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\*：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. Root盘
3. Ephemeral盘

与root盘相同

1. Swap盘

与root盘相同

1. **对网易私有云项目的其他用途**
2. Root盘
3. Ephemeral盘

与root盘相同

1. Swap盘

与root盘相同

1. **类似agent比较**
2. **风险分析**