阿里云
专有云Agility版

产品简介

产品版本：V1.1.0

文档版本：20180416

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的”现状“、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。
5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含”阿里云”、Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

表 -1: 格式约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **禁止**： 重置操作将丢失用户配置数据。
	该类警示信息可能导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **警告**： 重启操作将导致业务中断，恢复业务所需时间约10分钟。
	用于警示信息、补充说明等，是用户必须了解的内容。	 **说明**： 导出的数据中包含敏感信息，请妥善保存。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 **说明**： 您也可以通过按**Ctrl** + **A**选中全部文件。
>	多级菜单递进。	**设置** > **网络** > **设置网络类型**
粗体	表示按键、菜单、页面名称等UI元素。	单击 **确定**。
courier字体	命令。	执行 cd /d C:/windows 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	bae log list --instanceid *Instance_ID*
[]或者[a|b]	表示可选项，至多选择一个。	ipconfig *[-all|-t]*
{}或者{a|b}	表示必选项，至多选择一个。	swich *{stand | slave}*

目录

1 产品概述

随着云服务的不断成熟，专有云也在同时迅猛地发展。目前，以政务云为代表的专有云在国家政策的大力支持下发展迅速。金融、安防等传统行业上云进程加快，专有云市场逐渐得到云服务商、系统集成商、IDC服务商以及各行业用户的广泛关注。

此外，专有云市场还有以下特点：

- 传统专有云解决方案部署周期长、价格贵、技术要求高。
- 中小规模专有云占据了大部分专有云市场需求，典型用户的部署规模偏小。
- 主要面向需求相对简单，满足简易、快速、经济的部署专有云的企业客户。

为了针对不同企业级市场的使用特点，为客户提供一个开放、统一、可信的企业级云平台，增强在云市场上的核心竞争力，满足更广泛的业务需求，我们推出面向企业中小规模专有云场景的敏捷云应用平台，可以直接部署在企业已有的x86等硬件基础之上并加以管理，支持研发运维一体化、云原生应用架构和机器学习等场景，允许应用在公共云和自有数据中心物理机统一部署管理，支持应用无缝迁云、弹性伸缩、应对突发流量等场景。提供真正企业级的安全和稳定服务保障。

2 系统架构

2.1 架构介绍

专有云Agility版基于开放API模型，为您提供企业级云安全架构，与统一的管理运维体验。

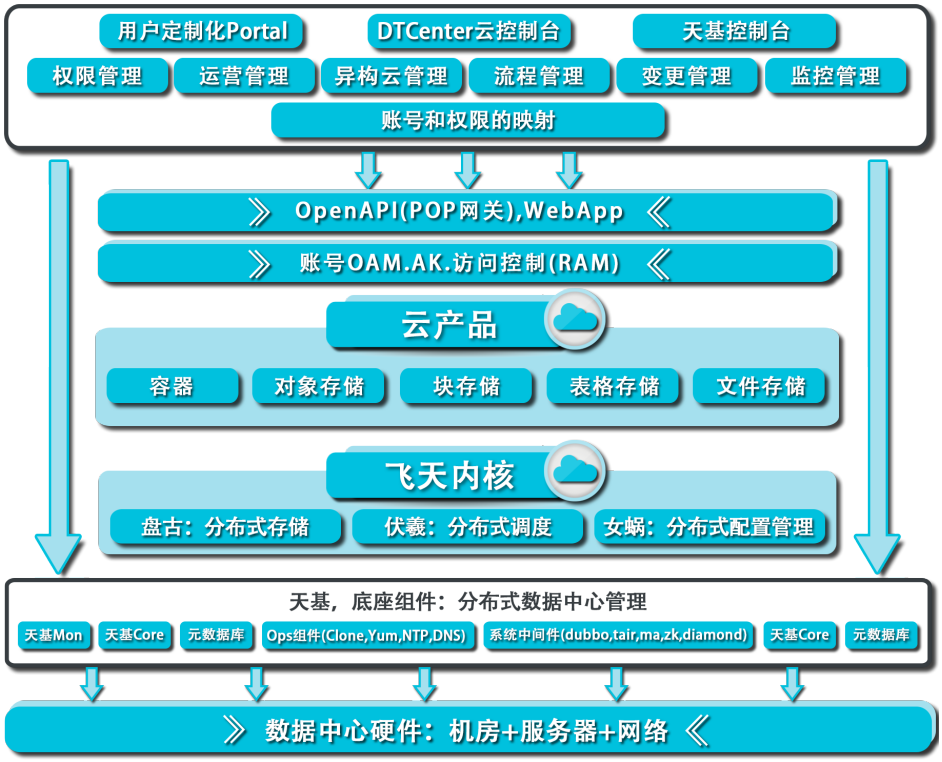
2.2 逻辑架构

专有云Agility版通过将物理服务器的计算和存储能力虚拟化成虚拟计算、分布式存储，并在此基础上提供容器和存储服务，为您的应用系统提供IT基础服务的支撑能力，同时可以和您现有的账号体系，监控运维系统进行对接。

专有云Agility版逻辑架构有如下特点：

- 硬件基础是IDC+X86服务器+网络设备。
- 飞天内核（分布式引擎），基于飞天提供了各种云产品。
- 所有云产品都要求遵从一个统一的API框架，管理与运维（账号、授权、监控、日志）体系及安全体系。
- 保证所有产品遵从一致性的使用体验。

图 2-1: 逻辑架构


用户定制化Portal
DTCenter云控制台
天基控制台
权限管理
运营管理
异构云管理
流程管理
变更管理
监控管理
账号和权限的映射
OpenAPI(POP网关),WebApp
账号OAM.AK.访问控制(RAM)
云产品
容器
对象存储
块存储
表格存储
文件存储
飞天内核
盘古：分布式存储
伏羲：分布式调度
女蜗：分布式配置管理
天基，底座组件：分布式数据中心管理
天基Mon
天基Core
元数据库
Ops组件(Clone,Yum,NTP,DNS)
系统中间件(dubbo,tair,ma,zk,diamond)
天基Core
元数据库
数据中心硬件：机房+服务器+网络

3 网络架构

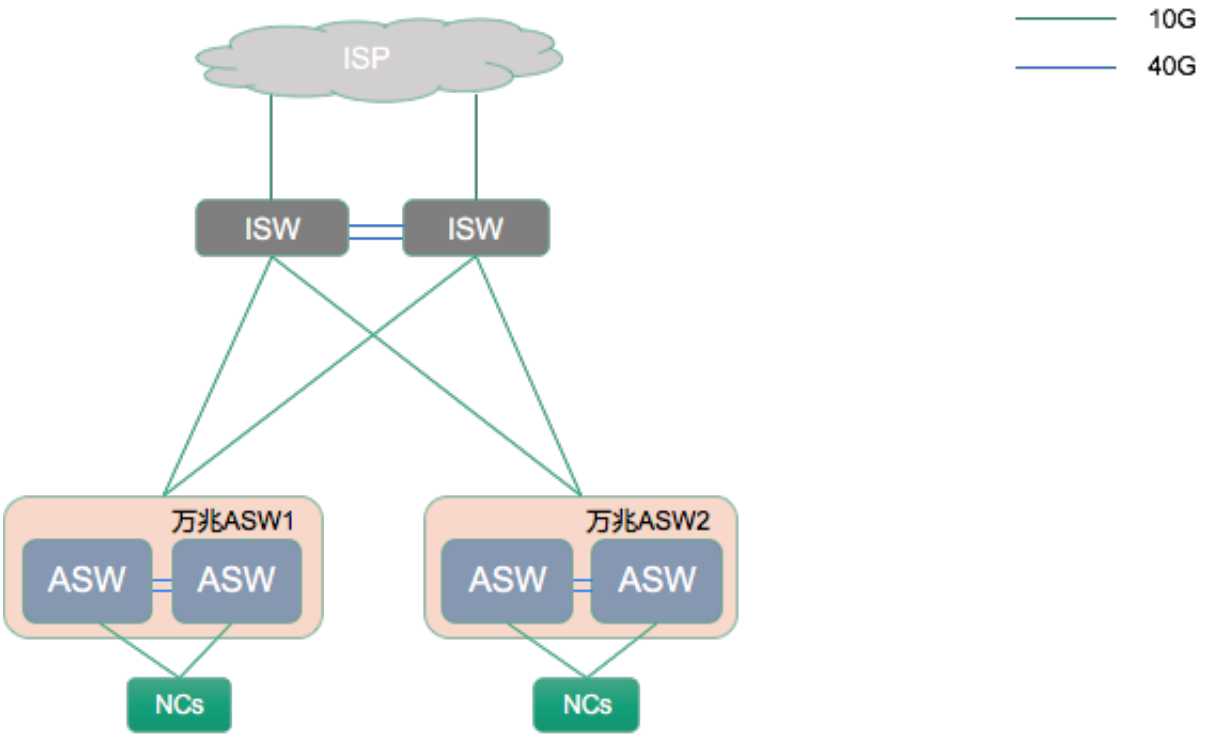
专有云Agility版是Apsara Stack的最小化产品输出，网络架构经过优化和精简，仅包含ISW和ASW两种设备角色，最大支持96台服务器规模，采用MiniLVS代替SLB，支持MiniLVS扩展BGP协议。

各层交换机的角色和作用如下表所示。

表 3-1: 角色定义

角色名称	作用
ISW	互联交换机。对外互联ISP，对内互联ASW。
ASW	接入交换机。接入云服务器，上行互联ISW。
OOB	带外管理（Out of Band）。
OMR	带外管理核心交换机。
OASW	带外管理接入交换机，互联服务器IPMI控制卡。
ACS	串口服务器，互联网络设备console口，用于设备管理。

图 3-1: 网络架构逻辑分区

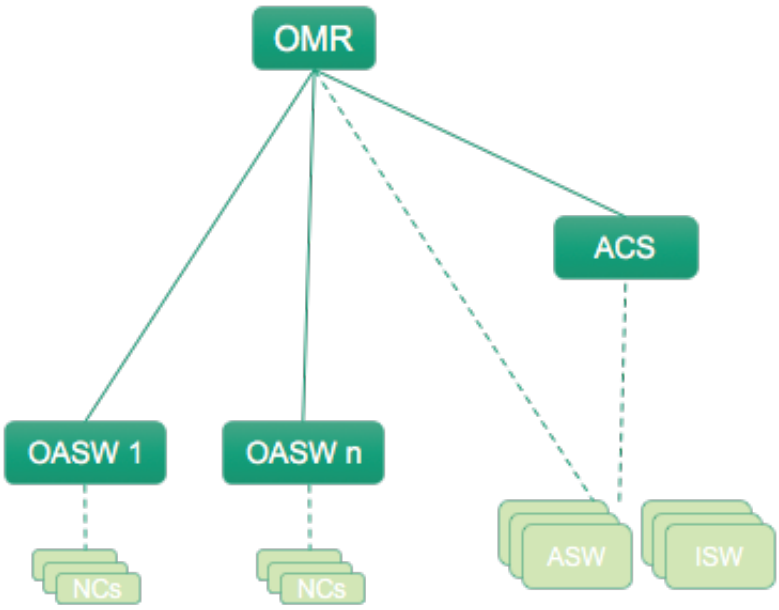


专有云Agility版网络架构中ISW对外互联ISP运营商（外网带宽取决于用户实际需求），对内互联ASW接入交换机。ISW固定输出2台，ISW之间互联带宽为2*40G，下行互联每组ASW带宽为320G。

