Nova image create流程

本文主要讨论nova/virt/libvirt/driver.py:\_create\_image的相关流程，只讨论file磁盘，不包括EBS盘（block设备）。

1. **Resize过程的镜像拷贝优化**
2. 优化之前

首先通过libvirt的XMLDesc()方法拿到虚拟机的配置文件，然后从配置文件中读取所有file类型磁盘的信息（路径，driver，qcow2的backing file）；然后如果是不同host之间resize，则qemu-img convert合并base和子镜像为qcow2（无backing file），之后通过rsync ssh方式拷贝合并后的镜像到新的host对于instance目录下，然后在if not os.path.exists(self.path) or not os.path.exists(base):则创建镜像，resize过程这个self.path是已经拷贝过来的，所以不需要创建镜像，也就是什么都不做。

1. 优化之后（仅优化了resize过程，创建过程与优化之前相同）

拷贝镜像是用rsync的daemon push模式，并且不合并base和子镜像，只拷贝子镜像部分，然后在目标host上检查base是否存在，不存在则下载，扩容，最后qemu-img rebase把子镜像rebase到新的base上；目前第二块盘（disk.local）以及swap盘（disk.swap）是不拷贝的，因为如果拷贝过去的仅仅是子镜像，会导致base找不到，为disk.local、disk.swap准备base镜像这部分代码没有实现，所以在拷贝子镜像过程中忽略了disk.local，disk.swap目前没有配置，所以代码里面没有忽略，如果开启了swap的配置，则resize过程会出现问题（base找不到导致虚拟机无法启动）。

优化后的镜像生成流程：

# nova/virt/libvirt/driver.py:LibvirtDriver.\_create\_image()

**if** snapshot\_optimization \

**and** **not** self**.**\_volume\_in\_mapping**(**self**.**default\_root\_device**,**

block\_device\_info**):**

self**.**\_create\_snapshot\_image**(**context**,** instance**,**

disk\_images**[**'image\_id'**],**

basepath**(**'disk'**),** size**)**

# nova/virt/libvirt/driver.py:LibvirtDriver

# use optimized snapshot image

# 优化后的resize过程镜像生成流程

**def** \_create\_snapshot\_image**(**self**,** context**,** instance**,** image\_id**,**

target**,** size**):**

# NOTE(hzwangpan): for resize operation, the 'disk' is copied from

# source node before \_create\_image(), so if we fetch the 'disk' here,

# it will cover the 'disk' copied from source

# 只有当'disk'不存在的时候才下载'disk'，M3从快照恢复流程优化的遗留代码

# M3的快照只有COW部分也即'disk'，所以创建虚拟机的时候要先下载'disk'，

# 然后根据其backing file的名称从glance下载它的base，这里就是下载'disk'

# 的流程，因为社区原有的下载镜像代码会转换镜像格式，而我们不需要转换，

# 所以这里新加了一个fetch\_orig\_image()方法。

# resize的时候instance目录下'disk'是存在的，已经从源端拷贝过来了。

**if** **not** os**.**path**.**exists**(**target**):**

libvirt\_utils**.**fetch\_orig\_image**(**context**=**context**,** target**=**target**,**

image\_id**=**image\_id**,**

user\_id**=**instance**[**"user\_id"**],**

project\_id**=**instance**[**"project\_id"**])**

**if** **not** os**.**path**.**exists**(**target**):**

LOG**.**error**(**\_**(**"fetch image failed, image id: %s"**),** image\_id**,**

instance**=**instance**,** context**=**context**)**

**raise** exception**.**CouldNotFetchImage**(**image\_id**)**

# 查询'disk'的backing file信息，也即查找其base

backing\_file **=** libvirt\_utils**.**get\_disk\_backing\_file**(**target**)**

**if** **not** backing\_file**:**

LOG**.**error**(**\_**(**"get backing file of image %s failed"**),** image\_id**,**

instance**=**instance**,** context**=**context**)**

**raise** exception**.**ImageUnacceptable**(**image\_id**=**image\_id**,**

reason**=**\_**(**"%s doesn't has backing file"**)** **%** target**)**

virtual\_size **=** libvirt\_utils**.**get\_disk\_size**(**target**)**

size **=** max**(**size**,** virtual\_size**)**

# get base image by backing file

# 根据backing file名称下载base镜像

# 如果没有M3那种不完整的快照存在，则从backing file名称下载base镜像

# 的流程可以简化为根据image id下载镜像，因为每一个虚拟机都是从一个

# 完整的镜像/快照创建的，所以resize的时候根据虚拟机的image id下载到

# 的镜像就是'disk'的base。

base\_dir **=** os**.**path**.**join**(**FLAGS**.**instances\_path**,** '\_base'**)**

