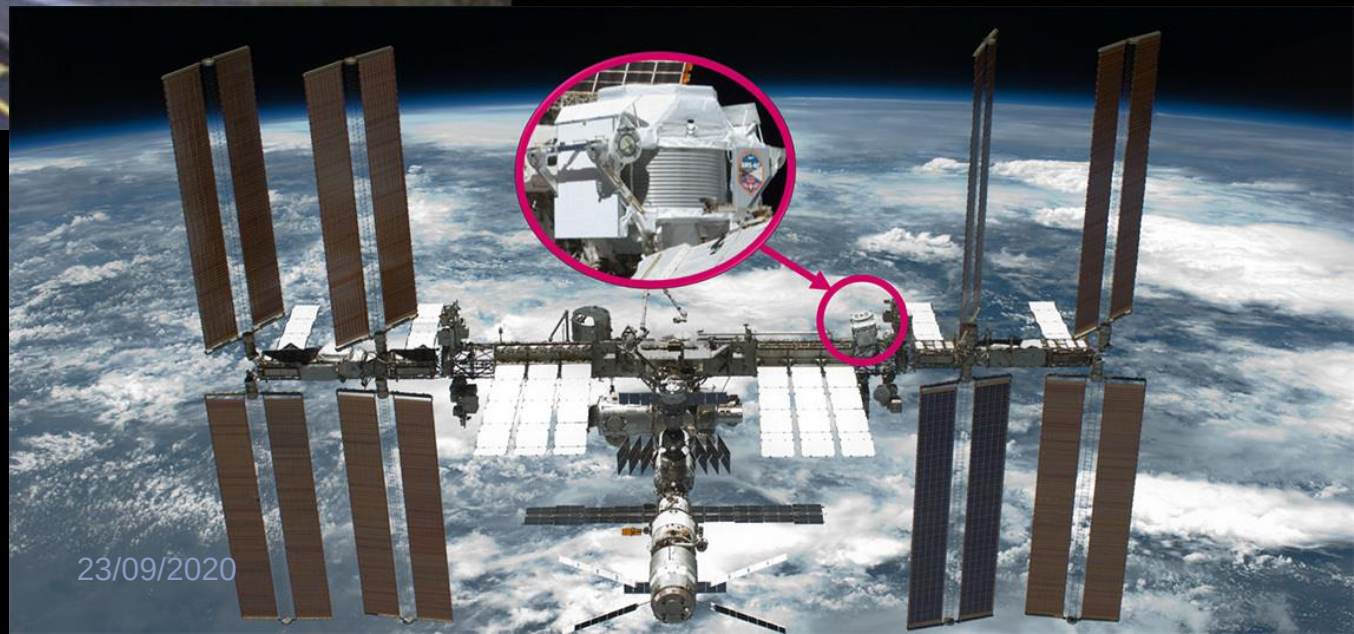
CERN

Running a large multi-purpose HTCondor pool at CERN

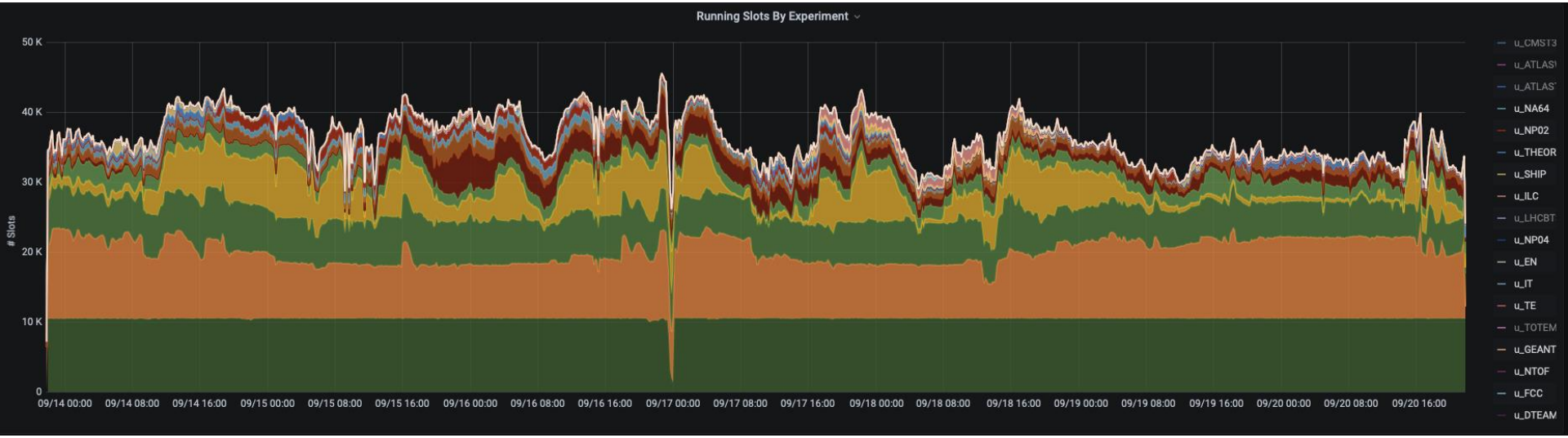
Ben Jones

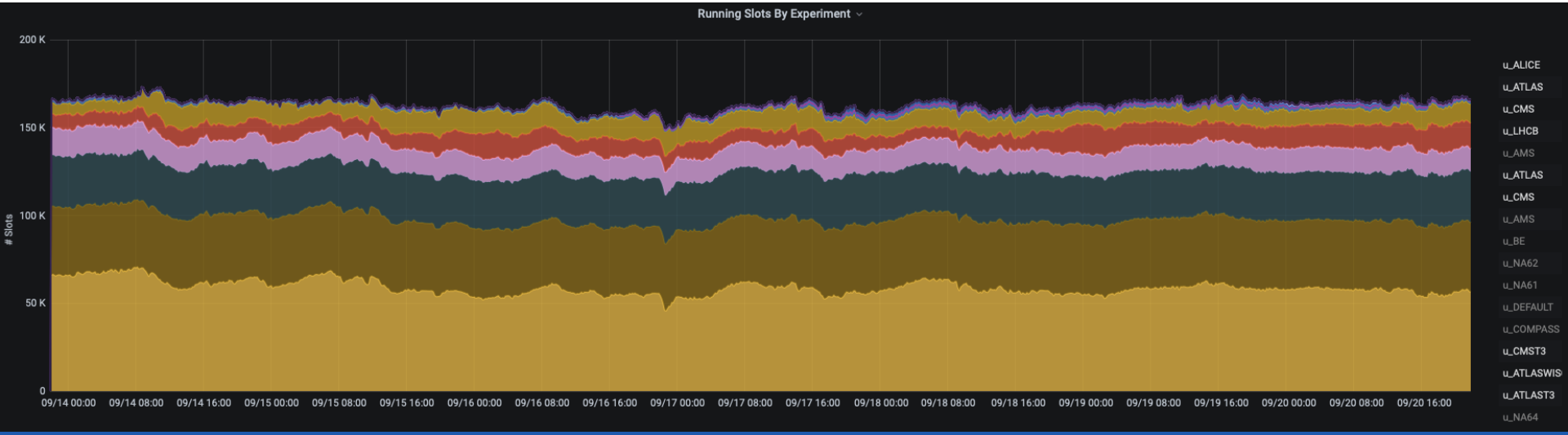
Not just the LHC…

AD & AMS

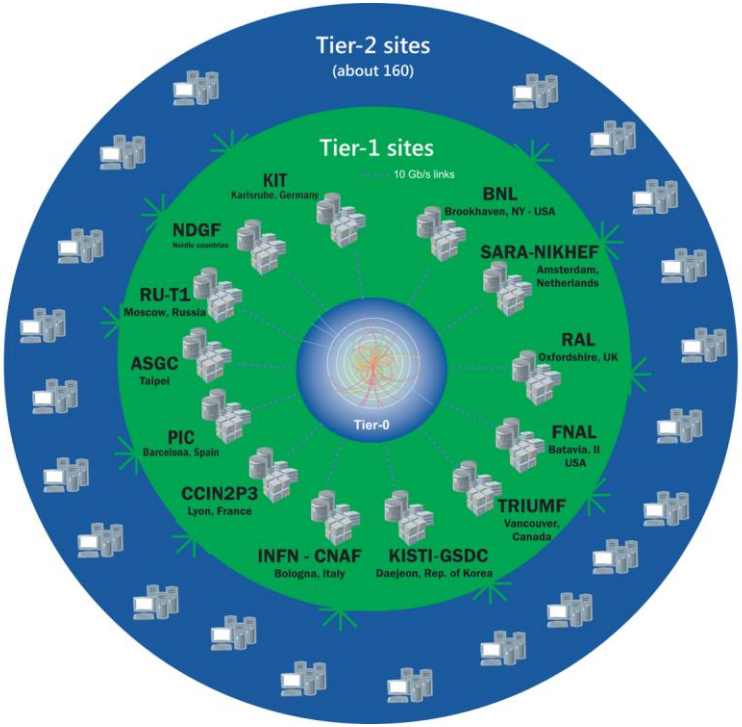


LHC vs Non-LHC





WLCG



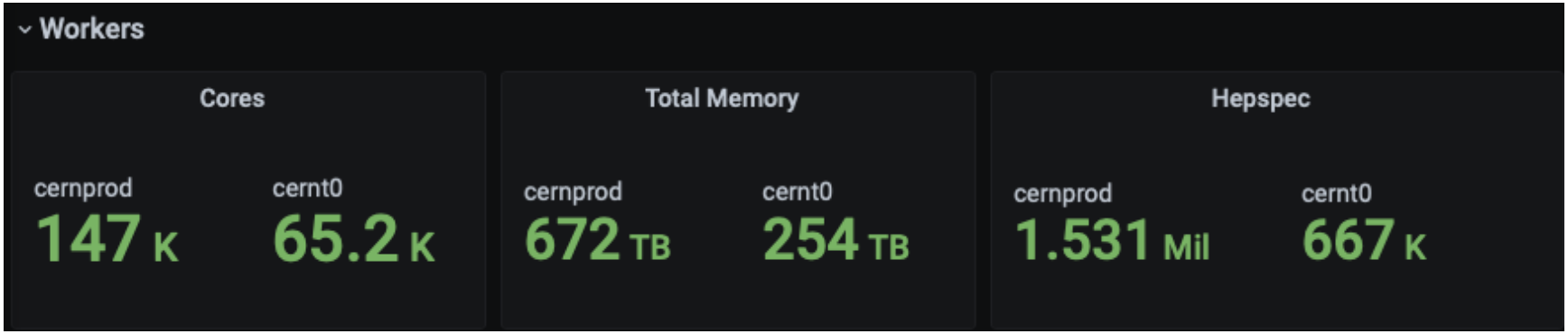
- CERN as Tier-0 of the WLCG
- Hosts tier-0 functions of LHC experiments
- 700 000 cores
- 200+ sites
- ~30 GiB/s
- ~70 PB/year

Grid vs local

CERN's batch system…

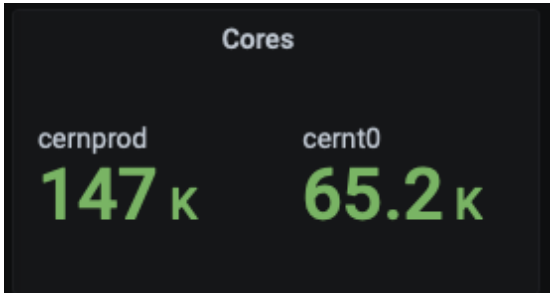
- Isn't just for LHC experiments
- Isn't just Grid