专有云Agility版网络架构中ASW为接入交换机，互联服务器，为提供所有云业务的网络承载。ASW采用堆叠设计，两台为一组，敏捷版最多支持2组ASW，最大接入服务器能力为96台，每组ASW互联ISW带宽为320G，互联服务器带宽为960G，网络收敛比为1:3。

所有服务器配置双网卡，通过bond双链路上行ASW交换机，单台服务器出口达到20G。

图 3-2: 带外管理网



专有云Agility版OOB管理网提供集群内部服务器及交换机的管理能力，是天基系统对物理服务器进行装机、重启等操作的必要途径。

服务器管理网络

每台服务器的IPMI口（控制卡管理口）上联GE网线至OASW交换机。每台OASW交换机配置48千兆网口，OASW交换机二层透传，上联至OMR交换机，网关配置在OMR交换机。

网络设备管理网络

每台网络设备MGMT口（management管理口）上联GE网线直接连至OMR交换机。

网络设备Console口网络

每台网络设备Console口上联GE网线至ACS ，ACS上联至OMR交换机。

4 底座组件

专有云底座有三类组件，共同为云平台的部署和运维提供支撑。

表 4-1: 底座组件

组件		功能说明
Ops组件	Yum	安装软件包 软件源在初始装机阶段已经部署完毕，主要用于物理机上安装操作系统、部署飞天等专有云的应用软件包及其依赖的组件。
	Clone	机器克隆服务
	NTP	时钟源服务 部署在专有云的物理机，从标准NTP时间源同步，并授时给其他宿主机。
	DNS	域名解析服务 为专有云内部环境提供域名的正解和反解服务，在两个 OPS 机器上各运行一个 bind 实例，通过keepalived 提供高可用服务，在一台失效的情况下，另外一台能够主动接管。
底座中间件	dubbo	分布式RPC服务
	tair	缓存服务
	mq	消息队列服务
	ZooKeeper	分布式协同
	Diamond	配置管理服务
	SchedulerX	定时任务服务
底座基础组件	天基	数据中心管理
	天基Mon	数据中心监控
	OTS-inner	表格存储服务
	SLS-inner	云平台日志服务
	元数据库	元数据库
	POP	云平台开放接口Open API
	OAM	账号系统

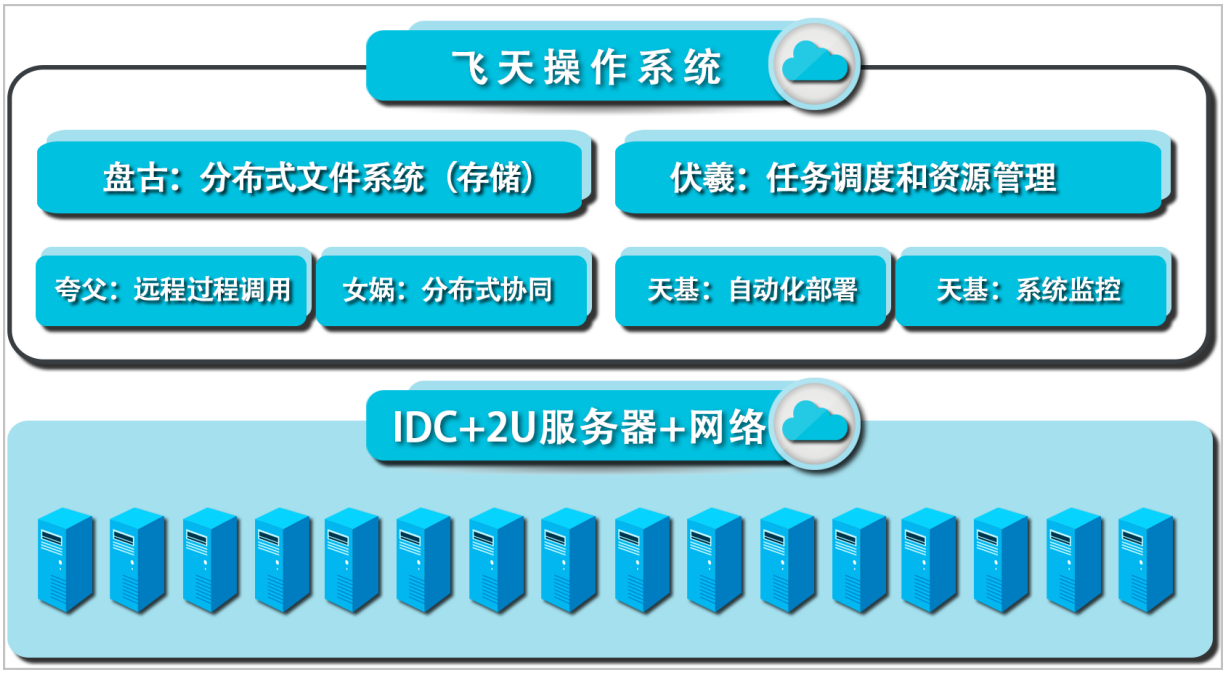
组件		功能说明
	RAM	认证授权系统
	WebApps	运维控制台支撑

5 产品优势

5.1 飞天大规模分布式计算系统内核

飞天大规模分布式计算内核，为上层的服务提供存储、计算和调度等方面的底层支持，其中包括：远程过程调用、安全管理、资源管理、分布式文件系统、任务调度和协调服务。通俗地讲，飞天就是把几千台通用的服务器整合成一台超级计算机。

图 5-1: 飞天内核系统架构



飞天平台内核包含的模块覆盖了以下主要的功能：

- 分布式系统底层服务

  提供分布式环境下所需要的协调服务、远程过程调用、安全管理和资源管理的服务。这些底层服务为上层的分布式文件系统、任务调度等模块提供支持。

- 分布式文件系统

  提供一个海量的、可靠的、可扩展的数据存储服务，将集群中各个节点的存储能力聚集起来，并能够自动屏蔽软硬件故障，为您提供不间断的数据访问服务；支持增量扩容和数据的自动平衡，提供类似于POSIX的用户空间文件访问API，支持随机读写和追加写的操作。

- 任务调度

为集群系统中的任务提供调度服务，同时支持强调响应速度的在线服务和强调处理数据吞吐量的离线任务；自动检测系统中故障和热点，通过错误重试、针对长尾作业并发备份作业等方式，保证作业稳定可靠地完成。

- 集群监控和部署

  对集群的状态和上层应用服务的运行状态和性能指标进行监控，对异常事件产生警报和记录；为运维人员提供整个飞天平台以及上层应用的部署和配置管理，支持在线集群扩容、缩容和应用服务的在线升级。

5.2 天基部署与管控系统

天基提供了云服务产品的统一部署、验证、授权和管控能力，为云服务提供基础性的支撑。天基框架中包含了部署框架、资源库、元数据库、认证授权、接口网管、日志服务、管控服务等模块。

- 部署框架为所有的云服务提供了统一的接入平台部署和服务间的依赖关系管理功能。
- 资源库保存了所有云服务和依赖组件的执行文件。
- 认证授权组件为云服务提供访问控制能力。
- 接口网关为云服务提供统一的API管理平台。
- 日志服务为云服务提供了日志存储、检索、获取等功能。
- 管控模块监控各云服务的基础健康状态，支撑云平台的运维体系。

5.3 久经考验的阿里云服务

容器服务是面向企业中小规模专有云场景的敏捷云应用平台，支持Docker企业版。可以直接部署在企业已有的x86等硬件基础之上并加以管理，支持研发运维一体化、云原生应用架构和机器学习等场景，支持混合云管理，允许应用在公共云和自有数据中心物理机统一部署管理，支持应用无缝迁云、弹性伸缩应对突发流量等场景。

存储包含了对象存储服务（简称OSS）、表格存储服务、文件存储NAS、块存储EBS。

- OSS可给用户提供基于RESTFul API的弹性扩展、高安全和高可靠的云存储服务。
- 表格存储是构建在阿里云飞天分布式系统之上的NoSQL数据存储服务，提供海量结构化数据的存储和实时访问。
- 文件存储NAS是面向阿里云ECS实例、HPC和Docker等计算节点的文件存储服务，提供标准的文件访问协议，您无需对现有应用做任何修改，即可使用具备无限容量及性能扩展、单一命名空间、多共享、高可靠和高可用等特性的分布式文件系统。

- 块存储EBS提供基于标准接口协议的开放API，满足客户对接Openstack云平台的业务需求，支持KVM、容器等计算虚拟化技术。

5.4 统一的运维管理系统

统一的运维管理系统包含云服务控制台和运维监控控制台。您可以通过控制台来进行账号管理、分配云服务资源、处理告警、升级系统、审计管理等操作。

图 5-2: 统一运维和运营管理Portal

5.5 开放的云服务接口

云服务通过OpenAPI平台，提供丰富的SDK包和RESTful API接口。您可以使用开放接口来灵活访问专有云提供的各种云服务。您还可以通过OpenAPI获取云平台的基础管控信息，将专有云平台接入到您统一的管控系统。

6 容器服务

6.1 产品概述

容器服务是面向企业中小规模专有云场景的敏捷云应用平台，支持Docker企业版。可以直接部署在企业已有的x86等硬件基础之上并加以管理，支持研发运维一体化、云原生应用架构和机器学习等场景，支持混合云管理，允许应用在公共云和自有数据中心物理机统一部署管理，支持应用无缝迁云、弹性伸缩应对突发流量等场景。阿里双十一大规模验证，Docker公司战略合作，提供真正企业级的安全和稳定服务保障。

6.2 功能特性

资源调度

- 支持大规模集群的统一资源池化管理，支持管理用户的物理机、VMware等现有IAAS环境。
- 支持根据应用需求动态调度容器，可选择多种维度的调度策略，比如资源维度（CPU、内存、GPU等）、可用性要求维度、应用拓扑的亲和性维度等等。

微服务

- 内置通用的服务注册、发现、路由、负载均衡等机制，对开发语言和中间件无特殊需求。
- 提供声明式方式配置，无需编码。
- 支持Spring Cloud等开源微服务框架。

DevOps

- 内置容器化DevOps最佳实践，可以实现一键式从代码提交到应用变更上线的全自动流程。
- 支持与三方、开源CI/CD方案整合。
- 提供不间断发布、蓝绿发布、灰度发布等发布机制，支持灵活、可控的服务更新。

日志与监控

- 提供企业级日志采集和输出方案。无缝集成容器日志采集，支持采集标准输出或指定目录的日志。
- 容器服务集成了ELK 第三方开源日志框架，并做了功能扩展，支持企业对容器日志服务的需求。
- 提供容器级别、应用级别和宿主机级别的多维度监控，提供服务和应用视角聚合数据。
- 支持脚本、URL等自定义监控。

- 容器服务集成了Grafana、InfluxDB、Elasticsearch等三方开源监控工具，并做了功能扩展，支持灵活的自定义监控Dashboard能力。

安全

- 与企业用户目录无缝集成，支持统一用户认证管理。
- 支持基于角色的授权模型，对集群资源灵活控制。

开放的接口

- 兼容Docker原生工具链和API。
- 兼容Docker Swarm和Compose编排能力。
- 兼容三方Docker工具。
- 无厂商锁定。

容器存储

- 支持本地和分布式数据卷管理，支持企业已有NAS和SAN存储。
- 可以通过插件扩展机制对接更多存储实现。

容器网络

- 支持容器间高性能跨宿主机网络通信。
- 支持与企业现有网络方案对接。
- 支持打通云上云下，混合管理。
- 提供插件扩展机制支持更多网络方案。

混合云

- 与阿里云公共云无缝实现混合云。
- 支持对自有数据中心和公共云容器集群的统一资源、镜像、应用和安全管理。
- 支持工作负载动态迁移。
- 支持云突发、云灾备、异地多中心管理等典型场景。

