Invirt: A Technical Overview

Evan Broder and Greg Price
invirt@mit.edu

December 9, 2008

What is Invirt?

Invirt is software to let users create and manage virtual machines, independently.

Invirt powers SIPB's XVM service (`http://xvm.mit.edu`), a virtual machine hosting service funded by ISDA and free for the MIT community.

Features for Users

Some of the user-visible features of Invirt include...

- VM creation and management on the web
- Runs a wide range of Linux distributions, or even Windows
- 3-minute autoinstaller for Debian and Ubuntu
- Access to the VGA console on the web
- Serial console access over ssh
- DNS server: `inode.xvm.mit.edu`
- DHCP server for easy VM configuration
- Ownership and access control through AFS lockers

Features for Admins

Some of the non-user-visible features that make Invirt more useful...

- VMs are load-balanced across multiple servers
- We can migrate VMs from one server to another with minimal service interruption
- The autoinstaller runs as a guest, keeping security problems from affecting the hosts
- VMs are network-isolated and prevented from binding to other VMs' IP addresses
- Few services running on hosts; most services run on VMs

Virtualization and Xen

- Xen uses two different methods to virtualize a machine
- Hardware virtualization
  - AMD-V or Intel VT extensions to x86
  - Privileged instructions attempted inside VM trap to hypervisor
  - Xen uses qemu to emulate devices like the BIOS, the bootloader, block devices, and network devices
- Paravirtualization
  - Guests have limited privilege, privileged instructions fail
  - Instead, modify guest OS to make "hypercalls", invoke hypervisor
  - Available for Linux, a few other OSes

The Database

machines:

machine_id	name	owner	type_id	...
74	inode	price	linux	
731	jos-virt	geofft	linux-hvm	
192	remus	broder	linux	
753	mpp	mpp	linux-hvm	
82	cups	ternus	linux	
175	macathena	macathena	linux	
715	scripts-f9-test	scripts	linux-hvm	
730	j	jmandel	linux	
433	youtomb-sql	freeculture	linux-hvm	
165	water-buffalo	geofft	linux-hvm	
132	metaphysical	rwbarton	linux-hvm	
...				

nics:

machine_id	mac_addr	ip	hostname
74	00:16:3e:07:74:1e	18.181.2.32	fsck
192	00:16:3e:0c:fc:45	18.181.0.95	remus.mit.edu
731	00:16:3e:36:97:1f	18.181.2.96	jos-virt
...			

. . .

Xen config: simple

```
/etc/xen/myvm.cfg
-----------------
name         = 'myvm'
kernel       = '/boot/vmlinuz-2.6.18-6-xen-amd64'
ramdisk      = '/boot/initrd.img-2.6.18-6-xen-amd64'
memory       = '128'
disk         = [ ',sda1,w' ]
vif          = [ 'mac=00:16:3E:CF:CF:CC, ip=18.181.0.142' ]
on_poweroff = 'destroy'
on_reboot    = 'restart'
on_crash     = 'restart'

aperture-science# xm create myvm
```

Xen config: simple

```
/etc/xen/myvm.cfg
-----------------
name         = 'myvm'
kernel       = '/boot/vmlinuz-2.6.18-6-xen-amd64'
ramdisk      = '/boot/initrd.img-2.6.18-6-xen-amd64'
memory       = '128'
disk         = [ ',sda1,w' ]
vif          = [ 'mac=00:16:3E:CF:CF:CC, ip=18.181.0.142' ]
on_poweroff = 'destroy'
on_reboot    = 'restart'
on_crash     = 'restart'

aperture-science# xm create myvm
```

One per VM?

Xen config: simple

```
/etc/xen/myvm.cfg
-----------------
name         = 'myvm'
kernel       = '/boot/vmlinuz-2.6.18-6-xen-amd64'
ramdisk      = '/boot/initrd.img-2.6.18-6-xen-amd64'
memory       = '128'
disk         = [ ',sda1,w' ]
vif          = [ 'mac=00:16:3E:CF:CF:CC, ip=18.181.0.142' ]
on_poweroff = 'destroy'
on_reboot    = 'restart'
on_crash     = 'restart'

aperture-science# xm create myvm
```

One per VM? How to update?