- As we'll see in a presentation later today, isn't even the only HTCondor batch system at CERN
- As the IT batch system, it is responsible for providing batch computing to experiments, groups and users associated with CERN

Worker stats



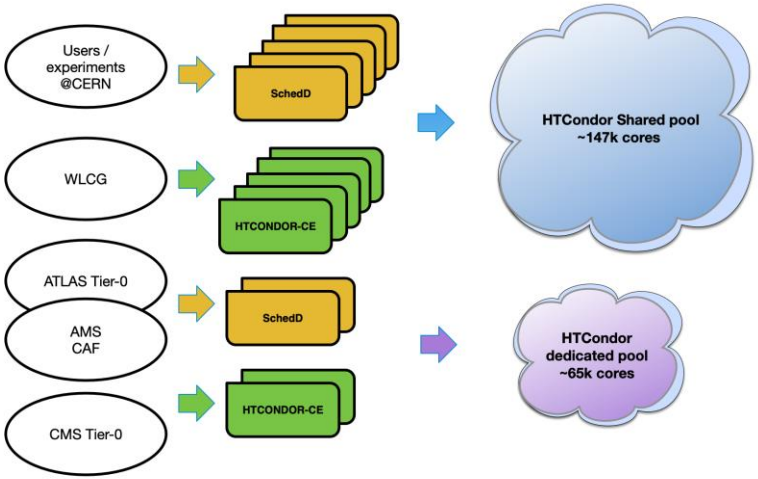
- CERN Batch system represents about 70% of the CERN Cloud
- All Batch machines are (still) SMT-on
- Majority of shared resources have 2 VMs per NUMA zone
  - VMs are scheduled by OpenStack to the physical sockets
  - (Yes, in prehistoric non-NUMA aware days we had VMs scheduled to SMT ”cores”)
  - 8 / 10 / 12 CPU VMs