自动化运维与管理

- 基于阿里云飞天基础架构管理平台。
- 提供硬件资产管理、自动化监控、运维能力。
- 支持基础网络服务管理。

无缝对接阿里云公共云

- 与阿里云公共云无缝实现混合云。
- 提供通过专线、VPN等网络互通方式。
- 复用阿里云提供数据复制机制，可实现数据迁移、异地灾备等多种混合云业务应用场景。

6.3 产品优势

Docker Engine

- 经过双十一大规模线上业务验证。
- 阿里和Docker战略合作；提供OS和插件的安全认证。
- 提供专业化技术支持团队。

敏捷

- 应用秒级启动。
- 打造软件研发流水线，应用迭代速度提升13倍，加速企业业务迭代。
- 良好的隔离性，可以多种应用类型混合部署，系统利用率提升5-10倍。

弹性

- 密切监控应用性能，根据负载情况自动伸缩资源，从容应对突发流量。

开放

- 商业化产品和Docker社区开源保持一致。
- 开放的产品技术路线。
- 活跃的生态。

6.4 应用场景

6.4.1 容器化应用上云

传统运行在数据中心的应用如果要向云上迁移面临的挑战是应用的运行环境的适配以及应用部署方式的改变。原有数据中心的环境和云环境的差异越大，迁移的成本越高。

由于容器技术具有跨平台移植的特性，可以解决不同环境下应用运行差异问题。无论用户的应用是何种语言，何种技术栈，都能够通过容器技术封装成为跨平台可用的Docker镜像。从而将原来繁重的应用迁移变成一系列标准化的操作。

一般情况下，在数据中心里运行的虚拟机镜像无法直接迁移到云上环境，原来的镜像构建和部署脚本都要重新开发。采用容器技术可以使用容器镜像替代虚拟机镜像。利用容器技术将原有的传统应用镜像化，可以在尽量减少或者没有改变原有代码的情况下快速上云，并保证部署的一致性。

利用Docker Compose部署模版，可以描述完整的应用栈和所需云资源。容器服务扩展了Compose模板，从而支持服务治理：包括服务调度约束、服务注册/发现策略、弹性伸缩等；以及容器与外部资源的动态集成，如对象存储、文件存储等。这样可以采用应用为中心的管理方式，大大提升了部署、运维效率。

这个方案的特点是：成本低、上线速度快，并且在完成迁移后可以利用容器服务的微服务支持进行进一步微服务化改造。

6.4.2 容器化DevOps

开发和运维中交接的代码，由于同一代码在不同环境中的差异性，往往导致很高的调试成本，软件的交付和测试周期长。开发运维团队需要花费很大的精力来屏蔽系统差异，标准化交付手段。

容器技术作为软件的交付手段，具有同一应用在多个平台的可移植性，和不同应用在同一环境中的相互隔离的特点。利用这一特性，使容器成为开发环境、测试环境、生产环境的标准代码交付手段。在任何环境中运行的都是同一个容器，这样保证了由于环境不同所带来的差异被最大限度地屏蔽。利用这种标准交付手段，可以大大提高开发和运维的效率。

容器服务-专有云Agility版提供了容器化Jenkins的一键式部署模板，可以快速灵活地搭建一套CI/CD环境。同时还提供了Jenkins插件来充分发挥容器服务-专有云Agility版的能力，比如多种发布模式，保证新系统和服务上线的可靠性。比如通过灰度发布，可以灵活控制新老版本的流量切换，快速、可靠地回滚。

6.4.3 微服务支持

微服务作为云应用常见的架构模式正在被越来越多地接受。但是随着服务实例数量的快速增长，对服务治理的要求会越来越高。容器服务提供了和语言无关的微服务治理能力，您不用限定语言和开发框架，不用改变应用逻辑，就可以轻松应对微服务的管理和伸缩。所有这些能力都可以通过在编排模版中以声明的方式指定，或者在容器服务控制台中可视化地指定。

从对SLA保证角度，容器服务提供了利用自定义命令/HTTP等健康检查的能力，可以细粒度地评估应用的健康状况。容器服务还可以按照资源约束调度和再平衡，当节点故障时自动重调度，按照服务亲和性和跨可用区调度约束等机制保证可用性。

服务发现是微服务平台的一个重要能力，所有服务的启动和停止都能够自动注册到平台，服务之间可以非常容易地进行服务发现。更进一步，配合Routing Mesh，简单路由服务可以为应用提供灵活的4层、7层路由方式。

对于任何运维团队来说，日志的集中和分析，云资源的集中监控都必不可少。容器服务提供了声明式日志、监控采集方案，只需要简单声明要采集的日志目录，就可以自动收集日志。支持对接多种日志、监控存储。

配合监控服务，在您自定义弹性伸缩策略后，容器服务对应用的指标进行监控，在触发条件时按照策略进行应用弹性扩容和缩容。容器服务同时也支持集群弹性伸缩。

6.4.4 混合云

容器服务支持计算能力跨界迁移，通过容器服务可以管理本地专有云和公共云容器集群，提供了对所有环境完全一致的管理方式，您无需针对不同的其他云服务商或者IDC机器寻求特定的管理方案。使用同一套镜像和编排模板让应用可以无缝地在云间迁移。

混合云的典型场景包括：

- 场景1：应用快速在云端部署伸缩，应对大促突发流量峰值
- 场景2：异地、同城灾备
- 场景3：本地的机器开发测试，云端预发生产；或反之

针对混合云，为了提供最佳的网络实现，容器服务提供了不同的网络方案来适配不同环境。对于IDC与阿里公共云容器互通的需求，提供基于VPC的混合云互连方案，使用VPC+专线/VPN，可以实现IDC机器、IDC容器、ECS机器、ECS容器的全部互通。容器跨宿主机互连方案包括：Overlay（VXLAN）/Layer 3（Calico）等。

图 6-1: 混合云架构

6.5 基本概念

容器服务管理Docker集群所需的重要信息，包括集群、节点、容器、镜像、编排模版、服务和应用。这些元信息同时是支撑容器服务的重要概念。

集群

一个集群指容器运行所需要的资源组合，关联了若干服务器节点等资源。

节点

一台服务器（可以是虚拟机实例或者物理服务器）已经安装了Docker Engine，可以用于部署和管理容器；容器服务的Agent程序会安装到节点上并注册到一个集群上。集群中的节点数量可以伸缩。

容器

一个通过Docker镜像创建的运行时实例，一个节点可运行多个容器。

镜像

Docker镜像是容器应用打包的标准格式，在部署容器化应用时可以指定镜像，镜像可以来自于Docker Hub，阿里云容器镜像服务，容器服务的DTR，或者用户的私有Registry。镜像ID可以由镜像所在仓库URI和镜像Tag（缺省为latest）唯一确认。

编排模板

编排模板包含了一组容器服务的定义和其相互关联，可以用于多容器应用的部署和管理。容器服务支持Docker Compose模板规范并有所扩展。

服务

一组基于相同镜像和配置定义的容器，作为一个可伸缩的微服务。

7 对象存储OSS

7.1 什么是对象存储OSS

对象存储服务（Object Storage Service，简称 OSS）提供海量、安全、低成本、高可靠的云存储服务。它可以理解为一个即开即用，无限大空间的存储集群。相比传统自建服务器存储，OSS 在可靠性、安全性、成本和数据处理能力方面都有着突出的优势。使用 OSS，您可以通过网络随时存储和调用包括文本、图片、音频和视频等在内的各种非结构化数据文件。

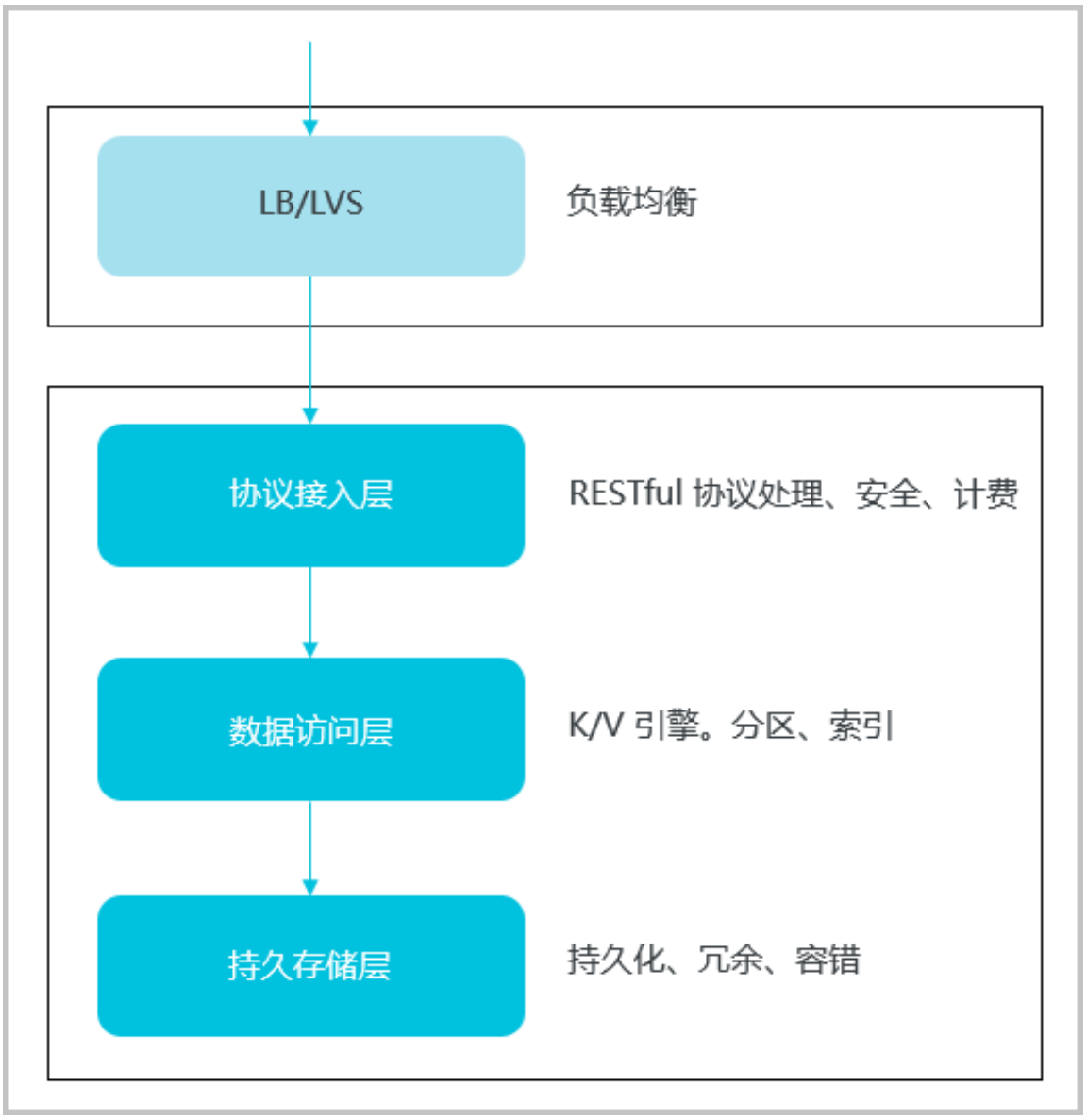
OSS 将数据文件以对象/文件（Object）的形式上传到存储空间（Bucket）中。 您可以进行以下操作：

