

HTCondor in Production

Seamlessly automating maintenance, OS and HTCondor updates with (almost) zero downtimes

Oliver Freyermuth, Peter Wienemann

University of Bonn
{freyermuth,wienemann}@physik.uni-bonn.de

13th October, 2020

Physics Institute at University of Bonn

- 240 members
- Biggest particle accelerator run by a German university ('ELSA', 164.4 m circumference) with two experiments (≈ 50 people)
- Groups from:
 - Particle physics: ATLAS, Belle II
 - Hadron physics
 - detector development
 - photonics
 - theory groups

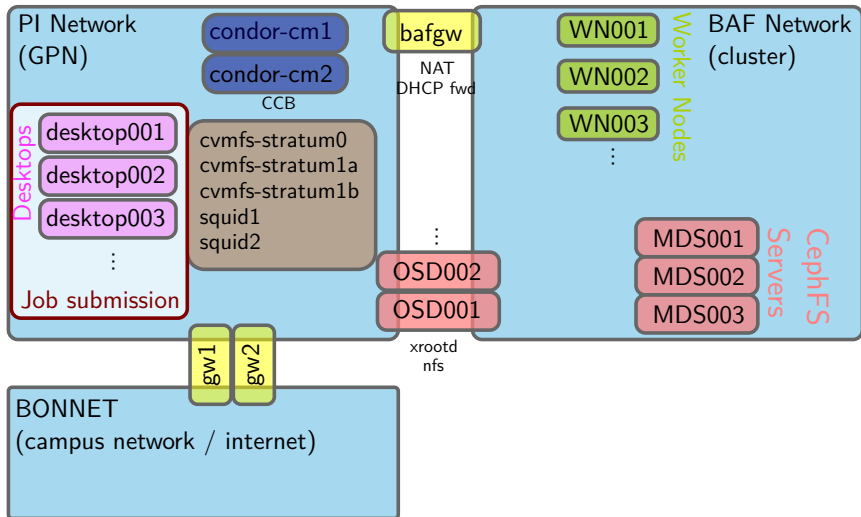
One cluster with growing hardware diversity & slowly growing resources for all users.

Provided by us — we are...

2 'full time' people, but cluster operations is only a small fraction
+ various part-time helping hands

⇒ **High degree of automation needed!**

Services surrounding the cluster



Key points of our setup

- No login / submission nodes ('use your desktop')
- Condor central managers in desktop network
- Desktops running Ubuntu 18.04 LTS --> Debian 10
- Cluster nodes running CentOS 7.8 --> CentOS 8
- Full containerization (all user jobs run in containers)
- Containerization decouples OS upgrades from user jobs
- Cluster file system (CephFS) directly accessible from Desktop machines via NFS.
- Cluster worker nodes interconnected with InfiniBand (56 Gbit/s), second data centre with Ethernet (1 Gbit/s per node, 10 Gbit/s total)
- All desktops, worker nodes, condor central managers fully puppetized, for HTCondor: [HEP-Puppet/htcondor](#)
Module allows to set up queue super-users, block users from submission, set up HTCondor for Singularity, set up a health check, . . .

HTCondor Automation

STARTD_CRON jobs executed by HTCondor on worker nodes

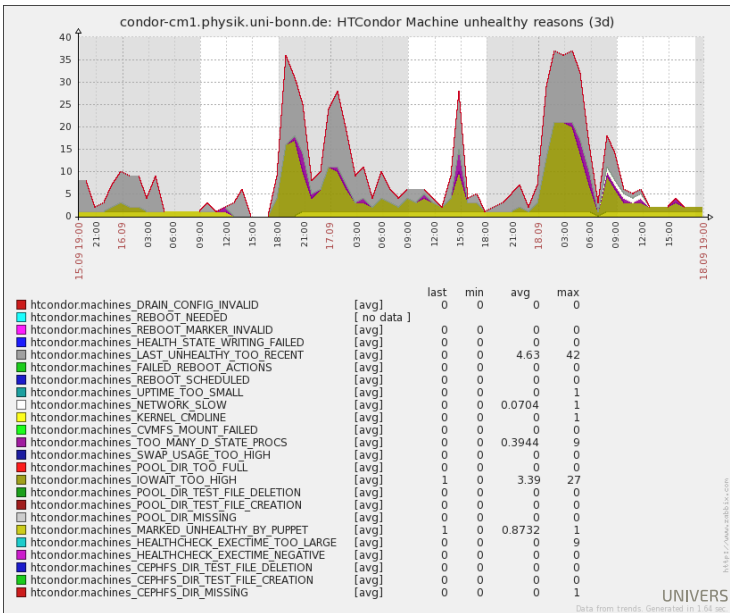
- node health check, covered in detail in this talk
runs every minute
- script extracting job and machine information (*every 5 min*):
 - `condor_who` to determine the latest job end time
(we ask users to specify maximum runtime, more later)
 - `condor_status` to find out applied drain settings
(more later)

If you have not used STARTD_CRON yet...

