脑裂的来龙去脉

相信很多非专业人士都听过脑裂，但是对于到底什么是脑裂，可能就一知半解了。由于之前很多同事问什么是脑裂，很早就想写一篇文章阐述一下脑裂问题，今天终于能够静下心来写一篇关于脑裂的文章，本文尽量以浅显易懂的方式为大家解惑，什么是脑裂，脑裂怎样出现的，怎样解决脑裂。

1、什么是脑裂

脑裂是基于副本的分布式存储特有的一种数据不一致问题，是的，脑裂是数据不一致的一种，但是数据不一致并不一定是脑裂。搞晕了~~~，不用急，看下面的 3 张图：

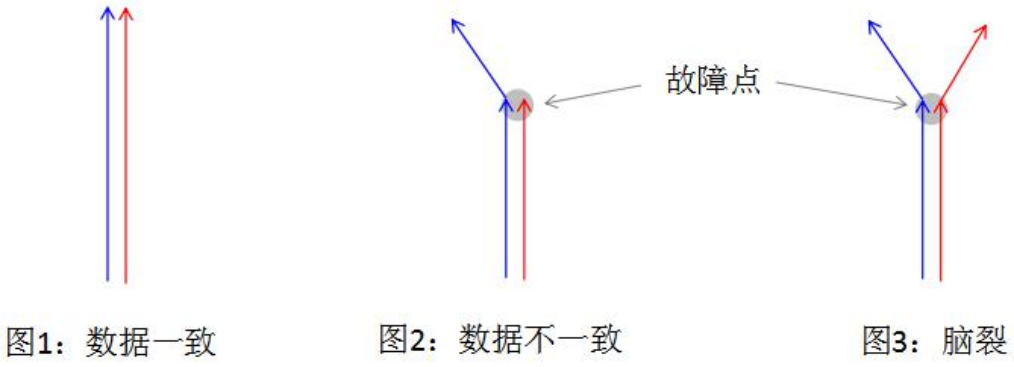


图1：数据一致　　图2：数据不一致　　图3：脑裂

上面 3 张图中，蓝色、红色线条分别表示 2 个副本的数据更新过程。

图 1：两个副本的数据更新完全是一致的，2 个副本处于数据一致状态；

图 2：2 个副本在第一阶段的数据更新是一致的，但是到了故障点之后，只有蓝色的副本被更新了，红色的副本由于离线或其他原因没有得到数据更新，还是处于旧的状态，这时 2 个副本数据不一致了，但是这 2 个副本还没有发生脑裂；

图 3：故障发生后，2 个副本的数据也不相同了，但是跟图 2 不同的是，在故障点之后，蓝色副本和红色副本被分别更新了，2 个副本被写入了不同的数据，走了不同的岔路，这时 2 个副本发生了脑裂。

我们来看看图 2 跟图 3 的区别：尽管 2 张图中，2 副本都是处于数据不一致状态，但是图 2

的 2 个副本是一新一旧，只是数据不一致，并没有脑裂；而图 3 的 2 个副本分别被更新了，形成了“Y”字型更新图，脑裂了，**记住：“Y”字型就是脑裂的标志**。

