API 网关中创建服务 A，配置 API 网关指向函数 B release 别名，并发布到 API 服务的 release stage 中。
- 修改 API 网关，指向函数 B prehub 别名，并发布到 API 网关服务的 prehub stage 中。
- 修改 API 网关，指向函数 B 的默认流量，并发布到 API 网关服务的 dev stage 中。
  至此 API 网关的配置完成，后续无需在 API 网关上再次修改及发布配置。

开发测试发布过程：持续开发测试发布上线

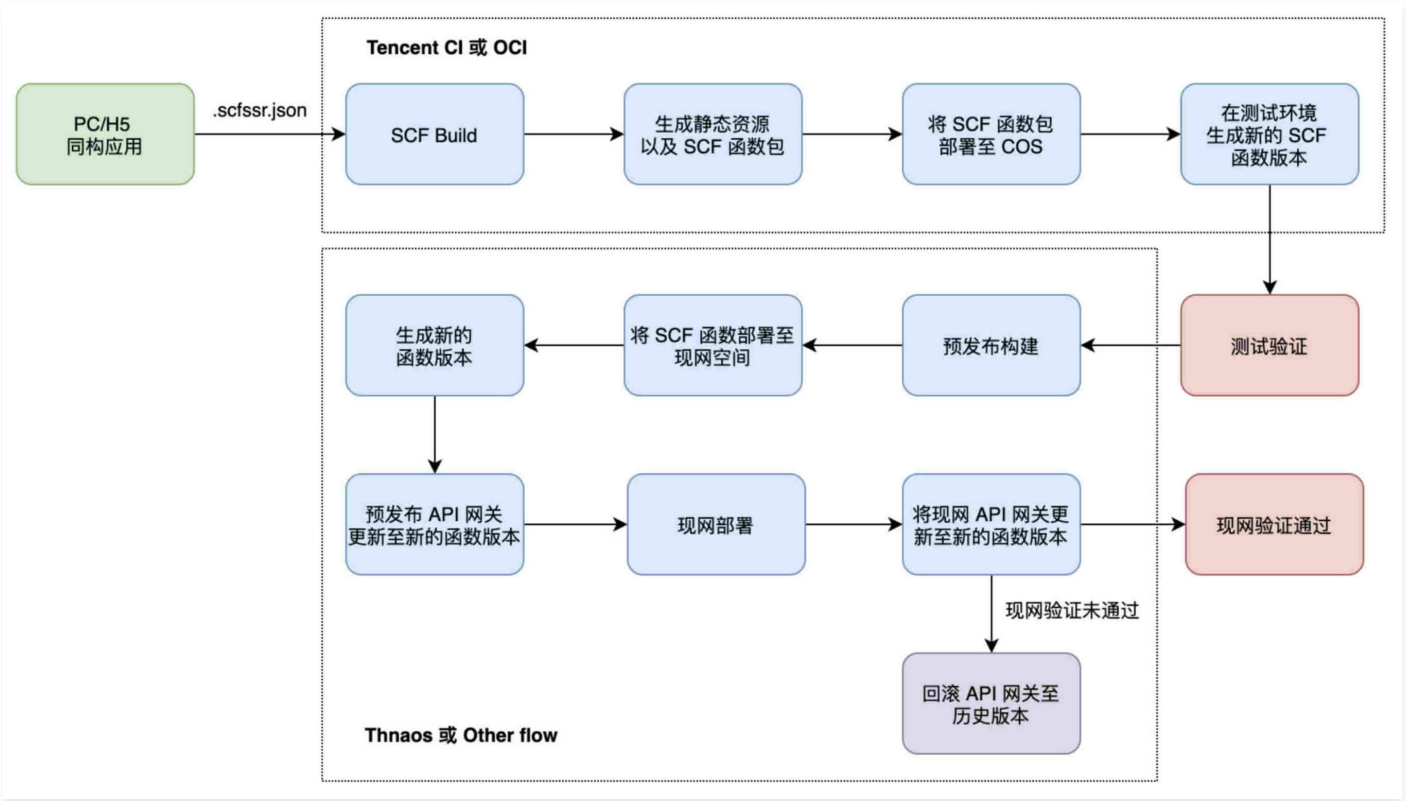
在函数上持续开发，一次发布版本 1、2、3。

- 需开发并测试最近版本时，配置 `$DEFAULT` 别名指向 `$LATEST` 版本，可基于此版本持续开发修改，修改完成后发布版本。
- 版本3可进入预发布环境时，配置 prehub 别名指向版本3，在预发布环境可进行测试和体验。
- 版本2已经在预发布层完成体验，可上线，将 release 别名灰度从版本1切换至版本2。
- 通过监控及日志查看灰度过程，版本2的流量是否正常上涨，版本1的流量是否正常下降，及发布过程中的各版本错误情况以及总体错误情况。