Xen config: simple

```
/etc/xen/myvm.cfg
-----------------
name         = 'myvm'
kernel       = '/boot/vmlinuz-2.6.18-6-xen-amd64'
ramdisk      = '/boot/initrd.img-2.6.18-6-xen-amd64'
memory       = '128'
disk         = [ ',sda1,w' ]
vif          = [ 'mac=00:16:3E:CF:CF:CC, ip=18.181.0.142' ]
on_poweroff = 'destroy'
on_reboot    = 'restart'
on_crash     = 'restart'

aperture-science# xm create myvm
```

One per VM? How to update? Yuck. Better...

Xen config: ours

/etc/xen/invirt-database

```
# -*- mode: python; -*-
from invirt.database import models

machine = models.Machine.get(name=machine_name)

memory = machine.memory
vif = [ 'mac=%s, ip=%s, script=vif-invirtroute netdev=eth2'
        % (n.mac_addr, n.ip)  for n in machine.nics ]
# ...
```

```
shadow-moses# xm create invirt-database machine_name='inode'
```

How servers communicate

Mentioned that most services don't run on hosts.

How servers communicate

Mentioned that most services don't run on hosts.
Website runs on VM.

How servers communicate

Mentioned that most services don't run on hosts.
Website runs on VM. How does it communicate with hosts?

remctl

"remctl is a client/server protocol for executing commands on a remote system"

remctl

"remctl is a client/server protocol for executing commands on a remote system"
A remctl command runs some executable on the server,

remctl

"remctl is a client/server protocol for executing commands on a remote system"
A remctl command runs some executable on the server, ability to restrict access

remctl: Example

```
remus:~ evan$ remctl zsr v get
30
```

remctl: Example

```
remus:~ evan$ remctl zsr v get
30

root@zygorthian-space-raiders:~# less /etc/remctl/conf.d/volume
volume set  /usr/local/bin/volume-set    ANYUSER
volume get  /usr/local/bin/volume-get    ANYUSER
```

remctl: From the website

Website issues remctls for privileged operations:

- Creating and mutating LVM volumes for VM disks
- Getting list of running VMs
- Booting and shutting down VMs

remctl: From the website

Website issues remctls for privileged operations:

- Creating and mutating LVM volumes for VM disks
- Getting list of running VMs
- Booting and shutting down VMs

Easy if there's one server.

remctl: From the website

Website issues remctls for privileged operations:

- Creating and mutating LVM volumes for VM disks
- Getting list of running VMs
- Booting and shutting down VMs

Easy if there's one server. But we have 4.

remctl: From the website

Website issues remctls for privileged operations:

- Creating and mutating LVM volumes for VM disks
- Getting list of running VMs
- Booting and shutting down VMs

Easy if there's one server. But we have 4. So what now?

Invirt remote server

Setup a server to proxy remctl requests through

Invirt remote server

Setup a server to proxy remctl requests through

- Boot a VM on the server with the least RAM in use.

Invirt remote server

Setup a server to proxy remctl requests through

- Boot a VM on the server with the least RAM in use.
- Shutdown, reboot, or get info about a VM based on the server where it's already running.

Invirt remote server

Setup a server to proxy remctl requests through

- Boot a VM on the server with the least RAM in use.
- Shutdown, reboot, or get info about a VM based on the server where it's already running.
- Compile and combine the list of running VMs from all servers

And for bonus points, let users control their own VMs

Invirt remote server

To boot a VM, use

```
root@xvm:~# remctl xvm-remote.mit.edu control remus create
```

Invirt remote server

To boot a VM, use

```
root@xvm:~# remctl xvm-remote.mit.edu control remus create
```

Can have a bunch of lines like

```
control remus /usr/sbin/invirt-remote-proxy-control /etc/remctl/acl/web
```

or...

Invirt remote server

To boot a VM, use

```
root@xvm:~# remctl xvm-remote.mit.edu control remus create
```

Can have a bunch of lines like

```
control remus /usr/sbin/invirt-remote-proxy-control /etc/remctl/acl/web
```

or...what if we could generate the ACL file based on VM access?

