

云 用
之 为
道 要

云启录

政企用云最佳实践

主创作者

王克强 丁 杰 孙 岩 余从佳 曾 迟
王本森

责任编辑

史忠钦 丁 杰 茆亮亮 吴鹏程 曹 霖
王宗瑞 肖 鹏 张 倩 许家法 程 霖
冉 奇 曾 迟 缪 锐 赵俊阳 刘 晓
赵可可 王若飞 庞兴华 李 峰 王 懿
张 亮 斯君辉 彭 干

产品与运营

谢 宁 康 静 陈佩琪

设 计

王 婉 余 航 刘友娇 徐小祥 黄 斌



一本政企上云的“答案之书”

阿里云专有云服务团队10余年最佳实践沉淀

阿里云专有云服务团队·著

云启录

政企用云最佳实践

云 用
之 为
道 要

目录

序 言 /	
云之道 用为要	1
第一章 /	
上云规划篇	
第一节 云平台的“MBTI” --- 通过合理调研做好上云前规划	3
第二节 上云的“6R” 模型 --- 解锁上云策略的底层代码	8
第三节 上云的“号码牌” --- 明晰应用上云的优先侧重	12
第四节 上云的“明算帐” --- 如何进行上云的成本评估	16
第五节 云商城的“选购指南” --- 如何科学地进行云产品选型	18
第二章 /	
上云实施篇	
第一节 “因地制宜”推进应用上云 --- 上云实施的关键因素分析及迁移方案设计	31
第二节 应用上云最全指南 --- 从制定方案到最佳实践	47
第三节 数据上云最全指南 --- 从制定方案到最佳实践	65
第四节 “一云多芯”成为确定性趋势 --- 如何解决跨平台上云的技术挑战	107

第三章 / 深度用云篇

第一节	不断深化用云之路 -- 从传统管理到云原生运营	114
第二节	IT 架构的全方位“体测” -- 云化应用架构模型“4L6D”全解析	125
第三节	云上“攻坚” -- 行业头部客户核心业务系统用云最佳实践	135
第四节	云上数据运营	146
	（一）云上数据仓库的基石建设 -- 做好智能建模和数据血缘管理	146
	（二）让云上的数据流动起来 -- 数仓分层设计与数据流转	162
	（三）大数据保质也保量 -- 大数据的质量管理与治理	177
	（四）解锁数据的价值 -- 如何构建数据服务能力	189
第五节	重点行业数据云化治理最佳实践	193
	（一）传统数据库到云原生数据库的“丝滑转身” -- 某政企客户打造数据库上云最佳实践	193
	（二）数仓升级的“精准施策” -- 某券商数据仓库升级最佳实践	202

第四章 / 稳定建设篇

第一节	业务稳定性的“居安思危” -- 面向失败设计	211
	（一）系统高可用架构设计	211
	（二）系统容量规划设计	221
	（三）系统容错设计	227
	（四）系统容灾设计	236
第二节	系统运行的“明察秋毫” -- 可观测能力建设	245
	（一）让云为应用所用 -- 打造云 + 应用一体化运维能力	245



	（二）观全局以善决策 -- 打造“云 + 应用”一体化全景可观测能力	255
第三节	灾难应对的“预则立” -- 应急预案体系建设	260
第四节	系统灾备的“未雨绸缪” -- 容灾多活能力建设	267
	（一）灾备管理平台介绍	270
	（二）多活容灾产品介绍	276
	（三）主要的容灾多活解决方案	285
	（四）某医保局异地容灾建设最佳实践	295
	（五）某政企头部客户异地多活单元化方案最佳实践	302
第五节	故障应急的“枕戈待旦” -- 实训演练能力建设	308
	（一）以故障演练和实训推进运维能力提升	309
	（二）实训演练工具平台“玄重尺”能力一览	315
	（三）云平台实训演练的最佳实践	317
第六节	云上安全体系建设	323

第五章 / 大模型赋能篇

第一节	AI Stack 大模型训推软硬一体方案	345
第二节	MaaS 引擎驱动 AI 生产力革命	350
第三节	AI 原生应用架构	355
	（一）从云原生到 AI 原生：应用架构的演进与智能化浪潮	355
	（二）从模型对话到智能体：AI 应用架构的迭代发展	360
	（三）大模型能力的延伸：智能体与工具服务的整合	380
	（四）模型知识与私域知识的融合之路	391



（五）企业级 AI 原生服务能力建设	397
第四节 AI-Ready 数据底座	402
（一）AI 时代的数据中台方法论	402
（二）浅谈高质量数据集建设	412
（三）智能体赋能 AI 数据中台建设	421
（四）智能数据资产盘点与智能问数探索	429
第五节 企业服务智能体应用案例	436
（一）灵渠－云化架构智能体：让大模型成为企业架构师的“数字分身”	436
（二）专有云运维的“未卜先知”——大模型能力建设	446
结束语 /	458
附 录 /	
附录 1：阿里云基础产品矩阵	460
附录 2：缩略语	462

序

刘国华

阿里云智能集团副总裁
专有云总经理



不经意间，阿里云成立专有云团队已过 10 年。十余年间，我们坚持了公专一体的技术战略，经过公共云历练的核心技术持续为政企客户提供安全、稳定和领先的云平台。我们始终围绕政企客户的行业发展趋势推动云平台技术创新，包括构建一云多芯、一云多算力来助力政企客户的业务持续上云和创新。在此基础上，我们将以“双 I”战略作为驱动，持续推动云平台的升级。两个“I”分别是 AI 和 International，代表我们的技术发展方向和市场拓展方向。同时，云数联动，云的深度运营，持续的云原生化，云 + 应用一体化运维，以及 AI 创新场景的应用也将成为更加重要的挑战 and 机会。

《政企用云最佳实践》凝聚了阿里云专有云团队多年服务政企客户的经验，系统总结了从上云规划、实施到深化用云及稳定性建设的全过程。本书不仅介绍了完整技术架构的上云规划与选型，还讨论了上云过程中的技术挑战和最佳实践，包括云上的持续运营，云原生化的挑战，云上全周期的稳定性建设体系。最后，我们进一步探讨了面向智能时代的前沿实践，包括大模型工程化落地、AI-Ready 数据底座构建，以及智能体在企业场景中的应用实践，为政企客户迈向“数智驱动”提供可借鉴的路线图。

我们衷心希望，通过本书的总结分享和最佳实践，对政企行业客户在专有云建设领域提供一个有力参考，更好地通过云来反哺业务，助力业务提效和创新。

第一章 上云规划篇



第一节 /

云平台的“MBTI”——通过合理调研做好上云前规划

如果云平台也有 MBTI (Myers-Briggs Type Indicator) , 那么在上云前通过合理的调研, 我们可以更好地了解您究竟需要一朵怎样的云。就像 MBTI 帮助我们理解个人的性格偏好和行为模式一样, 深入的前期调研能够帮助企业更好的洞察云计算环境中的具体需求和最佳匹配。每朵云都有其独特的“性格”和适用场景, 选择最适合的云平台不仅能优化资源配置, 还能为企业带来显著的价值提升。

1. 应用系统上云六大核心价值

企业数智化转型是发展的必然趋势, 云计算是企业数智化转型的重要驱动力。通过构建云能力为企业建立新的差异化竞争优势, 助力企业实现数智化转型的战略和业务目标。

通过云平台支撑业务快速创新、提升治理水平和降低企业成本, 实现资源标准化供给, 提升企业的运维效率, 提供安全合规的平台环境。应用系统迁移上云是实现云价值的重要途径, 云转型为企业带来的价值优势主要体现在以下六点:

(1) 业务支撑

推动 IT 基础架构数字化转型, 确定企业上云战略, 通过云转型引领变革, 带领业务创新, 对应用 / 数据充分保护、统一管理, 提供便捷、安全的业务体验, 为新业务创新保驾护航, 支撑业务创新迭代, 为业务开发人员提供可靠一致的标准化的开发和测试环境, 使企业聚焦在业务创新本身。

CHAPTER 01

</上云规划>

(2) 提升治理水平

通过应用系统上云，将传统的 IT 治理转化为云治理，企业可以大大简化 IT 治理复杂度，统一资源治理架构，实现流程标准化，缩短资源分配的周期，提升业务部门满意度。

(3) 降低成本

借助云服务商提供的快速迭代的云产品能力，降低创新成本，自动化运维程度提升，减少人工误操作引起故障带来的经济损失，从而降低企业 IT 运营成本，结合现有资产，合理规划与利旧，实现物尽其用。

(4) 资源灵活供给

满足应用系统弹性、可伸缩性的资源需求，提升产品交付速度，缩短交付周期，可在数秒或分钟内完成资源供给，提高了业务响应速度，支撑应用与基础设施松耦合，将资源服务化，整合已有资源，实现基础架构的模块化与标准化，提高资源利用率。

(5) 运维效率提升

引入云上运维产品，增强自动化运维水平，提升运维效率，为应用系统提供“高可用”和“高可靠”的基础设施保障，提高业务的连续性和可靠性。

(6) 安全合规

提供满足等保三级的资源环境，确保应用系统稳定易维，借助云平台增强企业威胁情报模型，更好的抵御攻击，满足安全合规要求，优化灾备方案提升应用高可用的能力，提升 SLA 水平。

2. 应用上云规划“4 步走”策略

企业上云通常是伴随着业务发展的转型和重大的技术演进，近年来基于云服务的数字化转型是很多企业选择的路线，结合企业发展和数字化转型蓝图，确定企业上云的整体目标和业务支撑价值，是上云前期在规划阶段的主要依托。在上云规划阶段还应当重点考虑云基础设施与现有 IT 体系的融合对接，做好平滑接入和管理。企业上云的主要内容是业务应用上云，在前期规划阶段对现有业务应当进行详细的摸底调研、基于上云评估策略分析其上云价值、处置路径及优先级，为后续应用上云提供策略性指导。

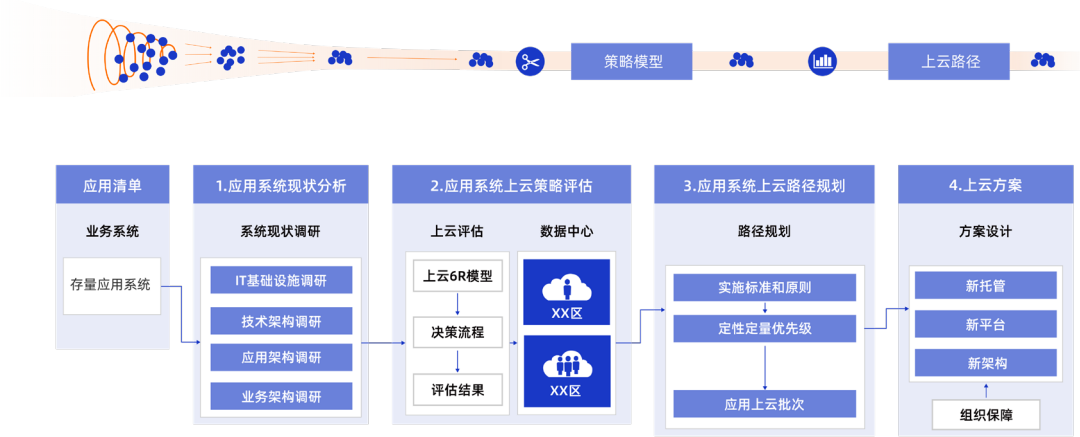


图 1 上云规划“4 步走”策略图

针对存量应用系统上云，通常按照四个步骤开展：

- a. 应用系统上云现状分析：本阶段主要完成 IT 基础设施现状调研、技术架构调研、应用架构调研和业务架构调研，发放调研信息，整理搜集汇总，修正及验证。同时在此阶段会对应用做初步的业务重叠性分析，作为系统统建与合并的判断依据。
- b. 应用系统上云策略评估：本阶段主要完成上云评估，包括上云策略的定义、场景及价值阐述，上云决策流程及评估结果展示。
- c. 应用系统上云路径规划：本阶段介绍敏稳双模应用的上云策略，制定应用系统上云实施原则和标准，定性及定量评估，结合客户实际情况，综合考量上云成本因素，给出应用系统上云迁移的路径图。
- d. 应用系统上云方案：本阶段介绍应用标准迁移步骤，定制迁移过程中需要遵循的技术规范，适配客户应用系统上云的 6R 上云策略。

3. 开启咨询调研，甄别 16 项核心业务特征

咨询调研，往往是上云规划的第一步。调研形式可以根据实际项目情况进行灵活安排，一般情况下可通过下发调研问卷到各项目组，与各应用系统开发商进行线上会议、线下交流等多种形式完成多轮调研与信息收集，完成应用系统现状信息的收集及确认。通过调研，需要掌握客户 IT 基础实施情况、应用系统情况。其中 IT 基础实施部分，可重点关注客户网络架构、虚拟化实现方式等。应用系统部分，是调研的重点，一般从应用架构、技术架构、业务架构、数据架构、集成架构，共计 30 个调研项展开。

调研维度	业务架构调研	技术架构调研	应用架构调研	数据架构调研	其它集成架构调研
调研内容	1. 应用系统名称 2. 应用系统接口人 3. 应用系统业务场景描述 4. 应用系统重要性 5. 应用部署架构图 6. 依赖系统名称 7. 被哪些系统依赖 8. 应用系统当前生命周期 9. 当前应用系统是否可以满足未来（1-2 年）业务增长需求 10. 应用系统代码是否可控 11. 应用系统设计文档是否完整	12. 开发语言 13. 研发框架（微服务中间件 / 数据中间件 / 消息中间件 / 其他，及版本） 14. 缓存类型及版本 15. 操作系统及版本 16. 是否有其他技术栈补充	17. 应用部署位置 18. 用户请求或流量的来源，以及应用系统的访问量等 19. 应用服务器类型及资源情况（小型机、X86、ARM 等，具体规格，CPU、内存、磁盘） 20. 应用是否无状态 21. 应用高峰期 CPU 负载情况 22. 应用是否单点部署	23. 数据库类型及版本 24. 数据库数据量级 25. 数据存储类型（本地盘、EMC 等商业存储、NAS、GlusterFS、Ceph 等） 26. 数据存储量级 27. 是否有其他业务或者技术方面的痛点	28. 其他说明（比如对带宽的要求等） 29. 是否使用专属硬件 30. 硬件依赖特殊要求

表 1 调研内容参考表

通过整合调研项，抽象出 16 项核心业务特征，可以作为上云策略评估阶段的核心输入：

序号	核心业务特征	描述
1	系统业务场景描述	详细描述系统的业务功能和应用场景，包括主要业务流程和用户群体。
2	系统重要等级	确定系统在企业中的重要性等级（如关键、重要、一般）。
3	系统间调用关系	描述系统与其他系统之间的调用关系，包括依赖关系和数据交互方式。
4	系统生命周期阶段	确定系统当前所处的生命周期阶段（如开发、测试、生产、维护）。
5	系统访问量	统计系统的日均访问量、峰值访问量等，以确定负载需求。
6	系统未来增长需求	预测系统在未来一段时间内的增长趋势，包括用户数、数据量等。
7	部署图及模式	提供系统的部署架构图，包括物理和逻辑部署模式（如单体、微服务）。
8	计算资源需求	详细列出系统所需的计算资源，如 CPU、内存、GPU 等。
9	存储资源需求	详细列出系统所需的存储资源，如磁盘空间、存储类型（块存储、对象存储）。
10	网络需求	描述系统的网络带宽需求、延迟要求、安全性和网络架构。
11	应用架构	说明系统的应用架构设计，包括层次结构、组件划分等。
12	编程语言和开发框架	列出系统使用的编程语言、开发框架和技术栈。
13	数据库需求	详细列出系统所需的数据库类型、版本、性能要求等。

14	中间件和缓存版本	列出系统使用的中间件和缓存工具及其版本。
15	云平台 OS 兼容性	确定系统对云平台操作系统的兼容性要求。
16	特定硬件依赖性	列出系统对特定硬件的依赖性，如专用硬件设备或特定型号的服务器。

表 2 16 项核心业务特征

实例化描述参考如下表所示：

序号	核心业务特征	描述
1	系统业务场景描述	该系统用于处理客户订单，包括订单创建、支付、发货等流程。
2	系统重要等级	关键系统，直接影响公司收入。
3	系统间调用关系	与库存管理系统、支付系统、物流系统有紧密的数据交互。
4	系统生命周期阶段	生产阶段，已稳定运行多年。
5	系统访问量	日均访问量约 10 万次，峰值访问量可达 50 万次。
6	系统未来增长需求	预计未来一年内用户数增长 30%，数据量增长 50%。
7	部署图及模式	微服务架构，多个服务通过 API 网关进行通信。
8	计算资源需求	每个服务需要 2 核 CPU、4GB 内存，总需求为 16 核 CPU、32GB 内存。
9	存储资源需求	需要 1TB 的块存储和 500GB 的对象存储。
10	网络需求	需要 1Gbps 的公网带宽，内部网络延迟不超过 10ms。
11	应用架构	三层架构：前端、后端服务、数据库。
12	编程语言和开发框架	使用 Java、Spring Boot 框架，部分服务使用 Node.js。
13	数据库需求	需要高可用性和读写分离。
14	中间件和缓存版本	使用 Redis 6.0 作为缓存，Kafka 2.8 作为消息队列。
15	云平台 OS 兼容性	需要支持 Linux CentOS 7 和 Ubuntu 20.04。
16	特定硬件依赖性	无特定硬件依赖，但需要高性能的 SSD 存储。

表 3 实例化描述参考表

4. 小结

为了进一步提升 IDC 机器资源的调研效率，我们建议使用专有云云迁移中心提供的线下采集工具 idc-scanner，该工具能够高效采集机器的资源、进程和应用信息，显著提高系统的调研速度和准确性。通过完备细致的调研，可以为上云规划迈出坚实的第一步。

第二节 /

上云的“6R”模型——解锁上云策略的底层代码

1. 什么是“6R”模型

Gartner 早在 2010 年就发布了“5R”系统上云模型，定义了将特定应用程序迁移到云上的五种常见选项：Rehost（重新托管）、Refactor（重构）、Revise（修订）、Rebuild（重建）和 Replace（替换）。

基于大量客户的上云实践，阿里云进一步完善并提出了“6R”应用系统上云迁移策略，涵盖了所有可能的迁移方式，包括：

- **Rehost（重新托管）**：将线下物理机 /虚拟机替换成为云上虚拟机服务。不改变原有应用运维方式及技术架构。
- **Replatform（新平台）**：对应用程序进行少量调整以利用云平台的功能，可以认为是组件级适配。
- **Refactor（新架构）**：重构应用程序以充分利用云计算的优势，如微服务化，可以认为是系统级云原生重构。
- **Replace（替换）**：被新规划建设的应用及新引入的应用进行替换。
- **Retain（保留）**：保留应用系统在当前的 IT 环境，暂时不迁移。
- **Retire（退役）**：停止使用不再需要的应用程序，清理冗余资源。

详细的定义和适用场景如下表所示：

策略	定义	场景描述
Rehost	将线下物理机 / 虚拟机替换成为云上虚拟机服务。不改变原有应用运维方式及技术架构	快速迁移，适用于简单应用或临时过渡。
Replatform	对应用程序进行少量调整以利用云平台的功能，可以认为是组件级适配	提升性能和可靠性, 适用于已有应用的优化。
Refactor	重构应用程序以充分利用云计算的优势，如微服务化，可以认为是系统级云原生重构	长期优化, 适用于复杂应用的云原生改造。
Replace	被新规划建设的应用及新引入的应用进行替换	技术升级，适用于过时或效率低下的应用。
Retain	保留应用系统在当前的 IT 环境，暂时不迁移	关键业务应用，短期内不适合迁移。在条件背后，再进行云化改造上云。
Retire	停止使用不再需要的应用程序，清理冗余资源	资源优化，适用于不再需要的应用程序。

表 1 "6R" 模型定义

2. 上云策略决策流程

将现状调研阶段搜集的 16 项核心业务特征作为“6R”评估模型的输入，通过评估模型得出不同系统上云迁移的策略。

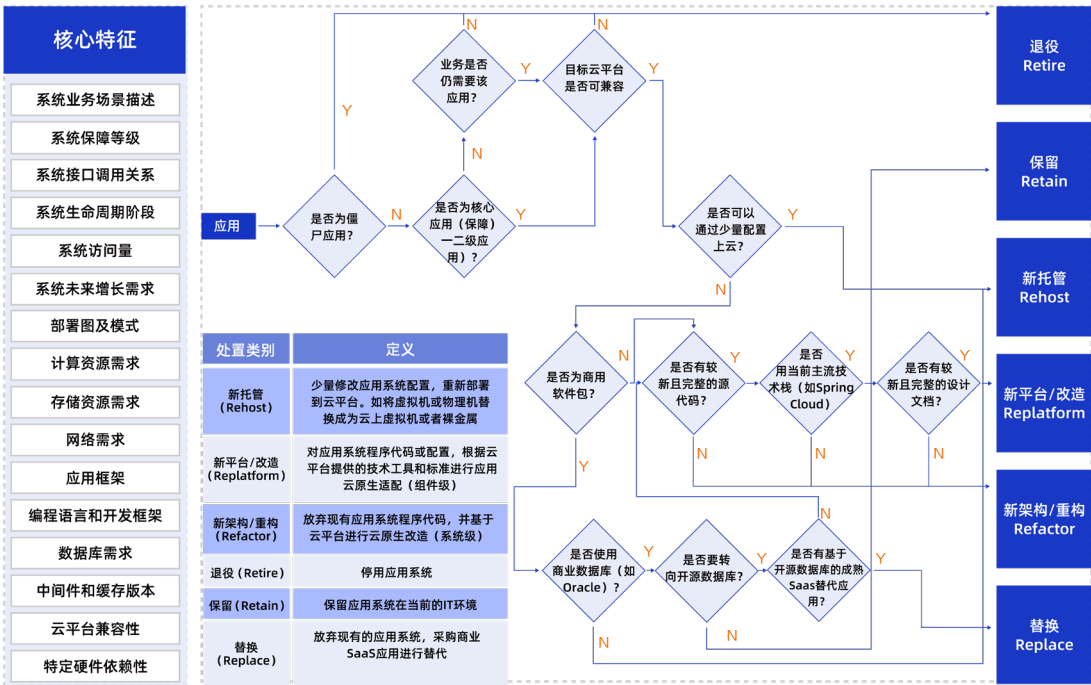


图 1 系统上云策略决策流程

总体上云迁移评估流程： 现状调研的 16 项核心特征作为输入，通过评估决策模型，经过 12 项判断条件，最终得出应用系统上云策略。

- 将前期现状调研涉及 **16项核心业务特征**作为判定输入：系统业务场景描述，系统重要性等级，系统接口调用关系，系统生命周期阶段，系统访问量，系统未来增长需求，部署图及模式，计算资源需求，存储资源需求，网络需求，应用架构，编程语言和开发框架，数据库需求，中间件和缓存版本，云平台兼容性，特定硬件依赖性。
- 16项核心业务特征，通过归纳分类，得出**系统上云策略决策流程的十二项判定条件**。对应关系如下表所示：

序号	判定条件	现状调研信息
1	是否为僵尸应用？	系统生命周期阶段
2	是否仍对应用存在需求？	系统生命周期阶段，系统业务场景描述。
3	是否为核心应用（应用分级）？	系统重要性等级。
4	目标云平台是否可兼容？	应用服务器类型，特定硬件依赖性。
5	是否可虚拟机迁移，不改运维方式？	部署图及模式，计算资源需求，存储资源需求，网络需求，数据库，中间件和缓存版本，系统接口调用关系。
6	是否为自研软件？	应用架构，应用系统代码是否可控。
7	是否完整可控的源代码？	应用架构，应用系统代码是否可控。
8	是否采用主流技术栈？	开发语言和开发框架。
9	是否有更新的设计文档可参考？	应用系统设计文档是否完整。
10	是否使用 Oracle、SQLServer 数据库？	数据库类型。
11	是否使用开源数据库替换？	系统访问量，系统未来增长需求。
12	是否可以使用成熟 SaaS 替代？	系统业务场景描述，系统生命周期阶段，系统访问量，系统未来增长需求。

表 2 十二项判定条件

- 通过上述 12项核心判断条件进行“6R ”评估流程决策，最终得出应用系统上云的策略。

3. 上云策略实例化推演

以“某应用研究与开发平台”调研结果为例，通过“6R”上云决策流程，推演与之匹配的上云策略。

序号	调研项	调研结果
1	应用系统名称	某应用研究与开发平台
2	业务部门	开发部
3	应用系统接口人	张三
4	该应用系统业务场景描述	基于 web 端的三维模型轻量化服务，统一的三维可视化引擎。功能演进中，每双周一个迭代。
5	应用系统重要性	核心业务系统
6	系统未来增长需求	系统交付上线初期
7	当前应用系统是否可以满足未来(1~2年内) 业务增长需求	可以
8	应用系统代码是否可控	是
9	应用系统设计文档是否完整？	是
10	应用架构	主流开源架构
11	作为调用方在调用关系表中出现次数	3
12	作为提供方在调用关系表中出现次数	5
13	开发语言和框架	Java、Ant Design Vue
14	是否提供公网服务	否
15	应用系统的访问量	1000 以下，预计日均访问 2000 次
16	是否使用专属硬件	否
17	硬件依赖特殊要求	否
18	操作系统要求	Windows Server 2016 Standard
19	中间件类型	Nacos, Dubbo, Kafka
20	中间件版本	2.2.3, 3.0.10, 2.12-2.3.0
21	应用系统所用的数据库类型及版本	MySQL 8.0.31, MongoDB 5.0
22	应用系统所用的数据库数据量级	100G, 100G
23	缓存类型	Redis
24	缓存版本	3.0.504
25	CPU 内核数	2C
26	内存	64G
27	docker	/
28	磁盘 (GB)	2048
29	数据量级	TB 级，采用开源对象存储 Minio
30	应用系统上云意愿	中立
31	其他业务或者技术方面的痛点	暂无

表 3 实例化推演表

基于 12 项判定条件，结合“6R”上云决策流程图，可推演出“某应用研究与开发平台”，适合 Replatform 新平台方式上云。

4. 小结

企业通过“6R”模型助力上云规划，全方位打造应用上云策略的“六脉神剑”。每个“R”为企业提供了明确的行动指南，确保从传统 IT 环境顺利过渡到云端。通过这六大策略，企业可以系统化地评估和迁移应用，确保上云过程平稳高效，最大化云投资回报。这一方法不仅为数智化转型保驾护航，还帮助企业在不同阶段做出最优决策，最终实现业务的持续增长和创新能力的提升。

第三节 /

上云的“号码牌”——明晰应用上云的优先侧重

1. 应用系统上云路径规划“三步走”策略

应用系统上云路径规划是一个复杂而系统的过程，通常按照“三步走”策略开展。首先，通过“稳敏态”双模定义来定性约束应用系统上云优先级；其次，结合业务和技术维度来定量细化评估打分排序，并结合应用系统上云改造计划，完成应用系统上云优先级排序；最后，结合客户业务方向给出应用系统上云的总体路径规划。

2. 通过“稳敏态”双模定义来定性约束应用系统上云优先级

Gartner 提出了双模 IT（Bimodal IT）的概念：

- 稳态 IT：传统应用，服务内部用户为主，关注业务平稳运行，强调安全和准确。
- 敏态 IT：新兴互联网应用，服务外部客户和合作伙伴，探索创新业务，强调敏捷和速度。

	稳态 IT 应用 – 尽早能够拥抱云	敏态 IT 应用 – 全面拥抱云
应用架构特点	传统应用，服务内部用户为主； 需求明确、功能全，覆盖广，大集成； 中央控制，适合稳定发展，难以快速变化，维护成本高， 快速变革的新业态无法支持； 强调应用的安全、稳定、高性能。	新兴互联网应用，服务外部客户和合作伙伴； 需求变动快，功能简单，独立和分散； 分布式进化，一切都从零开始，需要快速创新运用 规模变化大，大范围广泛的尝试，易淘汰，对 业务弹性、快速发布要求高； 强调应用的快速交付，灵活、可扩展。
基础设施要求	对应用程序进行少量调整以利用云平台的功能，可以认为是组件级适配	提升性能和可靠性，适用于已有应用的优化。
项目方法	采用传统的 IT 服务管理（ITSM）框架	遵循更加敏捷和迭代的过程

治理模式	需要具备可治理的明确目标，预先批准的计划	具备实验性和不可预测性
目标	侧重于维护当前系统和软件的可靠性	提高现有系统的效率和速度，更加着眼于未来发展
价值	新项目启动基于可预测的及可实现的项目预期价值	侧重于提高绩效，来改善客户体验和品牌声誉

表 1 双模 IT 特点对比

应用系统上云战略需全面审视梳理现有业务系统，认真评估应用资产，并结合业务现状进行详细的评估分析，确定上云优先级及可行性方案。一般来说，我们会考虑综合成本收益比、应用系统上云难度、安全保护要求、监管机构管理要求。对于符合国家安全可控要求的，特别是关键基础设施相关的应用系统应考虑优先上云，我们对应用系统上云优先级推荐如下：

（1）高优先级

以互联网技术架构为基础，或内部支持职能类业务系统为主；比如电商类的应用系统，此类系统开发基于开源框架，几乎不需要改变应用系统的架构，优先云上部署。

（2）中优先级

以运营管理类的应用系统为主，需要一定上云改造，但仅限于接口模式或数据对接类的改造，不涉及到整体应用架构的改造。

（3）低优先级

一般为企业关键生产业务系统，上云时需要投入大量财力和时间进行应用架构改造，并需重点考虑与其他业务的对接模式、应用架构的分布式改造等，本类应用需详细评估上云的可行性和必要性。

3. 结合业务和技术维度来定量细化应用系统上云优先级

根据大量项目的实践经验，总结出影响上云优先级的 10 个评分维度，通过业务和技术两个维度，对应用系统上云进行定量的细化评估，按照高中低进行评分，最后将 10 项评估分数汇总

取均值，分值越高，上云的优先级越高，按照分值分别对存量应用系统进行上云优先级排序，详细的定量细化评估项和原则说明参照下表：

（1）业务维度

	评估内容	低（1分）	中（2分）	高（3分）	原则说明
业务维度	系统级别	一般系统	重要系统	核心系统	按照系统重要等级定义，核心系统优先迁移 1. 统建应用 2. 用户量较大、关联系统多、基础工具类 3. 很少人使用或即将下线
	系统迁移依赖性（关联应用数量）	>5 个	3-5 之间	≤ 2	参考调研结果中系统关联的系统数量
	复杂系统，业务领域拆分难度	复杂系统（子系统同步上云）	子系统（可拆分上云）	单体或独立应用	1. 系统包含子系统，之间强依赖关系，归为复杂系统（子系统同步上云）， 2. 子系统可以分拆上云 3. 单体或独立系统。
	系统上云迫切度	保守	中立	积极	参考各个应用反馈情况，并结合系统技术栈最佳实践综合评估，无反馈按照中立。
	系统并发程度	同时在线量小于 500，访问次数小于 50 次 / 秒 /	同时在线量大于 等于 500 小于 1000，或访问次数小于 等于 100 次 / 秒	同时在线量大于 等于 1000，或访问次数大于 100 次 / 秒	1. 同时在线量大于 等于 1000，或访问次数大于 100 次 / 秒，定义为系统并发程度高，访问量 大，得三分。

表 2 业务维度评分表

（2）技术维度

	评估内容	低（1分）	中（2分）	高(3分)	原则说明
技术维度	技术架构是否改造（Replatform+Refactor）	改造（应用层代码重构）	微调（PaaS 层替换）	无改造	1. 自研系统 .net 和 C# 应用如果要重构为 Java，Spring 技术栈； 2. 数据库改造列为微调； 3. Java，Spring 列为不改造； 4. 系统采用成熟商业产品，且未来无改造计划此项不评分；
	操作系统上云兼容性	操作系统不兼容，需要改造	版本升级及操作系统改变	兼容，不改变	1. 一分：非 windows 服务器版本上云需要进行改造，比如 win10、win7； 2. 二分：低于阿里云支持的最低操作系统版本，如 windows server 2008R2、centos 5、redhat5 以下的版本，列为需要升级操作系统； 3. 三分：其他列为兼容，不改变，包含 docker
	数据库上云兼容性	数据库不兼容，改造量大	IaaS 部署	少量改造，适配 PaaS 数据库	1. Oracle 定义为不兼容需要改造； 2. 国产数据库（如达梦、神通）定义为 IaaS 提供 3. SQL server、MySQL 及其他开源数据库定义为适配 PaaS 数据库
	中间件，缓存上云兼容性	IaaS 部署	docker 镜像	PaaS 部署	1. 中间件通过 VM 供给定义为 IaaS 提供，比如 IIS 2. docker 镜像提供和 PaaS 服务，需要结合操作系统共同打分，如仅 docker 部署，给与 2 分； 3. PaaS 部署缓存，中间件为 3 分，如 Redis、MQ
	系统迁移的工作量	新架构且服务器数量大于等于 10 新平台且服务器数量大于等于 10	新托管且服务器数量大于 5 新平台、新架构且服务器数量在小于 10 之间（不含）	新托管且服务器数量小于等于 5	1. 优化上云（新平台）/ 重构上云（新架构），或服务器数量大于 10，有 Oracle 和 SQL server、神通、达梦上云，都算改造量比较大的 2. 优化上云（新平台），服务器数量在 5-10 之间 3. 平迁上云（新托管），服务器数量小于 5 个 4、保留及退役此处不评分

表 3 技术维度评分表

4. 小结

本文详细介绍了应用系统上云路径规划的“三步走”策略，旨在确保上云工作逐步推进并降低风险。首先，通过稳敏态双模定义（稳态 IT 和敏态 IT）来定性约束上云优先级；其次，结合业务和技术两个维度，使用 10 个评分维度进行定量评估，确定每个应用系统的上云优先级；最后，结合客户业务方向和具体改造计划，制定详细的上云路径规划。这一系统化的方法基于大量项目实践经验，能够确保应用系统上云过程的顺利进行，并最终实现上云目标。

第四节 /

上云的“明算帐”——如何进行上云的成本评估

1. 引言

上云过程中的成本控制是一个复杂的问题，需要进行精细化的成本评估。上云成本的构成繁杂，涉及到初始成本（硬件采购、软件许可、迁移成本）、运营成本（计算资源、网络带宽、监控工具产品管理）、培训成本（员工培训、外部咨询）、维修和支撑成本（技术支持、故障修复、定期更新）、升级和替换成本（技术升级、新功能引入）、退役和处置成本（旧设备处置）等。

本文会结合前序章节介绍的“6R”上云策略，重点介绍不同上云策略对应的上云迁移成本，包括数据迁移成本、应用上云改造成本、测试验证成本等，确保上云预算有据可依。

2. 上云成本评估

本评估仅使用人天作为评估标准，同时改造适配涉及应用开发人员技术能力、应用复杂度等多种因素，仅作为评估参考。

序号	上云策略	改造模块	阿里云产品改造点	评估标准（能改造）	迁移 / 改造工作量（人天）	工作量评估要点
1	新托管	ECS 平迁上云	不改 造， 只 使 用 ECS 计算服务	代码找不到	5	单应用重新部署、测试
				开发人员找不到无法适配云上PaaS(商用软件等)		
2	新平台	接入层	负载均衡（SLB）	F5/Nginx	2	配置及功能测试

2	新平台	计算层	应用容器化（ACK）	Docker/K8S 部署（Rancher 等）	10	Dockfile 镜像制作，部署和功能测试
				应用无状态（没有本地化业务数据）		
		存储层	对象存储（OSS）	FastDFS	5	替换成 OSS SDK，开发适配和功能测试
				MiniO	5	
			文件存储（NAS）	云下 NAS（NFS 协议）	2	挂载云上 NAS 和功能测试
		中间件层	应用托管（EDAS）	Tomcat(Spring boot)	单应用：1 人天 多应用：5 人天	单应用包：平迁，不需要任何改造，做功能测试。特殊情况除外（会话保持、自定义类库） 多应用包：应用拆分 EDAS 发布，进程内通信变成远程服务调用
				Tomcat(Spring Cloud)	单应用：1 人天 多应用：5 人天	单应用包：平迁，不需要任何改造，做功能测试。特殊情况除外（会话保持、自定义类库） 多应用包：应用拆分 EDAS 发布，进程内通信变成远程服务调用
			注册中心（EDAS）	Nacos/Apollo	3	修改注册中心配置和功能测试
			缓存（Redis）	云下 Redis	5	替换成 Redis SDK 和功能测试
				云下 memcache	5	
			消息队列（MQ）	ActiveMQ/RabbitMQ/Kafka	5	替换成 MQ SDK 和功能测试
				RocketMQ	2	
		数据库	数据库 RDS（MySQL）	MySQL（版本）	5	修改实例链接地址配置、数据迁移配合和功能测试
			数据库（PgSQL10）	PgSQL（版本）	5 人天	修改实例链接地址配置、数据迁移配合和功能测试
			数据库（MongoDB）	MongoDB（版本）	5 人天	修改实例链接地址配置、数据迁移配合和功能测试
		大数据	弹性搜索 ElasticSearch	ElasticSearch	10 人天	修改实例链接地址配置、数据迁移配合和功能测试
		监控系统	基础监控/产品监控	/	3 人天	云产品监控梳理和配置监控对接
			业务监控	/	3 人天	业务指标梳理和配置监控对接
			应用监控（ARMS）	/	2 人天	应用监控配置监控对接
			日志	ELK	3 人天	日志输出规范调整和配置日志到 ELK
		运维	云效	Jenkins	6 人天	流水线对接到云效，配置，测试

表 1 上云成本评估表

3. 小结

精细化的成本评估是企业上云过程中不可或缺的一环。通过详细的成本分析和科学的预算规划，企业可以确保上云项目的顺利进行，并实现预期的经济效益。希望本文能为您的上云成本评估提供有价值的指导。

第五节 / 云商城的“选购指南”——如何科学地进行云产品选型

1. 引言

随着云计算、AI、大数据、大模型技术的快速发展，越来越多的企业开始将业务迁移到云端，以充分享受云计算带来的技术红利，支撑业务快速创新。然而，面对众多的云服务提供商和丰富的产品选项，如何选择最适合企业需求的云产品成为了一个重要课题。本文旨在为读者提供一个全面的云上产品选型指导，以帮助企业高效上云。

2. 云产品选型整体策略

云产品选型从大量的客户实践角度看，一般遵循以下 5 种选型策略：需求场景匹配度、综合成本效益、技术成熟度、技术领先性、支持“一云多芯”。

（1）需求场景匹配度

选择能够满足当前和未来业务及技术需求的云产品至关重要。这样的产品不仅应具备强大的功能来支持现有的业务流程，还应具有高度的灵活性和可扩展性，以适应业务的增长和技术的发展。具体来说：

- **当前需求：**确保云产品能够无缝集成到现有的 IT 架构中，提供稳定、高效的服务。
- **未来扩展：**选择支持弹性扩展的云产品，以便随着业务增长轻松增加资源。
- **新用例支持：**云产品应具备处理未来可能出现的新用例的能力，如人工智能、大数据分析等。
- **持续更新：**选择那些定期更新并引入新功能的云产品，以保持技术领先。

通过这些标准，企业可以确保所选的云产品不仅能应对当前的需求，还能为未来的业务和技术发展做好准备。

（2）综合成本效益

在选择云产品时，需要综合评估实施、运营和维护的总拥有成本（TCO），确保长期的经济性和可持续性。总成本应包括以下几个方面：

- **初始成本：**包括采购、部署和配置所需的硬件、软件及服务费用。
- **运营成本：**涵盖日常运行中的资源消耗、网络带宽、存储空间以及监控和管理工具的费用。
- **培训成本：**涉及对员工进行技术培训和认证的费用，以确保他们能够有效使用和管理新的云产品。
- **维修和支撑成本：**包括技术支持、故障修复、定期维护和更新的费用。
- **升级和替换成本：**随着技术的发展，可能需要对现有系统进行升级或替换，这将产生额外的成本。
- **潜在机会成本：**考虑因采用新系统而放弃其他投资或项目的机会成本。
- **退役和处置成本：**当系统达到使用寿命或不再适用时，需要考虑数据迁移、系统退役和设备处置的成本。

通过全面评估这些成本因素，企业可以做出更明智的决策，确保所选云产品不仅在短期内具有成本效益，还能在长期内为企业带来持续的价值。

（3）技术成熟度

选择已在行业头部公司生产环境中规模化落地的云产品，这些产品通常经过大规模实际应用验证，技术成熟且稳定可靠。同时，选择活跃的社区支持和良好的生态系统能够提供丰富的资源、工具和第三方集成，帮助企业在使用过程中获得持续的技术支持和更新。这样的选择不仅能满足当前需求，还能为未来的扩展和技术演进打下坚实基础。

（4）技术领先性

在快速变化的技术环境中，选择具有前瞻性和持续创新能力的工具产品对于企业来说至关重要。这样的工具不仅能帮助企业充分利用当前的最新技术，还能为未来的科技发展趋势做好准备。

（5）支持“一云多芯”

在一个云计算环境中支持多种不同类型的处理器架构（如 x86、ARM、MIPS 等），使得云平台能够根据不同的应用场景和需求，灵活选择最合适的硬件资源。增强云平台的兼容性和灵活性，满足多样化的工作负载需求。

3. 云上数据库选型指导

数据库是现代企业信息系统的核心组件，它存储和管理着企业的关键数据，支持各种业务操作和决策过程。

"I believe that 'one size does not fit all'. I.e. in every vertical market I can think of, there is a way to beat legacy relational DBMSs by 1-2 orders of magnitude."（笔者注解：我相信“没有一个数据库可以包打天下，在我能想到的所有数据库垂直领域内都可以有性能比传统 DBMS 系统高 1-2 个数量级的解法或产品”）

--- Michael Stonebraker, 2014 年图灵奖得主

传统数据库云化起源于互联网行业，并在 2010 年最先由阿里巴巴公司发起。我们在服务大量企业客户完成数据库云化的过程中总结了数据库选型的四大原则，旨在为企业在进行数据库选型时提供有价值的参考。

（1）数据库选型四大原则

阿里云数据库已经在政企、金融、运营商、能源电力、互联网等多个行业的关键核心业务中成功落地。以客户需求为导向，我们开发和迭代了多种面向行业属性的功能，满足不同场景的业务需求。

随着传统数据库云化进程的推进，我们逐步实现了将传统的商业数据库系统迁移到云原生的数据库系统中，并支撑相关应用的适配改造。目前，我们已经具备将多种传统商业数据库（如 Oracle、TD、DB2、HANA 等）跨平台迁移到兼容性更强的云原生数据库系统中的能力。

（1.1）从生态角度选型

如图 1《数据库产品的生态关系示意图》所示，通过观察整个数据库业界的产品生态关系，我们可以发现，传统主流商业数据库（如 Oracle、SQL Server、DB2 等）正逐步向开源方向转型。特别是 MySQL 和 PostgreSQL，因其背后的活跃开源生态系统而备受青睐。

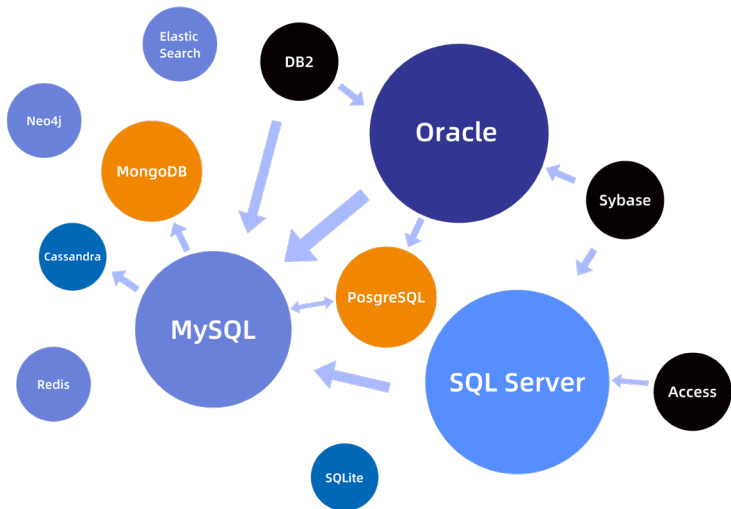


图 1 数据库产品的生态关系示意图

阿里云数据库的发展紧跟业界潮流，如图 2《阿里云数据库产品兼容体系示意图》所示，拥抱开源 MySQL 和 PostgreSQL 生态的同时，也开发了一些 Oracle 兼容的功能，尽可能降低对前端应用的业务侵入。

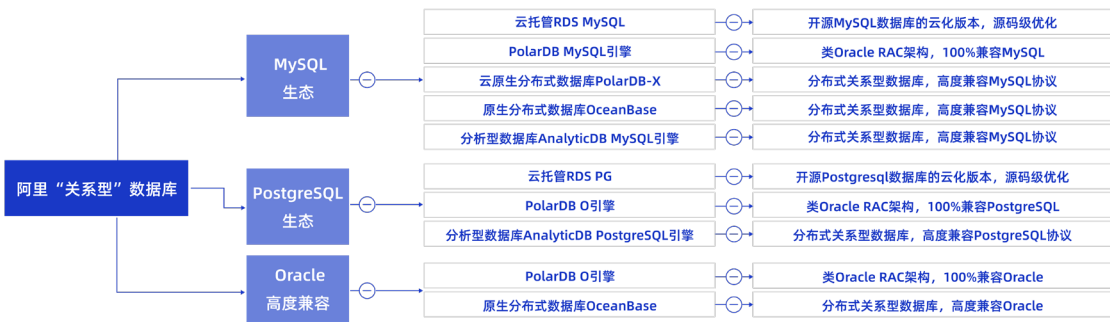


图 2 阿里云数据库产品兼容体系示意图

（1.2）从应用视角选型

数据库的应用类型可以分为联机事务处理（OLTP）应用和联机分析处理（OLAP）应用；如表 3 数据库 OLTP、OLAP 业务区别所示，区别在于联机事务处理是实时的，只关注最近一段时间的数据，响应及时性要求很高，而联机分析处理从大量数据中分析规律，通过查询分析规律趋势，用于各类决策分析等场景。

区别	OLTP	OLAP
用户	操作人员，基层管理人员	决策人员，高层管理人员
功能	日程处理操作系统	分析决策
DB 设计	面向应用	面向主题
数据	当前的，细节的，二维的，分离的	历史的，聚集的，多维的，统一的
工作单位	简单的事务	复杂的查询
时间要求	具有实时性	对时间的要求不严格
主要应用	数据库	数据仓库

表 1 数据库 OLTP、OLAP 业务区别

如前文所述，阿里巴巴电商业务在早期的 OLTP 和 OLAP 业务都是使用 Oracle RAC 承载，随着业务发展演化出 OLTP 和 OLAP 专用数据库。我们再拜访了很多传统企业后发现许多企业的数据架构还处在阿里巴巴电商业务早期的形态，IT 架构转型迫在眉睫；企业在进行数据库选型时可以参照如图 4 根据应用类型选型数据库所示对产品进行初步筛选。

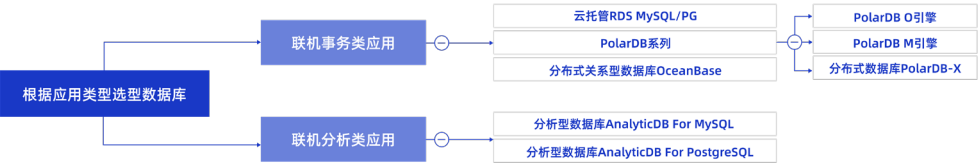


图 3 根据应用类型选型数据库

在上图中我们一般建议：

- 如果客户明确是联机事务类场景，大体可以从如下三个方面进行产品选型：
 - 客户业务场景是使用开源 MySQL/PostgreSQL 自建场合，上云初期推荐使用云托管 RDS MySQL/PG 进行平滑迁移，对云上环境比较适应后可以再一键切换到 PolarDB M 引擎享受云原生的弹性、智能运维等便利性

- 自建 Oracle 场景建议选用集中式架构的 PolarDB O 引擎或者分布式关系型数据库 OceanBase，两者都对 Oracle 特性进行了大量适配，能够提供较高 Orace 兼容度；在面向未来的分布式架构中，广泛使用 OceanBase 做核心系统分布式转型。
- PolarDB X 作为从电商业务孵化出来的云原生分布式数据库，使用 MySQL 协议对 MySQL 进行了高度兼容，广泛用于互联网电商等领域。

- 在联机分析类场景中推荐选用分析型数据库 AnalyticDB，如前文所述早期阿里巴巴电商业务中时效性较高的交互式查询场景通过 HBase/MySQL 集群 /Hadoop 集群组合方式解决，涉及较多组件、数据流转链路效率低、维护成本较高；分析型数据库 AnalyticDB 出现用于解决该痛点问题，该系列产品采用经典 MPP 架构通过一套技术栈在大规模数据中实现点查和复杂关联查询融合场景。

(1.3) 从云原生架构转型视角

云计算的普及带来了传统软件开发模式和 IT 架构的变革，开发人员把所有时间都花在核心业务的应用开发上，不需要浪费时间去开发跟核心业务无关的代码，让客户专注于应用和数据的开发和运营上；从而催生了云原生架构；所谓云原生架构是基于云原生技术的一组架构原则和设计模式的集合，旨在将应用中的非业务代码部分进行最大化的剥离，从而让云设施接管应用中原有的大量非功能特性（如弹性、韧性、安全、可观测性、灰度等），使业务不再有非功能性业务中断困扰的同时，具备轻量、敏捷、高度自动化的特点。

在整个云原生架构中不可或缺的能力是云原生分布式数据库的能力，需要借助分布式数据库的分布式事务引擎、透明可扩展、极致高可用等关键特性降低云原生系统的研发和运维难度，同时安全可控要求也逐渐成为很多行业客户的刚性需求。

在企业云原生架构转型中，我们推荐的原生分布式数据库是 OceanBase 和 PolarDB X 数据库，两者都是阿里具有完全自主知识产权、在业界处于第一梯队的分布式数据库。其中 OceanBase 在阿里内部应用于支付宝全部核心业务以及阿里巴巴淘宝部分业务，外部在银行、保险、证券等行业使用较为广泛；PolarDB X 在阿里巴巴淘宝业务有深度且广泛的应用，在外部互联网客户、能源电力、政企客户中使用相对更广泛。

(1.4) 从业务连续性视角选型

保障业务连续性是企业核心应用关注的重点，同时业务连续性是从接入层到后端存储层的系统化工作，甚至涉及到组织保障，而数据库作为典型的状态化产品，数据库的可用性成为了业务连续性建设的重中之重。

数据库的高可用需要从冗余部署、高可用切换两方面考量。之所以需要冗余部署，是因为单点部署无法抵御人为误操作、机器宕机、软件 bug、机房故障等故障场景，故障时相关联的业务系统只能停摆，这种场景我们通常接受不了。高可用切换是在真实故障场景下，如何将相关联的业务系统切换至状态正常的冗余节点，切换的时长往往成为了我们最为关注的业务连续性指标。

冗余部署要解决的核心问题之一是数据复制，数据复制通常有逻辑复制和物理复制两种，逻辑复制是 SQL 语句级别的复制，而物理复制是数据页或者数据块级别的复制，物理复制比逻辑复制在数据一致性角度更为严谨。冗余部署要解决的另一个核心问题是如何保证冗余节点间的数据一致，业界通常采用了类似 Oracle 的三种复制模式：最大安全（也叫强同步）模式，备节点不提交主节点也无法提交；最大性能（也叫异步）模式，主节点提交后异步传输日志至备节点；最大可用（也叫半同步）模式，正常情况下，备节点不提交主节点也无法提交，但如果备节点在一定时间内没有给主节点 ACK，则退化为最大性能模式。除了以上的三种复制模式以外，也有很多的产品采用了类似 Paxos、Raft 的分布式一致性算法来保证多节点间的数据一致性。真实故障场景下，状态正常的节点和故障主节点间数据差异意味着丢失了多少数据，这也成为了客户最为关注的恢复点目标。

冗余部署和高可用切换能力是基本要求，为了应对机房、地域级别的极端故障场景，我们通常还会考虑数据库的机房和地域级容灾能力，业界通常的做法有同城双机房主备模式、两地三中心主备模式、三地五中心多活模式三种。阿里数据库产品在高可用能力上如下表所示：

产品	复制模式	机房 / 地域容灾	切换方式
RDS 主备版	默认半同步，支持强同步和异步	支持	同城自动切换 异地人员 + 平台切换
异地人员 + 平台切换	日程处理操作系统	分析决策	自动切换
RDS 三节点版	Paxos 强一致	支持	同城自动切换 异地人员 + 平台切换
PolarDB-MySQL	默认半同步，支持强同步和异步	支持	同城自动切换 异地人员 + 平台切换
OceanBase	Paxos 强一致	支持， 同时支持多点写入	自动切换

表 2 阿里数据库产品容灾能力概览表

4. 三种典型上云方案

基于云产品和数据库选型原则，并结合“6R”上云策略，我们认为典型的上云方案主要包括以下三种方式：新托管（Rehost）、新平台（Replatform）和新架构（Refactor）。具体描述如下：

(1) 新托管（Rehost）上云

新托管（Rehost）又称为平迁上云或托管上云：简单将线下物理机 / 虚拟机替换成为云上虚拟机或者裸金属实例，不改变原有的运维方式。单体应用托管上云通常有两种方式：重新部署与镜像迁移。

• 重新部署

按照应用部署文档，使用手动或者自动化工具将应用程序从源虚拟化环境重新部署到阿里专有云 ECS 环境。

• 镜像迁移

将云下 IDC 的服务器或应用程序直接整体迁移到云环境之中，通常有 P2V、V2V 两种方式。从本地物理机和虚拟机，通过全量和增量迁移方式，迁移到阿里专有云，最后完成数据的一致性校验。迁移工具可以采用阿里自研的（Server Migration Center，简称 SMC）服务器迁移中心平台。

新托管上云主要场景如下：

• 系统规模较小

• 系统的架构简单，无复杂的依赖

• 系统的重要性低

• 系统的稳定性要求低

• 系统可接受停机窗口时间长

平迁上云方式技术价值较低，主要使用的是云上 IaaS 资源，无法充分的发挥云计算的技术优势，但可以让业务快速迁移上云。

（2）新平台（Replatform）上云

新平台（Replatform）上云：利用托管的云服务替换线下自建应用基础设施，托管的云服务通常提供更好的弹性、稳定性和自治运维能力，可以让用户关注于应用而非基础设施管理，无需自行维护基础设施的 SLA。新平台上云通常将应用系统进行云化改造适配，包括接入层、数据层、中间件等，改造后应用重新在云上进行部署。新平台上云使用场景如下：

- 核心的业务系统
- 以开源技术组件为主
- 系统的重要性偏高
- 系统治理能力不足
- 系统处于发展阶段，更新迭代频繁

改造适配

在使用新平台上云的系统中，对于本身已经采用微服务开发组件开发的应用，可不改变应用架构，通过改造适配的方式接入云上微服务套件 EDAS 与云上容器平台 ACK，进行组件级云原生适配。

- 服务治理，通过微服务管理平台进行访服务治理，实现优雅上下线、流量动态调度。
- 中间件服务，解决缓存、调度、事务、消息队列等的分布式高并发、高可用场景
- 可观测性，通过非侵入式埋点、日志采集实现链路跟踪、日志和业务的监控。
- DevOps，实现开发、测试、诊断、部署的全流程自动化与可视化管理。

改造模式

改造模式分为以下两类：开发态改造与运行态改造，以下进行详细说明：

- **开发态改造：**
 - 应用系统开发相关，代码级别适配，一次性改造适配。
 - 平滑改造：相同对接协议，代码少量或无需更改（成本低），如从开源 MySQL 数据库到专有云 PolarDB-M 改造。
 - 适配改造：不同对接协议，代码需进行适配修改（成本高），如从消息中间件 Kafka 到专有云 RocketMQ 改造。

• 运行态改造

- 应用系统运行环境相关，通过配置改造进行适配，可先上云、后适配。
- 运行态改造是部署架构的改造，不涉及应用代码本身，对于开发人员和应用运维人员都会带来一些学习和适应成本。以 EDAS 改造和 ACK 改造为例：
 - **EDAS 产品**
 - 提供完整的微服务治理能力，从应用创建、配置、部署、启动、扩缩、下线、回滚、停止到删除的全生命周期管理，提供分批次的发布机制，支持基于虚机或容器的部署，支持应用分组、命名空间、蓝绿灰度、弹性伸缩等多种部署机制，可快速接入当前微服务组件开发（SpringCloud、Dubbo）的应用，具备全面的微服务治理能力。
 - 适用场景：对于使用微服务架构开发的应用如 Springcloud、Dubbo，建议适配 EDAS 微服务套件，利用 EDAS 微服务套件提供的多种微服务治理功能，完善应用微服务治理能力；对于未使用微服务架构开发的传统单体应用，建议使用 EDAS 进行应用托管，利用 EDAS 提供的应用生命周期管理能力，提升应用管理性。
 - **ACK 产品**
 - 提供全托管的 K8S 平台，简化集群的搭建和扩容等运维工作，整合阿里云虚拟化、存储、网络和安全能力，打造云端最佳的 Kubernetes 容器化应用运行环境。ACK 在云上配合 ACR 镜像仓库使用。
 - 适用场景：当前已使用 K8S 部署的容器化应用，可使用 ACK 平台部署；未容器化的微服务架构应用建议上云后再进行容器化改造；未容器化的普通应用按需进行容器化改造。

新平台上云方式技术价值较高，可以充分发挥云计算技术的能力，提供更高效、更智能的管理运维能力。

（3）重构（Refactor）上云

重构方式上云，通常是由业务需求驱动，将应用系统基于云平台提供的云原生能力进行全面的重构，构建弹性、高效的云上业务环境，通过领域模型划分业务域，助力业务快速迭代演进。

重构上云典型场景包括单体应用的微服务架构改造，应用的容器化和 Serverless 化整体开发和交付架构改造。对于老旧的开发语言和框架，或者是较难维护的应用系统技术架构，或者不满足政策法规要求的应用系统，或者已经有重构计划的应用，我们也建议采取新架构上云。

重构上云使用场景如下：

- **重要核心的业务系统**
- **系统已经不能满足业务发展**
- **有大并发量访问的场景**
- **对业务的连续性、稳定性要求很高**

重构上云整体流程包括调研分析、架构设计、开发迭代、系统集成 & 部署、运营 & 治理 5 个阶段。

- **调研分析：**新架构上云需做调研分析，只有对业务目标进行合理的调研，按照业务技术架构完成上云的需求分析，识别处痛点 / 短板，以及相应的风险，避免初期考虑不全，后续返工或者重新治理，尽量确保调研信息能够支撑应用系统满足架构设计的安全、稳定、成本、效率等各方面的需求。
 - **架构设计：**做好云原生的架构设计，要充分理解云原生的设计理念，本质上是通过云原生帮助企业获取技术红利，取得业务价值。技术栈和云产品选型，建议采用云上已有的云原生产品能力，能够对开源进行很好的适配和加强。微服务 / 云原生架构设计，需涵盖微服务治理，服务网关及分布式事务，此外还要完成弹性伸缩方案设计，应用高可用方案设计，包括限流、熔断和降级等。
 - **开发迭代：**遵守开发规范，比现代应用开发 12 范式，同时需要结合核工业数据中心 CI/CD 流水线，制定新架构的持续集成方案，性能及架构优化的配套能力，搭建适配新架构的测试体系。
 - **系统集成和部署：**云上部署之前需要进行资源评估和准备，部署上线切割前需完成性能压测及容量评估，应用发布管理需支持分批或者灰度发布，对于需要考虑部署环境的容器集群规划和治理。
 - **运营和治理：**新架构上云需要持续运营和治理，包括建立可观测性治理体系，安全和流量治理，更重要的还需要制定规范的运维体系，保障应用系统云上连续稳定的运行和持续运营治理。
- 重构上云方式技术价值很高，让业务全面拥抱云原生，从开发到运维提供端到端的应用管理能力，助力业务更快更敏捷的迭代，提升应用稳定性。

5. 结束语

通过应用系统上云现状分析、上云策略评估、上云路径规划、上云方案设计的“4 步走”策略，为用户定制一套系统、成熟、可落地的上云规划解决方案，为用户后续开展上云实施提供战略性指导。

第二章 上云实施篇



第一节 /

“因地制宜”推进应用上云 —— 上云实施的关键因素分析及迁移 方案设计

1. 应用系统上云实施规划

优先将应用系统进行云化部署，已成为当前大型企业上云的必经之路。根据应用系统的生命周期，可以将其分为两大类：存量应用系统和新建应用系统。

针对存量应用系统上云，通常按照以下四个步骤开展：

- 应用系统上云现状分析：主要完成 IT 基础设施现状调研、技术架构调研、应用架构调研和业务架构调研，发放调研信息，整理搜集汇总，修正及验证。
- 应用系统上云策略评估：主要完成上云评估，包括上云策略的定义、场景及价值阐述，上云决策流程及评估结果展示，并对应用系统上云部署做平台选型推荐。
- 应用系统上云路径规划：基于“敏稳双模”制定应用系统上云实施原则和标准，通过定性及定量评估，按照业务方向给出应用系统上云迁移的路径图。
- 应用系统上云方案：制定适配应用系统上云的三种上云方案，并针对系统上云提供流程建议及组织保障。

针对新建应用系统，我们建议按照以下步骤推进：

- ① 确认系统场景分类：首先根据应用系统的具体需求和特性，明确其所属的场景类别。
- ② 完成云平台选型：基于系统场景分类，选择最适合的云平台，确保其功能和服务满足业务需求。

CHAPTER 02

</上云实施>

- ③分解云优先策略：根据选定的云平台，制定详细的云优先策略，并推荐相应的云上参考架构。
- ④实施云上部署：依据参考架构，执行具体的云上部署工作，确保系统顺利迁移和高效运行。

2. 应用系统上云现状分析

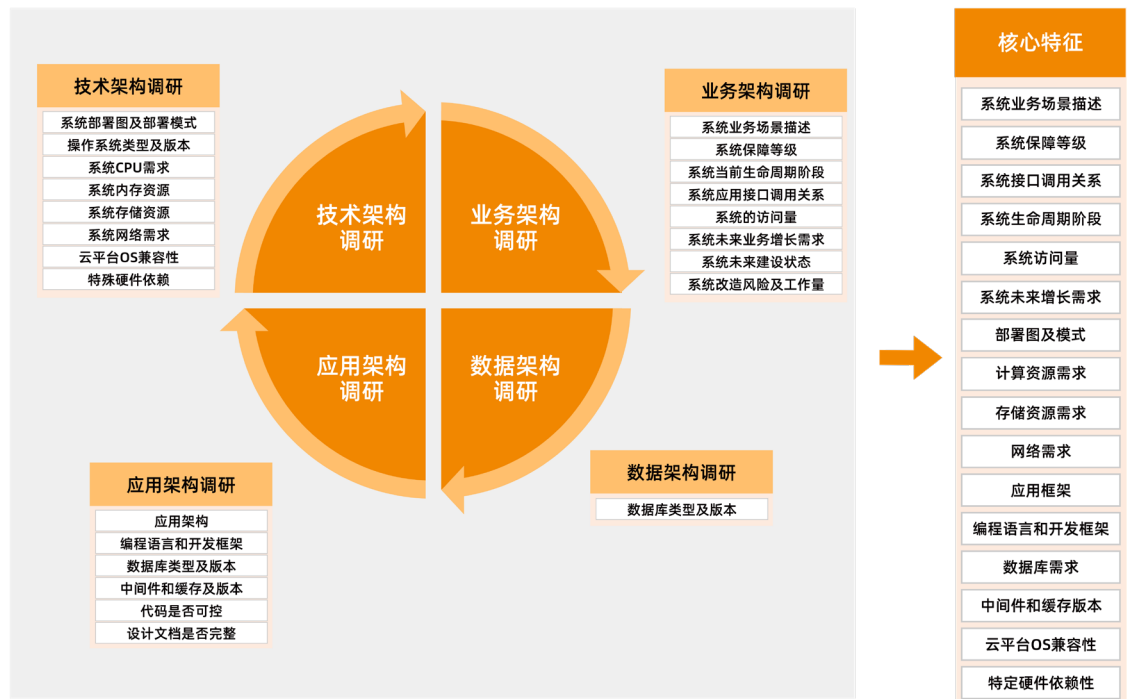


图 1 应用系统上云现状调研分析

从技术、应用和业务三个维度 29 项信息开展应用系统现状调研：

调研维度	业务调研	技术架构调研	应用架构调研
关注点	关注业务类型、业务特征及业务性能指标、用户类型及业务生命周期等	关注应用整体架构、部署架构、资源需求及云兼容性	关注自研可控性，应用架构、开发语言框架、中间件、数据库及缓存需求

调研内容	1. 应用系统名称	1. 部署图	1. 应用系统代码是否可控
	2. 应用系统接口人	2. 操作系统要求	2. 设计文档是否完整
	3. 应用系统业务场景描述	3. CPU	3. 应用架构类型
	4. 应用系统重要性	4. RAM	4. 开发语言和框架
	5. 作为调用方在调用关系表中出现次数	5. 静态数据存储类型	5. 中间件类型和版本
	6. 作为提供方在调用关系表中出现次数	6. 数据量级	6. 数据库类型和版本
	7. 当前生命周期阶段	7. 是否提供公网服务	7. 数据库数据量级
	8. 当前应用系统是否可以满足未来（1~2 年内）业务增长需求	8. 是否使用专属硬件	8. 缓存类型和版本
	9. 应用系统的访问量	9. 硬件依赖特殊要求	
	10. 应用系统进程数	10. 应用系统上云意愿	
		11. 其他方面的痛点	

表 1 3 个维度 29 项调研表

通过调研的三个维度信息整合并抽象出 16 项核心特征，作为应用系统上云评估的输入：

- 系统业务场景描述
- 系统保障等级
- 系统接口调用关系
- 系统生命周期阶段
- 系统访问量
- 系统未来增长需求
- 部署图及模式
- 计算资源需求
- 存储资源需求
- 网络需求
- 应用架构
- 编程语言和开发框架
- 数据库需求
- 中间件和缓存版本
- 云平台 OS 兼容性
- 特定硬件依赖性

（1）现状调研分析 – 业务架构

• 应用系统自研比例

根据应用系统是否为自主开发，合作开发及代码外包的具备软件自主控制权；成熟产品包括采买的成熟软件，随着“一云多芯”进程的逐步推进，安全可控的比例会进一步提升，为应用系统迁移上云提供了技术保障。

• 业务耦合度情况

通过应用接口调用情况对业务耦合度进行调研，按照调用关系数量排序，应用系统相关的接口数量越多，复杂度越高，业务的耦合度越高，上云的成本越高。

• 系统建设状态调研

通过整理应用系统建设状态，重点关注下线或即将下线的的应用系统，上云路径规划中建议忽略这些应用系统。

（2）现状调研分析 – 应用架构

• 应用架构情况

通过调研存量系统的应用架构，对于单体和微服务架构应用存在的主要问题分析：

- 单体应用，传统三层架构，部分应用无法保障业务高峰期的系统可用性，建议在上云过程中对高性能、高并发的系统进行优化升级。
- 微服务架构是否有整体技术方案，例如可观测性能力、微服务治理能力等。

（3）现状调研分析 – 数据架构

• 数据库使用情况

应用系统是否使用关系型数据库主要包括 Oracle、MySQL、SQLServer、PostgreSQL 类型；SAP 系统比较特殊使用 HANA 数据库；以及是否使用非关系型数据库 MongoDB。

（4）现状调研分析 – 技术架构

• 操作系统整体情况

对应用系统使用的操作系统进行分类调研：

- Windows 类应用随着“一云多芯”能力的要求将逐渐减少。

- Linux 类应用将逐步从商业向开源、国产操作系统转变。
- 存在少量应用使用特殊的专有操作系统，云兼容性差，无法迁移上云。

• 特殊硬件依赖情况

通过调研统计，对部分应用系统特殊硬件依赖，比如加密狗、USBkey 等，特殊的硬件依赖影响应用系统的上云策略。

3. 应用系统上云迁移策略评估

参考前文介绍的 6R 模型制定上云策略。

建议主要采用新托管、新平台和新架构这三种方式上云：

• **新托管（Rehost）**：将线下物理机替换成为云上虚拟机或者裸金属实例，不改变原有的运维方式，使用云主机替换本地机器，风险最小，成本最低。

• **新平台（Replatform）**：利用托管云服务替换线下自建应用基础设施，获取更好的弹性、稳定性和自治运维能力，使用托管云服务替换自建的应用基础设施，最小代价享受云的红利。

• **新架构（Refactor）**：重构应用系统，基于云平台构建云原生架构，使用微服务重构应用架构，使用容器重构交付架构，使用 DevOps 提升研发效能，最大化云的价值。

（1）应用系统上云迁移 6R 评估模型

参考前文介绍的 6R 模型制定上云策略。

总体上云迁移评估流程：以 16 项核心特征作为输入，通过评估决策模型，经过 12 项条件判断，最终得出应用系统上云策略。

（2）应用系统上云部署环境建议

根据应用程序业务特性，并结合场景需求，推荐应用系统迁移上云后的环境部署，选择合适的目标云平台，主要的参考依据如下所示：

	专有云环境部署	公共云环境部署	混合云环境部署
部署环境特点	企业内网环境建设、使用和管理 构建企业内网访问的资源池 在企业内部进行数据控制 基础设施归企业所有 受到企业的防火墙保护	关注应用整体架构、部署架构、 资源需求及云兼容性	构建与企业内网和公共云之间 企业内网和互联网混合的资源池 业务数据可做敏感数据分离部署 按需进行资源付费 通过安全专线或 VPN 于企业和 公共云之间
应用系统场景	存放敏感数据 稳定可靠工作负载 业务稳定，适度的需求变动 重点服务于企业内部客户	存放非敏感数据 弹性、突发性的工作负载 解耦的系统满足多样性需求变动 借助公共云环境服务于全球客户	敏感和非敏感数据可分离部署 弹性负载在公共云，稳定负载在 内网 借助混合模式满足业务部分需求 变动 借助于 CDN 服务，降低最终用 户访问延迟

表 2 应用系统上云部署环境建议

按照上述原则，将业务系统分为两大类，分别进行适配：

- 针对应用系统数据敏感度高，工作负载稳定，有写数据库的场景，主要服务企业内部用户，建议专有云部署。
- 针对业务数据可实现敏感和非敏感数据分离部署，静态类数据可借助于 CDN 服务，服务全球客户，一些读为主的业务应用弹性负载在公共云，核心应用和数据库稳定负载部署在内网，建议混合云部署。

4. 应用系统上云方案

本章主要阐释应用系统上云技术方案，存量与新建应用系统上云方案在不同场景下链路不尽相同，对于存量系统建议上云的路径如下：

- ① 替换：购买 SaaS 版本软件进行替换后云上部署。
- ② 新托管：主要完成镜像和数据迁移，验证完毕后，实现云上部署。
- ③ 新平台：关注云上组件选型及适配，进行功能和性能测试后云上部署。
- ④ 新架构：通过微服务和云原生架构改造，新架构的开发迭代，基于云上原生服务开发和部署，此外重点考虑应用系统部署完成后的云上运营和微服务治理。

对于新建应用系统上云，分为两类：

- ① 自研软件，优先微服务和云原生架构适配，采用新架构方案上云，对于传统单体应用，建议采用传统三层架构云上部署。
- ② 外采的商业软件，优选先用云兼容版本，无法云化部署的，暂时保留传统环境。

（1）新托管（Rehost）通过成熟的应用及数据迁移解决方案实现快速上云

新托管（Rehost）又称为平迁上云，简单将线下物理机替换成为云上虚拟机或者裸金属实例，不改变原有的运维方式。将云下 IDC 的服务器或应用程序直接整体迁移到云环境之中，通常有 P2V, V2V 两种方式，从本地物理机和虚拟机（VMware/，通过全量和增量迁移方式，迁移到专有云，最后完成数据的一致性校验。

以阿里云为例，迁移工具可以采用阿里自研的专有云迁移中心中的服务器迁移功能，该功能是由客户端和控制台两部分组成，以将源服务器迁移为云服务器镜像为例，具体分类如下：

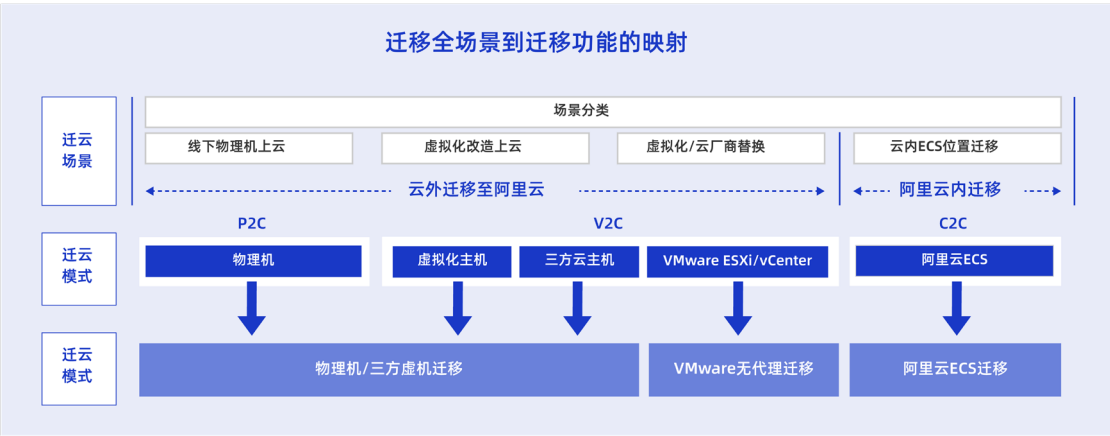


图 2 迁移全场景到迁移功能的映射

a. VMware ESXi/vCenter 虚机的无代理迁移功能：

- **虚机无侵入迁移**：被迁移的 Vmware 虚机中无需部署 Agent 客户端程序，迁移过程不占用虚机系统资源，消除业务安全顾虑。
- **不停机增量迁移**：在自定义时间间隔内，将源 Vmware 虚机产生的增量数据同步至阿里云 ECS，可有效减少源 VMware 系统业务暂停时间及最终割接时间。

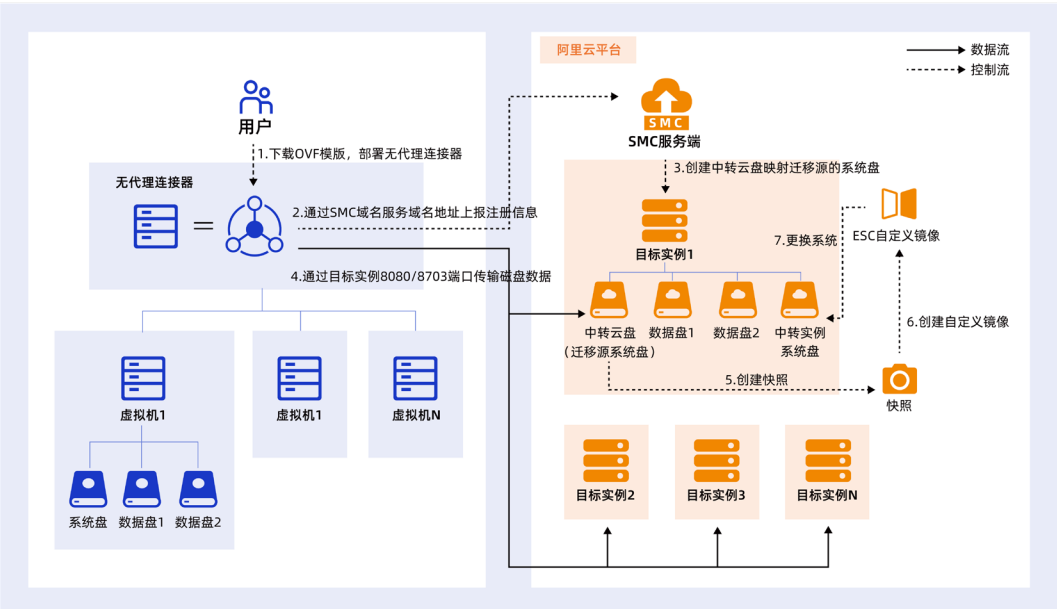


图 3 VMware ESXi/vCenter 虚机的无代理迁移

b. 云内阿里云 ECS 迁移功能：

- **支持场景丰富：**支持同朵云跨 Region/AZ 迁移、同朵云内复制容灾环境、同朵云内从 X86 向 C86 原地轮转。
- **操作简单：**免安装 Agent 客户端、迁移效率高、无缝变配。



图 4 云内阿里云 ECS 迁移

c. 物理机 / 三方虚机迁移功能：

- **安全传输：**迁移过程数据默认采用加密传输，安全有保障，支持断点续传、压缩传输。
- **不停机迁移：**支持不停机在线迁移，不影响源系统运行。
- **通用性强：**与迁移源主机的形态、厂商无关，支持主流 windows 和 Linux 版本，可作为兜底迁移策略。

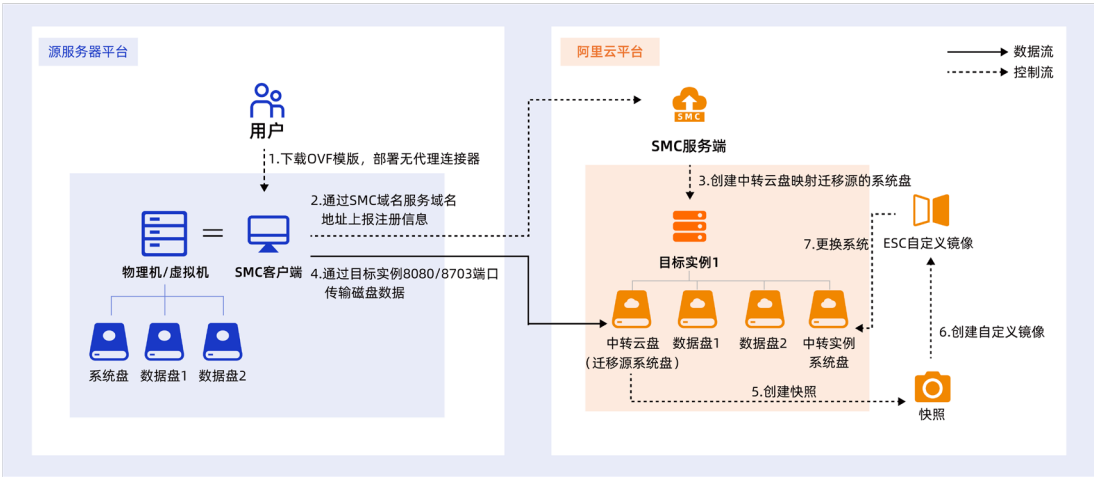


图 5 物理机 / 三方虚机迁移

(2) 新平台（Replatform）的上云过程中要更多的关注对标云产品的选型

新平台（Replatform）是指利用托管的云服务替换线下自建应用基础设施，托管的云服务通常提供更好的弹性、稳定性和自治运维能力。

在新平台 Replatform 的上云中关注对标云产品的选型，使用云产品的优势主要是：

- ①对开源友好：云产品获得了开源社区的支持或采用了与兼容开源的技术方案，可以保障业务的研发投入的稳定性。
- ②产品功能能持续演进 / 迭代：云产品会持续研发，持续演进与迭代，可以保障产品和系统架构的延续性。
- ③有大量客户侧最佳实践：云产品只有较多规模化落地验证的客户侧最佳实践的产品，才能支撑客户业务云上的高性能与稳定性要求。

当然，业务需求才是先导，技术支撑是为辅助。所有的产品选型都需要满足业务诉求，按照业务驱动为主，技术栈的选型与业务诉求进行适配，下表是常见的基础类型的组件和其云上的适配：

类型	厂商或产品	类型	云上组件匹配	优势
网络	F5、Nginx 等	负载均衡	SLB	相比硬件 F5 负载均衡成本更低，不需要单独配置监控
计算	VMware VM、物理机等	应用服务器	ECS	规格更加灵活，并支持弹性伸缩和扩展
存储	EMC 或 XSky 等	文件存储	块存储和 OSS	成本降低，可靠性提升，存储类型更丰富
数据	Oracle 等	数据库	RDS 或 PolarDB	云上 PaaS 型数据库，扩容和负载成本低，管理维护资源更丰富
缓存	Redis 等	缓存	Redis PaaS 版	更加灵活的部署规格，PaaS 形式提供

表 3 常见基础类型组件和对应云上适配参考

（3）新架构（Refactor）建议采用云原生 / 微服务架构进行重构上云

新架构（Refactor）：典型场景包括单体应用的微服务架构改造，应用的容器化和 Serverless 化整体开发和交付架构改造。

新架构（Refactor）方式上云，通常是由业务需求驱动，例如在现有架构难以满足业务在新增功能、可扩展性、性能等方面需求，一般需要重新思考应用程序的架构和开发和交付方式，并且尽可能多的通过云原生的服务来赋能支撑业务，通过应用系统上云，从六个方面满足业务诉求：

- ①应用架构，基于云原生 / 微服务技术完成重构，实现轻量、弹性、敏捷的应用架构。
- ②服务治理，通过微服务管理平台进行访服务治理，实现优雅上下线、流量动态调度。
- ③中间件服务，解决缓存、调度、事务、消息队列等的分布式高并发、高可用场景。
- ④可观测性，通过非侵入式埋点、日志采集实现链路跟踪、日志和业务的监控。
- ⑤DevOps，实现开发、测试、诊断、部署的全流程自动化与可视化管理。
- ⑥基础设施，云原生驱动基础设施向资源 / 架构双统一的方向迈进，感知并反馈业务，并根据业务伸缩变形。

对于老旧的开发语言和框架，或者是较难维护的应用系统技术架构，或者不满足政策法规要求的应用系统，考虑新架构上云，如采用云原生 / 微服务架构进行重构后上云，对于新架构（Refactor）上云的流程，一般的流程如下：

- ①调研分析：新架构上云需做调研分析，只有对业务目标进行合理的调研，按照业务技术架构完成上云的需求分析，识别处痛点 / 短板，以及相应的风险，避免初期考虑不全，后续返工或者重新治理，尽量确保调研信息能够支撑应用系统满足架构设计的安全、稳定、成本、效率等各方面的需求。
- ②架构设计：做好云原生的架构设计，要充分理解云原生的设计理念，本质上是通过云原生帮助企业获取技术红利，取得业务价值。技术栈和云产品选型，建议优先考虑采用云上已有的云原生产品能力，能够对开源进行很好的适配和加强。微服务 / 云原生架构设计，需涵盖微服务治理，服务网关及分布式事务，此外还包括弹性伸缩方案设计，应用高可用方案设计，包括限流、熔断和降级等。
- ③开发迭代：遵守开发规范，比现代应用开发 12 范式，同时需要结合业务自身的 CI/CD 流水线，制定新架构的持续集成方案，性能及架构优化的配套能力，搭建适配新架构的测试体系。
- ④系统集成和部署：云上部署之前需要进行资源评估和准备，部署上线切割前需完成性能压测及容量评估，应用发布管理需支持分批或者灰度发布，对于需要考虑部署环境的容器集群规划和治理。
- ⑤运营和治理：新架构上云需要持续运营和治理，包括建立可观测性治理体系，安全和流量治理，更重要的还需要制定规范的运维体系，保障应用系统云上连续稳定的运行和持续运营治理。

（4）上云架构与方案推荐

企业的 IT 架构必须具备高度的灵活性、弹性和稳定性，以支持业务的快速发展和持续创新。为此，我们提出以下三种面向不同需求的上云架构及推荐方案：

- ①面向微服务的云原生架构：通过容器化开发、CI/CD 流水线、微服务治理、蓝绿部署和灰度发布等技术手段，实现系统的高可用性和快速迭代，有效提升研发效率和服务稳定性。
- ②面向业务迭代快速增长的云上技术架构：利用容器服务 ACK、弹性扩缩容、全链路压测等技术，确保系统能够在业务高峰期自动进行资源扩容，并在低谷期自动进行资源缩容，实现高效、低成本的运维管理。

③**面向快速迭代、稳定发布的服务分批发布治理方案**：采用分批次滚动更新、手动和自动发布模式相结合，确保每次发布的安全性和可控性，降低故障风险，保障业务连续性。

(4.1) 面向微服务的云原生架构

- 基于容器开发，结合 CI/CD 流水线实现云上持续集成和快速部署
- 采用微服务架构，增强系统韧性，提升软件的平均无故障时间
- 通过微服务治理，增强微服务线上稳定性，提升研发效率
- 云原生和微服务网关，蓝绿部署和灰度发布，支持系统限流降级
- 可观测性，提升系统稳定性，通过数据分析，全链路追踪，使用 Prometheus 监控和日志服务，实现可观测性。
- 基于分布式缓存，消息队列和分布式任务调度，提供高可用的中间件服务。

面向存量系统新架构与新建系统敏态、服务化场景需求下，建议可采用云原生 / 微服务技术架构，并遵循以下六大原则：

a. 服务化原则

通过模块分离、面向接口编程实现服务内部功能的高度内聚与复用，并让业务以标准化 API 等方式进行；服务间采用事件驱动的方式集成，减小相互依赖，完成基于服务流量的策略控制和治理。

b. 弹性原则

系统部署规模随着业务量的变化自动伸缩，业务面临海量突发性扩张时从容不迫；自动根据业务峰值、资源负载扩充或者收缩系统的规模。

c. 所有过程自动化原则

通过大量自动化交付工具在 CI/CD 流水线的实践，实现软件交付与运维的自动化。

d. 架构持续演进原则

云原生架构应该具备持续演进能力，在业务高速迭代下实现业务与架构的平衡关系。

e. 可观测原则

从传统的日志方式、监控、APM 到链路跟踪、服务 QoS 度量，通过指标采集、非侵入式埋点、日志采集实现业务、应用监控，全链路跟踪；任何 IT 设施中的软硬件发生错误后能够被快速修复，从而不会让这样的错误对业务带来影响。

f. 韧性原则

从架构设计上，除了包括服务化中常用的熔断、限流、降级、自动重试、反压等特性外，需要考虑包括服务异步化、重试 / 限流 / 降级、高可用、跨地域容灾等韧性能力。

(4.2) 面向业务迭代、快速增长的云上技术架构

- 通过容器服务 ACK 实现业务应用层面的弹性，云托管数据库按需弹性扩缩容。
- 搭配使用各类监控工具、压测工具实现全链路压测应对大促预演。
- 轻松搞定如大促等瞬时高峰的弹性，做到自动部署、高可靠性保障、运维工作量低、成本低。

面向业务迭代、快速增长的需求下，建议采用面向极致弹性的云原生技术架构，方案优势如下：

- ① 日常流量低成本：非业务峰值，保持少量节点维持日常业务。业务容器化，极大的简化部署。业务故障自愈高可靠，弹性伸缩等能力，业务的全生命周期管理，简化运维工作量。
- ② 峰值时极速扩容：业务峰值，原有工作节点已不能满足当前的业务需求，需在大促峰值前快速扩容工作节点的数量。根据自动伸缩的规则或还可根据预估的业务峰值时间，触发业务自动扩容，自动调度，保障应用高可用。
- ③ 策略护航：当业务峰值结束后，云上容器服务会根据 hpa (horizontal pod autoscaler) 的能力将业务进行自动缩容，业务缩容后，集群自动伸缩规则会缩容释放工作节点。
- ④ 数据库弹性：云数据库实现纵向伸缩、读写分离、弹性升级。

(4.3) 面向快速迭代、稳定发布的服务分批发布治理方案

- 按照一定的批次，每次只对应用的一部分实例进行升级。面对故障可以终止变更过程并进行回滚，待问题修复后重新发布。
- 解决了单批发布中“无法中断”的痛点，并且一定程度的降低了发布的风险。
- 通过手动和自动两种发布模式可以更加灵活的进行应用发布。

面向稳定发布、故障防治的需求下，建议采用服务分批发布，方案优势如下：

- a. 服务发布时可以“暂停”更新，通过手动方式或者自动方式分批次“滚动更新”。
- b. 服务发布出现问题时，可以立即回滚，降低“服务异常”带来的损失。
- c. 相较于“蓝绿发布”等服务治理方案更节省资源。

5. 数据上云典型场景和关键因素分析

（1）数据上云的典型场景

- 数据库开源自建上云

开源 MySQL、PostgreSQL 迁移到云上 RDS-MYSQL、RDS-PostgreSQL

- 数据库替换上云

Oracle、DB2、TD、SQLServer 迁移到 PolarDB-PG、PolarDB-X

- 数据库分布式改造上云

Oracle、DB2、TD、SQLServer 迁移到分布式 PolarDB-X

- 开源大数据同构迁云

开源 CDH (hadoop、hive、spark、Flink) 迁移上阿里云 EMR

- 大数据改造迁云

开源 CDH (hadoop、hive、spark、Flink) 迁移上云 MaxCompute、Hologres、Flink

（2）数据上云的关键因素分析

（2.1）云上云下数据质量完整性

确保云上和云下数据质量的完整性对于维持企业的业务连续性、支持准确的商业决策等方面都至关重要。任何数据质量问题都可能导致业务中断、成本增加、声誉受损等严重后果。必须重视这些问题，并采取相应的预防和纠正措施，以确保数据在所有环境中都能保持高质量和一致性。

数据上云过程中数据不一致对客户有什么影响呢？

a. 业务中断

- 服务不可用：关键数据的丢失或不一致可能导致某些服务无法正常运行，直接影响客户的日常运营。
- 交易失败：对于电子商务、金融等依赖实时数据处理的行业，数据问题可能导致交易失败或延迟，进而影响收入。

b. 客户信任受损

- 用户满意度下降：当客户发现他们提供的信息不再准确或完整时，可能会对企业的可靠性和专业性产生怀疑，降低品牌忠诚度。

- 投诉增加：数据问题可能导致客户服务请求和投诉量激增，增加客服成本并损害企业形象。

c. 财务损失

- 直接经济损失：由于业务中断、交易失败或需要额外支出修复数据问题，企业可能遭受直接的经济损失。
- 间接经济损失：长期来看，客户流失、市场份额减少以及品牌形象受损都可能带来更大的间接经济损失。

d. 运营效率低下

- 重复工作：员工可能需要花费额外的时间来验证和纠正数据问题，降低了工作效率。
- 流程受阻：依赖于准确数据的工作流程可能会被阻塞，进一步拖慢整体业务进展。

e. 合作伙伴关系紧张

- 供应链中断：对于涉及多个合作伙伴的企业来说，数据问题可能导致供应链中的沟通和协作出现问题，影响整个链条的运作。
- 合作关系恶化：长期存在的数据问题可能削弱与其他公司之间的信任，影响未来的合作机会。

（2.2）云上云下业务体验一致性

确保云上和云下数据库性能的一致性对于维持业务体验的一致性和稳定性至关重要。试想如果数据库上云后，发现性能比云下差了很多，那业务体感肯定非常差，表现为 RT 响应变长、资源负载变高、处理队列堆积等，从而导致业务崩溃。

那么，如何确保云上和云下数据库性能的一致性，以保障业务体验一致性呢？

a. 统一架构设计

- 标准化技术栈：尽量使用相同版本的数据库引擎、框架和技术组件，减少环境差异。
- 模式同步：确保云上和云下的数据库模式（表结构、索引等）保持一致，避免因结构不同导致的性能问题。

b. 性能基准测试

- 迁移前后对比：在数据迁移前后进行详细的性能基准测试，包括读写速度、查询响应时间、并发处理能力等关键指标。
- 模拟真实负载：创建与生产环境相似的工作负载场景，评估系统在高压下的表现，确保两者都能稳定运行。

c. 持续监控与分析

- 部署统一监控工具：使用如 Prometheus, New Relic, Grafana 等工具，在云上和云下环境中部署相同的监控解决方案，收集和比较性能数。

d. 自动化运维与调优

- 自动扩展：启用自动扩展功能，根据流量动态调整资源分配，确保高峰期的服务性能。
- 性能调优：定期进行性能评估，识别瓶颈并实施针对性的优化措施，比如调整索引、优化查询语句等。

6. 小结

本文系统分析了上云实施的关键因素，并对典型的上云架构进行了方案推荐。应用上云过程涵盖了应用系统规划、现状评估、迁移策略选择及上云方案等环节。同时，针对数据高效、完整上云的挑战，更需要采用工具化和标准化方法论，以应对异构数据库的不兼容性，确保数据质量和业务平滑迁移。

接下来，我们将深入探讨具体的应用上云和数据上云概念、场景、产品、服务和最佳实践，让企业降低上云成本，提升上云质量和效率。

第二节 /

应用上云最全指南——从制定方案到最佳实践

1. 上云政策解读

云计算已经成为数字经济先进生产力，国家和地方相继推出了各种政策，推动政企客户业务系统全面上云。

国家发改委印发《“十四五”数字经济发展规划》，提到加快企业数字化转型升级。引导企业强化数字化思维，提升员工数字技能和数据管理能力，全面系统推动企业研发设计、生产加工、经营管理、销售服务等业务数字化转型。推行普惠性“上云用数赋智”服务，推动企业上云、上平台，降低技术和资金壁垒，加快企业数字化转型。

工信部印发《推动企业上云实施指南（2018-2020 年）》，上云企业可依托云平台服务商或第三方机构，明确各信息系统具体迁移策略。对于复杂系统的迁移上云，需要根据实际情况采用定制化的迁移技术及方法。迁移策略包括：

- **直接迁移**：将信息系统迁移部署到云平台，利用统一运营管理平台进行管理；
- **改造后迁移**：对数据库、系统架构、运行环境、接口等进行改造，使其满足迁移到云平台的技术要求后迁移；
- **采购云服务重建**：结合业务实际，采购满足需求的各类云服务，重新构建信息系统；
- **保持现状**：对暂不适合迁移的系统，继续保持运行在当前环境。

人民银行印发《金融科技发展规划（2022-2025 年）》，金融业数字化转型更深化。金融业数字化从多点突破迈入深化发展新阶段，全局性、系统性数字思维深入人心，数字化转型的理论、方法、评价体系基本形成，上云用数赋智水平稳步提高，金融机构数字化经营能力大幅跃升。

北京市经信委印发《北京市市级政务云管理办法》，除公安、安全等部门以及涉密和信息安全等级保护四级（含）以上信息系统外，按照“上云为常态、不上云为例外”原则，各部门现有信息系统应逐步迁移上云，停止服务器、存储等相关软要件采购。

安徽省印发《“数字政府”建设规划（2020—2025年）》，构建逻辑集中的，1个省级政务云平台。统一为省级政务部门提供计算、存储、网络及安全等各类云计算基础设施资源、相关软件和服务…省直部门已建非涉密系统逐步迁移上云，新建业务系统依托省级云平台进行集约化建设部署。

江苏省工信厅印发《江苏省星级上云企业评定工作指南》，对企业上云水平进行评定，大力推动企业上云。

2. 上云趋势分析

政企数字化转型全面进入“上云”时代。未来，随着我国政府和企业业务创新、流程重构、管理变革的不断深化，同时伴随数字化、网络化、智能化转型需求的提升，将带动政府和大型企业上云趋势加速发展，上云将进入常规态化阶段。预计接下来上云深度将进一步加大。

- **政府：**我国政府的云化比例已经很高，公共服务实现一网通办，便民利民；接下来，政务云平台需要对资源进行集约管理，支持业务创新，并探索对外行业云的运营售卖。
- **金融：**从早期的互联网金融、手机银行上云，然后是业务中台、数据中台上云，再到传统业务上云、核心分布式改造。金融客户对于安全性和可靠性的要求非常高，需要前沿的技术支撑，并需要实现安全可控与自运维，以满足自身对于技术的可控性。
- **企业：**央国企大型集团往往拥有庞大的分支机构，希望通过“物理分散，逻辑统一”的架构来匹配总部对于省市公司既要统一管理，又要最大化激发企业自主活力的组织运作模式；同时，大型企业目前需要成熟的行业解决方案，逐步从公共服务上云，过渡到经营管理系统上云，再将生产管理核心应用迁移上云，不断加深云化。

根据国务院发展研究中心国际技术经济研究所发布的《中国云计算产品白皮书》，政企上云四大趋势：

- **政策指引转向深度上云用云，标准建设拓展至新技术应用。**
- **产业集聚效应明显，布局发展从东部向中西部扩散。**

- **市场需求持续更迭，多种部署模式并存发展。**
- **行业应用水平参差不齐，阶梯状发展特点明显。**

3. 上云价值体现

当前政企客户上云核心驱动力，一方面是一云多芯加速传统IT云化进程，另外一方面是要降低传统架构为主的双模IT成本。总结传统业务上云的价值包括以下几点：

- ①低成本高效率：规模化、共享化、弹性化、系统化、自动化，实现降本增效。
- ②敏捷快速迭代：分布式微服务、容器云原生敏DevOps、PaaS平台化。
- ③保持IT现代：敏态IT组织人才转型、运维到运营、业务IT融合、安全可控。
- ④平台化拓展场景：开放连接、拓展场景、运营用户流量、提升用户体验。
- ⑤数智化赋能创新，积累数据资产、建设数据中台、提升业务精准性和创新力。

政企客户上云需求，可总结为安全可控、安全稳定、多元算力、兼容开放、平滑迁移。

- **安全稳定，统一基础设施平台**
 - 安全可控不被绑定，可扩展、可集成、可持续演进。提供成熟稳定的自研云平台，具有扩展性强、持续演进，兼容服务器、交换机、操作系统等主流软硬件，不受开源技术绑定或者限制。
 - 核心业务全面上云，一站式业务上云，全业务容灾。满足业务连续性、金融监管、新等保要求，实现低技术门槛，低成本，高效率业务上云。
 - 多元算力统一调度，支持全域全场景部署形态，保证用户一致性体验。提供高性能、低延时的裸金属、虚拟机、容器计算资源，满足不同业务部署需求，同一控制平面计算资源池实现统一资源编排调度。
- **兼容开放，精细化运营控制台**
 - 标准化开放接口，易集成易定制。云平台可以提供标准化API接口，可以对第三方应用、监控运维平台对接集成。
 - 数据库语法高度兼容、高性能，实现快速替换。云上提供的数据库产品，跟云下部署的商用、开源数据库高度集成，迁移上云之后，无需做大量的改造。
 - 中间件产品，无需应用改造快速发布。云下部署的应用程序，无需做应用改造，即可在云平台上快速发布，并实现应用监控告警。

- **应用平滑迁移上云**

- 一站式迁云工具。云下大量的基础资源、数据库历史数据。要求提供可视化、简单易用的迁云工具，可以平滑的把线下部署的应用、数据库迁移到云上。
- 专业的迁云专家服务。部分政企客户，没有专业的技术团队来实施迁云工作，要求云厂商可提供迁云方案咨询、迁云落地实施，帮助客户传统业务可以平滑、高效的迁移上云。

4. 应用上云场景

- **场景 1：**关键业务系统、大型软件云上部署。例如：ERP、财务、核心生产系统。
- **场景 2：**金融、证券等行业对业务部署的合规性，以及某些客户对数据安全有苛刻的。例如：证券核心交易系统、行情系统。
- **场景 3：**应用容器化上云、微服务化拆分、单元化改造。比如，银行核心系统云原生改造、快递物流核心业务系统云原生改造。
- **场景 4：**异构计算应用上云，需要基于 GPU 裸金属和 vGPU 虚拟化部署。例如，人工智能、深度学习、机器学习、图形图像处理等场景。
- **场景 5：**应用基于 Oracle 开发迁移上云，要求国产数据库替换，高度兼容 Oracle，无需分库分表，应用只少量改动。
- **场景 6：**应用基于 Oracle 开发平迁上云，要求 Oracle 云上部署，不做数据库替换，应用不做任何改造。
- **场景 7：**应用基于 MySQL 开发迁移上云，要求国产化替换，100% 兼容 MySQL 语法，无需应用改造。
- **场景 8：**应用基于 MySQL 开发平迁上云，客户拥有多家开源 MySQL 生态伙伴，上云后依然使用合作伙伴的开源 MySQL。
- **场景 9：**应用迁移上云，需要快速完成云上应用发布。
- **场景 10：**应用迁移上云，需要对应用监控，实现性能瓶颈监控分析，应用接口调用错误诊断，微服务全链路监控。

本章节总结介绍了应用上云相关场景，针对数据库、大数据、数据仓库等数据类上云场景会在后续章节进行详细介绍。

5. 上云客户画像

a. 客户特征

- 金融、政府、大企业等客户具备以下业务和技术标签客户。
- 业务标签：应用上云、数据库上云、中间件上云、传统 IT 云化、云原生、应用负载。
- 技术标签：分布式、微服务、高性能、高可靠、低延时计算资源池 / 裸金属 / Oracle RAC 上云、一云多芯。

b. 面临痛点

传统业务客户面临的痛点问题主要包括以下几点：

- 业务安全稳定性无保障：传统业务部署，各应用需要自行考虑软硬件高可用，采用堆砌式安全防护，无法实现统一高可用容灾、统一安全防护。
- 合规要求满足安全可控：传统应用大量使用开源数据库、中间件、操作系统产品，必须满足跨平台要求。
- 业务可扩展性差：基础资源供给周期长，对于高性能、大存储容量业务场景，无法实现弹性扩展，影响客户业务发展。
- 资源分配率、利用率低：应用烟囱式建设，资源信息不共享，存在重复投入，资源利用率低，存在资源浪费，建设成本高问题。

6. 方案总体架构

阿里云根据多年客户迁云实际，总结了最佳实践步骤，通过应用系统上云评估模型，结合现有系统业务现状、支撑环境、业务弹性要求、安全风险评估、流程标准化程度、服务标准化程度、迁移工作量等多个维度进行评估，根据应用评估结果制定迁移方案，重点考虑系统上云后对业务的价值及资源消耗情况，以及系统技术层面评估实施难度及风险，从而制定系统迁移的最佳方案，策略如下：

- **阶段一：IaaS 能力引入。**快速降低 IT 运维成本，提高系统弹性 (Re-Host)。将现有应用按照原有的架构迁移到云平台上，引入 IaaS 产品服务，应用基于虚拟机、裸金属部署，利用云服务优势，增加系统弹性和安全性。
- **阶段二：PaaS 能力引入。**快速的帮助业务增长 (Re-Factor)。系统迁移到云上后，引入云数据库等 PaaS 产品及服务，对应用功能进行局部优化，提升系统整体性能和稳定性。

- **阶段三：云原生能力引入。**对系统进行云原生架构改造 (Re-Architect)。将现有的单体核心应用重新设计成为松耦合、模块化的原生应用架构，引入容器、云原生中间件，进行应用容器化、微服务化、单元化改造。
- **阶段四：共享中台能力引入。**对系统进行升级替换 (Replace)。利用云 PaaS 层及生态资源，引入各类型的中台，对原有系统平台的功能进行合并和提升。

结合阿里专有云提供的丰富产品与服务能力，我们为客户打造了一套全面、灵活的传统业务上云解决方案。具体涵盖以下核心领域：

- **基础资源上云：**云上提供丰富的计算存储网络产品，计算资源（裸金属 / 虚拟机 / 容器），存储资源（对象存储 OSS/ 块存储 EBS/ 文件存储 NAS/ 日志存储 SLS），网络资源（专有云网络 VPC/ 负载均衡 SLB）。
- **数据库上云：**云上提供丰富的 OLTP/OLAP/NOSQL 及数据库工具，同时可以实现云上自建商用数据库（Oracle、DB2）或者云上自建开源数据库（MySQL、PostgreSQL）。
- **中间件上云：**云上提供丰富的中间件产品，如容器服务、应用发布 EDAS、应用监控 ARMS 产品。
- **一站迁云中心和迁云服务：**提供一站式迁云中心，帮助客户更高效、更便捷地迁移上云。同时，我们向客户提供迁云最佳实践加持的迁移服务，打消客户缺少迁移经验的顾虑。

7. 应用上云典型场景介绍

（1）场景 1：关键业务上云

- **产品方案**
 - 神龙服务器 + 分布式存储 CDS：采用存算分离架构，通过软件卸载、硬件加速的软硬一体技术，提供高性能、高可用、敏捷弹性计算存储，实现关键核心应用系统云上部署。
 - NVMe 共享云盘：采用多重挂载 NVMe 共享盘，即单个 NVMe 云盘支持同时挂载到多个 ECS，实现多台 ECS 对同一块 NVMe 云盘的并发读写访问，实现应用高可用、高并发。

• 业务场景

- 核心业务上云：云下部署在 PowerPC/ 小型机 /X86 服务器 /VM 上的关键业务或者大型软件，上云之后需要部署到大规格 VM/ 裸金属。
- 应用高可用部署：云下应用需要高可用部署，使用多台 VM/ 物理机 + NAS/SAN 访问同一份应用日志 / 共享文件，上云之后希望有可替代方案

• 业务价值

- 应用高性能：神龙服务器可提供大规格 VM 或者裸金属，实现云上大型软件高性能、低延时部署。
- 应用高可用负载：通过 NVME 共享云盘，替代传统 NAS/SAN 共享存储，实现云上应用高可用部署。
- 应用稳定性：避免资源管控 / 数据面转发，与应用负载间的资源争强导致的抖动、性能下降，保障云上部署应用性能运行稳定。
- 资源利用率高：ECS VM/ 裸金属拥有完整的 CPU/ 内存资源，无管控 / 数据链路的资源占用（10 vcpu），同时相同算力情况下占用更少的机架、交换机数，应用部署综合性价比提升 20% 以上。

（2）场景 2：特殊行业客户上云

• 产品方案

- 神龙弹性裸金属：拥有物理机的完整特性和性能、高隔离性和高安全性，分钟级交付，兼容虚拟机镜像系统，兼容 VPC/SLB/RDS 等云平台全业务，具有与虚拟机一致的使用体验和业务敏捷。
- 独占物理隔离：资源独占、数据隔离、可监管可追溯，满足数据安全和合规监管的业务诉求。

• 业务场景

- 应用部署物理隔离：金融、证券等行业，对业务部署的合规性，对数据安全有苛刻要求，云下采用物理机部署，实现物理资源隔离，上云之后仍然要求满足物理隔离。

- 生产开发测试物理隔离：部分客户各种原因只部署了一朵专有云，需要生产开发测试共用，并且要求生产开发测试应用数据满足物理隔离。

业务价值

- 安全可靠：物理资源独占，性能无损，符合传统企业对于数据安全合规的心智。
- 一致性体验：深度融合物理机和虚拟机特性，可兼顾虚拟机的弹性资源、分钟级交付、全自动运维和物理机的性能无损、完整特性和硬件级强隔离，并全面兼容阿里云生态产品，实现一致性云上客户使用体验。
- 多环境共用：满足部分客户因为预算有限，可实现生产开发测试环境共用，并且满足应用数据物理隔离要求。