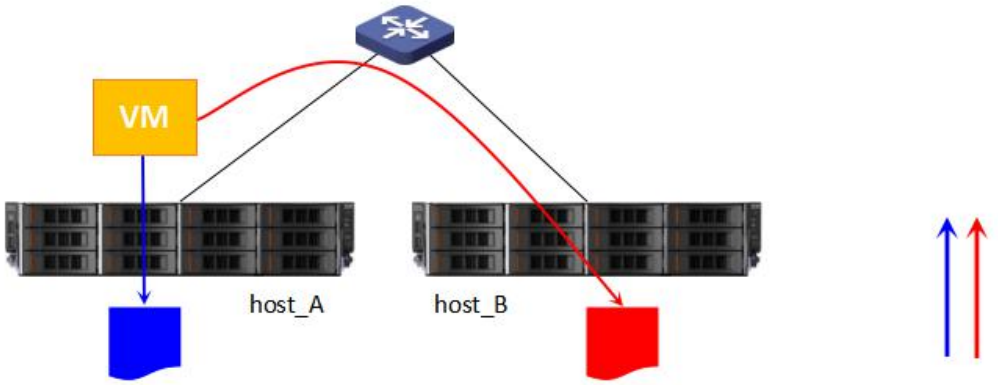
2、脑裂是怎样出现的

那么脑裂是怎样出现的呢？我可以随便举几个例子：

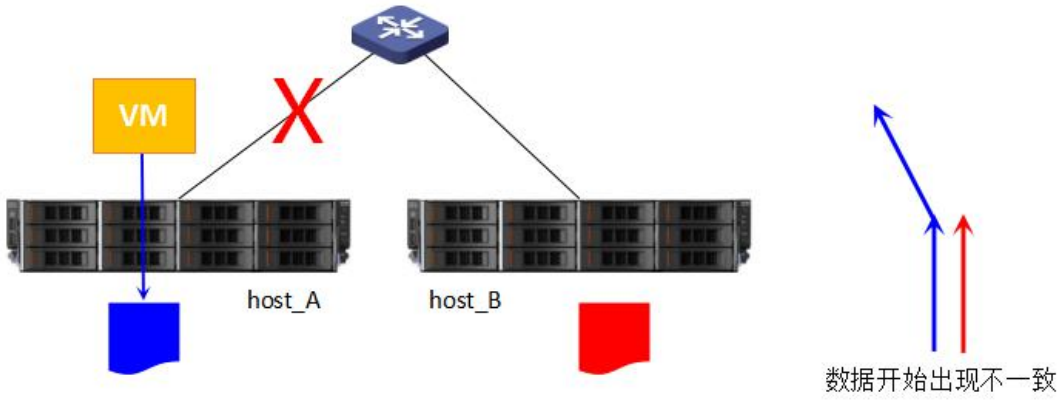
假定有 host_A、host_B 2 个主机，2 个副本分别存储在 host_A 和 host_B 上面

例 1：

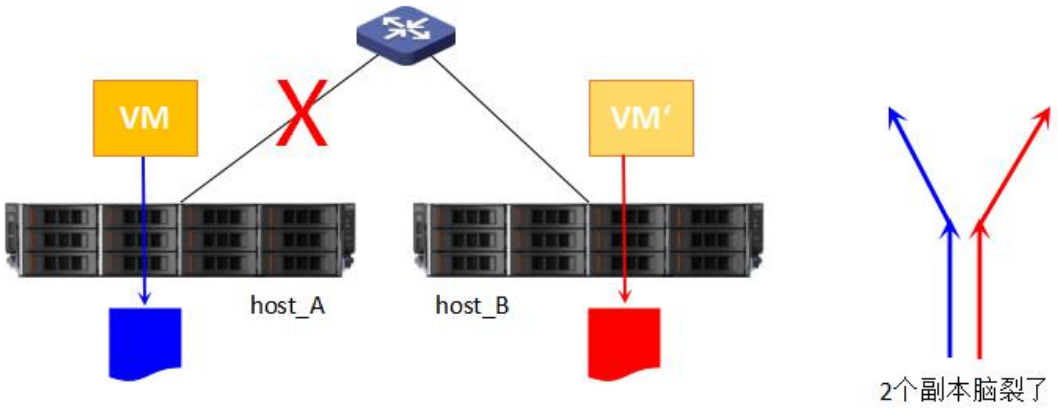
第一步：虚拟机运行在 host_A 上面，最开始的时候，一切正常，2 个副本处于一致状态



第二步：host_A 主机的所有网络都断了，host_A 完全失去了跟 host_B 的联系，这时候，在 host_A 上的虚拟机是还在运行的，但是由于网络断了，VM 只能往蓝色的副本写数据，没办法往红色的副本写数据，这时候，蓝色的副本被 VM 更新了，但是红色的副本还处于旧状态，一新一旧，数据不一致了，但是还没有出现脑裂。

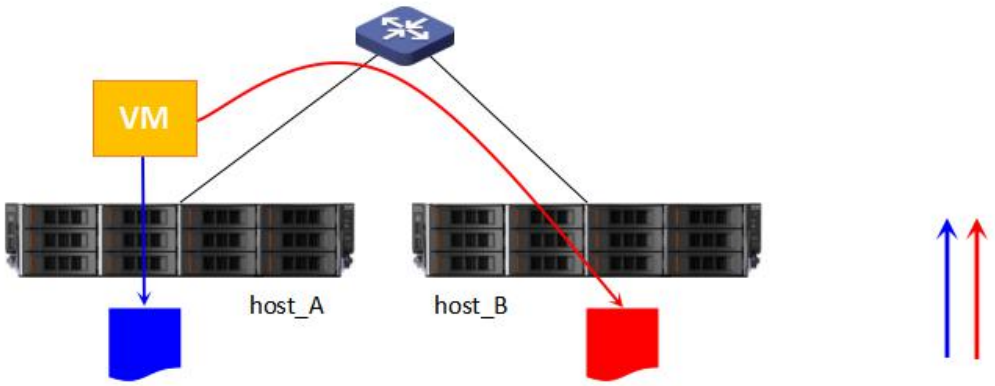


第三步：这个虚拟机启用 HA 机制，host_B 经过探测，发现没有找到 VM 虚拟机，这时候，HA 机制生效，host_B 重新启动了 VM 的一个新实例，记为 VM’，VM’运行在 host_B 上，并且往红色副本写入了新的数据，这时候，红、蓝两个副本出现了“Y”字型分岔，脑裂了。



