目 录

1 snapshot 的流程

1.1 数据结构

1.1.1 req

```
# nova/api/openstack/wsgi.py Request
```

1.1.2 body

```
{u'createImage': {u'name': u'snap1', u'metadata': {}}}
```

1.1.3 Instance

该类是代表虚拟机数据库抽象的类。

```
# nova/objects/instance.py Instance
# nova.db.sqlalchemy.models.Instance
```

1.1.4 bdms

```
# nova/objects/block_device.py BlockDeviceMappingList
```

1.1.5 Image

```
# glance.db.sqlalchemy.models.Image
```

1.2 快照入口函数

```
# nova/api/openstack/compute/servers.py Controller._action_create_image()
def _action_create_image(self, req, id, body):
    # id是instance id
    context = req.environ['nova.context']
    entity = body.get("createImage", {})

    image_name = entity.get("name")

    ...

    props = {}
    metadata = entity.get('metadata', {})

    try:
        props.update(metadata)
    ...
    # 根据context和req得到Instance类
    instance = self._get_server(context, req, id)

    bdms = objects.BlockDeviceMappingList.get_by_instance_uuid(
                context, instance.uuid)

    try:
        # 判断root分区是否是volume
        if self.compute_api.is_volume_backed_instance(context, instance,
                                                      bdms):
            img = instance['image_ref']
            if not img:
                properties = bdms.root_metadata(
                        context, self.compute_api.image_api,
                        self.compute_api.volume_api)
                image_meta = {'properties': properties}
            else:
                image_meta = self.compute_api.image_api.get(context, img)

            # Snapshot the given volume-backed instance
            image = self.compute_api.snapshot_volume_backed(
                                                    context,
                                                    instance,
                                                    image_meta,
                                                    image_name,
                                                    extra_properties=props)
        else:
            # 做快照的正常流程
            image = self.compute_api.snapshot(context,
                                              instance,
                                              image_name,
                                              extra_properties=props)
    ...

    # 以下代码的功能: build location of newly-created image entity
    image_id = str(image['id'])
    url_prefix = self._view_builder._update_glance_link_prefix(
```

```
54                 req.application_url)
55         image_ref = os.path.join(url_prefix,
56                                  context.project_id,
57                                  'images',
58                                  image_id)
59
60         resp = webob.Response(status_int=202)
61         resp.headers['Location'] = image_ref
62         return resp
```

1.3 使用 RPC 与 nova-compute 服务通信

```
1     # nova/compute/api.py API.snapshot()
2     def snapshot(self, context, instance, name, extra_properties=None):
3         # instance: nova.db.sqlalchemy.models.Instance
4         # name: name of the snapshot
5         # extra_properties: dict of extra image properties to include
6         #                   when creating the image.
7         # returns: A dict containing image metadata
8
9         # 函数功能: Create new image entry in the image service.
10        #           This new image will be reserved for the compute
11        #           manager to upload a snapshot or backup.
12        image_meta = self._create_image(context, instance, name,
13                                        'snapshot',
14                                        extra_properties=extra_properties)
15
16        # 更改instance的task_state
17        instance.task_state = task_states.IMAGE_SNAPSHOT_PENDING
18        # 调用Instance.save()方法更改数据库
19        instance.save(expected_task_state=[None])
20
21        self.compute_rpcapi.snapshot_instance(context, instance,
22                                              image_meta['id'])
23
24        return image_meta
```

```
1     # nova/compute/rpcapi.py ComputeAPI.snapshot_instance()
2     def snapshot_instance(self, ctxt, instance, image_id):
3         # server: the destination host for a message.
4         # server == instance['host']
5         cctxt = self.client.prepare(server=_compute_host(None, instance),
6                 version=version)
7         cctxt.cast(ctxt, 'snapshot_instance',
8                    instance=instance,
9                    image_id=image_id)
```

