gLExec

gluing grid jobs to the Unix world

… of job submission, #'s, pilot jobs and traceability …

David Groep, Gerben Venekamp, Oscar Koeroo
Nikhef

The Problem

- translation is needed between grid identity and local identity
- this translation has to happen somewhere
- something needs to do that

What is gLExec

gLExec

a thin layer
to change Unix domain credentials
based on grid identity and attribute information

you can think of it as:

- **'a replacement for the gatekeeper'**
- **'a *griddy* version of Apache's `suexec`'**
- **'a program wrapper around LCAS, LCMAPS or GUMS'**

Why was glexec devised?

1. **Gatekeepers and schedulers are complex: why run with super-user privileges all the time?**
   - like apache's httpd, where user *cgi* scripts may run as user, but without the web server itself having to run as root!
   - to accomplish this a small program is needed with *setuid* power to change *uid*: 'suexec'

   gLExec is the 'griddy' suexec clone

2. **Variety of grid job submission systems is increasing**
   - need a common way of enforcing site policy and id mapping
   - without the need to modify each and every system
   - gLExec can be used as an alternative to having authorization and mapping *call-outs* in each system

Job Submission Models

There are several 'traditional' job submission models, where glexec has a role in two of these

1. direct per-user job submission to a 'gatekeeper' running with root privileges
2. a CE or scheduler not running as the super user

Site-manager controlled, running with 'generic' uid

Real User Job
(with *uid* matching actual workload being run)

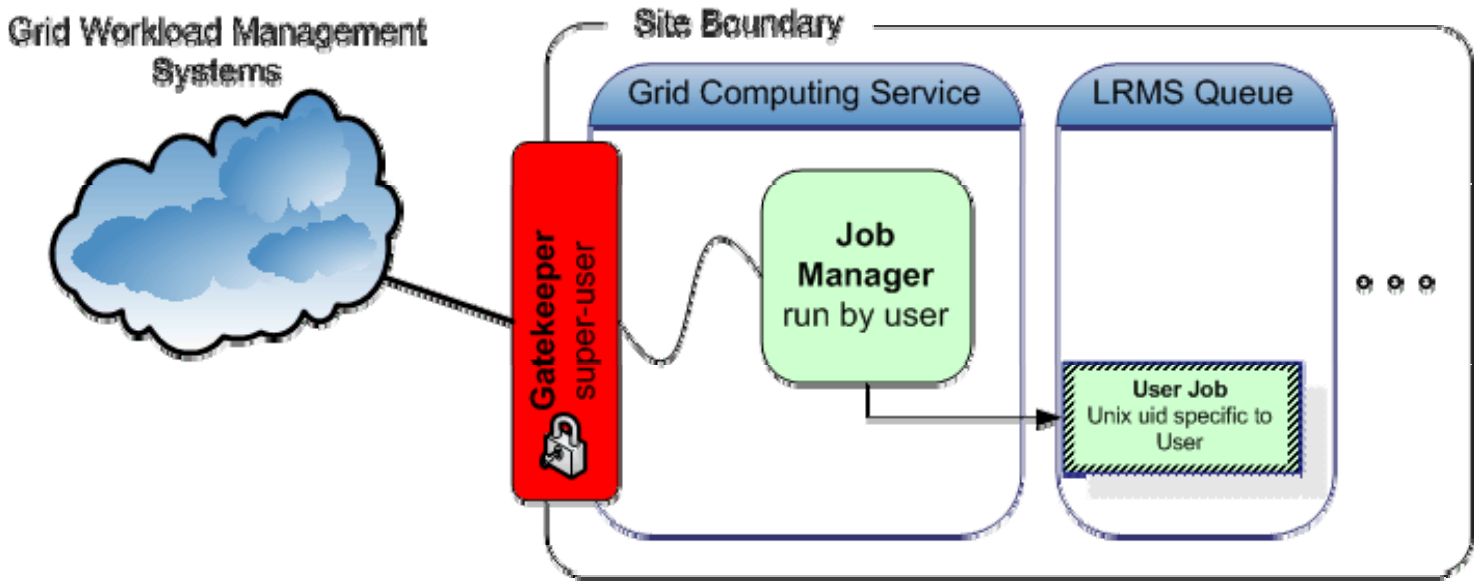
Site-manager controlled, running as super-user

or a super-user daemon

VO-run process
(potentially generic VO *uid*, or generic VO pool)

