GridPP / DIRAC / Cloud

Andrew McNab
University of Manchester
GridPP and LHCb

Overview

DIRAC in the title is NOT dirac.ac.uk!

- High Throughput Computing in HEP
- The Grid and Pilot Frameworks
- DIRAC
- GridPP DIRAC and the Vacuum Model
- Vcycle and Vac
- VM architectures
- HTC vs HPC
- How DIRAC does production requests
- Some questions and ideas

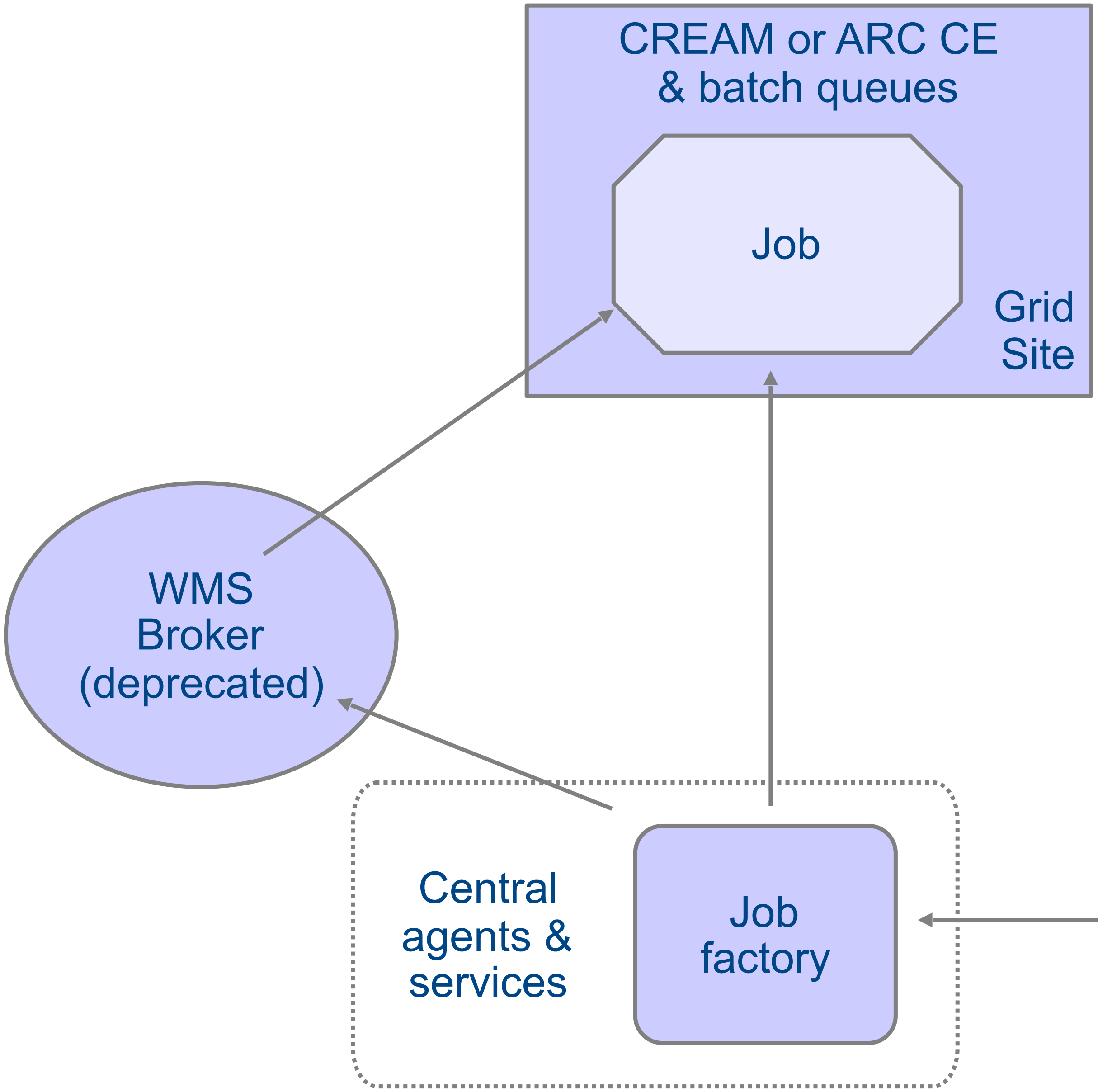
HEP’s High Throughput Computing model

- We have a series of pieces of computation (“jobs”) that need doing
  - From more than one person/role in the experiment collaboration
  - With different priorities
  - With different requirements
- Only very loose connections between jobs
  - “Do 9,000 Stage1 jobs; do 5,000 Stage2 jobs with Stage1 output files; do 8,000 Stage3 jobs with Stage2 outputs”
- So we don’t need “coherence”: work doesn’t arrive on the fabric as a wavefront, spread across it, and then finish at the same time.
  - We can run jobs from different workflows next to each other, all over the place, mostly in any order
  - But we have to put a lot of effort into book keeping!