**if** **not** os**.**path**.**exists**(**base\_dir**):**

utils**.**ensure\_tree**(**base\_dir**)**

old\_backing\_file **=** os**.**path**.**join**(**base\_dir**,** backing\_file**)**

old\_size **=** 0

**if** "\_" **in** os**.**path**.**basename**(**old\_backing\_file**):**

base\_img **=** old\_backing\_file**.**rsplit**(**"\_"**,** 1**)[**0**]**

old\_size **=** int**(**old\_backing\_file**.**rsplit**(**"\_"**,** 1**)[**1**])** **\*** \

**(**1024L **\*** 1024L **\*** 1024L**)**

**else:**

base\_img **=** old\_backing\_file

# 先检查不带大小信息的base是否存在，如果存在就不需要从glance下载了

# 如果不存在，则需要从glance下载base

**if** **not** os**.**path**.**exists**(**base\_img**):**

self**.**\_get\_base\_image\_by\_backing\_file**(**context**,** instance**,**

image\_id**,** base\_img**)**

lock\_path **=** os**.**path**.**join**(**FLAGS**.**instances\_path**,** 'locks'**)**

@utils.synchronized**(**base\_img**,** external**=True,** lock\_path**=**lock\_path**)**

**def** copy\_and\_extend**(**base\_img**,** target\_img**,** size**):**

**if** **not** os**.**path**.**exists**(**target\_img**):**

libvirt\_utils**.**copy\_image**(**base\_img**,** target\_img**)**

disk**.**extend**(**target\_img**,** size**)**

# NOTE(wangpan): qemu-img rebase 'Safe mode' need the old backing file,

# refer to qemu-img manual for more details.

# 从没有大小信息的base拷贝扩容出'disk'的老的backing file，因为qemu-img

# rebase默认是采用“安全模式”的，这种模式需要COW部分的新老backing file

# 都存在才能正常执行。

**if** old\_size**:**

copy\_and\_extend**(**base\_img**,** old\_backing\_file**,** old\_size**)**

# 从没有大小信息的base拷贝扩容出'disk'的新的backing file，也即resize之后的大小

new\_backing\_file **=** base\_img

**if** size**:**

size\_gb **=** size **/** **(**1024 **\*** 1024 **\*** 1024**)**

new\_backing\_file **+=** "\_%d" **%** size\_gb

copy\_and\_extend**(**base\_img**,** new\_backing\_file**,** size**)**

# when old\_backing\_file != new\_backing\_file, rebase is needed

# 如果新老backing file不一样，则需要对'disk'进行rebase操作

**if** old\_backing\_file **!=** new\_backing\_file**:**

libvirt\_utils**.**rebase\_cow\_image**(**new\_backing\_file**,** target**)**

**def** \_get\_base\_image\_by\_backing\_file**(**self**,** context**,** instance**,**

image\_id**,** backing\_file**):**

base\_image\_id\_sha1 **=** os**.**path**.**basename**(**backing\_file**)**

LOG**.**debug**(**\_**(**"image id sha1 of backing file %(backing\_file)s "

"is: %(base\_image\_id\_sha1)s"**)** **%** locals**(),**

instance**=**instance**,** context**=**context**)**

**(**image\_service**,** image\_id**)** **=** glance**.**get\_remote\_image\_service**(**

context**,** image\_id**)**

# 根据base名称，从glance查询镜像/快照信息

image\_info **=** image\_service**.**get\_image\_properties**(**context**,**

"image\_id\_sha1"**,**

base\_image\_id\_sha1**)**

**if** **not** image\_info**:**

LOG**.**error**(**\_**(**"can't find base image by base\_image\_id\_sha1 "

" %(base\_image\_id\_sha1)s, snapshot image\_id: %(image\_id)s"**)** **%**

locals**(),** instance**=**instance**,** context**=**context**)**

**raise** exception**.**ImageNotFound**(**image\_id**=**base\_image\_id\_sha1**)**

base\_image\_id **=** str**(**image\_info**[**0**].**get**(**"image\_id"**))**

lock\_path **=** os**.**path**.**join**(**FLAGS**.**instances\_path**,** 'locks'**)**