Submission via a Gatekeeper

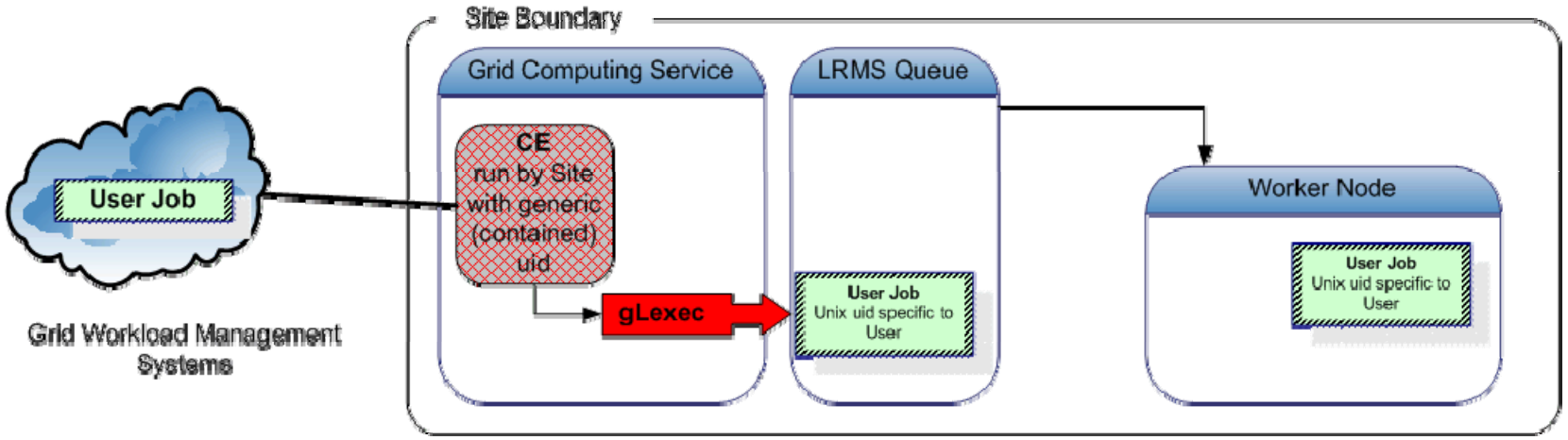
Traditional job submission scenario, model ‘gatekeeper’



- change of credentials at the site edge
- networked service (‘gatekeeper’) with super-user privileges
- job management in a per-user account (be it for single or multiple jobs)

Grid Computing Service: Site-CE

- **Deployment model with a CE 'service'**
  - running in a non-privileged account or
  - with a CE run (maybe one per VO) on a single front-end per site



examples
- CREAM
- GT4 WS-GRAM (via sudo)