Invirt remote server

To boot a VM, use

```
root@xvm:~# remctl xvm-remote.mit.edu control remus create
```

Can have a bunch of lines like

```
control remus /usr/sbin/invirt-remote-proxy-control /etc/remctl/acl/web
```

or...what if we could generate the ACL file based on VM access?
But we don't want caching

FUSE!

- Before: filesystems are kernel modules.

FUSE!

- Before: filesystems are kernel modules. Written in C.

FUSE!

- Before: filesystems are kernel modules. Written in C.
- Now: filesystems are userspace applications.

FUSE!

- Before: filesystems are kernel modules. Written in C.
- Now: filesystems are userspace applications. Bindings available C,

FUSE!

- Before: filesystems are kernel modules. Written in C.
- Now: filesystems are userspace applications. Bindings available C, Java,

FUSE!

- Before: filesystems are kernel modules. Written in C.
- Now: filesystems are userspace applications. Bindings available C, Java, Haskell,

FUSE!

- Before: filesystems are kernel modules. Written in C.
- Now: filesystems are userspace applications. Bindings available C, Java, Haskell, Perl,

FUSE!

- Before: filesystems are kernel modules. Written in C.
- Now: filesystems are userspace applications. Bindings available C, Java, Haskell, Perl, Ruby,

FUSE!

- Before: filesystems are kernel modules. Written in C.
- Now: filesystems are userspace applications. Bindings available C, Java, Haskell, Perl, Ruby, Lua,

FUSE!

- Before: filesystems are kernel modules. Written in C.
- Now: filesystems are userspace applications. Bindings available C, Java, Haskell, Perl, Ruby, Lua, and most importantly,

FUSE!

- Before: filesystems are kernel modules. Written in C.
- Now: filesystems are userspace applications. Bindings available C, Java, Haskell, Perl, Ruby, Lua, and most importantly, Python

RemConfFS

Idea: Use a FUSE filesystem to generate the ACLs

RemConfFS

Idea: Use a FUSE filesystem to generate the ACLs and the list of running VMs

RemConfFS

Idea: Use a FUSE filesystem to generate the ACLs and the list of running VMs from the database

RemConfFS

Idea: Use a FUSE filesystem to generate the ACLs and the list of running VMs from the database on the fly.

RemConfFS

Idea: Use a FUSE filesystem to generate the ACLs and the list of running VMs from the database on the fly.

```
root@xvm-remote:~# mount | tail -n 1
invirt-remconffs on /etc/remctl/remconffs type fuse.invirt-remconffs (rw)
```

RemConfFS

Idea: Use a FUSE filesystem to generate the ACLs and the list of running VMs from the database on the fly.

```
root@xvm-remote:~# mount | tail -n 1
invirt-remconffs on /etc/remctl/remconffs type fuse.invirt-remconffs (rw)
```

```
root@xvm-remote:~# cat /etc/remctl/remconffs/conf
control j /usr/sbin/invirt-remote-proxy-control /etc/remctl/remconffs/acl/j
control remus /usr/sbin/invirt-remote-proxy-control /etc/remctl/remconffs/acl/remus
control mpp /usr/sbin/invirt-remote-proxy-control /etc/remctl/remconffs/acl/mpp
control cups /usr/sbin/invirt-remote-proxy-control /etc/remctl/remconffs/acl/cups
control macathena /usr/sbin/invirt-remote-proxy-control /etc/remctl/remconffs/acl/macathena
control scripts-afs-test /usr/sbin/invirt-remote-proxy-control /etc/remctl/remconffs/acl/scripts-afs-test
control scripts-f9-test /usr/sbin/invirt-remote-proxy-control /etc/remctl/remconffs/acl/scripts-f9-test
control youtomb-sql /usr/sbin/invirt-remote-proxy-control /etc/remctl/remconffs/acl/youtomb-sql
...
```

