Integration of Grid and Local Batch Resources at DESY

C. Beyer, Th. Finnern, <u>A. Gellrich</u>, Th. Hartmann, B. Lewendel, Y. Kemp DESY IT

ID: 179 CHEP2016 13 Oct 2016 San Francisco, CA, USA





Introduction

- > DESY has a manifold scientific programme incl. HEP, photon science, astroparticle, accelerator, and theoretical physics
- > DESY IT provides computing and storage resources for all these scientific areas
- After the end of mainframes in the late 80s and an intermediate period of workstations of various vendors and OS', PCs and Linux took over
- > HEP groups built their own Linux-PC farms and deployed batch systems



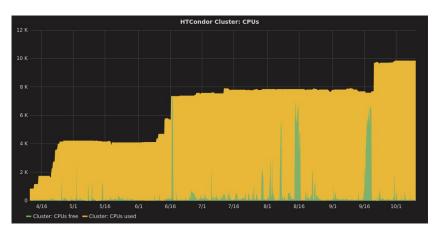


- > In 2000s Grid computing became a *key technology* for WLCG to find the HIGGS (*explicitly* mention in July 2012)
- > As a complement to the Grid, analysis facilities were built



History and Status

- > Grid computing since 2004: Tier-2, WLCG, EGEE/EGI, Belle II, ILC
- > National Analysis Facility (NAF) since 2007: ATLAS, CMS, LHCb, Belle II, ILC
- Solution > Grid and NAF access the same storage elements!
- > Grid: ~15k cores (150kHS06) (PBS/mysched) (2015)
- > NAF @ local batch: ~ 5k cores (50kHS06) (SGE) (2015)
- > PBS and SGE reach limits of scalability
- > Operation and maintenance is an issue
- > Consolidation of batch resources
- Merging LRMS' into HTCondor (2016)
- > Final cluster size: ~20k cores





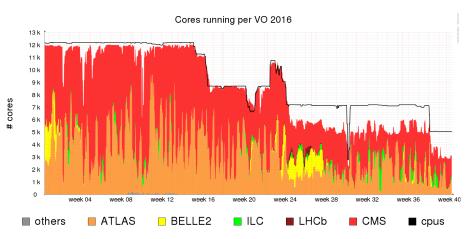
History and Status: Jobs at DESY-HH in 2016

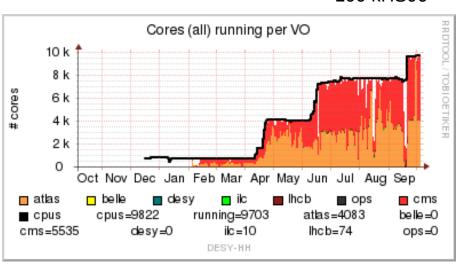
PBS/MyS(c)hed resources

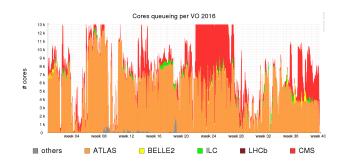
HTCondor (started 2016)

