Cloudbursting computational clusters

Riccardo Murri

<riccardo.murri@uzh.ch>

GC3: Grid Computing Competence Center,
University of Zurich

Nov. 29, 2012

VM-MAD / 1

Virtual **M**achines **M**anagement
and **A**dvanced **D**eployment

Joint project of ETH, UZH, FGCZ, SWITCH
funded under the AAA/SWITCH scheme

"Provide simple mechanisms to deploy complex scientific applications on heterogeneous hardware and software resources using virtualization techniques."

VM-MAD / 2

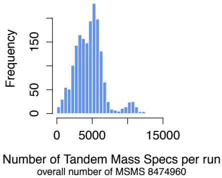
Minimal impact on current usage patterns

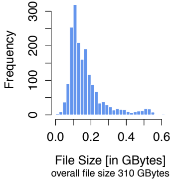
Progressive migration
from classic "HPC cluster in the basement" model
towards virtualized infrastructures

Cloudbursting

Integration with the SMSCG
national grid infrastructure

Example use case: Drosophila proteome





Some numbers from the “bio” side:

- 1’800 (LC)-MS/MS runs
- $\pm$3Da peptide mass tolerance
- $\approx$ 10’000 peptides in the MS window
- 8’474’960 MS/MS

Reference: Nature Biotechnology 25, 576–583 (2007).

On the “IT” side:

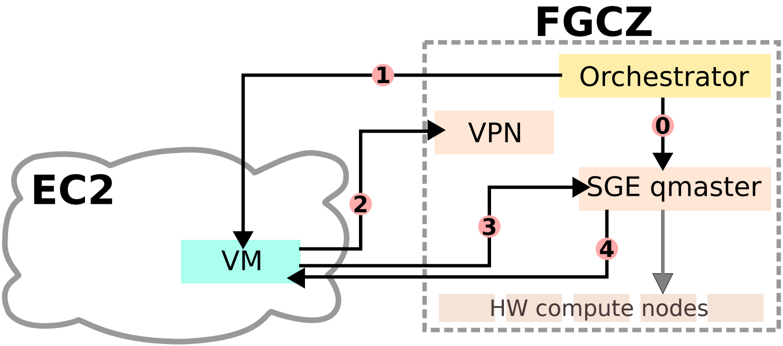
- Many *independent* single-thread jobs
- Perfect match for a batch-computing cluster!
- But local compute cluster already quite busy. . .

Implementation idea

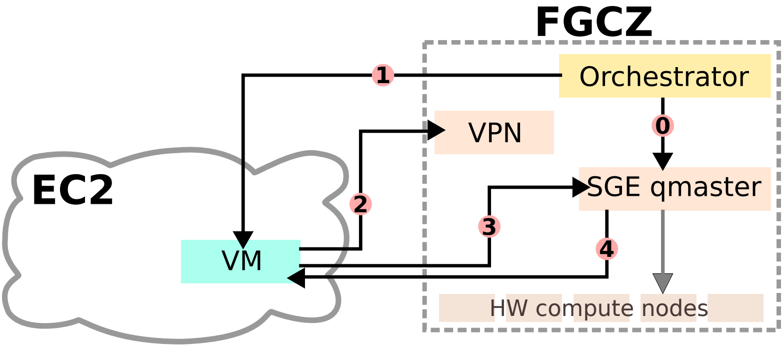
Expand FGCZ computing resources on demand.

"Orchestrator" to control the VM infrastructure:

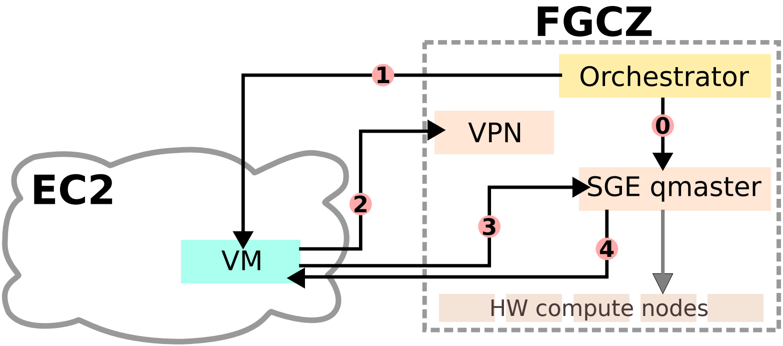
- Monitors batch system queues
- Starts and shuts down VM instances according to *configurable policies and metrics*
- Adds/removes VMs as compute nodes to the cluster