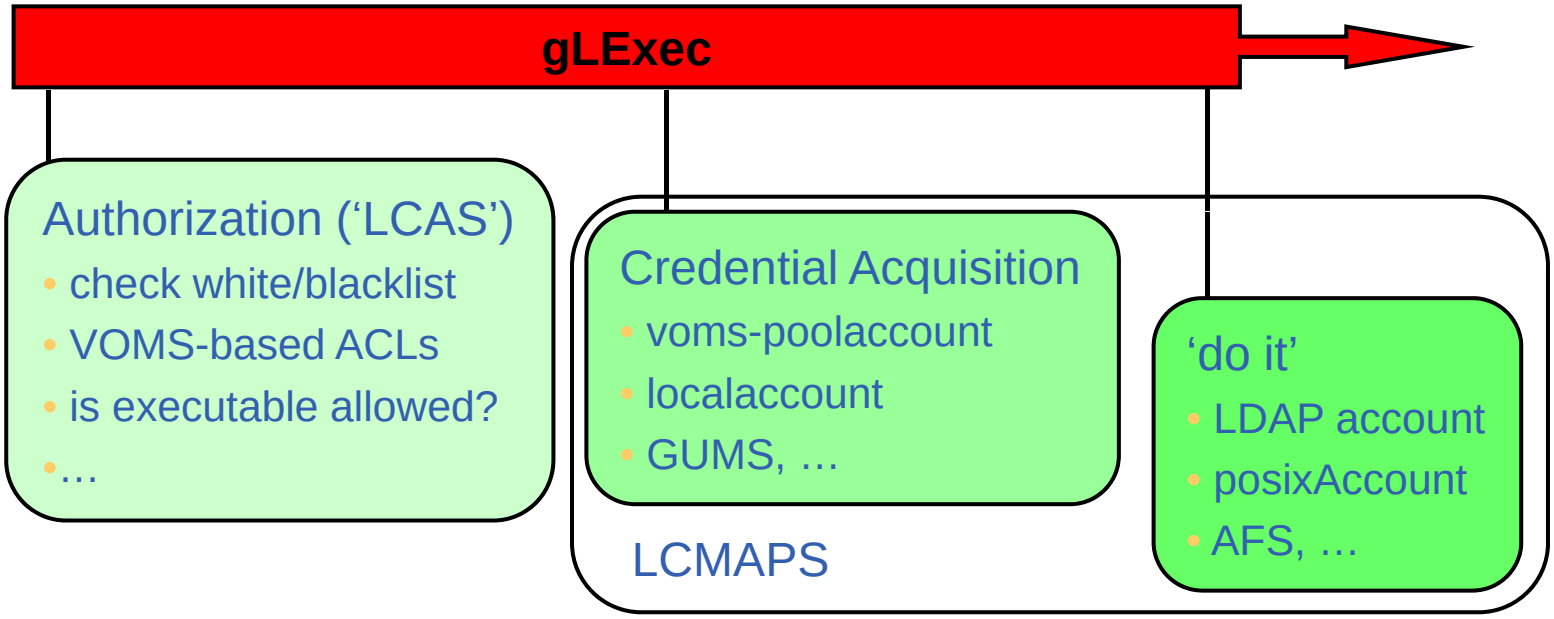
Characteristics of the traditional models

- In all these models, the submission of the user job to the batch system is done with the *original job owner's* mapped (uid, gid) identity
- grid-to-local identity mapping is done *only* on the front-end system (CE)
  - batch system accounting provides per-user records
  - inspection shows Unix process on worker nodes and in batch queue per-user

What gLExec does …

- **User grid credential** (subject name, VOMS, …)
- **command to execute**
- ***current uid allowed to execute gLExec***

cryptographically protected by CA or VO AA certificate



- **Execute command with arguments**
- **as user (*uid, pgid, sgids …)***

Pilot jobs: ‘late binding’

