

HTCondor at DESY.

Experiences and Outlook



Thomas Hartmann

2017.June.07



Migration → HTCondor

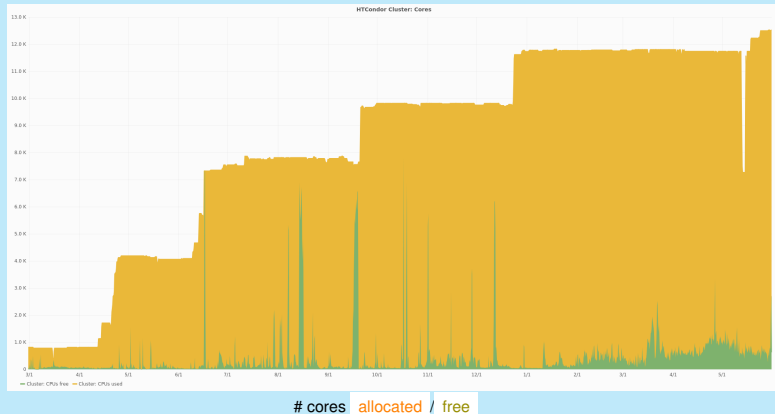
DESY Grid on HTCondor pool

- > running Condor pool in production for >1y now
- > have migrated Grid resources into Condor pool
 - now @ ~12500 cores
- > scaling the pool has been straight forward ☺
- > serving 5 larger VOs with 2 ARC CEs
- > grid universe
- > partitioning for single/multi-core jobs
- > cgroup resource handling
- > memory ~3GB/core



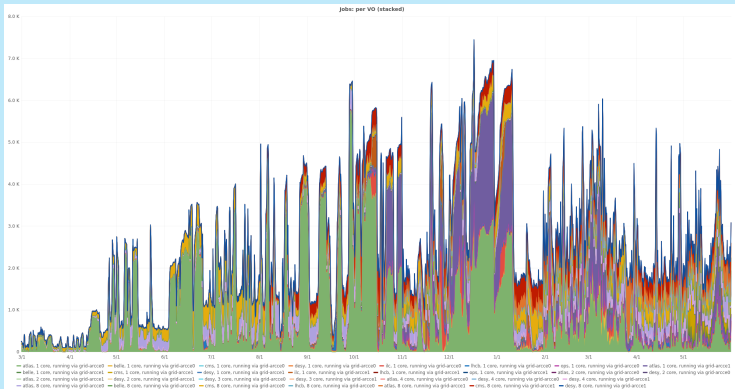
Migration → HTCondor

pool over the last year: # cores



Migration → HTCondor

pool over the last year: single/multi core jobs



colors: stacked cores per VO per sched



Grid on Condor

Batch Nodes

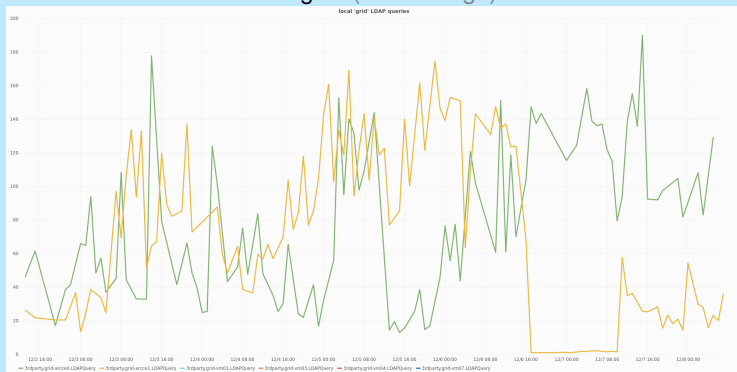
- > plain nodes: basic OS only
 - mainly SL6
 - EL7 batch nodes available (but limited use so far)
- > providing only CVMFS for Grid job
- > Grid clients and VO software via CVMFS
- + Singularity @EL7



ARC CE

ARC CE

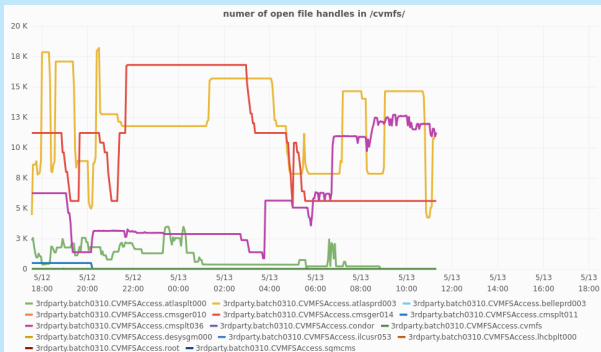
- > encountered scaling issues
- > ARCs now more stable for us
- > will look at Condor CE again (accounting?)



