无服务器架构技术白皮书

（2019 年）

云计算开源产业联盟

OpenSource Cloud Alliance for industry，OSCAR

2019年4月

版权声明

编写说明

牵头编写单位：中国信息通信研究院
参与编写单位：华为技术有限公司、阿里云计算有限公司、腾讯云计算（北京）有限公司、毕威拓科技（中国）有限公司、百度云、京东云、小米科技有限责任公司、国际商业机器（中国）有限公司、蚂蚁金服、西云数据、灵雀云、中兴通讯股份有限公司、时速云

编写组成员：

中国信息通信研究院：栗蔚、陈屹力、刘如明、闫丹、郑立
华为技术有限公司：俞岳、张红、高巍
阿里云计算有限公司：朱松、傅海雯、穆寰
腾讯云计算（北京）有限公司：黄文俊，卢萌凯，罗茂政，张浩，肖雨浓
毕威拓科技（中国）有限公司:刘鹏
百度云：张乐添、陈寿送、周岳骞
京东云：张金柱、高幸
小米科技有限责任公司：王小锋
国际商业机器（中国）有限公司：郭迎春
蚂蚁金服：李克鹏、俞仁杰、朱佳敏、董一韬
西云数据：高斌
灵雀云：陈凯、邢海涛
中兴通讯股份有限公司：余鹏
时速云：涂家英、魏巍、王金秀

前 言

十几年前云计算技术诞生，掀起了物理主机托管的基础设施变革风潮，云计算实现了计算资源与物理硬件的解耦，虚拟化技术的发展运用，使得云主机替代了物理主机，基础设施及服务（IaaS）开始广泛使用。随着容器技术普及，PaaS 平台逐渐兴起， IaaS 的运维工作持续下沉，PaaS 平台将开发人员与运维人员进一步的分离，开发人员能够更加专注于计算与存储资源的分配与使用。尽管 PaaS 平台已经广泛应用，但仍有优化空间，是否能有一种全新的架构，能将业务与基础设施完全剥离？无服务器架构（Serverless）应运而生，Serverless 将应用与基础设施彻底分离，开发人员无需关心基础设施的运维工作，只需专注于应用逻辑的开发，仅在事件触发时才调用计算资源，真正做到了弹性伸缩与按需付费。

无服务器架构在我国仍在初级阶段，业界对无服务器架构的认知尚不清晰，项目实践与成功案例亟待普及。本书将给出无服务器架构的定义及其涵盖范围，简述无服务器架构的发展历程，剖析架构的优势与不足，介绍无服务器架构现有的技术生态体系及其体系内的技术对比，本书还将讨论无服务器架构在众多领域的应用前景。

目 录

一、 初识 Serverless

（一）无服务器架的产生背景

基础设施架构总是伴随软件架构演进。单体架构时代应用比较简单，应用的整体部署、业务的迭代更新，物理服务器的资源利用效率足以支撑业务的部署。随着业务的复杂程度飙升，功能模块复杂且庞大，单体架构严重阻塞了开发部署的效率，业务功能解耦，单独模块可并行开发部署的微服务架构逐渐流行开来，业务的精细化管理不可避免的推动着基础资源利用率的提升。虚拟化技术打通了物理资源的隔阂，减轻了用户管理基础架构的负担。容器/PaaS 平台则进一步抽象，提供了应用的依赖服务、运行环境和底层所需的计算资源。这使得应用的开发、部署和运维的整体效率再度提升。

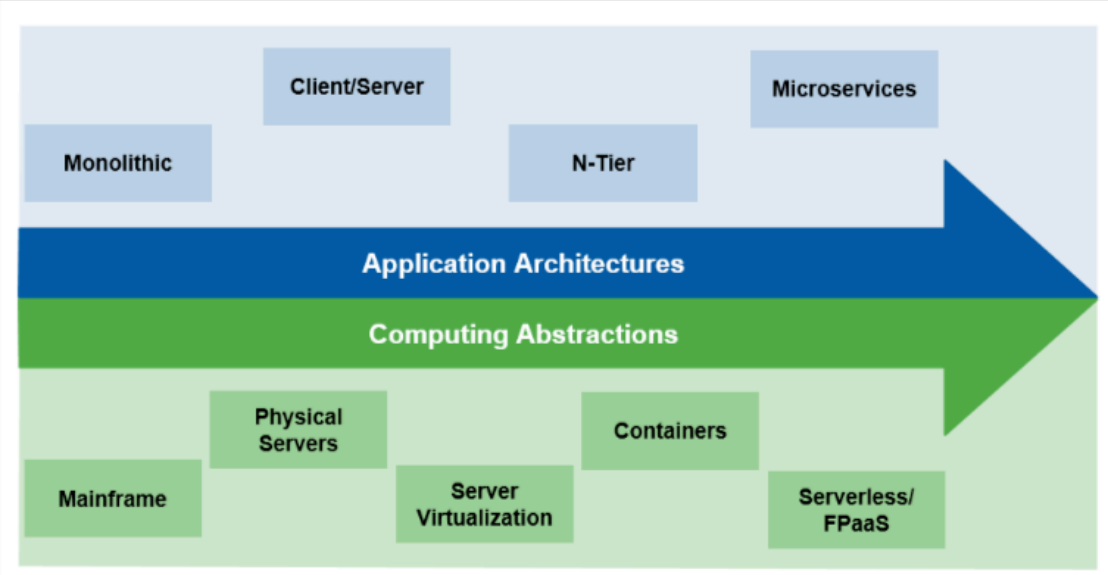

图 1：应用架构与计算抽象的演进示意图

能否有一种更加彻底的架构，能够将应用架构堆栈中的各类资源的管理全部委托给平台，免去基础设施的运维，使用户能够聚焦高价值的业务领域，进一步提高软件应用和运营的生产力？无服务器架构的思想应运而生。

（二）无服务器架构技术发展大事记

无服务器（Serverless）的概念最早要追溯到 2012 年，Ken Fromm 在《软件和应用的未来是 Serverless》中率先提出了无服务器的概念，但却并未引起广泛关注。2014 年 AWS 重磅发布函数计算产品 Lambda，开启了无服务器架构的新时代，这使得无服务器架构变得触手可及并逐步流行开来，无服务器架构开始正式走向云计算的舞台。

随后两年国外云计算巨头谷歌、微软、IBM 等陆续推出无服务架构的函数服务，2017 年起国内公有云厂商也开始加紧布局 Serverless，开源的无服务器架构框架也日渐丰富，2018 年谷歌开源 knative，尝试将无服务器架构标准化，Serverless 生态初具规模。无服务架构广受拥趸，已经悄然成为技术发展的风向标。