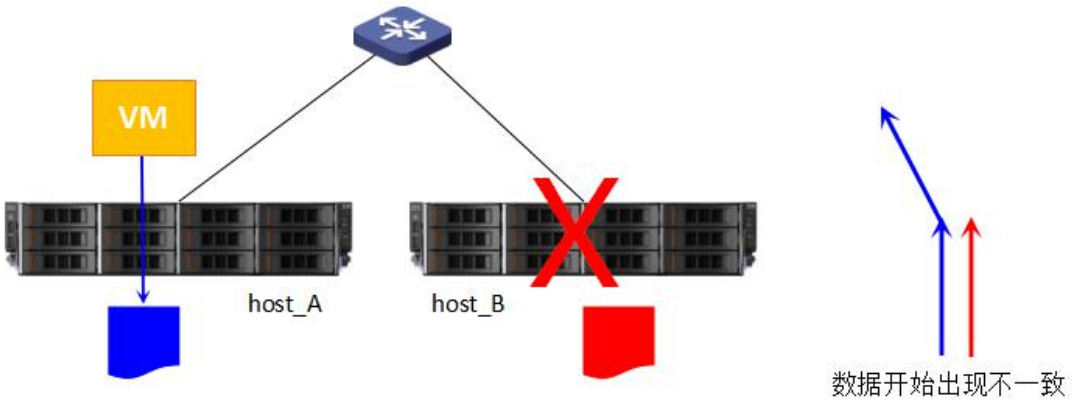
例 2：

第一步：虚拟机运行在 host_A 上面，最开始的时候，一切正常，2 个副本处于一致状态

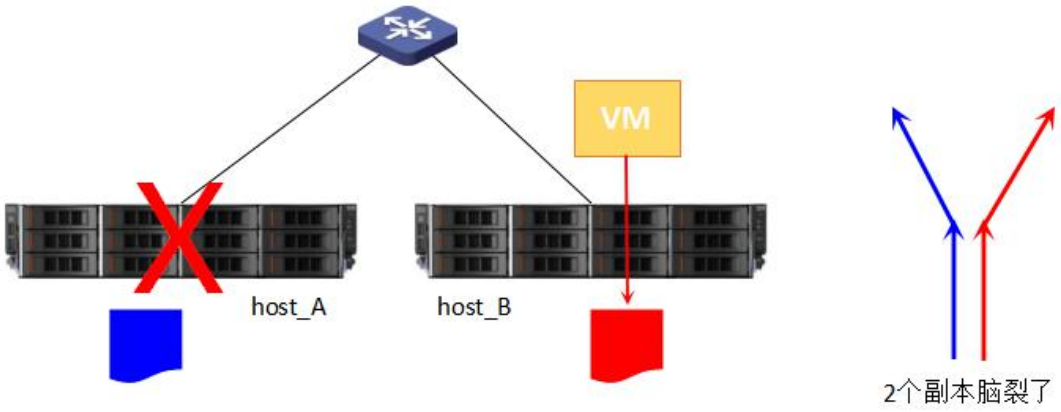


注：蓝色、红色分别表示 2 个副本

第二步：把 host_B 关机，这时候 VM 只能往蓝色副本写入数据，红色副本的数据得不到更新，两副本处于一新一旧状态，出现了数据不一致，但是还没有出现脑裂。



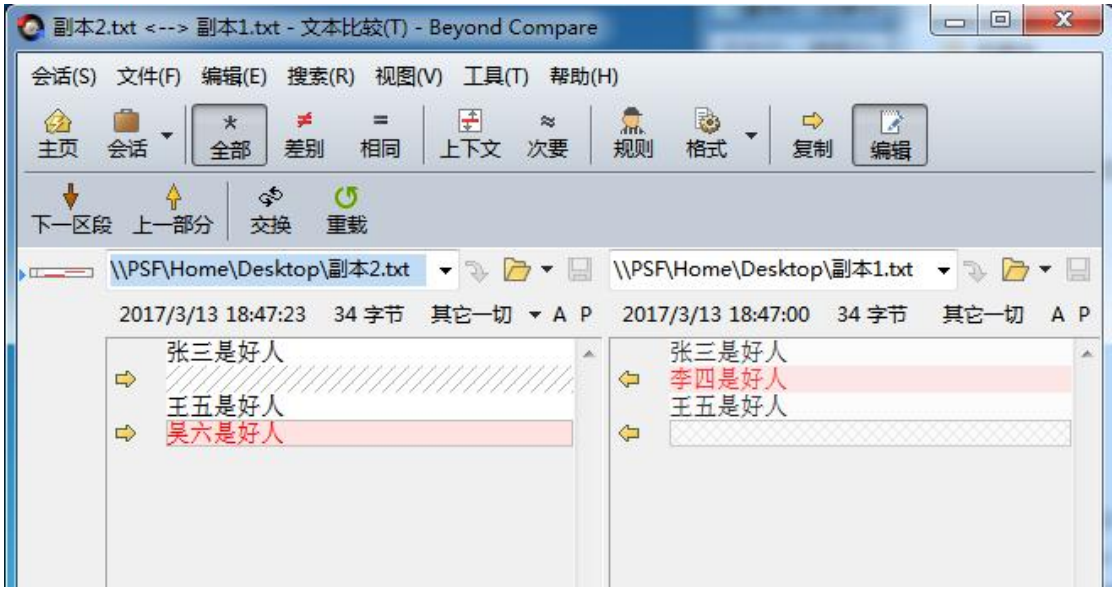