Last 15 years: The Grid

- (Here “The Grid” is the infrastructure of European Grid Infrastructure (EGI) and Worldwide LHC Computing Grid (WLCG))
- The Grid gives access to remote batch systems in a scalable fashion
  - Uses X.509 and VOMS certificates for identifying users and services and for virtual organization (VO = experiment) membership
  - An information system to discover suitable resources
  - Lots of operational support (tickets, mailing lists, monitoring)
- Makes it possible to automate the submission of jobs
  - Originally via a Broker or Workload Management System
  - Now via jobs submitted as part of a pilot framework

The Grid



Underlies the infrastructure run by EGI/WLCG

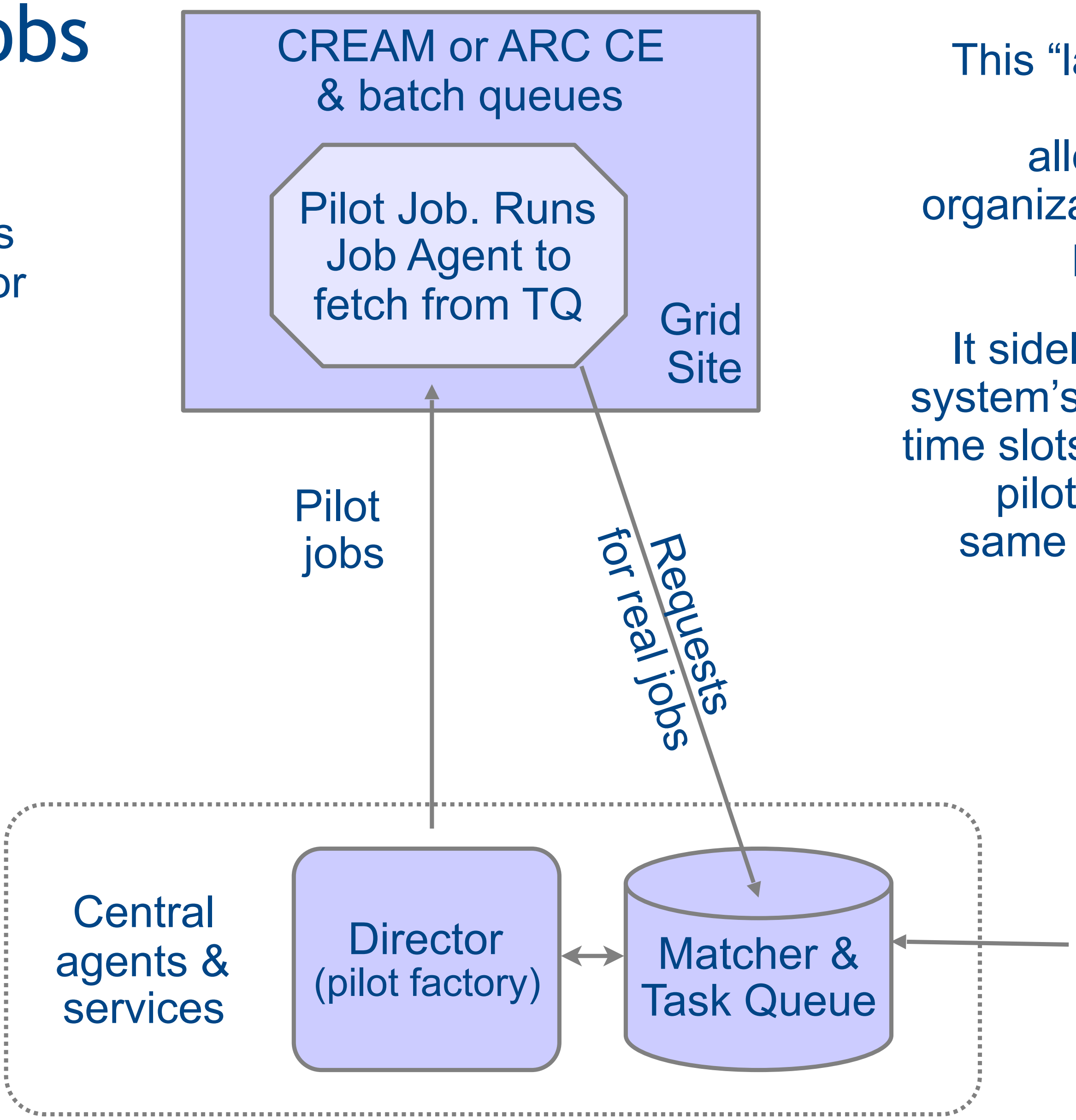
Originally proposed by EU DataGrid in 2000

In practice, we found this “central planning” or “push model” to be fragile, and bad at dealing with distributed infrastructure at 10s or 100s of sites.

The Grid with Pilot Jobs

The Grid + pilot jobs is the dominant model for running HEP jobs.

Well established, and gives access to resources around the world.



This "late binding" or "pull" model allows the virtual organization to decide priorities itself.

It sidelines the batch system's scheduling of time slots: each type of pilot job are all the same irrespective of the contents.

