Qemu guest agent调研

[Qemu guest agent调研 1](#_Toc361064555)

[1. 原理分析 2](#_Toc361064556)

[2. 实现程度 3](#_Toc361064557)

[2.1 已有功能 3](#_Toc361064558)

[2.2 功能扩展方式 4](#_Toc361064559)

[3. 社区活跃度 7](#_Toc361064560)

[4. 实现监控方案的可行性 7](#_Toc361064561)

[4.1 监控方案现状 7](#_Toc361064562)

[4.2 当前方案存在的问题 7](#_Toc361064563)

[4.3 采用qga方式的监控方案 8](#_Toc361064564)

[4.4 改为qga方式需要做的工作 9](#_Toc361064565)

[4.5 qga方式存在的问题 9](#_Toc361064566)

[5. 对网易私有云项目的其他用途 9](#_Toc361064567)

[5.1 云主机内部操作系统状态检查 9](#_Toc361064568)

[5.2 冻结云主机内部文件系统 9](#_Toc361064569)

[6. 类似agent比较 10](#_Toc361064570)

[6.1 ovirt-guest-agent 10](#_Toc361064571)

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\*：宿主机发送一个int数字给qga，qga返回这个数字，并且在后续返回字符串响应中加入ascii码为0xff的字符，其作用是检查宿主机与qga通信的同步状态，主要用在宿主机上多客户端与qga通信的情况下客户端间切换过程的状态同步检查，比如有两个客户端A、B，qga发送给A的响应，由于A已经退出，目前B连接到qga的socket，所以这个响应可能被B收到，如果B连接到socket之后，立即发送该请求给qga，响应中加入了这个同步码就能区分是A的响应还是B的响应；在qga返回宿主机客户端发送的int数字之前，qga返回的所有响应都要忽略；
* guest-sync\*：与上面相同，只是不在响应中加入0xff字符；
* guest-ping\*：Ping the guest agent, a non-error return implies success；
* guest-get-time\*：获取虚拟机时间（返回值为相对于1970-01-01 in UTC，Time in nanoseconds.）；
* guest-set-time\*：设置虚拟机时间（输入为相对于1970-01-01 in UTC，Time in nanoseconds.）；
* guest-info\*：返回qga支持的所有命令；
* guest-shutdown\*：关闭虚拟机（支持halt、powerdown、reboot，默认动作为powerdown）；
* guest-file-open：打开虚拟机内的某个文件（返回文件句柄）；
* guest-file-close：关闭打开的虚拟机内的文件；
* guest-file-read：根据文件句柄读取虚拟机内的文件内容（返回base64格式的文件内容）；
* guest-file-write：根据文件句柄写入文件内容到虚拟机内的文件；
* guest-file-seek：Seek to a position in the file, as with fseek(), and return the current file position afterward. Also encapsulates ftell()'s functionality, just Set offset=0, whence=SEEK\_CUR；
* guest-file-flush：Write file changes bufferred in userspace to disk/kernel buffers；
* guest-fsfreeze-status：Get guest fsfreeze state. error state indicates；
* guest-fsfreeze-freeze：Sync and freeze all freezable, local guest filesystems；
* guest-fsfreeze-thaw：Unfreeze all frozen guest filesystems；
* guest-fstrim：Discard (or "trim") blocks which are not in use by the filesystem；
* guest-suspend-disk\*：Suspend guest to disk；
* guest-suspend-ram\*：Suspend guest to ram；
* guest-suspend-hybrid：Save guest state to disk and suspend to ram（This command requires the pm-utils package to be installed in the guest.）；
* guest-network-get-interfaces：Get list of guest IP addresses, MAC addresses and netmasks；
* guest-get-vcpus：Retrieve the list of the guest's logical processors；
* guest-set-vcpus：Attempt to reconfigure (currently: enable/disable) logical processors inside the guest。

参考资料：源码

<http://git.qemu.org/?p=qemu.git;a=blob;f=qga/qapi-schema.json;h=7155b7ab55fc4ef5336fd771ca06905c485fad62;hb=refs/heads/master>

1. 功能扩展方式

qga功能扩展十分方便，只需要在qapi-schema.json文件中定义好功能名称、输入输出数据类型，然后在commands-posix.c里面增加对应的功能函数即可，下面的补丁即在qga中增加一个通过statvfs获取虚拟机磁盘空间信息的功能：

**diff --git a/qga/commands-posix.c b/qga/commands-posix.c**

index e199738..2f42a2f 100644

--- a/qga/commands-posix.c

+++ b/qga/commands-posix.c

@@ -21,6 +21,7 @@

#include <stdio.h>

#include <string.h>

#include <sys/stat.h>

+#include <sys/statvfs.h>

#include <inttypes.h>

#include "qga/guest-agent-core.h"

#include "qga-qmp-commands.h"

@@ -1467,6 +1468,36 @@ void qmp\_guest\_fstrim(bool has\_minimum, int64\_t minimum, Error \*\*err)