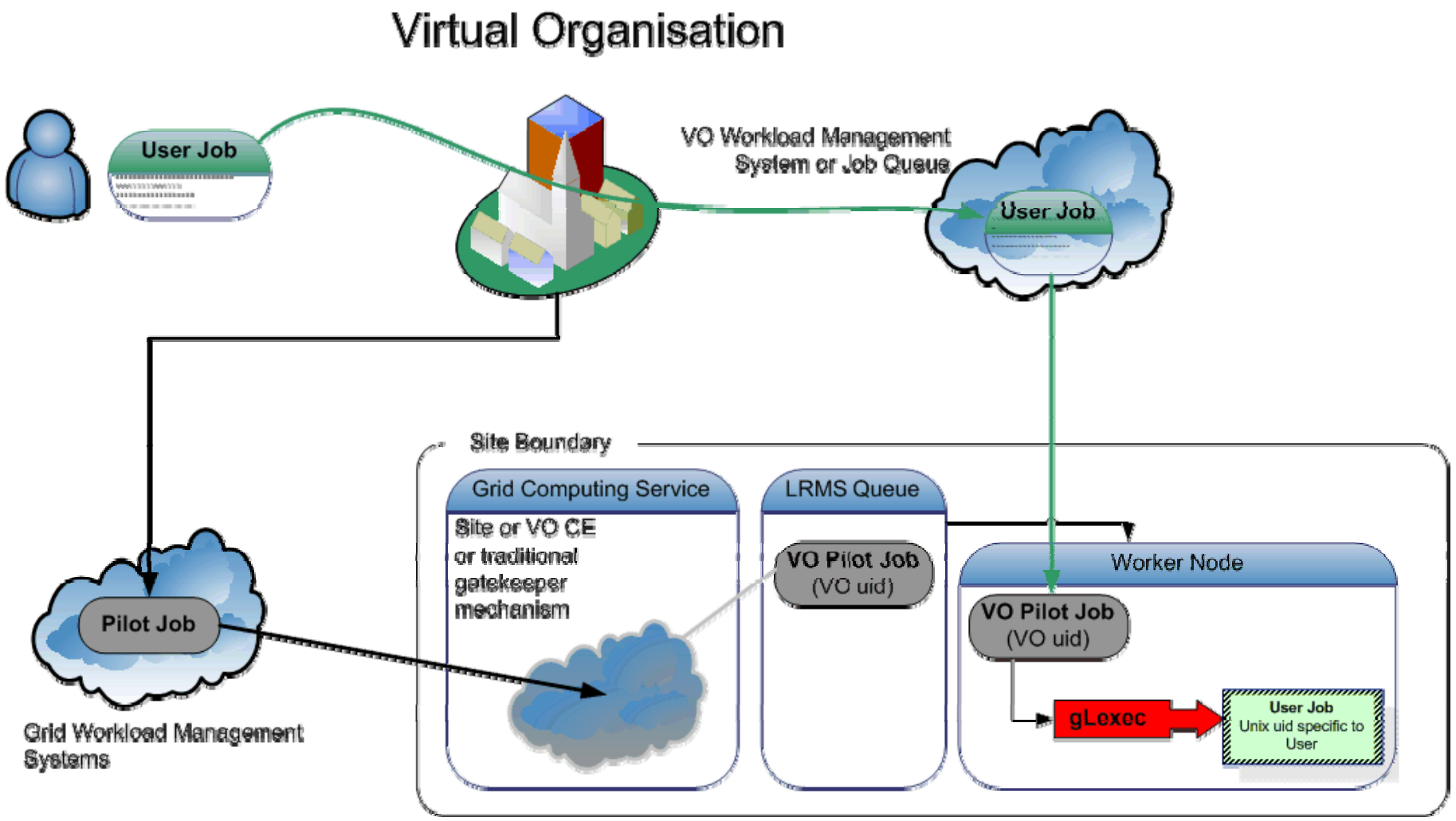
But job submission gets more and more intricate …

- Late binding of jobs to job slots via *pilot jobs*
  ‘some larger user communities develop and prefer to use proprietary scheduling & job management’
  - pilot is a small placeholder that downloads a real job
  - it is not committed to any particular task, or perhaps even a particular user (‘VO pilot’), until that point
  - ‘first establishing an overlay network
  - subsequent scheduling and starting of jobs is faster’
- this scheduling is orthogonal to the site-provided systems

Situation today

- **'VO-type' pilot jobs submitted as if regular user jobs**
  - run with the identity of one or a few individuals from a VO
  - obtain jobs from any user (within the VO) and run that payload on the WN allocated
  - no effective mechanisms today can deny this use model
  - site 'sees' only a single identity, not the true owner of the workload
- **regular 'per-user' pilot jobs have no such issues**
  - user-specific pilot glided in, binding to the own user's workload

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 on WN deployment model

- 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 ‘badged’ user at each site