- 创建一个或者多个存储空间。
- 每个存储空间中添加一个或多个文件。
- 通过获取已上传文件的地址进行文件的分享和下载。
- 通过修改存储空间或文件的属性或元信息来设置相应的访问权限。
- 通过云控制台执行基本和高级 OSS 任务。
- 通过 SDK 或直接在应用程序中进行 RESTful API 调用执行基本和高级 OSS 任务。

7.2 产品架构

对象存储 OSS 是构建在阿里云飞天平台上的一种存储解决方案。其基础是飞天平台的分布式文件系统、分布式任务调度等基础设施。这些基础设施提供了 OSS 以及其他阿里云服务所需的分布式调度、高速网络、分布式存储等重要特性。OSS 的架构如图 *7-1: OSS* 架构图所示：

图 7-1: OSS 架构图



- 最上层是协议接入层，负责接收用户使用 RESTful 协议发来的请求，进行安全认证。如果认证通过，用户的请求将被转发到 Key-Value 引擎继续处理；如果认证失败，直接返回错误信息给用户。
- 数据访问层负责数据结构化处理，即按照 Key 来查找或存储数据，并支持大规模并发的请求。当协调服务集群变更导致服务被迫改变运行物理位置时，可以快速协调找到接入点。
- 最底层是持久存储层，即大规模分布式文件系统。元数据存储在 Master 上，Master 之间采用分布式消息一致性协议（Paxos）保证元数据的一致性，从而实现高效的文件分布式存储和访问，保证数据在系统中有 3 个备份以及在软硬件错误发生以后的故障恢复。OSS 系统的这一设计提供了不低于 99.9% 可用性和 99.99999999% 数据可靠性。

7.3 功能特性

OSS提供的主要功能如表 *7-1: OSS主要功能*所示：

表 7-1: OSS主要功能

功能	描述
创建存储空间	在上传任何文件到 OSS 之前，您需要首先创建存储空间来存储文件。
删除存储空间	如果您不再需要存储空间，请将其删除以免产生不必要的费用。
修改存储空间读写权限	OSS 提供权限控制 ACL（Access Control List），您可以在创建存储空间的时候设置相应的 ACL 权限控制，也可以在创建之后修改 ACL。
设置静态网站托管	将存储空间配置成静态网站托管模式，并通过存储空间域名访问该静态网站。
设置防盗链	为了减少您存储于 OSS 的数据被其他人盗链而产生额外费用，OSS 支持设置基于 HTTP header 中表头字段 referer 的防盗链方法。
管理跨域资源共享	OSS 提供 HTML5 协议中的跨域资源共享 CORS 设置，帮助您实现跨域访问。
设置生命周期	定义和管理存储空间内所有对象或对象的某个子集的生命周期。设置生命周期一般用于文件的批量管理和自动碎片删除等操作。
上传文件	您可以上传任意类型文件到存储空间中。
新建文件夹	您可以像管理 Windows 文件夹一样管理 OSS 文件夹。
搜索文件	在存储空间或文件夹中搜索具有相同的名称前缀的文件。
获取文件访问地址	通过获取已上传文件的地址进行文件的分享和下载。
删除文件	删除单个文件或批量删除文件。
删除文件夹	删除单个文件夹或批量删除文件夹。
修改文件读写权限	您可以在上传文件的时候设置相应的 ACL 权限控制，也可以在上传之后修改 ACL。
管理碎片	删除存储空间内的全部或部分碎片文件。
图片服务	对保存在OSS上的图片进行格式转换、剪裁、缩放、旋转、水印、样式封装等各种处理。
API	提供 OSS支持的 RESTful API 操作和相关示例。
SDK	提供主流语言 SDK 的开发操作和相关示例。

7.4 产品优势

OSS与自建存储对比

OSS与自建存储对比如表 7-2: OSS与自建存储对比表所示。

表 7-2: OSS与自建存储对比表

对比项	对象存储 OSS	自建服务器存储
可靠性	• 服务可用性不低于 99.9%。 • 规模自动扩展，不影响对外服务。 • 数据持久性不低于 99.99999999%。 • 数据自动多重冗余备份。	• 受限于硬件可靠性，易出问题，一旦出现磁盘坏道，容易出现不可逆转的数据丢失。 • 人工数据恢复困难、耗时、耗力。
安全	• 提供企业级多层次安全防护。 • 多用户资源隔离机制，支持异地容灾机制（需要选配容灾套件）。 • 提供多种鉴权和授权机制及白名单、防盗链、主子账号功能。	• 需要另外购买清洗和黑洞设备。 • 需要单独实现安全机制。
数据处理能力	提供图片处理功能。	需要额外采购，单独部署。

方便、快捷的使用方式

提供标准的 RESTful API 接口（部分接口与Amazon S3 API兼容）、丰富的 SDK 包、客户端工具、控制台。您可以像使用文件一样方便地上传、下载、检索、管理用于 Web 网站或者移动应用的海量数据。

- 不限文件数量和大小。您可以根据所需存储量无限扩展存储空间，解决了传统硬件存储扩容问题。
- 支持流式写入和读出。特别适合视频等大文件的边写边读业务场景。
- 支持数据生命周期管理。您可以自定义将到期数据批量删除。

强大、灵活的安全机制

灵活的鉴权、授权机制。提供 STS 和 URL 鉴权和授权机制，以及白名单、防盗链、主子账号功能。

丰富的图片处理服务

支持 jpg、png、bmp、gif、webp、tiff 等多种图片格式的转换，以及缩略图、剪裁、水印、缩放等多种操作。

7.5 基本概念

本部分将向您介绍 OSS 中涉及的几个基本概念，以便于您更好地理解 OSS 产品。

对象/文件（Object）

对象是 OSS 存储数据的基本单元，也被称为 OSS 的文件。对象由元信息（Object Meta），用户数据（Data）和文件名（Key）组成。对象由存储空间内部唯一的 Key 来标识。对象元信息是一个键值对，表示了对象的一些属性，比如最后修改时间、大小等信息，同时用户也可以在元信息中存储一些自定义的信息。

根据不同的上传方式，对象的大小限制是不一样的。分片上传最大支持 48.8TB 的对象大小，其他的上传方式最大支持 5GB。

对象的生命周期是从上传成功到被删除为止。在整个生命周期内，对象信息不可变更。重复上传同名的对象会覆盖之前的对象，因此，OSS 不支持修改文件的部分内容等操作。

OSS 提供了追加上传功能，用户可以使用该功能不断地在 Object 尾部追加写入数据。

> **说明**：
> 如无特殊说明，本文档中的对象、文件称谓等同于 Object。

存储空间（Bucket）

存储空间是您用于存储对象（Object）的容器，所有的对象都必须隶属于某个存储空间。您可以设置和修改存储空间属性用来控制访问权限、生命周期等，这些属性设置直接作用于该存储空间内所有对象，因此您可以通过灵活创建不同的存储空间来完成不同的管理功能。

- 同一个存储空间的内部是扁平的，没有文件系统的目录等概念，所有的对象都直接隶属于其对应的存储空间。
- 每个用户可以拥有多个存储空间。
- 存储空间的名称在 OSS 范围内必须是全局唯一的，一旦创建之后无法修改名称。
- 存储空间内部的对象数目没有限制。

强一致性

Object 操作在 OSS 上具有原子性，操作要么成功要么失败，不会存在有中间状态的Object。OSS 保证用户一旦上传完成之后读到的 Object 是完整的，OSS 不会返回给用户一个部分上传成功的 Object。

Object 操作在 OSS 上同样具有强一致性，用户一旦收到了一个上传（PUT）成功的响应，该上传的 Object 就已经立即可读，并且数据的三份副本已经写成功。不存在一种上传的中间状态，即 read-after-write 却无法读取到数据。对于删除操作也是一样的，用户删除指定的 Object 成功之后，该 Object 立即变为不存在。

强一致性方便了用户架构设计，可以使用跟传统存储设备同样的逻辑来使用OSS，修改立即可见，无需考虑最终一致性带来的各种问题。

OSS与文件系统的对比

OSS 是一个分布式的对象存储服务，提供的是一个 Key-Value 对形式的对象存储服务。用户可以根据 Object 的名称（Key）唯一地获取该Object的内容。虽然用户可以使用类似 test1/test.jpg 的名字，但是这并不表示用户的 Object 是保存在test1 目录下面的。对于 OSS 来说，test1/test.jpg 仅仅只是一个字符串，和 a.jpg 这种并没有本质的区别。因此不同名称的 Object 之间的访问消耗的资源是类似的。

文件系统是一种典型的树状索引结构，一个名为 test1/test.jpg 的文件，访问过程需要先访问到 test1 这个目录，然后再在该目录下查找名为 test.jpg 的文件。因此文件系统可以很轻易的支持文件夹的操作，比如重命名目录、删除目录、移动目录等，因为这些操作仅仅只是针对目录节点的操作。这种组织结构也决定了文件系统访问越深的目录消耗的资源也越大，操作拥有很多文件的目录也会非常慢。

对于 OSS 来说，可以通过一些操作来模拟类似的功能，但是代价非常昂贵。比如重命名目录，希望将 test1 目录重命名成 test2，那么 OSS 的实际操作是将所有以 test1/ 开头的 Object 都重新复制成以 test2/ 开头的 Object，这是一个非常消耗资源的操作。因此在使用 OSS 的时候要尽量避免类似的操作。

OSS 保存的 Object 不支持修改（追加写 Object 需要调用特定的接口，生成的 Object 也和正常上传的 Object 类型上有差别）。用户哪怕是仅仅需要修改一个字节也需要重新上传整个 Object。而文件系统的文件支持修改，比如修改指定偏移位置的内容、截断文件尾部等，这些特点也使得文件系统拥有广泛的适用性。但另外一方面，OSS 能支持海量的用户并发访问，而文件系统会受限于单个设备的性能。

因此，将 OSS 映射为文件系统是非常低效的，也是不建议的做法。如果一定要挂载成文件系统的话，建议尽量只做写新文件、删除文件、读取文件这几种操作。使用 OSS 应该充分发挥其优点，即海量数据处理能力，优先用来存储海量的非结构化数据，比如图片、视频、文档等。