Output of `STARTD_CRON` becomes part of slot classads by default.

- Allows to adjust the `START` expression based on node health.
- Allows to use information in matchmaking with d-slots.

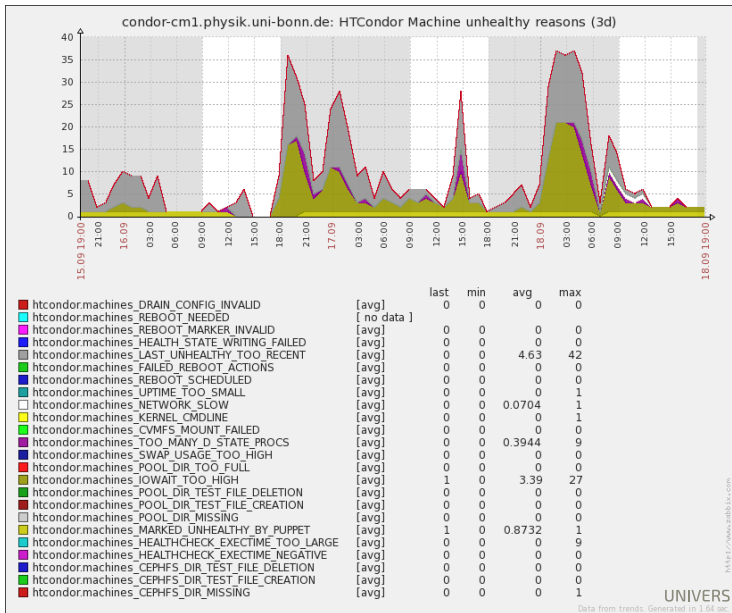
Node health checking



Node health checking: Reasons for 'unhealthiness'

- 🐛 last 'UNHEALTHY' too recent (debouncing, ≤ 10 min)
 - writing of status files failed or syntax bad (drain configuration, reboot marker, health state)
 - failed reboot actions (more later!)
 - reboot scheduled (i.e. `shutdown` command with timeout)
 - minimum uptime (≤ 20 min)
 - slow network interface (≤ 100 Mbit/s)
 - bad kernel command line (should contain 'console=')
 - unhealthy CVMFS mounts
- 🐛 swap usage is too high ($> 80\%$, HTCondor does not monitor swap)
- 🐛 iowait too high ($> 15\%$)
- 🐛 number of processes in D state too large ($> \frac{\text{\#logical cores}}{2}$)
 - read / write of execute directory or $> 80\%$ used (don't limit disk use yet)
 - administrative 'UNHEALTHY' marker
 - read / write of cluster file system, check if mount healthy
 - execution time of health check (> 10 s)

Node health checking



Node reboot handling

Two kinds of reboot reasons:

Reasons triggering staggered draining, reasons that don't drain

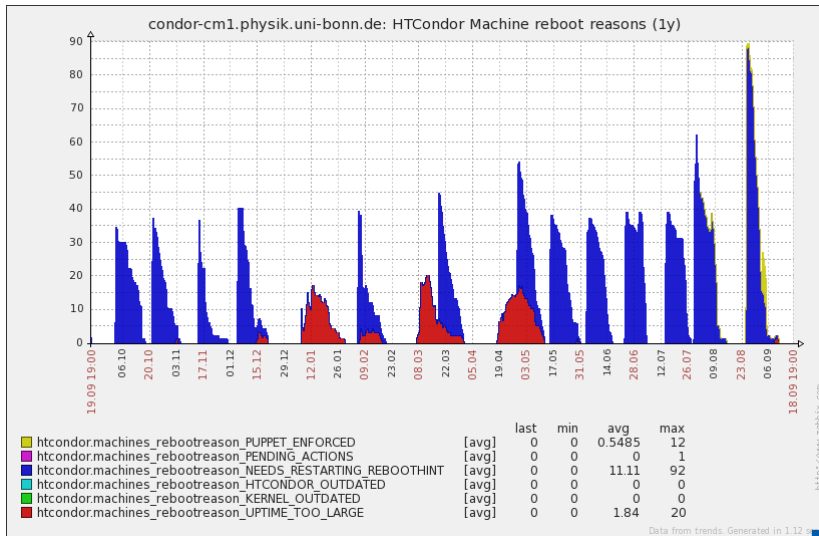
Draining reboot reasons

- Reboot for updates via `needs-restarting -r`
- Other possibilities: Kernel version too old, uptime > 30 d
- Start of drain smeared out over 10 d
- If drain starts, node marked for **immediate draining**

Non-draining reboot reasons

- Admin-enforced (Puppet)
- Pending actions to execute on reboot
- 'HTCondor version too old' can either be draining or non-draining

Node reboot handling



Draining: How a node becomes empty

Several techniques to drain a node...

