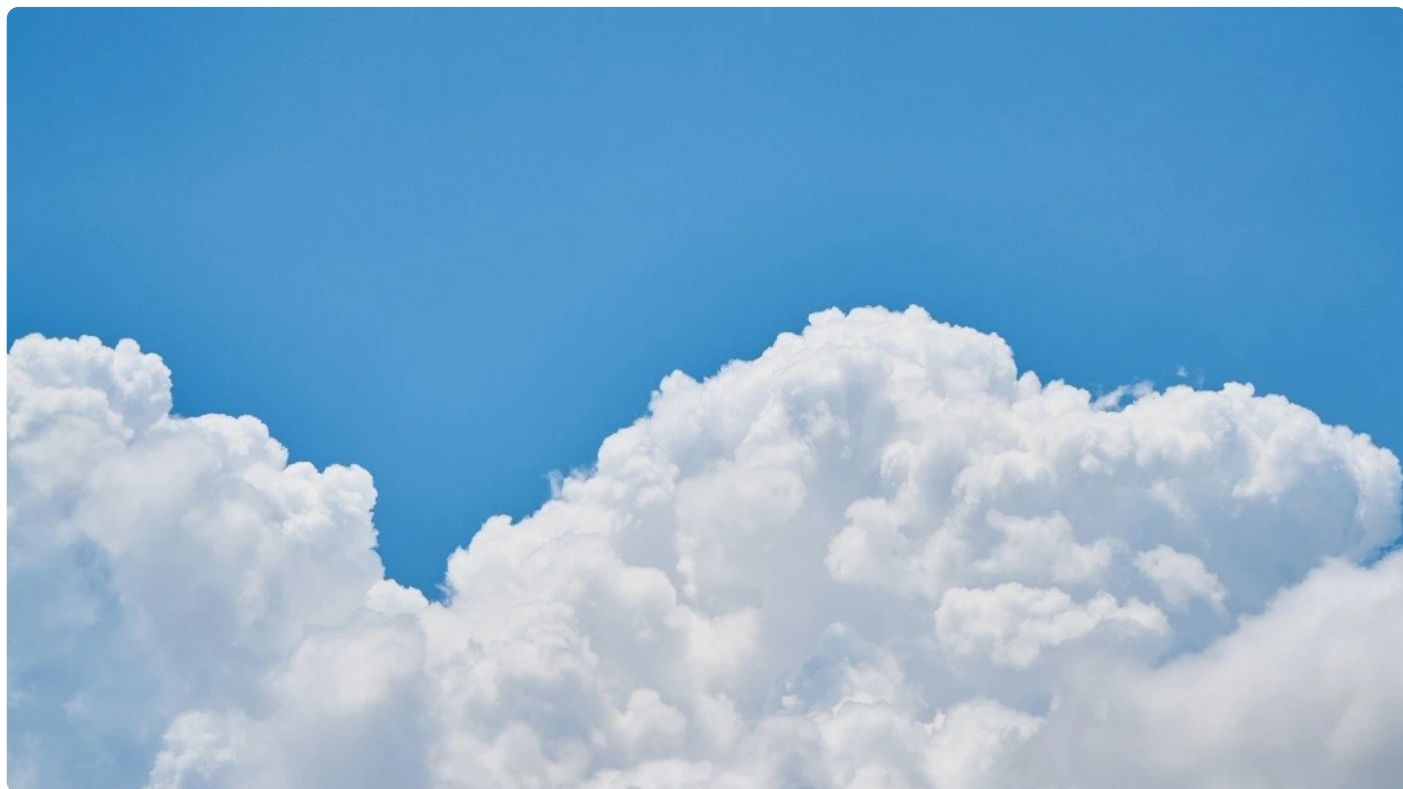


07 | 云端架构最佳实践：与故障同舞，与伸缩共生

2020-03-18 何恺铎 来自北京

《深入浅出云计算》



你好，我是何恺铎。这一讲，我们来谈谈云上架构的注意事项和最佳实践。

云上架构最需要注意什么呢？就像我在标题所描述的那样，云端架构一方面需要处理和应对可能出现的**故障**，保证架构和服务的可用性；另一方面则是需要充分利用好云端的**弹性**，要能够根据负载进行灵活的伸缩。

