Stealth Cloud: How not to waste CPU during grid to cloud transitions

Daniela Bauer & Simon Fayer

Imperial College London

The Problem

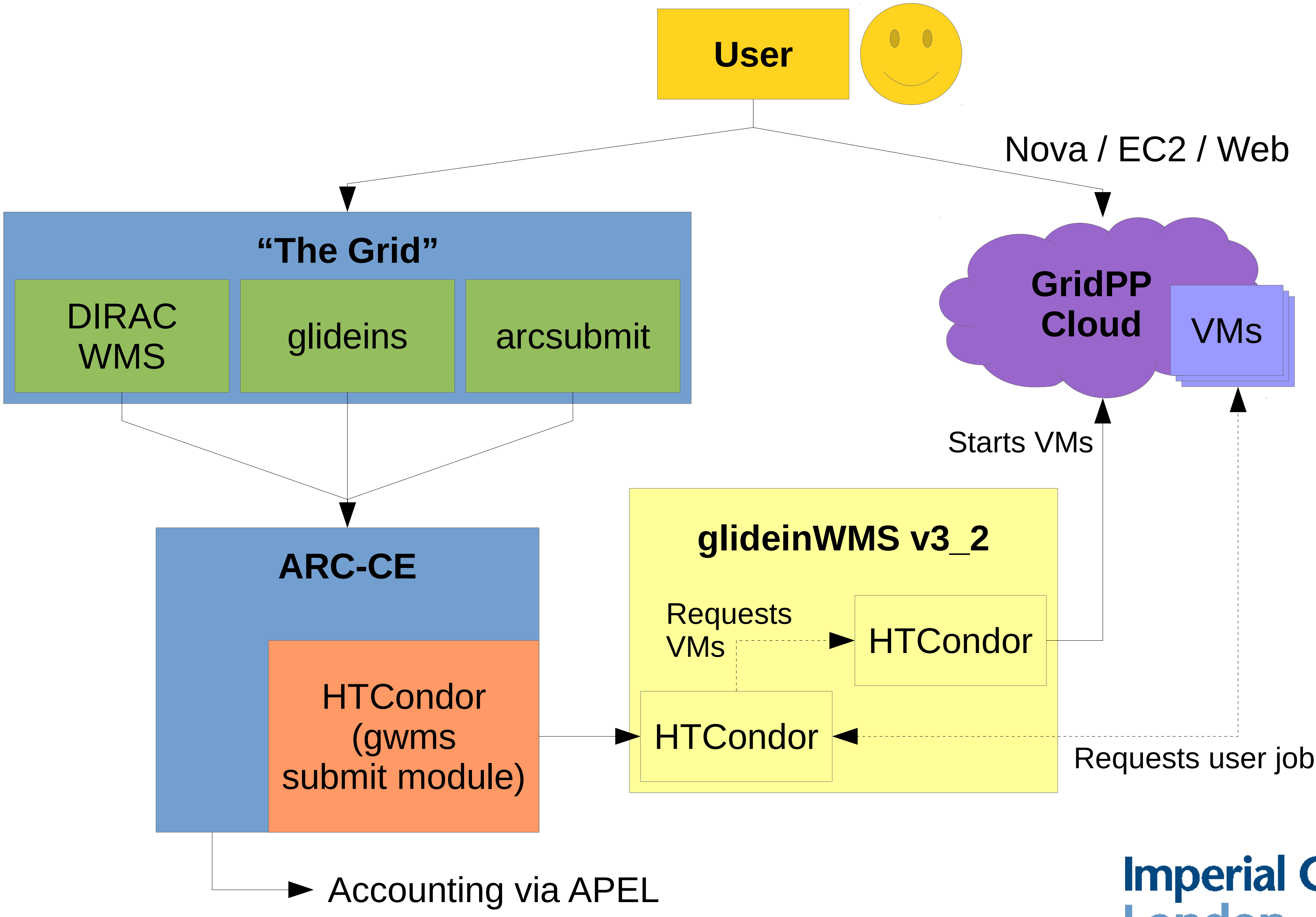
- In 2013 when we first started looking at clouds there was little expertise by users and admins on how to best make use of them
- In order to solicit interest, our cloud had to be a reasonable size...
- …but we couldn't afford to have too many compute nodes sitting idle, especially with work submitted via various grid channels waiting

Scenario

- User wants to run a Grid job
  - Submits using any usual mechanism to “UKI-LT2-IC-HEP”
  - Job either runs on local batch system or a cloud WN depending on matched CE
    - Completely transparent; user happy
- User actually needs a cloud
  - They either have an image to boot or want to run a service
  - Direct submission to a specific cloud using the usual cloud interfaces