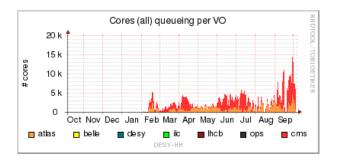














Use Patterns

> Grid

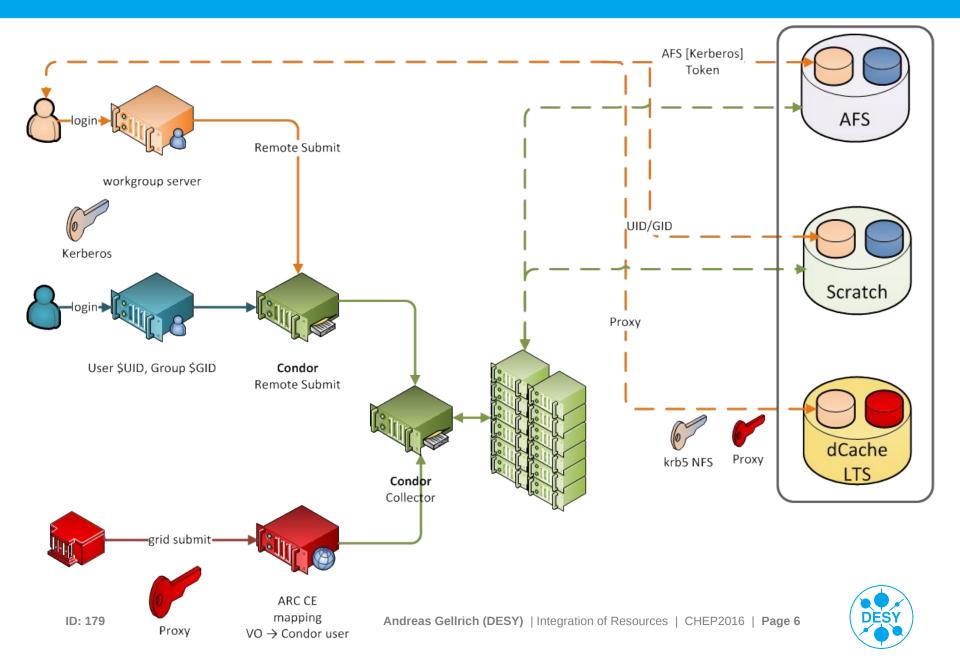
- Batch-like approach with submitting via (CREAM, ARC)
- Support for several VOs with user/group mapping to pools (VOMS)
- Due to pilot factories small number of active users (per VO)
- Established storage clients and protocols (lcg-*, gfal-*, xrootd*)

> Local batch

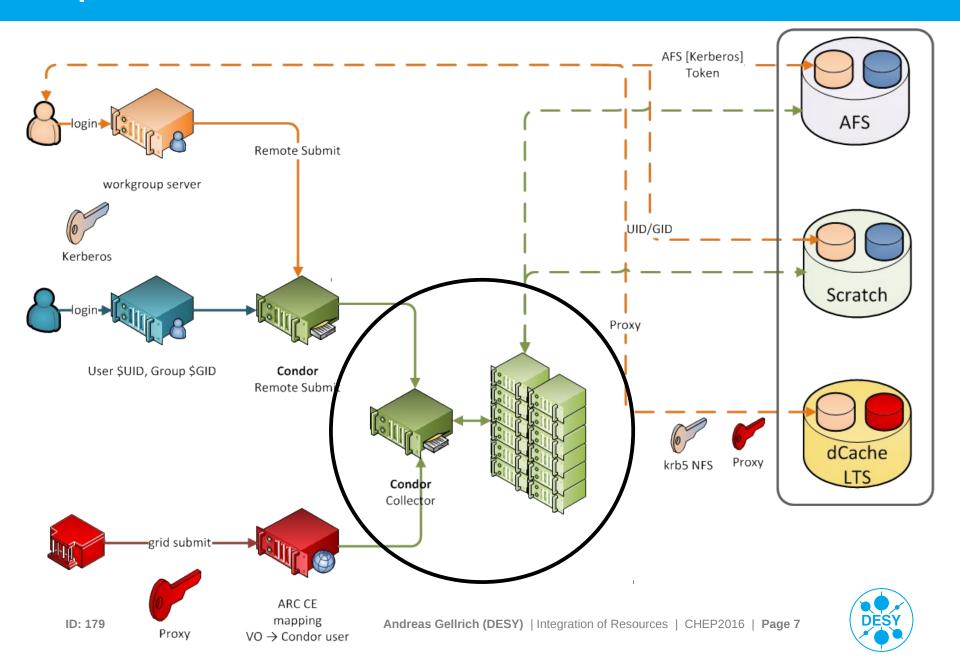
- Batch Interactive Resource at DESY (BIRD)
- Interactive approach with fast response and short turnaround times
- O(1000) individual users, potentially in multiple projects
- Resource management and usage accounting required
- KERBEROS, AFS, scratch space, mass storage (dCache) ...
- Storage access via NFS4, Grid-clients, ...