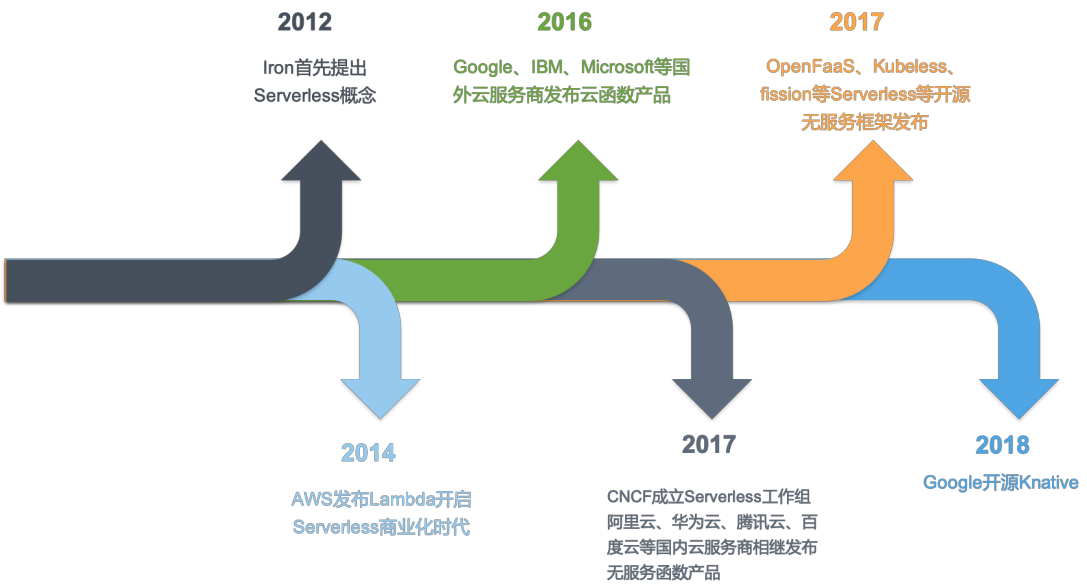

图 2：无服务器架构发展历程

（三）无服务器架构概念

无服务器是一种架构理念，其核心思想是将提供服务资源的基础设施抽象成各种服务，以 API 接口的方式供给用户按需调用，真正做到按需伸缩、按使用收费。这种架构体系结构消除了对传统的海量持续在线服务器组件的需求，降低了开发和运维的复杂性，降低运营成本并缩短了业务系统的交付周期，使得用户能够专注在价值密度更高的业务逻辑的开发上。由于大量服务均由厂商负责维护，这也使得无服务器架构的厂商绑定现象较为严重。

目前业界较为公认的无服务器架构主要包含两个方面，即提供计算资源的函数服务平台 FaaS，以及提供托管云服务的后端服务 BaaS:

1） 函数即服务（Function as a Service）

函数即服务是一项基于事件驱动的函数托管计算服务。通过函数服务，开发者只需编写业务函数代码并设置运行的条件，无需配置和管理服务器等基础设施，函数代码运行在无状态的容器中，由事件触发且短暂易失，并完全由第三方管理，基础设施对应用开发者完全透明。函数以弹性、高可靠的方式运行，并且按实际执行资源计费，不执行不产生费用。

函数即服务带来了前所未有的开发体验。**开发交付更加敏捷**，开发人员只需编写应用程序逻辑，计算资源以服务形式提供，无需考虑资源容量与基础设施运维，进一步缩短了开发交付时间；**资源利用更加高效**，函数单元仅在触发时运行，处理完成后迅速释放，几乎没有闲置时间，资源利用率近乎百分之百。

但现阶段函数即服务的局限性也较为明显。**代码调试较为复杂**，FaaS平台的代码调试大多需要下载到本地,调试成功后上传至函数，在线调试工具功能尚不完善，调试的复杂度较高；**低延时业务暂不适用**，FaaS 中的代码通过事件触发，如果执行结束一段时间没有再次触发，执行函数的容器会销毁，再次启动会有启动的开销，增加启动延迟，所以目前不适用低延迟的业务，如金融交易等。

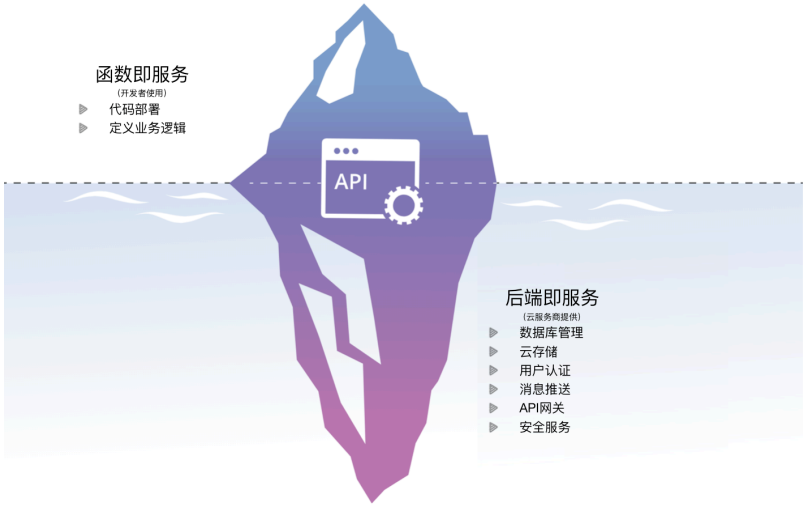

图 3: 无服务器架构的两个重要组成部分

2) 后端即服务（Backend as a Service）

BaaS 的概念涵盖范围较广，覆盖了应用有可能依赖的所有第三方服务，如云数据库、身份验证（如 Auth0、AWS Cognito）、对象存储等服务，开发人员通过 API 和由 BaaS 服务商提供的 SDK，能够集成所需的所有后端功能，而无需构建后端应用，更不必管理虚拟机或容器等基础设施，就能保证应用的正常运行。这在很大程度上省去了开发人员学习各种相关技术和中间件的成本，降低了开发的复杂度。BaaS服务大多有云服务商提供,用户无需关心和运维底层基础资源。目前常见的 BaaS 服务包括:

- 数据库管理
- 云存储
- 用户认证
- 推送通知
- 远程更新

- 消息队列

二、 Serverless 技术生态现状

目前我国 Serverless 的技术生态主要活跃在公有云的云函数服务领域，国内主要云服务商都具备云函数产品，与之匹配的 BaaS 服务也日臻完善，公有云服务基本可满足用户无服务器应用的搭建。私有云的解决方案领域仍旧以国外开源技术为主，Serverless 私有云解决方案提供商多处在产品研发阶段。工具层来看，独立 BaaS 服务（开源、商业产品）主要由国外服务商提供，目前尚不能很好的兼容国内公有云产品。国内服务商提供的相关工具主要供给各自产品使用，普适多云平台的工具产品多集中在开发框架层面。

无服务器架构产品的使用场景大多集中在各大云服务商的内部业务改造小范围试水，大规模的企业级应用尚属少数。中小企业基于无服务器架构的业务应用已初见端倪，未来可期。

图 4：中国无服务器架构技术生态图

文章下述将对 Serverless 技术生态中的每一环做简要介绍，并对其中的典型产品予以剖析:

（一）平台层

平台层提供全托管的运行环境，提供函数单元所需的计算环境并自行维护服务器资源、网络资源、消息分发和负载均衡等功能。是整个 Serverless 架构的基础。

国内主要公有云服务商均已推出云函数产品，开源的无服务器架构框架也层出不穷，下文将选取较为典型的几个平台进行介绍。

1. 公有云函数计算服务

◆ 华为云函数工作流

函数工作流（FunctionGraph）是华为云提供的一款无服务器计算服务，无服务器计算是一种托管服务，服务提供商会实时为用户分配充足的资源，而不需要预留专用的服务器或容量，真正按实际使用付费。华为无服务器计算包含函数和工作流两个功能模块，分别实现函数计算和函数编排的功能。

FunctionGraph 函数是一项基于事件驱动的函数托管计算服务。使用 FunctionGraph 函数，只需编写业务函数代码并设置运行的条件，无需配置和管理服务器等基础设施，函数以弹性、免运维、高可靠的

方式运行。此外，按函数实际执行资源计费，不执行不产生费用，具有以下特点:

- 提供多种 Runtime，适应各种编程需求。
- 支持多种事件源触发函数，满足复杂的业务需求。
- 支持函数访问公有云内其他服务或 Internet 上的服务
- 提供图像化函数指标页，实时监控函数运行情况。
- 提供详细的日志信息，方便排查函数错误。
- 提供多种使用场景的函数模板，实现模板代码、触发器、运行环境自动填充，快速构建应用程序。
- 提供函数初始化入口，分离初始化逻辑和请求处理逻辑。