RemConfFS

Idea: Use a FUSE filesystem to generate the ACLs and the list of running VMs from the database on the fly.

```
root@xvm-remote:~# mount | tail -n 1
invirt-remconffs on /etc/remctl/remconffs type fuse.invirt-remconffs (rw)
```

```
root@xvm-remote:~# cat /etc/remctl/remconffs/conf
control j /usr/sbin/invirt-remote-proxy-control /etc/remctl/remconffs/acl/j
control remus /usr/sbin/invirt-remote-proxy-control /etc/remctl/remconffs/acl/remus
control mpp /usr/sbin/invirt-remote-proxy-control /etc/remctl/remconffs/acl/mpp
control cups /usr/sbin/invirt-remote-proxy-control /etc/remctl/remconffs/acl/cups
control macathena /usr/sbin/invirt-remote-proxy-control /etc/remctl/remconffs/acl/macathena
control scripts-afs-test /usr/sbin/invirt-remote-proxy-control /etc/remctl/remconffs/acl/scripts-afs-test
control scripts-f9-test /usr/sbin/invirt-remote-proxy-control /etc/remctl/remconffs/acl/scripts-f9-test
control youtomb-sql /usr/sbin/invirt-remote-proxy-control /etc/remctl/remconffs/acl/youtomb-sql
...
```

```
root@xvm-remote:~# cat /etc/remctl/remconffs/acl/remus
broder@ATHENA.MIT.EDU
broder/root@ATHENA.MIT.EDU
include /etc/remctl/acl/web
```

Serial console server

How do you get to the serial console?

```
xm console d_remus
```

Serial console server

How do you get to the serial console?

`xm console d_remus`

But we don't want people connecting directly to the hosts.

conserver

Run some arbitrary command to get to something like a serial console.
Then expose that locally or over the network.

```
console -M arklay-mansion.mit.edu remus
```

conserver: Simple config

```
console remus {
    type exec;
    exec xm console d_f;
    execsubst f=cs;
}
```

conserver: Simple config

```
console remus {
    type exec;
    exec xm console d_f;
    execsubst f=cs;
}
```

Again, one per VM?

conserver: Our config

```
default * {
    type exec;
    exec xm console d_f;
    execsubst f=cs;
}

console remus { master arklay-mansion.mit.edu; }
```

conserver on console server

- conserver can be used as a proxy for other instances of conserver.

conserver on console server

- conserver can be used as a proxy for other instances of conserver.
- Just need to tell it where the "master" conserver is

conserver on console server

- conserver can be used as a proxy for other instances of conserver.
- Just need to tell it where the "master" conserver is
- console server needs a series of lines of the form
  `console remus { master arklay-mansion.mit.edu; }`

Accessing the console server

Give access to a list of users with .k5login file

```
vinegar-pot:~# cat .k5login
tabbott/root@ATHENA.MIT.EDU
nelhage/root@ATHENA.MIT.EDU
jbarnold/root@ATHENA.MIT.EDU
andersk/root@ATHENA.MIT.EDU
broder/root@ATHENA.MIT.EDU
```

Accessing the console server

Give access to a list of users with .k5login file

```
vinegar-pot:~# cat .k5login
tabbott/root@ATHENA.MIT.EDU
nelhage/root@ATHENA.MIT.EDU
jbarnold/root@ATHENA.MIT.EDU
andersk/root@ATHENA.MIT.EDU
broder/root@ATHENA.MIT.EDU
```

But how to keep it up to date?

Accessing the console server

FUSE!

```
root@xvm-console:~# cat /consolefs/remus/.k5login
broder@ATHENA.MIT.EDU
broder/root@ATHENA.MIT.EDU
```

Accessing the console server

FUSE!

```
root@xvm-console:~# cat /consolefs/remus/.k5login
broder@ATHENA.MIT.EDU
broder/root@ATHENA.MIT.EDU
```

I can login as `remus@xvm-console.mit.edu`.

Accessing the console server

FUSE!

```
root@xvm-console:~# cat /consolefs/remus/.k5login
broder@ATHENA.MIT.EDU
broder/root@ATHENA.MIT.EDU
```

I can login as `remus@xvm-console.mit.edu`.
Why does that user exist?