Implementation: Overview

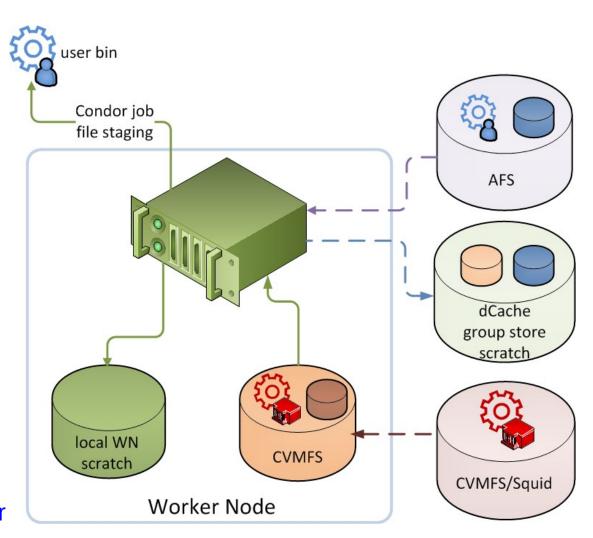


Implementation: Generic Batch Nodes



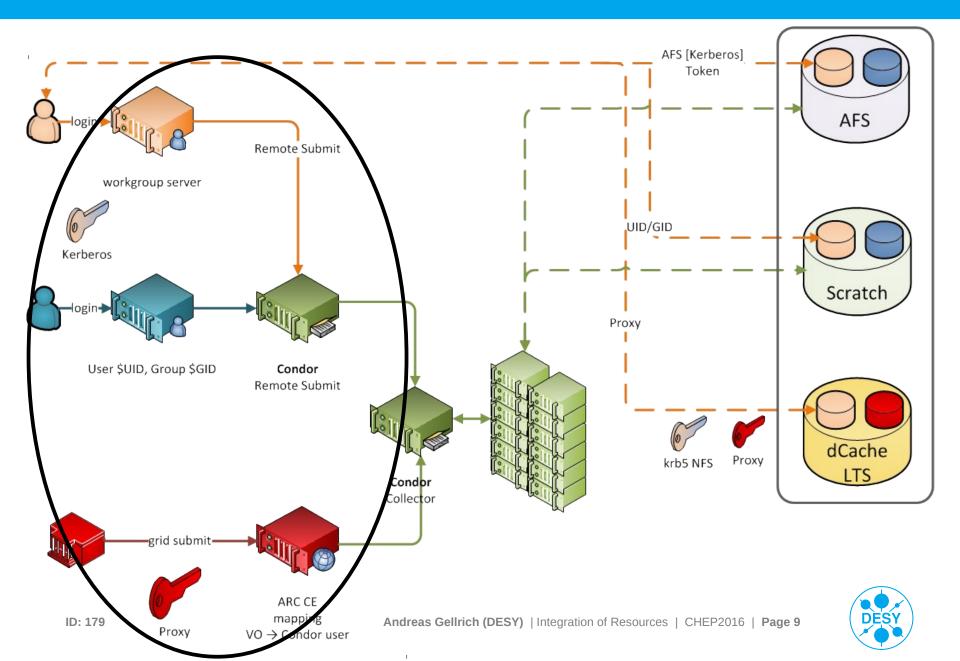
Implementation: Generic Batch Nodes cont'd

- > Generic batch nodes
- Concept of 'cattle' nodes
- > Plain SL6/EL7
- > HTCondor batch client
- Caching/remote FS
 - HTCondor staging
 - NFS4
 - (AFS / KERBEROS)
 - CVMFS
- > Client software via CVMFS
- > 233 nodes / 9822 cores
- Easy scaling with HTCondor





Implementation: User Mapping



Implementation: User Mapping cont'd

- Solution > Grid: Pool accounts/groups linked to VOMS groups/roles
 - Submission to ARC-CEs
 - Jobs: atlasusr007:atlasusr ... cmsplt001:cmsplt ... belleprd002:belleprd
 - Data: one account per VOMS group/role via proxy
 - Static mapping via table
- Local batch: Users choose project (via WGS)
 - Submission to local batch (to HTCondor schedd)
 - Dynamical setting of primary group depending on project
 - project determined via project-specific WGS
 - Data: storage controlled via GID (or Kerberos token in AFS)
 - Grid storage available via proxy



