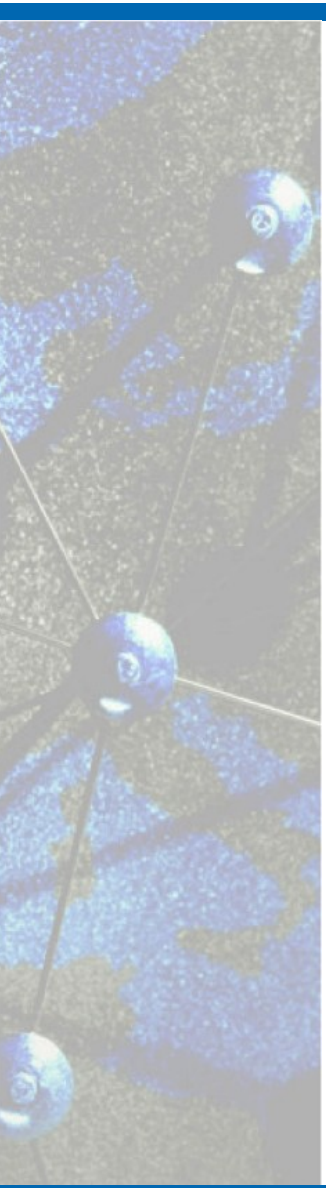

AppPot
User-mode Linux virtualization on the Grid

Sergio Maffioletti
<sergio.maffioletti@gc3.uzh.ch>

Grid Computing Competence Centre,
University of Zurich
http://www.gc3.uzh.ch/

What are we trying to solve?

Two basic problems:

- **Deploying complex applications (at Grid sites)**
- **Running own-written code on a Grid**

One solution:

- virtual machines!

Example 1: GAMESS(US)

Many scientific codes require experience to be built properly.

GAMESS(US) is a program for ab initio molecular quantum chemistry.

Legacy build system, consisting in a number of C-shell scripts, FORTRAN code preprocessors (written themselves in FORTRAN), hand-editing files and much patience.

It's easy to get an incorrect build.

And there's no validation suite, so you will only find out when users complain.

Example 2: Alpine 3D

Code developed by WSL for high resolution simulation of alpine surface processes.

Depends on the large MeteoIO library, plus some not-so-standard build procedure.

Alpine3D is an ongoing development effort. Developers need to test and validate new part of Alpine3D with some well-defined simulation suites.

Even simple simulations can take hours. **How can we support running *modified* Alpine3D on the Grid?**

What is UML?

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 process within another Linux system *(host)*.

Other than that, it's a regular kernel, so it can run *any* Linux distribution with *any* configuration.

See: `http://user-mode-linux.sf.net/`

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.

Idea:

Run a UML machine as a Grid job.

So what is AppPot?

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, or **on the Grid**.

How does it help with. . . *(I)*

. . . 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.

Sysadmins do not need to be involved. . .

How does it help with. . . *(II)*

. . . Running own code on the Grid?

A knowledgeable person creates a *base image*.

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 **just run the modified image as a Grid job.**

Different image for each dataset?

Do I need a separate image for each dataset?

AppPot automatically mounts the current directory on `/home/user/job`.

You can read and write files on to the host filesystem through that.

The job execution is mounted on `/home/user/job`. So you have access to all the files you included in the input- and output-sandboxes.

Running commands

How do I run a specific command in AppPot?

AppPot comes with a standardized startup script, and a modified Linux boot script (installed as `/etc/rc.local`).

Supports 4 different ways of executing a command:

- Give it on the command-line of `apppot-start.sh`,
- or provide an `apppot-run` script in the current directory,
- or provide an `apppot-autorun` script *within the AppPot image*,
- else, starts a shell on the main console and you just type it.

Collect UML output?

Did I tell you the job execution directory is mounted on `/home/user/job`? Just copy your output there.

As for STDIN and STDOUT, the `apppot-start` script takes care of redirecting them.

A filesystem image is too large?

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 UML instance.

Example: Running GFIT3C/ABC

ABC: Quantum mechanical atom-diatom reactive scattering program.

- Input data (PES) produced by the GFIT3C program

GFIT3C: Global fitting routine for triatomic systems.

- Input data is *compiled in*
- So, need to recompile before each run.
- Old FORTRAN code, needs a specific compiler.

Solution:

- Install ABC + GFIT3C + the G95 compiler in a single VM image.
- `apppot-autorun` script takes an input file and performs the correct GFIT3C+ABC passes.
- Run the image as a single Grid job.

(contact: A. Costantini, U. Perugia)

Limitations

Need to tweak `/proc/sys/vm/max_map_count` to allow UML VMs to use $>$ 256MB of memory.

Updating a base image requires all users to update their local images as well.

Running code in a UML machine comes with a performance penalty (esp. for disk I/O). Precise data yet to come.

Only tested with Debian as a guest OS. No plans to try other OSes as we don't have the manpower for that.

Future work

Support MPI. Two issues:

- UML-to-UML networking backs on IP multicast. Does the cluster network fabric support it?
- The startup script has to extract host allocation details for each and every batch system / MPI library combination.

Any questions?

