自动化应用驱动的容器弹性

面向希望加快产品上市速度
同时确保应用性能的平台
和 DevOps 工程师

目录

执行摘要

企业的竞争优势取决于从创意转换到业务交易的速度，以及这种转换给客户带来的益处。技术正是这种转换的推动力。

容器提供的速度、敏捷性、弹性与规模从根本上改变了我们构建、部署和运行应用的方式。在容器开创的世界里，应用可以真正在任何地方运行；无论何时何地，更新和新功能每天都可以多次部署到生产中，而且可以通过弹性基础架构供应来管理动态波动的工作负载需求。Kubernetes 平台可以帮助组织提升敏捷性和弹性，但无法做到在保证性能的同时提高效率。

基于容器化所提供的简洁性和敏捷性，编排平台仅能提供一种管理这些服务生命周期的方法，即“以您描述的方式部署和维护您的服务”。

容器平台本身不能保证服务满足 SLO，也不能动态管理资源

基于阈值的策略不能解决持续性能问题；这种方法已然不再奏效，而且鉴于容器平台的变化速度，非关联触发自动缩放最终也可能会导致问题。弹性基础架构是确保性能的关键，但需要通过自动化分析来持续管理需求、供应和约束，以满足所需的服务级别目标 (SLO)。

本白皮书将为您介绍采用容器平台作为业务运营平台时需要考虑的关键概念，以及如何通过自动化保护方面的投资，在确保性能的同时最大限度地降低成本并保持合规。

本文简要阐述了为何您需要通过自上而下驱动的分析让自我管理的 Kubernetes 平台高效运行您的服务。在云之旅早期构建多云规模可为您的 IT 组织提供运营方面的“肌肉记忆”，这将会从根本上改变您交付更多创新的方式和时间。

速度、敏捷性、弹性与规模的承诺

Kubernetes 有助于实现弹性；它不会自动确保您满足应用服务级别目标 (SLO)。

若要在容器化采用方面取得成功，您需要为开发人员提供他们所需的敏捷性、规模化适应不断波动的需求所需的弹性，并确保应用以所需的速度运行。

采用云原生方法并将应用分解为不同的服务集，有助于推动更敏捷的应用开发和部署。容器提供了有助于确保服务可移植性、可扩展性的“包装”。Kubernetes 可为您提供数字应用和服务运行所需的框架和控制点。不过，若要提供业务所需的高性能企业级平台，您还需要添加各种功能，释放容器平台支持的弹性，以满足并确保应用 SLO。

利用 CICD 和生产反馈加快部署

基于自动化的持续集成持续部署（CICD）方法是加快产品上市速度的关键。在 Google Cloud 发布的《2021 年 DevOps 现状》报告[1]中，受访者提到了通过实施 CICD 所带来的重大改善：

部署频率	**每周 - 每月**	**每小时 - 每天**
变更准备时间	6 个月以上	不到 1 小时
变更失败率	16% - 30%	0% - 15%

随着速度的提升，需要一种方法来管理生产过程中的不断变更，还需要一个关于如何执行服务、如何预测基础架构需求的反馈循环。其目的在于找出一种定义 SLO 的方法，并让平台提供有关如何配置容器和基础架构，进而降低性能问题风险的反馈。

– 资源分配给服务的方式由谁决定？他们如何决定？是不是通过针对设定 SLO 进行压力测试和基准测试等方式？
– 您如何衡量性能？您的 CICD 管道中是否存在有助于确保正确配置容器和 Pod 的反馈循环？
– 您如何确保新部署始终有足够的容量可供使用？

选项	限制事项	Turbonomic 的解决办法
手动分析容器和 Pod 利用率数据，进而确定资源规格。	- 数据收集设置 - 分析需要投入人力	- 通过自上而下的应用驱动型分析确定调整容器大小的方式 - 反馈至 CICD - 可以减少非必要请求
手动分析堆栈中所有点的资源数据，进而确定生产容量。	- 从多个来源收集数据需要投入人力 - 分析需要投入人力	通过基于利用率的分析确定整个堆栈中的资源需求