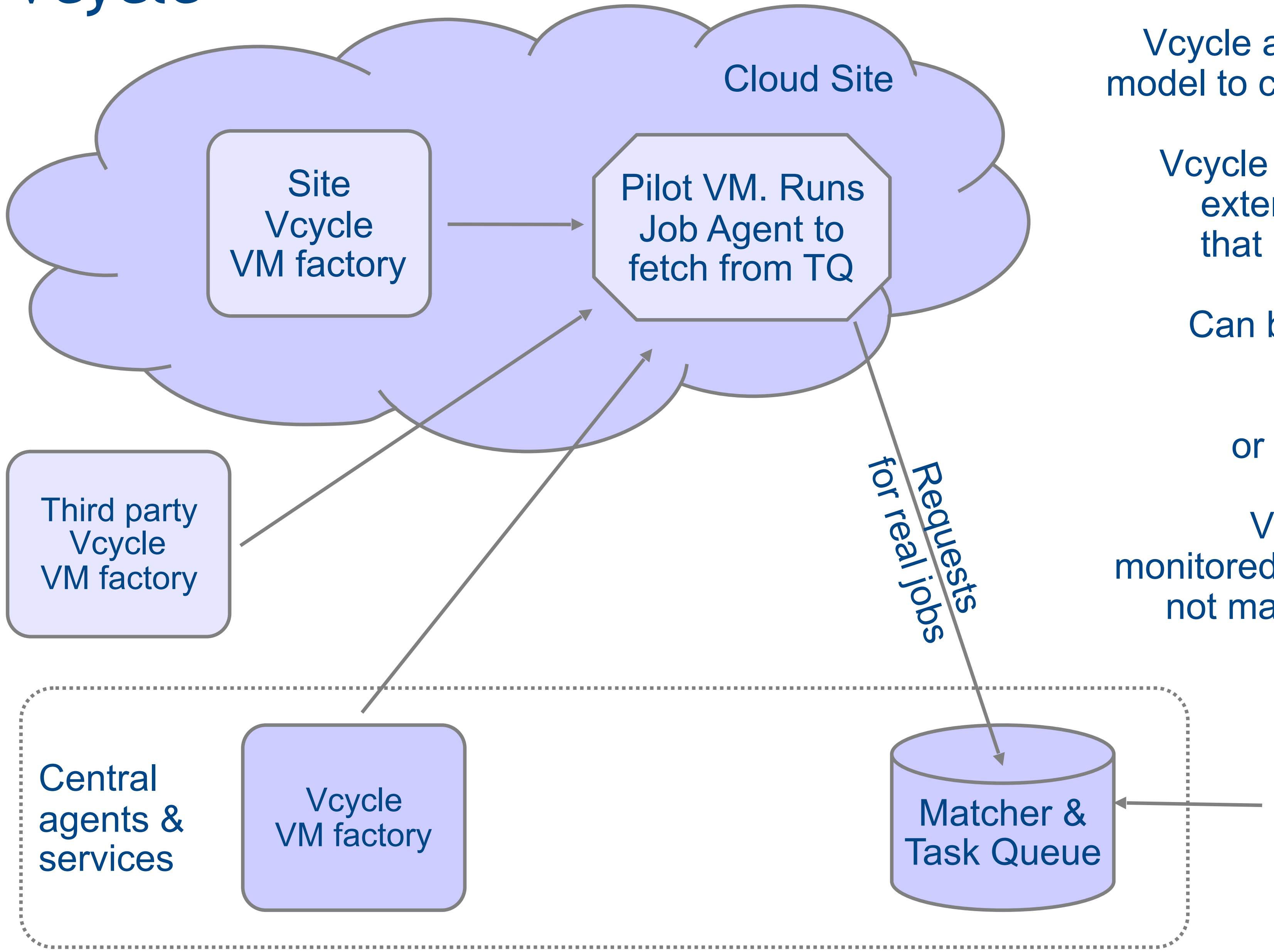
Pilot frameworks: DIRAC

- Pilot job concept was introduced by LHCb's DIRAC project in 2003
  - DIRAC now spun-off as a separate project and used by other collaborations, and provided by infrastructures like EGI and GridPP
  - GridPP DIRAC service is intended to support experiments and VOs without their own pilot framework
  - ATLAS and CMS also use pilot frameworks, and have done some common work on using ATLAS's BigPanda
- Pilot framework becomes a higher level batch system
  - The virtual organization can monitor and manage all of the resources available to it centrally
- **Can be used to hide the details of the underlying batch system, or even whether jobs are running inside VMs at cloud sites**
  - Users don't need to know. Or care.

