降本之源

云原生成本管理白皮书

腾讯云

CAICT 中国信通院

降本之源

云原生成本管理白皮书

编写单位

腾讯云计算（北京）有限责任公司

中国信息通信研究院

作业帮教育科技（北京）有限公司

腾讯云原生 让用云更简单 更有效

目录 CONTENTS

前言 01

一、背景介绍 02

二、云原生成本管理模型 03

2.1 资源利用率现状 04

2.2 云原生成本管理模型 08

2.2.1 成本洞察 08

2.2.2 成本优化 08

2.2.3 成本运营 09

三、最佳实践 10

3.1 成本洞察 10

3.1.1 成本采集及资源追踪 10

3.1.2 资源使用可视化 10

3.1.3 费用可视化 11

3.1.4 成本分配 11

3.1.5 账单管理 13

3.2 成本优化 14

3.2.1 设置合理的资源请求和限制 14

3.2.2 动态调度 17

3.2.3 多维度弹性 19

3.2.4 在离线混部 22

3.2.5 GPU 共享 25

3.2.6 优化矩阵 27

3.3 成本运营 27

3.3.1 建立成本优化团队 28

3.3.2 推动成本意识文化 28

3.3.3 数据驱动成本优化 29

3.3.4 在流程中考虑成本 29

3.3.5 量化成本优化交付的业务价值 29

四、总结 31

五、企业客户降本案例 32

前言

云原生使组织能够在现代云环境（例如公共云、私有云和混合云）中构建和运行可扩展的应用程序， 更快地创新并使企业能够更敏捷地对市场作出反应。

凭借弹性计算、自动扩展、计量计费和按使用付费模型等功能，云原生计算可帮助组织摆脱昂贵的永远在线的基础架构，并将这些节省用于新功能开发。 冗余、容错、松散耦合的服务和云原生架构自动化的自动恢复导致的弹性增加和停机时间减少也可能导致间接成本节约。

使用容器和微服务构建应用程序的一个巧妙之处在于，云原生应用程序使开发人员可以更轻松地访问和重用为早期项目创建的组件。 这有几个云原生的优点，可以降低开发成本并制作更好的应用程序：

- **降低开发过程的复杂性：**开发人员可以将更多时间花在项目的细节上，而不是构建通用框架上。它还允许在更短的时间内开发更复杂的应用程序。
- **缩短上市时间：**更快的交付意味着更快的客户更满意，这也意味着抓住时间敏感机会的潜力。
- **简化测试：**经过审查的微服务出现的问题更少，减轻了管道后期的负载。当开发人员知道服务有效时，他们所要做的就是设计兼容性。
- **模块化设计：**模块化设计使外观和功能的标准化更容易。拥有多个应用程序或服务的公司可以利用这一点来降低客户的学习曲线。

云原生是应用程序开发的未来，具有巨大的业务影响潜力——能够快速有效地将想法转化为生产。云原生已赋能许多其他技术，例如 边缘计算、人工智能、区块链、5G 应用等。

抓住云原生变得特别重要, 你准备好了以下了没:

1. 创建一个更加面向服务的组织。而不是传统的基于功能的结构，围绕特定的服务或能力组织您的团队。

2. 使用现代和最新的架构。云原生意味着使用微服务和反应速度更快的架构类型。

3. 重新组织您的架构以跟上云原生开发的步伐。业务应该能够以与技术人员相同的速度足够快地生成需求。

4. 大多数云原生关键基础设施都是开源的，例如 Kubernetes，你知道如何参与开源社区以了解云原生的最新发展，甚至成为贡献者的一份子吗?

Keith Chan
CNCF 中国区总监兼 Linux 基金会亚太区策略规划总监、腾讯云 TVP

一、背景介绍

数字经济已成为我国经济增长的重要引擎，云计算从部分企业数字化转型的载体，转变为整个经济社会发展的基石与枢纽。万千企业数字化转型提速换挡，对云计算的使用效能提出了更高的要求，云计算迎来了全新的发展机遇。

相较于传统云计算架构，云原生具备更灵活的资源管理、更敏捷的应用迭代、更高效的模块协同以及更稳定的业务保障能力，云原生带动技术架构、应用效能、云化效益的全方位提升，为企业数字化转型提供了有效的实践工具和方法论。

根据中国信息通信研究院调查数据显示，云原生已进入黄金发展期，其核心技术开始在大规模生产环境中深入应用，金融、政府、制造、电信、医疗等行业的云原生用户占比较 2020 年均有显著提升，云原生化开始从业内的头部企业逐步下沉到中小企业，从领先企业的尝鲜变为主流企业的必备。同时，云原生技术给企业带来的价值中，提升资源利用率节约成本连续两年排名第一，2021 年已有九成用户认可该项价值，排名前五的另外四项价值分别是：提升弹性效率、提升交付效率、简化运维系统以及便于现有系统的功能扩展。云原生已成为企业基于云实现降本增效的最佳实践。

但随着企业用云程度不断加深，云上支出浪费严重，云原生平台自身的成本治理成为企业上云突出诉求。弹性按需是云原生的资源利用优势，但如果资源配置策略设置不合理可能会导致资源的浪费；云原生资源利用的计量方式如果不够灵活，会使得企业难以准确调控用云成本；云边缘、混合多云、云数智融合等模式在帮助提升工作效率、实现应用灵活部署的同时也带来了异构资源管理的新挑战。因此，企业在应用云原生架构之后，需要考虑如何管理、优化和使用云原生服务，如何通过降低云原生成本，进一步提升业务的数字化转型效果。云原生成本管理面临问题难定位、路径难选择、成效难持续三大挑战，资源成本优化是云原生降本的关键路径。

云原生成本管理一是需要掌握成本支出态势，准确定位资源浪费的根源；二是需要选择适合平台架构和业务特点的资源优化策略，并规避因资源优化导致对业务稳定性的影响；三是实现成本优化后还需要保证其持久性。资源成本优化从基于账单优化的成本可视、基于资源优化的成本节约以及基于模式优化的成本运营从三方面帮助降低云原生平台成本。

《云原生成本管理白皮书》将基于中国信息通信研究院与腾讯云对行业发展趋势和企业客户诉求的准确把握、对云原生优化的技术研究与实践经验，提出一套体系化的云原生成本优化方法论和最佳实践路径，结合行业优秀案例，帮助企业改善用云成本充分发挥云原生的效能和价值，为企业的数字化转型提供可靠的保障。

二、云原生成本管理模型

云原生并非一项单纯的技术，更是一种思想，是技术、企业管理方法的集合，云原生追求的是在包括公有云、私有云、混合云等动态环境中构建和运行规模化应用的能力，追求的是业务持续平滑的变更能力。

Kubernetes 是云原生技术栈的核心，是众多云原生项目的粘合剂。Kubernetes 遵循声明式系统原则，将其管控的对象都抽象成标准 API，并通过多种控制器完成云原生平台的高度自动化。比如云用户可以通过定义 Pod 对象，并指定需要运行的容器镜像，以及容器所需的 CPU、内存等计算资源。该对象被提交至 Kubernetes 以后，Kubernetes 会依据用户请求运行容器进程，并按需求确保该应用进程的资源配额。

这使得应用可以以较低成本接入到 Kubernetes 平台中来，并依靠 Kubernetes 自动化机制实现低运维乃至免运维。基于监控平台收集的数据，Kubernetes 可根据应用的实时指标数据，预测应用资源用量，并通过自动化横向或纵向扩缩容能力，及时调整应用的副本数量以及资源用量，及时回收空闲资源，提升资源使用率。

提升资源利用率是云原生技术栈的核心目标之一，资源利用率的提升意味着以更少的计算节点用承载更多的应用实例，极大的降低云用户的资源开销，也契合国家节能减排的政策号召。

2.1 资源利用率现状

根据中国信息通信研究院调查数据显示，云原生技术给企业带来的价中，提升资源利用率以节约成本连续两年排名第一，2021 年，已有九成用户认可该项价值。

选项	2020年	2021年
提升资源利用率以节省成本	76%	90.59%
提升弹性伸缩效率	63%	76.98%
提升交付效率	38%	66.83%
简化运维系统	30%	67.57%
开放架构方便现有系统上的功能扩展	25%	48.02%

如图1所示，《2020 年 CNCF 中国云原生调查》报告中指出，生产系统中使用 Kubernetes 的比例已从 2019 年的 72% 增长到了 82%，越来越多的企业在云上使用基础设施资源、通过 Kubernetes 平台来管理应用，Kubernetes 已经无处不在。

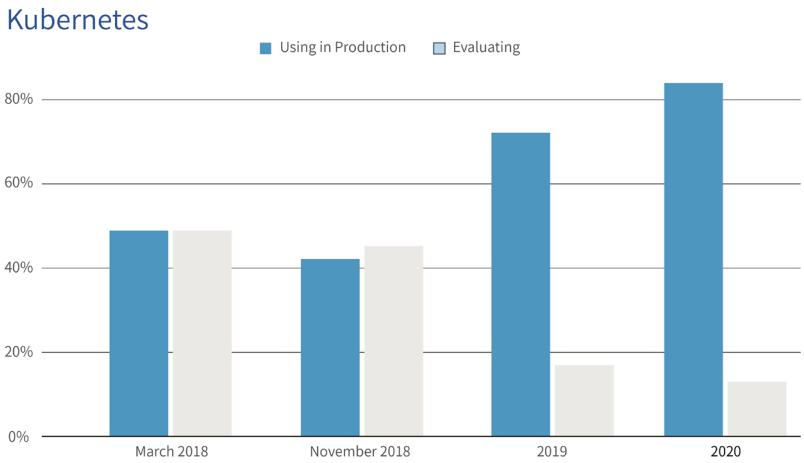

图1　Kubernetes 近年部署率

但是，随着企业用云程度加深，越来越多的应用迁移到云原生架构上，原本天然具备降本增效特点的云原生架构如果资源配置不当也会引起大量云上资源闲置、云支出浪费。2021 年 CNCF《FinOps Kubernetes Report》调研报告显示，迁移至 Kubernetes 平台后，68% 的受访者表示所在企业计算资源成本有所增加，36% 的受访者表示成本飙升超过 20%，调查结果如图 2 所示。这都说明即使是资源利用率更高的云原生架构也需要合理的资源成本管理。