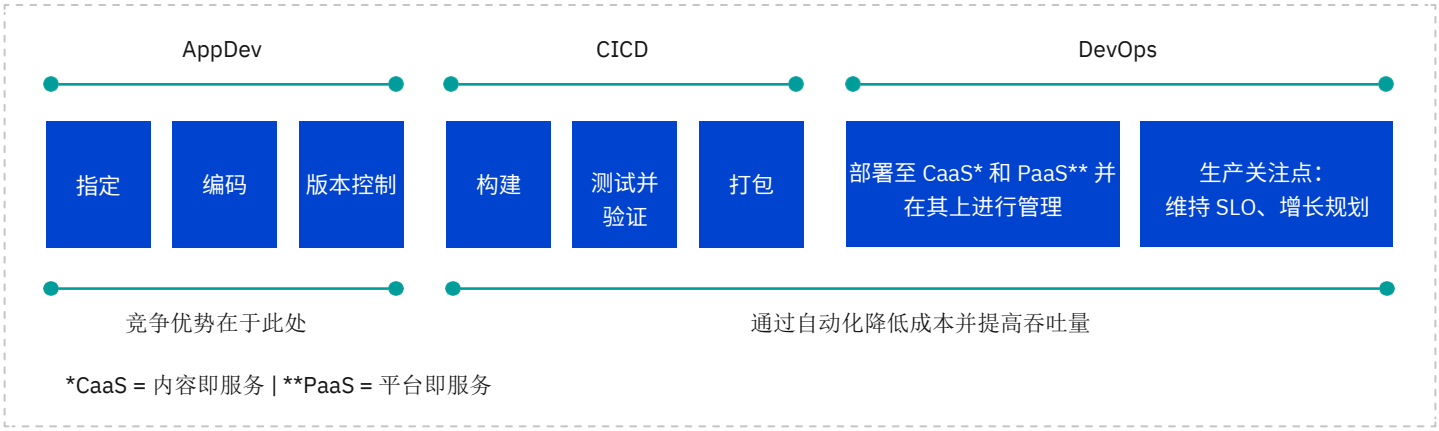


图 1. 应用敏捷性流程

平台与基础架构

为什么需要应用驱动型全栈管理

无论您选择容器平台还是底层基础架构，比如私有云、公有云、混合云、多云，甚至是裸机，在平台即服务 (PaaS) 方面都会面临如下运营挑战：

– 如何确定是否有足够的容量来满足当前需求和扩展需求？
– 如何决定何时启动更多应用节点？
– 如何决定何时暂停？
– 如何应对需求高峰？
– 如何充分利用公有云资源实现云爆发？
– 如何确保整个堆栈的高可用性 (HA) 和弹性？
– 如何实施业务约束？

借助容器平台支持的弹性，您可以根据应用平均需求总和（而非应用峰值需求总和）进行配备。若要利用这种弹性，就需要一个能够随着需求的波动而持续扩展或者收缩的平台，这就需要能不断做出资源调度决策的软件，以确保应用能够在需要时获得所需的计算、存储和网络。

选项	限制事项	Turbonomic 的解决办法
基于可提供自动缩放组（例如附属服务组 (ASG)、可用性集等）的服务提供商进行运行。	- 基于阈值的策略 - 无法扩展特定节点：所有节点必须具有相同的约束、节点标签等	- 自上而下的应用驱动型 SLO - 持续不断调整基础架构资源来满足应用需求 - 持续对适当的容器、Pod 和节点进行缩放、纵向扩展和横向扩展 -持续将 Pod 放置在适当的节点
分析堆栈中所有点的资源数据，进而确定生产容量。	- 从多个来源收集数据需要投入人力 - 分析需要投入人力	- 通过基于利用率的分析确定整个堆栈中的资源需求 - 持续对适当的容器、Pod 和节点进行缩放、纵向扩展和横向扩展 - 持续触发各种操作，以防止瓶颈

基于 SLO 的规模化运营

容器平台旨在以业务所需的服务级别运行应用。随着应用数量的增加，需要不断确保性能。通常，我们看到客户需要花费 12 个月以上的时间才能完成前 1 到 3 个应用的部署。对于后续应用的部署，基于学到的技能和最佳实践，可能需要额外 6 到 12 个月的时间。在业务线识别出各种可能性时，会发现需管理的单个服务数量已超出人力所能管理的极限。即使您已构建了无状态服务，可以利用容器的短暂特性，但最终用户体验势必会下降，您对这种下降又能容忍多少？您可以采取哪些手段来管理需求、应对不断加快的变化？答案在于自动化，即权衡分析需要多少服务实例才能确保 SLO，同时分析工作负载的大小和放置的配置、如何从基础架构获取合规资源，然后采取相应的操作。