- Older machines: just under 2Gb/core RAM
- Newer machines: just under 3Gb/core
- Significant minority of machines with non-standard configurations

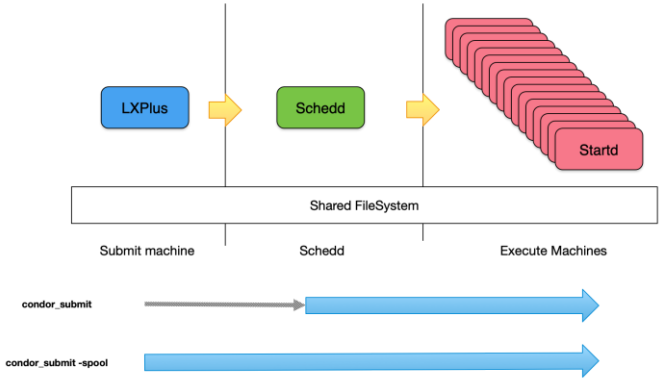
Pools



- **cernprod / share**
  - CM: 2x 40 core 120Gb RAM
    - 42 sub collectors
    - 3 negotiators
      - 2 split grid, 1 for “special” resources
  - 1 CCB machine
  - 12 schedds for “local” submission
  - 15 CEs for grid
- **tzero / dedicated**
  - CM: 2x 16 core 28Gb RAM
    - 8 sub collectors
    - 1 negotiator
  - 2 schedds for “local” submission
    - 1 of which is dedicated to ATLAS T0
    - 3 CEs for grid

