ZStack 技术白皮书精选

ZStack--级联框架

扫一扫二维码，获取更多技术干货吧

版权声明

摘要

大道至简·极速部署，ZStack 致力于产品化私有云和混合云。

ZStack 是新一代创新开源的云计算 IaaS 软件，由英特尔、微软、CloudStack 等世界上最早一批虚拟化工程师创建，拥有 KVM、Xen、Hyper-V 等成熟的技术背景。

ZStack 创新提出了云计算 4S 理念，即 Simple （简单）、Strong （健壮）、Smart （智能）、Scalable （弹性），通过全异步架构，无状态服务架构，无锁架构等核心技术，完美解决云计算执行效率低，系统不稳定，不能支撑高并发等问题，实现 HA 和轻量化管理。

ZStack 发起并维护着国内最大的自主开源 IaaS 社区——zstack.io，吸引了 6000 多名社区用户，对外公开的 API 超过 1000 个。基于这 1000 多个 API，用户可以自由组装出自己的私有云、混合云，甚至利用 ZStack 搭建公有云对外提供服务。

ZStack 拥有充足的知识产权储备，积极申报多项软著和专利，参与业内标准、白皮书的撰写，入选云计算行业方案目录，还通过了工信部云服务能力认证和信通院可信云认证。ZStack 面向企业用户提供基于 IaaS 的私有云和混合云，是业内唯一一家实现产品化，并领先业内首家推出同时打通数据面和控制面无缝混合云的云服务商。选择 ZStack，用户可以官网直接下载、1 台 PC 也可上云、30 分钟完成从裸机的安装部署。