第三步：客户机房停电了，过了一段时间，重新来电，碰巧 host_A 的电源线烧坏了，host_B 起来了，启动了一个虚拟机，只能写红色副本，这时候，2 个副本出现了“Y”字型分岔，脑裂了。



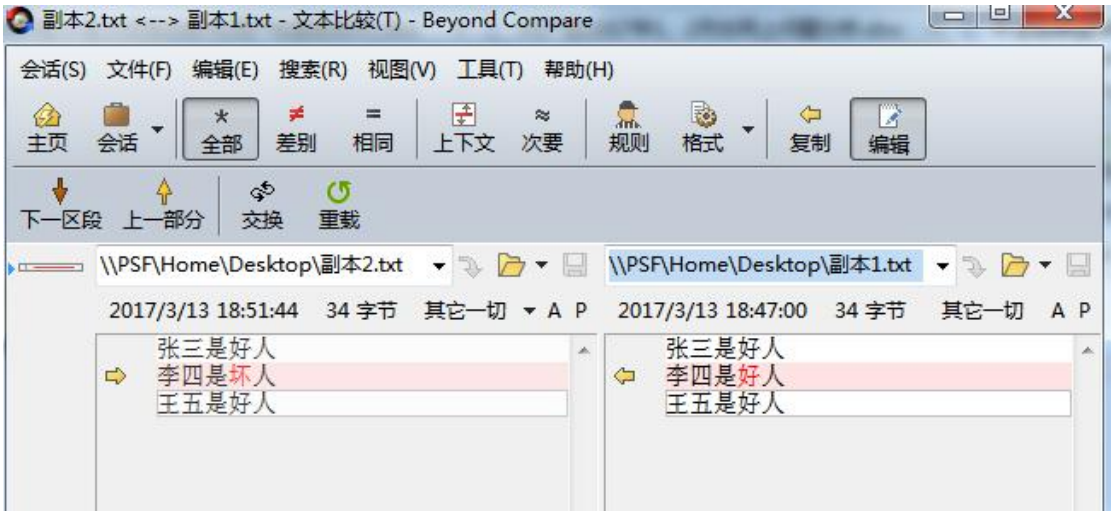
3、怎样解决脑裂问题

不知大家对第 1 节中关于脑裂的描述理解了没有，2 个副本分裂了，我有部分数据是你没有的，你有部分数据是我没有的，即“Y”字型上面分岔的那部分数据。脑裂一旦出现，分裂的数据是没办法合并的，唯一的办法是采用其中一个副本的数据，丢弃另一个副本的数据，所以一旦出现脑裂就意味着数据的丢失。