1.4 nova-compute 中的快照动作

```
# nova/compute/manager.py ComputeManager.snapshot_instance()
def snapshot_instance(self, context, image_id, instance):
    # context: security context
    # instance: a nova.objects.instance.Instance object
    # image_id: glance.db.sqlalchemy.models.Image.Id
    try:
        # 修改instance的task_state
        instance.task_state = task_states.IMAGE_SNAPSHOT
        # 调用Instance.save()方法更改数据库
        instance.save(
                    expected_task_state=task_states.IMAGE_SNAPSHOT_PENDING)
    ...

    self._snapshot_instance(context, image_id, instance,
                            task_states.IMAGE_SNAPSHOT)
```

```
# nova/compute/manager.py ComputeManager._snapshot_instance()
def _snapshot_instance(self, context, image_id, instance,
                       expected_task_state):
    # self.driver.get_info(instance)["state"]
    current_power_state = self._get_power_state(context, instance)
    try:
        # 修改instance的power_state
        instance.power_state = current_power_state
        instance.save()

        if instance.power_state != power_state.RUNNING:
            state = instance.power_state
            running = power_state.RUNNING

        def update_task_state(task_state,
                              expected_state=expected_task_state):
            instance.task_state = task_state
            instance.save(expected_task_state=expected_state)

        # 调用LibvirtDriver.snapshot()函数实现快照功能
        self.driver.snapshot(context, instance, image_id,
                             update_task_state)

        # 记录instance状态
        instance.task_state = None
        instance.save(expected_task_state=task_states.IMAGE_UPLOADING)
    ...
```

