26 | 容器化：分布式数据库要不要上云，你想好了吗？

2020-10-09 王磊

分布式数据库30讲 进入课程 >

讲述：王磊
时长 14:11 大小 13.01M

你好，我是王磊，你也可以叫我 Ivan。

今天，我想和你分享的话题是分布式数据库的容器化部署。当数据库遇到容器，我知道这一定是个很有争议的话题。但是，在容器化技术大规模落地的背景下，这也是一个无法回避的话题。

容器化技术可以将资源虚拟化，从而更灵活快速地调配。容器镜像为应用打包提供了完美的解决方案，也为 DevOps 理念的落地扫清了技术障碍。可以说，容器已经成为现代软件工程化的基础设施，容器化已经成为一个不可逆的发展趋势。

但是，具体谈到数据库的容器化，我们又有太多的纠结。常见的反对意见就是数据库因为有状态、高 I/O 消耗和稳定运行等要求，所以不适合容器化部署。

那么，随着技术的快速发展，这些理由是不是还成立呢？为了说清楚这个问题，我们先来介绍一些 Kubernetes 的基本概念。

Kubernetes 基本概念

1. Container

容器化就是将物理机划分为若干容器（Container），应用程序是直接部署在容器上的，并不会感知到物理机的存在。具体来说，这个容器就是 Docker，它使用的主要技术包括 Cgroup 和 Namespace。

Cgroup 是控制组群（Control Groups）的缩写，用来限制一个进程组能够使用的资源上限，包括 CPU、内存、磁盘、网络带宽等，本质上实现了资源的隔离。Namespace 修改了进程视图，使当前容器处于一个独立的进程空间，无法看到宿主机上的其他进程，本质上实现了权限上的隔离。这两项其实都是 Linux 平台上的成熟技术，甚至在 Docker 出现前已经被用在 Cloud Foundary 的 PaaS 平台上。

Docker 能够快速崛起的重要原因是，它通过容器镜像可以将文件系统与应用程序一起打包。这个文件系统是指操作系统所包含的文件、配置和目录。这样，就不会出现各种环境参数差异导致的错误，应用的部署变得非常容易，完美解决了应用打包的问题。