- **immediate draining**

accept only jobs shorter than the latest allowed finish time of all running jobs (*users need to specify runtime!*)

⇒ like 'peaceful', but more efficient

- **timed draining**

don't accept jobs which may run longer than until a given point in time (*users need to specify runtime!*)

⇒ *planned maintenance periods*

- **unhealthiness**

don't accept any jobs anymore

⇒ *really like 'peaceful', used for hardware or software issues*

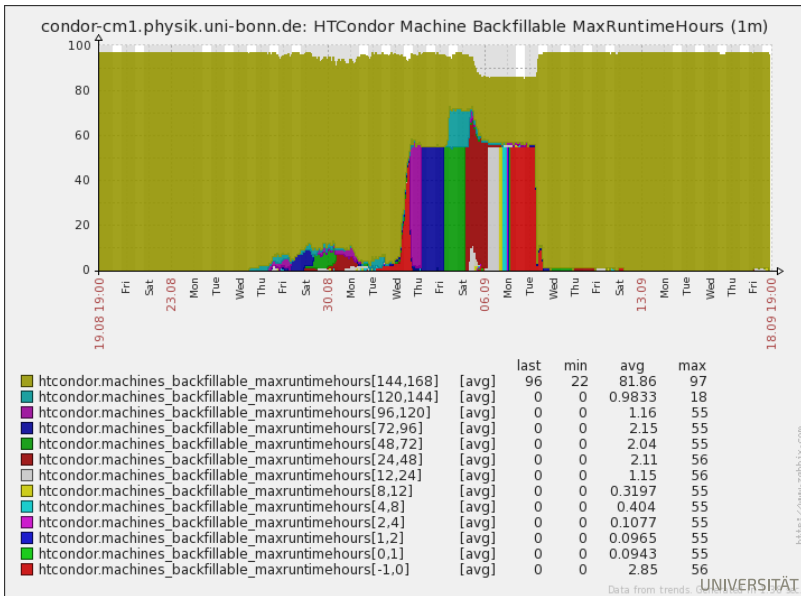
⇒ After the draining, the 'reboot' code runs:

reboot, shutdown or notification, extendable with scripts

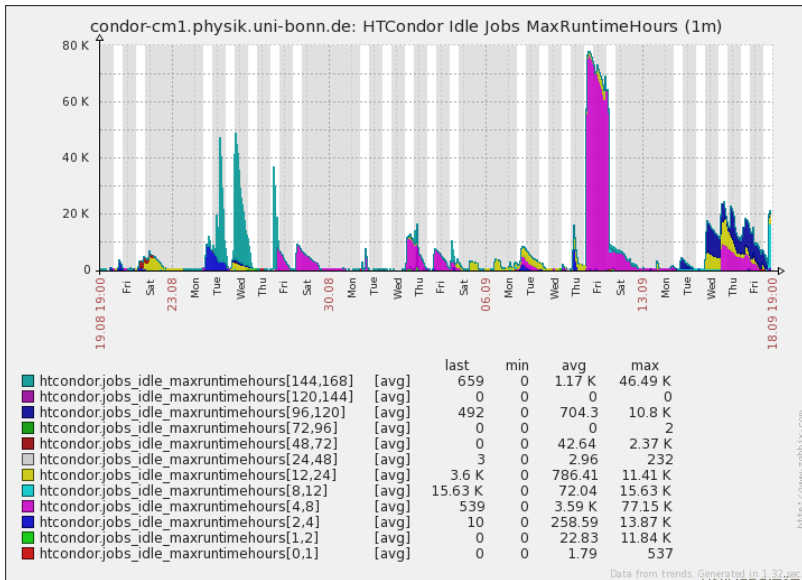
Feedback to users and admins

- Self-regulating: Users specify correct `+MaxRuntimeHours` , get more resources
- Jobs exceeding `+MaxRuntimeHours` are removed
- Enforced limits on `+MaxRuntimeHours` :
 - Maximum `+MaxRuntimeHours` for interactive jobs: 24 h
 - Maximum `+MaxRuntimeHours` for batch jobs: 168 h
- `SUBMIT_REQUIREMENT` on `schedd` creates good error messages, enforces setting of these attributes
- Mails sent out before machines are rebooted (possibility to interfere), contain output from reboot actions
- All attributes of machines are exported and visible to users
- All attributes and their distributions monitored by Zabbix
- Additional 'human-readable' attributes
(e.g. `NODE_REBOOT_REASONS` , `NODE_HEALTH` ,
`DRAIN_REASONS` , ...)