GridPP DIRAC in VMs: Vacuum model

- We've extended the GridPP DIRAC service on to cloud and other VM-based systems using the "Vacuum Model"
- This applies the late-binding and pull ideas of DIRAC to VM management
  - Instead of the virtual organisation centrally creating VMs, the resource providers create them using provided VM definition
  - If they find work to do, they run; otherwise they shutdown
  - More VMs of successful types created
- This gives resource providers control of what runs on their system
- Means that the same VM definitions can be used on different cloud platforms, sites etc.
- APIs for all this defined in an HEP Software Foundation technical note
- We also have LHCb and ATLAS VMs in production using this model, and demonstrated it with CMS

Vcycle


Cloud Site
Site Vcycle VM factory
Pilot VM. Runs Job Agent to fetch from TQ
Requests for real jobs
Third party Vcycle VM factory
Central agents & services
Vcycle VM factory
Matcher & Task Queue

User and production jobs

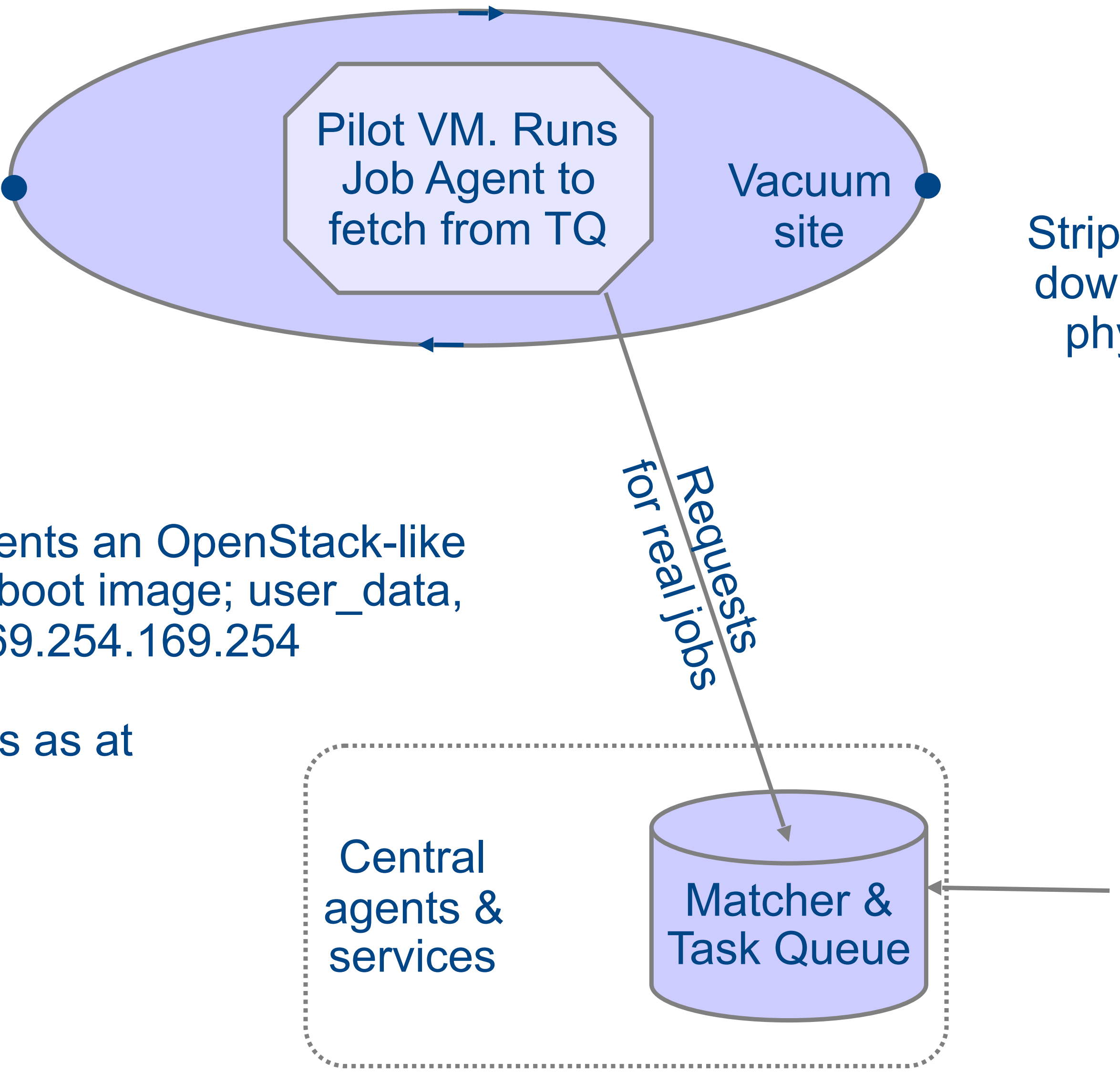
Vcycle applies vacuum model to cloud resources

Vcycle implements an external VM factory that manages VMs.

Can be run centrally by experiment or by site itself or by a third party

VMs started and monitored by Vcycle, but not managed in detail ("black boxes")