# 下载找到的镜像/快照

@utils.synchronized**(**base\_image\_id\_sha1**,**

external**=True,** lock\_path**=**lock\_path**)**

**def** fetch\_base\_image**(**context**,** target**,** image\_id**,** user\_id**,** project\_id**):**

**if** **not** os**.**path**.**exists**(**target**):**

# 使用原有的下载镜像的方法，会转换镜像格式

libvirt\_utils**.**fetch\_image**(**context**=**context**,**

target**=**target**,**

image\_id**=**image\_id**,**

user\_id**=**user\_id**,**

project\_id**=**project\_id**)**

fetch\_base\_image**(**context**,** backing\_file**,** base\_image\_id**,**

instance**[**"user\_id"**],** instance**[**"project\_id"**])**

**if** **not** os**.**path**.**exists**(**backing\_file**):**

LOG**.**error**(**\_**(**"fetch base image failed, image id: %s"**),**

base\_image\_id**,** instance**=**instance**,** context**=**context**)**

**raise** exception**.**CouldNotFetchImage**(**base\_image\_id**)**

1. **公共流程**

创建和resize的公共流程：

# nova/virt/libvirt/driver.py:LibvirtDriver.\_create\_image()

# syntactic nicety(为了语法好看，定义了三个内部方法)

**def** basepath**(**fname**=**''**,** suffix**=**suffix**):**

**return** os**.**path**.**join**(**FLAGS**.**instances\_path**,**

instance**[**'name'**],**

fname **+** suffix**)**

**def** image**(**fname**,** image\_type**=**FLAGS**.**libvirt\_images\_type**):**

**return** self**.**image\_backend**.**image**(**instance**[**'name'**],**

fname **+** suffix**,** image\_type**)**

**def** raw**(**fname**):**

**return** image**(**fname**,** image\_type**=**'raw'**)**

# ensure directories exist and are writable

# 创建instance目录，用来存放镜像和libvirt.xml配置文件

utils**.**ensure\_tree**(**basepath**(**suffix**=**''**))**

# 写入libvirt.xml配置文件

libvirt\_utils**.**write\_to\_file**(**basepath**(**'libvirt.xml'**),** libvirt\_xml**)**

# 写入console.log控制台输出文件

libvirt\_utils**.**write\_to\_file**(**basepath**(**'console.log'**,** ''**),** ''**,** 007**)**

# get image type（为了优化镜像流程而新增的代码）

image\_type **=** **None**

has\_base\_id\_sha1 **=** **False**

**(**image\_service**,** image\_id**)** **=** glance**.**get\_remote\_image\_service**(**

context**,** disk\_images**[**'image\_id'**])**

**try:**

image\_info **=** image\_service**.**show**(**context**,** image\_id**)**

**if** image\_info **and** 'properties' **in** image\_info**:**

**if** image\_info**[**'properties'**].**get**(**'image\_type'**)** **==** "snapshot"**:**

image\_type **=** "snapshot"

**else:** # 如果不是快照，则认为是普通镜像

image\_type **=** "image"