}

#endif

+GuestFileSystemStatistics \*qmp\_guest\_get\_statvfs(const char \*path, Error \*\*errp)

+{

+ int ret;

+ GuestFileSystemStatistics \*fs\_stat;

+ struct statvfs \*buf;

+ buf = g\_malloc0(sizeof(struct statvfs));

+

+

+ ret = statvfs(path, buf);

+ if (ret < 0) {

+ error\_setg\_errno(errp, errno, "Failed to get statvfs");

+ return NULL;

+ }

+

+ fs\_stat = g\_malloc0(sizeof(GuestFileSystemStatistics));

+ fs\_stat->f\_bsize = buf->f\_bsize;

+ fs\_stat->f\_frsize = buf->f\_frsize;

+ fs\_stat->f\_blocks = buf->f\_blocks;

+ fs\_stat->f\_bfree = buf->f\_bfree;

+ fs\_stat->f\_bavail = buf->f\_bavail;

+ fs\_stat->f\_files = buf->f\_files;

+ fs\_stat->f\_ffree = buf->f\_ffree;

+ fs\_stat->f\_favail = buf->f\_favail;

+ fs\_stat->f\_fsid = buf->f\_fsid;

+ fs\_stat->f\_flag = buf->f\_flag;

+ fs\_stat->f\_namemax = buf->f\_namemax;

+

+ return fs\_stat;

+}

+

/\* register init/cleanup routines for stateful command groups \*/

void ga\_command\_state\_init(GAState \*s, GACommandState \*cs)

{

**diff --git a/qga/qapi-schema.json b/qga/qapi-schema.json**

index 7155b7a..a071c3f 100644

--- a/qga/qapi-schema.json

+++ b/qga/qapi-schema.json

@@ -638,3 +638,52 @@

{ 'command': 'guest-set-vcpus',

'data': {'vcpus': ['GuestLogicalProcessor'] },

'returns': 'int' }

+

+##

+# @GuestFileSystemStatistics:

+#

+# Information about guest file system statistics.

+#

+# @f\_bsize: file system block size.

+#

+# @f\_frsize: fragment size.

+#

+# @f\_blocks: size of fs in f\_frsize units.

+#

+# @f\_bfree: free blocks.

+#

+# @f\_bavail: free blocks for non-root.

+#

+# @f\_files: inodes.

+#

+# @f\_ffree: free inodes.

+#

+# @f\_favail: free inodes for non-root.

+#

+# @f\_fsid: file system id.

+#

+# @f\_flag: mount flags

+#

+# @f\_namemax: maximum filename length.

+#

+# Since 1.5.10(NetEase)

+##

+{ 'type': 'GuestFileSystemStatistics',

+ 'data': { 'f\_bsize': 'int', 'f\_frsize': 'int', 'f\_blocks': 'int',

+ 'f\_bfree': 'int', 'f\_bavail': 'int', 'f\_files': 'int',

+ 'f\_ffree': 'int', 'f\_favail': 'int', 'f\_fsid': 'int',

+ 'f\_flag': 'int', 'f\_namemax': 'int'} }

+

+##

+# @guest-get-statvfs:

+#

+# Get the information about guest file system statistics by statvfs.

+#

+# Returns: @GuestFileSystemStatistics.

+#

+# Since 1.5.10(NetEase)

+##

+{ 'command': 'guest-get-statvfs',

+ 'data': { 'path': 'str' },

+ 'returns': 'GuestFileSystemStatistics' }

+

中间复杂的类型定义代码，以及头文件包含关系处理都由一个python脚本在编译的时候动态生成出来，这对开发人员来说是非常方便的，开发人员在扩展功能的时候只需要关注输入、输出的数据类型，以及功能的函数内容即可。

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. **实现监控方案的可行性**
2. 监控方案现状

目前云主机监控的实现方法是，在创建云主机的过程中，增加监控脚本及其配置文件、定时任务及监控信息推送配置文件的注入过程，包括四个文件，其中监控信息推送配置文件(/etc/vm\_monitor/info)由管理平台根据云主机所属用户的注册信息以及监控平台相关配置生成，并传入创建云主机的API来实现文件的注入，监控脚本(/etc/vm\_monitor/send\_monitor\_data.py)及其配置文件(/etc/vm\_monitor/monitor\_settings.xml)、定时任务文件(/etc/cron.d/inject\_cron\_job)是包含在NVS经过base64编码后的监控脚本文件inject\_files.json中。

工作模式为，在root账户增加定时任务inject\_cron\_job，其中有一条任务为：root su -c 'python /etc/vm\_monitor/send\_monitor\_data.py' > /dev/null 2>&1，也即每60s收集并推送一次监控信息给监控平台。