面对故障，提升冗余

故障，是 IT 业界的永恒话题。故障的原因多种多样，无论是由于硬件的自然寿命造成的，还是数据中心的极端天气捣鬼，或是人工运维操作上的失误，不论我们多么讨厌它，故障似乎总是不可避免。

你也许会问，**那么，云计算会有故障吗？比如说，云上创建的虚拟机，是否百分之百会工作正常呢？**

很遗憾，虽然公有云们为了避免故障，在许多层面上做了冗余和封装，但云也不是可以让你永远无忧无虑的伊甸园。我们需要牢记，云端的服务仍然是有可能出故障的，只是概率上的不同而已。这也是云供应商们为云服务引入**服务等级协议**（Service Level Agreement，简称 SLA）的原因，它主要是用来对服务的可靠性作出一个预期和保证。

SLA 的可用性等级可能是 99.9%，也可能是 99.99%，它能够表明某项云服务在一段时间内，正常工作的时间不低于这个比例，也代表了厂商对于某项服务的信心。不过你要知道，再好的服务，即便是 SLA 里有再多的 9，也不可能达到理论上的 100%。

小提示：当实际产生的故障，未达到 SLA 的要求时，云厂商一般会给予受到影响的客户以消费金额一定比例金额的赔付。不过很多时候，赔付的金额不足以覆盖业务上的经济损失，你不应该依赖它。

所以，从架构思维的角度上来说，我们需要假定故障就是可能会发生，对于它的影响事先就要做好准备，事先就进行推演并设置相关的冗余和预案。AWS 有一个非常著名的架构原则，叫做 **Design For Failure**，讲的也就是这个意思。

好在云上做高可用架构同样有自己的特点和优势，我们可以轻松地调用各个层面的云端基础设施来构建冗余，规避单点的风险。

那么，云上可能出现哪些不同层面的故障？相应的故障范围和应对措施又会是怎样的呢？我们不妨从小到大，依次来看我们可能遇到的问题和解决办法。

第一种故障是在宿主机的级别，这也是从概率上来说最常见的一种故障。当宿主机出现硬件故障等问题后，毫无疑问将影响位于同一宿主机上的多个虚拟机。为了避免产生这样的影响，当我们承载重要业务时，就需要创建多台虚拟机组成的集群，共同来进行支撑。这样，当一台虚拟机出现故障时，还有其他几台机器能够保证在线。

这里需要注意的是，**我们需要保证多个虚拟机不在同一台宿主机上，甚至不处于同一个机架上，以免这些虚拟机一起受到局部事故的影响。**那么，要怎么做到这一点呢？

虚拟机的排布看似是一个黑盒，但其实在公有云上是有办法来对虚拟机的物理分配施加干预，让它们实现分散分布，隔开一段距离的。这一特性，在 AWS 称为**置放群组**（Placement Group），Azure 称为**可用性集**（Availability Set），阿里云对应的服务则是**部署集**。比如说，我们对阿里云同一个可用区内的虚拟机，在创建时选择同一个部署集，就可以保证相当程度的物理分散部署，从而最大限度地保证它们不同时出现故障了。

第二种规模更大的故障，是在数据中心，也就是可用区的层面。比如火灾、雷击等意外，就可能会导致数据中心级别的全部或者部分服务类型的停摆。有时一些施工导致的物理破坏，也会挖断光纤，影响可用区的骨干网络。

要应对这类故障，我们就需要**多可用区的实例部署**，这也是云抽象出可用区概念的意义所在。你的实例需要分散在多个可用区中，这样，可用区之间既可以互为主备，也可以同时对外服务，分担压力。另外，也不要忘记我在 [🔗 上一讲](#)中所提到的，虚拟私有网络可以跨越可用区，这会大大方便我们多可用区架构的搭建。

第三种更严重的故障，就是整个区域级别的事情了。当然这种一般非常少见，只有地震等不可抗力因素，或者人为过失引发出的一系列连锁反应，才有可能造成这么大的影响。

区域级别的事情一般都难免会对业务造成影响了。这时能够进行补救的，主要看**多区域架构层面是否有相关的预案**。如果是互联网类的服务，这时最佳的做法，就是在 DNS 层面进行导流，把域名解析到另外的一个区域的备用服务上，底层的数据则需要我们日常进行着跨区域的实时同步。

