gLExec and OS compatibility

David Groep
Nikhef

Outline

- **What is glExec-on-WN (again)**

- **OS and Batch System Interoperability**
  - Starting and killing jobs
  - Cleaning up files
  - Pruning stray processes

Use Case for '*gLExec on the WN*'

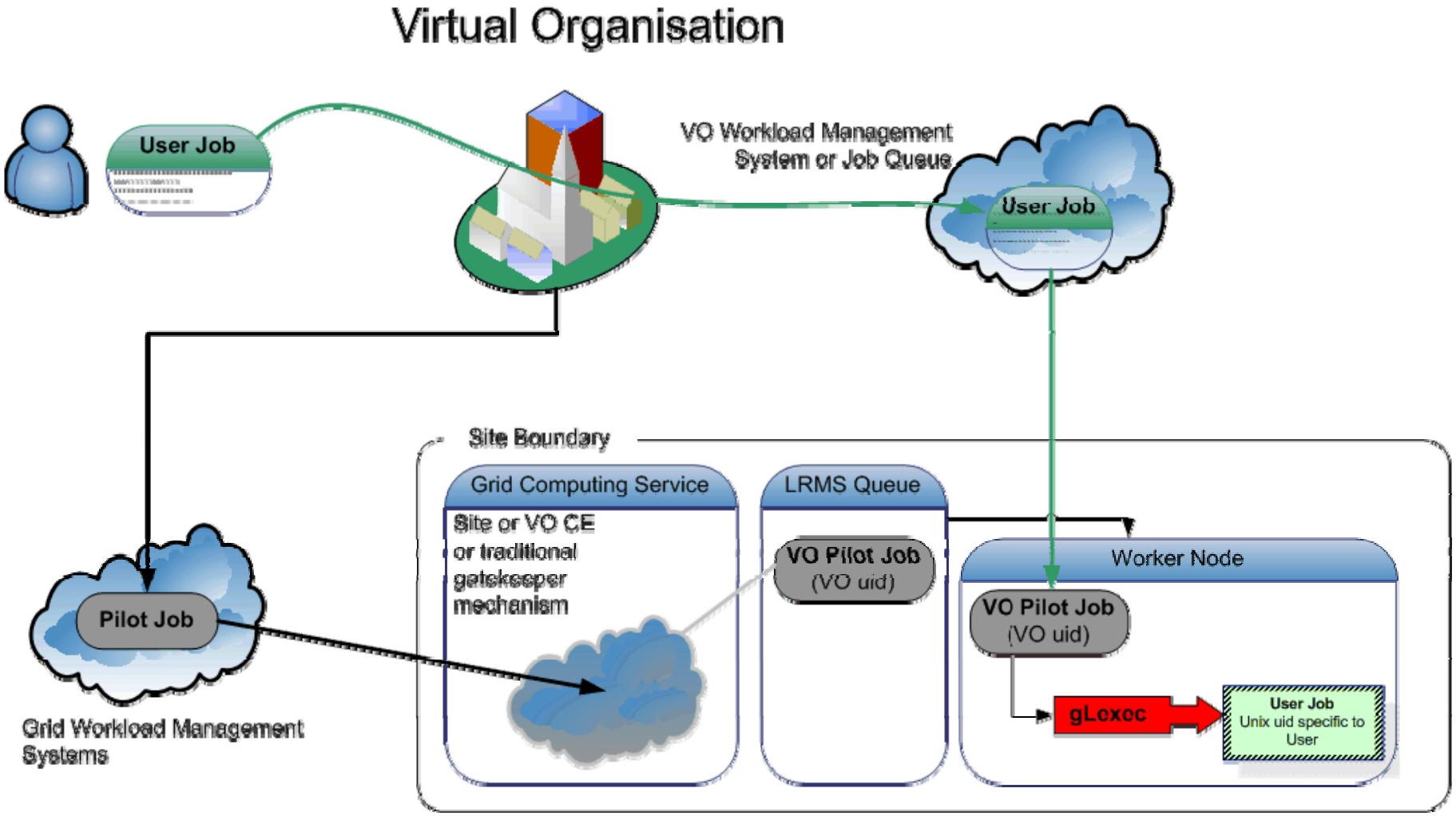
1. Make pilot job subject to normal site policies for jobs

- VO submits a pilot job to the batch system
  - the VO 'pilot job' submitter is responsible for the pilot behaviour

    this might be a specific role in the VO, or a locally registered 'special' user at each site
  - Pilot job obtains the true user job, and presents the user credentials and the job (executable name) to the site (gLExec) to request a decision on a cooperative basis

2. Preventing 'back-manipulation' of the pilot job
   - make sure user workload cannot manipulate the pilot
   - project sensitive data in the pilot environment (proxy!)
   - by changing uid for target workload away from the pilot