FUSE induced kernel crashes

CVMFS vs. kernel

- > colliding FUSE inode accesses crashed batch nodes



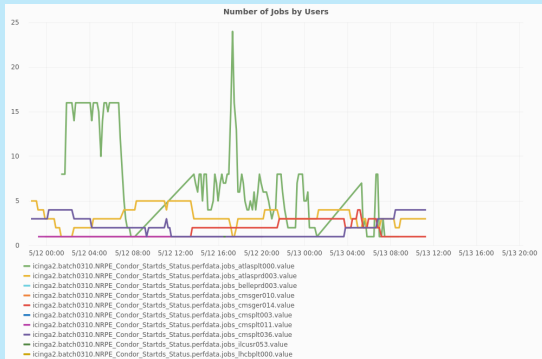
#k CVMFS file handles until node crashed



FUSE induced kernel crashes

CVMFS vs. kernel

- > reweighting batch node offers to increase diversity? [↗]
- > optimize entropy: users? cores?
- > crashes not necessarily correlated w. user/job entropy ☺



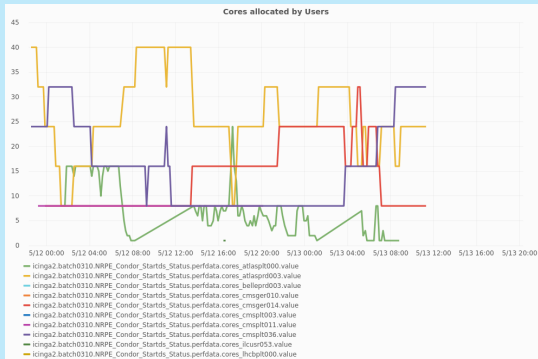
jobs per user until node crashed



FUSE induced kernel crashes

CVMFS vs. kernel

- > reweighting batch node offers to increase diversity? [↗]
- > optimize entropy: users? cores?
- > crashes not necessarily correlated w. user/job entropy ☺



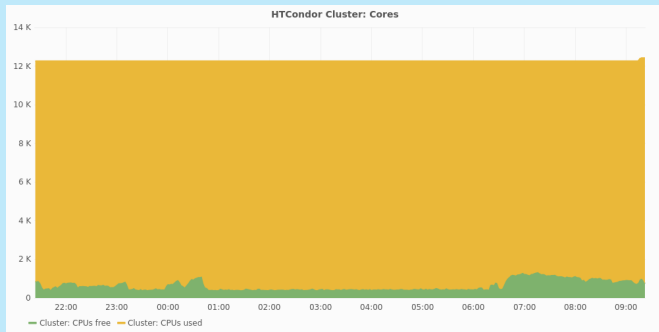
cores per user until node crashed



S/MCORE defragging/drainng

Draining losses

- > *default*: unallocated $\sim 3\%$ cores
- > following RAL's defrag approach [†]
- > mcore \rightarrow score regime: no losses (no surprise)
- > score \rightarrow mcore: $\nearrow \sim 10\%$ for 3-4h



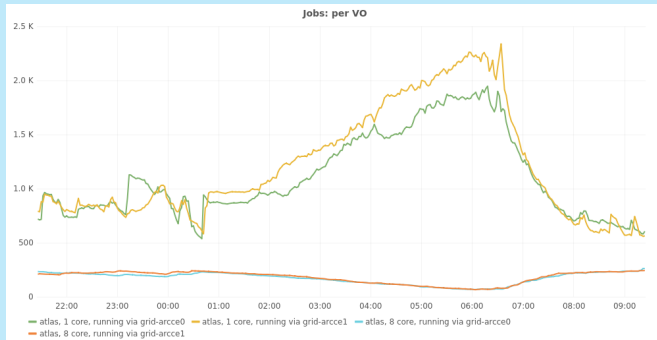
core allocation



S/MCORE defragging/draining

Draining losses

- > *default*: unallocated $\sim 3\%$ cores
- > following RAL's defrag approach [†]
- > mcore \rightarrow score regime: no losses (no surprise)
- > score \rightarrow mcore: $\nearrow \sim 10\%$ for 3-4h