再更进一步的万全之策，就需要考虑**多云**了，也就是同时选用多家云厂商的公有云，一起来服务业务。虽然集成多个异构的云会带来额外的成本，但这能够最大限度地降低服务风险，因为两家云厂商同时出问题的概率实在是太低了。更何况，多云还能带来避免厂商锁定的好处，现在其实也越来越多见了。

综上所述，不论是哪种级别的故障，我们应对的基本思想其实没有变化，都是化单点为多点，形成不同层面、不同粒度的冗余。当故障发生时，要能迅速地发现和切换，平滑地过渡到备用的服务和算力上。

当然，盲目地追求可用性也不可取。**根据业务需求，在成本投入与可用性之间获得一个最佳的平衡，才是你应该追求的目标。**试想一下，构建一个个人博客网站，和建立一个金融级系统，两者在可用性架构方面的要求显然天差地别，所以我们最后的架构选择也会大相径庭。

随机应变，弹性伸缩

弹性伸缩，这是云上架构的另一个原则，也是云端的重要优势。

由于云的本质是租用，而且它便捷的操作界面、丰富的 SDK 和自动控制选项，使得云上“租用”和“退租”的成本很低，可以是一个很高频的操作，这就为弹性伸缩在云上的出现和兴起提供了土壤。在妥善应用之下，弹性伸缩既可以提高工作负载洪峰来临时的吞吐和消化能力，提高业务稳定性，又能够在低谷期帮我们显著地节约成本。

在 IaaS 端，能够弹性伸缩的最实用的产品形态，莫过于**虚拟机编组**了，也就是功能相同的多个虚拟机的集合。把它们作为一个单位来创建、管理和伸缩，是一种普遍应用的最佳实践。AWS 中相关的产品命名是 **EC2 自动伸缩** (Auto Scaling)，Azure 中是**虚拟机规模集** (VM Scale Set)，阿里云则叫做**弹性伸缩**。

我们把多个虚拟机以弹性伸缩组的方式进行统一管理，能够极大地提高效率，减轻负担。因为弹性伸缩服务，会帮我们动态地创建和销毁虚拟机实例，自动根据我们指定的数量和扩缩容规则，来协调虚拟机的生命周期。我们只需要从高层进行指挥就可以了。

弹性伸缩服务，在云端还有一个最佳拍档，就是**负载均衡器**。它特别适合将流量均匀地，或者按照一定权重或规则，分发到多台虚拟机上，正好可以和提供计算资源的弹性伸缩服务形成配合。当负载增大、虚拟机增加时，负载均衡也能够自动动态识别，将流量分发到新创建的虚拟机上。

所以，你可以尝试使用弹性伸缩服务来实现云端弹性架构，用它来管理一组虚拟机，并与负载均衡一起配合。这特别适合处理无状态类的计算需求，因为它会为你代劳底层计算资源的管理。

高可用的弹性架构实战

结合上面的介绍，让我们进入这一讲的实战环节。

我们来模拟一个线上高可用服务的场景，来看下如何用阿里云进行服务的搭建。我会在上一讲搭建的虚拟私有网络的基础上来提供服务，并做到一定程度的故障隔离和弹性扩展。

我们先用 Node.js 来搭建一个简单的 Web 服务，用来计算著名的“斐波那契数列”。相关的源码如下，供你参考：

 复制代码

```
1  const express = require('express');
2  const ip = require('ip');
3  const os = require('os');
4  const app = express();
5  //使用递归计算斐波那契数列
6  function fibo (n) {
7      return n > 1 ? fibo(n-1) + fibo(n-2) : 1;
8  }
9  app.get('/', function(req,res) {res.write('I am healthy'); res.end();} );
10 app.get('/fibo/:n', function(req, res) {
11     var n = parseInt(req.params['n']);
12     var f = fibo(n);
13     res.write(`Fibo(${n}) = ${f} \n`);
14     res.write(`Computed by ${os.hostname()} with private ip ${ip.address()} \n`);
15     res.end();
16 });
17 app.listen(80);
```