# base\_image\_id\_sha1是为了兼容M3时的快照（只上传COW部分）

**if** image\_info**[**'properties'**].**get**(**'base\_image\_id\_sha1'**):**

has\_base\_id\_sha1 **=** **True**

**except** Exception**:**

image\_type **=** **None**

has\_base\_id\_sha1 **=** **None**

LOG**.**warn**(**\_**(**"get image type of %s faild"**)** **%** image\_id**,**

context**=**context**,** instance**=**instance**)**

**pass**

# 检查镜像是否有backing file，也即是否只是COW部分

backing\_file **=** **None**

**if** os**.**path**.**exists**(**basepath**(**'disk'**)):**

backing\_file **=** libvirt\_utils**.**get\_disk\_backing\_file**(**

basepath**(**'disk'**))**

# 下面的这些判断都是为了检查是否需要走我们自己修改的镜像流程

# snapshot\_optimization为True则需要走修改后的流程

snapshot\_optimization **=** **False**

# check use image snapshot optimization or not

use\_qcow2 **=** **((**FLAGS**.**libvirt\_images\_type **==** 'default' **and**

FLAGS**.**use\_cow\_images**)** **or**

FLAGS**.**libvirt\_images\_type **==** 'qcow2'**)**

# only qcow2 image may be need to optimize, and images with

# 'kernel\_id' or 'ramdisk\_id' shouldn't be optimized

**if** FLAGS**.**allow\_image\_snapshot\_optimization **and** use\_qcow2 **and** \

**not** disk\_images**[**'kernel\_id'**]** **and** **not** disk\_images**[**'ramdisk\_id'**]:**

# 下面的这些if语句是为了判断当前属于哪种镜像的哪个操作

# 然后就可以判断是否需要走修改后的流程，这种判断方式比较人肉，

# 以后改起来也比较麻烦，但目前没有更好的办法了。

# normal image, when create instance（普通镜像的创建虚拟机过程）

**if** image\_type **==** "image" **and** backing\_file **is** **None** **and** \

**not** has\_base\_id\_sha1**:**

snapshot\_optimization **=** **False**

# normal image, when resize（普通镜像的resize过程）

**if** image\_type **==** "image" **and** backing\_file **is** **not** **None** **and** \

**not** has\_base\_id\_sha1**:**

snapshot\_optimization **=** **True**

# unbroken snapshot, when create instance（完整快照的创建虚拟机过程）

**if** image\_type **==** "snapshot" **and** backing\_file **is** **None** **and** \

**not** has\_base\_id\_sha1**:**

snapshot\_optimization **=** **False**

# unbroken snapshot, when resize（完整快照的resize过程）

**if** image\_type **==** "snapshot" **and** backing\_file **is** **not** **None** **and** \

**not** has\_base\_id\_sha1**:**

snapshot\_optimization **=** **True**

# only cow part snapshot, when create instance

# （只有COW部分的快照（M3修改）的创建过程）

**if** image\_type **==** "snapshot" **and** backing\_file **is** **None** **and** \

has\_base\_id\_sha1**:**

snapshot\_optimization **=** **True**

# only cow part snapshot, when resize

# （只有COW部分的快照（M3修改）的resize过程）

**if** image\_type **==** "snapshot" **and** backing\_file **is** **not** **None** **and** \

has\_base\_id\_sha1**:**

snapshot\_optimization **=** **True**

# 生成base的文件名

root\_fname **=** hashlib**.**sha1**(**str**(**disk\_images**[**'image\_id'**])).**hexdigest**()**

1. **创建过程**
2. 概述

对于qcow2格式镜像root盘，原有流程是先下载镜像（或者说先创建base），然后qemu-img create生成子镜像（disk），对于qcow2格式的第二块临时盘和第三块swap盘，首先是通过mkfs/mkswap创建base，之后qemu-img create生成子镜像（disk.local/disk.swap）。

传入参数：

# nova/virt/libvirt/driver.py:LibvirtDriver.spawn()

self**.**\_create\_image**(**context**,** instance**,** xml**,** network\_info**=**network\_info**,**

block\_device\_info**=**block\_device\_info**,**

files**=**injected\_files**,**

admin\_pass**=**admin\_password**)**

1. Root盘

目前如果不是M3版本的快照文件，完整快照或者镜像的创建过程与社区F版本流程一致。首先根据image id下载镜像，之后转换、copy、扩容后生成并Cache base镜像，最后qemu-img create创建COW部分的disk。

# nova/virt/libvirt/driver.py:LibvirtDriver.\_create\_image()

**elif** **not** self**.**\_volume\_in\_mapping**(**self**.**default\_root\_device**,**

block\_device\_info**):**

# image是上面说的三个内部方法之一，初始化为一个对象，具体的对象是

# 根据镜像的格式来确定的，FLAGS.libvirt\_images\_type默认是default，

# 然后会再判断FLAGS.use\_cow\_images是否为True，默认值为True

# 如果是True则image是Qcow2类的对象，目前这两个值都是保持默认。

# 否则则是Raw，LVM则需要配置libvirt\_images\_type='lvm'。

# 'disk'参数是root盘的文件名，也就是加上instance目录后的image.path

# cache就是image类里的一个方法，用来缓存base

# fetch\_func就是如果base不存在，用来从glance下载镜像的方法

# filename是base文件的名称

image**(**'disk'**).**cache**(**fetch\_func**=**libvirt\_utils**.**fetch\_image**,**

context**=**context**,**

filename**=**root\_fname**,**

size**=**size**,**

image\_id**=**disk\_images**[**'image\_id'**],**

user\_id**=**instance**[**'user\_id'**],**

project\_id**=**instance**[**'project\_id'**])**

# nova/virt/libvirt/imagebackend.py:Image.cache()

**def** cache**(**self**,** fetch\_func**,** filename**,** size**=None,** **\***args**,** **\*\***kwargs**):**

"""Creates image from template.

Ensures that template and image not already exists.

Ensures that base directory exists.

Synchronizes on template fetching.

:fetch\_func: Function that creates the base image

Should accept `target` argument.

:filename: Name of the file in the image directory

:size: Size of created image in bytes (optional)

"""

# 根据base的文件名加锁，防止两个创建过程同时下载导致的镜像损坏

@utils.synchronized**(**filename**,** external**=True,** lock\_path**=**self**.**lock\_path**)**

**def** call\_if\_not\_exists**(**target**,** **\***args**,** **\*\***kwargs**):**

# 这里的判断必不可少，因为可能拿到锁的时候另外一个创建流程已经下载过了这个镜像

**if** **not** os**.**path**.**exists**(**target**):**

fetch\_func**(**target**=**target**,** **\***args**,** **\*\***kwargs**)**

# 如果instance目录下'disk'文件已经存在，则什么都不做，否则生成'disk'

**if** **not** os**.**path**.**exists**(**self**.**path**):** # self.path的初始化见下面的代码

base\_dir **=** os**.**path**.**join**(**FLAGS**.**instances\_path**,** '\_base'**)**