为什么说两个副本不能合并呢？



如果 2 个副本是上面这样的，当然是可以合并的。



如果 2 个副本是这样的，怎么合并呢？而在现实世界中，一旦发生脑裂，都是这种情况，特别是非 txt 文件，一定是这样的，合并不了。

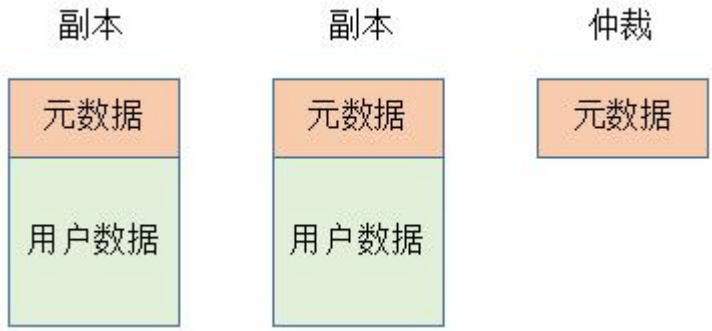
既然发生脑裂就意味着丢数据，那么脑裂的解决办法就是不能让脑裂发生，阻止脑裂发生的理论性基础就是仲裁机制，**仲裁机制的核心思想是“少数服从多数”**：

对于 n 个副本，为其设置 n-1 仲裁，副本+仲裁总共 n+(n-1)=2n-1 个对象，只有超过一

半的对象都在线的情况下，才能写入，否则禁止写入。对于 2 副本来说，就是 2 个副本+1 个仲裁，总共 3 个对象，3 副本就要设置 3 个副本+2 个仲裁=5 个对象，依次类推。

什么是仲裁呢？

VMware 把它叫 witness，我们可以把仲裁看成一个特殊的副本，它只有元数据，没有用户数据，因此仲裁占用的空间很小，如下图所示：

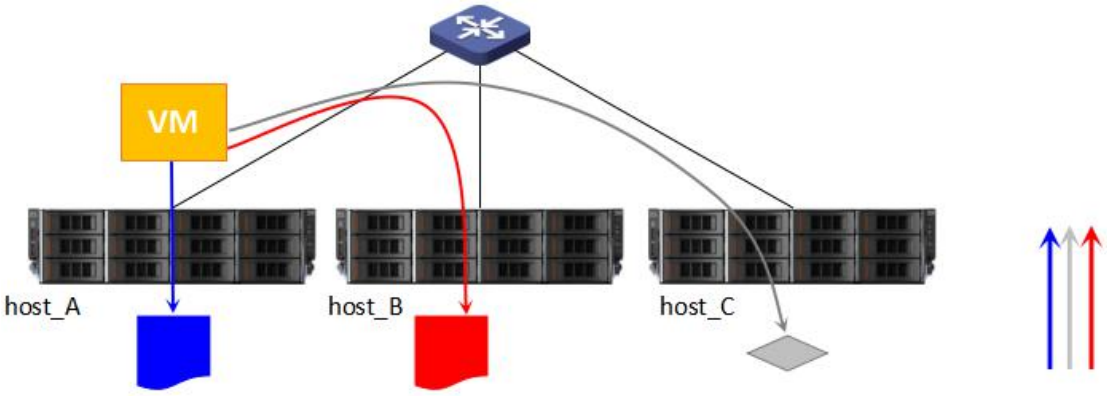


下面我们就以 2 副本为例说明一下，仲裁是如何阻止脑裂的发生的：

由于 2 副本+1 仲裁总共有 3 个对象，所以至少需要 3 个主机才能支持，假定 3 主机分别是 host_A、host_B、host_C，2 个副本分别位于 host_A、host_B，仲裁位于 host_C。

例 1：（跟上面第 2 节的例 1 一模一样）

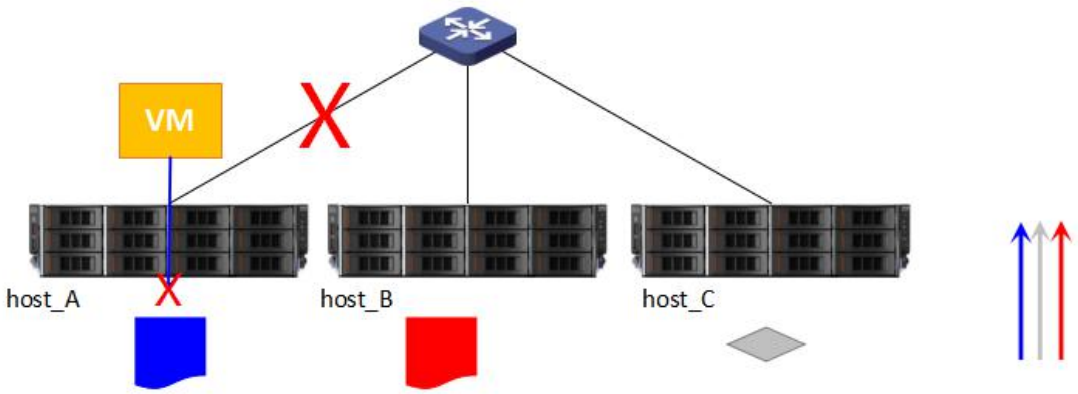
第一步：虚拟机运行在 host_A 上面，最开始的时候，一切正常，3 个副本对象都处于一致状态。