我们在上一讲创建的虚拟机“vm1-in-vpc1”中安装好 Node 环境，将上述代码放入一个起名为“app.js”的文件中，用 npm 安装 express 等相关依赖后，就可以用命令“node app.js”直接运行了。然后，我们需要把这个服务设置为**开机自动启动**（你可以通过 npm 安装 pm2 组件来帮助实现开机自动启动），这样一个简单的 Web 服务就搭建好了。

为了让之后的外部流量能够进入到内部网络的多台虚拟机中，我们来建立对外的负载均衡实例。要注意，**负载均衡器本身也是需要是高可用的**，我们这里主要选择华东 2 区域下的可用区 D，让可用区 E 作为备可用区，和我们的 VPC 保持一致。

付费模式

包年包月

按量付费

地域及可用区

中国

亚太

欧洲与美洲

中东与印度

华北1（青岛）

华北1可用区B

华北2（北京）

华北2可用区B

华北3（张家口）

华北3可用区B

华北5（呼和浩特）

华北5可用区A

华东1（杭州）

华东1可用区F

华东2（上海）

华东2可用区D

西南1（成都）

成都可用区A

中国（香港）

香港可用区B

可用区类型

多可用区

单可用区指实例只在一个可用区存在；多可用区指实例在两个可用区存在,当主可用区不可用时会在备可用区恢复服务。[详情参考>>](#)

备可用区

华东2可用区B

华东2可用区E

然后，在负载均衡器上配置一个 HTTP 协议 80 端口的监听，后端服务器可以先指向我们的测试机 vm1-in-vpc1，然后从外部测试负载均衡器的连通性。

<input type="checkbox"/>	实例名称/ID	服务地址	状态	监控	实例体检	端口/健康检查/后端服务器
<input type="checkbox"/>	slb-for-auto-scale lb-uf6o6lfpjtpxzea2x9j1 未设置标签	47.101.77.110(公网IPv4)	✓ 运行中			HTTP: 80 ✓ 正常

复制代码

```
1 [client@clientVM ~]$ curl http://47.101.77.110/fibo/35
2 Fibo(35) = 14930352
3 Computed by vm1-in-vpc1 with private ip 192.168.1.80
```

可以看到，curl 命令的响应中，成功地返回了斐波那契数列第 35 项的结果值，以及相关服务器的名称、IP 等信息，说明负载均衡已经初步正常工作了。

接下来，我们要创建一个能够弹性伸缩的虚拟机集群，来大规模地对外输出这个计算服务。

作为准备工作，我们要先为 vm1-in-vpc1 创建一个镜像，作为新建虚拟机的“种子”：



<input type="checkbox"/>	镜像ID/名称	标签	镜像类型	系统位数	状态	进度
<input type="checkbox"/>	m-uf69qi8t0tviird8bvb1 vm-template-image1	 	自定义镜像	64位	可用	100%

然后，我们就可以创建弹性伸缩实例了。我们来建立一个最小数量为 2，最大数量为 10 的伸缩组。在这个过程中，你尤其需要注意，**要选取上一讲中建立的 VPC 作为目标网络，同时选择两个分属不同可用区的交换机，并设置为均匀分布策略。**如下图所示：

*伸缩组名称:

autoscale-group1

名称为2-40个字符，以大小写字母、数字或中文开头，可包含“-”，“_”或“.”

*伸缩组内最大实例数 (台) ?:

10

取值范围为 [0, 1000]

*伸缩组内最小实例数 (台) ?:

2

取值范围为 [0, 1000]

伸缩组内期望实例数 (台) ?:

2

取值范围在最小实例数与最大实例数之间
创建伸缩组时若未指定期望实例数，则修改时不能再次指定期望实例数。
只有设置了期望实例数的伸缩组才可以支持伸缩活动并行。 [前往查看详情](#)

*默认冷却时间 (秒) ?:

60

最小为0，必须为整数

☐ 开启伸缩组删除保护 ?