8 块存储EBS

8.1 什么是弹性块存储EBS

弹性块存储（Elastic Block Storage，简称EBS）是基于分布式文件系统实现的一个高可靠、高可用的块设备存储服务。

它可以部署在通用X86架构的服务器上，将多台服务器的内置本地硬盘组成虚拟的存储资源池，对外提供块存储访问能力。

EBS提供基于标准接口协议的开放 API，满足客户对接Openstack云平台的业务需求，支持KVM、容器等计算虚拟化技术。

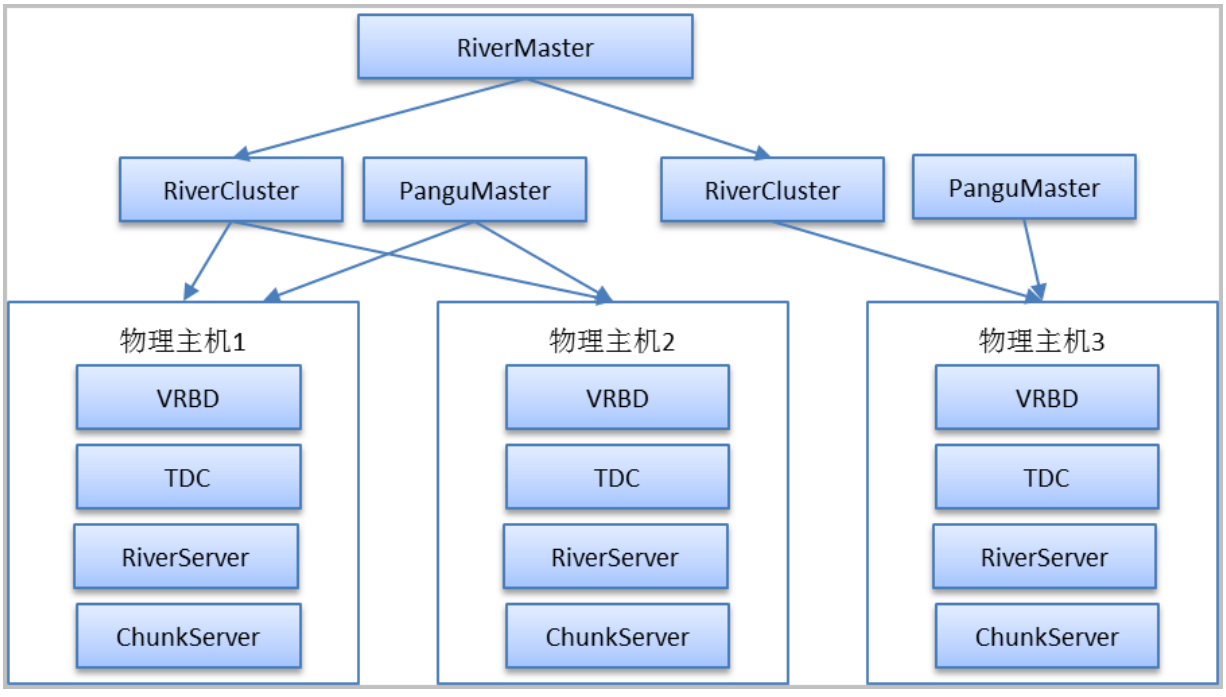
EBS支持快照、镜像、多租户、自动精简配置等高级数据服务功能，支持从最小规模无缝弹性扩展存储资源池，支持HDD机械硬盘和SSD闪存，满足不同业务模型下对存储性能和容量的多样性要求。

8.2 产品架构

8.2.1 技术架构

块存储的技术架构如图 *8-1: EBS软件架构*所示。

图 8-1: EBS软件架构



其中，各组件及其功能如表 *8-1:* 各组件功能说明所示。

表 8-1: 各组件功能说明

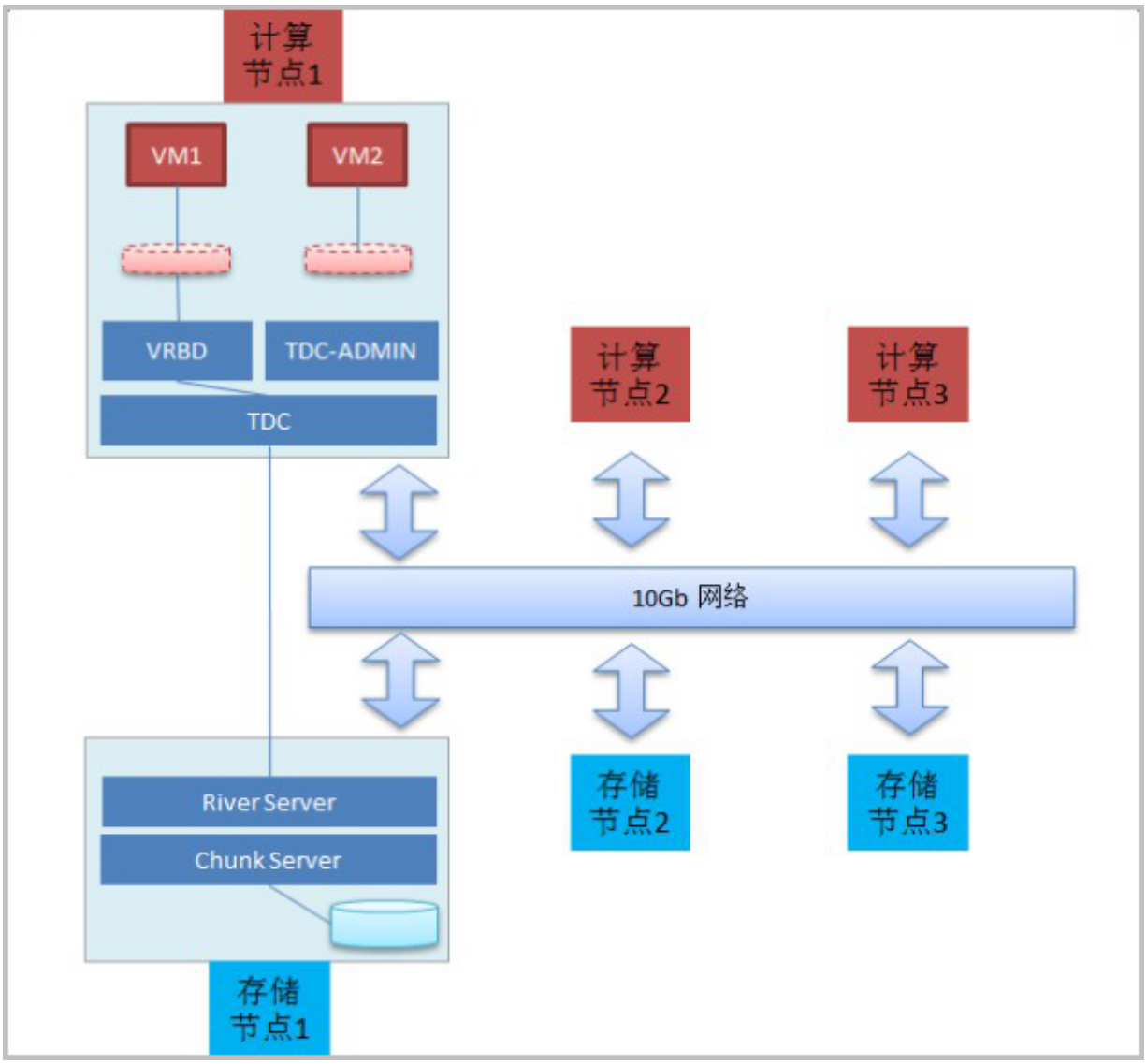
组件	功能
VRBD	虚拟块设备驱动，部署在计算节点，通过VRBD能创建虚拟的块设备，能够被挂载到物理主机上，能进行IO读写，完全和物理介质的磁盘一样的使用体验。
TDC	TDC负责将后端存储和前端组件如VRBD桥接起来，承载块设备IO，同时提供形态丰富的CLI接口，如磁盘的创建、销毁，快照的创建与回滚等。
PanguMaster	盘古中最重要的角色，维护了文件和数据块之间的映射quota数据、chunkserver元数据、和checkpoint等。盘古通过多master机制保证master的高可用性，同时使得盘古具备热升级的能力。master之间log的同步和主master的选举采用了Paxos算法来保证其一致性。
ChunkServer	运行在数据物理服务器上，主要的职责是管理本地的硬盘和支撑对硬盘的读写删操作。
RiverMaster	在块存储系统中管控角色，负责磁盘在集群见的调度，包括磁盘创建的的调度、动态负载调度等，以及相关的管理工作，如存储库存水位管理等。
RiverCluster	负责管理一个集群，RiverCluster通过心跳维护RiverServer的状态。当RiverServer不可服务时，负责将RiverServer的磁盘动态迁移到另一台RiverServer上。
RiverServer	运行在每台物理主机上，RiverServer负责块层的IO读写处理，同时将IO请求转化后投递到Pangu文件层。

8.2.2 部署架构

阿里云EBS块存储系统采用分离部署模式，即计算端节点只部署计算端的软件组件，存储端节点只部署存储端的软件组件。

如图 *8-2:* 分离部署模式所示，分离部署模式中，计算节点仅部署与计算有关的软件组件，如VRBD、TDC和TDC_ADMIN等，存储节点仅部署与存储有关的软件组件如RiverServer和ChunkServer等。