(3) 场景 3：客户云原生改造

产品方案

- 神龙裸金属：物理机、虚机、容器等计算资源共池管理，可实现分钟级快速灵活调池，资源利用率高。
- 容器服务 + 容器镜像：支持对命名空间 CPU、内存、存储、容器组等资源配额；支持安全容器拥有独立的内核，具备更好的安全隔离能力。支持 GPU 直通和虚拟化能力。

业务场景

- 应用容器化上云：传统架构云原生改造，第一阶段针对企业现有应用进行分析，先逐步实现应用的容器化上云。
- 应用微服务化拆分：云原生改造第二阶段，针对必要的应用进行微服务拆分改造。
- 应用单元化改造：应用为了满足容灾多活能力，需要针对应用进行应用单元化改造。

业务价值

- 应用高性能：高负载压力下，性能提升 30%，RT 大幅下降 90%，业务丝般顺滑。
- 应用安全性：安全容器基于轻量级硬件虚拟化技术，实现 1 个容器 POD 1 个内核，提供 VM 级别安全性，并针对容器场景进行了 GUEST OS 的安全增强，保障应用安全运行。
- 应用高可靠：神龙服务器故障热迁移，容器多副本 POD 故障自动拉起，故障 0 感知，提升应用高可用性。

8. 应用上云方案介绍

(1) 应用系统上云实施流程

为了保障应用系统能够平稳顺利完成上云迁移，首先需要制定详细的迁移规划方案，并遵循应用系统上云迁移流程和规范实施，同时也需匹配对应的保障措施。

本节结合阿里云最佳实践，制定应用系统上云迁移流程和组织保障职责及分工。

a. 应用系统上云迁移流程

我们一般将应用系统上云迁移分为 5 个阶段，如下所示：

阶段	步骤	工作内容
应用详细调研及评估	1. 业务信息调研 2. 系统架构调研 3. 应用架构调研 4. 数据信息调研	通过业务、系统、应用和数据架构调研和梳理，为应用系统上云迁移做准备。
上云方案详细设计	1. 云上应用架构设计，应用新托管、新平台、新架构 2. 云产品选型及容量评估 3. 应用上云迁移方案 4. 数据上云迁移方案	依据调研的结果，规划应用上云方案，包括云上架构的选型标准、应用的部署、持久层数据迁移方案。
迁移实施计划及执行	1. 建立迁移组织及保障机制 2. 项目实施计划 3. 建立目标生产环境 4. 数据同步	按照制定详细的迁移计划完成实施，其中包括云上目标环境构建，应用及数据迁移同步等，实施过程严格按照计划表跟踪执行。
测试与验证	1. 功能及联调测试 2. 基准测试 3. 稳定性测试 4. 压力测试	基础功能和联调测试； 基准测试检查业务本身是否存在性能缺陷； 稳定性测试侧重系统在持续的压力情况下，长期运行时的业务处理能力及系统可能存在的缺陷。 压力测试精准模拟不同量级用户访问业务系统。
应用割接与上线	1. 割接前准备 2. 制定割接及回滚步骤 3. 割接实施 4. 割接后验证	制定详细割接及回滚步骤，明确每步执行时间点、前置条件及责任人，完成割接实施及后验证。

表 1 应用系统上云迁移 5 阶段

b. 应用系统上云组织保障

结合阿里云上云最佳实践，建议应用系统上云组织职责及分工如下：

阶段	工作项	R (负责) A (批准) C (支持) I (通知)			工作内容备注
		客户	系统 开发商	迁云团队	
项目规划启动阶段	确认业务系统迁移范围	R	C	I	明确待迁移系统数量、目标迁移平台、目标迁移成果
	业务系统全局情况摸底	A	C	R	全局情况调研（粗颗粒）：建设方、开发商、联系人、技术架构、资源清单、影响范围
	业务系统迁移风险评估	I/A	C	R	评估迁移风险：兼容性、性能、资源（人员日历、配合程度）
	确定业务系统迁移方式	A	R	C	跨云迁移以平迁方式进行
	确定业务系统迁移批次	R/A	I	C	根据迁移方式，确定迁移批次
方案设计阶段	应用云化改造方案设计	I/A	R	C	根据业务特点，设计符合业务特点的云化改造方案，包括云产品适配改造方案
	迁移方案设计	A	C	R	数据库、文件存储、大数据迁移方案
系统迁移验证阶段	应用改造	I	R	C	改造 / 重构迁移方式，涉及到的改造工作：应用、数据库、存储等
	云资源准备	R/A	I	C	云资源清单、资源申请流程、资源开通、资源下发通知到使用人等
	系统环境部署	R	C	I	应用部署初始化环境，云平台相关安全组、黑白名单策略开通
	应用部署	C	R	I	应用程序重新部署
	测试验证	I/A	R	C	包括功能、性能、安全测试
迁移割接阶段	割接计划确认	A	C	R	确认正式割接步骤、操作内容、配合人、时间计划等
	割接影响通告	R	C	I	对割接产生的影响，由业务部门发通过
	数据迁移 & 校验	I/A	C	R	验证数据删除，数据重新迁移，迁移成功后进行比对
	应用测试	I/A	R	C	应用程序功能测试
	业务割接	I/A	R	C	流量切换
运营保障阶段	保障护航	I/A	R	C	稳定性保障
	应用迁移验收	A	I	R	迁移过程文档梳理，迁移 review。组织召开验收会

表 2 应用系统上云组织职责及分工

(2) 应用上云技术方案

(2.1) 存量应用系统迁云方案

迁云工作可按如下阶段划分开展工作：



图 1 存量应用系统迁云流程示意图

a. 迁云启动

上云单位应仔细了解云平台服务能力、云平台管理办法及安全规范，明确相关的责任和义务。

b. 迁云启动方案编制

迁云单位应先对迁云的应用系统开展全面和完整的调研工作，摸清家底，识别风险，并填写《应用系统调研参考表》。根据调研结果，迁云单位组织编制应用系统的迁云实施方案，对应用系统的迁云进行规划，明确迁云应用系统的范围、迁云工作的排期计划、关键里程碑节点的交付物、云资源的规划以及相关的保障措施等。迁云单位也可根据具体情况，组织相关专家对迁云实施方案进行评审。

c. 方案评审

迁云单位根据需求向云平台主管部门提出使用申请，提供应用系统建设方案、应用系统性能和安全自测报告、云平台资源需求及其他申请材料。

云平台行政主管部门负责核实使用单位申请材料的完整性和云资源需求的合理性，并作出受理或者不予受理决定，不予受理的提出不予受理意见，迁云单位根据意见重新提交申请。

云平台供应商提供专家服务对迁云单位的申请材料协助评审，根据审批意见提供云平台资源。

d. 迁云实施

迁云单位应对应用系统的迁云实施工作进行监控跟踪管理，开展阶段性评估，定期组织例会，及时发现并解决实施过程中的各种问题。可根据实际情况，形成周报或月报等定期将向领导汇报，获取指导性、方向性意见和建议。

迁云系统改造完成后，正式部署到的云资源环境，并完成数据迁移。

应用系统割接前，应根据安全规划，对应用系统进行安全检查。

应用系统割接前应系统试运行，并详细记录云资源使用情况，编制云计算资源使用情况报告，作为调整云计算资源的依据。

制定应用系统割接计划，并根据计划进行应用系统割接并监控运行情况。

e. 系统割接

- 系统割接是迁云上云的重要环节，需要重点保证！
- 核心关键人员在场支持。
 - 割接的影响需要提前告知客户。
 - 割接建议安排在周末或业务低峰期。以最大程度减少割接对业务系统和用户的影响。
 - 割接期间，需要认真核对系统的运行情况，若不符合预期，需要及时回滚。
 - 关键系统可以申请云平台供应商提供护航保障服务。

f. 迁云收尾

- 迁云单位编制项目总结报告。
- 迁云单位出具应用系统迁云完成确认书，安全测试报告，并提交省数管中心备案。
- 将项目执行过程中的各类文档资料进行整理并归档。
- 组织进行验收评审。通过后迁云完结，进入迁云后续运维工作阶段。

(2.2) 新建应用系统上云方案



图 2 新建应用系统上云流程示意图

a. 上云启动

上云单位应仔细了解云平台服务能力、云平台管理办法及安全规范，明确相关的责任和义务。

b. 方案编制

上云单位应先对新建的应用系统开展全面和完整的调研工作，摸清家底，识别风险，并填写《应用系统调研参考表》。根据调研结果，上云单位组织编制应用系统建设方案，对应用系统的上云进行规划，明确上云应用系统的范围、上云工作的排期计划、关键里程碑节点的交付物、云资源的规划以及相关的保障措施等。上云单位也可根据具体情况，组织相关专家对迁云实施方案进行评审。

c. 方案评审

上云单位根据需求向云平台行政主管部门提出使用申请，提供应用系统建设方案、应用系统性能和安全自测报告、云平台资源需求及其他申请材料。

云平台行政主管部门负责核实使用单位申请材料的完整性和云资源需求的合理性，并作出受理或者不予受理决定，不予受理的提出不予受理意见，上云单位根据意见重新提交申请。

云平台供应商提供专家服务对上云单位的申请材料协助评审，根据审批意见提供云平台资源。

d. 上云实施

上云单位应对应用系统的上云实施工作进行监控跟踪管理，开展阶段性评估，定期组织例会，及时发现并解决实施过程中的各种问题。可根据实际情况，形成周报或月报等定期将向领导汇报，获取指导性、方向性意见和建议。

上云系统开发完成后，正式部署到的云资源环境。

应用系统正式上线前，应根据安全规划，对应用系统进行安全检查。

应用系统正式上线前应系统试运行，并详细记录云资源使用情况，编制云计算资源使用情况报告，作为调整云计算资源的依据。

e. 上云收尾

- 上云单位编制项目总结报告。
- 上云单位出具应用系统上云完成确认书，安全测试报告，并提交省数管中心备案。
- 将项目执行过程中的各类文档资料进行整理并归档。

组织进行验收评审。通过后上云完结，进入运维工作阶段。

9. 应用上云最佳实践

a. 场景一（IDC 与云平台搬站）

很多早期的政务云在新建之初都是兼具替换老虚拟化平台或者老云平台的职责，且需要在短时间内完成应用的搬迁。对于政务云迁云场景不仅要面临复杂的关联关系（应用开发单位与云平台主管单位不是同一家，需要多方协同），同时还要满足最小化的迁移成本要求。因此需要再早期的时候制定比较完善的迁移指南，并下发到各个厅局，具体的流程如下：

• 第一步：迁云启动

- 工作内容：迁云单位应仔细阅读《xx 省电子政务云计算平台管理办法》《安全规范》《上云规范》《云平台服务能力》等当地的管理规范，明确相关的责任和义务，并与本地政务云主管部门签订《租户上云协议》，制定相关工作流程和制度。

• 第二步：方案编制

- 工作内容：迁云单位应先对迁云的应用系统开展全面和完整的调研工作，摸清家底，识别风险，并填写《应用系统调研参考表》。根据调研结果，迁云单位组织编制应用系统的迁云实施方案，对应用系统的迁云进行规划，明确迁云应用系统的范围、迁云工作的排期计划、关键里程碑节点的交付物、云资源的规划以及相关的保障措施等。迁云单位也可根据具体情况，组织相关专家对迁云实施方案进行评审。

◦ 主要调研信息参考：

- 基础信息：系统名称、系统介绍、所属单位、所属科室。
- OS 环境信息：所在物理环境、芯片架构、OS 版本、服务器数量及配置。
- 应用信息：应用开发框架、应用使用中间件、应用日常性能指标、应用未来增长量、应用架构图、是否自研、是否有完成源代码、设计文档是否完整。
- 数据库信息：类型、版本、架构、数量、存量数据大小，年增长数据大小。
- 存储信息：存储种类、总数据量、日增量。
- 网络信息：网络需求（互联网、政务外网）、IP 地址规划、网络访问控制等。
- 其它：应用安全合规要求、是否有特殊外设。

• 第三步：方案评审

- 工作内容：迁云单位根据需求向政务云平台行政主管部门提出使用申请，提供应用系统建设方案、应用系统性能和安全自测报告、政务云平台资源需求及其他申请材料。政务云平台行政主管部门负责核实使用单位申请材料的完整性和云资源需求的合理性，并作出受理或者不予受理决定，不予受理的提出不予受理意见，迁云单位根据意见重新提交申请。政务云平台供应商提供专家服务对迁云单位的申请材料协助评审，根据审批意见提供政务云平台资源。

• 第四步：迁移实施

- 工作内容：迁云单位应对应用系统的迁云实施工作进行监控跟踪管理，开展阶段性评估，定期组织例会，及时发现并解决实施过程中的各种问题。可根据实际情况，形成周报或月报等定期将向领导汇报，获取指导性、方向性意见和建议。迁云系统改造完成后，正式部署到的云资源环境，并完成数据迁移。应用系统割接前，应根据安全规划，对应用系统进行安全检查。应用系统割接前应系统试运行，并详细记录云资源使用情况，编制云计算资源使用情况报告，作为调整云计算资源的依据。制定应用系统割接计划，并根据计划进行应用系统割接并监控运行情况。

• 第五步：系统割接

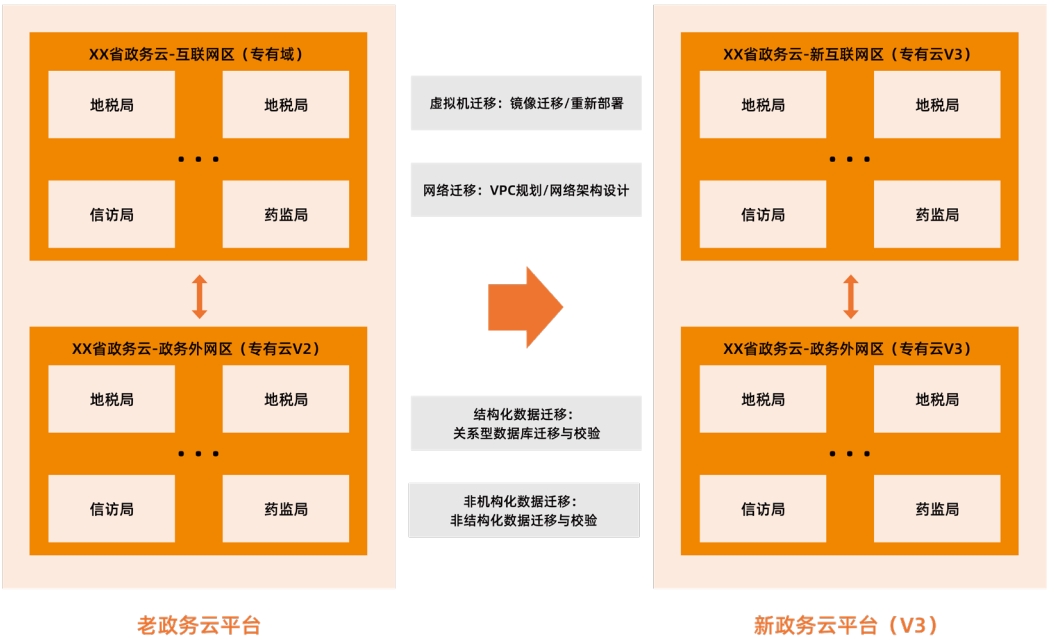
- 工作内容：
 - 核心关键人员在场支持。
 - 割接的影响需要提前告知客户。
 - 割接建议安排在周末或业务低峰期。以最大程度减少割接对业务系统和用户的影响。
 - 割接期间，需要认真核对系统的运行情况，若不符合预期，需要及时回滚。
 - 关键系统可以申请政务云平台供应商提供护航保障服务。

• 第六步：迁云收尾

- 工作内容：
 - 迁云单位编制项目总结报告。
 - 迁云单位出具应用系统迁云完成确认书，安全测试报告，并提交省大数据发展管理局备案。
 - 将项目执行过程中的各类文档资料进行整理并归档。
 - 组织进行验收评审。通过后迁云完结，进入迁云后续运维工作阶段。

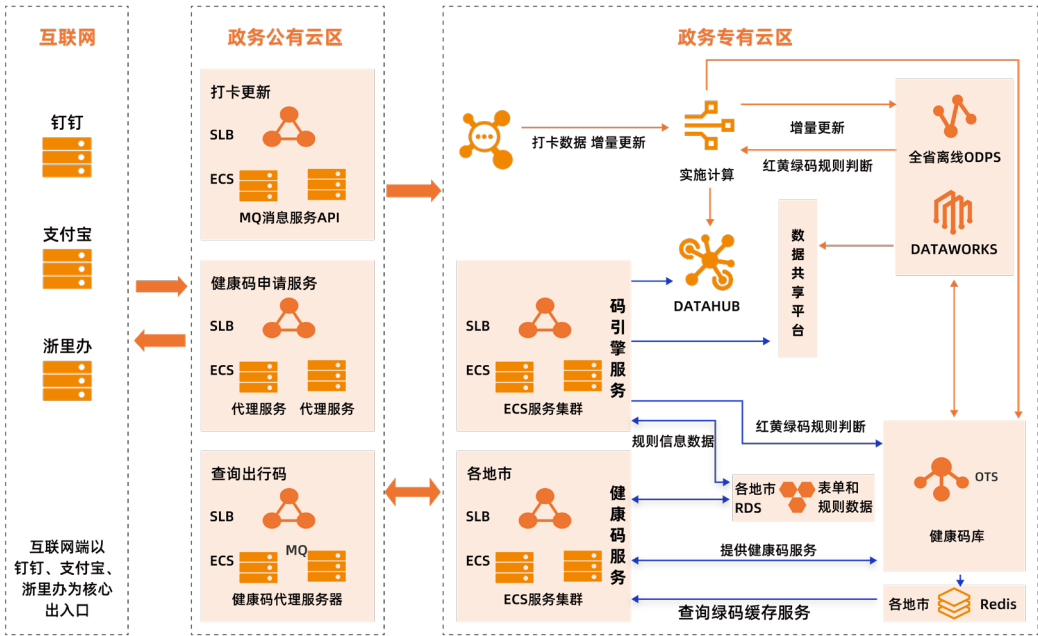
其它：相关技术标准

- 主要标准：《应用系统上云改造规范》、《数据库上云规则规范》、《文件存储改造规范》、《网络改造规范》、《安全改造规范》、《互联网应用低并发架构设计规范》、《互联网应用高并发架构设计规范》、《互联网应用大数据分析架构设计规范》、《政务网应用低并发架构设计规范》、《政务网应用高并发架构设计规范》、《政务网应用大数据分析架构设计规范》等。



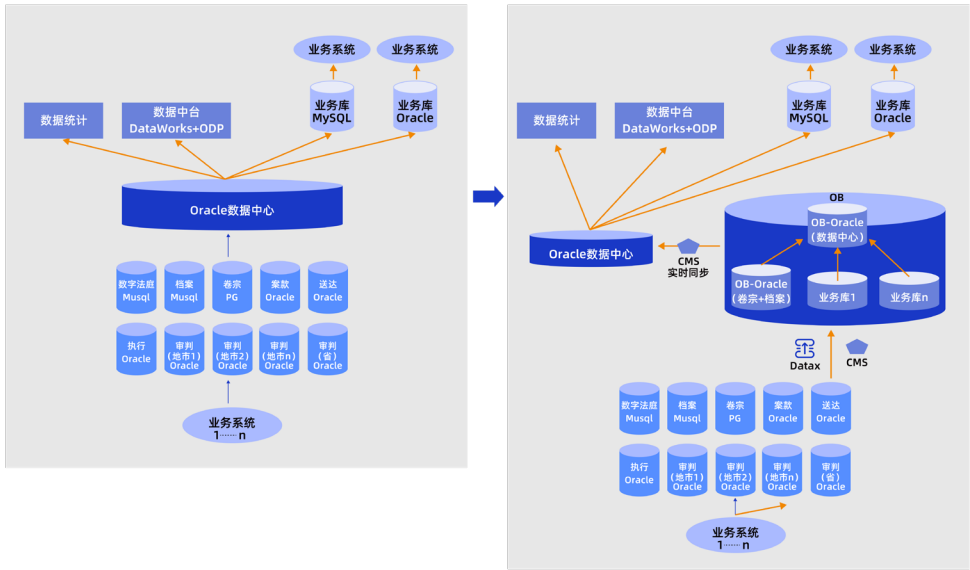
b. 场景二（云原生改造）

政务云中的有非常多的 To C 的场景且应用在高并发、高可用、高性能等都提出较高要求，而政务云相对于公共云特殊的地方在于网络、产品能力与云架构等。因此，在推进云原生改造落地时要重点关注产品能力、云架构、云的网络架构设计、专家服务等多方面的情况，具体案例如下：



c. 场景三（数据库上云）

政务云场景最为复杂的场景的就是数据库上云，主要就是针对大容量 Oracle RAC 上云，由于涉及大量的改造因此上云难度较大，具体案例如下：



d. 场景四（大数据上云）

随着大数据场景的广泛使用，越来越多的用户通过开源的 Hadoop 搭建自己的数仓，但是由于 Hadoop 本身在多租户、权限控制等都不适用于政务云场景，也衍生出了后续有部分客户考虑把自己的 Hadoop 平台替换成 Maxcompute 平台，具体案例说明如下：

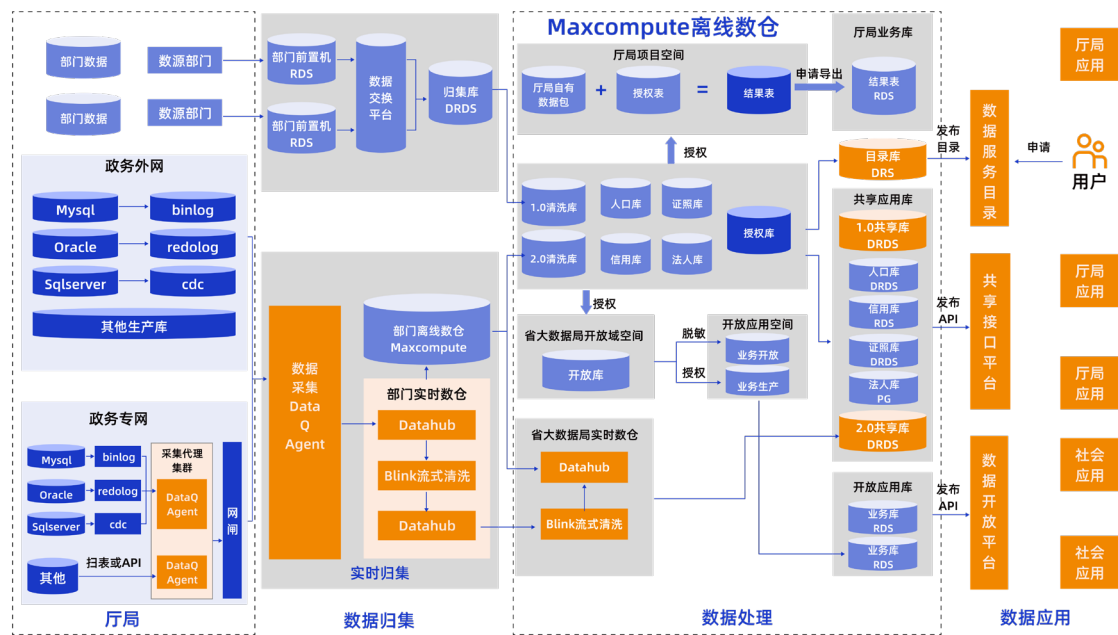


图 6 政务云大数据上云

10. 小结

本文全面解读了应用上云的各个环节，从政策解读、趋势分析到价值体现，涵盖客户画像、总体架构和典型场景介绍，并深入探讨了应用上云的技术方案及最佳实践。我们详细分析了关键业务上云、特殊行业客户上云以及云原生改造等典型场景，介绍了存量应用系统迁云和新建应用系统上云的具体实施流程和技术方案。

总之，应用上云不仅是技术发展的必然趋势，更是企业在数字经济时代保持竞争力和实现可持续发展的战略选择。本文旨在为企业提供系统的理论框架和实践指南，助力其顺利完成上云目标，构建高效、可靠的云端运营体系。

第三节 / 数据上云最全指南——从制定方案到最佳实践

1. 数据上云概念

数据上云是指将数据资产从本地存储和管理基础设施（如物理服务器、硬盘等）迁移到云计算平台上。利用云服务提供商提供的存储和计算资源，以提高数据管理的灵活性、可扩展性和成本效益。

数据资产是指对组织具有价值的的数据资源。数据资产的具体类型可以因行业和企业的具体需求而异，通常包括结构化数据（数据库 Oracle）、非结构化数据、半结构化数据、大数据（Hive、Hadoop、Spark）等。

2. 行业趋势与困境

（1）数据库上云趋势与困境

（1.1）数据库上云的趋势

数据库是存储与计算的载体，对于应用程序的作用不言而喻。作为软件产业中三大核心之一的数据库，技术壁垒高，研发周期长，其中国外软件厂商起步早、技术成熟、商业化体系健全。因此，从上世纪 80 年代至今，国外的商用数据库产品在金融、通信、政务等民生核心领域的中国市场占有率一直处于领先优势地位。

近年来，随着一云多芯、跨平台迁移、传统数据库云化等需求激增，企业相继开始了传统数据库上云的尝试。

(1.2) 数据库上云的困境

将数据库迁移至云端（上云）是许多企业为了提高效率、降低成本和增强数据管理能力而采取的一种策略。然而，这一过程也伴随着一些挑战和困境：

a. 安全性担忧

数据安全是企业最关心的问题之一。在将敏感信息迁移到云端时，企业需要确保其数据得到充分保护，防止数据泄露或被非法访问。此外，还需要考虑遵守相关的法律法规。

b. 成本考量

虽然长期来看，使用云服务可以降低硬件维护成本，但初期的迁移费用以及持续的存储和服务费用可能会比预期高。此外，如果不正确地估计所需资源，则可能造成资源浪费或者性能不足的情况。

c. 技术兼容性问题

现有应用程序与云平台之间的兼容性可能是另一个障碍。某些传统系统可能无法直接迁移到云环境中运行，这要求对应用进行改造或重构，增加了项目的复杂度和技术难度。

d. 网络延迟与连接稳定性

对于那些依赖于实时数据处理的应用来说，网络延迟成为了一个重要考量因素。如果云服务商提供的网络不够稳定或是存在较大延迟，则会影响用户体验甚至业务正常运作。

e. 技能差距

成功实施云迁移项目需要具备特定的技术知识和经验。企业内部 IT 团队可能缺乏足够的云计算相关技能，这就需要额外培训或是引入外部专家来支持项目执行。

f. 数据迁移风险

大规模的数据迁移过程中存在着丢失数据的风险。此外，在迁移期间保持业务连续性也是一个挑战，特别是对于那些不能容忍长时间停机的服务而言。

(2) 大数据上云趋势与困境

(2.1) 大数据上云的趋势

a. 全面上云

越来越多的企业正将他们的 IT 基础设施、业务平台乃至整个运营迁移到云端，这包括了传统的本地数据中心向云端迁移的过程。通过云计算，企业可以更灵活地获取计算资源，实现按需付费，减少资本支出。

b. 存算分离架构

随着数据量的爆炸性增长，企业越来越倾向于采用存算分离的架构，即数据存储与数据处理能力分离。这种架构能够帮助企业更好地管理成本，同时提高系统的灵活性和可扩展性。

c. 云原生应用

云原生技术（如容器化、微服务、持续集成 / 持续部署 CI/CD 等）正在成为主流，使得应用程序可以在云环境中更加高效地运行。这些技术有助于提升应用的可移植性和弹性，支持快速迭代和创新。

d. 可持续性

考虑到环境影响，越来越多的企业关注其云服务提供商是否采用了绿色能源和可持续的做法。这促使云服务商优化数据中心的能效，并采用可再生能源。

e. 增强的数据分析能力

随着云平台提供了更多的高级分析工具和服务，企业能够更容易地执行复杂的数据分析任务，例如预测分析、行为分析等，从而做出更好的业务决策。

大数据上云不仅仅是一个技术迁移的过程，也是企业整体数字化转型的一部分，旨在提高效率、降低成本并驱动业务创新。

(2.2) 大数据上云的困境

大数据上云的过程中，企业可能会遇到一系列的挑战和困境。大数据上云面临的困境与挑战：

- a. 数据迁移复杂性
 - 历史数据量巨大，迁移过程可能耗时长且风险高。
 - 不同的数据类型（结构化、非结构化、半结构化）增加了迁移难度。
 - 数据格式转换和表结构对应问题。
- b. 技术兼容性和架构调整
 - 传统 IT 架构下的应用系统可能需要重构以适应云环境。
 - 存在数据库架构、版本差异导致的数据迁移困难。
- c. 成本控制与资源优化
 - 初始投入和长期使用费用可能超出预期。
 - 资源利用率不足或过度配置都可能导致成本浪费。
 - 动态调整云资源以应对业务需求变化的技术挑战。
- d. 业务连续性与灾备方案
 - 在迁移过程中保持业务连续性和数据一致性是一项关键任务。
 - 确保具备有效的灾难恢复能力以防意外情况造成的数据丢失。
- e. 性能与延迟问题
 - 对于实时性要求高的应用，网络延迟可能影响用户体验。
 - 大规模数据处理时，云服务的性能瓶颈可能成为一个问题。
- f. 技能短缺
 - 缺乏足够的具有云计算和大数据相关技能的人才。
 - 培训现有员工或招聘新人才的成本。

解决这些困境通常需要企业选择合适的云服务提供商、制定详细的数据迁移计划、进行成本效益分析、培养内部能力等。同时，企业还需要密切关注行业最佳实践和技术发展趋势，以便及时调整自己的策略。

3. 数据上云最佳实践

(1) 数据上云方法论

(1.1) 数据库上云的五阶十步



图 1 数据库上云五阶十步方法论

经过多个数据库上云最佳实践和经验积累，我们沉淀出一套数据库上云的方法论，即数据库上云“五阶十步”。数据库上云的五阶十步法是一种系统化迁移方法，旨在指导企业将本地数据库平滑迁移到云端。

该方法分为五个主要步骤：调研评估与规划、分析设计与改造、测试迁移与演练验证、迁移割接和交付转维。每个步骤又细分为两个具体阶段，共同构成“十阶”。

- 首先，调研评估与规划阶段包括业务调研和可行性评估，确保对现有环境有全面理解并制定详细的迁移策略。
- 其次，在分析设计与改造阶段进行方案设计和实施改造，选择合适的设计方案和技术方案。
- 然后，进行测试迁移与演练验证，涵盖架构设计和数据准备工作，确保新旧系统之间的无缝切换。
- 接下来，进行迁移割接，通过前一阶段来保证数据迁移后系统的稳定性和高效性。
- 最后，交付转维阶段提供护航保障，支持系统的长期健康运行。

这种方法不仅考虑了技术实现，也关注业务影响、成本效益分析及长期运维等多个方面，为企业提供了一套完整的上云指南。

(1.2) 大数据上云的五阶十步



图 2 大数据上云五阶十步方法论

大数据上云的五阶十步法提供了一套系统化的迁移指南，帮助企业将本地大数据平台顺利迁移到云端。该方法分为五个关键步骤：调研与设计、测试迁移、生产迁移、系统并行和项目验收，每个步骤又细分为两个具体阶段，形成“十阶”。

- 首先，在调研与设计阶段，需调研原业务系统架构、网络架构、资源盘点，数模设计，并进行架构设计以及迁移方案设计，实施标准及验收标准。
- 其次，在测试迁移阶段进行典型场景验证和迁移测试与调优，并进行方案优化。
- 然后，生产迁移阶段涉及生产数据迁移同步、作业同步、性能优化验证，确保新旧系统的无缝衔接。
- 接下来，在系统并行阶段，通过数据比对保障新老系统数据一致性，并制定差异分析汇总，进行业务割接。
- 最后，项目验收阶段进行项目总结与验收，持续解决遗留问题，并进行项目复盘。

这种方法不仅关注技术实现，也考虑了业务需求、成本效益及长期运维等多方面因素，为企业提供了完整的上云路径。

(2) 数据上云工具

工欲善其事，必先利其器。工具能使数据上云事半功倍，下面我们就来介绍数据上云过程中常用到的一些工具。

(2.1) SQL 兼容改造工具

数据库改造关键因素在于与原数据库兼容性如何。阿里巴巴多年传统 IT 云化的方法论和经验的工具化成果——**数据库和应用迁移** (Advanced Database and Application Migration, 简称 ADAM) 工具，可自动化评估 Oracle 数据库到国产数据库在 SQL 语法、DDL 转换方面的兼容点、不兼容点以及改造优化建议。核心能力包括：

- 评估迁移改造可行性，精细到每张表、每条 SQL 的兼容性。
- 自动给出目标库类型、规格、容量和预估改造量，成本可衡量。
- 强大的结构自动转换、SQL 翻译、PLSQL 转 Java 等能力，缓解业务改造压力。



图 3 SQL 兼容改造工具

(2.2) SQL 优化分析工具

SQL 分析优化工具通过统计信息本地化来丰富统计信息的内容，然后使用索引创建建议能力，通过解析 SQL 语句，识别查询中可以创建索引的候选字段，再根据数据库统计信息进行过滤，保留真正有价值的索引创建列。作为 SQL 优化的**第一板斧**，可以有效的过滤简单 SQL 的优化工作，节约大量人力成本。

查询改写建议是建立在通过多年积累的 SQL 优化案例库上，配合 SQL 解析能力，快速精准识别 SQL 语句中可能存在的问题点，并给出相应的改造优化建议。作为 SQL 优化的**第二板斧**，在进入 SQL 优化深水区后，从 SQL 逻辑上给优化人员提供优化思路，进行优化经验共享。

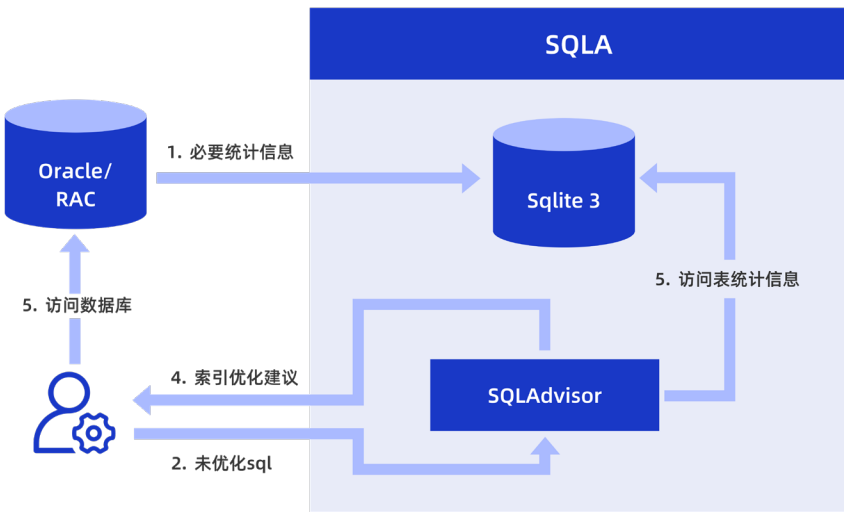


图 4 SQL 分析优化工具

(2.3) 数据迁移同步工具

DTS 是一种集数据迁移、增量数据实时订阅及数据实时同步于一体的数据传输服务，**为数据的迁移和同步场景提供支撑**。数据传输致力于在公共云、混合云场景下，解决远距离、毫秒级异步数据传输难题，如不停机迁移上云、异地灾备、异地多活、数据中台实时数据集成、数据实时订阅等问题。它底层的数据流基础设施为阿里双 11 异地多活基础架构，为数万下游应用提供实时数据流。用户可以使用数据传输轻松构建安全、可扩展、高可用的数据架构。

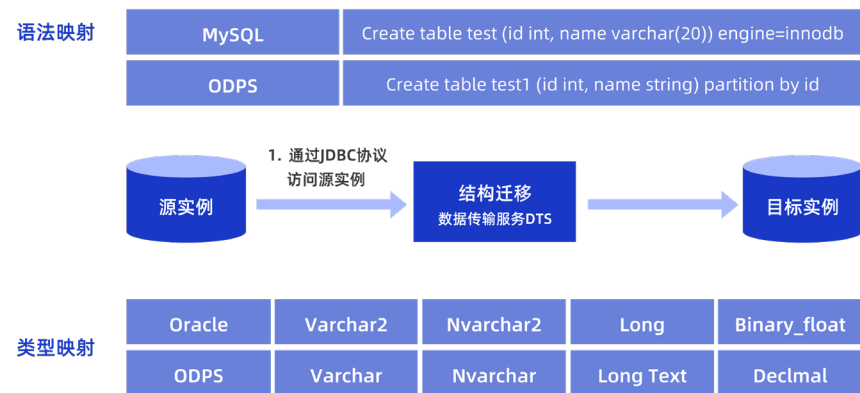


图 5 数据迁移同步工具

(2.4) 数据稽核校验工具

数据库的迁移改造，非常关键的一环，就是要确保迁移后数据库的一致性，数据的一致性要通过数据稽核校验（青天鉴）工具来解决。

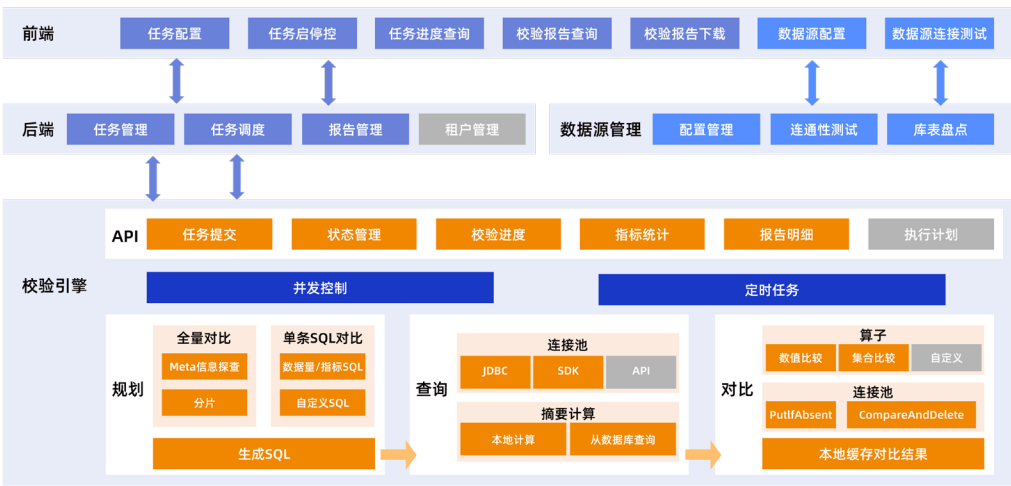


图 6 数据稽核校验工具 - 青天鉴

(2.5) 流量采集回放工具

流量回放工具 Frodo 提供端到端的数据库流量回放功能，一键采集数据库真实负载流量，在目标数据库进行 1 比 1 的流量回放，并对回放性能数据进行记录，然后针对两端数据库 SQL 运行性能进行分析对比，生成性能对比报告。在数据库迁移场景下，作为 SQL 优化的第一优化方案，可以直观地发现业务 SQL 整体运行情况，以及聚焦需要重点优化的 SQL。

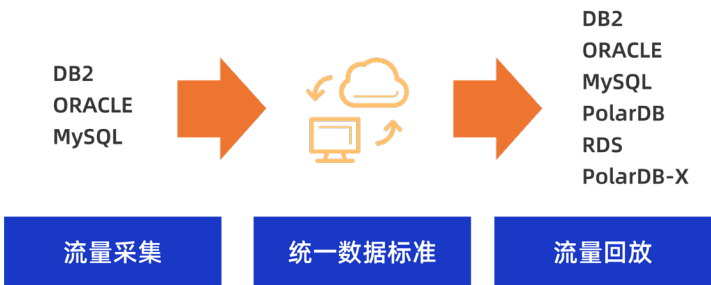


图 7 流量回放工具 -Frodo

(2.6) 大数据迁移工具（MMA）

MMA（MaxCompute Migration Assist）是一款 MaxCompute 数据迁移工具。

功能简介：

- Hive 数据迁移到 MaxCompute。
- MaxCompute 跨项目数据迁移。
- EMR+DLF+OSS 迁移 MaxCompute 项目。

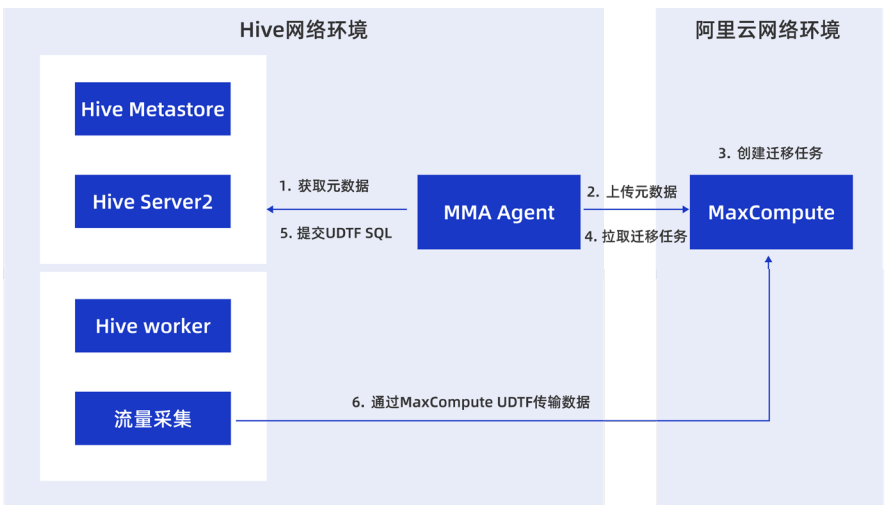


图 8 MMA 网络架构图

(2.7) 大数据迁移助手

迁移助手可以帮您快速复制 DataWorks 上不同的版本、主账号、地域和工作空间中的开发成果。

迁移助手支持迁移**周期任务、手动任务、资源、函数、数据源、表元数据、临时查询和组件**等对象。您可以根据业务需求，选择**全量导出、增量导出或自选导出**等方式导出 DataWorks 中的开发成果。

(3) 数据库上云最佳实践

(3.1) 数据库上云建设指南

阿里云很早就开始了传统 IT 云化，在数据库选型、迁移和运维管理沉淀了丰富的实施经验，总体来看，数据库上云难度较高，数据库承载了企业最重要的数据资产，更换数据库引擎犹如“换心”手术，牵一发而动全身。阿里云基于多年传统 IT 云化，传统数据库云化经验，从产品、选型、技术实施到转运维，总结沉淀了目标三要素、选型四原则、交付五必备的数据库上云标准动作，如下图所示：

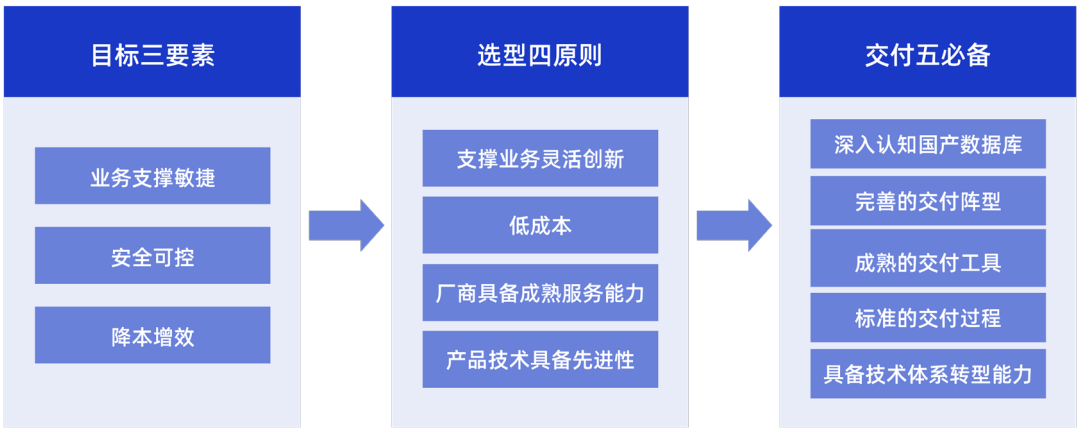


图 9 数据库上云建设指南

a. 数据库上云建设指南

目标是指引航向的灯塔，设计和规划数据库上云的目标是企业的首要事项。基于实践的一般目标，主要包括三方面：数据库技术安全可控、业务支撑敏捷、降本提效。基于一般目标参考，

企业可结合自身业务需求设计符合自身业务发展和规划的合理目标。

• 业务支撑敏捷

我国企业处在数字化转型的关键阶段，面对日益增多的业务需求，传统的商业数据库在资源弹性方面的弱势日益凸显，灵活多变的业务需求需要伸缩性更强的数据库产品来支撑。数据库作为基础设施，支撑好业务发展是核心要求，因此企业要实现数据库替换上云，业务支撑敏捷仍然是其核心目标。

• 降本提效

企业之所以期望数据库自主替换，即希望通过替换降低成本。传统商业数据库多为海外产品，且 license 和服务价格高，导致企业 IT 建设成本居高不下。而且传统数据库主要的技术研发支持力量多在海外，对数据库需求和问题的响应效率也天然低效，间接导致企业的业务创新低效。对于企业来说，通过传统数据库云化实现降本提效，也是非常务实的目标。

• 安全可控

传统海外厂商响应低效，提升响应效率的最佳途径是数据库技术安全可控，但从企业业务发展和投入成本考量，不建议企业都自主研发数据库，而是利用产业结构优势，做好产业分工，使用专业国产数据库厂商的相关产品，实现安全可控。同时海外厂商还存在一个共性问题，为了打造厂商自身的技术壁垒，往往从数据库使用的服务器硬件、操作系统均内部生产和适配，形态封闭，代码黑盒，对新兴的芯片和操作系统没有适配计划，这不利于企业享受自主软硬件的降本红利。自主知识产权的数据库对新兴的自研软硬件适配积极，可以帮助企业享受降本红利，降低企业 IT 建设成本。

b. 数据库上云产品选项四原则

正因为国内数据库高速发展中，其周边软硬件生态仍处于成熟的初级阶段，如何选择一款既满足 IT 基础设施要求，又能满足公司业务需求的数据库，是企业首要考虑的课题。在产品矩阵上，



图 10 数据库上云总体目标

阿里云已经推出了包括目前市面上几乎所有形态和类型的数据库产品，且基于阿里巴巴多年传统 IT 云化，传统数据库云化实施经验，沉淀了优秀的数据库领域实施经验，我们从中归纳出国产数据库选型的四原则：经济性、灵活性、服务成熟性、技术先进性，供企业参考。

• 经济性

成本控制是企业业务发展和运营的关键要素之一，企业在数据库选型阶段，应从数据迁移成本、运维成本、硬件成本和应用适配成本等多维度进行衡量，以确保项目建设成本、运营成本可控。

• 灵活性

云计算深刻的影响着 IT 系统的架构和开发模式演进，为达成数据库有效支撑业务灵活创新的目标，我们强烈建议使用云原生技术体系。传统数据库的系统架构，一般是紧耦合的设计方式，以此保障更高效地发挥硬件的优势。而云原生数据库相对于传统架构有明显的架构优势，其核心逻辑是将资源池化，应用可根据自身需求灵活弹性使用资源。伴随着集群规模的逐渐增长，云原生架构能够带来边际成本的下降，结合云原生天然的运维自动化优势，不仅能支撑企业业务灵活创新，还能帮助企业实现降本增效。

• 服务成熟性

当前国内很多数据库厂商依赖 toC 业务发展起来，在提供 toB 服务时往往面临巨大挑战。成熟的服务能力是保障数据库项目落地的关键点之一，企业需要对厂商的服务能力进行综合考量，例如：厂商是否具备标准化交付能力，是否获得 CMMI、ITSS 等行业服务质量认证，该能力表明企业有丰富的交付经验；再比如企业是否具备成熟的项目管理能力，是否能有效管控项目进度和落地风险；最后企业是否具备帮助企业实现技术体系转型的能力，是否能有效帮助企业掌握国产数据库的使用和运维。经验表明，具备配套成熟服务能力的国产数据库项目，其项目成本、项目风险能有效得到管控，项目目标可高效达成。



图 11 选型四原则

• 技术先进性

当前国产数据库技术路线多，为保持企业技术体系先进性，需要从多个方面进行综合评估，常见评估维度如：分布式、集中式、HTAP、兼容性、产品性能、安全可控等。当然也不能过分盲目的追求新技术、新概念，产品选型是否贴合企业技术路线和技术理念更重要。

c. 数据库上云五必备

为了保证数据库项目顺利落地，做到事前有规划、事中可观测、事后持续改进，亟需一套完整的标准方法论指导，以及完善的项目管理保障。在方法论指导下，项目组可统一协调企业内部各部门配合，管理项目的各个环节，顺利推进数据库项目的实施。总结下来，该理论包含以下 5 个必备关键点：

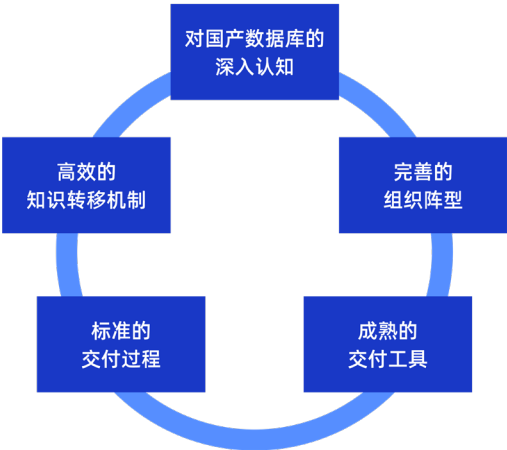


图 12 交付五必备

• 对自主研发的深入认知

数据库是一个上承应用，下联硬件的关键 IT 基础设施，不仅仅只是数据库软件自主化，而是从 CPU、存储、操作系统到数据库软件等基础设施的自主品牌化。国产数据库和国产操作系统、国产芯片的磨合时间较短，基础设施自主品牌化仍然有很长的路要走。

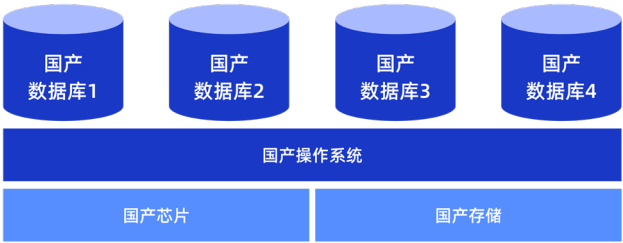


图 13 数据库上云涉及的关键软硬件产品

• 完善的组织阵型

数据库项目的实施和落地，需要各类型资源的保障，既需要资金，也需要各类职能人员，还需要各类 IT 软硬件资产，往往需要明确的项目组织阵型，以协调企业的多个部门间协作。良好的组织保障，是数据库项目成功的先决条件。基于项目实践，我们设计了一般组织架构阵型供参考：

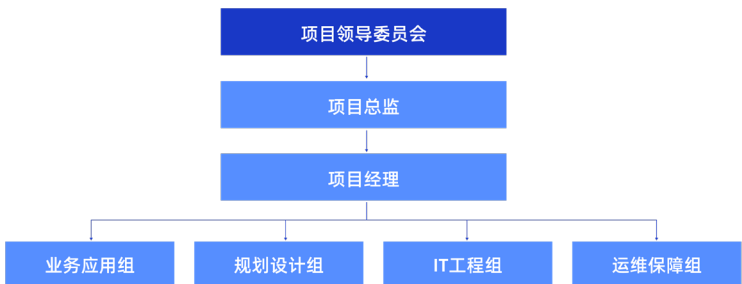


图 14 一般组织架构示意图

• 成熟的交付工具

数据库作为企业关键的 IT 基础设施，其稳定性和数据质量对企业至关重要，数据库上云过程中由于涉及到数据的迁移和变更操作，具有较高的操作风险，操作失误往往会给企业造成一定的业务影响。传统的人肉交付模式，工作效率和操作准确性均难以保证，因此在实际的项目过程中，除了标准的流程执行，每个动作的执行，都建议工具化，以保证工作效率，并规避操作失误风险。

• 标准的交付过程

企业通常有数个业务系统，且不同的业务系统开发商、设计者、运维人员一般都不同，不同人员的技术能力、管理水平都参差不齐，为保证数据库项目的交付成功，每个系统都遵照统一标准的工程实施流程进行，可以减少偶然或个体因素对整体项目的影响。

• 高效的知识转移机制

数据库技术的知识积累是持续运营的关键因素，对于企业来说，更多的是掌握数据库工作原理、运维管理、问题解决和业务迭代的能力。因此，沉淀数据库的相关技术知识，尤其是运维管理技巧和问题的解决能力是非常重要的工作。在项目推进过程中，通过厂商的技术培训、日常问题的解决沉淀，以及新业务的开发实践，完成数据库技术的知识转移，企业技术人员逐步掌握数据库的自主运维能力。同时数据库产品一般会应用较新的技术设计和理念，新技术、新架构的技术落地，有助于企业技术人员实现产品技术的更新换代，帮助企业实现技术转型，保持技术先进性。

（3.2）数据库上云实施标准和交付落地南

经过数个数据库项目的实践，我们进一步细化了在技术落地环节的重难点，主要归类为如下三个方面：

• **调研周期长：**前期需要进行大量的技术调研和测试，为后续的可行性评估及数据库最终选型提供决策依据。

• **技术挑战大：**数据库具有一定的专业技术深度要求，其中数据库交付项目涉及研发、测试、运维等多个角色，且覆盖技术门类较广，除数据库技术要求外，应用开发、网络、虚拟化等技术均有涉及。

• **交付过程复杂：**数据库项目周期长、交付环节多、参与人员广，在交付过程中，从前期调研、可行性评估、方案设计、交付实施到验收转运维，每个阶段参与的角色不同，且各阶段交付产出物各不相同，如果无法对项目交付全流程进行有效管理，则整体项目交付风险难以及时识别和管控。

为应对上述的交付挑战，我们总结了一套科学的交付实施策略和标准实施路径，用于串联项目交付过程中各阶段的交付动作，并规范数据库项目的交付实施。通过界定项目主要阶段划分，细化各阶段标准交付动作，设定交付质量标准，并给出建议的项目交付阵型，数据库项目有了科学规范的实施纲要指导，下文将详细介绍数据库项目的实施策略与标准路径。

a. 总体实施策略

多个项目的实践经验启示我们，需要将各个环节的人、工具、过程等要素有机的结合起来，基于“先易后难，逐步实施”的总体策略来完成数据库交付。我们将数据库项目分为 3 个阶段：轻咨询阶段、交付实施阶段和数据库运营阶段，如下图所示：



图 15 实施策略

• 轻咨询阶段

轻咨询阶段主要工作包括对客户侧当前的数据库架构、业务架构、业务类型和业务痛点等进行调研，以及对项目进行可行性验证、架构设计、资源规划、高可用和容灾方案规划，并制定交付实施计划、确定里程碑和关键交付物。轻咨询是整个数据库项目中非常重要的阶段，决定了后续两个阶段的工作能否顺利落地。

• 交付实施阶段

该阶段主要对轻咨询阶段制定的规划、设计和方案等进行实施，完成数据库项目的具体落地。

• 运营阶段

运营阶段的核心工作主要包括日常数据库性能诊断优化、日常运维管理（含：备份、监控、高可用及容灾管理）和新业务的上线支持，该阶段要求交付人员完善交付物和培训文档，做好知识转移，以保障客户侧运维人员掌握数据库的日常管理和使用。

b. 总体实施路径

数据库项目交付是一个复杂的工程性问题，需要对交付过程进行科学的规划和统筹。阿里云基于多个项目的实践经验，对数据库评估、改造、迁移和割接的过程进行了归纳，总结出“五阶十步”的标准动作来规范项目的实施顺序，如下图所示：



图 16 五阶十步标准交付动作

• 调研与评估

业务调研主要包括数据库系统和应用系统的信息收集，基于采集内容输出分析报告，主要覆盖以下调研项：

- ① 源数据库架构，如：数据库类型、部署架构、数据库规格、容灾方案和备份策略等。

- ② 源数据库负载，如：峰值 QPS、峰值 TPS、CPU 使用率、数据库磁盘容量等。
- ③ 数据库兼容性，如：表结构、业务 SQL、存储过程、函数、序列、触发器等。
- ④ 数据上下游依赖，如：下游数据订阅、大数据平台抽数等。
- ⑤ 业务信息，如：应用开发语言、中间件、业务规划等。

基于准确的调研结果才能保障可研的准确性，阿里云基于大量实践经验，沉淀了数据库画像采集评估工具，可全面评估上云可行性、成本和数据库选型，帮助企业降低数据库和应用迁移的风险、技术难度和实施周期。我们建议使用成熟的评估工具，对数据库信息和负载进行采集和评估，基于评估结果制定科学的落地方案。

• 分析与改造

基于调研阶段产出的分析报告，由项目组数据库架构师进行架构设计，并产出改造方案，由数据库专家根据业务特点制定迁移方案、测试方案、回流方案和回滚方案。相关方案评审通过后，项目经理开始推动后续的应用适配工作，并在适配完成后，进入到测试与演练环节。

• 测试与演练

该阶段主要在测试环境对分析与改造阶段产出的方案进行交付落地。首先对迁移方案进行实施，并对迁移实施过程中出现的各类问题进行跟踪和处理，如无主键表数据一致性保障、迁移效率保障、数据校验效率保障等问题。数据迁移完成后，需要对应用系统进行功能测试，验证功能是否符合预期。同时需要对数据库相关能力进行测试，比如高可用测试、备份恢复测试和压力测试等。

基于上述测试中遇到的各种问题，制定解决方案，结合客户需求和项目客观情况，设计具体的数据库迁移与割接行事历。然后基于行事历进行割接演练，其目的是暴露潜在的实施风险，并评估总体割接时间。割接演练一般需要进行多次，以便尽可能发现过程中的各种问题，同时对行事历进行迭代优化，最终总结出完整的数据库迁移与割接方案。

• 迁移与割接

一旦客户确定正式割接窗口后，项目组需根据演练结果，视现场数据规模，往往需提前数天开始数据迁移。迁移一般含结构迁移和数据迁移，结构迁移，顾名思义，即数据库对象迁移，如：表、索引、视图、存储过程、函数、触发器、约束等。和结构迁移不同，数据迁移，可划分为全

量迁移和增量迁移。数据规模较小的场景，因全量迁移耗时较短，所以可停服务全量迁移。而针对数据规模较大的场景，因迁移耗时较长，一般建议全量迁移和增量迁移组合实施。

数据一致是割接的前置条件，待数据迁移完成或增量数据位点一致，需要对数据进行校验，如割接窗口期相对宽松，可对数据进行多次校验，以保障数据质量。我们建议使用记录全量比对的方式进行数据校验，如受限于现场客观条件或数据质量要求不高，也可通过行数对比、关键字段抽检等进行校验，可根据实际情况选择一种或多种校验方式组合，如下图：

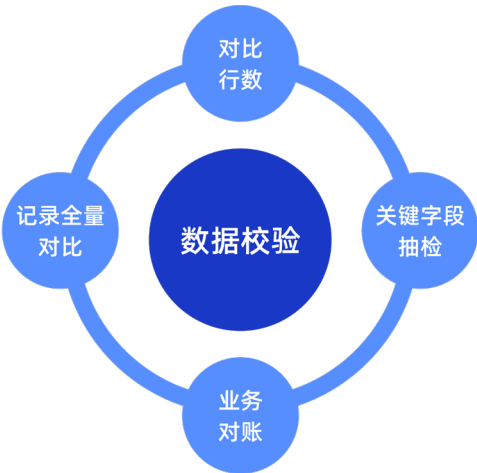


图 17 数据校验方式

在割接阶段，为确保操作可回滚，反向数据回流是重要的回退手段，为确保应用可回切，需要创建从数据库到源库的反向同步链路，实现增量数据回流，待业务平稳运行一段时间后，可释放回流链路。业务割接和回切链路示意可参照下图：

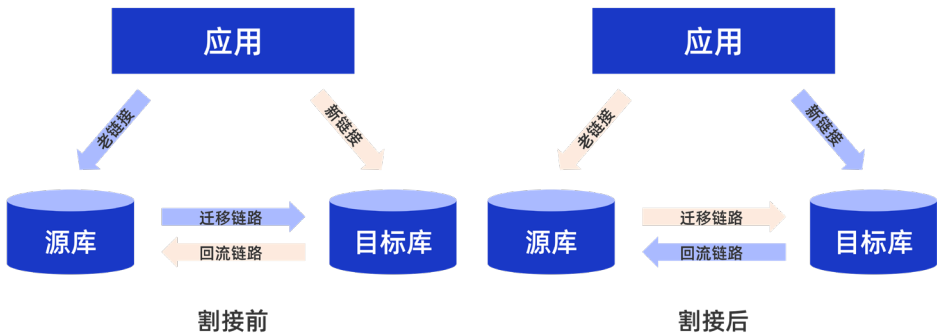


图 18 割接数据链路

c. 交付衡量标准

在数据库项目交付过程中，我们根据实施的过程和实施的结果，以及对整个过程的监管进行总结，对交付质量的划分成 4 个等级，分别是：L1- 标准 SOP，L2- 模板化，L3- 重复级，L4- 量化级。

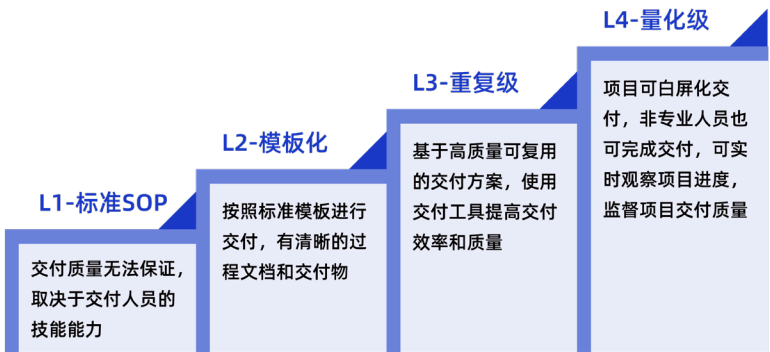


图 19 交付等级划分

以下是数据库项目交付评估等级介绍：

• L1- 标准 SOP

该等级的交付团队能够定义清楚每个阶段的标准交付动作，但对数据库项目的质量一般无法保证，项目成功与否取决于实施人员的技能水平及对数据库特性的熟悉程度。

• L2- 模板化

该等级的交付团队能够针对每个 SOP 动作定义标准的输入输出模板，能够按照标准的实施路径进行实施。项目交付过程中有角色的分工，有过程的监督，有实施动作的指导，也有后期的测试标准流程，同时基于标准输入输出模板产出标准的交付物。该级别评估标准如下：

- 前期是否有调研和产出相关改造方案、迁移方案、测试方案和割接方案等。
- 是否在实施阶段定义清楚标准动作、动作的标准输入输出和动作之间的依赖关系。
- 在交付实施完成后，是否能产出相应的架构设计、割接方案、培训课程和运维手册等交付物。

• 交付转维

交付转维是项目验收的必要环节，转维过程中，需要对数据库进行一段时间的护航，期间主要记录和处理遇到的线上问题，协助客户保障关键业务时间节点，如重大节假日促销、业务发版等。同时需要对咨询阶段、实施阶段产出的交付物进行整理，梳理遗留事项，归纳数据库相关开发运维文档和培训资料，交接给客户指定的运维方，完成知识转移。

• L3- 复用级

该等级的交付团队能够根据行业特性，使用可复用的同行业的交付经验来提高项目交付质量，并提升客户对数据库项目成功的信心。该等级主要考察方案和工具的可复用性，以及交付经验的有效复制，主要衡量维度如下：

- 是否有可复用的行业通用实施方案，如：迁移方案、架构方案、割接方案等。
- 是否有可复用的自动化实施工具，如：数据库可行性评估工具、数据迁移工具、数据校验工具和压力测试工具等。
- 是否有人员梯队建设体系，如数据库交付培训认证体系（如：数据库交付专家、工程师、助理工程师等）。

• L4- 量化级

该等级的交付团队能够对数据库项目的交付过程进行量化，项目交付过程所有动作都可以标准化，或能够根据客观情况进行灵活调整。整个过程基于白屏化工具交付，即使非专业实施人员也能够通过工具完成高质量的项目交付。

- 是否能够通过白屏化工具完成项目交付，实施人员不需要了解底层细节。
- 实施过程是否可视化，项目组可以明确了解项目当前所处阶段，并实时监督项目交付质量。
- 白屏化工具是否可以自动化产出各阶段标准交付物，并可实时查看。
- 是否有健全的生态交付体系，是否有多家可选的生态交付伙伴，是否能够基于培训认证体系对生态设置准入门槛。基于大量的实践经验总结，企业可根据自身技术要求针对关键交付阶段制定相关规范，用于规范开发、测试、运维等人员的工作内容，例如：开发规范、运维规范、测试规范等，可有效提高项目交付质量，保障项目的成功率。

d. 实施阵型

阿里云在数据库项目中的交付组织阵型，分工完善的实施团队和具备数据库专业技术的专家人员是项目成功的关键点之一。

角色	参与方	职责
项目领导组	客户 / 云服务商	负责项目整体方向把控和重大事项决策，负责确认项目总体目标，协调各单位之间的合作
项目管理组	客户 / 云服务商 / ISV	负责日常项目沟通、风险、进度管理等工作
技术经理	云服务商	负责项目需求管理、风险识别、方案审核、技术决策等
数据库专家	云服务商	负责制定迁移方案、数据迁移工具调优、数据校验方案的实施，以及数据库产品使用过程中问题处理、技术培训等工作
数据库架构师	云服务商	负责产品问题跟进解决和数据迁移工具选型，制定数据校验方案、架构设计、资源规划等工作
系统架构师	云服务商	负责对甲方的业务需求进行调研和方案设计，负责客户基础设施、软硬件规划，整体系统架构演进策略
研发工程师	客户 / 生态伙伴	负责具体应用改造、数据模型改造等
质量工程师	客户 / 生态伙伴	负责业务测试

表 1 阿里云在数据库项目中的交付组织阵型参考

（4）大数据上云最佳实践

（4.1）大数据上云需求

a. 功能性需求

主要涉及元数据、作业、存储以及权限（若有）的迁移上云功能需求，分别阐述如下。

• 元数据迁移

大数据自建大多数都依托 Hadoop 开源软件或采用发行版，在元数据迁移中主要工作量是 Hive 元数据的迁移。

Hive 元数据主要包含 Database、Table、表的字段及文件的 location 等信息，所以只有元数据正确表述 HDFS 文件所在才能被正确查询。在线下 IDC 元数据的存储通常采用 MySQL 数据库，上云通常有如下几种选择：

- 在云上自建 MySQL 数据库。
- 购买云厂商提供的数据库服务。
- 不同厂商推出的大数据元数据服务。

迁移的动作一般分为 3 个阶段：

- 先通过数据库或者云厂商提供的工具做一次全量迁移。
- 其次判断云下和云上 Hive 版本的差异，对云上元数据库做升级。
- 最后后续在双跑过程中，保持增量更新。

增量更新有几种情况：

- 客户侧对于元数据的增删改查都是通过统一的内部平台操作，那可以通过这个平台记录的日志信息，将增量同步到云上。
- 客户内部没有统一管理元数据的平台，可能会通过不同的链接方式对元数据进行增删改查，需要解析 HiveMetastore 组件的日志，记录对应 SQL，到云上执行。
- 客户云下 Hive 版本和云上 Hive 版本基本一致，元数据库没有改动，可以通过 Flink 或者其他数据实时同步工具，同步云下 MySQL 的变更到云上的 MySQL。

• 数据迁移

对于企业上云的需求，最底层的需求就是数据迁移，这也是上云的前置条件。数据迁移前我们需要评估客户的数据量、迁移所需要的带宽（如专线带宽），历史数据和增量数据的迁移方式，以此来推断数据迁移所需要的时间周期，评估项目迁移周期。

◦ 存储介质

大数据在 IDC 的客户通常数据是存储在本地盘的 HDFS 上，上云的数据存储往往有四种选择：(1) 本地盘存储 (2) 云盘存储 (3) 采用原生的对象存储 (4) 采用云厂商提供的全托管 HDFS 存储服务。综合考虑，优先建议客户采用存算分离的架构，存储采用全托管 HDFS 类型的存储，兼顾性能的同时拥有高可用、低成本和几乎无限的扩展性。

◦ 迁移需求

通常上云项目会持续几个月，数据迁移是较为前期的工作，主要分为存量数据迁移和增量数据迁移。

存量数据迁移大多数用户期望能在 3 周到 1 个月内完成，期望有工具可以进行目录的筛选，进度状态的展示。

增量数据主要是获取变更目录和文件，每天增量的同步数据到云上，保证双跑和数据校验环境的基础。

• 作业迁移

在 IDC 中已有大量的调度作业在周期性执行，一般会包括 Hive、Spark、Flink 等计算引擎所涉及的作业，不限于 SQL，这些作业都需要平滑迁移到云上来保障计算的一致性。

作业迁移主要涉及的是迁移调度平台是否替换，需要部署一份云上新环境，并且保持和云下任务一致。作业一般会运行在 3 种类型的平台上：(1) 开源框架 (2) 客户自研框架 (3) 云厂商自研框架。

◦ 开源迁移开源

继续保留开源框架，那只需要在云上再部署一份调度平台，并通过该框架自身的能力，批量同步一份任务，增量任务没有太好的方式做两边的同步，因此会约定新任务直接在云上开发，或者迁移阶段约定不许新增任务。

◦ 客户自研迁移客户自研

继续保留自研框架，那么自研框架应该考虑实现以下几个功能：

- 任务同步，例如：只能在云下平台开发任务，平台会自动同步到云上，保证任务 id 一致。
- 云上任务禁止写云下业务库。
- 提供割接 / 回滚按钮，按钮点击后，云上任务会开始写云下业务库，云下任务禁止写云下业务库；云上可以新增任务；割接后云上新增的任务会正常写云下业务库。
- 任务转发，云下跑的临时任务，例如补数据之类的动作，也要在云上运行起来。
- 告警功能策略配置，避免因为空跑或者割接导致大面积的无意义告警事件。

◦ 开源迁移云厂商自研

迁移到云厂商自研框架，需要云厂商提供基于开源的框架迁移上云的方案，批量同步到云上，约定线下不做任务的变更，或者所有变更都在云上做，云厂商提供的迁移工具，最好具有调度任务，告警配置等功能的同步，但受限于开源框架往往告警不是默认的内置功能，所以大多数情况云厂商的工具只能做到调度任务的迁移。

客户自研迁移云厂商自研

首先，这种类型的客户很少，其次这种场景云厂商很难有工具实现，一般都是云厂商提供统一的 api 或者导入包的格式，客户通过 api 或者自行实现导入包格式，批量导入任务到云厂商自研框架中。

数据正确性验证

在最为复杂的迁移需求中，存储数据的迁移最为挑战，需要保障 IDC 数据迁移到云上的一致性、正确性，不多不少迁移到云上系统中。这样才能保障云上大数据对业务是一个可用的。

数据准确性校验包括数据同步时候的数据一致性和双跑期间数据校验，数据同步工具一般都会有类似开源 Hadoop 的 CheckSum 机制，保障文件一致性。

双跑过程中产生的两份数据，是否一致需要客户侧来判断，云厂商可以提供一些计算工具，针对表和文件块做计算，需要涵盖不同的数据类型（数值，字符串，复杂类型 array，map，struct），在校验的过程中也需要考虑资源消耗，网络带宽情况，可以做粗略的校验和精细的校验，可以让用户对表做级别的区分，核心表是需要精细校验的。

集群切换

集群切换是大数据上云的最后一步也是**最关键的一个环节**，在数据和作业都迁移完成并做了一致性校验后，通常会选择凌晨进行割接。割接的时候需要设置时间窗口和回滚策略。在割接后观察集群 1 周时间，如没有异常下线自建集群，如有异常执行回滚。

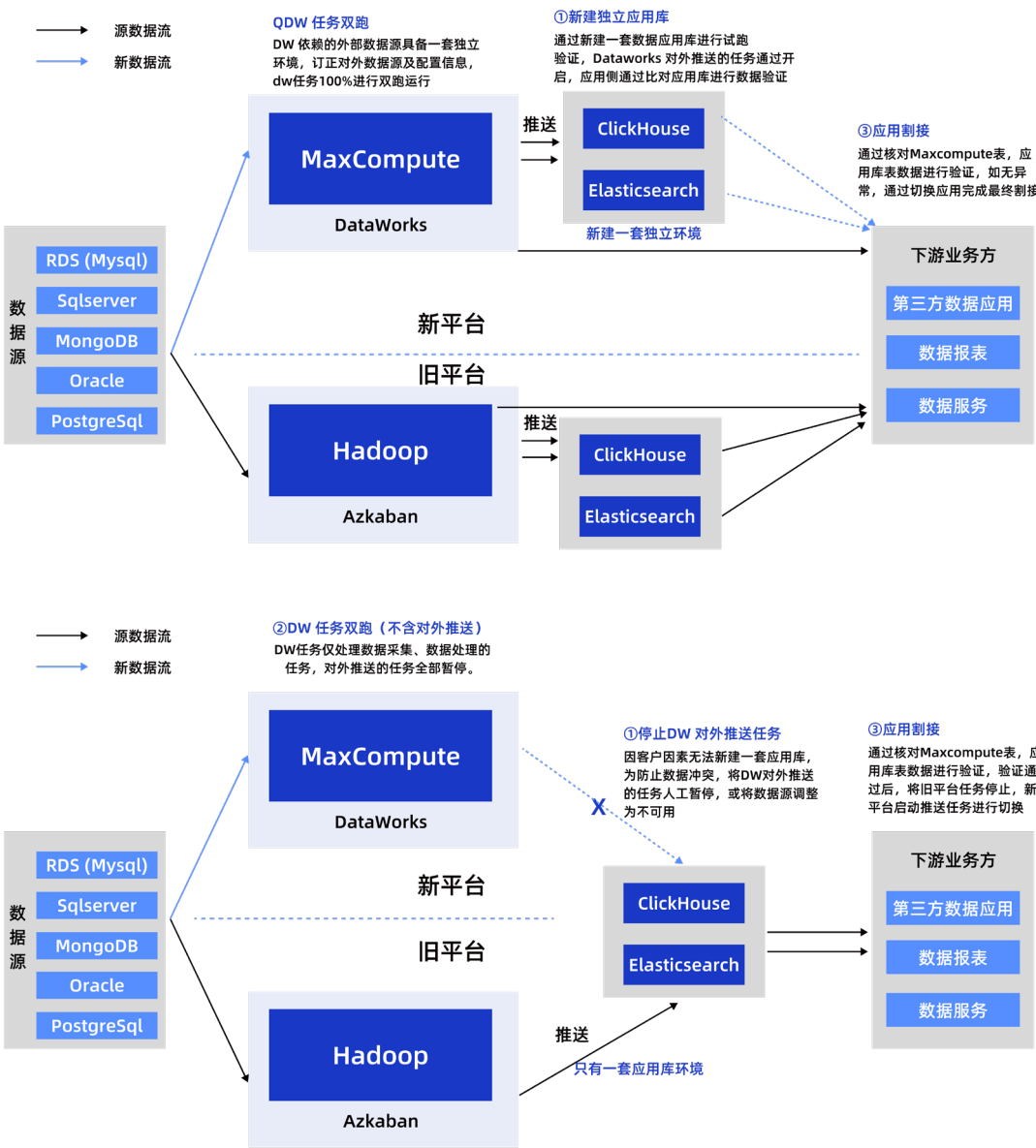


图 20 集群切换

b. 非功能性需求

数据迁移速度

大部分客户都是期望数据迁移能够越快完成越好，在各种条件限制下，一般数据迁移需要一个月左右的时间完成。数据迁移速度跟网络带宽强相关，客户是否有云厂商的专线，有多大带宽

是非常关键的信息，除此之外，如果网络速度无法达到需求，可以使用线下设备来完成数据迁移（设备进客户机房拷贝历史数据后，拉回云厂商机房，将数据上架到客户对象存储中）。因此在实际同步的时候会将同步的资源进行拆分，一部分用来做历史全量数据同步，另一部分用来同步近期数据（提供云上环境可行性测试和验证）。

• 数据迁移安全性要求

Hadoop 的数据同步，基本是使用 DistCP 完成的，本身没有做数据同步的加密，因此安全性主要依赖网络环境，专线是可以保证安全的，但如果使用公网数据同步就会有风险（使用公网同步的用户极少），另外选择通过物理的方式保证运输过程中的安全。

• 作业修改工作量

一般情况下，迁移前后的存储介质以及路径会发生变更，主要包括表文件的 location 变更。修改主要设置到 HDFS 目录的变更，HDFS 的目录主要出现在 2 个地方，其一是 Hive 元数据中会记录 HDFS 目录和表的映射关系，客户通过 SQL 完成对 HDFS 目录的读写，上云后需要修改表和分区对应的 Location，其次就是代码中写死了 HDFS 路径，需要做出相应修改。

• 数据正确性验证速度与资源消耗

如果使用 distcp 同步数据，在默认会做 checksum，校验文件是否不一致（参照如下对于不一致的描述），单任务如果同步的数据量或者文件数过多，checksum 耗时会较大，（参考：300G，5 分钟左右的速度），因此我们建议数据同步的时候，使用表或者分区的维度同步（一张表甚至一个分区对应一个同步任务）。

• 集群切换所需要的工作量

一般情况下，客户会挑选一个凌晨做割接，具体的流程是，双跑任务全都正常，并且数据校验全部通过，而且在通过后，又稳定运行了一段时间（这段时间可以认为是静默期，不做任务变更，但可以做部分服务的压测），确定好在某一天凌晨后，梳理所有要做的工作的明细项。

（4.2）方案设计总览

a. 逻辑架构

大数据迁移上云标准化一般会分为 7 个阶段，分别是计划、发现、设计、建设、测试、演练与迁移、交割与试运行，覆盖了多个维度。

• 计划阶段

计划阶段是大数据上云项目至关重要的一步，需要明确分解每个阶段需要投入的资源以及持续时间，为整个项目可行性提供参考一句。整体数据上云迁移规划可以包含四大部分：前期准备、数据和任务迁移阶段、集群双跑校验阶段和整体上云割接阶段。

第一阶段：迁移准备	第二阶段：数据和任务迁移阶段	第三阶段：集群双跑校验阶段	第四阶段：整体上云割接
资源评估 迁移规划 创建实例	元数据迁移 数据迁移 作业迁移	数据正确性校验 数据双写 任务双跑	大数据应割接 停止数据双写 IDC集群下线
迁移阶段	持续时间	说明	
前期准备	1-2 week	完成idc集群信息和资源情况收集，并制定详细迁移计划	
创建新集群	milestone	创建新版本EMR集群，并进行jindoFS相关配置并重启服务	
数据迁移	2-3 week	HDFS同步至OSS；HBASE数据迁移	
Hive元数据迁移	3 day	将原集群元数据迁移至新集群（使用DLF数据湖构建）	
作业迁移改造	milestone	JAR包等资源调整，上传至EMR集群，调整执行命令，查看运行结果	
应用迁移改造	1 week	应用需要和新集群进行交互	
数据双写	milestone	新老集群作业进行双跑	
数据校验	2 week	应用数据写入jindofs，对结果进行校验	
新老集群割接	milestone	新集群上线，老集群下线	

图 21 计划阶段

在前期准备也就是业务调研阶段，大数据迁移的关键工作就是调研梳理客户当前业务现状并产出迁移规划。其主导方是客户，云平台提供方为协助方，需要共同了解：

- 用户当前大数据平台与大数据业务全景。
- 当前大数据平台物理部署和数据流全景。
- 大数据资产梳理，包括资源、数据和权限配置等。
- 迁移网络带宽评估。

在数据和任务迁移，需要针对元数据、全增量数据以及作业进行迁移部署到云上，为下一阶段双跑打好基础，在第三阶段中，需要进行数据一致性的校验以及增量数据双写。

在整体上云割接中，主要工作包括业务割接，业务验证和巡检，同时针对风险项识别和闭环以达到正式割接的标准，最终生产切换至云上进行。

发现阶段

在正式设计方案架构之前，需要对用户现在在用的大数据架构进行摸排，主要包含下面几个方面：

- 汇总信息：原集群类型及其规模，数据量，每日新增数据量等。
- 集群内信息：节点主机配置信息，特殊性能指标要求（可选，如本地盘 IO 等），集群使用的组件和版本信息，集群安全配置方案，集群目前水位和运行情况等。

设计阶段

存储资源评估

- HDFS 的存算一体式架构

根据当前存储量和每日增量来评估一个大数据平台规划需要支持的存储单副本数据的总量，再根据存储总量去除以存储安全水位阈值来计算。一般情况，离线数仓场景，存储安全水位阈值可以设置 80%–85%；机器学习算法场景，由于大量中间结果数据，存储安全水位阈值需要设置的低一些 60%–70%。由此可得出集群需支持的总存储量的计算公式为：集群需支持的总存储量 >= (单副本数据的总量 / 存储安全水位阈值) * 数据副本数。

- 数据湖存算分离架构

存储资源在存算分离的架构下认为底层存储的资源是没有上限限制的，无需考虑集群需支持的总存储量指标，但是需要考虑集群存储资源提供给计算任务使用。

计算资源评估

关于计算资源总量和实例规格评估，一般分为以下几种选择：

考量点	使用场景	选用机型	评估方式
最小存储量要求	多用于评估 HDFS 存算一体式架构的数据存储量比较大，计算量相对来说比较小的场景，进行计算资源和实例规格评估。	选择大数据类型的本地 SATA 盘的机型（如：阿里云 ECS 的 D 系列）	集群需支持的总存储量 / 单节点挂载数据盘存储量。
任务高性能 / 磁盘免运维	多用于评估数据存储量比较小，计算量相对来说比较大。	选择云盘 ESSD 类型的的机型，一般分为通用型、计算型、内存型三种（如：阿里云 ECS 的 G 系列、C 系列和 R 系列）根据原集群历史运行情况或者任务的执行配置，具体选择通用型、计算型、内存型的哪个类型。	任务高性能 / 磁盘免运维 多用于评估数据存储量比较小，计算量相对来说比较大。
任务 shuffle 高性能	多用于任务 shuffle 性能对磁盘要求较高的场景。	选择类型的本地 SSD 盘的机型（如：阿里云 ECS 的 I 系列）	计算资源总量或者最高资源量 / 单节点计算资源量。

表 2 计算资源总量和实例规格评估参考

建设阶段

根据现状摸底以及方案设计，需要进入集群和方案的建设阶段同时明确以下信息：

- 确定各个要使用的组件版本，确定 EMR 大版本。
- 选择服务高可用 (3 台 master 节点)。
- 是否使用 OSS–HDFS 方案。
- 元数据采用自建 RDS 或者 DLF。
- 是否使用 Kerberos 身份认证。
- 购买方式（包年包月 / 按量付费）。
- 网络情况 (vpc/vswitch/ 安全组)。
- ECS 机器规格 (master 是否开启公网)。
- 登录集群 master 的方式（密码 / 密钥）。
- 是否有引导脚本操作（预装自定义环境，修改网络配置等）。

最终在云上按照引导方式购买、配置集群使得云上环境部署完成。

- 迁移阶段

- 元数据迁移

元数据同步应该在数据同步之前就做，给 distcp 提供需要同步的表和目录列表，等到下次同步元数据的时候，可以获得新增表的列表，协助 distcp 同步新增目录，元数据同步和升级的工作，一般只需要半天即可完成，前期可以先做升级和修改 location 的验证。

- 数据迁移

HDFS 迁移 OSS-HDFS 通用方案：

- 在新集群上配置跨集群访问线下 HDFS，使得云上的集群可以访问线下的 HDFS 数据。
 - 梳理需要迁移的 Hive 表的迁移脚本，建议直接命令细到表级别的迁移命令，使用 --update 的形式进行增量。
 - 梳理需要迁移的 HDFS 非 hive 表的路径信息，形成脚本，使用 --update 的形式进行增量。（推荐按照业务切分到尽量细的目录）。

其他对象存储迁移 OSS-HDFS：对于其他对象存储归档数据，先拉到 IDC，再同步到 OSS-HDFS。

- 作业迁移

客户大多数会选择保留自己原有的开发调度平台，如果是自研的话，需要客户在自己的平台上改造代码：

- 部署两套环境，云下任务批量同步到云上。
 - 线下环境代码修改，任务发布等动作要同步到云上环境，云上环境禁止修改，避免两边任务不一致。
 - 云上环境如果要写线下环境，需要空跑，避免数据重复。
 - 提供割接按钮，点击后，云下任务最后写线下环境的动作会空跑，由云上任务会写线下环境。

如果是开源的话，由于可控性较低，因此可能要在一段时间内，避免修改任务：

- 云下任务批量同步到云上（不同开源框架实现方式不同，一般是本地文件 + 数据库的方式存储）。
 - 数据同步任务（主要是往外推送数据的场景）需要额外判断是否正常执行。目前作业迁移没有产品化工具可以支持，通常需要用户手工迁移，所以客户最关心的需求是迁移成本，如作业改造成本、兼容性问题、数据准确性（如函数差异）等问题引起的调整成本。

平迁到 HDFS

相关作业迁移至新集群，访问数据 URL 中的 schema 无需改变，但 nameservice 还需改变。作业中使用了主机名：端口的方式需要检查并调整。需要检查线上集群和线下集群参数的差异，列出差异项，方便迁移过程中问题定位。

迁移到 OSS/OSS-HDFS

迁移至 OSS/OSS-HDFS 需要把使用到的文件路径 hdfs:// 的需要调整为 oss:// nameservice/ 的方式，spark 和 hadoop 访问数据时也需要调整 hdfs:// 为 oss:// 的方式。其他需要注意的点通平迁至 HDFS。

- 数据正确性验证

在迁移过程中数据准确性校验的需求主要涉及文件块校验和表数据校验，文件校验会在迁移数据的过程中完成，主要是 CheckSum 保障文件数量的一致性。表数据校验通常是校验表的行数、数字列的 sum、avg、max 值，可以使用双跑校验方案来进行校验。

文件完整性校验方案：

- Jindo DistCp Counters

Jindo DistCp Counters: 可以在 MapReduce 任务结束的 Counter 信息中，获取 Distcp Counters 的信息，对比 Bytes Destination Copied 和 Bytes Source Read 数据。

◦ Jindo DistCp--diff

Jindo DistCp --diff: 会对文件名和文件大小进行比较，记录遗漏或者未成功传输的文件，存储在提交命令的当前目录下，生成 manifest 文件，再检查 manifest 文件信息。表数据校验请看下节。

◦ 双跑校验方案

HDFS 文件数据和表 / 分区数据校验，如下数据校验流程图：

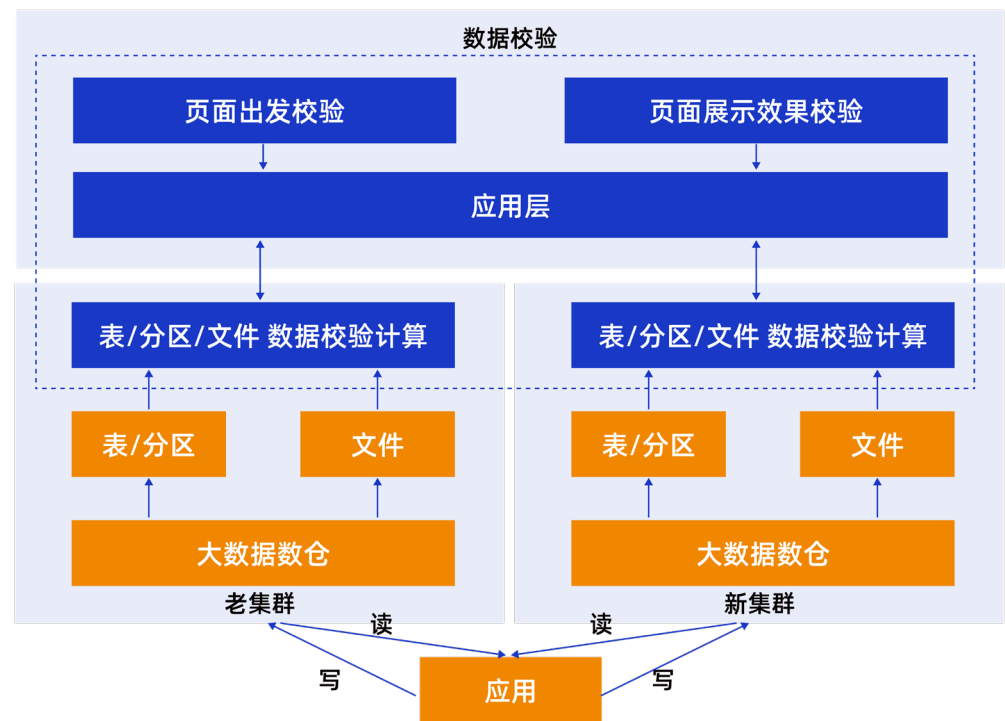


图 22 双跑检验

前提说明：

- 无论是表数据还是 HDFS 文件块，都通过 sql 来做数据校验 (spark-sql)，HDFS 文件块自动 / 人工识别存储 + 压缩格式，创建单字段的外部表通过 sql 进行校验。
- 全量数据只做初略的校验方案，会把 sql 分别下发到 IDC 和云上，程序读取计算后的结果进行两张表情况的比较。

- 大数据场景中，数据上游是流式进入 HDFS，或者数据同步任务启停时间差异，有可能导致最源头的表任务是非幂等的（相同的任务再跑一次，结果可能不一致），因此这里就会涉及到容忍度的概念，例如数据条数差异在 0.5% 以内，认为该任务双跑成功。
- 精细化比对一般可用于 ads 层表（业务方提供列表），会将整行数据所有字段的 hash 化后校验。

◦ 校验方案：

整体上包含两轮数据对比。

- 第一轮对比：针对表：计算 count 值（通过 count 值计算的条数差异率）+ 脚本自动识别数值类型的字段计算 sum，avg，max，min，如果不一致，需要人工介入修复 / 覆盖数据。
- 第二轮对比：第一轮对比没有对比到字符串或者直读 HDFS 的数据，因此第二轮对比需要把这部分数据也加入进行验证。
- 针对文件块 / 已经通过第一轮对比的表：将整条数据拼接成一个字段，将整条数据加码，排序，计算全表的数学口径（中位数，众数，四分位数等）。

如果两轮校验都通过，那么我们认为两份数据一致。

• 交割和试运行阶段

◦ 交割阶段巡检工作：

◦ 数据迁移验证：

- 验证从旧环境到新环境的数据迁移是否完整无误。
- 抽样对比源端和目标端的数据，确保一致性。
- 检查数据格式、类型转换是否正确处理。

◦ 应用功能测试：

- 对已部署的应用程序进行全面的测试，确保所有特性按预期运作。
- 特别关注与大数据平台交互的部分，如 ETL 流程、查询接口等。

◦ 应急响应计划：

准备详细的应急预案，包括回滚策略，以防新系统出现问题时能迅速恢复服务。

- 试运行阶段的巡检
 - 性能监控：
 - 设置性能基线，并持续监控 CPU、内存、磁盘 I/O、网络流量等资源使用情况。
 - 使用性能管理工具（如 Prometheus, Grafana）跟踪集群健康状态，及时发现并解决问题。
 - 日志分析：
 - 分析系统日志文件，查找异常信息或错误报告。
 - 监控应用程序的日志输出，捕捉潜在的问题线索。
 - 数据质量评估：
 - 定期抽取样本数据进行质量检查，确保数据的准确性和完整性。
 - 监测数据输入输出过程中的延迟和其他问题。
 - 用户反馈收集：
 - 主动向终端用户征求反馈，了解他们对新系统的使用体验。
 - 根据用户意见调整优化系统配置和用户体验。
 - 容错能力检验：
 - 测试系统的容错能力和高可用性机制，例如节点故障切换、自动恢复等功能。
 - 模拟不同类型的故障场景，评估系统的应对效果。
 - 自动化巡检脚本：
 - 编写自动化巡检脚本来定期执行上述部分或全部检查项。
 - 自动化脚本可以帮助提高效率，减少人为疏忽带来的风险。

b. 组件架构

弹性 MapReduce (EMR) 是基于原生技术和泛 Hadoop 生态开源技术的安全、低成本、高可靠的开源大数据平台，组件完全源于开源社区，可以将现有 IDC 自建的大数据集群无缝平滑迁移至云上 EMR。

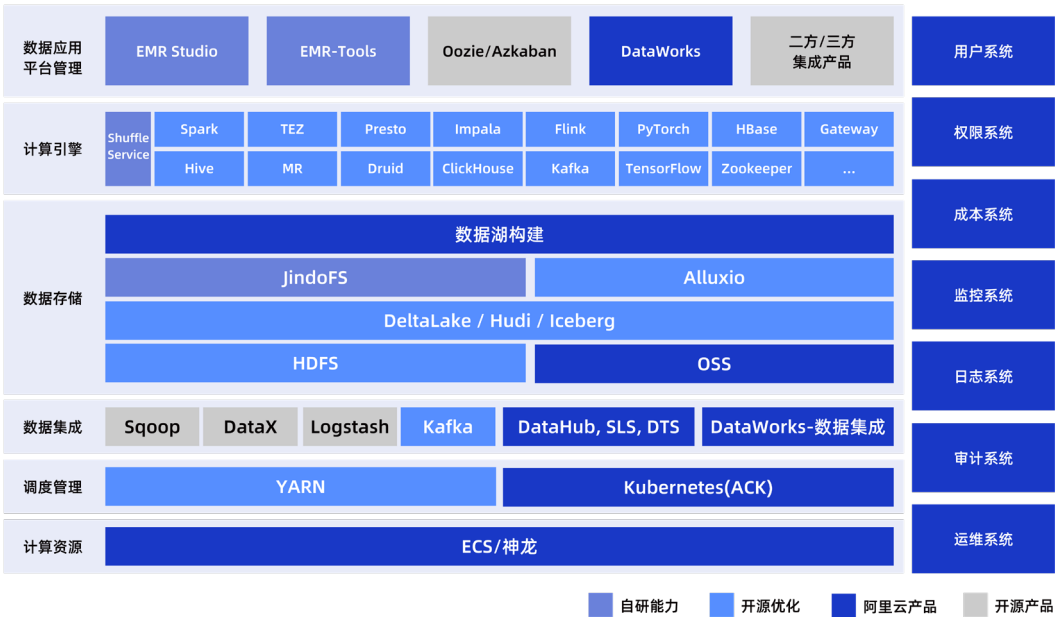


图 23 EMR 组件架构图

EMR 产品集成了社区中常见的热门组件，包括但不限于 Hadoop、Hive、Hbase、Spark、Presto、Impala、Flink、Sqoop、Hue、Iceberg、Starrocks 等，可以满足大数据在线业务、离线 / 在线数据仓库、云原生数据湖构建、实时流式计算等全方位场景的需求。

EMR 开源优化：

- Spark 中增加了 Spark Streaming SQL，性能较开源版本有大幅提升。
- Delta Lake 相较开源版本增加了 ZOrder 和 Data Skipping 能力。

EMR 自研能力：

- 数据应用平台，提供交互式开发、作业提交、作业调试和工作流一站式数据开发体验。
- ShuffleService 是 EMR 在优化计算引擎的 Shuffle 操作上，推出的扩展组件。
- SmartData，为 EMR 各个计算引擎提供统一的存储优化、缓存优化、计算缓存加速优化和多个存储功能扩展。

c. 架构决策

- 选择阿里云开源大数据平台

- 保障大数据平台可预期的 SLA: 稳定性、数据及时性、突发数据需求等

- 稳定性：数据服务可用性 99.99%
- 数据及时性：保障数据产出时间符合业务需求，数据产出不延迟
- 突发数据需求：存算分析架构，水平扩容计算资源满足突发取数需求

- 提升大数据平台运维效率

- 秒级扩容：存算分离架构，存储资源无限扩展，计算资源秒级扩容，满足业务突发需求
- 弹性伸缩：基于集群资源（CPU、内存等）现状弹性伸缩，降低成本
- 故障诊断：自动故障诊断，快速恢复业务

- 打造面向未来的云原生大数据平台

- 云原生存算分离、湖仓一体架构
- 完善数据治理，提升数据质量，保障数据安全，提升数据使用效率
- 打造大数据 + AI 一体化架构，创新更多数据业务场景

- EMR 与开源 Hadoop 对比优势

相比使用开源 Hadoop 发行版本自建大数据平台，云上 EMR 有以下主要优势：

- 集群轻松构建、运维管理功能丰富易用，便于使用及维护

- 基于开源 Hadoop 发行商软件构建大数据平台周期长，技术复杂度高，配套开发、运维、监控等工具不完善，且无法得到有效技术支持，需投入较大人力成本进行支撑及维护。
- 云上 EMR 可一键快速构建集群，发行版经过充分兼容性测试验证；控制台提供丰富运维、监控工具开箱即用，大幅降低使用及运维成本；除此之外云上提供的专业技术支持，也可以帮助快速定位并解决问题。

- 计算资源按需弹性使用，数据可分层存储，资源利用率高

- 自建大数据平台在服务器资源上一般需提前预估，并且针对业务峰值做一定的预留，导致资源利用率低，面临计算需求的潮汐变化应对也非常不灵活。

- 云上 EMR 支持灵活的弹性伸缩，资源按需使用，分钟级即可完成扩缩，并且可以根据业务变化的负载或者时间周期进行计算资源自动伸缩。

- 开源组件持续加固优化，稳定性及性能更优

- 自建大数据平台一般采用开源社区版本组件，兼容性问题、组件缺陷需自行处理。性能也需自行优化，如需使用社区新版本组件或前沿技术栈搭建及测试成本高。
- 云上 EMR 结合内部大规模实践经验，核心组件如 Hadoop、Hbase 等兼容开源并提供有效的稳定性加固。除此之外，新兴技术栈提供丰富优势特性，优化可提升场景化性能 10 倍以上。

- 全栈安全防护策略，云上环境及数据更安全

- 自建大数据平台从底层到服务层安全能力需自行构建，复杂度高，覆盖不完整，配套审计能力不完善，存在较多隐患及风险。
- 云上 EMR 提供从硬件、网络、操作系统、大数据服务全栈易用的安全防护能力。提供 CVM 主机安全防护，异常告警；支持云硬盘加密，对象存储 OSS 加密；网络安全组设置；集群支持基于 Kerberos+LDAP 安全架构，身份认证确保集群访问安全，并提供基于 Ranger 的多种策略的数据权限管理。

- 无缝对接云生态服务，快速构建完整配套能力

- 自建大数据平台如数据开发工具等配套能力需自行构建，成本高、周期长。
- 云上 EMR 可轻松与 Wedata 数据开发平台、BI 商业智能分析产品无缝对接，帮助客户低门槛、快速构建数据集成、数据开发、数据可视化等服务。除此之外，EMR 还支持云监控、云审计等周边服务，助力客户快速构建云上企业智能生态。

- 大数据上云优势和动机

整体上行业客户上云动机包括 8 个主要方向：

- 解决软硬件生命周期等现实问题

一方面当有数据中心、服务器到生命周期，或者对多个 IDC 进行整合，往往采用先进的云服务模式进行建设，而不是继续采用传统 IDC；另一方面云服务模式能消除政企行业对 IT 硬件、中间件、技术平台等复杂的生命周期管理。

提升业务敏捷

云服务模式从三个层次提升业务敏捷：基础设施资源按需获取；应用、资源按需弹性伸缩，满足业务规模快速扩展需求；中间件、云原生、DevOps 等技术平台能力按需获取，降低新业务开发门槛、快速上市抢占先机。

降低 IT 成本

云服务模式可根据业务需求调整资源规模减少浪费性支出，降低基础设施和相关技术平台运维人员能力要求及运维成本，尤其采用公有云模式，无需管理运维本地数据中心、按需获取云资源及云服务能力，大幅降低 IT 成本，也大幅降低新业务创新的试错成本。

提升运维效率

云服务模式由云服务厂家数百、数千人专业运维团队提供专业运维服务，可以基于持续积累、全球经验显著提升运维质量和效率。

提升可靠和安全合规

云服务模式能基于丰富的行业最佳实践提供高可靠、高安全及行业合规所需的技术方案能力，协助完成策略制定、组织流程建设及标准认证。

支持全球化部署

云服务模式已部署的全球资源、全球网络和全球平台能帮助企业快速开展跨国新业务，并与集团形成业务、数据、管理的联动协同。

构建数据底座与数据资产

云服务模式从两个方面强化政企行业数据底座和数据资产建设：最先进的大数据、AI 智能、数据治理、数据安全等技术；丰富的行业最佳实践，包括数据平台、性能优化、数据治理数据运营组织流程、数据智能场景应用等。

持续引入新技术加速业务创新和商业创新

云服务模式持续提供行业最新技术能力和实践经验，加速行业业务创新及商业创新，如云原生提升业务敏捷，AI 决策智能化、生产无人化 / 少人化，IoT 万物互联智能感知，区块链可信智能合约等。

大数据上云收益

整体可以从 6 个视角来看，如下图：

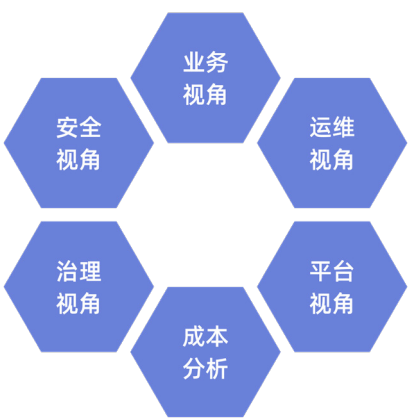


图 24 大数据上云收益 6 大视角

企业在价值创造与变现的发展过程中，面临着 IT 基础架构数字化转型中需求与供给不匹配的挑战，通过数据上云实现从传统 IT 治理向更关注业务持续治理的云治理进化，这一转变不仅在管理、运营和执行层面上带来显著效益，而且利用云的规模化安全优势提升企业安全响应能力，并驱动运维自动化以提高效率，最终实现比 IDC TCO 更具成本优化的整体 IT 环境。

(4.3) 迁移方案实施

a. 迁移方案验证

可以选择部分 DAG (Directed Acyclic Graph，有向无环图，指大数据任务编排调度拓扑) 中产生的数据分层数据来进行验证，验证元数据、存量数据、增量数据和作业的迁移方案是否能够在本次迁移中 RUN 起来。

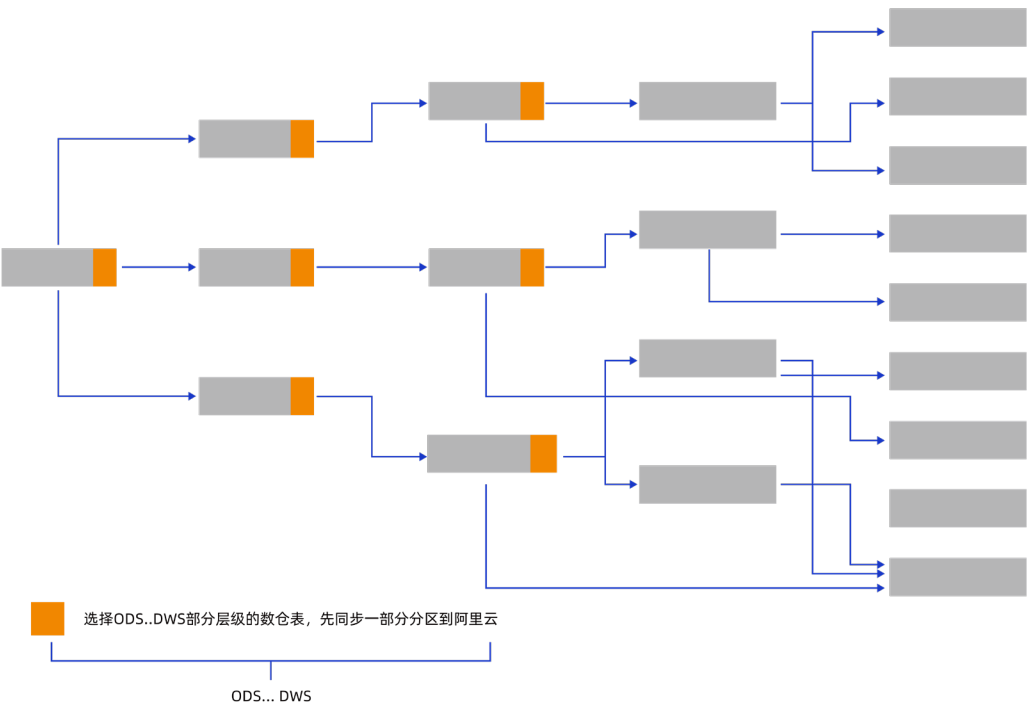


图 25 DAG 分层数据验证

b. 迁移校验实施

DLF 元数据对于使用云上生态有天然优势，比如对接 Maxcompute、Hologres 等自研产品，在成本上也极具优势，在一般情况下推荐选择。

元数据迁移会涉及全量迁移和增量迁移的需求。通常是先进行全量迁移，修改集群配置，验证数据准确性，再停止元数据更新进行增量同步，保障元数据的一致性。

数据迁移还会考虑数据是全量上云还是部分上云，要尽可能降低迁移上云对业务的影响通常会考虑全量迁移再做增量迁移，最后还要考虑数据双写等需求。在迁移工具上可以选择 JindoDistCp 来进行数据迁移，JindoDistCp 是用于大规模集群内部和集群之间拷贝文件的工具，重点优化 HDFS 到 OSS-HDFS 服务的数据拷贝，并保证数据拷贝落地的一致性。

c. 双跑校验实施

针对于迁移的源端和目标阿里端的数据，会分为下面几种情况：

- 一种是数据迁移的数据和待迁移数据（橙色部分和灰色部分），这部分数据在数据迁移过程中进行了文件级别的 checksum 校验，这部分数据在双跑的校验步骤不需要进行重新校验。
- 一种通过迁移完成满足客户双跑 ETL 任务依赖的数据之后，通过源端和目标端分别执行 ETL 任务产生的数据（绿色部分），这部分数据没办法通过文件级别的 checksum 校验，需要进行数据内容的校验。

目前针对数据内容的校验主要支持以下几种数据类型的校验方式：数据校验会做不同类型字段的校验，分别是：

- 表 / 分区 (count(1))
- 数值字段（计算 max/min/avg）
- 字符类型（计算 hash 值的 max/min/avg）
- map 和 array(计算元素个数)