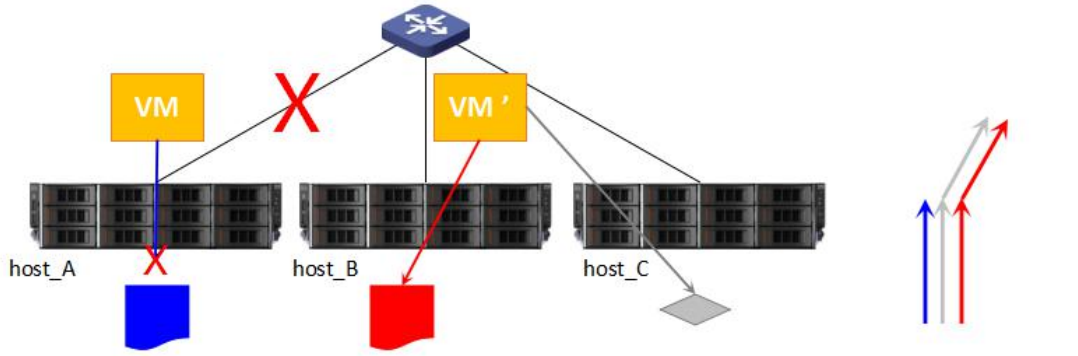
注：蓝色、红色表示副本，灰色菱形表示仲裁

第二步：host_A 主机的所有网络都断了，host_A 完全失去了跟 host_B、host_C 的联系，这时候，VM 只能看到蓝副本，红副本和仲裁都跟这个 VM 失去了联系，3 个对象中只有 1

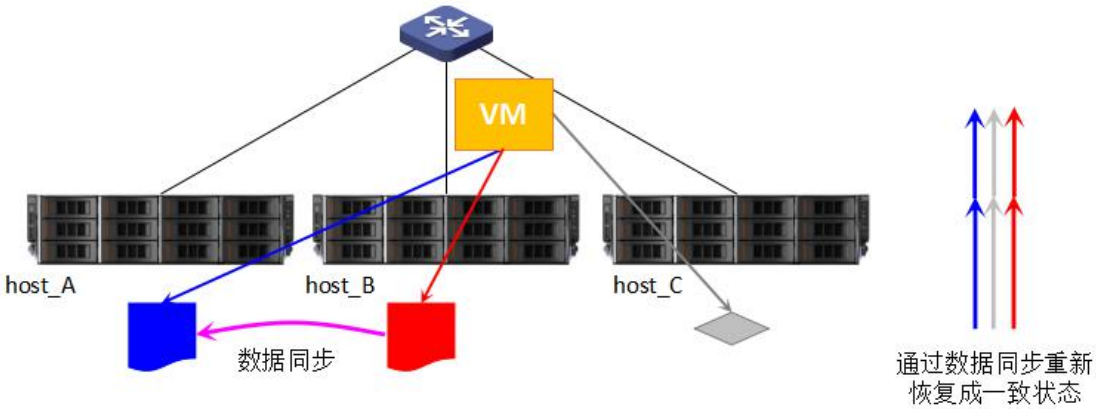
个对象在线，少于一半，仲裁逻辑就会阻止 VM 写入到蓝副本，这时候 3 个对象还是处于一致状态。



第三步：由于 host_A 离线，host_B 和 host_C 检测不到虚拟机 VM，于是就重新拉起了一个虚拟机实例 VM’，由于 VM’可以同时看到红副本和仲裁，超过半数了，所以它可以正常往红副本写入数据，同时，更新红副本和仲裁的元数据。这时候由于 VM’没办法写入到蓝副本，两副本处于一新一旧状态，并没有出现“Y”字型。