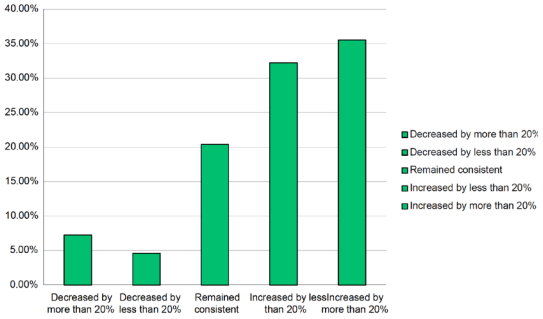

图2　迁移至 Kubernetes 平台后成本变化调查结果

Kubernetes 集群的成本主要集中在工作节点的运行成本，资源利用率过低是计算成本高企的最主要原因。在 Kubernetes 集群中，资源利用率体现为运行的所有 Pod 消耗的资源与集群上可用资源总量的比率，据麦肯锡早期报告，全球服务器资源利用率不到 6%，资源利用率低下导致了大量的资源浪费。

针对相同统计口径，腾讯云对 1000 多云客户进行了资源利用情况分析，抽样超过一万计算节点，实际使用率如图 3 所示：

42% 的节点资源利用率低于 10%	72% 的节点资源利用率低于 20%
15% 的节点资源利用率在 20%-30%	只有不到13%的客户点利用率大于 30%

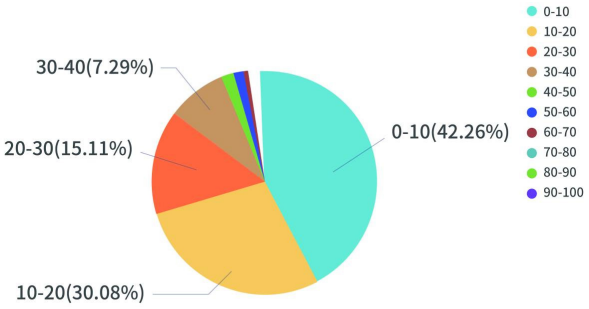

图3　资源利用率调查

将统计数据汇总，我们可以看到，计算节点的平均资源利用率在 10% 左右，这意味着云用户 90% 的计算资源成本是闲置的，这造成了高企的云成本，形成了极大的浪费。

针对此种状况，下文分析了主要原因：

资源预留过多，普遍存在 50% 以上的浪费

Kubernetes Pod 的资源需求（Resource Request）字段用于管理容器对 CPU 和内存资源预留，保证容器至少可以达到的资源量，该部分资源不能被其他容器抢占。如何设置资源需求是一个难题，若设置过小，当业务负载变高时，业务所需的计算资源无法被确保，可能会造成计算变慢、延迟过高等影响业务指标的情况。

为避免此情况发生的可能性，用户通常会将 Request 设置得很高，以保证服务的可靠性。但实际上，业务在大多数时段时负载不会很高。以 CPU 为例，图4是某个实际业务场景下容器的资源预留（Request）和实际使用量（CPU_Usage）关系图：资源预留远大于实际使用量，两者之间差值所对应的资源不能被其他负载使用，因此 Request 设置过大势必会造成较大的资源浪费。

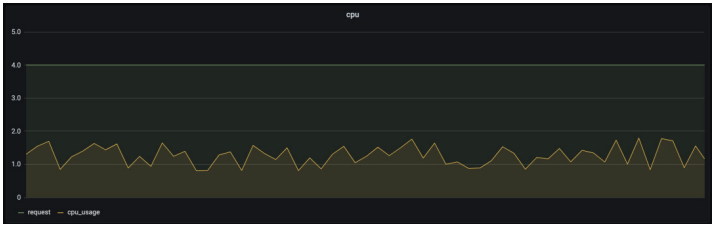

图4　CPU 资源利用率

业务资源使用率波峰波谷现象普遍，低峰时资源浪费严重

在线业务流量通常有较明显的波动周期，且波谷时间常大于波峰时间，波谷时期的资源有较多闲置，这也成为了资源利用率较低的另一主因。例如公交系统通常在白天负载增加，夜晚负载减少；游戏业务通常在周五晚上开始出现波峰，在周日晚开始出现波谷。如图 5 所示，同一业务在不同的时间段对 CPU 资源的用量不同，若用户以固定资源需求申请 CPU 资源，当业务负载较低时，CPU 资源利用率就会很低。

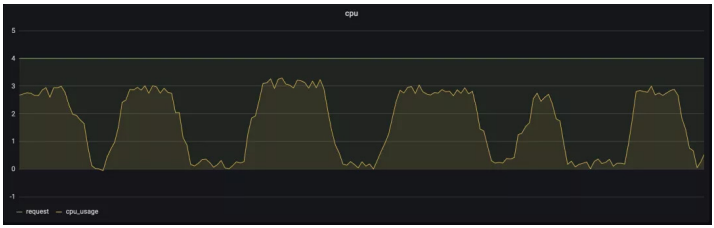

图5　资源利用率的波动现象

不同类型的业务，对资源的需求不同

计算作业依据其提供服务的特性和业务目的可分为在线业务和离线业务两类：在线业务通常白天负载较高，且对时延要求较高，必须优先调度和运行；而离线的计算型业务通常对运行时段和时延要求相对较低，可以在在线业务负载波谷时段运行。此外，有些业务属于计算密集型，对 CPU 资源消耗较多，而有些业务属于内存密集型，对内存消耗较多。如图 6 所示，将负载峰值在不同时段的业务，或者将在线和离线作业混部在一起，可有效的以离线作业填补在线作业的负载低谷时段，进而提升资源利用率。将需求不同资源的作业有效的部署在相同节点，可充分利用各种资源，提升总体资源利用率。

图6　负载峰值在不同时段的业务混部

针对资源利用率低的现状，我们对数十个客户进行了访谈，了解到他们在成本管理时的挑战：

- 相比于传统云服务器，Kubernetes 的动态性为资源计量和成本分摊变得更复杂，Kubernetes 作为一个开放式平台，鼓励资源共享，但同时缺少有效的成本观测和分配机制。多个容器应用可以被动态调度在同一个计算节点上，简单将底层云资源与应用一一映射并评估成本的传统手段已经失效。
- 如何发现和观测成本管理上的问题，比如针对资源闲置、应用闲置、以及前述资源利用率过低等问题，如何能快速有效的发现并分析。
- 定位到问题后，如何制定优化方案，这些方案会不会对现有的架构带来挑战，如何在权衡成本、性能、稳定性、业务价值后作出最佳决策，如图 7 所示。

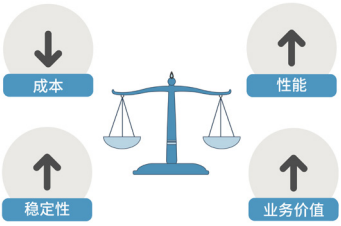

图7　优化方案的决策平衡

- 部分客户做完成本优化项目后，短期内确实节省了很多成本。但是随着时间推移，业务的变迁，旧的成本优化方案可能会失效，云成本又开始上升，这个持续成本优化带来很大挑战。
- 业务开发团队在系统架构、系统设计时，缺乏对成本的考量，这可能使业务上线后的资源开销较大。客户需要从组织、文化、流程等多维度提升对云成本的认知，在系统设计之初就将云成本纳入考量范围之内。

2.2 云原生成本管理模型

基于资源利用率的现状和挑战，我们整理了三阶段云原生成本管理模型，如图 8 所示：

- **成本洞察：**成本采集及资源跟踪、成本可视化、成本分配和账单管理
- **成本优化：**提供可靠、便利、智能的优化方案
- **成本运营：**从组织、文化、流程等方面建设成本运营体系

图8　云原生成本管理模型

2.2.1 成本洞察

成本采集及资源追踪

云环境下的资源使用和成本支出是比较复杂的，团队或组织需要通过一定的工具或方案对各种产品进行定义、定价、计费及统计，并对各类资源的使用率、使用量等指标进行持续追踪，能帮助团队尽可能多地搜集到云成本情况，为后边成本可视化以及成本分配提供数据基础。

成本可视化和成本分配

Kubernetes 的成本分配比传统的云环境面临更多挑战，团队需要定义一致的标签和命名空间来改善分配，对于任何组织、部门及职能团队，基于多维度（如云产品、环境、业务线）的资源和成本的可视化分析，能够帮助团队有效地建立起相应的问责机制，并根据获取到的实时数据快速制定优化方案及措施。