1.5 实现快照功能的核心函数

```
def snapshot(self, context, instance, image_id, update_task_state):
    try:
        # 调用virConnect.lookupByName()返回virDomain对象
        virt_dom = self._lookup_by_name(instance['name'])
    ...
```

```
6
7           base_image_ref = instance['image_ref']
8
9           # 得到instance的image的相关数据
10          base = compute_utils.get_image_metadata(
11              context, self._image_api, base_image_ref, instance)
12
13          # Retrieves the information record for a single disk image by image_id
14          snapshot = self._image_api.get(context, image_id)
15
16          # 通过virtDomain.XMLDesc(0)得到xml配置文件，从而得到instance的磁盘路径
17          disk_path = libvirt_utils.find_disk(virt_dom)
18          # 使用“qemu-img info disk_path”获得磁盘信息
19          source_format = libvirt_utils.get_disk_type(disk_path)
20
21          image_format = CONF.libvirt.snapshot_image_format or source_format
22
23          # NOTE(bfilippov): save lvm and rbd as raw
24          if image_format == 'lvm' or image_format == 'rbd':
25              image_format = 'raw'
26
27          # metadata = {'is_public': False,
28          #             'status': 'active',
29          #             'name': snp_name,
30          #             'properties': {
31          #                            'kernel_id': instance['kernel_id'],
32          #                            'image_location': 'snapshot',
33          #                            'image_state': 'available',
34          #                            'owner_id': instance['project_id'],
35          #                            'ramdisk_id': instance['ramdisk_id'],
36          #                            }
37          #             }
38          metadata = self._create_snapshot_metadata(base,
39                                                    instance,
40                                                    image_format,
41                                                    snapshot['name'])
42
43          # 获得快照的名称
44          snapshot_name = uuid.uuid4().hex
45
46          # LIBVIRT_POWER_STATE = {
47          #     VIR_DOMAIN_NOSTATE: power_state.NOSTATE,
48          #     VIR_DOMAIN_RUNNING: power_state.RUNNING,
49          #     VIR_DOMAIN_BLOCKED: power_state.RUNNING,
50          #     VIR_DOMAIN_PAUSED: power_state.PAUSED,
51          #     VIR_DOMAIN_SHUTDOWN: power_state.SHUTDOWN,
52          #     VIR_DOMAIN_SHUTOFF: power_state.SHUTDOWN,
53          #     VIR_DOMAIN_CRASHED: power_state.CRASHED,
54          #     VIR_DOMAIN_PMSUSPENDED: power_state.SUSPENDED,
55          # }
56          # 调用virDomain.info()返回[state, maxMemory, memory, nbVirtCPU, cpuTime]
57          state = LIBVIRT_POWER_STATE[virt_dom.info()[0]]
58
59          # 动态快照要求QEMU 1.3 and Libvirt 1.0.0
60          # Instances with LVM encrypted ephemeral storage只支持静态快照
61          if (self._has_min_version(MIN_LIBVIRT_LIVESNAPSHOT_VERSION,
```

```
62                                      MIN_QEMU_LIVESNAPSHOT_VERSION,
63                                      REQ_HYPERVISOR_LIVESNAPSHOT)
64                  and source_format not in ('lvm', 'rbd')
65                  and not CONF.ephemeral_storage_encryption.enabled):
66              live_snapshot = True
67              try:
68                  # 终止虚拟机磁盘上的active block job
69                  virt_dom.blockJobAbort(disk_path, 0)
70              ...
71          else:
72              live_snapshot = False
73
74          if state == power_state.SHUTDOWN:
75              live_snapshot = False
76
77          # virDomain.managedSave() does not work for LXC
78          # 如果是静态快照，需要执行virDomain.managedSave()函数
79          if CONF.libvirt.virt_type != 'lxc' and not live_snapshot:
80              if state == power_state.RUNNING or state == power_state.PAUSED:
81                  # 卸载虚拟机的pci设备
82                  self._detach_pci_devices(virt_dom,
83                      pci_manager.get_instance_pci_devs(instance))
84                  # 关闭SR-IOV端口
85                  self._detach_sriov_ports(context, instance, virt_dom)
86                  # This method will suspend a domain and save its memory contents to
                        a file on disk.
87                  virt_dom.managedSave(0)
88
89          # 返回Qcow2类
90          # snapshot_backend是nova.virt.libvirt.imagebackend.Qcow2 object
91          snapshot_backend = self.image_backend.snapshot(instance,
92                  disk_path,
93                  image_type=source_format)
94
95          ...
96
97          update_task_state(task_state=task_states.IMAGE_PENDING_UPLOAD)
98          # snapshot_directory == /var/lib/nova/instances/snapshots
99          snapshot_directory = CONF.libvirt.snapshots_directory
100         # 确保有这个目录存在
101         fileutils.ensure_tree(snapshot_directory)
102         with utils.tempdir(dir=snapshot_directory) as tmpdir:
103             try:
104                 # 得到快照路径
105                 out_path = os.path.join(tmpdir, snapshot_name)
106                 if live_snapshot:
107                     os.chmod(tmpdir, 0o701)
108                     # 动态快照的步骤如下:
109                     # disk_path为/var/lib/nova/instances/vm-uuid/disk
110                     # out_path为/var/lib/nova/instances/snapshots/snapshot_name
111                     # 首先调用”qemu-img create -f qcow2 -o backing_file=disk_path
                            的backing_file,size=disk_path的virtual_size outpath.delta
                            “创建镜像
112                     # 然后调用domain.blockRebase(disk_path, disk_delta, 0,
113                     #           libvirt.VIR_DOMAIN_BLOCK_REBASE_COPY |
114                     #           libvirt.VIR_DOMAIN_BLOCK_REBASE_REUSE_EXT |
```

```
115                     #           libvirt.VIR_DOMAIN_BLOCK_REBASE_SHALLOW)函数将
                            disk_path的内容拷贝给outpath.delta
116                     # 然后使用”qemu-img convert“命令将out_path.delta拷贝到
                            out_path
117                     # 这条命令在拷贝过程中，会先把后端镜像和增量镜像合并，然后在拷
                            贝到另外一个镜像文件中，所以会比较久
118                     self._live_snapshot(virt_dom, disk_path, out_path,
119                                         image_format)
120                 else:
121                     # 使用”qemu-img convert“命令将虚拟机磁盘拷贝到out_path
122                     # 这条命令在拷贝过程中，会先把后端镜像和增量镜像合并，然后在拷
                            贝到另外一个镜像文件中，所以会比较久
123                     # 如果不想新建立一个镜像，只是想做个快照，那么这是可以改进的一
                            点
124                     snapshot_backend.snapshot_extract(out_path, image_format)
125             finally:
126                 new_dom = None
127                 # NOTE(dkang): because previous managedSave is not called
128                 #              for LXC, _create_domain must not be called.
129                 if CONF.libvirt.virt_type != 'lxc' and not live_snapshot:
130                     if state == power_state.RUNNING:
131                         # 这种情况new_dom == virt_dom
132                         new_dom = self._create_domain(domain=virt_dom)
133                     elif state == power_state.PAUSED:
134                         # 这种情况new_dom == virt_dom.createWithFlags(libvirt.
                                VIR_DOMAIN_START_PAUSED)
135                         new_dom = self._create_domain(domain=virt_dom,
136                                                       launch_flags=libvirt.VIR_DOMAIN_START_PAUSED)
137                     if new_dom is not None:
138                         # 安装原有的pci设备
139                         self._attach_pci_devices(new_dom,
140                             pci_manager.get_instance_pci_devs(instance))
141                         # 开启SR-IOV端口
142                         self._attach_sriov_ports(context, instance, new_dom)
143 
144             # Upload that image to the image service
145             with libvirt_utils.file_open(out_path) as image_file:
146                 self._image_api.update(context,
147                                        image_id,
148                                        metadata,
149                                        image_file)
```

2 snapshot 优化方案