Qemu guest agent调研

本文主要是对Qemu guest agent调研结果的总结，主要内容如下：

* Qemu guest agent的原理
* Qemu guest agent的实现程度
* Qemu guest agent的社区活跃度
* 基于Qemu guest agent实现监控方案的可行性分析
* Qemu guest agent对网易私有云项目的其他用途
* Qemu guest agent风险分析

为了叙述方便，下文中将用qga来代替Qemu guest agent。

1. **原理分析**

qga是一个运行在虚拟机内部的普通应用程序（可执行文件名称默认为qemu-ga，服务名称默认为qemu-guest-agent），其目的是实现一种宿主机和虚拟机进行交互的方式，这种方式不依赖于网络，而是依赖于virtio-serial（默认首选方式）或者isa-serial，而QEMU则提供了串口设备的模拟及数据交换的通道，最终呈现出来的是一个串口设备（虚拟机内部）和一个unix socket文件（宿主机上）。

qga通过读写串口设备与宿主机上的socket通道进行交互，宿主机上可以使用普通的unix socket读写方式对socket文件进行读写，最终实现与qga的交互，交互的协议与qmp（[QEMU Monitor Protocol](http://wiki.qemu.org/QMP)）相同（简单来说就是使用JSON格式进行数据交换），串口设备的速率通常都较低，所以比较适合小数据量的交换。

QEMU virtio串口设备模拟参数：

/usr/bin/kvm(QEMU) \

……\

-device virtio-serial-pci,id=virtio-serial0,bus=pci.0,addr=0x6 \

-device isa-serial,chardev=charserial1,id=serial1 \

-chardev socket,id=charchannel0,path=/var/lib/libvirt/qemu/test.agent,server,nowait \

-device virtserialport,bus=virtio-serial0.0,nr=1,chardev=charchannel0,id=channel0,\

name=com.163.spice.0

通过上面的参数就可以在宿主机上生成一个unix socket文件，路径为：/var/lib/libvirt/qemu/test.agent，同时在虚拟机内部生成一个serial设备，名字为com.163.spice.0，设备路径为：/dev/vport0p1，映射出来的可读性比较好的路径为：/dev/virtio-ports/com.163.spice.0，可以在运行qga的时候通过-p参数指定读写这个设备。

Libvirt支持QEMU串口相关配置，所以上述参数已经可以通过libvirt进行配置，且更简单直观，配置方式如下：

<channel type='unix'>

<source mode='bind' path='/var/lib/libvirt/qemu/test.agent'/>

<target type='virtio' name='com.163.spice.0'/>

</channel>

需要注意的是libvirt-qemu:kvm用户要有权限读写'/var/lib/libvirt/qemu/test.agent'。

参考资料：<http://wiki.qemu.org/Features/QAPI/GuestAgent>，<http://wiki.libvirt.org/page/Qemu_guest_agent>

1. **实现程度**
2. 已有功能

目前qga最新版本为1.5.50，linux已经实现下面的所有功能，windows仅支持加\*的那些功能：

* guest-sync-delimited\*：
* guest-sync\*
* guest-ping\*
* guest-get-time\*
* guest-set-time\*
* guest-info\*
* guest-shutdown\*
* guest-file-open
* guest-file-close
* guest-file-read
* guest-file-write
* guest-file-seek
* guest-file-flush
* guest-fsfreeze-status
* guest-fsfreeze-freeze
* guest-fsfreeze-thaw
* guest-fstrim
* guest-suspend-disk\*
* guest-suspend-ram\*
* guest-suspend-hybrid
* guest-network-get-interfaces
* guest-get-vcpus
* guest-set-vcpus

参考资料：源码<http://git.qemu.org/?p=qemu.git;a=tree;f=qga;hb=refs/heads/master>

1. 功能扩展方式

参考资料：源码及guest-get-time功能提交记录

<http://git.qemu.org/?p=qemu.git;a=commitdiff;h=6912e6a94cb0a1d650271103efbc3ac2299e4fd0>

1. **社区活跃度**

QEMU社区从2011年7月20号开始在QEMU代码仓库中增加qga功能，最近一次提交在2013年5月18号，总共有100多次提交记录，代码维护人员主要来自redhat和IBM，社区的活跃度不高，但是QEMU本身的提交记录从2003年至今已有27200多条，还是比较活跃的，qga的功能及代码都比较简单，也是活跃度不高的一个重要原因。

QEMU代码仓库地址：git clone git://git.qemu-project.org/qemu.git

qga代码位于QEMU代码的根目录下的qga目录中。

参考资料：代码仓库git log

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