账单管理

云账单的繁冗性和复杂性给用户带来非常大的不便，团队或组织需要对账单进行统一高效的管理，一是根据实际的业务需求，将账单按组织、部门、项目业务维度，年、月周期维度等进行详细化的拆分。二是将各类账单进行统一分析，包括过去一段时间的使用情况、未来使用趋势分析、各部门成本使用情况等方面，并给出合理的规划及更改建议。三是将分析后账单情况按业务部门、周期、自定义等维度定期推送给团队，方便团队及时得到相应的账单。

2.2.2 成本优化

提供可靠、便利、智能的优化方案

基于成本可视化和成本分配等手段，有了数据作为度量依据，团队能够围绕其业务目标及业务场景制定对应的成本优化目标。针对云原生场景，云上资源成本优化不仅仅是对云资源规格、数量的调整，也包含了对业务的架构优化、以及通过弹性能力和资源混部等手段提升资源利用率。此阶段的优化方案包括：

- 正确评估应用容量，设置合适的资源请求
- 通过动态调度解决资源碎片的问题，提高装箱率
- 回收业务波谷时的冗余，通过弹性和混部做到按需使用
- 对于固定资源池，对负载峰值在不同时段的在线应用、在离线应用进行混部，做到分时复用
- 针对 GPU 资源，实现资源的池化和共享

2.2.3 成本运营

从流程、组织、文化等方面建设成本运营体系

云上资源优化并非是通过一系列标准动作后得出的一个终态恒定值。如同追求系统稳定性冗余大量资源，追求极致成本而牺牲业务稳定性同样不可取。业务本身是变化的，适合的系统架构及管理方式同样如此，我们鼓励从组织、文化、流程等方面建设成本运营体系，根据目标持续不断调整和优化。此阶段的优化方案包括：

建立成本优化团队	推动成本优化意识
数据驱动成本优化	在流程中考察成本
量化成本优化交付的业务价值	

三、最佳实践

本章内容将从成本洞察、成本优化、成本运营三个维度展开，结合企业实际落地情况提供成本管理的最佳实践。 帮助企业上云、云原生改造时兼顾成本优化， 助力企业数字化转型。

3.1 成本洞察

成本洞察是成本管理的第一步，现有的 Kubernetes 集群中，只能看到资源的使用情况，而无法分析和观察更具体的成本维度的数据。成本洞察的重点在于从成本的角度观察集群的成本使用情况，主要包含资源使用可视化、费用可视化，以及成本分配。

3.1.1 成本采集及资源追踪

随着云计算的发展，云上各种的产品和资源也是日益复杂，要想实现对成本的可视，那么必须要有专业的方案以及统一的标准对云成本进行定量、定价及采集，并对资源进行持续跟踪，从而做到云成本使用心中有数，以下是实现成本采集及资源追踪具体的要求：

- 对各类公有云成本产品进行收集，并详细记录账单情况。
- 对企业私有云进行资源的定义、定价、计费、统计，然后对私有云成本使用情况账单进行采集
- 对企业资源使用情况进行持续追踪，包括资源使用量、资源使用方式、资源利用率等。

3.1.2 资源使用可视化

得益于云原生的发展，Kubernetes 周边的生态已经相对完善。监控方面，Prometheus 和 Grafana 的组合已经成为了云原生领域监控的标准。成本的管理实际上就是资源的使用管理，如何快速并且准确的识别出资源的分配、消耗、以及浪费，是成本优化的前提条件。因此成本洞察首先要做到的是资源消耗和资源利用率的可视化，这里举几个资源使用情况可视化的例子：

- 展示业务的资源分配、消耗，资源使用效率
- 分析 Top10 资源消耗的业务，分析 Top10 资源利用率低的业务
- 展示资源消耗的走势图
- 展示不同维度的资源使用情况，包含 Kubernetes 概念的维度：例如命名空间、工作负载、Pod 等；也包含用户自定义的维度：例如部门、组织、团体、项目等

3.1.3 费用可视化

在获取资源实际用量后，对接云资源计费系统获取资源单价，即可计算出云资源的实际使用费用。相比资源用量可视化，费用视角的可视化是企业财务和采购角色更关注的视角，他们更加关注费用支出的合理性、费用归属是否明确等等，提升资源利用率的本质就是控制和减少费用支出，从而节约成本。

下面展示费用可视化的日常应用场景：

- 分析集群的下月预估成本，分析每个资源对象的下月预估成本
- 查看集群成本曲线
- 查看业务增长曲线和成本曲线对比，评估成本增长是否过快
- 根据推荐的方案预估的成本节省，为执行动作做成本角度的建议
- 多维度的成本观测：从计算资源维度以及用户自定义的业务维度
- 多周期的观测：支持不同时长的周期定义，如日、周、月、或由用户自定义
- 保存历史重要数据，定期生成报告，回顾和对比

3.1.4 成本分配

Kubernetes 的成本分配比传统的云环境面临更多挑战，团队需要定义一致的标签和命名空间来改善分配，精确分配资源成本是在 Kubernetes 环境中创建成本可见性和实现高效成本利用的首要步骤。以下是成本分配的建议：

明确业务云资源费用类型

企业往往选择公共的 IT 团队来支付云费用，公共的 IT 团队再将成本分配给业务部门或业务应用的所有者。假如在共享 Kubernetes 集群上运行不同业务团队的工作负载，通过云厂商本身的费用账单再分配就变得非常困难。所以，成本分配的第一步是梳理并明确共享的资源有哪些，如计算资源、存储资源、网络资源、云厂商的 PaaS 服务如日志等等。

云原生架构业务标签配置

理清楚共享成本类型之后，就要建立可持续的公平分配的方法。基于云原生架构，用户可以为云上运行的应用添加标签来标识资源的所有者。建立合理的标签模型是成本分配最为关键的一个步骤。

云原生成本分配标签设计原则

成本分配是一件很有挑战的事情，不同的团队关心的成本角度是不一样的，比如：

- 对于开发来说，他们希望能够区分开发环境的成本和生产环境的成本。
- 对于管理着来说，想知道到底是什么业务驱动了成本的上涨，即时发现不合理的商业逻辑。

成本的标记也有多种方式，例如腾讯云提供标签帮助云用户有效分隔不同的云资源，此外还有多层级的 CAM（Cloud Access Management，腾讯云访问管理）账户结构、项目等功能帮助有效分类不同的成本。资源标签是云用户分类云资源的有效手段，不同云用户对标签的使用不尽相同，云服务商对标签的语法限制、字符集和长度等合法性进行校验。这种极大的灵活性会让云用户有相同的疑问，到底该如何定义标签呢？

越早越好

获得较好的成本分配方式的关键是：尽早且始终如一的执行某种分配策略。例如在月初创建一个资源，到月底才打标签，那有一个月的成本没有分配。标签设置的策略也应该尽早进行，也应该尽量保证始终如一的执行下去，否则每次标签的改动实际上都会导致历史数据的丢失。

标签定义原则

该原则应该明确如何定义标签的键，以及标签可能包含的适当值，这些键值对将应用于所有资源。工程师开始开发应用程序之前，就应该将标签定义原则传达给工程师以确保良好的标签覆盖率。虽然许多公司有很多没有被标记的存量业务，但不必担心，梳理业务标签为时未晚。 标签标准在开始制定时越深思熟虑，未来修订的可能性就越小。随着企业在云上的扩展，它将产生数千甚至数百万个资源，每个资源都有自己的标签，更改标记策略意味着团队必须更新所有这些资源。

选择正确数量的标签

标签原则不应该强迫团队应用太多标签。要求过多的标签只会导致对标准的抵触，从而导致不合规。最好只针对所需的核心元数据要求标签。与其根据需要定义所有标签，不如定义可选标签，明确描述团队在选择这样做时应如何实施它们。

易读简化且一致

标签的设计尽量易读，并且要保证简化，满足业务诉求即可。用户应该能直接从标签读懂其含义，但又不能设计得过于冗长。另外，简化的同时也要保证标签一致性，如果一个团队使用 “prod” 的标签值，而另一个团队使用 “production”，这些标签的分组方式会有所不同。

命名标准化

采用标准化命名格式，使后续的 API 集成更加便捷。例如：统一用英文小写，单词之间用下划线隔开。值得注意的是：您可能开始使用“R&D”标记资源，但后来意识到某些服务不支持 & 字符，因此命名尽量考虑使用最通用标准的方式，例如此时应该使用 r_and_d。

最佳实践

如果还是不确定如何设置标签，这里有一个使用标签的最佳实践：

- 一个成本中心/业务标签，它清楚地定义了资源成本应在组织的位置。是属于固定的、集的成本中心？还是单独计算某种业务的具体成本？例如：可以设置类似 cost_allocation=cost_center 或 cost_allocation=business 的标签
- 标识资源属于哪个服务，允许组织区分团队正在运行的服务之间的成本。是订单服务的成本？还是物流服务的成本？ 例如：可以设置类似 business_type=order 或 business_type=logistics 的标签
- 资源所有者标签，用于帮助识别负责资源的个人/团队。例如：可以设置类似 resource_owner=xiaoming 或 resource_owner=wechat 的标签
- 资源名称标签，可以理解成自定义资源名称。例如云厂商定义的云服务器的名字为 CVM，某个实例名为 ins-dfkjdkd，为了方便资源的统计，可以给这些云资源加上一个 cloud_resource=computer 的标签，表明这是个计算资源
- 帮助确定开发、测试和生产之间的成本差异的环境标签。例如：可以设置类似 environment=development 或 environment=production 的标签
- 用于标识资源所属的服务层部分的层标签（例如，前端、网络、后端）。例如：可以设置类似 service_layer=frontend 或 service_layer=backend 的标签