- Pilot job is subject to normal site policies for jobs

- 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

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
- **Site-Isolation Mode – protect only**
  - make setuid to a single 'nobody' user
  - no per-user auditing, but well separated from pilot (or container)
- **'Empty Shell' – do nothing but execute the command…**

Pieces of the solution

VO supplied pilot jobs must observe and honour

the same policies the site uses for normal job execution

(e.g. banned individual users)

Three pieces that go together:

- **glexec on the worker-node deployment**
  - mechanism for pilot job to submit themselves and their payload to site policy control
  - give ‘incontrovertible’ evidence of who is running on which node at any one time (in mapping mode)
    - at some sites for regulatory compliance (remember Igor’s talk)
    - ability to nail individual culprits
    - by requiring the VO to present a valid delegation from each user
  - VO should want this
    - to keep user jobs from interfering with each other
    - honouring site ban lists for individuals may help in not banning the entire VO in case of an incident

Pieces of the solution

- glexec on the worker-node deployment

- **way to keep the pilot jobs submitters to their word**
  - mainly: monitor for compromised pilot submitters credentials
  - process and system call level auditing of the pilot jobs
  - logging and log analysis

- **‘internal accounting should be done by the VO’**
  - the regular site accounting mechanisms are via the batch system, and these will see the pilot job identity
  - the site can easily show from those logs the usage by the pilot job

  - auditing data on the WN is useful for incident investigations only
  - making a site do accounting based glexec jobs requires a large and unknown effort

Status today

- **Status of ‘glexec’ today**
  - implementation ready & tested, deployed in production at FNAL
  - uses the LCAS and LCMAPS for mapping and enforcement both in their library-based implementation
  - extensive logging via syslog
  - new modules have been added
    - LCAS: RSL (executable path) constraints
    - validation of cert chain and proxy lifetime
  - restrictions
    - policy should be located on local POSIX-style file systems
    - policy transport should be ‘trustworthy’ (but is within the site)
    - gLExec executable restrictions to specific users only is today via Unix permissions only

gLExec future directions

- **gLExec, LCAS, LCMAPS improvements planned …**
  … especially nice for the '–on-WN' model
  - make the credential acquisition process (LCAS/LCMAPS) work with a *site-central policy engine*
    - actual credential application will have to stay local
  - changeover to standard callouts for both LCAS and LCMAPS
    - interoperation between LCAS/LCMAPS and GUMS servers
  - add site configuration capabilities

Needed components, procedures

- **Auditing the VO placeholder job/scheduler on the WN**
  - check number of ‘fork-execs’ done by the placeholder with the number of glexec invocations
    a discrepancy means the VO is cheating on you
  - check the VO placeholder job is not using too much CPU
    the CPU-time / Walltime should be close to zero

- **credential mapping auditing/logging**
  - ‘JobRepository’ fits the bill
    - *schema allows for recording and retrieving all aspects of credential mapping*
    - *records both user identity and any VO attributes*
    - *retains the credential mapping for each ‘job’ or glexec invocation*
  - JR is part of the stack, but not widely deployed yet

Towards end-to-end trust

Issue:

- **gLExec trusts submitter to match credentials to jobs**
  - like any site-managed ingress point trusts resource brokers today do this correctly
  - also RBs are unknown quantities to the receiving site

- **longer term solution: jobs signed by submitting user**
  - but today …
    … job description is modified by intermediaries (brokers)
  - but signature is on original content …
    … site has to evaluate if job received matches the signed JDL
  - Use an inheritance model for the job description?

Summary

- gLExec part of the ‘modular job submission’ scenarios
  - less code runs as the super-user
  - does the implicit mapping needed for most submissions
- gLExec-on-WN gives VO tools to comply with site policies
  - Realize that today some VOs are doing ‘pilot’ jobs today
  - some sites may even just don’t care yet, whilst others have hard requirements on auditability and regulatory compliance
  - but you, as a site, will miss that warm and fuzzy feeling of trust
- gLExec-on-WN is always replaceable
  there are 4 deployment models to choose from
- but gLExec-on-WN is for just one of the scenarios
  - it is still needed for the Site-CE scenarios