Pilot Jobs and gLExec



On success: the site will set the uid/gid to the new user's job

On failure gLExec will return with an error, and pilot job can terminate or obtain other user's job

gLExec deployment modes

- **Identity Mapping Mode – 'just like on the CE'**
  - have the VO query (and by policy honour) all site policies
  - actually change uid based on the true user's grid identity
  - enforce per-user isolation and auditing using uids and gids
  - requires gLExec to have *setuid* capability

- **Non-Privileged Mode – declare only**
  - have the VO query (and by policy honour) all site policies
  - do not actually change uid: no isolation or auditing per user
  - the gLExec invocation will be logged, with the user identity
  - does not require setuid powers – job keeps running in pilot space

- **'Empty Shell' – do nothing but execute the command…**

Identity change

Let's assume you make it *setuid*. Fine. Where to map to:

- **To a shared set of common pool accounts**
  - Uid and gid mapping on CE corresponds to the WN
  - Requires SCAS or shared state (gridmapdir) directory
  - Clear view on who-does-what
- **To a per-WN set of pool accounts**
  - No site-wide configuration needed
  - Only limited (and generic) set of pool uids on the WN
  - Need only as many pool accounts as you have job slots
  - Makes cleanup easier, 'local' to the node
- *Or something in between ... e.g. 1 pool for CE other for WN*

Starting and Killing Jobs

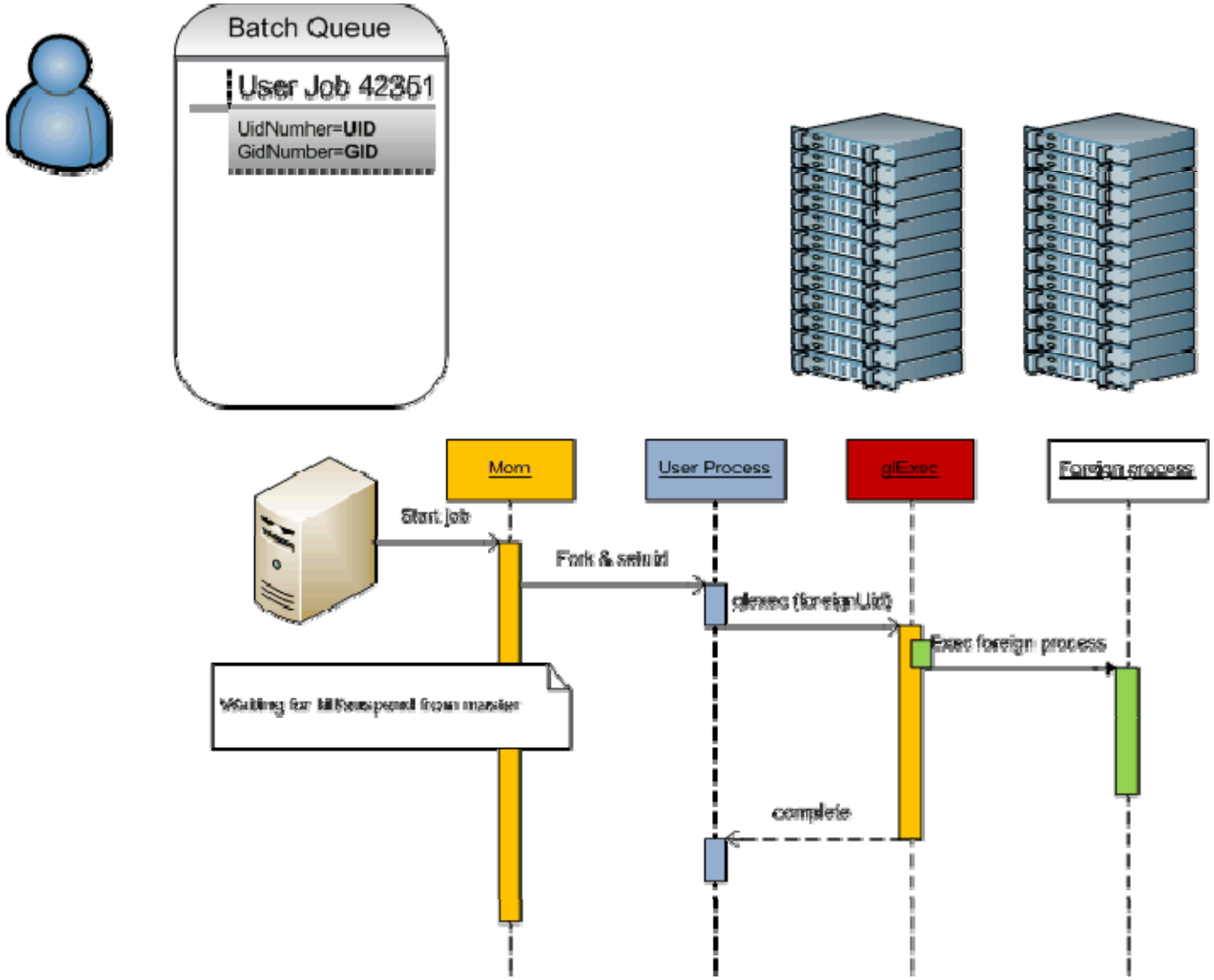
The batch system performs the following basic functions

1. Job Submission
2. Job Suspend/Resume
3. Job Kill
4. CPU time accounting

does not yet address enforcing sanity and user compliance

from the test description of Ulrich Schwickerath

Starting and Killing Jobs



Testing it

Can batch system suspend/kill the gLExec'ed processes?