3.1.5 账单管理

云账单是企业获得成本支出的第一手资料，账单的可读性往往影响到企业对成本开支的判断，一个完备的云账单体系应满足各部门精细化的需求，能摸清各种云产品的支出情况，并能及时给到部门或组织反馈。因此对于企业的云账单体系要提高管理水平，帮助企业提高账单分析的效率，使账单更好地服务于企业，以下几个方面是账单管理的落实步骤：

- 对企业账单按云产品维度进行账单的拆分。
- 对企业账单按组织、部门、项目等业务维度进行账单的拆分。
- 对企业账单按包年、包月等周期维度进行账单的拆分。
- 对企业账单按自定义维度进行账单拆分。
- 对企业账单的使用情况进行分析，包括过去支出、浪费情况，未来支出趋势，优化建议等方面。
- 提供业务维度包括组织、部门、项目等方面的账单推送。
- 提供周期维度包括年度、季度、月度等方面的账单推送。

3.2 成本优化

有了完整的成本洞察能力后，即可查看企业整体成本效率。企业进而能够围绕其业务目标及业务场景制定对应的优化方案，本小节将具体介绍成本优化的最佳实践。我们再回顾下前文分析的资源浪费的常见现象：

- 应用设置了过大的资源配额
- 应用波峰波谷明显，波谷时资源浪费严重
- 存在不同类型的业务，对资源要求不同

主要资源优化方向包括：

- 正确评估应用容量，设置合适的资源请求
- 通过动态调度解决资源碎片的问题，提高装箱率
- 回收业务波谷时的冗余，通过弹性做到按需使用
- 对应用进行混部，做到分时复用
- 针对 GPU 资源，实现资源的池化和共享

3.2.1 设置合理的资源请求和限制

设想你是个集群管理员，现在有四个业务部门使用同一个集群，你的责任是保证业务稳定性的前提下，让业务真正做到资源的按需使用。为了有效提升集群整体的资源利用率，这时就需要限制各业务使用资源的上限，以及通过一些默认值防止业务过量使用。

理想情况下，业务应该根据实际情况，设置合理的 Resource Request 和 Limit。Request 用于对资源的占位，表示容器至少可以获得的资源；Limit 用于对资源的限制，表示容器至多可以获得的资源。这样的设置有利于容器的健康运行，资源的充分使用。此外，当多个团队将业务部署到同一个共享集群以后，每个团队都倾向将自己容器的 Resource Request 和 Limit 设置得很高，以保证自己负责的业务的稳定性。如果你使用的是腾讯云容器服务 TKE（Tencent Kubernetes Engine，简称TKE）的控制台，创建负载时会给所有的容器设置如下默认值。

	Request	Limit
CPU（核）	0.25	0.5
CPU（核）	256	1024

为了更细粒度的划分和管理资源，可以在腾讯云容器服务 TKE 上设置命名空间级别的 Resource Quota 以及 Limit Ranges。

使用资源配额（Resource Quota）划分资源

如果你管理的某个集群有四个业务，为了实现业务间的隔离和资源的限制，你可以利用命名空间和资源配额。

命名空间是 Kubernetes 集群里面的一个隔离分区，一个集群里面通常包含多个命名空间，资源配额用于设置命名空间资源的使用配额。例如 Kubernetes 用户可将不同的业务部署在不同的命名空间里，再为不同的命名空间设置不同的资源配额，以限制一个命名空间对集群整体资源的使用量，达到预分配和限制的效果。资源配额主要作用于如下方面，详细信息可查看[2]。

- **计算资源：**所有容器对 CPU 和 内存的 Request 以及 Limit 的总和
- **存储资源：**所有 PVC 的存储资源请求总和
- **对象数量：**PVC/Service/Configmap/Deployment 等资源对象数量的总和

资源配额使用场景

- 给不同的项目/团队/业务分配不同的命名空间，通过设置每个命名空间资源的资源配额以达到资源分配的目的
- 设置一个命名空间的资源使用数量的上限以提高集群的稳定性，防止一个命名空间对资源的过度侵占和消耗

使用Limit Range限制资源

资源需求是 Kubernetes Pod 中容器定义的可选属性，用户可以不为 Pod 设置资源Request 和 Limit，也可以将值设置得很大。集群管理员可以为不同的业务设置不同资源使用默认值以及范围，这可以有效减少业务创建时的工作量同时，限制业务对资源的过度侵占。

Limit Range 适用于一个命名空间下的单个容器，可以防止用户在命名空间内创建对资源申请过小或过大容器；当用户不设置容器资源需求时，Limit Range 可为容器添加默认资源。Limit Ranges 主要作用于如下方面：

- **计算资源：**对所有容器设置 CPU 和内存使用量的范围
- **存储资源：**对所有 PVC 能申请的存储空间的范围
- **比例设置：**控制一种资源 Request 和 Limit 之间比例
- **默认值：**对所有容器设置默认的 Request/Limit，如果容器未指定自己的内存请求和限制，将为它指定默认的内存请求和限制

Limit Range使用场景

- 设置资源使用默认值，以防用户遗漏，也可以避免 QoS 驱逐重要的 Pod
- 将不同特征的业务部署在不同的命名空间中，且为不同命名空间设置不同的Limit Range可在一定程度上提升资源利用率
- 限制容器对资源使用的上下限，保证容器正常运行的情况下，限制其请求过多资源

智能 Request 推荐

即使有了资源配额和 Limit Range 设置，多业务视角的资源调配依然无法做到灵活多变。例如，当用户的业务负载提升，确实需要较多资源时，配额的限制会阻止快速扩容。配额的分配通常需要申请和审批流程，这限制了弹性和创新速度。并且，资源配额限制的是同一个命名空间的应用实例，当一个命名空间里存在多个工作负载时，工作负载之间也会发生资源抢占，导致某些负载资源使用受限。Limit Range 的作用范围也是针对一个命名空间下的所有业务实例，而同一命名空间里可能存在主业务容器也会存在辅助容器，如果都设置成固定的大小值也会造成资源浪费或不够的现象。因此不管是资源配额，还是 Limit Ranges，它们的限制范围都是命名空间级别，粒度较粗。

图 9 是一个实际生产集群的资源使用情况，该集群的 CPU 总量为 1906 核，CPU 的分配率接近 60%，即集群中所有容器的 CPU Request 之和占集群 CPU 总量的 60%（CPU 申请量）。但实际集群中所有容器的 CPU 使用量之和仅为 13.4 核，CPU总体利用率不到 0.8%，CPU 的利用率非常低。出现这种现象的根本原因在于容器的 Request 设置不合理，Request 的设置占集群整体资源的 60%，而实际使用量仅占不到 0.8%。Request 和 Usage 之间存在较大的优化空间。

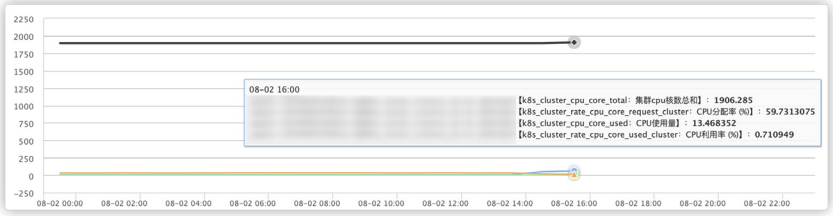

图9　某生产集群的资源分配和实际用量

Request 的设置基于云用户对业务负载的深刻理解，不同业务的 Request 设置各不相同，并且同一业务在不同时间段的 Request 设置也应该随业务负载波动及时调整。业务部门通常习惯将 Request 设置较大，以锁定资源保证业务工作负载的稳定性。但从图 9 可以看出，即使设置使用量十倍的冗余，集群总体核数也仅需要 134 核左右，但实际用户的 Request 为 1906*60%=1143 核。

智能 Request 推荐基于业务的实际资源使用情况，给用户推荐更合理的 Requst 数值，且可以自定义冗余倍数，避免因为流量突发导致的业务不稳定性。配合上弹性集群能力（Cluster Autoscaler），可以在负载低谷期及时缩减集群的整体规模，以达到降本的目的。以图 9 为例，CPU 的实际使用量细节如图 10 所示，没有超过 40 核，即使设置十倍的 Requst 的冗余，此时集群规模仅需 400 核，相较之前的 1906 核，使用智能推荐的 Request 将节省（1906-400）/1906= 79%。

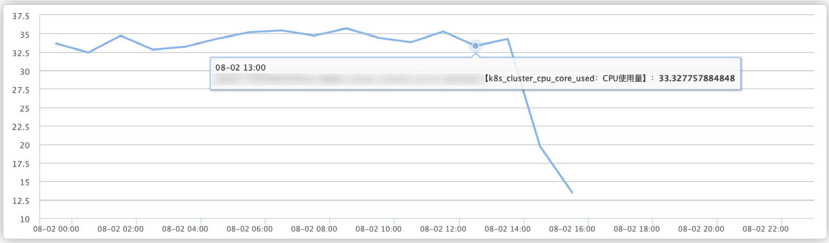

图10　某生产集群的CPU实际用量细节

3.2.2 动态调度

节点亲和性

若某个 CPU 密集型业务实例，被调度到内存密集型的节点上，导致内存密集型的 CPU 被占满，但内存几乎没怎么用，会造成较大的资源浪费。如果能为此类节点设置一个标记，标明该类节点是 CPU 密集型，随后在创建业务负载时也设置一个标记，标明这个负载需要在CPU密集型节点运行。Kubernetes 的调度器会将这个负载调度到 CPU 密集型的节点上，这种寻找最合适的节点的方式，将有效提升资源利用率。