Local submission





- "Local" submission: remote submission to schedds, using Kerberos and passing tokens
- Shared filesystem between remote submit machines (LXPLUS) and schedds
- Why remote submission?
  - We don't trust LXPLUS and nodes have short lifetime
  - We have privileged tokens on schedds
- Kerberos token renewal via our "ngauth" service, and HTCondor's SEC_CREDENTIAL_* knobs
- Mapping users to schedds

```
LOCAL_CONFIG_FILE = /usr/bin/myschedd.sh |
```

Routing / transforming: ATLAS

StartD:

```
HostGroup = ”bi/condor/tzero/atlas”
STARTD_ATTRS = $(STARTD_ATTRS) HostGroup
START = ((SendCredential =?= True) && (regexp("^group_u_ATLAS.TZERO.*", AccountingGroup)
|| regexp("^group_u_ATLAS.CAF.*", AccountingGroup)) || ((AtlasGridJob =?= True ||
EtfTestJob =?= True) && (SendCredential =!= True)))
```

CE
job_router:

```
[
     MaxJobs = 6000;
     MaxIdleJobs = 4000;
     TargetUniverse = 5;
     name = "AtlasT0";
     Requirements = (regexp("atlas", x509UserProxyVoName)) && (TARGET.queue =?= "AtlasT0");
      set_Requirements = (TARGET.Hostgroup =?= "bi/condor/tzero/atlas" && OpSysAndVer =?=
"CentOS7")
     set_AtlasGridJob = True;
     set_NiceUser = True;
   ]
```

Schedd
job transform:

```
JOB_TRANSFORM_To_Atlas_Tier0 @=end
    NAME To_Atlas_Tier0
    REQUIREMENTS ((regexp("group_u_ATLAS.TZERO.*", AccountingGroup) ||
regexp("group_u_ATLAS.CAF.*", AccountingGroup))  && jobUniverse =?= 5 )
    COPY Requirements VanillaRequirements
    SET Requirements (TARGET.Hostgroup =?= "bi/condor/tzero/atlas") && VanillaRequirements
@end
```

Preemptible resources

- Preemptible resources both from CERN & Public Cloud
- No drain, no real warnings
- Tidy up resources in Condor afterwards

Draining

```
# cat /usr/libexec/condor/scripts/staged drain.sh
#!/bin/bash

FILE=/etc/shutdowntime

if [ -f $FILE ];
then
    UNIX_SHUTDOWN=`awk '{print $0}' $FILE`
    echo "InStagedDrain = True"
    echo "ShutdownTime = $UNIX_SHUTDOWN"
else
    echo "InStagedDrain = False"
    echo "ShutdownTime = 0"
fi
```

- We often like to drain machines for interventions.
- Jobs specify max walltime, then we accept those jobs that will finish before the intervention time.
- We kill jobs that exceed their advertised wall time
- Jobs are accepted via more START expr magic

```
((InStagedDrain =?= True && (time() + MaxRuntime < ShutdownTime)) || InStagedDrain =?= False)'
```