AppPot home page: http://AppPot.googlecode.com

Source code: `svn co http://AppPot.googlecode.com/svn`

e-mail:
riccardo.murri@gmail.com,
sergio.maffioletti@gc3.uzh.ch

Thank you!

Additional material

Customise the UML instance *(I)*

```
$ apppot-start.sh --apppot ./apppot_a3d.img /bin/bash
...
==== Starting AppPot 0.13 ...
== Setting up AppPot user user(11527) in group users(4200)
usermod: no changes
== Mouting host directory '/home/rzuext/sergio' on local di
== Running command-line '/bin/bash' ...
user@rootstrap:~$
```

Now you've got a *sudo*-enabled promt *in* the UML VM:

- Customize the UML instance
- Install new system components (apt-get)
- Install new specific packages (i.e. MeteoIO_2.0 for the A3D project)
- if necessary, modify the UML configuration

Customise the UML instance *(II)*

Before closing the session, do:

```
user@rootstrap:~$ sudo apppot-snap base
WARNING: Deleting old base state file '/var/lib/apppot/app
user@rootstrap:~$ ls -lrt /var/lib/apppot/apppot-snap.base
-rw-r--r-- 1 root root 389444 May  5 15:33 /var/lib/apppot/
user@rootstrap:~$ exit
exit
==== AppPot done, commencing shutdown ...
INIT: no more processes left in this runlevel
INIT: Switching to runlevel: 0
...
```

Deployed RTE

```
$ cat TEST/A3D_UML
...
export A3D_UML_IMAGE=/share/grid/local_repo/uml/images/apppot_a3d.img
...

$ cat TEST/APPPOT-0
...
export APPPOT_IMAGE=/share/grid/local_repo/apppot/apppot.img
export APPPOT_KERNEL=/share/grid/local_repo/apppot/bin/linux
export APPPOT_STARTUP=/share/grid/local_repo/apppot/bin/apppot-start.sh
export PATH=$PATH:/share/grid/local_repo/apppot/bin
...
```

Related `.xrsl`

```
$ cat A3D_uml.xrsl
&(executable = "$APPPOT_STARTUP" )
(arguments = "--apppot" "a3d.img.cow,$A3D_UML_IMAGE")
(jobname = "A3D_UML" )
(inputFiles =
  ("A3D_source.tgz" \./A3D_source.tgz")
  ("A3D_simulation.tgz" "./A3D_simulation.tgz")
  ("apppot-run"
   "gsiftp://idgc3grid01.uzh.ch/local_repo/A3D/apppot-run-a3d.sh
(stdout = "a3d.stdout" )
(outputFiles = ("results.tgz" ""))
(runtimeenvironment = "TEST/APPPOT-0")
(runtimeenvironment = "TEST/A3D_UML")
(gmlog = ".a3d_log")
(join = yes)
(executables = "apppot-run")
```

Running a job

```
$ ngstat -l gsiftp://idgc3grid01.uzh.ch:2811/jobs/8943130458710019357 2822
Job gsiftp://idgc3grid01.uzh.ch:2811/jobs/8943130458710019357 2822
  Job Name: A3D_UML
  Status: FINISHED
  Owner: /DC=ch/DC=switch/DC=slcs/O=Universitaet Zuerich/CN=Sergio Maffioletti FD0DDA88
  Cluster: idgc3grid01.uzh.ch
  Queue: all.q
  Requested Number of CPUs: 1
  Execution Nodes:
    compute-1-395.local
  stdout: stdout.txt
  stderr: stdout.txt
  Grid Manager Log Directory: .a3d_log
  Submitted: 2011-05-05 11:18:20
  Completed: 2011-05-05 21:23:05
  Submitted from: 130.60.40.14:37810;ocikbgtw.uzh.ch
  Submitting Client: nordugrid-arc-0.8.3.1
  Requested Runtime Environments:
    TEST/APPPOT-0
    TEST/A3D_UML
  Used CPU Time: 10 hours, 2 minutes
  Used Wall Time: 10 hours, 4 minutes
  Used Memory: 5473566 KiB
```

Example: Running GAMESS

Base image contains GAMESS, correctly installed for Debian.

Startup script runs GAMESS on the input files specified by the user.

No visible difference between calling this GAMESS appliance and the real application.

Example: GAMESS development

Base image contains GAMESS sources in `/home/user/gamess`, configured for compilation on Debian. All needed development tools installed.
GAMESS developers can hack the sources, recompile at will and test locally.
Then, they can send the *changes* tarball as a Grid job to try out the new version with some larger data set.

Example: Alpine 3D *(I)*

Specialized base image with a predefined startup script *inside*:

- expects a tarball with the (modified) sources to be available in the sessiondir
- expects simulation data to be available there as well
- compiles the sources and run the simulation with it, reporting on results.

Intended as a base image to be available at Grid sites, for *both:*

- Running a simulation with unmodified Alpine 3D code (for A3D users)
- Running tests of customized A3D code (for developers)

Example: Alpine 3D *(II)*

Usage model:

- Developers commit new code of the Alpine3D to SVN.
- They launch the validation suite:
  1. create tarball from SVN repo (`A3D_source.tgz`)
  2. simulation suites already available on a SE
  3. submit *A3D_uml.xrsl* job, using *A3D* AppPot image available on SMSCG sites
  4. fetch results when job is done