节点亲和性使用场景

节点亲和性非常适合在一个集群中有不同资源需求的工作负载同时运行的场景。比如说，腾讯云的云虚拟机（CVM节点）有 CPU 密集型，也有内存密集型。如果某些业务对 CPU 的需求远大于内存，此时使用普通的 CVM 机器，势必会对内存造成较大浪费。此时可以在集群里添加一批 CPU 密集型的 CVM，并且把这些对 CPU 有较高需求的 Pod 调度到这些 CVM 上，这样可以提升 CVM 资源的整体利用率。同理，还可以在集群中管理异构节点（比如 GPU 机器），在需要 GPU 资源的工作负载中指定需要 GPU 资源的量，调度机制则会帮助你寻找合适的节点去运行这些工作负载。

动态调度（负载感知）

原生的 Kubernetes 调度策略倾向于调度 Pod 到节点剩余资源较多的节点上，比如默认的 LeastRequestedPriority 策略。但是原生调度策略的资源分配是静态的，Request 不能代表资源真实使用情况，因此当业务负载降低时，Kubernetes 调度器的可用资源与集群的实际闲置资源会有较大偏差。如果调度器可以基于节点的实际资源利用率进行调度，将一定程度上解决资源浪费的问题。

腾讯云容器服务TKE自研的动态调度器所做的就是这样的工作。动态调度器的核心原理如下图 11 所示：

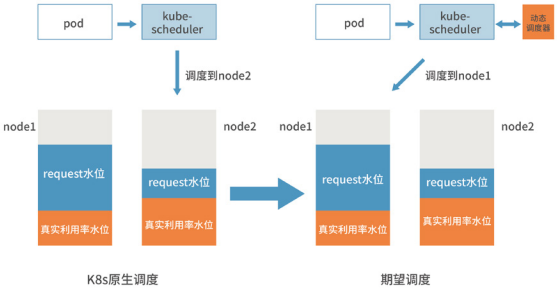

图11　动态调度器

动态调度器的使用场景

除了降低资源浪费，动态调度器还可以很好的缓解集群调度热点的问题。

- 动态调度器会统计过去一段时间调度到节点的 Pod 数目，避免往同一节点上调度过多的 Pod
- 动态调度器支持设置节点负载阈值，在调度阶段过滤掉超过阈值的节点

重调度

在企业的运维工作中，除了成本，系统的稳定性也是十分重要的指标。如何在两者间达到平衡，可能是很多运维人员心中的“痛点”。一方面，为了降低成本，资源利用率当然是越高越好，但是资源利用率达到一定水位后，负载过高极有可能导致节点资源过载而触发的主动驱离或因 CPU 竞争导致的业务指标抖动等问题。为了减小企业成本控制之路上的顾虑，腾讯云容器服务 TKE 还提供了“兜底神器”—— 重调度器 —— 来保障集群负载水位在可控范围内。重调度器是动态调度器是一对好搭档，它们的关系可以参考下图 12，就像它的名字一样，它主要负责“保护”节点中已经负载比较“危险”的节点，优雅驱逐这些节点上的业务。

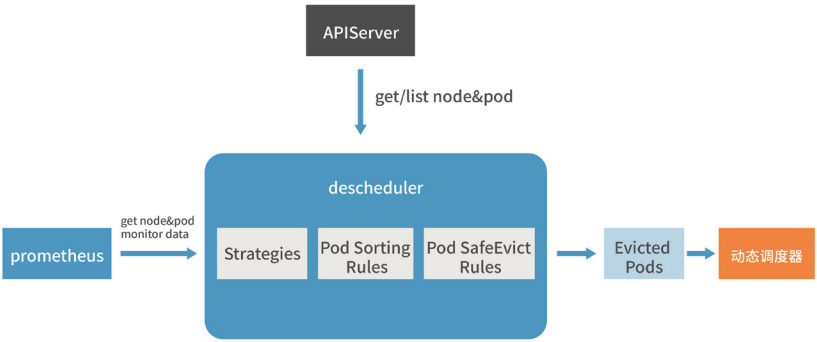

图12　重调度器和动态调度器的关系

3.2.3 多维度弹性

通过 HorizontalPodAutoscaler 按指标弹性扩缩容

如果你的业务是存在波峰波谷的，固定的资源 Request 注定在波谷时会造成资源浪费，针对这样的场景，如果波峰的时候可以自动增加业务负载的副本数量，波谷的时候可以自动减少业务负载的副本数量，将有效提升资源整体利用率。

HPA（Horizontal Pod Autoscaler）可以基于一些指标（例如 CPU、内存的利用率）自动扩缩 Deployment 和 StatefulSet 中的 Pod 副本的数量，达到工作负载稳定的目的，真正做到按需使用。

HPA 使用场景

- **流量突发：**突然流量增加，负载过载时会自动增加 Pod 数量以及时响应
- **自动缩容：**流量较少时，负载对资源的利用率过低时会自动减少 Pod 的数量以避免浪费

通过 HorizontalPodCronscaler 定时扩缩容

假设你的业务是电商平台，双十一要进行促销活动，这时可以考虑使用 HPA 自动扩缩容。但是 HPA 需要先监控各项指标后，再进行反应，可能扩容速度不够快，无法及时承载高流量。针对这种有预期的流量暴增，如果能提前发生副本扩容，将有效承载流量井喷。

HPC（HorizontalPodCronscaler）是腾讯云容器服务 TKE 自研组件，旨在定时控制副本数量，以达到提前扩缩容、和提前触发动态扩容时资源不足的影响，相较社区的 CronHPA，额外支持：

- 与 HPA 结合：可以实现定时开启和关闭 HPA，让你的业务在高峰时更弹性
- 例外日期设置：业务的流量不太可能永远都是规律的，设置例外日期可以减少手工调整 HPC
- 单次执行：以往的 CronHPA 都是永久执行，类似 Cronjob，单次执行可以更灵活的应对大促场景

HPC 使用场景

以游戏服务为例，从周五晚上到周日晚上，游戏玩家数量暴增。如果可以将游戏服务器在星期五晚上前扩大规模，并在星期日晚上后缩放为原始规模，则可以为玩家提供更好的体验。如果使用 HPA，可能因为扩容速度不够快导致服务受影响。

通过 VerticalPodAutoscaler 垂直扩缩容

Kubernetes Pod 垂直自动扩缩（Vertical Pod Autoscaler，以下简称 VPA）可以自动调整 Pod 的 CPU 和内存预留，帮助提高集群资源利用率并释放 CPU 和内存供其它 Pod 使用。相较于 水平自动伸缩功能 HPA，VPA 具有以下优势：

- VPA 扩容不需要调整 Pod 副本数量，扩容速度更快。
- VPA 可为有状态应用实现扩容，HPA 则不适合有状态应用的水平扩容。
- Request 设置过大，使用 HPA 水平缩容至一个 Pod 时集群资源利用率仍然很低，此时可以通过 VPA 进行垂直缩容提高集群资源利用率。

VPA 使用场景

VPA 自动伸缩特性使容器服务具有非常灵活的自适应能力。应对业务负载急剧飙升的情况，VPA 能够在用户设定范围内快速扩大容器的 Request。在业务负载变小的情况下，VPA 可根据实际情况适当缩小 Request 节省计算资源。整个过程自动化无须人为干预，适用于需要快速扩容、有状态应用扩容等场景。此外，VPA 可用于向用户推荐更合理的 Request，在保证容器有足够使用的资源的情况下，提升容器的资源利用率。

通过 ClusterAutoscaler 自动调整节点数量

上面提到的 HPA 和 HPC，都是在业务负载层面的自动扩缩副本数量，以灵活应对流量的波峰波谷，提升资源利用率。但是对于集群整体而言，资源总数是固定的，HPA 和 HPC 只是让集群有更多空余的资源，是否有一种方法，能在集群整体较“空”时回收部分资源，能在集群整体较“满”时扩充集群整体资源？因为集群整体资源的使用量直接决定了账单费用，这种集群级别的弹性扩缩将真正节省使用成本。

如图 13 所示，CA（Cluster Autoscaler）用于自动扩缩集群节点数量，以真正实现资源利用率的提升，并直接作用于用户的费用，是降本增效的关键。

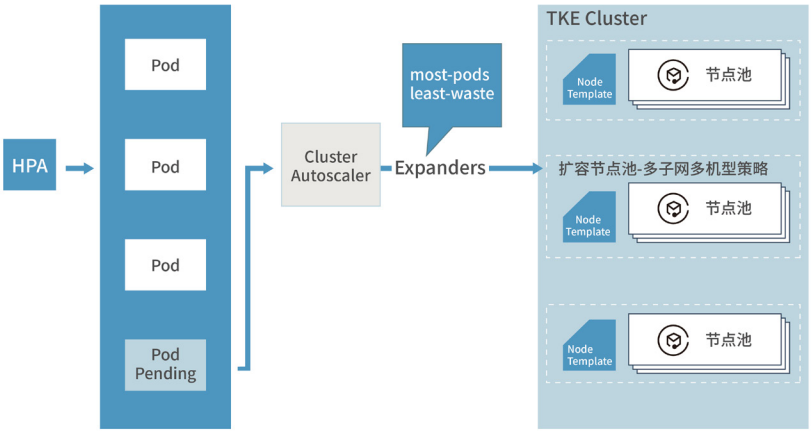

图13　ClusterAutoscaler 原理

CA 使用场景

- 在业务波峰时，根据业务突增的负载扩容合适的节点
- 在业务波谷时，根据资源的空闲情况释放多余的节点

通过虚拟节点获得 Serverless 能力

虚拟节点并不是节点，而是一种调度能力，支持将标准 Kubernetes 集群中的 Pod 调度到集群服务器节点之外的资源中。腾讯云容器服务的虚拟节点会将开启该功能的集群中，符合调度条件的 Pod 调度到由弹性容器服务维护的云上计算资源中。