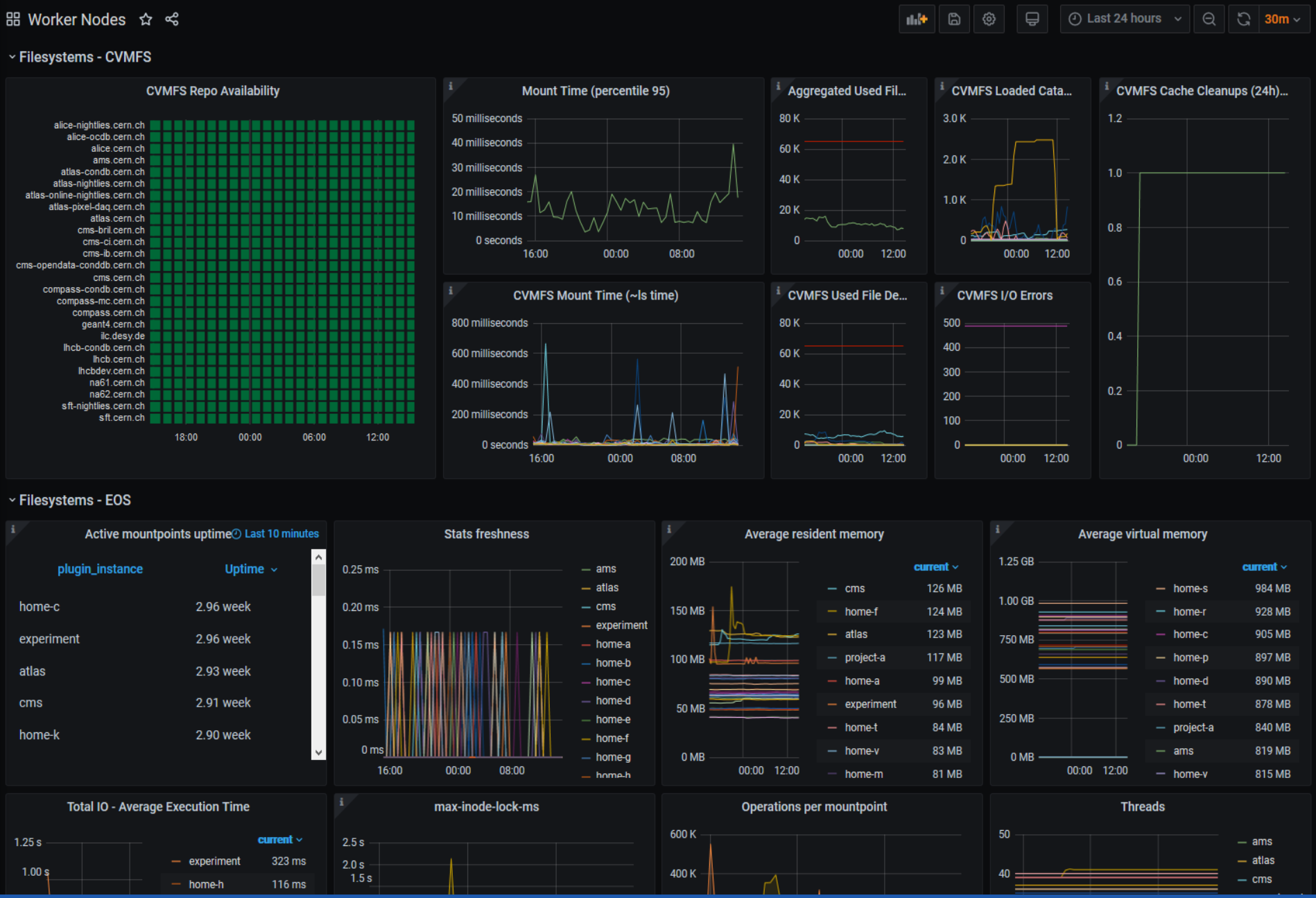
Healthchecks: CVMFS

- Previously: blacklisting startds if one cvmfs repo is broken.
  - Suboptimal: jobs might need only one of the healthy ones, or not require cvmfs at all.
- Need of more visibility on CVMFS led to the development of collectd-cvmfs.
  - Deployed and monitors most popular CVMFS repositories in our infrastructure.
- Extended startd crons to inject:
  - NODE_MOUNTS_CVMFS
  - CVMFS_HEALTHY_REPOS: string list with healthy repos.

CVMFS health (II)

- New attributes enables us to…
  - Mapping of Grid VOs to predefined repos.
  - Allow advanced users to define their requirements.

```
+CvmfsCustomExpression = false
+CvmfsCustomExpression =
stringListMember(”myrepo.cern.ch”,
CVMFS_HEALTHY_REPOS)
```


Worker Nodes
Last 24 hours
30m
Filesystems - CVMFS
CVMFS Repo Availability
alice-nightlies.cern.ch
alice-ocdb.cern.ch
alice.cern.ch
ams.cern.ch
atlas-condb.cern.ch
atlas-nightlies.cern.ch
atlas-online-nightlies.cern.ch
atlas-pixel-daq.cern.ch
atlas.cern.ch
cms-bril.cern.ch
cms-ci.cern.ch
cms-ib.cern.ch
cms-opendata-conddb.cern.ch
cms.cern.ch
compass-condb.cern.ch
compass-mc.cern.ch
compass.cern.ch
geant4.cern.ch
ilc.desy.de
lhcb-condb.cern.ch
lhcb.cern.ch
lhcbdev.cern.ch
na61.cern.ch
na62.cern.ch
sft-nightlies.cern.ch
sft.cern.ch
18:00
00:00
06:00
12:00
Mount Time (percentile 95)
50 milliseconds
40 milliseconds
30 milliseconds
20 milliseconds
10 milliseconds
0 seconds
16:00
00:00
08:00
Aggregated Used Fil...
80 K
60 K
40 K
20 K
0
00:00
12:00
CVMFS Loaded Cata...
3.0 K
2.0 K
1.0 K
0
00:00
12:00
CVMFS Cache Cleanups (24h)...
1.2
1.0
0.8
0.6
0.4
0.2
0
00:00
12:00
CVMFS Mount Time (~ls time)
800 milliseconds
600 milliseconds
400 milliseconds
200 milliseconds
0 seconds
16:00
00:00
08:00
CVMFS Used File De...
80 K
60 K
40 K
20 K
0
00:00
12:00
CVMFS I/O Errors
500
400
300
200
100
0
00:00
12:00
Filesystems - EOS
Active mountpoints uptime
Last 10 minutes
plugin_instance
Uptime
home-c 2.96 week
experiment 2.96 week
atlas 2.93 week
cms 2.91 week
home-k 2.90 week
Stats freshness
0.25 ms
0.20 ms
0.15 ms
0.10 ms
0.05 ms
0 ms
16:00
00:00
08:00
ams
atlas
cms
experiment
home-a
home-b
home-c
home-d
home-e
home-f
home-g
Average resident memory
200 MB
150 MB
100 MB
50 MB
0 MB
00:00
12:00
current
cms 126 MB
home-f 124 MB
atlas 123 MB
project-a 117 MB
home-a 99 MB
experiment 96 MB
home-t 84 MB
home-v 83 MB
home-m 81 MB
Average virtual memory
1.25 GB
1.00 GB
750 MB
500 MB
250 MB
0 MB
00:00
12:00
current
home-s 984 MB
home-r 928 MB
home-c 905 MB
home-p 897 MB
home-d 890 MB
home-t 878 MB
project-a 840 MB
ams 819 MB
home-v 815 MB
Total IO - Average Execution Time
1.25 s
1.00 s
current
experiment 323 ms
home-h 116 ms
max-inode-lock-ms
2.5 s
2.0 s
1.5 s
Operations per mountpoint
600 K
400 K
Threads
50
40
ams
atlas
cms