**if** **not** os**.**path**.**exists**(**base\_dir**):**

utils**.**ensure\_tree**(**base\_dir**)**

base **=** os**.**path**.**join**(**base\_dir**,** filename**)**

# 把下载镜像的方法作为参数传给创建disk的方法

self**.**create\_image**(**call\_if\_not\_exists**,** base**,** size**,**

**\***args**,** **\*\***kwargs**)**

# nova/virt/libvirt/imagebackend.py:

**class** **Qcow2(**Image**):**

**def** \_\_init\_\_**(**self**,** instance**,** name**):**

super**(**Qcow2**,** self**).**\_\_init\_\_**(**"file"**,** "qcow2"**,** is\_block\_dev**=False)**

# instance=instance['name']，name='disk'

# self.path就是instance目录下的disk文件

self**.**path **=** os**.**path**.**join**(**FLAGS**.**instances\_path**,**

instance**,** name**)**

**def** create\_image**(**self**,** prepare\_template**,** base**,** size**,** **\***args**,** **\*\***kwargs**):**

# 加锁，防止镜像在使用过程中中被删除或修改

@utils.synchronized**(**base**,** external**=True,** lock\_path**=**self**.**lock\_path**)**

**def** copy\_qcow2\_image**(**base**,** target**,** size**):**

qcow2\_base **=** base

**if** size**:**

size\_gb **=** size **/** **(**1024 **\*** 1024 **\*** 1024**)**

qcow2\_base **+=** '\_%d' **%** size\_gb

**if** **not** os**.**path**.**exists**(**qcow2\_base**):**

**with** utils**.**remove\_path\_on\_error**(**qcow2\_base**):**

# 根据flavor拷贝后扩容base

libvirt\_utils**.**copy\_image**(**base**,** qcow2\_base**)**

disk**.**extend**(**qcow2\_base**,** size**)**

# 使用qemu-img create命令行创建COW部分也即disk文件

libvirt\_utils**.**create\_cow\_image**(**qcow2\_base**,** target**)**

# 使用传入的下载镜像的方法下载镜像，也即准备base

prepare\_template**(**target**=**base**,** **\***args**,** **\*\***kwargs**)**

**with** utils**.**remove\_path\_on\_error**(**self**.**path**):**

copy\_qcow2\_image**(**base**,** self**.**path**,** size**)**

下载时是先把镜像保存在\_base目录下，命名为root\_fname.part，然后转换为raw格式，转换过程中的目标文件命名为root\_fname.converted，转换完成后删除root\_fname.part，并把root\_fname.converted改为root\_fname，扩容后的后面加上size信息例如root\_fname\_10。

生成的libvirt.xml配置文件中root盘的配置为：

<disk type=**'file'** device=**'disk'**>

<driver name=**'qemu'** type=**'qcow2'** cache=**'none'**/>

<source file=**'/home/openstack/nova/instances/instance-000005ac/disk'**/>

<target dev=**'vda'** bus=**'virtio'**/>

</disk>

1. Ephemeral盘

首先qemu-img create创建base（mkfs.ext3格式化），之后qemu-img create创建COW部分的disk.local，配置文件与root盘相同，只是file文件的名称（disk改为disk.local）、target dev（vda改为vdb）不同而已。

# nova/virt/libvirt/driver.py:LibvirtDriver.\_create\_image()

ephemeral\_gb **=** instance**[**'ephemeral\_gb'**]**

**if** ephemeral\_gb **and** **not** self**.**\_volume\_in\_mapping**(**

self**.**default\_second\_device**,** block\_device\_info**):**

# 如果有第二块盘'disk.local'，则swap盘作为第三块盘vdc

swap\_device **=** self**.**default\_third\_device

# 封装创建第二块盘的方法\_create\_ephemeral

fn **=** functools**.**partial**(**self**.**\_create\_ephemeral**,**

fs\_label**=**'ephemeral0'**,**

os\_type**=**instance**[**"os\_type"**])**

fname **=** "ephemeral\_%s\_%s\_%s" **%** **(**"0"**,**

ephemeral\_gb**,**

instance**[**"os\_type"**])**

size **=** ephemeral\_gb **\*** 1024 **\*** 1024 **\*** 1024

# 与root盘的创建流程类似，差别只是将从glance下载镜像改为qemu-img创建base

image**(**'disk.local'**).**cache**(**fetch\_func**=**fn**,**

filename**=**fname**,**

size**=**size**,**

ephemeral\_size**=**ephemeral\_gb**)**

**else:**

swap\_device **=** self**.**default\_second\_device

# nova/virt/libvirt/driver.py:LibvirtDriver

**def** \_create\_ephemeral**(**self**,** target**,** ephemeral\_size**,** fs\_label**,** os\_type**):**

# 创建未格式化的空磁盘文件

self**.**\_create\_local**(**target**,** ephemeral\_size**)**

# 格式化为ext3格式

disk**.**mkfs**(**os\_type**,** fs\_label**,** target**)**

@staticmethod

**def** \_create\_local**(**target**,** local\_size**,** unit**=**'G'**,**

fs\_format**=None,** label**=None):**

"""Create a blank image of specified size"""