部署在虚拟节点上的 Pod 具备云服务器一致的安全隔离性，具备与部署在集群既有节点上的 Pod 一致的网络隔离性、网络连通性，如图 14 所示。

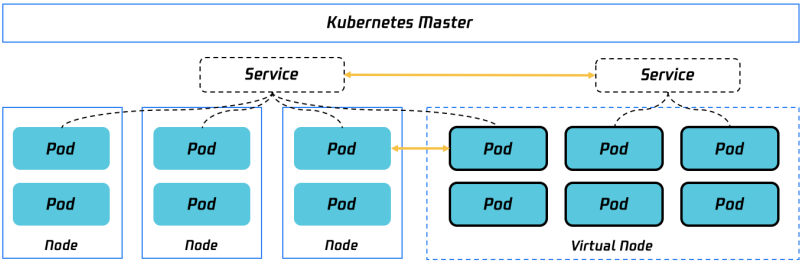

图14　在虚拟节点部署业务

图 15 展示了虚拟节点与普通 Kubernetes 集群扩缩容的耗时对比，从中可以看出，虚拟节点的扩容比节点的创建流程更快，有效应对流量突发场景；虚拟节点瞬时缩容，有效避免浪费。

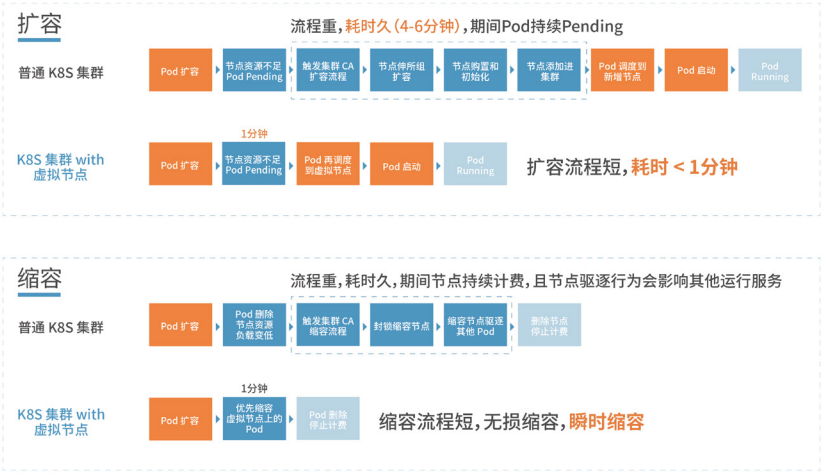

图15　普通 Kubernetes 集群与支持虚拟节点的 Kubernetes 集群扩缩容对比

虚拟节点使用场景

- 减少集群资源 buffer，应对长期运行服务波峰
  - 虚拟节点可以不占用集群服务器节点资源，快速的部署大量 Pod。适用于长期运行且资源负载特征为潮汐型的应用。
  - 使用虚拟节点可以减少集群预留 buffer，将集群的节点维护在资源利用率更高、使用和预留更合理的水平。
  - 在业务波峰进行扩容时会自动的优先调度到节点上，消耗预留的节点资源，再调度到虚拟节点上为集群补充更多的临时资源，这些资源会随着 Pod 缩容自动退还。
- 替代节点扩缩容，应对短期运行任务
  - 适合将短时间运行、资源需求量大的任务，手动调度到虚拟节点上。无需在部署这些负载前后进行集群节点的扩缩容，降低了扩缩节点的时间周期和维护成本。
  - 可以在部署这些任务的时候，指定虚拟节点进行调度，就不会占用集群其他服务运行的节点资源。且任务运行结束 Pod 退出会自动退还资源并停止计费，不需要人力或程序再干预。

竞价实例

竞价实例（Spot Instance）是云服务器 CVM 的一种新实例运作模式，它最核心的特点是折扣售卖和系统中断机制。但正如它的名字一样，您和其他同时使用竞价实例的用户存在一定的竞争关系。在特定场景下，实例可能会被回收，我们官方将这种回收定义为系统主动中断（库存波动）。当前阶段，在腾讯云的竞价实例模型下，仅会因为竞价实例资源池库存不足而产生中断。资源管控系统会自动根据实时库存变化回收这些折扣售卖的实例。当您成功购买一个竞价实例后，它的使用和按量计费的 CVM 实例基本毫无区别，包括控制台操作、远程登录、服务部署、关联 VPC 等。

竞价实例使用场景

在丰富的实践与探索中，我们发现，Spot 非常适合容器、无状态服务、CI/CD、强化学习、离线转码、大数据分析等具有容错能力的业务应用，尤其是基于云原生框架构建的应用，在这些场景下可以在巨幅降低成本（80% 以上）的前提下，保证业务的稳定性。

3.2.4 在离线混部

混部概念

提高集群资源利用率有几种方式，一是集群本身合理配置应用申请资源，尽量运行更多的作业。二是在波谷时段填充其他作业，运行更多的作业。第一种方式适合不同类型的应用混部，应用之间资源互补，高峰时段错开。若是同种类型的应用，应用都在同一时段处于高峰，这种情况适合第二种方式。本节主要阐述基于方式二的混部，即在线离线混部。

在线离线混部是通过在在线作业运行过程中填充离线作业，来提高资源利用率。离线任务不能无限填充，需要保证在线作业不受影响，保证其 SLO 在可接受范围内，同时离线作业要能快速上线下线，当在线作业需要资源的时候，及时出让。另外，离线运行起来之后，还要保证离线作业的成功率，不能因为频繁出让资源，而导致失败率很高。

混部场景

混部概念中将应用类型分为在线作业和离线作业，混部要解决的问题是如何通过填充离线作业把集群各个时段的在线空闲资源利用起来。集群每个时段的空闲资源会发生变化，这就要求离线作业要快速上线下线。那什么样的作业适合混部？ 在线作业特点包括但不限于运行时间长，有很强的时延敏感型，资源潮汐现象，如广告业务。而离线作业的特点包括但不限于运行时间短、计算需求大、容错率高、时延不敏感，允许重运行，典型的是 Hadoop 生态下的 MapReduce、Spark 作业。

结合一些实际的混部场景：例如在线应用分为两类，容器化和非容器化的。容器化的应用有基于 Kubernetes 和 Mesos 的。离线场景主要也有两类，分别是 Hadoop 类的大数据，以及基于 Kubernetes 的各种离线应用虑。由于场景比较多，混部也确定了两个主要目标：在保证服务质量的前提下，尽可能提升资源使用率。技术路线有几个原则：一是通用技术，方便开放到社区。二是要符合云原生使用方式，第三，降低对应用的依赖，不能引入太多假设，也要兼容主要生态。

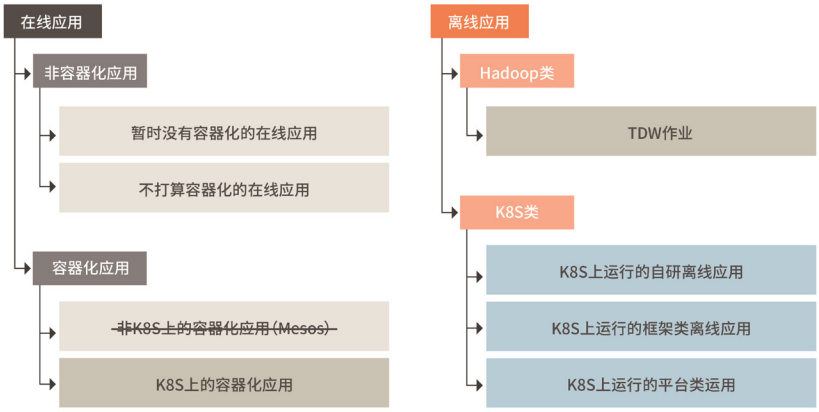

图16　混部场景

混部架构

腾讯数据平台部通过多年大规模资源调度的研究，设计了如图 17 所示的在线离线混部架构：

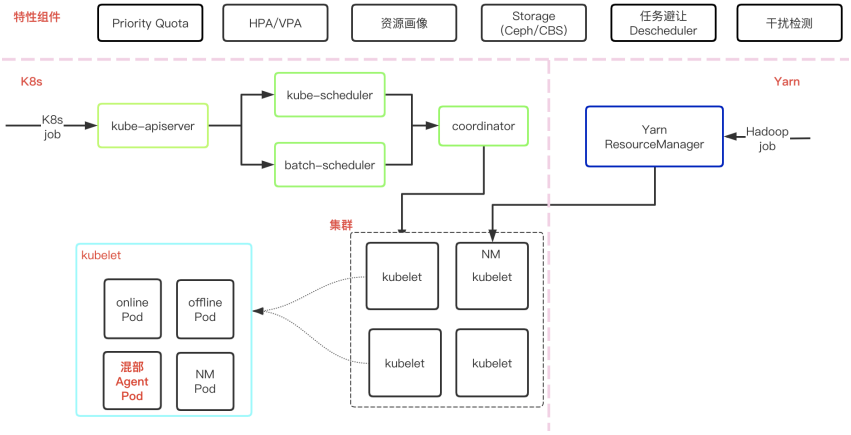

图17　在离线混部架构

设计以 Kubernetes 为依托，实现了以下关键技术，如：

（1）**任务定级：**制定了任务级别标准，用于对应不同优先级资源。

（2）**调度增强：**因离线任务量大，运行时间短，原生的 Kubernetes 调度器无法满足大批量需求，同时也缺少一些批量调度特性，如 gang scheduling。基于此增加一个批量调度器，跟原生调度器并行，通过一个协调器进行协调，防止资源被重复调度。