图 8-2: 分离部署模式



8.3 功能特性

本节主要介绍EBS的功能和特性。

8.3.1 分布式文件系统

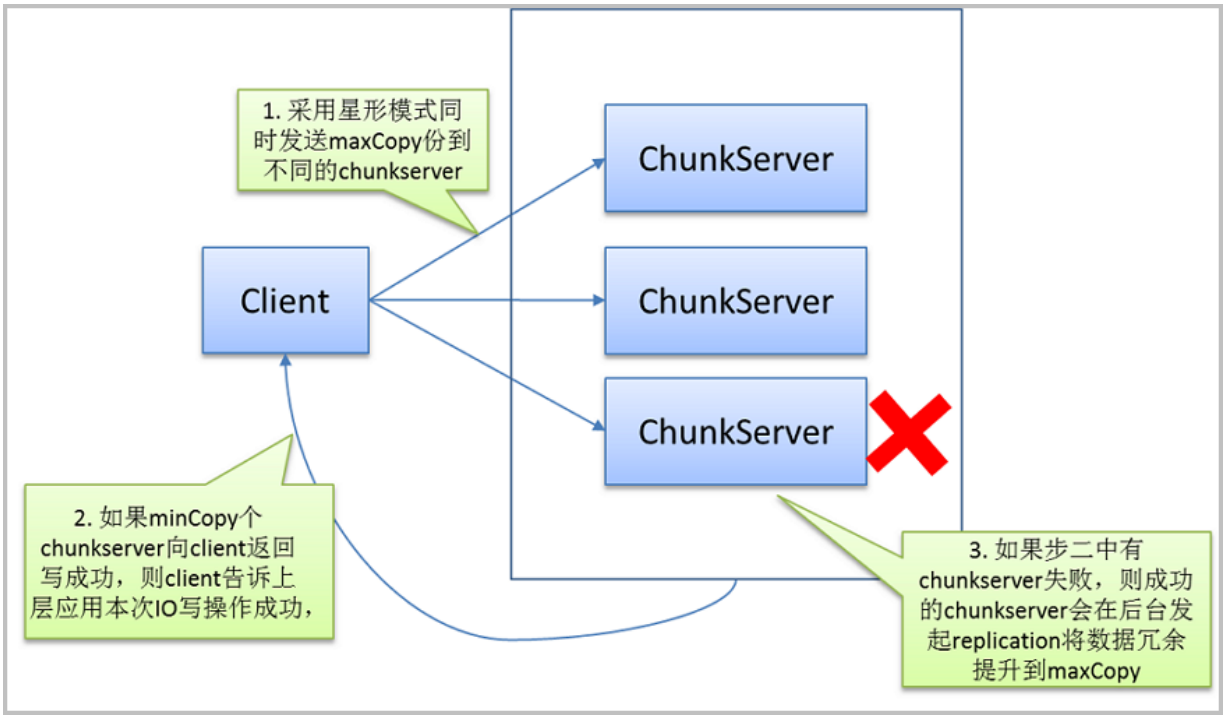
盘古采用多副本机制实现分布式文件系统。

如图 *8-3:* 多副本机制所示，盘古多副本机制的内部实现原理如下：

1. Client采用星形模式，同时向服务器发送maxCopy份副本到不同的chunkserver。
2. 如果minCopy个chunkserver向Client返回写成功，则Client报告上层应用，本次IO写操作成功。

3. 如果上一步中有chunkserver失败，则成功的chunkserver会在后台发起replication，将数据冗余提升到maxCopy。

图 8-3: 多副本机制

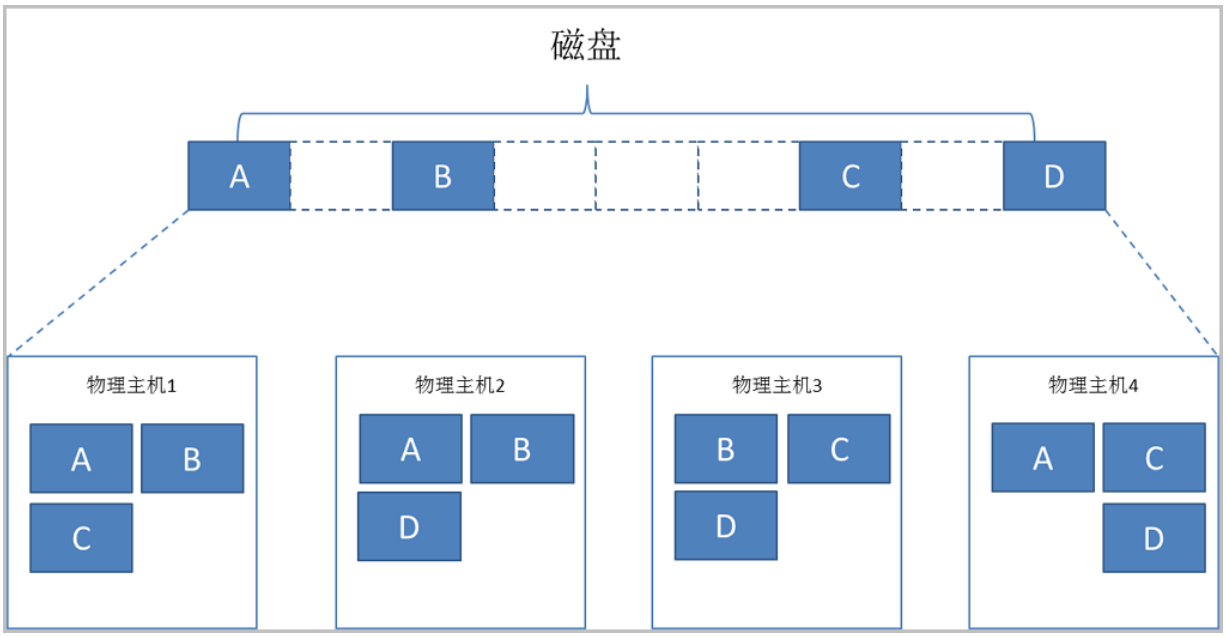


8.3.2 自动精简装置

EBS默认支持精简配置，以降低存储的成本，提高资源利用率。

如图 *8-4:* 自动精简装置所示，虽然磁盘有9个块的容量，用户磁盘真正写入的只有4个块。所以，落在底层盘古，也并不是按照9个块空间进行分配与存储，而只是存储了真实有数据的块，并进行索引维护，从而实现了精简配置。

图 8-4: 自动精简装置



8.3.3 快照

EBS卷的快照具备如下特点：

- 快照通过增量技术实现，创建的第一个快照是卷的数据的全量备份，后续创建的快照均是自上次卷的数据量，进行增量备份。
- 快照是按照一定大小进行切割，并将数据有变化的数据块压缩后，备份到对象存储系统中。
- 快照的指令下发是秒级完成，快照数据同步是在系统后台进行的，不需要停止卷的运行状态。
- 快照删除后，不会影响后续快照的数据完备性，被删除的快照的数据块，只有引用计数为0时，才会被回收。

8.3.4 数据重建

盘古具备强大的数据保护和智能恢复机制。

数据存储时已被切片打散到多个节点上，这些分片数据通过算法被分配在不同的存储节点、不同机柜之间以提供最大数据安全保障，同时数据存储时采用多副本技术，支持两副本或三副本，数据会自动保存多份，每一个分片的不同副本也被分散保存到不同的存储节点上。

在硬件发生故障导致数据不一致时，盘古一方面通过异步机制来保证服务的可用性，另一方面通过优化的自检机制，能够快速比较不同节点上的副本分片，自动发现数据故障，并在发现故障后启动数据修复机制。

在后台修复数据，由于数据被分散到多个不同的存储节点上保存，数据修复时，在不同的节点上同时启动修复，每个节点上只需修复一小部分数据，多个节点并行工作，配合精准的流控机制不仅能充分利用剩余网络资源，还可以有效避免单个节点修复大量数据所产生的性能瓶颈，对上层业务的影响做到最小化。

8.4 典型业务场景

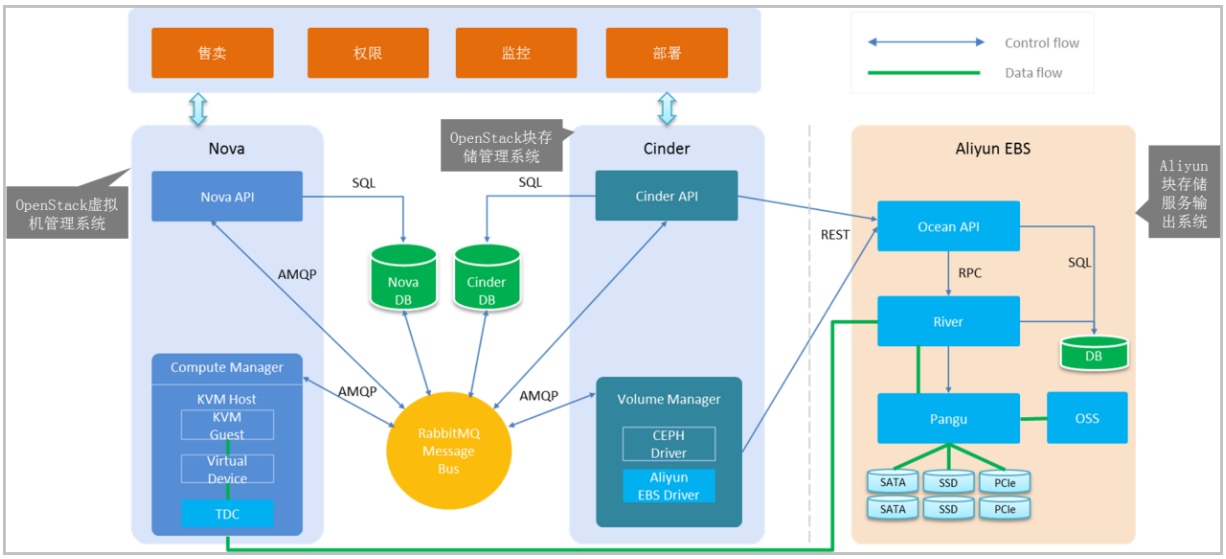
典型业务场景主要包括OpenStack场景和Docker场景。

8.4.1 OpenStack场景

阿里云EBS块存储服务作为一个基础的云存储基础设施，能很敏捷地与现有源云平台如OpenStack进行集成。

如图 *8-5: OpenStack场景*所示，OpenStack系统中的存储子系统是Cinder，阿里云为Cinder开发了Aliyun EBS Plugins for cinder，将阿里云EBS块存储和OpenStack进行无缝集成。这样用户只需关注计算端的调度和管控，块存储服务由阿里云EBS来承载，并向上层提供丰富的功能，如磁盘类功能、快照类功能、镜像类功能、运维类功能及监控类功能。

图 8-5: OpenStack场景

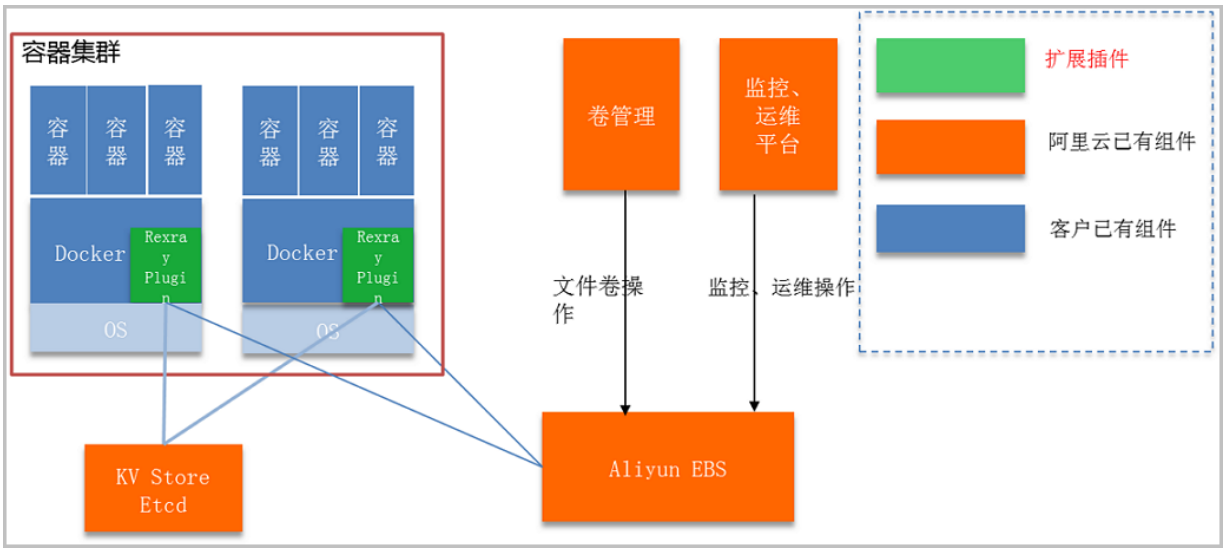


8.4.2 Docker场景

阿里云EBS同样能支持Docker容器技术。

如图 *8-6: Docker场景*所示，通过实现一个基于阿里云EBS的plugin，即可将阿里云EBS和Docker系统无缝集成在一起。

图 8-6: Docker场景



9 表格存储Table Store

9.1 什么是表格存储

表格存储（Table Store）是构建在阿里云飞天分布式系统之上的 NoSQL 数据存储服务，提供海量结构化数据的存储和实时访问。

- 表格存储以实例和表的形式组织数据，通过数据分片和负载均衡技术，达到规模的无缝扩展。
- 表格存储向应用程序屏蔽底层硬件平台的故障和错误，能自动从各类错误中快速恢复，提供非常高的服务可用性。
- 表格存储管理的数据全部存储在 SSD 中并具有多个备份，提供了快速的访问性能和极高的数据可靠性。

9.2 产品优势

扩展性

- 动态调整预留读/写吞吐量

  在创建表的时候，应用程序可以根据业务访问的情况来配置预留读/写吞吐量。表格存储根据表的预留读/写吞吐量进行资源的调度和预留。在使用过程中，还可以根据应用情况动态修改预留读/写吞吐量。

- 无限容量

  表格存储中表的数据量没有上限，随着表数据量的不断增大，表格存储会进行数据分区的调整，从而为该表配置更多的存储。

数据可靠性

表格存储将数据的多个备份存储在不同机架的不同机器上，并会在备份失效时进行快速恢复，提供了极高的数据可靠性。

高可用性

通过自动的故障检测和数据迁移，表格存储对应用屏蔽了机器和网络的硬件故障，提供了高可用性。

管理便捷

应用程序无需关心数据分区的管理、软硬件升级、配置更新、集群扩容等繁琐的运维任务。

访问安全性

表格存储对应用程序的每一次请求都进行身份认证和鉴权，以防止未授权的数据访问，确保数据访问的安全性。

强一致性

表格存储保证数据写入强一致，写操作一旦返回成功，应用就能立即读到最新的数据。

灵活的数据模型

表格存储的表无固定格式要求，每行的列数及不同行同名列的类型可以不相同，支持多种数据类型，如 Integer、Boolean、Double、String 和 Binary。