Automated user notification

- Anticipate known issues and do early notifications to users.
  - Too many files on AFS, wrong paths, out of quota,…
- Deploy a lightweight pipeline from logs to mail notification
  - Grok_exporter, Prometheus & Alertmanager.

Automated user notification (II)

HTCondor Schedd

grok_exporter

```
metrics:
- type: counter
  name:
sched_writeuserlog_error_total
  match:
WriteUserLog::initialize:
safe_open_wrapper\("%{AFS_AREA:
afs_area}/%{USERNAME:username}%
{UNIXPATH:path}"\) failed -
errno %{INT:error_number}
  paths:
  - '/var/log/condor/SchedLog'
  labels:
    err_num: '{{.err_num}}'
    username: '{{ .username }}'
```

Prometheus

schedd scrape target

```
scheddxy.cern.ch:9144/metrics
```

alert rule

```
UserFullAFSFolder:

delta(sched_writeuserlog_erro
r_total{err_num ="27"}[5m]) >
10
```

AlertManager

Notification

```
receivers:
 - name: 'email_users’
   email_configs:
   - to: '{{ template
"batch.user_notif.recipient" .
}}'
```

mail.tpl

```
<p>
Dear {{.GroupLabels.username}}
</p>
```

Never underestimate users…

- Our users often find ways to take full advantage of HTCondor's rich features in ways we did not anticipate
- Our submit requirements now becoming a historical record of various adverse reactions in the correct functioning of our plant