Imperial College London

The Setup



Imperial College London

Cloud Configurations

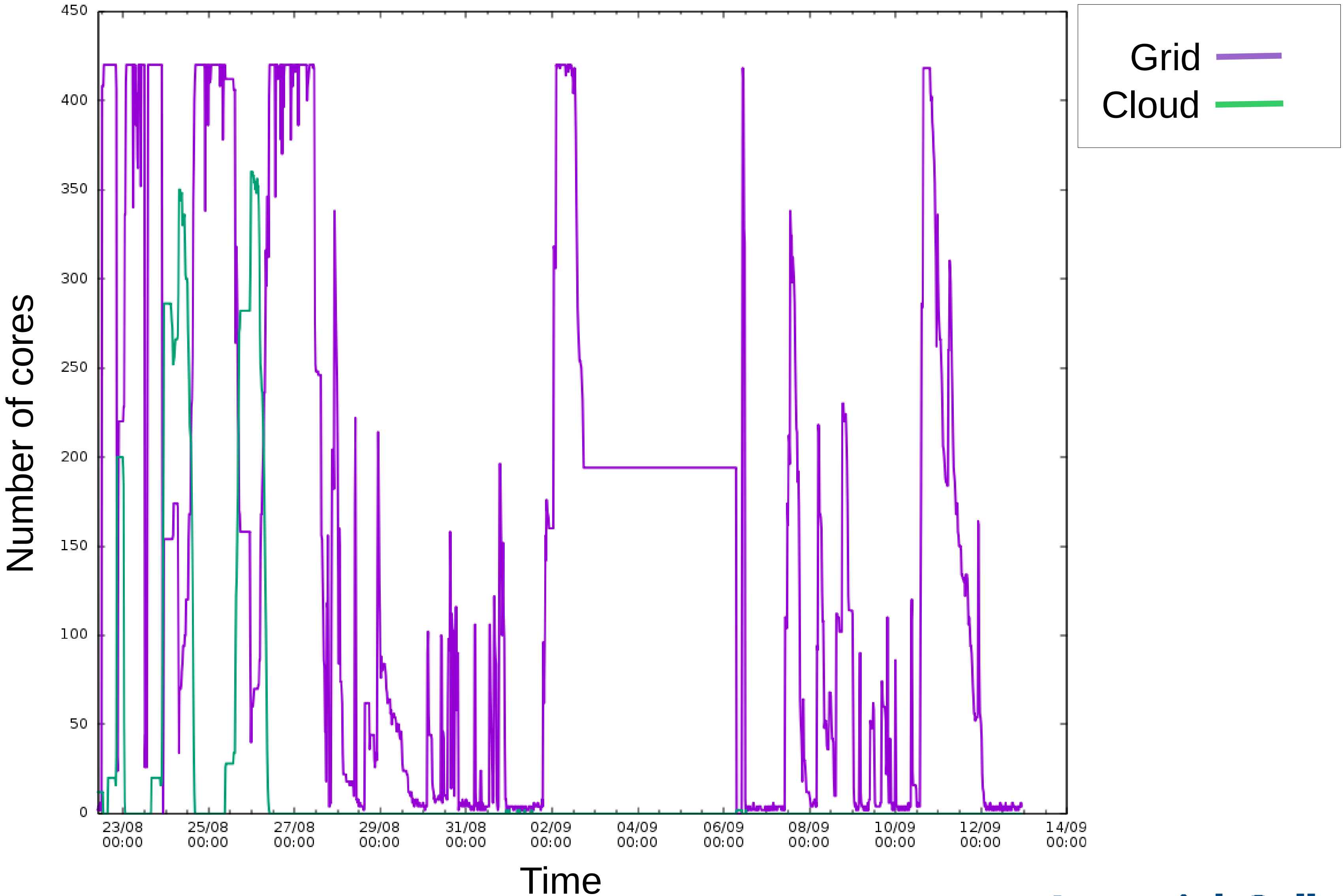
- Tested with multiple OpenStack versions / configurations:
  - Icehouse & Juno
    - In-built EC2 interface
    - Gluster glance backend
    - Nova networking initially → Migrated to Neutron as part of the Juno upgrade
  - Kilo
    - External EC2 with VPC patch
    - CEPH glance backend
    - Neutron networking
  - Mitaka (for testing only)
    - Unpatched external EC2 interface
    - Nova networking
    - Local/POSIX glance backend

Payload Image

- Images are built using CloudStamp[1]:
  - Kickstart based RHEL installation
  - One-shot puppet application
    - Installs base WN / GlideinWMS bootstrap
    - Configures users, start-up scripts, etc...
  - Image compression
- Images are bundled as compressed QCOW2 and uploaded to OpenStack
- On boot, GlideinWMS bootstrap processes instance user-data and pulls job
- Once job has completed, VM waits 10 mins (blackhole WN prevention) and halts itself