FunctionGraph 工作流提供可视化的函数工作流编排，可以编排分布式应用函数和微服务组件。用户可以使用一系列能够单独执行的离散函数构建应用程序，并且能够快速扩展和更改，满足快速增长的、复杂的业务需求，具有以下特点:

- 提供图形控制台
- 使函数（程序）编排可视化，简化多步骤应用函数（程序）的构建和运行。
- 自动触发和跟踪各个步骤
- 使应用程序按照预期顺序执行，并在出现错误时重试。
- 记录每个步骤的状态
- 在出现错误时，能够迅速定位并调试问题。

- 简化应用程序的扩展和更改
- 无需编写代码就可以更改和添加步骤，可以轻松地完善应用程序。

◆ 阿里云函数计算

阿里云函数计算是事件驱动的全托管计算服务。使用函数计算，用户无需采购与管理服务器等基础设施，只需编写并上传代码。函数计算服务会准备好计算资源，弹性地可靠地运行任务，并提供日志查询、性能监控和报警等功能。借助函数计算，用户可以快速构建任何类型的应用和服务，并且只需为任务实际消耗的资源付费。函数计算以事件驱动的方式连接其他服务。借助这些方式，用户可以构建弹性的、可靠的以及安全的应用和服务，甚至在数天内就能完成一套多媒体数据处理后端服务。当事件源触发事件时，我们会自动调用关联的函数处理事件。例如，对象存储会自动触发函数处理新对象创建/删除事件，或者 API 网关在收到 HTTP 请求时自动触发函数处理请求。此外，函数还可以由日志服务或者表格存储等其他阿里云服务触发。函数计算工作流程如下图所示：

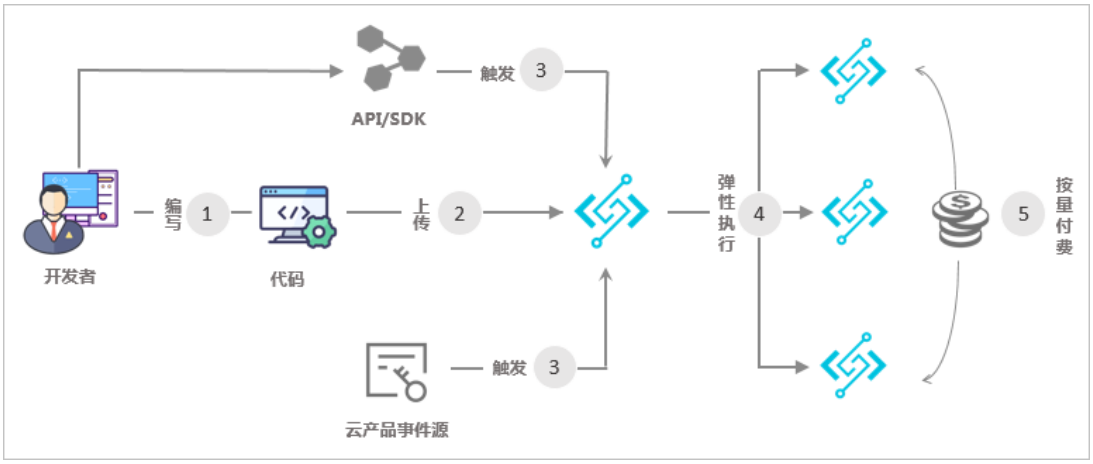

图 5: 阿里云函数计算工作流程示意图

编号说明

①. 开发者使用编程语言编写应用和服务。
②. 开发者上传应用到函数计算。上传途径包括函数计算控制台、API/SDK 以及命令行工具 fcli。
③. 触发函数执行。触发方式包括 OSS、API 网关、日志服务、表格存储以及函数计算 API/SDK 等。
④. 动态扩容以响应请求。函数计算可以根据用户请求量自动扩容，该过程对您和您的用户均透明无感知。
⑤. 根据函数的实际执行时间按量计费。函数执行结束后，可以通过账单来查看执行费用，收费粒度精确到 100 ms。

◆ 腾讯云函数计算

无服务器云函数（Serverless Cloud Function，SCF）是腾讯云为企业和开发者们提供的无服务器执行环境，帮助用户在无需购买和管理服务器的情况下运行代码。用户只需使用平台支持的语言编写核心代码并设置代码运行条件，即可在腾讯云基础设施上弹性、安全地运行代码。SCF 是实时文件处理和数据处理等场景下理想的计算平台。

使用云函数时，用户只需关注自己的代码。腾讯云完全管理底层计算资源，包括服务器 CPU、内存、网络和其他配置/资源维护、代码部署、弹性伸缩、负载均衡、安全升级、资源运行情况监控等，用户只需使用平台支持的语言提供代码。代码在执行时将根据请求负载扩缩容，无需人工配置和介入即可满足不同情景下服务的可用性和稳定性，从每天几个请求到每秒数千个请求，都由云函数底层自行伸缩。云函数自动地在地域内的多个可用区部署，提供极高的容错性。用户只需为运行中的云函数付费，代码未运行时不产生任何费用。

◆ 百度云函数计算

函数计算 CFC（Cloud Function Compute）提供基于事件触发、全托管的云端计算能力。由于其“无服务器”特性，用户仅需要开发业务代码并上传，无需关注和配置服务器资源，CFC 会托管用户的代码并在代码需要执行时自动分配计算资源，使用户业务高效、可靠地运行。

CFC 支持通过事件触发器连接到其他服务，当对应服务发出请求时，函数即会响应运行，做出相应处理和反馈。例如：通过 BOS 触发器将函数连接到百度云对象存储服务，设置函数响应 BOS 的新建对象事件。当用户的对象存储中增加新的文件时，函数即被触发运行，对新上传的文件进行诸如内容检查等操作。

利用函数计算，开发者完全无需创建和管理后端服务器，只要开发完业务代码即可快速启用服务，大大提高工作效率。函数计算支持多种编程语言和函数触发器，满足多样化的事件触发场景，同时提供弹性、高可用、扩展性好、极速响应的资源调度能力，保障用户业务的高性能运转。用户仅需为函数运行时实际占用的资源付费，当函数不被调用时不会产生任何费用，从而达到成本的最优化。

目前 CFC 的已经在智能设备、IoT、小程序、边缘计算等场景广泛投入使用，比如为百度人工智能操作系统 DuerOS 的开发者提供便捷的响应式技能开发平台。

◆ IBM 云函数

IBM云函数是IBM公有云平台上的无服务器计算平台，采用了Apache OpenWhisk的核心代码。用户可以通过交互界面或者命令行工具，将函数代码发送到云平台上去，配置好函数代码执行的事件源以及准备好代码处理的数据流，就可以安心的等待程序代码在合适的时候被执行了。IBM云函数则会管理基础架构，分配计算资源，调度容器运行，峰值自动扩展。费用则根据函数具体运行时间来计算，精确到毫秒级，不会将金钱浪费在空闲的服务器上。

IBM云函数与IBM公有云平台进行了很好的集成，可以导入公有云中应用的事件，例如数据库、消息队列、移动应用事件等，也可以在函数代码中方便地接入公有云上的服务，例如Watson上的认知服务、对象存储、API网关等 。

目前，IBM云函数广泛应用在Web应用开发、流媒体处理、物联网等多个领域。

函数平台	华为云	阿里云	腾讯云	百度云	Lambda
函数内存（MB）	128/256/512/768/1024/1280/1536/1792/2048/2560/3072/3584/4096	64*K (K=2,3,4…24)	64*K	128/256/384/512/640/768/896/1024	64*K (K=2,3,4…24)
CPU	与内存成比例	与内存成比例	与内存成比例	与内存成比例	与内存成比例
Runtime OS	EulerOS	Aliyun Linux	CentOS7	None	Amazon Linux
部署方式	在线编辑、zip包方式上传、OBS包上传	ZIP/Jar/OSS	Zip 包上传，对象存储文件引用	在线编辑、zip包方式上传	支持zip包上传
语言环境	Java、Python、Node.js、Go、C#等	Java、C、C#、Python、PHP、Go等	Python，Nodejs，PHP，JAVA，Go	Java、Python、Node.js	Java、C、C#、Python、PHP、Go等