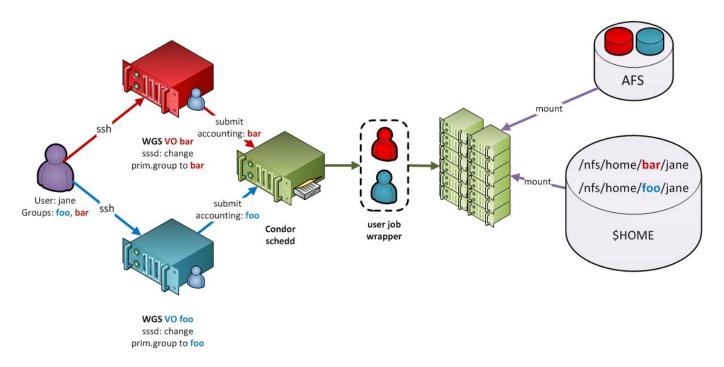
Implementation: User Mapping cont'd

> Currently:

- Submit from (some) generic WGS'
- \$HOME=/afs/...

> Future:

- per-project light-weight WGS w/ \$HOME=/nfs/home/\$GROUP/\$USER
- Shares and quotas bound to group



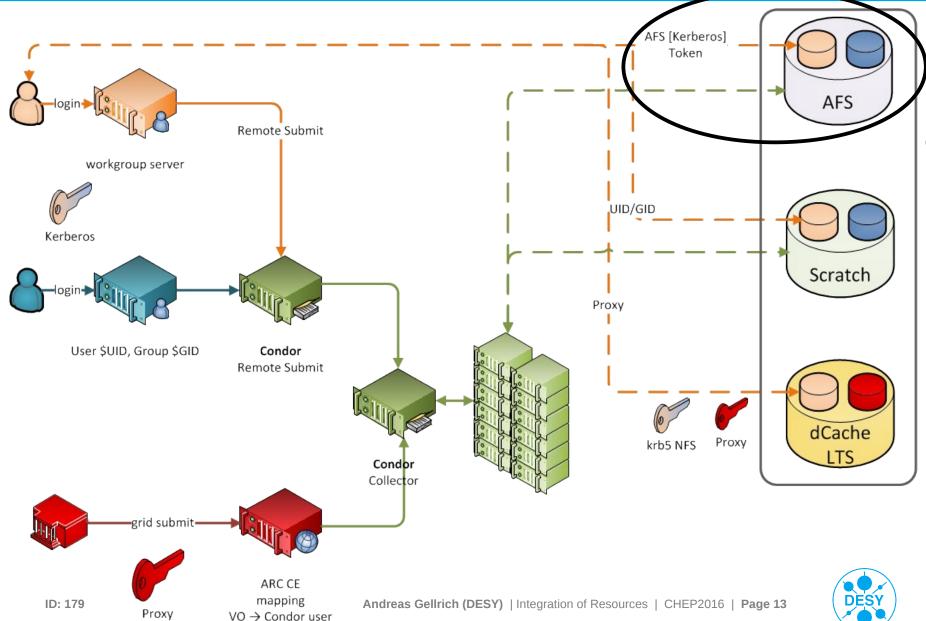


Implementation: Accounting, Shares, and Quotas

- Process accounting based on (UID/GID)
- Solution > Group specific \$HOME is under discussion
- > Used for quota
 - very easy for Grid (small number of users and groups)
 - challenging for local batch (large number of users and groups, non disjunct)
 - quota surplus handled by built-in fair-share mechanism (local batch → Grid)
- Interplay between fabric-like Grid jobs and interactive-style local batch jobs?
 - Local batch: Fast response for small #jobs rather than overall large throughput
 - Grid: Large throughput more important than fast response



Implementation: KERBEROS + AFS





Implementation: KERBEROS + AFS cont'd

- Supporting Kerberos for local batch users
- > Working Kerberos → AFS support trivial
 - Local batch: kerberized submission
 - Spawning long-term tokens for long-running campaigns (currently: 2 weeks max, up to 4 weeks optional)
 - Grid: auth* by ARC (HTCondor schedd)
- > AFS support currently needed
 - Users have legacy code, scripts, data,...
 - Encourage users to migrate to dedicated 'cluster local store' (NFS)
 - (Open)AFS future at DESY under discussion