[1] https://github.com/sfayer/cloudstamp

Grid vs Cloud usage



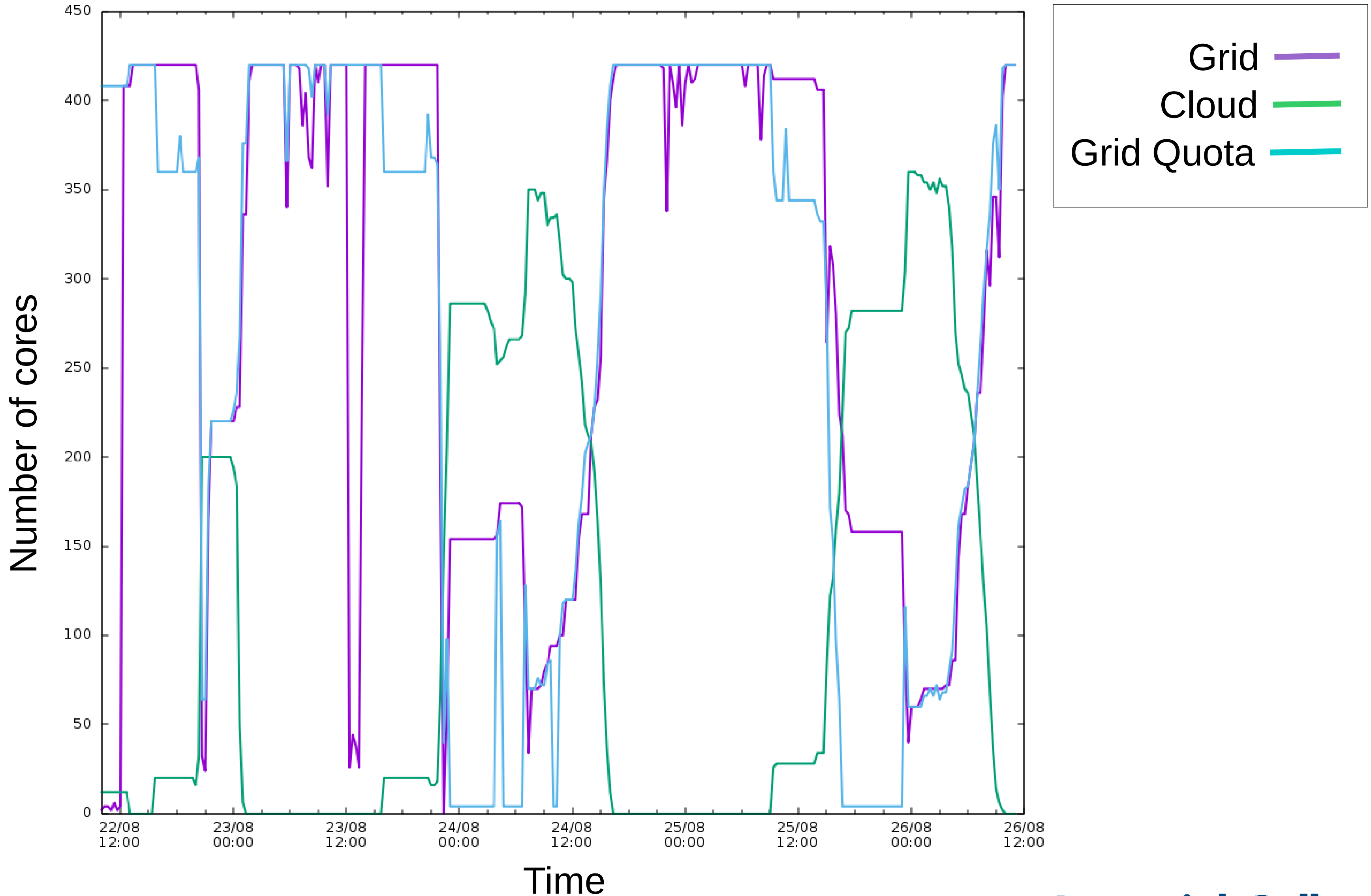
Imperial College London

No queuing in OpenStack

- New VM requests to OpenStack fail if resources aren't available
  - Very unlike a traditional batch system
- Partition the cloud into “grid” and “cloud” tenants
  - The grid tenant must be dynamically expanded to use up the free space
    - If set too large, cloud work will never get to run
    - If too small, too many slots are left idle
- A python script manages the available tenant quota
  - Leaves a small window (currently ~10 slots) idle for “real” cloud workloads
  - Scales grid quota down more if window is full and vice versa

Imperial College London

Making room for the cloud



Imperial College London

Problems

- OpenStack needs an expert
  - Generating a constant stream of VM requests from the grid side helped find a large number of deployment errors:
    - E.g. size of database hard to predict
- GlideinWMS makes extensive use of SSH keys
  - Older versions lost track of keys and eventually exceeded tenant quota
- gLexec CMS test warnings
  - Expects different mappings based on VOMS role, current VMs only match by DN
- Regular rebuilding of images
  - Images contain trust anchors which need updating quite frequently → Maybe switch to CVMFS for this?

Imperial College London

The Future

- This has been working nicely (for years!), but...
  - Lacks monitoring
  - No Multi-core job support
  - Longer lived VMs would be more efficient (Requires Machine Job Features (MJF)?)
- Switch to newer technologies
  - vcycle (needs to be specifically supported by experiments)
  - HTCondorCE
  - VMDirac (part of GridPP DIRAC server)