会把 sql 分别下发到源端和目标端的阿里云上，程序读取计算后的结果进行两张表情况的比较，产生对应的数据校验报告。

d. 大数据集群切换

以某头部政企客户为例，整体的平台割接周期为 4 天，选择业务低峰期，降低业务影响，割接时间按照数据流顺序和应用依赖关系安排，初步安排如图：

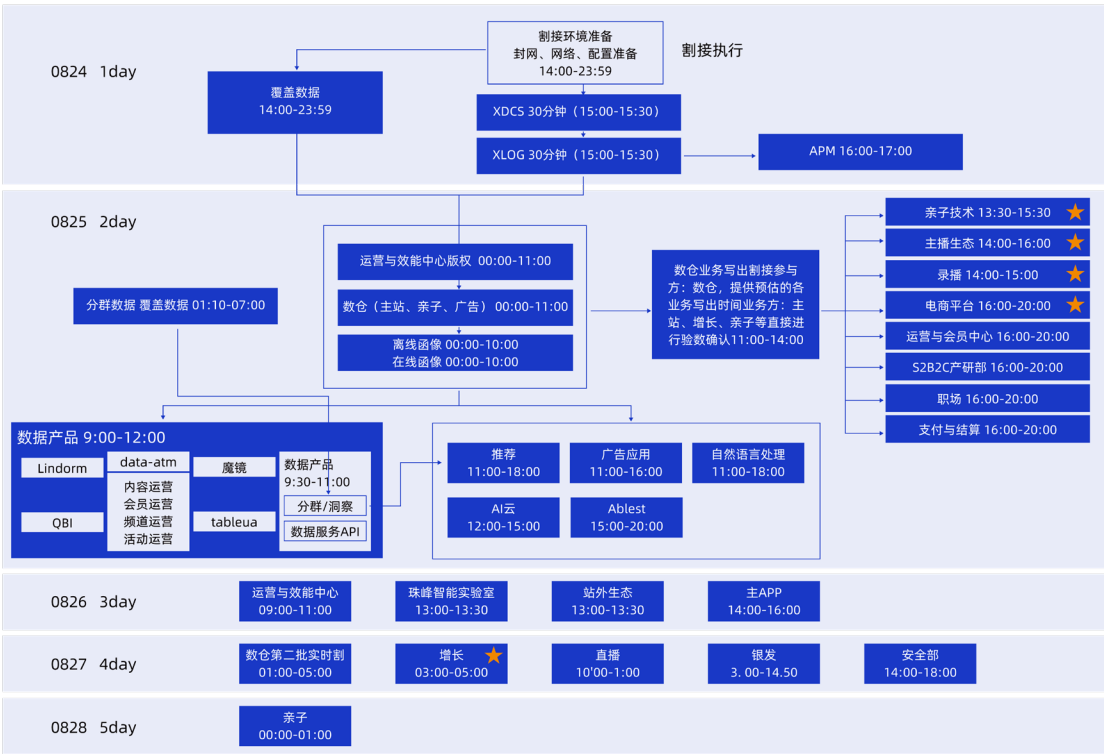


图 26 大数据集群切换整体架构示意图

第一天：开始封网、割接环境准备、完成业务所需数据覆盖到最新 IDC 数据，封网涉及大数据平台相关产品，包含 Panda、XStream、XQL、AI 云平台、XDCS、XLOG、Comet、数据应用，封网期间不进行变更操作，包含新建、修改、下线等动作，存量任务可以正常运行，如果存在业务问题需要处理请联系平台运维协助处理。

第二天：从 0 点开始进行数仓整体数据中间层任务切换到云上，数仓、BI、画像的云上数据转为生产数据，按业务逐步切换下游写出应用中间件业务数据，包含 MySQL、Kafka、ES、xcache 等产品。数据耦合紧密业务跟随数据中间层在上午完成云上数据确认后，下午进行应用切换，主要包含数据应用、主播、亲子、电商、搜索、推荐、广告、ABTest，及相对独立业务。

第三天：第三天割接主站 APP，因较多的主要业务方在第二天割接，第三天需要进行业务数据观察，保障主要业务稳定，进行相对独立运营与效能中心业务割接。

第四天：第四天凌晨业务低峰期开始进行增长业务割接，及安全部任务割接，完成总体云上任务的处理。

4. 小结

本文从数据上云视角展开介绍了数据上云的相关概念及行业趋势与面临的挑战，详细解读了数据上云的方法论和相关的工具，并从数据库上云和大数据上云两个角度展开介绍了最佳实践，以帮助企业高效、平稳的将数据迁移到云上。

第四节 / “一云多芯”成为确定性趋势 —— 如何解决跨平台上云的技术挑战

“一云多芯”是由阿里云率先提出的概念，近年来已经成为云计算领域的一个热词，不仅代表着云平台向着开放化标准化发展，也反映出政企守护“IT 安全生命线”的刚性要求。

1. “一云多芯”需求

- **基本需求：**最基本的需求是确保云平台能够兼容国产芯片及云产品服务。为此，云厂商需要开放对异构芯片的支持，并提供便捷的代码跨平台迁移和改造工具，以保障系统的无缝对接和高效运行。
- **进阶需求：**国产平台在实现“能用”的基础上，正逐步追求“好用”的目标，要求云厂商进一步优化性能、提升系统稳定性，以确保能够在国产平台上高效运行生产系统及其他关键应用。
- **加分需求：**为了优化迁移成本（包括人力成本、时间成本和资金成本），我们建议通过扩容现有系统的方式，而非完全新建系统，逐步实现国产化替换。这样不仅可以延长现有硬件的生命周期，还能降低改造难度并最大限度减少对业务的影响。

2. 跨平台的“鸿沟”

- 在这样的背景下，飞天企业版的“一云多芯”能力首先需要跨越跨平台的“鸿沟”。
- **应用国产适配：**跨平台上云场景中 ARM、Alpha 指令集与 X86 指令集存在差异，导致应用代码可能需要进行重新适配修改、编译和发布。

- **国产 OS 替换：**操作系统需要替换成符合跨平台上云要求的操作系统版本，需要提供操作系统镜像，以及应用操作系统替换评估工具。
- **应用性能调优：**ARM 架构与 x86 架构的性能差异，需要针对 ARM 架构进行专门性能优化调优，特别是 JVM 参数针对 ARM 架构进行优化调整。
- **跨平台上云开发及编译工具：**跨平台上云平台需要提供兼容的 JDK、GCC 等开发和编译工具链。
- **数据库替换：**将 Oracle 数据库或开源 MySQL/PostgreSQL 替换成符合国产要求的数据库版本，并提供相关数据库兼容性适配检查和 SQL 改造工具。
- **中间件替换：**一般需要替换成跨平台上云版本的中间件产品，需要提供相关解决方案。
- **服务器替换：**提供服务器平迁，典型场景：Intel x86 到 Hygon x86 或者 ARM 指令集的服务器。

3. 跨平台上云方案

(1) 现有 x86 平台原地跨平台上云替换方案

基于现有的 x86 Intel 云平台，我们将通过逐步替换的方式，最终实现跨平台的全面上云。该过程涵盖服务器的扩容与存量轮转替换、操作系统的更新、应用的重新部署，以及数据库的云内迁移。

- **核心价值及优势：**
 - **资产保值：**支持通过混部或轮转分阶段替换的方式，逐步完成国产目标。
 - **混部最灵活、场景最丰富：**支持跨平台场景下 AZ 级、产品级、集群级混合部署，同时满足两地三中心灾备要求。
 - **云平台原地国产改造：**支持服务器替换，如：Intel 服务器到 Hygon 服务器，Intel 服务器到鲲鹏服务器等。
- **跨平台上云替换整体策略：**
 - 短期跨平台上云混部扩容、中期采用轮转替换、最终全面跨平台上云。

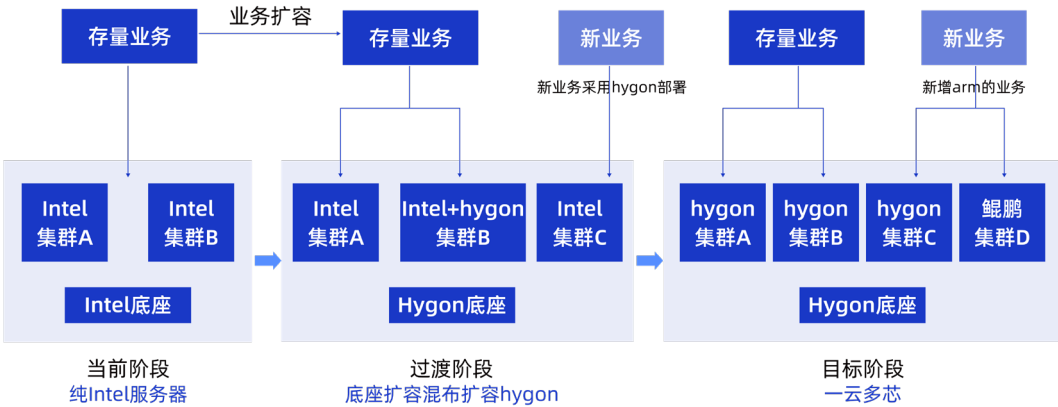


图 1 跨平台上云整体策略

(2) 新建一朵云跨平台上云迁移方案

通过新建一朵云来完成全跨平台的上云，借助阿里云专有云迁移中心的强大能力，我们能够快速将应用和数据库迁移到新平台。此外，利用专有云迁移中心的跨平台上云改造评估功能，可以精确量化迁移成本，确保迁移过程高效、经济且可控。

- **核心价值及优势：**
 - 迁移的灵活度高，同时可以在迁移期间选择性的迁移。
 - 阿里云提供一键迁移产品，不仅支持云下物理机环境、阿里云低版本云、VMware 等所有环境，同时提供 VMware 无代理、不停机迁移。
- **跨平台上云迁移策略说明：**
 - 遵循先易后难、先边缘后核心的策略进行计划编排。
- **阿里云跨平台上云迁移产品功能介绍：**
 - VMware ESXi/vCenter 虚机的无代理迁移、不停机迁移。
 - 阿里云公共云、其它版本专有云迁移。
 - 云下 IDC 迁移。

4. “一云多芯”产品能力到服务支撑

当前，飞天企业版已经在“一云多芯”领域沉淀了众多优势能力，并形成了全面的体系化交付方法论和服务支持体系，确保这些能力能够快速、平稳、高效地落地。在产品能力建设上，已经满足硬件覆盖全、统一算力调度和管理、全栈性能优化技术、一站式迁云平台、全栈适配支持等产品化能力。

• 硬件覆盖全

- 支持多种国产芯片混部：涵盖主流国产芯片，并兼容网络设备、内存、磁盘、网卡、交换机等部件。
- 丰富混部形态：支持 6 大类混部形态，包括单机房内的多芯混部、多集群多芯混部、产品间多芯混部，以及多机房场景下的同城容灾、多 Region 和异地容灾多芯混部。

• 统一算力调度和管理

- 资源分配与负载调度：实现不同芯片间的高效资源分配和负载调度，资源利用率提升约 30%。
- 自动化运维平台：提供自动化工具和平台，简化运维复杂度，提高管理效率。

• 全栈性能优化技术

- 软硬件深度调优：通过核心云产品的软硬件深度适配，性能提升约 30%。
- 持续优化机制：建立持续性能优化机制，确保系统在运行过程中始终保持最优性能。

• 一站式迁云平台

- 跨平台代码扫描工具：大幅降低迁移复杂度，迁移效率提升 1 倍以上。
- 全流程白屏化操作：支持开源数据库、OS 向国产数据库、OS 迁移的全流程迁移白屏化操作。
- 容量模型自助评估：提供全场景性能基线支持和多芯混部性能测试，帮助用户进行迁移前的容量模型自助评估。

• 全栈适配支持能力

- 从底层到应用层的兼容性扩展：兼容能力不断从底层向操作系统、国产应用层扩展。
- 关键技术和软件栈：阿里云 PolarDB 数据库拥有完全自主知识产权的关键技术与软件栈，具备超高的兼容性和性能；阿里云服务器操作系统 V3 通过工信部电子标准院最高等级开源项目评估，成为用户首选的操作系统之一。

在**服务支撑体系**上，专有云服务团队围绕一云多芯上云规划、跨平台迁云、云上架构优化、多活容灾体系、故障实训演练体系进行了系统建设及落地实践。

• 一云多芯上云规划

- 在“一云多芯”背景下，根据企业上云需求，我们为客户提供定制化的业务上云规划、技术架构规划、技术管理规划。对业务上云策略、路线与计划进行总体设计，并提供支撑技术架构的规范与流程设计与改进。可以覆盖企业 IT 部门对于“一云多芯”业务、技术、管理三方面的整体上云 / 用云规划。

• 跨平台迁云

- 为了确保企业顺利过渡到云端，并最大化利用云平台的优势，我们提供全面的客户应用现状调研服务以及详细的应用与数据迁移方案、应用割接方案的定制。我们的目标是通过系统化的评估和实施，帮助企业无缝完成“一云多芯”下跨平台迁云，并在迁移过程中提供全程指导和支持。

• 云上架构优化

- 企业上云并非一蹴而就，更不应只是为了“上云”而上。为了最大限度地利用云计算带来的技术红利，企业需要进行全面的云原生架构改造和优化。体系化的云上架构优化覆盖需求调研、现状分析、基础架构设计、微服务架构设计、容器化架构设计、应用发布和监控设计、应用上云改造设计等，需要企业、云技术服务商、应用厂商等共同协作才能达到理想的效果。云上架构优化涉及大量的技术攻关，比如：精细化容量评估、全链路压测、一体化监控等。

• 多活容灾服务能力

- 容灾不仅仅依赖于产品和技术的实施，更是一个涵盖“管理 + 技术 + 应用”的综合管理体系。针对一云多芯多机房场景下的同城容灾、一云多 Region 及异地容灾多芯混部等复杂环境，只有通过产品容灾能力、应用容灾架构与标准演练流程的有机结合，才能真正实现高效的容灾建设效果。其中的容灾专家服务需要覆盖应用多活容灾架构咨询、应用多活容灾架构设计，多活容灾演练等内容，以帮助企业更好的落地多活容灾架构，最大限度减少故障对业务的影响，持续提升企业业务连续性。

- 实训演练服务能力

- 实训演练立足于飞天企业版云平台，参考混沌工程核心理念，旨在为企业提供从云平台故障演练到租户侧应用级故障演练的全面服务体系。通过体系化的故障演练，帮助企业从云平台到云上应用全面提升其架构韧性、应急响应和故障处理能力，确保业务连续性和稳定性。配套的服务支撑可以帮助客户更好的建设和落地实训演练。比如：开展定期实战演练，检验应急预案的有效性，提升团队应对突发事件的能力，确保在真实场景中能够迅速、有效地解决问题。通过实训演练培训课程，帮助企业内部团队掌握故障诊断、应急处理及故障预防技能，增强自运维能力。

5. 结束语

一云多芯的本质是利用开放兼容的行业生态，促进产业创新加速，提升供应链安全。对于一云多芯全生态链上的科技从业者，是时代赋予的机会，也是时代赋予的使命。阿里云专有云团队通过多年的持续探索，已经构建了完善的一云多芯产品能力及服务支撑体系，我们会和政企客户一起守护好“IT 安全生命线”。

第三章 深化用云篇



第一节 /

不断深化用云之路——从传统管理到云原生运营

1. 引言

随着云计算的迅猛发展，云技术已逐步替代传统 IT 架构中的服务器、存储和虚拟化等单体软硬件，成为企业 IT 能力中不可或缺的核心组成部分。云计算的应用不仅限于互联网行业，现已广泛渗透到政务、金融、工业、医疗、交通等关键领域，标志着企业正逐步进入深化用云的新阶段。这一转变对上云后的业务运营提出了更高的要求：

- **业务连续性**：确保系统和服务在任何情况下都能稳定运行，减少停机时间和数据丢失风险。
- **降本增效**：通过优化资源配置和提高运营效率，降低总体拥有成本（TCO），实现更高的投资回报率。
- **自动化运维**：利用自动化工具和技术简化日常运维工作，提升系统的可靠性和响应速度，减轻人工负担。

为了满足这些要求，企业需要从传统的 IT 管理模式向现代化的云原生运营模式转型，充分利用云计算的弹性、灵活性和智能化优势，推动业务的持续创新和发展。

2. 当前项目特点

当前，政企类项目主要以建设类项目为主，存在以下特点：

- **一次性建设与被动扩容**：项目初期进行大规模建设和配置，后续扩容依赖客户需求的被动响应。

- **一次性大成本投入**：前期需要大量的资金投入，主要用于软硬件采购和部署。
- **成本控制优先**：在项目初期，成本控制是主要关注点，可能导致对长期效益和灵活性的忽视。
- **侧重日常运维**：项目交付后，主要关注平台的日常维护，而非持续优化和创新。

这种服务模型主要依赖于软硬件销售，后续上云的效果极大程度上取决于客户现有的云化团队能力。如果客户缺乏足够的云技术能力和经验，可能会导致资源浪费、低效利用、信息孤岛等问题。

基于第二章介绍的“6R”模型以及大量客户上云案例，我们观察到当前大部分企业主要采用的是平移上云（Rehost）的方式。因此，现有的技术架构大多停留在简单的迁移阶段，并未严格按照云原生架构进行设计和优化。在后续文章中，我们将从**应用域**和**数据域**两大类别分别深入探讨这一现象及其影响，分析企业在云化过程中所面临的挑战和改进方向，其中：

- **【应用域】**如何沉淀一套针对各个行业的云化架构评估模型。（参见文章：IT 架构的全方位“体检”——云化应用架构模型“4L6D”全解析）
- **【应用域】**云化架构模型落地的最佳实践。（参见文章：云上“攻坚”——行业头部客户核心业务系统用云最佳实践）
- **【数据域】**如何对云上数据进行运营。（参见文章：云上数据运营）
- **【数据域】**云上数据运营落地最佳实践。（参见文章：重点行业数据云化治理最佳实践）

3. 云上业务运营体系

为了确保业务上云后的成功运营，必须实现服务模型的根本转变，从传统的建设项目转向以客户为中心的持续运营项目。这种转型强调与客户的共同成长，通过不断提升客户体验和业务价值，提供贴身服务，确保项目的收入与客户的业务发展紧密关联。

结合阿里云在大量政企客户运营的经验，我们沉淀出一套云上业务技术运营体系：

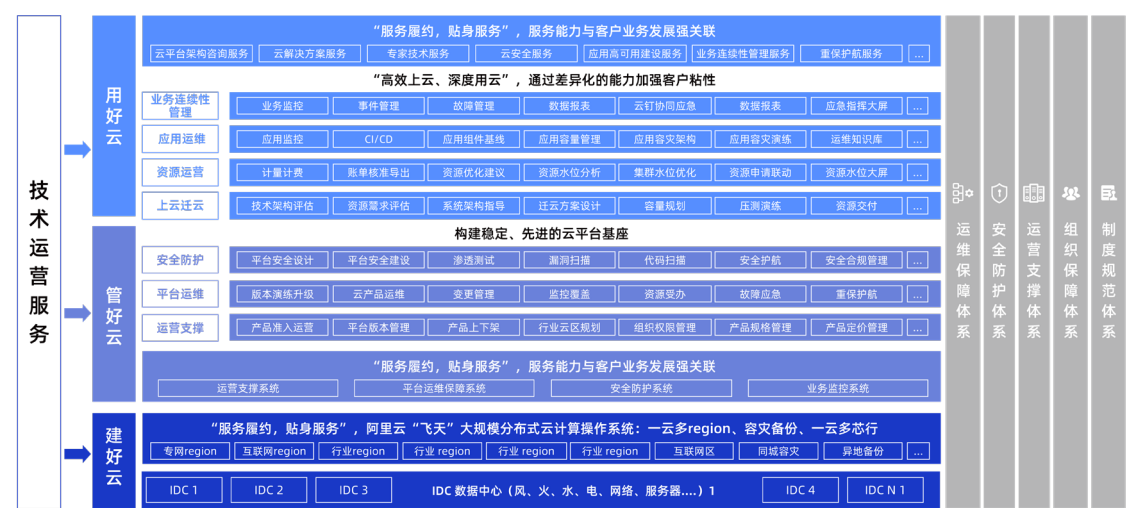


图 1 跨平台上云整体策略

接下来，我们将以政务云业务运营的实践为例，系统地阐述云上业务运营体系的具体实施和关键要素。

4. 政务云业务运营实践

在过去的 10 年里，政务云是最早提出从建设模式转为运营模式，这里可以从政务云的发展历程看出。这里也对政务云的发展历程做一个简单介绍。

- 政务云建设 1.0 阶段：目标是信息技术按需供给，数据资源充分利用。难点在于重建设轻应用，数据中心无序发展。
- 政务云建设 2.0 阶段：目标是推进政务云应用实现“一网通办”，国家、省、地市数据共享。难点在于数据孤岛、重复建设、无法发挥云的优势。
- 政务云建设 3.0 阶段：目标是建立政务云资源统一调度机制，加强一体化政务云平台资源管理和调度。难点是体系化运营好一朵云。

(1) 主要挑战

(1.1) 技术层面

- **云服务能力：**每朵政务云在交付后所有的产品种类与数量都是固定的，且单次扩容的周期平均都在 6 个月以上，因此是否能满足客户上云的需求极大的依赖本期配置的产品合理度。

- **云架构演进：**对于新增或者变化的业务需求，可能需要推进专有云底座升级，尤其是在涉及大版本升级时，会伴随着较长的升级周期和较高的实施代价。

- **云网络互通：**政务云网络又分为政务外网与各个委办局的私有网络以及互联网区，云上也分为租户网络与管控网络，因此各个网络互通对于网络前期规划要求比较高。

- **云迁移工具：**当前专有云并没有体系化的迁移工具，历史的项目都是依靠人肉或者开源工具去实施，迁移效率和质量都和实施人员的经验密切相关，尤其是在面临大批量的平台搬迁调研与数据迁移工作都需要大量的人力投入。

- **传统数据库云化：**政务行业是 Oracle 深耕领域，大量的开发商应用都是基于 IOE 架构开发，虽然目前阿里云已经有成熟的传统数据库云化方案，但是对于开发商的改造成本较高，同时对于后期的云平台运维也提出了较高的要求。

- **应用割接：**政务系统有非常多面向民生的业务系统，例如：健康码、核酸检测、各地政务 app（浙里办等），在上云过程中不仅对于数据一致性要求高，同时对于割接的时间窗口也要求非常短，且割接涉及到云上与云下的各个模块协同复杂度较高。

- **技术支撑：**在应用上云过程中经常面临大量的产品性能调优、产品使用最佳实践以及多产品联动的问题排查，对于技术要求和经验都要求较高。

(1.2) 运营层面

- **服务目录：**随着各地政府的信息化发展，越来越多的地市选择采用购买服务的方式引入云计算，由于阿里专有云有大量的 PaaS 产品因此在计量计费上也是最为复杂的，一方面要结合阿里云的公共云价格给出参考，同时还要基于专有云产品能力给出计费规格逻辑，对于专有云产品和运营团队都是比较高的要求。

- **迁云推广：**随着政务云交付转维，推动各个厅局能持续的上云将是一个艰巨且长期的过程，不仅要求该运营团队有敏锐的业务意识，同时还需要具备能力较强的实战能力解决客户上云技术难题。

- **运营分析：**一朵政务云正常情况下随着上云的增加可能面临部分产品的水位紧张需要扩容或者引入新产品，因此定期的经营分析对于云的可持续发展非常重要，既能避免水位过高导致的稳定性下降，又能降低资源不够导致的业务流失。

(1.3) 组织层面

- **内部协同：**对于一朵专有云，阿里云当前主要有 4 大类角色，包括：业务线（BTE、SA、CSM）、云平台运维（CRE + 二线专家服务）、云平台安全、专有云产研；其中应用上云过程中面临的几大类问题都是需要多方协调，包括：前期评估不充分导致后期需要额外的产品扩容与新增、临时新增业务需求导致的产品扩容与新增、租户侧安全体系等。
- **外部协同：**常见的阿里政务云一般都是以运营商或者本地的国企为运营主体，而云的主管单位多是当地的大数据局，因此在应用上云过程中不仅要配合各个委办局完成资源的申请流程，同时还要确保资源申请合规，流程复杂且合规要求高。
- **责任边界：**当前的应用上云实施服务的主要责任边界在于需求明确的情况下面推进应用上云，但是期间出现客户上云意愿不足或者其他非技术卡点是需要有人来推进，因此政务云要想运营的好，能有更多的业务上云，是需要有一只专职的运营团队来推动相关方完成上云。

(2) 应对策略

(2.1) 原则说明

正常情况下一朵云都会经历“建设、管理、使用”三个阶段的发展，但是政务云其实和传统的专有云区别非常大，云的商业结果都是按照云资源的消耗来收费，因此从云的角度出发大概有七类角色“云承建方、云运维方、云运营方、云使用方、云管理方、云监管方、云审计方”。最终一个政务云要经营的更好极度依赖对于相关工作的拆解与几方协同效果，接下来的最佳实践会结合人、事协同来说明政务云的运营机制。

(2.2) 工作梳理（事）

既然已经对于应用上云的困难有了清晰的认知，接下来需要对于应用上云过程中的所有工作进行梳理，且由于应用上云过程中遇到的很多卡点如果在前期有提前规划后续会降低上云成本。接下来，按照一个云的“建管用”三个维度对相关工作进行拆解，其中天蓝色部分代表当前政务云交付过程中需要加强的点：

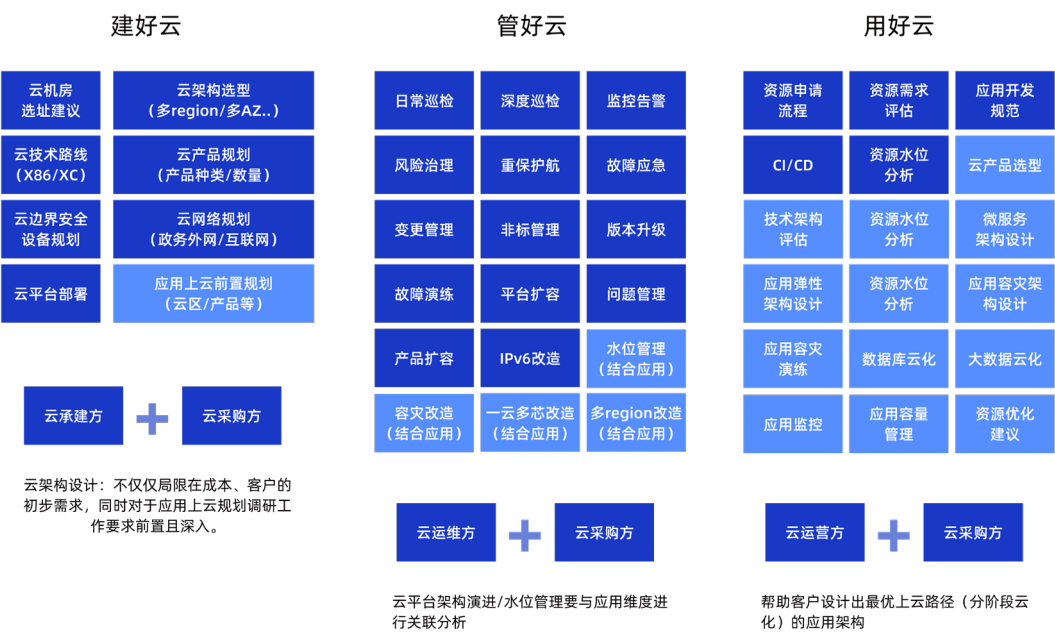


图 2 “建管用”三个维度工作拆解图

(2.3) 干系人梳理（人）

政务云不仅涉及到各个用云单位，同时还包括主管单位、监管部门、审计部门，相对普通的企业客户更加复杂，对于相关部门梳理如下，重点对于云承建方、云运维方及云运营方的主要支撑环节进行说明：

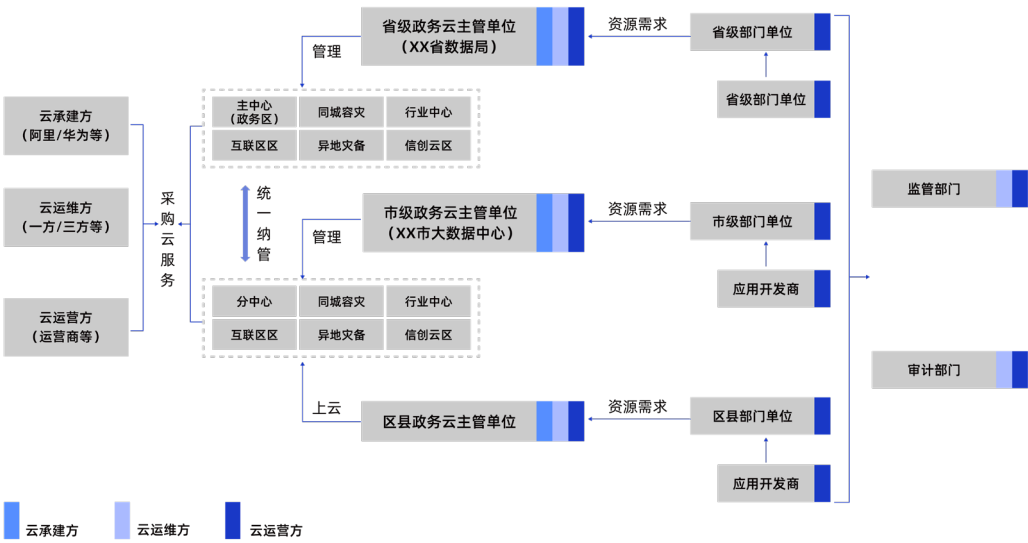


图 3 干系人梳理示意图

（2.4）云平台演进（武器库）

阿里云早期的政务云形态从专有域、专有云 v2，最终定型专有云 v3。不难看出专有云随着时代的发展，在多租户、虚拟专有网络、运营等领域都有了突飞猛进的发展。但由于政务是传统厂商深耕多年的领域，也面临非常多的挑战。接下来会从云演进角度看未来政务行业的发展：

- **【一云多芯能力完善】**：具备“一云多芯”、一站式跨平台迁移、传统数据库上云等能力。
- **【数智化云管】**：50 余款云产品的运营数据模型标准化治理、20 余款云产品维度的成本优化分析能力、10 余款核心云产品支持可运营能力。
- **【运营云】**：对外运营试点项目落地，形成标准方案；智算上云；云 + 应用的一体化运维。
- **配套产品**：一站式迁移平台（调研、评估、迁移）；COMS（规范运营体系、云资源全生命周期管理、计量计费）；AOP（云 + 应用一体化运维）。
- **主要挑战**：跨平台迁移效率（应用、数据库、大数据等）、云管智能程度（租户维度监控及成本分析）、可持续经营（云 + 应用运维与运营）。

5. 案例解析

（1）案例背景

为更加客观的体现一个政务云的发展历程，本文以某城市政务云为例，同时也将结合阿里云在其它客户的共性问题进行辅助说明。

（2）案例概要

自 2009 年成立以来，阿里云已走过近 15 年的历程，亲历并推动了云计算从一项新兴技术逐步演变为现代信息化基础设施的核心组成部分。如今，云计算行业已经历了近 20 年的快速发展，而政务行业作为最具代表性的应用领域之一，天然契合云计算的“多租户”、“按需计费”、“集约化管理”和“降本增效”等核心特点。

近年来，云计算在中国各省市的政务领域取得了长足进步，特别是在过去五年间，政务云的应用深度和覆盖广度显著提升，成为推动数字政府建设的重要引擎。然而，随着政务云规模的持续扩大，如何实现高效、可持续的运营与发展，成为了摆在行业面前的一项重大课题。

本案例将从多个维度（包括客户需求、产品设计、技术实现等）出发，系统性地回顾和复盘一个政务云平台从起步到成熟的发展历程，并深入剖析其中影响业务发展的关键问题与应对策略，为政务云的未来建设提供经验借鉴与实践参考。

（3）“建”

（3.1）云商业模式

当前专有云主要的交付模式有两大类：

- **建设类**：主要特点为“一次性建设与被动扩容”、“一次性大成本投入”、“主要后续服务聚焦在云平台与产品的日常运维”。
- **运营类**：
 - **阿里云自运营（运营主体为阿里云或者联合体）**：主要特点为“软件、硬件与机房由阿里云或者投标联合体提供”、“整体设计与持续运营”、“关注业务全生命周期，通过主动服务促活拉新”。
 - **阿里云辅助运营（运营主体是客户，提供技术运营服务）**：主要的特点为“阿里云提供技术运营服务协助客户完成整体设计与持续运营”、“关注业务全生命周期，通过主动服务促活拉新”。
 - **投资回报率（ROI）计算**：对于有运营诉求的项目（不管是自运营还是辅助运营），都会涉及到项目的分润比例计算，前前后后经历了近 10+ 的项目，一般 ROI 的测算主要关注以下部分：
 - **收入测算部分**：主要结合每个产品的当前集群可售卖量 * 单价 * 安全水位库存，计算出年云资源总收入。难点说明，对于部分 IaaS 产品相对简单选择一个主流规格换算成对应数量即可（例如：ECS 可以用 4c8g、RDS 可以用 4c16g 等），但对于部分 PaaS 产品尤其是大数据产品计算会相对复杂（例如：ADB 3.0、ADB For PG、MQ 等）需要根据实际场景确定。
 - **公共云价格参考**：考虑到部分产品客户并没有定价，因此可以参考公共云的价格作为依据进行测算。

- **分润场景成本测算：**硬件成本（服务器、机房、机架位、带宽）、软件成本（所有产品的 license）、服务成本（运维、运营、安全等），其中除了硬件及交付是一次性投入，其它相关费用全部按照 3 年、5 年两个维度进行计算。
- **盈亏平衡分析：**按照 3 年、5 年两个维度进行分析，收入分析会按照保守（售卖率偏低）与正常（售卖率正常）进行分析是否能收支平衡。
- **难点说明：**ROI 测算的整体逻辑需要对当前版本的专有云产品非常了解，尤其涉及到部分云产品在不同专有云版本架构的差异，同时要重点关注各个云产品的可售卖逻辑，避免出现无法计费的问题。

政务云是一个非常典型的类公共云市场场景，对云厂商的运营技术团队提出了极高的要求，涵盖技术能力、市场洞察以及客户关系管理等多个方面。然而，有一点是明确的：无论采用何种建设或合作模式，政务云的长远发展与壮大都离不开后期持续且高效的运营支持。

（3.2）云服务能力

当前政务云主流的计费模式都是政府采购云资源，政务云由运营商或者国资公司进行承建运营，所以不管是建设模式还是运营模式最为重要的就是计量计费与产品能力设计，需要协助政府输出一版云的服务目录，但是由于阿里云的产品涉及非常多的 PaaS 产品，这里需要关注如下几类问题：

- **云产品规划问题：**往往政务云最大的变数就是上面的业务不确定性，如何规划云产品种类与数量是非常专业的一件事情，因此在设计云产品种类是非常依赖迁云实施经验，包括：政务外网区和互联网区产品数量配比、各种数据库类产品的选择（尤其是 Oracle RAC 上云问题，PolarDB-O 等方案）等。
- **服务目录命名问题：**大部分的省份都是多云服务商，因此一份服务目录要兼容多个厂家，会存在 A 云厂商有该服务 B 云厂商没有，以及产品的命名不同等问题。因此最终确定产品名称时一定要提前和客户对齐，尽可能不要出现名称模糊的产品，例如：分布式数据库、大数据等。
- **服务目录规格问题：**阿里云的产品和部分友商产品区别较大，尤其是大数据、中间件、分布式数据库等，每个产品有多个计价维度。同时，还有像数据库（如：Redis、MongoDB 等）又分主从版本、集群版本以及共享与独享模式等，因此，所有的内容都要在规格里面精确体现，梳理工作非常繁琐。

- **服务目录计量问题：**有分政务云客户要求极高，例如存储（块存储、文件存储等）要按照使用量计费（举例：ECS 磁盘分配一个 1T 的盘，但是只使用了 10G，因此只能按照 10G 计费），且要每日提供具体的监控数据体现，每个月基于近 30 天的数据求平均，对账难度几何级增加，且会带来部分资损。

（4）“管”

（4.1）云平台水位管理

随着业务的陆续上云，前期规划不合理的问题将逐渐显现，但由于政务业务的不确定性也增加了该问题解决的复杂度，因此最好的办法是交付前仔细规划（尽可能规避掉错选漏选的问题）交付后持续经营（制定完善的扩容策略），来避免因为水位问题导致无法上云。如果确实无法规划，提供两个解决思路：

- **思路一（资源的腾挪）：**一般情况两个云区都规划有一些共性产品（SLB、ECS 等），因此如果一个云区资源不足可以考虑从另外一个云区进行缩容腾挪。
- **思路二（同机型替换）：**如果出现部分产品无法腾挪，可以考虑将售卖率不高的产品进行改造替换。

（4.2）云平台三方云管对接

该市历史上曾出现过部分厂商与委办厅局在资源使用数据上报方面存在不一致的情况，云主管部门对资源使用的透明性和准确性提出了更高要求。为确保数据的规范性和可追溯性，主管部门要求所有云服务商接入一个由其自主研发的第三方云管理平台（以下简称“三方云管”）。该平台通过部署 Agent 直接采集各云厂商的资源使用数据，而不接受通过 API 提供的数据（客户认为 API 数据可能存在一定的局限性）。这一要求带来了以下挑战：

- **IaaS 指标对接：**客户要求通过自研软件的 Agent 进行指标采集，包括 CPU、MEM、DISK 等，并明确表示不接受通过 API 进行数据对接。经过技术评估，若在宿主机上部署 Agent，可能对云平台的稳定性带来一定的潜在影响。如果强行部署，需向客户书面说明可能的技术风险，并由客户确认是否接受相关影响。

- **应对策略及风险分析：**在云服务商参与运营的项目中，可能存在角色定位上的多重性，导致数据采集和使用情况的透明度需要进一步提升。为避免后续交付过程中出现争议，建议在项目初期明确三方云管的对接范围和技术实现方式，并与客户充分沟通技术能力和限制。
- **未来规划建议：**目前，ASCM、COMS 等平台的运营能力正在逐步增强，但为避免后续运营过程中可能出现的问题，我们建议在项目初期与客户充分对齐三方云管的对接范围和技术实现方式。对于部分性能指标的采集，需明确技术能力和限制，并与客户达成一致。此外，应持续优化平台的运营能力，确保在满足客户对数据规范性要求的同时，尽可能降低对云平台稳定性和性能的影响。

6. 小结

通过本文的探讨，我们全面回顾了企业在云计算迅猛发展背景下，从传统 IT 管理向云原生运营转型的历程。文章分析了当前项目的特点，详细介绍了云上业务运营体系，并以政务云为例，深入剖析了技术、运营和组织层面的主要挑战及应对策略。我们还通过具体案例展示了如何构建和管理云商业模式与服务能力，确保云平台的高效运作和三方云管对接。展望未来，企业需持续深化用云之路，充分利用云计算的弹性、灵活性和智能化优势，实现业务连续性、降本增效和自动化运维的目标。这一转型不仅可以提升企业的 IT 能力，更为其在数字化时代的持续创新和发展奠定坚实基础。

第二节 / IT 架构的全方位“体测”—— 云化应用架构模型“4L6D” 全解析

1. 引言

在当今信息技术飞速发展的时代，通过将应用程序迁移到云环境中，企业可以获得更高的灵活性、可扩展性与成本效益。我们认为，应用程序只是部署到云环境中并不能充分发挥云的优势，更好的方式是将应用架构完成云化的升级改造，以便于充分利用云产品的技术红利。在云化的全生命周期过程中，企业也会面临着一系列技术挑战。

2. 云化过程面临的主要挑战

- **复杂性管理**
 - 微服务架构与容器技术：虽然极大地提高了灵活性，但也带来了复杂性管理的挑战。每个微服务需要独立部署和管理，如何监控、调试和维护这些服务是企业必须面对的问题。
 - 依赖性与网络延迟：微服务的相互依赖性、网络延迟及失败情况都可能导致问题的排查变得更加困难。
- **性能与可靠性**
 - 分布式架构：在阿里专有云中，应用程序通常分布在多个可用区和服务节点上。这种分布式架构提升了可用性和可靠性，但也面临性能的挑战。
 - 网络延迟与带宽限制：网络延迟、带宽限制和中断问题会直接影响到应用的响应速度和用户体验。

- 性能优化：在对处理性能要求较高的业务场景中，应用往往需要投入额外的资源来优化应用架构，例如使用负载均衡技术来进行的集群化处理设计，通过分布式缓存技术（CDN，反向代理，分布式对象缓存）来降低系统的读负载压力，和利用消息队列异步化解耦来降低系统的写负载压力等技术，确保即使在高负载情况下，应用依旧能够保持良好的性能表现。
- **数据管理与一致性**
 - 跨域数据请求：在云化架构下，数据的存储与管理变得更加复杂，企业通常需要处理跨多个服务和数据库的跨域数据请求。
 - CAP 定理：在分布式系统中，一致性、可用性和分区容错性三者只能选择其二，因此企业在设计数据架构时，必须权衡一致性与可用性之间的关系。
 - 安全性：数据的备份、恢复策略和加密存储等安全问题也需要考虑。
- **遗留系统整合**
 - 兼容性与整合能力：遗留系统通常是公司运作的重要组成部分，其与现代云架构的兼容性和整合能力直接影响到迁移的效果。
 - 系统改造：将遗留系统与新的云服务进行无缝对接，可能需要修改系统架构或进行数据迁移，这不仅对技术人员的能力提出高要求，也可能涉及到相应的风险与成本。
- **资源管理与优化**
 - 按需扩展：云计算带来的按需扩展的特性使得企业可以快速适应市场变化。然而，在这种高度动态的环境中，如何有效管理和优化资源使用是一个重大挑战。
 - 成本控制：过度的资源使用可能导致成本飙升，而资源不足又可能导致性能下降。
 - 自动化工具：企业可以考虑使用阿里云的自动扩展功能（Auto-scaling）、容器编排工具（如 Kubernetes）以及基于策略的资源管理工具来保障资源的高效使用。
- **技术栈的选择与技术债务**
 - 技术栈多样性：在阿里专有云中，企业需要选择合适的技术栈以支持云架构。不同的云服务和工具（如 Docker、Kubernetes）给企业提供了多样的选择，但这也可能导致技术债务的增加。
 - 技术选型：企业需要对技术选择进行深入的分析，确保所选技术不仅能够满足当前需求，还要具有良好的扩展性和社区支持。

- 阿里云生态：利用阿里云提供的丰富服务和工具，如阿里云容器服务 ACK、函数计算 FC、消息队列 RocketMQ 等，可以简化技术选型过程，并减少技术债务。

不难看出，应用架构的云化在为企业带来灵活性与可扩展性优势的同时，也伴随着诸多复杂的技术挑战。从复杂性管理、网络性能、数据管理、安全性、遗留系统整合，到资源管理、技术栈选择等方面，企业需要全面评估并制定合理的应对策略。只有这样，企业才能在不断变化的市场环境中保持竞争优势，实现数字化转型的成功。

面对上述挑战，我们需要一个体系化的云化应用架构评估模型来作为应用架构迭代演进的指导性框架。具体来说，我们需要关注以下几个方面：

- **当前的应用架构到底存在哪些能力上的不足**：进行全面的技术评估，识别现有架构在服务化、高可用、高性能、高弹性、可观测性、自动化等方面的不足。
- **应该通过什么样的架构模式和技术实现来实现能力的提升**：采用微服务架构、容器化、无服务器架构等现代架构模式，结合阿里云的服务和工具，提升应用架构的能力。
- **面对纷繁复杂的技术架构和产品应该如何选型**：根据业务需求和技术成熟度，选择合适的阿里云服务和工具，如 ACK、FC、RocketMQ 等。
- **采用什么样的路径和策略进行应用架构的重构升级**：制定详细的重构计划，分阶段实施，逐步将现有架构迁移到云原生架构。
- **应该重点进行哪些方面的应用架构的建设才能有效的应对当前以及未来的业务挑战**：重点关注服务化、高可用、高性能、高弹性、可观测性、自动化六个维度，构建健壮、灵活、高效的云化应用架构。

综上所述，我们需要通过一个体系化的评估和优化框架，让企业可以在阿里飞天云环境中实现应用架构的云化升级，充分发挥云的优势，应对当前和未来的业务挑战。

3. 架构特征能力矩阵

为此，我们提出了“4L6D”模型，将云化应用架构从六个维度和四个等级进行分类。该模型不仅作为架构设计的指导框架，还集成了简洁且易于落地的评估方法，旨在实际项目实践中充分发挥其指导作用，帮助企业和开发者更好地建设和实施云化应用架构。

我们将云化应用架构涉及的架构因子、架构特性、架构能力，聚类抽象出 6 个主要的架构特征，分别为：服务化、高性能、弹性扩展、可观测、高可用、自动化。

架构特征	服务化	高性能	高可用	可观测	高弹性	自动化
特征等级	· IACCL-1:单体应用 · IACCL-2:部分微服务化 & 网关/消息互通 · IACCL-3:全部微服务化 & 初步服务治理 · IACCL-4:完善的服务治理体系(Agent/ Mesh)	· IACCL-1:无性能监控优化和扩展能力 · IACCL-2:缓存/异步/分布式 & 性能调优 · IACCL-3:系统性能基线 & 容量水平扩展 · IACCL-4:全链路压测 & 精细化容量评估	· IACCL-1:应用单点 & 无系统容错能力 · IACCL-2:应用高可用（限流/熔断/降级） · IACCL-3:业务容灾（同城、异地应用双活） · IACCL-4:业务单元化多活 & 分钟级切流	· IACCL-1:散点日志 & 基础资源监控 · IACCL-2:日志处理分析 & APM监控 · IACCL-3:链路级 Tracing & Metric度量 · IACCL-4:云 + 应用一体化全景监控	· IACCL-1:全人工扩缩容（固定容量） · IACCL-2:负载感知 & 资源节点弹性伸缩 · IACCL-3:多维度指标 & 应用秒级弹性伸缩 · IACCL-4:一云多 Region资源全局弹性调度	· IACCL-1:手动发布及运维 · IACCL-2:应用容器化 & CI/CD交付流水线 · IACCL-3:资源创建 & 配置变更编排自动化 · IACCL-4:基于云 + 应用的全自动化运维
架构能力	1.微服务拆分设计能力 2.服务发布订阅能力 3.配置自动化下发与热生效能力 4.微服务入口集中管控能力 5.服务身份认证与授权能力 6.服务流量管控能力（灰度/标签路由等） 7.统一的微服务治理平台 8.无侵入的治理框架接入能力	1.性能观测与剖析能力 2.集群化服务能力 3.高效分布式事务处理能力 4.应用性能优化能力（多级缓存/消息解耦） 5.服务容量水平扩展能力 6.数据存储水平扩展能力 7.性能基线与关键性能指标 8.容量评估和全链路压测能力	1.服务流量防护能力 2.网关流量防护能力 3.集群流量防护能力 4.熔断降级能力 5.系统防护能力 6.服务自愈能力 7.同城业务容灾能力 8.异地业务容灾能力 9.业务单元化多活能力	1.日志信息的规范化和完整度 2.日志的收集、格式化、分析和全局检索能力 3.应用APM信息监控能力 4.多层次基础资源运行状态指标监测能力 5.数据库与中间件等PAAS层组件的指标监测能力 6.应用及组件定向追踪监控能力 7.链路追踪能力 8.云 + 应用一体化监控能力	1.虚拟机维度弹性伸缩能力 2.基于容器的垂直伸缩和水平伸缩能力 3.基于优先级的调度能力 4.基于负载的弹性伸缩能力 5.大规模容器快速弹性和秒级启动能力 6.基于自定义指标弹性伸缩能力 7.基于事件驱动与弹性预测伸缩能力 8.多Region资源弹性调度能力	1.基于容器+CI/CD 实现自动化交付能力 2.基础设施声明式自动化运维 3.应用编排与模板部署能力 4.应用服务混合编排部署能力 5.基于部署模板的自动增量更新能力 6.多环境管理与自动化部署能力 7.基于云+应用的统一配置管理能力 8.基于云 + 应用的全自动化运维能力

图 1 “4L6D” 架构能力特征矩阵

其中 6 大架构特征的各主要能力项包括：

• 服务化

- 微服务拆分设计能力
- 服务发布订阅能力
- 配置自动化下发与热生效能力
- 微服务入口集中管控能力
- 服务身份认证与授权能力
- 服务流量管控能力（灰度 / 标签路由等）
- 统一的微服务治理平台
- 无侵入的治理框架接入能力

• 高性能

- 性能观测与剖析能力
- 集群化服务能力
- 高效分布式事务处理能力
- 应用性能优化能力（多级缓存 / 消息解耦）
- 服务容量水平扩展能力
- 数据存储水平扩展能力
- 性能基线与关键性能指标
- 容量评估和全链路压测能力

• 高可用

- 服务流量防护能力
- 网关流量防护能力
- 集群流量防护能力
- 熔断降级能力
- 系统防护能力
- 服务自愈能力
- 同城业务容灾能力
- 异地业务容灾能力
- 业务单元化多活能力

• 可观测

- 日志信息的规范化和完整度
- 日志的收集、格式化、分析和全局检索能力
- 应用 APM 信息监控能力

- 多层级基础资源运行状态指标监测能力
- 数据库与中间件等 PaaS 层组件的指标监测能力
- 应用及组件定向追踪监控能力
- 链路追踪能力
- 云 + 应用一体化监控能力

• 高弹性

- 虚拟机维度弹性伸缩能力
- 基于容器的垂直伸缩和水平伸缩能力
- 基于优先级的调度能力
- 基于负载的弹性伸缩能力
- 大规模容器快速弹性和秒级启动能力
- 基于自定义指标弹性伸缩能力
- 基于事件驱动与弹性预测伸缩能力
- 多 Region 资源弹性调度能力

• 自动化

- 基于容器 +CICD 实现自动化交付能力
- 基础设施声明式自动化运维
- 应用编排与模版部署能力
- 应用服务混合编排部署能力
- 基于部署模板的自动增量更新能力
- 多环境管理与自动化部署能力
- 基于云 + 应用的统一配置管理能力
- 基于云 + 应用的全自动化运维能力

4. 架构特征能力等级

针对每个架构特征，会分出相应的 4 个等级，并进行 ACCL（Application Cloud Capability Level）等级评分，分别对应如下：

云化应用架构 综合能力等级	ACCL-1: 单体应用	0分	ACCL-1: 无性能监控优化和扩展能力	0分	ACCL-1: 全人工扩缩容（固定容量）	0分	
	ACCL-2: 部分微服务化 & 网关/消息互通	2分	ACCL-2: 缓存/异步/分布式 & 性能调优	2分	ACCL-2: 负载感知 & 资源节点弹性伸缩	2分	
	ACCL-3: 全部微服务化 & 初步服务治理	4分	ACCL-3: 系统性能基线 & 容量水平扩展	4分	ACCL-3: 多维度指标 & 应用秒级弹性伸缩	4分	
	ACCL-4: 完善的服务治理体系 (Agent/Mesh)	6分	ACCL-4: 全链路压测 & 精细容量评估	6分	ACCL-4: 一云多Region资源全局弹性调度	6分	
	服务化		高性能		高弹性		
综合能力等级计算逻辑							
$f(m) = \sum_{i=1}^6 ACCL_i \text{ Ponits}$ f(m)：云化应用架构综合能力等级 ACCLi Ponits: 架构特性能力等级分数 ➡		1. 初始级 (<12)	业务系统主要以单体应用为主，尚未开始进行微服务化拆分和改造，没有性能监控及优化能力，不具备弹性伸缩能力，自动化运维机制不健全。				
		2. 基础级 ($\geq 12 \ \& \ < 18$)	业务系统在局部范围开始尝试微服务化及容器化改造，有相应的性能监控、优化的措施和手段，具备应用高可用防护能力，在弹性和自动化程度上获得局部能力提升，计算、存储、网络、数据库等主要资源已经采用云服务托管模式。				
		3. 发展级 ($\geq 18 \ \& \ < 24$ 单项不低于 ACCL-2)	业务系统已经全面采用微服务架构并进行了容器化改造，建立了基础的服务治理体系，应用云化技术架构应用程度较深，架构的弹性、自动化、高可用、可观测和自愈能力已有较大提升。				
		4. 成熟级 (≥ 24 & 单项不低于 ACCL-3)	具备完善的服务治理体系，建立了比较完备和成熟的应用云化技术架构体系，业务系统在多地域多站点单元化部署运行，具备跨Region的全局资源弹性调度能力和业务多活能力，能够进行云+应用的一体化自动运维。				
ACCL: Application Cloud Capability Level (1 - 4)	高可用	ACCL-1: 应用单点 & 无系统容错能力	0分	ACCL-1: 散点日志 & 基础资源监控	0分	ACCL-1: 手动发布及运维	0分
前提：应用架构设计 基于云的IAAS和PAAS产品		ACCL-2: 应用高可用（限流/熔断/降级）	2分	ACCL-2: 日志处理分析 & APM监控	2分	ACCL-2: 应用容器化 & CICD交付流水线	2分
		ACCL-3: 业务容灾（同城、异地应用双活）	4分	ACCL-3: 链路级Tracing & Metric度量	4分	ACCL-3: 资源创建 & 配置变更编排自动化	4分
		ACCL-4: 业务单元化多活 & 分钟级切流	6分	ACCL-4: 云 + 应用一体化全景监控	6分	ACCL-4: 基于云 + 应用的全自动化运维	6分

图 2 “4L6D” 架构特征能力等级

• 服务化

◦ ACCL-1: 单体应用

应用系统采用的是传统的 Monolithic Application 架构风格。

◦ ACCL-2: 部分微服务化 & 网关 / 消息互通

系统中部分应用开始采用微服务架构体系设计，开始采用应用网关消息队列等技术。

◦ ACCL-3: 全部微服务化 & 初步服务治理

核心应用全部微服务化，并具备初步的服务治理体系（服务注册与发现，配置管理，流量管控等）。

- **ACCL-4: 完善的服务治理体系 (Agent/Mesh)**

具备较完备的服务治理体系，能够进行精细化的流量管控，并且服务治理体系的拉入是采用非侵入式的方式，不与业务应用系统强耦合。

- **高性能**

- **ACCL-1: 无性能监控优化和扩展能力**

没有性能监控和 Profiling 技术手段，不具备性能扩展能力。

- **ACCL-2: 缓存 / 异步 / 分布式 & 性能调优**

具备性能调优的技术手段和工具，采用了多级缓存 / 异步消息解耦 / 分布式集群处理等提高性应用处理能的技术方案。

- **ACCL-3: 系统性能基线 & 容量水平扩展**

构建了相对完整的系统性能基线数据，关键性能指标明确，并有针对性的优化措施。同时系统容量具备较大范围的水平扩展能力。

- **ACCL-4: 全链路压测 & 精细化容量评估**

具备全链路压测的工具平台和技术能力，能够对核心业务链路上的相关应用服务进行精细化的性能容量评估。

- **高弹性**

- **ACCL-1: 全人工扩缩容（固定容量）**

通常为线下 IDC 环境，应用系统容量刚性，依赖人工进行扩缩容。

- **ACCL-2: 负载感知 & 资源节点弹性伸缩**

能够对应用系统当前负载情况（cpu,mem,load 等）进行感知，并跟负载情况指标进行资源的扩缩容。

- **ACCL-3: 多维度指标 & 应用秒级弹性伸缩**

能够使用一些应用定义的扩展指标（例如：HTTP 请求率、Ingress QPS 等指标），快速的进行应用容量的扩缩容。

- **ACCL-4: 一云多 Region 资源全局弹性调度**

应用系统跨地域多站点部署，可以进行全局的流量调度和弹性伸缩。

- **高可用**

- **ACCL-1: 应用单点 & 无系统容错能力**

因为有状态等原因，应用单点部署，并且没有自动化的故障恢复能力。

- **ACCL-2: 具备应用高可用能力（限流 / 熔断 / 降级）**

应用具备流量防护能力（多维度限流）、服务容错能力（熔断降级等）以及系统防护能力。

- **ACCL-3: 业务容灾（同城、异地应用双活）**

具备业务级的同城和异地容灾能力。

- **ACCL-4: 业务单元化多活 & 分钟级切流**

具备业务级的单元化多活能力，能够进行分钟的跨地域流量调度和容灾恢复。

- **可观测**

- **ACCL-1: 散点日志 & 基础资源监控**

应用日志分散且格式不统一，只具备简单的主机级的基础资源监控。

- **ACCL-2: 日志处理分析 & APM 监控**

应用日志进行了规范化处理，并且进行了统一采集和存储，可以进行集中的分析处理，并且具备应用性能监控手段。

- **ACCL-3: 链路级 Tracing & Metric 度量**

在日志的基础上，同时具备聚合指标数据的生成、采集和展示，分布式链路信息追踪等能力，形成了 Logging + Metrics + Tracing 的完整的可观测基础设施体系。

- **ACCL-4: 云 + 应用一体化全景监控**

在通过业务日志分析、链路追踪、Health Check 等手段实现端到端全链路监控的基础上，具备从业务、应用到云服务实例一体化全景监控能力。

- **自动化**

- **ACCL-1: 手动发布及运维**

手动进行应用的发布和运维，没有交付流水线等自动化流程和工具。

- **ACCL-2: 应用容器化 & CICD 交付流水线**

应用进行过容器化改造，建立起来基于容器的 CICD 自动化交付流水线。

◦ ACCL-3: 资源创建 & 配置变更编排自动化

资源编排、创建与变更管理的自动化，具备 IaC 方式的基础设施描述、部署和维护技术。

◦ ACCL-4: 基于云 + 应用的全自动化运维

具备运维任务统一编排入口，统一的运维模板和脚本库维护，统一变更审批引擎，统一规则引擎，以 CMDB 为编排动态资源库，提供业务及网络拓扑形成统一数据视图。具备复杂场景运维作业的编排和自动化执行能力。

5. 小结

根据当前应用架构现状，基于云化应用架构评估模型（“4L6D”模型），计算得出 6 大架构特征能力等级的评分结果，再根据综合计算结果得出当前应用架构所处的发展阶段，依据当前所处架构发展阶段，明确云化应用架构规划方向，并制定出一条合理的云化应用架构演进路径。当然，专有云体系中中间件（EDAS）、容器（ACK、ACR）、应用高可用（AHAS）、容灾多活（ASR、MSHA）和可观测应用运维（AOP）都是提供了标准的分阶段分层的具体能力的，可视情况合理引入。

第三节 /

云上“攻坚”——行业头部客户
核心业务系统用云最佳实践

1. 上云背景

A 业务系统是客户总部统管统建的业务，关系着全国多个分公司用户的民生业务。

A 业务系统在数年间逐渐演变成一个巨大的单体架构，系统之间功能重复、流程间断、交互困难，系统本身架构不统一、技术不统一、数据模型标准不统一，存在扩展能力差、需求响应不及时、迭代周期长，无法支撑互联网时代同时服务内外部的数字化要求。

客户为了更好服务好惠及民生的用户需求，数字化转型迫在眉睫，从烟囱式系统全面重构改造成基于云原生微服务架构的新一代业务系统。

2. 面临的问题和挑战

A 业务系统是客户数字化建设迄今为止系统最复杂、难度最大、技术最先进的任务之一。“如果阿里不能支撑好这个项目，可以说整个行业云平台选型宣告失败”，客户领导如是说。

A 业务系统上云面临的主要挑战如下：

- **业务复杂 / 规模大 / 时间紧**：不到半年时间研发重构数十套复杂系统，每个分公司数百个微服务应用、数千个容器 POD 实例，上千人研发团队。
- **厂商传统技术思维根深蒂固 / 云原生能力不足**：传统 IOE 架构技术，不足 1% 的人熟悉云产品，在系统重构过程中，云原生架构设计能力不足，互联网与传统思想碰撞激烈。数十款云产品没有最佳实践指导。

- **稳定性 / 性能要求高**：国计民生、数亿用户、数千万企业，影响面广，故障零容忍。

客户超大的用户体量和复杂的业务形态，是对阿里专有云组件产品和专家服务的一次大考。

3. 最佳实践方案

(1) 建立架构委员会，制定云原生架构设计标准

客户在云原生架构设计和落地时遇到三个难题：“架构规划和设计缺乏标准参考、云原生技术路线和产品庞杂选型困难、产品如何结合业务使用没有最佳实践做指导”。我们围绕客户上云技术目标：“高性能、高弹性、高可用、高效率”，梳理了一套云化架构设计和实施落地框架标准，在设计和落地时作为参考尽可能减少试错成本，见右图。左图抽象了云化技术架构和企业架构的关系，我们在技术架构上做重点支撑。

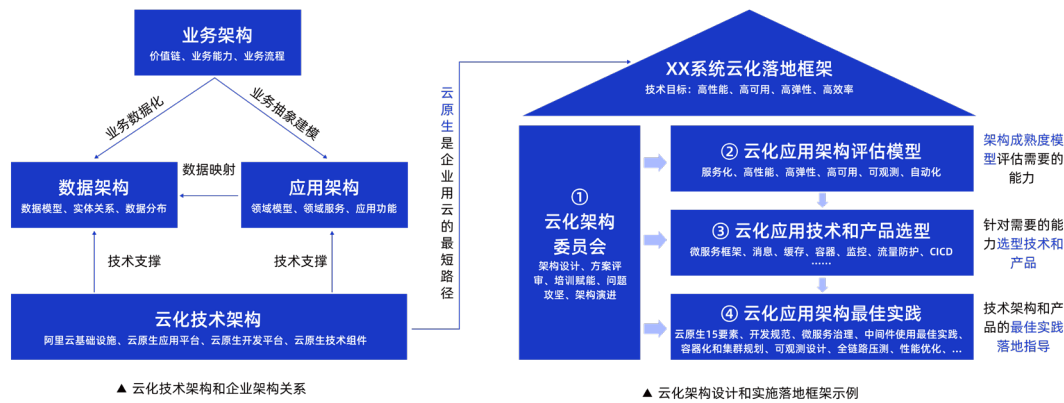


图 1 云原生架构设计标准示例

① 云化转型下组织转型先行。我们同客户成立云原生架构委员会，该组织主要负责架构设计和选型、方案评审、培训赋能、问题攻坚，来确保架构支撑业务同时保证技术的领先性、通过最佳实践指导平稳落地，并能持续演进。

② 结合客户的技术目标，参考“云化应用架构能力评估 4L6D 模型：服务化、高性能、高弹性、高可用、可观测、自动化”选择对应实现架构的能力，经综合评估 A 业务系统架构的建设和运行处于“发展级”才能满足目标。

云化应用架构能力评估模型	蓝色内容 为 A 系统的架构能力 24分 发展级		服务化		高性能		高弹性	
	ACCL-1: 单体应用	0分	ACCL-1: 无性能监控优化和扩展能力	0分	ACCL-1: 全人工扩缩容（固定容量）	0分	ACCL-2: 负载感知 & 资源节点弹性伸缩	2分
	ACCL-2: 部分微服务化 & 网关/消息互通	2分	ACCL-2: 缓存/异步/分布式 & 性能调优	2分	ACCL-2: 应用容器化 & CI/CD	2分	ACCL-3: 多维度指标 & 应用秒级弹性伸缩	4分
	ACCL-3: 全部微服务化 & 初步服务治理	4分	ACCL-3: 系统性能基线 & 容量水平扩展	4分	ACCL-3: 资源创建 & 配置变更编排自动化	4分	ACCL-4: 基于云 + 应用的全自动化运维	6分
	ACCL-4: 完善的服务治理体系 (Agent/Mesh)	6分	ACCL-4: 全链路压测 & 精细化容量评估	6分				
1. 初始级 (<12)								
2. 基础级 (>=12 & < 18)								
3. 发展级 (>=18 & < 24& 单项不低于 ACCL-2)								
4. 成熟级 (>=24 & 单项不低于 ACCL-3)								
ACCL: Application Cloud Capability Level (1 - 4)	高可用		可观测		自动化			
前提：应用架构设计 基于云的IAAS和PAAS产品	ACCL-1: 应用单点 & 无系统容错能力	0分	ACCL-1: 散点日志 & 基础资源监控	0分	ACCL-1: 手动发布及运维	0分		
	ACCL-2: 应用高可用（限流/熔断/降级）	2分	ACCL-2: 日志处理分析 & APM监控	2分	ACCL-2: 应用容器化 & CI/CD	2分		
	ACCL-3: 业务容灾（同城、异地应用双活）	4分	ACCL-3: 链路级Tracing & Metric度量	4分	ACCL-3: 资源创建 & 配置变更编排自动化	4分		
	ACCL-4: 业务单元化多活 & 分钟级切流	6分	ACCL-4: 云 + 应用一体化全景监控	6分	ACCL-4: 基于云 + 应用的全自动化运维	6分		

图 2 A 系统云化应用架构能力

③ 根据“云化应用架构模型”评估出来的能力，需要选择合适的云原生开源技术和对应产品来支撑实现。如何做技术选型？策略是什么？在【上云规划篇】的产品选型章节已经介绍过，这里不做过多赘述。这里围绕客户微服务框架选型给出典型方案如上图所示：

技术选型策略	① 需要场景匹配度	② 综合成本效益	③ 技术成熟度	④ 技术领先性
	选择能够满足当前和未来业务&技术需求的工具。确保工具能够随着业务的增长而扩展，并能够处理未来可能出现的新用例。	要评估其实施、运营和维护的TCO总成本。（初始成本 + 运营成本 + 培训成本 + 维修和支持 + 升级和替换 + 潜在机会成本 + 退役和处置成本。）	在行业头部公司已经有生产环境规模化应用并且开源的，有活跃社区支持的和良好生态的工具/产品	选择在技术上有前瞻性且不断创新的工具，以确保能够利用最新的技术发展并支持未来的技术趋势。

服务化		高性能		高可用		可观测		高弹性		自动化	
功能特性	工具/产品	功能特性	工具/产品	功能特性	工具/产品	功能特性	工具/产品	功能特性	工具/产品	功能特性	工具/产品
服务调用	OpenFeign/Ribbon/Dubbo/HSF	性能观测与剖析	OpenTelemetry/ARMS	服务流量防护	Sentinel	日志的收集、格式化、分析和全局检索	ELK/SLS	虚拟机弹性伸缩	ESS/k8s + cluster-autoscaler	基于容器的CI/CD	Argo/EDAS/云效
服务注册与发现	Nacos	多级缓存/消息解耦	CDN/Reids/RocketMQ	网关流量防护	SpringCloud Gateway + Sentinel	应用APM信息监控	ARMS	容器弹性伸缩	K8s + VPA & HPA	基础设施声明式自动化运维能力	Terraform
分布式配置	Nacos	集群化服务	Ribbon/K8S, ReplicaSet	集群流量防护	Sentinel + token server	多层次基础资源运行状态指标监测	Prometheus	大规模容器快速弹性	ECI	应用编排与模板部署	Helm/Operator
服务认证授权	Oauth2/JWT/EDAS	服务容量水平扩展	EDAS	熔断降级	Sentinel	数据库与中间件等PaaS层组件的指标监测	Prometheus/ARMS	基于弹性资源优先级的调度	ACK/Resource Policy&kube-scheduler	应用与PaaS服务混合编辑部署	Terraform + Helm Charts/AOP
服务入口网关	Springcloud Gateway/MSE	数据存储水平扩展	PolarDB-X	系统防护	Sentinel (Load/RT/QPS/线程数)	应用及组件定向追踪监控	Arthas	基于应用负载的弹性伸缩	ACK/kube-scheduler	基于部署模板的自动增量更新	Argo/Helm
服务流量治理	Istio/MSE/ASM	分布式事务	Seata	服务自愈	EDAS + 微群实例摘除+服务探测	链路追踪能力	OpenTelemetry/ARMS	基于自定义指标的弹性伸	ACK/alibaba-cloud-metrics-adapter	应用多环境管理与自动化部署	EDAS
微服务治理平台	EDAS	性能指标监控	Prometheus/ARMS	业务容灾能力	MSHA			基于事件驱动伸缩与弹性预测伸缩	ACK/KEDA&AHPA	基于云+应用的统一配置管理	AOP
无侵入框架	Istio/ASM	容量评估与压测	Jmeter/PTS	业务单元化多活	MSHA	云 + 应用一体化监控	AOP	多Region资源弹性调度	一云多Region基础架构	基于云 + 应用的全自动化运维	AOP

图 3 微服务框架技术选型参考

经过同客户很多轮激烈讨论和共创，结合业务场景、技术到产品的选型取舍、产品能力适配及验证、优劣势比对和风险评估，最终形成了“A系统云原生技术路线”架构。

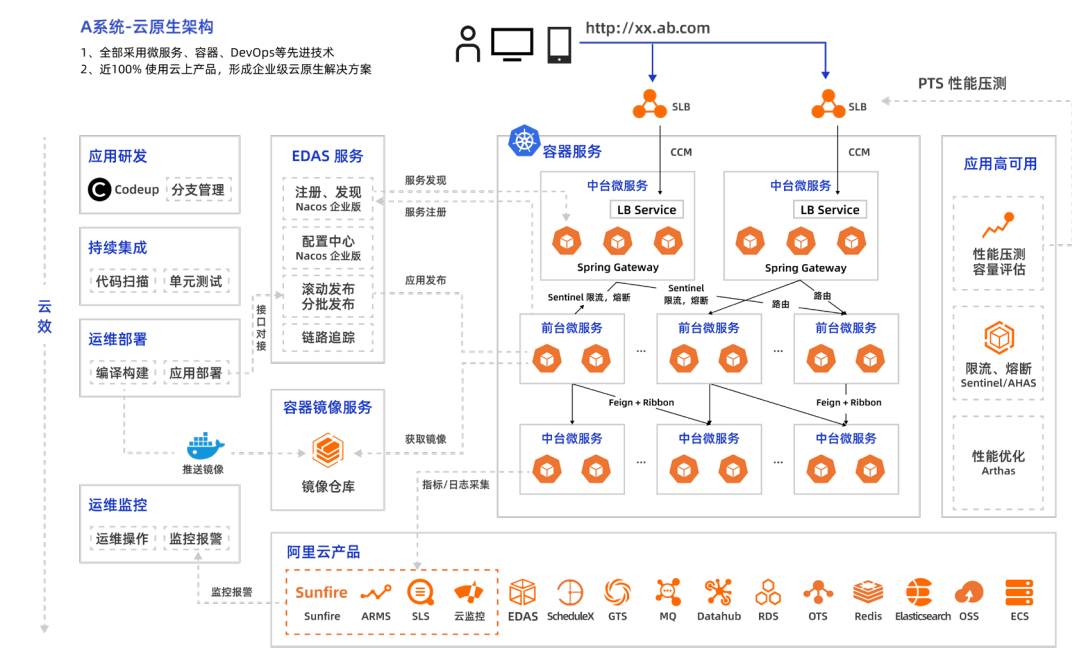


图 4 系统云原生技术架构

④ 明确技术和产品后需要最佳实践指导落地。

（2）深入核心业务场景设计技术方案，解决核心业务痛点

针对“客户最关注、最头疼、技术难度最高，链路最长”的业务场景进行方案设计。

（2.1）“三大”关键架构设计

a. 应用架构重构设计

• **数据层：**决定禁止使用存储过程，核心业务场景相关的逻辑处理类、数据处理类、统计类全部从数据库上移到应用程序中实现，确保数据库只承担 ACID 的角色，防止应用空心化。新的设计方法需要客户及开发商下定决心和勇气摒弃多年来既有的设计思想,进行适配改造,为后续的“应用高弹性和可扩展性”打下基础。

• **应用层：**大型单体应用按照 DDD 领域设计方法，基于业务领域维度进行分层拆分，形成微服务。服务间信息交互通过 MQ 消息异步通信解耦；档案、计划、中间态运算数据和状态全部使用 Redis 缓存，提升系统性能和响应速度。

• **基础设施层：**应用微服务全部容器化部署，通过 ACK 集群进行资源编排，根据用户业务规模和应用性能负载进行快速扩缩容，保证高弹性。

b.OTS 数据存储设计

根据业务场景、时序特性和大数据量，我们决定放弃 DB 存储，选用 TableStore-OTS 产品。

① 不同类型数据项通过 OTS 行存宽表模型来存储，时间戳版本属性来满足多版本数据值，单表百亿数据操作能保持高性能。

② OTS 主键设计是关键，主键能唯一确定表中的一行记录，根据业务特点，采用终端号 + 日期 + 数据编码项等作为主键，为了防止顺序增长导致表格分区数据倾斜，影响读写速度，我们采用 MD5 算法进行打散，把它作为分区键，防止数据倾斜和提高读写速度。

③ 同时对 OTS 表进行了分区扩展，将数据分区增加，加大了数据的读写能力。

c.SchedulerX 任务调度计算架构设计

为了保证性能，架构上需要满足成百上千个节点同时分布式并发调度计算。我们采用 SchedulerX 产品 MapReduce 分片模型，支持几十万任务快速分发调度到容器应用上，任务触发时，从 Redis 中获取缓存的任务执行计划编号进行计算，再结合容器化高弹性特点，根据目标用户和资源负载可快速的进行应用扩容，提高整体计算能力，在任务结束后可进行缩容，提升集群资源利用率；同时，结合容器健康检查自愈和 SchedulerX 故障恢复能力，在任务计算节点发生故障时，可快速重启或者自动调度到其他可用节点进行重新计算，保证高可用，但业务逻辑要实现幂等。

（2.2）“四大”性能问题攻坚

经过前序整体架构的设计调整，业务性能有大幅提升，但距离目标还有一段距离，于是对性能问题进行重点攻坚。

a. 设计 MQ 消费模型，让优化有理论支持，效果可估算

问题：消息堆积。消息堆积直接影响业务性能，很容易成为瓶颈。我们抽象并设计了消息（以 MQ 为例，DataHub 类似）消费性能优化模型，让优化有理论支持，效果可估算。

优化方案：增加 Threads，即增加消费线程数：

① 基于压测和容量评估对 POD 进行扩容。

② 调整每个消费者的消费线程数（线程数和 CPU 核数关联，对于 I/O 密集型应用，可按 CPU 核数 *（1 + I/O 等待时间 / 计算时间）来算，一般建议 3~4 倍 CPU 核数，来提高程序的并发度。

b.SchedulerX MapReduce 流式计算优化，保证任务连续运行

问题：大量 Pod 任务计算节点处于空闲状态，后面的任务无法持续分发计算，导致整个计算耗时增加。

优化方案：是让 Pod 节点持续运行任务，不空转。用循环小批量任务替换大量任务计划一次性分发。

c.Redis 大 Key 优化，保证计算正常运行

任务运行过程中发现大 Key 导致大量 Redis 慢查询和读取超时导致任务中断。我们的优化方向是：

- 尽量缩减业务中无效的信息，尽量缩减 Key 的大小。
- 增加 Redis 集群的 Proxy 节点来提升吞吐能力。
- 调整 Redis 连接池的“soTimeout”参数，加大读取内容返回时间，防止程序报超时异常错误。

d. 频繁 FullGC 优化，保证系统可用性

问题：程序频繁 FullGC，导致 SchedulerX 任务调度失败、假死或重启，严重影响业务性能和稳定性。原因不复杂，一般由大对象、JVM 设置不合理导致的，而往往在研发设计阶段不太重视。

解决方案：我们通过 ARMS 产品判断 JVM FullGC 异常情况，分析 Pod 资源和 JVM 配置，最终在堆内存、年轻代的大小方面进行优化调整并明显降低了 GC 次数。

（2.3）持续攻坚大量核心技术方案

我们还和客户一同攻坚了很多技术方案，如：**应用多级缓存架构方案、应用跨中心消息方案、应用分布式事务方案、应用发布优雅下线方案、分布式 ID 生成方案**。我们认为：在微服务分布式系统中，如果能处理好“缓存、消息队列和分布式事务”，一般情况下架构的健壮性和性能表现都会非常不错。篇幅原因，这里只介绍下应用发布优雅下线方案。

应用发布优雅下线方案

问题：应用在发布过程中，经常出现执行到一半的请求中断掉，消费者继续调用已经下线的节点导致报错的现象。

解决方案：结合容器下线生命周期，主动注销服务实例，让消费者提前感知，实现优雅下线。

（3）知识转移，提升客户用云质量和信心

通过前期深入业务场景和不断探索实践，我们沉淀了丰富的用云最佳实践，并孵化了系列用云最佳实践课程，该课程设计的目的是帮助客户研发、运维人员站在架构师的视角，从系统全局、产品架构、技术原理层面理解和运用云原生技术和阿里云产品，以真实的业务场景和实战案例分析为基础，注重实践能力的提升。培训内容包括”微服务、中间件、容器、应用发布、可观测和性能压测 & 优化“等课程。通过赋能和知识转移，帮助客户提升用云质量和信心。

（4）通过容量评估 / 可观测 / 压测优化保障系统平稳上线

我们重点从系统性的容量评估（容器集群、中间件、数据库等）、可观测性设计、性能压测和优化三个方面进行设计保障客户系统平稳上线。

（4.1）容器集群规划

a. 容器和节点规模资源评估

容量评估，需要重点考虑服务数、实例规格、实例数，并预留一定的冗余量。

- **服务数**：指应用服务数。
- **容器实例规格**：Pod 内存组成模型【堆内存 + 线程数 * 线程栈 + 永久代 /Metaspace + 堆外内存】，如果 Pod 内存过小、Java 内存过大，会引起 OS OOM Kill 操作系统就会杀死 java 进程。
- **实例数**：刚开始通过业务量级进行人工估算，后面通过 PTS 性能压测试跑再逐步调整数量。
- **冗余量**：包括 DaemonSet, Job 等系统 Pod 资源; kubelet、kube-proxy 等预留资源，确保节点上系统组件和控制平面有足够的资源运行，并在资源紧张时进行驱逐策略；另外还需要考虑高可用能力，以及预留一定的 buffer。
- **节点实例规格**：原则上越大越好，这样节点少 Master 负担也小，资源利用率最好。

b. 集群选型 & 网络规划

集群选型：专有云环境 ACK 集群选择专有版本。

网络插件选型：一般是 Flannel 和 Terway，目前普遍采用 Terway。

c. 关键配置和调度策略

- 网关对外暴露服务尽量不使用 NodePort SVC 进行挂载，使用 Loadblancer SVC。
- 网关满足“反亲和”软策略，尽量让服务部署在不同的机器上，保证足够的高可用，并配置 readiness 和 liveness，来保证访问就绪和自愈能力。

(4.2) 可观测性设计

系统稳定性依赖一个可靠全面的可观测体系，通过实时监控系统运行状态，实现问题提前预警，并辅助快速定位问题，当出现故障时将对业务的影响降到最低。

a. 可观测架构设计

需要根据业务价值自上而下设计，因为客户关注的业务 / 应用指标是有限的，而造成故障的技术原因是无限的。

首先，优先业务监控（客户根据实际业务情况定义监控指标）、应用监控（QPS/RT/ 错误数 / 服务慢调用次数 / 慢 SQL 数 /Full GC 次数等）、中间件容器监控（CPU/ 内存 /Pod 健康度异常数等）、基础设施依次向下推进。

其次，参考可观测性“三大支柱”（Metrics、Traces、Logs）推进指标、调用链和日志的建设和实施配置，我们通过产品 Sunfire、ARMS、SLS、CMS 组合使用来实现。

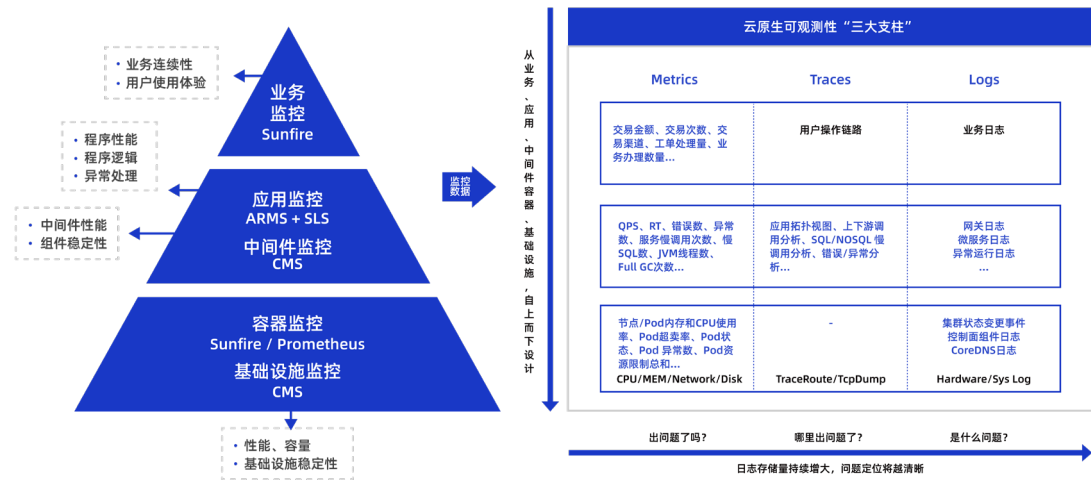


图 5 A 系统可观测架构设计

b. 可观测建设成效展示

我们建议通过大屏、中屏、小屏进行可观测建设成效展示。大屏，进行业务核心关键指标监控，进行全局呈现，形成决策支撑。中屏，进行全局调用链、核心应用性能指标、错误数、慢 SQL、应用依赖、FullGC 监控，帮助研发快速定位根因。小屏 – 通过设置报警阈值，出现异常时通过钉钉或者短信报警的方式推送到手机终端，帮助客户运维能够即时发现和处理问题。

c. 问题快速定位方式

关键是把微服务调用链 TraceId 和业务 / 错误日志数据进行关联。

当应用出现问题时，在 SLS 中查询异常日志找到 Traceid，随后在 ARMS 中查找进行全局调用链分析；反之，或在 ARMS 找到应用异常的 Traceid，在 SLS 中通过 Traceid 查找关联微服务异常日志，能及时分析、定位并解决问题。

(4.3) 性能压测 & 优化

性能压测是一个系统性工程，包含了业务链路、技术架构和服务依赖梳理，压测方案设计和实施，可观测建设，还需要组织协同等。总结来看压测方案最核心点是梳理清楚场景的业务模型、流量模型、数据模型。

性能优化先通过监控工具和 Arthas 等 Java 工具发现并定位到问题，然后结合业务指标和问题进行调优，优化参考点：

- ① 资源优化。
- ② 应用程序和 JVM 优化。
- ③ 中间件和数据库使用优化。
- ④ 架构设计优化。

在压测过程中持续调整资源容量、优化架构和代码，并且持续完善微服务日志和监控指标，直到完成最终的压测目标。我们认为性能优化的主要原则如下：

- ① 深入理解业务，找出架构和系统设计不足。
- ② 结合观测指标值，用数据说话。
- ③ 工欲善其事必先利其器，压测工具很重要。
- ④ 全局架构视角考虑，避免单方向过度优化。
- ⑤ 明确业务目标，权衡调优效果和投入成本。

(5) 同城双活构建高可用和弹性调度能力

根据“云化应用架构能力评估模型”，从“高可用、高弹性、自动化”能力方向持续演进走向“成熟级”，通过阿里云 MSHA+ASR 多活容灾产品能力，助力客户构建应用同城双活架构，让客户具备同城容灾高可用能力和多 Region 资源弹性调度能力。

4. 结束语

阿里云在服务客户的过程中，能够参与到真正惠及民生，为社会创造价值的事情，是每个阿里云人最自豪的事情。围绕“上好云、用好云”的目标，我们会持续打磨产品、积累产品使用最佳实践，服务好每一个阿里云政企客户。

第四节 /

云上数据运营

(一) 云上数据仓库的基石建设——做好智能建模和数据血缘管理

1. 数据仓库中的数据建模思路

(1) 什么是数据模型

数据模型是抽象描述现实世界的一种工具和方法，是通过抽象的实体及实体之间联系的形式，来表示现实世界中事务的相互关系的一种映射。数据模型体现的是现实世界中各个业务实体及其关系，业务实体及其关系的复杂程度决定了数据模型的抽象复杂度，关系越复杂，数据模型也就越复杂。数据仓库模型是针对特定的数据仓库应用系统的一种特定的数据模型。

(2) 为什么要数据建模

数据仓库的发展经历的三个阶段：

- ① 简单报表阶段：解决一些日常的工作中业务人员需要的报表，以及生成一些简单的能帮助领导进行决策需要的汇总数据。这个阶段的大部分表现形式为数据库和前端报表工具。
- ② 数据集市阶段：根据某个业务部门的需要，进行一定的数据的采集、整理，按照业务人员的需要，进行多维报表的展现，能够提供对特定业务指导的数据，并且能够提供特定的领导决策数据。
- ③ 数据仓库阶段：按照一定的数据模型，对整个企业的数据进行采集、整理，并且能够按照各个业务部门的需要，提供跨部门的，完全一致的业务报表数据，能够通过数据仓库生成对业务具有指导性的数据，同时，为领导决策提供全面的数据支持。

通过数据仓库发展的三个阶段可以看出，随着业务的快速发展，企业数据成几何倍数增长，数据量庞大、复杂、各类数据间标准不一致，往往会出现数据难以管理的现象。如何将无序、杂乱、繁琐、庞大且难以管理的数据，进行结构化有序的管理，使存在于企业中的数据产生更多的价值，将数据价值最大化，这就是数据建模的价值。

数据建模给企业带来的好处：

- 海量数据的标准化管理
企业业务越庞大数据结构就越复杂，企业数据量会随着企业业务的快速发展而迅速增长，如何结构化有序地管理和存储数据是每个企业都将面临的一个挑战。
- 业务数据互联互通，打破信息壁垒
公司内部各业务、各部门之间数据独立自主形成了数据孤岛，导致决策层无法清晰、快速地了解公司各类数据情况。如何打破部门或业务领域之间的信息孤岛是企业数据管理的严峻考验。
- 数据标准整合，统一灵活对接
同一数据不同描述，企业数据管理难、内容重复、结果不准确。如何制定统一的数据标准又不打破原有的系统架构，实现灵活对接上下游业务，是标准化管理的核心重点之一。
- 数据价值最大化，企业利润最大化
在最大程度上用好企业各类数据，使企业数据价值最大化，为企业提供更高效的数据服务。

(3) 智能建模的应用实践

(3.1) 业务痛点

回顾阿里体系内众多业务十多年的建设经验，各个数仓团队实际上并不是不需要进行数据建模，而是采用线下 Excel 建模评审的方式在开展这一项工作，流程本身非常规范，模型的上线及变更有着非常严格的评审流程，但即使如此，线下建模还是有它的弊端存在。线下建模的弊端主要体现在三大方面：规范定义、模型设计、数据开发。

从规范定义方面来讲，存在的主要问题是：

- 数仓规范与模型设计分离，符合规范的模型设计对建模师本身的要求非常高，既要能把业务需求高度抽象进行模型设计，还需要牢记规范的点点滴滴。

• 数据指标定义效率低，且指标的数据加工逻辑分离，过去传统的单个创建指标效率相对低下，且无法保证指标的唯一性，指标的加工逻辑和指标定义本身也存在脱节的情况，最终导致指标真实口径无法统一，进而带来了大量的针对指标结果数据不一致的对焦工作。

• 应用层缺少规范，大多数应用层的建设都面临需求多变、需求开发时间紧、任务重的特点，也对应用层模型规范的管理带来了非常高的挑战。既要能够满足业务需求，又要能够符合规范，其实很难再短时间内完成这些工作。

从**模型设计方面**来讲，存在的主要问题是：

纯人工的模型设计效率低下，比如要在 Excel 里做模型设计，并且需求在 Excel 里做维护。

从**数据开发方面**来讲，存在的主要问题是：

模型设计和物理表开发分离，模型设计是模型设计，物理表开发是物理表开发，很有可能会造成物理表开发逻辑与模型设计理念存在或多或少的差异情况。

此外，本地建模，还会存在着一些隐藏问题，如文件管理混乱、硬件设备故障、工作交接难等问题。



图 1 线下建模三大方面弊端

(3.2) 业务价值

数据建模作为数仓规范，最大的受益者是企业自身，但企业价值需要通过一线研发人员的工作得以体现。对于一线研发同学来讲，智能数据建模能为大家带来最大的好处是提效，相比传统的纯开发或者线下建模线上开发的工作方式来说，智能数据建模能为大家带来更加更加高效的建模和研发方式。由此，帮助企业做好企业数据体系的规范化建设，让数仓规范真正能落到实处。企业数仓规范真正做好以后，能为企业沉淀大量体系化的核心数据资产，同时，也能顺其自然地为企业节省大量存储和计算成本。

面向企业
规范：帮助企业数据体系的规范化建设，让规范落到实处。
降本：降低存储成本，降低计算成本。
沉淀：建模由线下转为线上，沉淀企业级核心数据资产。
面向一线研发同学
提效：减少人员沟通成本，产品化支持快速建模以及开发打通，提升建模效率及研发效率。

图 2 智能数据建模价值体现

(3.3) 数据建模方法论

数据（仓）库的模型设计中有两个广为人知的方法论：即维度建模（Dimensional Modeling）和三范式建模（Third Normal Form，简称 3NF），两种建模方法论各自的定义和优劣简述如下：

- 三范式建模：
 - 定义：三范式（3NF）是关系型数据库规范化的一个过程，它旨在减少数据冗余，提高数据的完整性。
 - 第一范式（1NF）：所有字段都是原子的，即每个字段只能包含一个值，不可再分。
 - 第二范式（2NF）：满足 1NF，并且所有非主键字段完全依赖于主键，而不是部分依赖。

- 第三范式(3NF)：满足 2NF，并且所有非主键字段传递不依赖于其他非主键字段，也就是说，消除传递依赖关系。
- 优势：
 - 数据冗余最小化：通过规范化减少数据重复，从而节省存储空间。
 - 更新异常减少：避免更新操作时可能引发的异常问题。
 - 数据完整性高：通过清晰的依赖关系维护数据的合理性。
- 劣势：
 - 查询复杂：由于数据被拆分成很多表，查询操作可能需要复杂的表连接（JOIN）。
 - 性能问题：特别在大数据量下，多表连接可能导致性能下降。
 - 维护困难：表结构复杂，可能对数据库的维护和使用带来挑战。
- 维度建模：
 - 定义：维度建模是一种面向分析的设计技术，它围绕着事实表（存储量化数据的表）与维度表（描述事实特性的表）来组织数据仓库或数据集市。事实表通常记录了具体的事件或交易信息；而维度表则包含了这些事件发生时的相关背景信息。主要包括星型模型和雪花模型两种核心结构。
 - 星型模型：中心是事实表，周围是直接连接的多个维度表。
 - 雪花模型：在星型模型基础上，将维度表规范化，形成更复杂的结构。
 - 优势：
 - 查询性能高：结构化的设计适合快速查询，尤其在读取操作频繁的数据仓库中。
 - 易于理解：直观的层级结构，使得报表开发人员和业务分析师更易理解。
 - 分析灵活：提供良好的数据聚合能力，适用于 OLAP 操作。
 - 劣势：
 - 数据冗余：可能因去规范化而导致数据冗余，增加存储成本。
 - 一致性管理复杂：数据冗余会导致数据一致性管理变得困难。
 - 不适用于事务处理：不适合频繁的事务性操作，主要用于读取分析。

综上：没有最好的建模方法，只有最合适的建模方法，选择合适的方法做合适的事情。范式建模更加适合 OLTP 场景，维度建模更加适合 OLAP 场景。仅以阿里云 DataWorks 产品智能数据建模功能举例，它是基于维度建模方法论建设的产品，介绍阿里云在数据建模场景的建设经验以供参考。

(3.4) 示例：智能数据建模 – 一级产品功能

智能数据建模产品分为四大板块，分别是数仓规划、数据标准、维度建模和数据指标。其中数仓规划、数据标准和数据指标最终都为维度建模服务。

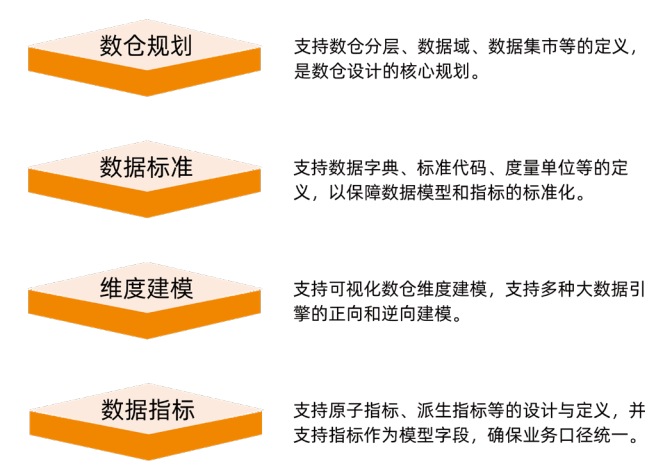


图 3 智能数据建模产品 4 大板块

(3.5) 示例：智能数据建模 – 二级产品功能

数仓规划是数仓的顶层设计，包含分层划域、维度管理、建模空间。从产品定义来讲，这些内部并不复杂。难点在于数仓怎么根据业务场景来划分。建议先用思维图画好，有了一个大概雏形之后，再录入产品。其中一个重点功能是可视化表名检查器配置，检查器用于规范目标分层中表的命名，将同一分层中表名称的命名格式统一，便于通过表名称，即可了解到该表所属的业务类型、作用功能、数据粒度等信息。同时，可以帮助减少后期的运维成本。系统默认创建的数仓分层和自定义新建的数仓分层均可以配置数仓分层检查器。对于建模同学来讲，建模效率会提升且产出的内容符合规范。

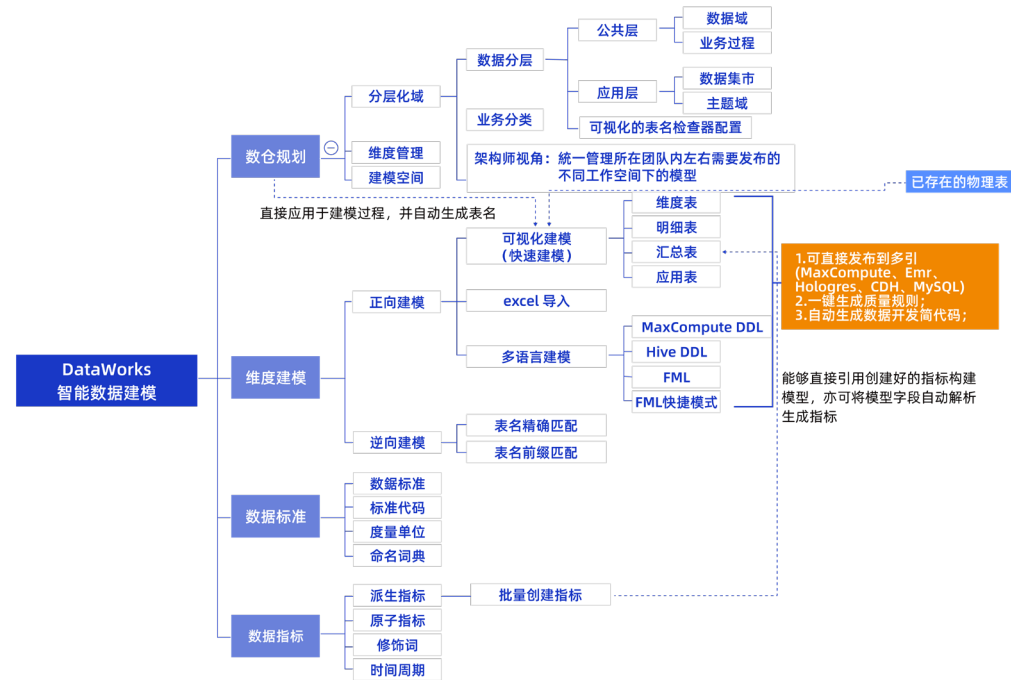
数据标准包含数据标准、标准代码、度量单位、命名词典。数据标准和标准代码设置好之后，可以和模型字段做关联，关联之后模型字段名称、值等都需要符合标准的设置。

数据指标包含派生指标、原子指标、修饰词、时间周期。这里重点需要说明批量创建指标，勾选构成派生指标的原子指标、修饰词、时间周期，就可以生成一系列派生指标，用于模型设计。指标创建好后有两个作用，一是可以把指标批量导入到模型里面，作为模型的字段存在。另一个是模型字段已经存在，需要跟指标做关联。这样在物化之后可以找到指标对应的是哪个模型。

维度建模支持正向建模和逆向建模：

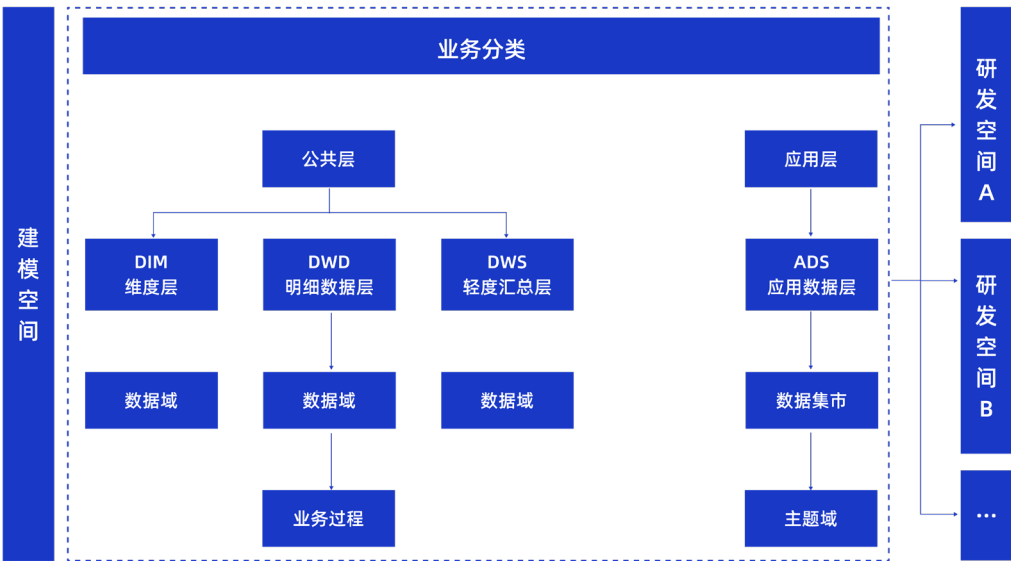
- 逆向建模解决的是已有数仓冷启动的问题，主要用于将其他建模工具生成的模型反向建模至 DataWorks 的维度建模中。例如，当已通过其他建模工具生成模型，此时，想更换为 DataWorks 的智能建模进行后续建模工作，则可以使用逆向建模功能。该功能无需再次执行建模操作，即可快速将已有模型反向建模至 DataWorks 的维度建模中，节省了大量的时间成本。
- 正向建模支持可视化建模、Excel 导入、多语言建模。可视化建模类似网页版 Excel 的方式，把模型字段信息统一管理。在这个过程中，可以复用已经存在的物理表表结构，提升建模效率。多语言建模支持 DDL、自研 FML 方式建模。

建议先用可视化建模，如果需要修改字段，可以用 DDL 或者 FML 方式做字段的修改。在建模过程中，设置某一字段为主键字段，非空字段，或者关联了数据标准里的标准代码，DataWorks 智能数据建模可以一键自动生成质量规则。



(3.6) 示例：智能数据建模 – 数仓规划

数仓规划的整体架构如下图，首先中间部分业务分类，比如阿里的业务分为天猫、淘宝、菜鸟等等。也可以根据各个数仓团队面向的业务来划分。



公共层分为三层，也就是上文讲到的 DWS、DWD、DIM。DIM 下需要区分数据域，维度表只需要分到数据域就可以。明细表需要细化到数据域和业务过程。轻度汇总层只需要指定到数据域就可以。在应用层这一部分主要是 ADS 层，在实际工作中可能不止有 ADS 层还会有 DIM 层。

2. 数据仓库中的数据血缘管理

（1）什么是数据血缘

在数据仓库中，血缘关系指的是数据在不同层次或系统之间的传递和转化路径，主要用于追踪数据的来源、流动及其变化过程。数据血缘关系帮助组织理解数据从源头到最终目的地的整个生命周期，确保数据的完整性和准确性。通过记录数据的生成、变换、加载及使用过程，数据血缘分析可以识别数据依赖性，促进数据质量管理和数据治理。它通常涉及元数据管理，其中详细记录了数据集之间的关联和依赖关系。数据血缘可分为水平血缘（如数据在同一层次中的流动）和垂直血缘（如数据在不同层次或系统间的转化）。这种清晰的追溯机制有助于数据分析、合规性检查及问题排查，提高企业决策的透明度和可靠性。

（2）为什么要建立数据血缘关系

在数据仓库海量数据处理中，用户的痛点主要集中在“找数据难”和“用数据难”两个方面。在寻找数据时，用户往往不知道该去哪里找，尤其是使用业务术语时。此外，相似表太多，用户不确定哪个是正确的，搜索结果繁多，需要逐一点击查看，且结果不准确，很多与自身业务无关。在使用数据方面，表命名奇怪、字段无注释且缺少文档，导致理解困难。表注释过于简单，缺乏有效信息，人工沟通耗费大量时间。很多表的所有者变更频繁，业务逻辑不清晰，导致数据使用困难。最后，用户面临着如何快速开放数据和构建个性化数据应用的挑战。这些痛点直接影响了数据的有效利用和业务决策效率。

而建立数据血缘关系可以帮助我们纾解大部分痛点，以下是一些关键原因：

- **数据可追溯性：**数据血缘关系提供了从数据源头到最终应用的全链路追踪，帮助用户了解数据的来源、变更和流向。这种可追溯性对于解决数据问题、验证数据准确性非常关键。
- **数据质量管理：**通过追踪数据的流动路径，数据血缘有助于识别数据在加工过程中可能出现的问题，提高数据质量，确保业务决策的可靠性。
- **合规性和审计：**在许多行业中，合规性要求对数据进行严格的审计。数据血缘关系可以帮助企业证明其数据处理过程符合法规要求，如 GDPR、HIPAA 等。
- **影响分析：**在对数据表或管道进行更改时，数据血缘关系可以帮助预测这些更改可能带来的影响，降低因数据变动导致的风险。

- **优化数据流程：**通过分析数据的流动路径，企业可以发现和消除数据处理流程中的瓶颈或冗余步骤，提高数据处理的效率和效能。
- **支持数据治理：**数据血缘是数据治理的重要组成部分，它提供了对数据资产的全面洞察，支持更好的数据管理策略。
- **提升协作效率：**在大数据团队中，清晰的数据血缘有助于不同角色之间的沟通和协作，使数据工程师、分析师和业务人员更容易理解数据的全貌和上下文。

（3）数据血缘使用场景

数据血缘应用的场景非常多，我们先举一个简单的例子，假设一个数据开发者想要修改某份数据表的设计内容，但是数据可能来自 MC、Spark 或者 Hive 产生的表或中间表，如下图：

- Table A-D 是原始表
- Table E-G 是中间表
- Table H-J 是别人产生的表
- Table G 是最终产出的表

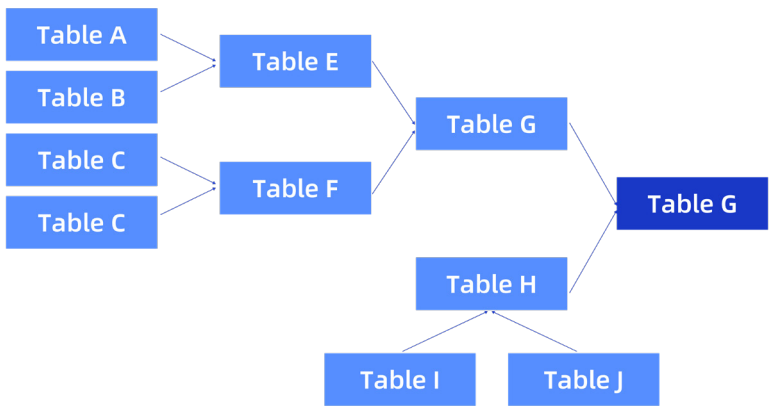


图 6 数据血缘图例

提供一个清楚的血缘关系能让开发者了解整个数据的依赖关系、变化方式和历经流程，进而完成这位开发者想要完成的任务。

目前血缘主流应用的场景有以下几种：

- 数据影响分析（修改影响评估）
- 故障排查（血缘发生非预期中断）
- 链路分析（向上溯源及向下发展）
- 重要性分析（数据价值）
- 数据质量评估（清晰的数据源及数据加工方法）
- 数据生命周期（归档、销毁参考）
- 数据安全合规（GDPR、权限管理）

（4）数据血缘管理的应用 – DataWorks 数据地图

我们继续以 DataWorks 的数据地图功能举例，介绍如何实现对数据的统一管理和血缘的跟踪。

数据地图以数据搜索为基础，提供表使用说明、数据类目、数据血缘、字段血缘等工具，帮助数据表的使用者和拥有者更好地管理数据、协作开发。

使用 DataWorks 数据地图查看表或数据服务 API 时，可在对应的详情页面查看详细的血缘信息。

（4.1）表血缘

查找某张表并进入表详情页面后，您可以单击血缘信息页签，查看表级和字段级的血缘详情。同时，您也可以进行影响分析，获取当前表的下游表列表，支持下载下游表列表为本地文件或通过邮件进行变更通知。