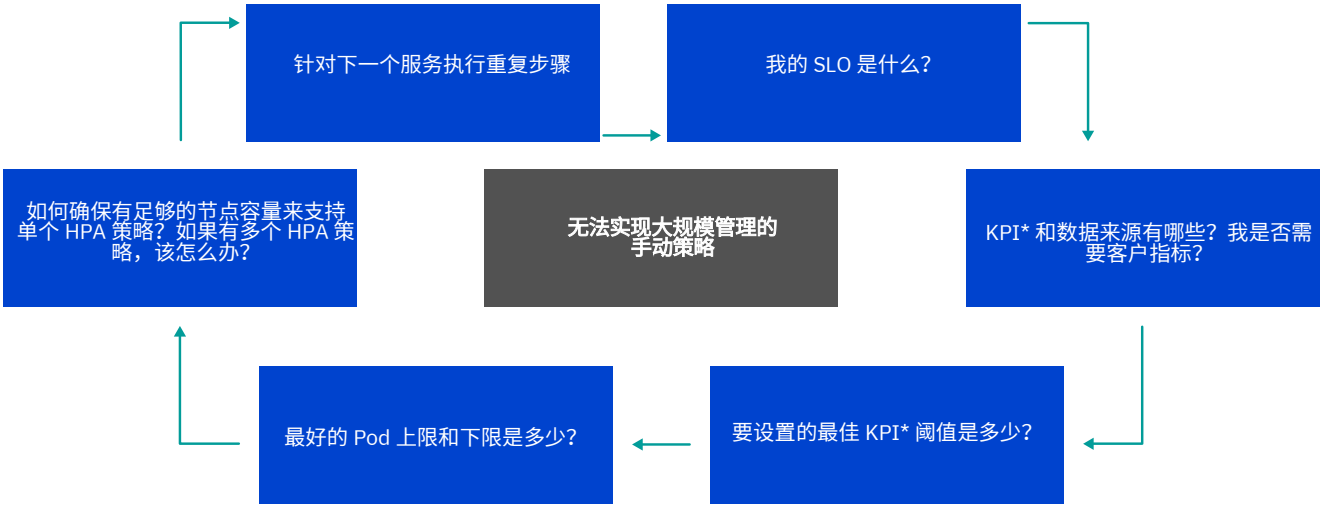
阈值不能解决问题

容器平台能够确保您只需部署最少数量的可用服务即可满足需求；如果某个服务崩溃，平台会尝试再次启动。不过为了确保良好的用户体验，您就会希望系统在出现性能下降和崩溃之前做出响应。您可以通过设置原生横向自动缩放来满足这一需求，但同时还需要决定最能表达所需资源的指标，配置阈值及上下限，测试和推断其是否能满足生产需求，然后对所部署的每个服务重复这些步骤。

想象一下，如果每个应用有 100 多个服务，情况将会怎样？这些策略都互不关联。您如何确保在添加更多服务 Pod 之后不会在另一个区域造成拥堵？您所克隆 Pod 的配置是否适当？是否需要首先进行纵向扩展？您如何管理节点拥堵、解决相邻干扰并找出未使用且可释放的已分配资源？

此外，容器、Pod 及横向 Pod 自动缩放器 (HPA) 或集群自动缩放策略的配置并非一劳永逸。这么一来，您就必须持续监控并重新定义最有可能的投入。如果您的团队不必手动设置和重置这些阈值，那么他们可以用节省的时间去做什么？

确保这些配置得到适当设置非常重要，对成功实施数字化转型战略有着直接影响。糟糕的部署可能会显著拖慢您所构建的平台和系统的采用速度。手动配置这些控制点需要投入过多的时间和人力，可能会严重阻碍贵组织平台优先战略的实施。您的企业能承受起这样的延迟吗？因此我们需要这么一个控制系统：该系统可以管理所有资源之间的权衡，并定义容器纵向扩展限制和请求、所需的 Pod 数量和放置决策，借助单个分析引擎重新分配 Pod 并管理集群资源。