容器的本质是进程，但复杂的应用系统往往是一个进程组。如果为每个进程建立一个容器，那么容器间就有非常密切的交互关系，这样管理起来就更加复杂。

那么，有什么简化的办法吗？

Kubernetes 给出的答案就是 Pod。

2. Pod

在 Kubernetes 的设计中，最基本的管理单位是 Pod，而不是容器。

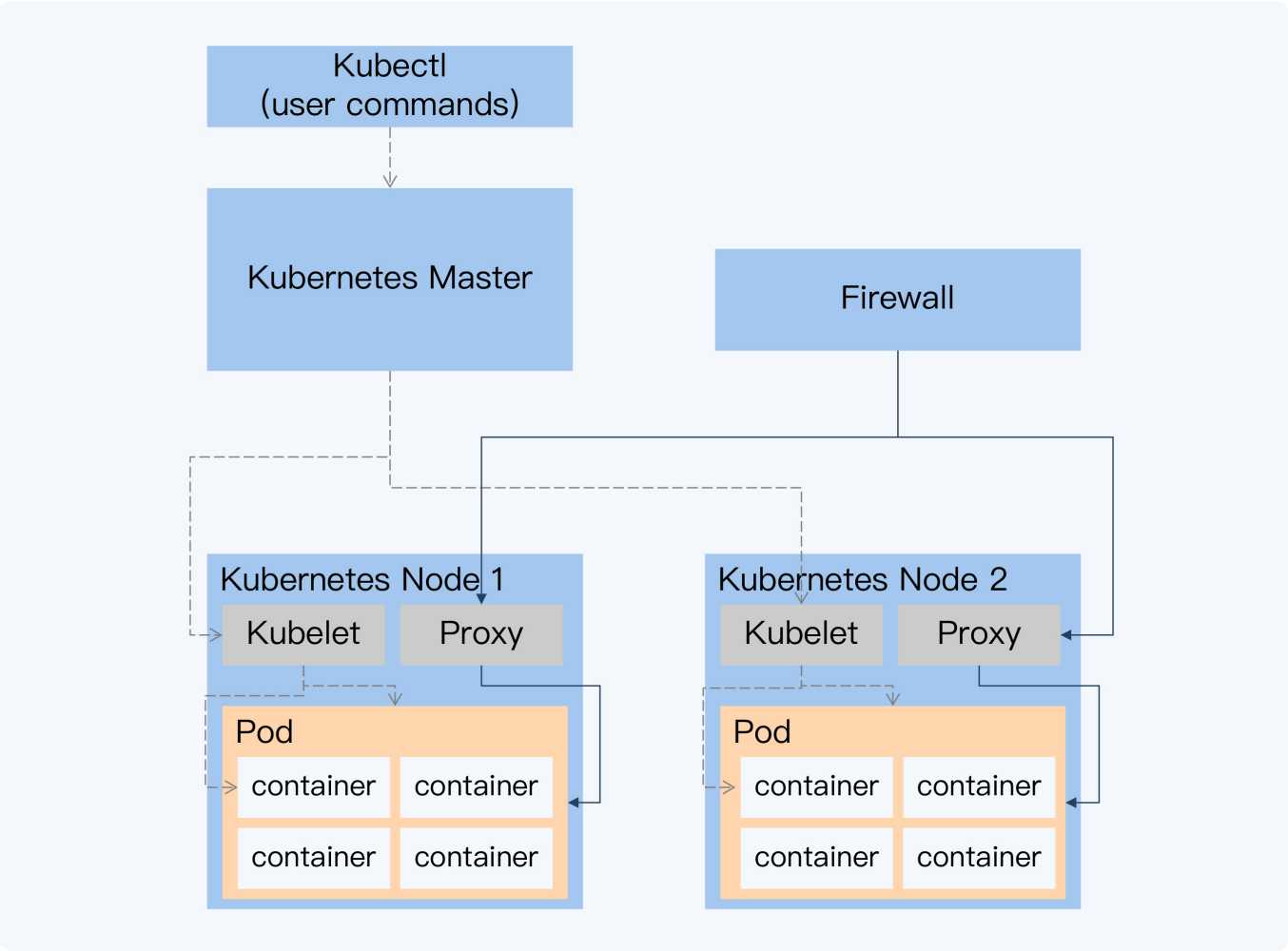
Pod 是 Kubernetes 在容器之上的一层封装，它由运行在同一主机上的一个或者多个容器构成。而且，同一个 Pod 中的容器可以共享一个 Network Namespace 和同一组数据卷，从而高效的信息交换。因为具有了这些特性，Pod 通常会被类比为物理机。

Pod 存在的意义还在于可以屏蔽容器之间可能存在依赖关系，这样做到 Pod 在拓扑关系上的完全对等，调度起来更加简单。

那么 Pod 又是如何被管理的呢？这就要说到 Kubernetes 的整体架构。

3. 整体架构

总体上，Kubernetes 是一个主从结构（Master-Slave）。其中 Master 是一个控制面板，每个 Salve 就是物理节点上代理 Kubelet，Master 与 Kubelet 通讯完成对物理节点上 Pod 和 Container 的管理。外部网络可以直接通过节点上的 Proxy，调用容器中的服务，不需要经过 Master。



Master 内部又包括 API Server、Scheduler、Controller 和 ETCD 四个部分，它们的职责分工和这一讲的内容没有直接关系，为了降低学习难度，这里就不做具体介绍了。如果你想再深入了解 Kubernetes 相关知识，可以在张磊老师的专栏找到更多资料。

有状态服务（StatefulSet）

对于无状态服务的管理，用户可以定义好无状态服务的副本数量，Kubernetes 调度器就会在不同节点上启动多个 Pod，实现负载均衡和故障转移。多个副本对应的 Pod 是无差别的，所以在节点出现故障时，可以直接在新节点上启动新的 Pod 替换已经失效的 Pod。整个过程不涉及状态迁移问题，管理起来很简单。

但是，作为应用系统的基石，数据库和存储系统都是有状态服务。如果将它们排除在外，那容器化的价值将大打折扣。所以，Kubernetes 在 V1.3 版本开始推出 PetSet 用于管理有状态服务，并在 V1.5 版本更名为 StatefulSet。