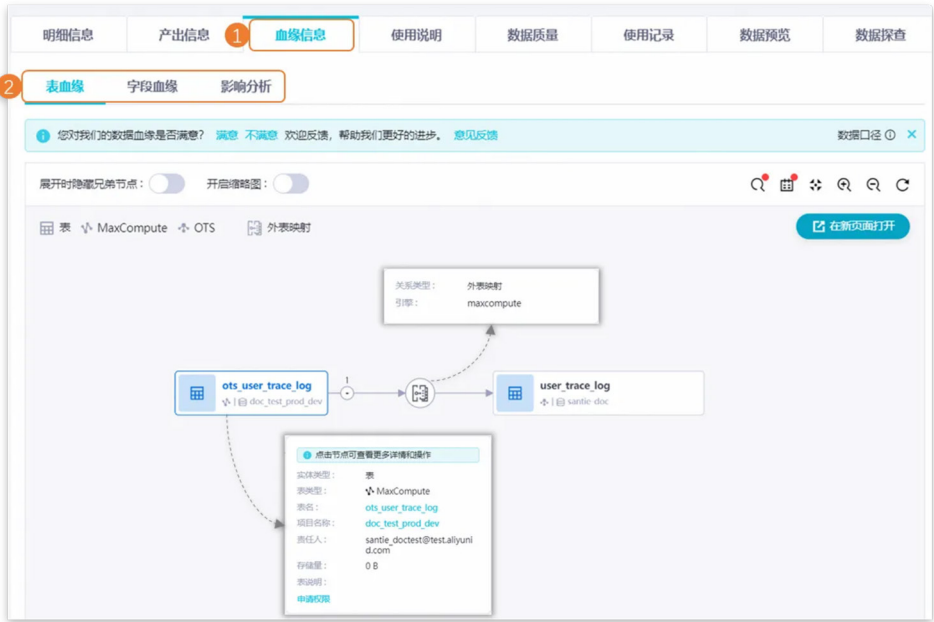


图 8 表血缘图例

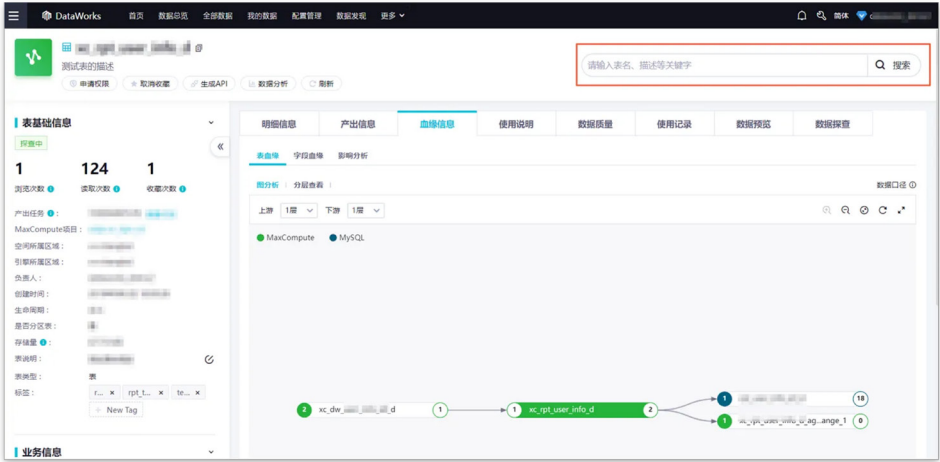


图 7 DataWorks 数据地图

表血缘各数据源血缘展示情况说明：

角色	数据集成	字段级血缘	数据开发	字段级血缘
	表级血缘		表级血缘	
MaxCompute ✔ 详情页 ✔ 详情页 – 血缘信息	✔ 离线同步 ✔ 实时同步（MySQL/ Kafka/PolarDB MySQL/LogHub 同步至 MaxCompute）	✔ 离线同步 ○ 实时同步	✔ insert into /insert overwrite table ✔ create as select from table ✔ create external table	✔ insert into /insert overwrite table ✔ create as select from table ✔ create external table
Hologres ✔ 详情页 ✔ 详情页 – 血缘信息	✔ 离线同步 ✔ 实时同步（MySQL/ Kafka/LogHub 同步至 Hologres）	✔ 离线同步 ○ 实时同步	✔ insert into /insert overwrite table ✔ create as select from table ✔ create external table	✔ insert into /insert overwrite table ✔ create as select from table ✔ create external table
E-MapReduce ✔ 详情页 ✔ 详情页 – 血缘信息	✔ 离线同步 ○ 实时同步	✔ 离线同步（OSS、Hive） ○ 实时同步	Hive、Spark（spark-submit）、Spark SQL（支持 Hudi 格式）、Shell（使用 beeline 提交的 Hive SQL） ✔ insert into /insert overwrite table ✔ create as select from table ○ create external table	Hive、Spark（spark-submit）、Spark SQL（支持 Hudi 格式）、Shell（使用 beeline 提交的 Hive SQL） ✔ insert into /insert overwrite table ✔ create as select from table ○ create external table
AnalyticDB MySQL ✔ 详情页 ✔ 详情页 – 血缘信息	✔ 离线同步（OSS、Hive） ○ 实时同步	✔ 离线同步 ○ 实时同步	✔ insert into /insert overwrite table ✔ create as select from table ○ create external table	✔ insert into /insert overwrite table ✔ create as select from table ○ create external table
AnalyticDB PostgreSQL ✔ 详情页 ✔ 详情页 – 血缘信息	✔ 离线同步 ○ 实时同步	○ 离线同步 ○ 实时同步	✔ insert into /insert overwrite table ✔ create as select from table ○ create external table	✔ insert into /insert overwrite table ✔ create as select from table ○ create external table
StarRocks ✔ 详情页 ✔ 详情页 – 血缘信息	○ 离线同步 ○ 实时同步	○ 离线同步 ○ 实时同步	○ insert into /insert overwrite table ○ create as select from table ○ create external table	○ insert into /insert overwrite table ○ create as select from table ○ create external table
ClickHouse ✔ 详情页 ○ 详情页 – 血缘信息	○ 离线同步 ○ 实时同步	○ 离线同步 ○ 实时同步	○ insert into /insert overwrite table ○ create as select from table ○ create external table	○ insert into /insert overwrite table ○ create as select from table ○ create external table

CDH/CDP ✔ 详情页 ✔ 详情页 – 血缘信息	○ 离线同步 ○ 实时同步	○ 离线同步 ○ 实时同步	Hive、Impala、Spark、Spark SQL ✔ insert into /insert overwrite table ✔ create as select from table ○ create external table	Hive、Impala、Spark、Spark SQL ✔ insert into /insert overwrite table ✔ create as select from table ○ create external table
MySQL ✔ 详情页 ○ 详情页 – 血缘信息	✔ 离线同步 ✔ 实时同步（MySQL/（MySQL 同步至 MaxCompute/Hologres）	✔ 离线同步 ○ 实时同步	○ insert into /insert overwrite table ○ create as select from table ○ create external table	○ insert into /insert overwrite table ○ create as select from table ○ create external table
PostgreSQL ✔ 详情页 ○ 详情页 – 血缘信息	✔ 离线同步 ○ 实时同步	✔ 离线同步 ○ 实时同步	○ insert into /insert overwrite table ○ create as select from table ○ create external table	○ insert into /insert overwrite table ○ create as select from table ○ create external table
SQL Server ✔ 详情页 ○ 详情页 – 血缘信息	✔ 离线同步 ○ 实时同步	✔ 离线同步 ○ 实时同步	○ insert into /insert overwrite table ○ create as select from table ○ create external table	○ insert into /insert overwrite table ○ create as select from table ○ create external table
Tablestore(OTS) ○ 详情页 ○ 详情页 – 血缘信息	✔ 离线同步 ○ 实时同步	✔ 离线同步 ○ 实时同步	○ insert into /insert overwrite table ○ create as select from table ○ create external table	○ insert into /insert overwrite table ○ create as select from table ○ create external table
Oracle ✔ 详情页 ○ 详情页 – 血缘信息	✔ 离线同步 ○ 实时同步	✔ 离线同步 ○ 实时同步	○ insert into /insert overwrite table ○ create as select from table ○ create external table	○ insert into /insert overwrite table ○ create as select from table ○ create external table
OceanBase ○ 详情页 ○ 详情页 – 血缘信息	✔ 离线同步 ○ 实时同步	✔ 离线同步 ○ 实时同步	○ insert into /insert overwrite table ○ create as select from table ○ create external table	○ insert into /insert overwrite table ○ create as select from table ○ create external table
OSS ○ 详情页 ○ 详情页 – 血缘信息	✔ 离线同步 ○ 实时同步	○ 离线同步 ○ 实时同步	○ insert into /insert overwrite table ○ create as select from table ○ create external table	○ insert into /insert overwrite table ○ create as select from table ○ create external table
PolarDB MySQL ○ 详情页 ○ 详情页 – 血缘信息	✔ 离线同步 ✔ 实时同步（PolarDB MySQL 同步至 MaxCompute	✔ 离线同步 ○ 实时同步	○ insert into /insert overwrite table ○ create as select from table ○ create external table	○ insert into /insert overwrite table ○ create as select from table ○ create external table

PolarDB PostgreSQL ○ 详情页 ○ 详情页 – 血缘信息	● 离线同步 ○ 实时同步	● 离线同步 ○ 实时同步	○ insert into /insert overwrite table ○ create as select from table ○ create external table	○ insert into /insert overwrite table ○ create as select from table ○ create external table
Kafka ○ 详情页 ○ 详情页 – 血缘信息	● 离线同步 ● 实时同步 (Kafka 同步至 MaxCompute/Hologres)	○ 离线同步 ○ 实时同步	○ insert into /insert overwrite table ○ create as select from table ○ create external table	○ insert into /insert overwrite table ○ create as select from table ○ create external table

表 1 表血缘各数据源血缘展示情况说明表

(4.2) API 血缘

查找某个数据服务 API 并进入 API 详情页面后，您可以单击**血缘信息**页签，查看 API 的血缘详情。

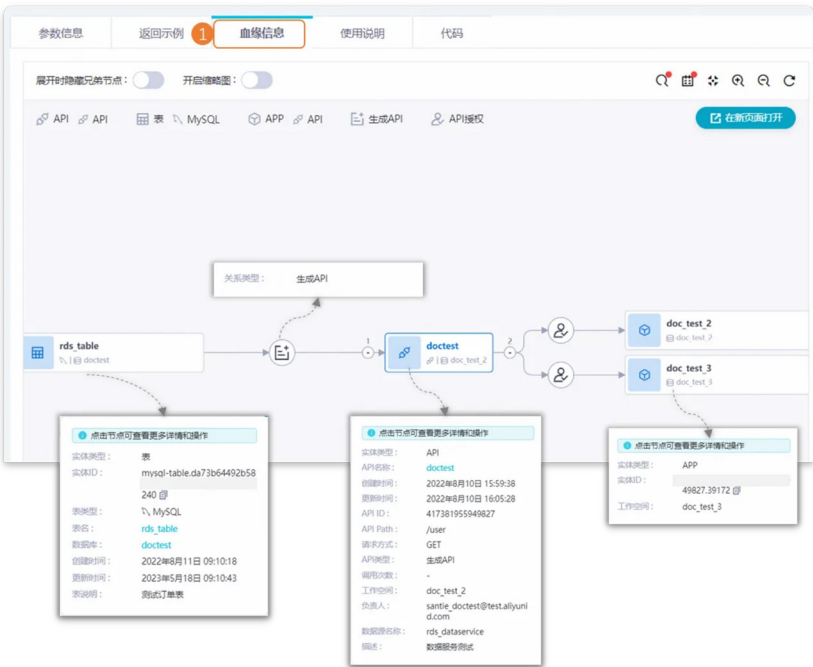


图 9 API 血缘图例

3. 小结

本文系统介绍了数据仓库中的数据建模思路和数据血缘管理。我们阐述了数据模型的定义及其重要性，强调了智能建模在解决业务痛点、提升业务价值方面的关键作用，并通过具体示例展示了其应用。同时，深入探讨了数据血缘的概念，解释了建立数据血缘关系的必要性，并展示了如何通过 DataWorks 数据地图实现对数据的统一管理和血缘的跟踪。

总之，数据建模与血缘管理是构建高效数据仓库的两大支柱，为企业数据治理提供了坚实基础，助力企业在数字化转型中实现数据驱动的创新和发展。未来，随着技术进步，这两者将继续演进，支持企业保持竞争力并实现可持续发展。

(二) 让云上的数据流动起来——数仓分层设计与数据流转

1. 数据仓库中的分层设计

(1) 数仓分层

我们一般建议将数据仓库分为三层，自下而上为：数据引入层（ODS，Operation Data Store）、数据公共层（CDM，Common Data Model）和数据应用层（ADS，Application Data Service）。

数据仓库的分层和各层级用途如下图所示。



图 1 数据仓库分层架构

• 数据引入层 ODS（Operation Data Store）：存放未经过处理的原始数据至数据仓库系统，结构上与源系统保持一致，是数据仓库的数据准备区。主要完成基础数据引入到 MaxCompute 的职责，同时记录基础数据的历史变化。

• 数据公共层 CDM（Common Data Model，又称通用数据模型层），包括 DIM 维度表、DWD 和 DWS，由 ODS 层数据加工而成。主要完成数据加工与整合，建立一致性的维度，构建可复用的面向分析和统计的明细事实表，以及汇总公共粒度的指标。

- 公共维度层（DIM）：基于维度建模理念思想，建立整个企业的一致性维度。降低数据计算口径和算法不统一风险。公共维度层的表通常也被称为逻辑维度表，维度和维度逻辑表通常一一对应。
- 公共汇总粒度事实层（DWS）：以分析的主题对象作为建模驱动，基于上层的应用和产品的指标需求，构建公共粒度的汇总指标事实表，以宽表化手段物理化模型。构建命名规范、口径一致的统计指标，为上层提供公共指标，建立汇总宽表、明细事实表。公共汇总粒度事实层的表通常也被称为汇总逻辑表，用于存放派生指标数据。
- 明细粒度事实层（DWD）：以业务过程作为建模驱动，基于每个具体的业务过程特点，构建最细粒度的明细层事实表。可以结合企业的数据使用特点，将明细事实表的某些重要维度属性字段做适当冗余，即宽表化处理。明细粒度事实层的表通常也被称为逻辑事实表。

• 数据应用层 ADS（Application Data Service）：存放数据产品个性化的统计指标数据。根据 CDM 与 ODS 层加工生成。

其中 ODS，又称为贴源层。ODS 主要是用户存储业务系统同步的业务数据。一般情况下，不会对 ODS 层的数据做过多加工，以便于后续在 ADS 和 CDM 数据出错时溯源。换句话说，ODS 不是数仓同学设计出来的，是对业务系统数据的直接同步。

数仓建设最重要的公共层 CDM 层，CDM 层需要对业务进行高度抽象，需要具备通用性、易用性、复用性，因此，公共层的建设对数仓同学的要求是非常高，既精通建模方法，同时也对业务情况了如指掌。CDM 层再进行细分，一般会分为 DIM 层 – 维度表，DWD 层 – 明细数据表，DWS 层 – 轻度汇总层。

数仓建设最难管但管好了效果非常明显的是应用层 ADS 层，ADS 层主要面向业务进行模型设计。因此，大家一定要先了解清楚模型的主要应用场景，是普通的报表分析，还是数据产品的调用等等，不同的应用场景，模型设计需要考虑的因素也不一样。如果规范化 ADS 层，需要建设的表会减少，通过统一逻辑去查询，会使计算和存储成本降低。

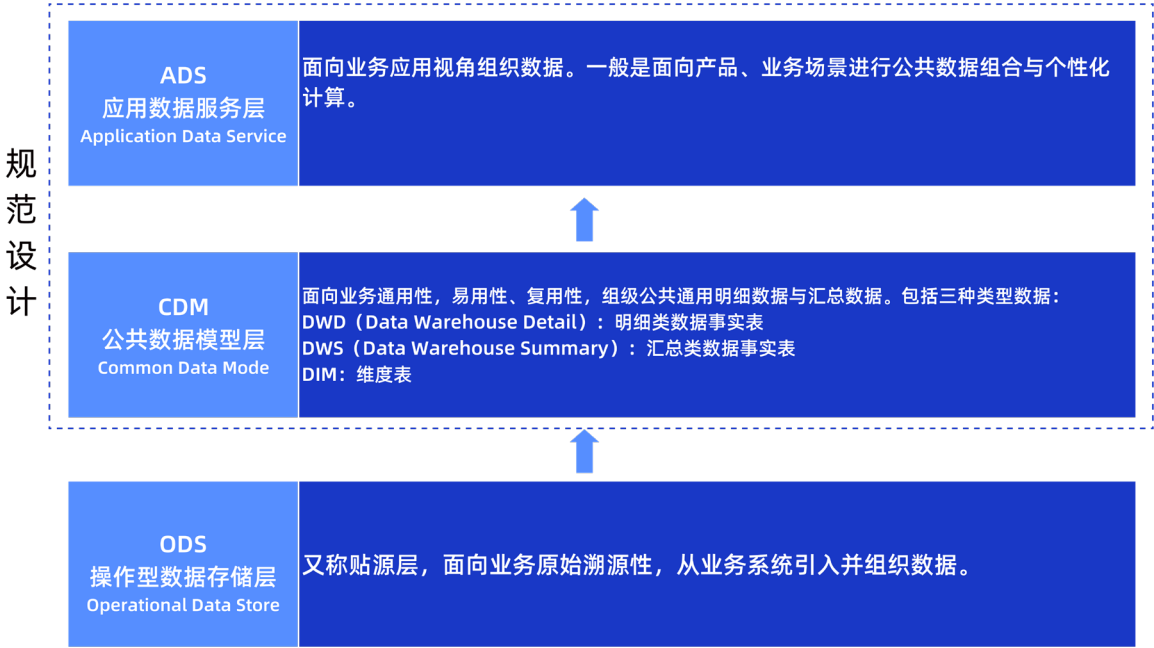


图 2 数据仓库规范设计

（2）通过数仓规划进行数据分层

数仓规划的整体架构如下图所示。架构的中间部分是业务分类，例如阿里的业务分为天猫、淘宝、菜鸟等。也可以根据各个数仓团队面向的业务来划分。

公共层分为三层，也就是数据公共层讲到的 DWS、DWD、DIM。DIM 下需要区分数据域，维度表只需要分到数据域就可以。明细表需要细化到数据域和业务过程。轻度汇总层只需要指定到数据域就可以。在应用层这一部分主要是 ADS 层，在实际工作中可能不止有 ADS 层还会有 DIM 层。

产品侧是支持大家灵活设置，如果有需要可以自行创建。ADS 层需要指定到具体的数据集市和主题域。这是模型在分层化域时需要考虑到的一整套体系。如果数仓团队负责多个业务，多个工作空间，需要复用同一套数仓规范，可以使用建模空间功能。

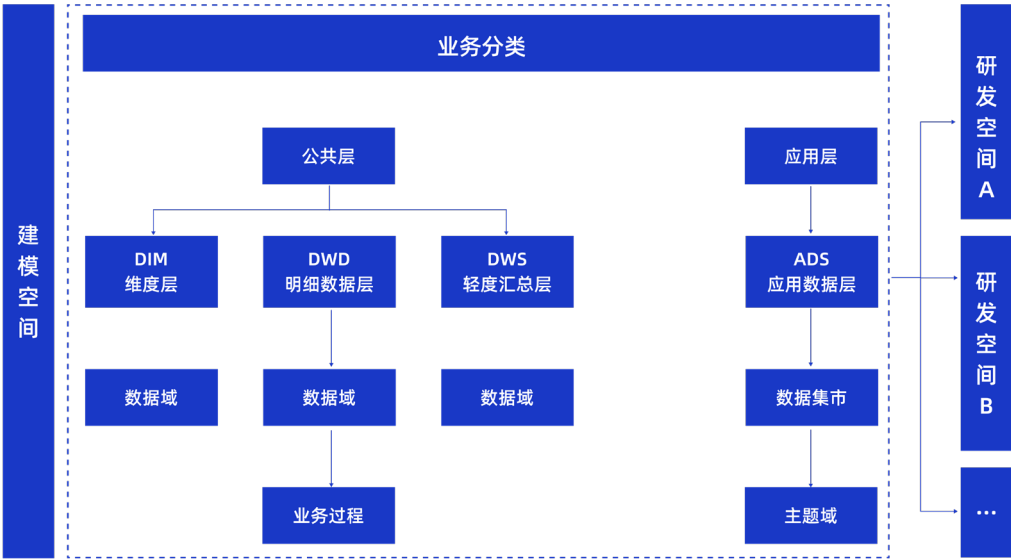


图 3 数仓规划整体架构

数仓规划架构中的名词释义：

数据域：数据域是指一个或多个业务过程或者维度的集合，如交易域，日志域。数据域是一个较高层次的数据归类标准，是对企业业务过程进行抽象、提炼、组合的集合，是企业业务人员在使用数据时第一个分组入口，可以帮助企业业务人员快速的从海量的数据中快速圈定到自己的业务数据。

业务过程：业务过程指企业的业务活动事件，如下单，支付。业务过程是对业务活动流程的描述，例如在电商领域，加购、下单、支付等都可以是一个业务过程。进行业务效果分析时，业务过程有非常典型的应用，例如常用漏斗分析，即将购买商品的业务活动分解为浏览商品、加入购物车、下订单、付款、确认收货等业务过程，统计每个业务过程的“订单数”，将可以针对“订单数”这一指标做漏斗分析。

数据集市：面向某个应用场景或者产品的数据组织，一般会依赖数据公共层。数据集市是对某个业务分类制定细化的业务主题，并通过主题域，基于不同分析视角对目标集市中的数据进行主题划分，最终面向业务应用统计分析数据。例如，运营平台数据集市。

主题域：主题域用于将数据集按照分析视角进行划分，通常是联系较为紧密的数据主题的集合。您可以根据业务的关注点，将这些数据主题划分至不同的主题域。例如，电商行业通常分为交易域、会员域、商品域等。

建模空间：当您所需要管理多个工作空间且需要复用一套数仓规划时，面对跨多个工作空间的复杂数据体系，可以通过设计空间来共享一套数据建模工具，针对整个数据体系进行统一地数仓规划、维度建模及指标定义等工作。

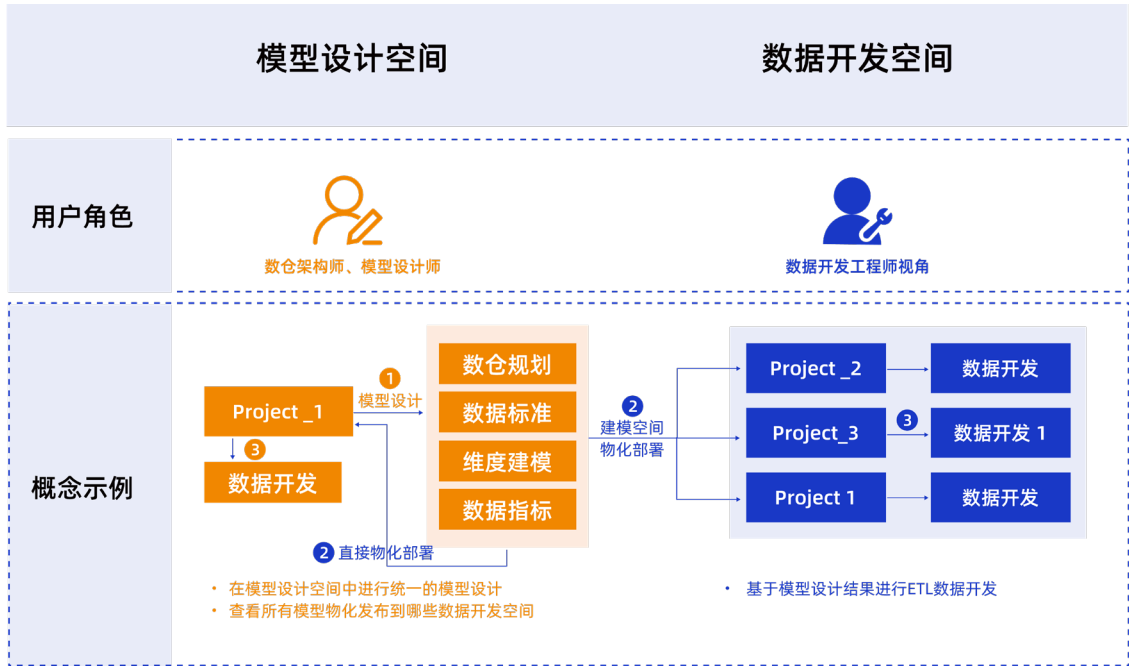


图 4 数仓分层名词及关系映射

（3）数仓规划的实现 –DataWorks 数仓规划

使用 DataWorks 进行数据建模时，数仓架构师或者模型小组成员可以在数仓规划页面对数据分层、业务分类、数据域、业务过程、数据集市、主题域进行设计。完成设计后，模型设计师在建模过程中可以依赖数仓规划中的数据分层、业务分类、数据域、业务过程等对所建模型进行分层化域管理。

（3.1）创建数据分层

DataWorks 的数仓分层功能，默认为您创建了数据引入层 ODS（Operational Data Store）、公共维度层 DIM（Dimension）、明细数据层 DWD（Data Warehouse

Detail）、汇总数据层 DWS（Data Warehouse Summary）及应用数据层 ADS（Application Data Service）等五层，各个分层的功能介绍如下：

- 数据引入层 ODS（Operational Data Store）

ODS 层用于接收并处理需要存储至数据仓库系统的原始数据，其数据表的结构与原始数据所在的数据系统中的表结构一致，是数据仓库的数据准备区。ODS 层对原始数据的操作具体如下：

- 将原始的结构化数据增量或全量同步至数据仓库中。
- 将原始的非结构化数据（例如，日志信息）进行结构化处理，并存储 MaxCompute。
- 根据实际业务需求，记录原始数据的历史变化或对原始数据进行简单的清洗。

ODS 层的数据表，命名必须以 ods 开头，并且生命周期为 366 天。

- 明细数据层 DWD（Data Warehouse Detail）

DWD 层通过企业的业务活动事件构建数据模型。基于具体业务事件的特点，构建最细粒度的明细数据表。您可以结合企业的数据使用特点，将明细数据表的某些重要维度属性字段适当冗余，即宽表化处理。同时，也可以减少明细数据表及维度表的关联，提高明细表的易用性。

- 汇总数据层 DWS（Data Warehouse Summary）

DWS 层通过分析的主题对象构建数据模型。基于上层的应用和产品的指标需求，构建公共粒度的汇总指标事实表。

例如，从 ODS 层中对用户的行为做一个初步的归类汇总，抽象出来一些通用的维度，假设维度为时间、IP、ID，并根据这些维度统计出相关数据，比如用户每个时间段在不同登录 IP 购买的商品数。则在 DWS 层可以进一步添加一层轻度的汇总，可以让计算更加的高效。例如在此基础上计算仅 7 天、30 天、90 天的行为会节省很多时间。

- 应用数据层 ADS（Application Data Service）

ADS 层用于存放数据产品个性化的统计指标数据，输出各种报表。例如，某电商企业，在 6 月 9 日至 6 月 19 日，杭州地域出售的各大球类商品的数量及排行情况。

公共维度层 DIM (Dimension)

DIM 层使用维度构建数据模型。可基于实际业务，存放逻辑模型的维度表；或存放概念模型的维度定义，通过定义维度，确定维度主键，添加维度属性，关联不同维度等操作，构建整个企业的一致性数据分析维表，帮助您降低数据计算口径和算法不统一的风险。



图 5 DataWorks 创建数据分层图例

(3.2) 创建业务分类

如您所在的集团或公司业务复杂且较为庞大时，可创建业务分类，对数据进行业务区分，便于对业务数据进行更加方便的管理。您可以在 DataWorks 上根据业务需求创建业务分类。



图 6 DataWorks 创建业务分类图例

(3.3) 创建数据域

数据域是联系较为紧密的数据主题的集合，通常是根据业务类别、数据来源、数据用途等多个维度，对企业的业务数据进行的区域划分，将同类型数据存放在一起，便于您快速查找需要的内容。不同使用目的数据，分类标准不同。例如，电商行业通常分为交易域、会员域、商品域等。DataWorks 的数据建模默认为您创建了名为默认的数据域，如果业务过程没有指定所属数据域，则创建的业务过程会被默认放至该数据域中。您可以根据需要建模的业务类型，创建符合企业实际应用的数据域。

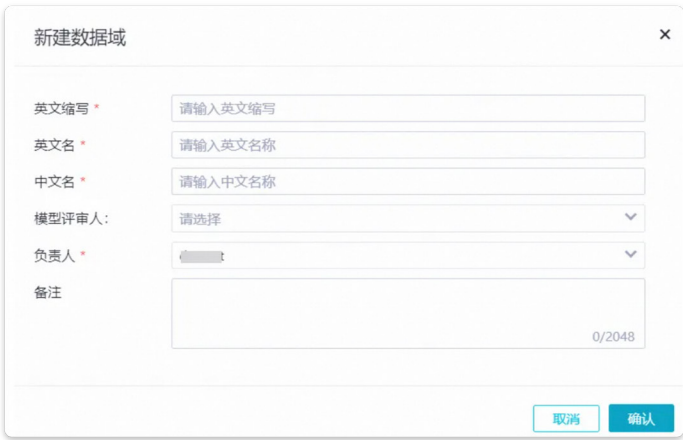


图 7 DataWorks 创建数据域图例

(3.4) 创建业务过程

业务过程是对业务活动流程的描述，例如在电商领域，加购、下单、支付等都可以是一个业务过程。进行业务效果分析时，业务过程有非常典型的应用，例如常用漏斗分析，即将购买商品的业务活动分解为浏览商品、加入购物车、下订单、付款、确认收货等业务过程，统计每个业务过程的“订单数”，即可以针对“订单数”这一指标做漏斗分析。

DataWorks 的数据建模已为您创建好一个默认的业务过程（英文名为 default，中文名称为默认），如果您的业务比较简单，无需明细多个业务过程，那您无需再创建新的业务过程，在后续建模配置过程中，直接使用此默认业务过程即可。如果您的业务比较复杂，则您可以根据实际业务需求，创建符合企业实际应用的业务过程。

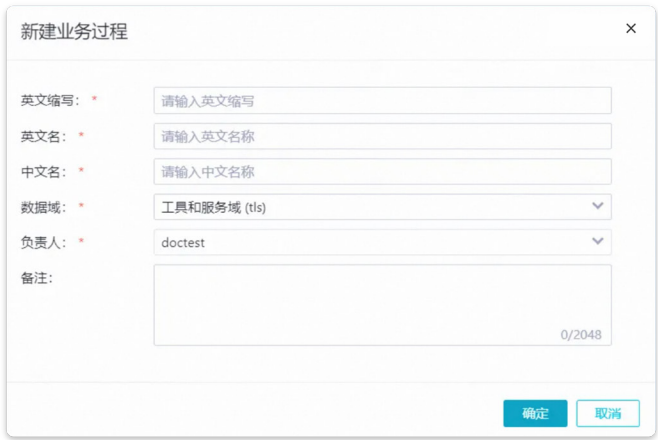


图 8 DataWorks 创建业务过程图例

(3.5) 创建数据集市

数据集市是对某个业务分类制定细化的业务主题，并通过主题域，基于不同分析视角对目标集市中的数据进行主题划分，最终面向业务应用统计分析数据。例如，运营平台数据集市。您可以在 DataWorks 上根据业务需求创建数据集市。



图 9 DataWorks 创建数据集市图例

(3.6) 创建主题域

主题域用于将数据集市按照分析视角进行划分，通常是联系较为紧密的数据主题的集合。您可以根据业务的关注点，将这些数据主题划分至不同的主题域。例如，电商行业通常分为交易域、会员域、商品域等。您可以在 DataWorks 上根据业务需求创建主题域。



图 10 DataWorks 创建主题域图例

2. 大数据中的数据流转

(1) 数仓分层中的数据流转原理

前面我们已经通过数仓分层了解了数据仓库中的业务架构以及分层逻辑，而不同层级之间的数据流向如下图所示。其中，ODS 层到 DIM 层的 ETL（萃取（Extract）、转置（Transform）及加载（Load））处理是在数据仓库中进行的，处理完成后会同步到所有存储系统。ODS 层和 DWD 层会放在数据中间件中，供下游订阅使用。而 DWS 层和 ADS 层的数据通常会落地到在线存储系统中，下游通过接口调用的形式使用。

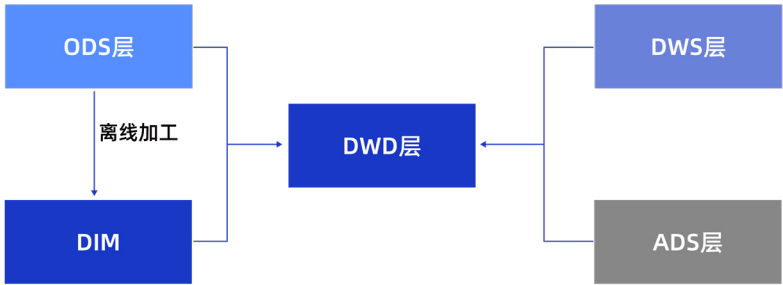


图 11 数据流转原理

(2) 数据流转的实现方案

在数据仓库中，ODS 层的数据来自不同的业务系统，数据源类型多样，存在结构与非结构数据，我们需要将我们需要的业务数据通过萃取 (Extract)、转置 (Transform) 及加载 (Load) 等方法将数据整理到 ODS 层，而如何对这些数据源进行处理，阿里云目前提供多种工具进行数据加工、数据流转来实现数仓分层之间实时数据的高效流动，解决实时数仓分层问题。

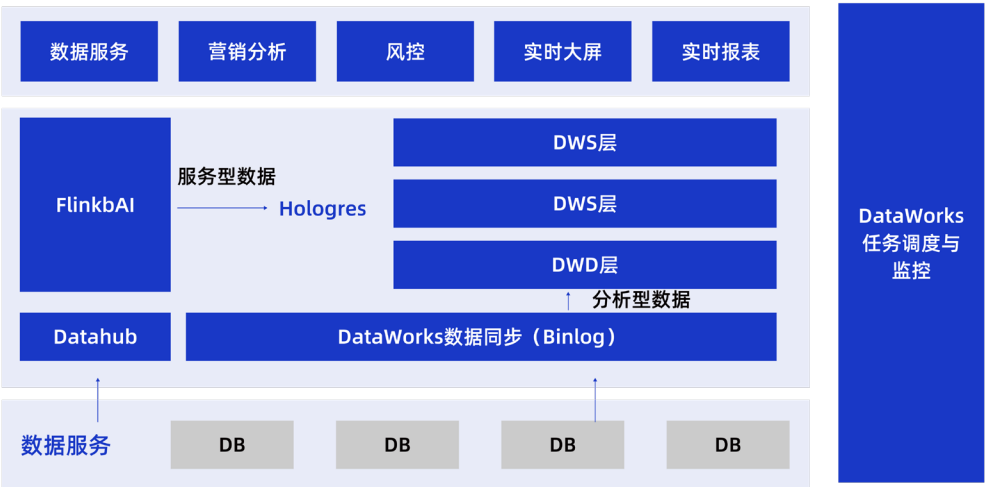


图 12 数据流转实现

(2.1) 通过 DataWorks 数据集成进行数据流转

DataWorks 数据集成是一个稳定高效、弹性伸缩的数据同步平台，致力于提供在复杂网络环境下、丰富的异构数据源之间高速稳定的数据移动及同步能力。

DataWorks 数据集成支持且仅支持结构化（例如 RDS、DRDS 等）、半结构化、无结构化（OSS、TXT 等，要求具体同步数据必须抽象为结构化数据）的数据的同步。

离线（批量）同步

DataWorks 数据集成主要用于离线（批量）数据同步。离线（批量）的数据通道通过定义数据来源和去向的数据源和数据集，提供一套抽象化的数据抽取插件（Reader）、数据写入插件（Writer），并基于此框架设计一套简化版的中间数据传输格式，从而实现任意结构化、半结构化数据源之间数据传输。

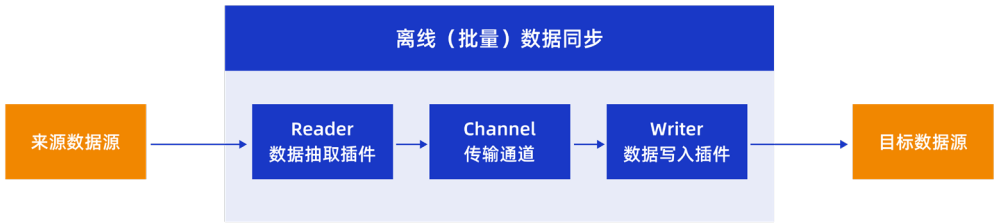


图 13 DataWorks 数据集成实现数据流转图例

实时同步

DataWorks 数据集成提供的实时数据同步功能，使用单表或整库同步方式，将源端数据源中部分或全部的数据变化实时同步至目标数据源。数据集成的实时同步包括实时读取、转换和写入三种基础插件，各插件之间通过内部定义的中间数据格式进行交互。一个实时同步任务支持多个转换插件进行数据清洗，并支持写入插件实现输出功能。

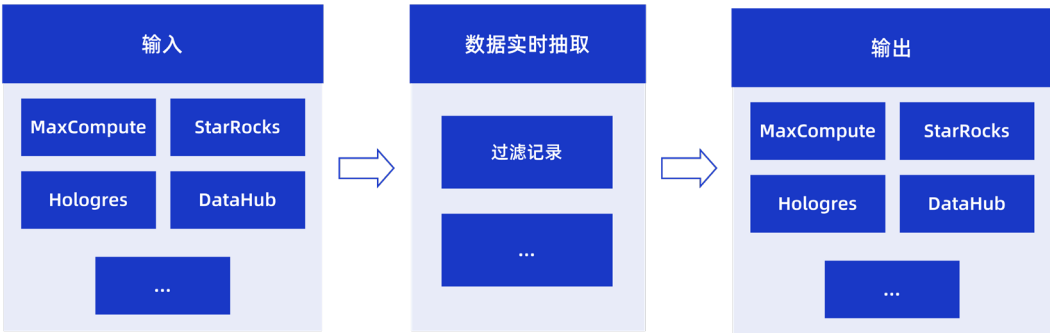


图 14 DataWorks 实时同步图例

全增量同步任务

实际业务场景下，数据同步通常不能通过一个或多个简单离线同步或者实时同步任务完成，而是由多个离线同步、实时同步和数据处理等任务组合完成，这就会导致数据同步场景下的配置复杂度非常高。

为了解决上述问题，DataWorks 提出了面向业务场景的同步任务配置化方案，支持不同数据源的全增量同步任务。

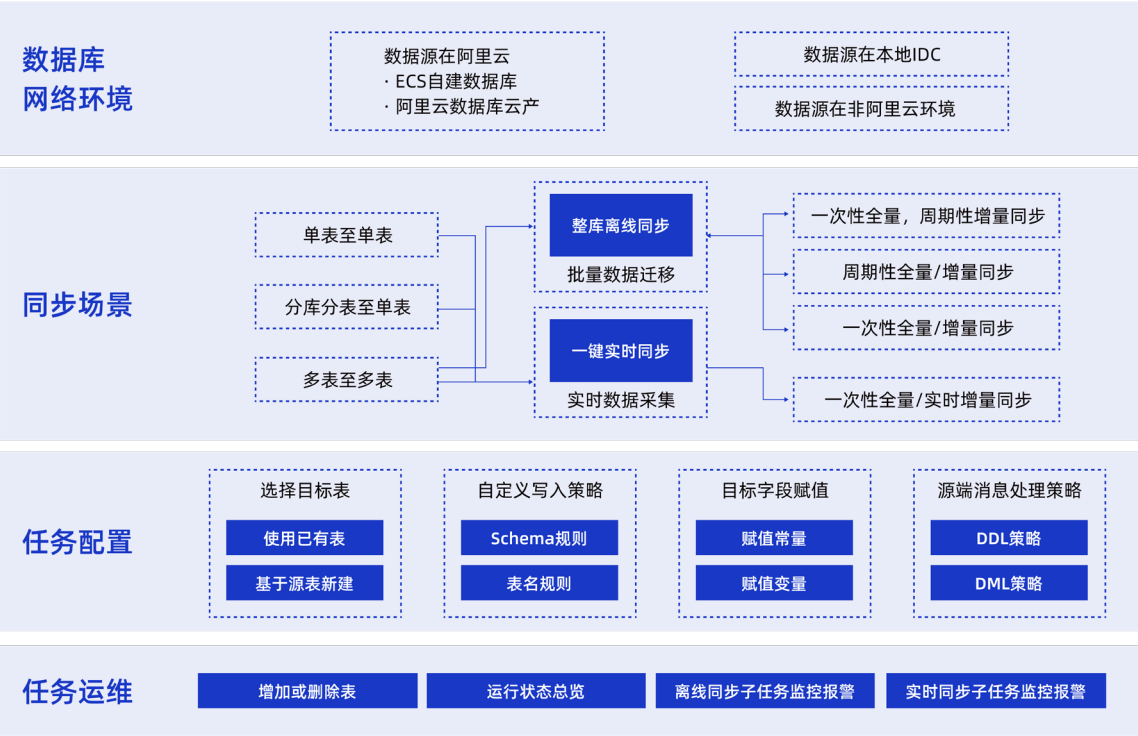


图 15 DataWorks 全增量同步任务图例

（2.2）通过 Flink 进行数据流转

作为流式计算引擎，Flink 可以广泛应用于实时数据处理领域，例如 ECS 在线服务日志，IoT 场景下传感器数据等。同时 Flink 还能订阅云上数据库 RDS、PolarDB 等关系型数据库中 Binlog 的更新，并利用 DataHub、SLS、Kafka 等产品将实时数据收集到实时计算产品中进行分析和处理。最终，分析结果可写入不同的数据服务中，例如 MaxCompute、MaxCompute-Hologres 交互式分析、人工智能平台 PAI、Elasticsearch 等，以提高数据利用率，满足业务需求。



图 16 Flink 数据流转场景

利用 Flink 的实时计算能力，数据可以在数仓分层之间实时流动。

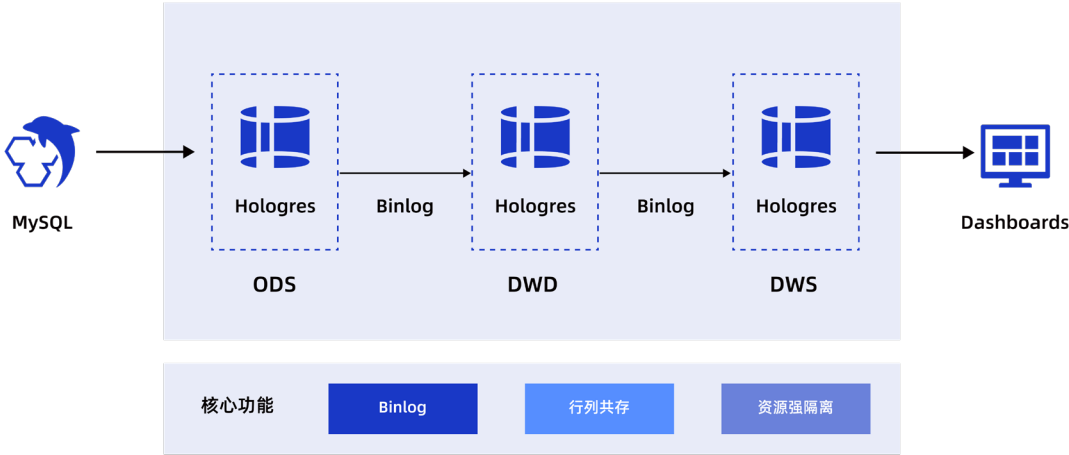


图 17 Flink 实现数仓分层实时流动图例

（2.3）通过 DataHub 进行数据流转

阿里云流数据处理平台数据总线 DataHub 是流式数据（Streaming Data）的处理平台，提供对流式数据的发布（Publish），订阅（Subscribe）和分发功能，让您可以轻松构建基于流式数据的分析和应用。数据总线 DataHub 服务可以对各种移动设备，应用软件，网站服务，传

传感器等产生的大量流式数据进行持续不断的采集，存储和处理。用户可以编写应用程序或者使用流计算引擎来处理写入到数据总线 DataHub 的流式数据比如实时 Web 访问日志、应用日志、各种事件等，并产出各种实时的数据处理结果比如实时图表、报警信息、实时统计等。

用户可以编写应用订阅数据总线 DataHub 中的数据，并进行实时的加工，把加工后的结果输出。用户可以把应用计算产生的结果输出到数据总线 DataHub 中，并使用另外一个应用来处理上一个应用生成的流式数据，来构建数据处理流程的 DAG。

通过数据总线（DataHub）搭建原始数据层（ODS），实时明细层（DWD）和实时汇总层（DWS），打造实时数据仓库。



图 18 DataHub 数据流转图例

3. 小结

本文系统介绍了数据仓库的分层设计和大数据中的数据流转。我们阐述了数仓分层的概念及其规划实现，包括通过 DataWorks 创建数据分层、业务分类、数据域、业务过程、数据集市和主题域。同时，深入分析了数仓分层中的数据流转原理，并探讨了通过 DataWorks 数据集成、Flink 和 DataHub 进行数据流转的具体方案。

总之，合理的分层设计和高效的数据流转是构建高效数据仓库的核心。这不仅提高了数据管理的规范性和灵活性，还确保了数据的及时性和一致性，为企业在数据驱动决策中提供坚实支持。

(三) 大数据保质也保量 —— 大数据的质量管理与治理

1. 大数据中的数据质量管理

(1) 大数据的数据管理痛点

理想中，一个企业的数据管理应该是从业务系统采集的数据，能供给所有的部门进行使用，各部门共用一份大数据，这样的数据最好管理，不容易出错，且不存在冗余数据，管理效率最高。

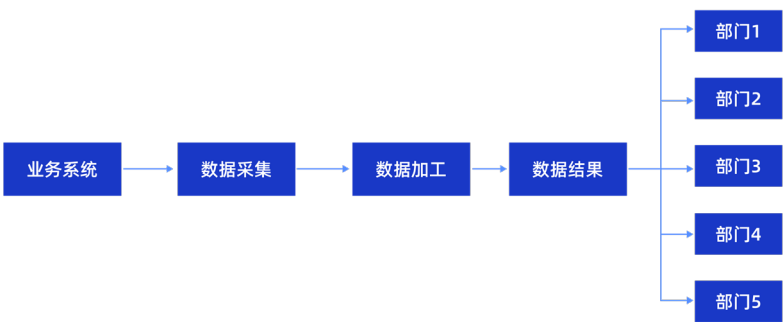


图 1 企业理想数据管理流程

而实际情况中，大部分公司存在部门之间使用的数据来自不同的大数据平台，各个部门只关注自己部门的数据，站在企业的角度看，数据管理杂乱无章，且数据存在大量的冗余，部门间的数据孤岛严重，管理起来比较繁琐。

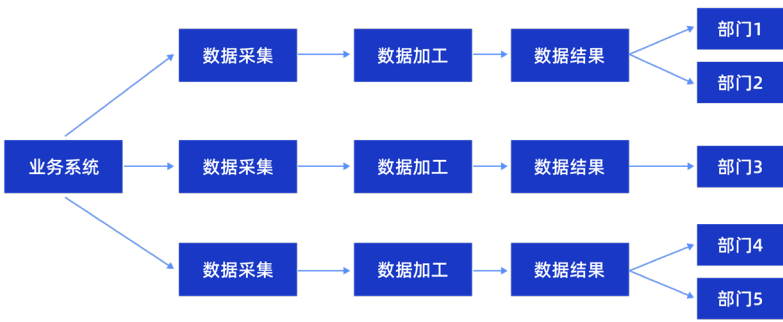


图 2 多数企业实际数据管理流程

通过对企业数据管理的分析中，我们可以发现，当前企业大数据的管理存在以下痛点：

数据稳定性不足

存在大量的大数据平台进行数据管理，每个数据平台的逻辑都不一样，无法保障各个数据平台的数据质量，是否及时，是否准确。

数据应用效率低

数据越来越大，表数量越来越多，存在大量的冗余，缺少数据管理标准与规范，当需要跨部门统计数据时非常麻烦，需要大量的沟通成本去解决数据孤岛问题。

数据成本高

数据存在大量冗余，平台成本与数据空间成本是单个大数据平台的几何倍。

数据安全风险高

数据平台数量多，使用的人也更多，面对的数据安全风险也比较高。

通过以上痛点分析，我们可以得知，在各行各业日趋强烈的大数据建设、管理及应用的诉求，数据管理的多样性及复杂性随之增加。因业务系统的原始数据的规范性无法得以保障，所以我们要定义数据管理的质量，以满足资产质量的时效性、准确性、完整性、一致性、有效性，助力企业真正能够依赖数据进行经营决策。

（2）数据质量的定义

在大数据管理中，数据质量是指存储在数据仓库中的数据在准确性、完整性、一致性、及时性等方面的优良程度。高质量的数据能够确保分析结果的可信性和决策的有效性。为了维护数据质量，通常需要实施严格的数据治理和管理流程，包括数据清洗、标准化、验证以及定期的质量审计。这确保了用户在使用数据进行分析和报告时，能够得到准确和有用的信息，从而支持企业的战略决策和运营效率。

数据质量衡量标准：**完整性、一致性、准确性、及时性。**

完整性：完整性指的是数据信息是否存在缺失的状况，数据缺失的情况可能是整个数据记录缺失，也可能是数据中某个字段信息的记录缺失。

一致性：一致性是指数据是否遵循了统一的规范，数据集是否保持了统一的格式。数据质量的一致性主要体现在数据记录的规范和数据是否符合逻辑。

准确性：准确性是指数据记录的信息是否存在异常或错误，符合业务预期。

及时性：及时性是数据的产出时间是否及时、准时。

（3）如何进行数据质量管理

我们已经了解了数据质量管理的重要性，那我们如何将数据质量管理进行落地呢？具体可以分为以下三步：

a. 定义数据质量标准

明确数据质量的标准和指标，包括准确性、完整性、一致性、及时性和唯一性等，以便衡量和评估数据质量。数据质量标准是对常见大数据存储进行质量校验。从完整性、准确性、有效性、一致性、唯一性和及时性等多个维度，配置质量监控规则。并可以将质量监控规则与调度节点进行关联，当任务运行完成后便会触发质量规则校验，帮助企业第一时间感知问题数据，按需设置规则的强弱来控制任务是否失败退出，从而避免脏数据影响扩大，有效降低数据恢复处理的时间成本和费用成本。

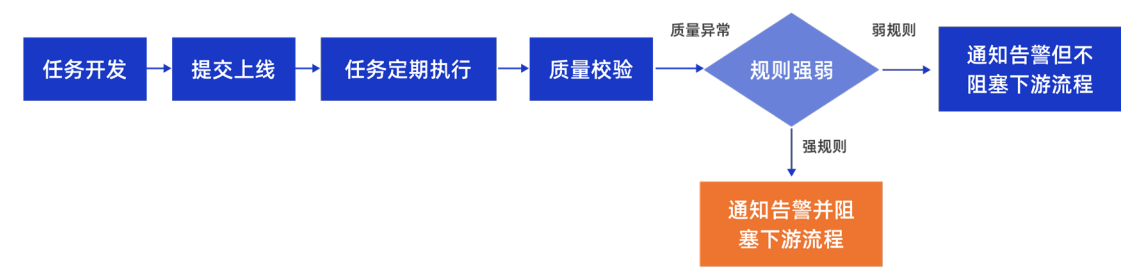


图 3 数据质量标准规则示例

b. 数据质量治理

数据质量治理是针对校验过程中的出错情况及针对该质量校验问题发起整改、忽略、通知等治理操作，从而实现资产质量从制定计划到实现整改的 PDCA 完整流程，有效提升数据质量。

c. 数据质量监控

针对数据质量进行全链路监控，监控项包括质量规则列表、配置质量规则、查看校验记录、查看质量报告等。从数据质量监控我们可以评估大数据管理中经过数据质量治理后的数据质量情况，是否满足数据质量的衡量标准。

2. 大数据的数据治理

随着数据建设的深入，愈发意识到数据治理是数据资产化建设、加速数据价值释放不可或缺的关键工作。构建“质量可靠、安全稳定、生产经济、消费便捷”的数据资产体系目标，是所有企业的共识，我们需要围绕这个目标来开展数据治理工作。

(1) 企业数据治理的痛点



图 4 企业数据治理常见痛点

数据治理的工作在很多企业已经广泛开展或正准备开展，数据治理实施，有以下四个方面的典型痛点：

• 数据治理入手难

开展数据治理工作，通常会参考 DAMA 或者 DCMM 理论体系，可以发现数据治理涵盖内容极为广泛。从哪里优先入手，以什么样的路径来推进，这是企业进行数据治理工作首先要回答的问题。数据治理的目标和执行路径不清晰，是第一个典型痛点。

• 数据治理落地难

无论是企业内部自发地开展、还是请专业的咨询机构，构建出数据治理咨询方案、产出一系列的规范和管理办法后，往往只能停留于纸面，没有恰当的治理平台工具来支撑落地，这是会面临的第二个典型挑战。

• 数据治理成效的可视化不足

如何客观地评估治理、将治理成效量化、可视化。当这个工作没有做好时，治理的推进难度会显著加大。

• 数据治理工作不可持续

数据治理的工作容易陷入“运动式治理”，通过集中的突击、在一段时间内看到一定的效果。但如果不融入到日常的数据开发生产链路中去，这项工作就不持续，不能长久地、从根本性地解决治理的问题。

(2) 企业数据治理体系

构建企业的数据治理体系，从计算、存储、质量、安全、模型和成本等多个维度进行治理切入，采用统一的方法和策略，构建量化的评估模型，并使用统一的治理平台工具来承接落地，这样能很高效的帮助企业完成数据治理目标。

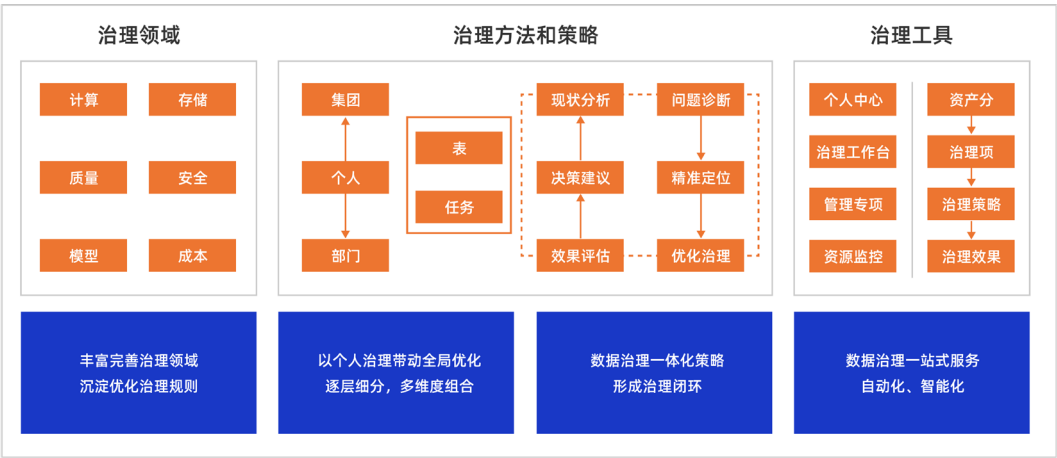


图 5 企业数据治理体系大图

数据治理的工作在很多企业已经广泛开展或正准备开展，数据治理实施，有以下四个方面的典型痛点：

企业数据治理体系有几个关键点：

- 首先，明确治理的核心对象是大数据的数据对象。数据治理是治理客观的对象，不是治理人。但治理实施的一个关键前提，是对大数据的数据对象的确定归属，梳理并定义清楚对象的具体负责人，来确保治理问题有着落、有跟进。通过具体到人，进而汇聚到部门、到企业整体。
- 其次，数据治理采取的实施路径是“现状分析 -> 问题定位 -> 优化治理 -> 效果评估”，构建一个闭环流程。
- 最后，数据治理的核心，要落在量化上：将问题量化、将成效量化。并基于局部的明细给出全局的决策建议，比如为全企业的资源调配、各部门的预算制定、成本优化目标设定等，提供参考。并且，这些量化的评估和治理问题的发现、修复，都会通过一个统一的平台工具来承接。

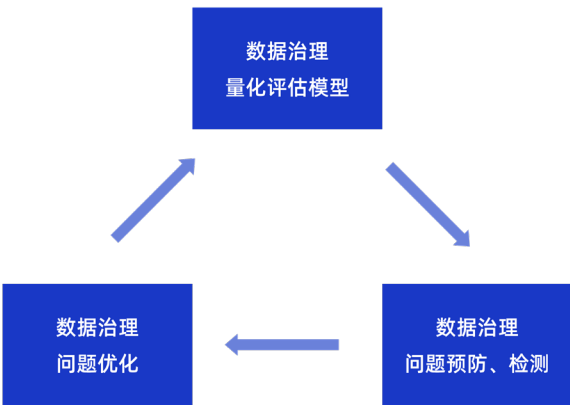


图 6 企业数据治理关键点

数据治理体系基于治理问题驱动，构建了一个治理量化评估 - 问题发现 / 预防 - 治理问题的优化处理的闭环提升机制。基于事前预防、事后整治相结合的方式，提供了几大核心产品功能。这里要说明下，这个“事前”、“事后”的“事”，定义为大数据当中，数据加工的正式数据生产这一个环节。

- 数据治理通过检查项的功能，可以做到在任务的提交、发布等关键环节，对于 SQL 代码的质量、性能消耗等进行自动扫描和检查卡点，来预防新问题的引入。这个有点类似于编译和优化的提示。

- 当前面临的一个现实问题是大数据的建设可能已经进行了较长时间，会存在许多存量的问题需要优化治理。数据治理的治理项功能，就是为此而设计，可以发现系统中存量需要优化的问题，并给出对应的解决办法。与检查项一样，这也是一种全自动的方式。

- 数据治理最具特色的，或者说是企业内部数据治理体系实践的特色，是这套量化评估机制。基于治理“健康分”的概念，从“计算”、“存储”、“质量”、“安全”和“研发”五个基础维度进行量化评估，进而给出整体的治理健康度评估。便于治理实施前了解现状、同时也会数据治理实施后的成效提供客观评估。

- 此外，数据治理在成本优化治理方面，也提供了资源使用分析等一系列的产品能力，可以清晰了解单个任务、单张表粒度的资源消耗、费用预估以及资源异动情况，帮助公司有针对性地进行计算和存储的优化治理，来达成降本增效的目标。

(3) 数据治理的业务体系架构



图 7 数据治理业务体系架构图

数据治理的业务体系架构，大致可以分为数据层、治理应用层和管理运营层。

• **数据层：**是整个产品模块的关键基础，数据治理中心汇聚了任务、表、模型、数据服务 API 等一系列对象的元数据信息，并构建用以分析洞察的元数仓，来支撑上层的治理应用。

• **治理应用层：**数据治理中心的主体功能所在。基于内置的方案模板，提供用于事前问题自动预防、事后存量问题的自动发现，以及对应的优化处理指南等系列功能。资源使用分析是面向成本治理构建的产品能力，包含资源明细和异动分析等，以及规划中的资源智能优化建议。对象 360 用于汇聚展示对象的全景信息，尤其是需要治理优化的问题，并全生命周期追踪对象的事件变化情况。

标签体系作为额外的支撑体系，便于有效的对任务进行类型打标区分，然后进行集中式的治理。场景化治理是基于 PDCA 理念构建，来帮助按照业务需要，灵活圈选需要治理的对象、评估现状、设定治理目标，并有效监督治理实施进度，最终来达成治理落地。

• **管理运营层：**数据治理核心服务于数据治理管理员以及数据治理具体参与的一线同学两类用户群体。在管理运营层，提供了治理评估报告、治理健康分、治理排行榜和治理运营推送等一系列功能。

(4) 数据治理的实现路径

数据治理的实现路径，大致可以分为现状评估、治理实施和治理运营 & 成效查看三个环节。

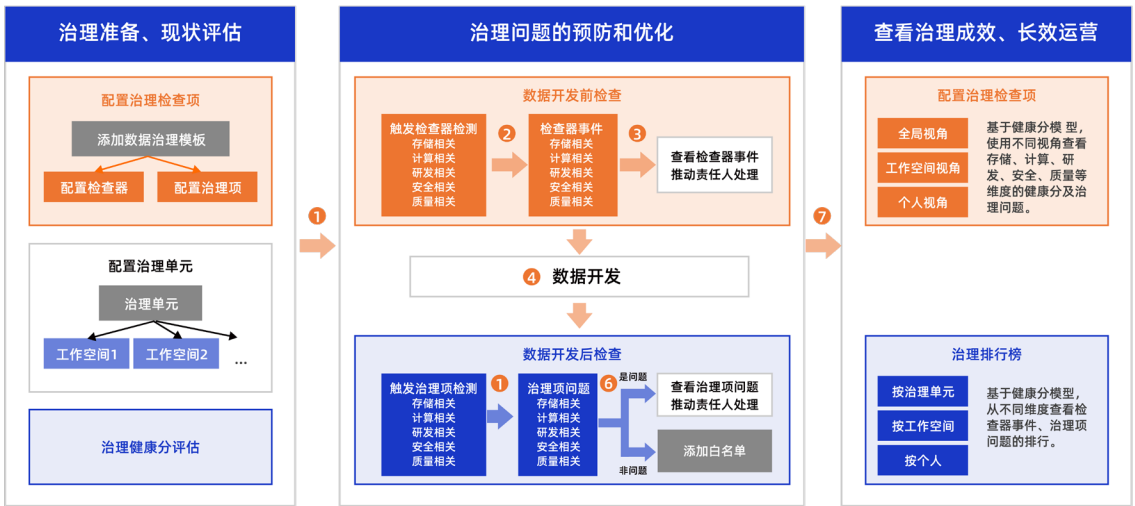


图 8 数据治理实现路径图

(4.1) 数据治理现状评估

根据数据治理的治理项和检查项查看整体的治理评估报告，也就是治理的健康分评估。治理评估报告提供租户全局、单个工作空间以及具体个人三个视角的报告，覆盖研发、质量、安全、计算和存储五个维度，给出量化的具体评估。最关键的一点，对于不同的工作空间、不同的个体，这个评估模型采用的是同一套标准，保证评估的客观一致性。这份报告，可以作为治理工作正式开始实施前的一个基础参照。

数据治理健康分评估模型

数据治理健康分基于治理项发现的问题、按照定义的模型计算得出。采用的扣分逻辑为满分 100 分，通过内置的算法模型，按需要治理问题减掉扣分后得到健康分。

数据治理健康分评估模型									
研发规范健康分		数据质量健康分		数据安全健康分		计算资源健康分		存储资源健康分	
命名规范检测	注释检测	未配质量规则表检测	告警未处理表检测	数据下载检测	数据保护模式	费用消耗检测	任务参数不合理	空表检测	无生命周期表检测
运行时长检测	分区检查	基线任务质量规则检测	指定规则未配置检测	数据存储加密	安全访问模式	简单加工检测	产出表未被读取	长时间为访问检测	表大小检测
distinct检查	select * 检查	脏数据检测	质量规则覆盖率检测	数据源访问控制	开发生产数据资源隔离	输入为空检测	数据倾斜检测	重复表检测	表引用次数检测
禁用insert into	...	连续告警检测	...	成员与角色控制	...	暴力扫描检测	...	无效表检测	...

图 9 数据治理健康分评估模型

数据治理细分了研发、质量、存储、安全、计算和存储五个维度的单项健康分，并综合计算得出整体健康分。这个逻辑可能看起来并不复杂，复杂的在于底层数据获取、加工构建、治理问题洞察。

(4.2) 数据治理实施：问题预防、发现和优化

数据治理的实施根据检查项和治理项来进行管理，检查项面向事前治理问题预防，它会侵入日常任务的提交、发布等环节，如果检测不通过会阻塞流程。治理项面向事后治理问题发现，针对已发现的问题进行专项处理。

治理问题的处理优化 – 自动预防（检查项）

检查项开启后，可以作用在某一个具体空间，在任务提交或发布环节，能够自动触发扫描。数据治理的检查项根据业务需求专项制定，检查项也在随着在企业使用沉淀，以及依据业务使用的反馈，进行累积。

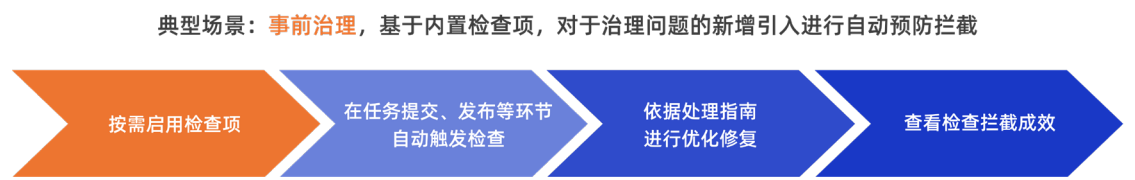


图 10 数据治理实施 – 自动预防

治理问题的处理优化 – 自动发现（治理项）

事后治理使用到的是“治理项”的能力，治理项和检查项不同，治理项需在数据治理中扫描出需要治理优化的问题，并提供相应的处理指南、指导对问题进行优化。与检测项相同，治理项也随着在企业使用沉淀，以及业务使用的反馈，慢慢积累。



图 11 数据治理实施 – 自动发现

（4.3）数据治理的长效运营机制

在企业内部数据治理的演进中，能看到三个明显的方向，分别从组织、平台、业务三个方向来描述。

首先，数据治理不单纯是大数据团队一直在搞技术、建平台，它更多的是一个组织协同的问题，会跨越过原先单技术团队，到影响到公司整体的架构设计，如下图左侧，有数据平台团队，有业务团队，还有财务、风控等协同团队。

涉及到跨团队，对于整个组织来说，一个很头疼的问题就是如何来衡量效果？如何更好地发挥组织的主动性？在企业内部做治理，经常会发现，有一个很好的规范，但是没有平台工具来落地。在企业内部，这是设计治理健康分一个很大的出发点。对于某个团队来说，比如今年的目标之一，就是把健康分从 70 分提升到 80 分，可以从计算、存储、研发、治理、安全等各个方面入手，有什么需求可以提给数据平台团队，将这些能力都沉淀到平台工具上，目标大家一起来共背。

通过这种方式，各个团队就会有一个统一的考核指标来指引进行数据治理的工作。在长效推进上，会启动各类的数据治理战役，各个业务团队之间的治理成效比武等等长效的运营工作，也可以通过健康分做不断地延展，达到组织数据的协同目的，发挥数据治理组织的主动性。



图 12 数据治理长效运行机制

从长远角度来看，数据治理也着眼于将数据治理从小部分人的工作转变为有良好群众基础和参与度的普遍性工作。数据治理可以让治理参与同学清楚感知其所处的位置，让优秀的得到表扬，不足的得到鼓励；同时面向治理管理员和普通同学提供不同的视角，让其清晰了解治理健康度水平和需要优化的问题，有的放矢地进行优化治理。

（5）数据治理的未来目标

数据治理立足降本增效的核心诉求，围绕治理问题的自动预防和治理演进，提升治理问题的处理效率。基于这些述求，数据治理的未来目标有：

- 根据企业数据治理最佳实践，持续丰富治理项和检查项，让治理问题得以更全面地发现和预防。

- 为任务下线、表删除等治理操作提供优雅处理方案，解决治理风险顾虑，提升问题优化治理效率和处置完成率。
- 持续夯实资源使用分析洞察功能，切实帮助控制不合理的资源使用花费。
- 拓展支持的大数据产品，使数据治理能广泛运用到实际场景中。
- 面向不同行业提供数据治理最佳实践和数据治理行业模板。

3. 小结

本文详细探讨了大数据中的数据质量管理与数据治理。我们首先分析了大数据管理的痛点，明确了数据质量的定义，并详细介绍了如何通过规范流程和技术手段进行数据质量管理。接着，深入探讨了企业数据治理的挑战，构建了完整的数据治理体系和业务体系架构，提出了从现状评估、问题预防与发现到优化和长效运营的数据治理实现路径。

总之，高效的数据管理和治理是确保数据准确、一致和可用的关键。合理的数据质量管理措施能够解决数据不一致和质量问题，而完善的数据治理体系则为企业提供了可持续发展的保障。未来，随着技术进步和业务需求的变化，数据治理将继续演进，助力企业在数字化转型中保持竞争力并实现长期稳定发展。

（四）解锁数据的价值——如何构建数据服务能力

在当今数字化转型的大背景下，数据已成为企业最宝贵的资产之一。如何高效地管理和利用这些数据，成为企业面临的重要课题。大数据和数据仓库体系的建设为企业提供了强大的数据处理能力，而数据服务则是将这些数据转化为业务价值的关键环节。本文将详细介绍数据服务的定义、典型场景、常见难题以及解决方案，旨在为企业提供一个全面的数据服务能力建设指南。

1. 数据服务的定义

在大数据和数据仓库体系建设中，数据服务通常指的是一种面向最终用户和应用程序的数据访问方式。数据服务的目的在于抽象数据访问的复杂性，提供可重用、可靠和高性能的数据接口。它是数据治理和数据架构设计的重要组成部分，旨在简化和优化数据的提供与消费。数据服务通常包含以下几个范畴：

- **数据访问接口**：数据服务通常通过标准化的 API（如 RESTful API、GraphQL、SOAP 等）提供经过处理的数据访问，这些接口是为了满足不同业务需求和使用场景而专门设计的。
- **数据抽象层**：数据服务提供对底层数据的抽象，使得用户无需了解数据存储的细节，即数据库的表结构、技术实现等。通过这种抽象，可以方便地处理数据存储技术多样化的问题，如关系型数据库、NoSQL、数据湖等。
- **数据集成和聚合**：数据服务通常负责从多个数据源提取、转换和加载数据，并可能对数据进行聚合整合，为用户提供一个统一的数据视图。
- **安全和权限管理**：数据服务提供对数据访问的安全控制，包括权限验证和访问记录，确保数据被授权的用户按预期访问。
- **性能优化**：数据服务常常包括缓存、负载均衡等技术手段，以提高查询性能和系统响应速度。

2. 企业数据服务的典型场景

数据服务在企业中的应用非常广泛，以下是几个典型的场景：

- **对内数据服务：**企业数据部门提供数据给企业业务部门，用数据赋能业务，比如客户管理、仓储查询等。通过内部数据服务，业务部门可以快速获取所需的数据，提高决策效率。
- **对外数据服务：**企业进行数据能力变现，例如常见的物流查询、短信验证码、实名认证等。对外数据服务不仅提升了用户体验，还为企业带来了新的收入来源。

• **企业运营分析 - BI：**数据仓库提供已经汇总好的企业运营指标，提供给商业分析部门，用于进一步的商业决策分析。通过 BI 工具，企业可以实时监控关键指标，及时调整策略。

3. 数据服务化中常见难题

数据服务化过程中，针对不同角色定位，往往会有不同的难题，主要表现在如下几类：

（1）管理者的难题

- **数据服务开发和管理体系不完善：**缺乏规范的数据服务开发流程和管理体系，导致数据服务的质量和稳定性难以保证。
- **烟囱式独立获取数据：**各业务系统各自独立获取数据，导致重复建设和资源浪费。缺乏统一的数据服务入口，使得数据孤岛问题严重。
- **权限体系不方便：**权限管理体系不完善，网关策略不健全，导致数据安全无法得到有效保障。数据泄露风险增加，影响企业的整体安全性。

（2）开发者的难题

- **业务分析查询场景多样：**业务分析查询场景多样，个性化需求多，导致开发工作量大，维护成本高。
- **数据接入方式多样：**数据接入方式多样，接入效率低，增加了开发人员的工作负担。
- **数据和接口无法复用：**数据和接口无法复用，无法共享，导致开发效率低下。
- **多物理表关联查询复杂：**多物理表关联查询 SQL 代码复杂，维护困难，容易出现性能瓶颈。

（3）数据应用者的难题

- **没有全局服务市场：**没有全局服务市场或没有详细的服务资料，不知道怎么复用或查找已有服务，导致资源浪费。
- **只能使用已有服务：**只能使用已有服务，不够灵活，无法快速响应业务需求。
- **定制化需求开发响应缓慢：**定制化需求开发响应缓慢，排期数天才能支持，影响业务发展速度。

4. 数据服务难题的治理体系

针对以上难题，行业内通用的解决方案是建设统一的数据服务能力，以实现面向业务提供统一数据服务出口和统一业务查询逻辑；同时面向底层技术实现频避底层物理细节，使数据开发者专注于实现业务逻辑本身。具体可以通过以下几个方面的能力建设来实现：

• 统一数据服务市场

- **构建 API 市场：**构建 API 市场，实现接口复用。开发者可以在 API 市场查看已有的接口，申请使用权限授权后调用数据，无需重复开发。API 市场还可以提供详细的接口文档和示例，帮助开发者快速上手。
- **屏蔽中间存储差异：**数据服务对各个数据应用屏蔽了不同的中间存储，提供统一规范的服务访问。无论底层是关系型数据库、NoSQL 还是数据湖，开发者都可以通过统一的接口访问数据。

• 企业数据服务中心

- **认证和权限管理：**数据服务具备认证和权限管理功能，保障接口安全和数据安全。通过 OAuth2.0、JWT 等标准协议实现身份验证，确保只有授权用户才能访问数据。
- **限流和监控：**数据服务提供服务限流、监报告警等功能，保障数据被合理访问与调用。通过设置合理的请求速率限制，防止恶意攻击和资源耗尽。同时，监控系统可以实时跟踪服务状态，及时发现并解决问题。

- **统一规范的高效开发能力**
 - **配置化和 SQL 化的服务开发：**提供配置化和 SQL 化的服务开发方式，屏蔽物理数据源底层实现，研发人员可以专注在业务实现逻辑上，高效开发。通过可视化界面配置数据服务，减少手动编写代码的工作量。
 - **自动化测试和部署：**引入自动化测试和持续集成 / 持续部署（CI/CD）机制，确保数据服务的质量和稳定性。通过自动化测试，可以快速发现并修复问题，缩短开发周期。
- **服务全生命周期管理**
 - **一站式 API 生命周期管理：**从 API 开发、测试、调试、调用、监控、运维，提供一站式 API 生命周期管理解决方案。开发者可以通过统一的平台管理整个 API 生命周期，提高工作效率。
 - **版本管理和发布：**支持 API 版本管理和发布机制，确保新旧版本的兼容性和稳定性。通过版本管理，可以逐步升级服务，避免一次性大规模变更带来的风险。

5. 小结

随着大数据和数据仓库体系的不断发展，数据服务已成为企业信息化建设中的重要一环。通过构建统一的数据服务市场、企业数据服务中心、统一规范的高效开发能力和服务全生命周期管理，可以有效解决数据服务化过程中面临的各种难题。这不仅提升了数据服务的可用性和安全性，还极大地提高了企业的数据利用效率和业务响应速度。

未来，随着技术的不断进步和应用场景的拓展，数据服务将更加智能化、自动化，为企业带来更大的价值。企业应持续关注数据服务领域的最新进展，不断提升自身的数据服务能力，以适应快速变化的市场环境。通过有效的数据服务建设，企业能够更好地利用数据资产，推动业务创新和发展。

第五节 / 重点行业数据云化治理最佳实践

（一）传统数据库到云原生数据库的“丝滑转身”——某政企客户打造数据库上云最佳实践

1. 上云背景

- **数字化转型的需求：**随着信息技术的发展，某政企客户越来越依赖于数据来支持决策过程、提高运营效率和服务质量。云计算提供了一种高效且经济的方式来管理和分析海量的数据。
- **成本效益考量：**传统的本地部署数据库解决方案需要大量的前期投资，包括硬件采购、软件许可费用以及维护成本。而云计算提供了按需付费的模式，降低了初期投入，并且可以根据业务需求灵活调整资源。
- **安全性提升：**近年来，网络安全威胁日益严重，客户对数据安全的要求也不断提高。云服务提供商通常会提供更高级别的安全措施，如加密、访问控制等，以确保用户数据的安全性。
- **技术创新驱动：**云平台能够快速集成最新的技术成果（例如人工智能、机器学习），客户可以利用这些先进技术优化业务流程、创新产品和服务。

2. 数据上云面临问题与挑战

- **数据迁移复杂性：**将现有系统迁移到云端涉及到复杂的规划和技术实施，尤其是对于类似该客户这样的大型政企而言，它们往往拥有多个相互关联的应用程序和庞大的历史数据，这使得迁移工作变得更加棘手。

- **合规性和监管要求：**政企客户在处理敏感信息时必须遵守严格的法律法规。不同国家和地区有不同的隐私保护法规（如 GDPR），因此选择合适的云服务商和制定相应的策略变得尤为重要。
- **性能和可靠性担忧：**尽管大多数主流云服务商都承诺了高可用性和低延迟的服务级别协议（SLA），但客户仍然担心云环境下的性能波动可能影响关键任务型应用的表现。
- **技能短缺：**云技术和传统 IT 基础设施存在差异，导致市场上缺乏足够的专业人才来进行有效的设计、开发、部署和支持。此外，内部员工也需要接受培训才能适应新的工作方式。

3. 数据库上云最佳实践

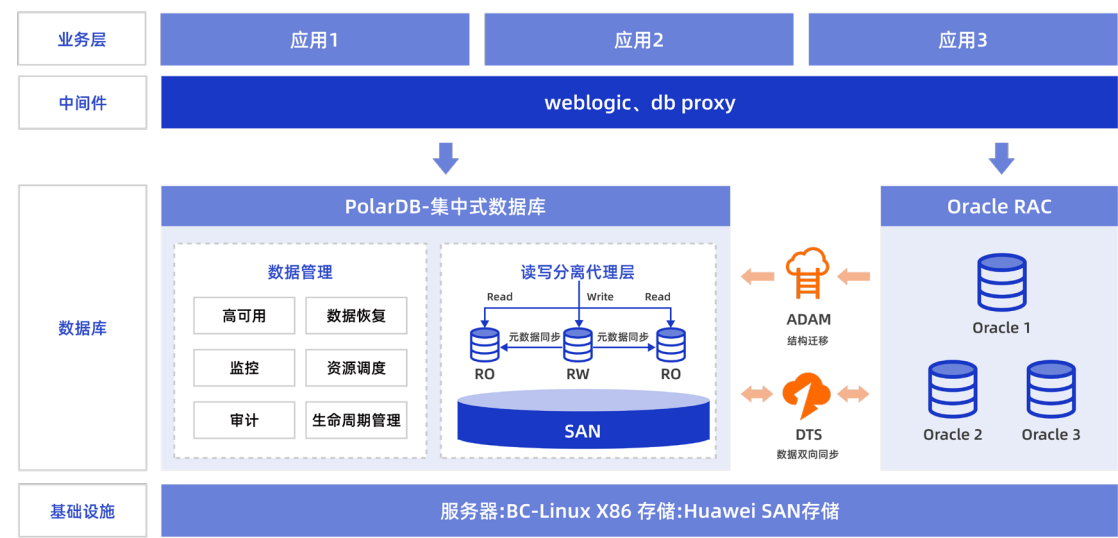


图 1 客户云上数据库整体架构示意图

因为实施的业务是客户侧核心业务，涉及上下游各方的应用适配和协调测试，从开始适配测试到正式割接上线，耗时约 3 个月，顺利完成所有上下游业务的整体迁移，平稳度过当天业务高峰期，帮助客户实现数据库软件安全可控，实现从传统数据库架构到云原生数据库架构的平稳过渡。

接下来我们围绕整个迁移过程，详细描述下是怎么一步步推进项目进程，怎么提高交付效率，确保迁移实施高质量高稳定，帮助客户实现业务转型。

（1）标准迁移流程指导

经过多年多个数据上云实践我们总结出了数据库迁移上云的方法论：“五阶十步”，该方法论是通用的数据库交付沉淀，是对数据库交付工程化的抽象，用于指导通用数据库交付。基于五阶十步方法论的指导，将数据库迁移抽象为五大主要步骤：



图 2 数据库迁移五大主要步骤图

在项目初始阶段主要是咨询工作，该阶段主要是梳理需求，梳理交付范围，确定组织阵型，确定交付方案、交付计划及里程碑。初始阶段主要对照五阶十步中的调研与评估、设计与改造。其次是项目主要交付阶段，包含测试与演练、迁移与割接，最后是转维阶段。

（2）调研与评估

数据库选型

当前国内数据库厂商数量多，数据库产品数量多达 200+，在如此多厂家以及数据库产品中如何选择一款适合企业业务的数据库是当前的首要问题。经过测试和调研，并结合客户的业务需求，选择 PolarDB 作为前期试点产品，选型的主要考量点如下：

- ① 是否能支持业务的持续发展以及架构演进。
- ② 数据库代码是否安全可控。
- ③ 是否有广泛的应用案例。
- ④ 是否能够兼容原有开发习惯，降低开发成本。
- ⑤ 是否能完整适配国产操作系统和国产硬件。
- ⑥ 是否能够充分利用原有硬件设备。

可行性评估

在做完需求调研之后，要尽快完成可行性评估，评估整体迁移的工作量和改造工作量，我们从如下几个维度来评估：

评估项	描述	工作量占比
数据迁移	根据数据条数、存储大小和数据库对象数量，结合模拟迁移次数评估迁移成本	30%
结构迁移	根据对象兼容性和 sql 兼容性评估对象迁移成本	5%
数据链路	上下游数据链路梳理，评估上下游链路改造工作量	5%
上层应用改造	应用统计，包括脚本和定时任务等，评估应用改造量	30%
功能测试	评估功能测试工作量	20%
性能评估	压力测试和性能优化评估	10%

表 1 迁移工作量评估表

这一步非常关键，如果数据库兼容性不高，会带来较大的对象改造工作量和应用改造工作量，在整体项目平迁的逻辑下,过高的改造成本会导致平迁逻辑演化到重构逻辑,可能导致成本不可控，项目失败。再比如如果存在下游数据链路，但是没有对应的替代方案，会导致无法进行数据库割接。所以在前期我们通过多个维度的成本评估，提前暴露风险，确保项目成本可控、方案可行。

在该项目中，客户侧业务系统数据量数 TB，数万张表，总数据行数在百亿级别。经过 ADAM 工具的评估，数据库对象和 SQL 兼容性在 99% 以上，兼容度很高。但是因为核心系统，上层业务功能复杂，测试功能点多，所以要投入更多精力在功能测试、应用改造和性能优化上。

（3）设计与改造

基于前期的需求调研和现状评估，我们为该项目量身定制了完整迁移方案，内容主要包括：架构设计、迁移方案、压测方案、割接方案、回退方案等。

方案设计	实施内容	对应工具
系统架构	PolarDB 集中式架构方案	/
对象改造方案	对象兼容性改造列表	ADAM
迁移方案	结构迁移和数据迁移	ADAM、DTS
压测方案	压力测试和性能优化	Frodo
割接方案	割接操作手册	/
回退方案	回退操作手册	/

表 2 迁移方案设计实施参考

整体项目推进涉及多个角色：交付技术架构师、解决方案架构师、客户和生态伙伴（ISV）。交付技术架构师负责系统架构设计、对象改造、迁移方案设计等，解决方案架构师负责内外部的沟通和协调，参与方案评审；客户负责协调提供服务器等硬件资源；生态伙伴（ISV）主要负责应用适配、功能测试，协同交付技术架构师进行方案实施。

（4）工具支撑测试与数据迁移

工欲善其事必先利其器，在交付的每个环节，大量使用工具来解决交付的效率问题，每个工具在交付过程中的不同阶段发挥不同的作用，尽量使用工具把交付流程串联，做到可复用。基于丰富的实施经验，沉淀了大量可复用的交付工具，下图介绍了工具在数据库上云落地过程中的应用。

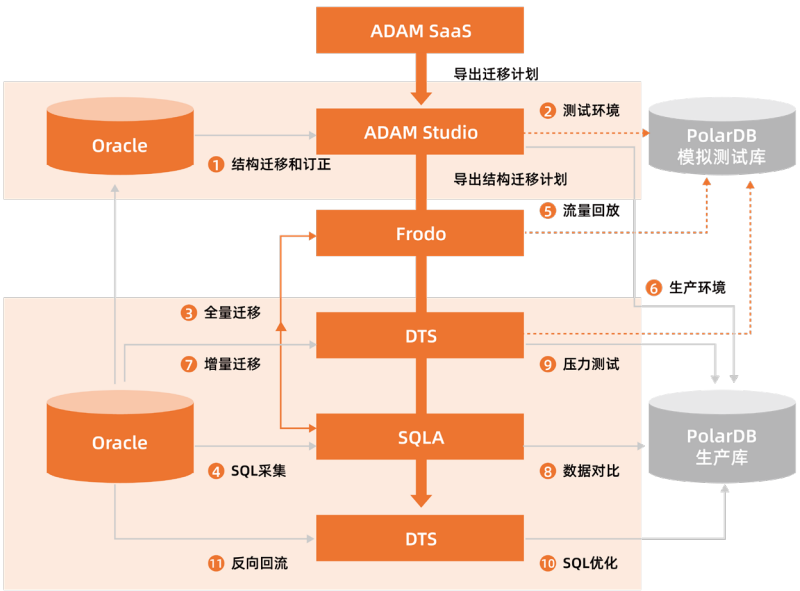


图 3 交付工具集

模拟迁移

模拟迁移阶段主要是验证数据库以及相关工具组件，测试新业务在数据库是否正常。通过 2-3 次的模拟迁移，确认迁移工具功能的完备性；基于多轮的测试模拟，尽可能的暴露 SQL 兼容性问题、性能等问题；详细记录每一次模拟迁移过程中遇到的问题和调整，为最终的生产数据迁移和割接方案奠定基础。

模拟迁移	交付内容	交付工具
结构迁移	全量数据结构迁移，包括表结构、索引、触发器、函数、存储过程等	ADAM
数据迁移	全量迁移 + 增量迁移	DTS
数据校验	全量数据校验	DTS、青天鉴
流量回放	真实 SQL 流量回放，进行性能压测	Frodo

表 3 模拟迁移交付参考

我们把性能优化、全量数据校验工作放到模拟迁移阶段完成，因为模拟迁移完全使用生产环境，所以在正式生产环境迁移阶段，不需要重复做性能优化工作和频繁的全量数据校验工作。在模拟迁移阶段的每一项工作都是关键点，完整且细致的完成每一次模拟迁移，同时详细记录过程中的配置调整和实施细项，是该阶段最核心的工作内容。

生产迁移

得益于模拟迁移阶段的充分准备，基于模拟迁移阶段沉淀的迁移方案，准备生产环境的数据迁移。和模拟迁移阶段不同，该阶段数据校验主要覆盖行数校验和核心表全量校验，对于大量历史表不在进行校验，该调整节省了大量时间。

生产迁移	交付内容	交付工具
结构迁移	全量数据结构迁移，包括表结构、索引、触发器、函数、存储过程等	ADAM
数据迁移	全量迁移 + 增量迁移	DTS
数据校验	常态化数据行数校验、核心表全量校验	DTS、青天鉴

表 4 生产迁移交付参考

（5）割接与回滚

事前充分准备，事中轻松面对。在模拟割接阶段沉淀了完整的迁移操作步骤、性能优化方案等操作，通过总结前期的测试过程和操作文档，结合阿里巴巴去 IOE 最佳实践，准备最终的割接手册和回滚手册。

割接准备

步骤	操作说明	执行条件	check
目标库配置调整	调整参数，提升同步速度，避免日志打满磁盘	必须	✔

结构迁移	使用 ADAM 进行结构迁移	必须	✔
关闭表触发器	关闭表触发器	存在触发器	✔
关闭外键约束	关闭外键约束	存在外键约束	
数据迁移	根据模拟演练的经验进行任务切分提高迁移速度	根据数据量选择是否切分	✔
	全量迁移	必须	✔
	增量迁移	根据数据量和割接方案选择是否需要增量	✔
结构变更检查	检查是否存在结构差异，并修复	必须	✔
数据校验	行数校验，可周期性执行	必须	✔
全量校验	检查是否存在结构差异，并修复	必须	✔
兼容性解决方案	执行在测试和演练阶段沉淀的兼容实施性解决方案	必须	✔
性能问题解决方案实施	执行在测试和演练阶段沉淀的性能优化方案，如索引优化、表结构优化	必须	✔
第二次数据迁移	提前数小时对不一致大表进行数据迁移，减少停服后的第三次数据迁移的时间	存在结构差异或数据不一致表，且数据量较大，无法在停应用后单独同步	✔
恢复数据库配置	恢复数据库到生产配置	必须	✔

表 5 割接准备操作表

割接操作

步骤	操作说明	执行条件	check
停应用	停止应用	必须	✔
锁定 oracle 应用账号	锁定 oracle 库应用账号，确保 oracle 无业务写入	必须	✔
停止数据同步任务	确认同步追上，且无新增数据，然后停止数据同步任务	必须	✔
抬高 sequence	抬高 sequence，避免 sequence 冲突	必须	✔
核心表校验	一般使用行数校验工具快速检验	必须	✔
结构变更检查	检查是否存在结构差异	必须	✔
第三次数据迁移	对数据不一致的表进行全量迁移	存在数据差异或数据不一致的表	✔
恢复 polardb 触发器	恢复 polardb 触发器	存在触发器	✔
恢复 polardb 外键约束	恢复 polardb 外键约束	存在外键约束	✔

关闭 oracle 触发器	关闭 oracle 表级别的触发器	必须	✔
启动回流任务	启动 polardb->oracle 增量任务，并验证回流任务的有效性	必须	✔
启动应用	启动应用	必须	✔
功能测试	功能测试	必须	✔

表 6 割接操作表

紧急回退（如果割接出现异常，直接终止割接）

步骤	操作说明	执行条件	check
停止回流任务	停止回流任务	必须	✔
恢复 oracle 触发器	恢复 oracle 触发器	必须	✔
应用回切	应用回切	应用启动失败或核心功能测试失败或割接操作超时	✔

表 7 紧急回退操作表

回滚手册模版

应用割接成功后，建立快速应急响应值班机制，做到问题发现和问题解决的快速闭环。坚持遇到解决问题的原则，如果没有出现灾难性的场景，不会轻易做回滚操作，回滚是最后的选择。尽管回滚概率不高，但是依然需要准备回滚手册，以应对小概率的回滚场景。

a. 回滚准备

步骤	操作说明	执行条件	check
结构变更检查	检查是否存在结构差异并修复	必须	✔
数据校验	常用 count 条数校验	必须	✔
	全量校验	必须	✔
数据修复	提前数小时对不一致的大表进行数据迁移，减少停服后的数据迁移时长	存在数据不一致大表或结构不一致的大表	✔

表 8 回滚准备操作表

b. 回滚操作

步骤	操作说明	执行条件	check
停应用	停止应用	必须	✔
polardb 锁定账号	polardb 锁定账号，杀连接禁止业务访问	必须	✔
停回流任务	停止回流任务	必须	✔
oracle 抬升 sequence	抬高 oracle 的 sequence，确保 sequence 一致或更高	必须	✔
结构变更检查	检查是否存在结构差异并修复	存在新增表或不支持的 DDL	✔
数据修复	不一致的表重新进行数据同步	存在数据差异或数据不一致的表	✔
数据校验，确保数据一致性	一般使用行数校验工具快速校验	必须	✔
恢复 oracle 表触发器	恢复 oracle 表触发器	必须	✔
应用启动	修改连接配置，启动应用	必须	✔
功能测试	验证功能可用性	必须	✔

表 9 回滚操作表

从上述步骤可以看到回滚步骤操作内容繁多，势必会导致回滚时间过长，所以回滚在大部分场景下是最坏的选择。我们在前期测试和模拟阶段做好充分准备，把大部分精力投入到保障割接成功的工作中，完成了生产数据库的割接操作。

（6）交付转维

转维阶段的核心是做好两件事：线上护航和交付物收集。新业务上线后，或多或少会存在一些问题，例如请求慢、请求失败等，做好问题的及时处理和记录。同时做好交付后的检查工作，回顾交付过程，有哪些还没有完成，例如数据库监控、数据库备份、高可用是否有恢复等，准备好 checklist，统计遗留事项，进入常态化运维及稳定性建设阶段。

4. 结束语

本文概述了上云背景，分析了数据上云面临的问题与挑战，并通过一个详细的数据库上云实战案例，系统介绍了“五阶十步”方法论及其具体实施流程。该案例覆盖了从调研评估到交付转维的全生命周期，确保数据迁移的安全性和可靠性，为企业提供了一套切实可行的数据库上云指南。

（二）数仓升级的“精准施策”——某券商数据仓库升级最佳实践

1. 行业变化与客户痛点

数字经济已被公认为经济发展的重要力量，也在不断的深刻改变和重塑着当下的经济发展和活动方式。证券行业作为服务实体经济的先锋力量，对于加快数字化转型的发展有着重大意义。近几年监管部门先后出台了一系列政策文件，强调推动证券行业数字化转型并以科技赋能业务发展，实现金融科技与证券业务发展互相促进、良性循环。这为证券行业的数字化转型和高质量发展奠定了坚实的基础。证券行业的数字化转型可总结为三大利好，具体如下：

- ① 有利于全面提升该行业乃至金融服务实体经济的整体效率与水平，为经济的整体高质量发展提供强大引擎。
- ② 有利于实现对资本市场发展的强大支撑力量。
- ③ 有利于激发数字化经营新动力。

证券行业对数据的高质量要求正逐步推进证券行业的数字化转型步伐。目前，证券业务的信息系统已积累和储备大量业务数据，这些数据体量大、种类多、更新快、价值高，更为重要的是其数据价值可以有效的为证券业务创新、业务产品升级优化、企业决策、风险管理等层面提供重要支持，而这些价值数据也是证券行业的核心资产。

该客户作为国内领先券商，一直重视挖掘数据资产的价值，此前采用了十几年的国外传统数据仓库系统为业务发展带来良好的业务支撑。但随着大数据时代的到来，原有的数据仓库架构硬件老化、资源耗尽、扩容和数据迁移成本高、难以升级，无法满足客户日益增长的业务数据需求。在业务快速变化的今天，尤其是在核心业务系统要求满足安全可控的大背景下，数据仓库升级成为券商的普遍选择。

2. 数据仓库升级交付目标

基于以上原因，该客户决定选择全新的云原生数据仓库进行升级，在综合对比多家厂商性能后，选择携手阿里云通过采用 AnalyticDB for PostgreSQL 等云上组件成功将服役了十几年

的某厂家传统数据仓库平台云化升级，在云平台底座上，成功搭建了具备数据接入、数据存储、数据计算、横向扩展、数据共享的先进数据仓库平台。阿里云结合证券期货行业数据模型（简称 SDOM）经验，高效实现了数据模型从原数据仓库向新平台的平滑迁移。该客户数据仓库系统整体性能提升 40% 以上，支撑了数万张核心表，为运营及业务快速发展提供了可靠、高效的云上基础设施环境。该项目的数据仓库平台升级实施涉及一百多套上游业务源系统、三十余套下游系统、两万多个任务、近 1000 多个服务接口、500 多 TB 数据（日新增 500 多 GB 数据）的业务规模，整体的迁移过程如下图 1 所示。

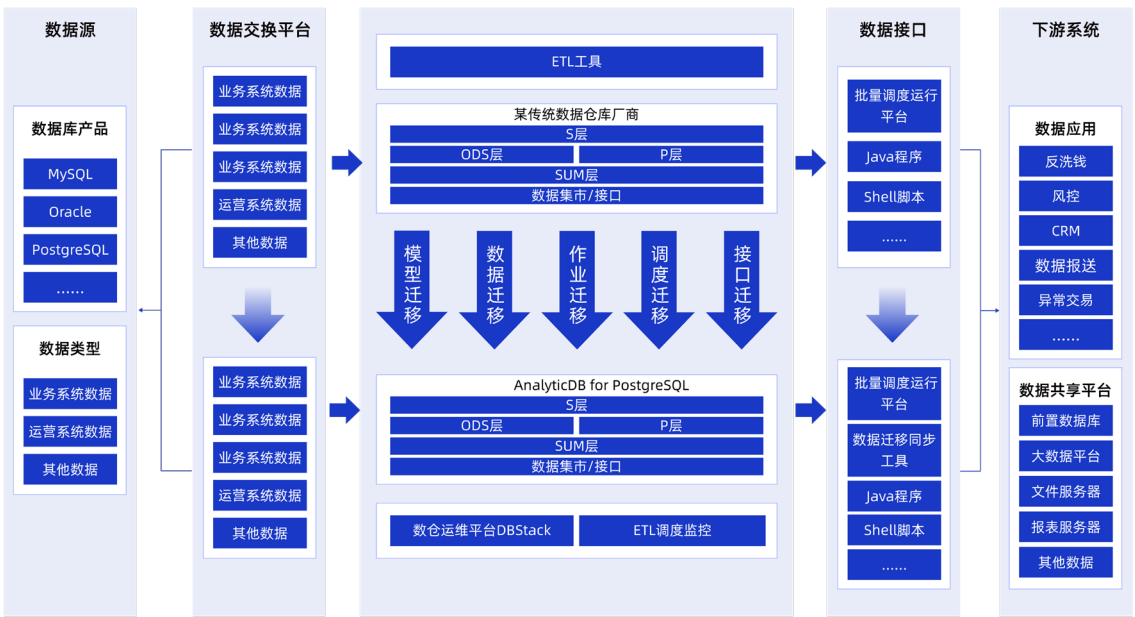


图 1 数据仓库迁移过程图

由于数据协同的上下游系统较多，原厂家的传统数据仓库平台升级替换耗时约 12 个月，在各方的共同努力下，最终完成所有预计目标的平稳迁移，同时保障现有业务平稳有序运转，帮助该证券公司实现安全可控、成本与风险双降，数据快速赋能业务。那么，数据仓库升级是如何一步一步展开的？如何通过交付标准化手段，围绕交付标准流程、交付工具及数字化工作台，确保数据仓库迁移高质量高稳定履约，帮助客户数据业务转型升级？以下主要将结合该项目中总结沉淀出的数仓升级经验进行介绍。

3. 数据仓库升级最佳实践

（1）升级交付流程标准化

项目启动后，首先进入轻咨询阶段，由客户数据中心、阿里云专有云服务团队以及 ISV 合作伙伴一起进行项目前期的交付咨询梳理。包括项目组织实施机制以及前期环境准备等。

此外快速构建起由客户、阿里云和 ISV 伙伴组成的项目交付阵型，明确职责矩阵、分工界面及衔接方式，结合一站式交付工作台 KOC 编排的工作需求说明书及标准流程动作，形成项目交付的里程碑节点计划。

在阿里云技术服务团队的统筹下，基于项目交付里程碑节点要求、项目执行计划及项目管理章程，高质量高效率推进项目交付标准流程动作。下文以需求调研和方案设计两个关键动作示例说明。

（1.1）业务需求调研

由于证券业务系统较多，数据仓库升级需要对接的上下游系统不但要保证平稳迁移，更需要根据不同的系统需求反馈，逐步按阶段按计划进行排期，业务调研阶段需对原数据仓库系统上下游每一个系统做详实调研，调研内容包括原数据仓库系统架构及环境、数据调研与分析、模型调研与分析、数仓 ETL 任务调研与分析等，输出调研分析报告，并与业务方做深入讨论与修正。

业务需求调研	交付任务	交付工具
原系统资源盘点清单	业务需求调研会议纪要 业务需求分析说明 技术需求调研会议纪要 数据需求调研会议纪要	SQL 兼容改造评估工具
原数仓系统架构反馈		
原数仓数据交互流程反馈		
原数仓技术性能要求清单		
原数仓库表统计清单		
原数仓 ETL 任务 / 脚本统计清单		
原数仓用户 / 权限统计清单		

表 1 业务需求调研表

交付分工上，整体由解决方案架构师负责，交付技术架构师、客户和 ISV 伙伴协助配合；项目经理负责组织对产出物进行评审；交付技术架构师结合交付工作台产出调研清单进行数据盘点，梳理原有数仓系统架构和数据流向，并参与客户的全局业务架构调研及核心业务流程调研；客户负责整理系统调研和数据盘点所需资料，组织业务调研和技术调研，并对业务、数据和技术调研的问题进行解答，参与产出物评审及确认；伙伴配合交付技术架构师进行需求调研和数据盘点实施。

（1.2）迁移方案设计

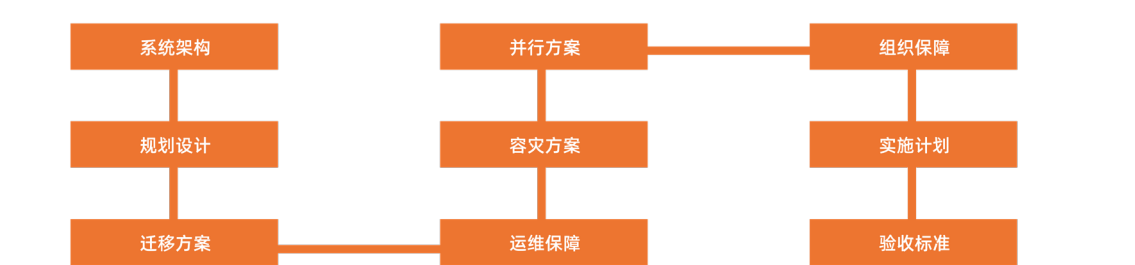


图 2 方案设计示意图

方案设计阶段是根据前期的业务需求调研以及客户系统实际情况进行量身定做，这个阶段是多方一起参与共创的阶段，最终设计并编制可落地的执行方案。内容包括：系统架构设计、规划设计、迁移方案、并行方案以及实施计划等。

方案设计	交付任务	交付工具
新系统架构说明	数仓升级技术架构设计方案 ADB PG 集群容量规划方案 ADB PG 应用开发方案 上下游系统交付设计 某传统数据仓库厂家迁移 ADB PG 方案 云上数仓容灾方案	SQL 兼容改造评估工具
规划设计说明		
迁移方案说明		
容灾方案说明		
并行方案说明		
实施计划说明		
验收标准说明		

表 2 方案设计详表

交付分工上，整体由交付技术架构师负责，解决方案架构师、客户和伙伴协助配合；交付技术架构师负责 ADB PG 集群容量规划、ADB PG 集群应用开发规范、迁移方案、容灾方案等的落地实施方案的设计编写。

解决方案架构师协助确认整体方案的内外部沟通协调，参与方案设计评审；客户协助确认实施的软硬件资源环境，参与评审并确认方案设计，并对业务与技术问题解析解答；伙伴配合技术架构师和业务架构师进行方案设计实施。

（2）交付内容工具化

（2.1）测试迁移

迁移测试阶段主要是对阿里云新数据仓库平台的功能性验证和典型场景验证，测试新系统在业务对接上是否存在问题。

同时该阶段还会论证迁移步骤的合理性，评估迁移工具的功能完备性；通过本阶段的典型场景验证，期望找到更多迁移交付中的技术性问题（如 SQL 兼容性、SQL 复杂度、作业复杂度等），用于更为准确评估迁移工作量，以便后续进行合理的工作安排。

方案设计	交付任务	交付工具
迁移工具验证	数据迁移验证报告 ADAM for TD 评估分析验证报告	数据迁移 DTS ADAM for TD
典型场景验证	调度迁移工具验证报告 测试迁移复盘报告	

表 3 测试迁移表

（2.2）生产迁移

生产迁移主要工作包括数据迁移、模型迁移和调度迁移三大块，而每一块的迁移都会涉及数据核对。该阶段迁移的主要难度在于三块工作完成迁移后，新数据仓库平台的数据要和原有数据仓库平台的数据一致，包括历史数据一致和新增数据处理的一致，并能够经过一个稳定周期的验证，任何一个环节的错误都将导致整体返工重做。

KOC 阶段：生产迁移	交付任务	交付工具
数据源迁移	数据源迁移及完整性报告	数据迁移 DTS
模型迁移	模型迁移及完整性报告	ADAM for TD
作业迁移	作业迁移及完整性报告	
数据迁移	数据迁移及完整性报告	
数据校验		

表 4 生产迁移表

在其中会用到一系列的交付工具，例如数据迁移工具 DTS，确保铺底数据的原始拷贝和数据自动导出导入，保证数据一致性，简化数据迁移流程；而数据模型迁移工具 ADAM（亚当），则确保不同系统的语法转换，以促进迁移的平滑过度。

由于项目涉及的上下游系统较多，该阶段实施中具体的工作内容可以分为以下几个部分，这里同时总结了本项目中对应的工作量占比。

工作内容	工作量占比
数据源迁移	5%
模型迁移	25%
作业迁移	10%
数据迁移	10%
新老系统数据比对	25%
下游系统对接	25%

表 5 项目工作量统计表

交付分工上，本阶段需要由交付技术架构师负责，解决方案架构师、客户和伙伴协助配合；交付技术架构师负责制定迁移方案设计和编写，并参与迁移及数据校验工作；解决方案架构师协助参与方案设计及评审工作；客户协助并确认迁移和校验结果；伙伴配合具体实施迁移工作。

(2.3) 系统并行

本阶段为迁移投产前的数据稽核校验阶段，由于数据仓库平台涉及核心业务系统，稳定性要求极高，有必要在迁移期间保证数据仓库新旧双系统并行，待数据迁移完整性校验通过后，再正式进入生产割接前的最后准备阶段。

KOC 阶段：系统并行	交付任务	交付工具
系统并行	新老数据仓库数据比对分析报告	青天鉴
业务割接	数据迁移成功确认函	

表 6 系统并行表

由于涉及的上下游系统较多（例如反洗钱，风控，CRM 等均与数据仓库做大量连接），因此在系统并行阶段，交付工作主要面临数据不一致、新旧平台特性差异以及系统配置不同这三方面的挑战。

数据不一致方面，统计数据迁移不仅涉及数据表及数据量。数据核对的工作更为繁琐，数据核对可分为对象级、记录级、指标级、全表级、每日批量几个级别的核对。每个级别都涉及大量的脚本编写和繁重的数据比对。基于此开发自动核对脚本生成工具，可根据元数据配置自动生成核对脚本，并可以对脚本进行批量调度，快速得出核对结果。这样，核对的主要工作就是发现不一致的数据并解决对应的问题。平台特性差异以及系统配置方面的差异挑战,则主要通过系统调整、推送端适配修改、以及人工校对修改等方式进行完善。

这个阶段也会用到一些标准化交付工具，例如数据校验工具青天鉴等，这些工具主要完成数据比对等功能，便于更好的实现双系统并行。

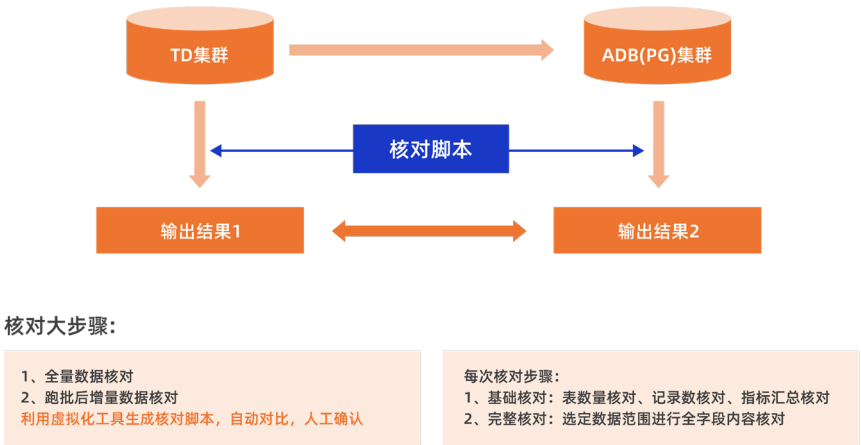


图 3 系统并行数据核对

在交付分工上，并行阶段需要由交付技术架构师负责，解决方案架构师、客户和伙伴协助配合；交付技术架构师负责数据比对策略指定及数据差异分析；解决方案架构师协助参与并行方案评审工作；客户协助确认数据比对结果的准确性、一致性；伙伴负责具体数据比对工作。