为优化云函数的发布流程，我们基于腾讯云 Serverless 所提供的 Node SDK 做了一键发布 SCF 的工具。如下图所示：

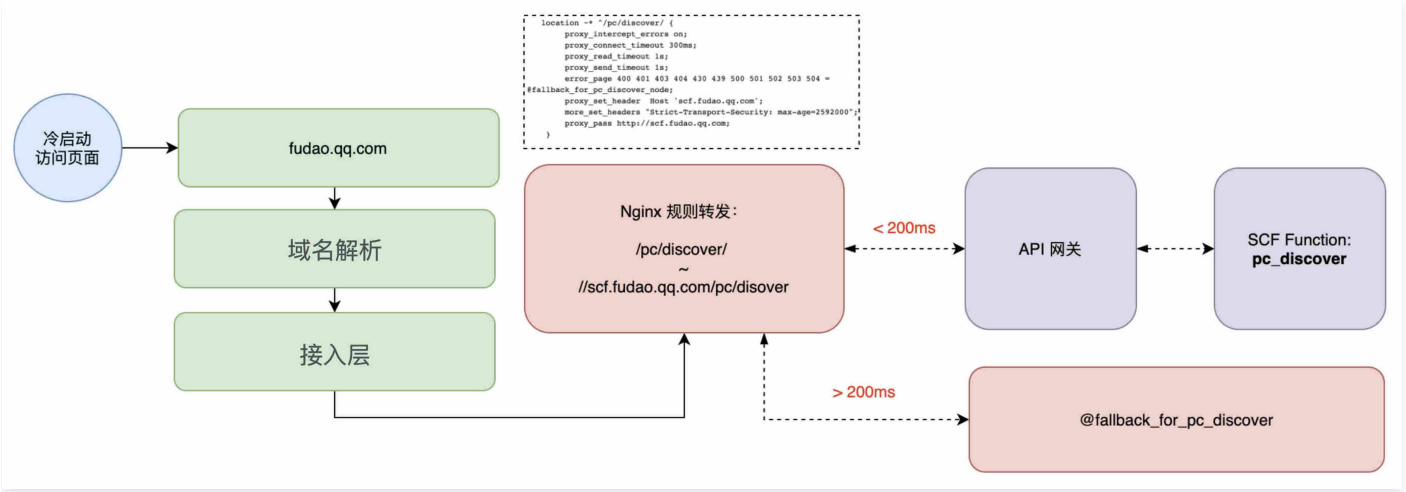


一个完整的 SCF SSR 应用生命周期如下图所示：



腾讯云 Serverless SSR 方案的优点及后续规划

使用基于 SCF 的 SSR 方案，节省了众多的服务运维成本。基于腾讯云 Serverless 的日志系统，所有的单个 SSR 应用请求在日志平台都有完整的链路，定位问题与处理问题的速度都有了质的提升。
Serverless 的架构模式存在冷启动时间较长的问题，SCF 针对此问题已经进行了技术优化，例如预启动容器等。我们在实际业务方面，也可尝试进行优化。例如，在接入层做了服务的降级优化。如下图所示：



后续的优化方案还可以从灰度、多维降级等方面来做改进。

使用 SSR 技术的建议

- 如果想追求更好的用户体验，建议针对核心业务做 SSR 优化，搭配 Serverless 来做服务的部署与运维。有了 Serverless 的支持，我们可以不用像以前一样关心机器的运维和扩容，可进一步提高团队生产力。
- 使用 SSR 后，建议可进一步完善自己业务的 devops 流程，将整个研发链路打通，从开发到测试再到部署都可高效进行。
- 推荐使用业务接入层来做服务降级，提高 SSR 应用的可用性。