双机房灾备架构搭建实践

2017-08-23 12:21 | 阅读 4.8k | 评论 0

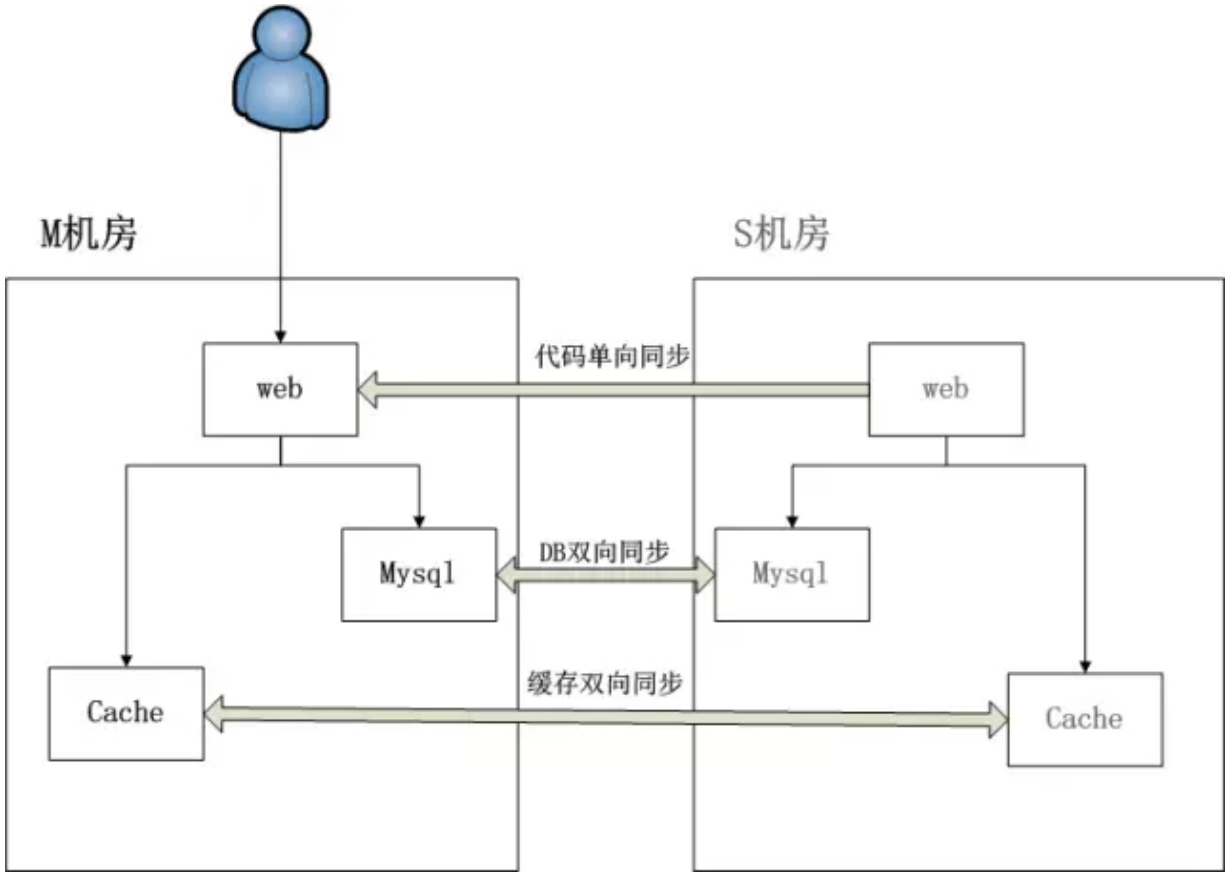
1. 背景

作为运维哥，每逢佳节倍担心，除了定期到机房贴符拜拜，我们还有什么办法，保障我们的服务不受硬件故障的困扰呢。

对很多互联网企业来说，服务长时间停摆，意味着无法估量的损失。接下来我们介绍一个简单快捷的方法，搭建双机房灾备服务。当业务所在机房发生网络或者机器故障时，能够迅速转移并恢复我们的服务。