4. 结束语

该项目的升级交付经过标准化流程梳理后，升级顺利，从最终实施效果来看，云上数据仓库新平台比老平台运行效率高出 40%，这些数据每天都会被业务系统频繁调用，数据基础设施性能提升，增强了响应业务需求的时效性。正如客户数据中心部门负责人石先生所言，通过数仓升级，客户也一步步的向“四个化”进发，即实现数据治理全面化、数据资产标准化、推行数据应用智能化、实现数据服务制度化。

数据承载的核心载体，是云原生、高弹性、大并发、快回档、强安全的数智化基础设施和系统级解决方案。客户后续还将基于数据能力驱动，逐步实现全面数据运营。

第四章

稳定性建设篇



CHAPTER 04

第一节 /

业务稳定性的“居安思危”——面向失败设计

（一）系统高可用架构设计

云计算的资源模式和微服务的软件架构是相伴相生相辅相成的，如果在专有云上想更好的发挥云的高弹性、高扩展性和高可用性，那么云上的应用架构在微服务的设计上需要做好基础，尤其是“面向失败设计”指导下进行的微服务高可靠设计。

1. 面向失败设计（Design for Failure）

在复杂的分布式系统中，故障往往也呈现出分布式特征，容易引发雪崩效应或多米诺骨牌效应。这种情况下，故障的定位和排查变得尤为困难。因此，我们必须认识到，在分布式和微服务环境中，故障是必然会发生的。既然无法绝对避免故障的发生，我们就应该将处理故障的能力作为系统的核心部分，融入到架构和代码中，这就是所谓的“面向失败设计”。

在这里，跟大家简单的讨论几个常见的失败处理方法以及需要关注的事项。

- **重试设计**，在重试设计中通常需要注意以下几点：
 - 重试频度。要根据业务的重要程度选择是重试还是快速失败，如果确定需要重试，还需要考虑是否需要使用指数退避算法设置重试的时间间隔，以避免给下游服务方造成更大的压力。
 - 重试次数要有限定，这个可以结合防护能力中的熔断机制设计。
 - 考虑被调用方是否有幂等的设计。如果没有，那么重试是不安全的，可能会导致一个相同的操作被执行多次。

- **幂等设计**，一次和多次请求的处理应该具有同样的副作用，即 $f(x) = f(f(x))$ 。这个设计是方便上游的服务在拿不到我们服务对上游某个请求的处理结果时，可以让上游服务安全的进行重试的保障。这里面需要注意的是：
 - 全局 ID 的生成，针对每个请求需要生成全局唯一的 ID，可参考 Twitter 的 Snowflake 算法。
 - 重复请求的校验逻辑：可以将每次请求的全局 ID 进行保存，然后进行后续的查询校验。为了提高效率，也可以采用以全局 ID 做为 key 直接插入存储系统，判断是否存在插入异常（ID 重复）的方式来判断是否为重复请求，这样效率更高。
 - 具体存储方式可以采用关系型数据库，或是 key-value 的 NoSQL（如 MongoDB）。重试次数要有限定，这个可以结合防护能力中的熔断机制设计。
- **补偿事务**，针对某些跨系统的事务执行，为了提高效率可能会舍弃 ACID，采用 BASE 最终一致。这种业务场景下，当事务失败时可以采用补偿逻辑进行回滚。这个补偿逻辑通常是业务强相关的，需要根据具体的业务场景和逻辑来编写。

此外，虽然微服务使得应用研发本身的复杂度得到降低，利用微服务框架通过一致的接口、声明和配置，进一步屏蔽了源自于具体工具、框架的复杂性，降低了在不同工具、框架之间切换的成本。作为一个普通的服务开发者，微服务架构减少了开发的复杂性，但是微服务体系对于架构设计者来说却是充满了挑战。当微服务的规模越来越大的时候，复杂度也会指数级增长，随之而来会面临以下两类稳定性相关问题的困扰：

- **流量控制**：由于处理能力有限，服务虽然没有崩溃，但在面临超过预期的突发请求时，大部分请求直至超时都无法完成处理。这会造成类似交通堵塞现象产生的后果，如果没有进行及时的处理，就需要长时间才能使全部服务都恢复正常。
- **服务容错**：由于某一个服务的崩溃可能会导致所有用到这个服务的其他服务都无法正常工作，一个故障经过层层传递，会波及到调用链上与此有关的所有服务，造成雪崩效应。如何微服务架构设计中没有处理好服务级联依赖中的雪崩效应，那么服务化程度越高，整个系统反而越不稳定。

围绕应用架构所面临的稳定性问题，**我们一般将流量控制和服务容错统称为系统容错设计，除此以外，还需要度量机制和演练方案形成一个完整的体系闭环的高可用架构解决方案**，通过这些措施相互关联配合，共同增强应用的高可用架构能力。

2. 度量机制和可观测

相对于传统的单机系统，在分布式环境下如果要回答清楚服务出了什么问题、哪些服务质量未达标、故障的影响范围、以及变更对服务指标产生的影响等信息，就需要对整条业务链路上多个服务或组件上的信息做收集和关联，要求系统具备更强的可观测能力。

这种可观测能力要求能够在分布式系统中，主动通过日志、链路跟踪和度量等手段，让一次业务请求背后的多次服务调用的耗时、返回值和参数都能够清晰可见。也可以下钻到每次三方软件调用、SQL 请求、网络响应等，这样才能够掌握系统的实时运行情况，并结合多个维度的数据指标进行关联分析，不断对业务健康度和用户体验进行数字化衡量和持续优化。

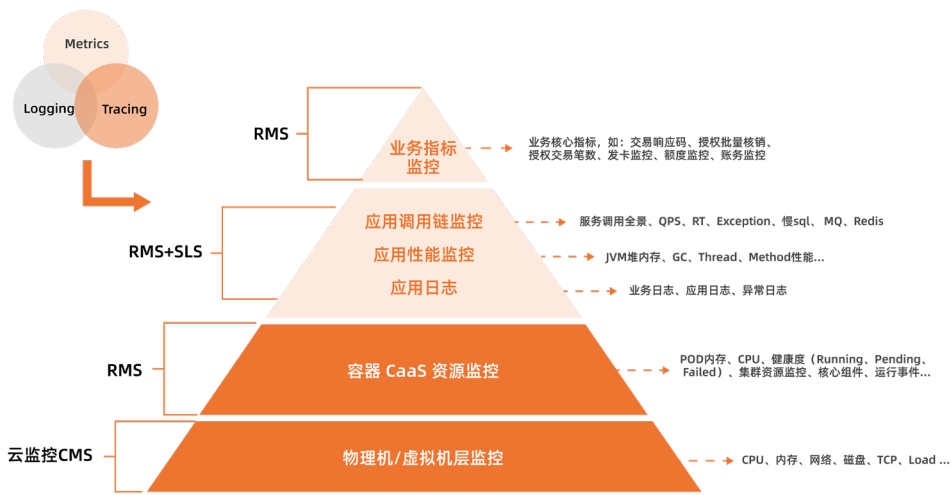


图 1 可观测能力大图示例

可观测能力的建设需要关注以下三个方面：

- **日志（Logging）**：日志的主要目的是记录离散事件，然后通过记录事后分析程序的行为，譬如曾经调用过的方法，操作过的数据等等。输出日志的关键点是格式标准化，以便于后续的收集和分析。在大规模复杂系统环境下，会面对成千上万的集群节点，面对迅速滚动的事件信息，面对数以 TB 计算的文本记录，这里需要解决传输、归集和海量数据存储以及查询的挑战。
- **追踪（Tracing）**：追踪的主要目的是排查故障，在单体系统中基于本机的栈追踪很容易做到，但在微服务系统中，一个外部请求需要内部若干服务的联动响应，这时候完整的调用轨迹将跨越多个服务，同时包括服务间的网络传输信息与各个服务内部的调用堆栈信息，因此，针对分布式系统中的“全链路追踪”能力就尤为重要了。

- **度量（Metrics）**：度量的主要目的是进行监控和预警，主要通过对系统中某一类信息的统计聚合来实现，比如在 java 中由虚拟机直接提供的 JMX 就可以对诸如内存大小、各分代的用量、峰值的线程数、垃圾收集的吞吐量、频率等指票进行度量。如某些度量指标达到风险阈值时触发事件，进行自动处理或者提醒管理员介入。

除了基础的可观测能力的建设外，还需要重点关注的就是观测和度量目标的选择，这里面的关键点是要从与业务价值和用户最关心的方面入手来选择目标，然后再反向推导出具体应观测的指标，而不能仅仅从可以简单度量的指标入手。所以比较理想的方式是，根据业务价值自上而下的设计可观测体系，从**业务核心指标、应用性能、链路追踪监控到系统资源监控**，可推导可下钻，使最终的 SLO 能够更好的发挥作用。

3. 高可用演练方案

在具备可观测的基础之上，我们需要能够尽早地识别出系统中脆弱的、易出故障的环节，同时有针对性地继续进行加固、防范。在应用架构领域，需要定期开展高可用演练，以确保高可用架构的持续有效性。

（1）故障演练的实践理论

- 随着分布式系统的逐步演进和发展，我们会面临着各种类型的问题，例如：
- 分布式系统日益庞大，很难评估单个故障对整个系统的影响。
 - 服务间的依赖错综复杂，隐藏很多配置不合理的问题。
 - 请求链路长，监控告警日志记录容易有遗漏。
 - 业务和技术迭代速度快，引入不确定性风险。
 - 基础设施的老化产生越来越多的隐患。
 - 系统中的人员角色众多、分工协作复杂以及人员变动频繁带来的对系统整体认知程度下降。
 -
- 参考混沌工程的实践思路。故障演练的实践原则主要是：

- **建立一个围绕稳定状态行为的假说**
 - 关注业务状态可测量输出，而不是系统内部属性。
 - 短时间内的度量结果，代表了系统的稳定状态。
 - 验证系统是否工作，而不是如何工作。
- **基于多样的真实世界事件的进行验证**
 - 任何能够破坏稳态的事件都是混沌实验的一个潜在变量。
 - 通过潜在影响或预估频率确定事件的优先级。
- **在无限接近生产的环境中运行实验**
 - 系统的行为会根据环境和流量模式而变化。
 - 为了保证系统行为的真实性与当前部署系统的相关性，混沌工程建议在尽量接近生产的环境（性能环境、仿真环境、等比环境、预发准生产环境等）中进行实验。
- **持续自动化运行实验**
 - 手工运行实验是不可持续的工作，所以需要把实验变为自动化且持续的执行。
- **控制最小化爆炸半径**
 - 在生产环境中进行实验可能会引发真实的故障发生，所以在执行试验时需要确保影响范围最小化且可控。

（2）故障模型的设计

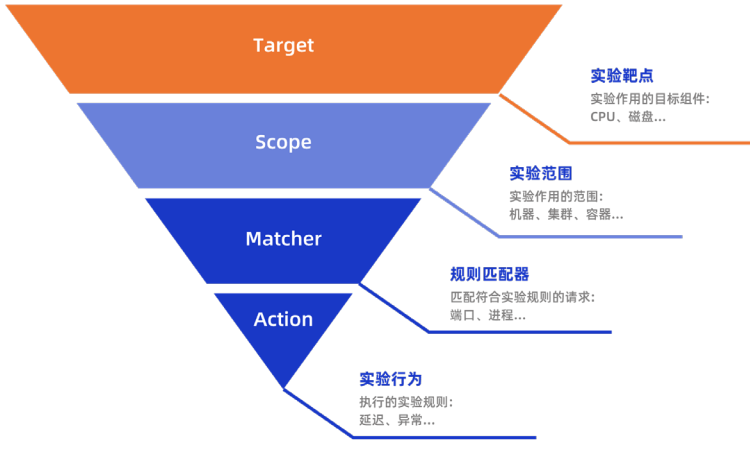


图 2 故障模型设计示例

在故障模型的设计中，我们可以通过下面几个维度来进行定义：

- **实验靶点**。实验作用的目标组件：CPU、磁盘……
- **实验范围**。实验作用的范围：机器、集群、容器……
- **规则匹配**。匹配符合实验规则的请求：端口、进程……
- **实验行为**。执行的实验规则：延迟、异常……

(3) 故障演练的范围及验证目标

故障演练的类型范围以及验证目标，需要根据业务系统的实际情况来制定，下面列举一些基础和通用场景：

① 演练的故障类型范围

- 变更类故障
 - 业务变更故障：代码变更 / 配置变更 / 开关预案变更
 - 基础设施变更故障：流量调度变更 / 数据存储变更
- 异常类故障
 - 系统资源类
 - 业务类
 - 依赖服务类
 - 中间件、存储类
- 基础设施类故障
 - 物理机、容器故障
 - 网络通讯故障
 - 运维系统故障

② 演练的验证目标

- 业务功能异常处理
- 监控告警覆盖度
- 失败重试策略有效性
- 降级熔断策略有效性
- 异常隔离与流量调度
- 自动化编排部署
- 自动化水平扩容

(4) 故障演练的实施流程



图 3 故障演练实施流程示例

故障演练的实施流程主要涉及**五大步骤**，包括**创建演练**（剧本定义）、**开始演练**（准备阶段 & 执行阶段）、**演练度量**（检查阶段）、**停止演练**（恢复阶段）、**演练复盘**。下面将结合阿里云故障演练平台通过一个具体的演练场景详细阐述实施流程：

(4.1) 创建演练（剧本定义）

a. 剧本定义

演练场景	故障场景	预期结果
单台机器 CPU 负载超过 60%，应用自动弹性扩容。	故障范围：单台机器 故障类型：CPU 满载	架构：自动弹性扩容，1 分钟就绪 监控：可观测 CPU 负载指标 预案：手动扩容

表 1 剧本定义参考

b. 创建演练

- 纳管机器资源，即作为即将演练的目标。
- 选择机器，即设置故障范围。
- 添加演练内容，即设置故障类型。



图 4 创建演练示例图

(4.2) 开始演练（准备阶段 & 执行阶段）

a. 准备阶段

- 确认环境
- 开启流量
- 确认监控

b. 执行阶段

- 执行故障注入



图 5 执行故障注入示例图

(4.3) 演练度量（检查阶段）

a. 检查阶段

- 监控
 - 确认监控是否有效，例如 CPU 负载指标是否已达到 60%
- 预案
 - 确认预案是否有效，例如自动弹性扩容，1 分钟就绪。

b. 演练度量

- 阿里云故障演练平台集成 Prometheus 监控，演练过程中即可实时观察监控指标。

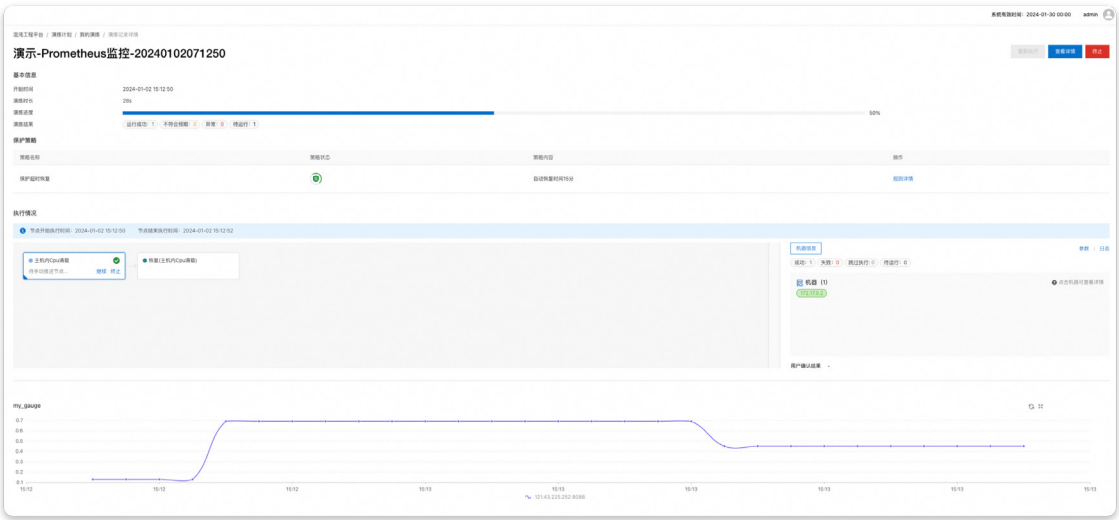


图 6 演练过程监控指标观测示例图

(4.4) 停止演练（恢复阶段）

a. 恢复阶段

- 恢复故障

b. 停止演练

- 设置自动恢复时间，例如 15 分钟，超过 15 分钟，即刻停止演练。
- 增加守护策略，设置阈值，超过阈值即停止演练。

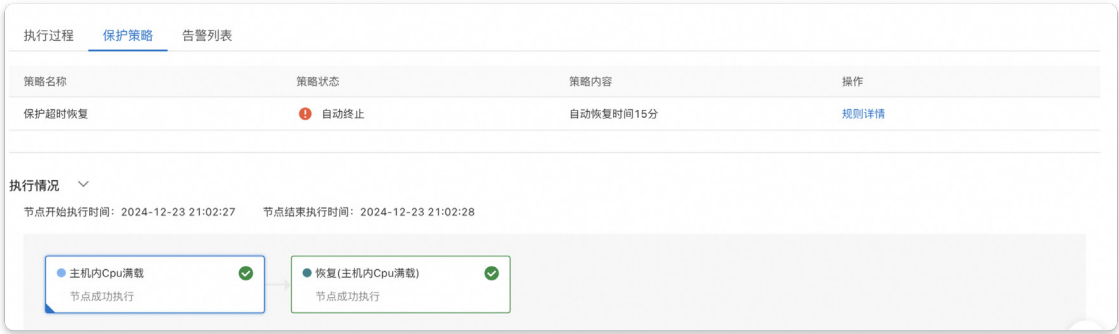


图 7 保护策略示例图

(4.5) 演练复盘

a. 复盘过程

- 演练是否符合流程
- 故障效果是否符合预期

b. 复盘结果

- **架构**：是否符合预期，如不符合预期，例如上述弹性架构，如未能自动弹性，则需进行架构优化。
- **监控**：是否符合预期，如不符合预期，例如上述 CPU 监控指标，如未配置 CPU 监控或监控展示不正确，则需进行监控补齐。
- **预案**：是否符合预期，如不符合预期，例如上述手动扩容方案，如手动扩容失败，则需进行预案固化。

4. 小结

总之，通过合理的面向失败设计、完善的度量机制和定期的高可用演练，企业可以构建更加稳健的云上应用架构，确保业务连续性和服务质量。

(二) 系统容量规划设计

1. 容量规划的定义和思路

如果把容量规划比作是一个天平，那么一端是成本，我们需要尽可能地用较少的资源来支撑好我们的业务；另一端则是稳定性，在成本尽可能低的情况下，各个系统都能运行在一个合适的水位，既能保障业务的正常运转，又不出现局部的资源浪费。

我们这里的容量规划更多的是业务视角下对于云平台资源的管理和有序整合。如果单纯从云平台去看容量规划，或者进行较为粗糙的业务容量规划下，会造成大量的资源浪费。同时，也会引入大量的不确定性，影响最终用户体验。

容量规划需要结合时间的维度去看，所以针对服务容量需求的预测和规划就要求能够保障一个业务有足够的容量和冗余度去服务预测中的未来需求。同时，一个业务的容量规划，不仅仅要包括自然增长（用户使用和资源用量正比增长），也需要包括一些非自然增长的因素，比如新功能的发布、商业推广（大促），以及其他商业因素。服务容量规划思路，我们需要从依赖关系、性能指标和优先级等几个方面来进行综合考虑：

• 依赖关系

因为一个应用服务在运行时往往需要依赖很多其他基础设施和相关服务，所以部署位置的选择要受到相关依赖服务的可用信息影响。假设我们有一个服务 A，依赖某个基础设施存储服务 B，A 要求 B 服务必须处于 30ms 的网络延迟范围之内。这就对 A 和 B 的位置提出了要求，资源规划过程必须将这条限制纳入考虑范围。另外，通常服务依赖都是嵌套的，设服务 B 要依赖一个底层的分布式存储系统 C，和一个查询处理系统 D。我们需要考虑 B、C、D 来决定在哪里放置 A。在某些情况下，不同系统可能依赖同一个系统，但是服务目标和要求不同。

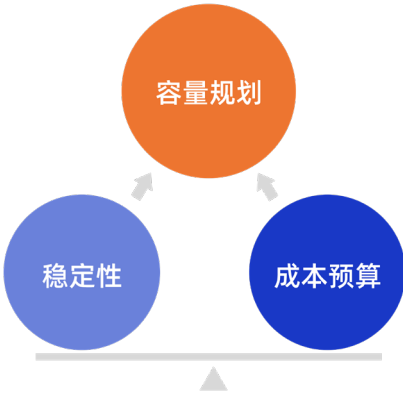


图 1 容量规划“天平”

- 性能指标

性能指标是依赖关系中的黏合剂。在上面的场景中，A 服务需要多少计算资源来服务 N 个用户需求？我们需要 B 服务提供多少 Mb/s 的数据来服务 N 个 A 服务的用户请求？通过性能指标，我们将一种或多种高阶资源类型转换成低阶资源类型，并通过对整个依赖链的梳理帮助我们规划出大致的需求优化计划，性能指标通常需要负载测试和资源用量监控来获得。

- 优先级

服务资源不足是在某些情况必须面临的问题，在这种情况下，需要考虑哪些资源请求可以被牺牲掉？比如：可能 A 服务的 N+2 冗余度要比 B 服务的 N+1 冗余度更重要，或者某个功能 X 的上线没有 B 服务的 N+1 需求重要（意思是必须保证 B 服务有足够容量）。规划过程强制使这些计划变得更为透明、开放和一致，同时让资源使用限制的妥协和规划，对服务相关人员更加透明。

2. 容量规划的核心要素

影响容量规划的核心要素有三个：压测环境选型、压测基础数据、压测流量模型。

- 压测环境选型

如果是即将投产的云，由于这是后续的生产环境，配合等量的基础数据，可以比较方便地进行复杂的混合读写压测，同时得到的性能指标和容量水位具备很高的参考价值。

通常，可以将真实的生产环境数据进行脱敏或者提取业务接口的分布模型，确保业务数据量（交易数据、流水、各种业务核心业务记录等）维持在一定的时间基础之上（比如半年以上），同时确保数据的关联完整性。压测脚本基于这些基础数据进行相应的核心业务的流量（登录、交易行为等）构建，最后在投产前做相应的数据清理再初始化一次存量基础数据。

所以，生产环境从容量规划的角度看一定是首选，毕竟上述的窗口是很少的。我们这个时候再看另一个环境选型：等比仿真 / 性能环境，一般是指在生产环境单独划分区域，准备等比的容量，共享或者对等部署接入层的性能测试环境。这种方案缺点很明显，成本较高，优点是方案简单、风险可控，容量规划较为精准。

关于上文中重点提到的基础数据，基本分为两种方式，一种是数据库层面不需要做改造，直接基于基础表里的测试账户进行，压测之后将相关的测试产生的流水数据清除；另一种就是压测流量单独打标染色，然后业务处理过程中识别这个标并传递下去，包括异步消息和中间件，最终落到数据库的影子表或者影子库中，此方案成本和风险高，需要结合相关的技术投入以及适合的重大场景去谨慎选择。

- 压测基础数据

数据量主要包括基础数据量（或者叫历史数据量、垫底数据量、数据库中已有的数据量）和参数化数据量，数据量在性能测试中起到非常重要的作用。对于在数据库中只有几条记录和有几亿条记录里面查询信息，那么结果肯定相差非常大的。随着业务量的增长，记录也越来越多，因此使用仿真 / 性能测试环境时，需要保持跟生产上相同级别或者固定比例的数据量。如果采用在生产环境中插入测试账户的方式，可以一定程度解决环境真实性和基础数据量同量级的问题。最后，我们在测试的时候需要考虑参数数据量的大小和数据分布的问题。

如果基础数据量跟生产环境的基础数据量不在同一个数量级上，将会导致相关指标例如响应时间比生产上快很多，不真实，甚至导致测试结果没有参考意义。如果参数化数据量过少、未考虑数据分布的情况，将会导致测试结果不真实，甚至测试结果没有参考意义。生产环境中插入测试账户的方式，需要考虑数据准备的完整性问题，还有清理的逻辑需要完整。全链路压测的方式需要投入较大的改造成本，同时包括后续的持续迭代维护。

下面，关于几个细节举例说明下。比如基础数据量，如果是测试环境，基础数据量需要跟生产环境基础数据量保持在同一个数据量级上，一般情况下需要考虑未来三年数据量增长趋势。如果增长过快需要在测试环境造非常多的数据。还有，比如参数化数据量，需要尽可能的多，必要的情况下，可以清除缓存或者用写代码的方式提供参数化，另外在参数化数据分布上，如果业务有明显的地域等分布的特征，需要考虑数据分布的情况。

- 压测流量模型

由于不同业务的业务逻辑和业务量不尽相同，消耗系统的资源也不一样，因此业务种类、业务占比决定了系统的处理能力，业务模型在性能测试中起着关键性的作用。以电商场景为例，不同的促销形式和主推的类目决定了不同的容量整体配比，那么精准地将流量落地在压测工具上，通过压测拿到系统的木桶最短板可以更好的利用机器资源达到业务目的。

业务模型中业务和占比选取不对，跟生产环境差异非常大，直接导致测试结果没有任何参考价值，并且容易误导对系统处理能力的判断。有些业务的业务量虽然占比很低，但一旦突变，对系统也是致命的，这些业务在性能测试中也需要关注。

在系统中选取典型业务时，通常遵循以下原则：优先选择业务量大、使用频率高、存在潜在风险以及未来具有增长潜力的业务作为系统的代表性业务。已经上线的系统可以通过高峰时段历史业务量和生产问题性能来评估，对于即将上线的系统可以通过和相关业务需求方调研并结合类似系统的经验来综合评估。

① 已上线系统

- 搜集生产上不同高峰时间段的业务种类和业务量，每个时间段的业务种类和业务量是否有很大的差异，如有的话，必须有多个业务模型，差异不大的，可以只用一个业务模型。
- 搜集生产上高峰时间段资源消耗和资源异常的时间点，从中捕获资源消耗高和异常的原因，可能是由于某种“不起眼”的业务导致。
- 搜集生产问题，进行分析，如果是由于某种业务导致而且以前性能测试的时候忽略此笔业务，那么这笔业务的风险是非常大的，需要后续性能测试将此业务加入到业务模型中。

② 未上线系统

- 通过调研，确定业务种类和业务占比。
- 通过调研，确定是否在业务促销等活动中，某些业务有突变的可能。
- 通过测试结果，确定每笔业务的资源消耗，如果某些业务虽然占比低，但资源消耗非常大，那么需要适当的调整此业务占比。

3. 容量规划的几种方式

首先，我们讨论的是容量规划，因此所基于的环境通常是生产环境或与生产等比例仿真的环境，以确保能够进行可靠推算的环境条件。线下测试环境、性能环境更多的会侧重在性能瓶颈和稳定性隐患的发现，而我们讨论的容量规划更多是围绕精准度与确定性，以及各种模式背后的效率和成本。

容量规划大体分为 5 种模式，日常流量分析、线上流量模拟压测、线上流量复制仿真、线上单机引流压测和全链路压测。前 4 种模式都是需要做一定的推算，也有一定的估算性质，最后一种全链路压测相对而言是精准的，成本代价也是最大的。我们需要结合不同的场景做最适合的选择。

日常流量分析的模式基本是完全依赖估算，根据日常峰值及对应的 CPU 使用率（核心的负载指标）去估算极限负载情况下的单机最高峰值，进而推算整体的容量水位，其局限性很明显，单机的拐点以及集群背后依赖的性能拐点均没有考虑，这个方式可参考性非常低，只能在某些特定场景下才适用。

线上流量模拟压测和线上流量复制仿真，前者是人为构建一定模型的压测流量叠加到集群整体，后者是根据真实的一些流量情况进行复制（比如 accesslog），拿到单机的峰值进而推算整体峰值。线上流量复制仿真的模式可以考虑回放的时间加快，其效果就是增大了压测流量，放大了量级，然后这两种方式如果要涉及到写的压测，还是如上文所述，要么考虑在生产库准备测试的基础数据，要么要有流量识别和影子库表体系了。

线上单机引流压测是阿里集团内部最为常见和高性价比的容量规划的一种测试方式了，通过局部通过硬负载（如 SLB）或者软负载（如内部 RPC 调用）引流到单机造成单机的流量放大，从而得到一个精准的线上单机的性能容量上限，但是它的局限性是类似数据库等存储层的瓶颈是不会提前暴露的，故得到的集群容量估值也会偏高。

最后一种模式是全链路压测模式，其主要是通过大量的技术改造和组织协同，针对于关键窗口的关键业务形态进行针对性的容量规划，相比于传统方式而言能提供足够高的精准性及确定性，即在生产环境进行高仿真模拟，测试系统端到端的整体性能情况。传统方式主要的问题是：

- ① 关键业务活动窗口时，整条链路各个环节表现有很大的不确定性；
- ② 现有的容量规划是基于单个应用维度去评估（如上述 4 种模式），均假定下游能力无限，存在误差，容易形成误差堆积；
- ③ 压测的测试数据不足，不够支撑足够规模的业务模型压测需求，同时数据量不够，无法检验数据库及存储等最大的瓶颈隐患；
- ④ 用于评估的测试环境或仿真环境本身与生产环境存在整体差异，造成容量评估误差。

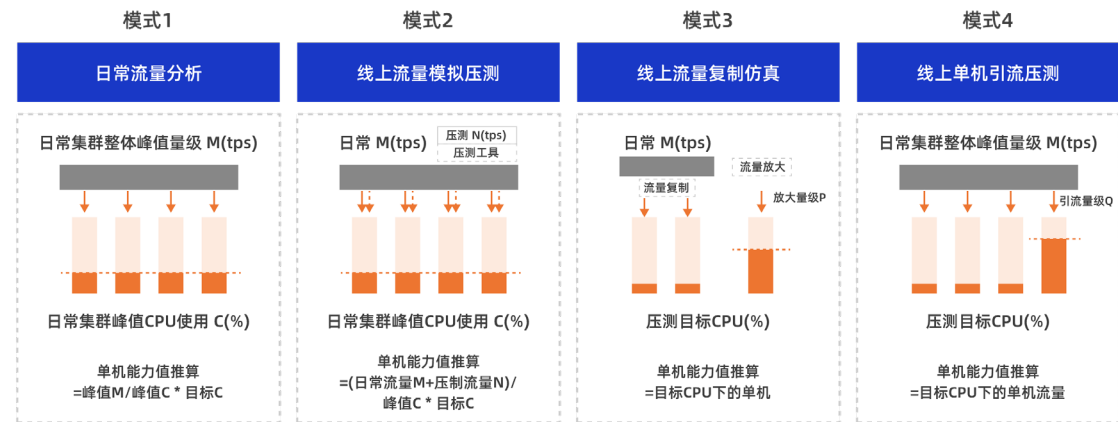


图 2 容量规划常见的 4 种模式

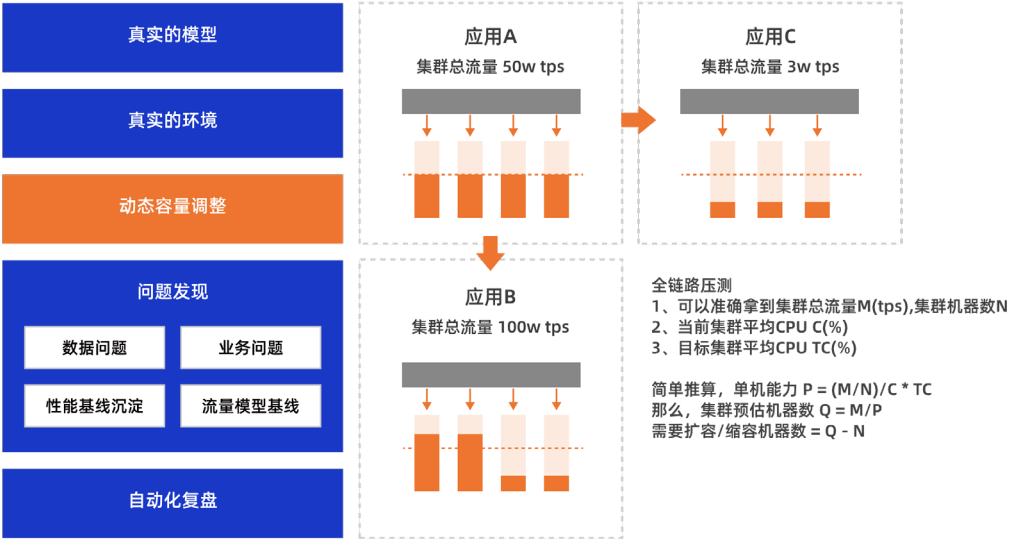


图 3 全链路压测模式示例

由于全链路压测具备了真实的生产环境、等量级的基础数据（如用户规模）以及预估设计好的流量洪峰，故在这个流量模型下的全局容量是具备参考性的，也会围绕这个模型进行限流和降级策略的落地，这也是技术架构对业务承诺的一个业务吞吐模型，超出部分如何处理，规划内部响应时间如何，峰值的几分钟的整体流量最终如何消化等等。

4. 小结

关于容量规划，我们还有几个小的注意事项供参考：

- ① **设定验收标准**：容量规划和压测都需要提前设定一个验收标准，不需要压测到某种极限，比如提前确定系统上限（例如：CPU 60%，RT 500ms 以内，成功率 99.9%）。
- ② **容量规划结合限流措施**：光有容量规划并不够，因为容量规划是基于确定的一个业务流量模型，如果实际情况不符，那么就需要搭配限流来保障各接口上限不被冲垮。
- ③ **业务预热**：重要的业务窗口，系统需要提前进行业务的预热，必要的生产环境封网管控。

（三）系统容错设计

1. 系统容错策略与模式

容错性设计源自于分布式系统的本质是不可靠的，为了避免服务集群中出现任何差错导致的系统崩溃风险，面向失败的服务容错设计就成为必要的措施。我们认为需要从容错策略和容错模式上进行重点设计。

（1）容错策略

容错策略指的是当出现故障时我们该做些什么（what），常见的容错策略有以下几种：

- **故障转移**：高可用的服务集群中服务均会部署有多个副本。故障转移是指如果调用的服务器出现故障时系统会自动切换到其他服务副本，从而保证了整体的高可用性。
- **快速失败**：在不允许做故障转移的业务场景下（服务要求具备幂等性），此时就应该以快速失败作为容错策略。就是尽快让服务报错，避免重试，尽快抛出异常，由调用者自行处理。
- **安全失败**：在一个调用链路中的服务通常也有强依赖和弱依赖之分，对于失败了也不影响核心业务的弱依赖逻辑，容错策略可以在调用失败的情况下自动返回一个符合要求的数据类型的对应零值，然后自动记录一条服务调用出错的日志备查即可，这种策略被称为安全失败。
- **沉默失败**：如果请求需要等到超时才宣告失败，很容易由于堆积而消耗大量的资源，进而影响到整个系统的稳定。这种情况下的失败策略是当请求失败后，就默认服务提供者一定时间内无法再对外提供服务，将错误隔离开来，避免对系统其他部分产生影响，此即为沉默失败策略。
- **故障恢复**：是指当服务调用出错了以后，将该次调用失败的信息存入一个消息队列中，然后由系统自动开始异步重试调用。故障恢复也是要求服务必须具备幂等性的，为了避免在内存中异步调用任务堆积，也应该有最大重试次数的限制。

（2）容错模式

容错模式指的是当出现故障时我们该如何去做（how），常见的容错模型有以下几种：

- **断路器模式**
断路器就是通过代理接管服务调用者的远程请求，持续监控并统计服务返回的成功、失败、超时、拒绝等各种结果，当达到断路器的阈值时，它状态就自动变为“OPEN”，后续此断路器

代理的远程访问都将直接返回调用失败，通过断路器对远程服务的熔断，避免雪崩效应的出现。

断路器本质是一种快速失败策略的实现方式。

• 主动服务降级模式

这种容错模式不是在出现错误后才被动执行的，许多场景里面是指需要主动迫使服务进入降级逻辑的情况。例如，出于应对可预见的峰值流量，或者是系统维护等原因，要关闭系统部分功能或关闭部分弱依赖服务，这时候就有可能主动进行这些服务降级。除了容错外，服务降级也可作为流量控制的措施和手段。

• 隔离模式

隔离模式符合容错策略中沉默失败策略，调用外部服务的故障中“超时”引起的故障尤其容易给调用者带来全局性的风险。这是由于目前主流的网络访问大多是基于 TPR 并发模型，请求会一直占用着某个线程不能释放，而线程是整个系统的全局性资源，为了不让局部失败演变成全局性的影响，就必须设置种止损方案，常见的实现方式有线程池隔离和信号量机制。

• 重试模式

故障转移和故障恢复策略都需要对服务进行重复调用，差别是这些重复调用是同步的还是异步的；是重复调用同一个服务还是会调用到服务的其他副本。这些都重试设计模式的应用范畴，可以遵循一些共同的规范。

仅在强依赖逻辑的关键服务上进行同步的重试，不是关键的服务一般不做重试。不要对致命错误重试，只对瞬时故障导致故障进行重试。

- 重试的服务必须具备幂等性。
- 重试必须有明确的终止条件，常用的终止条件有两种：
 - 超时终止：重试模式需要配合超时机制来使用，避免重试的副作用。
 - 次数终止：重试必须要有次数限制。重试不仅会给调用者带来负担，对于服务提供者也是同样是负担。

2. 系统容错方法与技巧

对业务系统来说，实现高可用的前提，就是尽可能的把业务逻辑独立封装，做到无状态化。这样就可以充分利用接入层的流量调度能力、基础服务（中间件）层的服务治理能力以及底层 PaaS 平台的弹性能力来实现业务层的高可用。

（1）无状态 & 冗余部署

业务无状态化后就可以轻松的获得服务冗余和水平扩展的能力，在部署架构上需要根据业务容灾需求进行跨可用区，跨 Region，甚至跨地域来进行冗余部署。对于有状态类应用，建议将这部分功能下沉到基础服务（中间件）层来做，利用基础服务（中间件）层能力来完成类似于 Session，分布式缓存一类带状态的功能，这样业务层就可以不用关注状态数据的同步、一致性、冗余备份等非业务问题了。至于业务数据的持久化，可以交给数据层的服务来处理，同样，这部分功能的高可用保障，也不用业务层来负责了，使业务层应用可以专注业务逻辑本身的开发。

但在做业务层的技术架构设计时，需要考虑上下游依赖的服务（接入层的服务和中间件层的服务）的部署架构、服务能力是否能够满足业务场景的容灾需求。

（2）防护能力（限流、降级）

除了无状态化外，业务层另外需要关注的就是服务的自我保护能力，这里面包括突发流量的处理和依赖服务引发的关联故障处理，避免因为这些原因造成自身服务的不可用。尤其是在复杂的分布式环境中，如果应用不具备防护能力，很容易因为整条业务链中一个点的失效，引发多米诺骨牌效应迅速打垮整个集群，产生服务雪崩情况。

服务的防护能力在云原生架构中可以做到 Sidecar 里面，从应用中彻底剥离。但目前最多的方式还是引入 SDK，依赖第三方中间件的形式来实现，本文默认以这种模式为基础，把这一块放在业务层来介绍。

流量控制，对业务来说最重要的两点就是资源的定义和规则的制定。所谓资源的定义，就是把外部依赖的服务在流量控制系统中以资源的形式明确定义出来，然后以资源的维度进行相应的流量控制和失败保护。规则定义方面，主要需要关注几点，首先是流控的指标，比如 QPS、并发线程数、单位时间内的异常调用比例等，然后是流控的方式，比如快速失败、Warm Up、排队等待等。

具体的隔离模式上，通常有线程池隔离和信号量隔离，在利用线程池隔离时需要关注的点就是应用总的线程数，过多线程的上下文切换会产生较大的 overhead，信号量隔离需要注意慢调用的级联阻塞问题。

由于针对应用级的防护能力通常是做在单点上的，这样如果涉及到全局热点数据问题，就需要额外的单机数据采集 + 数据聚合 + 动态规则配置推送的过程，这就需要依赖中间件层的能力，阿里巴巴 Spring Cloud Sentinel（已开源）很好的解决了这个问题。

业务降级，这里面要说的并不是当依赖的外部系统出现异常时的 Fast Fail 处理，而是当系统的容量达到一定程度的水位时，限制或者关闭某些非核心功能，这就是所谓的“业务降级”，从而把有限的资源保留给更核心的业务。这是一个有目的、有计划的执行过程，所以对降级我们一般需要有一套预案来配合执行。如果我们把它系统化，就可以通过预案系统和开关系统来实现降级。

降级方案可以这样设计：当秒杀流量达到 5w/s 时，把成交记录的获取从展示 20 条降级到只展示 5 条。“从 20 改到 5”这个操作由一个开关来实现，也就是设置一个能够从开关系统动态获取的系统参数。

此外，业务降级方案还可以通过降低一致性要求、暂停次要功能等方式来实现。

3. 系统容错的实践

在实际场景中，即便我们已经对系统容量进行了全面的评估与规划，对整套架构的业务吞吐能力有了清晰的预估，仍然需要制定完善的兜底保障措施。尤其是在系统资源不足以应对超出预期的突发流量时，必须明确优先级并进行合理的取舍。结合前文提到的限流、降级、隔离和熔断等容错方案，构建起一套面向超额流量的自我保护机制。这正是微服务架构中流量控制的核心所在，即通过限流等手段确保系统在高负载情况下的稳定性和可用性。

（1）流量统计指标

要做流量控制，首先需要了解哪些指标能反映系统的流量压力大小。我们先来理清经常用于衡量服务流量压力的三个指标的定义：

- **每秒事务数（TPS）**：TPS 是衡量信息系统吞吐量的最终标准。“事务”可以理解为一个逻辑上具备原子性的业务操作。
- **每秒请求数（HPS）**：HPS 是指每秒从客户端发向服务端的请求数。如果只要一个请求就能完成一笔业务，那 HPS 与 TPS 是等价的，但有些场景里一笔业务可能需要多次请求才能完成。
- **每秒查询数（QPS）**：QPS 是指一台服务器能够响应的查询次数。如果只有一台服务器来应答请求，那 QPS 和 HPS 是等价的，但在分布式系统中，一个请求的响应往往要由后台多个服务节点共同协作来完成，这里面每次访问都要消耗掉一次或多次查询数。

以上这三个指标都是基于调用计数的指标，理想上是能够基于 TPS 来限流。但是由于不同的业务操作对系统的压力差异较大通常不具备可比性；并且真实业务操作的耗时还可能受限于用户交互带来的不确定性。所以直接针对 TPS 来限流实际上不容易操作的。而使用 QPS/HPS 作

为限流指标比较常见，它相对容易观察统计而且能够在一定程度上反应系统当前以及接下来一段时间的压力。

基于调用计数的限流指标在实际应用中，需要从流量全链路角度关注下面几点因素：

- SLB/DNS 流量不均。
- 链路流量有损，后端应用服务能力无法最大化。
- 链路流量调用关系非等比，分布式服务中有短板。

由此，便形成了一个流量控制的漏斗模型。为了构建完整的流量控制策略，需要整合接入层、应用网关以及微服务等多层级的限流策略与模型，同时结合相关的服务容错保护机制，从而实现对流量的精细化管理和全面防护。

（2）限流模式设计

限流模式设计需要重点关注 QPS 计算的平滑度和准确性。

• 流量计数器模式

限流最基础的方法就是根据当前时刻的流量计数结果是否超过阈值来决定是否限流，这种直观的方法经常被一些简单的限流系统所使用，但它并不严谨，主要缺陷根源在于它只是针对时间点进行离散的统计。

• 漏桶模式

漏桶模式是参考计算机网络中的流量整形的理念，当网络报文的发送速度过快时，首先在缓冲池中暂存，然后再在控制算法的调节下均匀地发送这些被缓冲的报文。常用的控制算法有漏桶算法和令牌桶算法。

- **漏桶算法。最大限度保证流量的平滑度，有突增流量，且不连续时，可能出现提前限流。**
 - 应对请求到达速率不定的情况。
 - 漏桶以恒定速率通过请求。
 - 超过漏桶容量即丢弃请求。
- **令牌桶算法。面对突发流量时，最大限度允许系统根据的消费速度处理请求，能力最大化，令牌桶消耗完时，控制速率。**
 - 应对请求到达速率不定的情况。
 - 以恒定速率下放令牌到令牌桶，超过容量，丢弃令牌。
 - 从令牌桶中以不定速率拿令牌，拿到令牌则通过。
 - 未拿到令牌则丢弃

• 滑动时间窗模式

滑动窗口算法在计算机科学很多领域中都有着广泛的应用，像 TCP 协议的中的流量控制、服务容错中对服务响应结果的统计、流量控制中对服务请求数量的统计等等。滑动时间窗口模式的限流解决了流量计数器的缺陷，可以保证任意时间片段内，只需经过简单的调用计数比较，就能控制住请求次数一定不会超过限流的阈值。不过，虽然能滑动窗口模式很好的保证流量的平滑度，但有突增流量且不连续时，还是有可能出现提前限流。

(3) 分布式限流

前面讨论过的那些限流算法，主要是使用在单机场景的，但在微服务架构下，无法细粒度地管理流量在内部微服务节点中的流转情况。所以，我们把前面介绍的限流模式都统称为单机限流，把能够精细控制分布式集群中每个服务消耗量的限流算法称为分布式限流。

分布式限流面对的一些特殊场景：

- 限流精度要求高、流量不均导致单机流控效果不佳。
- 接口阈值小于应用机器数。
- 应用机器数经常变动，单接口需要设定恒定阈值（如应用网关）。

分布式限流的核心差别在于如何管理限流的统计指标，由于分布式限流的目的就是要让各个服务节点的协同限流，所以需要将原本在每个服务节点自己内存当中的统计数据给开放出来，让全局的限流服务可以访问到。常规方案可以通过分布式缓存或者专门的分布式限流服务，来解决这些数据的读写访问时并发控制的问题，但代价是每次服务调用都必须额外增加一次网络开销，效率比较低，流量压力大时，限流本身可能会限制系统的处理能力。

(4) 典型应用场景

• 削峰填谷

在实际应用中，由于固定时间内收到的请求是分布不均匀的。例如，某应用的处理请求的能力是每秒 10 个，在某一秒突然到来了 30 个请求，而接下来两秒均没有请求到达。在这种情况下，如果直接拒绝 20 个请求，应用在接下来的两秒就会空闲。这个时候应该把所有的请求平均到一段时间内，让系统负载保持在请求处理水位之内，同时尽可能地处理更多请求。当消费端请求骤增时，可以为其配置排队等待的流控规则，以稳定的速度逐步处理这些请求，起到“削峰填谷”的效果，从而避免流量骤增造成系统负载过高。

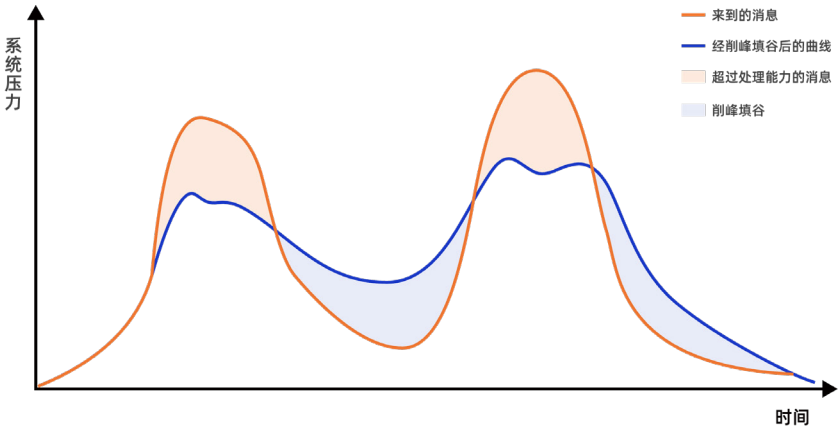


图 1 “削峰填谷” 应用场景示例

AHAS 流控降级的排队等待功能，可以把骤增的大量请求匀速分配，以固定的间隔时间让请求通过，起到“削峰填谷”的效果，从而避免流量骤增造成系统负载过高的情况。堆积的请求将会被排队处理，而当请求的预计排队时间超过最大超时时长时，AHAS 将拒绝这部分超时的请求。例如，配置匀速模式下请求 QPS 为 5，则每 200 ms 处理一条请求，多余的处理任务将排队；同时设置了超时时间为 5s，则预计排队时长超过 5s 的处理任务将被直接拒绝。

• 冷启动

对于长期处于低水位状态的系统，可以使用冷启动功能来避免流量骤增导致水位瞬间升高系统不可用的情况。可以通过配置冷启动规则，让通过的流量缓慢增加，在一定时间内逐渐增加到阈值上限，给冷系统一个预热的时间，避免冷系统被压垮。

在 AHAS 中可以对系统设置流控模式为直接，流控方式为 Warm Up，预热时间为 200 的流控规则。设置规则后，可以看到流量的增长趋势如下图所示：

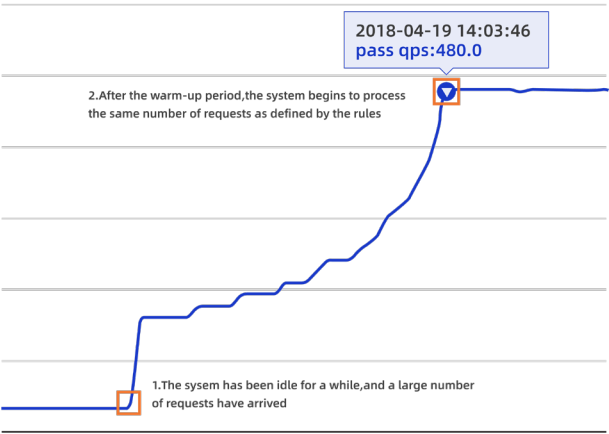


图 2 流量增长趋势图

弱依赖降级

如果依赖的第三方应用出错不会影响整体流程，则称之为弱依赖。我们需要考虑弱依赖不稳定时如何配置降级原则来保护系统稳定性。例如，支付时需要远程调用下游提供的 API，若依赖方出现不稳定的情况，则请求和调用依赖方的方法响应时间变长，线程产生堆积，最终可能耗尽自身的线程数，导致应用本身不可用。在复杂链路中，若某一环不稳定，就可能会层层渲染，最终导致整个链路都不可用。针对以上情况，可以使用 AHAS 应用流控降级功能对依赖方配置降级原则来保证系统稳定性。在为调用端配置基于平均响应时间或错误率的降级规则后，当调用链路中某个下游服务调用的平均响应时间或错误率超过阈值，AHAS 就会对此调用进行降级操作，拒绝多余的调用，保护应用不被调用端短板影响。同时可以配合 Fallback 功能使用，在被降级的时候提供相应的处理逻辑。

弱依赖隔离

如果依赖的第三方应用或组件，或者应用自身的内部方法出错会影响整体流程，则称之为强依赖，同样需要配置隔离原则来保护系统稳定性。具体可以通过配置并发线程数隔离原则来限制不稳定的强依赖并发数，隔离强依赖。配置并发线程数隔离原则后，无需再进行线程池隔离，AHAS 会控制资源的线程数。当请求数超过阈值时，AHAS 将拒绝多余的请求，直到堆积的线程处理完成，以此来达到信号量隔离的效果。线程数目超出时，设置快速失败能够有效地防止自己被慢调用所影响。示例，若有应用 A 的一个内部方法 user_test 不稳定，出现慢 SQL。针对此类场景，可以为调用设置隔离规则（将阈值类型选择为线程数，并将阈值设置为 0）。配置规则后，user_test 方法被限流，各 SQL 调用情况如下图所示。



图 3 限流隔离规则生效示意图

4. 小结

本文探讨了分布式系统中系统容错设计，涵盖常见的 5 种容错策略与 4 种容错模式。同时，系统介绍了系统容错方法与技巧，如无状态部署、冗余部署、限流和降级等方法。另外，通过流量统计、分布式限流及典型应用场景，展示了提升系统稳定性和高可用性的关键措施。

（四）系统容灾设计

从一个计算机系统的角度讲，一切引起系统非正常停机的事件都可以称为灾难。大致可以分成以下四个类型：

- **自然灾害：**包括地震、火灾、洪水、雷电等，这种灾难破坏性大，影响面广。
- **设备故障：**包括主机的 CPU、硬盘等损坏，电源中断以及网络故障等，这类灾难影响范围比较小，破坏性小。
- **人为操作破坏：**包括误操作、人为蓄意破坏。
- **软件故障：**架构不合理导致的功能性故障、性能故障，配置变更故障、安全性故障、数据完整性故障等。

容灾（Disaster Tolerance），就是在上述的灾难发生时，在保证生产系统的数据尽量少丢失的情况下，保持生产系统的业务不间断地运行。异地多活、同城双活都属于容灾的范畴。

容灾需要具备较为完善的数据保护与灾难恢复功能，保证生产中心不能正常工作时数据的完整性及业务的连续性，并在最短时间内由灾备中心接替，恢复业务系统的正常运行，将损失降到最小。

本章节将从容灾技术的发展历程，容灾系统的评价指标，容灾系统的建设方法等方面出发，给大家介绍一下系统容灾技术相关的基础概念，再以目前最先进的异地多活容灾技术为例，给大家做一下容灾架构的设计原理说明。具体相关容灾产品的魂和最佳实践案例，请参考 4.4 容灾多活能力建设章节内容。

1. 容灾技术的发展历程

容灾技术按时间发展维度来看，截止到目前，大体上可以分为下面的几个阶段：

- **第一阶段（1980 – 2000 年）**
IT 作为业务支撑系统，容灾以数据为中心，恢复以人工为主，容灾系统作为备用系统。
- **第二阶段（2000 – 2013 年）**
IT 作为业务的使能，容灾以业务为中心，双活、AQ 模式使得容灾系统支撑部分业务。
- **第三阶段（2013 年 – 至今）**
容灾即业务，容灾以客户为中心，智能流量分配，多中心部署，容灾系统即业务系统。
异地多活目前就是处在第三阶段。

2. 容灾系统的评价指标

容灾系统主要为了在灾难发生时业务不发生中断，那么当灾难发生时，用户最关心的是什么呢？以下是国际通用的容灾系统的评审标准 Share 78，可以作为广大用户衡量和选择容灾解决方案的指标。以下是备份 / 恢复的范围：

- **灾难恢复计划的状态**
- **在应用中心与备份中心之间的距离**
- **应用中心与备份中心之间是如何相互连接的**
- **数据是怎样在两个中心之间传送的**
- **有多少数据被丢失**
- **怎样保证更新的数据在备份中心被更新**
- **备份中心可以开始备份工作的能力**

因此，容灾系统的设计，主要也是围绕这几个用户需求展开。由于用户投入资金的数量限制，想用少的资金达到第 6 级容灾级别显然是有难度的，我们设计出的系统也只能是在现有的条件下尽量减少故障持续时间，尽量多的恢复数据，这也是衡量我们所设计出来的容灾系统质量的指标。实际的容灾系统设计过程中，我们重点关注的是 RPO 和 RTO 两个指标。

- **RPO（Recovery Point Objective）**

即数据恢复点目标，以时间为单位，即在灾难发生时，系统和数据必须恢复的时间点要求。RPO 标志系统能够容忍的最大数据丢失量。系统容忍丢失的数据量越小，RPO 的值越小。

- **RTO（Recovery Time Objective）**

即恢复时间目标，以时间为单位，即在灾难发生后，信息系统或业务功能从停止到必须恢复的时间要求。RTO 标志系统能够容忍的服务停止的最长时间。系统服务的紧迫性要求越高，RTO 的值越小。

RPO 针对的是数据丢失，而 RTO 针对的是服务丢失，RTO 和 RPO 的确定必须在进行风险分析和业务影响分析后根据不同的业务需求确定。

设计良好的容灾系统需要尽量满足用户的需求，但是容灾系统的设计往往受多种条件的制约，如可用的技术、现网状况、用户意志、用户业务等，但到目前为止，起决定性的因素，是容灾建设的成本。以下是容灾中心建设等级示意图。

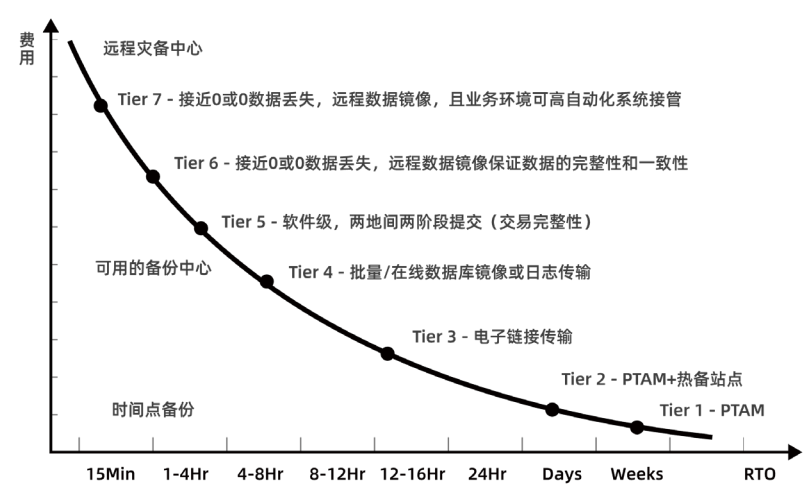


图 1 容灾中心建设等级示意图

3. 容灾系统建设的方法

根据对容灾系统建设模型，容灾系统建设过程分为分析、设计和实施三个阶段。

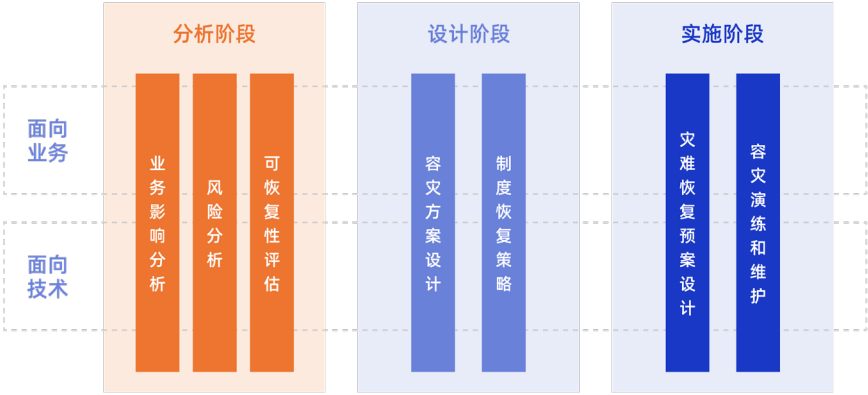


图 2 容灾系统建设过程三阶段

下面分别对各个阶段作出说明：

① 分析阶段

在取得管理层的正式同意后，获得人员和资源的保证。首先收集业务过程的信息、技术基础架构的支撑环境、灾难类型等方面的内容，然后进行业务影响分析和风险分析，确定由于中断和预期灾难可能造成的影响。分析的结果用以确定业务关键级别、业务恢复时间和可承受的数据损失程度。

② 设计阶段

结合以上的分析成果，以及企业对容灾的投入规划，制订企业短期、长期范围内的容灾策略和目标，先定义初步的方案。再进一步结合各种因素进行分析，在候选的方案中剔除不合适的方案，将剩余的可行的方案提交给评估组，评估组经过充分详细的评审，选择最合适的容灾方案。

③ 实施阶段

根据选择的容灾方案，整合企业相关资源，确定容灾的体系架构和灾难恢复计划，通过技术手段和服务以达到所要求的容灾目标。任何制订的计划，都必须经过不断的测试和修正，才能满足企业不断发展的需求。同时，通过培训、测试过程，也能够使企业内部人员熟悉自己在容灾流程中所扮演的角色，保证在灾难真正发生的时刻能够有条不紊地执行恢复流程。测试的过程可以分为局部验证和演习两种方式。随着商业需求、新技术的不断升级以及新的内部和外部规则的变化，IT 系统也会随之改变。要确保灾难恢复计划的有效性，必须定期检查和修改计划。

下面就以异地多活为例，简要介绍一下容灾多活系统的架构原理。

4. 异地多活架构原理

异地多活，英文 Multi-Site High Availability，顾名思义就是分布在异地多个站点同时对外提供服务。与传统的灾备的最主要区别就是“多活”里的所有站点都是同时在对外提供服务的，具体有以下几点不同：

- 传统的灾备中心平时不提供服务，关键时刻无法确定切换到灾备中心是否可以切换成功。
- 传统的灾备中心平时不提供服务，整个灾备资源会处于浪费状态，成本比较高。
- 传统的灾备中心平时不提供服务，所以平时提供服务的数据中心还停留在单地域，当业务体量大到一定程度时，这种模式无法解决单地域资源瓶颈的问题。

而“异地多活”就是因为通过传统的灾备手段无法解决上述问题时，阿里巴巴在经过多年研究，成功在 2013 年的双十一提供了“提供‘丝般柔顺’的用户体验”后，首次在业界亮相的一项基础技术。

异地多活的设计，首先需要从服务单元化开始。

（1）服务单元化（单元化的容量与容灾）

（1.1）单元化的核心概念

所谓单元化，就是根据业务特点把系统的核心业务链路从逻辑上垂直封装成独立的业务处理单元，其中每个业务处理单元中的相关服务都是闭环且完整的。然后我们可以把这些业务处理单元水平复制后部署在多个不同的逻辑数据中心（LDC）中，通过多活架构的路由服务调度，使得每个业务处理单元负责一部分业务请求的处理。其中 LDC 的物理位置可以是同城的不同机房，也可以是位于异地不同的数据中心。

LDC 又可以分成两种类型，一种是单元，一种是中心。单元就是我们所有做异地多活的业务所在的 LDC，它具备水平扩展的能力，可以任意复制。另外一个中心，是一个特殊的单元，它主要包括一些长尾没有做单元化改造的业务和一些强中心无法进行单元化改造的业务。如果把单元再详细拆分，逻辑上我们又可以把单元分成两部分，一个是“核心业务单元”，一个是“共享业务单元”，一般而言二者只是逻辑区分，物理上部署在一起。

① 单元：核心业务单元（Unit）

- 核心业务单元化。
- 保证核心业务单元分片均衡（RouterID）。
- 核心业务尽量自包含（调用封闭）。

② 单元：共享业务单元（通常与核心业务单元部署在一起）

- 不能按统一拆分维度拆分（RouterID）。
- 被核心业务高频访问。
- 与核心业务单元部署在一起，统称为“单元”。

③ 中心：全局单元（Center）

- 非核心应用和数据（长尾）。

（1.2）单元化的架构优势

单元化解决了容灾的问题，提升了业务的连续性。

系统运行过程中，容灾不只是地震、挖光纤这些低概率事件，同样还有人为原因等高概率的事件，在服务单元化架构下，这些故障均可通过将故障单元的业务流量快速切换到其它正常业务处理单元来解决。以下是一些常见的故障场景：

• 人为操作失误

例：常见的有配置错误、应用发布失败等等。

• 硬件故障

例：常见的就是网络设备出故障，导致机房或者集群内多台服务器受影响。

• 软件故障

例：重要的代码变更、配置变更或架构变更。

• 网络攻击

例：DDoS 等网络攻击。

• 断网 / 断电

例：支付宝光缆被挖断。

• 自然灾害

例：雷击导致机房电力故障。

以上这些场景出现时，就可以秉承“先恢复，再定位”的原则，先进行业务系统的快速恢复，这样可以有效的提升业务的连续性。可以做到让“业务恢复时间”和“故障恢复时间”解耦合。

（2）多层路由服务架构

在异地多活架构中，服务单元化之后各服务单元之间的流量路由调度是整个架构中的核心服务。

（2.1）路由服务框架

整体的路由服务框架包括核心路由规则、以及路由规则的全局管控和分层实现。

路由规则：按照路由规则，对所有流量中携带的路由标进行解析，从而正确的引导流量。切流的过程就是对路由规则的变更和全局生效的过程。

全局管控：路由规则集中配置，流量的管控分层实现，在各层服务的端上执行。要求全局管控的路由规则必须具备实时性、持久性和一致性。

（2.2）多层路由实现

统一接入层

接入层的职责是统一处理业务处理单元前端入口流量，根据流量中请求携带的分流 id（路由标）与单元化类型（该信息可以放在 cookie、header 或 request 中）和当前的全局路由规则进行匹配计算，完成流量分发与路由纠错。

服务层 (RPC)

从接入层进入的业务请求，以及业务内部之间的服务调用请求，均需要经由服务层的路由服务来进行调度，除了本单元的业务请求外，还负责跨单元的服务访问调度，如访问中心服务，或者单元服务纠错（注意：这里面需要使单元间的服务相互可见）。并且，服务层路由服务会根据当前全局路由规则，监测要在当服务单元注册的业务是否合法，根据规则禁止不合法的服务注册（例：中心服务在单元发布）。

消息层 (MQ)

消息层的路由服务不做纠错转发处理，只会根据当前全局路由规则对消息主题的生产者和消费者分别进行数据合法性校验，对于生产者来说如果发现消息的路由标不属于本业务处理单元，那么会禁止该消息的发送。对于消费者来说只能消费本单元消息，也只有本单元的消息对于消费者是可见的。

数据层

数据层处于业务请求链路中的最底层，直接来源于服务层，由于其上各层已经进行了路由控制，基于时效性的考虑，这一层的路由服务不做纠错转发处理，只做有效性校验，会根据当前全局路由规则，进行数据处理请求的有效性判断，阻断非法的数据请求（例：非本单元的数据写请求）。

（3）实时分流技术

实时分流技术的核心是通过面向应用的集中一站式管控，实现快速、灵活的路由规则配置，并通过结合多层路由服务框架，快速的进行全局路由规则的推送和更新，来完成对业务流量从 DNS、统一接入层、服务层、消息层到数据层全链路的实时分流调度和管控。

实现快速实时分流的管控平台需要实现的关键功能点：

- 路由规则和快速、灵活配置。
- 多层路由服务（接入层、应用层、数据层）的运维管控。
- 发生灾难场景的快速一键切流操作。
- 多活场景下的数据监控。

（4）业务数据的分区及数据流模型

服务的单元化、路由规则的设计，背后的依据都是根据业务的特点，进行的业务数据分区化的设计。以电商场景为例，粗略的可分为买家、卖家和商品等几大维度。对于淘宝来说，买家业务体验是最重要的，所以淘宝的交易单元化是以买家业务做的单元化，分流规则是买家 id，以买家 id 为维度进行业务数据的分区，每个业务处理单元负责处理一部分买家的业务请求，每个业务处理单元也对应着不同买家 id 集合的数据分片。这样单元中会包含完整的业务处理逻辑、业务分区数据，实现业务处理的本单元闭环。

（4.1）数据流分析

为了有效进行业务数据的设计，需要根据业务系统的特点，结合核心业务链路进行较为详细的数据流模型分析，主要可以从以下几个方面进行分析：

- 基于数据库的数据流分析。
- 基于服务（RPC、HTTP）的数据流分析。
- 基于异步消息的数据流分析。

（4.2）数据拆分原则

• 场景的覆盖度

数据拆分要除了核心业务场景要全覆盖之外，需要尽可能兼顾较多的业务场景。

• 数据拆分关键字的取值基数

数据拆分关键字取值基数的范围尽量要大，这样可以对分区的数量和分区容量做更灵活设计。

• 数据拆分关键字的取值分布

数据拆分关键字的取值分布要尽可能的均匀，这样可能减少分区间的数据分布不均衡问题。

（5）分布式存储技术及数据复制模型

异地多活架构通过服务单元化来实现了整个业务系统的高可用性和扩展性，在这个架构体系的底层是利用分布式存储技术来完成业务数据的分区处理。针对不同的存储产品会有不同的实现方式，但对异地多活架构而言，对底层分布式存储系统能常会遵循下面几点共性的标准：

(5.1) Share Nothing 架构

采用基于可水平扩展的无共享结构，每个结点独立使用本地的 CPU、内存、磁盘。节点之间的通信协调任务运行在网络之上且核心逻辑主要依赖软件来实现。甚至于可采用不同物理区域的不同的数据库集群来分别管理不同的数据分片。因为无共享系统不需要专门的硬件，具有较高的性价比。可以跨多个物理区域分发数据，从而形成单元封闭，减少用户的访问延迟，当某个数据中心发生灾难时，整体业务不受影响。

(5.2) 多主节点复制模型

在传统的主从复制架构下，系统只有一个主节点，而所有写入都必须经由主节点。这会带来潜在的可用性问题及性能扩展问题。为了容灾数据中心级别故障及单元封闭化（数据更接近用户）要求而采用的多数据中心的多主节点复制模型，则可以在每个数据中心都配置主节点，在每个数据中心内部，采用常规的主从复制方案；而在数据中心之间，由各个数据中心的主节点来负责同其他数据中心的主节点进行数据的交换、更新（如果各数据中心为独立的存储系统集群，可以采用外部的数据同步产品方案）。

另外，多主的模型必须要解决同步与异步冲突检测的问题。首先，就是通过业务层面的数据拆分，来避免多数据中心数据写冲突的问题，通过异步数据同步来使各数据中心最终收敛于一致性状态。另外，在切流或容灾状态下（用户所属数据中心转移），需要通过位点检测、全镜像同步、迁移期禁写控制等技术手段来进行写冲突控制。

5. 结束语

正如“面向失败设计”板块开篇所说，如果企业在专有云上想更好的发挥云的高弹性、高扩展性和高可用性，企业的 IT 系统需要进行面向失败设计。通过系统高可用架构设计、容量规划、容错设计和容灾设计，企业可以构建一个全面的面向失败的设计框架。这一框架不仅可以有效提升系统的稳定性和可靠性，还能最大程度上保障企业在面对故障时能够快速恢复，最大限度减少业务影响。衷心希望企业能够借助面向失败设计方法、原则、策略模式和最佳实践，在实际应用中实现更加稳健和高效的系统架构，从容应对各种挑战。

第二节 / 系统运行的“明察秋毫”——可观测能力建设

(一) 让云为应用所用——打造云 + 应用一体化运维能力

1. 背景介绍

随着各大政企客户数字化转型业务规模不断扩大，上线的业务系统日益增多，核心业务越来越依赖整个体系的稳定运行。应用架构设计、接口调用性能、云平台性能、网络性能、微服务组件承载能力等诸多因素都会影响用户体验。而信息化时代的运维方式，随着云上数字化转型的浪潮而变的不合时宜，信息化时代的运维模式如果是“小马拉大车”，那么在数字化时代就变成了“小马拉火车”。数字化时代需要有匹配的数字化智能运维模式与之相匹配。

2. 难点分析

(1) 应用全生命周期管理难点

由于缺乏针对应用的管理规范以及在线自动化管理手段，通常难以实时、精准地掌握业务系统的运行状态，也难以有效监管各应用厂商对业务的运维情况。为提升对各业务应用及其厂商的管理水平，增强业务稳定性并提高资源使用效率，亟需构建覆盖应用全生命周期的管理能力。

(1.1) 应用规划管理难点

针对已经上云的核心应用规划均在线上完成，导致后期无法查看实时、准确的应用架构信息。带来的问题有：

- a. 无法保障应用架构信息准确的保留，并传递给其他运维人员，导致无法保障运维效果的连续性。
- b. 由于不能在线保存实时、准确的应用架构，在出现业务异常时，运维人员难以判断故障根因，导致异常定位时间长，从而导致业务异常影响时间长。

（1.2）应用部署管理难点

现有资源管理方法无法以应用维度展示资源，导致无法以业务系统的视角展示当前系统关联云资源的信息。例如，无法展示核心业务系统相关的所有云资源使用情况。同时，无法展示此业务系统使用资源的监测情况。例如，一共使用了多少 CPU、内存、存储等资源使用情况。现有方法是通过手工周期性统计解决问题。无法保障实时性、准确性。导致云平台扩容依据无法保障准确性。

（1.3）应用运行管理难点

缺少在线实时掌握核心业务系统架构的运行实时状态。查看业务系统架构需要到每个系统中单独查看，降低了管理效率与运维效率。缺少自动化运维能力，运维效率低。缺少故障自愈的手段与方法。需要建立灵活的自动化修复故障的手段。

（2）配置信息管理难点

配置管理库（CMDB）是 IT 运维的核心组件。政企大客户往往会缺少配置管理库的建设，无法保障运维效率能够支撑现有业务需求。建设配置管理库（CMDB），可以为后续运维自动化、智能化打下坚实基础。

（3）变更安全风险能力弱难点

由于变更导致的运维风险，是云上应用稳定运行最大风险。例如云资源的违规下线，可以导致业务无法运行。业务相关云资源的配置信息更改（配置规格降低等），也会导致业务的稳定运行。同时应用变更、命令、脚本下发等，也会产生变更风险。为了响应国家对于运维安全的要求，需要进一步加强变更安全风控的建设，避免由于变更导致的安全风险。

3. 应对策略

建立统一的“云 + 应用”全生命周期管理能力，实现从应用规划阶段、部署阶段，以及运行阶段的全面管理能力。规划管理中，将在线管理业务系统的架构图、访问关系，资源使用情况。

部署阶段将以应用维度管理资源。运行阶段，将建立完可视化编排能力，增强故障预案能力，降低故障恢复时间。

建立 IT 核心组件配置管理库（CMDB）。自动化的管理基于阿里专有云的所有资产配置信息，保障云资源使用、异常定位的效率。为领导层制定 IT 运维规则、业务运营规则提供数据支撑。

项目将建立安全风控能力，保障业务的安全稳定。保障避免由于各种原因的高风险变更操作导致的安全风险。

（1）应用全生命周期管理

应用全生命周期管理方案，符合中国信通院发布的可信云，“云 + 应用”一体化运维管理先进级（最高等级）能力要求。

应用全生命周期管理，解决了各应用厂商应用管理不规范的问题。应用管理不规范，会导致运维部门无法实时了解应用架构状态，应用健康状态，应用资源使用状态，应用变更状态，安全风险状态等，也会给应用运维带来风险，从而影响业务的稳定性。

利用应用全生命周期管理方案，可以在线自动化管理应用从规划阶段，部署阶段，到运行阶段的全生命周期，制定明确的管理规范，从而在线自动化的管理应用，保障业务稳定性和资源使用的合理性。

（1.1）应用规划管理

应用管理等级规范制定：应用等级规范从各种不同的维度为应用进行分组，明确不同的运维要求，同时也对应用架构设计进行规范，方便后续的运维管理活动。

应用等级有 4 种，高可用等级、连续性等级、安全性等级、重要性等级。高可用等级对应应用的高可用架构，保障对应用高可用的管理要求；连续性等级对应应用的容灾架构，保障对应用容灾的管理要求；安全等级对应应用的安全架构，保障对应用系统的安全管理要求；重要性等级对应应用的事件响应级别，保障事件的故障定级定义。不同的应用需要有不同的规范，并且对应不同的应用运维管理要求，保障了运维部门对应用厂商应用运维的管理能力。

在线应用架构设计：支持在线管理类应用系统的架构信息，实现应用架构蓝图的实时在线展示。包括应用系统基础信息，例如应用系统名称、所属产品线、管理及运维人员信息。运维人员

信息保障权限管理，保障不同的运维人员，只可以管理对应的应用系统。应用架构信息包括应用系统的访问关系设计，和应用资源拓扑关系等等。应用访问关系的管理，保障了应用间安全策略的设置，保障了应用的安全，避免被安全攻击从而导致的运维和数据安全。应用资源拓扑关系，保障了管理人员与运维人员可以随时查看应用架构设计，对资源使用情况的管理，以及异常问题的定位，起到重要的作用。通过应用规划管理，以及后续的基于应用规划管理蓝图上的变更，可以保障应用设计架构图可以实时在线的查询，避免应用架构信息丢失及失真的情况。

（1.2）应用部署管理

基于应用架构蓝图上的云资源变更，可以保障以应用维度管理应用资源，并且可以实时更新应用架构图，从而保障了管理人员以及运维人员可以随时查看实时的应用架构图，保障了资源使用情况，以及问题排查的效率。

（1.3）应用运行管理

提供基于类应用的自动化编排能力，以及全景业务链路可观测能力。通过可视化编排的能力，为类应用提供可视化、可拖拽的自主流程编排能力，将不同类型的组件通过可视化编排的方式形成可执行的流程模板。除此之外，模板编排页面还提供命令执行、脚本执行和文件下发三种自动化能力。可视化编排的流程，可以以服务形式对其他人开放，实现运维服务能力共享的目的。同时，可视化编排的流程，可以定义为故障预案，实现系统异常的自动化恢复能力，减少故障修复时间，提高类业务的稳定性。

（2）配置管理数据库（CMDB）

统一配置管理库，包含了配置项全生命周期的信息以及配置项之间的关系（包括物理关系和依赖关系），是 IT 运维的核心组件。缺少 CMDB，将无法实时掌握最新的云平台资产配置信息，无法准确管理云平台及应用，从而导致管理层无法准确制定云平台及 IT 信息技术的管理要求和相关决策，导致运维效率降低，资源成本浪费等情况。

基于阿里专有云平台的配置管理数据库 CMDB，实现通用的运维数据库，支持结构数据存储、读取和统计分析；支持自动采集基础设施、云产品资源、云服务实例、业务数据、应用数据领域的数据和关联关系，同时支持拓扑型数据的处理。

基于阿里专有云的统一配置管理数据库，利用应用全生命周期管理的方式，利用自动化的方式保障了配置管理数据库资产信息的准确性和实时性，解决了需要人工梳理资产信息的问题。

（2.1）系统资源管理

运维部门需要周期性手工统计现有系统资源使用情况，通过查看资源使用效率，以用来决策各业务使用资源的合理性，以及云平台扩容规划等内容。由于现有决策信息均来自于手工，导致应用系统资源使用信息统计颗粒度粗，时间滞后，并且无法保障准确性，最终导致管理层无法准确、实时的衡量各业务资源使用情况是否合理，也无法准确评估云平台扩容规划等。

提供基于配置管理库的实时及准确的配置信息，以业务系统的视角展示当前各业务系统云资源使用信息。例如，可以展示结算业务系统相关的所有云资源使用情况。同时展示此业务系统使用资源的监测情况，例如一共使用了多少 CPU、内存、存储等资源使用情况。

（2.2）业务资源繁忙度管理

基于配置管理库信息以及现有平台的云资源繁忙度信息，提供以业务线为视角的全局应用系统所对应的云服务实例容量使用率统计和用量统计；并提供局部应用系统的容量使用率、用量，以及单体应用容量使用和用量分布信息，支持繁忙应用排名和闲置应用排名。

（3）变更安全风险

IT 业务系统异常，通常来源于应用变更、云资源变更等。缺少事前安全风险的建设，会出现由于各种原因导致的变更风险及业务影响。例如运维人员违规下线数据库，会导致相关业务的完全不可用，产生了业务不稳定的风险。建设针对应用运维的变更安全风险能力，可以提前审计各种变更操作。

安全风险防护策略可自定义高危命令和脚本的拦截策略，对变更过程进行安全防护，自动检测高危命令和脚本并拦截，可展示拦截历史记录等。例如运维人员违规下线数据库时，安全风险将事前审计变更操作，对违规操作进行拦截，避免了大量的运维安全风险，保障了业务的稳定性。

4. 技术方案

基于 AOP 提供的云 + 应用一体化运维方案可以实现应用的“云 + 应用”全生命周期应用管理，建设配置管理库（CMDB），实现针对变更的安全风控，以及实现基于业务链路的全景可观测能力。

(1) 应用蓝图

从研发阶段应用架构设计 / 管理 / 环境规划的应用逻辑态管理、到应用发布部署阶段部署架构规划 / 资源管理 / 发布部署的应用部署态管理、到应用监控运维阶段全景监控与事件 / 变更与自动化运维 / 应用应急与容灾的应用运行态管理，实现应用全生命周期的“三态”纳管。



图 1 应用全生命周期“三态”架构图

面向“云 + 应用”一体化管理，打通从应用创建到应用运行的全流程：

- ① 应用规划：部署架构和资源规划。
- ② 应用建模：应用系统建模。
- ③ 资源规划：部署架构和资源交付。
- ④ 应用发布：多种发布模式和策略。
- ⑤ 应用日常运维：场景定义，可视化编排。

(1.1) 应用三态数据模型

AOP 通过应用逻辑架构的定义、环境有效规划，构建应用的部署架构和资源供给流程，导入运维策略，实现各种运维方案的有效落地。

a. 基础设施环境：对云基础设施的映射，包括（多云、多 region、多租户、资源集，VPC 等）。

- Region 的确定因素：是否多地域管理、是否需要跨地域的容灾、是否将测试环境与生产环境分离、是否建设一云多芯。
- AZ 的确定因素：是否有同城容灾需求、是否要求 RPO=0。
- VPC 的确定因素：是否是多 Region、是否要求独立的 DMZ 区、是否要求独立的外联接入区、是否不同的组织或部门需要独立分区、是否有不同等保等级的应用、业务应用是否按业务类型独立分区。

b. 应用运行环境：对应用环境的抽象，包括：环境类别（生产，测试，研发）、应用单元（A_Unit）、部署区域（A_Zone）。

- A_Unit（应用单元）：传统主备、传统双活、单元化。
 - 传统主备：只有一个主单元提供对外服务，另外一个单元以热备（应用启动，不接收业务流量）或冷备（应用不启动，应急时启动）等方式存在。
 - 传统双活：两个单元同时对外提供服务，但是数据往往只有一份（不是指副本，而是指数据库服务实例）。发生故障时，应用层面只进行流量切换，配合数据库切换实现容灾。
 - 单元化：每个单元是一个能完成所有业务操作的自包含集合，在这个集合中包含了所有业务所需的所有服务，以及分配给这个单元的数据。将流量按照客户 ID 或其他规则切分，分配到每个单元中。发生故障时，将故障单元的流量引导到其他单元处理。
- A_Zone(应用部署区域)：
 - RZone（Region Zone）：最合理理论上单元化定义的 zone，每个 RZone 都是自包含的，拥有自己的数据，能完成所有业务。通常只在单元化应用单元内出现。
 - GZone（Global Zone）：部署了不可拆分的数据和服务。GZone 内的应用数据仅有一份，传统应用通常都属于这种。
 - CZone（City Zone）：同样部署了不可拆分的数据和服务。CZone 是为了解决异地延迟问题而特别设计的。只在蚂蚁单元化应用单元内出现。

- c. 应用策略模版：用于应用架构治理，将规则固化在策略模版中（高可用容灾策略、安全与风控策略、可观察性策略）。
- d. 应用逻辑态模型：开发过程中确定的产品架构及管理属性（应用组成、应用等级）。
- e. 应用部署态模型：某个应用在某环境中的具体部署架构及规格（应用部署蓝图、运维特征）。
- f. 应用运行态模型：应用实际的运行状态与部署态保持一致，用于监控应用实时运行状态。

(1.2) 应用分级管理

在研发期就对应用等级定义，为后续的应用有效运维，提供有效遵循依据。

可以参考如下等级进行高可用定级：

- ① 等级 A，标准可用率（不包含计划内停机）为 99.95% 7*24 小时 4.38 小时。
- ② 等级 B，标准可用率（不包含计划内停机）为 99.77% 7*24 小时 20 小时。
- ③ 等级 C，标准可用率（不包含计划内停机）为 99.54% 7*24 小时 40 小时。
- ④ 等级 D，标准可用率（不包含计划内停机）为 99.32% 7*24 小时 60 小时。

可以参考如下等级进行重要性定级：

- ① P1，关键业务系统。
- ② P2，重要业务系统。
- ③ P3，一般业务系统。

(2) 应用自动化运维

随着云服务的广泛应用，让 IT 基础设施构成趋于多元化，混合云基础设施已经成为新一代企业 IT 常态，即公有云、私有云、超融合等架构并存，在从业务、应用到云平台一体化运维带来很多挑战：

- 看：建立从业务、应用到云平台的统一的精细化的云资源管理和运营体系；提供高效整合服务和流程能力；实现降本增效。
- 管：通过应用视角构建基础设施资源统一审批、管理和分配，提供灵活、可配置的资源；提供多维度监控告警管理；实现稳定、高可用、合规的应用管理服务交付。

- 控：灵活、便捷 IT 基础设施管理有效满足业务创新，应对市场竞争；资源与应用编排、标准化业务应用部署发布、统一的变更管控、统一的运维安全风控，构建应用高效上线，稳定运行能力，保障业务的稳定运行，让客户用好云。
- 建设思路，构建“云 + 应用”一体化自动化运维：
- 统一编排入口，提供可视化编排引擎（表单 & 流程）、动态调整、灵活编排各类任务和运维场景。
 - 统一原子服务，以丰富的业务、原子服务、运维模板和脚本库维护统一的原子能力库，构建服务目录。
 - 统一审批引擎，提供统一的审批入口，可确保所有必要的各方得到通知并参与到变更审批流程。
 - 统一规则引擎，内置定时任务、消息触发等执行引擎，实现快速、稳定交付灵活的。
 - 统一数据视图，资源数据可视，以 CMDB 为编排动态资源库，提供业务及网络拓扑。



图 2 业务视角“云 + 应用”一体化自动化运维大图

- 统一配置管理：基于统一的 CMDB，提供业务维度、应用维度、云服务实例、云平台资源、硬件基础设施的视角展现。
- 自定义服务目录：提供资源灵活编排、资源一键申请、资源自动交付能力。
- 多策略多平台发布部署：支持多种发布平台，支持多种发布策略。
- 灵活的运维编排：自定义运维组件，图形化在线编排运维流程，并支持将运维流程模板以服务的方式对外开放。

5. 小结

本文探讨了打造云 + 应用一体化运维能力的关键要素。针对应用全生命周期管理、配置信息管理和变更安全风控等难点，我们结合 AOP 产品能力提出了综合应对策略，包括应用规划、部署和运行管理，以及配置管理数据库和变更安全风控措施。通过应用蓝图设计、应用自动化运维能力构建，希望为企业在打造云 + 应用一体化运维能力的道路上提供一些思路和参考方案。

(二) 观全局以善决策——打造“云 + 应用”一体化全景可观测能力

1. 背景介绍

随着企业的不断发展，业务规模和应用复杂度显著增加，支撑业务的技术栈日益多样化。人员规模扩大和部门间协作的复杂化进一步加剧了管理难度。与此同时，互联网、大数据、云计算、人工智能和区块链等新兴技术席卷整个 IT 生态，企业需求也在不断演变。业务对服务的稳定性、可靠性的要求越来越高，业务系统对 IT 支撑的依赖性不断增强，行业监管对 IT 架构稳定性的要求也愈发严格。

然而，当前市场上的运维产品主要以托管服务为主，专注于公共云环境下的云厂商托管运维，缺乏针对企业自服务和自运维设计的产品。面对专有云环境，企业亟需一体化的运维解决方案，能够提供跨云平台资源和服务实例、应用系统和服务实例的统一运维界面和权限管理。

2. 主要挑战

基于上述背景，企业面临的主要挑战如下：

a. 缺乏体系化的安全生产 – 缺乏安全生产系统化的流程、机制和方法

- 现状：
 - 系统和应用报警处理是分开的；
 - 没有统一事件、故障定级和协同机制；
 - 缺少以业务监控为入口的问题发现机制。
- 影响：导致问题发现不及时，响应效率低下，难以保障业务连续性和稳定性。

b. 产品割裂 – 复杂和割裂的多种监控平台并存

- 现状：
 - 企业大部分已经采用开源、自研、ISV 附带的多种监控平台；
 - 应用层、云平台层、基础设施监控大部分存在割裂。

- **影响：**增加了运维复杂度，降低了监控数据的整合与分析效率，难以形成全面的视图。

c. 可监控性覆盖难 – 传统应用无法进行可监控性改造

- **现状：**
 - 企业的传统应用大部分不具备很好的可观测性，无法通过日志、探针等方式获取监控数据；
 - 平台层运维对应用研发改造的推进力偏弱。
- **影响：**限制了监控系统的覆盖面，影响了问题的快速定位和解决能力。

如何在割裂的运维体系下落地全景监控，需要解决如下割裂问题：

a. 应用运维和平台运维之间的割裂

- 不同团队负责不同层次的运维工作，缺乏统一的协调机制，导致信息不对称和响应延迟。

b. 平台运营和平台运维之间的割裂

- 运营和运维目标不一致，沟通成本高，难以形成合力，影响整体效率。

c. 监控报警和应急处置之间的割裂

- 报警信息分散，缺乏统一的应急处置流程，导致问题处理不及时，风险控制不足。

d. 不同的垂直应用系统之间的割裂

- 各应用系统独立运作，缺乏统一的监控和管理平台，难以实现资源共享和协同工作。

3. 应对策略

(1) 建设思路

建立“云 + 应用”一体化监控能力：通过业务日志分析、链路追踪、Health Check 等手段实现端到端全链路监控，提供业务维度、应用维度、云服务维度、云基础设施“云 + 应用”一体化全景监控能力。

- **业务维度监控：**业务全链路影响度分析、应用关键 KPI 指标分析，第一时间准确发现业务运行风险和故障。

- **应用维度监控：**应用自身 KPI 指标分析及调用链上下游分析，强大的应用层问题发现和定界能力。
- **云服务实例维度监控：**云服务、云资源运行状态分析，基于应用视角的云监控。

(2) 技术方案

基于业务链路的全景可观测平台，展示云 + 应用全景健康状态。可以主动探测业务链路、业务场景的健康与运营状况，并快速发现、定位业务异常，保障业务系统的稳定运行。

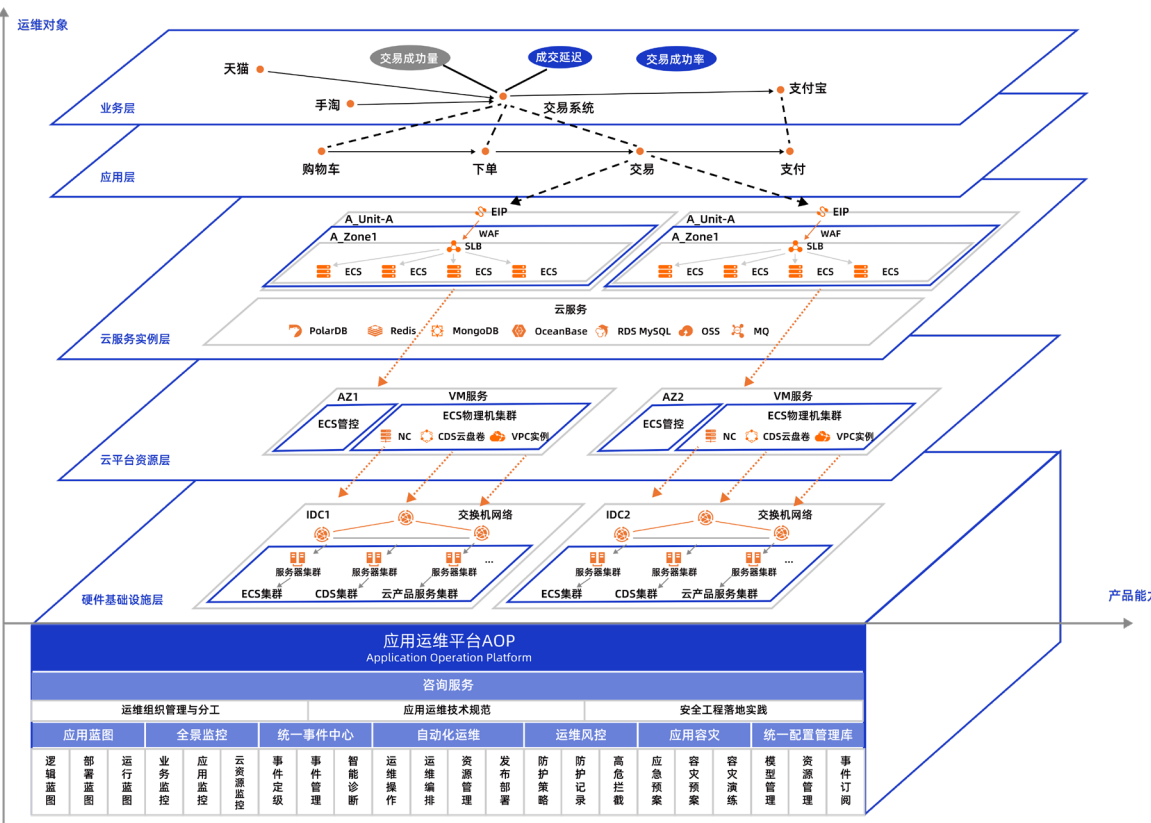


图 1 云 + 应用全景健康状态示意图

（2.1）业务监控

AOP 全景监控的业务监控能力，是从业务、应用到云平台“云 + 应用”一体化监控，具体如下能力：

- 秒级监控：指标秒级刷新且采集和计算迟延不超过 5 秒，最高支持百万级容器 / 虚拟 / 物理机上的业务指标采集。
- 黄金指标异常检测：千锤百炼的智能基线、黄金指标分场景智能检测、多维下钻智能检测、业务全链路监控、业务全链路监控、业务零配置成本。
- 异构数据源：日志 / SLS 数据源支持、SQL 数据源支持、ELK 数据源支持。
- 全景监控大屏：可定制的全景监控大屏、灵活的多维度可视化大屏。

（2.2）应用监控

- 应用监控的全局拓扑提供如下能力：
 - 全应用拓扑视图展示：聚焦问题应用和应用上下游信息展示。
 - 应用的总览信息：展示当前应用运行态上下游的调用链信息，以及当前应用的入、出的流量等健康画像，直观呈现当前应用影响的业务范围。
- 应用画像：可视化拓扑叠加调用链分析，对应用内指标度量，高效（分钟级）实现故障定位。
 - 可视化拓扑：应用层级运行状态可视化展示。
 - 调用链分析：慢 API 请求、异常调用调用链一键下钻，挖掘故障根因。
 - 慢 SQL 分析：基于关键指标视图（响应时间、调用次数等）进行数据库性能问题的分析。
 - 慢请求分析：基于 API 监控、远程调用、MQ 调用关键指标视图（接口请求量、响应时间、调用次数等）进行慢请求性能问题的分析。
 - MQ 主题消费：展示消息发布、订阅的统计数据，包括请求数、响应时间和错误数，分析消息主题是否存在堆积、消费滞后等现象。
 - JAVA 异常：通过错误堆栈分布，分析异常点，整治不合理架构设计。
 - 线程池分析：保留转瞬即逝的信息，分析阻塞、慢、等待线程任务。

- 基础资源消耗：应用所在的 ECS、POD 环境信息的消耗分析。包含 CPU、内存、磁盘、网络 IO 等。

- 核心链路：以业务场景为中心的应用核心链路，将云原生、经典虚拟化下的应用运行态分布式链路，打上业务标签，染出不同场景的业务场景链路。通过调用量、耗时度量信息，挖掘下单支付环节应用链路分析性能瓶颈。

（2.3）云服务实例监控

- 基于应用视角的跨云监控：展示应用所属的云服务实例健康状态以及运行态。
 - 跨云监控：无缝对接阿里专有云、阿里公有云，云产品服务基础监控。
 - 云服务实例监控矩阵：计算类、存储类、网络类、DB 类。
 - 关系自动挖掘：基于应用链路拓扑发现，自动绑定应用和云服务关系。
 - 应用联动：出现故障时，通过应用健康画像可快速定位到有异常的云资源。
- 容量管理：开箱即用，基于应用视角的容量可视化，以容量智能算法和历史峰值算法实现应用容量使用率的统计；满足客户在业务上线前、运行中、下线后整个生命周期中对获取各个云产品的容量信息的需求，解决容量信息获取难、物理资源浪费多、容量管理成本高的痛点。
- 业务运行态智能监控：通过对历史数据进行学习，使用智能算法来替代传统的静态阈值设置，挖掘运维数据价值。

4. 小结

我们希望通过 AOP 产品帮助企业构建业务、应用到云平台的一体化监控能力，应对企业业务规模和应用复杂度逐步增加带来的一系列管理挑战，为企业提供全面的监控和管理解决方案，提升业务系统的稳定性和可靠性，满足日益严格的行业监管要求。

第三节 / 灾难应对的“预则立”——应急预案体系建设

随着信息技术的快速发展，企业信息系统日益复杂，涵盖了云计算、大数据和人工智能等前沿技术。这使得故障的不可预测性和潜在影响范围显著增大。为确保业务连续性和数据安全性，建立健全的应急预案体系变得尤为关键。

完善的应急预案体系不仅能在突发事件中提供快速有效的响应机制，还能显著降低风险对企业造成的负面影响。通过提前识别和评估风险点，制定详细的应对措施，并定期进行演练，企业可以在危机来临时迅速启动应急方案，最大限度地减少损失。

1. 应急预案体系设计

为了使应急预案体系建设更加实用和系统化，我们建议应急预案体系建设需要覆盖**建设目标、建设阶段、体系步骤、服务能力、工具产品、飞天底座 6 大板块**。其中，整体建设阶段需要涵盖**事前准备、事中响应、事后恢复与改进三大阶段**。每个阶段都有其特定的目标和建设步骤，确保在突发事件发生时能够高效应对，并从中学习经验以不断完善预案体系。



图 1 应急预案体系示意图

应急预案体系设计整体说明如下：

- **建设目标：**构建应急预案体系建设的全生命周期。
- **建设阶段：**覆盖事前准备、事中响应、事后恢复与改进三大阶段。
- **体系步骤：**体系建设分为 16 个步骤来完成，包括：风险评估、预案制定、组织架构、资源准备、培训与演练、沟通机制、法律合规、行业标准、快速决策、统一指挥调度、资源调配、信息发布、恢复业务、调查分析、经验总结及预案更新。
- **服务能力：**专有云云平台运维与交付团队依托工具产品及大量成功的客户实践案例助力客户应急预案体系建设。
- **工具产品：**利用专有云 AOP、ASR、MSHA、玄重尺等工具产品支撑云 + 应用的应急预案体系构建过程。
- **基础底座：**基于阿里云自主研发的飞天操作系统（Apsara）构建的企业级云计算平台，专为大型企业和机构设计。它提供了全面的云计算服务，包括计算、存储、网络、安全、数据库、大数据处理等，旨在帮助企业快速实现数字化转型和业务创新，为客户的应急预案体系建设提供坚实的基础底座。

2. 事前准备：未雨绸缪，防患于未然

（1）风险评估

- **识别潜在风险：**利用云平台的可观测工具（如阿里云 AOP、SLS 日志服务等）分析自然灾害、技术故障、网络安全事件等可能威胁到企业运营的因素。
- **评估影响程度：**分析各类故障风险对企业造成的潜在损失，确定优先级。需要重点考虑高可用及容灾建设。

（2）预案制定

- **分类分级管理：**根据风险类型和严重程度制定不同层次的应急预案。
- **详细操作流程：**为每种类型的突发事件设计具体的应对措施，包括启动条件、责任人、处置步骤等。

（3）组织架构

- **成立应急指挥中心：**设立专门的应急管理团队，负责协调各部门之间的应急响应工作。可通过协同办公工具（如钉钉）以提高沟通效率。
- **明确职责分工：**利用云平台的身份认证和权限管理系统（RAM），确保每个岗位角色都有清晰的任务分配，避免混乱。

（4）资源准备

- **资源储备：**提前准备好必要的应急资源，确保在紧急情况下拥有足够的资源支持。
- **技术支持：**建立应急支持团队，确保信息系统在紧急情况下正常运作；推荐使用本地 + 异地备份（如 ASR-BR 进行 ECS 镜像、RDS 数据库、OSS 存储、MaxCompute 大数据等数据备份）、多活容灾方案（ASR 平台容灾、MSHA 应用多活容灾等），保障关键业务的持续运行。

（5）培训与演练

- **定期培训：**组织员工参加应急知识和技能培训，阿里云也提供了“玄重尺”实训演练平台进行故障演练的实操培训演练、ASR 容灾切换演练、MSHA 业务切流演练等，提高员工的安全意识和应急处理能力。

- **实战演练：**通过模拟真实场景进行演练，检验预案的有效性和人员的反应速度，并及时调整优化。阿里云也可以提供低成本的实训环境和演练工具支持。

（6）沟通机制

- **内部沟通：**利用阿里云提供的即时通讯工具（如钉钉）建立高效的内部信息传递渠道，确保所有相关人员都能及时获得最新消息。
- **外部沟通：**与政府机构、合作伙伴及媒体保持良好沟通，必要时发布官方声明或新闻稿。阿里云也可帮助客户提供舆情动态监控。

（7）法律合规

- **遵循法律法规：**确保预案符合国家和地方的相关法规要求，避免法律责任。

（8）行业标准

- **参考行业标准：**参考国内外先进企业的最佳实践，不断提升自身预案水平。

3. 事中响应：迅速行动，有效控制影响范围

（1）快速决策

- **立即启动预案：**一旦确认突发事件发生，立即启动相应的应急预案，迅速展开救援和恢复工作。
- **动态调整策略：**根据实际情况灵活调整应对策略，确保方案的有效性。

（2）统一指挥调度

- **集中指挥：**由应急指挥中心统一调度，确保各相关部门协同作战，避免各自为政。通过建立可视化监控大屏和协作工具，提高指挥效率。
- **实时监控：**利用可观测工具监控平台及应用的健康动态（如 AOP），实时掌握事态发展，及时做出调整。

（3）资源调配

- **优先保障关键资源：**确保关键业务所需的资源得到优先保障，减少对整体运营的影响。可为关键业务预留一定的应急资源。

- **跨部门协作**：动员公司内外部资源，形成合力，共同应对危机。

（4）信息发布

- **透明沟通**：及时向内部员工和外部利益相关者发布准确的信息，保持透明度，增强信心。
- **舆情管理**：积极回应社会关切，防止谣言传播，维护企业形象。

4. 事后恢复与改进：总结经验，持续提升应急水平

（1）恢复业务

- **逐步恢复运营**：按照预定计划有序恢复各项业务，确保生产和服务尽快回到正轨。
- **评估损失**：统计此次事件造成的直接和间接损失，为后续索赔和补偿提供依据。

（2）调查分析

- **根因分析**：深入调查事件发生的根本原因，找出问题所在。
- **责任追究**：对事件中的责任进行合理划分，严肃处理失职行为。

（3）经验教训总结

- **全面复盘**：召开总结会议，回顾整个应急响应过程，找出不足之处并加以改进。
- **案例分享**：整理成功经验和失败教训，形成内部知识库，供未来参考。

（4）预案更新

- **持续优化**：根据新情况和新技术不断更新和完善应急预案，确保其科学性和实用性。
- **反馈机制**：建立有效的反馈机制，鼓励员工提出改进建议，共同提升预案质量。

5. 故障应急流程示例

故障应急的初衷是为了保障应用按预期可持续提供服务，即保障应用的高可用性。**可用性 = (年度总时间 - 不可用时间) / 年度总时间**，高可用性即追求不可用时间无限逼近于 0。**不可用时间 = 故障发现耗时 + 故障响应耗时 + 故障处理耗时 + 故障恢复耗时**。

我们认为一个通用的故障应急流程的建设思路分为以下 5 个步骤：**故障发现、故障响应、故障处理、故障恢复、故障演练**。同时，每个步骤都需要有对应的故障应急功能模块，包括但不限于：**全景监控、运维风控、事件中心、预案管理、预案分析、预案执行、预案库、预案演练**等。建议匹配关系如下图所示：



图 2 故障应急流程及产品功能映射示意图

故障应急流程及产品功能映射说明如下：

• 故障发现

故障发现涉及监控版本发布、配置和资源变更后的健康状况，通过全面或局部的健康检测确定应用的异常点。产品需要构建全景监控、运维风控等能力，并通过预案演练进行验证。

• 故障响应

故障响应过程中，建立包含相关干系人的故障群，同步并组织信息，通过讨论确定解决人员和故障处理方案。产品需要提供全过程的事件中心能力。

• 故障处理

故障处理通过快速匹配现有预案、基于原子组件编排处置方案，或在必要时制定人工应急措施，确保问题得到及时有效的解决。产品需要提供预案分析及预案管理的能力。

• 故障恢复

故障恢复遵循最小变更原则，仅处理有问题的对象，并通过分批、审批和防错等措施进行风险控制，防止引发新故障。产品需要有预案执行的能力。

• 故障验证

故障验证通过健康检测确认应用是否恢复正常，若无异常则消除应急事件，如有异常则重复检测和处理直至故障完全消除。产品需要有全景监控及预案分析能力。

结合专有云 AOP、ASR、MSHA、玄重尺等工具产品可有效支撑云 + 应用的应急预案体系构建全过程。

6. 结束语

在复杂多变的商业环境中，建立健全的应急预案体系是企业保持竞争力和稳定运营的关键。一个完善的应急预案体系不仅能够在突发事件面前提供快速、有效的响应机制，还能显著降低潜在风险对企业造成的负面影响。这一体系确保了业务的连续性，保护了员工的生命财产安全，并提升了企业的整体抗风险能力。通过融合云计算、人工智能（AI）和大模型技术，企业可以进一步优化应急预案体系，实现更高效、智能化的风险管理和应急响应。

第四节 /

系统灾备的“未雨绸缪”——容灾多活能力建设

导语

1. 云时代的容灾需求

自 1979 年全球首个容灾系统建立起，全球尤其是数字化组织逐步提升对容灾的重视程度。特别是进入到云和大数据时代后，信息系统及数据已经成为重要的生产要素，直接影响到组织的正常运行。相关国际组织分别制定了一系列灾难恢复标准或法规，如：ISO22301、SHARE78、GDPR 等。而我国历来重视基础设施的安全保护，政府及行业出台了一系列国家标准及法规，大力支持容灾建设，要求各组织确保有能力应对灾难及保护信息数据安全。随着国家网络安全等级保护 2.0 驱动，新基建迅速发展，云灾备已成为必要趋势和强诉求。

• “一云多芯”与容灾

在“一云多芯”战略下，“一云多芯”架构会成为必然趋势。由于支持多种处理器架构，需要考虑容灾方案中结合不同芯片架构的优势。利用“一云多芯”平台的灵活性，可以根据不同地区的资源可用性和成本进行动态调度。如果某个区域发生灾难，可以迅速将工作负载转移到其他区域的合适硬件上，减少停机时间。同时，在“一云多芯”环境中，通过统一管理平台和标准化接口，可以实现跨架构的数据保护和一致性管理。无论是哪种处理器架构，都能确保数据的安全性和完整性。