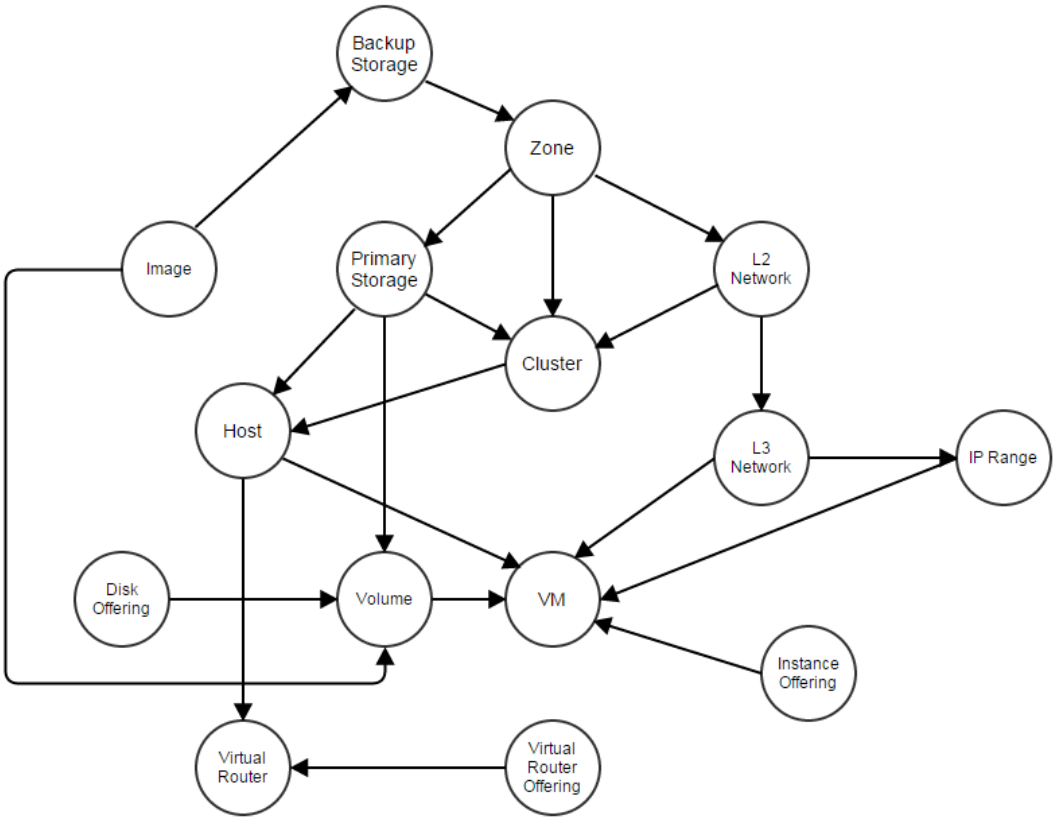
目前已有 1000 多家企业用户选择了 ZStack 云平台。

ZSTACK--级联框架

云中的资源相互都有关系。操作一个资源通常会引发连锁反应；例如，当删除一个集群的时候，是非常合理地去删除属于该集群的所有主机并停止所有在这些主机上运行的虚拟机。传统的 IaaS 软件要么硬编码连锁反应，要么简单地禁止这些操作，例如，禁止用户删除有虚拟机运行的集群。ZStack 提供一个级联框架，用以散布本来只对一个资源的操作到所有相关的资源。资源可以通过实现一个简单的扩展点以加入级联框架，使得资源的业务逻辑与框架解耦。

动机

云中的资源多多少少都彼此依赖；例如，一个主机是一个集群的子资源，一个主存储是一个集群的兄弟资源，L3 网络是一个区域的后裔资源。资源之间的关系可以被描述为一个有向图：



上图，我们展示了 ZStack 的主要资源；不同的 IaaS 软件可能使用不同的术语，上图主要是想让你有一个粗略的概念。由上图所暗示的，当对资源进行操作时，不仅仅是目标资源，

相关资源也将受到影响；例如，当删除一个区域时，比较理想的是属于区域的集群、主机、主存储、L2 网络等资源也同时被删除。为了处理这个问题，IaaS 软件必须满足级联（cascading）操作的需求。

问题

大多数 IaaS 软件很少考虑级联操作。它们要么硬编码业务逻辑，例如，你需要显式删除一个将要被删除帐户的所有资源；要么直接不允许这种操作，例如，当你试图删除一个 IP 地址范围时，抛出一个错误信息“仍有 VM 使用在这个 IP 范围中的 IP”。这两种方法都会带来很多麻烦。对硬编码而言，它使软件不能灵活的添加新的资源，因为你必须修改现有的代码来添加级联操作，例如，修改删除帐户的代码使得账户删除时，新资源也被删除。对于完全没有责任感的错误信息，用户要么去做无聊的工作，例如，在删除一个 IP 范围之前，手动删除 100 个虚拟机；要么摧毁现有的一切，然后从零开始，例如，重新部署整个云。

> ***避免误操作不是借口：****有些人可能会声称不允许级联删除是慎重考虑的结果，因为用户可能会误操作，误操作可能带来灾难性的后果；例如，错误地删除区域会导致损失掉所有虚拟机。然而，这种说法只是一个错误的借口，并且是一种为用户做决定的自作聪明。你能想象吗，当你为了删除一个区域必须手动删除 10,000 个虚拟机，因为软件认为你可能会做错事，所以迫使你枯燥的重复 10,000 次任务确认？一个好的软件应该为用户提供选择，并让他们做出决定。在我们的例子中，IaaS 软件应该在进行到最后删除之前警告用户，还有 10,000 台虚拟机在运行；但一旦用户承认他们需要这么做，软件就应该这么做。*