通常，我们说数据库是一个有状态的服务，是指服务要依赖持久化数据，也就是存储状态。而对于 StatefulSet 来说，状态又分为两部分，除了存储状态，还有系统的拓扑状态。

拓扑状态

拓扑状态是指集群内节点之间的关系，在 Kubernetes 中就是 Pod 之间的关系。

对于应用服务器来说，因为它们互相之间是不感知、不依赖的，所以可以随意创建和销毁。但数据库就不一样了，即使是单体数据库，也会有主从复制的关系，从节点是依赖于主节点的。

而分布式数据库的节点角色更加复杂。除了 CockroachDB 这样的 P2P 架构，多数分布式数据库中的节点拓扑关系是不对等的，会明确的划分为计算节点、数据节点、元数据和全局时钟等。

拓扑状态管理的第一步是标识每个 Pod。我们已经知道，Pod 随时会被销毁、重建，而重建之后 IP 会发生变化，这将导致无法确定每个 Pod 的角色。StatefulSet 采用记录域名的方式，再通过 DNS 解析出 Pod 的 IP 地址。这样 Pod 就具有了稳定的网络标识。

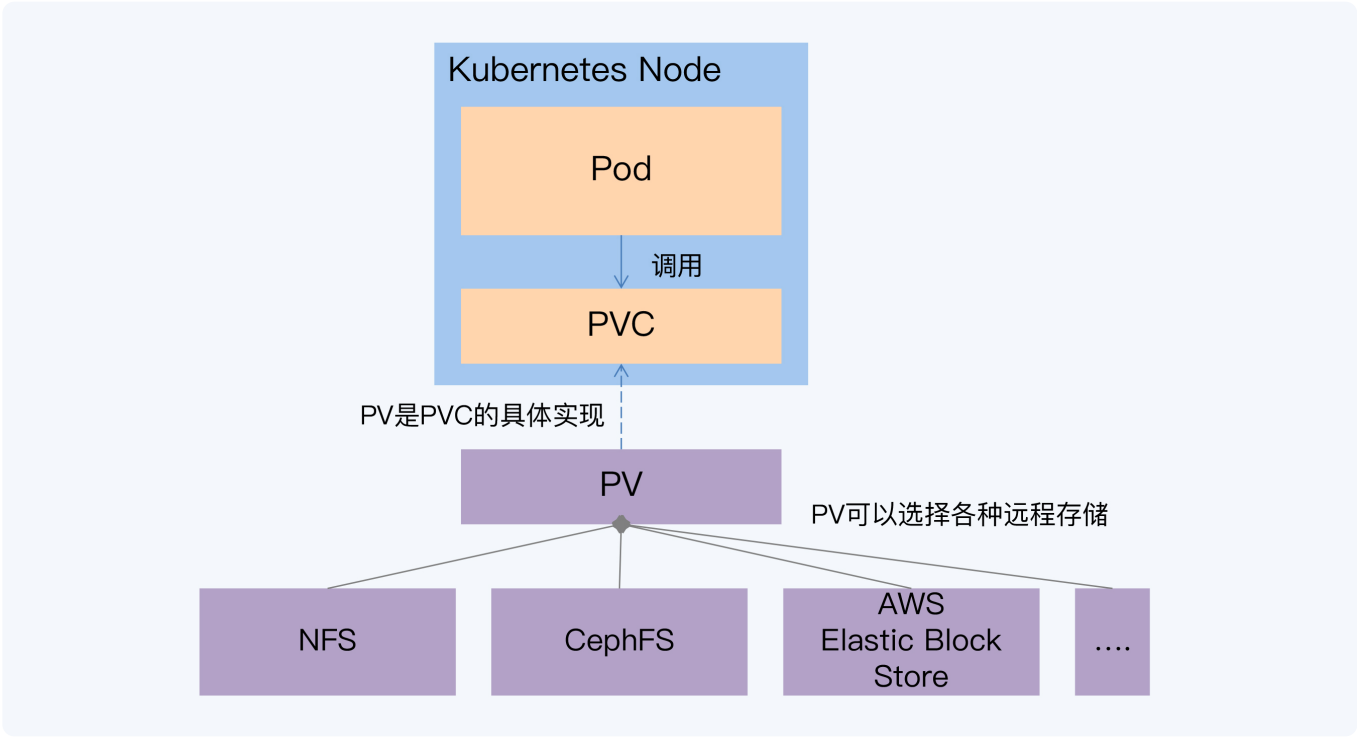
同时，为了确保体现 Pod 之间的依赖关系，StatefulSet 还引入顺序的概念，将 Pod 的拓扑状态，也就是启动顺序，按照 Pod 的“名字 + 编号”的方式固定了下来。