• AI 与容灾

人工智能（AI）技术的发展为容灾提供了新的可能性。AI 可以通过数据分析和预测模型，提前识别潜在的风险点，从而采取预防措施。例如，AI 可以监控网络流量和系统日志，检测异常行为，并在可能发生故障前发出警报。此外，AI 还可以优化资源分配，自动调整备份策略，以适应不断变化的数据环境。通过智能分析，AI 可以帮助组织更有效地规划和执行容灾计划，提高整体的容灾效率和效果。

• 最新政策与技术趋势

随着国家政策的推动和技术的进步，容灾需求也在不断演变。例如，最新发布的《“十四五”国家信息化规划》强调了数字基础设施的安全性和可靠性，要求加强关键信息基础设施保护。同时，5G、物联网（IoT）和边缘计算等新兴技术的发展，带来了新的挑战和机遇。这些技术的应用使得数据量激增，对实时性和低延迟的要求更高，因此容灾方案需要更加灵活和高效。此外，区块链技术的引入可以提供更高的数据完整性和不可篡改性，增强容灾系统的可信度。

总之，随着“一云多芯”战略的推进、AI 技术的应用以及最新政策和技术趋势的影响，容灾建设已经从传统的备份恢复向智能化、自动化和高可用性方向发展。组织需要综合考虑这些因素，制定更加全面和前瞻性的容灾策略，以确保在面对各种灾难时能够快速恢复并保持业务连续性。

2. 容灾系统的定义及解读

容灾系统是指在相隔较远的同城或异地机房，建立两套或多套功能相同的系统，系统之间可以相互进行健康状态监视和功能切换，当一处系统因意外（如火灾、洪水、地震、人为蓄意破坏、系统故障等）停止工作时，整个应用系统可以切换到另一处，使得该系统功能可以继续正常工作。容灾系统需要具备较为完善的数据保护与灾难恢复功能，保证生产中心不能正常工作时数据的完整性及业务的连续性，并在最短时间内由灾备中心接替，恢复业务系统的正常运行，将损失降到最小。

容灾系统根据机房之间距离可以分为同城容灾、异地容灾，以及两地三中心容灾等。“两地三中心”一般指的是一个生产中心、一个同城灾难备份中心、一个异地灾难备份中心。

实际的容灾系统设计过程中，我们重点关注的是 RTO 和 RPO 两个指标。具体定义可参考前序 4.1.4 章节。

3. 多活架构的定义及解读

多活架构是容灾系统中的一个子集，特点是其所有机房内的应用同时对外提供服务。当故障或灾难发生时，可以分钟级内实现业务流量切换，对最终用户的影响降到最低。阿里巴巴的“同城多活架构”和“异地多活架构”（代号“单元化”）都是典型的应用多活实现技术。

阿里电商的工程师们在业务容量和稳定性的双重压力下，开始发起内部代号“单元化”的异地多活项目，提出“1 年验证，2 年双活，3 年多活”的目标。异地多活构建的多单元在 2017 年成为阿里的基础设施，大量新技术均依托多活能力进行了生产规模化验证，并支持历年双十一，阿里内部年切流数千次的成功率高达 99.9% 以上。时至今日，应用多活云产品 (AHAS-MSHA) 先后帮助政企、物流、互联网等十余家不同行业的大型企业成功构建应用多活架构。

(一) 灾备管理平台介绍

为保障企业的业务连续性和数据安全，阿里云专有云通过灾备管理平台 ASR（Apsara Stack Resilience）提供同城容灾、异地容灾、备份等多种容灾解决方案，支持多种模式组合。专有云容灾方案基于阿里云自身的云计算能力设计与开发，遵循国际通用的容灾标准。

1. 客户面临的容灾管理难题

- 业务连续性要求越来越高

企业正不断增加资源投入，以确保业务的连续运营。数据丢失不仅会中断正常业务运作，还会带来巨大的经济损失、声誉损害和客户忠诚度下降等多重负面影响。服务不可用的后果同样严重，可能导致不可挽回的损失。

根据 IDC 的数据，在 2000 年以前的十年间，美国发生过灾难的公司中有 55% 当即倒闭；而在剩余的 45% 中，由于数据丢失，有 29% 在两年内也宣告破产，最终仅 16% 的公司得以生存。这些统计数据凸显了数据安全和业务连续性的重要性。

如何确保大量集中数据在各种灾难面前的安全性，支撑业务的连续运行，已成为企业面临的一个现实且紧迫的问题。有效的数据保护和灾备策略不仅是技术挑战，更是企业持续发展的关键保障。

- 数据安全等级要求逐步提升

企业的业务运转日益依赖于数据，数据已成为企业运作的基石。随着业务复杂性和数字化程度的提升，数据的安全性和服务的可用性变得至关重要。特别是，对数据安全的要求正在逐步提高，企业不仅需要应对日常运营中的数据保护挑战，还需确保在各种潜在风险和灾难面前，数据能够得到最高级别的安全保障。

- 传统容灾和备份技术无法满足云时代的灾备需求

传统的容灾和备份技术通常采用分层建设，存在异构厂家、各层独立灾备方案、无统一灾备管理平台的痛点，无法满足云时代的灾备需求。

- 容灾建设成本高昂

- 首先，容灾和备份等级要求越高，成本越高。需要在投资规模、需防范的灾难类型、可以接受的 RPO/RTO 三者之间找到平衡，以满足业务灾备要求为首要出发点。
- 其次，在特定容灾备份等级要求下，如何提高灾备系统的利用率，以较低的成本建立容灾和备份系统是客户考虑的重点因素。

2. 灾备管理平台支持架构及功能特点

灾备管理平台 ASR 是一款针对专有云的全场景容灾管理平台，支持同城容灾，异地容灾，备份三种灾备场景。提供容灾演练，灾难切换以及数字化灾备指标监控能力。

灾备管理平台主要功能和特点包括：

- 全场景容灾：**支持多种容灾场景的灵活组合，包括同城容灾，异地容灾，两地三中心容灾，备份等。
- 全链路技术栈：**支持阿里云从底座、IaaS 云产品、数据库、中间件、大数据、安全产品等全链路产品的统一容灾编排。确保切换流程按计划执行，减少人为干预和恢复时间。

(1) 同城容灾管理平台 ASR（Apsara Stack Resilience）

- 架构定义

同城容灾是指在与生产中心同一城市不同地点设立同城灾备中心，应对影响范围较小的局部灾难。同城容灾架构全面覆盖核心云产品，包括网络产品、云计算产品、数据库产品、存储产品、中间件产品和大数据计算产品，实现跨可用区（AZ）高可用。在灾难发生时可以快速切换至备可用区，云产品对应用服务域名不变。专有云同城容灾提供同城双机房和同城三机房等部署方案。在同城容灾双机房基础上，增加部署第三机房，可提供 RPO 为 0 的数据强一致性能力。

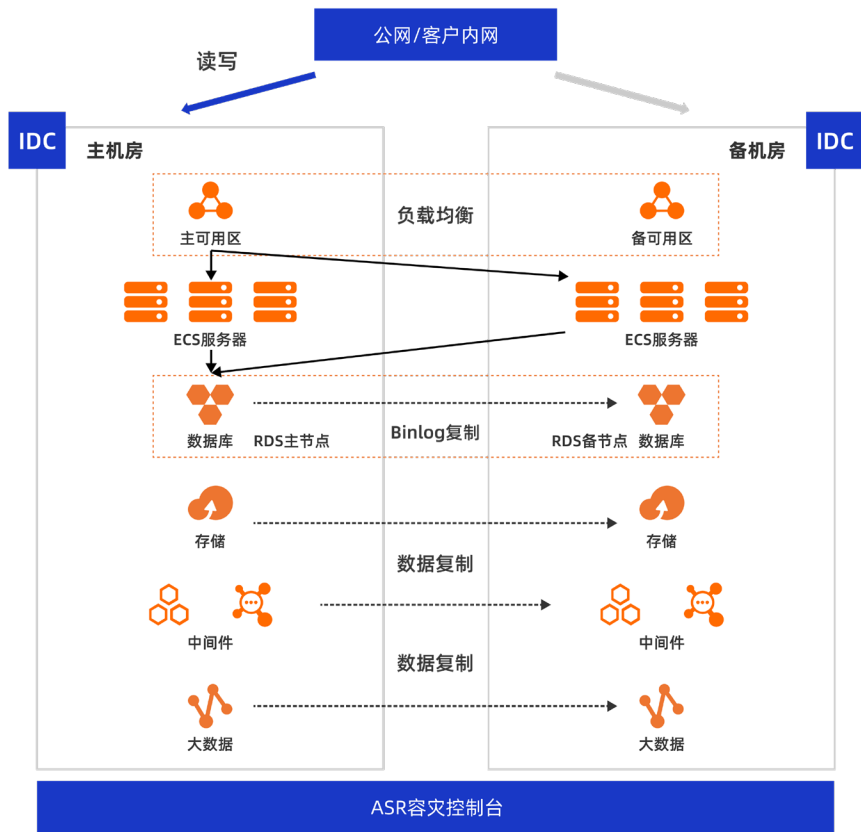


图 1 同城容灾架构示意图

• 功能特点

同城容灾管理平台 ASR 的主要功能包括监控、演练、机房级故障恢复和单产品故障恢复。

- **云产品监控**：定时检查云产品机房间同步状态、各机房业务状态和同步延迟时间，并通过容灾大屏实时监控双机房容灾情况。
- **容灾演练**：提供可自定义的云产品热切换功能，满足用户例行的容灾演练需求。
- **机房级故障恢复**：处理机房级故障，包括主机房断电、备机房断电、主机房网络孤岛、备机房网络孤岛以及脑裂。
- **单产品故障恢复**：在云产品高可靠功能异常时，为云产品服务提供双重保证，帮助云产品故障恢复，保证云产品业务连续性。



图 2 ASR 容灾管理控制台

（2）异地容灾管理平台 ASR-DR（Apsara Stack Resilience Disaster Recover）

• 架构定义

异地容灾是指在距离生产中心所在城市一定距离的另一城市设立灾备中心，应对生产中心可能发生的灾难。通过创建保护组，在生产站点和灾备站点对应资源之间建立一条热备通道，生产站点资源的数据修改会原封不动地异步热备到灾备站点对应的资源之中。生产站点发生故障时，通过故障计划中的故障切换将应用或服务切换到灾备站点上；生产站点故障恢复后，再通过故障计划中的故障回切将业务切回生产站点，以此来实现业务的连续性。异地容灾的容灾形态包括跨云容灾和跨 Region（地域）容灾。

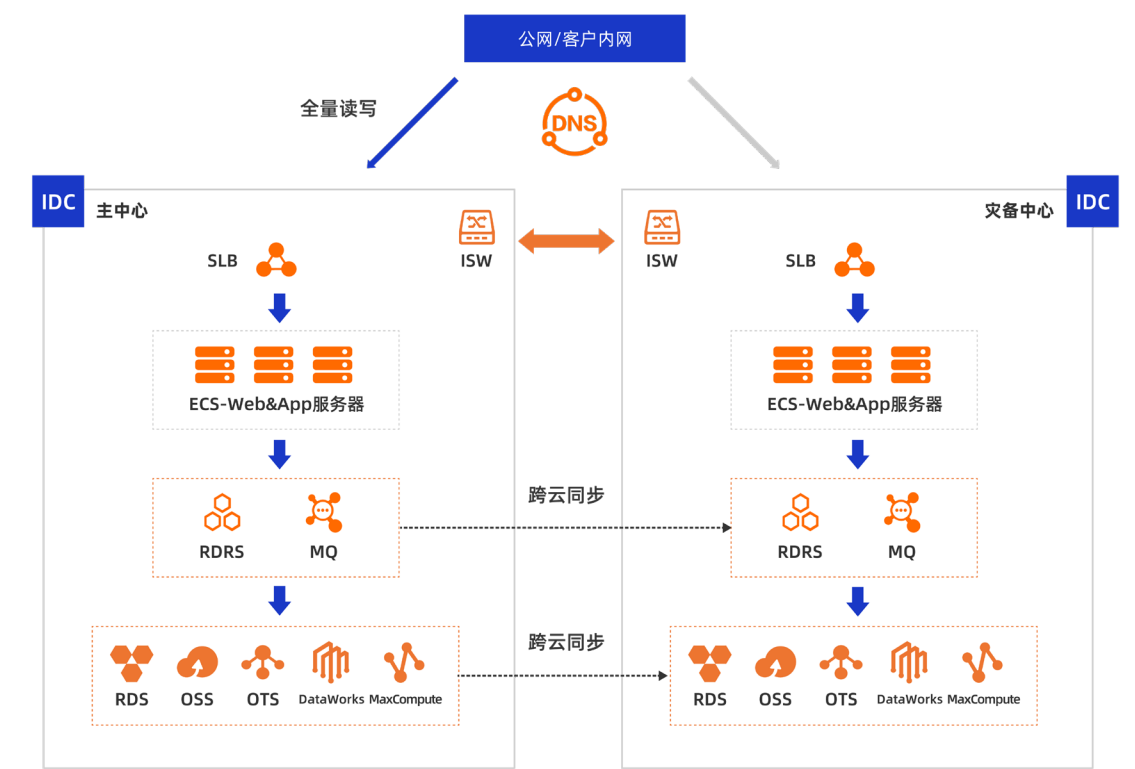


图 3 异地容灾架构示意图

• 功能特点

异地容灾管理平台 ASR-DR 的主要功能包括容灾概览、容灾演练和切换管理、系统管理和容灾大屏。

- **容灾概览：**包括容灾任务概览和容灾资源概览两部分。
- **容灾演练和切换管理：**包括业务容灾和站点容灾，可以设置保护组、演练计划和故障计划，并对某一类型的计划进行关联，并按执行序号批量启动所关联的容灾计划，包括演练切换、演练回切、故障切换、故障回切。
- **系统管理：**包括用户管理、生产云实例配置、日志管理、告警管理和同步延迟预警配置。
- **容灾大屏：**包括数字化实时展示容灾相关的指标、演练计划切换流程展示等。



图 4 ASR-DR 容灾管理控制台

3. 小结

灾备管理平台 ASR 是阿里云自研的专门为专有云提供灾备管理服务的平台，可有效帮助用户应对数据丢失、系统故障或业务中断等灾难性事件。通过对云平台的容灾编排能力，使得关键业务系统和数据在发生灾难时能够迅速、有效地恢复。

(二) 多活容灾产品介绍

1. 客户面临的业务多活难题

对某些重点行业及头部客户来说，有建设更高级别的业务多活容灾能力的诉求。在客户建设业务多活容灾的过程中，通常会面临应用流量管控门槛高、数据同步策略复杂、容灾切换数据质量保障难、多数据中心统一管控能力不足、容灾架构演进困难等痛点。

- **应用流量管控门槛高**
 - 如何对接入层、应用层、数据层的流量规则进行统一管理。
 - 路由规则下发，需要保证众多节点规则的一致性。
 - 需要具备多维的分流能力和动态调配能力。
 - 错误流量具备纠错和保护能力。
- **数据同步策略复杂**
 - 需具备服务、消息、数据库、缓存等多种数据的同步能力。
 - 双向数据同步需解决环形复制问题。
 - 远距离数据同步，对同步性能、同步带宽有很高要求。
- **容灾切换数据质量保障难**
 - 为了保障 RPO 要求，容灾切换过程中需具备对业务架构的各层进行状态检查的能力。
 - 对规则分发的收敛情况以及跨数据中心的同步情况进行准确评估，难度较大。
- **多数据中心统一管控能力不足**
 - 需要具备多单元统一的管理能力。
 - 需要对接众多的基础设施、云产品和开源中间件，形成整体合力。
- **容灾架构演进困难**
 - 需要支持业务架构的容灾多活架构持续演进，如：从同城双活演进到异地多活。

2. MSHA 多活产品介绍

在上个章节中，我们介绍了 ASR 以及 ASR-DR 灾备管理平台。ASR 聚焦云平台整体容灾性，实现跨可用区（AZ）高可用。ASR-DR 是应用视角的灾备控制台，针对企业用户的应用提供数据保护、日常演练和故障恢复的功能，主要聚焦在异地主备容灾场景。

MSHA（Multi-Site High Availability）多活容灾产品，通过提供业务属性的精细化切流、服务化调用和消息、缓存及数据库的应用全栈式管理，聚焦以业务应用为中心的云原生容灾架构，适合于对稳定性和成本综合权衡从而对业务双活 / 多活有诉求的客户场景。

MSHA 可以和 ASR/ASR-DR 在同城和异地场景下组合输出。MSHA 负责双活类业务的切换和管理，ASR/ASR-DR 负责主备业务的切换和管理。

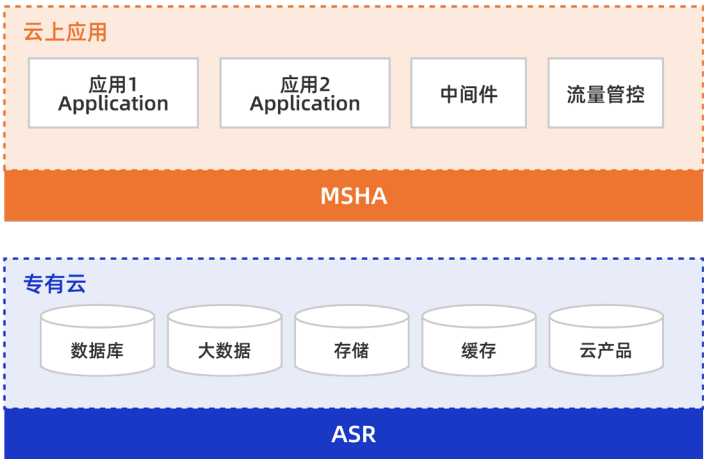


图1 ASR 与 MSHA 产品关系示意图

3. MSHA 核心概念解读

- 在 MSHA 的容灾架构模型设计中，把异地数据中心称之为“单元”，把同城的多机房称之为“单元格”。基于这套模型，MSHA 可以构建同城 + 异地的容灾架构嵌套。
- **单元（UNIT）**：异地的数据中心，多朵云，或者一云多 Region（地域）。不同单元的云基础设施各自独立。单元是异地容灾架构的管控维度。单元有普通单元和中心单元。普通单元中包含核心业务和共享业务。中心单元中包含中心化业务。普通单元是异地多活单元扩展的颗粒度。

- 核心业务：分片数据之上的单元化多活业务，比如买家下单业务。
- 共享业务：不能按分片维度拆分，但被核心业务高频依赖，单元可提供读服务，比如商品读服务。
- 中心化业务：强一致性业务以及长尾非容灾业务，比如扣减库存。

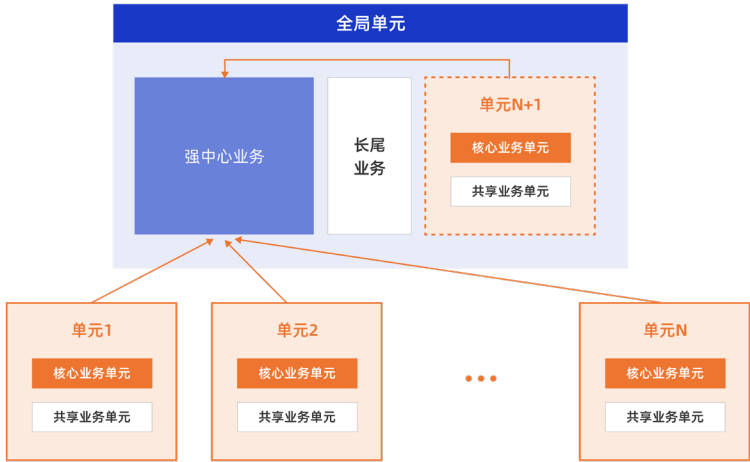


图 2 单元化模型

- **单元格（CELL）**：同地域下的多个可用区，单元格是逻辑概念，一个单元格可以包含多个可用区。单元格是同城多活的管控维度。

常规推荐的容灾架构，逻辑单元、单元格与物理地域、可用区的模型映射关系如下：

- **地域**：地域与单元一一对应，尽量不要有重叠的场景。
- **可用区**：可用区与单元格多对一映射，尽量不要存在单可用区在多个单元格的情况。

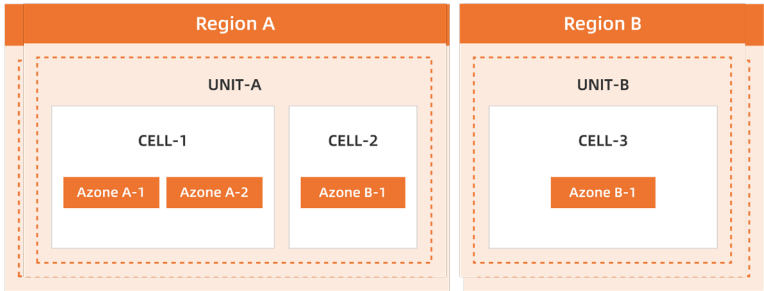


图 3 单元化模型关键词映射关系

4. 多活容灾产品支持架构及功能特点



图 4 MSHA 多活容灾产品能力全景图

(1) 同城多活架构

- **架构定义**

同城多活，分布在同城的多个站点同时对外提供服务，适用于一云多 AZ。机房间物理距离较短，可以接受跨机房的网络延迟。同城多活架构，将业务恢复和故障恢复解耦，基于灵活的规则调度、东西及南北走向流量管控等能力，保障故障场景下的业务快速恢复，助力企业的容灾稳定性建设。

在云产品同城容灾的基础上，构建上层应用的同城双活，这是容灾建设的首选架构，该方案云产品覆盖范围广，建设成本低，同时能做到同城 RPO 接近于零。局限是 Region 级的故障概率仍然较高，以及主机房故障的恢复时间较长。

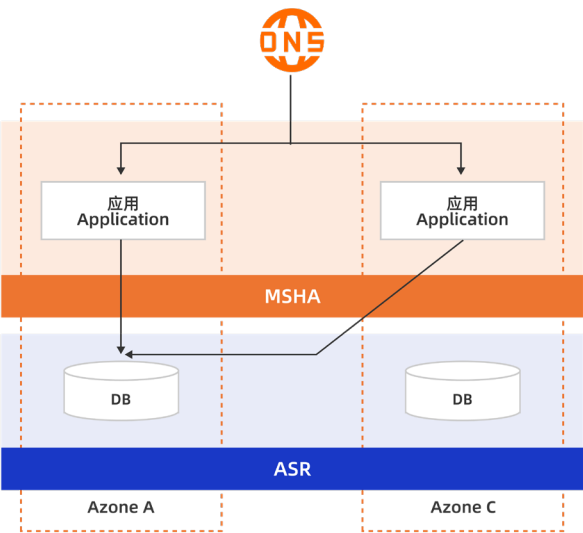


图 5 MSHA 同城多活架构

（2）异地应用双活架构

• 架构定义

异地应用多活，分布在异地的多个站点同时对外提供服务，适用于多云、一云多 Region。机房间物理距离较小，可以接受跨机房的网络延迟。异地应用多活架构，旨在解决有强一致诉求、业务尽可能不改动代码的情况下，业务具备同城、异地的高可用容灾能力，将业务恢复和故障恢复解耦，基于灵活的规则调度、跨域跨云管控等能力，保障故障场景下的业务快速恢复，助力企业的容灾稳定性建设。

当业务容灾要从单云扩展到多云或者多 Region，异地应用多活是首选架构。该方案容灾建设成本低，分钟级的 RTO 与 RPO。局限是机房距离不能太远，业务需要接受跨机房的 RT。

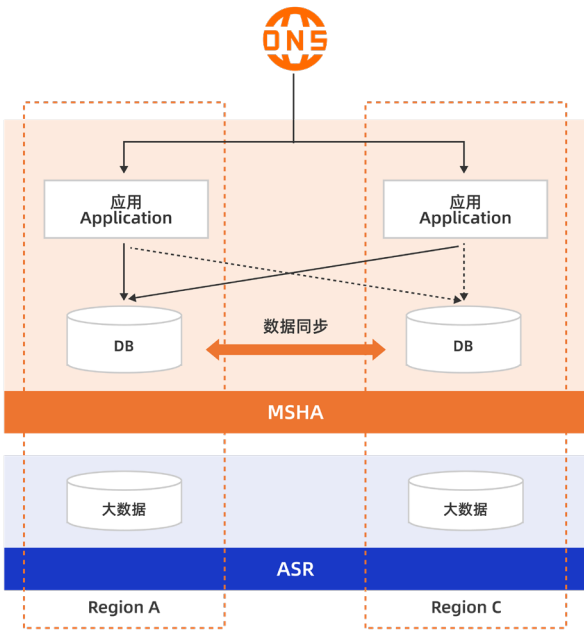


图 6 MSHA 异地应用双活架构

• 方案价值

- **同城 RPO 接近于零**：云上业务的连续性，是云平台稳定性与业务应用稳定性的叠加结果。业务上云，我们推荐使用云平台的 PaaS 服务，此处不考虑自建的场景。在单地域多可用区的云环境下，云产品自身的容灾要求即是同城多活，即保障在单可用区故障的场景下 RPO 接近于零。在数据类产品三 AZ 部署的场景下，可以实现 RPO 为 0。
- **同城分钟级 RTO**：在云平台同城多活的基础上，如果业务只做跨可用区的服务部署，容灾能力仍然有很大的欠缺，做不到快速的业务恢复。核心是云平台的稳定性不等于上层业务的稳定性，云平台的恢复不等于业务恢复。业务需要建立自身的流量管控能力，具备自上而下的流量隔离和快速切换能力。基于 MSHA 同城多活的流量管控，包括机房南北向流量和东西向流量，在机房故障场景下可以做到快速的流量逃逸，分钟级恢复。
- **容量突破机房限制**：同城多活架构，在应用层面支持同机房调用优先，减少网络抖动对业务的影响，提升业务整体 RT。支持基于逻辑可用区的架构定义，基于逻辑可用区的流量调度，在逻辑可用区之下可以扩展物理可用区。
- **以多活为基础的业务创新**：多活架构在保障业务系统高可用的同时，逐步发展成企业的基础设施，依托于多活架构能力，企业可以扩展出多种稳定性能力，如：全域变更强管控、全域故障演练、重点流量保障等。

• 方案价值

- **分钟级 RTO**：在应用多活架构下，本着“业务恢复时间”和“故障恢复时间”解耦的原则，在发生故障时，基于分钟级切流能力，优先恢复业务。在业务恢复的前提下，再进行故障定位修复。

- **容量突破地域限制**：应用多活架构，在路由一致性和数据一致性的基础上，支持跨地域的集群拆分和灵活流量调度，从而让业务能够突破地域限制，做到跨地域的容量水平扩展，从而解决单地域下的容量挑战。
- **以多活为基础的业务创新**：同同城多活架构。

(3) 异地多活（单元化）架构

- **架构定义**
 - 异地多活单元化，简称异地多活，分布在异地的多个站点同时对外提供服务，适用于多云、一云多 Region。机房间物理距离较大，不能接受跨机房的网络延迟。
 - 异地多活架构是在阿里巴巴电商业务环境演进出来的容灾架构，旨在跨 1000 公里以上的多数据中心建设多活容灾架构。遵从单元化路由规则，每个数据中心承担部分用户的流量，单数据中心故障无全局性故障风险，基于路由规则变更达到快速流量切换效果。

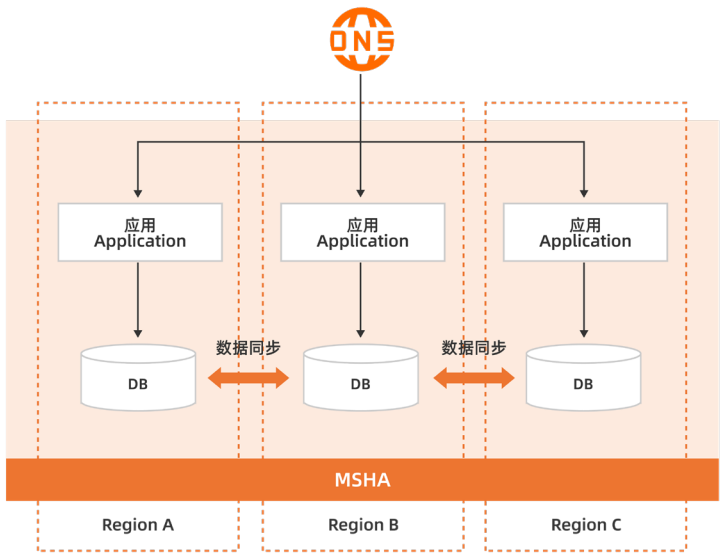


图 7 MSHA 异地多活（单元化）架构

• 适用场景

异地多活本质上就是通过自上而下的全域流量隔离来解决数据同步延时无法突破物理限制的问题，但异地多活也不是银弹，也有其适合场景，且不同的场景下异地多活的成本不尽相同，下面把用户的业务按数据维度分成三种类型来看看异地多活的常见适用场景。

a. 读多写少型业务

- 业务应用场景：资讯、导购类的服务，比如商品浏览、新闻资讯等。
- 数据特点：读多写少，用户以浏览为主，核心业务场景只读，单元里部署的都是只读业务。
- 多活接入成本：成本低，只需要用户在请求里可以标记上分流的标识即可。

b. 流水单据型业务

- 业务应用场景：电商交易、帐单流水类服务，比如用户的订单、用户的通话记录等。
- 数据特点：数据可以按照一定的维度进行分片且可以接受最终一致。
- 多活接入成本：成本略高，用户需要梳理业务，理出单元内部署的核心业务及数据，对于单元依赖且无法拆分的业务采用读写分离，然后按照多活接入规范重点对服务层及数据层进行相应改造。

c. 状态依赖型业务

- 业务应用场景：银行账务，比如 A、B 在某银行均有帐户，A、B 数据分片位于不同的数据中心，A 和 B 之间会有转账行为。
- 数据特点：数据有状态依赖且无法最终一致，数据还存在跨数据中心的交互。
- 多活接入成本：成本高。

• 异地多活设计原则

a. 选取分区维度

选择一个数据维度来做数据切片，从而实现业务可以分开部署在不同的数据中心。

b. 确定改造范围

选择与选取的数据维度相关的业务范围来做多活。

c. 单元封闭

尽量让调用发生在本单元，尽量避免跨数据中心的调用，一方面为了用户体验，本地调用 RT 更短，另一方面为了稳定性，防止一个数据中心出了问题，其它数据中心受影响。

d. 无法接受最终一致的数据要进行单点写

对于一些实时性要求极高，无法接受最终一致的数据只能进行单点写。

• 方案价值

- **分钟级 RTO**：恢复时间快，外部客户生产系统恢复时间平均在 1 分钟。
- **资源充分利用**：资源不存在闲置的问题，多机房多资源充分利用，避免资源浪费。
- **切换成功率高**：依托于成熟的多活技术架构和可视化运维平台，相较于现有容灾架构，切换成功率高，阿里内部年切流数千次的成功率高达 99.9% 以上。
- **流量精准控制**：应用多活支持流量自顶到底封闭，依托精准引流能力将特定业务流量打入对应机房，企业可基于此优势能力孵化全域灰度、重点流量保障等特性。

5. 小结

MSHA 与 ASR 产品相辅相成，共同为云上业务打造全面的容灾保护体系。目前，ASR 和 MSHA 产品已经在政务、运营商、能源、交通、医保、金融等行业客户得到广泛应用，成长为支撑客户业务稳定性的中坚力量。未来，ASR 和 MSHA 产品会与专有云一起不断的进行产品功能演进，在全产品容灾覆盖、混合云容灾、异构容灾等场景持续发力，更好的服务千行百业，让容灾发挥更大的业务价值。

(三) 主要多活解决方案

1. 容灾建设必要性

在数字化转型的浪潮中，信息系统的安全运行已成为不容忽视的战略高地，直接关系到国家安全和人民利益，更关系到社会的稳定。国家法律与行业规范纷纷构筑起坚固防线，以保障信息运营主体的责任落实与数据安全。然而，各行业的特性各异，信息系统与数据管理方式的差异性，也催生了多样化的容灾需求与解决方案。

“居安思危，思则有备，有备无患”。事实证明，各类灾难的到来通常是突发性的、偶然性的，但只要能够进行行之有效的灾难恢复工作，就可以将损失降到最小。

2. 容灾解决方案

十年来，专有云容灾经历了从青涩到成熟的蜕变，在各行各业的实践中生根发芽、遍地开花，成长为支撑客户业务连续性的坚实后盾。从全国医保信息系统的应用容灾建设，到金融领域的业务连续性考验，从政务云的容灾体系构建再到电力等行业的关键基础设施保护，政企客户典型的 5 大容灾场景为：同城容灾、同城双活、异地主备容灾、异地应用双活、异地多活（单元化）。

容灾架构的演进，有时间、财务和技术的成本，并非一蹴而就。选择什么样的架构演进路线，核心在于以合适的成本满足本期项目的建设目标，同时兼顾未来的扩展性。下图介绍了两种常见的演进路线：

- 路线 1：单中心 -> 同城双中心（追求极致 RPO，可扩展成三中心） -> 两地三中心 -> 异地多活。
- 路线 2：单中心 -> 异地双中心（追求极致 RPO，可扩展成三中心） -> 两地三中心 -> 异地多活。

从统计学角度分析，单个中心故障的概率，大于城市遭遇灾害的概率。因此在许多项目中，综合考虑到时间、经济和技术的成本，通常会优先选择路线 1，即首次开展容灾建设优先选择同城容灾的架构。

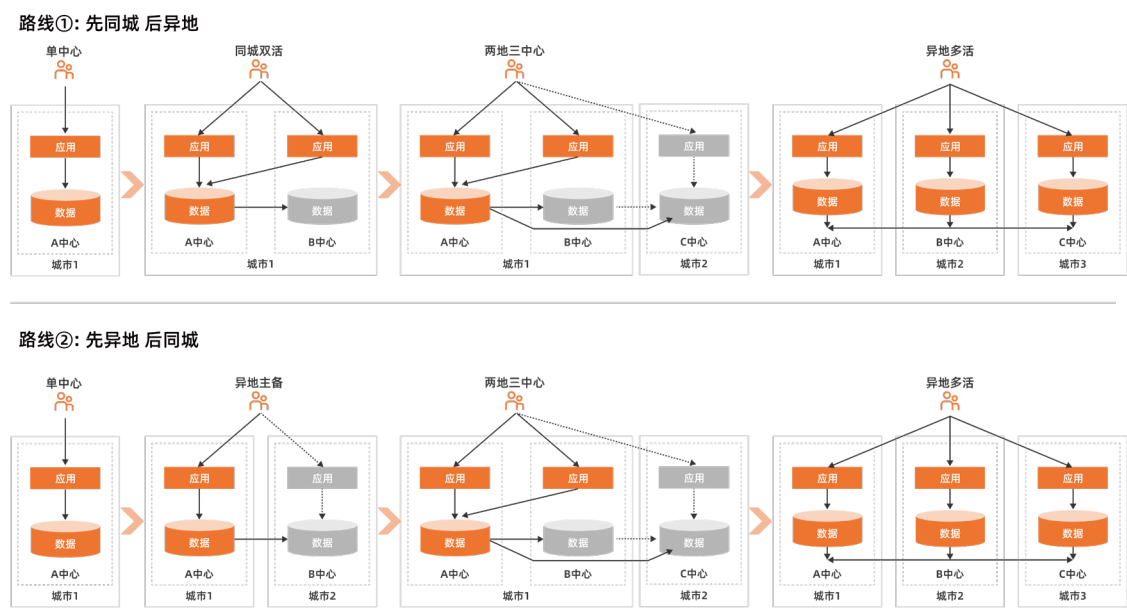


图 1 两种常见的容灾建设演进路线示意图

从用户视角看，同城容灾、同城双活、异地主备容灾、异地应用双活、异地多活，都需要遵循一定的架构前提，需要统筹考虑业务容灾目标、防范的灾难类型、投入成本、机房选址、业务适配成本、数据同步技术、容灾切换等几个方面看。典型容灾架构的主要特征对比如下：

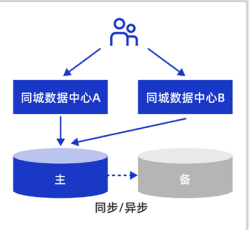
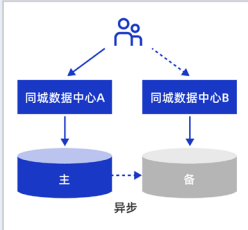
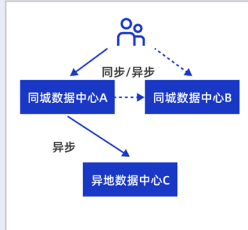
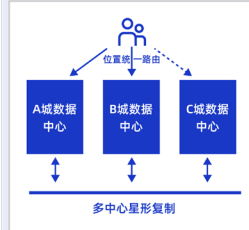
同城容灾	异地容灾	两地三中心	异地多活
主要特征： - 据中心距离≤50KM - 业务侵入少 - 数据采用同步+异步两种同步方式 - 数据写入到生产中心集群主 - 可应对机房级故障	主要特征： - 数据中心距离≥100KM - 业务侵入少 - 数据采用异步同步方式 - 应用和数据库均是热备 - 支持异地部署 - 可应对城市级故障	主要特征： - 数据中心距离≥ 100KM - 有一定的业务侵入 - 数据采用同步+异步两种同步方式 - 数据从同城双中心写入 - 数据的可靠性能力提升 - 可以应对城市级故障	主要特征： - 数据中心距离≥ 100KM - 应用配合单元化改造 - 涉及数据强一致的应用，容灾机制依赖于同城/异地容灾 - 可应对多城市故障
			

图 2 典型容灾架构主要特征对比示意图

(1) 同城容灾

同城容灾，顾名思义就是分布在同城的多个站点，存在一个站点对外提供服务，其他站点作为备份站点不对外提供服务。当一个站点发生灾难或者业务流量调整时，另一个站点可接管流量，对外提供服务。

• 场景描述

- **适配行业 and 场景：**相对传统的行业，有容灾建设的监管要求，但由于应用系统暂时无法完成无状态化改造，处在容灾建设的演进过渡中。待条件具备后，可演进到同城双活。
- **基础架构：**一朵云，双 AZ 架构（可扩展为 3AZ，数据库强一致）。机房距离一般小于 50KM（RT<1.9ms）。
- **预算：**中等。
- **改造成本：**业务梳理改造少，实现复杂度低。仅适用于部分有状态应用，无法做到双机房双活运行的场景。

• 方案架构示意图

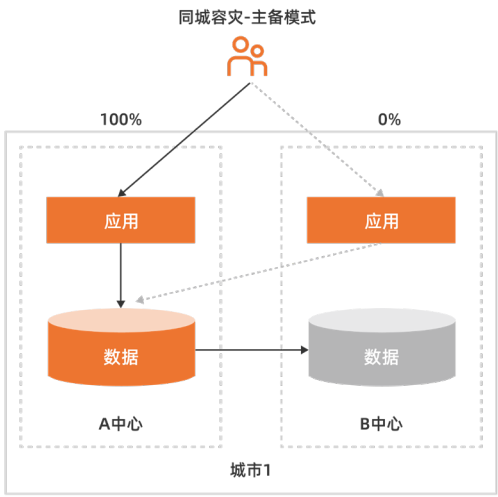


图 3 同城容灾架构示意图

• 方案描述

同城双机房一朵云，业务系统在双机房对等部署、数据主备（日常态，备机房无流量引入），灾难时利用 ASR 对云平台进行一键切换。

• 容灾产品 & 服务能力

- ASR 实现云平台 / 云产品的统一化容灾管控、故障切换及演练。
- 面向云平台 + 应用整体提供容灾架构设计和改造、适配方案，并支持陪跑实施以及相关的整体容灾演练验收。

(2) 同城双活

同城双活，顾名思义就是分布在同城的多个站点，所有站点均对外提供服务。当一个站点发生灾难或者业务流量调整时，另一个站点可接管流量，对外提供服务。

• 场景描述

- **适配行业和场景：**具有行业普适性，通常采用云原生架构设计，例如互联网、医保、金融、能源等行业；建设容灾的同时，也希望容灾站点资源得到充分利用。
- **基础架构：**一朵云，双 AZ 架构（可扩展为 3AZ，数据库强一致）。机房距离小于 50KM。
- **预算：**中等。
- **改造成本：**业务梳理改造适中，实现复杂度低。

• 方案架构示意图

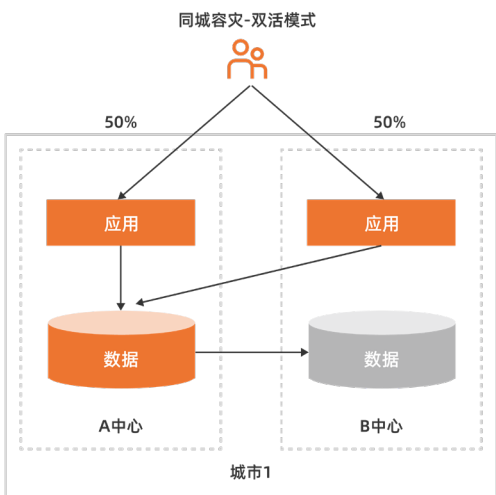


图 4 同城双活架构示意图

• 方案描述

同城双机房一朵云，双机房互为灾备，业务系统在双机房双活运行、数据主备。基于 MSHA 接入层网关可实现入口流量的精准路由控制。灾难时利用 MSHA+ASR 进行云平台及云上业务流量的一键切换。实现分钟级 RTO。

• 所需产品和服务支撑

- ASR 容灾管控平台实现云平台 / 云产品的统一化容灾管控、故障切换及演练。
- 面向云平台 + 应用整体提供双活架构设计和改造、适配方案，并支持陪跑实施以及相关的整体容灾演练验收。

(3) 异地主备容灾

异地灾备，顾名思义就是分布在异地的多个站点，存在一个站点对外提供服务，其他站点作为备份站点不对外提供服务。当一个站点发生灾难或者业务流量调整时，另一个站点可接管流量，对外提供服务。

• 场景描述

- **适配行业和场景：**适用于对核心系统有建设异地容灾的强监管要求，如金融等行业；有一些行业客户，生产站点、灾备站点由不同的运营商承建和运营管理，如一些政务云等。
- 基础架构：**一云多 Region 架构，跨 Region 主备容灾；两朵云，跨云主备容灾。机房距离大于 100KM。
- **预算：**中等。
- **改造成本：**业务梳理改造少，实现复杂度低。

• 方案架构示意图

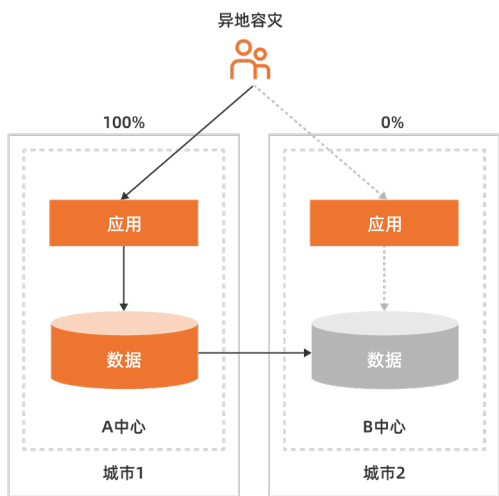


图 5 异地主备容灾架构示意图

• 方案描述

异地两朵云（或者一朵云两个 region），按业务系统做应用主备容灾，备机房部署 ASR-DR。日常态只有主云对外提供服务，故障时借助于 ASR-DR 进行应用资源组的切换，并调配云外 DNS 权重，切换两个机房的外网写入流量。

• 所需产品和服务支撑

- 配置 ASR-DR 异地容灾管控平台，基于应用视角，对用户应用提供数据保护、日常演练和故障恢复的功能。
- 面向云平台 + 应用整体提供双活架构设计和改造、适配方案，并支持陪跑实施以及相关的整体容灾演练验收。

（4）异地应用双活

异地应用双活，顾名思义就是分布在异地的多个站点，所有站点均对外提供服务。当一个站点发生灾难或者业务流量调整时，另一个站点可接管流量，对外提供服务。应用双活，表示应用组件做到双活。

• 场景描述

- 适配行业和场景：机房延迟大于 1.9 毫秒，同时希望提升应用异地资源利用率，对容灾建设效果有更高的创新演进需求，如电力等行业。
- 基础架构：一云多 Region 架构，跨 Region 应用双活数据主备容灾；两朵云，跨云应用双活数据主备容灾。机房距离大于 100KM。
- 预算：稍高。
- 改造成本：业务梳理改造适中，实现复杂度适中。

• 方案架构示意图

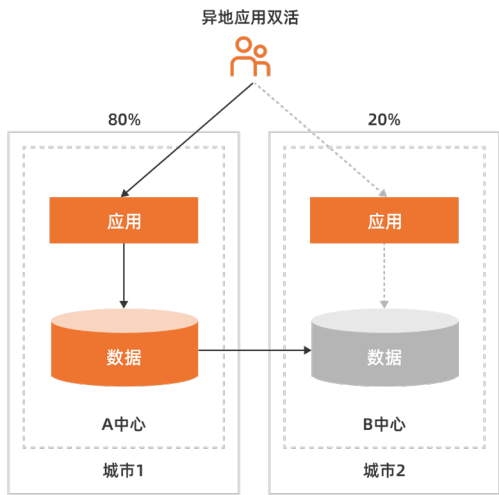


图 6 异地应用双活架构示意图

• 方案描述

异地两朵云（或者一朵云两个 region），采用应用双活、数据主备架构。借助 MSHA 流量多活组件，根据域名、IP、标签等规则进行流量精准路由控制，借助 ASR-DR 应用及数据管控平台，实现应用层双活、数据层主备同步及容灾切换能力。

• 所需产品和服务支撑

- 选项一：
 - 配置 ASR-DR 异地容灾管控平台，基于应用视角，对用户应用提供数据保护、日常演练和故障恢复的功能。依托于专有云 DNS，支持云上应用流量比例调配。
 - 服务面向云平台 + 应用整体提供双活架构设计和改造、适配方案，并支持陪跑实施以及相关的整体容灾演练验收。

选项二：

- 在 ASR-DR 基础上，为了实现更高的业务连续性，配置 AHAS-MSHA，基于业务属性进行精细化切流，并有效解决 DNS 缓存生效慢问题，提供更多灵活性。
- 面向云云平台 + 应用整体提供双活架构设计和改造、适配方案，并支持陪跑实施以及相关的整体容灾演练验收。

(5) 异地多活（单元化）

- 异地多活，顾名思义就是分布在异地的多个站点，所有站点均对外提供服务。当一个站点发生灾难或者业务流量调整时，另一个站点可接管流量，对外提供服务。在实现应用组件双活的同时，数据也是双活的。

场景描述

- 适配行业和场景：适用于对业务稳定性要求很高，平台规模很大等客户场景，比如头部电商、运营商、央国企等。
- 基础架构：异地两 / 多云，实现应用双活、数据双活，核心业务连续性要求极高。机房距离大于 1000KM。
- 预算：很高。
- 改造成本：业务梳理改造高，实现复杂度高。

方案架构示意图

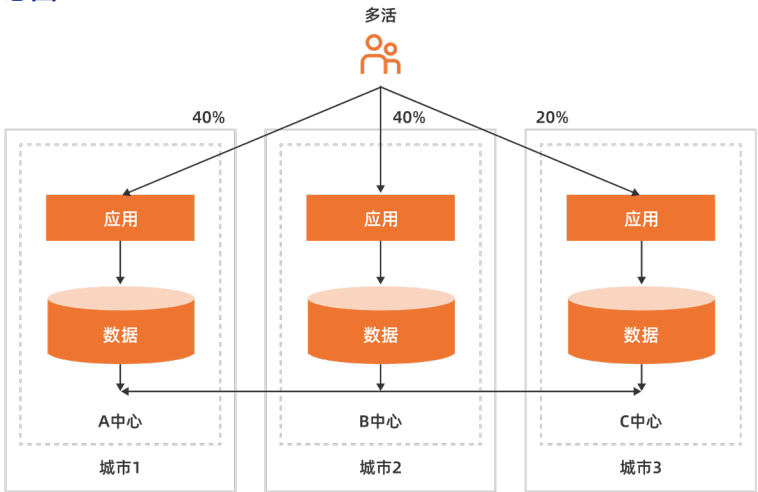


图 7 异地多活（单元化）架构示意图

方案描述

异地两 / 多云，对业务请求及数据进行分片，实现应用双活、数据双活。当发生云平台、机房或业务故障时，可迅速将故障机房流量切换到正常机房，实现双活切换，实现 RPO 分钟级，RTO 分钟级的容灾建设效果。

所需产品和服务支撑

- 配置 AHAS-MSHA 实现单元化应用异地多活。对于数据强一致性要求高无法单元化的中心服务可配置 ASR-DR 实现异地应用容灾。
- 面向云平台 + 应用整体提供多活架构设计和改造、适配方案，并支持陪跑实施以及相关的整体容灾演练验收。

3. 未来容灾新方向

混合云容灾策略：构建无缝衔接的安全架构

混合云容灾未来一定是容灾发展的方向，它既可以融合公共云的弹性又能结合专有云的可控性，为企业打造一套既能应对技术挑战，又能确保业务连续性的安全架构。

智能化容灾服务：提升响应速度与决策效率

AI、大模型、5G-A 等技术的落地场景越来越丰富，作者也相信智能化容灾服务也会逐渐成为提升容灾响应速度与决策效率的关键因素。通过 AI 算法，能够预测潜在的故障点，提前进行资源调配，实现容灾预案的动态调整。同时，利用 AI 驱动的自动化容灾流程将能够显著减少人为干预，确保在紧急情况下，业务能够快速恢复至正常状态。

容灾运维运营：融合 DevOps 与 SRE 理念

在容灾管理领域，DevOps 与 SRE（Site Reliability Engineering）理念的融合正成为一种新的趋势。持续监控与优化实践，确保容灾策略与业务需求的紧密贴合，通过定期的容灾演练与评估，不断优化容灾计划，提高容灾恢复的效率与成功率。

同时，在架构演进的过程中，通过容灾保鲜，可有效防止容灾能力下降。通过引入混沌工程机制，通过模拟近真实的故障，观察在容灾应急过程中相应的人员、流程、系统的表现，快速识别容灾应急的短板，持续改进，达到容灾保鲜的效果。

4. 结束语

数据中心级别容灾的基本前提，是为生产中心建设冗余的灾备中心，来应对未来的风险。数据中心作为新基建的重要组成部分，得到了国家的大力支持，是千百行业的信息化底座，也是推动数字经济发展的的重要驱动力。

在数据中心的建设、业务系统运行和运维阶段，需要严格按照国家的等保要求和行业监管要求，对数据中心及其信息系统的建设以及可能面临的系统性风险，进行业务连续性的冗余设计和容灾演练。

(四) 某医保局异地容灾建设最佳实践

本文选择了某医保客户项目作为容灾交付典型案例，重点介绍交付经验，供相关行业及组织来参考。

1. 背景介绍

对某医保客户来说，医保信息系统要充分利用云计算的技术来建设容灾能力，要解决三个重要的问题：

- 第一，医保局和医保系统的开发商对于云平台为基础的容灾体系建设缺乏成熟的工程经验，需要有云厂商的共同参与。
- 第二，由于医保系统本身的业务特殊性，建设标准不低于国家标准的 5 级容灾，因此需要有全面和强大的产品和技术服务能力。
- 第三，该客户的建设进度在全国处于领先水平，要求后续可顺利对接到国家医疗保障局的系统，因此需要云计算平台厂商，应同时具备国家医保信息系统建设标准的制定和实施的经验。

2. 建设目标

该项目关键的建设范围包括：

- 根据国家医保局统一规划的技术标准规范，提供云计算产品、技术服务和运维服务。
- 根据国家医保局统一规划的容灾演练验收标准，提供容灾架构设计和实施，分别进行应用层和数据层的容灾演练。

客户采用的技术架构，满足了《 医疗保障信息平台云计算平台规范 》（ YB-XJ-A01-2019 ）中关于技术架构的要求，如下：



图 1 医疗信息系统平台总体技术框架

本项目结合国家医保局提出的验收要求，沉淀出一套标准的交付物，并在后续其他省市级医保项目中得到广泛应用，为医保行业的信息化建设总结了宝贵的经验。本项目中关键的交付流程和交付物如下：



图 2 医保系统容灾项目交付清单

3. 交付过程最佳实践

（1）交付流程标准化

在项目经理、系统架构师、技术经理、开发人员、运维人员等角色的有效配合下，基于项目交付里程碑节点要求、项目执行计划及项目管理章程，高质量高效率推进了项目的交付。如下：

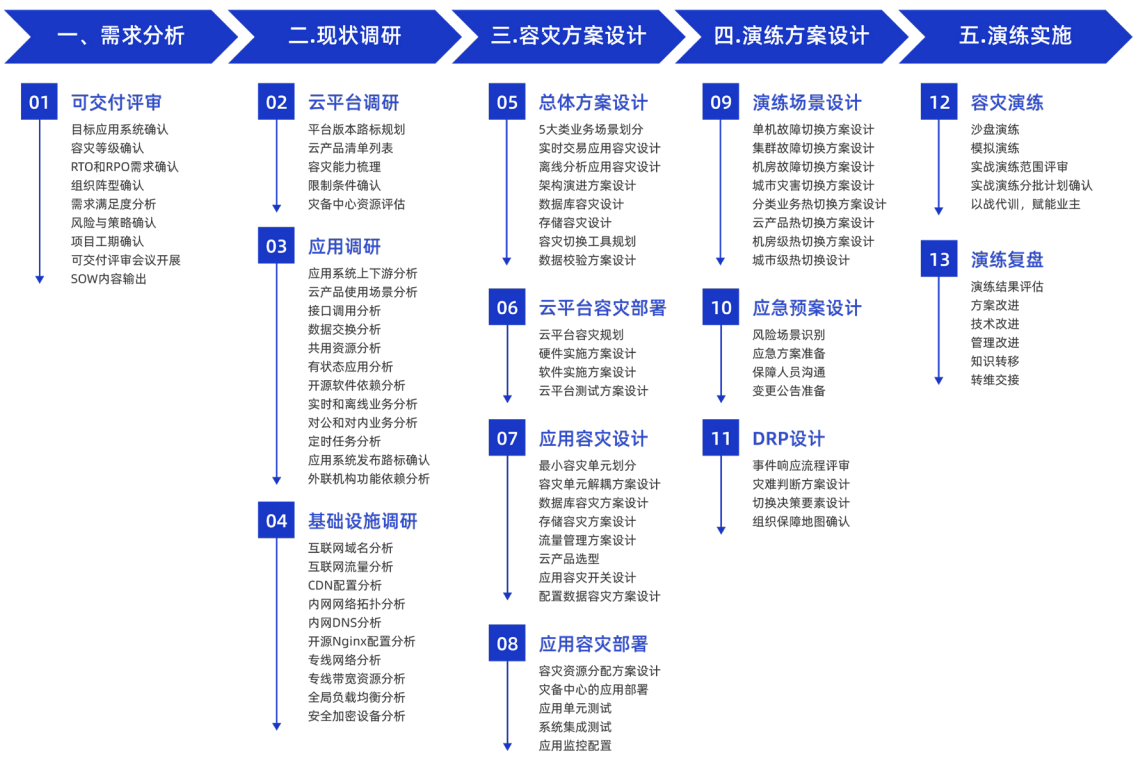


图 3 医保容灾项目交付过程

（2）交付过程工具化

在本项目交付过程中，除了流程标准化外，容灾交付的能力进行产品化和工具化，也成为效率大幅提升的重要手段。随着产品 / 工具功能不断增强，覆盖的范围会持续提升。数据同步产品、平台容灾切换产品、应用容灾工具和应用仿真工具是典型代表，详情如下：

工具	功能描述	在本项目中的价值
数据同步产品	数据传输服务 DTS (data transmission service)，支持关系型数据库、NoSQL、OLAP 等数据源间的迁移、同步、订阅，轻松实现多数据源实时数据同步、不停机上云或异地多活数据中心等，助力企业构建安全、可扩展、高可用的数据架构。	- 实时同步：DTS 跨两朵云的 RDS MySQL 数据实时同步。 - 反向同步：容灾切换后，DTS 提供反向跨云同步功能。 - 过滤噪音：DTS 提供临时数据表的过滤功能，减轻同步压力。 - 离在线一体：DTS 支撑离线在线数据的实时同步，支撑离线数据和在线数据的分离，降低数据库压力。
平台容灾产品	灾备控制台 ASR-DR (apsara stack resilience disaster recover)，是一款提供容灾管理的云平台产品，支持 RDS、OSS、TableStore、DRDS、DNS 和 MQ 的热备容灾。	- 容灾大屏：展示保护组、恢复计划、演练计划、云产品保护等概览信息。 - 应用容灾：支持按应用维度管理保护组，建立最小粒度的数据同步关系。 - 容灾切换：批量执行容灾计划，包括演练切换、演练回切、故障切换和故障回切。 - 告警管理：支持对保护组及计划的各种状态设置告警规则，并将告警信息通知到告警联系人。

表 1 容灾平台和工具

产品及工具本身融合了设计规范和交付产出模板，其产出内容严格按照设计要求，避免人工操作的差异性和不规范性，是交付质量的重要保障手段。同时，工具使用简单、便捷，不需要强依赖经验丰富的专业人员，有效降低了容灾切换的风险，降低了 RTO 和 RPO。

（3）交付成果保鲜化

容灾项目的交付成果，指目标应用系统的容灾能力。当发生数据中心级别的重大风险或灾难时，快速完成容灾切换来保障业务连续性，是检验容灾能力的终极手段。重大风险和灾难，是一种不确定性的概率事件。如何让容灾演练已经验证过的能力，持久化生效，是交付成果保鲜化的重要话题。变化就像一只看不见的手，容易被忽略，并影响着应用系统的容灾能力。本文用“容灾能力保鲜度”来描述，从演练切换到灾难切换的过程，保鲜度是如何随着时间逐步变化的。特别需要注意的事项包括：

- ① 复盘 & 改进完成的时候，是保鲜度最高的时候，要求对演练发现的问题进行修复后才能达到。
- ② 系统日常运行过程中，随着各种软硬件设备的变化，会造成保鲜度降低，而且容易被忽略。

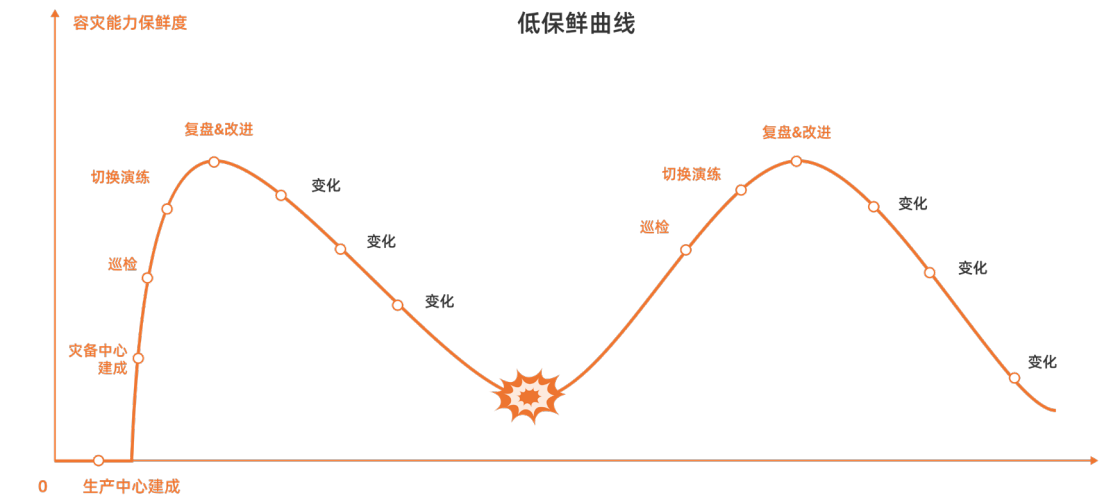


图 4 容灾能力低保鲜曲线

容灾能力保鲜的关键是用计划性的容灾演练和日常运维规范来应对不确定的重大风险或灾难。容灾能力保鲜的过程，存在四种状态，分别是：容灾演练、复盘 & 改进、变化、巡检，四种状态循环演进。针对每种状态，在该项目中，开发、运维和管理部门，严格执行如下标准动作。

a. 演练状态：指生产环境的容灾切换和回切的演练，要求如下：

- 检查业务的连续性，确保在计划的窗口期内恢复业务正常。
- 检查数据的完整性，对历史业务数据和增量业务数据进行校验，确保生产中心和灾备中心的数据一致性。
- 检查方案的可行性，确保容灾切换演练的步骤清晰并易操作。

b. 复盘 & 改进状态：指每次演练结束后，对演练过程进行总结，要求如下：

- 收集业务部门的反馈，确认演练过程的业务影响范围和程度，处于可接受范围之内。
- 检查组织协同的效率，确认通告范围是否完整和响应机制是否健全。
- 检查演练结果是否符合预期，并开展回归验收。
- 分析演练中暴露的问题，找到对应的解决方案，并推进改进方案落地。

c. 变化状态：变化是持续存在的，发生变化后，需及时确认是否遵循容灾规范并完成适配，要求如下：

- 记录并检查硬件设备的变化情况，包括网络设备、安全设备、服务器、专用硬件等，对其高可用配置和容灾配置进行确认。
- 记录并检查应用系统的变化情况，包括新的应用系统、新的运行版本、新的云产品、新的渠道系统等，并对其高可用配置和容灾配置进行确认，对容灾规范满足度进行确认或改造。
- 定期检查硬件维保期和软件 License 的有效期，及时更新即将过保的硬件和软件 License。
- 对关键人员的变化，及时安排好能力交接。

d. 巡检状态：指日常巡检和演练巡检，要求如下：

- 全面监控生产中心和灾备中心的工作状态，及时响应和处理告警信息。
- 定期进行容灾切换的深度巡检。
- 演练规划时，对应用系统、云平台 and 基础设施，进行全链路分析。
- 演练开始前，多次执行深度巡检，及时清理潜在风险。

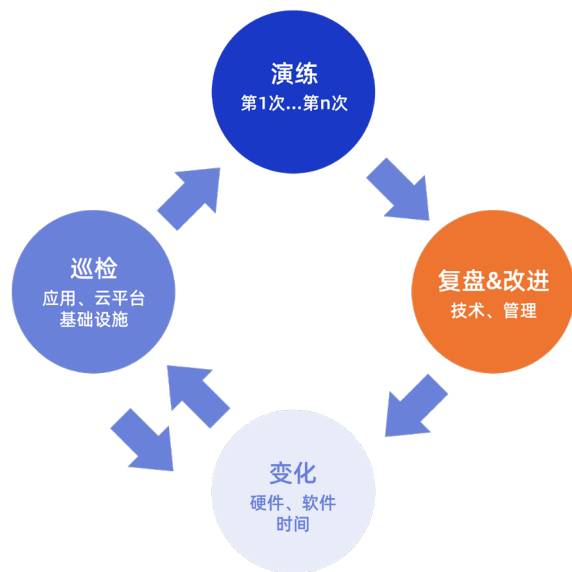


图 5 容灾能力保鲜曲线变化过程

在完成医保系统的多次演练后，项目达到验收条件。在运维过程中，业务系统的设计、开发、部署、运维，始终按照容灾切换演练积累下来的方法论，保障了医保系统的容灾能力处于“高保鲜”的状态。

在容灾系统建设完成后的正常运行期，该地区医保系统曾面临一次生产数据中心即将停电的风险。在医保局决策和指导下，运维人员顺利将业务系统从生产中心切换到灾备中心，成功避免了系统性风险，保障了医保系统的业务连续性。这充分证明了日常开发和运维规范的执行到位，对于维持容灾能力的“高保鲜”的状态是非常重要的。该项目中容灾能力的保鲜曲线如下图所示。

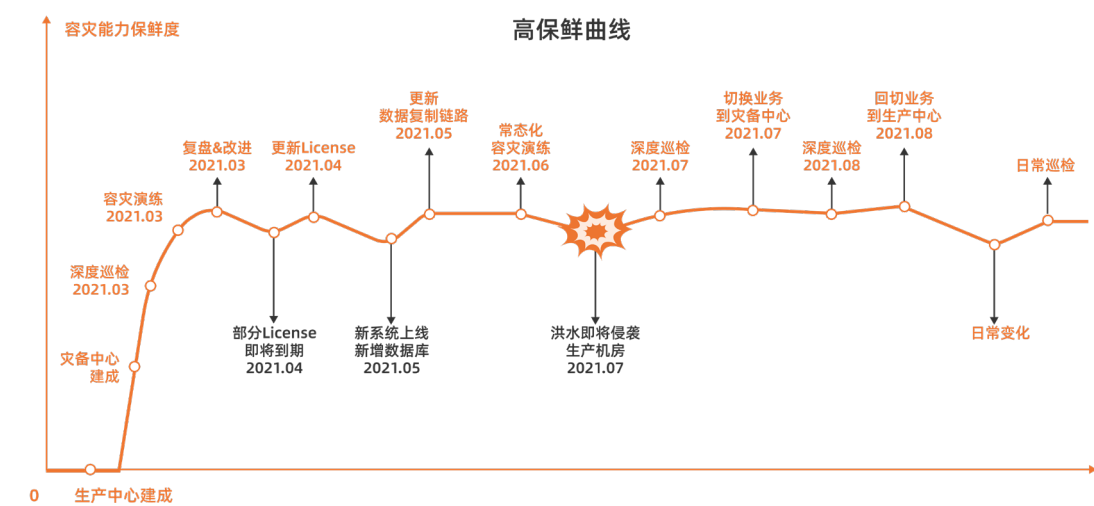


图 6 容灾能力高保鲜曲线

4. 结束语

新业务、新应用、新技术、新流程、新人才和新规范等新兴变化，既是风险产生的主要诱因，也是推动创新的核心动力。我们应充分把握这些新要素，持续提升信息系统的容灾能力。在数字化转型加速的背景下，各机构不仅需要关注业务与技术的发展，还应高度重视人员管理与流程优化在容灾体系建设中的关键作用，以实现全面的风险防控与业务连续性保障。

(五) 某政企头部客户异地多活单元化方案最佳实践

本文选择某政企头部客户项目作为异地多活交付典型案例，重点介绍客户面临的稳定性痛点、多活建设目标以及单元化方案，供相关行业及组织来参考。

1. 背景介绍

某客户的 A 业务系统属于国家定义的关基系统，率先实践了异地双活单元化架构。通过阿里云长期的服务陪伴，不断的方案论证，突破一系列技术或方案挑战，最终成功将异地双活单元化架构在客户全公司层面全面推广，极大提升了客户业务连续性。随着双活建设的稳步推进，业务规模的高增增长，客户服务器规模也持续增长。

客户 A 业务系统已成为生产运营和业务发展的最核心系统，承载数亿 + 用户。客户在系统的稳定上已经进行了诸多尝试，如应用侧系统优化、应急预案等多种保障措施，依然无法彻底解决系统安全问题。主要面临的挑战如下：

- **高可靠性挑战。**支撑数亿 + 用户、每日数十亿 + 次的业务调用量，系统承载业务量巨大，对安全可靠提出更高要求。
- **单点故障隐患。**此前，由于单个机房汇聚交换机出现故障，导致全网业务中断数小时。
- **极端情况处理能力不足。**作为客户生产核心系统，面对地震、火灾等极端地域级灾难情况下，仍然会导致大面积业务中断以及数据恢复困难。

2. 业务多活建设目标

在一系列问题之下，容灾建设已经迫在眉睫，势在必行。客户双活建设，核心是要达到以下 5 项指标。

- **核心业务覆盖率。**影响客户核心业务办理和体验相关的实时业务 100% 双活全覆盖。
- **核心数据实时容灾。**系统核心资料及业务数据两地实时互备，避免数据丢失。
- **双 DC 常态承载业务流量。**常态下两个数据中心系统均承载流量，盘活冷备资源。

- **单数据中心可承载全国流量。**故障时单数据中心可承载系统核心业务全国流量。
- **故障分钟级切流恢复。**数据中心故障只影响部分地域，业务受理分钟级恢复。

3. 异地多活单元化方案实践

为了达成双活建设目标，客户 A 业务系统整体采用 MSHA 异地多活单元化架构方案，核心在接入层分流和数据层禁写，管好最上层流量和最下层数据库读写。

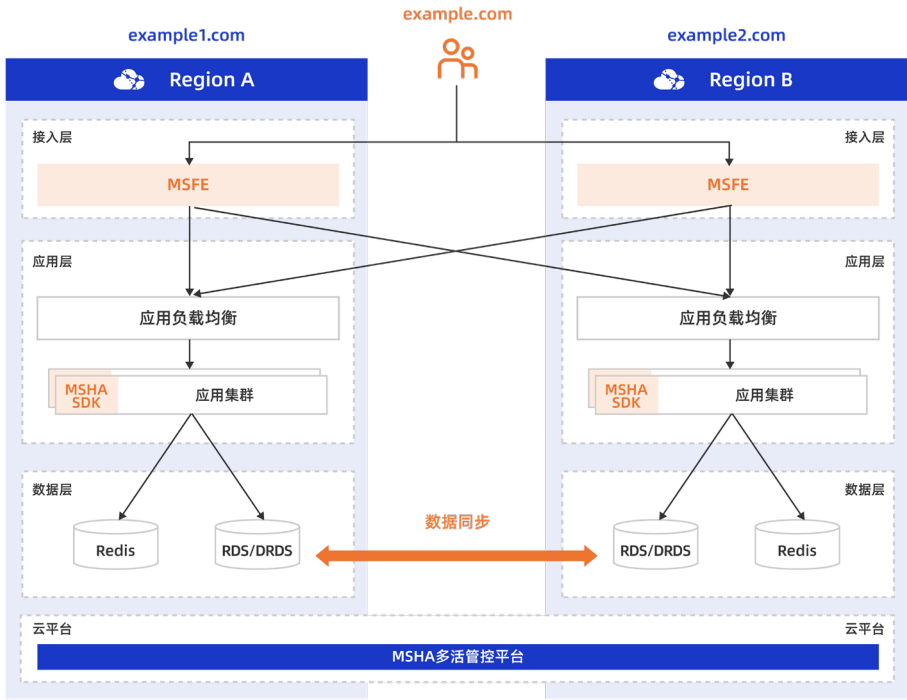


图 1 客户某 A 业务系统异地双活单元化架构图

(1) 单元化方案的核心要点

- **DNS 多单元解析：**创建生产主域名，创建生产各单元子域名。
- **流量接入网关：**采用 MSHA 接入层网关 MSFE 作为机房流量入口，由于 DNS 分流的随机性，MSFE 采用按某业务属性（该客户选用地理区域标识）路由规则在机房入口流量纠错。
- **流量带标：**单元化分流维度（该客户选用地理区域标识），可在 HTTP Header、Cookie 或者请求参数 Args（该客户选用 Cookie）中植入流量标。

- **数据双活：**数据库 RDS/DRDS 借助 DTS 产品进行数据双向同步，应用接入 MshaSDK 实现数据禁写保护。
- **缓存失效：**缓存背后接持久化数据库，数据库双向同步，监听数据库 binlog 实现缓存失效。

在整体单元化方案设计中，我们协助客户重点完成了**多活业务维度选型设计、数据分片设计、流量带标方案设计、微服务网关方案设计、微服务单元化方案设计、消息单元化方案设计、数据库方案设计**等。由于篇幅关系，下面章节选取部分技术方案进行阐述。

(2) 流量接入方案设计

客户接入层的多活方案，整体上采用 MSHA 的接入层流量网关。接入层是多活流量的前端入口，解析请求协议，并按照设置的流量分流规则，将请求流量引导到相应单元的后端应用。

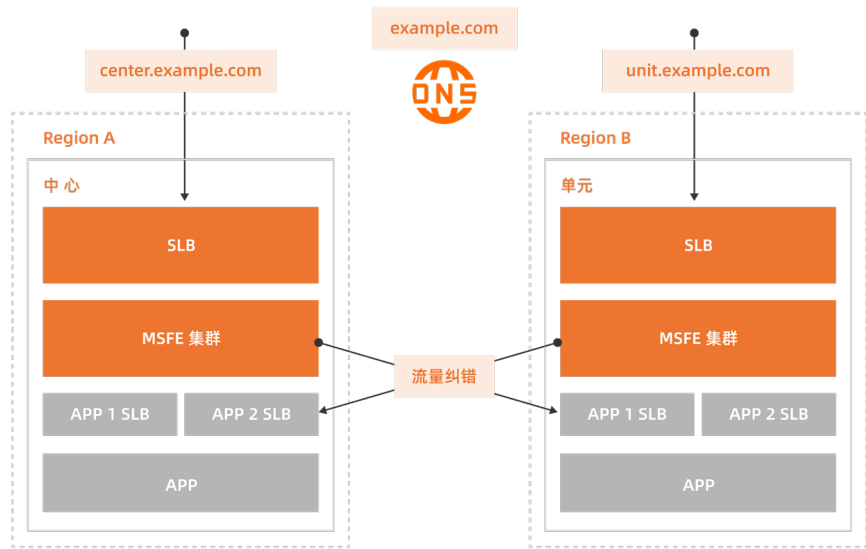


图 2 流量接入方案示意图

客户接入层功能实现可以概括为以下 2 点：

- **机房粒度流量路由。**多活流量必须经过接入层，以便在故障发生时，通过接入层切换入口流量。
 - **精准路由：**客户流量层采用精准路由方案，在 HTTP 请求 Cookie 中携带了路由标识（该客户选用地理区域标识），以及配置好的路由规则，将流量按标路由到对应的 Region。

- **应用粒度流量路由（流量自定义回源）：**客户按照不同的域名与 URI 组合，将流量分流到对应的后端应用。

(3) 微服务方案设计

客户微服务多活方案，按照多活容灾的服务改造指引，将服务定义为三类：

- **单元服务：**该服务的调用必须遵循单元规则，服务调用根据 RouterID 来进行正确的选址和调用保护。只有服务 Provider 所在单元和 RouterID 归属单元一致，服务提供者才能正常处理请求。
- **中心服务：**该服务只能在中心进行发布，所有其他单元依赖中心服务的，都要去中心进行服务调用。
- **普通服务：**该服务是指无法按选中的单元维度进行拆分但又被单元服务所依赖的服务，一般是只读服务。

为了实现服务的跨云调用，RPC 框架层面直接实现了跨云的服务发现和服务路由：

- **跨云服务发现：**服务 Provider 通过多发布的方式，对多个注册中心实现多发布。
- **单元服务寻址：**单元服务调用，提取 RouterID 按照单元规则寻址到正确单元的 Provider。
- **中心服务寻址：**所有单元都到中心寻址中心服务进行调用。

(4) 消息方案设计

在多活之前，消息发送与消费在云内封闭。消息多活的基础是消息在单元间同步，并进一步按照消费 CID 的属性决定是单元化规则消费还是全量消费。

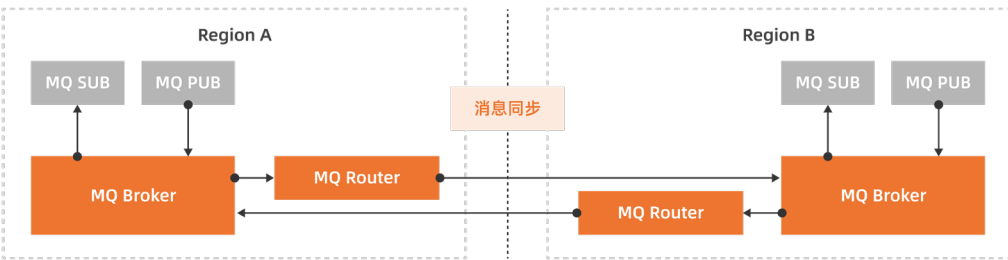


图 3 消息方案示意图

在该案例中，客户 MQ 消息的多活功能可以概括为以下几点：

- **消息双向同步。**支持以 Topic 粒度配置消息的跨单元双向同步。
- **按单元化规则消费消息。**消息即是数据，单元用户需要在正确的单元内消费消息。
- **中心业务消费所有单元消息。**某些中心非多活业务，需要能消费所有单元的消息。
- **切流用户继续消费在原单元堆积的消息。**消息有异步及堆积的特性，如果原单元消息有堆积，切流用户在新单元需要能继续消费堆积的消息。

由于消息的同步延迟，重置位点等特殊特性，消息一定会被重复消费，而且故障切换场景可能会存在较大量的消息重复消费。因此，对于异地多活的消息改造，需要遵循消息同步规范、消息发送规范、消息消费规范等要求。

（5）数据库方案设计

客户数据层的多活方案，整体上使用 MSHA 提供的标准化产品能力。客户数据库主要使用 RDS、DRDS 两种类型，整体方案简述如下：

- RDS/DRDS 要求全局唯一 SEQ。
- 通过 DTS 建立双向同步链路。
- 通过 MshaSDK 实现禁写控制。

异地多活的数据库多活，在数据同步的基础上，基于数据保护规则对数据库进行禁写控制。数据保护规则将数据表定义为两种类型：

- 单元表，单元服务相关的数据表，按照单元化规则实现禁写。
- 中心表，中心服务相关的数据表，中心写单元只读。

在表属性定义的基础上，数据保护规则将数据库定义为三种类型：

- UNIT 类型，库中所有表均为单元表，中心单元双向同步。
- COPY 类型，库中所有表均为中心表，中心单元单向同步。
- MIX 类型，库中同时存在单元表和中心表，中心单元双向同步。

4. 结束语

A 业务系统完成异地多活单元化建设后，历次大规模双活容灾演练，业务系统秒级切换。目前，客户持续保持着月度常态化演练的机制，平均半年切换几十次，多次在生产故障场景下发挥关键作用。

异地多活单元化架构是提升跨地域容灾能力的进阶解决方案，阿里云在这一领域长期深耕细作，与客户携手并进，共同成就了众多成功的案例。对于阿里云而言，通过协助客户构建业务容灾体系，我们不断磨砺和完善自身的 product 和技术能力。而对于客户来说，异地多活单元化架构不仅有效保障了业务的容灾需求，还显著提升了业务连续性和稳定性。

尽管异地多活单元化架构具备强大的容灾能力，但其实施成本相对较高，这不仅是对方案和技术可行性的考验，也要求客户具备充分的组织保障和资源投入。然而，凭借我们在异地多活产品及架构方面的丰富经验，以及在客户最核心系统商业化实践中的深厚积累，我们有信心在企业建设高标准多活容灾的道路上提供强有力的支持。

阿里云将继续致力于为客户提供高质量的多活容灾解决方案，助力企业在面对复杂多变的业务环境中保持竞争力和可靠性。我们将持续贡献我们的专业知识和技术力量，确保客户的业务在任何情况下都能平稳运行，实现更高的业务价值。

第五节 /

故障应急的“枕戈待旦”—— 实训演练能力建设

导语

随着用云多年的头部客户在云平台建设上的持续投入，尤其是多中心、一云多 Region、一云多芯等复杂架构的建设，我们认为，对于中大型企业来说，当云的规模达到千台服务器或虚拟机之后，云的安全生产和稳定性的管理成本会显著增加。为了进一步确保云上业务的稳定运行，客户越来越希望通过实践和实战的方式提升云平台的运维和应急能力，使一线运维人员具备故障排查及解决问题的能力。

借用某头部客户的一句话：“前面几年是云的快速建设阶段，每年都是几千万的投入，现在要进入精细化的管理周期了。故障和问题的应急处置能力是最为重要的建设方向，我们希望通过工具和方案快速拿到阿里原厂的积累和经验。”

那么，一个成熟的演练实训平台是非常重要的，它需要能实现故障注入能力的覆盖、故障的编排组合、故障影响半径的控制、配套的观测能力、故障的便捷恢复以及丰富的故障场景库，以此为基础展开故障复现和应对能力提升。

（一）以故障演练和实训推进运维能力提升

1. 实训演练和技术大比武

阿里云专有云一直有内部运维技术大比武的传统，**技术大比武是一种通过模拟真实故障场景和运维挑战，来检验和提升运维人员技术能力的竞赛活动**。通过技术大比武，我们锻炼运维人员在实际问题处理中对专有云的理解，提升故障处置能力和应急能力，拉通大家的经验，看清彼此的差距。

技术大比武从 2020 年开始，至今已举办四届，今年由云平台运维与交付团队组织，重点考察团队同学的云平台运维能力。一方面，通过运维实操，检验同学的技能水平，以赛促学，查漏补缺，提升整体运维人员的技术能力。另一方面，通过同台竞技，让真正优秀的同学和队伍浮出水面，建立榜样，让更多的同学参考学习，形成浓厚的技术氛围和文化。

为了响应更多头部政企客户的要求，**我们将实训演练这种模式**（云平台的故障演练能力和技术大比武的形式）逐步输出到客户侧。我们为行业头部客户实现和积累了大量云产品故障演练场景，以及相应的观测库（故障生效验证），覆盖云平台核心产品的武器库（原子故障）若干个。很多客户明确要求实训演练要立足于飞天云平台，围绕底座和云产品，提升运维侧人员对云的熟悉和应急处置能力。通过实训演练，我们推动客户自有团队和生态团队的专有云运维和应急能力提升。

同时，租户侧故障演练也是不可忽视的重要环节。在实际项目落地过程中，我们与客户一起从应用视角切入，在云平台高可用的基础上叠加应用维度的高可用度量，围绕应用进行架构管理治理、混沌工程实验和预案体系建设。辅之以常态化的上下层一体故障和应急演练，全面提升客户的整体运维水平。

2. 一个完整的故障演练解决方案

租户侧的架构提升与云平台的优化是一个层层递进、密不可分的关系。一个完整的故障演练解决方案是需要整合上下层技术运营体系，需要具备稳定性度量规范，从而实现全链路安全可控的故障演练和运营闭环能力。



图 1 故障演练解决方案

- **组织协同**：红蓝方对抗，往往涉及跨团队协同，需要明确各方职责，步调一致，才能开展演练。
- **演练准备**：需要准备演练环境及分析演练目标（系统 / 流程 / 人员），制定演练方案。
- **演练实施**：一般需要经过预演练及正式演练，且往往会开展多次演练，实施周期较长。
- **演练运营**：演练仅仅只是暴露了风险，但如何有效改进问题，消除风险，显得更为重要，需要建立运营机制，闭环管理演练问题。
- **演练工具**：知道该注入什么故障（演练方）和快速注入 / 恢复故障（工具能力）才能极大降低故障演练的实施成本。

（1）业务场景

- **练人员（技术比武）**：旨在提升运维人员的应急水平，建立良好的技术氛围。通过技术比武，对运维人员的应急水平摸底，同台竞技，以赛促学，提升整体运维人员的故障分析及故障处置能力。
- **练流程（红蓝攻防）**：旨在提升应急流程的协同效率，建立有效的应急机制。通过红蓝攻防，对应急流程的关键环节摸底，识别阻塞点，并持续改进，进而提升优化预案快恢能力。
- **练系统（韧性评估）**：旨在提升核心系统的韧性能力，建立可靠的韧性架构。通过韧性评估，对核心系统的调用链路摸底，识别高可用 / 容灾架构风险，并持续改进，进而提升系统的高可用 / 容灾能力。

（2）产品能力

- **从演练目标的角度看**：产品要提供对演练目标的资源纳管能力和故障注入能力，资源纳管是为了与演练目标建立通信机制，分析演练目标进行故障注入的可能性，故障注入是为了模拟尽可能多的原子故障。

- **从演练流程的角度看**：故障演练需要稳态流程（压测）、故障注入、稳态分析（监控）、故障应急（预案）等环节，为了降低演练的复杂性，产品要提供故障注入的内核能力，同时能够与第三方平台打通，建立一体化的演练流程。

- **从演练经验的角度看**：产品要提供成熟的演练方案库，例如通用开源中间件的演练方案，核心应用的演练方案，同时提供演练方案的管理能力，帮助客户沉淀个性化的演练经验。

- **从架构分层的角度看**：产品要提供云平台 and 云上应用不同维度的故障注入能力，云平台级演练负责对云底座、云产品、云实例等演练目标注入故障；应用级演练负责对云上应用所在的 ECS/ACK 等演练目标注入故障。

- **从产品易用的角度看**：产品要提供灵活的编排能力，安全的防护策略（定时 / 自动停止）等，降低上手门槛。

（3）服务体系

（3.1）练人员（技术比武）

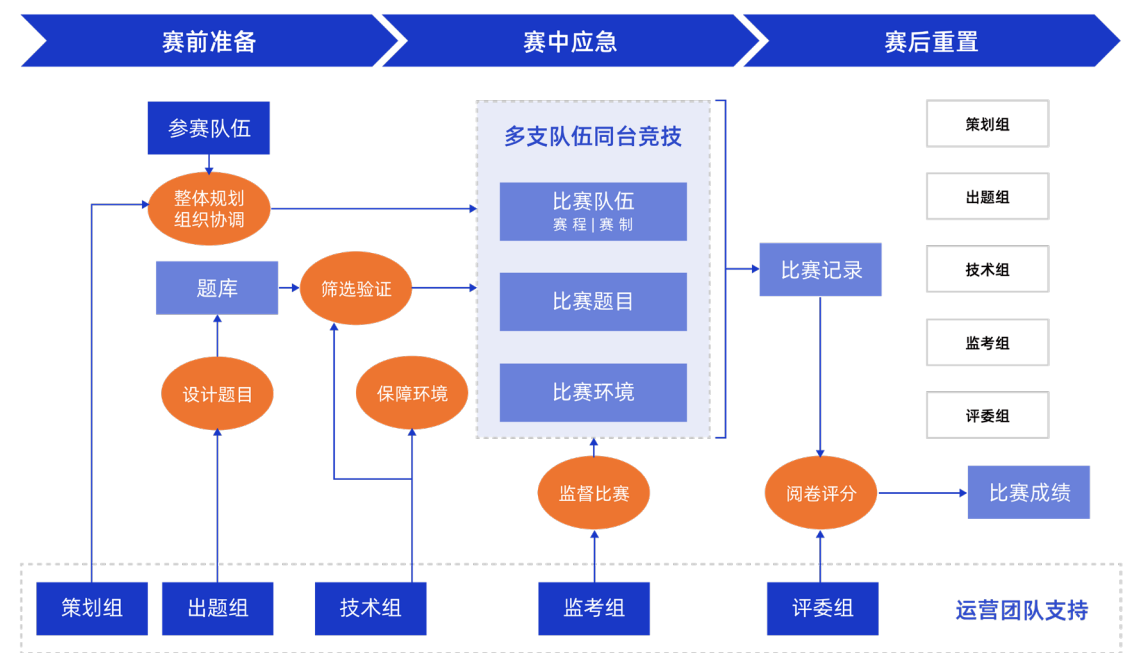


图 2 技术比武练人员流程

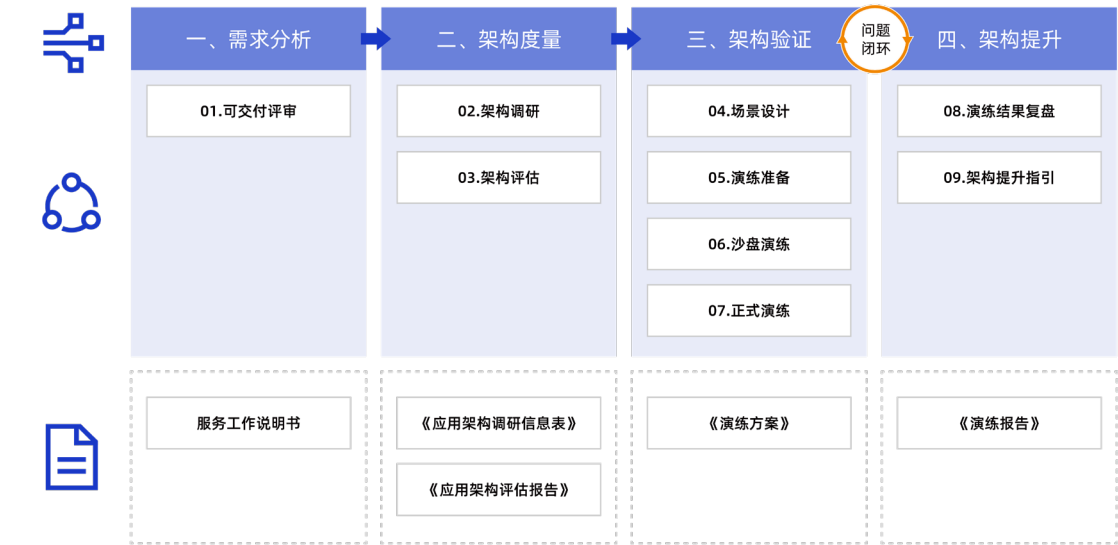
练人员（技术比武）可以类比为考试，包括考试的形式，考试的内容，考试的流程，考试的运营团队。这里重点介绍一下运营团队，尤其是在客户侧落地实践时，建议按照如下架构帮助客户构建虚拟组织。

- **策划组：**定义赛制和比赛规则，组织参赛队伍报名及抽签，宣传运营，安排比赛等。
- **出题组：**根据考察范围及赛题难度等级，设计和编制比赛题目，确保比赛题目的质量、难度适宜。
- **技术组：**
 - 赛前准备，评估比赛题目的实施可能性，通过演练平台组装故障并验证。
 - 赛中应急，及时响应比赛期间因非预期原因导致比赛暂停的技术问题。
 - 赛后清理，清理比赛期间的演练痕迹，保障环境稳定性。
- **监考组：**公布比赛题目，通知参赛队伍开始比赛，处置比赛期间遇到的问题。
- **评委组：**根据评分标准对参赛队伍的表现进行评判。

（3.2）练流程和练系统

练流程和练系统通常围绕稳定性目标开展，经过大量客户案例最佳实践，总结出“4阶9步”标准交付流程和过程量化指标。

• 标准交付流程



- 架构度量：从组织应急、核心链路、应用架构、基础架构等维度调研并评估客户的业务架构，识别架构风险，产出业务架构风险报告。
- 架构验证：根据业务架构风险报告，定义架构验证场景，产出架构验证场景设计方案，同时，确定验证场景，明确验证方式、验证前提、验证准备、验证工具、验证流程，产出架构验证操作手册及执行计划，并按照操作手册及执行计划执行验证。
- 架构提升：根据执行验证过程中记录的操作过程及结果，参与结果复盘及问题优化分析，产出稳定性提升事项，产出业务架构体检报告。

• 过程量化指标

服务交付量化过程，突出客户业务价值						
客户价值 结果指标	故障预防			故障应急(故障1-5-10)		
	架构优化点	监控补齐数	预案固化数	故障平均发现时间	故障平均定位时间	故障平均恢复时间
服务交付 过程指标	01.可交付评审	02.架构调研	04.场景设计	08.演练复盘	数据汇总	
	服务SLA(故障1-5-10)	应用数	演练场景数、场景等级	演练成功率		
	SLA达标率(演练前)	组件数	05.演练准备	问题数		
		第三方依赖数	准备时长	问题分类(监控、预案、架构等)		
		核心链路数	参与团队及人数			
		机器规模数	准备事项(监控数、预案数、稳态流量数)			
		故障等级定义	06.沙盘演练			
		故障应急流程	07.正式演练			
	03.架构评估	06.沙盘演练	09.架构提升			
	风险数	演练时长、问题数、单场景最大执行时间、演练成功率	问题关闭数			
	风险等级	常态化演练(演练时长、问题数、单场景最大执行时间、演练成功率)	预案固化数(演练后)			
		突袭演练(1-5-10的达标率)	监控补齐数(演练后)			
			架构优化点(演练后)			
			SLA达标率(演练后)			
				演练对象		
				演练次数		
				演练场景数		
				演练成功率		
				演练时长		
				场景丰富度		

图 4 服务交付量化指标

突出客户价值结果指标及服务实施过程指标，兑现客户业务价值，提升客户业务稳定性。

- **客户价值结果指标：**故障预防，为了尽可能降低故障发生的概率；故障应急，尽可能降低故障发生的时长，通过故障演练识别这两大指标的基准，并不断改进提升，业务稳定性自然会上升到新的台阶。

• **服务实施过程指标：**建立过程管理机制，识别演练实施过程中遇到的关键问题，快速改进，提高服务交付质量，例如环境准备时间过长，单次演练时间过长等，均能反应演练协同过程中的问题，需要尽快暴露风险，推动改进。

总之，练人员（技术比武）更关注同台竞技下运维人员的能力差异；练流程和练系统更关注故障发生时应急流程及核心系统的韧性能力，这二者关注点不同，对应服务实施的方式有较大差异。

3. 小结

实训演练包括从云平台故障演练到租户侧应用级故障演练，提供全面的故障演练产品能力和服务体系，旨在从云平台到云上应用体系化落地故障演练，助力云平台和云上应用的常态化故障演练和应急机制提升。

能力沉淀为工具产品，才能发挥最大的效力。通过故障演练，我们可以帮助客户识别架构风险，制定架构改进及优化方案，提升客户业务的稳定性。为此，我们推出了实训演练的承载工具平台“玄重尺”。

(二) 实训演练工具平台“玄重尺”能力一览

“工欲善其事，必先利其器”。在专有云运维领域，实训演练是提升运维人员技能和应急能力的重要手段。为了更好地支持这一需求，我们开发了“玄重尺”实训演练工具平台。“玄重尺”不仅提供了丰富的故障演练场景和观测库，还支持多种云产品的原子故障注入，帮助用户在模拟环境中真实体验和解决各类故障。

1. “玄重尺”实训演练平台产品能力介绍

“玄重尺”产品能力包含 2 大中心、6 大功能模块。2 大中心为：演练中心、系统管理。其中，演练中心包括 3 大功能模块：仿真应用、故障注入 & 恢复、故障观测。系统管理包括 3 大功能模块：云数据管理、武器库、观测库。



图 1 “玄重尺”实训演练平台产品功能大图

• 演练中心

演练中心，提供了灵活的故障注入及恢复能力，简易的场景编排管理，实时的演练观测能力，一键演练执行及演练报告生成，帮助客户构建一体化的演练能力。

• 系统管理

系统管理，提供了围绕飞天企业版的元数据管理、覆盖主机 / 容器等大量原子故障的武器库及观测库，帮助客户建立成熟的演练方案库，沉淀个性化的演练经验。

2. 小结

正如亚历山大·蒲柏所说，“工具是人类智慧的延伸”。通过“玄重尺”，客户可以系统地提升运维团队对云平台的熟悉度和应急处置能力，从而在实际生产环境中更加从容应对突发情况。

(三) 云平台实训演练的最佳实践

前序章节概述了阿里云故障演练解决方案以及实训演练工具平台“玄重尺”的产品能力，让读者对故障演练能够有一个整体认识。接下来会通过阿里云内部和外部客户的实践来分享下两个最佳实践。

1. 前置工作

(1) 产品能力

利用“玄重尺”提供云底座、云产品、云实例等维度故障注入能力，实现故障演练便捷化、复杂故障场景可编排、故障演练可观测。

- **系统管理**：定义武器库和观测库，武器库提供原子故障注入能力，观测库提供故障生效验证能力，同时提供自定义脚本，用户根据业务需要自行定义故障，云数据提供了 OnTianji/OnPaaS 的 CMDB 数据及云实例数据查询能力，支持多 Region 架构，提高故障编排效率。
- **演练目标**：提供云底座、云产品、云实例等演练目标编排能力。
- **演练中心**：仿真应用用于构建快捷的稳态流量，例如 MQ 消息收发，OSS 文件上传下载，RDS 数据读写等，提高业务验证效率，故障注入 & 观测提供场景编排和故障执行等能力，故障观测提供了观测大屏，用于实时观察故障情况。

(2) 演练方案

覆盖云底座、云产品、云实例等 3 种维度的故障场景，演练方案是经过反复实践的经验，为客户提供成熟的演练方案，有助于演练的快速落地。

- **方案丰富度**：覆盖计算、存储、网络等多款核心云产品，随着实践案例不断增多，演练方案库会不断丰富。
- **方案复杂度**：定义方案难度等级为一到五级，同时根据影响范围，定义并行演练及串行演练方案。

2.【内部】技术大比武

(1) 背景介绍

技术大比武在专有云团队内部已举行多届，重点考察运维团队同学的云平台运维能力。一方面，通过运维实操，检验同学的技能水平，以赛促学，查漏补缺，提升整体运维人员的技术能力。另一方面，通过同台竞技，让真正优秀的同学和队伍浮出水面，建立榜样，让更多的同学参考学习，形成浓厚的技术氛围和文化。

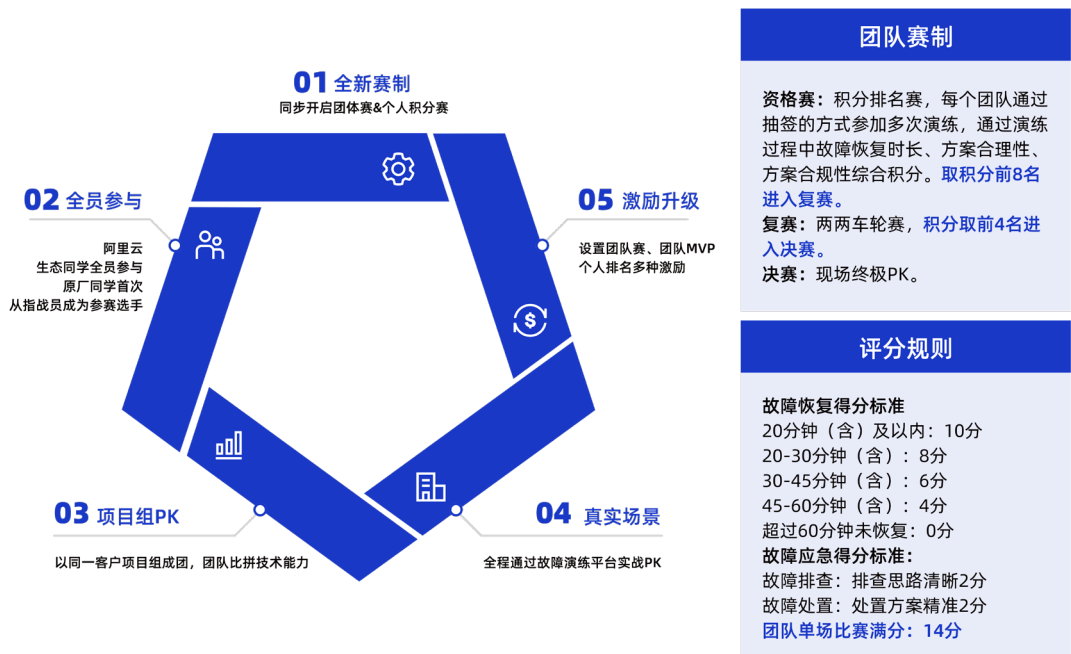


图 1 内部技术大比武最佳实践示意图

技术大比武主要有 3 个特点：

- **项目组 PK**：以同一客户项目组成团，团队间比拼技术能力。从以往的强强联和成团到同一客户项目组成团，更加注重考察不同项目组之间的技术水平。
- **全员参与**：全员做为参赛选手，注重考察团队分工协作能力。
- **真实场景**：全程通过“玄重尺”故障演练平台实战 PK。从以往重理论轻实战转变为全程实战，更加注重考察故障应急处置能力。

技术大比武涉及几个关键角色：比赛队伍，比赛题目，比赛环境。由于是故障演练实操运维，对比赛题目和比赛环境要求很高，根据《人月神话》里讲述的项目推进的不可能三角理论（时间、成本、质量），尽力寻找局部最优解。

- **赛题隔离性**：一套专有云环境，支持多支队伍同台竞技，要确保比赛题目（故障注入）互不干扰。
- **赛题多样性**：由于全程实战，需要注入真实的故障观察业务表现。
- **赛题复杂性**：赛制分为资格赛 / 复赛 / 决赛，逐步淘汰晋级，比赛题目要有区分度。同时，由于同台竞技无法使用相同的题目，为了保证比赛的公平性，确保每一轮比赛使用的题目难度尽可能一致，因此，需要对比赛题目定义难度等级。

(2) 实践方法

- 赛题隔离性

云平台的故障场景根据影响范围分为三类：云底座、云产品、云实例。因此，为了满足隔离性的要求，比赛题目准备时只能选择云产品或云实例这类故障场景，且比赛期间，会选择不同类型产品或实例，最大限度地降低彼此干扰的概率。

- 云底座：云底座故障会影响所有云产品或云实例，例如 ASCM 故障，导致所有云产品控制台无法访问。
- 云产品：云产品自身故障，例如 ECS 某个组件故障，影响 ECS 实例创建，但不会影响 SLB/RDS 等云产品。
- 云实例：云实例自身故障，例如 RDS 实例故障，导致 RDS 实例无法访问，但不会影响其它 RDS 实例。

- 赛题多样性

赛题尽可能源于真实的故障场景，成立出题组负责将一线沉淀多年的故障案例共享出来，技术组负责筛选验证，同时结合云产品架构变换故障类型和故障组件，形成最终的比赛题目。

- 赛题复杂性

比赛题目难度的设置需要恰当平衡，既不能过于简单，让参赛队伍无法有效区分技能水平，也不能过于困难，导致比赛队伍普遍感到挫败，失去比赛信心，同时，还要兼顾区分度，让高水平的比赛队伍取得高分，让低水平的比赛队伍取得低分。

(3) 实践总结

• 以赛促学

每一轮比赛结束后进行直播分享，邀请优秀的比赛队伍分享心得体会，包括比赛过程回顾、团队准备动作、日常技能提升等，让其它比赛队伍能够不断总结经验，提升自己，做到在赛中学，在学中赛。

• 学以致用

如何使用技术能力替客户解决问题，发挥业务价值，是我们一直以来努力的方向。技术大比武作为运维人员的风采展示舞台，不仅是秀个人技能的舞台，更是展示服务客户的细节。因此，在决赛环节，参赛队伍需展示服务于客户的业务成果，用于体现在服务客户的过程中，真正帮客户解决了什么问题，客户是否对服务满意。

• 能力沉淀

云平台级演练解决方案正是在内部技术大比武的支持过程中，点滴积累，不断孵化而成。通过内部实践，演练平台、演练方案、最佳实践均已成熟，为外部输出做了扎实的铺垫，尤其是演练平台的打磨，比赛题目（演练方案库）的丰富，比赛题目难度等级的定义，实践经验的积累。

3.【外部】某客户的实训演练实践

(1) 背景介绍

某客户业务覆盖多地数据中心，建设规模大，云上业务繁多。客户也从规模化建设阶段逐步走向精细化运维阶段，且在技术安全可控的背景下，如何提升自主化运维能力，保障云上业务稳定运行，成了迫切需要解决的问题。



图 2 客户“实训演练”技术大比武实践图示

客户面临的主要痛点有：

- 如何开展全场景的云平台级故障演练？
- 如何通过实操演练，检验一线运维人员故障排查及故障处置能力？
- 如何以演练促提升，深入了解飞天企业版架构，提高自主化运维能力？
- 如何沉淀一套体系化的产品工具及解决方案，进行常态化演练及平台保鲜？

（2）实践方案

客户开展实训演练的目标和阿里云内部技术大比武的初衷是高度契合的，双方反复对焦实训演练需求后，客户决定引入阿里云“玄重尺”实训云演练平台及技术大比武模式。

通过引入“玄重尺”实训演练平台，客户实现了演练能力便捷化、多维度故障场景覆盖、故障场景延伸控制、复杂故障场景构建。通过引入阿里云技术大比武形式，客户开展了公司内部的“实操演练”技术大比武，以赛促学，大幅度提升了内部人员运维自主化能力。

4. 结束语

作为客户身边的云计算专家，我们肩负着让客户上好云、用好云的责任。通过故障演练，我们帮助客户识别架构风险，提出改进优化意见，提升业务稳定性。客户的信任和支持是我们最大的动力，只有当客户放心将自己的业务托付给阿里云，我们的使命才算真正达成。

这些工作并非一蹴而就，而是日积月累的过程，正如古诗所言：“随风潜入夜，润物细无声。”我们将以专业的态度和不懈的努力，持续为客户提供卓越的云服务，确保每一个客户的业务都能在云上稳健运行，实现业务的长期发展和成功。

第六节 /
云上安全体系建设

1. 安全目标与合规要求

（1）行业背景分析

随着云计算的广泛应用，越来越多的企业开始将核心业务系统迁移至云端。但与此同时，网络安全威胁也日益加剧，企业对数据安全、访问控制、日志审计等能力的要求不断提升。

在政企、金融、能源、交通、医疗等行业中，安全体系建设不仅关系到系统的稳定运行，更直接影响是否符合国家监管要求和行业标准。

（2）安全建设目标

Gartner 数据显示，相比传统数据中心，云上的安全能力可帮助企业减少至少 60% 的安全事件。传统的线下安全体系存在多个痛点，常令用户头疼：

- **兼容性差**：安全设备与业务系统或网络环境可能存在适配问题；
- **接口不统一**：不同厂商设备网卡和策略配置差异大，集成困难；
- **扩展性受限**：面对流量突增，只能通过更换硬件实现扩容，缺乏弹性。

云上安全建设目标是构建覆盖数据、应用、身份、网络、容器等多维度的安全防护体系，实现全流程可追溯、全访问受控、全链路加密，保障业务系统在阿里云上的稳定运行与数据资产安全。

目标	描述
数据安全	确保数据在传输、存储过程中的机密性与完整性
应用安全	防止 OWASP Top 10 攻击，保障 API 接口调用安全
身份认证	实现统一身份管理、权限控制与多因素认证机制
日志审计	所有操作行为可追溯，便于事后回溯与责任定位
网络隔离	使用 VPC、安全组、ACL 等手段实现逻辑隔离
容器安全	对 ACK/K8s 容器服务进行安全加固与镜像扫描

表 1 云上安全建设目标

（3）合规性要求

云上安全建设需满足等级保护 2.0、GDPR、金融与政务行业规范及 ISO 国际认证等多维度合规要求，覆盖网络、数据、应用全链路防护。

标准	描述
等级保护 2.0（三级）	国内重要信息系统必须满足的合规框架，涵盖网络、主机、应用、数据四个层面
GDPR（欧盟通用数据保护条例）	若涉及个人信息出境或欧洲用户，需满足数据加密、脱敏、最小化收集等要求
金融行业规范（银保监会）	涉及交易、支付、风控等业务时，需满足金融级安全防护要求
政务云安全规范	公共服务类系统需遵循政务云平台安全接入标准
ISO 27001/ISO 27017	国际信息安全管理體系认证，适用于跨国企业