监控集成

您可以从表格存储云控制台实时获取每秒请求数、平均响应延时等监控信息。

9.3 功能特性

针对表的功能和操作

- ListTable：列出实例下的所有表。
- CreateTable：创建表。
- DeleteTable：删除表。
- DescribeTable：获取表的属性信息。
- UpdateTable：更新表的预留读/写吞吐量配置。
- ComputeSplitPointsBySize：将全表的数据在逻辑上划分成接近指定大小的若干分片，返回这些分片之间的分割点以及分片所在机器的提示。

针对数据的功能和操作

- 单行操作
  - GetRow：读取单行数据。
  - PutRow：新插入一行。如果该行内容已经存在，先删除旧行，再写入新行。
  - UpdateRow：更新一行。应用可以增加、删除一行中的属性列，或者更新已经存在的属性列的值。如果该行不存在，则新增一行。
  - DeleteRow：删除一行。
- 批量操作
  - BatchGetRow：批量读取一张或者多张表的多行数据。

  - BatchWriteRow：批量插入、更新、删除一张表或者多张表的多行数据。
- 范围读取
  - GetRange：读取表中某个范围内的数据。

针对写的功能

- 原子性

  PutRow、UpdateRow、DeleteRow 操作的结果保证原子性，即：要么全部成功，要么全部失败，不会存在中间状态。
- 强一致性

  应用程序获得写操作成功的响应后，本次操作的修改会立即生效，应用程序可以读取到该行最新的修改。

另外，表格存储提供 BatchWriteRow 操作，对多个单行写操作进行聚集，应用程序可以将多个 PutRow、UpdateRow、DeleteRow 操作放到一个 BatchWriteRow 操作中。需要特别注意的是，BatchWriteRow 操作是多个单行写操作的聚集，本身不保证原子性，可能会出现部分行操作执行成功，部分行操作执行失败的情况，但是 BatchWriteRow 的子操作具有原子性。

9.4 使用限制

表 *9-1:* 使用限制为表格存储的使用限制汇总，部分限制范围表明用户能够使用的最大值，而不是建议值。为保证更好的性能，请合理设计表结构和单行数据大小。

表 9-1: 使用限制

限制项	限制范围	说明
一个阿里云用户账号下可以保有实例数	不超过 10	如有需求提高上限，请联系管理员。
一个实例中表的个数	不超过 64	如有需求提高上限，请联系管理员。
实例名字长度	3~16 Bytes	字符集为 [a-z、A-Z、0-9] 和连字符（-），首字符必须是字母且末尾字符不能为连字符（-）。
表名长度	1~255 Bytes	字符集为 [a-z、A-Z、0-9] 和下划线（_），首字符必须是字母或下划线（_）。
列名长度限制	1~255 Bytes	字符集为 [a-z、A-Z、0-9] 和下划线（_），首字符必须是字母或下划线（_）。

限制项	限制范围	说明
主键包含的列数	1~4	至少 1 列，至多 4列。
String 类型主键列列值大小	不超过 1 KB	单一主键列 String 类型的列列值大小上限 1 KB。
String 类型属性列列值大小	不超过 2 MB	单一属性列 String 类型的列列值大小上限 2 MB。
Binary 类型主键列列值大小	不超过 1 KB	单一主键列 Binary 类型的列列值大小上限 1 KB。
Binary 类型属性列列值大小	不超过 2 MB	单一属性列 Binary 类型的列列值大小上限 2 MB。
一行中属性列的个数	不限制	不限制单一行拥有的属性列个数。
单行数据大小	不限制	不限制单一行中所有列名与列值总和大小。
读请求中 columns_to_get 参数的列的个数	0~128	读请求一行数据中获取的列的最大个数。
单表 UpdateTable 的次数	上调：无限制 下调：无限制	需要遵循单表的调整频率限制。
单表 UpdateTable 的频率	每 2 分钟 1 次	单表在 2 分钟之内，最多允许调整 1 次预留读/写能力值。
BatchGetRow 一次操作请求读取的行数	不超过 100	-
BatchWriteRow 一次操作请求写入行数	不超过 200	-
BatchWriteRow 一次操作的数据大小	不超过 4 MB	-
GetRange 一次返回的数据	5000 行或者 4 MB	一次返回数据的行数超过 5000 行，或者返回数据的数据大小大于 4 MB。以上任一条件满足时，超出上限的数据将会按行级别被截掉并返回下一行数据主键信息。
一次 HTTP 请求 Request Body 的数据大小	不超过 5 MB	-

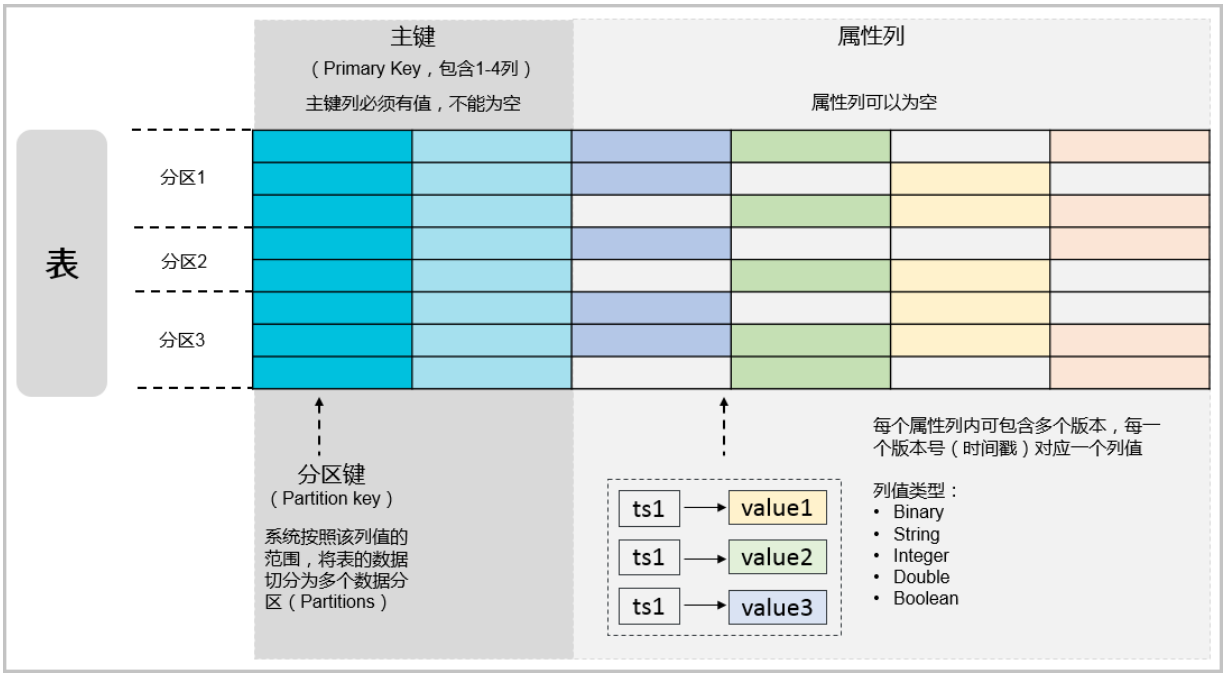
9.5 基本概念

9.5.1 数据模型

表格存储的数据模型概念包括表、行、主键和属性。

数据模型图如图 *9-1:* 数据模型图所示：

图 9-1: 数据模型图



- 表是行的集合，行由主键和属性组成。
- 主键列和属性列均由名称和值组成。
- 所有行都必须包含主键列，其主键列的数目和名称必须相同。
- 每行包含的属性列的数目可以不固定，名字和数据类型也可以不同。
- 与主键列不同，每个属性列可以包含多个版本，每个版本号（时间戳）对应一个列值。

> **说明**：
> 时间戳是从 1970-01-01 00:00:00 UTC 时间到当前写入时间的毫秒数。

如下例子展示了同一张表中的两行。

表 9-2: 示例

ID	Type	ISBN	PageCount	Length
‘4776’	timestamp = 1466676354000, value = ‘Book’	timestamp = 1466676354000, value = ‘123*45678912345’	timestamp = 1466676354000, value = 666	空
‘6555’	timestamp = 1466676354000, value = ‘Music’	空	空	timestamp = 1466676354000, value = 400; timestamp = 1466762754000, value = 500

上述表格的含义为：

- ID 是表的主键，ID 为 ‘4776’ 和 ‘6555’ 的行拥有不同的属性，它们可以被存在一张表中。
- ID 为 ‘4776’ 行的 Type 属性列只有一个版本数据，版本号为 1466676354000 的数据为 ‘Book’ 。
- ID 为 ‘6555’ 行的 Length 属性列有两个版本数据，版本号为 1466676354000 的数据为 400，版本号为 1466762754000 的数据为 500。

9.5.2 最大版本数

最大版本数（Max Versions）是数据表的一个属性，表示该数据表中的属性列能够保留多少个版本的数据。当一个属性列的版本个数超过 Max Versions 时，最早的版本将被异步删除。

建表后，您可以通过 UpdateTable 接口动态更改数据表的 Max Versions。

说明：

- 超过 Max Versions 的数据版本为无效数据，即使数据还没有被真正删除，该数据对用户已经不可见，无法读出。
- 当调小 Max Versions 时，如果数据版本个数超过新设的 Max Versions，最早的版本会被系统异步删除。
- 当调大 Max Versions 时，如果以前版本个数超过旧的 Max Versions 但还没有被系统删除的，数据会被重新读出来。

9.5.3 数据生命周期

数据生命周期（Time to live，简称 TTL）是数据表的一个属性，即数据的存活时间，单位为秒。表格存储会在后台对超过存活时间的数据进行清理，以减少用户的数据存储空间，降低存储成本。

- TTL 由用户在建表时进行设置，如果希望数据永不过期，将其设置为 -1。
- 建表后，可以通过 UpdateTable 接口动态更改 TTL。
- TTL 的单位为秒，例如期望过期时间为 30 天，TTL 应设置为 2592000（即 30 * 24 * 3600）。

假设数据表的 TTL 设置为 86400（一天），在 2016-07-21 00:00:00 UTC 时，该数据表上所有版本号小于 1468944000000（除以 1000 换算成秒之后即 2016-07-20 00:00:00 UTC）的属性列都将过期，系统会自动清理这些过期的数据。

说明：

- 超过 TTL 的过期数据为无效数据，即使数据还没有被真正删除，该数据对用户已经不可见，无法读出。
- 当调小 TTL 时，可能会有数据因为 TTL 变小而过期，这部分数据会被系统异步删除。
- 当调大 TTL 时，如果有版本号在上个 TTL 之外的数据还没有被系统删除，数据会被重新读出。

9.5.4 有效版本偏差

有效版本偏差（Max Version Offset）是数据表的一个属性，单位为秒。

为了防止非期望的写入，服务端在处理写请求时会对属性列的版本号进行检查。当版本号小于当前写入时间减去 Max Version Offset，或者大于等于当前写入时间加上 Max Version Offset 的值时，该行数据写入失败。

属性列的有效版本范围为：[数据写入时间 - 有效版本偏差，数据写入时间 + 有效版本偏差)。数据写入时间为 1970-01-01 00:00:00 UTC 时间到当前写入时间的秒数。属性列版本号为毫秒，其除以 1000 换算成秒之后必须属于这个范围。