移出策略 ?:

先筛选

最早伸缩配置对应的实例

在结果中再筛选

最早创建的实例

移出

[如何保证手工添加的ECS实例不被移出伸缩组](#)

*组内实例配置信息来源 ?:

☒ 自定义伸缩配置 ☐ 启动模板

*网络类型:

☒ 专有网络 ☐ 弹性公网IP ☐ 弹性网卡

*专有网络:

专有网络ID: vpc-uf6irhi8124x59a168761

虚拟交换机: test-vpc1-vsw1 (华东 2 可用区 D) test-vpc1-vsw2 (cn-shanghai-e)

[创建专有网络](#)

多可用区扩容策略 ?:

☐ 优先级策略 ☒ 均衡分布策略 ☐ 成本优化策略

回收模式 ?:

☒ 释放模式 ☐ 停机回收模式

负载均衡 ?:

-- 请选择负载均衡 --

[管理我的负载均衡](#)

只有配置过监听的负载均衡才能被伸缩组使用

负载均衡配置详情

伸缩组下负载均衡最多可配置 1 / 10

负载均衡 ID: lb-uf6o6lfpijtpxzea2x9j1

负载均衡名称: slb-for-auto-scale

服务器组	端口(1-65535)	权重(1-100)
默认服务器组 ?	-	在伸缩配置里指定

+ 默认服务器组

+ 虚拟服务器组

同时在这里，我们还为伸缩组和刚才建立的负载均衡器建立了关联，这样弹性伸缩实例中的机器，会自动地进入到负载均衡后端服务器的列表中。

下一步是建立伸缩配置，这里主要是指定虚拟机模板，记得选取我们刚才创建好的自定义镜像：

镜像

公共镜像

自定义镜像

共享镜像

?

选择镜像前请确保至少已
选择一个实例规格

弹性伸缩报警任务依赖云
监控插件

vm-template-image1

▼

↺

启用伸缩配置后，很快就能看到弹性伸缩服务为我们建立了两台虚拟机了：

弹性伸缩架构

SLB 实例

ECS 实例

默认服务器组 ?

负载均衡 ID	权重
lb-uf6o6lfpijtpxzea2...	在伸缩配置里指定

总数量

2台

服务中

2台

备用中

0台

保护中

0台

停用中

0台

加入中

0台

在 ECS 控制台，你也可以清楚地看到，这两台机器被自动分配到了不同的可用区中，分属不同的交换机：

<input type="checkbox"/> 实例ID/名称	可用区 ▼	IP地址	状态 ▼	网络类型 ▼	专有网络属性
<input type="checkbox"/> i-uf67wymbgnd69wkf31 ESS-asg-autoscal...	华东 2 可用区 E	192.168.1.89(私有)	运行中	专有网络	vpc-uf6irhi8124x59al68761 vsw-uf6ls7t8l8lpt35c6a37u
<input type="checkbox"/> i-uf68viqv1vrqntkpyiha ESS-asg-autoscal...	华东 2 可用区 D	192.168.0.234(私有)	运行中	专有网络	vpc-uf6irhi8124x59al68761 vsw-uf6nbt7032jedor7jbnnf

我们再设置一下非常重要的伸缩规则，**这会告诉伸缩组何时进行自动扩缩容**。这里我们选择监控平均 CPU 指标，我们希望理想状态下控制在 50% 左右。换句话说，如果平均 CPU 偏离 50% 太远，系统就会自动地为我们增加或减少机器。

创建伸缩规则

*规则名称:

auto-scale-by-cpu

名称为2-64个字符，以大小写字母，数字或中文开头，可包含“_”、“-”、“.”

* 伸缩规则类型 ?

☐ 简单规则

☒ 目标追踪规则

☐ 预测规则

☐ 步进规则

* 指标类型 ?

平均CPU使用率

* 目标值 ?

50

%

* 实例预热时间 (秒) ?

60

* 禁用缩容 ?