**if** **not** fs\_format**:**

fs\_format **=** FLAGS**.**default\_ephemeral\_format # 默认为None

# qemu-img create命令创建raw格式的base

libvirt\_utils**.**create\_image**(**'raw'**,** target**,**

'%d%c' **%** **(**local\_size**,** unit**))**

**if** fs\_format**:** # =None，这里不执行

libvirt\_utils**.**mkfs**(**fs\_format**,** target**,** label**)**

1. Swap盘

流程与ephemeral盘相同，只是base格式不同，首先qemu-img创建base，并用mkswap格式化，之后qemu-img create创建COW部分的disk.local。配置文件与root盘相同，只是file文件的名称（disk改为disk. swap）、target dev（vda改为vdb/vdc，根据有无Ephemeral盘而定）不同而已。

1. **Resize/冷迁移过程**
2. Root盘

resize源端：

# nova/virt/libvirt/driver.py:LibvirtDriver

@exception.wrap\_exception**()**

**def** migrate\_disk\_and\_power\_off**(**self**,** context**,** instance**,** dest**,**

instance\_type**,** network\_info**,**

block\_device\_info**=None):**

LOG**.**debug**(**\_**(**"Starting migrate\_disk\_and\_power\_off"**),**

instance**=**instance**)**

# 获取虚拟机上所有的type='file'类型的disk的信息

disk\_info\_text **=** self**.**get\_instance\_disk\_info**(**instance**[**'name'**])**

disk\_info **=** jsonutils**.**loads**(**disk\_info\_text**)**

# 关机

self**.**power\_off**(**instance**)**

# 块设备处理，我们目前没有使用cinder，所以这里不处理

block\_device\_mapping **=** driver**.**block\_device\_info\_get\_mapping**(**

block\_device\_info**)**

**for** vol **in** block\_device\_mapping**:**

connection\_info **=** vol**[**'connection\_info'**]**

mount\_device **=** vol**[**'mount\_device'**].**rpartition**(**"/"**)[**2**]**

self**.**volume\_driver\_method**(**'disconnect\_volume'**,**

connection\_info**,**

mount\_device**)**

# copy disks to destination

# rename instance dir to +\_resize at first for using

# shared storage for instance dir (eg. NFS).

# 拷贝disk到目标host

same\_host **=** **(**dest **==** self**.**get\_host\_ip\_addr**())**

inst\_base **=** "%s/%s" **%** **(**FLAGS**.**instances\_path**,** instance**[**'name'**])**

inst\_base\_resize **=** inst\_base **+** "\_resize"

clean\_remote\_dir **=** **False**

**try:**

# 先把instance目录改为instance-xxxxx\_resize，备份过程

utils**.**execute**(**'mv'**,** inst\_base**,** inst\_base\_resize**)**

**if** same\_host**:**

dest **=** **None**

utils**.**execute**(**'mkdir'**,** '-p'**,** inst\_base**)**

**else:**

# 不同宿主机之间的resize

**if** **not** FLAGS**.**use\_rsync**:**

# 优化前的流程是用ssh创建目标端的instance目录

utils**.**execute**(**'ssh'**,** dest**,** 'mkdir'**,** '-p'**,** inst\_base**)**

**else:**

# 新流程是用rsync创建目录

libvirt\_utils**.**make\_remote\_instance\_dir**(**inst\_base\_resize**,**

dest**,** instance**[**'name'**])**

clean\_remote\_dir **=** **True**

# 遍历所有disk

**for** info **in** disk\_info**:**

# assume inst\_base == dirname(info['path'])

img\_path **=** info**[**'path'**]**

fname **=** os**.**path**.**basename**(**img\_path**)**

# FIXME(wangpan): when resize, we ignore the ephemeral disk

# 我们在这里忽略了第二块盘'disk.local'，不拷贝到目标端

# 这里我们还应该忽略第三块盘'disk.swap'，不过暂时没用到

**if** fname **==** "disk.local"**:**

LOG**.**debug**(**\_**(**"ignore disk.local when resize"**),**

instance**=**instance**)**

**continue**

from\_path **=** os**.**path**.**join**(**inst\_base\_resize**,** fname**)**

remote\_path **=** "%s/%s" **%** **(**instance**[**'name'**],** fname**)**

**if** info**[**'type'**]** **==** 'qcow2' **and** info**[**'backing\_file'**]:**

tmp\_path **=** from\_path **+** "\_rbase"