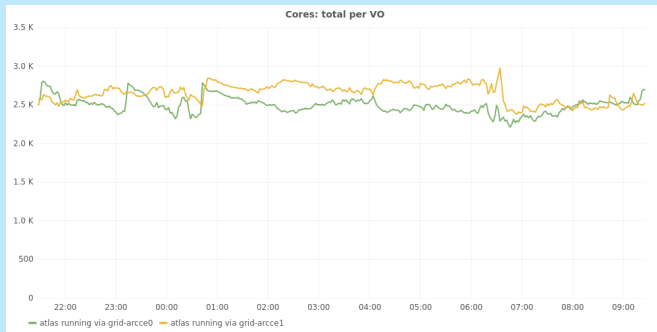
VO's active 1 / 8 -core jobs per schedd



S/MCORE defragging/draining

Draining losses

- > *default*: unallocated $\sim 3\%$ cores
- > following RAL's defrag approach [†]
- > mcore \rightarrow score regime: no losses (no surprise)
- > score \rightarrow mcore: $\nearrow \sim 10\%$ for 3-4h



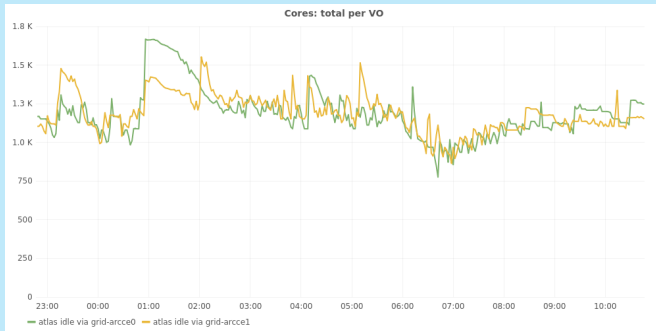
\sim stable core allocation per VO per schedd



S/MCORE defragging/draining

Draining losses

- > *default*: unallocated $\sim 3\%$ cores
- > following RAL's defrag approach [†]
- > mcore \rightarrow score regime: no losses (no surprise)
- > score \rightarrow mcore: $\nearrow \sim 10\%$ for 3-4h

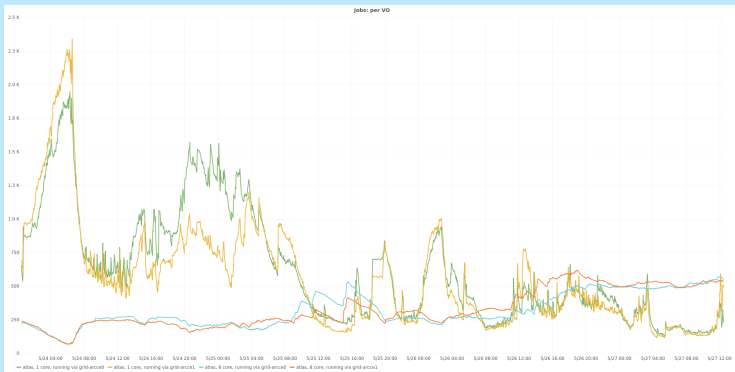


idle VO (job) cores per schedd



S/MCORE defragging/drainig

stable core inflow would be nice



zooming out: VO's active 1 / 8 -core jobs per schedd



Outlook

OS

- > EL7: taking care of CVMFS/kernel clashes(?)

IPv6

- > have been playing a bit with it
- > aiming with storage also towards enabling dual-stack

Containers

- > for users: singularity available on EL7 WNs for confining namespaces
- > Dockerize the cluster for own namespacing??
- ~ network namespacing could be nice to have!
 - layer 2/3 ~ monitoring(?!)