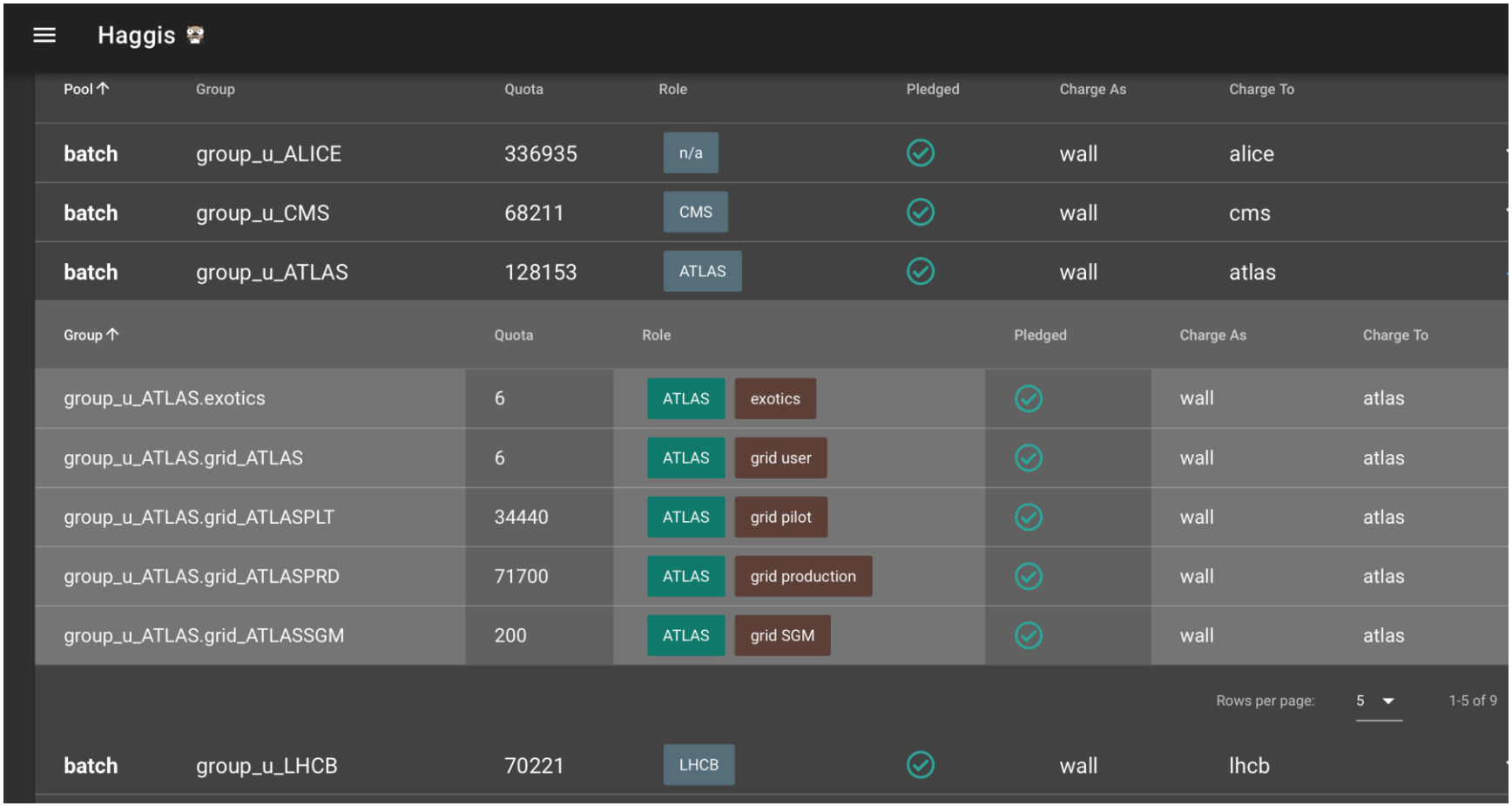
```
# Submit Requirement: NoJobDelay
SUBMIT_REQUIREMENT_NoJobDelay = isUndefined(NextJobStartDelay)
SUBMIT_REQUIREMENT_NoJobDelay_REASON = "Setting next_job_start_delay or
+NextJobStartDelay in submission is not allowed:
https://batchdocs.web.cern.ch/troubleshooting/commonexceptions.html#job-delay"

# Submit Requirement: NoUsageAds
SUBMIT_REQUIREMENT_NoUsageAds = (isUndefined(RemoteWallClockTime) ||
real(RemoteWallClockTime) =?= 0.0 ) && (isUndefined(CumulativeSlotTime) ||
real(CumulativeSlotTime) =?= 0.0)
SUBMIT_REQUIREMENT_NoUsageAds_REASON = "The following attributes cannot be set:
RemoteWallClockTime, CumulativeSlotTime"
```