☐

创建伸缩规则

取消

回到最佳拍档**负载均衡**的管理界面，我们也看到弹性伸缩组中的两台机器，已经位于后端服务器列表中了（这时可以将测试机 vm1-in-vpc1 从后端服务中删去）：

添加

云服务器名称

请输入名称或ID进行精确查询


<input type="checkbox"/>	云服务器ID/名称	地域	VPC	公网/内网IP地址	状态	权重	操作
<input type="checkbox"/>	ESS-asg-autoscale-group1 i-uf67wyymbgnnd69wkf31	上海 可用区E	vpc-uf6irhi8124x59al68761	192.168.1.89(私有)	✓ 运行中	50	移除
<input type="checkbox"/>	ESS-asg-autoscale-group1 i-uf68viqv1vrqntkpyiha	上海 可用区D	vpc-uf6irhi8124x59al68761	192.168.0.234(私有)	✓ 运行中	50	移除

我们试着来反复地访问负载均衡端的同一个入口 URL，会获得来自不同可用区中不同机器的响应，这说明负载均衡的**随机分发**起作用了：

复制代码

```
1 [client@clientVM ~]$ curl http://47.101.77.110/fibo/35
2 Fibo(35) = 14930352
3 Computed by iZuf68viqv1vrqntkpyihaZ with private ip 192.168.0.234
4 [client@clientVM ~]$ curl http://47.101.77.110/fibo/35
5 Fibo(35) = 14930352
6 Computed by iZuf67wyymbgnnd69wkf31Z with private ip 192.168.1.89
```

最后也是最精彩的部分，我们来使用 **siege** 命令来持续冲击这个负载均衡，使集群的平均 CPU 升高，看看它是否会自动扩容。

 复制代码

```
1 [client@clientVM ~]$ siege -c 15 -t 20m http://47.101.77.110/fibo/35
2 ** SIEGE 4.0.2
3 ** Preparing 15 concurrent users for battle.
4 The server is now under siege...
5 HTTP/1.1 200      0.14 secs:      88 bytes ==> GET   /fibo/35
6 HTTP/1.1 200      0.16 secs:      87 bytes ==> GET   /fibo/35
7 HTTP/1.1 200      0.28 secs:      88 bytes ==> GET   /fibo/35
8 HTTP/1.1 200      0.29 secs:      87 bytes ==> GET   /fibo/35
9 HTTP/1.1 200      0.41 secs:      88 bytes ==> GET   /fibo/35
10 ...
```

果然，流量到来后，虚拟机的 CPU 飙升，伸缩组就自动地进行了新实例的创建，一直达到了我们设定的十台上限，以满足汹涌到达的计算请求。



伸缩组的峰值状态

伸缩活动	变化后总实例数	开始时间	停止时间	描述	状态(成功) ▾	操作
asa-uf6ezrfgoyndoywm8II3	2	2020年1月31日 12:52	2020年1月31日 12:52	Remove "8" ECS ...	成功	查看详情
asa-uf6ezrfgoyndor0hsdnk	10	2020年1月31日 12:35	2020年1月31日 12:36	Add "3" ECS ins...	成功	查看详情
asa-uf6ezrfgoyndoj4dc5ph	7	2020年1月31日 12:18	2020年1月31日 12:19	Add "3" ECS ins...	成功	查看详情
asa-uf6iyjj2n706h8yt0kzh	4	2020年1月31日 12:16	2020年1月31日 12:17	Add "2" ECS ins...	成功	查看详情
asa-uf6h8gp7qsxr6tc4k85w	2	2020年1月31日 11:33	2020年1月31日 11:34	Add "2" ECS ins...	成功	查看详情

伸缩活动历史记录

当 siege 命令停止后，平均 CPU 大幅降低，伸缩组还能自动地缩容，减少实例数量。上面的伸缩活动的截图也体现了这个过程。

至此，我们的跨可用区负载均衡的实验就大功告成了。

你也可以结合你实际的场景，来进一步地实验和拓展这个范例。比如在生产环境中，你通常需要为负载均衡的外部 IP 绑定正式的域名；或者你的 Web 服务很可能不是完全无状态的，需要依赖后端数据库；再比如，你可以尝试在别的区域再建立一个 VPC，让两个 VPC 互相连接，新 VPC 可以作为冷备，或者承担日志数据分析的工作，这样能够形成一个类似“两地三中心”的强壮架构。