（3）**资源复用：**每个 Kubernetes 的 kubelet 节点通过 daemonset 部署一个 agent 组件，用于实现负载预测、闲置资源回收、离线资源配额分配和资源隔离等功能。

（4）**资源画像：**预测在线作业的各类资源使用情况，指导离线作业调度和隔离。

（5）**存算分离：**通过 Ceph 或腾讯云云硬盘CBS（Cloud Block Storage，简称CBS）分布式存储技术，解决离线作业需要大量存储空间及磁盘 IO 问题。

（6）**任务避让：**解决节点负载不均衡问题，重新调度离线作业到低负载节点和实现负载升高按优先级驱逐任务功能。

（7）**干扰检测：**分析在线作业的时延数据，如本身暴露的时延指标、CPI 数据或硬件指标数据，来判断在线作业是否受影响。

3.2.5 GPU 共享

随着机器学习的不断发展，使用 GPU 提供的并行算力已经非常普遍。很多厂商基于 Kubernetes平台，将应用容器化，并使用 GPU 资源，但是通常都是将一张完整的 GPU 卡分配给一个 Kubernetes Pod。对于某些场景，比如深度学习模型训练，这种分配方式可以提供比较好的隔离性，单卡的利用率也可维持一个不错的水准。但对于模型开发和模型推理则会比较浪费。随着 GPU 卡越来越多，独占 GPU 卡带来的资源浪费会越来越严重。大家的诉求很自然的会想到将更多推理服务共享同一张 GPU 卡，进而提高集群 GPU 利用率，正如图 18 所示。

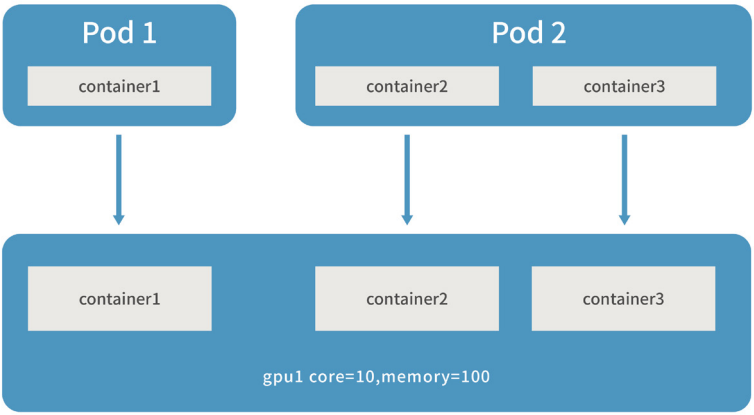

图18　GPU共享

GPU 共享包含两个关键技术点：精细调度与 GPU 隔离。前者在保证业务负载可被调度到同一张 GPU 卡上的同时，通过诸如 binpack、spread 或负载感知等调度策略，保证负载按照要求分布。亦或者依赖 GPU 拓扑感知调度，将一批负载亲和到同一 GPU 拓扑上，保证之间计算和数据传输的最大性能。而后者则保证在共享 GPU 资源时，业务互相之间不受干扰，这也是维持业务稳定性的关键所在。GPU 资源包括算力和显存，对这两种资源均需提供强有力的 QoS 保障和完全的隔离能力。只有如此，才能使得 GPU 共享带来的利用率提升产生的价值最大化。

如图 19 所示，腾讯云TKE qGPU 满足的正是这样的场景。你可以通过安装该产品在腾讯云容器服务 TKE 集群上创建 qGPU 节点池，并在创建负载时指定 qGPU 算力与显存。TKE qGPU 会帮助你将负载调度到同一张 GPU 卡上，并同时提供算力和显存的强隔离。GPU 利用率会随着共享技术有质的提升，你也无需担心共享带来的资源竞争会损害业务。

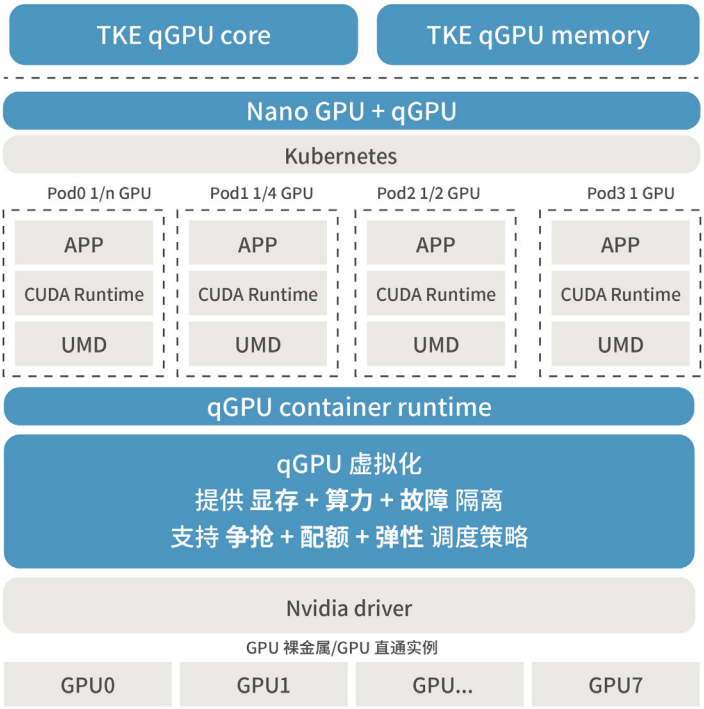
TKE qGPU core
TKE qGPU memory
Nano GPU + qGPU
Kubernetes
Pod0 1/n GPU
Pod1 1/4 GPU
Pod2 1/2 GPU
Pod3 1 GPU
APP
CUDA Runtime
UMD
APP
CUDA Runtime
UMD
APP
CUDA Runtime
UMD
APP
CUDA Runtime
UMD
qGPU container runtime
qGPU 虚拟化
提供 显存 + 算力 + 故障 隔离
支持 争抢 + 配额 + 弹性 调度策略
Nvidia driver
GPU 裸金属/GPU 直通实例
GPU0
GPU1
GPU...
GPU7

图19　TKE qGPU

3.2.6 优化矩阵

上面的章节概述了成本优化的各种手段，那么在何种场景下应该用何种技术手段，每种手段的效果和实现难度如何评估呢？我们将这些能力归入四个象限，横坐标为成本，纵坐标为难度，如图 20 所示。

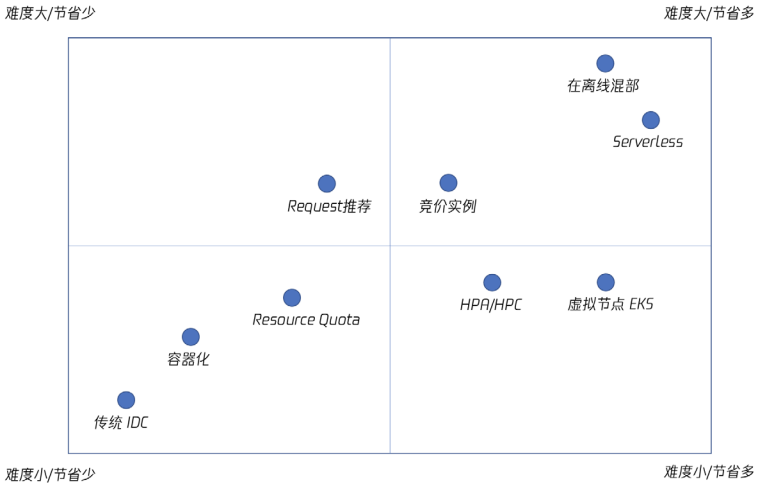

图20　优化方案的难度和效果

注释：(EKS，腾讯云弹性容器服务 Elastic Kubernetes Service)

3.3 成本运营

云成本优化并非是通过一系列标准动作后得出的一个终态恒定值，而是一个持续迭代和优化的过程。很多企业做成本优化都是项目制，做完项目后效果很好，但是很快反弹了。所以我们需要从组织、文化、流程等方面建设成本运营体系，这已经成为业界共识，FinOps 应运而生。

FinOps 基金会的定义很好的涵盖了它的本质：

“FinOps 是云的运营模式。FinOps 实现了一种转变——系统、最佳实践和文化的组合——以提高组织了解云成本和进行权衡的能力。与 DevOps 通过打破孤岛和提高敏捷性来彻底改变开发的方式相同，FinOps 通过将技术、业务和财务专业人士与一组新流程聚集在一起来增加云的业务价值。”