1. 当前方案存在的问题

* 依赖云主机内部的python解释器
* 云主机必须存在root账户
* 依赖NVS文件注入功能；并且为了注入这些监控文件对nova的改动也比较大，也无法与社区同步；windows镜像也会注入这些无用的文件，可能导致一些意想不到的问题；另外如果有的镜像的操作系统不在第一个分区上，则注入的监控文件会失效
* 已经运行的云主机内部的监控相关文件更新困难，导致新监控项的添加、推送周期、推送地址等的修改也比较困难，灵活性较差
* Nova中base64编码的注入脚本的代码可读性很差，代码更新及维护困难
* 定位问题一般都需要登录到云主机内部进行，对于采用密钥对登录的云主机来说定位问题比较困难

1. 采用qga方式的监控方案

首先为每个云主机增加virtio-serial的配置，这个只需要修改生成libvirt配置文件的代码即可，并且应该可以提交给社区；其次需要在虚拟机内部安装qga服务；最后需要在宿主机上新增一个服务进程（这里暂定为monitor服务），用来通过与qga交互从云主机内部获取监控信息；总的模块交互流程如下：



图表 1 云主机创建流程中的监控相关操作

monitor服务单次监控信息获取及推送流程如下：



图表 2 单次监控数据获取及推送流程图

1. 改为qga方式需要做的工作

* 需要扩展qga的功能，增加获取文件系统信息的功能（1人天）
* 为qga打包的相关工作（3~5人天）
* 需要重构监控信息获取脚本，把读本地/proc目录相关文件改为通过qga读文件（3~5人天）
* 计算节点新增monitor服务（5~10人天）
* nova代码修改，包括新增virtio-serial配置生成流程以及去掉原有的监控文件注入流程（3人天）
* 已有镜像需要更新，安装qga（1人天）

1. qga方式存在的问题

* 与当前方案的兼容性问题：已创建的云主机仍然采用原有的监控方式进行数据推送，新创建的云主机则采用qga方式进行监控数据的推送（可以通过检查宿主机上没有socket文件来区分新创建还是已有云主机）；监控项保持不变；
* qga升级问题：除非需要增加读取云主机内部相关文件不能获取到相关信息的监控项，否则不需要特别对qga进行升级；如果确实需要升级，可以在云主机内部配置包含新版本qga的apt相关源，通过apt方式进行升级；
* qga稳定性问题：已经简单对其进行过测试，文件读取功能稳定性没有发现问题（每秒读取一次，连续运行超过24小时，测试代码：<https://github.com/aspirer/study/blob/master/qemu-guest-agent/poll_qemu_guest_agent.py>）
* 监控项扩展问题：在不需要对qga进行升级的情况下，只需要升级monitor服务即可（已有云主机也不需要改动）；如果需要同时升级qga（这种情况比较麻烦，如果不升级qga则新的监控项无法推送）；
* 安全问题：主要包括暴露在计算节点的socket文件以及暴露在云主机内部的virtio-serial，这两个文件或设备的安全性问题；外部的socket文件与云主机镜像文件类似，都是有权限控制的，普通用户无法访问；云主机内部的virtio-serial设备可能会被用户攻击，但由于其速率较低，所以对宿主机影响不大，主要的问题可能是用户会伪造监控文件，然后通过virtio-serial设备返回给monitor服务，导致报警异常，之前的监控文件注入方式比qga方式在这一问题上更严重，因为监控完全是通过注入的脚本在云主机内部获取并推送的，用户可以做任何修改，所以这个问题也应该不是问题；
* 用户隐私保护问题：用户可能会担心我们使用qga获取云主机内部的某些敏感文件，但这个问题其实在云计算环境是无法完全避免的，即使不通过qga，我们也可以通过拷贝云主机镜像并重新创建新的云主机的方式获取云主机内部的任意文件，所以这应该也不是问题；
* LXC支持问题：目前不支持LXC，只支持基于kvm的虚拟机。

1. **对网易私有云项目的其他用途**
2. 云主机内部操作系统状态检查

主要是心跳获取过程可以改为主动通过qga进行查询，而不是依赖注入的定时任务进行上报，这样会更准确及时一些。

1. 冻结云主机内部文件系统

可以用于保证快照操作的安全性\*（不影响云主机正常运行，目前有潜在问题）；可以通过扩展qga功能来保证文件系统一致性（sync功能）。

1. **类似agent比较**
2. [ovirt-guest-agent](http://www.ovirt.org/Guest_Agent)

该agent的原理与qga完全相同，是redhat公司为自己的[OVirt虚拟化管理平台](http://www.ovirt.org/Home)开发的与虚拟机交互的方案，与qga不同之处在于redhat采用了python作为其guest agent的开发语言，其支持的协议也是基于JSON格式的，并且它支持部分windows系统（但是配置起来比较复杂），另外它没有提供通用的文件读写功能，对我们的监控实现来说比较麻烦。其支持的功能列表可以在其主页查看到：<http://www.ovirt.org/Guest_Agent>。

参考资料：<http://www.ovirt.org/Category:Ovirt_guest_agent>