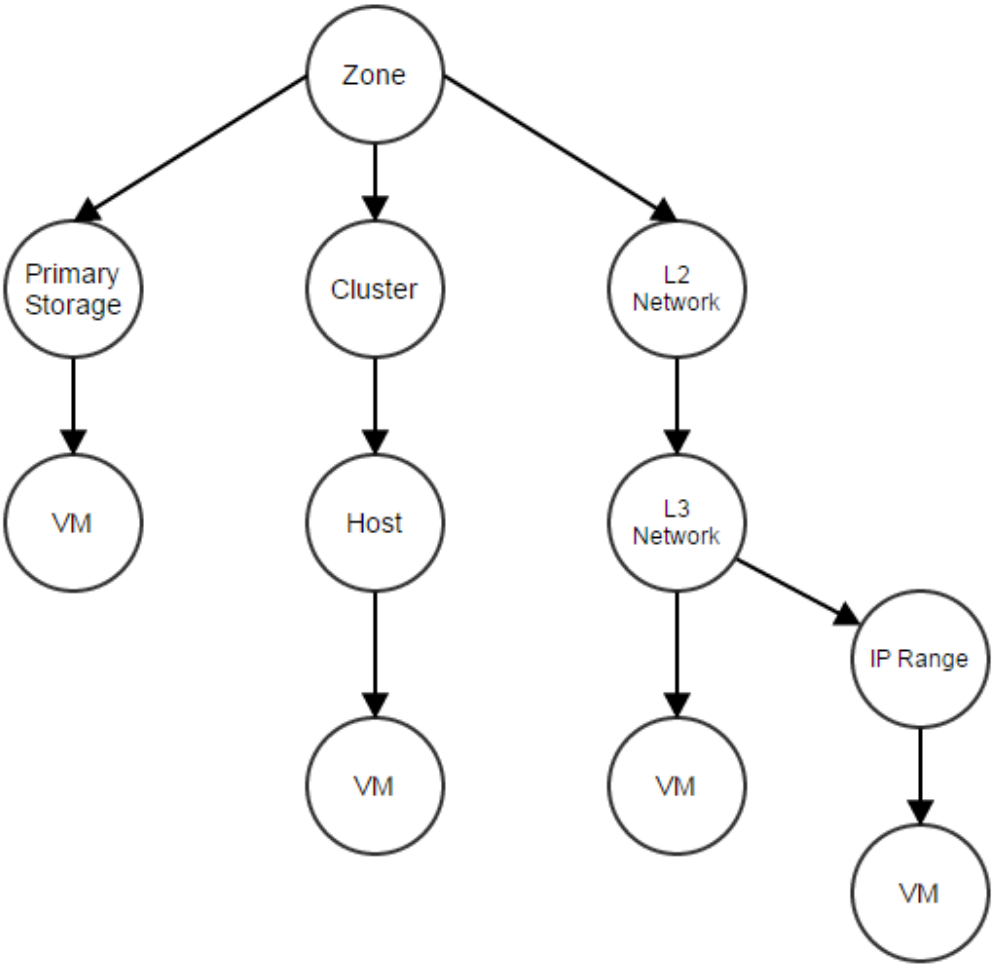
级联框架

ZStack 通过一个级联框架解决这一问题；顾名思义，级联框架允许一个操作能从一个资源级联到其他资源。为了解耦整个架构，这个级联框架被作为一个单独的组件创造出来，资源可以按意愿加入框架。要加入框架，资源所需要做的全部事情就是实现一个扩展点 `CascadeExtensionPoint`（在我们的例子中 `AbstractAsyncCascadeExtension` 是一个实现 `CascadeExtensionPoint` 的类）：

```
class VmCascadeExtension extends AbstractAsyncCascadeExtension {

    @Override

    public void asyncCascade(CascadeAction action, Completion completion) {

        if (/* this is from deleting Primary Storage*/) {

            /* delete VMs that have root volumes on the primary storage*/

        } else if (/*this is from deleting L3 Network*/) {

            /* stop VMs that have nics on the L3 network, and remove those nics */

        } else if (/* this is from deleting IP range*/) {

            /* stop VMs that have nics whose IP is in the IP range */

        } else if (/* this is from deleting host*/) {

            /* stop VMs that run on the host */

        }



        completion.success();

    }



    @Override

    public List<String> getEdgeNames() {

        return Arrays.asList(

                PrimaryStorageVO.class.getSimpleName(),

                L3NetworkVO.class.getSimpleName(),
```

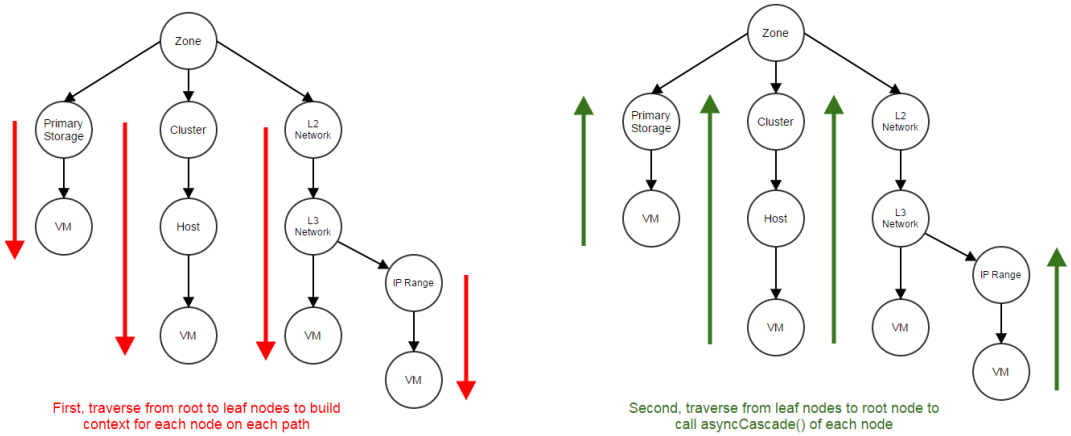
```
                IpRangeVO.class.getSimpleName(),
                HostVO.class.getSimpleName()
        );
    }

    @Override
    public String getCascadeResourceName() {
        return VmInstanceVO.class.getSimpleName();
    }

    @Override
    public CascadeAction createActionForChildResource(CascadeAction action) {
        return convertContextToVmRelatedContext(action);
    }
}
```