ARRET D'URGENCE
GENERAL
GENERAL
EMERGENCY STOP
TELEPHONE ROUGE
CTER-00502
EUB60140
EXTINCTEUR
FIRE EXTINGUISHER

Backup

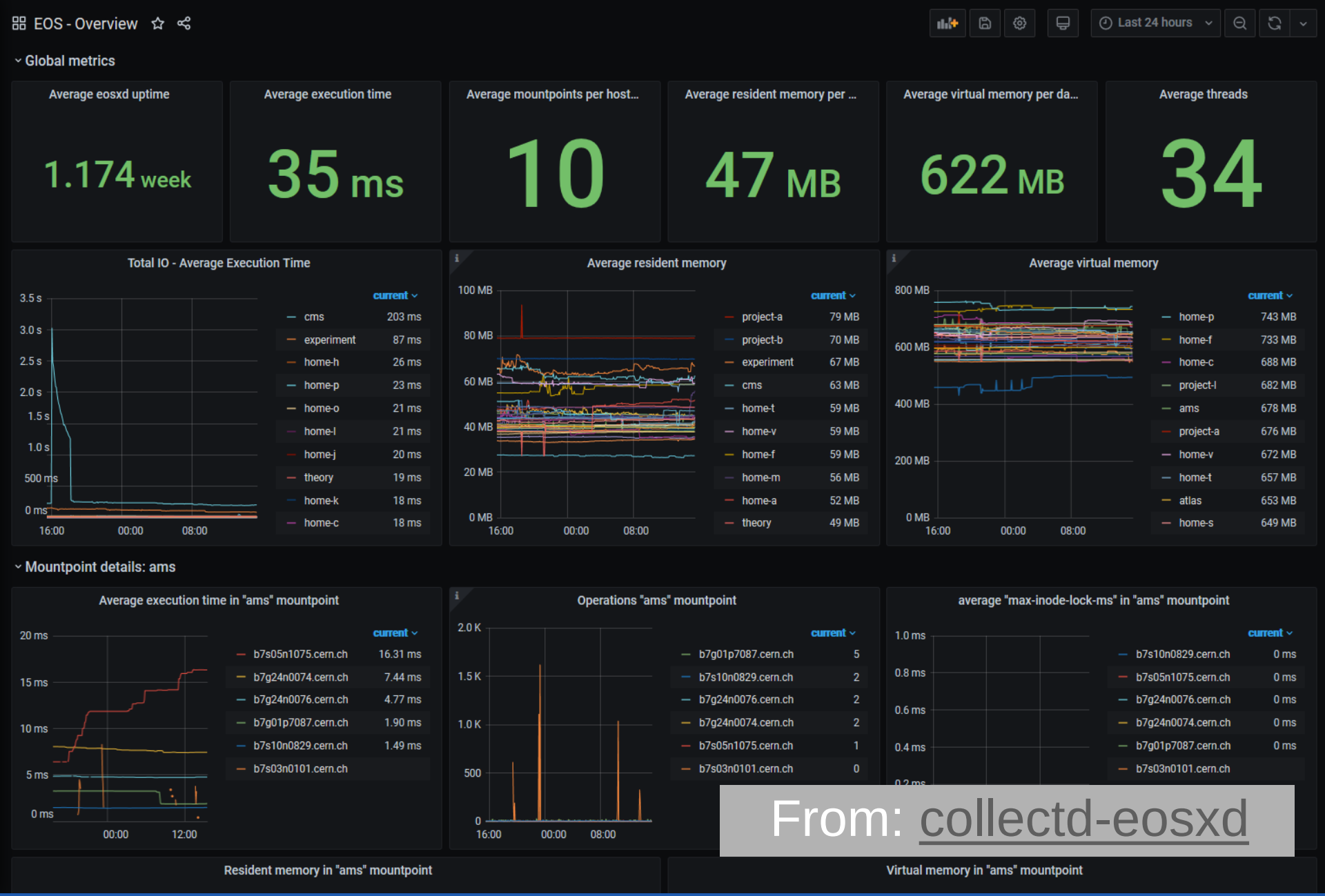
HAGGIS Accounting Groups



Accounting Group transforms

```
JOB_TRANSFORM_AccountingGroupCheck @=agc
    NAME AccountingGroupCheck
    EVALSET AccountingGroup (userMap("CERNGroups", Owner, AccountingGroup) =!= undefined) ?
strcat(userMap("CERNGroups", Owner, AccountingGroup), ".", Owner) : strcat("group_u_DEFAULT.all", ".", Owner)
    EVALSET MaxRuntime (userMap("JobFlavours", JobFlavour, undefined) =!= undefined) ?
int((userMap("JobFlavours", JobFlavour, undefined))) : (MaxRuntime =?= undefined) ? 1200 : MaxRuntime
@agc
```

- Accounting Groups map defined by HAGGIS dump
- userMap to assign to AccountingGroup, and dump to low-pri “DEFAULT” if there’s no assignment
- Also set default (low!) maxRuntime (we really should’ve called it maxWalltime)


EOS - Overview
Last 24 hours
Global metrics
Average eosxd uptime
1.174 week
Average execution time
35 ms
Average mountpoints per host...
10
Average resident memory per ...
47 MB
Average virtual memory per da...
622 MB
Average threads
34
Total IO - Average Execution Time
current
cms 203 ms
experiment 87 ms
home-h 26 ms
home-p 23 ms
home-o 21 ms
home-l 21 ms
home-j 20 ms
theory 19 ms
home-k 18 ms
home-c 18 ms
Average resident memory
current
project-a 79 MB
project-b 70 MB
experiment 67 MB
cms 63 MB
home-t 59 MB
home-v 59 MB
home-f 59 MB
home-m 56 MB
home-a 52 MB
theory 49 MB
Average virtual memory
current
home-p 743 MB
home-f 733 MB
home-c 688 MB
project-l 682 MB
ams 678 MB
project-a 676 MB
home-v 672 MB
home-t 657 MB
atlas 653 MB
home-s 649 MB
Mountpoint details: ams
Average execution time in "ams" mountpoint
current
b7s05n1075.cern.ch 16.31 ms
b7g24n0074.cern.ch 7.44 ms
b7g24n0076.cern.ch 4.77 ms
b7g01p7087.cern.ch 1.90 ms
b7s10n0829.cern.ch 1.49 ms
b7s03n0101.cern.ch
Operations "ams" mountpoint
current
b7g01p7087.cern.ch 5
b7s10n0829.cern.ch 2
b7g24n0076.cern.ch 2
b7g24n0074.cern.ch 2
b7s05n1075.cern.ch 1
b7s03n0101.cern.ch 0
average "max-inode-lock-ms" in "ams" mountpoint
current
b7s10n0829.cern.ch 0 ms
b7s05n1075.cern.ch 0 ms
b7g24n0076.cern.ch 0 ms
b7g24n0074.cern.ch 0 ms
b7g01p7087.cern.ch 0 ms
b7s03n0101.cern.ch
From: collectd-eosxd
Resident memory in "ams" mountpoint
Virtual memory in "ams" mountpoint