Vac - bare metal vacuum



Vac applies vacuum model to independent hypervisors

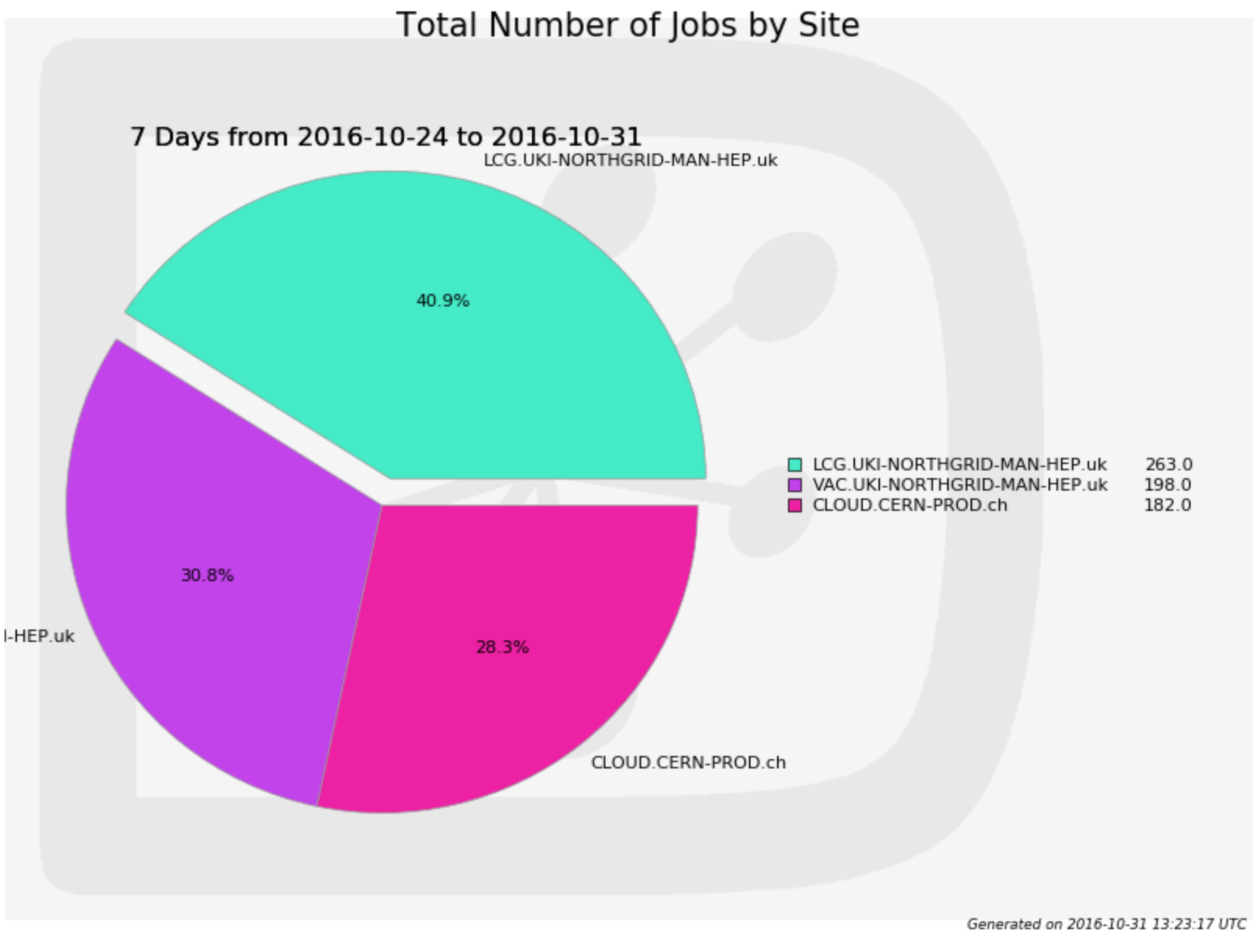
Strip the system right down and have each physical host at the site create the VMs itself.

Each hypervisor presents an OpenStack-like environment to VMs: boot image; user_data, meta data on http://169.254.169.254

So can use same VMs as at cloud sites.

Example with skatelescope.eu VO

- HelloWorld jobs run via GridPP DIRAC in skatelescope.eu VO
  - (Details tomorrow)
- All the jobs submitted to DIRAC are the same
- Same VM running on Vac at Manchester and Vcycle/OpenStack at CERN
- Transparent to the user, whether jobs run on “LCG” (=Torque/CREAM) or the two VM sites



🟩	LCG.UKI-NORTHGRID-MAN-HEP.uk	263.0
🟪	VAC.UKI-NORTHGRID-MAN-HEP.uk	198.0
🟥	CLOUD.CERN-PROD.ch	182.0

Virtual grid sites

- Cloud resources also being used in two other ways in HEP for HTC:
- Most popular (eg at CERN):
  - Lots of static VMs created, e.g. on OpenStack
  - Run a conventional batch system on top
  - May be easier to manage than before, but not elastic
- “Use cloud API like grid API” (eg CloudScheduler)
  - Centrally create VMs at remote site
  - VMs then join a batch system run from elsewhere
  - Elasticity, but currently requires giving a fixed sized tenancy to the virtual organisation at each site
  - (Quite complicated too)

VM architecture: CernVM

- HEP experiments are standardising on CernVM image as the basis for their VMs
- This is a small (20MB) boot image that works on OpenStack, Amazon EC2, Google GCE, MS Azure, VMware, VirtualBox, kvm, Xen, …
- Root filesystem is taken from CernVM-FS, a wide-area read-only filesystem, on demand, and with copy-on-write for any files you change locally
    - Looks like Scientific Linux 6 (or 7 in the next version)
    - Security updates done centrally. This is a major benefit for us.
- We also use CernVM-FS for experiment software
    - All backed by a hierarchy of Squid caches
    - Strong versioning so updates appear coherently
- Available on our conventional batch worker nodes too