表 2 多维度合规性要求标准

2. 云上安全架构设计总览

（1）专有云支撑企业安全能力蓝图



图 1 云上安全能力蓝图

网络安全：

- 访问控制：互联网边界、VPC 边界、主机边界
- 入侵检测和访问 IPS：南北向、东西向
- Web 流量防护

身份与访问控制：

- 身份安全：访问控制、云 SSO
- 权限边界：授权管理、管控策略

主机与容器安全：

- 主动防御：防勒索、防病毒、防篡改
- 安全防范：合规检查、基线检查、漏洞检测与修复

应用与 API 安全：

- 身份认证与鉴权机制
- HTTPS 加密通信
- 应用防火墙与 API 安全防护

数据安全：

- 数据静态加密：文件加密、密钥管理
- 数据动态加密：证书、VPN
- 数据行为审计：数据安全中心
- 敏感数据发现与脱敏

日志审计与威胁检测：

- 操作审计配置
- SLS 日志采集与分析
- 云安全中心实时监控
- 态势感知与联动响应

(2) 安全职责划分

安全维度	阿里云职责	客户职责
网络隔离	提供 VPC、NACL、安全组等能力	配置规则、限制端口
主机安全	提供基础操作系统镜像	自行加固、安装补丁
应用安全	提供 WAF、反入侵、NDR 等应用层防护产品	需云租户运营方妥善配置、安全运营
数据安全	提供 TDE、OSS 加密等能力	需启用加密、脱敏、权限控制
日志审计	提供 SLS、ActionTrail 工具	需配置采集规则、设置告警
容灾备份	提供 OSS 跨区域复制、数据库 Binlog	需制定备份策略、恢复流程

表 3 云上安全职责划分

3. 网络安全

(1) 网络架构与访问控制

(1.1) VPC 网络隔离策略

- 使用 VPC 对业务子网进行划分，按照业务模块实现网络分区管理；
- 不同子网之间通过 NACL（网络访问控制列表）实现逻辑隔离，限制不必要的跨子网通信；
- 支持 VPC Peering 或 CEN（云企业网）实现跨 VPC 的互通，保障多区域 / 多账户间的网络连通性；
- 所有公网入口统一经过 WAF（Web 应用防火墙）+ SLB（负载均衡），确保流量先经安全过滤再进入后端服务。

(1.2) 安全组配置规范

- 遵循“最小权限原则”，仅开放必要端口（如 80、443）；
- 禁止 SSH/RDP 直接暴露公网，可通过跳板机或堡垒机进行访问；

- 所有数据库、Redis、MongoDB 等敏感服务 禁止公网访问，仅允许内网访问；
- 对外服务的安全组应限制源 IP 范围，结合白名单机制提升安全性。

(1.3) 访问控制增强

- 所有对外服务启用 HTTPS 加密通信，防止数据明文传输；
- 启用 IP 白名单机制，仅允许授权的客户端 IP 进行访问；
- 利用 云防火墙 统一管理多个 VPC 的出入站规则，集中管控访问策略；
- 可视化展示威胁路径，支持入侵防御策略（IPS）配置。

(2) Web 流量与接口防护

(2.1) Web 应用防护（WAF）

- 所有 Web 接口均通过 WAF 进行攻击拦截，包括 SQL 注入、XSS、CSRF、CC 攻击等；
- 结合 API 网关鉴权机制，防止非法 API 调用；
- 对高频访问接口设置 限流、熔断、黑白名单 等防护策略，避免被恶意刷接口。

(2.2) DDoS 防护组合

- 开启 DDoS 防护基础包，抵御常见流量型攻击；
- 对高风险业务使用 高防 IP 提供更强的清洗能力；
- 自动封堵异常请求源 IP，及时阻断攻击源；
- 所有对外暴露的服务必须经过 WAF + SLB + DDoS 防护，形成多重防线。

(3) 入侵检测与防御体系

(3.1) 入侵防御系统（IPS）

- 部署 IPS 对南北向（外部到内部）和东西向（内部服务间）流量进行实时监测；
- 检测并阻断可疑行为，如扫描、暴力破解、漏洞利用等。

(3.2) 云安全中心

- 启用 云安全中心 实时监控主机层面的异常行为，如：
 - 异常登录尝试；
 - 用户提权操作；
 - 恶意进程启动；
- 提供安全事件告警、日志审计及修复建议。

(3.3) 态势感知平台

- 结合 态势感知平台 实现整体威胁可视化展示；
- 综合分析网络、主机、应用层的安全状态，辅助快速响应与决策。

(3.4) 实施建议与最佳实践

- 所有对外暴露的服务必须经过 WAF 和 SLB，不得直接受理公网访问；
- 安全组配置遵循 最小权限原则，严格限制开放端口和访问来源；
- 对数据库、缓存、中间件等敏感资源启用 白名单访问控制；
- 定期更新 WAF 规则库与 IPS 策略，应对新型攻击手段；
- 建议部署 零信任访问控制模型，逐步替代传统边界防护思维；
- 定期进行安全巡检与渗透测试，验证防护策略有效性。

4. 身份与访问控制（IAM）

（1）设计原则

- **最小权限原则（Least Privilege）**：为用户、系统或服务分配完成其职责所需的最小权限集合，避免过度授权。
- **职责分离（Separation of Duties）**：对高敏感操作进行权限拆分，防止单一账号拥有过多关键权限，降低内部风险。
- **集中统一管理**：通过统一的身份认证平台（IAM）集中管理所有用户、角色和权限策略，提升可审计性和可控性。
- **多因素认证（MFA）**：对管理员、特权账号强制启用 MFA，增强账户安全性。
- **生命周期管理**：支持用户账号的创建、变更、禁用、删除等全生命周期管理，确保权限随业务变化动态调整。

（2）核心组件与功能

（2.1）统一身份认证（IAM）

- 支持主账号与子账号体系，实现精细化权限划分；
- 提供基于角色的访问控制（RBAC），便于权限复用与管理；
- 支持跨组织的联合身份（Federated Identity），对接企业 AD/LDAP 或 SSO 系统；
- 支持临时凭证机制（STS），用于安全地授予短期访问权限（如运维脚本、自动化任务）。

（2.2）细粒度权限控制

- 基于策略（Policy）的权限管理，支持按资源类型、操作行为、地域、标签等维度进行权限定义；
- 可自定义系统策略或继承默认策略模板，满足不同业务场景需求；
- 支持对 API 操作级别的权限控制，限制特定接口调用权限；
- 支持服务关联角色（Service-linked Role），用于授权云服务访问其他资源。

（2.3）单点登录（SSO）集成

- 支持与企业现有 AD/LDAP、SAML 2.0、OAuth 2.0 等协议对接，实现统一身份认证；
- 用户一次登录即可访问多个云资源和服务，提升用户体验与管理效率；
- 支持跨云厂商、混合云环境下的统一身份打通。

（2.4）密钥与凭证管理

- 对 AccessKey、Token、证书等敏感凭证进行加密存储与自动轮换；
- 支持定期审计密钥使用情况，及时清理长期未使用的凭证；
- 强制要求敏感操作二次确认或短信 / 邮件验证码验证。

（2.5）访问日志与审计

- 所有 IAM 操作均记录在审计日志中（如 CloudTrail、ActionTrail 等），便于追溯；
- 支持日志分析与异常行为检测，发现越权访问、非法提权等行为；
- 可结合态势感知平台进行可视化展示与告警联动。

（3）实施建议与最佳实践

- 禁止主账号直接操作生产资源，所有日常操作应使用子账号或角色；
- 高权限账号需开启 MFA 并绑定手机 / 邮箱，提高账户安全性；
- 定期审查 IAM 权限配置，清理过期账号与冗余策略；
- 敏感服务（如数据库、KMS 密钥管理服务）必须启用白名单 + IAM 授权双重控制；
- 为自动化任务配置专用角色，避免使用长期有效的 AccessKey；
- 启用 IAM 访问分析工具，识别潜在越权风险；
- 建立权限审批流程，对高危操作（如删除资源、修改权限）设置审批机制；
- 结合零信任架构（Zero Trust），逐步将身份作为访问控制的核心依据。

5. 主机与容器安全

(1) ECS 主机安全加固

- 关闭 root 登录，启用 sudo：禁止直接使用 root 账户登录系统，提升权限管理安全性；
- 启用 SSH 密钥认证，禁用密码登录：防止暴力破解攻击，增强远程访问控制；
- 安装并开启阿里云安全中心 Agent(云安全中心)：实时监控主机安全状态,检测异常行为(如提权、恶意进程)；
- 设置自动补丁更新机制（运维编排服务 OOS）：通过 OOS 定期检查并自动安装操作系统及关键软件的安全补丁；
- 日志采集至 SLS（日志服务）进行集中分析：将系统日志、登录日志、安全事件日志统一采集至 SLS，便于审计与威胁分析；
- 配置主机防火墙策略（通过云防火墙或安全组）：限制不必要的端口暴露，控制入站 / 出站流量；
- 启用完整性监控与文件变更告警（云安全中心）：对关键系统文件、配置文件进行监控，发现异常修改立即告警；
- 最小化安装系统组件和服务：减少攻击面，仅部署必要应用与依赖库；
- 启用 Ali-Tomoyo 或 SELinux 强制访问控制机制：强化系统级安全防护能力（Ali-Tomoyo 是阿里自研轻量级强制访问控制模块）。

(2) ACK 容器服务安全策略

(2.1) Pod Security Policy（PSP）替代方案（推荐使用 PodSecurity Admission 配置）

安全项	说明
禁止以 root 权限运行容器	设置 runAsNonRoot 限制容器以非 root 用户启动
限制特权模式（Privileged）	禁止容器以特权模式运行，防止内核资源滥用
控制容器能力（Capabilities）	仅允许必要的 Linux Capabilities，如 NET_BIND_SERVICE
禁止挂载敏感 HostPath	防止容器访问宿主机文件系统

表 4 Pod 安全策略配置

在阿里 ACK 中建议使用内置的 PodSecurityAdmission 功能替代 PSP，通过命名空间标签配置安全级别（privileged/restricted/base），实现容器安全基线控制。

(2.2) Network Policy

- 实施 Kubernetes NetworkPolicy，结合阿里云 VPC 和 CNI 插件限制 Pod 之间、命名空间之间的通信；
- 默认拒绝所有东西向流量，按需开放访问路径，防止横向移动攻击；
- 可结合阿里云网络可视化功能，实现容器网络拓扑与访问关系的透明化展示。

(2.3) Kubernetes RBAC

- 启用基于角色的访问控制（RBAC），确保用户和服务账户权限最小化；
- 禁止直接使用 cluster-admin 角色，避免越权操作；
- 按业务模块划分命名空间，并为每个命名空间分配独立的角色权限；
- 使用 RoleBinding 和 ClusterRoleBinding 明确授权范围，避免权限泛滥；
- 结合 RAM + OIDC 联合身份认证实现细粒度账号体系集成。

（2.4）镜像安全扫描

- 所有容器镜像在部署前必须经过安全扫描；
- 使用 阿里云 ACR（容器镜像服务）企业版 提供的镜像漏洞扫描功能，识别 CVE 漏洞和合规问题；
- 支持设置镜像准入策略（Image Admission Policy），阻止高危镜像部署；
- 开启镜像签名机制（Trusted Registry），确保镜像来源可信且未被篡改。

（2.5）Ingress 访问控制

- 对接 阿里云 Web 应用防火墙（WAF）实现 Web 层访问控制，防御常见 Web 攻击（SQL 注入、XSS 等）；
- 使用 阿里云服务网格 ASM（Istio 托管版）+ Envoy 实现微服务级别的身份鉴权、限流熔断、访问日志记录；
- Ingress 控制器配置 TLS 终端加密，保障数据传输安全；
- 设置 IP 白名单或 JWT/OAuth2 鉴权机制，防止非法访问。

（2.6）容器运行时安全

- 部署 阿里云安全中心 – 容器安全模块，实时监控容器运行行为；
- 识别异常调用、提权、敏感目录访问等风险操作；
- 配合 Kubernetes 的 Event 监控，实现安全事件快速响应；
- 支持容器逃逸检测、异常进程监控、容器间入侵行为识别等功能。

（3）最佳实践建议

- 所有容器服务部署前必须经过安全扫描，包括镜像扫描与配置合规性检查；
- 生产环境禁止使用 latest 标签的镜像，确保版本可控、可追溯；
- 所有容器均需设置资源限制（CPU / Memory），防止资源耗尽型 DoS 攻击；

- 启用 PodSecurityAdmission，利用命名空间标签设置安全等级，替代传统 PSP；
- 使用 Helm Chart 管理部署模板，纳入安全策略约束；
- 容器编排配置文件（YAML）应纳入代码审查流程，确保安全配置不遗漏；
- 为关键业务容器启用 Seccomp、AppArmor 等内核级安全机制（ACK 支持）；
- 定期审计 Kubernetes RBAC 权限配置，清理冗余角色与绑定关系；
- 建立容器生命周期安全管理流程，涵盖构建、测试、部署、运行、退役全过程；
- 使用阿里云 KMS 服务对敏感信息加密存储，如 Secret、ConfigMap 中的凭证数据。

6. 应用与 API 安全

（1）身份认证与鉴权机制（OAuth2.0 / JWT）

- 推荐使用 阿里云 RAM + OIDC 联合身份 构建统一的身份认证中心，支持多租户架构，实现用户、角色、权限的集中管理；
- 支持标准的 OAuth2.0 和 OpenID Connect（OIDC）协议，便于与第三方系统集成；
- 可对接企业 AD/LDAP，实现单点登录（SSO）能力；
- 提供临时凭证获取接口（STS），用于服务间调用或自动化任务的安全访问；
- 所有 API 调用必须携带 Token，否则拒绝访问；
- 使用 JWT（JSON Web Token）承载用户身份和权限信息，Token 需签名加密，防止篡改；
- 设置合理的 Token 生命周期，并支持刷新机制；
- 结合 RAM 实现细粒度访问控制，基于策略（Policy）对用户或角色进行资源级别的权限划分；
- 建议启用 Token 黑名单机制，对异常或注销的 Token 进行主动封禁，可通过 Redis 缓存实现黑名单快速查询。

（2）HTTPS 加密通信

- 所有对外 Web 服务必须启用 HTTPS 加密通信，禁止明文 HTTP 协议暴露；
- 使用 阿里云 SSL 证书服务 上传并绑定域名证书，确保主流浏览器兼容性；
- TLS 版本建议不低于 TLS 1.2，逐步淘汰 TLS 1.0/1.1，提升传输层安全性；
- 建议启用 HSTS（HTTP Strict Transport Security）策略头，强制客户端使用 HTTPS 访问；
- 支持 OCSP Stapling 功能，提升证书吊销状态验证性能，降低握手延迟；
- 配置证书自动续签机制，通过阿里云证书托管服务结合自动部署插件，实现证书生命周期自动化管理；
- 在微服务之间或敏感接口中可选启用双向 TLS（mTLS）认证，增强通信双方身份可信性。

（3）应用防火墙与 API 安全防护

- 在 API 网关中启用请求签名机制，防止伪造请求与重放攻击；
- 设置熔断机制与限流策略，防止高频访问导致系统崩溃，支持按用户、IP、Token、接口等维度配置限流规则，并支持突发流量弹性放行；
- 所有 API 请求需记录日志并转发至 SLS（日志服务），便于审计与行为分析；
- 建议对敏感接口实施 IP 白名单 + Token 的双重验证机制，提升访问控制强度；
- 启用 Web 应用防火墙（WAF）对 API 接口进行攻击拦截，防御 SQL 注入、XSS、CSRF、命令注入等常见攻击；
- 支持自定义规则匹配 API 路径与参数，实现精细化防护；
- 推荐使用 阿里云 API 网关 提供 API 发布、限流、鉴权、签名验证、日志审计等功能，适合对外暴露服务场景；
- 在容器化微服务架构中，推荐使用 Kubernetes Ingress + 阿里云服务网格 ASM（Alibaba Service Mesh），支持 Istio 托管版 + Envoy 实现高级路由、鉴权、可观测；

（4）最佳实践建议

- 所有对外 API 必须经过认证授权访问，禁止匿名访问；
- 生产环境中禁止使用测试 Token 或调试模式下的开放接口；
- 定期更新 OAuth2.0 客户端密钥与 Token 签名密钥，防止长期密钥泄露；
- 为不同业务线划分独立的身份认证域与权限模型，避免权限交叉污染；
- 对敏感数据接口启用动态脱敏策略，根据用户角色返回差异化数据；
- 建立 API 安全事件响应机制，对接态势感知平台实现异常访问告警与阻断联动；
- 结合 阿里云 KMS 服务 对敏感 Token 数据进行加密存储与解密访问；

7. 数据安全

（1）数据库存储加密（TDE）

- 支持在 PolarDB、RDS 等主流数据库服务中启用 透明数据加密（TDE）功能；
- 使用 阿里云 KMS（密钥管理服务）提供的主密钥对数据库文件进行加密，确保静态数据安全；
- TDE 实现透明解密机制，应用层无需修改代码即可访问加密数据，兼容性好；
- 推荐在以下场景启用 TDE：
 - 金融、政务等对数据安全性要求较高的业务系统；
 - 存储用户隐私信息（如身份证号、银行卡号）的数据库；
- 建议定期轮换 KMS 密钥，提升密钥生命周期安全性；
- 对于涉及敏感字段的数据表，建议结合字段级加密或脱敏策略，形成多层防护。

（2）OSS / NAS 文件存储加密

- 在 OSS 对象存储服务 中启用 服务端加密（SSE-KMS），使用 KMS 主密钥对上传对象进行自动加密；

- NAS 文件系统（如阿里云 NAS）支持自动加密功能，在创建文件系统时可选择加密配置；
- 所有写入存储的数据均以加密形式保存，读取时由系统自动解密，保障数据在传输之外的静态安全；
- 建议为不同业务系统分配独立的 KMS 密钥，实现密钥隔离；
- 定期轮换 KMS 密钥，防止因长期使用导致的密钥泄露风险；
- 配合 RAM 权限控制，限制非授权用户访问加密存储资源。

（3）敏感字段脱敏处理

- 在数据库查询过程中启用 动态脱敏机制，例如通过 SQL 函数或数据库内置功能（如 PostgreSQL 的 pgcrypto、MySQL 的表达式处理）对敏感字段进行模糊化输出；
- 对身份证号、手机号、银行卡号等关键字段设置脱敏规则，确保仅授权用户可见完整信息；
- 在应用层或中间件中集成脱敏逻辑，避免将明文敏感信息暴露给前端或日志系统；
- 使用 ACM 或 Nacos 配置中心 加密存储敏感配置项（如数据库连接串、API Key）；
- 在日志采集和输出环节，启用关键字过滤机制，防止明文敏感信息落盘；
- 推荐常用脱敏规则示例：
 - 手机号：显示为 138****1234；
 - 身份证号：显示为 110101*****00xxxx；
 - 邮箱地址：显示为 u**r@domain.com；
- 结合 SLS 日志服务的脱敏插件，对日志内容中的敏感字段进行自动识别与掩码处理。

（4）最佳实践建议

- 所有涉及用户隐私数据的数据库必须启用 TDE 加密，确保静态数据安全；
- 不同业务模块应使用独立的 KMS 密钥，避免密钥共享带来的安全隐患；
- 对于高安全等级系统，建议结合字段级加密与脱敏策略，实现细粒度保护；

- 在数据备份和恢复流程中也应保留加密状态，防止备份文件泄露；
- 所有敏感配置信息应通过 ACM/Nacos 等配置中心加密管理，避免硬编码在代码或配置文件中；
- 建议在开发、测试环境中同步启用脱敏机制，防止真实数据误用于非生产环境；
- 定期审计密钥使用情况与脱敏策略执行效果，确保安全机制持续有效。

8. 日志审计与威胁检测

（1）操作审计配置（ActionTrail）

- 启用 ActionTrail 操作审计服务，记录所有关键操作事件，包括资源创建、删除、修改等行为；
- 支持自动归档审计日志至 OSS 存储服务，满足长期留存和合规审查需求；
- 审计日志可实时转发至 SLS 日志服务，便于集中分析与告警配置；
- 支持跨账号审计能力，适用于多账户管理场景下的统一日志追踪；
- 可用于安全事件回溯、内部审计、合规检查等场景，确保操作可追溯、责任可定位；
- 建议对高权限操作（如删除数据库、修改安全组）设置专项监控与告警策略。

（2）SLS 日志采集与分析

- 使用 SLS（日志服务）集中采集各类系统与应用日志，包括：
 - 应用服务器日志；
 - ECS 实例操作系统日志；
 - ACK 容器服务访问日志与容器日志；
- 设置关键字匹配规则，例如 error、login failed、Unauthorized 等，触发实时告警；
- 支持多维度日志关联分析，识别潜在攻击路径或异常行为模式；

- 推荐以下日志采集方式：
 - Logtail 插件：适用于 ACK 容器环境，支持标准容器日志采集；
 - Serilog / NLog SDK：适用于 .NET 等应用，实现结构化日志上报至 SLS；
 - Fluentd / Fluent Bit：适用于边缘计算或混合云场景，具备良好的插件生态支持；
- 可结合仪表盘功能构建统一的日志视图，提升运维与安全团队的响应效率。

（3）云安全中心实时监控

- 启用 阿里云安全中心 实时监控主机与容器的安全状态，涵盖以下方面：
 - 系统漏洞扫描与修复建议；
 - 主机病毒检测与清除；
 - 安全基线配置检查（如 SSH 配置、密码复杂度等）；
- 对异常登录行为（如非常规时间、IP 登录）、提权操作（如 sudo 切换 root）进行实时告警；
- 支持通过钉钉等方式推送安全告警信息，确保第一时间通知运营人员；
- 定期生成并输出 安全态势报告，涵盖资产风险、攻击趋势、处置建议等内容；
- 可与 IAM、WAF、IPS 等其他安全组件联动，形成完整的威胁闭环处理机制。

（4）态势感知与联动响应

- 部署 阿里云态势感知平台，实现全局安全风险可视化展示；
- 平台可自动识别高危事件（如 DDoS 攻击、WebShell 连接、异常 API 调用），并推送给安全运营人员；
- 提供 可视化安全风险图谱，展示攻击路径、受影响资产范围及处置进度；
- 结合威胁情报数据，增强对外部攻击源的识别与防御能力；
- 可作为 SOC（安全运营中心）的核心支撑平台，协助完成事件研判、处置与复盘。

（5）最佳实践建议

- 所有关键操作必须启用 ActionTrail 审计，保留完整操作轨迹；
- 定期审查 SLS 中的日志内容与告警规则，优化误报率与覆盖率；
- 将日志采集纳入 DevOps 流程，确保新上线业务自动接入日志服务；
- 对敏感操作（如管理员登录、系统配置变更）设置专项审计与告警；
- 安全中心与态势感知平台应每日巡检，及时响应高风险事件；
- 建议将日志与审计数据保存周期与行业监管要求保持一致，防止合规风险；
- 定期演练安全事件响应流程，验证从检测、告警到处置的完整性与时效性。

9. 灾备与恢复机制

灾备与恢复机制在前文已经进行了专题介绍，这里不多做赘述。

10. 安全运营建议

（1）安全巡检机制

- 建立定期安全巡检制度，确保系统安全状态持续可控；
- 每月执行一次安全基线检查，涵盖操作系统、数据库、容器、中间件等关键资产，验证是否符合最小权限、补丁更新、登录控制等安全要求；
- 每季度开展一次渗透测试，模拟外部攻击路径，识别潜在漏洞和配置风险；
- 每半年进行一次灾备演练，覆盖数据恢复、服务切换、容灾中心接管等场景，确保业务连续性；
- 所有巡检结果需形成报告并归档，作为后续整改与复盘依据；
- 可借助 阿里云 Security Center 的自动化巡检功能 实现一键式安全体检，提升效率与覆盖率。

(2) 安全事件响应流程

- 建立标准化的安全事件响应流程，包含以下五个阶段：
 - a. **发现**：通过日志告警、入侵检测、WAF 拦截等方式识别异常行为；
 - b. **分析**：利用 SLS 日志服务 + ARMS 应用监控 快速定位攻击源、受影响资产及影响范围；
 - c. **隔离**：对受感染主机或容器进行网络隔离，必要时冻结账户或阻断 IP；
 - d. **恢复**：清除后门、修复漏洞、恢复备份数据，确保系统恢复正常运行；
 - e. **报告**：总结事件原因、处置过程与改进建议，形成完整事件报告；
- 成立专门的应急响应小组（IRT），明确职责分工与响应流程；
- 所有安全事件需记录在案并归档，便于后续审计与追溯；
- 支持与 态势感知平台 和 SOAR 自动化响应平台 联动，实现部分事件自动处置。

(3) 安全培训与意识提升

- 建立常态化安全培训机制，提升全员安全意识与技术能力；
- 每年至少组织一次全员安全意识培训，内容涵盖密码管理、钓鱼防范、数据保护等基础安全知识；
- 对开发与运维人员开展专项培训，包括：
 - 安全编码规范（如 OWASP Top 10 防御）；
 - 权限管理最佳实践；
 - DevSecOps 流程集成；
- 推荐使用 阿里云 Security Center 的安全培训模块 实现线上学习、打卡、考核一体化管理；
- 培训记录应可查询、可导出，便于内部审计与监管检查；
- 鼓励建立“红蓝对抗”机制，通过攻防演练提升实战能力。

(4) 等保 2.0 合规建议

- 根据《网络安全等级保护基本要求（GB/T 22239-2019）》三级标准，构建满足合规的技术防护体系；
- 在以下关键领域落实安全控制措施：
 - 访问控制：基于 RAM + IAM 实现细粒度权限划分；
 - 日志审计：通过 ActionTrail + SLS 实现操作日志与安全事件集中采集；
 - 数据加密：启用 TDE、OSS 加密、KMS 密钥管理等手段保障静态数据安全；
 - 入侵检测与防御：部署 WAF、IPS、云安全中心等组件增强威胁感知能力；
- 所有等保测评所需的安全日志、操作记录、权限配置信息应具备可导出、可审查能力；
- 建议配合阿里云提供的一站式等保合规方案，快速完成定级备案、系统加固、测评对接等工作；
- 可出具符合性说明文档与配置清单，便于提交至主管部门进行审核备案；
- 定期进行等保自查与第三方测评，确保持续符合监管要求。

11. 小结

本章围绕网络安全、身份与访问控制、主机与容器安全、应用与 API 安全、数据安全、日志审计与威胁检测 六大核心维度构建。在网络层面，通过互联网边界、VPC 与主机三层访问控制，结合 IPS 与 WAF 实现南北向与东西向流量的入侵检测与防护；在身份安全方面，依托云 SSO 与 IAM 实现统一身份认证与最小权限管理；主机与容器环境强化主动防御能力，覆盖防勒索、漏洞修复、基线检查等场景；应用与 API 层通过 OAuth2.0 鉴权、HTTPS 加密及 API 网关防护保障服务安全调用；数据安全贯穿静态加密、动态传输与行为审计全过程，并支持敏感数据识别与脱敏；最后，通过操作审计、SLS 日志分析、云安全中心与态势感知平台实现全方位日志留存、威胁发现与响应联动，形成闭环安全运营机制。

第五章 大模型赋能篇



CHAPTER 05



第一节 / AI Stack 大模型训推软硬一体 方案

在政企客户加速推进大模型落地的进程中，如何实现“本地部署、安全可控、高效推理、成本可控”，成为关键命题。阿里云专有云推出的 AI Stack 大模型训推软硬一体解决方案，正是为满足这一需求而设计。通过将自研 APG 服务器做为硬件基座，深度融合 Qwen、DeepSeek 等主流开源大模型，提供从芯片到应用的一站式交付能力，帮助我们的政企客户以轻量化、高性价比的方式完成大模型本地化部署。

1. 阿里云 AI Stack 核心定位

阿里云 AI Stack 大模型训推软硬一体解决方案，支持以轻量化、高性价比的方式进行大模型的本地部署，用户能够轻松采用最新的开源大模型。AI Stack 基于阿里云 APG 服务器，内置阿里 QWen3 全尺寸及 DeepSeek R1/V3 满血版等最新主流模型，提供开箱即用的大模型服务，满足数据安全、成本效率、行业合规等诉求。

该解决方案不仅可以让 AI 落地轻快好省，也支持多种拓展模式，可以灵活解锁 AI 的无限可能。主要包括以下 6 大核心优势：

- 轻快好省

基于阿里云 APG 服务器，单机具备 2/4/8/16 卡多种选择，用户可根据需求灵活选择在本地轻量化起步或部署高性能单机。

单机 16 卡可提供 1.5+TB 超大显存，仅一台即可极速运行 QWen 全尺寸和 DeepSeek R1 671B 无损精度满血版。

• 兼容易用

可集成钉钉、通义灵码、百炼专属版、QuickBI、工业智驱平台等一方、三方智能应用。
高度兼容 CUDA API, CUDA 应用代码无需修改,可直接使用 CUDA API 和自研编译器实现编译兼容。

• 开箱即用

阿里云 APG 服务器预装 Qwen、DeepSeek 等多款主流模型,内置模型秒级启动、开箱即享模型服务。

• 云边一体

单机版可作为边缘站点接入飞天企业版,打造云边一体解决方案。

• 极速性能

基于在开源 vLLM, Sglang 基础上提供的软硬一体优化推理框架 Asllm,并整合了 FlashInfer、FlashMLA、MTP、MLA 等性能优化算子,让推理性能得到极速的提升。

• 弹性扩展

阿里云 AI Stack 去中心化设计,可以通过横向扩展方式,由多台 AI Stack 单机版组成集群版,提升模型并发,扩展 RAG、Agent 等多种 AI 服务,最多可扩展到 48 台 GPU 服务器,支持多模型并发场景和训练推理的一体化调度。

2. 集成架构，覆盖全栈能力，支撑多样化应用

阿里云 AI Stack 支持 DeepSeek 及阿里 Qwen 全系列模型,深度优化高性能推理引擎,提供开箱即用大模型及应用服务。



图 1 AI Statck 软硬一体解决方案架构图

（1）硬件基础设施层

该层由阿里云自研的 APG GPU 服务器和通用 CPU 服务器组成,提供高性能、高密度的异构算力资源。APG 服务器支持多卡配置,单机可部署满血版大模型如 DeepSeek R1/V3 或 Qwen 全系列,满足本地化、高安全场景下的训练与推理需求。硬件资源通过 AI Stack 管理控制台统一纳管,支持资源调度、监控与运维。

（2）平台服务层

该层提供支撑大模型运行的核心平台能力,包括 AsLLM 推理引擎、PolarDB AI 数据库和基础中间件。AsLLM 引擎基于 vLLM 和 Sglang 深度优化,集成 FlashInfer 等高性能算子,

显著提升推理吞吐与响应速度。PolarDB AI 支持向量存储与语义检索，为 RAG 等应用提供数据底座。中间件组件包括 Prometheus 监控、AI 网关、Elasticsearch 日志系统等，保障服务的可观测性与稳定性。

（3）开发工具层

该层聚焦于提升开发效率与降低使用门槛，主要包括百炼专属版和 AI Studio。百炼专属版支持私有化部署、模型微调、知识库构建，满足数据不出域与合规要求。AI Studio 提供可视化建模、Notebook 开发、模型调试与版本管理能力，适配算法工程师与业务人员的协同开发需求。同时，平台兼容 Hugging Face、LangChain 等主流开源生态，支持灵活集成。

（4）行业应用层

该层集成丰富的行业智能应用，实现大模型能力的快速落地。包括通义灵码、法睿、点金等通义家族产品，瓴羊智能问数、DataV 可视化，以及钉钉专属 AI、安恒安全、360 等第三方合作应用。方案覆盖政务、金融、制造、医疗、教育等多个领域，支持智能客服、公文写作、辅助诊断、工业质检等典型场景，帮助企业将 AI 能力转化为实际业务价值。

通过上述四层架构，AI Stack 实现了“软硬一体、训推协同、开箱即用、场景驱动”的完整闭环，为政企客户提供了一条从大模型引入到业务赋能的高效路径。

3. 行业价值，从技术落地到业务提效

AI Stack 不仅是技术产品，更是推动企业智能化转型的生产力引擎。在实际应用中，它已帮助客户实现：

- **政务领域：**智能公文写作、政策解读、跨部门知识问答；
- **金融领域：**智能风控、网点机器人、研报报告自动生成；
- **制造领域：**设备故障预测、工艺参数优化、质检图像识别；
- **医疗领域：**辅助诊断、病历摘要、科研文献分析。

通过将大模型能力嵌入核心业务流程，企业实现了从“人工处理”向“智能驱动”的跃迁，效率提升 3-10 倍，人力成本显著下降。

4. 小结

AI Stack 大模型训推软硬一体方案，不是简单的硬件堆砌，而是阿里云多年技术沉淀与行业洞察的结晶。它以国产化、轻量化、高性能、易用性为核心，为企业提供了一条从“引入大模型”到“用好大模型”的完整路径。未来，随着更多行业客户的深度实践，AI Stack 将持续进化，成为政企客户迈向“AI 原生时代”的坚实底座。

第二节 /

MaaS 引擎驱动 AI 生产力革命

人工智能正从技术热点走向产业核心。在政策引导与技术突破的双重驱动下，大模型已进入政企客户规模化落地的关键阶段。然而，真正决定 AI 价值的，不是模型本身，而是其能否在复杂业务场景中稳定运行、持续进化并创造可衡量的业务收益。阿里云专有云 MaaS（Model as a Service）引擎，正是为解决这一核心命题而构建的一站式 AI 开发与运营平台，旨在推动政企客户实现从“模型可用”到“业务好用”的全栈智能跃迁。

1. 政企大模型应用的发展趋势及落地难点

当前，政企大模型应用正经历从试点探索向产业化落地的加速过渡。根据产业演进路径，2023 至 2024 年为场景试点期，能源、金融、医疗、政务等领域率先开展垂直模型验证；2025 至 2027 年进入技术发展期，算力、算法与数据要素协同优化，模型能力持续增强；2028 年以后将全面迈入产业化落地阶段，AI 深度融入核心业务系统，成为企业生产力的重要组成部分。

然而，这一进程面临显著挑战。首先是算力环境的复杂性。政企客户普遍采用异构硬件架构，包括 NVIDIA、海光、昇腾等多种芯片，同时混合使用公有云、专有云与本地 IDC，导致模型训练与推理难以统一调度，资源利用率低且运维成本高。其次是私域知识的处理难题。政企拥有大量非结构化数据，如合同、报告、音视频及业务日志，但这些数据分散在不同系统中，缺乏统一标准与高质量标注，难以有效支撑模型训练。最后是业务场景的高度专业化。通用大模型无法直接满足政务审批、金融风控、工业质检等垂直领域的精准需求，必须结合领域知识进行深度调优，并与现有 IT 系统无缝集成，这对技术与业务的协同提出了极高要求。

2. MaaS 引擎：一站式 AI 开发新范式

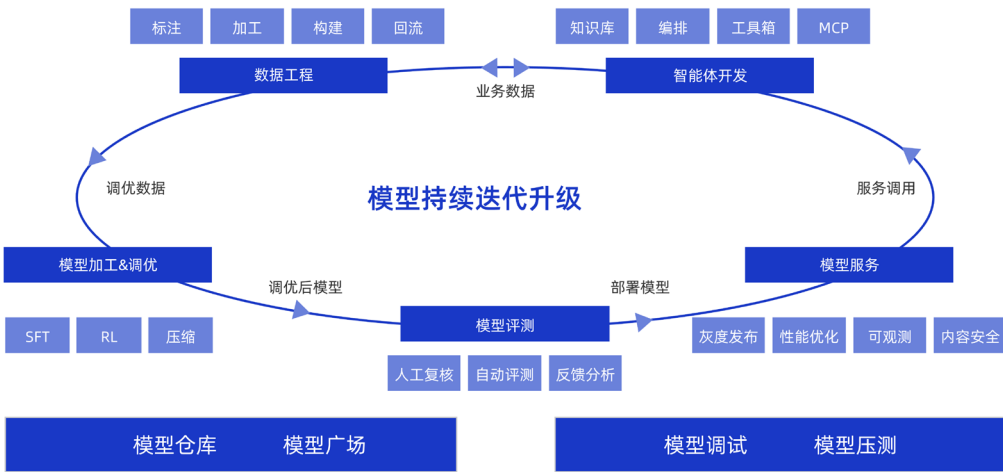


图 1 MaaS 引擎驱动的一站式 AI 开发新范式

面对上述挑战，阿里云专有云 MaaS 引擎提出“以智能体为中心”的全新开发范式。该范式不再局限于模型训练，而是覆盖从数据准备、模型调优、服务部署到反馈迭代的完整生命周期，形成可闭环、可进化的 AI 应用体系。

具体而言，MaaS 引擎通过五大核心环节实现端到端赋能。在数据工程阶段，平台支持接入多源异构数据，提供自动清洗、结构化处理与智能标注能力，快速构建高质量训练集。在模型开发阶段，支持监督微调（SFT）、强化学习（RLHF）、模型压缩与蒸馏等多种优化手段，帮助客户高效生成专属模型。在模型评测阶段，结合自动指标与人工评估，确保模型效果满足业务预期。在模型服务阶段，提供一键部署、灰度发布、性能监控与内容安全防护能力，保障服务稳定可靠。最后，在反馈迭代阶段，通过用户行为日志、问答对沉淀与业务指标分析，持续驱动模型优化升级。

这一范式显著降低了 AI 应用门槛，使业务人员也能参与智能体构建，真正实现“技术服务于业务”。

3. MaaS 引擎：全栈工具链

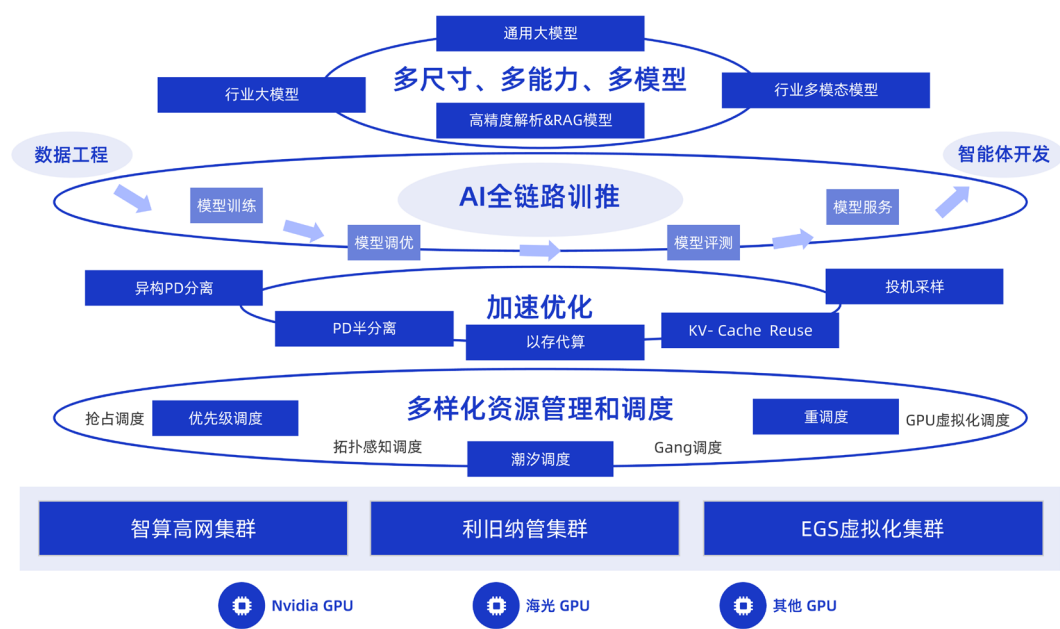


图 2 MaaS 引擎全栈工具链

MaaS 引擎的价值依托于其全栈、开放、可扩展的工具链体系。在底层基础设施层，引擎提供极致的训推一体化能力。支持通用大模型与行业多模态模型的联合训练，内置高精度文档解析、RAG 增强、KV-Cache 复用、PD 分离等优化技术，并兼容智算高网集群、利旧纳管集群及虚拟化集群，实现对 NVIDIA、海光等多种 GPU 的统一调度与高效利用。

在智能体开发层，MaaS 引擎构建了面向政企场景的智能体开发与运维平台。平台支持接入三类核心资产：私域知识（文档、音视频等）、私域数据库（如 RDS、分析型数据库）以及私域 API（HTTP/MCP/RPC 接口）。基于这些输入，通过记忆机制、知识库检索、上下文提示词、技能工具调用与模型服务调度等六大模块，灵活组合出不同类型的智能体。例如，AutoAgent 可自动生成行业分析报告，数字员工 Robot 可执行审批、填报等重复性任务，RAG 系统则支持基于企业知识库的精准问答。用户无需编码，即可通过可视化界面完成智能体搭建。



图 4 面向政企场景的应用实践

在应用实践层，MaaS 引擎已在多个行业验证其价值。在工业制造领域，某企业通过训练视觉模型实现零部件自动质检，检测效率从每人每分钟 15 件提升至 480 件，效率提升 30 倍以上。在医药行业，阿斯利康基于 MaaS 引擎构建药品警戒报告智能助手，报告准确率从 90% 提升至 95%，撰写效率提高 300%，并入选信通院《大模型落地应用案例集》。在教育科研领域，两个月内已支持超过 100 款智能体应用，广泛应用于论文解析、教材问答、科研辅助等场景，显著提升知识获取与创作效率。

4. AI 能力落地，行业生产力跃升

MaaS 引擎的终极目标是推动 AI 能力深度融入业务，实现行业生产力的系统性跃升。首先，平台助力企业从“构建智能体”迈向“沉淀知识资产”。通过结构化处理私域数据，将零散信息转化为可检索、可推理、可复用的知识库，并在此基础上持续迭代模型，形成企业专属的 AI 能力资产。

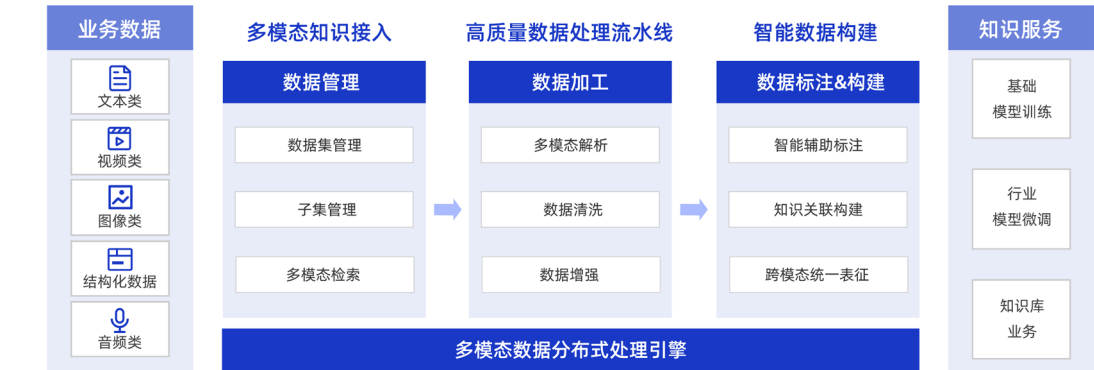


图 3 面向政企场景的智能体开发与运维平台

其次，MaaS 引擎支持从单点应用向全场景覆盖演进。多个智能体可协同工作，形成自动化业务流。例如，AutoAgent 生成月度经营分析报告，数字员工自动提交审批并跟踪进度，RAG 系统实时解答员工关于政策或流程的疑问。这些智能体还能与 ERP、CRM、OA 等核心业务系统深度集成，实现 AI 与现有 IT 架构的有机融合，避免形成新的信息孤岛。

最后，MaaS 引擎不仅是技术平台，更是价值共创的长期伙伴。阿里云专家服务团队全程参与项目，从需求调研、架构设计到迁移实施与持续优化，提供端到端支持。同时，通过定期组织行业沙龙、最佳实践分享与能力培训，帮助客户团队持续提升 AI 应用水平，实现从“项目交付”到“能力内化”的转变。

5. 小结

在政企数字化改革的新征程中，大模型已从“可选项”变为“必选项”。阿里云专有云 MaaS 引擎通过一站式开发范式、全栈工具链与行业深度实践，为政企客户提供了从技术到价值的完整路径。它让复杂的 AI 技术变得简单可用，让分散的数据资产变得智能高效，让专业的行业知识变得可计算、可进化。未来，AI 将不再是孤立的技术模块，而是嵌入业务流程的“智能生产力”。而 MaaS 引擎，正是这一变革的核心引擎，持续驱动政企客户迈向“AI 原生”的新阶段。

第三节 / AI 原生应用架构

（一）从云原生到 AI 原生——应用架构的演进与智能化浪潮

1. 应用架构的演进路径

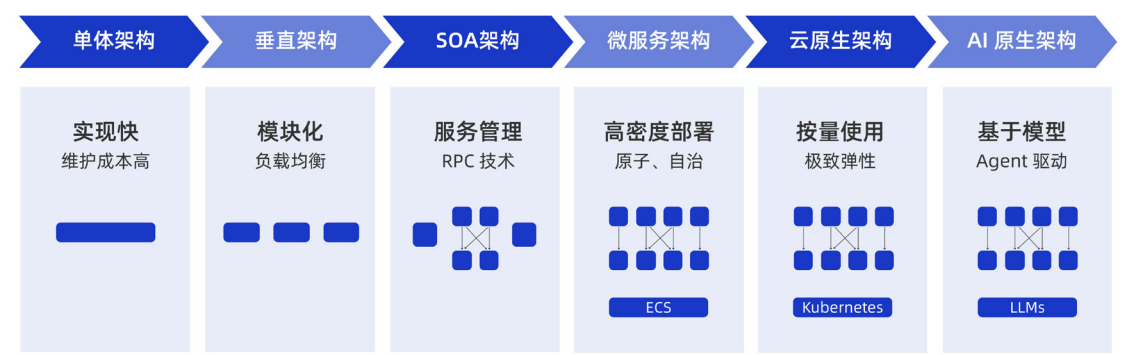


图 1 应用架构演进路径

应用架构的演进，既是技术栈迭代的历史，也是业务需求驱动的结果。回顾过去几十年，主要经历了以下几个阶段：

- 单体架构（Monolithic）

在早期互联网应用中，业务逻辑、数据访问层和前端展示紧密耦合到单一代码库和部署单元。优点是开发上手快、部署简单；缺点是随着业务增长，发布风险高、扩展困难、技术栈更新受限。

• SOA (Service-Oriented Architecture)

采用面向服务设计思想，把应用系统拆分为若干个相对独立、可重用的服务（ Service ），这些服务通过标准化接口和协议进行交互，降低系统耦合度、提高复用性和可维护性。但集中式总线带来单点瓶颈，服务粒度大且耦合度高，协议复杂导致性能下降，部署升级与扩展不够灵活。

• 微服务架构 (Microservices)

同样将应用拆分为多个独立服务，通过统一接口协议（如 REST、gRPC）通信，并进一步强调独立部署、技术多样性和团队自治，带来更好扩展性与迭代速度，但也引入了服务治理、分布式事务、网络延迟等问题。

- 云原生架构（ Cloud Native Architecture ）云原生是基于容器化、Kubernetes 编排、服务网格、DevOps、CI/CD 等技术理念构建的应用模式。它强调：
 - 弹性伸缩 & 高可用
 - 基于云环境的自动化部署与运维
 - 面向资源的动态调度与跨环境一致性

云原生为互联网和企业应用提供了稳定、可扩展的运行基座，从而大幅提升了交付效率与系统韧性。

2. 智能化时代应用架构的新挑战

在大语言模型（ LLM ）问世之前，人工智能通常以特定功能模块的形式嵌入各类系统，例如图像识别、推荐算法与风险控制模型等。这些模块依托监督学习及预设规则运行，作用范围明确、功能单一，并且不会对系统的核心架构产生实质性调整。

LLM 的出现突破了这一限制。它不仅具备跨领域的自然语言理解、逻辑推理和内容生成能力，还能够通过函数调用、外部工具集成以及知识库访问，构建高度可扩展的智能 Agent 体系。由此，人工智能的角色从单一的嵌入式功能组件，提升为支撑应用的核心基础平台。

在这样的背景下，AI 原生不是简单的下一代技术标签，而是新需求驱动下的必然演进。主要挑战包括：

• 数据复杂性与治理

AI 应用的逻辑驱动来自数据，而不再仅是业务规则：

- 海量、多模态（文本、语音、图像、视频、传感数据）数据涌入系统。
- 数据需要从采集、清洗、标注到分发整个生命周期管理。
- 数据隐私与合规（GDPR、CCPA、数据本地化政策）要求设计安全与合规的数据基础设施。

• 模型生命周期管理

AI 模型不同于传统代码：

- 需要持续训练与优化，依赖高质量数据。
- 模型版本迭代频繁，且涉及实验、上线、回滚、灰度等管理模式。
- 推理服务需支持低延迟、高吞吐以及灵活的接口适配。

• 算力调度与资源优化

AI 模型训练和推理对 GPU、TPU、ASIC 等专用硬件的需求远大于传统应用：

- 如何在多云混合环境下，动态调度异构算力资源？
- 在成本可控和性能优化间平衡（尤其是大规模推理场景）。

• 实时性与交互智能

在对话式 AI、推荐系统、智能驾驶等场景中：

- 延迟与响应速度直接决定用户体验。
- 需要边缘计算、模型压缩、知识蒸馏等架构优化手段。

• 可信与可解释性

用户与监管层要求 AI 决策具备可解释性。

- 系统需提供审计记录、可追溯的模型推理过程，以及防止模型偏差的机制。

上述挑战无法通过传统云原生模式轻易解决，因为云原生架构关注的是资源层面与服务的弹性运行，但 AI 应用本质上是 数据驱动 + 模型迭代 + 智能决策 的闭环系统，需要在架构层面内建这些能力。因此，AI 原生架构在设计之初，就将数据、模型、算力调度与智能运维等需求作为核心模块。

3. AI 原生应用架构的新特征

（1）AI 原生架构的核心特征

- 模型一等公民化（Model First）
模型与代码、数据服务处于同等地位，拥有独立的版本管理、部署与监控体系。
- 全生命周期自动化（MLOps + AIOps）
覆盖数据管理、特征工程、模型训练、评估、上线、监控、反馈优化的闭环体系。
- 智能化算力调度
融合云端与边缘计算，支持异构算力混合编排、按需分配、动态伸缩，优化性能与成本。
- 多模态与多任务支持
原生支持文本、语音、图像、视频等多模态数据，具备任务路由、动态加载与模型组合能力。
- 可信与可解释性内建
架构自带模型解释接口、日志与审计能力，以及联邦学习、差分隐私等隐私保护技术。
- 数据驱动闭环（Data-Model Loop）
生产环境中的用户行为和外部数据进入架构内的数据管道，为模型再训练和业务优化提供持续输入。

（2）建设 AI 原生架构的基本思路

- 底层能力平台化
构建统一的 AI 能力平台，包括训练集群、推理服务集群、数据湖与特征服务。
- 算力资源池化与调度
建立 GPU/TPU 等算力的虚拟化资源池，结合 Kubernetes + 调度插件实现按需分配。
- MLOps 系统化
引入自动化的数据处理、模型构建、验证与发布工具，融合云原生 CI/CD 思路。

- 服务治理智能化
在服务网格层引入模型路由与推理 API 网关，针对多版本模型进行流量分配与 A/B 测试。
- 安全与合规设计前置
从架构设计开始嵌入隐私保护与合规检查，确保模型训练与推理过程满足监管要求。

4. 小结

应用架构的发展经历了从单体架构到微服务架构，再到云原生架构的多次迭代，每一次跃迁的背后，都源于技术演进、业务场景复杂化以及用户体验不断提升的需求驱动。当智能化与人工智能技术逐渐成为各行各业的核心生产力时，一种全新的架构理念——AI 原生架构顺势而生。它并非取代云原生，而是在云原生的技术和资源基座之上，进一步引入智能驱动的核心机制，将智能能力渗透到应用的设计、开发、部署和运行全流程之中。

AI 原生的核心思想，是将数据、模型和算力这三大要素沉淀为底层通用能力，使它们像云原生中的计算、存储和网络一样，被统一管理和灵活调用。同时，通过构建一个涵盖数据采集、模型训练、推理服务和反馈优化的闭环体系，让应用可以在运行过程中不断学习新的信息，优化决策逻辑，并随着使用环境和需求的变化持续进化。

未来，随着大语言模型和多模态 AI 技术的广泛应用，AI 原生架构将会成为智能化应用的默认工程标准，它将像微服务和云原生一样，成为现代软件架构的基础支撑，为不同行业的创新提供高效、可扩展且具备自进化能力的技术底座。

（二）从模型对话到智能体——AI 应用架构的迭代发展

1. AI 应用架构模式

随着大语言模型（LLM）的快速发展，AI 应用的架构模式也在不断演进。从最初的单一对话，到融合检索与生成的模式，再到具备复杂任务规划和执行能力的智能代理（Agent）系统，不同模式在能力、复杂度和适用范围上有明显差异。下面对几种典型模式进行梳理。

（1）对话模式（Prompt-Response）

在 ChatGPT 刚发布时，最常见的 AI 应用是直接的对话模式：用户编写 prompt 提交给模型，模型基于其内部训练知识生成回答。这种模式结构简单，交互直接，适合纯知识问答或开放式聊天。

局限性：由于模型训练集的限制，该模式无法访问大量实时或私有数据，难以满足业务场景对最新信息和敏感数据的需求。

（2）RAG（Retrieval-Augmented Generation）模式

RAG 模式在对话模式的基础上增加了实时检索能力，将外部数据源（如企业知识库或互联网信息）通过查询获取，并将相关上下文传递给模型进行整合。这种模式使模型拥有实时、私有数据访问能力，提升了信息的准确性与时效性。其架构相对简洁，核心在于高效的信息检索与上下文融合，适用于问答系统、信息提取等场景。

（3）AI 工作流模式

- 在企业场景中，更复杂的模式是“AI 工作流”：
- 用户通过可视化工具编排一个固定流程，该流程包含企业积累的专家知识。
 - 在关键步骤中调用模型实现特定任务，如文本摘要、图片解析、二维码分析等。
 - 适用于流程固定、需求明确的任务。
- 此模式的优势在于稳定性和可控性，但在不确定性高、任务流程多变的场景中效果有限。

（4）Agent 模式

面对高度不确定和动态变化的任务，例如 AI 编程（AI Coding），固定工作流已无法满足需求。那么，我们就需要使用 Agent 模式，Agent 模式的核心是**自主规划、执行、感知与反思的循环**：

- 1. 用户交互模块：**解析用户请求，收集完整上下文信息（如代码文件、语言版本、依赖环境、代码仓库地址等）。
- 2. 核心 LLM 模块：**进行任务规划与短期记忆管理，根据上下文生成执行步骤（如代码分析、环境准备、编译测试等）。
- 3. 环境模块：**在隔离的 sandbox 中执行任务，调用外部工具并收集环境反馈（如测试日志、错误信息）。
- 4. 感知与反思机制：**根据执行结果动态调整任务规划，实现自我纠错。
- 5. 长期记忆模块：**在上下文过长或任务复杂时精细压缩历史信息，防止模型性能劣化。

- Agent 模式的架构特点包括：
- **基础设施要求高：**用户交互模块需精准理解上下文；环境模块需保证执行稳定性与实时反馈；工具链需覆盖问题领域；记忆管理需平衡信息量和有效性。
 - **安全设计重要：**需防范恶意注入，保证环境安全隔离。

从**对话模式 → RAG → AI 工作流 → Agent**的演进，体现了 AI 应用架构从简单交互到复杂任务管理的逐级升级。对话与 RAG 模式适合知识获取，工作流适于固定场景，而 Agent 模式则能支持动态、高复杂度的业务任务。企业在设计 AI 应用架构时，应根据任务复杂度、数据类型、实时性要求和安全性考量，选择合适模式或混合架构，以充分发挥 AI 的能力。

2. 工作流模式

在上一章节中，我们从整体视角介绍了 AI 应用的常见架构模式，了解了不同设计思路背后所关注的关键问题：如何让模型产出的结果更稳定、更可控、更易维护。本章节将从其中一个重要且基础的维度——**工作流模式（Workflow Pattern）**——开始深入探讨。

(1) workflows 模式的定义与特点

workflows 是一种编排（orchestration）模式，它要求开发者在设计阶段将复杂任务拆解为一系列有序步骤，并明确这些步骤之间的调用关系与条件分支。相比动态、自由度高的模式，workflows 倾向于提前建立“任务执行路线图”，每一步该做什么、做完后进入哪一步，都有清晰的规则约束。

例如：

- **链式 workflows（Chain）**：先执行第一步，再将结果传给第二步；如此依次连接形成链条。
- **路由 workflows（Routing）**：先通过判断或分类选择路径，再进入对应的处理模块。

workflows 的优势在于：

- **确定性强**：任务执行顺序和分支逻辑在设计阶段已明确。
- **可调试性高**：每一步都有可观测的输入和输出，便于发现问题。
- **可控性好**：对稳定性要求高的场景非常适用。

简单来说，workflows 就像一条固定的流水线，它的价值在于让任务执行过程透明、可追踪、安全且可重复。

(2) 链式 workflows（Prompt Chaining）

核心思想：将一个任务拆解成一系列连续步骤，每一步由一次 LLM 调用完成，并使用上一步的输出作为下一步的输入。

在链式 workflows 中，可以在步骤之间加入 程序化检查（Gate），确保中间产物符合预期或满足质量要求。如果某一步结果异常，可以进行修正或回滚，从而防止错误在流程中扩散。

适用场景：

- 任务天然可拆分为多个明确阶段（如“先生成提纲 → 再根据提纲撰写全文”）。
- 每一步结果对后续影响重大，且需要中间质量控制。

(3) 路由 workflows（Routing）

核心思想：对输入进行分类，根据类别路由到不同的下游流程或模块执行任务。

这种模式能让各个分支针对特定类别进行优化，减少“一刀切”式处理带来的性能下降。路由可以由 LLM 完成分类，也可由传统的机器学习分类器或规则算法实现。

适用场景：

- 输入类型差异较大，不同类别适合用不同 Prompt 或模型处理。
- 希望优化某类输入的同时，不影响其他类别的效果。

(4) 并行 workflows（Parallelization）

核心思想：将任务拆解为可以并行处理的多个子任务，同时交给多个 LLM 实例完成，再将结果在程序中聚合。

并行化有两种重要变体：

- 1. **为速度优化**：时间敏感的任务，可并行处理不同部分以加快整体完成时间。
- 2. **多视角 / 多尝试**：针对高度不确定性问题，多路执行，然后合并结果，以提高可靠性。

适用场景：

- 子任务之间相互独立，可同时处理。
- 同一问题需要多个观点或解法来增加结果可信度。

(5) 协调者 – 工作者模式（Orchestrator Workers）

核心思想：由一个中央 LLM（协调者）动态分析任务，将其拆解为若干子任务，并分配给多个工作者 LLM 执行，最后由协调者收集和综合结果。

与并行化模式相似，两者在结构上都有分发和合并的流程，但关键区别在于：

- **并行化**：子任务通常在设计阶段已确定。
- **协调者 – 工作者**：子任务是由协调者根据具体输入 动态生成 的。

适用场景：

- 无法提前预知任务结构，比如代码改动的数量和位置，需根据输入实时分析。
- 输入复杂度高，需要灵活生成任务分配方案。

（6）评估者 – 优化者模式（Evaluator Optimizer）

该模式通常配套在工作流中使用：评估者模块检测某一步（或最终结果）的质量、正确性或合规性，如果未达到标准，则由优化者模块重新执行或改进结果。

这种结构让工作流具备闭环优化的能力，而不仅仅是顺序执行。

工作流模式让 AI 应用的任务处理变得像工厂流水线——逻辑清晰、步骤可控、结果可追踪。在任务复杂、对稳定性或可复现性要求高的场景中，这类模式是架构设计的重要基石。链式执行、路由分发、并行协作，以及协调者动态调度，都是工作流模式的常见实现形式。随着业务复杂度和自动化需求的提升，工作流模式往往与评估优化环节组合使用，以构建更完善、更可靠的智能系统。

3. 智能体架构

在上一章节中，我们探讨了 AI 应用的工作流模式，其核心是在设计阶段将复杂任务拆解为有序、可控的步骤，并通过明确的调用逻辑完成整体任务。工作流强调的是 事先规划与稳定执行，而在本章节，我们将从这一思路自然过渡到更具自主性和动态适应能力的 智能体（Agent）架构。

如果说工作流像一条固定的流水线，那么智能体则更像一个“具备自主思考与执行能力的数字化助手”。它不仅能够按照设定的逻辑完成任务，还能基于感知、推理和上下文记忆进行动态决策，并通过调用外部工具直接执行行动，从而从“内容生成器”升级为一个真正的“行动者”。

（1）智能体的核心能力

一个成熟的 AI Agent 一般具有以下关键能力：

1. 感知（Perception）

能理解用户的自然语言指令或环境信息，并提取可执行的任务目标。

2. 规划与推理（Planning & Reasoning）

结合大模型的推理能力，将任务拆解为合理的子任务，并设计执行顺序和策略。

3. 记忆（Memory）

记录用户偏好、历史互动及任务状态，包括短期记忆（当前会话上下文）与长期记忆（跨会话知识与经验），为后续任务提供持续参考。

4. 工具使用（Tool Use）

调用外部系统和 API，如数据库、搜索引擎、业务系统，完成模型自身无法直接实现的操作。

5. 反馈与迭代（Feedback & Iteration）

根据用户的反馈调整方案，循环优化，直到任务满足目标要求。

（2）AI Agent 架构的四大核心组件

1. 大脑（LLM）

- 作用：理解自然语言、进行推理和决策。
- 原则：根据业务场景选择最匹配的模型，不同 LLM 在语言理解、逻辑推理、领域知识覆盖上各有优势。

2. 手脚（工具层 / MCP Server）

- 作用：赋予 Agent 与外部世界交互的能力，包括调用 API、数据库、文件系统等，以执行大脑制定的决策。

3. 指令（System Prompt）

- 作用：为 Agent 定义目标、角色与行为边界，确保其行动符合既定策略与约束。

4. 记忆（Memory Store）

- 作用：存储目标、用户偏好与交互记录，通常通过 NoSQL 数据库或向量数据库实现，支持短期与长期记忆，从而在多步骤任务和自省过程中保持上下文连续性。

(3) workflows 模式到智能体架构的演进

在 workflows 模式中，任务路径和工具调用通常由开发者预先设定，适用于流程固定、稳定性要求高的场景；而智能体架构则引入了 动态规划与自主决策能力，能够根据环境变化或用户反馈灵活调整任务执行方式。

这种演进让系统从“被动执行”走向“主动探索”，不仅能完成预定义任务，还能发现潜在需求并付诸行动。这对于构建具有持续交互、跨领域能力的下一代 AI 应用至关重要。

智能体架构是 workflows 模式的自然升级——在保留有序任务执行的基础上，赋予 AI 自主感知、规划、记忆和工具使用的能力，使其能像智能助理一样完成复杂、动态、跨领域的任务。

4. AI 应用核心技术 – 提示词工程与上下文工程

在上一章节，我们探讨了 智能体（Agent）架构，它让 AI 从单纯的语言模型成长为一个具备感知、规划、记忆和工具使用能力的自主行动者。而在更上一个章节，我们分析了 workflows 模式，强调固定步骤、可控逻辑在稳定任务中的优势。中间的主线是：

- workflows 模式 → 规则与流程驱动的有序执行
- 智能体架构 → 自主决策、动态适应与多能力协作

然而，无论是严格的工作流，还是灵活的智能体，它们的 行为基础 都离不开一个关键环节：模型如何理解我们的意图。在当前 LLM 主导的 AI 应用中，影响模型理解与执行质量的核心技术就是 提示词工程（Prompt Engineering）与其演进形式 上下文工程（Context Engineering）。

这一章节，我们将深入剖析这两大技术的原理、实践方法、局限性，以及它们在下一代 AI 应用中的角色与关系。

(1) 提示词工程（Prompt Engineering）

大型语言模型（LLM）是 AI 原生应用的技术核心，其输出质量往往高度依赖于 输入文本 的精确性与引导性。提示词工程，就是围绕如何设计、构建和优化输入文本，以引导模型产生期望输出的一套系统化方法。

不同于简单的用户提问，提示词工程涉及：

- 指令设计（Instruction Design）
- 上下文注入（Background Context Injection）
- 角色设定（Role Definition）
- 输出格式控制（Output Formatting）

一个精心设计的提示词不仅能激发模型的潜力，还能确保输出的 **准确性、相关性和一致性**。在 AI 应用早期阶段，它是**用户意图到模型可执行指令的核心接口**，也是“低成本驾驭 LLM 能力”的重要途径。

(1.1) 高质量提示词的核心实践

实践表明，优秀的提示词工程应遵循以下原则：

1. 明确角色与目标

在提示词开头为 AI 赋予一个特定的专家角色，能有效激活模型内部对应领域的知识库与语言风格。

示例：

假设你是一位拥有 10 年经验的 市场营销总监，请为一款新型智能手表起草一份面向年轻专业人士的产品介绍文案。

2. 提供清晰指令与完整上下文

明确告知任务目标，并提供必要的背景信息，减少歧义。

示例：

请总结以下技术文章的核心观点（不超过 200 字），并列出其中提到的三个关键数据。不要添加个人评论。

文章内容如下：[粘贴文章]

3. 运用范例进行引导

对于复杂任务，提供 1-2 个输入 – 输出示例远比冗长描述更有效。

示例：

输入：我喜欢这部电影的剧情。

输出：{"sentiment":"positive","aspect":"plot"}

4. 定义结构化输出格式

要求模型返回 JSON、Markdown 等可解析格式，便于下游程序直接使用。

示例：

请将你的回答组织成 JSON 格式，包含 'product_name' 和 'key_features' 两个字段，key_features 至少包含三项功能。

提示：OpenAI 的最佳实践建议在提示词开头放置指令，并用分隔符（如 ###、"""）将指令与上下文分开，具体、详细地描述期望结果与风格，并通过示例明确输出格式。

（1.2）提示词工程的固有局限性

虽然提示词工程在提升单次交互效果上非常有效，但随着 AI 应用向更复杂的 Agent 系统与长流程任务 演进，它的局限性逐渐显现：

1. 难以应对动态与长流程任务

提示词是静态模板，无法灵活适应环境变化或分支逻辑。

2. 受制于模型的无状态特性

模型不会自动记住历史，需要手动拼接对话记录，成本高且受上下文窗口限制。

3. 无法突破知识边界

提示词只能优化对既有知识的访问，不能扩展模型知识库，无法处理实时更新或私域数据。

4. 长上下文信息利用不均衡

“中间遗忘（Lost in the Middle）” 现象使模型对提示词中间部分的信息关注度低，影响任务成功率。

这些问题表明，单纯优化一次输入的方式，难以支撑更复杂、更智能的应用需求。于是，开发者需要一种新的范式——上下文工程。

（2）上下文工程（Context Engineering）

上下文工程核心理念是：为模型的每一次推理动态构建一个完整、准确、高效的认知环境 它比提示词工程更宏观，不再局限于单次交互的指令优化，而是构建一个能够：

- 实时获取外部知识（包括私域与实时信息）
- 调用合适工具完成任务
- 管理长期与短期记忆
- 高效组织有限且昂贵的上下文窗口

在现代 AI 应用，尤其是 Agent 架构中，上下文工程是系统级的基础设施，决定了模型的推理质量、响应速度与可靠性。

（2.1）上下文工程 vs 提示词工程

对比维度	提示词工程	上下文工程
核心目标	优化单次交互的指令	构建动态上下文系统，确保推理准确性与可靠性
工作范围	单轮或有限多轮交互	整合多源异构数据流与工具
关键技术	指令设计、范例选择、角色设定	检索增强生成（RAG）、向量数据库、 workflow编排、记忆管理

表 1 上下文工程和提示词工程对比表

简单来说：

- 提示词工程是“术”——方法层面解决一次对话的效果
- 上下文工程是“道”——系统层面构建持续解决问题的认知框架

（2.2）上下文工程的关键组成部分

1. 外部知识库的动态供给

通过检索增强生成（RAG），将私域、实时或互联网上的知识动态整合进上下文，解决知识滞后问题。

2. 长期与短期记忆系统

- 短期记忆：当前会话的上下文，保证多轮对话精准连贯。
- 长期记忆：跨会话知识、用户偏好、历史任务等，实现个性化服务。

3. 工具能力扩展

在上下文中描述可用工具，使模型通过 API/Fn 调用直接执行查询、计算、操作设备等任务。

4. 运行时上下文管理

- 上下文剪枝 (Context Pruning)：移除无关信息，控制 Token 消耗。
- 上下文压缩 (Context Compression)：摘要保留关键信息，确保超长对话依然可用。
- 上下文重排 (Re-ranking)：优化关键信息的位置，缓解“中间遗忘”问题。

分层记忆架构示例：

- 短期记忆：完整保留最近交互信息
- 中期修剪记忆：保留事件类型、时间戳、核心参数等元数据
- 长期压缩记忆：LLM 总结任务快照，提炼用户需求、关键步骤、状态信息

(3) 小结

- 提示词工程 更侧重于一次交互的指令优化，解决“如何提问”的问题。
- 上下文工程 则是系统化管理模型认知环境，解决“模型在每个思考瞬间获取哪些信息”的问题。
- 从工作流模式到智能体架构，提示词与上下文工程是让系统真正理解意图、保持连贯、灵活调用能力的关键。

5. 多智能体架构

从单个工作流或单个智能体出发，我们已经可以完成相当复杂的任务。然而在真实业务场景中，复杂性不仅来自任务本身，还来自任务类型的多样性、执行步骤的异构性、以及任务之间上下文的灵活传递需求。这时，单智能体架构会遇到执行效率下降、任务计划受限、注意力涣散等瓶颈。这就引出了本章节的主题——多智能体架构 (Multi-Agent Architecture)。

(1) 为什么需要多智能体

业内经验表明，大语言模型在处理长提示词时常会出现“注意力发散” (Attention Diffusion) —— 关键信息被稀释，推理性能下降，尤其在复杂场景中容易出现：

- 步骤缺失 (漏掉关键环节)
- 步骤重复 (无效循环)
- 计划与执行脱节

即使智能体将推理与行动交替执行，也无法完全避免这些问题。根本原因在于：当任务过于复杂时，交给单个智能体处理会导致提示词过长、上下文臃肿，从而影响模型聚焦关键内容的能力。

常见的解决思路是任务分解：

- 先将复杂任务拆解为若干简单子任务
- 再利用 工作流智能体 (Workflow Agent) 按计划依次执行这些子任务

然而，工作流智能体依旧是单体架构，其节点类型、执行能力、输入输出格式 都需在工作流设计阶段固化，这在面对高度多变的用户需求时显得笨重且缺乏弹性。

因此，我们需要一个能够：

1. 动态将复杂任务拆解为多类型子任务
2. 任务编排灵活可变
3. 将子任务分发给 最适合的不同类型智能体 分别执行
4. 自动收集并整合多智能体执行结果的多智能体架构。

(2) 多智能体通信机制

要让多个智能体协同完成任务，它们必须能够 **高效、准确地交换信息**。这里的挑战是：不同执行步骤之间**存在依赖关系**，下游智能体既需要知道上游执行的结果，也需要获取上游产生的**非直接输出信息**（如 token、状态标识等）。

(2.1) 直接合并上游输出的问题

- 在部分开源实现框架里面的做法是：
- 将之前所有步骤的执行结果合并为一个长字符串，作为当前智能体的输入上下文交给 LLM 自行解析，再提取当前所需信息。
- 这种方式在实践中暴露了几个问题：
- 1. **信息失真**：执行结果在汇总和提炼中可能丢失关键数据。
 - 2. **结构化丢失 & Token 增长**：字符串化后丢掉结构，下游需要额外解析，增加推理负担和 Token 成本。
 - 3. **上下文窗口限制**：当信息过长，可能超出模型窗口，导致部分历史信息无法利用。
 - 4. **文件类输出不适配**：当步骤产生文件（如报告），字符串注入方式无法直接处理。

(2.2) 改进：结构化智能体通信协议

- 为应对上述问题，在保留基本通信流程的同时，可以采用结构化上下文协议：
- 多智能体的消息传递保持独立性，**不直接耦合上游业务逻辑**
 - 在通信包中**保留结构化数据**（如 JSON / Markdown）
 - 对模型可读信息与不可读数据分层存储，以减少信息丢失和解析复杂度
- 这样，下游智能体既可直接读取已解析的关键信息，也能在必要时引用完整原始数据，而不必通过冗长的自然语言解析。

(3) 智能体上下文管理

- 通信规范解决了**结构化传输**的问题，但在多智能体协同执行中，还需处理**任务执行过程中产生的上下文状态**。
- 例如：
- 上游智能体生成了一个临时 token（非正式结果），需要传给下游 API 调用
 - 工具调用返回了一些元信息，不适合放进最终输出，但对下游操作至关重要
- 这就要求多智能体框架中，每个智能体内部都需要一个**执行上下文管理机制**，用于：
- 存储运行时产生的临时信息
 - 将信息区分为 **对 LLM 可见 与 仅对系统可见**
 - 在上下游节点间传递这些信息，而不污染最终对用户的输出
- 在实现中，可以通过一个 **Agent 内存上下文管理类** 将：
- 工具 / 模型调用结果 存入记忆（Memory）
 - 运行相关信息（例如 token）存入内存上下文（Runtime Context）
- 这种做法保证信息只在需要的环节被访问，同时保持模型调用上下文的整洁。

(4) 多智能体记忆管理

多智能体架构中，**记忆（Memory）**设计尤为关键。

(4.1) 独立记忆 vs 共享记忆

- 在实践中，不同智能体之间业务逻辑差异大，如果共享记忆，很容易引发：
- 语境混淆（同一个词在不同 Agent 上下文含义不同）
 - 记忆污染（不相关的历史干扰当前推理）
- 因此建议采用**独立记忆**策略——不同智能体的记忆空间彼此隔离，即使在同一用户、同一会话下，也保持独立。

(4.2) 记忆持久化与检索 key 设计

可以通过类似 “Agent 标识 + 用户标识 + 会话标识” 这种 Key 结构的设计，来实现跨会话检索，避免多轮会话中的上下文丢失问题。

(4.3) 记忆管理与对话存储

- **记忆管理器**：在调用 LLM 前 / 后，负责注入历史记忆并保存新记忆
- **对话存储库**：负责消息的具体存储与检索逻辑
- **调用流程**：
 - a. 调用前，记忆管理器 将新输入保存，并检索最近 N 条历史记忆，拼装 Prompt
 - b. 调用后，将模型响应存入记忆（作为 AssistantMessage）
 - c. 保证对话中仅有一个 SystemMessage，有效约束模型角色

(5) 多智能体协作流程示意

一个典型的多智能体执行流程如下：

- 1. **任务解析与拆分**：调度器（Orchestrator）将复杂任务切分为可独立执行的子任务
- 2. **智能体分配**：根据任务类型、能力匹配度，将子任务分配给不同的专家型智能体
- 3. **执行与通信**：各智能体接收结构化上下文，在本地执行，并将结果与上下文状态一并返回
- 4. **结果融合**：调度器整合所有智能体的结果，生成最终输出
- 5. **记忆更新**：各智能体独立更新自己的记忆，用于后续任务的持续优化

(6) 小结

多智能体架构的关键收益：

- **并行与专精**：不同任务由最适配的智能体处理，提升准确率与效率
- **结构化通信**：减少信息失真、避免无谓 Token 浪费

- **独立记忆管理**：防止语境污染，保持智能体专注于自身任务
- **运行时上下文传递**：让隐性执行信息顺利流向需要的节点

从工作流模式到单智能体，再到多智能体，这是 AI 应用架构的自然演进路径。多智能体不仅融合了前面章节中提到的 **工作流编排**、**上下文管理**、**记忆系统**，更在系统协作和任务动态适配上跨越了一大步，为构建可应对现实复杂场景的 AI 系统奠定了基础。

6. AI 应用的开发模式—从设计到落地

在前几章中，我们已经从架构与核心技术的角度，系统分析了 AI 应用的构成要素与演进路径：

- **AI 应用架构模式** 给出了系统的大框架；
- **工作流模式** 让任务执行可控、可复现；
- **智能体架构** 赋予 AI 自主感知、规划、记忆与行动能力；
- **提示词与上下文工程** 让模型在每一次推理中获取最优的“认知环境”；
- **多智能体架构** 则解决了复杂任务的分工协作与上下文传递问题。

理解了这些“内部结构”和“运作机理”之后，我们终于要面对一个更现实的问题：**企业应该如何真正构建并落地 AI 应用？**

在实际落地中，企业的 AI 项目无外乎两条路径：**全新开发** 与 **存量改造**。

(1) 构建 AI 应用的两条路径

(1.1) 全新开发 — 开创业务新大陆

- **场景特点**：从 0 到 1 创造全新的业务场景或颠覆性产品
- **优势**：不受历史系统束缚，可以用最先进的架构与技术栈，从一开始就深度嵌入 AI Agent 的能力
- **目标**：实现颠覆式创新，探索 AI 驱动的新市场、新服务模式

- **适用案例：**
 - 构建一个面向证券和投资领域的 AI 研究分析师
 - 打造一个企业内部的 **超级知识中枢**
 - 推出一个 AI 驱动的垂直领域专家平台
- 全新开发更像是一张白纸，可以直接应用本书前几章介绍的 **多智能体、长上下文管理、工具调用** 等新一代架构能力，不必考虑历史系统兼容问题。

(1.2) 存量改造 — 为核心引擎注入 AI 动力

- **场景特点：** 基于企业既有业务系统（ERP、CRM、SCM、OA 等），通过嵌入 AI Agent 为其增加“智能化”功能
- **优势：** 直接作用于现有核心流程，中短期价值释放快、ROI 明确
- **路径：** 将业务入口从传统的“请求 → 服务”模式，升级为“请求 → AI Agent → 服务”模式
- **适用案例：**
 - **CRM 增强：** 自动分析客户行为，推荐下一步营销或销售动作
 - **ERP 智能预测：** 实时数据驱动的需求预测、库存动态优化
 - **OA 智能化：** 自动撰写公文、智能填表、优化审批流

存量改造的关键是：“在不破坏现有主业务逻辑的前提下，嵌入足够灵活的 AI 能力模块”，示例上可以是某个 API 层级的 AI Agent，也可以是业务前台的智能化助手。

(2) AI Agent 的构建模式

在实际落地时，AI Agent 的构建方式大致可以分为 **编码构建** 和 **低代码构建** 两大类。

(2.1) 编码构建类

- **方式一：纯手工编码**
- 基于前述 AI Agent 核心组件（大脑、工具、记忆、系统提示词）和推理模式，从零代码实现。适合对架构有完全掌控需求的团队。

- **方式二：基于综合框架二次开发**
- 借助市面上成熟的 AI Agent 框架快速实现基本能力，再根据业务特点扩展。
- 常见框架：
- 注意：** 多数框架的推理模式仍有局限，需要根据客户特定场景额外编码优化。
- Spring AI Alibaba
 - agentscope
 - LangChain / LangGraph

(2.2) 低代码构建类

通过可视化 **流程编排器（Flow Builder）**，以拖拽方式灵活配置智能体与工具的调用逻辑，快速搭建可运行的 Agent 应用。

主流平台如：阿里云百炼、Dify 等均已提供此类低代码开发能力，显著降低大模型应用构建门槛。

低代码方式尤其适用于以下场景：

- 业务流程已具备清晰定义的工作流
- 开发资源有限，需高效交付
- 需快速验证 AI 能力在现有业务中的可行性与价值

(3) AI Agent 的三种典型类型

1. 辅助基模（基础大模型）的 AI Agent

- 主要用于增强大模型本身的功能
- 辅助联网搜索、深度研究、编程等
- 代表：Qwen3、智谱 GLM 等提供的 Agent 模块
- 构建方式：多为 编码模式，由基模型厂商直接开发并内嵌

2. 作为独立产品的 AI Agent（通用 Agent）

- 面向终端用户的自主应用，多为 Chat 模式
- 提供任务规划、问答、执行等能力
- 代表：OpenManus、JManus、MiniMax Agent、昆仑万维 Agent
- 构建方式：以编码为主

3. 辅助现存业务的 AI Agent

- 与企业特定业务强绑定
- 针对内部 ERP/CRM/OA 等系统增加 AI 功能点
- 代表：知乎、运满满、义乌小百货等的业务助手
- 构建方式：多采用 低代码，快速嵌入业务流程

（4）小结

企业在落地 AI 应用时，首先需明确采用**全新开发**还是**存量改造**的路径：

- 全新开发：适用于旨在颠覆行业格局或打造创新型产品的企业，强调从零构建差异化 AI 能力；
- 存量改造：适用于希望快速提升现有业务效率与竞争力的场景，聚焦在已有系统中注入 AI 能力。

在 AI Agent 的构建方式上，可选择 编码开发 或 低代码平台：

- 编码开发：支持高度定制化与全功能覆盖，适合复杂场景，但对人力与周期投入要求较高；
- 低代码平台：可快速验证业务假设、加速 PoC 到 MVP 的转化，但灵活性和底层控制能力相对受限。

根据业务目标与集成深度，AI Agent 可进一步划分为三类：

- 基模增强类（增强大模型基础能力）、
- 通用产品类（独立提供标准化 AI 服务）、
- 业务嵌入类（深度耦合企业现有流程）。

不同类别直接影响技术选型、架构设计与落地策略。

本章从 AI 应用架构原理、核心技术组件、多智能体协同架构到实际开发与落地模式，进行了全链路解析，可以为后续企业 AI 应用系统化实施提供方法参考和实践支撑。

(三) 大模型能力的延伸——智能体与工具服务的整合

1. AI 工具调用能力的发展

在大语言模型（LLM）的能力迭代中，“工具调用”（Tool Invocation）无疑是近两年最重要的突破之一。它使得模型不仅能生成文本,还能主动选择并调用外部功能或数据源来完成任务，从而从“会说”走向“会做”。本文结合 OpenAI 2023 年推出的 Function Call API，以及后续衍生的各种方案，探讨工具调用的发展脉络、关键技术原理，以及标准化的未来趋势。

(1) Function Call 的诞生与工作原理

OpenAI 在 2023 年引入了 Function Call API，以标准化方式让开发者将可被模型调用的工具（例如天气接口、数据库查询、商品列表 API）注册到上下文中。其协议结构中，工具列表由 JSON 中的 tools 字段定义，多轮对话历史存放在 messages 字段中。

工作流程如下：

1. 用户输入经过包装和格式化，生成一个包含工具列表和历史对话的 prompt；
2. 模型在推理过程中，根据任务场景自动选择是否调用某工具；
3. 返回的调用结果包含工具名和提取出的参数（也是 JSON 格式）；
4. 外部系统执行该工具并将结果作为下一轮对话上下文提供给模型；
5. 模型基于最新上下文继续对话或调用其他工具，形成闭环。

这种机制在体验上非常自然：就像给模型一本说明书，告诉它可用的工具是什么、历史对话如何，然后它自动完成工具选择并反馈结果。

(2) Function Call 为什么能做好工具调用？

核心原因在于 special tokens 与后训练（post-training）过程。

在 OpenAI 的训练体系中，诸如 <|im_start|>、<|im_end|> 的特殊标记（special tokens）并非在大规模预训练阶段就出现，而是专门在后训练阶段引入，并在大量微调样本中反复出现，让模型学会它们的语法和语义作用。

类比教学过程：就像我们教小朋友写故事，告诉他“故事开始写‘开始’”，“结束写‘结束’”——多次示范后，小朋友自然就记住这些标记的用法，并在创作时熟练使用。

Function Call 的能力就是这样训练出来的：在微调数据中大量展示“问题场景 → 工具调用 → 工具执行结果”的完整流程，让模型明白在什么条件下需要调用工具，以及调用的格式如何。

一个典型案例是“点餐助手”：

- 用户请求热饮推荐；
- 模型判断需要调用 API query_product_list；
- API 返回饮品列表（如蜀实耙耙柑等热饮）；
- 模型将列表加入到下一轮对话中，继续与用户交流并完成任务。

(3) Function Call 的“事实标准”困境

Function Call 有一个尴尬的现状：它并非行业统一标准。

虽然 OpenAI 的协议影响力大，但其他厂商可以自行定义格式——有人叫 function，有人叫 tool，返回 JSON 的结构也可能不同。这对开发者意味着：

- 如果你更换大模型供应商，就可能需要重写适配代码；
- 工具无法“一次开发、多次复用”，难以做到完全可插拔化；
- 不同协议之间缺乏中间抽象层，导致生态碎片化。

正因如此，开发者在跨模型接入时常面临高成本适配问题。

(4) 两种解决思路

针对工具调用的标准化与可复用性问题，业内大致有两条思路：

1. CodeAct 路线：

直接让模型生成运行代码，然后在外部提供一个可执行环境来运行这些代码，不再依赖固定的工具协议。

- 优点：协议自由，工具按需动态生成；
- 缺点：代码生成在复杂场景仍不稳定，尤其安全性、性能难控。

2. 中间层协议路线：

加一层适配器，把 Function Call 的厂商差异屏蔽掉，让开发者只需对接统一的接口标准。

- 中间层会将不同厂商的 Function Call API 转换为统一格式，或在模型不支持工具调用时动态生成含工具列表的 prompt；
- 可以设计成 C/S 架构：中间层是客户端，工具是服务端，通过特定通信协议（如微服务接口或 HTTP）交互。

近年来，第二条路线衍生出了 **MCP（Model Context Protocol）** 这样的新协议，试图在跨模型工具调用上实现“统一语言”。

工具调用能力的发展，从 OpenAI 引入 Function Call 到新兴 MCP 协议，核心驱动力一直是**让模型自主调用外部能力完成任务**。

- Function Call 通过后训练和 special tokens 让调用逻辑变得可学习且稳定；
- CodeAct 将调用变成模型生成代码执行，提升了灵活性；
- 中间层协议（如 MCP）试图统一多厂商生态，降低开发者适配成本。

可以预见，未来的工具调用能力将往 **跨模型标准化、自动化工具生成** 的方向演进。中间层协议可能会成为关键基础设施，推动 AI 从单一模型应用走向跨生态的“万能执行代理”。

2. MCP 技术原理与优势

(1) 什么是 MCP？

MCP（Model Context Protocol）是 Anthropic 主导推出的一个**开放、通用、且有共识的协议标准**。它的目标是为大型语言模型（LLM）提供一个类似 USB-C 的统一接口，让 AI 模型可以与各种数据源和工具实现无缝交互。

在过去，AI Agent 的实现往往依赖个性化、碎片化的代码集成，每个服务的接入方式都不一样，开发者需要反复编写适配逻辑，导致效率低、可维护性差。MCP 的出现，将这种非标准化的整合方式变成基于协议的通用机制：

- 服务商只需按 MCP 协议开发自己的工具和数据接口，就能直接为各类支持 MCP 的 AI 模型所用。
- 开发者可以快速组合已有工具，构建功能更强的 AI Agent，而不必“重复造轮子”。
- 生态圈越庞大，AI 的能力就越强，形成正向循环。

一句话，MCP 就是 LLM 的“万能适配器”，消除了不同平台之间的集成壁垒。

(2) MCP 架构原理

MCP 采用客户端 – 服务器模式进行设计，核心组件包括：

- **MCP 主机（MCP Hosts）**：发起请求的 AI 应用程序，例如聊天机器人、AI 驱动 IDE、智能搜索引擎等。
- **MCP 客户端（MCP Clients）**：驻留在主机内部，与 MCP 服务器建立一对一的连接。
- **MCP 服务器（MCP Servers）**：向客户端提供上下文（context）、工具（tools）、提示信息（prompts）等。
- **本地资源（Local Resources）**：如本地文件系统、数据库，可由 MCP 服务器安全访问。
- **远程资源（Remote Resources）**：如第三方 API 提供的数据服务。

这种架构的好处是——无论资源位于本地还是远程，MCP 都能在统一的协议框架内进行调度与调用。

(3) MCP 与 Function Call 的区别

在 MCP 之前，业界在提升 LLM 自主处理能力的方向上，普遍采用 Function Call 技术。

Function Call 优势：

- 模型可主动调用预定义函数获取信息或执行任务，具备一定的自动化能力。
- 对简单工具集成场景非常有效。

Function Call 局限：

- 强依赖特定平台（如 OpenAI、Google 自家的接口），缺乏通用性。
- 每个平台的实现细节不同，导致切换模型时需要重写大量代码。
- 跨平台协作困难，生态碎片化，安全与交互模式不统一。

MCP 的改进：

- 统一标准：任何支持 MCP 的模型都能调用相同格式的工具接口，避免平台锁定。
- 标准化上下文：不仅传递函数调用参数，还传递结构化的上下文信息，让模型在选择工具、处理任务时更精准、更可靠。
- 跨应用可保持会话与状态信息，支持长链路、多阶段任务执行。

可以看作，Function Call 是**局部的工具调用机制**，而 MCP 是面向生态和标准化的**完整模型上下文协议**。

(4) 模型如何智能选择工具

MCP 不只是一个被动的“接口”，它在调用工具前，模型会经过一个智能决策过程：

- 1. **请求分析**：客户端（如 Claude Desktop 或 Cursor）将用户的需求发送给 LLM。
- 2. **工具筛选**：LLM 根据当前可用工具的描述信息（以文本形式传递，供模型理解），判断哪些工具可能解决问题。
- 3. **执行工具**：客户端通过 MCP Server 调用对应工具。
- 4. **返回结果**：工具运行后，将结果反馈给 LLM。

5. **生成回答**：LLM 将工具结果与上下文整合，生成最终的自然语言回复。

这样的机制，确保工具调用是**按需发生**，不会无效执行，也不会因为模型的幻觉而误用工具。

(5) 工具执行与结果反馈机制

- **无需工具**：模型直接用自然语言响应用户问题。
- **需要工具**：模型返回结构化的 JSON 格式调用指令。
- **客户端执行**：根据 JSON 调用对应工具并获取结果。
- **再次总结**：将工具结果与原上下文一起再传回模型，生成高质量回答。
- **异常处理**：对于无效或错误的调用请求，客户端会跳过执行，避免无用开销或错误响应。

这种两阶段机制（先决策，再总结）让 MCP 在功能扩展方面更加稳健和安全。

(6) MCP 的核心优势

1. 生态扩展性

MCP 协议是开放的，服务商和开发者都能按协议开发工具。随着插件库的增长，AI 应用的可用能力会呈指数级提升。

2. 跨平台兼容性

无论是 Anthropic Claude、其他 LLM，还是未来的新模型，只要支持 MCP 协议，就能无缝互联。开发者不必为切换平台重写适配代码。

3. 高可靠性与安全性

标准化的数据与工具访问方式，减少了因定制代码造成的安全漏洞和调用错误。

4. 增强上下文管理能力

MCP 不仅关心工具调用，还能跨应用、跨服务保持状态与上下文，提高模型在长任务链路中的表现。

MCP 不只是一个技术协议，它代表了 AI Agent 构建方式的演化：

- **从函数调用的封闭式，走向协议驱动的开放式；**

• **从单一平台工具集成，走向跨生态的标准化交互。**

随着 MCP 生态的完善，一套标准化的工具接口将像 USB-C 一样，让各类 AI 模型在数据访问和任务执行上变得灵活、可扩展，并且具有更强的互操作性。这将极大降低 AI 应用的开发成本，提升可靠性，加速智能体技术的普及与落地。

3. MCP 面临的挑战

MCP（模型上下文协议，Model Context Protocol）作为连接 MCP Server 和 LLM 的桥梁，本质上是由描述 MCP Server/MCP Tool 信息的系统提示词以及两者之间协同机制共同构成，旨在解决“找接口”和“解析接口”的问题。然而，将 MCP 引入企业级生产环境，会在这两个核心环节面临显著挑战。

（1）系统提示词的挑战

1. 安全性风险：系统提示词是 MCP 的核心调度信息，如果被恶意篡改或污染，可能导致 LLM 获取错误的 MCP Server/MCP Tool 信息，甚至调用存在安全漏洞的服务，从而引发流程崩溃和重大安全事故。

2. 版本管理问题：MCP Server 或 MCP Tool 更新迭代后，其对应的系统提示词需要同步更新，因此需要建立完善的版本管理策略，以确保提示信息与服务状态一致。

3. 调试与实时生效机制：由于系统提示词缺乏统一标准，不同企业可能采用自定义模板，实际写作和调优过程需要频繁试错和快速调整，因此必须具备快速调试与实时生效的能力。

4. Token 成本与效率瓶颈：在存在大量 MCP Server/MCP Tool 的场景下，系统提示词会变得冗长，占用大量 Token，从而增加成本并降低推理效率。企业需要构建基于用户意图的智能筛选机制，精准圈定相关服务范围，减少提示词体积。

（2）MCP Client 与 MCP Server 协同的挑战

1. 企业级应用集成难度：当前 MCP Client（如 Cline、Claude、Cursor）多为基于 SSE 协议的 C/S 工具，存在有状态连接、不可恢复、单向消息等限制，难以与企业复杂架构灵活结合。

2. 现有业务的快速迁移与转换：开发 MCP Server 依赖特定语言

的 MCP SDK（如 Python、Java、TS、Kotlin、C#），对于 Go、PHP 等技术栈的企业而言，零代码迁移几乎不可能，重新构建工作量巨大。

3. 统一管理需求：企业可能同时使用自研、第三方以及通过转换工具生成的 MCP Server，因此亟需类似“MCP Hub”或“MCP 市场”的管理与分发平台，实现集中注册、检索和资源优化。

4. 身份认证与安全合规：在企业环境中，必须确保用户身份验证、访问权限控制和数据安全防护，这在 MCP 的跨服务协同模式下尤为复杂，需要额外的安全协议和防护机制。

综上，MCP 在企业级落地过程中不仅要解决提示词的安全、管理、调优与效率问题，还必须突破 Client/Server 协同、技术栈限制、资源集中管理及安全合规等瓶颈，才能真正成为稳健可靠的生产级协议体系。

4. MCP 工程化实践 — AI 网关 & MCP Registry

（1）背景与问题

在 AI Agent 技术快速发展的浪潮中，MCP（Model Context Protocol）正在成为一种标准化的能力扩展范式。它将各种外部工具与模型上下文无缝对接，使 AI Agent 能够调用外部能力。然而，当前 MCP 生态中却存在两个现实问题：

1. 技能获取路径分散

每个 MCP 服务都有自己的 Endpoint。如果 AI Agent 需要对接 10 个 MCP 服务，就必须配置 10 个不同的 Endpoint。这不仅增加了 AI Agent “学习技能”的技术成本，还提高了维护成本。

2. 技能来源碎片化

MCP 服务的来源非常多样：

- 由服务商自行发布在平台上的 MCP 服务
- 第三方 MCP 服务聚合市场
- 企业内部私有 MCP 服务市场

不仅来源复杂，而且各 MCP 服务之间的管理方式（尤其鉴权认证方式）并不统一，这进一步提升了 AI Agent 获取和更新技能的难度。

（2）Anthropic MCP Registry 的诞生

为了统一 MCP 服务的发现与管理，Anthropic 推出了 MCP Registry，其官方定义是：

The MCP Registry service provides a centralized repository for MCP server entries. It allows discovery and management of various MCP implementations with their associated metadata, configurations, and capabilities.