环境依赖	无	每种语言不同的	语言相关	None	每种语言不同的
触发器 类型	华为云服务+APIGateway+定时器	11 款云服务+API Gateway	API 网关、消息队列 CMQ、对象存储、CKafka，定时器，腾讯叮当	BOS 服务、CDN 服务、消息服务、http 触发器、DuerOS 触发器、Duedge 触发器	AWS 服务+API Gateway
函数最长执行时间	900S	300S	300S	300S	300S
日志监测工具	华为云 APM	云监控	日志服务、云监控	None	CloudWatch Logs and X-Ray
计费特点	函数执行时间 分配内存大小	函数执行时间 分配内存大小	函数执行时间 分配内存大小	调用次数、函数执行时间*分配内存大小	函数执行时间 分配内存大小
默认最大并发	100	100	单函数 100	100	none

表 1: 国内主流公有云函数服务主要参数对比

2. 开源无服务器架构框架

◆ Knative

Knative 是基于 Kubernetes 平台用来构建、部署和管理现代无服务器架构的工作负载的框架，它将云原生应用开发的三个领域的最佳实践结合起来，即构建容器（和函数）、为工作负载提供服务（和动态扩展）以及事件。Knative 是由谷歌与 Pivotal、IBM、Cisco、Red Hat 等云原生技术厂商紧密协作开发的。

Knative 扩展了 Kubernetes，提供了一组中间件组件，它们对于构建现代、源码中心化以及基于容器的应用至关重要，这些应用可以运行在企业内部、云端或第三方数据中心中。

Kubernetes 已经成为容器编排的事实标准。但是 Kubernetes 的定位是一个容器平台而不是代码平台。作为运行和管理容器的平台 Kubernetes 功能强大，但是这些容器是如何构建、运行、扩展和路由很大程度上是由用户自己决定，而这也是 Knative 的主要发力点。

Knative 构建在 Kubernetes 的基础上，为构建和部署无服务器架构和基于事件驱动的应用程序提供了一致的标准模式。Knative 减少了这种全新的软件开发方法所产生的开销，同时还把路由和事件的复杂性抽象出来。

图 6：Knative 架构示意图

Knative 包含的关键组件：构建应用程序，为其提供流量服务，以及确保应用程序能够轻松地生产和消费事件。

构建：通过灵活的插件化的构建系统将用户源代码构建成容器。比如 Google 的 Kaniko，它无需运行 Docker daemon 就可以在 Kubernetes 集群上构建容器镜像。Knative 的构建工作都是被设计于在 Kubernetes 中进行，和整个 Kubernetes 生态结合更紧密。另外，它希望提供一个通用的标准化构建组件。官方文档中的说构建系统，更多是为了定义标准化、可移植、可重用、性能高效的构建方法。Knative 提供了构建 CRD 对象，让用户可以通过 yaml 文件定义构建过程。

服务: 基于负载自动伸缩，包括在没有负载时缩减到零。允许你为多个修订版本应用创建流量策略，从而能够通过 URL 轻松路由到目标应用程序。服务的核心功能是让应用运行起来以提供服务。其提供的基本功能包括:

- 自动化启动和销毁容器
- 根据名字生成网络访问相关的 Service、ingress 等对象
- 监控应用的请求，并自动扩缩容
- 支持蓝绿发布、回滚功能，方便应用方法流程.

事件: 抽象出事件源，并允许操作人员使用自己选择的消息传递层。Knative 定义了很多事件相关的概念。

- EventSource
- Feed: 把某种类型的 EventType 和 EventSource 和对应的 Channel 绑定到一起。
- Channel: 对消息实现的一层抽象，后端可以使用 kafka、RabbitMQ、Google PubSub 作为具体的实现。
- Subscription: 把 channel 和后端的函数绑定的一起，一个 channel 可以绑定到多个 Knative Service。

Knative 是以 Kubernetes 的一组自定义资源类型（CRD）的方式来安装的，因此只需使用几个 YAML 文件就可以轻松地开始使用 Knative。因此，Knative 最大的特点是具有支持多云（不依赖于某一个 IaaS 云提供商），并且与 Kubernetes 的产品和生态具有延续性。

企业基于 Knative 的 Serverless 架构，可以完全借助已有的容器和 Kubernetes 环境，并进一步提供更加零运维、动态扩展和高度抽象的应用开发运行平台。

◆ Apache OpenWhisk

Apache OpenWhisk就是这样一款由IBM开源的事件驱动的无服务器计算（Serverless Computing）平台，目前是Apache基金会的孵化项目，使用了Apache 2.0的许可证，主要贡献者来自IBM，Adobe，Naver等公司。IBM公有云平台上的云函数就是使用了OpenWhisk的核心代码。Apache OpenWhisk支持主流的编程语言如Node、Java、Python等，定义并实现了基于事件处理的编程模型，可以安装在任何支持Docker的平台和操作系统上。

Apache OpenWhisk的事件处理编程模型，包括三个关键的概念：触发器（Trigger）、规则（Rule）与动作（Action）。触发器用于定义需要响应的时间源，例如数据库中的一条新记录、GitHub中的一次提交、或是某台移动设备进入了某个地理围栏。动作则是或大或小的代码片段，可以通过Node、Swift或Java等语言实现，也可以由Docker容器中的任意二进制文件进行定义。规则定义了触发器和规则之间的关联，当某个触发器启动时，相应的动作会被直接调用。动作除了被触发调用在之外，也可以通过HTTP调用直接执行。

Apache OpenWhisk预先定义好了一些常用的触发器、动作和规则，

利用它们，开发人员可以很方便的实现自己的代码逻辑。例如:

- 对Cloudant数据库的增删改查事件作出响应
- 对天气预报作出响应
- 对Github上的代码库的提交事件作出响应
- 对分布式消息系统 Kafka上的消息的产生作出响应

利用Apache OpenWhisk的编程模型和预定义好的触发器，用户可以非常方便的实现各种无服务器计算的场景，非常适合于执行计算任务，或者处理具有很高的并发性水平的系统。

作为开源的成熟无服务器项目，Apache OpenWhisk也在周边工具的开发上非常活跃，如部署工具、调试工具、与Knative集成工具等等。例如，部署工具WskDeploy，能够将一个完整场景所需要的资源定义为YAML文件，一键部署在OpenWhisk上。

◆ Riff Project

Riff 是与 Knative 紧密相关共同工作的一个项目，由 Pivotal 和 Google 公司为主要贡献。Riff cli 帮助开发人员使用 knative 构建和运行函数。Riff 包括在 kubernetes 集群中安装 knative 以及管理函数、服务、通道和订阅的命令。Riff 允许开发人员编写响应事件的函数。函数被部署为 kubernetes pods，其中包含自定义函数的特定语言调用程序，以及用于在函数范围内外获取数据的 I/O bound Sidecar。SideCar 负责读/写消息主题，并使用参数分派调用程序。

Riff 使用自定义资源定义来枚举 kubernetes 中的函数和主题。此外，它还部署了一对控制器盒来管理这些资源-主题和功能控制器。主题控制器使用基础事件代理处理主题状态更改。功能控制器监听主题事件并管理功能部署、销毁和扩展需求。包括: Riff Cli 安装 Knative 和使用 Knative serving 基于 Kaniko-based 集群的 builds，developer workflow 等等。

