XenoControl

High Availability
Management Console
For Xen Clusters

Jérôme Petazzoni <jp@enix.fr>

RMLL 2010

Outline

- Who ? (please allow me to introduce myself)
- Why ? (goals & purposes)
- What ? (results & samples)
- How ? (tools & libraries)
- XenoControl internals
- Roadmap
- Questions

Who

- Enix
  - Small hosting company
  - No money to buy expansive hardware
  - We love Open Source
  - Xen hosting since 2004
  - Before Xen : UML (UserModeLinux)

Who

- SmartJog / TVRadio (TDF group)
  - Part of a big company
  - Have money, but don't want to waste it
  - They love Open Source
  - Streaming, transcoding, etc.
  - Resource usage varies all the time

Why

- Need something to manage hosts & VM
  - List all VM in realtime
  - List resources usage and availability (CPU, RAM, storage, network)
  - Use standard commands & tools (xm, ***LVM***, standard Xen config files)
  - Live migration of VM *without SAN*
  - Scriptability (GUI are out of question)

Really, why ?

- Existing GUI also lack features
- Doing parallel SSH on 100+ dom0 is wrong
  - Needs some kind of registry/enumeration
  - Quickly turns into a hacky nightmare
- We want the system to be non intrusive
  - Must be able to plug/unplug XenoControl
  - Zero training to use the system (lazyness!)

The real killer feature

- Live migration of VM with local storage
  - Why live migration ?
    - Redeploy resources
    - Hardware maintenance
  - Why local storage ?
    - Excellent performance without requiring high speed network (IB, 10G)
    - Cheap boxes available from misc. makers (4U, dual-socket Nehalem, 72GB RAM, 24x1 TB HDD, less than 10KEUR)

What
(Xen requirements)

- Standard Xen setup
- VM configuration stored in /etc/xen/auto
- Uses LVM on a single VG
  - VG naming must be the same everywhere
- Network setup is irrelevant
  - As long as you use standard Xen facilities (i.e. /etc/xen/scripts/{vif,network}-*)
  - All dom0 should be on the same Ethernet

What
(other requirements)

- Spread toolkit (more about this later)
- Python (2.6 and above)
- DRBD in the dom0 (not needed in domU)
  (but DRBD is not inherent to XenoControl)
- Linux dom0 !

What (results)

```
milky:~# xenocontrol vmlist
Got a reply from #xenhost#xenlab
Got a reply from #xenhost#andromede
Got a reply from #xenhost#milky
Got a reply from #xenhost#medusa
      host                  name  vcpu  memMB      power_state
 andromede              Domain-0   16   976          Running
 andromede             enix.kran    1  1000          Running
 andromede           zeen.obiwan    1  1000          Running
         …                     …          …                …
    medusa              Domain-0    8   976          Running
    medusa         enix.arachnee    1   512          Running
    medusa       europnet.bisque    1   250          Running
         …                     …          …                …
     milky              Domain-0   16   976          Running
     milky       enix.dotcloud-1    1  4000          Running
     milky       enix.dotcloud-2    1  1000          Running
     milky       enix.dotcloud-3    1  1000          Running
     milky         libe.back-dev    2  7000          Running
```

What
(more results)

```
milky:~# xenocontrol hoststats
…
  name    hvm   enabled  cpu   cpu_usage memMB  memfreeMB   nr_vm     disk_usage      net_usage
andromed True      True   16       5.6 % 73718      62975       8     0 /   209 KB     1 /     2 KB
medusa   True      True    8     130.1 % 32766       1242      38    36 /  2822 KB   177 /  1302 KB
 milky   True      True   16       9.0 % 73719      11066      23   236 /  3030 KB   186 /  5919 KB
xenlab   True      True    4       2.1 %  8074       6841       2    91 /    14 KB     3 /    75 KB
4 host responded (4 enabled, 0 busy) for a total of 44 corethread hosting 71 vm
CPU    : 42.5 free on 44 ; Used at 3.34 %
Memory : 80.2G free on 183.9G ; Used at 56.38 %
Disks  : 7.7T free on 17.5T ; Used at 55.88 %

milky:~# xenocontrol hostlist
…
  name    hvm cpu  memMB freeMB  vm  stor_freeGB xen_ver   kernel_version       cpu_model
andromed True  16  73718  62975   8         3986   3.2   2.6.30.1-xen-amd64 Intel(R) Xeon(R) CPU E5520  @ 2.27GHz
medusa   True   8  32766   1242  38         1625   3.2   2.6.30.1-xen-amd64 Intel(R) Xeon(R) CPU E5405  @ 2.00GHz
 milky   True  16  73719  11066  23         2110   3.2   2.6.30.1-xen-amd64 Intel(R) Xeon(R) CPU E5520  @ 2.27GHz
xenlab   True   4   8074   6841   2          179   4.0   2.6.32-5-xen-amd64 Intel(R) Xeon(R) CPU X3430  @ 2.40GHz
4 host responded (4 enabled, 0 busy) for a total of 44 corethread hosting 71 vm
```

What (easier management)