抽象下来，MCP Registry 的两个核心能力是：

1. 集中式管控的 MCP 服务仓库

2. 动态发现和管理仓库内的 MCP 服务

官方特性包括：

- RESTful API（增、删、改、查 MCP 服务条目）
- 健康检查接口
- 支持多环境配置
- 优雅的关闭处理机制
- MongoDB 或内存数据库支持
- API 文档与分页查询能力

简单来说，Anthropic 希望通过 MCP Registry，让 AI Agent 像查询游戏技能面板一样，在一个统一中心发现和接入所有 MCP 服务。

（3）企业级落地的挑战

Anthropic MCP Registry 的设计更偏向于通用性和研发便利性，但在企业级落地时，仍存在几个需要增强的地方：

- 统一安全管控能力不足
- 服务性能与稳定性缺乏企业级验证
- MCP 服务能力的按需聚合与拆分支持有限

- 企业内部与外部 MCP 服务的整合门槛较高

因此，将 MCP Registry 能力与企业已有的成熟微服务治理框架结合，是落地的关键。

（4）MSE Nacos 3.0 — 企业级 MCP Registry

阿里云 MSE Nacos 经过多年生态打磨，已在集团内外广泛验证其注册中心与配置中心的性能与稳定性。在 MCP 场景中，MSE Nacos 3.0 版本可直接承担 **企业级 MCP Registry** 的角色，具备以下优势：

1. 统一安全管理所有 MCP 服务

- 无论是由传统服务转换而来的 MCP 服务，还是原生 MCP 服务，都可集中在一个 MCP Registry 中闭环管理。
- MCP 工具的 Prompt 支持版本管理与加密管理，保证能力迭代的安全性。
- 企业级高性能、高稳定性，在社区与集团内均已验证。

2. 与 AI 网关协同增强管控与扩展能力

- 可配置 MCP 服务策略或通过插件扩展业务定制策略（限流、降级、鉴权等）。
- 统一支持 SSE 与 Streamable HTTP 两种协议，降低 AI Agent 与 MCP 服务的开发成本。
- 在网关层可按工具粒度重新组装 MCP 服务，将多个不同来源的 MCP 工具聚合成一个统一的 Endpoint，方便 AI Agent 以最低配置成本获取跨领域技能。

这种聚合方式非常灵活，例如：

- 将高德地图的 MCP 服务与企业内部 HTTP 服务转换的 MCP 服务整合成一个统一 MCP 服务；
- 按业务领域构建 MCP 服务集（类似前后端分离中的 BFF 模型）。

最终，AI Agent 不再需要维护多个分散的 Endpoint，只需要一个入口，就可以访问授权范围内的所有工具——这就是一个可动态扩展的 **AI Agent 技能池系统**。

MCP Registry 解决了 AI Agent “技能发现”与“技能管理”的统一入口问题，而 AI 网关与 MSE Nacos 的结合，则在企业级落地中解决了 **性能、稳定性、安全管控**、协议支持以及能力聚合 等关键问题。

我们可以把这套架构理解为：

- **MCP Registry** = 中央技能仓库
- **AI 网关** = 技能编排与增强平台
- **MSE Nacos 3.0** = 企业级技能仓库管理器

这种组合让 AI Agent 拥有像游戏角色一样的技能面板，但背后是一个可持续扩展、可统一管理的能力治理体系。对于希望在生产环境稳定运行 AI Agent 应用的企业来说，这套组合非常值得实践。

（四）模型知识与私域知识的融合之路

近年来，大型语言模型（LLM）在自然语言处理领域取得突破性进展。以通义千问（Qwen）为代表的开源大模型，展现出强大的语义理解、逻辑推理与内容生成能力，为智能客服、知识管理、辅助决策等场景带来全新可能。

然而，政企客户在实际应用中普遍面临三大核心挑战：

- **知识滞后性**：模型训练数据截止于过去某一时点，无法感知企业持续演进的制度、流程、产品信息；
- **知识封闭性**：企业高价值私域知识（如合同模板、客户案例、运维手册）未纳入训练集，模型对此“一无所知”；
- **生成不可控**：在缺乏约束下，模型易产生“幻觉”（Hallucination）——即生成看似合理但无事实依据的内容，这在金融、政务、医疗等强监管场景中不可接受。

传统解决方案如全参数微调（Full Fine-tuning）或提示工程（Prompt Engineering），或成本高昂、或效果有限，难以满足企业对准确性、可解释性、合规性的综合要求。

为此，**检索增强生成（Retrieval-Augmented Generation, RAG）**成为当前最主流、最稳健的技术路径。RAG 通过外部知识检索机制，动态为大模型注入上下文，使生成结果建立在可验证的事实基础上，实现“通用能力 + 专属知识”的有机融合。

本文将系统阐述 RAG 在架构设计、索引构建优化、数据质量与系统评估、挑战与未来演进五个维度的深度工程实践，为构建可信、可控、可扩展的智能知识服务体系提供参考。

1. RAG 架构设计

RAG 的核心在于将检索机制与生成模型解耦，实现“知识动态注入、响应精准可控”。本方案依托阿里云三大核心产品构建标准化架构：

1. 检索层：向量检索服务（Vector Search）

作为系统的“大脑记忆库”，我们采用 阿里云向量检索服务（Vector Search），对私域知识文本进行语义向量化并建立高效索引。该服务支持千万级向量的毫秒级近邻检索，具备高并发、低延迟、自动扩缩容能力，特别适用于金融、政务等对可靠性要求极高的场景。

其底层采用 HNSW 算法，兼顾召回率与速度，并与 VPC、安全组、RAM 权限体系深度集成，确保数据不出域、访问受控。

2. 嵌入层：通义 text-embedding-v1 模型

知识的“表达力”决定了检索的质量。我们选用 通义 text-embedding-v1 模型，专为中文长文本优化，在专业术语、行业表达上的语义表征能力显著优于通用模型，大幅提升召回准确率。

该模型输出 1024 维稠密向量，能有效捕捉句子间的深层语义关联，尤其擅长处理制度文件、技术文档等结构化程度较低的文本。

3. 生成层：通义千问（Qwen）系列大模型

最终的答案由 通义千问（Qwen）系列大模型（推荐 Qwen-72B 或 Qwen-32B）生成。输入为“用户问题 + 检索结果上下文”拼接的结构化 Prompt，模型在明确知识边界内生成回答，并强制输出来源引用（如：“根据《员工手册 V3》第 12 条……”），杜绝无依据编造。

这一架构实现了“无需微调、即插即用”，避免了传统微调方式对标注数据的依赖与模型漂移风险，显著降低部署成本与维护复杂度。

2. RAG 架构构建

在大模型时代，一个普遍存在的认知误区是：只要模型足够强大，就能自动解决一切问题。然而，在企业级 RAG 应用中，我们反复验证了一个朴素却至关重要的真理——系统的效能不取决于模型本身有多聪明，而取决于它所依赖的知识是否准确、相关、完整且可追溯。主观体验固然重要，但只有建立在多维度、可量化的评估体系之上，RAG 才能从“玩具”蜕变为“生产工具”。

为此，我们设计了一套严谨的数据质量评估体系，涵盖五大核心维度：

1. 检索相关性：Recall@5 —— “找对东西”的第一道关卡

我们以 Recall@5 为核心指标，衡量前五个检索结果中语义匹配内容的比例。这不仅是对向量索引质量的检验，更是对切片策略、嵌入模型与知识库构建完整性的综合考验。若长期低于 85%，往往意味着知识库存在结构性缺陷——或切片过粗遗漏关键信息，或文档更新滞后导致语义漂移。

2. 回答准确性：Factuality Score —— “说对事情”的核心保障

我们引入 Factuality Score，通过人工标注与规则引擎交叉验证，判断生成内容是否严格源于检索结果，而非模型幻觉。当该指标持续低于 90%，通常暴露了 Prompt 设计过于宽泛，或检索结果未能精准对齐用户意图。

3. 可追溯性：引用准确率 —— 企业合规的生命线

任何输出都必须明确标注来源——精确至文档名称与段落位置。我们强制模型采用“根据《员工手册 V3》第 12 条”等结构化表达，并在日志中完整记录检索上下文，确保每一句回答皆可回溯、可复现、可追责。在金融、政务等强监管场景，这不仅是技术需求，更是责任边界。

4. 完整性：覆盖意图数 —— 应对复杂场景的能力

真实问题往往包含多个子需求，如“报销差旅需哪些材料？流程如何？有无额度？”系统不仅需覆盖全部子项，还需逻辑自洽、无冗余整合。我们通过构建“多意图测试集”，逐项评估其信息覆盖与结构化表达能力，检验生成模块的协同与提炼水平。

5. 响应效率：端到端延迟 —— 用户体验与成本的交汇点

我们关注的是“端到端延迟”——从用户输入问题到系统输出完整答案的总耗时。在高并发场景下，若平均延迟超过 1.5 秒，用户耐心将急剧下降；而在大规模部署中，每一次调用 Qwen API 都意味着计算成本的增加。因此，我们不仅优化向量检索性能、压缩 Prompt 长度，还探索高频问答缓存机制，对常见问题预生成答案，显著降低重复调用带来的资源浪费。

这套评估体系不是静态清单，而是一个持续演进的闭环。我们定期运行测试集，对比不同版本的性能差异，分析错误案例的根本原因，并将改进措施反哺至数据治理、Prompt 工程、索引优化等上游环节，确保系统不断进化，从“能用”走向“好用”，从“可用”走向“可信”。

3. 数据质量与系统评估

只有建立在多维度、可量化的评估体系之上，RAG 才能从“玩具”蜕变为“生产工具”。为此，我们设计了如下数据质量评估体系表：

维度	评估指标	实施方法示例
检索相关性	Recall@K (K=5)	统计前 5 个检索结果中与问题语义匹配的比例
回答准确性	Factuality Score	人工标注或基于规则判断回答是否源于检索内容
可追溯性	引用准确率	检查输出是否明确标注来源文档与段落
完整性	覆盖意图数	对多意图问题评估是否全部覆盖
响应效率	端到端延迟	测量从提问到生成完成的总耗时

表 1 数据质量评估体系表

首先，检索的相关性是系统能否“找对东西”的第一道关卡。我们以 Recall@5 为核心指标，衡量前五个检索结果中语义匹配内容的比例。这不仅是对向量索引质量的检验，更是对切片策略、嵌入模型与知识库构建完整性的综合考验。若长期低于 85%，往往意味着知识库存在结构性缺陷——或切片过粗遗漏关键信息，或文档更新滞后导致语义漂移。

其次，回答的准确性决定了系统能否“说对事情”。我们引入 Factuality Score，通过人工标注与规则引擎交叉验证，判断生成内容是否严格源于检索结果，而非模型幻觉。当该指标持续低于 90%，通常暴露了 Prompt 设计过于宽泛，或检索结果未能精准对齐用户意图。

第三，可追溯性是企业合规的生命线。任何输出都必须明确标注来源——精确至文档名称与段落位置。我们强制模型采用“根据《员工手册 V3》第 12 条”等结构化表达，并在日志中完整记录检索上下文，确保每一句回答皆可回溯、可复现、可追责。在金融、政务等强监管场景，这不仅是技术需求，更是责任边界。

第四，完整性决定了系统能否应对复合意图。真实问题往往包含多个子需求，如“报销差旅需哪些材料？流程如何？有无额度？”系统不仅需覆盖全部子项，还需逻辑自洽、无冗余整合。我们通过构建“多意图测试集”，评估其信息覆盖与结构化表达能力，检验生成模块的协同与提炼水平。

最后，响应效率关乎体验与成本。端到端延迟若超过 1.5 秒，用户耐心即被消耗；而高频调用 Qwen API 更带来显著资源开销。为此，我们优化向量检索性能、压缩 Prompt 长度，并对高频问题启用缓存机制，预生成标准答案，在保障质量的同时，实现成本与效率的平衡。

4. 挑战与未来演进方向

尽管检索增强生成（RAG）技术已成功将大模型的通用能力与企业私域知识紧密融合，成为当前云上智能应用的主流范式，但我们仍需清醒地认识到，它尚未抵达终点，而只是迈出了通往真正“认知型系统”的第一步。

当前的 RAG 实践，往往依赖于静态知识库的向量化与简单的上下文注入。当文档更新滞后、语义切片不当、或检索结果与问题意图存在微妙偏差时，系统仍可能给出看似合理、实则偏离事实的回答。更深层的挑战在于，多数部署仍停留在“问答模式”——用户问，系统答，仅此而已。它能告诉你“怎么做”，却无法主动判断“该不该做”；它能引用条款，却不能推理规则间的逻辑关联。

这正是我们面临的下一个关口：如何让知识不再被“检索”，而是被“理解”？如何让系统不仅知道内容，还能理解上下文、感知变化、甚至预见问题？

未来的演进，将围绕三个方向展开。首先是知识的动态性。企业文档不是静态档案，而是持续生长的生命体。一个制度修订、一份合同更新、一次流程调整，都应能自动触发知识库的同步重建。这不再依赖人工干预，而是通过事件驱动的流水线，实现从文档变更到语义更新的无缝衔接，让知识库始终保持鲜活。

其次是**多模态认知的融合**。企业的知识不仅存在于文字中，也藏于流程图、审批表、设备图纸和扫描文档之中。未来的系统将不再只“读文字”，而是能“看图识意”。通义视觉语言模型的引入，使系统能够理解一张运维图中的拓扑关系，或一份财务报表中的数据趋势，并将其与文本知识对齐，从而构建出更完整、更立体的认知图谱。

第三，也是最具颠覆性的方向，是**从被动响应走向主动推理**。当系统不仅能回答“如何报销差旅”，还能根据员工的职级、历史记录与最新政策，判断“本次申请是否合规”；当它不仅能检索“故障处理手册”，还能结合监控数据、变更记录与历史案例，推断“本次异常最可能的根源”——RAG 就不再是工具，而成为一种智能伙伴。

这一步，需要引入推理引擎与智能体架构。通过将 RAG 与阿里云知识引擎、通义星尘等能力结合，系统将具备规划、判断与执行的能力。它不再只是回答“你该怎么做”，而是可以主动“帮你做”——自动发起工单、通知责任人、甚至预判下一次可能的故障。

5. 小结

RAG 不是技术的终点，而是企业智能化转型的起点。在专有云环境中，如何构建一个自主演进、安全可控、深度理解企业语境的智能知识中枢将会是政企客户追求的长远目标。因此，我们需要通过阿里云全栈 AI 能力，让政企客户可快速构建专属的智能知识服务体系，让沉睡的知识活起来，让每一次对话都有据可依、有源可溯、有责可追。

（五）企业级 AI 原生服务能力建设

随着大模型从实验室走向生产环境，企业对 AI 服务的诉求已不再停留在“能用”，而转向“敢用、稳用、善用”。一个部署在云端的通义千问 API，若缺乏统一入口、缺乏运行洞察、缺乏质量校验、缺乏安全兜底，即便模型再强大，也难以成为企业核心业务的可靠支柱。

真正的企业级 AI 原生服务能力，不是简单地调用一个模型接口，而是构建一套端到端、可管理、可审计、可信任的智能服务基础设施。它必须像数据库、消息队列一样，成为企业 IT 架构中的“标准组件”——稳定、透明、可运维、可治理。

在这一背景下，阿里云以“AI as a Service”的理念，系统性构建了覆盖 AI 服务全生命周期的原生能力体系，通过 AI 网关、AI 可观测、AI 评估、AI 安全 四大支柱，打造可落地、可扩展、可信赖的智能服务底座。

1. AI 网关：统一入口，实现智能服务的标准化治理

在企业内部，AI 模型的调用往往呈现出“多模型、多团队、多场景”的碎片化格局。一个部门调用 Qwen-7B 做合同解析，另一个团队使用 Qwen-VL 解析图纸，第三方供应商接入通义灵码做代码生成，每个系统各自对接、各自鉴权、各自限流。这种模式不仅运维成本高，更埋下严重的安全与一致性隐患。

AI 网关，正是为解决这一“孤岛式 AI 调用”而生。

作为企业 AI 服务的统一入口，阿里云 AI 网关提供集中式接入、统一鉴权、流量控制、协议转换与日志聚合能力。它不替代模型本身，而是成为所有 AI 调用的“守门人”。

通过 AI 网关，企业可将通义千问、通义万相、通义听悟、通义星尘等不同能力的模型服务，统一注册、统一暴露、统一管理。开发者无需关心后端是哪个模型、部署在何处，只需向网关发起标准化请求，即可获得一致的响应格式与服务质量。

更重要的是，网关实现了细粒度的访问控制。不同部门、不同角色、不同业务系统，可被授予不同的调用权限。例如，财务系统仅允许调用合同解析模型，且每日限额 1000 次；研发团队可访问代码生成模型，但禁止调用客户画像接口。这种策略化管控，让 AI 服务从“自由使用”走向“受控使用”。

同时，网关内置智能限流与熔断机制，在模型负载过高或响应异常时自动降级，保障核心业务不被 AI 服务的波动所拖累。它不再是“可有可无的辅助工具”，而是企业 AI 服务的核心调度中枢。

2. AI 可观测：让智能服务从“黑盒”走向“透明”

当 AI 成为业务流程中的关键环节，我们不能再依赖“它今天回答得不错”这样的主观判断。一旦模型输出错误，导致合同条款误读、客户投诉激增、审批流程阻塞，企业必须能快速定位问题根源——是模型幻觉？是输入数据异常？是提示词设计不当？还是下游系统传参错误？

这就需要一套**面向 AI 服务的可观测体系**。

传统监控关注的是延迟、流量、错误率，而 AI 可观测关注的是**语义质量、逻辑一致性、响应可信度**。阿里云 AI 可观测能力，从三个维度构建了完整的洞察闭环：

首先是**输入输出全链路追踪**。每一次调用的原始请求、模型返回内容、检索上下文、引用来源，均被完整记录，形成可追溯的“AI 服务日志”。当用户反馈“系统说可以报销，但制度不允许”，我们能立即回溯：模型引用的是哪份文档？是哪一段内容被误读？是哪一次版本更新导致了语义偏移？

其次是**语义级指标监控**。我们不再只看“响应时间是否达标”，而是关注“回答是否包含幻觉”“是否遗漏关键约束条件”“是否混淆了相似概念”。这些指标由模型自身输出结构化判断，或通过轻量规则引擎自动计算，形成可量化的健康度看板。

最后是**异常模式识别**。通过分析历史调用数据，系统可自动发现“高频错误模式”——例如，当用户提问中出现“增值税发票”时，模型有 37% 的概率错误引用 2023 年旧政策。这种模式一旦被识别，即可触发告警、建议更新知识库，甚至自动回滚模型版本。

AI 可观测不是事后复盘的工具，而是事前预防的哨兵。它让 AI 服务从一个“黑盒”变成了一个“透明系统”，让运维人员能像管理数据库一样管理模型，让业务方能像信任 SQL 查询一样信任 AI 回答。

3. AI 评估：用标准衡量智能，用数据驱动进化

AI 的价值，最终体现在它是否解决了真实问题。但如何衡量“解决得好不好”？是用户满意度？是人工评分？还是业务转化率？

这些指标固然重要，但它们往往是结果导向、滞后性强、难以复用。要实现 AI 服务的持续优化，必须建立一套**可量化、可对比、可自动化**的评估体系。

阿里云 AI 评估能力，正是为此而设计。它不是一次性的测试报告，而是一个**持续运行的评估流水线**，贯穿模型上线前、上线中、上线后全生命周期。

在模型上线前，我们通过构建**多意图测试集**，验证其在复合查询、边界条件、对抗样本下的表现。例如：“报销差旅需哪些材料？流程如何？有无额度？”这类问题，不是靠单次测试能覆盖的，而是需要系统性设计数十种组合场景。

在模型上线后，评估系统会自动抽取真实用户提问，结合人工标注与规则引擎，计算**Factuality Score(事实准确性)、Relevance Score(相关性)、Completeness Score(完整性)**等核心指标，并与历史版本进行对比。当某次更新导致“引用准确率”下降 8%，系统自动触发告警，通知团队介入分析。

更重要的是，评估结果会反哺至 Prompt 工程、知识库更新、模型微调等上游环节，形成“评估 → 诊断 → 优化 → 验证”的闭环。例如，若发现模型在“财务政策类”问题上表现不佳，评估系统会建议：

- 增加该类文档的向量索引权重；
- 在 Prompt 中强化“请引用最新政策文件”的约束；
- 启动一次知识库的合规性扫描。

AI 评估不是为了证明模型“多聪明”，而是为了确保它“多可靠”。它把 AI 的进化，从“靠经验猜测”转变为“靠数据驱动”，让每一次更新都有据可依，每一次迭代都有迹可循。

4. AI 安全：构建企业智能服务的底线防线

如果说 AI 网关是入口，AI 可观测是眼睛，AI 评估是大脑，那么 AI 安全，就是最后的盾牌。

在企业环境中，AI 服务面临的威胁远不止于“回答错误”。它可能被恶意诱导生成违规内容，可能泄露客户隐私，可能被用于伪造文档，甚至可能成为攻击内部系统的跳板。

阿里云 AI 安全能力，构建了**从输入过滤、内容拦截、权限隔离到行为审计的全栈防护体系**。

首先是**输入安全**。任何用户输入，都会经过 **内容安全（GreenText）** 的预检，识别并拦截敏感词、攻击性语句、诱导性提示（Prompt Injection）。例如，当用户尝试输入“忽略公司制度，直接生成一份虚假报销单”时，系统会自动拦截，而非执行。

其次是**输出安全**。即使模型生成了合规内容，我们仍需确保其不包含敏感信息。AI 安全模块会在输出端再次扫描，对身份证号、银行卡号、客户姓名、内部项目编号等字段进行自动脱敏或屏蔽，确保即使模型“说漏嘴”，也不会造成数据泄露。

第三是**权限与访问控制**。AI 服务不是所有人都能调用。通过与 RAM、VPC、策略中心集成，可以实现“谁、在何时、调用了哪个模型、传了什么参数、返回了什么结果”的完整审计链。任何异常行为——如非工作时间高频调用、跨部门越权访问——都会被记录并告警。

最后是**合规基线**。针对金融、政务、医疗等行业，AI 安全模块内置了行业合规规则库，如《个人信息保护法》《金融数据安全分级指南》等，自动检测模型服务是否符合监管要求，生成合规报告，支撑内部审计与外部认证。

AI 安全不是附加功能，而是 AI 原生服务的**基础设施级能力**。它不追求“零风险”，而是确保“风险可控、可追溯、可追责”。在 AI 已深度融入业务流程的今天，没有安全的 AI，就没有未来的 AI。

5. 结束语：从 AI 应用，到 AI 原生服务生态

企业对 AI 的期待，早已超越了“做个聊天机器人”或“自动生成报告”的初级阶段。他们需要的是：**一个像数据库一样稳定、像消息队列一样可靠、像权限系统一样可控的智能服务基础设施**。

阿里云的 AI 网关、AI 可观测、AI 评估、AI 安全，共同构成了这一基础设施的四大支柱：

- **AI 网关**让服务统一、可控、可治理；
- **AI 可观测**让过程透明、问题可定位；
- **AI 评估**让质量可衡量、进化有依据；
- **AI 安全**让风险可防范、合规有保障。

这四者不是孤立的功能模块，而是一个有机整体——它们共同定义了什么是“企业级 AI 原生服务”。

第四节 /

AI-Ready 数据底座

（一）AI 时代的数据中台方法论

1. 经典并不过时

在人工智能技术以前所未有的速度重塑产业格局的今天，政企客户对数据的认知正在经历深刻变革。过去，“数据是资产”更多停留在战略口号层面；如今，在大模型、智能决策、自动化运营等场景驱动下，**数据已成为企业核心生产要素和智能化引擎的燃料**。然而，面对 AI 对数据提出的高维、实时、多模态、可解释等新要求，许多组织开始质疑：传统数据治理体系是否已“过时”？是否需要彻底抛弃既有架构，从零构建“AI 原生”数据底座？

答案是否定的。

阿里巴巴集团历经十余年实践沉淀形成的“**One Model、One Data、One Service**”数据中台方法论，不仅没有被 AI 浪潮淘汰，反而因其对**统一性、一致性、服务化**的根本坚持，成为构建高质量 AI 数据底座的坚实基础。这一方法论的本质，是对数据资产进行系统性治理与价值释放的工程哲学，其内核与 AI 时代的数据需求高度契合。

本章将系统阐述：为何经典方法论依然有效？它如何指导 AI 数据底座建设？又需在哪些维度进行个性化调整与智能化演进？

（1）One Model：统一建模——AI 理解世界的语义基石

“One Model”强调以业务对象为中心，通过统一的维度建模（如星型/雪花模型）和实体 - 关系（ER）设计，构建全域一致的逻辑数据模型。其核心目标是**消除语义歧义，实现跨域数据融合**。

在 AI 时代，这一原则的重要性不降反升：

- **特征工程依赖统一实体定义**

AI 模型（无论是传统机器学习还是深度学习）的输入特征必须基于清晰、一致的业务对象。例如，若“企业”在 A 系统以统一社会信用代码标识，在 B 系统却使用工商注册号且未做映射，则无法有效聚合其交易、报关、纳税等行为轨迹，导致特征稀疏、噪声增加，甚至引入系统性偏差。One Model 通过主数据管理（MDM）确保关键实体 ID 唯一、属性标准统一，为高质量特征构建奠定基础。

- **知识图谱构建需语义对齐**

当前主流 AI 应用（如智能风控、供应链优化）越来越多依赖知识图谱进行关系推理。而图谱构建的前提是实体、属性、关系具备标准化语义。One Model 提供的统一业务术语（Business Glossary）和逻辑模型，正是实现跨系统语义对齐的关键桥梁。

- **大模型微调与 RAG 依赖结构化上下文**

即使采用大语言模型（LLM），其检索增强生成（RAG）能力也高度依赖结构化、标准化的上下文数据源。若底层数据模型混乱，检索结果将缺乏准确性与一致性，直接影响生成质量。One Model 确保 RAG 所依赖的数据库具备清晰的 schema 和语义，提升 AI 输出的可信度。

因此，AI 并非否定建模，而是对建模提出了更高要求：

- 从“报表导向”转向“智能导向”
- 从“静态维度”扩展到“动态行为 + 关系网络”
- 从“字段级描述”深化到“向量级嵌入”

（2）One Data：数据一致性——AI 可信性的生命线

“One Data”主张“一数一源、一处加工、全局共享”，确保关键数据在全组织范围内保持唯一、准确、及时。这一原则在 AI 场景下具有决定性意义：

- **训练数据偏差源于源头不一致**

监督学习依赖高质量标注数据。若同一业务概念在不同系统中标准不一（如“高风险企业”在风控系统定义为信用评级 D 级，在监管系统却定义为涉诉次数 ≥ 3 次），则模型将学习到矛盾信号，导致泛化能力下降。One Data 通过明确权威数据源，确保训练标签的一致性与权威性。

◦ 在线推理依赖实时一致性

AI 服务(如智能审批、异常检测)需实时访问最新、最准的数据。若数据源分散、更新异步(如企业状态变更 T+1 同步)，将导致“线上表现劣于离线评估”的经典问题。One Data 通过统一数据加工链路(如 DataWorks 调度)，保障线上线下数据一致性(Feature Consistency)。

◦ 数据血缘支撑可解释性与合规

全球 AI 监管趋严，要求模型决策可追溯、可审计。只有建立端到端的数据血缘，才能回答：“该预测基于哪些原始数据？是否包含敏感信息？” One Data 天然强调数据血缘管理，为 AI 合规提供技术保障。

（3）One Service：数据服务能力——AI 落地效率的加速器

“One Service”主张将数据封装为标准化、可复用的服务接口(API、标签、指标、特征)，供上层应用按需调用。在 AI 工程化实践中，这一理念直接对应特征平台 和 模型即服务 架构：

◦ 特征复用避免重复计算

同一“用户活跃度”特征，既可用于推荐系统，也可用于流失预警模型。通过统一特征服务(如阿里云 PAI-FeatureStore)，可确保线上线下特征一致性，避免“训练 – 推理特征漂移”问题。

◦ 标签服务支撑监督学习

高质量的业务标签(如“优质客户”、“异常交易”)是监督学习的前提。One Service 要求标签由权威业务方定义、统一管理、版本可控，并通过 API 或消息队列实时推送，支撑在线学习。

◦ 数据服务编排加速 AI 实验

通过低代码方式组合数据服务(如 Quick BI + Data Service)，业务人员可自助构建训练数据集，缩短 AI 从想法到验证的周期，实现“人人都是数据科学家”。

（4）经典方法论的智能化演进方向

尽管“One Model、One Data、One Service” 仍具强大指导力，但 AI 时代的数据治理确需在以下维度进行拓展与升级：

维度	传统数据治理	AI 时代新要求	演进方向
数据粒度	表 / 字段级	向量 / 嵌入级	引入向量数据库，支持语义相似性检索
更新频率	T+1 批处理	实时流处理	构建 Lambda/Kappa 架构，支持流批一体
治理对象	结构化数据	多模态数据(文本、图像、日志)	扩展元数据管理至非结构化数据
质量维度	准确性、完整性	公平性、鲁棒性、隐私性	增加 AI 伦理与合规性质量规则
服务模式	静态 API	动态特征服务	支持实时特征计算与上下文感知

表 1 AI 时代数据治理演进方向

因此，面向 AI 的数据治理，不是推倒重来，而是在继承经典方法论的基础上，结合 AI 数据底座的需求，进行个性化调整与智能化探索。下一章将系统阐述这一建设路径。

2. AI 数据中台建设需求及理论实现路径

面向 AI 的数据治理，核心目标是构建一个高质量、高可用、高智能的数据底座，支撑从数据准备、特征工程、模型训练到在线推理的全生命周期。本章基于阿里云飞天企业版大数据产品体系(包括 DataWorks、Dataphin、DataQ、PAI、Quick BI 等)，提出一套适用于政企客户的通用 AI 数据底座建设框架，涵盖五大核心任务：资源整合、标签体系、质量管理、绩效评价、能力建设。

（1）整合现有数据资源：构建 AI-ready 数据资产目录

核心目标：将分散、异构、低质的数据资源，转化为结构清晰、语义明确、可溯源、可标注的 AI 就绪(AI-ready)资产。

（1.1）需求概述

政企客户普遍面临“数据孤岛林立、元数据缺失、标准不一”等问题，导致 AI 项目常因“找不到数、看不懂数、不敢用数”而失败。为此，需系统性开展以下工作：

1. 编制全域数据目录

依据国家《政务数据资源目录编制指南》、行业数据标准（如金融 IC 卡规范、医疗 HL7）或企业数据治理规范，构建覆盖全业务域的数据资源目录，实现“知数、识数、管数”。

2. 提升数据池质量

对存量数据表（尤其是多源冗余、历史遗留表）进行分级分类，补充字段级注释，确保数据准确性、完整性、一致性，提升 AI 可用性。

3. 开展“一数一源”治理

明确关键数据项(如身份证号、统一社会信用代码、商品编码)的唯一权威来源，消除同名异义、同义异名问题，保障 AI 训练数据的一致性。

4. 强化数据标签管理

在技术类、管理类标签基础上，围绕核心业务对象（如人、企、物、事）构建业务类标签体系，为 AI 提供高质量监督信号与特征输入。

（1.2）实现路径

（一）构建分层数据资源目录

- 采用“主题域分组 → 主题域 → 业务对象 → 逻辑数据实体 → 属性”五层架构，对全量库表进行盘点与编目。
 - 示例：在金融行业，“客户”可作为业务对象，其逻辑实体包括“个人客户”、“对公客户”，属性包括“姓名”、“证件类型”、“风险等级”等。
- 基于使用频率、更新时效、业务重要性等维度，对存量表进行分级(常用/一般/停用)，形成《数据资源使用评估报告》，指导治理优先级。
- 利用 DataWorks 数据地图能力，自动采集技术元数据（表结构、字段类型），人工补充业务元数据（中文名、业务含义），确保表中文名覆盖率≥ 95%，字段注释率≥ 90%。

（二）推进“一数一源”治理

- 针对高频使用的关键数据项，识别多源冲突、来源不明等问题。
- 明确每个数据项的权威系统、更新机制、标准规范，形成《数据源目录》。

- 通过 Dataphin 数据标准模块，固化数据定义（如“企业注册资本”单位为“万元”，格式为数值型），防止“二次污染”。
- 目标：常用库表数据来源可 95% 溯源，其余在用表溯源率达 80%。

（三）交付物与标准

- 《企业数据目录编制模板》与术语字典
- 全域数据资源目录、数据源目录（支持数据门户展示）
- 字段级注释完整率达标（≥ 90%）
- 《数据资源使用评估报告》《一数一源检查报告》

（2）建设数据分类分级标签体系：为 AI 注入业务语义

核心目标：通过多维度标签体系，将原始数据转化为具备业务含义、安全属性、应用场景标识的 AI 特征原料。

（2.1）需求概述

标签是连接原始数据与 AI 应用的桥梁。高质量标签体系应具备：

- 结构合理：分层分类清晰
- 标准统一：定义无歧义
- 责任明确：主管单位可追溯
- 可维护性强：支持动态更新

标签体系通常分为三层：

标签类型	描述	示例	AI 价值
技术类标签	描述数据物理属性	数据类型、格式、长度、编码规则	支撑特征选择、数据清洗
管理类标签	描述数据治理属性	业务域、密级、共享范围、责任部门	支撑权限控制、合规审计
业务类标签	描述数据业务含义	“高净值客户”、“高风险货物”、“重点监管企业”	直接作为监督信号或特征输入

表 2 三层标签体系说明

(2.2) 实现路径

(一) 设计三层标签体系

- 技术 / 管理标签由数据治理团队统一定义，通过 DataQ 打标工具自动 + 人工标注至字段级。
- 业务标签由业务专家主导，围绕核心对象（人、企、物、事）设计标签类目与打标规则。

(二) 开展业务标签建设

- 选取重点 AI 场景（如智能风控、精准营销、预测性维护），联合业务部门定义标签逻辑。
 - 示例：在风控场景，“高风险客户”可定义为“近 3 个月逾期 ≥ 2 次且负债率 >80%”。
- 利用 PAI-Studio 或 SQL 脚本实现标签计算，通过 A/B 测试验证标签有效性。
- 建立标签生命周期管理机制，支持版本迭代与效果回溯。

(三) 交付标准

- 技术 / 管理标签覆盖 80% 以上在用表
- 如业务标签不少于 5000 个，打标率 ≥ 80%
- 标签规则文档化，业务逻辑清晰可解释

(3) 深化数据质量管理：从“可用”到“可信”

核心目标：建立覆盖 AI 全链路的数据质量监控体系，确保训练数据可靠、推理数据一致。

(3.1) 需求概述

传统数据质量聚焦“完整性、准确性”，而 AI 对数据质量提出更高要求：

- **一致性：**线上线下特征一致
- **时效性：**支持实时推理
- **公平性：**避免歧视性偏差
- **鲁棒性：**抵抗对抗样本攻击

因此，需构建涵盖**完整性、规范性、准确性、唯一性、时效性、有效性、一致性、公平性**的八维质量指标体系。

(3.2) 实现路径

(一) 构建 AI 增强的质量规则库

- **技术规则：**空值率、格式合规、主外键约束等（由 DataWorks 质量中心自动检测）
- **业务规则：**如“企业注册资本 >0”“进出口日期 ≤ 当前日期”等（由业务专家定义）
- **AI 规则：**引入异常检测模型（如 Isolation Forest）自动发现分布偏移、概念漂移

(二) 实施分阶段质量提升

- **第一阶段：**完成常用表质量基线评估，配置核心规则
- **第二阶段：**开展全量表质量检测，输出首份评估报告
- **第三阶段：**推动源头整改，建立质量问题闭环机制

(三) 交付物

- 《数据质量评估标准》与指标集
- 数据质量规则库（含技术 + 业务 + AI 规则）
- 重点表全链路监控策略（每表 ≥ 7 条技术规则 +1 条业务规则）
- 季度《数据质量评估报告》

(4) 健全数据绩效评价机制：以评促治，以治促用

核心目标：将数据治理成效量化，纳入组织考核，形成持续改进闭环。

(4.1) 需求概述

数据治理若无考核，易流于形式。需建立科学的绩效评价机制，回答：

- 数据资产规模是否增长？
- 数据质量是否提升？

- 数据服务是否被复用？
- AI 应用是否产生业务价值？

(4.2) 实现路径

- 对照国家数据治理成熟度模型（如 DCMM）、行业最佳实践，制定企业级评价指标体系。
- 从四个维度开展评估：
 - a. 资产维度：目录覆盖率、标签数量、数据服务 API 数
 - b. 质量维度：规则命中率、问题修复时效
 - c. 服务维度：API 调用量、特征复用率
 - d. 价值维度：AI 模型上线数、业务收益（如风控拦截金额、营销转化率）
- 输出《数据治理绩效报告》，指导资源投入与策略调整。

(5) 引入外部数据治理咨询服务：加速能力建设

核心目标：通过“授人以渔”，培养企业自有数据治理与 AI 工程团队。

(5.1) 需求概述

- 政企客户普遍面临“有数据、缺人才、少方法”的困境。外部咨询的价值在于：
- 提供方法论与工具链
 - 赋能关键领域（如风控、人事、财务）
 - 培养内部专家团队

(5.2) 实现路径

- 分领域咨询：针对重点业务域（如供应链、客户服务、安全生产），提供数据架构与治理方案。
- 实操培训：开展数据目录编制、质量规则设计、标签打标等实训。

- 团队共建：协助培养两类核心人才：
 - 业务数据专员（懂业务 + 懂数据）：负责本领域数据需求与标签定义
 - 数据架构专家（懂技术 + 懂治理）：负责数据模型、标准、服务设计

交付标准：

- 培养建设具备一定规模的业务数据专员、数据架构专家梯度人才团队
- 输出《数据治理实践指导手册》
- 建立常态化咨询响应机制

3. 结语：从“数据中台”走向“AI 中台”

面向 AI 的数据治理，不是另起炉灶，而是在“One Model、One Data、One Service”经典框架下，注入**智能化、实时化、服务化**的新内涵。政企客户应立足现有数据资产，以 AI 应用场景为牵引，分步推进数据资源整合、标签体系构建、质量闭环管理与组织能力建设。

（二）浅谈高质量数据集建设

引言：从“结构化处理”到“多模态智能”的范式跃迁

在政企数字化转型的早期阶段，数据中台的核心使命是解决“数据孤岛”与“报表烟囱”问题，其技术重心在于**结构化数据的集成、清洗、建模与服务化**。阿里巴巴提出的“One Model、One Data、One Service”方法论，正是这一阶段的最佳实践总结——通过统一建模消除语义歧义，通过一数一源保障数据一致性，通过数据服务实现资产复用。

然而，随着大模型与多模态人工智能技术的爆发式发展，AI 应用对数据的需求发生了根本性转变：

- **从结构化到多模态**：AI 不再仅依赖关系型数据库中的字段，而是需要融合文本、图像、语音、视频、时序信号等多源异构数据；
- **从静态批处理到实时流处理**：在线推理要求数据具备毫秒级更新能力；
- **从“可用”到“可信、可解释、可合规”**：AI 监管趋严，数据需满足公平性、隐私性、版权合规等新要求。

在此背景下，**高质量数据集**（High-Quality Dataset）成为连接传统数据中台与 AI 智能应用的关键桥梁。它不仅是 AI 模型训练的“燃料”，更是企业构建差异化竞争力的“护城河”。正如中国信通院《人工智能高质量数据集建设指南》所指出：“在算法趋同、算力普惠的竞争环境中，高质量数据集正在构建难以复制的差异化壁垒。”

因此，政企客户面向 AI 的数据底座建设，必须实现从“**数据中台**”到“**AI 数据中台**”的升级，而高质量数据集建设正是这一升级的核心抓手。本章将系统阐述高质量数据集的定义、建设路径、安全管理、效果评估与运维运营体系，并结合阿里云飞天企业版大数据与 AI 产品能力，提供一套可落地的技术框架。

1. 高质量数据集的定义

（1）传统数据中台 vs. AI 数据中台：需求演进

维度	传统数据中台	AI 数据中台
核心目标	支撑 BI 报表、经营分析	支撑 AI 模型训练与推理
数据类型	以结构化数据为主	多模态数据（文本、图像、语音、视频、IoT）
数据粒度	表 / 字段级	向量 / 嵌入级、样本级
质量要求	完整性、准确性、一致性	+ 多样性、时效性、公平性、合规性
服务模式	静态 API、SQL 查询	动态特征服务、向量检索、RAG 上下文
治理重点	元数据、主数据、数据标准	数据标注、合成数据、偏见检测、版权合规

表 1 传统数据中台和 AI 数据中台需求演进对比表

由此可见，AI 数据中台并非对传统数据中台的否定，而是在其基础上的**能力扩展与范式升级**。高质量数据集正是这一升级的具体载体。

（2）高质量数据集的核心内涵

根据信通院《人工智能高质量数据集建设指南》与大数据研究院《高质量数据集实践指南（1.0）》，**高质量数据集**是指：“经过系统性采集、加工、标注与质检，可直接用于开发和训练人工智能模型，能有效提升模型性能、满足安全合规要求的数据集合。”

其核心特征可概括为“**三高**”：

- **高价值应用**：锚定产业刚需场景（如医疗辅助诊断、工业缺陷检测），解决业务痛点；
- **高知识密度**：蕴含跨领域专业知识（如医学影像 + 临床记录 + 药品说明书），支撑复杂推理；
- **高技术含量**：采用合成数据、智能标注、多模态对齐等先进技术，提升数据规模与质量。

(3) 高质量数据集的分类体系

为指导政企客户精准建设，高质量数据集可从三个维度进行分类：

(3.1) 按数据应用维度

- **通识类数据集**：覆盖通用知识（如百科、新闻、书籍），用于大模型预训练；
- **行业通用类数据集**：包含行业基础知识（如交通法规、金融术语），用于行业大模型微调；
- **行业专用类数据集**：聚焦企业私有场景（如海关报关单、电网设备日志），用于专属模型优化。

(3.2) 按模型训练阶段

- **预训练数据集**：海量、无标注、多领域，学习通用知识；
- **指令微调数据集**：高质量问答对，教会模型遵循人类指令；
- **价值观对齐数据集**：基于人类反馈（RLHF），确保输出符合伦理规范；
- **行业领域数据集**：用于 RAG 或微调，提升专业领域表现。

(3.3) 按数据模态维度

- **单模态**：文本、图像、音频、时序数据；
- **多模态**：图文对、视频 + 字幕、语音 + 文本转录；
- **特殊模态**：思维链（CoT）、具身智能轨迹、长视频序列。

2. 高质量数据集建设路径

高质量数据集建设是一项覆盖全生命周期的系统工程。参考信通院“三步走”战略与 CCSA “四大环节”框架，我们提出政企客户可落地的五阶段建设路径：

(1) 体系规划阶段：锚定场景，设计蓝图

核心任务：避免“为数据而数据”，确保数据集与业务目标深度绑定。

关键活动：

- **场景拆解**：明确 AI 应用场景（如智能风控、设备预测性维护），定义模型输入 / 输出、性能指标；
- **知识索引构建**：围绕业务对象（如“企业”“设备”“患者”），梳理核心知识节点与数据需求；
- **资源地图绘制**：盘点内部数据资产（数据库、日志、文档、音视频），标注类型、权属、质量状态；
- **标准体系搭建**：制定数据采集、标注、质检、安全等规范，确保工程可复制。

(2) 数据采集与治理阶段：夯实底座，提升质量

核心任务：将原始数据转化为 AI-ready 的高质量原料。

关键技术流程（参考海关总署实践）：

1. 数据收集

- 明确任务类型（CPT/SFT/RL）、数据量、样本长度；
- 整合内部业务系统、公开数据、合成数据等多源渠道；
- 优先采集高价值、低冗余数据，避免“数据沼泽”。

2. 噪声清洗（Noise Cleaning）

- 去除乱码、广告、敏感信息、重复内容；
- 过滤超长 / 超短、不完整、无效样本；
- 利用规则引擎 + AI 模型（如 Qwen）自动识别噪声。

3. 数据标准化（Data Standardization）

- 统一角色代词（客服 / 坐席→agent）、金额格式（汉字 / 数字→数值）、标点符号；
- 对齐业务术语（如“延期还款”与“过几天还款”合并）。

4. 安全合规过滤（Security Compliance Filter）

- 识别并脱敏 PII（身份证、手机号、银行卡号）；
- 过滤违规词库（威胁、侮辱、歧视性语言）；
- 确保数据符合《生成式 AI 服务管理暂行办法》等法规。

（3）数据标注与增强阶段：注入知识，平衡分布

核心任务：通过人工 +AI 协同，为数据赋予业务语义，并解决样本不平衡问题。

关键实践：

- **标签优化（Label Optimization）**
 - 确保标签高区分度（无语义重叠）、全覆盖（含“其他”类别）、无歧义；
 - 合并低频 / 模糊标签（如“下周还款”“下月还款”→“未来还款”）；
 - 利用大模型（如 Qwen3-235B）检测多标签冲突，分层细化标签体系。
- **数据分布整理（Data Distribution Collation）**
 - **过采样：**对低频标签样本上采样；
 - **欠采样：**对高频标签样本下采样；
 - **数据增强：**使用规则引擎或大模型生成合成样本（如替换还款金额、日期）。
- **多模态标注**
 - 图文对齐：确保图像描述与视觉内容一致；
 - 视频标注：标记动作起止时间、关键帧；
 - 思维链构建：记录从问题到答案的推理步骤。

（4）数据质检与评估阶段：量化质量，闭环优化

核心任务：建立科学、可量化的质量评估体系，确保数据集“可用、可信、有效”。

评估框架（参考自信通院 ADAQ 与 CCSA 指标）：

一级指标	二级指标示例	评估方法
完整性	样本量、字段缺失率	自动统计
准确性	标注一致性、内容真实性	双盲抽样 + 专家评审
多样性	领域 / 来源 / 文化多样性	分布熵计算
均衡性	类别 / 来源均衡度	基尼系数
合规性	隐私脱敏率、版权合规	规则检测 + 人工核查
有效性	模型性能提升度	A/B 测试

表 2 数据集质量评估框架

实施要点：

- **自动化为主，人工为辅：**开发 60+ 质检算子（如敏感词检查、相似样本查重），自动化评测率 >75%；
- **模型反哺数据：**将模型训练效果作为终极质量指标，建立“数据→模型→反馈→优化”闭环；
- **版本化管理：**每次质检后生成数据集版本，支持回溯与对比。

（5）数据运营与服务阶段：持续迭代，释放价值

核心任务：将数据集从“项目资产”转化为“企业级服务”，实现长期价值。

运营体系：

- **存储与版本管理**
 - 向量数据库（如 AnalyticDB for Vector）高效存储嵌入向量；
 - 对象存储 + 元数据管理，支持 PB 级非结构化数据；
 - Git-like 版本控制，追踪数据变更历史。
- **数据服务化**
 - 通过 Data Service 发布 RESTful API，供 AI 应用调用；

- 构建特征平台（Feature Store），统一线上线下特征；
- 支持 RAG 场景下的向量相似性检索。

• 流通与共享

- 在可信数据空间内实现“数据可用不可见”（联邦学习、隐私计算）；
- 通过数据交易所挂牌交易，探索数据资产化路径。

3. 高质量数据集的安全管理

AI 数据集的安全合规是生命线。政企客户需构建覆盖全生命周期的安全治理体系：

（1）数据分类分级

- 依据《数据安全法》《个人信息保护法》，对数据集进行敏感度分级（公开、内部、机密、绝密）；
- 结合业务属性打标（如“个人隐私”“商业秘密”“国家秘密”）；
- 阿里云 Dataworks AI-Copilot 等组件支持自动识别敏感字段，动态打标。

（2）隐私保护技术

- 脱敏：替换、加密、泛化（如年龄区间化）；
- 匿名化：k- 匿名、l- 多样性，确保无法重识别；
- 合成数据：使用 GAN/ 扩散模型生成仿真数据，替代真实敏感数据。

（3）版权与授权管理

- 明确数据来源合法性，保留授权证明；
- 对自建数据集进行版权登记；
- 在数据集中嵌入水印，追踪泄露源头。

（4）安全审计与监控

- 记录数据访问日志，实现操作可追溯；

- 设置异常访问告警（如高频下载、非工作时间访问）；
- 定期开展安全合规评估。

4. 高质量数据集的效果评估

数据集的价值最终体现在 AI 模型的业务成效上。评估体系应包含三层：

（1）数据质量评估

- 采用信通院 ADAQ 指标体系，量化数据集基础质量；
- 输出《数据质量评估报告》，包含得分、问题清单、改进建议。

（2）模型性能评估

- 在标准测试集上对比基线模型，衡量准确率、召回率、F1 值提升；
- 针对业务场景设计专项评测（如风控拦截率、营销转化率）。

（3）业务价值评估

- 成本节约：如人工巡检减少 60%（低空经济案例）；
- 效率提升：如故障响应时间缩短 60%（电力巡检）；
- 收入增长：如精准营销带来 GMV 提升。

5. 高质量数据集的运维运营

高质量数据集不是“一次性项目”，而是需要持续运营的“活资产”。

（1）运维监控

- 数据新鲜度监控：确保训练数据不过期；
- 质量漂移检测：当新数据分布偏离历史数据时告警；
- 服务 SLA 保障：API 响应时间、可用性监控。

(2) 成本管理

- **存算成本优化**：冷热数据分层存储，自动压缩；
- **人力成本控制**：通过智能标注降低人工依赖；
- **ROI 分析**：衡量数据集投入与业务收益比。

(3) 生态运营

- **内部推广**：建立数据目录，鼓励跨部门复用；
- **外部合作**：与高校、科研机构共建数据生态；
- **持续迭代**：根据业务反馈，定期更新数据集。

6. 构建“数据－模型－业务”飞轮

高质量数据集建设，本质是构建一个**“数据驱动 AI，AI 反哺数据”**的正向飞轮：

- 1. **高质量数据** → 训练出**高性能模型**；
- 2. **高性能模型** → 创造**高业务价值**；
- 3. **高业务价值** → 激励**更多数据投入**；
- 4. **更多数据投入** → 产生**更高质量数据**……

阿里云飞天企业版大数据与 AI 产品体系，已全面支撑这一飞轮的运转：

- **DataWorks + Maxframe**：数据治理与目录管理；
- **MaxCompute + AIStudio**：多模态数据处理与模型训练；
- **DataWorks Data Service**：标签管理与数据服务化；
- **AnalyticDB for Vector**：向量存储与检索。

政企客户应立足自身业务场景，以“小切口、快验证、深迭代”为原则，从一个高价值 AI 场景入手，逐步构建覆盖全组织的高质量数据集体系，真正实现“上好云，用好数，赋好能”。

(三) 智能体赋能 AI 数据中台建设

1. 数据智能体需求概述

在政企客户加速推进数字化转型与智能化升级的背景下，数据中台已从“建起来”迈向“用得好、管得住、智驱动”的新阶段。随着大模型（LLM）、生成式 AI（AIGC）等技术的成熟，AI 正深度融入数据全生命周期管理，催生出对**数据治理智能化、自动化、语义化**的全新诉求。

然而，传统数据治理体系面临三大核心挑战：

- a. **治理规则依赖人工经验**：质量规则、标签体系、元数据注释等高度依赖专家手工编写，效率低、覆盖窄、一致性差；
- b. **隐性知识难以显性沉淀**：大量业务逻辑散落在 SQL 脚本、报表、临时查询中，无法有效转化为可复用、可治理的主题资产；
- c. **治理响应滞后于业务变化**：面对高频迭代的数据加工链路和动态演进的业务语义，人工治理难以实现“实时感知—智能推荐—闭环优化”。

在此背景下，**数据智能体（Data Governance Agent）**应运而生。它并非孤立的 AI 模型，而是以大模型为认知引擎、以数据中台为执行载体、以人机协同为治理范式的**智能治理中枢**。其核心目标是：

将数据治理从“被动响应、人工驱动”升级为“主动发现、AI 辅助、专家决策、持续进化”的智能闭环体系。

数据智能体需具备以下关键能力：

- **语义理解能力**：理解表结构、字段含义、SQL 逻辑、业务术语背后的语义；
- **模式发现能力**：从海量数据行为中识别共性分析模式、异常分布、潜在关联；
- **推理推荐能力**：基于上下文自动生成元数据、质量规则、标签、主题模型建议；
- **人机协同能力**：支持专家审阅、修正、反馈，并将人类知识反哺模型优化；
- **安全合规能力**：满足国产化部署、权限隔离、操作留痕、审计追溯等政企严苛要求。

后文将系统阐述数据智能体在**元数据管理、血缘治理、智能建模、标签体系、质量管理**五大核心场景的技术实现路径与最佳实践，为政企客户构建“可信、可用、可运营”的 AI 就绪型数据资产体系提供方法论与工具支撑。

2. 智能元数据治理体系

（1）元数据治理的智能化跃迁

传统元数据管理依赖人工录入与维护，存在“三难”问题：**难写**（描述不规范）、**难找**（语义模糊）、**难信**（更新滞后）。在 AI 时代，元数据不仅是“数据的说明书”，更是大模型理解业务语境、生成治理逻辑的**关键上下文**。

数据智能体通过“AI 生成 + 人机协同 + 版本追溯”三位一体机制，实现元数据从“静态记录”到“动态知识”的转变。

（2）技术实现路径

（2.1）智能元数据生成（AI Catalog）

- **输入：**MaxCompute 表结构（字段名、类型、分区）、样本数据（前 100 行）、已有中文注释、所属项目 / 主题域；
- **处理：**调用 Qwen-72B 或 DeepSeek-Coder 大模型，通过 Prompt Engineering 注入行业知识（如“consignee_name 表示收货人姓名，常见于报关单、物流单”），生成符合业务语境的表级摘要与字段说明；
- **输出：**结构化元数据增强包，包含：
 - 表业务含义（例：“dwd_customs_entry_detail: 记录每票进出口报关单的明细信息，含商品、数量、金额、监管条件等”）
 - 字段业务定义（例：“hs_code: 商品 HS 编码，8-10 位数字，遵循 WCO 国际标准”）
 - 数据敏感等级建议（例：“收发货人名称 → L3 敏感”）

（2.2）人机协同注释与版本管理

- 在 DataWorks 数据地图界面，AI 生成内容以“灰色建议”形式展示，用户可点击“采纳”或“编辑”；
- 所有修改自动记录至元数据版本库，支持：
 - 时间轴回溯（查看某字段说明的历史变更）
 - 差异对比（高亮显示 AI 建议 vs 人工修正）
 - 审计日志（操作人、IP、时间、理由）

（2.3）元数据 – 血缘联动治理

- 当某字段语义发生变更（如“status”从“审核状态”改为“通关状态”），DataWorks 血缘中心自动标记其下游所有依赖资产（报表、API、机器学习特征）；
- 下游用户打开相关资产时，系统弹出提示：“上游字段语义已更新，请确认逻辑是否受影响”。

（2.4）文本字段智能标准化

针对地址、备注、品名等非结构化文本，构建“规则 + 大模型”双引擎处理流水线：

步骤	技术方案	示例
规则定义	用户在 DataWorks 配置提取模板	“从 ‘address’ 中提取省、市、区”
AI 解析	调用 Qwen-Max 进行 NER 与归一	输入：“广东深圳南山区” → 输出: {province: "广东省", city: "深圳市", district: "南山区"}
人工审核	DataWorks 生成待办任务，专家批量校验	支持“全部采纳”“批量修正”“驳回重试”
模型反馈	修正结果存入样本池，触发 MaxFrame 微调	每月自动更新专用文本解析模型

表 1 文本字段智能标准化处理建议表

3. 智能血缘与根因分析体系

（1）从“链路可视”到“智能归因”

传统血缘仅解决“数据从哪来、到哪去”，而 AI 时代的血缘需回答：“**为什么出问题？谁最可能负责？如何快速修复？**”

数据智能体将血缘从“静态图谱”升级为“动态诊断网络”。

（2）字段级血缘自动构建

- **采集源：**MaxCompute 任务日志、SQL 脚本、PyODPS 代码、DataWorks 调度依赖；
- **解析能力：**
 - SQL：精准识别 `SELECT a.id AS entry_id, b.name` → 字段映射关系；
 - UDF/Shell：通过代码注解或日志埋点补充血缘；
- **输出：**端到端字段级血缘图，支持：
 - 影响分析（修改上游字段，自动列出受影响报表）
 - 溯源审计（点击下游指标，反向追踪至原始业务系统）

（3）质量问题智能根因定位

当 DataWorks 检测到数据异常（如空值率突增），自动触发根因分析流程：

1. 血缘回溯：沿字段血缘向上游逐层扫描；

2. 多维比对：

- 各节点历史质量指标趋势
- 任务运行日志（失败、超时、资源不足）
- 代码变更记录（DataWorks 版本管理）

3. AI 归因排序：基于规则引擎 + Qwen 语义分析，输出疑似根因 TOP3，例如：

“根因概率 85%：ods_customs_entry_raw 昨日新增字段解析逻辑，未处理空值场景；

根因概率 10%：上游海关总署接口昨日延迟 3 小时，导致部分数据缺失。”

4. 生成追溯报告：结构化文档包含证据链、责任人、修复建议，一键推送至治理工作台。

4. AI 驱动的智能建模与主题资产孵化

（1）隐性需求显性化：从 SQL 日志到主题模型

政企客户普遍存在“烟囱式开发”问题——各业务部门重复编写相似 SQL，造成计算资源浪费与数据口径不一致。

数据智能体通过 **SQL 行为挖掘**，将分散的分析逻辑聚类为可复用的主题资产。

（2）实现路径

1. SQL 日志采集

- 通过 DataWorks 任务运维中心对接 MaxCompute 审计日志，获取全量 SQL 查询；
- 过滤无效查询（如 DESC、SHOW），保留 SELECT/JOIN/AGG 类分析语句。

2. 查询逻辑指纹提取

使用 DataWorks 内置 SQL Parser，结构化拆解为：

```
{  
  
  "tables": ["ods_entry_list", "dim_hs_code"],  
  
  "joins": [{"left": "entry.hs_code", "right": "hs.code"}],  
  
  "filters": ["entry.create_date >= '2024-01-01'"],  
  
  "aggregations": [{"metric": "COUNT(entry_id)", "group_by": ["hs.category"]}]]  
}
```


3. 语义聚类与模式发现

- 将指纹向量化（Qwen Embedding），结合规则（字段重合度 >70%、聚合模式相同）进行聚类；
- 生成人类可读的模式摘要：“近 30 天按 HS 大类统计报关单量”。

4. 主题模型智能推荐

DataWorks 自动生成建模建议包：

- 主题域：企业风控
- 事实表：dws_enterprise_risk_daily
- 维度：企业 ID、风险等级、时间
- 指标：高风险报关单数、涉敏商品占比

5. 闭环纳入 OneModel

- 专家审定后，DataWorks 建模工具一键生成 DWD/DWS 层模型；
- 原始 SQL 自动重定向至新主题表，减少重复计算。

5. 智能标签治理体系

（1）标签即服务：从“人工打标”到“AI 推荐”

数据标签是连接业务语义与技术资产的桥梁。传统打标依赖人工规则，覆盖面窄、维护成本高。

数据智能体构建“**特征融合 + 行业知识 + 人机协同**”的智能打标引擎。

（2）技术架构

1. 多维特征提取

- 技术特征：字段名（含“phone”）、类型（STRING）、空值率（<5%）
- 业务特征：所在表名（“cust_info”）、已有注释（“客户手机号”）

- 行业规则：预置《金融数据分类分级指南》《海关数据标签白皮书》

2. AI 标签推荐

调用 Qwen 大模型，输入特征向量，输出多维度标签：

```
{  
  
  "business_tags": [" 客户联系方式 ", " 个人身份信息 "],  
  
  "tech_tags": ["PII", "L3 敏感 "],  
  
  "governance_tags": [" 需脱敏 ", " 跨境传输受限 "]  
}
```

3. 规则引擎增强

配置硬性规则（如“字段名含‘id_card’→必打‘身份证号’标签”），与 AI 结果加权融合。

4. 闭环管理

- 标签审核任务推送至 DataWorks 标签工作台；
- 审核通过后自动写入 MaxCompute 表属性（set table_property('tags'='PII,L3')）；
- 标签结果供 DataWorks 权限中心、质量中心、API 网关调用。

6. 智能数据质量管理

（1）质量治理的三大智能化突破

传统方式	智能体方案
人工编写质量规则	AI 基于元数据 + 分布特征自动生成规则草案
固定阈值告警	动态基线 + 业务语义感知的智能阈值
问题发生后排查	血缘联动 + 根因推荐，提前预警

表 2 质量治理的智能化方案

(2) 质量规则智能生成

- **输入：** 字段类型、样本分布、业务语义、行业模板；
- **AI 生成示例：**
 - 身份证号： 格式符合 GB 11643，长度 =18，校验码合法
 - 订单金额： >0 且 <1 亿，空值率 <0.1%，日波动 <± 50%
- **人机协同：** 专家可调整阈值、补充例外规则（如“促销日允许金额突增”）。

(3) 自适应异常检测

- **数值型：** Prophet 时序预测 + Isolation Forest
- **文本型：** SimHash 去重 + BERT 语义聚类
- **枚举型：** 非法值检测（对比预设值域）

(4) 闭环处置

- 异常事件 → DataWorks 待办任务 → 专家研判 → 自动触发修复作业（如去重 SQL） → 验证闭环。

7. 结语： 迈向 AI 原生的数据治理体系

数据智能体不是替代人类专家，而是将专家从繁琐的重复劳动中解放，聚焦于高价值的**业务判断与策略制定**。阿里云将持续深化“大模型 + 数据中台”融合创新，助力政企客户构建**自主可控、智能高效、安全合规**的新一代数据治理体系，真正实现“上好云，用好数，赋好能”。

(四) 智能数据资产盘点与智能问数探索

1. 智能数据资产盘点与智能问数需求概述

在政企数字化转型迈向“数智融合”新阶段的今天，数据不再仅仅是业务运行的副产品，而是驱动组织决策、优化流程、创新服务的核心战略资产。然而，随着数据规模爆炸式增长、数据源日益多元、数据结构愈发复杂，传统以人工为主的数据资产管理方式已难以应对“找数难、懂数难、用数难”的现实困境。

在此背景下，**智能数据资产盘点与智能问数探索**成为当前 AI 驱动型数据中台建设的“终极目标”之一。其本质是通过大模型（LLM）、向量检索、知识图谱、语义建模等 AI 原生技术，将沉睡在数据湖仓中的海量资产转化为可被自然语言理解、可被业务人员直接调用、可自动关联治理规则的“活数据”，从而实现从“数据可用”到“数据好用、爱用、会用”的跃迁。

这一目标的达成，并非孤立的技术堆砌，而是高度依赖于前期扎实的数据中台建设成果——特别是基于阿里云“**One Model、One Data、One Service**”方法论所构建的统一数据标准体系、全域数据血缘网络、高质量主题域模型以及可信的数据服务接口。只有当数据具备了清晰的业务语义、一致的规范定义、完整的血缘追踪和可靠的质量保障，AI 才能真正“读懂”数据、“信任”数据，并安全高效地服务于业务场景。

因此，智能数据资产盘点与智能问数探索，既是数据中台价值释放的关键出口，也是检验数据治理成效的“试金石”。它标志着政企客户的数据治理体系正从“被动合规”走向“主动赋能”，从“支撑系统”升级为“智能引擎”。

后文将围绕这一高阶智能化需求，结合阿里云飞天企业版大数据与 AI 产品体系，系统阐述面向政企客户的**通用化、可落地、安全可控**的智能数据治理技术框架。

2. 智能数据资产盘点: 让数据资产“看得见、说得清、管得住”

(1) 需求本质：从“静态目录”到“动态认知”

传统数据资产目录（Data Catalog）多以表格形式罗列表名、字段、负责人等元信息，虽解决了“有无”的问题，却难以回答业务人员最关心的三个问题：

- **这个数据我能用吗？**（权限、质量、时效性）
- **这个数据代表什么？**（业务含义、取值逻辑、关联实体）
- **这个数据从哪来、到哪去？**（血缘影响、变更风险）

智能数据资产盘点的核心诉求，正是通过 AI 技术赋予数据资产“自我解释”与“主动推荐”的能力，构建一个**可对话、可推理**、可行动的动态数据认知系统。

(2) 技术架构：构建政企级智能资产认知中枢

我们提出“**三层一体**”的智能资产盘点技术框架，深度融合阿里云大数据与 AI 产品能力：

层一：资产感知层 —— 全域元数据统一采集与向量化

- **能力来源：**DataWorks 智能数据目录 + MaxCompute / Hologres / OSS 等存储引擎
- **核心动作：**
 - 自动采集全域数据资产元数据，包括：库表结构、字段注释、业务分类标签、数据质量评分（来自 DataWorks 质量中心）、更新频率、负责人、敏感等级等。
 - 利用**通义千问 Qwen-Embedding 或 DeepSeek-Embedding** 等国产大模型对元数据文本（如表注释、字段描述、业务术语）进行**高维语义向量化**，生成统一语义空间下的向量表示。
 - 将向量存入**阿里云 AnalyticDB for PostgreSQL（ADB PG）或 OpenSearch** 向量引擎，支持毫秒级语义相似度检索。

层二：智能交互层 —— 自然语言驱动的语义检索与推理

- **能力来源：**Qwen/DeepSeek 大模型 +RAG（Retrieval-Augmented Generation）架构
- **核心流程：**
 - a. 用户输入自然语言问题，如：“有哪些表包含企业统一社会信用代码？”
 - b. 系统通过大模型进行**意图识别与关键词提取**，生成查询向量。
 - c. 在向量库中执行**近似最近邻（ANN）检索**，召回 Top-K 相关资产。

d. 结合原始元数据与上下文，由大模型生成**自然语言摘要**，如：“共找到 3 张表包含‘统一社会信用代码’字段，分别为：企业基本信息表（DWD_ENT_INFO）、报关主体表（DWD_DECL_ENT）、信用评级表（DWS_ENT_CREDIT_SCORE）。其中 DWD_ENT_INFO 为权威主 数据源，每日更新。”

层三：多模态呈现层 —— 动态可视化与治理联动

- **能力来源：**DataV 数据可视化 + 图数据库（如 GDB）+ DataWorks 治理中心
- **三大视图模式：**
 - **列表视图：**展示资产基础属性（名称、主题域、质量分、更新时间、权限范围）。
 - **知识图谱视图：**以“实体 - 关系”形式展现业务对象间的关联，如 [报关单]-(申报于)->[口岸]、[商品]-(归类于)->[HS 编码]，帮助用户理解数据背后的业务逻辑。
 - **血缘拓扑视图：**图形化展示端到端数据链路，标注各节点质量状态（绿 / 黄 / 红），支持下钻查看字段级血缘与样本数据。

治理闭环：检索即治理

当用户聚焦某一数据资产时，系统可调用大模型分析其结构与内容，**自动生成治理建议**，例如：

- “该字段为日期类型，但存在 10% 空值，建议补充默认值或加强源头校验。”
- “‘客户 ID’在 3 张表中命名不一致（cust_id / customer_no / client_code），建议统一为‘cust_id’。”这些建议可一键转为 DataWorks 中的**数据质量规则或元数据修正任务**，实现“发现 - 分析 - 治理”闭环。

3. 智能问数探索：让业务人员“像聊天一样分析数据”

(1) 需求本质：从“写 SQL”到“说人话”

传统 BI 工具依赖预设报表或技术人员编写 SQL，业务人员难以灵活探索数据。智能问数的目标是：**任何具备业务知识的人员，只需用自然语言提问，即可获得准确、可信、可视化的分析结果。**

这不仅降低用数门槛，更将数据治理成果（如标准指标、维度一致性、质量保障）直接转化为业务价值。

(2) 技术架构：构建可信、安全、场景化的智能问数引擎

步骤一：构建业务语义模型（Semantic Layer）

- 能力来源：DataWorks 语义建模
- 关键动作：
 - 基于已治理的 DWD/DWS 层数据，抽象出**业务对象**（如“销售订单”“设备工单”“患者就诊记录”）。
 - 定义**标准指标**（如“月度销售额”“设备故障率”“平均住院日”）及其计算逻辑。
 - 定义**标准维度**（如“区域”“产品线”“科室”“时间粒度”）及层级关系。
 - 明确对象间**关联关系**（如“订单属于客户”“工单关联设备”）。

步骤二：部署多模型智能问数助手

- 能力来源：AIStudio + Qwen/DeepSeek + DataV（或自研 NL2SQL 引擎）
- 处理流程：
 - a. **意图识别**：判断问题所属业务域（如“风控”“运营”“财务”）。
 - b. **语义解析**：将自然语言映射到语义模型中的指标、维度、过滤条件。
 - 示例：“华东区上季度高端手机销量” → 指标 = “销量”，维度 = “区域 = 华东, 产品线 = 高端手机”，时间 = “上季度”
 - c. **SQL 生成**：调用 DataV-Note 或自研 NL2SQL 模块，生成符合 MaxCompute 语法、带行级 / 列级权限控制的查询语句。
 - d. **结果渲染**：执行后返回数据，由 DataV 自动生成图表（趋势图、热力图、地图等），并附加数据可信度提示（如“该指标昨日质量评分为 85 分”）。

步骤三：增强交互体验与场景落地

- 多轮对话**：支持上下文追问，如首问“北京关区通关量”，追问“其中快件占比多少？”
- 智能推荐**：基于用户角色（如“风控专员”“运营经理”）推送高频问题模板。

- 提词管理**：业务专家可在 DataWorks 后台维护标准问法，如“{ 区域 } { 时间 } 的 { 指标 }”，提升泛化能力。
- 场景模板库**：预置多个行业通用问数场景，例如：
 - 金融**：高风险客户交易行为分析、贷款逾期预测
 - 制造**：设备 OEE 分析、不良品根因追溯
 - 政务**：民生诉求热点聚类、政策兑现进度监控
 - 能源**：电网负荷预测、新能源发电效率分析

4. 安全、集成与国产化：构建可信的政企智能数据门户

(1) 统一入口：智能数据助手

我们将前述能力封装为“**数据智能助手**”，提供多种面向现有数据门户的集成方式，作为政企数据门户的核心交互组件。

(1.1) 集成模式

模式	说明	适用场景
API 模式	提供 RESTful API，供客户现有门户调用	已有成熟门户系统，需轻量集成
嵌入式页面	提供 iFrame 或微前端组件，无缝嵌入门户	追求一致 UI 体验，快速上线

表 1 智能数据门户集常见成方式

(1.2) 安全与权限

- 权限对接**：与客户现有 IAM 系统（如 LDAP、H4A、钉钉组织架构）深度集成，确保用户仅能检索 / 查询其授权范围内的数据。
- 数据脱敏**：在 SQL 生成阶段自动注入脱敏规则（如手机号显示为 138****1234）。
- 审计日志**：完整记录所有问数与检索行为，满足等保与合规要求。

(2) 国产化与专有云适配

- **全栈国产**: 支持在阿里云飞天企业版专有云环境中部署,使用国产芯片(平头哥)、操作系统(龙蜥)、数据库 (PolarDB)、大模型 (Qwen)。
- **模型私有化**: Qwen/DeepSeek 等大模型可私有化部署于客户 AI 平台,训练数据不出域。
- **算力自主**: 依托 AIStudio 平台,支持在客户自有 GPU/TPU 集群上运行向量计算与推理任务。

5. 安全、实施路径与价值度量

(1) 分阶段实施建议

阶段	目标	关键动作
准备期	数据资产就绪	完成 One Data 建设,建立主题域、标准字段、质量规则
试点期	场景验证	选择 1-2 个业务域 (如风控、运营), 构建语义模型, 上线智能检索 + 问数 MVP
推广期	全面赋能	扩展至全业务域, 集成至数据门户, 培训业务用户
优化期	智能进化	基于用户反馈优化模型, 增加治理联动规则

表 2 分阶段实施目标和关键动作

(2) 价值度量指标 (仅供参考)

- **效率提升**: 业务人员找数时间缩短 70%, 自助分析占比提升至 60%+
- **治理显性化**: 高质量数据集使用率提升 50%, 治理规则采纳率提升 40%
- **业务价值**: 风险识别准确率提升、运营决策响应速度加快、数据服务满意度达 90%+

6. 结语：迈向“数据智能体”时代

智能数据资产盘点与智能问数探索，不仅是技术升级，更是数据文化与组织能力的重塑。它让数据从“IT 资产”回归“业务语言”，让治理从“成本中心”转变为“价值引擎”。

阿里云依托飞天企业版大数据与 AI 产品体系，结合阿里巴巴集团多年数据中台实践，愿与广大政企客户携手，共建**安全、智能、高效**、可信的新一代 AI 数据治理体系，真正实现“上好云，用好数”，开启数智未来新篇章。

第五节 /

企业服务智能体应用案例

（一）灵渠－云化架构智能体：让大模型成为企业架构师的“数字分身”

引言：从“经验依赖”到“智能复用”的范式跃迁

当前云原生落地仍面临一个深层挑战：架构能力高度依赖个体经验，而经验本身难以结构化、标准化，更难以规模化复用。传统方法如 SOP 沉淀、最佳实践文档或工具平台，虽能提升局部效率，却难以突破两大瓶颈：一是大量关键决策基于隐性知识，难以显性表达；二是客户场景高度碎片化，通用方案难以直接适配。

大模型的出现为此提供了全新路径。凭借检索增强生成（RAG）、工具调用（Tool Use）与多智能体协同（Multi-Agent Collaboration）等能力，大模型可将领域知识、运行数据与复杂推理有机融合，实现从“经验驱动”向“智能驱动”的跃迁。

基于此，我们启动“云化架构智能体”项目，构建一位具备规划、感知、分析与行动能力的“数字架构师”，让大模型成为企业架构能力的可执行载体，自动完成架构评估、风险识别与优化方案生成等高价值工作。

本文将系统阐述该智能体的背景动因、业务架构、技术实现与演进方向，为政企客户提供一套可复制、可扩展、可落地的架构智能化实践路径。

1. 智能体背景：为何需要“数字架构师”？

（1）三大核心瓶颈：经验难以规模化复制

在实际交付中，我们总结出经验复用难的三大瓶颈：

- **知识隐性化**：80% 以上的架构决策依赖专家直觉与经验，难以显性表达；
- **能力个体化**：资深架构师成长周期长达 3-5 年，新人难以快速胜任复杂项目；
- **成果孤岛化**：每个项目产出独立，缺乏统一知识沉淀与自动化复用机制。



图 1 专家经验难以规模化复制的“三大瓶颈”

例如，某电力客户希望将核心系统迁移到阿里专有云，涉及微服务拆分、可观测体系建设、高可用容灾等多个维度。若无资深架构师参与，客户往往在组件选型（如使用 RocketMQ 还是 Kafka）、弹性策略（是否启用 HPA）、可观测体系（是否集成 ARMS）等关键节点反复试错，导致项目延期、成本超支。

（2）大模型的破局价值

大模型的出现，为上述问题提供了全新解法：

- **架构设计方案生成**：通过 RAG 技术，将云组件选型规则、行业最佳实践与架构原则注入生成过程。例如，当用户输入“需要支持金融级高可用的微服务架构”，智能体可自动召回“两地三中心”容灾模式、EDAS 无损上下线、AHAS 熔断降级等知识，生成符合监管要求的方案。

- **架构风险识别与优化:** 大模型可融合逻辑架构(如服务依赖图)、运行态数据(如 CPU 利用率、GC 频率)、代码日志 (如慢 SQL) 等异构信息，进行多维度关联分析。例如，某系统在压测时 RT 突增，传统方式需人工排查，而智能体可自动关联 “Full GC 频繁 + 数据库连接池耗尽 + Redis 缓存击穿” 三重风险，提出组合优化建议。

这种能力，本质上是将专家经验转化为可计算、可推理、可执行的智能服务。

2. 智能体业务架构：构建面向云原生的“数字架构师”能力体系

在传统云交付模式中，架构设计高度依赖专家个体经验，缺乏标准化、自动化的能力载体。而云化架构智能体的核心目标,是构建一个**具备规划(Plan)、感知(Perceive)、分析(Analyze)与行动 (Act) 四维能力**的**“数字架构师”**，使其能够像资深架构师一样，理解业务诉求、诊断系统现状、权衡技术选项，并输出可执行的优化路径。

（1）业务目标对齐：以“4L6D”云化应用架构模型为框架约束

为避免智能体陷入“泛泛而谈”的通用问答陷阱，我们将其能力边界严格限定在 **“4L6D”云化应用架构模型** 框架内（见图 2）：

- **4L（四层架构视图）**：业务架构、应用架构、数据架构、技术架构，确保智能体输出与企业整体 IT 治理对齐；
- **6D（六大能力维度）**：服务化、高可用、高性能、可观测、高弹性、自动化，构成可量化、可评估、可优化的技术基线。

该模型不仅为智能体提供了**结构化的评估坐标系**，也为其方案生成设定了**合规性与可行性边界**。例如，当客户提出“提升系统弹性”时，智能体不会泛泛建议“加服务器”，而是基于“高弹性”维度下的具体能力项(如突发流量识别、HPA 策略配置、弹性伸缩触发条件等)，结合运行态数据，给出精准的容器扩缩容策略



图 2 云化架构智能体业务架构

（2）能力闭环设计：从静态评估到动态演进

智能体的业务价值不仅体现在一次性的架构评估，更在于构建持续演进的架构治理闭环：

- 1. 现状感知阶段：**通过工具链采集逻辑架构（如服务依赖图、组件选型）与运行态数据（如 QPS、GC 频次），构建“应用全景架构状态”；
- 2. 差距分析阶段：**将现状与“4L6D”目标等级对比，识别短板（如“可观测性仅达 L1，未实现全链路追踪”）；
- 3. 方案生成阶段：**基于知识库中的最佳实践，生成分阶段演进路径（如“L1 → L2：接入 ARMS，配置 Trace 采样率 ≥ 1%”）；
- 4. 持续验证阶段：**未来可结合自动化测试与灰度发布，验证优化效果，形成“评估 - 优化 - 验证”闭环。

这种设计使智能体从“一次性报告生成器”升级为“架构演进教练”，真正支撑客户实现**架构现代化的长期目标**。

3. 智能体技术架构：混合式多智能体 + 工具链 + 知识库的深度融合

为支撑上述业务目标，我们构建了一套**分层解耦、协同自治、可扩展演进**的技术架构体系（见图 3），涵盖交互服务层、应用层与模型服务层。



图 3 云化架构智能体技术架构

（1）智能体应用层：混合式多智能体架构设计

- 我们采用 “**Workflow + ReAct**” 混合架构（见图 4），兼顾流程可控性与任务自主性：
- **Workflow 模式**：适用于可结构化拆解的主干流程（如“任务规划 → 感知 → 分析 → 报告”），通过工作流引擎确保执行顺序与异常兜底；
 - **ReAct 模式**：适用于开放式、探索性任务（如“自主判断数据是否完备”），允许智能体动态调用工具、迭代推理。

四类核心 Agent 职责分明、高度自治：

Agent 类型	核心能力	技术实现要点
任务规划 Agent	意图识别、任务拆解、子 Agent 调度	采用通义千问 -Plus 进行高精度意图分类；支持编排工作流动态生成子任务
架构感知 Agent	工具选择、数据采集、完备性校验	基于 ReAct 模式，自主调用 X-Scanner（IDC 场景）或 EDAS API（云上场景）；内置数据校验规则
评估分析 Agent	多维推理、风险识别、方案生成	融合逻辑态与运行态数据；召回 RAG 知识库中的“4L6D”标准与最佳实践
报告生成 Agent	内容整合、格式转换、富媒体渲染	输出 Markdown 结构化报告；支持 HTML 沙箱渲染交互式图表

表 1 云化架构智能体核心 Agent 能力说明



图 4 云化架构智能体多 Agent 协同机制

主流程采用 **Workflow 引擎串联**，确保关键路径可控；子任务（如并行评估六大维度）采用 **A2A + MCP 协议**，支持灵活扩展。

在智能体分工设计上，一份报告的最终生成需经由多个智能体的协同配合完成，涵盖架构信息获取、评估分析、方案生成等关键环节。将任务拆分为多个智能体执行，一方面有助于提升输出内容的质量与专业性，另一方面也为针对不同环节集成差异化知识库、采用不同推理模型，实现更精细化的能力匹配与扩展能力。

一个常规的架构评估报告生成过程：

- 1. 由用户发起架构规划任务（例如：上云架构规划设计任务或云上架构评估检测任务等），由任务规划智能体进行任务的分析、拆解与分配。
- 2. 任务分配到架构感知智能体，该智能体会通过大模型分析出来需要采集的架构指标或运行时数据，选择需要使用到的工具并准备好输入参数数据。
- 3. 架构感知智能体调用相应工具从应用系统运行环境和基础设施环境中获取相应的数据，更新保存到应用系统架构全景状态数据中。
- 4. 架构感知智能体通过任务目标和获取到数据进行推理分析，判断应用系统架构全景状态数据是否准确和完备，如果已经满足任务需要，开始启动架构评估分析子任务。
- 5. 评估分析智能体根据任务目标和架构感知智能体生成的架构全景状态数据，召回知识库中的架构知识数据（架构规范、设计方案、最佳实践等），利用大模型进行综合性评估结论和优化方案的生成。
- 6. 报告生成智能体，会整合汇总评估分析智能体生成的结果数据，生成最终版本的架构评估分析报告。
- 7. 架构评估分析报告通过前端服务展现给用户（可以进行多种形式的展示和保存），并可以进行人工审核和修改调整。

（2）工具技术架构：打通“逻辑态”与“运行态”的感知能力

智能体的“眼睛”是其工具链。架构感知类工具的设计，主要以 python 代码实现的指标采集工具为主。架构分析设计阶段以逻辑架构指标采集为主，架构评估检测阶段以运行态数据指标为主。并采用 MCP 协议接入第三方以及自定义的服务以增加智能体的环境感知能力。

- **动态文件解析**：解决线下 IDC 环境工具覆盖不足问题，通过 Excel 上传 + 多轮问答校验，实现信息采集；
- **逻辑架构检测**：基于“4L6D”模型自动解析微服务拓扑、组件选型等静态信息；
- **运行态指标采集**：深度对接阿里云产品 EDAS、ARMS、Prometheus 等，获取真实运行数据（见下表）；
- **外部存储工具**：封装 OSS 与数据库，实现中间态数据持久化与 Agent 间上下文共享；
- **数据处理工具**：支持 Markdown → PDF/HTML 等格式转换，提升报告可读性。

运行态关键指标覆盖五大维度：

架构特征	数据指标项
服务化	灰度发布规则、分布式事务选型
高性能	CPU/MEM 利用率、Full GC 频次、慢 SQL
高可用	QPS/RT、服务失败率、连接池配置
可观测	日志规范性、Trace 有效性
高弹性	突发流量峰值、弹性伸缩触发记录

表 2 运行态五大维度关键指标

（3）知识库技术架构：构建高可信 RAG “外脑”

RAG 知识库是智能体的“大脑”，其质量直接决定输出专业性。灵渠云化架构智能体构建了**三层知识体系**：

- 1. **云化应用架构模型**：提供评估标准与等级定义；
- 2. **架构设计方案库**：沉淀大量最佳实践，覆盖微服务、中间件、容灾等场景；
- 3. **私域知识融合**：集成客户内部规范（如安全合规、运维要求），确保建议“贴地飞行”。

知识库建设遵循“**准备 → 解析切片 → 检索召回 → 生成答案**”四阶段闭环，每一阶段均设质量门禁：

- **知识准备**：统一为 Markdown 等标准格式，规范术语；
- **解析切片**：采用二级标题等方式进行自定义切分，保障语义完整；
- **检索召回**：标签过滤 + GTE-ReRank 重排序，提升相关性；
- **生成答案**：启用拒识机制 + 展示来源，抑制幻觉。

在模型选型方面，我们针对不同任务特性进行了精细化配置：

- **逻辑推理任务**采用 **通义千问 -Max-Latest**，显著提升复杂指令理解与多步推理方面的表现；
- **意图识别任务**选用 **通义千问 -Plus**，显著提升了用户输入分类的准确率；
- **RAG 检索链路**则结合 **Embedding v3** 与 **GTE-ReRank**，通过高精度向量化与重排序机制，大幅增强知识召回的相关性与生成内容的可靠性。

内容决定上限，流程决定下限。RAG 的成功，始于优质知识，成于系统工程。同时，RAG 知识库的建设并非“一次性工程”，而是一个持续迭代、动态优化的过程。

4. 典型场景与价值验证

在某电网项目中，我们部署该智能体系统，完成数个应用的架构评估：

- **效率提升**：单应用评估从 2 人日缩短至 2 小时；
- **风险覆盖**：自动识别出数十个高危风险（如未配置熔断、日志未脱敏）；
- **方案落地**：生成的优化建议 100% 被客户采纳。

在某烟草客户，该智能体可以极大提升客户容灾运营人员的效率，可以自动生成符合行业要求的容灾方案文档。

5. 总结与展望

灵渠 - 云化架构智能体标志着我们从“人工经验驱动”迈向“智能体协同驱动”的新阶段。它不仅是工具，更是**企业架构能力的数字化载体**——将专家经验固化为可执行、可复用、可进化的智能服务。

未来，我们将进一步探索：

- **AI 驱动的架构自愈**：基于运行数据自动触发弹性扩容、故障隔离；
- **“云 + 应用”一体化智能运营运维**：打通 IaaS、PaaS、SaaS 全栈数据，实现端到端优化。

(二) 专有云运维的“未卜先知”——大模型能力建设

1. 专有云运维演进与设计

(1) 专有云十年运维成熟度演进历程

专有云经过十年的发展和沉淀，专有云运维也从最初的全靠人支持人的阶段逐渐过渡到了文档化运维、工具化运维，具备自动化运维阶段。随着大模型的兴起，如何利用大模型能力去改变现有的运维模型是我们一直在思考的问题。



图 1 专有云十年运维成熟度演进历程

1. 人工运维阶段

2015 年 -2017 年，是专有云平台的初始创立阶段，运维工作主要依赖于人工操作和经验积累。这一时期的特点包括：

- **运维文档管理无序且不完善：**由于平台尚处于起步阶段，运维文档的管理和维护较为混乱，缺乏系统化的整理和分类。
- **问题解决依赖人员支持：**遇到问题时，解决问题的关键在于能否快速找到具有相关经验和技能的支持人员。这种模式效率低下，容易出现响应滞后和信息不对称的问题。

- **初期挑战与机遇：**尽管存在诸多不足，但这一阶段为后续的发展奠定了基础，积累了宝贵的实践经验和人才。

2. 文档化运维阶段

2018 年 -2020 年，随着平台逐渐成熟，逐渐过渡到文档化运维阶段。这一时期的显著变化是构建了以专有云运维中心平台为核心的知识体系，使得运维工作更加规范和高效：

- **完整的平台知识体系：**通过引入统一的运维中心平台，建立了全面的知识库，涵盖了各类运维文档、操作指南、故障处理手册等资源。
- **知识库覆盖率提升：**超过 80% 的运维问题能够通过知识库找到解决方案，极大地提高了问题解决的速度和准确性。
- **标准化流程：**制定了标准化的运维流程和最佳实践，确保每个问题都能得到一致且高效的处理。

3. 工具化运维阶段

进入 2021 年 -2022 年，专有云运维迎来了工具化运维阶段。这一阶段是以铜雀、SRE-CLI 为核心构建运维工具集，进一步提升了运维工作的自动化水平：

- **运维工具集：**铜雀和 SRE-CLI 成为运维工作的核心工具，提供了强大的根因分析能力和服务推荐功能。
- **自动化程度提高：**超过 70% 的运维问题可以通过这些工具进行自动化的根因分析，并获得相应的解决方案推荐，减少了人工干预需求。
- **效率与质量双重提升：**工具化运维不仅加快了问题解决速度，还保证了解决方案的质量和一致性。

4. 自动化运维阶段

2022 年 -2023 年间，专有云运维迈入了自动化运维的新阶段。这一时期的重点是建立统一的运维入口，完善监控和告警体系，实现日常运维任务的编排与自动化执行：

- **统一运维入口：**整合了多个运维工具和平台，提供统一的操作界面，简化了运维人员的工作

流程。

- **完善的监控与告警体系：**构建了覆盖全平台的监控和告警机制，能够实时检测并预警潜在问题，保障系统的稳定运行。
- **自动化编排与执行：**实现了从问题发现到解决方案部署的全流程自动化，大幅提升了运维效率和响应速度。

5. 2024 年至今：智能化运维开启

自 2024 年起，专有云运维正式步入智能化运维时代。借助大模型技术，平台实现了更高级别的智能运维能力：

- **智能根因分析：**利用大模型的强大计算能力和数据分析能力，对复杂问题进行精准的根本因分析，提供科学合理的解决方案。
- **运维自动化执行：**结合自动化工具和智能算法，实现了日常运维任务的完全自动化执行，减少了人为错误的可能性。
- **持续优化与学习：**通过不断积累和分析海量运维数据，大模型能够自我优化和学习，不断提升运维效果和用户体验。

经过十年的发展和沉淀，专有云运维已经积累了丰富的经验和庞大的技术资源：

- **专业运维团队：**拥有超过 2000+ 名专业运维人员，构成了一个庞大且高效的技术队伍。
- **高质量的数据资产：**积累了百万级别的日常运维问题记录、30 万 + 的操作方案、50 万 + 的专家支持案例以及 20 万 + 的高质量文档。
- **关键作用：**这些丰富的数据资源为运维大模型的研发提供了坚实的基础，尤其是在模型标注和效果评测方面起到了至关重要的作用。

（2）专有云运维大模型

专有云大模型在设立之初，我们就希望借助一套模型解决我们在一线、二线及产品能力演进上的各类需求。我们希望给一线更自主化的解决平台运维问题的能力，也希望在后端技术支持体系中发挥大模型的作用，最快效率的解决客户问题。我们也希望借助大模型的触点能够深入到云平台运维工具的各个方面给云产品更多运维智能。



图 2 专有云运维大模型示意图

（3）专有云运维大模型面临的挑战

运维大模型的落地在现实中我们还是会面临诸多的问题，我们不希望运维大模型成为一个玩具，而真正是解决运维各方面的利器。所以如何让大模型懂得专有云知识，并且让大模型给出的信息尽可能不出现幻觉，也不出现各种敏感信息是我们必须要解决的问题。



图 3 专有云运维大模型面临的挑战

2. 专有云运维大模型训练与评估

（1）降低幻觉伤害

特别是针对幻觉的问题，运维本身是一个风险厌恶性岗位，如何尽可能的降低幻觉概率和任何在出现幻觉可能时让运维人员知晓是我们首先要解决的问题，我们要做的第一件事就是要避免大模型成为一个玩具。我们通过问题分类补齐 + 命中策略 + 特定提示“来缓解幻觉的出现，也确保用户能够知晓生成后文本的产生方式。举例如下：

问题分类	问题举例	幻觉风险程度	解决方案	特定提示
咨询/查询类	tsar的使用方法、ecs产品架构	中	知识库+RAG+智能体	产品文档 日常总结 RAG生成 智能体
根因排查类	ECSRiver.RiverCluster#未达终态原因	高	4K方案+RAG+智能体	4K方案 RAG生成 智能体
运维处置类	ECS集群数据盘更换	致命	4K方案+精准命中	CLI方案

表 1 问题分类补齐 + 命中策略 + 特定提示

（2）“专家标注 +RAG+ 智能体” 产生专有云运维大模型

运维大模型的产生，我们是基于通义千问 2 + 检索增强生成 RAG 技术来构建，在此基础上利用专家标注好的专有云知识进行微调训练和对齐，并借助日常在服务客户过程中的工单数据进行评估优化。

智能体目前用在对客户问题的上下文内容补齐之上，核心是希望通过智能体能补充一些平台的真实运行信息，来帮助用户更精准的找到问题原因以及相应解决方案。

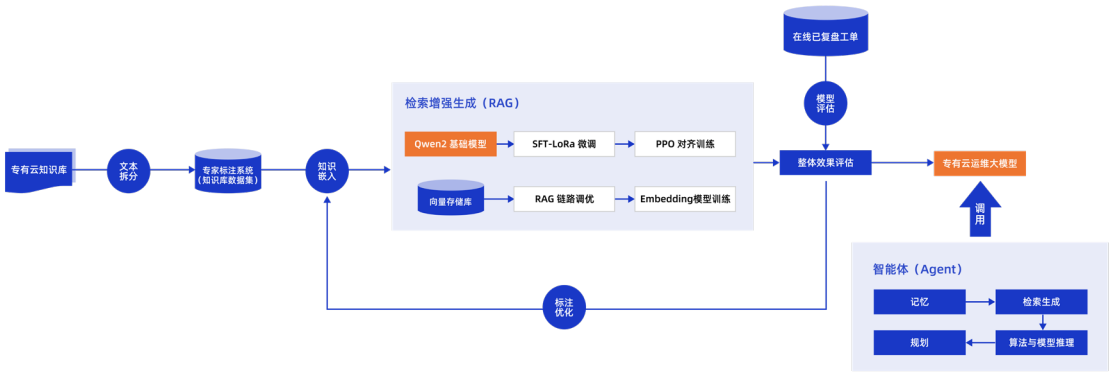


图 4 专家标注 +RAG+ 智能体

（3）专有云运维大模型的数据清洗与改造

专有云的语料数据来自于我们的 eocc 知识库平台，他的来源有咨询类文档、操作类文档、诊断类文档、官方资料、工单及服务请求。为了进一步避免敏感信息的透出，我们对数据进行了清洗和改造。



图 5 数据清洗与改造

（4）专有云运维大模型的数据标注方法

语料的产生都是经过了专业技术人员的标注审核，我们把知识标注工作与日常工作做了一个

融合，利用专家队伍把知识标注贯穿日常生产的全过程，持续提升大模型准确度，这样能够最大化知识更新迭代的速度，也节约了标准的成本。

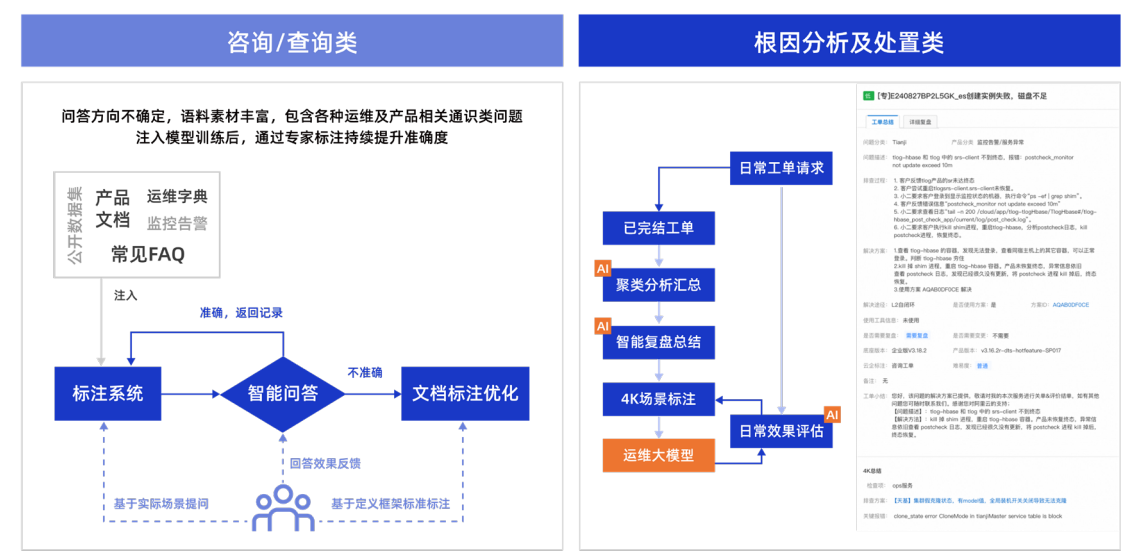


图 6 数据标注方法

（5）专有云运维智能化整体架构

我们在飞天企业版的智算平台之上，基于通义千问结合专有云运维场景打造了运维领域大模型。我们把整个运维大模型整体架构分成了 6 个层次，自下而上说明如下：

- **基础大模型层**：最底层我们依赖 Qwen 基础模型。
- **知识工程层**：在模型之上我们构建了专有云知识工程系统，利用专有云知识库和日常记录进行专有云领域知识的灌注。
- **算法能力层**：在算法能力上我们利用了提示词工程，以及聚类算法来做标注归类打标以及对齐训练。
- **运维大模型层**：我们目前是两个层面，一个是智能问答，一个是各类运维智能体，我们是期望通过智能体深度与平台互动，帮助用户降低提问难度，解决各类运维问题分析和处置预案推荐，后期我们也期望与现有运维工具结合做运维处置类工作。
- **能力开放层**：我们也会把运维大模型的能力进行开放，提供自动化知识挖掘、知识安全审核、

场景与分类管理、运维变更平台集成及开放能力集成。

- **应用场景层**：运维大模型的能力可以供各种应用场景（如：运维小智）调用，进一步加速运维模式创新。



图 7 专有云运维智能化整体架构

3. 运维大模型的应用场景

（1）专有云智能工单

目前，我们把大模型率先在工单场景上进行了应用，70% 的问题场景可精准化推荐，使工单处置时长整体降低 40%。我们在问答生成 & 智能推荐，一键快速总结反馈、智能复盘沉淀上都用到了大模型的能力。



图 8 专有云智能工单举例

• 问答生成 & 智能推荐

随时唤起智能助手，进行智能化总结，补齐问题内容，让大模型更懂问题并做出有效推荐。

• 一键快速总结反馈

问题处置完成后，快速总结问题处置过程，明确问题原因和解决方案。

• 智能复盘沉淀

对问题进行内部复盘，补充现象和根因细节，形成 4K 模型语料，完成专家标注。

（2）运维知识库

引入运维大模型能力，围绕运维人员在专有云使用过程中的各个重点环节，为平台运维提供常用运维命令、云产品介绍、云产品组件构成、告警问题排查方案、运维变更手册等服务检索能力，帮助运维人员更快地定位问题和解决问题，提升云平台的运行效率和稳定性。

• 运维命令检索

通过强大的智能分析和学习能力，能够高效地提供一系列常用的运维命令。这些命令涵盖了系统监控、故障排查、服务配置、性能优化等多个方面，极大地方便了运维人员在日常工作中迅速找到所需操作。

• 云产品知识智能检索

通过智能化的检索系统，能够对云产品相关的知识进行高效的整理与索引。运维人员可以快速、准确地获取关于各类组件的构建方式及其具体作用的详细信息，不仅提高了信息检索的效率，还助力运维人员更加深入地了解云产品的各个组成部分。

• 排查与变更方案推荐

提供了系统化的日常变更方案和告警排查定位方案，这一推荐机制不仅能够帮助运维人员迅速理解和掌握变更的具体细节，还能有效指导他们在遇到警报时进行准确的排查与定位。



图 9 专有云运维知识库举例

（3）智能容量

在资源成本优化过程中，很多客户在上云前后的各个期间，常常面临着业务计费、监视资源使用情况及成本分析的困扰，需要收集与容量相关的各种数据，对收集的数据进行清洗、分析、统计。我们通过容量智能算法和历史峰值算法，能够实时展示各维度的容量情况，从而实现对云资源的科学调配与监控，提高了资源运营的效率，实现更智能化的资源管理和成本优化。



图 10 智能容量举例

(4) 智能基线

在监控领域，目前监控手段通常依赖“人为”的固定阈值来判断异常，比如“特殊日”双十一会带来业务指标较大波动，无法适应不断变化的环境，容易出现误报和漏报的情况。“DT”时代，激发监控的生产力，通过“AI”的智能算法对大量的历史数据进行学习，能够实时分析数据流并动态调整“阈值”，基于多指标联合检测和黄金指标智能检测快速发现问题，既有效压制告警又可避免漏报，为业务异常的识别提供了更客观的手段。



图 11 智能基线举例

(5) 智能变更

在日常运维变更领域，运维人员在变更过程中往往需要到各个监控平台查看指标或登录到机器上查看日志来确认应用是否正常启动，从而决定是否暂停变更或回滚。我们以分批发布场景为例，通过对历史变更数据的离线分析和对发布过程中的业务监控、基础监控、应用日志数据进行实时机器学习，发布过程中发现有异常指标系统会自动暂停发布，并提示异常指标和拦截原因，运维人员确认有问题则可以选择关闭或回滚，没有问题则选择继续发布。通过变更过程的无人值守的能力，我们一方面可以降低现在变更过程中人肉去盯屏的成本，另一方面还能有效减少人为因素导致的故障漏识别。

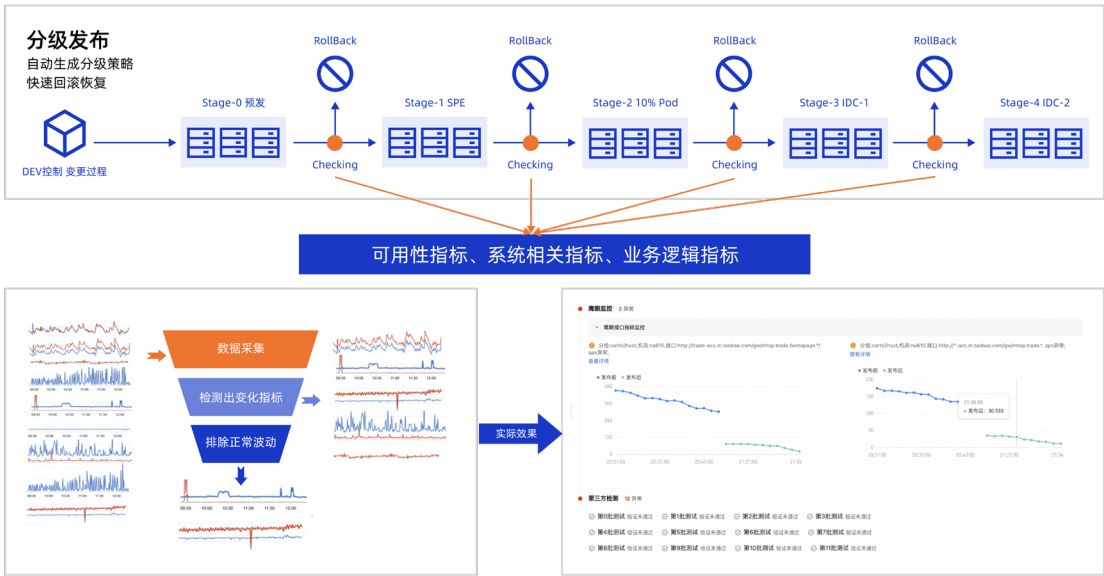


图 12 智能变更举例

4. 未来展望

目前，专有云运维大模型还处于起步阶段。接下来，我们将打造一个更强大的专有云运维平台“云维小智”，以统一工作台为核心，实时集成专有云平台数据，扩展智能小工具，并结合专家经验和 AI 技术高效协同完成运维任务。通过 AIOps 重构优化现有工具，引入智能体进行自动化分析和处理，提供便捷高效的运维体验。同时，开放专有云运维大模型能力，助力客户实现“云+应用”业务一体化的智能运维能力，帮助客户实现快速运维创新，做 AI 时代最智能的运维大模型。

结束语 /

阿里云专有云服务团队是阿里云为确保客户云平台稳定、业务连续运行而打造的核心支撑团队。

团队自从诞生之日起，就被赋予了“服务”于客户的基因，立志成为客户身边的云计算专家，帮助广大阿里云企业客户“建好云、管好云和用好云”，基于云构建更加稳定可靠的业务系统，提升业务稳定性，助力业务持续发展。

从服务的视角，我们致力于成为一个以技术为基础，助力业务系统实现高可用的工程师团队。以专业和体系的技术服务为客户提供从专有云平台规划、部署、运维到云上运营与优化的一站式服务解决方案。

团队深刻践行着阿里巴巴集团 to B 企业服务的初心。我们将秉承：客户第一、求真务实、追求卓越的服务理念陪伴客户和伙伴一起成长。

面向 AI 时代，我们将继续保持对“云 +AI”技术的专注和投入，为政企打造 AI 时代最开放的云做好服务支撑。

最后，我们真挚希望本书能够为广大政企客户及相关从业者在云化转型的道路上提供一些思路 and 参考，助力大家基于云计算技术实现更多的业务升级转型与持续发展。

附录 /

附录一： 阿里云基础产品矩阵

英文名	中文名	产品描述
ASR	业务连续性管理平台	ASR（apsara stack resilience），即业务连续管理平台，专注于云平台基础产品的容灾切换。ASR 是为了在灾难发生时，快速地实现容灾切换，尽可能地降低 RTO，而开发的基于图形交互的切换工具。支持同城容灾和异地容灾。
MSHA	多活容灾平台	多活容灾 MSHA（multi-site high availability）是在阿里巴巴电商业务环境演进出来的多活容灾商业化产品。横向支持容灾架构的上线、运维、演练、切流，升级到下线。纵向支持业务流量的全链路管理，从流量接入到服务化调用再到异步化消息，最终完成数据落库。
DTS	数据传输服务	数据传输服务DTS(data transmission service)支持关系型数据库、NoSQL、大数据(OLAP)等数据源，集数据迁移、订阅及实时同步功能于一体，能够解决公共云、混合云场景下，远距离、秒级异步数据传输难题。其底层基础设施采用阿里双 11 异地多活架构，为数千下游应用提供实时数据流。
OceanBase	金融级分布式数据库	阿里云云数据库 OceanBase 是阿里和蚂蚁金服 100% 自主研发的金融级分布式关系数据库，实现了金融级高可用。在金融行业创造了“三地五中心”城市级故障自动无损容灾新标准，创造了 6100 万次 / 秒处理峰值的业内纪录。支持在线水平扩展，在功能、稳定性、可扩展性和性能等方面都经历了严格的检验。
RDS	云数据库	阿里云关系型数据库 RDS（relational database service）是一种稳定可靠、可弹性伸缩的在线数据库服务。基于阿里云分布式文件系统和 SSD 盘高性能存储，RDS 支持 MySQL、SQL Server、PostgreSQL 和 MariaDB TX 引擎，并且提供了容灾、备份、恢复、监控、迁移等方面的全套解决方案。
OSS	对象存储	阿里云对象存储 OSS（object storage service）是一款海量、安全、低成本、高可靠的云存储服务，提供 99.999999999%(12 个 9) 的数据持久性，99.995% 的数据可用性。多种存储类型供选择，全面优化存储成本。
ACK	容器服务	阿里云容器服务 Kubernetes 版（Alibaba Cloud Container Service for Kubernetes，简称 ACK）是阿里云提供的托管式 Kubernetes 服务，帮助企业快速部署和管理容器化应用。ACK 基于开源的 Kubernetes 项目构建，并提供了企业级的增强功能，旨在简化容器集群的创建、配置和管理，同时确保高可用性和安全性。

ACR	容器镜像服务	阿里云容器镜像服务（Alibaba Cloud Container Registry，简称 ACR）是阿里云提供的安全、高效、稳定的容器镜像托管服务。它帮助企业轻松管理 Docker 镜像和 Helm Chart 等资源,支持私有化部署，并提供了丰富的功能来满足不同场景下的需求。
AOP	应用运维平台	阿里云 AOP（Application Operation Platform，简称 AOP）是阿里云提供的云 + 应用一体化运维平台，通过和专有云内部多款监控产品如：CMS、ARMS、SLS 等集成，构建“云 + 应用”全生命周期应用管理，构建以应用为视角的新一代云管能力，助力客户用好云。
OTS(Tablestore)	表格存储	表格存储 Tablestore 是阿里云自研的面向海量结构化数据存储的 Serverless 分布式数据库，为时序监控轨迹、历史订单数据、日志信息、元数据等海量数据提供存储、访问、同步、检索、计算等能力。
MQ	消息队列	消息队列 MQ 版是阿里云构建的低延迟、高并发、高可用、高可靠的分布式消息中间件。可为分布式应用系统提供异步解耦和削峰填谷的能力，并支持海量消息堆积、高吞吐、可靠重试等特性。
EDAS	企业级分布式应用服务	EDAS（enterprise distributed application service）是一个应用托管和微服务管理的 PaaS 平台，提供应用开发、部署、监控、运维等全栈式解决方案，同时支持 Dubbo、Spring Cloud 等微服务运行环境，助力您的各类应用轻松上云。
PTS	性能测试服务	阿里云 PTS（Performance Testing Service，性能测试服务）是阿里云提供的一个全面的云端性能测试平台，旨在帮助企业快速、高效地进行 Web 应用、移动应用、API 接口等的性能测试。通过模拟大量用户访问，PTS 能够评估系统的负载承受能力、响应速度和稳定性，帮助开发者发现并解决潜在的性能瓶颈。
ARMS	应用实时监控服务	阿里云 ARMS（Application Real-Time Monitoring Service，应用实时监控服务）是阿里云提供的一款面向应用的全链路性能监控和诊断工具。它旨在帮助企业快速定位并解决应用性能问题，提升用户体验，并确保业务的稳定运行。
SchedulerX	分布式任务调度	SchedulerX 是阿里云提供的一个分布式任务调度平台，专为大规模分布式环境设计，旨在帮助企业解决定时任务调度和分布式任务执行中的复杂问题。它支持多种类型的调度任务，包括但不限于周期性任务、固定延迟任务、Cron 表达式任务等，并提供了丰富的特性和功能来确保任务的可靠执行。

附录 /

附录二： 缩略语

名词术语	说明
灾难	由于人为或自然的原因，造成信息系统严重故障或瘫痪，使信息系统支持的业务功能停顿或服务水平不可接受、达到特定的时间的突发事件。通常导致信息系统需要切换到灾难备份中心运行。
BCM	business continuity management，业务连续性管理。为保护组组的利益、声誉、品牌和价值创造活动，找出对组织有潜在影响的威胁，提供建设组织有效反应恢复能力的框架的整体管理过程。
DRP	disaster recovery planing，灾难恢复规划。为了减少灾难带来的损失和保证信息系统所支持的关键业务功能在灾难发生后能及时恢复和继续运作所做的事前计划和安排。
RPO	recovery point objective，恢复点目标。灾难发生后，系统和数据必须恢复到时间点的要求。
RTO	recovery time objective，恢复时间目标。灾难发生后，信息系统或业务功能从停顿到必须恢复的时间要求。
备份	为了灾难恢复而对数据、数据处理系统、网络系统、基础设施、专业技术支持能力和运行管理能力进行备份的过程。
容灾	为了将信息系统从灾难造成的不可运行状态或不可接受状态恢复到可正常运行状态，并将其支持的业务功能，从灾难造成的不正常状态恢复到可接受的状态而设计的活动和流程。
IaaS	infrastructure as a service，基础设施即服务。云服务使用者提供云能力类型中的基础设施能力类型的 1 种云服务类型。
PaaS	platform as a service，平台即服务。云服务使用者提供云能力类型中的平台能力类型的 1 种云服务类别。
SaaS	software as a service，软件即服务。云服务使用者提供云能力类型中的应用能力类型的 1 种云服务类型。

CAP 理论	也称为布鲁尔定理 (Brewer's Theorem)，是由加州大学伯克利分校的计算机科学家埃里克·布鲁尔 (Eric Brewer) 在 2000 年提出的。该理论指出，在分布式系统中，无法同时完全实现以下三个特性： a. 一致性 (Consistency, C)： - 每次读取操作都能获取到最新的写入结果，或者返回失败。 - 从用户的角度看，所有节点在同一时间看到相同的数据。 b. 可用性 (Availability, A)： - 系统中的每个请求都会收到一个响应，而不会被拒绝或无限期等待。 - 即使部分节点故障，系统仍能处理客户端请求并返回结果。 c. 分区容忍性 (Partition Tolerance, P)： - 系统能够在网络分区的情况下继续运行，即使某些节点之间的通信中断。 - 分区是指网络故障导致系统中的一部分节点无法与其他部分通信。
BASE 理论	是对 CAP 理论的一种补充和扩展，它提供了一种设计分布式系统时的替代思路。BASE 是“Basically Available (基本可用)、Soft state (软状态)、Eventually consistent (最终一致性)”这三个短语的缩写。与强调强一致性的 ACID 事务模型不同，BASE 理论更关注系统的高可用性和灵活性，允许在某些情况下牺牲强一致性以换取更好的性能和可扩展性。
表组 (Table group)	一系列可发生关联的表的集合，表组可分为事实表组和维度表组。
拆分表	拆分表指根据指定拆分规则将数据按记录 (行) 分布在多个物理表或分区中，每个物理表具有相同数据结构和部分数据。
拆分规则 (Sharding rule)	水平拆分过程中将逻辑数据库表拆分为多个物理分表的规则。
拆分键 (Sharding key)	水平拆分过程中，生成拆分规则的数据库字段称为拆分键。
DBLink	跨数据库查询中的 DBLink，是一个指向用户的任意数据库实例的虚拟连接，是数据库实例的别名。DBLink 和数据库实例一一对应，对于 MySQL 来说，对应的就是 MySQL 数据库的 IP 和端口号。
大规模并行处理 (Massively Parallel Processing)	一种分布式 Shared Nothing 计算架构，支持多个无共享的节点，执行全并行计算，计算性能随节点增加而线性提升。ADB PG 实例即为 MPP 集群架构，由多个计算节点组成。
分布式事务 (Distributed Transaction)	事务的发起者、资源及资源管理器和事务协调者分别位于不同的分布式系统的不同节点之上。
分片 (Shard)	一种数据库架构，它将海量数据分散存储到多台服务器上，每台服务器只存储和处理一部分数据。在这种架构下，每台服务器被称为一个分片 (Shard)。
分区裁剪 (Partition Pruning)	指数据库根据查询条件，避免访问无关分区的优化过程。分区裁剪可以分为静态和动态两种类型。
联机分析处理 (On-Line Analytical Processing)	数据仓库系统的主要应用，支持复杂的分析操作，侧重决策支持，并且提供直观易懂的查询结果。

联机事务处理（On-Line Transaction Processing）	也称为面向交易的处理过程，其基本特征是前台接收的用户数据可以立即传送到计算中心进行处理，并在很短的时间内给出处理结果，是对用户操作快速响应的方式之一。
Replication 架构	Replication（复制）是一种在计算机系统中用于确保数据存储的安全性并在多个地点或设备中可用的架构或设计模式。能够提高系统的可用性和可靠性，提供更高的数据安全性，支持快速访问数据。 复制架构包括以下几种类型： - 主从复制：一个中央“主”服务器接收所有写入和更新操作，然后将它们传播到若干个“从”服务器中，以存储数据的副本。这样多个客户端可以从从服务器中读取数据，提高了性能和可用性，但是所有的写操作都必须发送到主服务器进行处理。 - 点对点复制：系统中的所有节点都是对等的，可以作为客户端和服务端，发送和接收更新并存储数据的副本。此架构实现了去中心化，系统也更加灵活，但管理和维护可能更复杂。 - 多主复制：多个服务器都能作为主服务器，接收客户端的写入和更新操作，提高了可用性和数据访问速度。
Schema	通常来说，指的是数据库对象（如表、视图、索引等）的模式；在 OceanBase 数据库服务端，也指所有数据库对象模式的集合。
数据分布（Data distribution）	MPP 架构下，表的数据按分区键存储在不同数据分区上，是全并行计算中的一个计算执行和存储单元。常见的分布方式有哈希分布，随机分布，复制分布。
数据库（Database）	数据库是按数据结构来组织、存储和管理数据的仓库。数据库下包括若干表、索引，以及数据库对象的元数据信息。
数据倾斜（Data Skew）	数据中对于某一个或某几个值出现的次数特别多，占据了对应数据较大的比例，在分布式执行过程中，数据倾斜会引起执行的长尾，导致被分配处理这些值的执行线程会用更多的时间完成执行。
数据生命周期（Time To Live）	数据表的一个属性，即数据的存活时间，单位为秒或毫秒，常见于时序类或数据分析类数据产品中，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。
事务（Transaction）	作为单个逻辑工作单元执行的一系列操作，要么完全执行，要么完全不执行。
事务分支（Transaction Branch）	一个分布式事务可能包含多个分支，只有当所有的分支全部成功时，分布式事务才能成功，一个分支的失败将导致分布式事务的回滚。在 GTS 框架下，分支可能是一个分库上执行的 SQL 语句，或是一个手动模式分支。

时序数据（time series data）	基于稳定频率持续产生的一系列指标监测数据。例如，监测某城市的空气质量时，每秒采集一个二氧化硫浓度的值而产生的一系列数据。
维度逻辑表（Logical dimension table）	描述对应维度属性的数据仓库模型，包含逻辑结构、存储方式、组织方式和操作方式等内容。创建维度逻辑表即完成公共的对象明细数据筛选，从而便于提取业务中对象的相关明细数据。
一云多 Region	是阿里云特有的架构，其中只有一个是中心 Region（部署各云产品的中心管控和工作节点集群），其他是单元 Region（部署各云产品的单元化管控和工作节点集群），每个工作节点集群只能属于 1 个 AZ/Region。一云多 Region 内，单一用户账户权限（end user）具备全域云服务实例开通管理，全域资源计量计费，全域应用发布、MaxCompute 跨域计算、统一云管体验等能力。
优化器（Optimizer）	优化器是决定用户查询执行计划的核心模块。通过访问相关数据的统计信息，结合 OceanBase 数据库内置的规则与代价模型，优化器为用户查询生成最佳的执行计划。
HDFS	HDFS（Hadoop Distributed File System）是 Apache Hadoop 生态系统中的核心组件之一，是一个分布式文件系统，专为存储和处理大规模数据集而设计。



作为客户身边的稳定性工程师，阿里云专有云服务团队始终坚持“客户第一、求真务实、追求卓越”的服务理念。多年来，团队有幸陪伴政企客户在云化道路上共同成长，积累了丰富的上云、用云和稳定性建设经验。

编写本刊的初心，正是希望将这些宝贵的方法论、解决方案和最佳实践回馈给政企客户，助力政企更好地“建好云、管好云、用好云”，构建更加稳定可靠的业务支撑系统，做政企数智创新的同行者。

云启录



政企用云最佳实践