（二）工具链

1. 应用框架

蚂蚁金服 SOFAStack 技术框架

SOFAStack 是蚂蚁金服自主研发的分布式中间件，为用户提供安全、稳定、可靠、高效、敏捷的基础架构能力，用于打造大规模高可用的分布式系统架构。SOFAStack 以轻量级服务框架为基础，兼容 Spring Boot、Spring Cloud、Dubbo 工程，提供应用中心、微服务、消息队列、数据访问代理、分布式链路跟踪、分布式事务、Serverless 等服务。

蚂蚁金服的 Serverless 服务配备文件储存，数据储存，服务托管，函数计算等诸多能力。文件储存方面，Serverless 平台为开发者提供了基于 CDN 的文件 BaaS 服务，开发者只需将文件通过接口上传，即可直接享受到 CDN 的能力，为文件带来最佳的访问性能以及海量的访问量。数据储存方面，用户可以通过客户端的 SDK 操作数据库里的

数据，无需服务端参与，即可完成数据的存取操作。通过服务托管，开发者无需再关系底层环境、后端运维的各种细节。开发者只需将业务代码提交到云端即可。通过函数计算，开发者可以将原有的复杂计算逻辑拆分为多个计算函数，然后通过事件或者 http 方式串接起计算业务，在实现对业务解耦的同时也能缩短对后端资源成本的依赖。

腾讯云无服务器框架

TCSAM 是用于在腾讯云上定义 Serverless 应用的模型。基于 TCSAM，腾讯云提供了 tcf 命令行工具用于云函数的管理、部署。

TCF 全称为 Tencent Cloud Function，是腾讯云无服务器云函数 SCF（Serverless Cloud Function）产品的命令行工具。通过 tcf 命令行工具，您可以方便的实现函数打包、部署、本地调试，也可以方便的生成云函数的项目并基于 demo 项目进一步的开发。

TCF 通过 TCSAM 规范的模板配置文件，完成函数及相关周边资源的描述，并基于配置文件实现本地代码及配置部署到云端的过程。同时 TCF 命令行工具提供本地事件模拟、本地调试等用于调试的相关功能，方便用户进行本地调试及测试。TCF 还提供了通过使用命令行工具将函数的管理、测试、部署、发布对接到持续集成及持续发布流程中的能力。

2. 监控/可视化

无服务架构的应用通常会部署成百上千个函数，访问调用的复杂性急剧上升。通过监控/可视化工具，可帮助用户或运维人员监测链路状态，掌握函数运行状态，快速定位问题源头。

Grafana

Grafana 是一个跨平台的开源的度量分析和可视化工具，可以通过将采集的数据查询然后可视化的展示，并对监测指标做告警通知，它常用于可视化基础设施和应用程序分析的时间序列数据。Grafana 搭载后端的 Prometheus 数据源，可以为多种开源 Serverless 框架构建函数计算监测平台。

3. 测试调试

由于公有云的函数服务没有开发环境，开发人员必须运行函数查看它们真实的运行情况，因此创建模拟测试环境并用于代码调试的工具变得非常必要。

华为云函数服务 Serverless Sandbox（HSS）

用户开发的函数在部署到华为云之前，可以使用华为云 Serverless Sandbox（HSS）在本地开发和测试 Serverless 应用。该工具可以用来在本地测试函数功能，验证华为无服务器应用模型（HSAM），并为各种事件源本地生成样本有效载荷；提供了丰富的

cloud event 命令，可以将来自华为云服务的事件直接路由到本地环境来调试本地函数功能。

百度云 CFC BSAM 工具

BSAM CLI 是一个基于 BCE SAM 规范的命令行工具，它提供了本地开发环境，帮助用户在把函数上传到百度云 CFC 之前，在本地进行函数的开发、分析和执行。

4. 安装部署

函数应用跨区域移植部署的配置非常繁琐，极易出问题。能够将应用的配置描述分离，复用给多个应用可以大大简化移植部署的难度。

阿里云 Fun2.0

阿里云 Fun2.0 是一款 Serverless 应用开发的工具，可以帮助用户定义函数计算、API 网关、日志服务等资源。Fun 2.0 版本引入了全新设计的 Serverless Application Model（SAM）规范。

SAM 作为一种基础设施即代码（Infrastructure as Code），允许用户描述函数计算及其相关云资源，可以使用同一份模板文件，进行跨 region 或者账户部署您的云应用。描述云资源的模板文件，也会成为项目代码的一部分，在不同开发者之间共享。这极大的降低了 Serverless 应用的交付难度、管理难度、移植难度。除了 1.0 版本支持的函数、API 网关的配置，Fun2.0 中新增了特性:

- 增强了对函数的描述能力：环境变量、日志服务、角色属性、VPC 属性等。
- 支持配置新的应用资源，比如 Table Store、日志服务等。
- 代码上传可以指定文件、目录、压缩包以及 OSS 路径。
- 更多的 API 网关参数配置。

三、 无服务器架构适用场景

结合无服务器架构的基于事件驱动，应用代码动态部署，完全动态的进行大规模资源资源扩缩等等特点。当前阶段，可以把无服务器架构的适用场景可以大致分为下面几类：

(一) 应用后端服务

通过将无服务器云函数和其他云服务紧密结合，开发者能够构建可弹性扩展的移动或 Web 应用程序，轻松创建丰富的无服务器后端，而且这些程序可在多个数据中心高可用运行，无需在可扩展性、备份冗余方面执行任何管理工作。

- 移动应用后端服务

使用无服务器架构技术构建移动后端服务是非常常用的场景。它允许开发人员在基于云平台的后端服务来构建应用。这使得开发人员可以更加专注在移动应用的优化上，只要按需选择云服务商提供的丰富的后端服务即可。例如微信小程序的开发。

- Iot 后端服务

物联网的应用场景中，设备传输数据量小，且往往是以固定时间间隔进行数据传输，数据传输存在明显的波峰波谷特征。数据传输的波峰时段触发后端函数服务集中处理，处理结束后快速释放，提升资源的利用效率。

(二) 大规模数据处理和计算类

- 人工智能推理预测

人工智能的推理预测的调用需求会随着业务的起伏而变化，具有一定的波动性，和人工智能训练时的较固定计算周期和运行时长有所不同。同时 AI 推理一般会使用 GPU 加速，这种明显的峰值变化会导致大量的资源浪费。使用无服务器架构技术可以有效解决上述问题。高业务请求到来时，云函数的执行实例自动扩容，满足业务需求；而在请求低谷或无请求到来时，云函数自动缩容甚至完全停止，节省资源使用。

- 批处理或计划任务

每天只需短期运行就能以异步计算的方式进行强大的并行计算能力，IO 或网络访问的任务非常适合无服务器架构。这些任务可以以弹性方式运行时消费所需的资源，并且在不被使用的当天剩余时间内不会产生资源成本。例如定期的数据备份等。

(三) 基于事件的内容处理类应用

- 实时文件处理

有些应用会根据不同的应用需求将图片裁剪成不同尺寸，或添加不同的标签水印。视频类的应用会将视频流转码成不同的清晰度推送给不同服务。当图片或者视频流通过对象存储上传时便会触发相应的函数计算，根据计算规则自动按需处理，整个过程无需再搭建额外服务器，也无需人工干预。

- 定制事件触发

以用户注册时发邮件验证邮箱地址的场景举例，可以通过定制的事件来触发后续的注册流程，而无需再配置额外的应用无服务器来处理后续的请求。

四、 无服务器架构与主流部署形态的对比

新的计算架构正在改变企业搭建和使用计算资源的方式，从物理机到虚拟机，提高了硬件的利用率，并使资源的使用和变更变得更加灵活。容器技术进一步降低了应用对运行操作系统的依赖，提高了应用的可移植性和交付效率，无服务器计算的出现，使得用户无需管理和运维服务器，只需要关注代码，进一步提升了开发效率。虚拟机、容器以及无服务器架构并非是完全替代的关系，需要根据各自业务应用的特点、需求选择合适的架构部署。