*KPI = 关键性能指标

图 2. 无法实现规模化管理的手动策略

选项	限制事项	Turbonomic 的解决办法
横向 Pod 自动缩放器 (HPA) - 采用基于阈值的策略（确定何时进行 Pod 的缩放）	- 按服务进行配置 - 基于服务所有 Pod 的平均数 - 手动定义 KPI 和阈值及 Pod 上下限	- 自上而下的应用驱动型 SLO - 使用响应时间数据来驱动服务的横向扩展，以确保满足 SLO - 持续对适当的容器、Pod 和节点进行缩放、纵向扩展和横向扩展 - 持续将 Pod 放置在适当的节点 - 持续不断调整基础架构资源来满足应用需求
纵向 Pod 自动缩放器 (VPA) - 采用基于阈值的策略来进行容器的纵向扩展	- 必须为每项服务进行定义 - Beta 项目：风险自负 - 采取操作时不访问节点容量	
放任 Pod 崩溃，然后再将其重新部署到更好的节点上	在接近崩溃的 Pod 上所进行的交易用户体验不佳	
Prometheus 的可观察性解决方案能够收集和整合数据	- 不提供数据分析 - 不提供操作	

应用驱动型方法

应由 SLO 来驱动基础架构

对任务关键型应用的容器化进行投资具有诸多优势。不过，要充分获得速度、弹性和可移植性这些优势，便需要软件时刻 (24x7x365) 都能够在适当的时间做出适当的资源调度决策。否则，复杂性便会拖滞速度。

Turbonomic 能够将您的任务关键型应用连接到 Kubernetes 平台和底层基础架构，无论其在何处运行。软件会基于实时应用需求，同时考虑堆栈每一层（从逻辑层到物理层）的约束和相互依赖性，在适当的时间确定适当的操作，进而确保应用始终能够准确地获得它们需执行的操作。然后，应用会实时、按计划执行这些操作，或将其作为 DevOps 管道的一部分予以执行。

智能化大小确定：您应该如何调整容器的大小？

– 自动化部署 - 作为管道的一部分确定大小并持续调整，例如 YAML、Jenkins 等。

– 实时自动化 - 通过 Kubernetes 动态执行。

持续放置：何时需要移动 Pod？移至哪些节点？

– 通过 Kubernetes 实时动态执行。仅适用于无中断的无状态服务。

动态缩放：何时需要缩放集群？缩放程度如何？

– 通过基础架构即代码或 Kubernetes 集群 API 实时动态执行集群扩展。

SLO 驱动型缩放：何时需要横向扩展或缩放 Pod 来满足应用响应时间的 SLO？缩放程度如何？

SLO 驱动型缩放的前提条件：

– 应用专为横向无状态微服务而设计。

– 具有 Kubernetes 不提供的 SLO 数据定义和来源。

这种智能自动化对您自身、您的团队和您的企业而言意味着什么？无论您是在本地、云端、裸机服务器或任何组合上运行 Kubernetes，Turbonomic 均可为您提供以下独特优势。

应用“巡航控制”：您的团队负责设置响应时间 SLO；AI 驱动型软件可帮助确保平台和底层基础架构始终提供满足这些 SLO 所需的资源，无论应用在何处运行。

尽量减少人工投入：开发人员、DevOps 和站点可靠性工程师 (SRE) 无需设置阈值、约束或自动缩放策略。软件可为您做出正确的资源决策，提供自动化操作。

不要在容量上投入过多：无需依赖开发人员来进行资源调度决策。他们经常会过度配置，目的只是为了确保安全，对吧？我们的软件可准确确定所需的资源服务 - 一切均基于应用需求。