存储状态

数据库的核心功能是存储，我们再来看看如何实现存储状态的管理。

早期版本中，提供了两种存储方式，分别是本地临时存储和远程存储。顾名思义，本地临时存储中的数据会随着 Pod 销毁而被清空，所以无法用做数据库的底层持久存储。

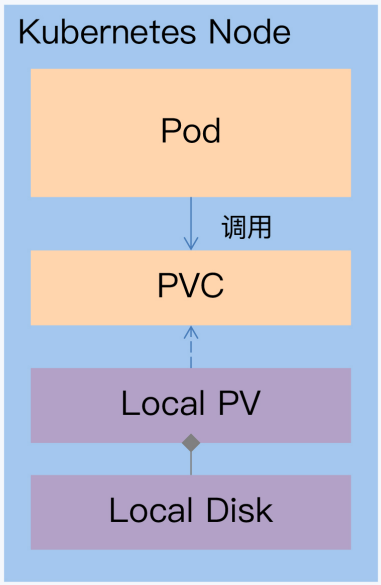
远程存储可以保证数据持久化，所以是一种可选方案。具体来说，就是使用持久化卷（Persistent Volume）作为数据的存储载体。当 Pod 进行迁移时，对应的 PV 也会重新挂载，而 PV 的底层实现是分布式文件系统，所以新的 Pod 仍然能访问之前保存的数据。



远程存储虽然可以使用，但是它与分布式数据库的架构设计理念存在很大冲突。我们在课程中介绍过，分布式数据库都是采用本地磁盘存储的模式，并且对存储层做了很多针对性的优化。如果改为远程存储，这些设计将会失效，导致性能上有较大的损失。

事实上，这就是很多人反对数据库容器化部署的最主要的原因，数据密集型应用对 I/O 资源的消耗巨大，远程存储无法适应这个场景。

针对这个不足，Kubernetes 尝试引入本地持久卷（Local Persistent Volume）来解决。简单来说，就是一个 Local PV 对应一块本地磁盘。开启 Local PV 特性后，调度器会先获取所有节点与 Local PV 对应磁盘的关系，基于这些信息来调度 Pod。



与普通 PV 不同，当挂载 Local PV 的节点宕机并无法恢复时，数据可能就会丢失。这和物理机部署方案面临的问题是一样的，而分布式数据库的多副本机制正好弥补了这一点。

不过 Local PV 推出得较晚，直到 V1.10（2018 年发布）Kubernetes 才相对稳定地支持调度功能，在 2019 年 3 月的 V1.14 中正式发布。

通过对拓扑状态和存储状态的管理，StatefulSet 初步解决了有状态服务的管理问题，但其中的一些关键特性（比如 Local PV）还不够成熟。另外，由于 Kubernetes 的抽象程度比较高，所以在真正实现分布式数据库的部署还需要很多复杂的控制。

Operator

控制逻辑的复杂性是与具体软件产品相关的，比如一个常见的问题就是对于 Pod 状态的判断。

Kubernetes 判断节点故障依据是每个节点上的 Kubelet 服务上报的节点状态。但是，Kubelet 作为一个独立的进程，从理论上说，它能否正常工作与用户应用并没有必然联系。那么就有可能出现，Kubelet 无法正常启动，但对应节点上的容器还可以正常运行。

这时，如果不处置原来的 Pod，立即在其他节点创建新的 Pod，应用系统可能就会出现异常。所以，调度过程还需要结合应用本身的状态来进行，其中的复杂性无法被统一封装，于是也就催生了一系列的 Operator。

