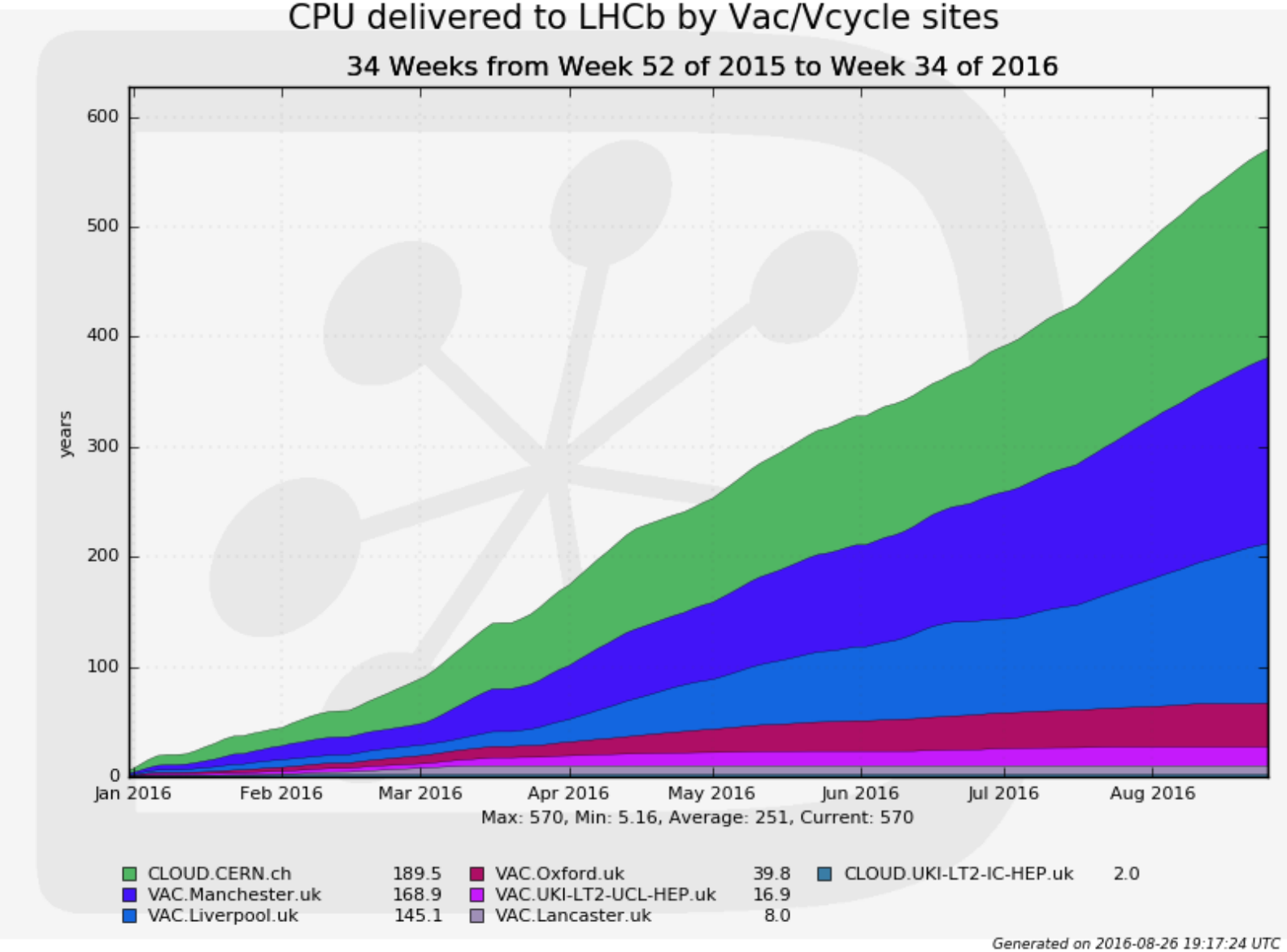
Vac, Vcycle, VMs status and plans

Andrew McNab
University of Manchester
LHCb

Vac vs Vcycle recap

- Two GridPP systems aimed at running VMs
- Vac - autonomous hypervisors
  - Each VM factory machine creates VMs in response to observed demand for each type of VM
  - More mature of the two, better documentation
- Vcycle - uses OpenStack etc
  - Factories created via Cloud API in response to observed demand for each type of VM
  - Code is solid, but docs are minimal: just man pages

LHCb Vac/Vcycle sites



Cloud Init ATLAS VMs

- GridPP's original ATLAS VMs ran the PanDA pilot script inside the VM
  - This worked but we were basically on our own supporting it
- Instead we made a second generation of ATLAS VMs, which use HTCondor to get pilots
  - Derived from VMs used on DBCE/CERN 2nd procurement
  - Same model as used by Sim@P1 VMs and CloudScheduler
  - Aim to converge all ATLAS VMs as much as possible
- Corresponding HTCondor services at CERN set up as production services, alongside pilot factories etc.
- Now rolled out across Vac based sites - comparable delivered CPU to LHCb
- Included in the ATLAS CLOUD/VMs CHEP2016 paper

Multiprocessor VM support

- ATLAS increasingly moving to “multicore jobs”
- Vac and Vcycle can already run multiprocessor VMs
  - UCL has been running dual processor LHCb VMs
- What about experiments that don’t have multiprocessor VMs?
- Support is being added to Vac for running a mix of VM sizes on the same hypervisor (VM factory) at the same time
  - Using “superslots” (next slide)
- This also needs “machinegroups”: target shares per experiment as well as per type of VM within that experiment
  - Experiment share is total of shares for each type of its VMs
  - Experiment retains its overall share even if one of its VM sizes has no work today

Superslots

- Vac 1.0 knows how many VM slots possible on the hypervisor
  - Same number of CPU/slot for each slot in normal running
- Next release will have “superslots” configured by the site
  - For example, 8 CPUs per superslot
- VMs will be created within a superslot
  - Each VM within a superslot has the same finish time
- This means CPUs can become available in a block of 8, ready for creating an 8 processor VM
- But if no multiprocessor VMs, can fill the superslot with single processor VMs (or 2, 3, 7, … processor VMs, whatever)
  - VM definitions include max_cpu and min_cpu limits.

Summary

- Vac, Vcycle, Vac-in-a-Box continue on a firm footing
- LHCb, ATLAS, and GridPP DIRAC
- Vac 1.0 released earlier this summer
- Preparing for mixed sized multiprocessor operation on Vac
  - Machinegroups and superslots