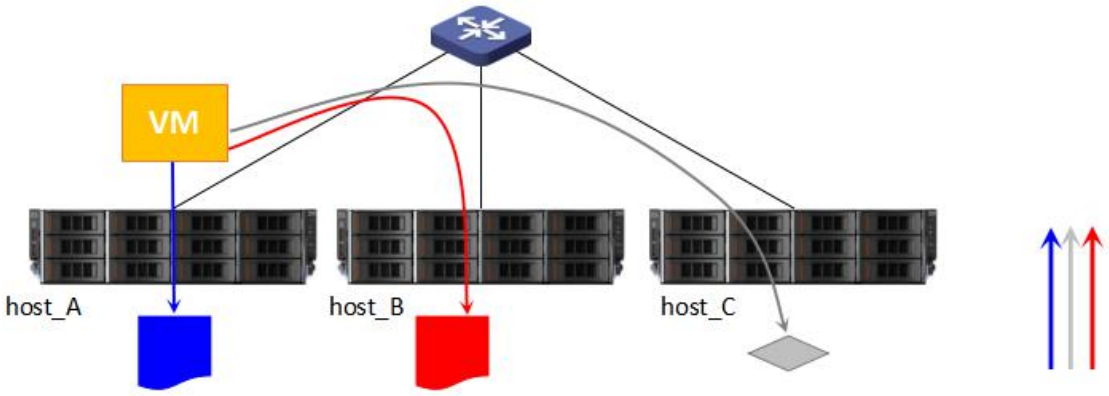


当 host_A 网络恢复后，host_A 上挂起的虚拟机实例被 kill 掉，留下 host_B 上的实例，由于这种情况只是数据不一致，软件可以自动修复，只要把红色副本的数据同步到蓝色副本，就可以重新让三者恢复到一致状态了。

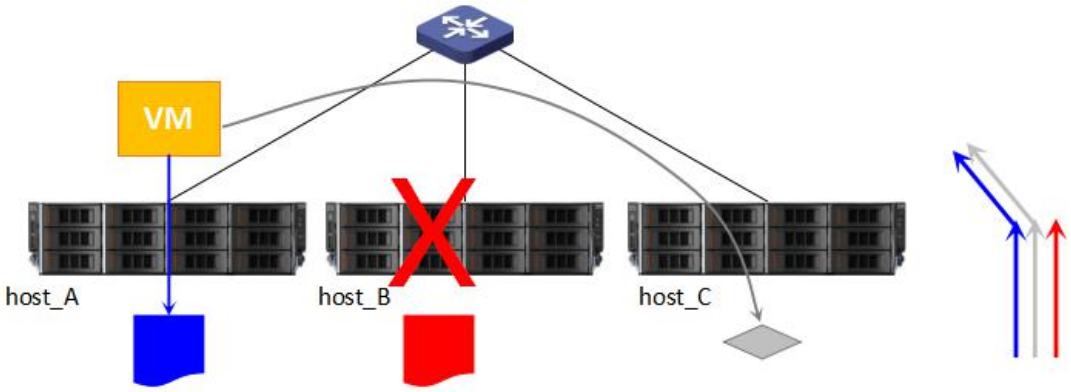


例 2：（跟上面第 2 节的例 2 一模一样）

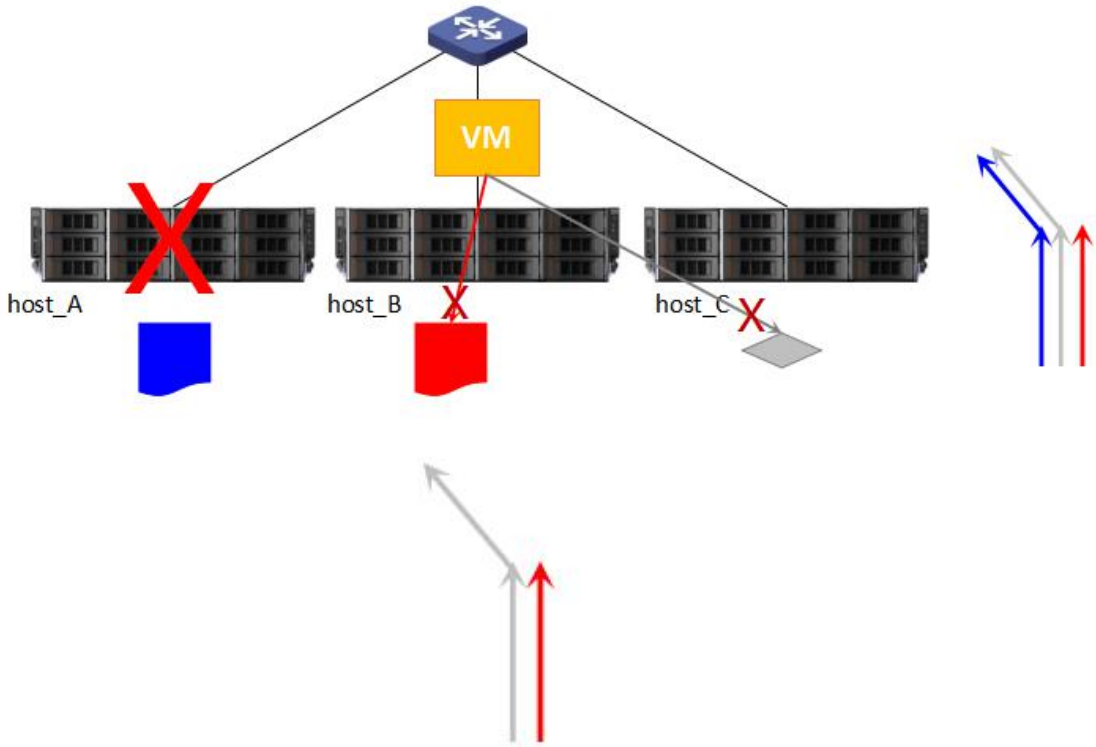
第一步：虚拟机运行在 host_A 上面，最开始的时候，一切正常，3 个副本对象都处于一致状态。