我们结合 FinOps 和客户实践整理了成本运营五大关键步骤：

建立成本优化团队	推动成本优化意识
数据驱动成本优化	在流程中考虑成本
量化成本优化交付的业务价值	

3.3.1 建立成本优化团队

建立一个成本优化团队，它可以是一个现有的个人或者团队，也可以是整个组织中由财务、技术和业务利益相关者组成的新团队。他们负责在团队里宣传成本意识的文化，确定成本优化的方向，推动成本优化的最佳实践，协调整个组织的成本优化工作。比如，可以定期举行成本优化会议，回顾和复盘成本管理中遇到的一些问题，从而推动持续改进。

此团队需要采取多学科方法，并具备项目管理、数据科学、财务分析和软件/基础设施开发的能力。他们可以执行成本优化（集中式方法）、影响技术团队执行优化（分散式）或将两者相结合（混合式），从而推进完成成本优化的目标。可以对照成本优化目标（例如资源利用率）来衡量团队的执行和交付能力。

同时必须确保团队获得高级管理层的支持。支持者即成本优化理念的倡导者，他们会为此部门提供升级支持，确保按组织确定的优先级开展成本优化活动。此部门及其支持者会共同确保组织在有效利用云资源并继续创造业务价值。

3.3.2 推动成本意识文化

成本意识是指节约成本和控制成本的观念，成本管理需要大家共同参与，只有团队具备了良好的成本意识，才能建立降低成本的主动性，才能让各种成本优化的举措、方法和要求得以很好的贯彻和执行。提高全员的成本意识是较长时期的任务，我们需要营造有利于提高成本意识的氛围和推动成本意识的管理。可以有如下方式推动成本意识文化：

- 在培训中强调成本的重要性和控制成本的必要性
- 建立激励机制，比如通过红黑榜的方式让业务团队之间进行竞争，对于成本管理做的好给予一定奖励，对于做的差的要定期复盘，确定后面的改进方向和举措，也可以运用一些 KPI 手段来进行管理，让员工意识到成本管理的严肃性。
- 积极倡导成本是企业的核心竞争力，成本优化能够带来真正意义上的商业价值。

3.3.3 数据驱动成本优化

所有成本优化都是由指标驱动的，所有指标都需要可实现的目标。

数据驱动是指将采集的数据有效组织，并对相关的信息进行整合和提炼，在数据的基础上经过训练和拟合形成自动化的决策模型。

对应到云原生成本运营，同样适用， 企业的IT团队往往会制定驱动团队成本优化的指标，通常是 CPU/内存使用率。 还有更精细的如闲置的存储资源、闲置的 IP 资源、无用的快照、无用的云盘等等。 通过设立目标来制定团队成本运营的指引方向。

需要注意的是：目标是可衡量的， 能够驱动团队采取下一步行动的，目标会受多方面的因素影响，明确目标之后就是要建立能够准确获取当前值与目标值的差距的途径， 并能够持续的观察当前值的变化，通过不断优化问题，做出正向的反馈，逐步靠近并达到目标值， 进而制定下一阶段的目标。

具体的落地可以借助云厂商或开源社区提供的各种可观测性仪表盘， 目前行业内的仪表盘的能力能够覆盖90%以上的场景， 如果需要更精细化且结合业务的数据，则需要企业自身投入研发。

3.3.4 在流程中考虑成本

在新的和现有的组织流程中建立成本意识，需要在软件生命周期全过程中去考虑成本，可以包括以下方案：

- 成本优化左移（ DevFinOps ），通过工具或者自动化加快成本优化，工程师在架构设计时需要考虑成本指标，保证能够和业务目标保持一致。
- 确保变更管理包含成本度量，以量化变更对成本的影响，这样有助于解决与成本相关的问题。
- 成本优化是运维或者说运营能力的核心组成部分；例如我们可以采用目前的故障管理流程来调查和确定成本管理异常的根本原因。
- 在组织中开展成本意识的培训和认证，有助于建立自我管理成本的组织。

3.3.5 量化成本优化交付的业务价值

由于成本优化是一项必要的投资，因此量化业务价值之后，您就可以向利益相关者说明投资回报。如果能够量化业务价值，在未来的成本优化投资中，就可以从利益相关者那里得到更多支持，并获得一个框架来衡量组织成本优化活动的成果。

这里有个单位经济学的概念，成本优化的最终目标是将成本追溯到业务收益，这不是在云上花费的美元，而是每次预订、每可用座位里程、每张机票、每笔客户交易、每百万活跃用户所花费的美元。

如果云成本在 6 个月内从 100 万上升到 200 万是不是很糟糕?

- 如果我的商品订单数在那段时间内翻了一番怎么办？那么它是中性的。
- 如果我的商品订单数在那段时间内增加了三倍怎么办？然后，我们只涨了 100 万就是非常好的一个成本优化的结果。

四、总结

资源配置不当是引起云原生成本浪费的主要原因

云原生底层的 Kubernetes 平台采用资源预留机制来管理容器可用资源，为避免因资源过小影响服务的可靠性，用户通常会预留远大于实际使用量的资源，导致大量资源浪费。部分业务流量存在波峰波谷，如果资源固定设置为波峰时期的需求量，那么波谷时资源利用率就会很低，长时间的低资源利用率也会导致资源浪费。不同类型的业务对资源的需求不同，如果不考虑业务资源需求的互补性，只针对单一业务配置资源，将很难实现总体资源利用率的最优化。

云原生成本管理面临问题难定位、路径难选择、成效难持续三大挑战

云原生平台底层资源共享、应用动态部署，底层云资源与应用无法一一对应，很难进行准确的成本观测，从而很难精准定位到成本浪费的根源。定位到问题根源后，选择成本优化的路径需要综合考虑成本、性能、稳定性和业务价值等多方因素，既要有切实有效的优化能力又不能对现有架构引入新的风险。由于业务动态变化，有效的成本优化机制运行一段时间后容易失效，如何实现持续的成本优化又成为新的问题。

成本洞察是定位云原生成本管理问题的基础

一是成本采集及资源追踪，需要针对复杂的云上资源提出统一的成本定量、定价和采集标准，并对资源进行持续跟踪。二是资源使用可视化，基于完善的云原生监测工具建立多维细粒度的资源利用率观测体系。三是费用可视化，相比资源利用率企业运营更关注成本，需要建立合理的计量计费方法，将资源利率可视化转换为费用可视化，并延展到费用历史分析、趋势预测等。四是成本分配，合理的资源标签模型是建立可持续的公平的成本分配方法的关键。五是账单管理，完备的云账单体系应满足各部门精细化的需求，能摸清各种云产品的支出情况，并能及时给到部门或组织反馈。

成本优化是减少云原生成本浪费的有效路径

成本优化方案需要围绕企业业务目标及业务场景制定，包括五大措施。一是设置合理的资源请求和限制，通过更细粒度的资源划分和管理，智能化的资源配额推荐和动态弹性伸缩，防止不同团队为保证业务稳定性过度侵占和消耗资源。二是动态调度，在调度应用时考虑业务资源利用类型与节点类型的匹配度、节点的真实负载，以及在应用部署后根据真实负载进行重调度。三是多维度弹性，通过灵活的负载、节点扩缩容策略应对多种业务负载波动。四是在离线混部，基于在离线业务对资源需求互补的特点，通过在在线作业运行过程中填充离线作业，来提高资源利用率。五是 GPU 共享，针对机器学习场景，通过精细调度与 GPU 隔离技术，在保证业务互不干扰的前提下实现推理服务共享同一张 GPU，大幅提升 GPU 利用率。

成本运营是实现云原生成本持续优化的长远之道

云原生成优化并不是成本管理的重点，因为成本优化的成效并不是恒定的，而是需要持续的迭代和优化。所以建立成熟的成运营体系才是成本管理的治本之法。成本运营包含五大关键步骤，包括建立成本优化团队、推动成本优化意识、数据驱动成本优化、在流程中考虑成本、量化成本优化交付的业务价值。成功的云原生成本管理非一时之功，需要从组织、文化、流程等多个方面来共同推动。

五、企业客户降本案例

本白皮书输出的成本管理模型和成本优化方案已在腾讯内外部客户大量落地，以下我们根据客户的使用实践整理出来成本优化案例，供大家扫码查看。

- 案例 | 腾讯广告 AMS 的容器化之路
- 案例 | 作业帮云原生成本优化实践
- 案例 | 云集云原生成本优化实践
- QQ浏览器信息流云原生应用之路
- 资源利用率提高67%，腾讯实时风控平台云原生容器化之路
- 宙斯盾 DDoS 防护系统“降本增效”的云原生实践
- 成本降低40%、资源利用率提高20%的 AI 应用产品云原生容器化之路
- 峰值利用率80%+，视频云离线转码自研上云TKE实践
- 揭秘日活千万腾讯会议全量云原生化上TKE技术实践
- 如何削减 50% 机器预算？“人机对抗”探索云端之路
- 案例 | 信安运维基于 TKE 平台的容器技术实践
- 案例 | 沃尔玛 x 腾讯云 Serverless 应用实践，全力保障消费者购物体验
- 技术赋能教育：51Talk 在线教育的 Serverless 及音视频 实践
- 用户案例 | 腾讯文档应用 Serverless 架构上云最佳实践

扫码查看

腾讯云原生 让用云更简单 更有效

腾讯云原生

扫码关注，更多干货

TKEStack

开放原子开源基金会项目

开源地址：https://github.com/tkestack/tke

扫码进入Github地址