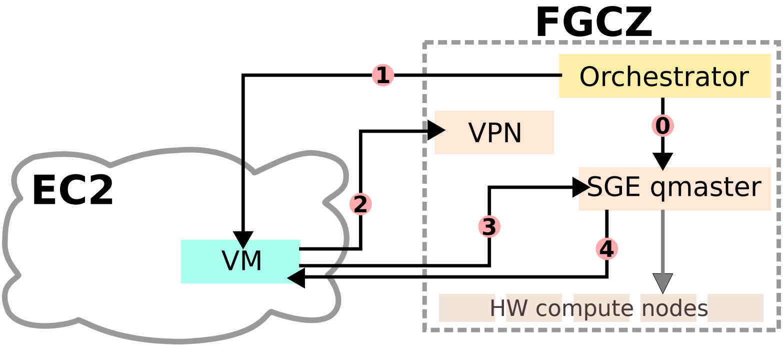
0. The orchestrator monitors the batch system state and determines a new compute node is needed.
1. A new VM is started
2. The VM connects back to the FGCZ network via VPN
3. The VM is added to the cluster as a compute node
4. SGE can now start jobs on the VM



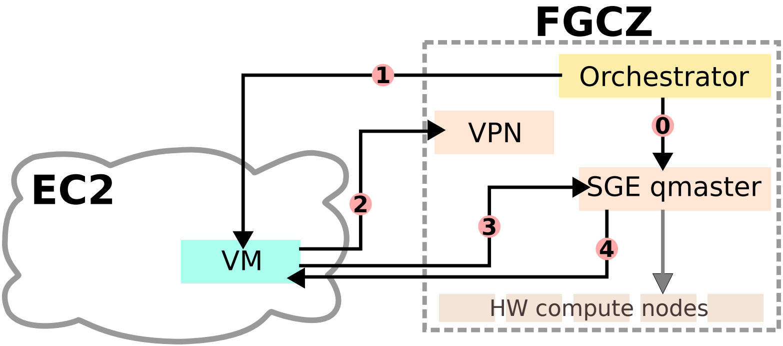
0. The orchestrator monitors the batch system state and determines a new compute node is needed.
1. A new VM is started
2. The VM connects back to the FGCZ network via VPN
3. The VM is added to the cluster as a compute node
4. SGE can now start jobs on the VM



0. The orchestrator monitors the batch system state and determines a new compute node is needed.
1. A new VM is started
2. The VM connects back to the FGCZ network via VPN
3. The VM is added to the cluster as a compute node
4. SGE can now start jobs on the VM



0. The orchestrator monitors the batch system state and determines a new compute node is needed.
1. A new VM is started
2. The VM connects back to the FGCZ network via VPN
3. The VM is added to the cluster as a compute node
4. SGE can now start jobs on the VM



0. The orchestrator monitors the batch system state and determines a new compute node is needed.
1. A new VM is started
2. The VM connects back to the FGCZ network via VPN
3. The VM is added to the cluster as a compute node
4. SGE can now start jobs on the VM

Orchestrator features

Entirely written in Python.

Web-based frontend to oversee the status.

Pluggable batch system interface: not limited to GridEngine.

Pluggable cloud backend: can use any cloud supported by Apache LibCloud, or the SMSCG grid.