Implementation: Cluster Monitoring

- > Rebooter script for automatic kernel upgrade
- Node health script + monitoring (*Icinga*)
 - "black hole", load, daemons, CVMFS, swap, partitions, ...
- > Feed into monitoring for operations/admin intervention (*Grafana*)

hostines AI *											
batch0201 ♂	batch0202 🗗	batch0203 ☑	batch0204 🕏	batch0205 🗗	batch0206 🗹	batch0207 🗗	batch0208 🗗	batch0209 🗹	batch0210 🗹	batch0211 🗗	batch0212 @
96%	100%	100%	98%	100%	100%	100%	100%	100%	100%	98%	96%
patch0213 ☑	batch0214 🗹	batch0215 🗹	batch0216 🗹	batch0218 🗗	batch0219 🗗	batch0220 C	batch0221 🗗	batch0222 🖸	batch0223 🗗	batch0224 🗹	batch0225 (
100%	98%	100%	100%	100%	100%	104%	100%	100%	100%	98%	98%
oatch0226 ☑	batch0227 🗹	batch0228 🗹	batch0229 🗹	batch0230 ☑*	batch0231 🗗	batch0232 🗗	batch0233 🗹	batch0234 ☑	batch0235 ☑	batch0236 ☑	batch0237
100%	97%	100%	N/A	100%	100%	96%	100%	97%	100%	100%	100%
batch0238 ♂	batch0239 🗗	batch0240 ☑	batch0241 ☑	batch0242 ☑*	batch0243 🗗	batch0244 🗗	batch0301 🗗	batch0302 🖸	batch0303 ☑	batch0304 E	batch0305
97%	100%	100%	88%	98%	100%	100%	97%	100%	100%	100%	100%
batch0306 ☑	batch0307 ☑	batch0308 🗗	batch0309 ☑	batch0310 ☑	batch0311 🗗	batch0312 ☑	batch0313 ☑	batch0314 ☑	batch0315 ☑	batch0316 ☑	batch0317
100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	98%
patch0318 ☑	batch0319 🗗	batch0320 ☑	batch0321 🗹	batch0322 ☑	batch0323 🗗	batch0324 ☑	batch0325 €	batch0326 ₺	batch0327 ☑	batch0328 🗗	batch0329
97%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
oatch0330 🗹	batch0331 🗗	batch0332 🗹	batch0333 🗹	batch0334 ☑*	batch0335 ☑	batch0336 🗹	batch0337 🗹	batch0338 ☑	batch0339 ☑	batch0340 ☑*	batch0341
100%	N/A	100%	100%	100%	100%	100%	100%	100%	0%	100%	100%
oatch0342 🕜	batch0343 🗗	batch0344 🗹	batch0345 🗹	batch0346 ☑	batch0347 🗗	batch0348 🗹	batch0349 🗹	batch0350 ☑	batch0351 🗹	batch0352 ☑*	batch0353
100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
hatch0354 [2	hatch0355 [2	hatch0356 [2]	hatch0357 [2]	hatch0358 [2]	hatch0601 [2	hatch0602 [2	hatch0603 FZ	hatch0604 FZ	hatch0605 [2]	batch0606 F	hatch0607



Conclusions and Outlook

- > Concept of generic nodes works (called batch nodes)
- > Grid: 2/3 of resources (~100kHS06) in HTCondor via 2 ARC-CEs
- > Local batch: Prototype submission
- (Maxwell) HPC cluster runs SLURM ...
 - HTCondor currently not optimized distributed parallel workflows in HPC
 - SLURM no option as Grid & local batch system
 - HTCondor has good community support for Grid and local batch
 - HTCondor for feeding HPC parasitically may be option in the future
- > Rebooter approach using ClassAds (→ Appendix)
- > Grid and Cloud ...
- Pilot factories replace batch systems ...



Appendix

Appendix



Rebooter: HTCondor ClassAds

- Configuration file for static parameters
 - Minimum uptime for rebooter, max ratio of nodes in draining or offline, list of paths for purge during reboot
- ClassAds per WN masters:
 - Rebooter active globally and on node
 - Kernel update active or force reboot FORCEREBOOT
 - Rebooter cycle states
 - status_YYMMDDHHMM and
 - inert (no current activity)
 - preparing (node ready to be drained)
 - draining (node waiting for collector's OK/to be drained)
 - rebooting (node finished draining and initiated reboot)
 - ERROR (unexpected state change etc.)

REBOOTER *

_ON, _HOST_ON

TARGETKERNEL,

HOST STATUS

status and change date locally advertised to/from collector