`getCascadeResourceName()`方法返回该资源的名称（VmInstance）；`getEdgeNames()`方法返回一个和资源直接关联的资源名列表，在我们的例子中返回主存储、L3 网络、IpRange 和主机；所以如果删除操作在这些 `edge resources` 或其上游资源（如区域）上发生时，该操作将被级联至在 `getEdgeNames()`方法中声明了这些资源的扩展。级联扩展可以在 `asyncCascade()` 中采取行动，并获取必须的信息比如操作码（如删除），根发起者（如区域，下文将很快给出解释），作为操作来源的父发起者（如主机，将很快给出解释）和操作上下文（例如，哪台主机正在被删除）。由于资源的关系是一个可能有环路的有向图，级联框架将把图压扁成一棵树，并把环路变为分支。例如，删除区域的操作将最终创建以下树（一部分）：



注：如你所见，删除区域操作将多次级联到虚拟机的级联扩展；这是刻意的，因为级联扩展通常依赖于父发起者去决定该采取什么行动；在这个例子中，虚拟机的父发起者为主存储、主机、L3 网络和 IP 范围；然而，对于不同的父发起者，扩展可能会采取不同的行动；例如，如果父发起者为主存储并且操作码为 `delete`*，该扩展将摧毁所有根云盘在该主存储的虚拟机；但如果父发起者是主机，扩展将会只停止在那台主机上的虚拟机，因为这些虚拟机稍后就可以在其他主机上启动。考虑到 ZStack 没有产生冲突的级联操作，例如，不会有一个操作导致虚拟机在路径 A 启动而在路径 B 停止，所以级联操作从不同路径进行多次延伸是没有问题的。*

当级联一个操作时，该框架从该操作被应用的`root issuer`开始；在上述删除区域的示例中，`zone`是根发起者；那么框架将从根发起者遍历树，并调用扩展的`createActionForChildResource()`方法为每一条路径上的每一个扩展创建上下文；一旦所有上下文创建成功，该框架将再次遍历树，不过是从叶子节点到根，并调用每个扩展的`asyncCascade()`方法；一个扩展可以依靠父发起者去决定应该做哪些操作，父发起者在`getEdgeNames()`方法中以资源名的方式声明；例如，如果父发行者是主机，则停止虚拟机；如果父发行者是主存储，则删除虚拟机。



这两个阶段的遍历保证，一个操作（例如删除）将只会被应用到根发起者，在所有下游资源都做完一些合适的操作后。例如，一个区域只在所有子孙资源都被删除后才能被删除。

由于并不是所有的操作都需要级联，一个资源可以在它需要的时候直接调用`CascadeFacade.asyncCascade()`。

总结

在这篇文章中，我们演示了 ZStack 的级联框架，这是一个强大的工具，用于扩散操作而不需要硬编码。ZStack 用很多方式使用了它，除了我们在文中提到的以外，一些操作，如卸载主存储（这将停止将被卸载的集群中的所有虚拟机），卸载 L2 网络（这将停止将被卸载的集群中的所有虚拟机）都是以这种方式实现的。有了它的帮助，管理员可以快速尝试不同的云部署而无需担心不方便；你可以只删除你的部署的一部分并重新创建一个新的，而不需要仅因为你在一个设计错误的 L2 网络上创建了许多虚拟机，就重新部署整个云（举个例子）。