虚拟机、容器和 Serverless 都对 IT 基础设施资源做了抽象，但三者之间也存在显著差异。无服务器架构并不能够完全替代虚拟机及容器的应用，在部署方式、安全隔离程度、生态完整程度、场景适用性等方面各有优缺点，拥有各自有的技术受众。

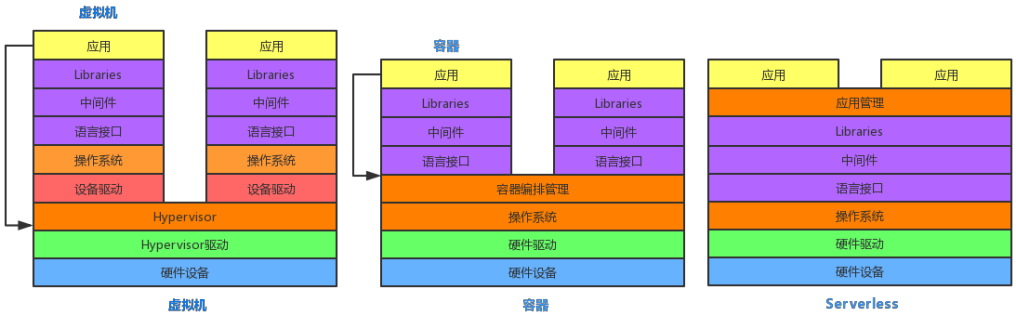

图 7：虚拟机、容器、Serverless 抽象示意图

（一） 虚拟机部署：稳态业务的首要选择

虚拟机从云计算诞生以来一直是计算基础设施的支柱，它的优缺点早已被广泛认知。尽管在过去的十年中，虚拟机的性能已经发生了重大转变，但是低延迟、可预测的高性能以及更好的隔离环境，仍然使它成为静态单体架构应用程序和性能敏感的工作负载，即高性能计算应用（例如视频编码、机器学习）的部署首选。对于大型应用程序，虚拟机也是最主要的部署对象，特别是强调数据持久性和有状态应用。虚拟机提供的强大安全性和隔离功能使其成为 IT 运营、风险管理和开发人员可以达成一致的默认计算抽象。

（二） 容器部署：微服务架构的最佳载体

虽然虚拟机仍然是大多数工作负载扩展的主要单位，但容器技术由于其面向应用的扩展方式以及超强的可移植性以及轻量级优势，也逐渐成为应用部署的重要组成部分。

对于微服务架构的应用程序，容器是最佳的部署方式。微服务很好地支持了语言技术栈的多元化，它通过切分系统的方式，为不同功能模块划定了清晰的边界，边界之间的通信方式很容易做到独立于某种技术栈，因此也就为纳入其它技术带来了空间。但是不同技术栈的微服务之间，除了需要考虑通信机制，还要确保这些技术能以较低成本整合成一个系统。容器技术将所有应用都标准化为可管理、可测试、易迁移的镜像，因此为不同技术栈提供了整合管理的途径。

容器可以运行在裸机内核基础架构上而不需要 hypervisor 层，避免了虚拟化带来的性能消耗，也使得业务的部署管理更为简化。因此，容器化部署也是面向裸金属服务器的工作负载的资源利用率和管理复杂性的一个重要解决方案。

由于资源占用空间较小，容器可以在主机上支持更高的租户密度，从而进一步加强服务器整合。这可能会显著影响商业软件的供应商许可和基础架构获取及运营的成本。

（三） 无服务器化部署：事件驱动下的部署新形态

Serverless 使开发人员能够专注于由事件驱动的函数组成的应用程序。结合平台多种后端服务，用户只需完成代码上传、触发器逻辑定义、扩展配置等一些基础配置，便可经由触发器所关联的后端事件源触发，实现整个业务的部署。无服务器化部署将对基础设施管理要求降到最低，无需考虑系统级的运维、容器的调度管理甚至运行时的问题。Serverless 可以实现自动扩展、弹性负载均衡以及最细粒度的“即用即付”计费模型。

无服务器计算的基础设施供应和管理基本上是从服务的消费者中抽象出来的。无服务器计算的主要优点是：它支持运行代码，且无需关心底层基础架构的运行。由于后端资源的自动缩放属性，它可以实现更轻松的水平扩展。当使用公有云的函数服务时，无服务器框架做到了真正的按需消费，没有空闲资源或孤立的虚拟机或容器，资源仅在触发时调用，处理完成后便会迅速释放。

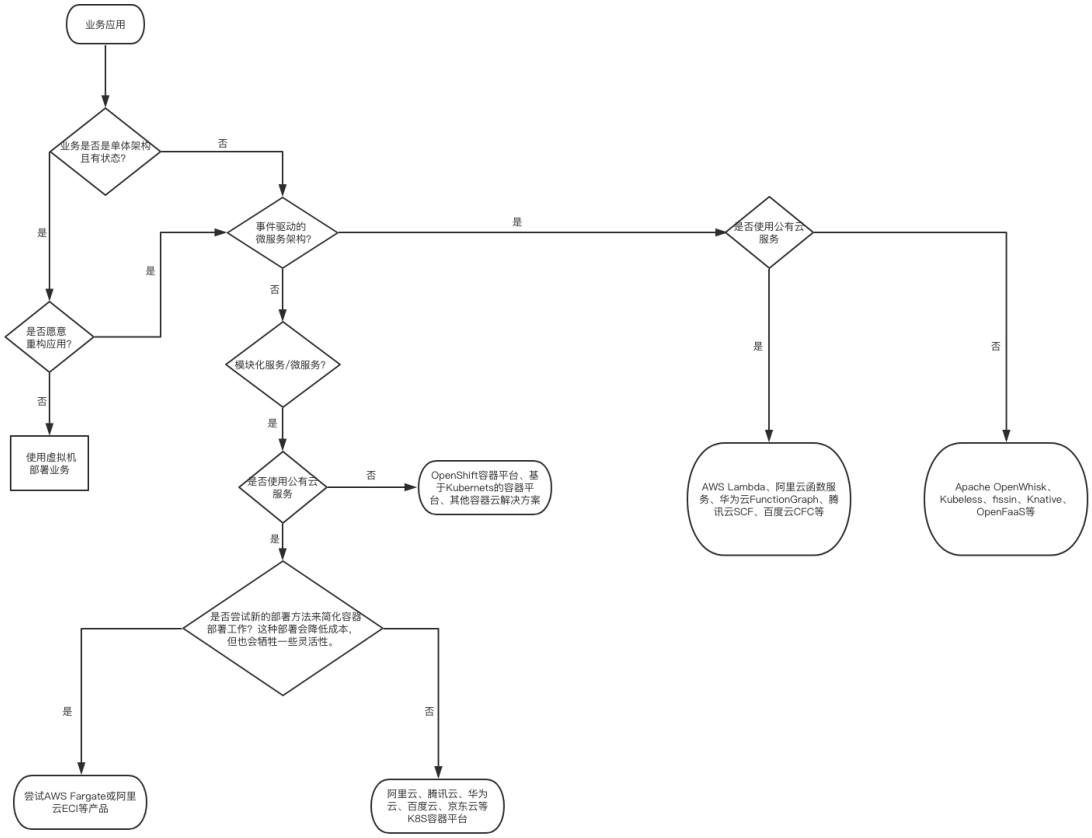

图 8: 虚拟机、容器、Serverless 部署选择参考

五、 无服务器架构的安全

从用户的视角看，采用无服务器架构后应用服务或者服务器都在调用时才会触发，并不长时间存在，服务的攻击面变小了，安全性相比有所提升。但是深层次来看，不同用户的 API 函数仍在后端服务器上运行，并使用不同的函数 runtime 来解析 API 请求，这反而使服务的攻击面变得更大。