- **Test using gLExec itself**
  - Most 'true' tests
  - Requires installation of gLExec and all dependencies
- **Test using the *sutest* mini programme**
  - Same logic, but a stand-alone small (50-line) C programme
  - No dependencies
  - A few hard-coded constants to be set before compilation
  - Trivial to test

Session Preservation

```
$ date && ps --forest -eo pid,ppid,sess,user,uid,euid,cmd

Fri Feb  8 14:02:41 CET 2008

  PID  PPID  SESS USER   UID UID CMD
3294     1  3294 root       0   0 /usr/sbin/pbs_mom
18668  3294 18668 davidg 502 502  \_ -bash
18713 18668 18668 davidg 502 502      \_ /bin/sh …/jobs/33.tbn05.ni.SC
18715 18713 18668 nobody  99  99          \_ /project/sutest /bin/sleep 120
18716 18715 18668 nobody  99  99              \_ /bin/sleep 120
...
```

... and Torque will kill all processes in the tree:

```
tbn05:~:1018$ cat tmp/tt.pbs
#! /bin/sh
date
/project/sutest /bin/sleep 120
date
```

```
$ date && qsub -q test tmp/tt.pbs
Fri Feb  8 14:02:21 CET 2008
33.tbn05.nikhef.nl
```

All vanishes after a 'qdel 33' ...

CPU accounting

- **No change with respect to current behaviour of jobs**
- **Times are accumulated on wait and collated with the gLExec usage**

Forcing havoc on yourself

```
$ ( date && ./sutest /bin/sleep 60 && date )
Fri Feb  8 16:41:24 CET 2008
Notice: identity changed to uid 99
```

```
...
 7508     1  7508 root     0 /usr/sbin/sshd
32122  7508 32122 root     0  \_ sshd: davidg [priv]
32124 32122 32122 davidg 502  |    \_ sshd: davidg@pts/0
32126 32124 32126 davidg 502  |        \_ -bash
17001 32126 32126 davidg 502  |            \_ -bash
17003 17001 32126 nobody  99  |                \_ ./sutest /bin/sleep 60
17004 17003 32126 nobody  99  |                    \_ /bin/sleep 60

# kill -9 17001
```

```
Killed
```

```
17003     1 32126 nobody       99    99 ./sutest /bin/sleep 60
17004 17003 32126 nobody       99    99  \_ /bin/sleep 60
```

Cleaning up files

File cleanup: what do sites use today?

- Check for files owner by users not currently running a job?
  - Who ‘is running’ becomes ill defined
  - Need a ‘back-mapping’ tool that can trawl log files or a state dir
- tmpwatch(8) for old files?
  - Change of uid does in influence this solution
- Transient TMPDIR facilities (PBSPro, Torque 2+)?
  - Runs with root privileges anyway
  - TMPDIR is inherited by the gLExec’ed child
  - And is thus unaffected by gLExec

Pruning stray processes

- **Killing stray user processes**
  'not owned by a user with a currently running process'
  - E.g. used at CERN/LSF
  - Need a 'back-mapping' tool that can trawl log files or a state dir
  - But: *is not trustworthy to begin with on multi-job-slot machines*!

- Kill processes that are 'too old'
  - Will run as root anyway
  - Unaffected, but is not trustworthy either

- Kill processes not 'parented' in a batch job
  - gLExec will preserve the process tree, and thus this will work
  - Will also slaughter daemonizing jobs today
    … which is a Good Thing™

Pruning User Processes

https://www.nikhef.nl/grid/sysutils/prune_users/

- For Torque in perl (simple migration to other systems)
- Kill processes that are not a child of a registered pbs_mom
- Uses the *momctl* command on the node

- Caveats
  - Will usually not kill processes with a uid < 99
  - May optionally preserve top-level sshd sessions (beware of MPI)
  - Does not protect against fork bombs

Where are we now?

You can deploy today if

- You run LSF or Torque and
  don't manage disk or processes
- You run LSF or Torque and
  use TMPDIR and prone_userproc style job slaughtering

You should wait for back-mapping tool (+update your script) if

- You use LSF or Torque and use uid recognition for pruning stray processes (but you ought to change this anyway)
- You use uid recognition for file cleaning

Back-mapping tool is expected to be out of development
in XXX weeks

Summary and References

References

- **https://www.nikhef.nl/grid/lcaslcmaps/glexec**
  - sutest program: https://www.nikhef.nl/grid/lcaslcmaps/glexec/osinterop
- **https://twiki.cern.ch/twiki/bin/view/FIOgroup/FsLSFGridglExec**