课堂总结与思考

今天涉及的点比较多，我们谈到了故障范围和故障处理，也谈到了云端的弹性优势。这次的实验也相对大一些，比较完整地构造了一个负载均衡加弹性伸缩的架构。不知道你掌握得怎样，有没有相关的问题，欢迎你在这里留言，和我一起探讨。

今天我留给你的思考题是：

大多数云上负载均衡产品都有一个重要特性，叫做“**会话保持**”，你知道它是用来做什么的吗？它的原理又是什么呢？

默认情况下，弹性伸缩服务会使用按量计费的虚拟机。那么成本上更有优势的包年包月虚拟机，或者竞价实例的虚拟机，能够融入弹性伸缩的体系吗？

好了，今天我们就到这里。如果你觉得有收获，欢迎把这篇文章分享给你的朋友。感谢阅读，我们下期再见。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

精选留言 (19)



何恺铎 置顶

2020-03-20

[上讲问题参考回答]

1. VPC内双机互联取决于机器的配置，云上单机内网带宽一般和自身性能正相关。许多云会给出具体数值，注意这里不要和公网带宽混淆。另外，两机物理距离也是实际效果的潜在影响因素，例如是否位于同一宿主机，是否位于同一可用区等等。
2. 反过来引导外界流量进入VPC，除了最简单的eIP绑定到虚拟机之外，主要考虑使用负载均衡和DNAT。



8



罗辑思维

2020-03-18

问题1

「负载均衡器」将同一客户端的会话请求转发给指定的一个后端服务器处理。
如何识别客户端：四层请求用源IP，七层请求用cookie。
如何赶走客户端：通过设置会话超时时间。

问题2

项目中以包年包月主机为主，资源不够时再触发生成按量计费虚拟机。

作者回复：赞



10



八哥

2020-03-18

如果代码要更新了，不知道镜像会不会自动更新，否则每次迭代发布，要重复执行自动伸缩的步骤。

作者回复: 镜像不会自动更新的。所以更新代码后需要重新打包镜像并替换原有镜像, 可以用脚本来完成这个操作。

共 2 条评论 >

👍 8



怀朔

2020-03-18

1、会话保持

为了同一个客户端多次连接是保证同一客户端多次连接路由到同一个服务上。

负载均衡关闭会话保持, 长链接在keep-alive的状态下, 也会路由到同一服务, 连接断开重连的情况下会负载均衡分布。

原理: 应该就是长链接吧...不是很懂。老师分析

2、计算问题 我觉得 有按量停机不收费。我觉得做的还是比较优秀的

按量和竞价本身的都可以容入弹性体系 核心的还是业务都是到底允不允许加入 目前阿里云 腾讯云 这一个只做到机器级别的弹 应用级别目前还没有到达 核心点 个人觉得主要点应用程序内容发布频率很高原因。

作者回复: 和长连接没有关系哦, 多次短连接也可以做到会话保持



👍 3



Christopher

2020-03-18

建议老师可在每节课开始之前说下上节课思考题的思路哈

作者回复: 谢谢你的建议。原本是打算后面统一整理回答的, 接下来会陆续在每讲的评论区给出前一讲问题的参考回答。



👍 3



一步

2020-03-18

像阿里云 SLB 服务, 后面的后台虚拟机是不是只能是阿里云的虚拟机? 也就是一个平台提供的IaaS服务只能和本平台提供的其他服务做对接? 不能和其他平台相互对接? 有没有办法突破这个限制?

作者回复: 一般来说, 云上的SLB只支持自家的虚拟机。如果想和其他云平台融合架构, 可以看看第三方厂商的多云解决方案。

共 2 条评论 >

👍 2



leslie

2020-03-18

跟着老师的课程一起学习确实感受到进步的收获: 不过可能个人会更加从实际的角度去看到一些本课程内容相关或者前面老师所答相关的问题。

关于之前课程的一些实践和感受:

- 1)购买了按流量的刻意数日没管, 发现总代价却是明显要比包月高出不少
- 2)上次课程中提及的传输和整体硬件相关, 这个个人觉得确实操作方面比私有云的灵活性就要显的差

今天课程的感受和问题:

- 1) 跨云应当是解决区域问题, 如区域断网或者整体硬件故障; 记得这种故障每年都会有片区级的发生, 这个应当是至少需要云厂内做异地多活或者跨云厂商吧? 尤其是对于数据文件。
- 2) 课程中的负载均衡个人感觉和现实中的负载均衡类似, 按流量会自动使用在服务器区域扩展的很大, 包年包月这个似乎。。。

关于今天课程的问题:

- 1) "会话保持"没有研究过: 这个从概念上去理解应当是一个connect或者一个session; 后续课后待进一步实战研究
 - 2) 包年包月的核心思路走的时候是和公有云类似: 有用过一些包年包月的, 整体感觉和公有云在诸多方面还是相对类似, 只是可扩展方面不像现实环境需要去人工添加硬件设备;
- 课程到今天老师想提及或阐述的一些思路已大致明了, 开课时所不明的诸多事宜已知其方向; 谢谢老师持续的解惑, 课程所需到今天已基本达到只待今天课程的答案。谢谢

共 1 条评论 >

👍 2



丁乐洪

2020-03-22

老师会讲一下迁移上云的注意点吗?

作者回复: 第8讲中我们会提到迁移, 你可以看一下。



👍 1



小狼

2020-03-19

“负载均衡器本身也是需要是高可用的”，这里需要建立两个同样的负载均衡器吗？

作者回复：不需要两个负载均衡。云上负载均衡的内部实现本身包含了高可用设计。这里我们还设置了备可用区，当主可用区故障时这个负载均衡实例仍能正常工作。

共 2 条评论 >

👍 1



我来也

2020-03-18

之前在阿里云上购买实例时,在最后一个界面上,有个选项是"部署集".

之前一直没有配置过,没有研究.

听老师这么一说, 原来是做这个用途的,哈哈!

在购买k8s的工作节点时,我觉得可以把这个功能用起来.

平常的服务,还是没有太大的必要.毕竟本身还都是单节点的,一个坏了, 就整个都不可用了.说起无状态的服务, 现在的云上,各种服务太全了, 只需要花钱就可以买现成的.

这样把有状态的数据Mysql Mongodb Redis, 把中间件Kafka, 把ES服务等都买现成的.

很容易可以把处理业务的模块改造成无状态的服务.

本身云服务就是高可用的,再自己把无状态的节点按老师说的,弄成多可用区,多机架.几乎也是高可用的了.



👍 1



Geek_761876

2022-06-26

弹性缩减实例时，如果实例正在运行生产业务，这时云平台一般会怎么解决这个问题？



Seven.Lin 澤耿

2020-05-15

1. lb都有类似的会话保持机制，这是lb的功能，例如用最简单的nginx也可以通过设置iphash的方式实现，就是把同一个客户端的请求转发给同一个服务端

2.可以的，类似线程池中core的可以一直活着





sipom

2020-04-01

在系统设计中，需要分析系统的故障概率，请问哪里能获得云平台上单台节点的故障率数据？
谢谢！



sipom

2020-04-01

请问下，云平台上单台节点的平均故障率，您有相关数据吗？



吴小智

2020-03-25

等我有 money 了，我也试一下



戴斌

2020-03-20

回话保持类似于nginx的sticky模块



TKbook

2020-03-18

真心建议老师本节末尾解答一下上一节的思考题

作者回复: 没问题。原本是打算后面统一整理回答的，接下来会陆续在每讲的评论区给出前一讲问题的参考回答。



我来也

2020-03-18

会话保持 的作用和原理

常见阿里云的文档[会话保持常见问题](https://help.aliyun.com/knowledge_detail/55202.html)

k8s中的Service也可以配置 session保持.

目的就是某些请求始终转发给同一个后端.

弹性伸缩 是否可以使用包年包月和竞价实例的虚拟机.
我觉得是可以的,只是官方的界面上没见这么人性化的配置.
按量付费适用的场景多,整体价格比包月付费和竞价的价格贵多了.

但是,在有必要的情况下,完全可以自己调用云厂商提供的API服务,自己弄.
自己调用API接口查询服务器负载, 自己购买机器, 加入集群.
本质上,人工可以操作的事情,几乎都可以调用API接口实现.



夜空中最亮的星

2020-03-18

弹性伸缩功能不错，还没实际用过