函数服务的载体是容器，而容器运行在物理/虚拟的服务器上。考虑无服务器架构的安全需要从容器安全、函数安全和鉴权管理三个方面考虑。

基础环境的容器安全，包括 CVE（通用漏洞披露）管理、漏洞管理、Runtime 安全、镜像和代码安全扫描、身份和访问管理、数据保护等，这些在无服务器架构中，仍然需要考虑并集成相应方案。

暂态运行环境下的函数安全。在应用程序代码方面存在一定风险，包括可以由外部用户执行的潜在攻击，利用无服务器功能代码、外部库、操作系统依赖性中的弱点或漏洞等。通过扫描函数代码中的开放源代码组件、外部库和操作系统依赖项，来最小化无服务器函数的应用程序代码攻击面。将扫描结果与漏洞数据库关联起来，从而识别潜在的漏洞。另外通过扫描代码，警告代码中存储的秘密不安全。以上可以通过集成来自动的、连续的对新发现的漏洞发出警报，强化函数安全。

无服务器函数的另外一个基本挑战是它们的权限定义。函数的权限是在一个单独的过程中定义的，通常是在函数代码的早期开发阶段，并且是独立地为每个函数定义的。基于细粒度划分的函数应用服务增加了错误、不当配置的可能性,进而导致潜在攻击者获得机会，访问超出自身权限范围之外的更多服务。同时，“最小权限模型”（即只赋予访问者最小可用的访问权限)的需求正在被相关的云安全研究所关注。

随着无服务器工作负载的使用增长，无服务器架构的安全问题日益凸显，越来越多的组织结构正在致力于安全问题的解决。

六、 无服务器架构技术发展趋势

无服务器架构技术诞生时间较短，但已引起广泛关注。随着技术应用推广的深入，无服务器架构技术将适配越来越多的场景，其相关生态系统环境也会不断完善，我们目前所知的 IT 基础设施在未来几年可能会发生重大变化。

更细粒度的函数规格有助提升资源利用效率。容器是承载函数代码的基本单元，目前主流云服务商提供的函数规格都在 64M 以上。随着应用逻辑的不断切分细化，现有函数规格的资源利用效率将会降低，与函数代码相匹配的更细粒度的函数规格。

无服务器架构技术的工具链生态将持续完善。2018 年全球范围内的无服务器架构的采纳率急速上升，企业级的大规模采用也正在发生。国内的 Serverless 市场尚未爆发，但采纳率也呈现抬头之势。广泛的应用将会刺激云服务商持续不断的优化产品，丰富和完善平台产品的工具链。同时无服务器架构私有化部署已有趋势，未来将会有更多的独立第三方服务商参与到无服务器架构的技术生态中。

Kubernetes 将成为无服务器架构的重要基础设施。Kubernetes 已经成为跨多个云提供商的容器编排的事实上的标准，正在成为默认的操作系统。Kubernetes 在云原生领域无处不在，也正在逐渐成为运行无服务器应用程序的标准。它可以轻松开发和运行无服务器应用程序，利用 Kubernetes 的内置功能如调度程序、集群管理、扩展、服

务发现、网络等，提供超强的可移植性和互操作性，所有这些都是无服务器运行时所需的。Kubernetes 作为无服务器基础架构的标准化能力，允许用户在私有云环境或者是多云环境下运行 Serverless 应用无需担心厂商锁定问题。随着 Kubernetes 围绕云原生架构的部署得到进一步完善，基于 Kubernetes 的无服务器框架将大有可为。

七、 落地应用案例

（一） Serverless 架构助力小米音乐曲库更新效能飙升

小米音乐需要定期更新曲库，从而给用户提供更好的音乐服务，更新曲库的时间不确定，但是更新一次需要处理的数据量比较大，一般是几百个 TB 的音乐数据，处理完成需要数天。处理的流程如下：

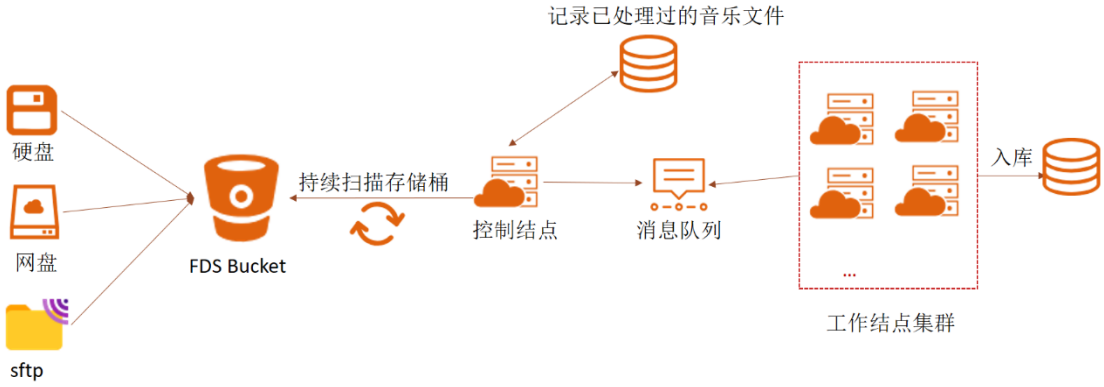

图 9：小米音乐早期架构

音乐文件存储到小米的文件存储服务器 FDS Bucket 之后，需要有一个控制结点来不断扫描 FDS Bucket 的文件，然后把已处理的文件记录到数据库中，对于未处理的文件，推入到小米的消息队列当中。

另外还需要工作结点集群来处理这些音乐文件，对文件进行转码，加密，最后把音乐入库。

业务痛点：

1) 工作结点扩容和缩容，完全依赖经验，一般会根据最大的量评估工作结点的规模，没有任务处理的时候浪费资源。
2) 需要有控制结点来不断的轮询 FDS Bucket，但并不是需要一直来处理音乐文件，只有在需要新增或者更新曲库的时候才需要处理，在空闲的时候造成了资源的浪费。
3) 部署困难，即使一次小的改动，也需要去重新编译部署整个项目。

在小米函数计算服务上线后，小米音乐决定重构自己的架构，采用无服务器的方式来处理音乐文件，重构后的架构如下：

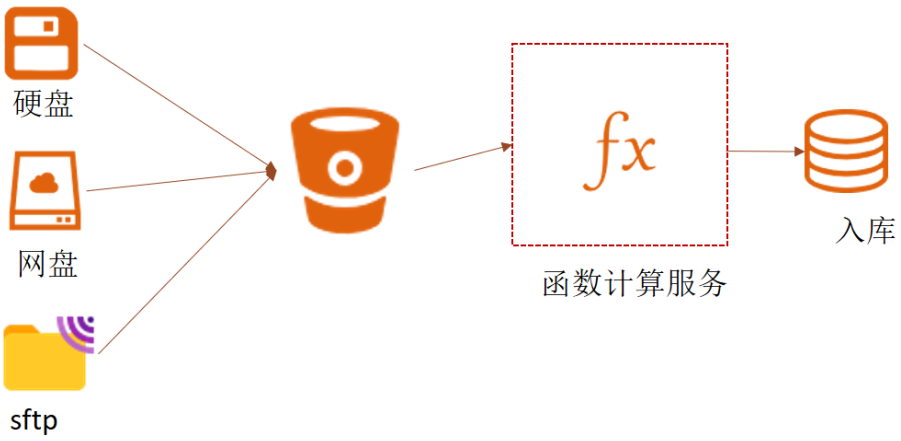

图 10：小米音乐 Serverless 改造后的架构

重构后的架构去除了所有的服务器资源，也不需要使用消息队列来接受任务，只需要编写核心的业务逻辑就可以了，由此带来的提升是很显著的：

1) 小米音乐团队不需要去运维和管理服务器资源了，大大节约了运维成本
2) 音乐转码，加密的任务只会在需要的时候才会被触发，函数计算服务才会给任务分配计算资源，从整体上提高了资源利用率
3) 不需要担心任务执行失败，函数计算服务会自动重试失败的任务
4) 部署发布更方便，小米音乐可以根据不同的功能打包成干个函数，使用函数计算服务管理函数的不同版本，并能够方便的切换不同的版本。