- All actions can use wildcards
- Need to do_something on a bunch of VM ?

  ```
  # xenocontrol do_something *webfront*
  # xenocontrol do_something host17/*sql*
  ```
- Various hooks everywhere
  - Existing deployment tool was successfully integrated with Xenocontrol

How
(architecture decisions)

- Single-file Python script
  - Easy distribution even without packaging
  - Can upgrade code automatically
- Communication : spread toolkit
  - Reliable and efficient group communication
  - Setup is boring, but easy

How
(live migration steps)

- Prepare DRBD and remote LVM
- Magically switch block backend to DRBD
- Wait for DRBD to synchronize data
- Call “xm migrate”
- Magically switch block backend to LVM
- Tear down DRBD
- Tear down local LVM

How
(live migration steps)

Watch the lovely drawings

(in separate file)

Internals (automations)

- No centralized master
- Each complex command is an “automation” (recipe made of simple tasks)
- Automation = high-availability parallel job, run on the whole cloud of dom0 nodes
- Automation is controlled by *initiator*
- Automation can be resumed or aborted (in case of error or crash)

Internals (requests)

- Automations can spawn workers (at least one, else no job is done)
- These workers will issue *requests*
- Requests can be executed by other hosts
- Requests are stateless
- State is kept on all nodes

Internals
(the magic)

How do we *magically* replace block devices ?

- “xm pause” the VM
- Use dmsetup to remap blocks (LVM trick)
- “xm unpause” the VM

The dmsetup call is quick :

- VM is not disturbed
- Unless running realtime tasks (VOIP...)

Internals (restrictions)

- Works only for Linux dom0
  - But domU can be anything
  - Won't work with Solaris or NetBSD dom0 (But there might be another way!)
- Is not limited to DRBD and LVM
  - Can work with iSCSI, AOE …
  - Should work with btrfs, glusterfs …
  - Requirement : dmsetup

Internals (black Python magic)

- Automations are made of workers
- Workers contain only logic control
- They are implemented using *continuations*
- Those *continuations* are implemented using generators and special *yield* syntax
- All real job is done by issuing *requests*
- *Requests* can execute elsewhere

Internals
(internals of internals)

- Each automaton = 1 spread group
- Each group = one master (initiator) elected
- Only the master will send data
- Other nodes will run the control logic, receive all data, but send nothing
- State is kept in sync on all nodes
- Allow for recovery in case of failure

Internals
(black magic unveiled, part 1)

- Engine code :
  start_at_least_one_worker()
- Worker code :
  … some control logic
  … then need to do something
  response = yield Request(requestparams)
  … some more control logic
  (hopefully using the response)

Internals
(black magic unveiled, part 2)

- Engine (pseudo-)code :

```
# For each worker...
request = worker.next() # first request
while not_finished:
    # initiator is the only one to send
    if initiator:
        send_to_network(request)
    # but everyone receives requests
    # and their responses, to keep state
    response = read_from_network()
    request = worker.send(response)
# And multiple workers run in parallel :)
```

Internals
(black magic by the book)

```
class DeployAutomation(AutomationProcessor):
   def s_init(self, args):
      vm_to_deploy = parse_opts(args)
      for vm in vm_to_deploy:
         possible_hosts = Request('can_host', vm, host='*')
         chosen_host = self.random.choice(possible_hosts)
         Request('set_busy', True, host=chosen_host)
         self.worker.add(self.s_deploy(vm, chosen_host))
   def s_deploy(self, vm, chosen_host):
      result = Request('install_vm', vm, host=chosen_host)
      if result != 'OK':
         Request('cleanup_vm', vm, host=chosen_host)
      else:
         Request('start_vm', vm, host=chosen_host)
      Request('set_busy', False, host=chosen_host)
```

Internals (limitations)

- Can't use random
  - Initialize each worker with a common seed
  - So you can use Python's random after all
- Can't interact with outer world (sockets...)
  - Workers must delegate communication
  - State must be consistent :
    inform other nodes of what's happening
    … this is actually done automatically

Roadmap
(what we want to do)

- Code cleanup
- Allow VG with different names
- Allow multiple VG
- Implement migration from/to iSCSI
- Add external control web server

Roadmap (what you can do)

- If you know kung-fu : new automations
  - Support for AOE, iSCSI, glusterfs, btrfs …
  - Integration with SAN management
- If you know Python : better output format
  - Add proper tabular/XML/JSON output
- If you know english : better messages
  - Right now everything is in frengrish

Roadmap
(cool stuff we dream about)

- Integration with ovirt
  - So you can get all the cool ovirt tools
- KVM / OpenVZ support
  - Because Xen sometimes sucks
  - And because KVM and OpenVZ are great
- Replace spread with some AMQP flavour
  - Spread has rough edges, but gracefully handles every nuke we throw at it

Conclusion

- Useful tool, used internally by two really different user profiles
- Neat architecture
  (or at least, that's what we think)
- Needs a great amount of cleanup, doc …
- You can also see XenoControl as a POC

Questions ?

http://xenocontrol.enix.org/

http://www.enix.fr/
jp@enix.fr

http://www.smartjog.com/