例如，当数据表的有效版本范围为 86400（一天），在 2016-07-21 00:00:00 UTC 时，只能写入版本号大于 1468944000000（换算成秒之后即 2016-07-20 00:00:00 UTC）并且小于 1469116800000（换算成秒之后即 2016-07-22 00:00:00 UTC）的数据。当某一行的某个属性列版本号为 1468943999000（换算成秒之后即 2016-07-19 23:59:59 UTC，小于一天）时，该行数据写入失败。

- 建数据表时，用户若不设置有效版本偏差，将使用默认值 86400。

- 建表后，可以通过 UpdateTable 接口动态更改有效版本偏差。
- 有效版本偏差为非 0 值，可以大于 1970-01-01 00:00:00 UTC 时间到当前时间的秒数。

9.5.5 主键和属性

主键

- 主键是表中每一行的唯一标识。主键由 1 到 4 个主键列组成。
- 创建表的时候，必须明确指定主键的组成、每一个主键列的名字和数据类型以及它们的顺序。
- 主键列的数据类型只能是 String、Integer 和 Binary。如果为 String 或者 Binary 类型，长度不超过 1 KB。

分区键

组成主键的第一个主键列又称为分区键。表格存储会根据表中每一行分区键的值所属的范围自动将这一行数据分配到对应的分区和机器上，以达到负载均衡的目的。

具有相同分区键的行属于同一个数据分区，一个分区可能包含多个分区键。分区键是最小的分区单位，一个分区键下的数据无法再做切分。为了防止分区过大无法切分，单个分区键下所有行的大小总和建议不超过 1 GB。

表格存储服务会根据特定的规则对分区进行分裂和合并，以达到更好的负载均衡。这个过程是自动的，应用程序无需关心。

属性

属性存放行的数据。每一行包含的属性列个数没有限制。

版本号

在一个属性列上，当写入的版本数超过数据表的最大版本数时，较早版本的数据会被删除，只保留最新的 Max Versions 的版本数。

在写入数据时可以指定属性列的版本号，如果不指定版本号，服务端会将当前时间的毫秒单位时间戳（从 1970-01-01 00:00:00 UTC 计算起的毫秒数）作为属性列生成版本号。比如属性列版本号为 1468944000000（即 2016-07-20 00:00:00 UTC），当数据表的 TTL 设置为 86400（一天）时，该版本的数据将会在 2016-07-21 00:00:00 UTC 过期，随后会被后台系统自动删除。

读取一行数据时，可以指定每列最多读取多少版本或者读取的版本号范围。

说明：

- 版本号的单位为毫秒，在进行 TTL 比较和有效版本偏差计算时，需要除以 1000 换算成秒。
- 当数据的版本号（即时间戳）完全由服务端决定时，写入的数据在写入后经过 TTL 秒后会被系统清理。
- 为了防止无效的写入，写入过期数据将会直接失败。

  例如在 2016-07-21 00:00:00 向 TTL 为 86400 的数据表中写入版本号小于 1468944000000（即 2016-07-20 00:00:00 UTC）的数据将会直接失败。
- 为了防止错误的写入，写入的属性列的版本号换算成秒后，需要在 [数据写入时间-有效版本偏差，数据写入时间+有效版本偏差) 的范围内。

列名的命名规范

主键列和属性列遵循如下命名规范：

- 必须由英文字母、数字或下划线（_）组成
- 首字符必须为英文字母或下划线（_）
- 大小写敏感
- 长度在 1~255 个字符之间

列值类型

表格存储支持 5 种类型的列值。

表 9-3: 列值类型

数据类型	定义	是否可为主键	大小限制
String	UTF-8，可为空	是	为主键列时最大为 1 KB，为属性列时请参考使用限制
Integer	64 bit，整型	是	8 Bytes
Double	64 bit，Double 类型	否	8 Bytes
Boolean	True/False，布尔类型	否	1 Byte
Binary	二进制数据，可为空	是	为主键列时最大为 1 KB，为属性列时请参考使用限制

9.5.6 读/写吞吐量

读/写吞吐量的单位为读服务能力单元和写服务能力单元，简称 CU （ Capacity Unit ），是数据读写操作的最小计费单位。

- 1 单位读能力表示从数据表中读一条 4 KB 数据。
- 1 单位写能力表示向数据表写一条 4 KB 数据。
- 操作数据大小不足 4 KB 的部分向上取整，如写入 7.6 KB 数据消耗 2 单位写能力，读出 0.1 KB 数据消耗 1 单位读能力。

应用程序通过 API 进行表格存储读写操作时，会消耗对应的写服务能力单元和读服务能力单元。

预留读/写吞吐量

预留读/写吞吐量是表的一个属性。应用程序在创建表的时候，可以为该表指定预留读/写吞吐量。预留读写吞吐量的配置不影响该数据表的访问性能和服务能力。

- 预留读/写吞吐量可以设置为 0。
- 当预留读/写吞吐量大于 0 时，表格存储根据该配置为表分配和预留相应的资源，从而获得更低的资源使用成本。
- 当预留读/写吞吐量大于 0 时，即使没有读写请求也会进行计费，所以表格存储限制用户能够自行设置的单表预留读写吞吐量最大为 5000 （ 读和写分别不超过 5000 ）。如果用户有单表预留读写吞吐量需要超出 5000 的需求，可以联系管理员提高预留读写吞吐量。
- 不存在的表将被视作预留读和预留写吞吐量均为 0，访问不存在的表将根据操作类型消耗 1 个按量读 CU 或者 1 个按量写 CU。

应用程序可以通过 UpdateTable 操作动态修改表的预留读/写吞吐量配置。

按量读/写吞吐量

按量读/写吞吐量是数据表在每一秒钟实际消耗的读/写吞吐量中超出预留读/写吞吐量的部分，统计周期为 1 秒。

假如某数据表设置的预留读吞吐量为 100，某 1 秒内读操作实际消耗 120 读吞吐量，则这 1 秒内消耗的按量读吞吐量为 20。如果数据表设置的预留读吞吐量为 0，则该数据表上所有的读访问消耗的读吞吐量均为按量读吞吐量。

由于按量读/写吞吐量的模式无法预估需要为数据表预留的计算资源，表格存储需要提供足够的服务能力以应对突发的访问高峰，所以按量吞吐量的单价高于预留吞吐量的单价。合理设置数据表的预留吞吐量能够有效地降低使用成本。

说明：

由于按量读/写吞吐量无法准确估计需要预留的资源，在某些极端访问情况下，若单个分片键每秒钟的访问需要消耗 10000 CU，表格存储可能会返回 OTSCapacityUnitExhausted 错误给应用程序。此时，应用程序需要使用退避重试等策略来减少访问该表的频率。

9.5.7 实例

实例是用户使用和管理表格存储服务的实体，用户在开通表格存储服务之后，需要通过云控制台来创建实例，然后在实例内进行表的创建和管理。实例是表格存储资源管理的基础单元，表格存储对应用程序的访问控制和资源计量都在实例级别完成。

用户可以为不同的业务创建不同的实例来管理相关的表，也可以为同一个业务的开发测试和生产环境创建不同的实例。

表格存储允许一个账号最多创建 10 个实例，每个实例内最多创建 64 张表。如果您需要增加限额，请联系管理员。

实例的名字在单个节点内必须唯一，用户可以在不同的节点内创建名字相同的实例。实例命名规范如下：

- 必须由英文字母、数字或连字符（-）组成
- 首字符必须为英文字母
- 末尾字符不能为连字符（-）
- 大小写不敏感
- 长度在 3~16字节之间

目前表格存储支持两种实例规格：高性能实例和容量型实例。

说明：

创建实例时请谨慎选择规格类型，创建后无法修改。

两种实例规格在功能上保持一致，都能够支持单表 PB 级别的数据量，主要区别在于使用成本及适用场景上。

- 高性能实例

  高性能实例能够提供百万级读写 TPS，单行的读写操作的平均延时为单个毫秒，适用于对读写性能和并发都要求非常高的场景，例如游戏、金融风控、社交应用、 推荐系统、舆情监控等。

- 容量型实例

容量型实例能够提供极高的写吞吐量，能够提供接近于高性能实例的写性能以及更低的使用成本，但读性能和并发能力均低于高性能实例，适用于写多读少、对读性能不敏感但对成本较为敏感的业务，比如日志监控数据、车联网数据、设备数据、时序数据以及物流数据。

说明：

容量型实例不支持预留读/写吞吐量，所有的读写访问均按照按量读/写吞吐量进行计费。

10 文件存储NAS

10.1 什么是文件存储

阿里云文件存储（Network Attached Storage，简称NAS）是面向阿里云ECS实例、HPC和Docker等计算节点的文件存储服务，提供标准的文件访问协议，您无需对现有应用做任何修改，即可使用具备无限容量及性能扩展、单一命名空间、多共享、高可靠和高可用等特性的分布式文件系统。

您创建NAS文件系统实例和挂载点后，即可在ECS、HPC和Docker等计算节点内通过标准的NFS协议挂载文件系统，并使用标准的Posix接口对文件系统进行访问。多个计算节点可以同时挂载同一个文件系统，共享文件和目录。

10.2 功能特性

无缝集成

支持NFSv3及NFSv4协议，使用标准的文件系统语义访问数据，主流的应用程序及工作负载无需任何修改即可无缝配合使用。

共享访问

多个计算节点可以同时访问同一个文件系统实例，非常适合跨多个ECS、HPC或Docker实例部署的应用程序访问相同数据来源的应用场景。

弹性伸缩

单文件系统容量上限10 PB，充分满足弹性伸缩需求。

安全控制

通过用户隔离（经典网络）、文件系统标准权限控制、权限组访问控制和RAM主子账号授权等多种安全机制，保证文件系统数据安全万无一失。

线性扩展的性能

可为应用工作负载提供高吞吐量与高IOPS、低时延的存储性能，同时性能与容量成线性关系，可满足业务增长需要更多容量与存储性能的诉求。

10.3 产品优势

多共享

同一个文件系统可以同时挂载到多个计算节点上，共享访问，节约大量拷贝和同步成本。

高可靠

提供99.99999999%的数据可靠性，相比自建NAS存储，可以大量节约维护成本，降低数据安全风险。

弹性扩展

单个文件系统容量上限达10PB，轻松应对业务的随时扩容和缩容。

高性能

单个文件系统吞吐性能随存储量线性扩展，相比购买高端NAS存储设备，大幅降低成本。

易用性

支持NFSv3和NFSv4协议，无论是在ECS实例内，还是在HPC和Docker等计算节点中，都可通过标准的Posix接口对文件系统进行访问操作。

10.4 应用场景

企业办公文件共享

企业员工办公需要访问和共享相同的数据集，管理员可创建NAS文件系统，为组织中的个人提供数据访问，并可设置文件或目录级别的用户和用户组权限。

数据备份

用户希望将线下机房的数据备份到云上，同时要求云上的存储服务兼容标准的文件访问接口。

服务器日志共享

将多个计算节点上的应用服务器日志存放在共享的文件存储上，方便后续的日志集中处理与分析。

10.5 基本概念

挂载点

挂载点是文件系统实例在经典网络内的一个访问目标地址，每个挂载点都对应一个域名，用户mount时通过指定挂载点的域名来挂载对应的NAS文件系统到本地。

权限组

权限组是NAS提供的白名单机制，通过向权限组内添加规则来允许IP地址或网段以不同的权限访问文件系统。

> **说明**：
> 每个挂载点都必须与一个权限组绑定。

授权对象

授权对象是权限组规则的一个属性，代表一条权限组规则被应用的目标。在经典网络内，授权对象只能是一个单独的IP地址（一般为ECS实例的内网IP地址）。