Operator 的工作原理就是利用了 Kubernetes 的自定义资源，来描述用户期望的部署状态；然后在自定义控制器里，根据自定义 API 对象的变化，来完成具体的部署和运维工作。简单来说，Operator 就是将运维人员对软件操作的知识代码化。

etcd-operator 是最早出现的 Operator，用于实现 etcd 的容器部署。Operator 封装了有状态服务容器化的操作复杂性，对于应用系统上云的有重要的意义，有的分布式数据库厂商也开发了自己的 Operator，比如 TiDB 和 CockroachDB。

小结

那么，今天的课程就到这里了，让我们梳理一下这一讲的要点。

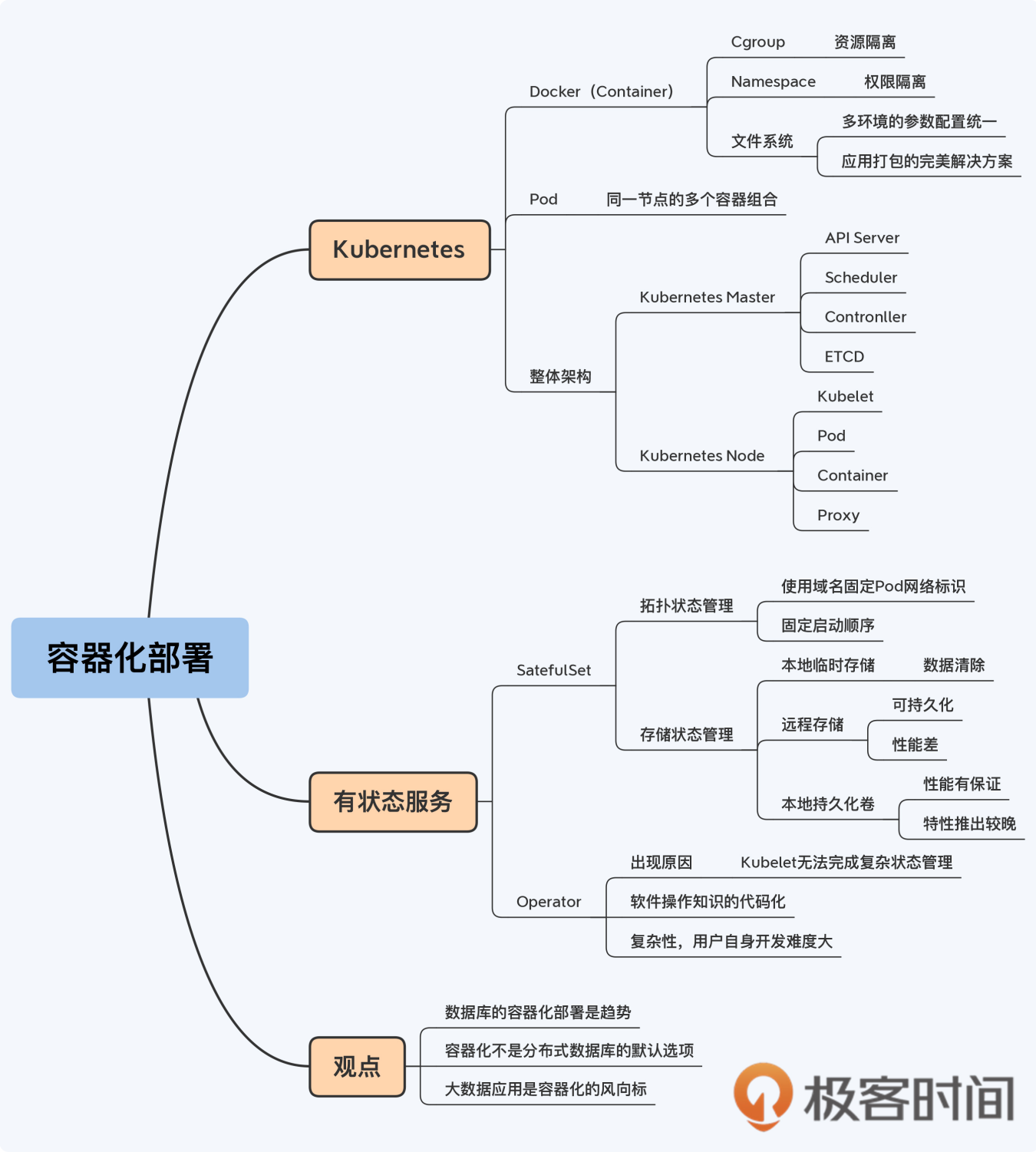
1. 容器可以更加精细的控制资源，并通过镜像功能实现应用打包部署，使开发、测试、生产等各个环境完美统一，因此得到了越来越广泛的使用。
2. 容器是单进程模型，为了适应更复杂的应用，Kubernetes 又引入了 Pod 概念，一个 Pod 包含了同一节点上的多个容器。Kubernetes 通过调度管理 Pod 将复杂的应用系统快速容器化部署，这种技术称为“容器编排”。因为容器的本质是进程，所以 Kubernetes 也被成为下一代操作系统。
3. Kubernetes 早期以支持无状态服务为主，Pod 完全对等，调度过程简单。为了适应更复杂的应用部署模式，Kubernetes 也在不断完善对有状态服务的支持，使用 StatefulSet 对象封装了相应的功能，包括拓扑状态管理和存储状态管理。在 1.14 版本中发布了本地持久化卷特性，增强了存储状态管理能力。
4. 对于分布式系统来说，有状态服务的管理非常复杂，通过简单的 Kubelet 无法把握各个服务的真实状态，所以就有了 Operator 这个扩展方式，每个产品厂商可以拓展自定义控制器，进行更有针对性的管理。目前，Operator 的接受程度在不断提高，TiDB、CockroachDB 都开发了自己的 Operator。

