***Cinder新attach&detach接口分析***

# 背景

在Ocata版本中，Cinder针对attach/detach操作开发了一套新的API[（参考）](http://specs.openstack.org/openstack/cinder-specs/specs/ocata/add-new-attach-apis.html)。新API的版本号为3.27，主要目的是简化attach/detach操作的步骤，使用nova和cinder在该过程中的耦合性更小。

在Ocata版本中，Nova沿用旧的attach/detach接口。在Pike版本中，nova将会在一些场景使用新接口（这些场景如：虚机热迁移，磁盘热迁移等）。

相关specs：http://specs.openstack.org/openstack/nova-specs/specs/pike/approved/cinder-new-attach-apis.html

# Ocata版本Nova使用的挂载旧接口

|  |  |  |  |
| --- | --- | --- | --- |
| ***接口名*** | ***功能*** | ***执行节点*** | ***备注*** |
| **os-reserve** | 修改卷状态为attaching | n-api |  |
| **os-initialize\_connection** | 获取backend的连接信息 | n-cpu |  |
| **os-attach** | 创建数据库attachment表记录 | n-cpu |  |

nova挂载磁盘时的流程如下：

1. **调用os-reserve修改卷状态；**
2. **调用os-initialize\_connection接口获取连接信息；**
3. **建立连接，挂载磁盘到VM；**
4. **调用os-attach添加一条数据库记录；**

# 新接口的逻辑

新接口和原来的接口区别在于：

1.新接口先建立数据库记录，然后Nova建立连接，原先是nova连接建立以后，nova通知Cinder添加记录，这样做的意图是防止一些场景实际建立了连接，但是没有attachment记录。

2.新接口使用更加简练，nova最少只用调用一次cinder接口就可以建立连接，而原来需要三次；

3.新的接口每个连接对应一个attachment信息，便于在流程发生错误时清理连接，控制回滚。

新接口定义在cinder/api/v3/attachments.py的AttachmentsController类中，包含以下方法：

|  |  |
| --- | --- |
| ***方法名*** | ***说明*** |
| attachment\_create | 修改卷状态为reserved，并且创建一条数据库记录。如果body中包含connector信息，会调用attachment\_update，否者直接返回。 |
| attachment\_update | 根据connector获取target的连接信息，并且填充数据中的attachment信息。 |
|  |  |

## attachment\_create方法

POST http://<ip>/volume/v3/<project\_id>/attachments

{

"attachment": {

"instance\_uuid": "<instance\_uuid>",

"volume\_uuid": "<volume\_id>"

"connector": {“attr1”:”val1”, “attr2”:”val2”, “attr3”:”val3”,……}

}

}

# 注意attachment方法要求发送ＲＥＳＴ　ＡＰＩ时，指定microversion值最小为3.27，因此在发送POST报文头，需要指定{"key":"OpenStack-API-Version","value":"volume 3.27"}。

调用create方法时connector是可选参数，如果body中没有指定该参数那么create接口只在volume\_attachment表中创建一条记录，并且将卷的状态设置为reserved，在该状态下的卷不能再调用create方法，也不允许通过旧的attachment API进行挂载。

attachment\_create接口可以一步完成以前挂载Volume的所有Cinder操作，也可以只创建一个虚拟连接（只有attachments记录，没有连接信息）。

## attachment\_update方法

PUT http://<ip>/volume/v3/<project\_id>/attachments/<attachment\_id>

{

"attachment": {

"connector": {

"initiator": "iqn.1993-08.org.debian: 01: cad181614cec",

"ip": "192.168.1.20",

"platform": "x86\_64",

"host": "tempest-1",

"os\_type": "linux2",

"multipath": false,

"mountpoint": "/dev/vdb",

"mode": "ro"