HTC vs HPC

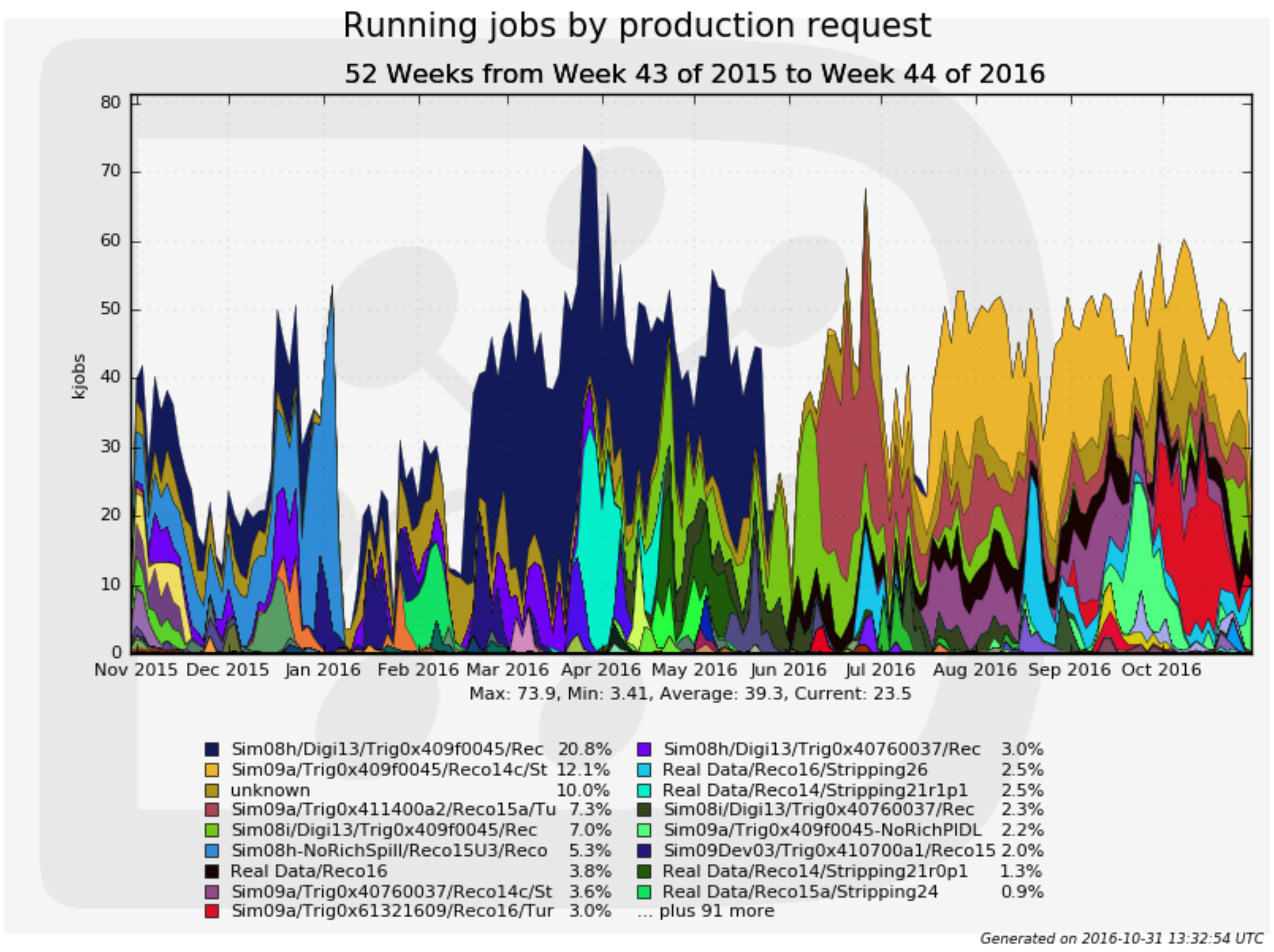
- There's more than just HTC in the world though
- What I've outlined for HTC doesn't provide the coherence that you need to use HPC resources
- We are starting to see some dedicated HPC facilities in HEP
    - eg CERN HPC service for theory, accelerator physics and engineering will be available as part of CERN batch next year
    - So HPC developments are interesting to us too
- (Some HPC sites also use HEP HTC jobs for backfilling)
- What about mixed workflows that are partially suitable for HTC but require some HPC stages?
- This requires an overall system for keeping track of workflows in both domains

DIRAC production requests

- Developed by LHCb for managing data+simulation processing
  - Adding 20 PB/year in 2016. Rate goes up x10 for Run3 in 2021.
- Manual Production Requests in very high level terms
  - eg “process the LHCb data from 2016”
  - Specify the stages that have to be done to the data files
- System breaks this down into a handful of Requests (~one per Stage)
- System then breaks the Requests down into individual jobs: millions in some cases
- Jobs run wherever DIRAC decides, depending on data location, remote data access, site type etc
- DIRAC keeps track of all this with DIRAC File Catalog and Bookkeeping service
- Could be extended to include HPC resources, with some stages only running at HPC sites?