# Note(hzzhoushaoyu): if allow optimization, just copy

# qcow2 to destination without merge.

# 优化后的流程是只拷贝COW部分，不合并COW和base

**if** FLAGS**.**allow\_image\_snapshot\_optimization**:**

tmp\_path **=** from\_path

**else:**

# merge backing file

# 老的流程是先合并COW和base之后再拷贝

utils**.**execute**(**'qemu-img'**,** 'convert'**,** '-f'**,** 'qcow2'**,**

'-O'**,** 'qcow2'**,** from\_path**,** tmp\_path**)**

**if** same\_host **and** \

**not** FLAGS**.**allow\_image\_snapshot\_optimization**:**

utils**.**execute**(**'mv'**,** tmp\_path**,** img\_path**)**

**elif** same\_host **and** FLAGS**.**allow\_image\_snapshot\_optimization**:**

utils**.**execute**(**'cp'**,** tmp\_path**,** img\_path**)**

**else:**

**if** **not** FLAGS**.**use\_rsync**:**

# 老的流程使用rsync的ssh模式拷贝磁盘文件

libvirt\_utils**.**copy\_image**(**tmp\_path**,** img\_path**,**

host**=**dest**)**

**else:**

# 优化后使用rsync的daemon push模式拷贝

libvirt\_utils**.**copy\_image\_to\_remote**(**tmp\_path**,**

remote\_path**,** dest**)**

**if** **not** FLAGS**.**allow\_image\_snapshot\_optimization**:**

utils**.**execute**(**'rm'**,** '-f'**,** tmp\_path**)**

**else:** # raw or qcow2 with no backing file

**if** **not** FLAGS**.**use\_rsync **or** same\_host**:**

libvirt\_utils**.**copy\_image**(**from\_path**,** img\_path**,**

host**=**dest**)**

**else:**

libvirt\_utils**.**copy\_image\_to\_remote**(**tmp\_path**,**

remote\_path**,** dest**)**

**except** Exception**,** e**:**

**try:**

# 异常处理，清理残留文件

**if** os**.**path**.**exists**(**inst\_base\_resize**):**

utils**.**execute**(**'rm'**,** '-rf'**,** inst\_base**)**

**if** clean\_remote\_dir **and** FLAGS**.**use\_rsync**:**

libvirt\_utils**.**clean\_remote\_dir**(**instance**[**'name'**],** dest**)**

utils**.**execute**(**'mv'**,** inst\_base\_resize**,** inst\_base**)**

**if** **not** FLAGS**.**use\_rsync**:**

utils**.**execute**(**'ssh'**,** dest**,** 'rm'**,** '-rf'**,** inst\_base**)**

**except** Exception**:**

**pass**

**raise** e

# 返回磁盘信息共目的端使用

**return** disk\_info\_text

resize目的端：

# nova/virt/libvirt/driver.py:LibvirtDriver

@exception.wrap\_exception**()**

**def** finish\_migration**(**self**,** context**,** migration**,** instance**,** disk\_info**,**

network\_info**,** image\_meta**,** resize\_instance**,**

block\_device\_info**=None):**

LOG**.**debug**(**\_**(**"Starting finish\_migration"**),** instance**=**instance**)**

# 生成libvirt.xml文件

xml **=** self**.**to\_xml**(**instance**,** network\_info**,**

block\_device\_info**=**block\_device\_info**)**

# assume \_create\_image do nothing if a target file exists.

# TODO(oda): injecting files is not necessary

# 这里生成镜像，但是实际上社区原有流程不会生成镜像，因为'disk'已经拷贝过来了

# 所以imagebackend.py里面的cache方法什么事情都不做

# 这里主要是创建instance目录，写入libvirt.xml和console.log文件

# 但是我们修改后的流程，会根据'disk'的backing file下载它的base，

# 还会在这里重新生成第二块盘的base和'disk.local'，

# 第三块盘'disk.swap'因为拷贝的时候没有忽略，所以这里不会重新生成，

# 所以这里可能会导致disk.swap找不到base，虚拟机启动失败。

self**.**\_create\_image**(**context**,** instance**,** xml**,**

network\_info**=**network\_info**,**

block\_device\_info**=None)**

# resize disks. only "disk" and "disk.local" are necessary.

# resize磁盘，忽略了第三块盘

disk\_info **=** jsonutils**.**loads**(**disk\_info**)**

**for** info **in** disk\_info**:**

fname **=** os**.**path**.**basename**(**info**[**'path'**])**

**if** fname **==** 'disk'**:**

size **=** instance**[**'root\_gb'**]**

**elif** fname **==** 'disk.local'**:**

size **=** instance**[**'ephemeral\_gb'**]**

**else:**

size **=** 0

size **\*=** 1024 **\*** 1024 **\*** 1024

# If we have a non partitioned image that we can extend

# then ensure we're in 'raw' format so we can extend file system.

fmt **=** info**[**'type'**]**

# 如果是qcow2格式的镜像，并且可以resize，则先把它转换为raw格式

**if** **(**size **and** fmt **==** 'qcow2' **and**

disk**.**can\_resize\_fs**(**info**[**'path'**],** size**,** use\_cow**=True)):**

path\_raw **=** info**[**'path'**]** **+** '\_raw'

utils**.**execute**(**'qemu-img'**,** 'convert'**,** '-f'**,** 'qcow2'**,**

'-O'**,** 'raw'**,** info**[**'path'**],** path\_raw**)**

utils**.**execute**(**'mv'**,** path\_raw**,** info**[**'path'**])**

fmt **=** 'raw'