自信加速 DevOps：在确保安全的前提下提升部署频率和规模。我们的分析功能与您的 DevOps 工作流程相集成，可帮助确保新部署服务和现有服务永续运行。

更轻松地规划增长：您可以使用我们的软件模拟新服务的注册。还可以准确确定支持新增长所需的节点数量。

客户亮点

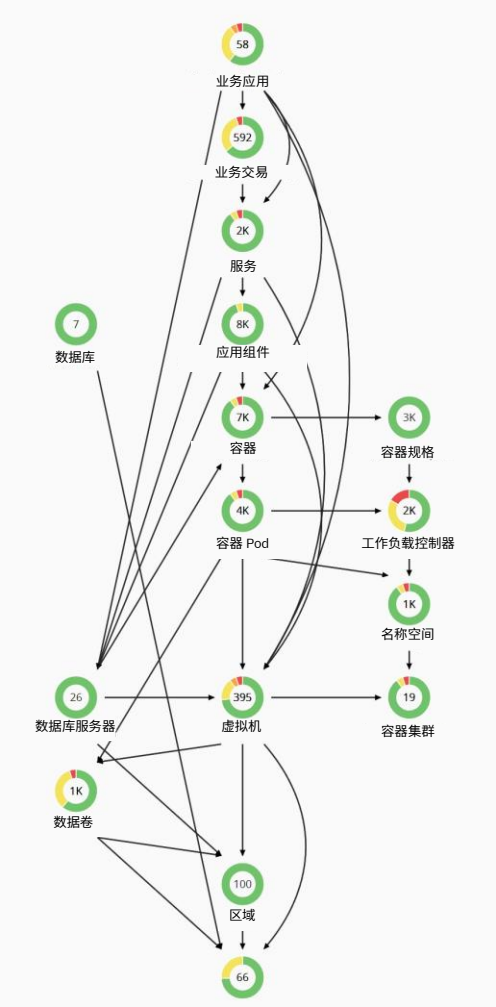
在疫情持续期间加速数字化转型

Turbonomic 在 Kubernetes 平台和底层基础架构中的动态资源调度响应速度非常快。

该客户是南美洲最大的保险公司之一，拥有超过 600 万名客户。其在现有环境和下一代环境资源调度方面采用的行业标准方法不仅拖滞了自身的数字化转型步伐，也导致公司在疫情应对方面不够快速。

在假期需求高峰期，Turbonomic 自动化软件可保持快速响应

该客户有一款业务应用与该地区规模最大的一家低成本航空公司应用相集成。旅行保险都是通过该应用预订的，因此我们可以在图 3 中看到，需求高峰与多日复活节假期有关。虽然需求增加，但 Turbonomic 在 Kubernetes 平台和底层基础架构中的动态资源调度速度非常快。



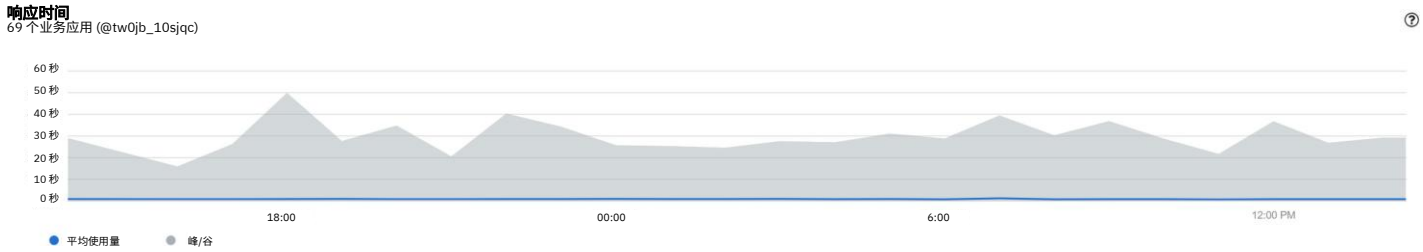


图 3. 单个业务应用及其响应时间的全栈视图，即使在需求高峰期，自动化的响应时间也保持在较低水平

57 个任务关键型应用

– 例如，车载 GPS：车辆报失、新保单报价等
– 约 3,000 个 Pod（包括约 7,000 个容器）
– 连接到 Dynatrace

自动化

– 容器大小调整（暂存）
– 持续放置（全部）

~70%
服务凭单减少量

关于 Turbonomic, an IBM Company

Turbonomic, an IBM Company 可提供应用资源管理 (ARM) 软件，该产品旨在通过跨混合和多云环境的动态资源调度应用来帮助客户确保应用性能和治理。Turbonomic 的网络性能管理 (NPM) 产品可提供现代化监控和分析解决方案，可帮助企业、运营商和托管服务提供商确保跨多供应商网络的规模化持续网络性能。

有关 Turbonomic 智能自动化的更多信息，敬请访问 ibm.com/cloud/turbonomic 或咨询 IBM 代表。

IBM Corporation
New Orchard Road
Armonk, NY 10504

美国印刷
2021 年 11 月

turbonomic
an IBM Company

[1] 2021 年 DevOps 现状，Google Cloud，2021 年