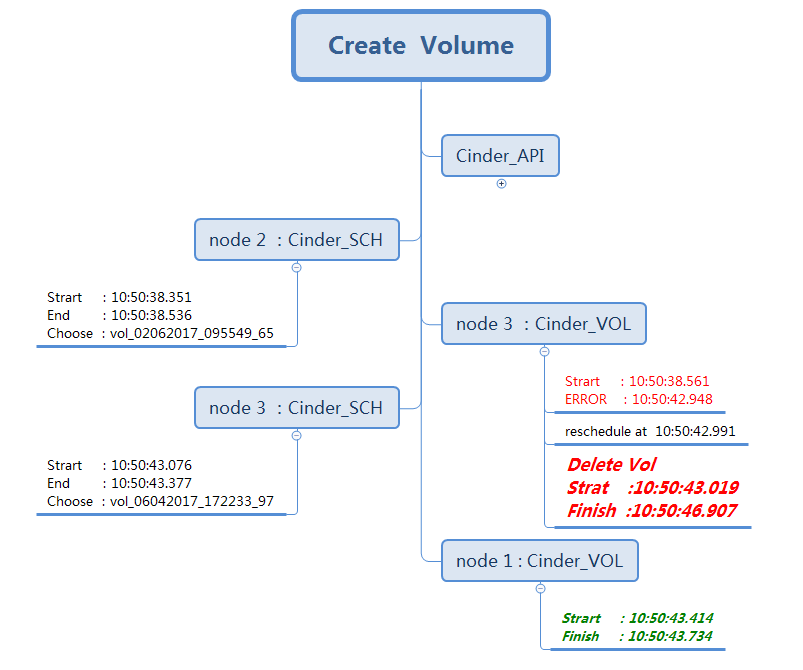
reschedule造成的创建卷失败问题分析

# 问题背景

当Cinder创建Volume失败（原因可能是选择的Pool中的LUN数目已经达到上限或者其他原因）时，Cinder会执行reschedule选择一个新的Pool，并且成功创建出Volume。在多节点部署的情形上，如果上述情形存在同步问题，导致Cinder会在Stroage删除创建的LUN，此时在stroage上找不到volume，但是在cinder上该volume的状态为available。

根据生产环境的日志整理时间轴如下：



从上面的图看出，第一次尝试在Storage创建卷的动作发生在Node3，但是失败（***原因是：Volume contains the maximum supported limit for LUNs***），因此在Node3开始re-schedule。

***re-schedule包括了两个环节：***

1.通知c-sch重新调度，选这一个新的Pool。选择完成后由c-sch通知c-vol在后端创建，这里通知到的c-vol并不一定是之前那个。

2.通知后端删除失败的LUN，尽管这个LUN可能并没有在后端被创建出来，但是删除操作还是会执行一次。

re-scheduler 包含了几个异步过程，不能保证完成的先后顺序，因此可能出现在node1节点已经完成了创建volume，但是node3删除Lun的过程还没有完成。导致node3最后删除的是node1创建的LUN。

上述过程中node3 删除LUN时，删除了 node1 新创建的LUN导致了整个问题。

实际上这里有逻辑上的错误，node1和node3创建LUN位于不同的Pool，为啥node3删除时删的是node1的？？？

re-schedule删除LUN时，调用的是Storage驱动的删除接口，具体的删除逻辑由驱动层代码实现。

**def delete\_volume**(self, volume):  
 *"""Deletes a volume.  
 If volume\_type extra specs includes 'replication: <is> True'  
 then the driver needs to delete the volume replica too.  
 """* **return**

delete\_volume的入参是volume对象（即数据库中volumes表的一条表记录），其实现逻辑一般是这样：

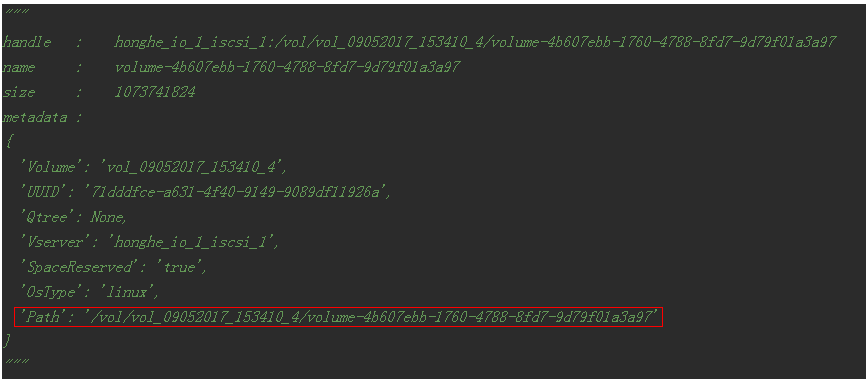
1.    通过volume中的信息定位出 LUN 在Storage 中的唯一标识符；

2.    发送请求到后端删除LUN；

在netapp中LUN的唯一标识符是 /vol /pool\_name /volume\_xxxxxxxx （path），并通过以下方法将cinder中的volume\_id 和 lun的 path 映射起来：

netapp驱动在内部维护一个字典（cinder.volume.drivers.netapp.dataontap.block\_base.NetAppBlockStorageLibrary.lun\_table）,字典中通过 volume\_id 对 LUN的信息进行映射：

字典里的LUN信息如下，删除时netapp 通过 metadata["Path"]进行删除。

******

所以问题的根本原因是这样的：

假设要创建卷的id为：volume\_123 ，第一次要调度Pool为A， 第二次成功调度的Pool为B。

1.  node 3 创建了一个卷，netapp在驱动中建立映射信息，但是创建失败，重新调度；

2.  node 1 重新调度，创建卷成功，映射信息： {volume\_123 ：/vol/B/volume\_123}。

3.  node 3 要删除/vol/A/volume\_123， 但是实际通过volume\_123映射到了/vol/B/volume\_123，导致错删（node 3 获取volume\_123的信息时，会先在内存中找，查找失败后直接从Storage获取，但是此时获取到的是/vol/B/volume\_123）

修改方案：

1.修改netapp 驱动，每次删除前判断DB中的Pool 信息是否和driver内存中的一致，如果不一致不能删除！

2.修改reschedule的逻辑，每次reschedule发生后使用为volume指定一个新的name\_id（相当于每次rescheduler都在后端建立一个新的映射）。