2. 双机房灾备架构

不同产品的组件和架构差别很大。下面是基于lnmp的web站点，最基本的双机房灾备服务架构， 仅供参考。



M机房为主机房，S机房为冷备机房，满足现有业务，并且方便未来扩展的前提下，我们尽可能把架构设计得越简单越好。华丽而复杂的组件，通常也伴随着可观的维护成本。

3. 一步步搭起来

3.1 web程序同步

web程序同步其实就是文件的同步，最简单的就是定时rsync。不过这个方式太古老，后来我们测试了lsyncd和sersync两款基于inotify的实时同步工具。最终选定整体性能和稳定性表现更好的lsyncd作为我们的代码同步工具。

这里简单贴下适合我们使用场景的测试条件和结果：

- 测试目录包含8215个子目录，51587个文件，共 4.3G。
- 源机和目标机均为标配8核16G内存146G硬盘，内网通过1000M交换机互联。
- 源机服务开启sshd，lsyncd/sersync。
- 目标机服务开启sshd，rsync daemon。

测试结果如下：

	lsyncd	sersync
全局同步(不带 z)	45s	44s
全局同步 (带 z)	70s	71s
解压/cp	解压约耗时 1min (同时开始同步), 继续同步耗时 21 ~ 23s	解压约耗时 1min (同时开始同步), 继续同步耗时 2min 24s ~ 2min 26s
rm	<60s	>4min , 实际同步完成后仍有同步操作 (应该是大量失败重试), 约 8min 才完全停止同步
mv	约 10min 后同步 , 实际同步耗时约 1min 40s	接近全局同步

lsyncd的官网： https://github.com/axkibe/lsyncd

rsync daemon和lsyncd的安装方法就不再赘述。这里给一个lsyncd同步web程序的参考配置（lua语法）：

```
settings { logfile = "/usr/local/lsyncd/logs/lsyncd.log", statusFile = "/usr/local/lsyncd/logs/lsyncd.status", maxDelays = 100, delay = 5, exitcodes = {[0] = "ok", [1] = "again", [2] = "die"}, maxProcesses = 5, statusInterval = 5 } — 同步到主机 sync { default.rsync, source = "/opt/web/ser", target = "rsy_user@19.68.111.222::ser_master", — 排除隐藏文件及临时文件，可以按具体需求修改 exclude={ ".*", "*.tmp", "*.bak", "*.swp" }, rsync = { compress = false, archive = true, verbose = true, — _extra用于手动指定rsync参数，默认建议设置超时，避免rsync进程卡死 _extra = {"–timeout=3600"}, password_file = "/usr/local/lsyncd/etc/ser_master.pas" } }
```

还可以把web服务端的配置文件（比如nginx vhost配置文件）也加到lsyncd自动同步中，然后定义触发重新加载配置的命令，这个就交给你自己发挥了。

3.2 mysql数据同步

mysql数据同步，是利用mysql双向主从来完成的，俗称mysql主主同步。

两个机房之间mysql数据同步，要考虑数据传输安全性和避免数据冲突。

(1) 数据传输安全性

可以参考之前我们的文章《远程访问mysql？教你为数据传输再加把安全锁！》。 下面介绍基于ssl进行数据加密同步。

首先使用openssl或者easyrsa配置生成ca、私钥和证书等。生成好证书后，配置到my.cnf中：

master

```
[mysqld] server-id=1 ssl ssl-ca=/opt/ssl/cacert.pem ssl-cert=/opt/ssl/srv.crt ssl-key=/opt/ssl/srv.key
```

slave

> [mysqld] server-id=2 ssl ssl-ca=/opt/ssl/cacert.pem ssl-cert=/opt/ssl/srv.crt ssl-key=/opt/ssl/srv.key

(2) 避免数据冲突

mysql采用奇偶分离的配置，表在设计时以自增id为主键，一边写入id为奇数，一边写入为偶数，即使发生两边同时写入的情况，也能保证主键不冲突。在前面的配置，继续增加下面内容：

master

> [mysqld] ##自动增量为2，允许最多2台数据库实例加入。 auto_increment_increment = 2 ##表示偏移值，每个数据库的偏移值必须唯一，且在1和auto_increment_increment之间。 auto_increment_offset = 1