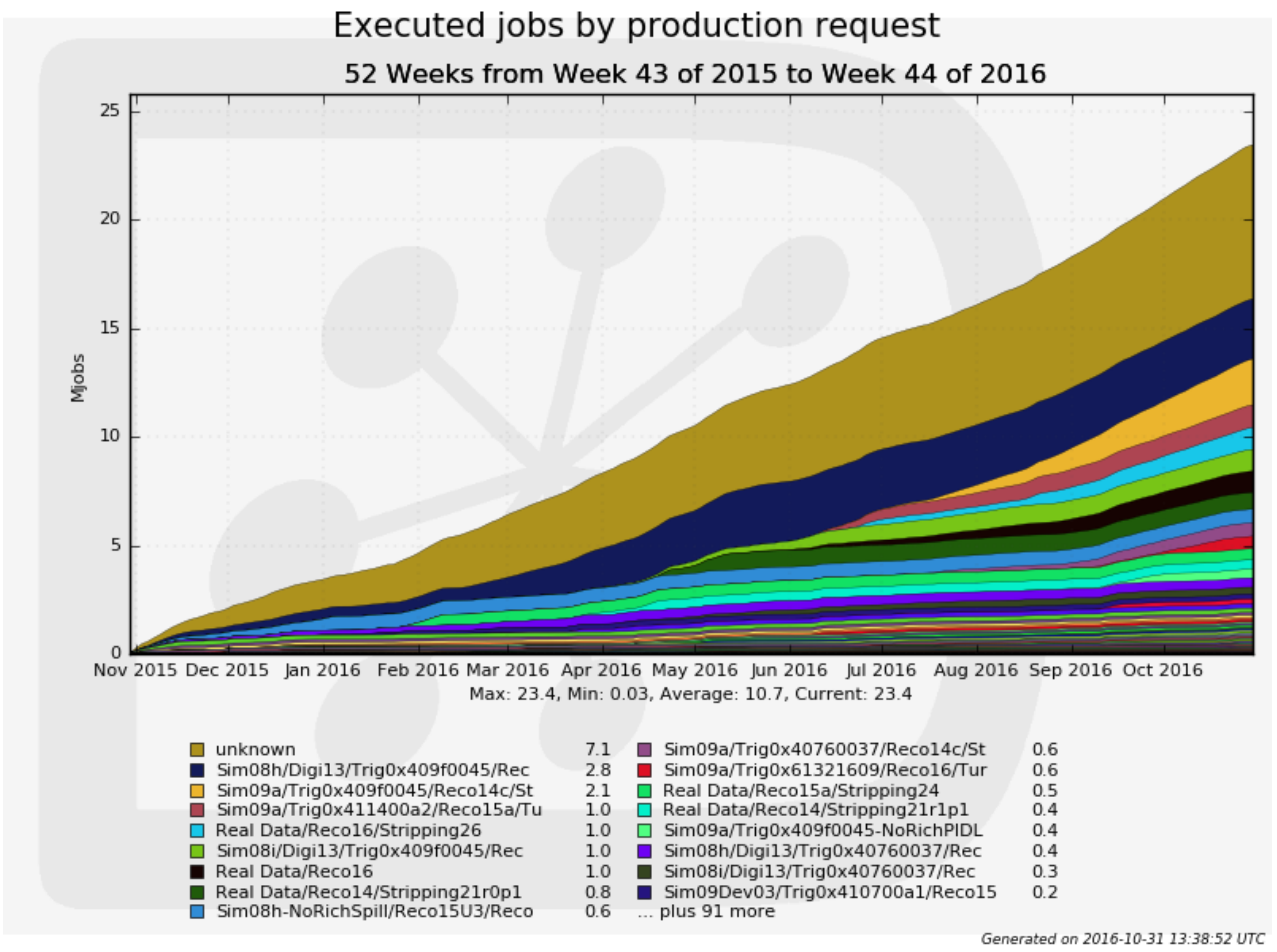
LHCb DIRAC production requests

"unknown"= user jobs etc



LHCb DIRAC production requests

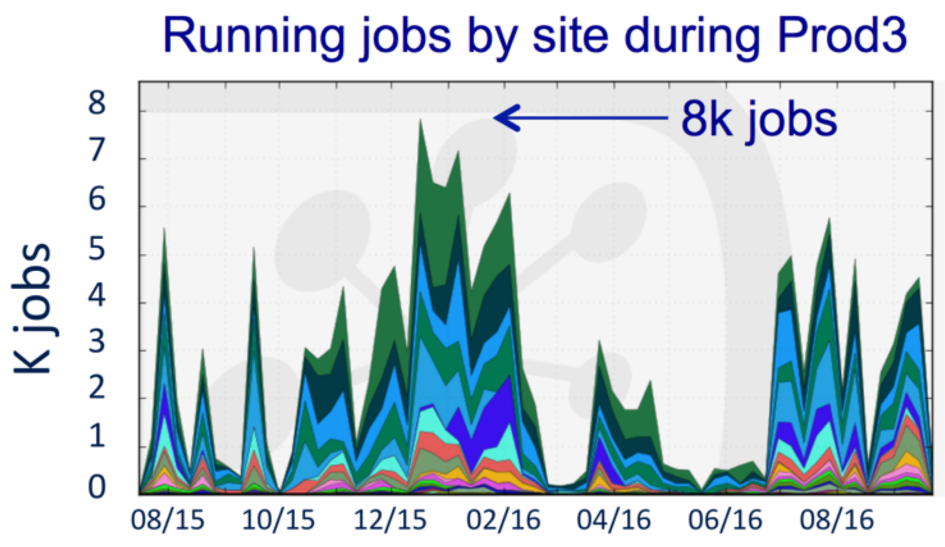
"unknown"= user jobs etc

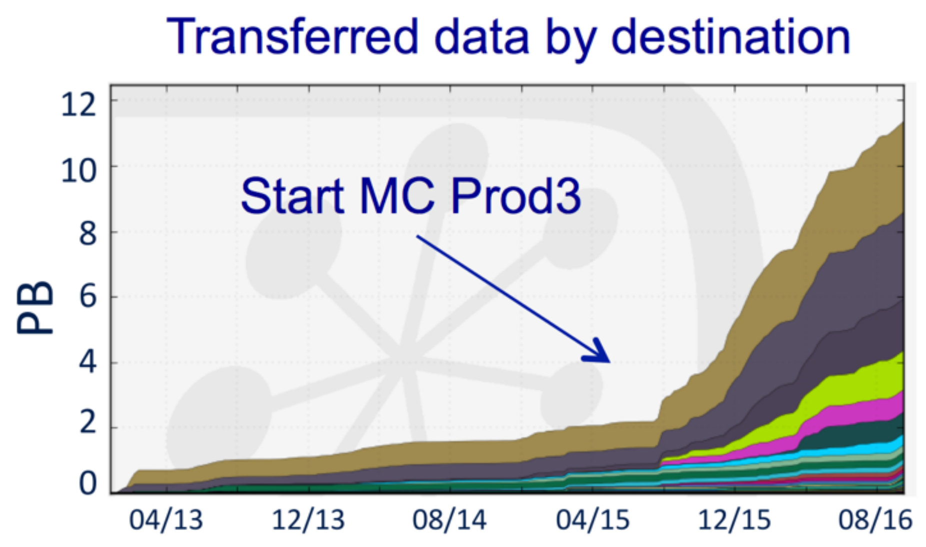


🟫	unknown	7.1	🟪	Sim09a/Trig0x40760037/Reco14c/St	0.6
🟦	Sim08h/Digi13/Trig0x409f0045/Rec	2.8	🟥	Sim09a/Trig0x61321609/Reco16/Tur	0.6
🟧	Sim09a/Trig0x409f0045/Reco14c/St	2.1	🟩	Real Data/Reco15a/Stripping24	0.5
🟥	Sim09a/Trig0x411400a2/Reco15a/Tu	1.0	🟦	Real Data/Reco14/Stripping21r1p1	0.4
🟦	Real Data/Reco16/Stripping26	1.0	🟩	Sim09a/Trig0x409f0045-NoRichPIDL	0.4
🟩	Sim08i/Digi13/Trig0x409f0045/Rec	1.0	🟪	Sim08h/Digi13/Trig0x40760037/Rec	0.4
⬛	Real Data/Reco16	1.0	🟩	Sim08i/Digi13/Trig0x40760037/Rec	0.3
🟩	Real Data/Reco14/Stripping21r0p1	0.8	🟦	Sim09Dev03/Trig0x410700a1/Reco15	0.2
🟦	Sim08h-NoRichSpill/Reco15U3/Reco	0.6		... plus 91 more	

Another example: Cherenkov Telescope Array

- Construction 2017-24; science from 2022; arrays in both hemispheres
- Been using DIRAC for last ~5 years for simulations
  - Requests, jobs, and DIRAC File Catalog
  - Using ~20 EGI grid sites
- Will need ~30PB/year storage, for all data and simulation





Graphs from L.Arrabito, LUPM

Questions, ideas

- Can you use DIRAC for HTC components of your workflows?
  - We can help with this, and it should be a quick/easy win
- Can you use DIRAC or parts of DIRAC (eg file catalog, request manager) for HPC?
  - If this helps, the code is there and well tested
- Can you make vacuum model VMs for non-DIRAC HTC work?
  - This would make it much easier to run your workflows on our Cloud sites (or OpenStack-compatible Vac sites) and vice versa
  - This would make it easier to use non-OpenStack cloud resources. Google, Amazon, Azure, …
- What about support? Ticket system, monitoring, mailing lists?
  - We use EGI GGUS (tickets) and Nagios/SAM/ETF (monitoring)

Summary

- HEP moved to the pilot model for HTC
- DIRAC is a generalised framework for doing this, used by several peta-scale projects
- Vacuum Model applies this to VMs for HTC
  - Vcycle and Vac implementations from GridPP
- CernVM and CernVM-FS useful components on which to build
- DIRAC Production Requests may be useful to manage work across HTC and HPC?
- LHCb and CTA demonstrate scaling of this at multiPB level
- Lots of interesting open questions