Policy configuration / 1

Criteria for starting/stopping a VM
are defined using Python code:

```
def is_cloud_candidate(self, job):
  # only jobs submitted to the `cloud` queue
  # are candidates for running on VMs
  return (job.queue == 'cloud.q')

def is_new_vm_needed(self):
  # if we have more jobs queued than started VMs,
  # start a new one
  if len(self.candidates) > 2*len(self.vms):
    return True
  else:
    return False
```

Policy configuration / 2

Criteria for starting/stopping a VM
are defined using Python code:

```
def can_vm_be_stopped(self, vm):
  TIMEOUT = 10*60 # 10 minutes
  if vm.last_idle > TIMEOUT:
    return True
  else:
    return False
```

Orchestrator construction kit

Actually, *all* configuration is done by subclassing the Orchestrator and customizing to taste:

```
class DemoOrchestrator(OrchestratorWebApp):
  def __init__(self, flaskapp):
    OrchestratorWebApp.__init__(
      self,
      flaskapp,
      interval=30,
      cloud=EC2Cloud(...)
      batchsys=GridEngine('bfabric'),
      max_vms=8,
      chkptfile='vm-mad.state')
```

Simulation mode

Has a *simulation mode*:

- Reads job descriptions from cluster accounting file
- Simulates spawning of VMs to run those jobs

Uses:

- To test policies against real cluster workload.
- To estimate the optimal ratio between own resources and rented resources.


add max 0 vms
Jan
Mar
add max 40 vms
Jan
Mar
May
Jul
Sep
Nov
add max 160 vms
2009
2010
add max 320 vms
2009
2010
add max 640 vms
2009
2010
add max 1024 vms
2009
2010

Interface to the cloud(s)

Apache Libcloud is Python adapter library that abstracts away differences among multiple cloud provider APIs.

```
def start_vm(self, vm):
  vm.instance = self.provider.create_node(
    name=str(vm.vmid),
    image=self._images[self.image],
    size=self._kinds[self.kind])
  [...]
```

Providers exist for EC2, Rackspace, CloudSigma, GoGrid, OpenStack, Eucalyptus, . . .
(more than 26 different providers)

Integration with SMSCG / 1

User-mode Linux is a Linux virtualization technology,
running entirely in user-space.

UML consists of a modified Linux kernel (guest),
that runs as a regular process
within another Linux system (host).

UML features

Any file in the host system can be a block device (*ubdX*). Uses *copy-on-write*, so one filesystem image can be used by many UML instances concurrently.

Can mount any directory in the host filesystem as a local *hostfs* filesystem.

Outbound net connectivity with a helper program (*slirp*).

Local networks of UML instances, backing on IP multicast.

Integration with SMSCG / 2

Idea:
run a UML machine as a Grid job†

This allows us to run a "virtual compute node" inside the compute node of another cluster.

All we need is a different backend for the Orchestrator, all the rest stays the same.

†: This idea can be taken much further and has spun off into a software project of its own, named AppPot.

Questions ?

Thank you!

References

VM-MAD software home:
http://vm-mad.googlecode.com

mailing list:
virtualization@gc3.lists.uzh.ch

Thank you!

Web frontend screenshot

DemoOrchestrator http://localhost:5000/

Orchestrator status

The current orchestrator status is as follows:

- 3 cycles have passed
- 3 VMs have been started
- 2 VMs are currently active (ready for processing jobs)

Started VMs

ID	State	Node name	
1	READY	vm-1	
2	READY	vm-2	
3	STARTING	(unknown)	Mark as ready

AppPot / 1

AppPot consists of:

- a base image (with the AppPot boot script)
  - *raw* disk image
  - can be run in *any* virtualization software: KVM, Xen, VirtualBox (and obviously UML!)
- a startup script `apppot-start`
- three support programs `linux`, `slirp`, `empty`

You can run an AppPot UML machine either locally on your computer, textbfon the Grid, or in a IaaS cloud.

▸ Back to main talk

AppPot / 2

AppPot supports a *base* + *changes* mechanism.

The command "`apppot-snap base`" records a snapshot of the current system state (file sizes, timestamps, etc.). This should be used by sysadmins / application experts when they are done preparing the base filesystem image.

The command "`apppot-snap changes`" creates a tarball with all the modifications since the last recorded base.

Users only submit the changes, the startup script automatically merges them into the running AppPot instance.

Back to main talk

AppPot / 3

...Complex application deployment?

- An application expert creates an AppPot base image with the software correctly installed and validated.
- Users just submit it as a Grid job.

...Running own code on the Grid?

- Users get a copy of the base image, install their code in it and do the development work (e.g., on their laptops).
- When they want to do a production run, they submit a job attaching the *changes* file.

Back to main talk