slave

> [mysqld] auto_increment_increment = 2 auto_increment_offset = 2

3.3 cache数据同步

稍有点用户量的网站，web程序直接读写mysql，机器很快会撑不住，这时就少不了缓存技术的使用。

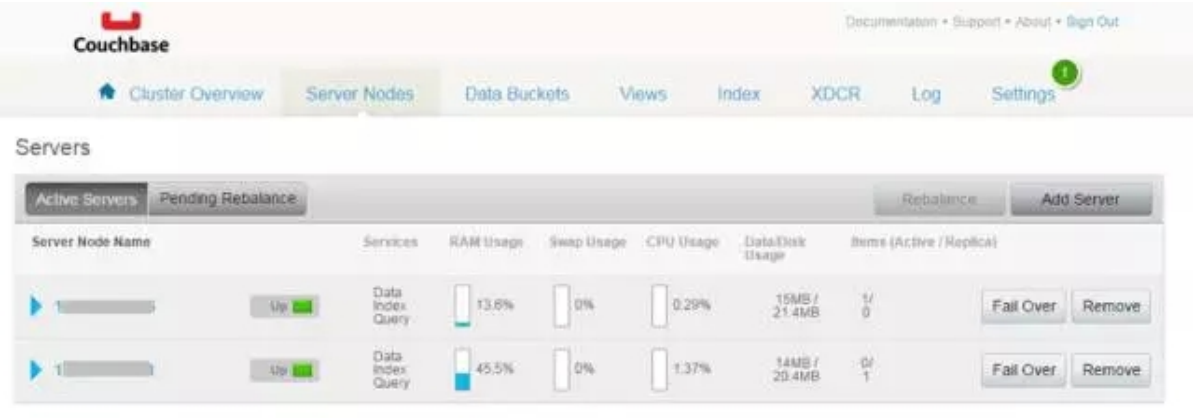
我们选用安装简单（rpm方式）、配置管理（web方式）同样简单的couchbase，而且完全兼容memcached的使用。

官网下载链接：https://www.couchbase.com/downloads

安装方案参考官网文档。

安装完成后，可以看到我们两个机房两台机器在同一个集群里，缓存数据就能够相互同步了。

异地服务两个节点：



memcached实例：

3.4 使用dnspod配置D监控

dnspod为我们解析站点域名，同时有一个D监控功能，在我们的双机房灾备中，起到自动进行故障转移，切换到其他正常运行机器的作用。

4. 结语和新的开始

到这里，我们已经明白搭建双机房灾备服务的关键步骤。不过想想这套冷备方案也有一些明显的缺点，比如

(1) 备机一直空跑，虽然可以拿来做定时数据备份，但是否还是有点浪费资源。

(2) 如果运气不错，一年半载都没切换过备机，如果下次真的需要切换时，是否有信心立即切换过去。

(3) dns切换期间，域名解析大概需要5分钟生效时间，其实用户是受影响的，可否把影响再继续降低些。

要解答这些问题，我们需要更高级的设计架构。冷备让我们心中有了保障，然而异地双活才是优雅的远方。

在异地双活改造时，一开始想着把冷备激活就可以了。然后发现有些业务必须依赖于两地数据的强一致性。就算拉个专线，或者提供多线路的切换容灾，但因为网络延时的不稳定，数据的一致性还是没法绝对保证。甚至有可能被利用来恶意重复刷取站点资源。

在茫然无解的时候，因为一个不完美思想的指导，重新找回了方向。接下来将从这几个方面进行异地双活的实现：

(1) 从业务层面，梳理出重点核心业务，优先考虑核心业务的异地双活。

(2) 再按双活实现难易程度，把既核心又容易实现的业务排在第一位实现。

(3) 同时一个业务流程下来，中间涉及的小业务模块也跟着实现异地双活，保证核心流程在一个地方就能独立顺利完成。

举个例子，有一个站点包括注册、登录、充值3个服务。

从业务量和影响面来分析，我们发现登录请求占90%以上，然后登录刚好也是最不依赖于数据实时一致性的业务，用户注册完，资料就开始进行同步了，以后再登录的时候，数据早已经两地同步完成，就算在两地快速切换，最多因为缓存的丢失，只是要求用户重新登录一次，不会出现登录失败的情况。

而注册或者充值平时业务量小，如果做了双活，在出现故障，快速进行两地切换时，可能存在注册信息冲突不一致，或者两地重复充值的情况。所以平时我们只要给登录、充值做好冷备服务就可以了。