第二步：把 host_B 关机，这时候，VM 可以看到蓝副本和仲裁，超过半数，所以可以正常往蓝副本写数据，蓝副本和仲裁都被更新了，而红副本由于离线处于旧状态，这时候出现了数据不一致。

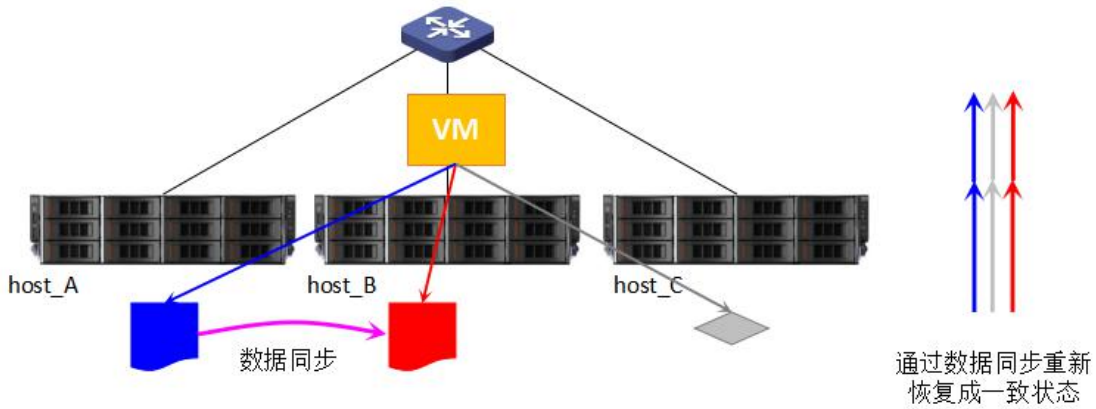


第三步：客户机房停电了，过了一段时间，重新来电，碰巧 host_A 的电源线烧坏了，host_B、host_C 起来了，host_B 尝试启动新的虚拟机实例 VM，这时候，红副本和仲裁都在线，超过了一半，但是仲裁会跟红副本进行元数据检查，一检查，发现红副本的数据是旧的，蓝副本才是新数据，但是已经离线了，所以这时候，仲裁逻辑会阻止对红副本的写入，阻止了“Y”字型的产生，所以也不会发生脑裂。



仲裁跟红色副本进行元数据对比，发现红色副本是旧数据，因此禁止对红色副本进行写入，避免了“Y”字型的产生，避免了脑裂。

当用户把 host_A 的问题解决掉并重新上线后，蓝色副本和仲裁都同时在线了，虚拟机就可以恢复 IO 了（对蓝色副本做 IO），同时蓝色副本的数据会同步到红色副本，让三个对象恢复成一致状态。



从上面的 2 个例子可以看到，通过引入仲裁逻辑后，由于至少要有超过半数的对象在线才允许写数据，这就阻止了脑裂的发生。

4、深信服脑裂解决方案

1）>=3 主机

HCI5.2（VS2.3）版本开始，引入了仲裁机制，VDI 会在下个版本 VDI5.3 版本合入，>=3 主机时，都有仲裁机制，不会发生脑裂。

1）2 主机

对于 2 主机的情况，深信服 HCI 做了特别的处理，对存储网络的可靠性做了 2 个改进：

- 每主机拿 2 网口做存储口，用 2 根网线直接对接，做成链路聚合，极大地提升了存储网络可靠性；

- HCI5.3（VS2.6）版本开始，提供了网络借用功能，当做了链路聚合的存储网络都断掉后，借用管理网络或数据通信网络作为存储网络，所以只有存储网络、管理网络、数据通信网络 3 张网络都断后，虚拟存储才会发生网络断开的问题。

通过上面 2 个牛逼的存储网络可靠性改进技术，保证存储网络具备非常高的可靠性，所以在客户真实的环境中，2 主机发生脑裂的可能性几乎为零。