（二） 无服务器架构实现支付宝小程序后台服务

传统模式下，小程序开发遭遇挑战。在传统模式中，程序员去开发一个小程序的时候，依旧需要采用像开发传统 App 一样的方式进行业务开发。在整体业务开发中，需要前端开发、后台开发、运维人员、安全人员等多个角色的协同，导致人力成本和资源成本高昂，不利于小程序的开发。

蚂蚁金服采用 Serverless 模式这种更高效的研发方式来实现小程序的快速布局。基于蚂蚁的 Serverless 产品 Basement，可以用更高效、简单的方式快速实现稳定、可靠的小程序后台服务。Basement 的技术架构如下:

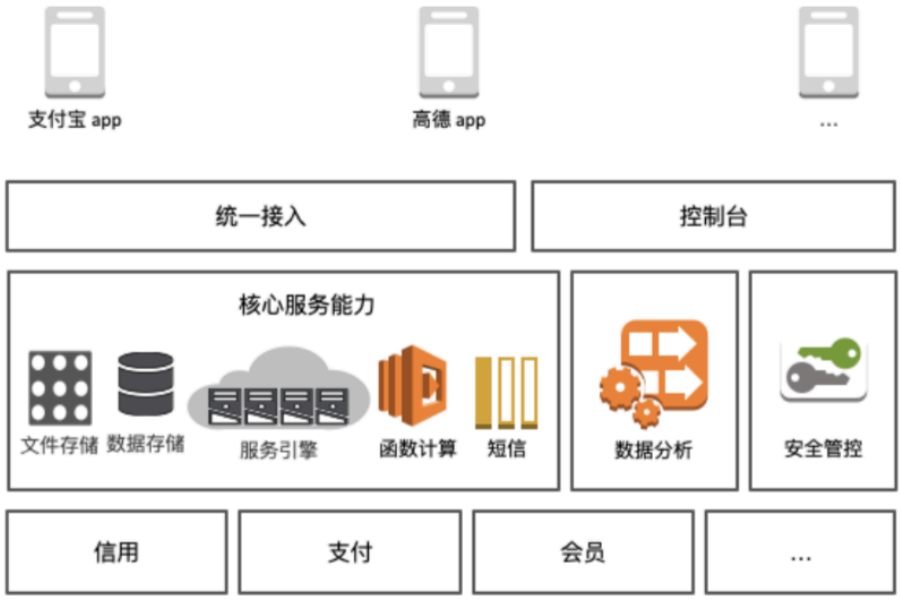

图 11：支付宝小程序后台架构示意图

其中，统一接入与控制台，可以避免客户去处理各个端上的认证差异，通过统一 API 实现快速认证，大大缩短开发时间；核心服务能力中的文件存储 BaaS 服务可以实现一键文件上传的功能，小程序可享受 CDN 的加速能力，并天然具备高并发能力；数据存储的 BaaS 能力可快速实现数据操作；服务引擎可实现底层的构建、部署、运维的托管，真正释放开发者，免运维、高效率。

函数计算和服务引擎共同组成了后端解决方案，可以通过事件触发或串接，实现功能封装，以达到业务的解耦和快速迭代。支付宝小程序 Serverless 模式带来的优势显而易见：

1) 研发效率提升，实现了复杂底层逻辑的托管，用户只需完成自己业务逻辑的开发即可，开发时间大大缩短，研发效率大大提升；
2) 高可用的服务能力，支持了同城多机房的容灾能力，所有服务的数据都会进行多机房的互备，同时在应用层，提供动态切换能力，可以保障服务高可靠性和业务高稳定性；

3) 专业的安全管控，为用户的服务提供了全方位的安全管控，包括流量防护、防火墙防护等接入层控制，涉黄、涉政、暴力等内容安全控制，保障数据不发生非法访问以及泄漏的访问控制。

4) 低成本，Serverless 模式下，人力投入成本低，资源成本低，收益高。

总体而言，Serverless 模式帮助支付宝提供可靠、稳定、安全的小程序服务，为开发者提供简单、高效的小程序开发方式。

(三) 小程序云开发助力腾讯相册小程序快速成长

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。

2018 年 12 月，腾讯相册累计用户量突破 1 亿，月活 1200 万，阿拉丁指数排行 Top30，已经成为小程序生态的重量级玩家。

客户痛点:

1) 小程序是新项目，人力非常紧缺，后台不能百分百投入

2) 后台系统有历史包袱，新功能开发难度大

3) 采用传统开发模式搭建小程序架构，费时费力。如下图所示，除了数据读取、文件管理、敏感逻辑的处理（如权限）外，还需考虑很多诸如弹性伸缩、容灾、负载均衡等技术问题。

小程序·云开发是微信团队和腾讯云联合开发的，集成于小程序控制台的原生 Serverless 云服务。核心功能包括: 云函数、云数据

库和云存储。用户只需编写核心逻辑代码，无需关注后端配置与运维，专注于业务开发。

以快速上线的点赞评论功能为例，存在如下典型问题:

1) 新增小程序评论、点赞等操作需要用户的鉴权信息;

2) 原有的后端服务架构太复杂，增加新功能非常困难

经过经由 Serverless 架构构建服务，利用云函数路由功能，在原有的相册服务端获取用户的鉴权信息，匹配原有后台服务，判断该用户在小程序端是否有权限进行敏感操作，轻松实现业务需求。

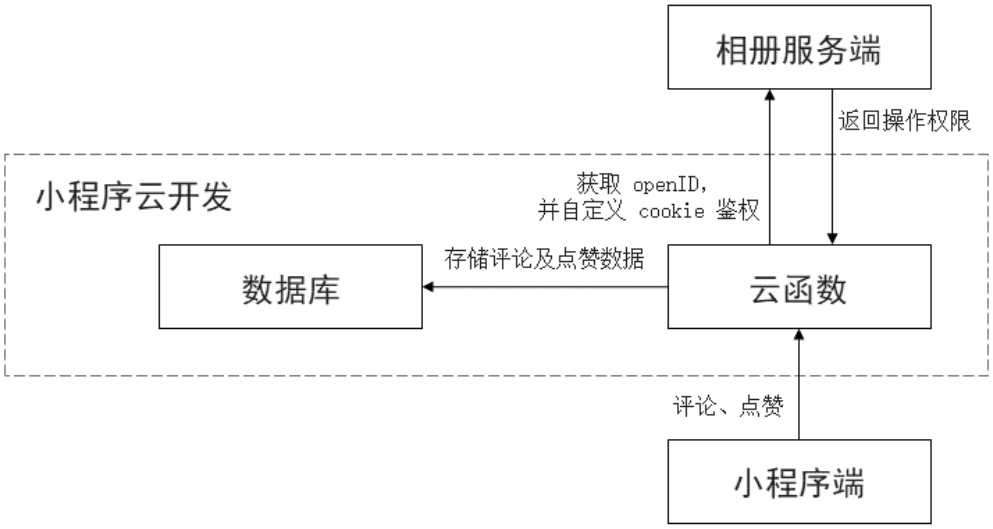

图 12：微信相册小程序评论功能后台架构示意图