今天，容器化对于分布式数据库来说并不是一个默认选项。少数产品如 TiDB 和 CockroachDB 开始提供容器化部署方案，这和他们正在开展的云服务商业模式有比较直接的关系。而更多的分布式数据库仍然使用物理机，例如 OceanBase 和 GoldenDB 等。没有厂商的支持，依靠用户自身的力量实现容器化，还有比较大的困难。

我认为在未来的一两年里，“数据库是否要容器化部署”仍然是一个有争议的话题。但数据库的容器化趋势应该是确定的，因为 Kubernetes 提供支持越来越完善，真正大规模落

地只是时间早晚的问题。而随着分布式数据库普及，数据库的节点规模必然会快速扩大，运维成本随之提升，这也给数据库容器化带来更强的推动力。

如果你想更积极地尝试数据库容器化，可以把大数据应用作为一个风向标。这是因为大数据应用的可用性要求低于 OLTP 场景，而集群规模更大，上百甚至上千节点都是正常规模。这意味着，大数据应用的容器化改造要承担的风险更小，但收益更大。



思考题

课程的最后部分是我们今天的思考题。今天的课程中，我们简要介绍了 Kubernetes 对有状态服务的支持情况，从中不难发现，资源调度是它非常核心的功能。我的问题是，除了 Kubernetes 你还知道哪些集群资源调度系统？它们在设计上有什么差异吗？

欢迎你在评论区留言和我一起讨论，我会在答疑篇和你继续讨论这个问题。如果你身边的朋友也对数据库的容器化部署这个话题感兴趣，你也可以把今天这一讲分享给他，我们一起讨论。

提建议

上一篇 25 | 容灾与备份：如何设计逃生通道保证业务连续性？

下一篇 27 | 产品测试：除了性能跑分，还能测个啥？

精选留言 (2)

写留言

piboye
2020-10-11

hbase使用hdfs这种方式会是以后的主流不？ k8s下，大量机器都有硬盘，计算资源被调度好了，硬盘其实挺浪费的，基于这些机器的硬盘建立的分布式文件系统，在基于分布式文件系统来建立弹性的数据库服务，不知道这个方向是否好？

展开

piboye
2020-10-11

cpu和内存k8s和knative，可以做到调用次数级别的弹性了。储存的目前调度的力度太大了，我觉得容器化和虚拟化作用很大，很多业务很难一下就定好容量，目前的云产品都要预定一个最大量，实际使用都存在很大的浪费，真正突破量级的时候扩容又是个大问题。

展开