NSS: normal config

NSS, or name service switch

NSS: normal config

NSS, or name service switch
Look at `/etc/passwd`, `/etc/group` by default

NSS: normal config

NSS, or name service switch

Look at `/etc/passwd`, `/etc/group` by default

```
mass-toolpike:~ broder$ grep broder /etc/passwd
broder:x:41803:101:Evan Broder,,,,6173244655:/mit/broder:/bin/athena/bash

mass-toolpike:~ broder$ getent passwd broder
broder:x:41803:101:Evan Broder,,,,6173244655:/mit/broder:/bin/athena/bash
```

NSS: our config

From /etc/nsswitch.conf:

```
passwd:         compat pgsql
group:          compat pgsql
```

NSS: our config

From /etc/nsswitch.conf:

```
passwd:         compat pgsql
group:          compat pgsql
```

From /etc/nss-pgsql.conf:

```
getpwnam = SELECT name, '*', name, '/consolefs/'|| name, \
    '/usr/bin/invirt-consolesh', machine_id + 1000, \
    machine_id + 1000 FROM machines WHERE name = $1
```

NSS: our config

From /etc/nsswitch.conf:

```
passwd:         compat pgsql
group:          compat pgsql
```

From /etc/nss-pgsql.conf:

```
getpwnam = SELECT name, '*', name, '/consolefs/'|| name, \
    '/usr/bin/invirt-consolesh', machine_id + 1000, \
    machine_id + 1000 FROM machines WHERE name = $1
```

```
root@xvm-console:~# getent passwd remus
remus:*:1192:1192:remus:/consolefs/remus:/usr/bin/invirt-consolesh
```

Console server shell

Finally, restrict what users can do.

Console server shell

Finally, restrict what users can do. Make this their shell:

```
#!/bin/bash
exec /usr/bin/console "$USER"
```

Configuring Invirt

/etc/remctl/acl/remote on remote-server

```
host/xvm-remote.mit.edu@ATHENA.MIT.EDU
```

/var/www/invirt-web/controls.py on web server

```
def lvcreate(machine, disk):
    """Create a single disk for a machine"""
    remctl('xvm-remote.mit.edu', 'web', 'lvcreate', ...)
```

Repeating ourselves?

Configuring Invirt

/etc/invirt/invirt.yaml everywhere

```
remote:
 hostname: xvm-remote.mit.edu
 ip: 18.181.0.188
...
```

/var/www/invirt-web/controls.py on web server

```
from invirt import config
def lvcreate(machine, disk):
    remctl(config.remote.hostname, 'web', 'lvcreate', ...)
```

Configuring Invirt

/etc/invirt/invirt.yaml everywhere

```
----------------------
remote:
 hostname: xvm-remote.mit.edu
 ip: 18.181.0.188
...
```

/var/www/invirt-web/controls.py on web server

```
-----------------------------
from invirt import config
def lvcreate(machine, disk):
    remctl(config.remote.hostname, 'web', 'lvcreate', ...)
```

But what about /etc/remctl/acl/remote?
Not a programming language.

Configuring Invirt

```
/etc/remctl/acl/remote.mako
---------------------------
<% from invirt import config %>\
host/${config.remote.hostname}@${config.kerberos.realm}
```

Initscript compiles real file from template.
Used in ten of our packages, on all our systems.

What's Next?

What's Next?

- Improve website

What's Next?

- Improve website
- Host information in Moira

What's Next?

- Improve website
- Host information in Moira
- Better usage documentation

What's Next?

- Improve website
- Host information in Moira
- Better usage documentation
- Deal with CPU limits, abandoned VMs

What's Next?

- Improve website
- Host information in Moira
- Better usage documentation
- Deal with CPU limits, abandoned VMs
- Make Invirt more generic

What's Next?

- Improve website
- Host information in Moira
- Better usage documentation
- Deal with CPU limits, abandoned VMs
- Make Invirt more generic
- Stronger advertising

What's Next?

- Improve website
- Host information in Moira
- Better usage documentation
- Deal with CPU limits, abandoned VMs
- Make Invirt more generic
- Stronger advertising
- Continue to follow new software

What's Next?

- Improve website
- Host information in Moira
- Better usage documentation
- Deal with CPU limits, abandoned VMs
- Make Invirt more generic
- Stronger advertising
- Continue to follow new software
- Take over the world!