}

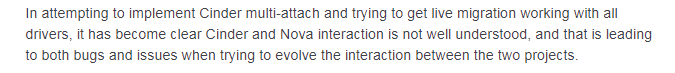
}

}

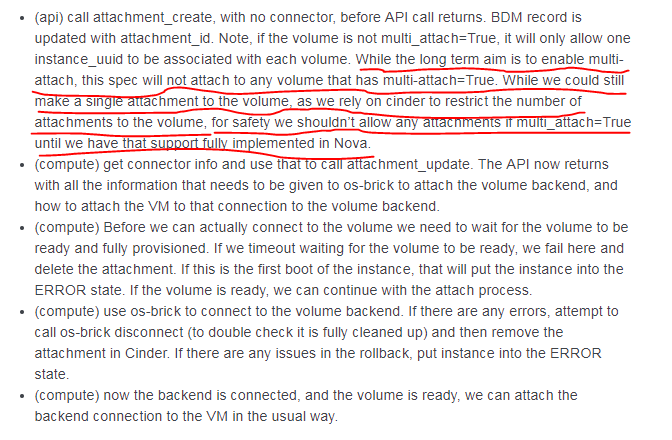
attachment\_update用来获取target的连接信息，并且将数据库记录填充完整。

# Nova计划的修改点

Nova引入新接口的目的主要有：1.尝试使用应用Cinder multi-attach特性；2.解决热迁移中的一些Bug；3.解决磁盘热迁移中的一些bug。



实际上，Nova的multi-attach功能并没有在Pike版本中实现，使用新接口的意图主要是为了后续实现该功能铺路。



Nova为了在Pike版本中禁用磁盘的multi-attach功能，修改了Cinder所有后端驱动代码，将存储池的multiattach特性全部写死成了False，通过常规手段已经无法创建具有multiattach能力的Volume，除非手动改DB（[相应的提交记录](https://github.com/openstack/cinder/commit/e499ab995de9c6dca353a115c6c68bd04594cdd9)）。

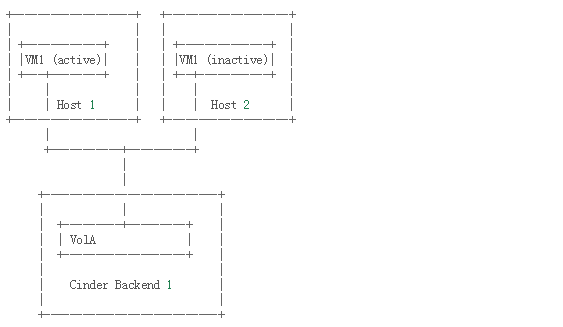
下面简述几个新接口的应用场景，这些场景都在spec中有提到，但是绝大部分还没有在Pike中有相应的代码修改。

## Live-migrate场景：

在Pike版本中的Live-migrate逻辑是这样的（假设从host1迁移到host2）：

1. 创建host2和backend的连接；
2. 执行VM从Host1到Host2的迁移；
3. 断开host1和backend的连接；

在热迁移场景中我们允许对Volume创建两个连接，即使这个卷不是multiattach的，但是必须注意，此时我们是在同一个VM和同一个Volume之间创建了两个连接（如下图）。



上面可以看出，VM迁移时在不同Host上有两个实例，一个时active一个为inactive，此时建立两个连接是安全的。上面的两个连接可以表示为：

attachment 1: VolA, VM1, Host 1

attachment 2: VolA, VM1, Host 2

当Live-migrate失败时有以下两种情况：

1. 删除attachment1失败（相当于最后一步发生失败），此时保留两个连接，虚机状态、迁移状态为ERROR。specs中提及保留attachment的初衷是提醒用户，需要手工介入去修复该状态。
2. 如果VM从host1移动到host2的过程发生失败（相当于步骤2失败），此时VM始终任然在host1上运行，此时回滚时只要删除attachment2，虚机状态依然正常，可以继续工作，但是迁移状态是ERROR。

原来的方案由于DB中只会有一条attachment记录，但是实际在nova的热迁过程中会出现两个连接，一旦迁移出现问题就会造成数据的attachment信息无法回滚，同时BlockDeviceMapping表的信息也会出现回滚异常。

## 磁盘热迁移场景

volume进行热迁移时，需要Nova参与拷贝数据。假设volume从backend1迁移到backend2。Cinder会现在backend2上创建一个new\_vol，之后通知nova将old\_vol上的数据拷贝到new\_vol中完成迁移。

在Pike版本中，上述过程存在以下两个attachment，对应数据库中两条记录：

attachment1 ；VM -🡪 old\_vol

attachment2 : VM -🡪 new\_vol

nova接到Cinder的磁盘迁移通知时，进行以下操作：

1. 和new\_vol建立连接；
2. libvirt进行数据拷贝；
3. 拷贝完成删除attachment1；

在ocata版本中上述过程也会建立attachment2连接，但是区别是attachment2不会存在对应的数据库记录。