# resize磁盘

**if** size**:**

disk**.**extend**(**info**[**'path'**],** size**)**

**if** fmt **==** 'raw' **and** FLAGS**.**use\_cow\_images**:**

# back to qcow2 (no backing\_file though) so that snapshot

# will be available

# 如果是raw格式或者刚刚被转换成raw格式，则再次转换成qcow2

path\_qcow **=** info**[**'path'**]** **+** '\_qcow'

utils**.**execute**(**'qemu-img'**,** 'convert'**,** '-f'**,** 'raw'**,**

'-O'**,** 'qcow2'**,** info**[**'path'**],** path\_qcow**)**

utils**.**execute**(**'mv'**,** path\_qcow**,** info**[**'path'**])**

### 上面的两次转换过程是很耗时的，所以不建议这么做

### 还好我们目前的root\_gb大小都是一样的，不会做resize动作

# 创建虚拟机

self**.**\_create\_domain\_and\_network**(**xml**,** instance**,** network\_info**,**

block\_device\_info**)**

# 等待虚拟机启动

timer **=** utils**.**LoopingCall**(**self**.**\_wait\_for\_running**,** instance**)**

timer**.**start**(**interval**=**0.5**).**wait**()**

1. Ephemeral盘

与root盘相同

1. Swap盘

与root盘相同

1. **热迁移过程（带block migration情况）**
2. Root盘

热迁移目的端：

# nova/virt/libvirt/driver.py:LibvirtDriver

**def** pre\_block\_migration**(**self**,** ctxt**,** instance**,** disk\_info\_json**):**

"""Preparation block migration.

:params ctxt: security context

:params instance:

nova.db.sqlalchemy.models.Instance object

instance object that is migrated.

:params disk\_info\_json:

json strings specified in get\_instance\_disk\_info

"""

# 与resize相同，disk\_info\_json也是找到的所有type='file'的disk

disk\_info **=** jsonutils**.**loads**(**disk\_info\_json**)**

# make instance directory

instance\_dir **=** os**.**path**.**join**(**FLAGS**.**instances\_path**,** instance**[**'name'**])**

**if** os**.**path**.**exists**(**instance\_dir**):**

**raise** exception**.**DestinationDiskExists**(**path**=**instance\_dir**)**

os**.**mkdir**(**instance\_dir**)**

# 遍历所有file disk

**for** info **in** disk\_info**:**

base **=** os**.**path**.**basename**(**info**[**'path'**])**

# Get image type and create empty disk image, and

# create backing file in case of qcow2.

instance\_disk **=** os**.**path**.**join**(**instance\_dir**,** base**)**

# 如果disk没有backing file（raw格式、或者不带backing file的qcow2）

# 则直接用'qemu-img create'创建空盘，磁盘内容会随着热迁移流程拷贝过来

# 这就是block migration的意义。

**if** **not** info**[**'backing\_file'**]:**

libvirt\_utils**.**create\_image**(**info**[**'type'**],** instance\_disk**,**

info**[**'disk\_size'**])**

**else:**

# 有backing file的disk

# 与创建虚拟机相同的镜像生成流程，也就是先准备base，

# 再qemu-img create 'disk'，这里生成的'disk'也是类似空盘

# 需要block migration拷贝过来

# 需要注意的是如果是M3的不完整快照，这里的流程会出错，

# 因为这里是根据image id下载base的，而不完整的快照的id就是它本身

# 我们需要的是根据快照的id找到并下载它的base

# M4的完整快照应该与普通镜像相同，所以没有这个问题

# Creating backing file follows same way as spawning instances.

cache\_name **=** os**.**path**.**basename**(**info**[**'backing\_file'**])**

# Remove any size tags which the cache manages

cache\_name **=** cache\_name**.**split**(**'\_'**)[**0**]**

# 下面的流程与创建流程相同

image **=** self**.**image\_backend**.**image**(**instance**[**'name'**],**

instance\_disk**,**

FLAGS**.**libvirt\_images\_type**)**

image**.**cache**(**fetch\_func**=**libvirt\_utils**.**fetch\_image**,**

context**=**ctxt**,**

filename**=**cache\_name**,**

image\_id**=**instance**[**'image\_ref'**],**

user\_id**=**instance**[**'user\_id'**],**

project\_id**=**instance**[**'project\_id'**],**

size**=**info**[**'virt\_disk\_size'**])**

1. Ephemeral盘

与root盘相同

1. Swap盘

与root盘相同