Monitoring BackfillableMaxRuntimeHours



Monitoring MaxRuntimeHours of user jobs



Admin feedback via the MOTD

```
Welcome to CentOS Linux release 8.2.2004 (Core) (GNU/Linux 4.18.0-193.14.2.el8_2.x86_64
↪ x86_64)
```

```
-----
This host is:
```

```
* a bare-metal (physical) machine
* using master: puppet-baf.physik.uni-bonn.de
* in hostgroup: Linux/Server/BAF/WN eth Ceph no IB
* in environment: production
* in location: Nußallee 12
* in room: 0.004
* in rack: 14
* in bay: 17
* commissioned on: 2020-09-16 (brand new!)
* in condor health state:
  MARKED_UNHEALTHY_BY_PUPPET:TEST_NODE
* last marked unhealthy: 0d 0h 0m 5s ago
  (at Fri Sep 18 00:30:00 CEST 2020)
* in timed drain
  backfillable MaxRuntimeHours: 26
  drain target time is in: 1d 3h 38m 5s
  (Sat Sep 19 04:08:10 CEST 2020)
* marked for draining due to the following reason(s):
  COOLING_PROBLEMS
* running 1 job(s)
* running the latest finishing job for up to 0d 7h 55m 1s
  (Fri Sep 18 08:25:06 CEST 2020)
```

```
Time: 00:30:05 up 1 day, 3:58, 1 user, load average: 0.08, 0.04, 0.00
-----
```

Use case examples

Memory upgrade of nodes

- Set drain target time to start of intervention (or immediate)
- Set reboot action to shutdown
- Inject a script running `dmidecode -t memory`

Broken hard disk

- Mark node as UNHEALTHY
- Enforce 'reboot' with Puppet
- Set reboot action to shutdown or notification

Intervention on cooling system

- Set a drain target time for the start of intervention (or UNHEALTHY if reduction is sufficient)
- Set reboot action to shutdown

Avoiding Downtimes: Multiple Ingredients

HTCondor

Operation of two Central Manager machines (negotiator, collector):

- VMs on different hypervisors (live migration possible)
- Redundant block device storage (Ceph RBD)
- Using high availability daemon of HTCondor

File systems

- CVMFS: Two Stratum one servers and two Squid servers (physical nodes, but in different racks --> buildings)
- CephFS: 8 file servers with 3 metadata and MON servers, redundancy via erasure coding (any two data servers can fail)

Networking: WIP

Cluster network NATted via redundant firewall cluster

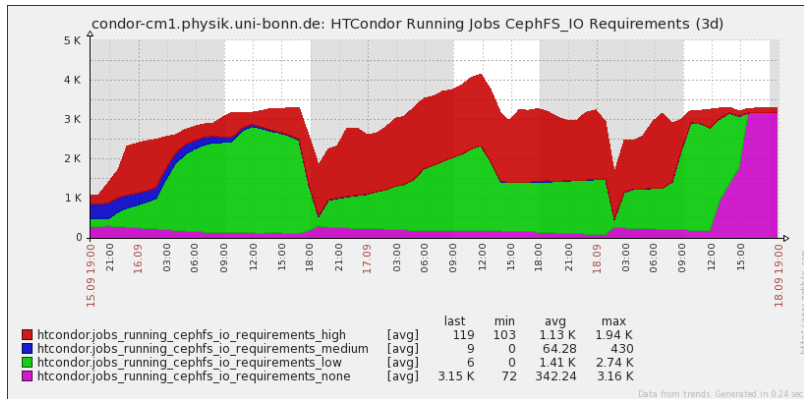
Avoiding Downtimes: Distributing a Cluster

Group with existing cluster joined our institute over Summer

- Nodes located in a different building
 - 1 No clean way to connect to our IPoIB CephFS
⇒ Use Ethernet
 - 2 Network switches replaced with models with faster uplink
- Reinstallation of all nodes fully automated
BIOS and BMC updates and initial configuration still manual labour
- Model difference of this cluster in HTCondor (one more dimension!):
Users choose `+CephFS_IO` from 'none', 'low', 'medium', 'high'
- Also paves the way for integration of other resources (HPCs, Desktops, ...)
- No hard fragmentation / partitioning needed!

Outages / maintenances increasingly unlikely to hit all resources.

Monitoring CephFS_IO



⇒ Huge diversity of jobs (and we backfill jobs with 'none' local I/O from WLCG).

Other news...

Singularity interactive jobs & `condor_ssh_to_job...`

Now work in 8.8.10, **already in production for our GPU machine!**

Many thanks to Greg Thain for persisting through many issues!

Some quirks remain with X11, see [HTCondor Wiki](#)

Functional MPI jobs with Singularity containers...

without requiring HTCondor inside the containers, using

<https://github.com/htcondor/htchirp>

Can now replace `condor_chirp`, thanks to Jason Patton!

Ongoing work: JupyterHub with CCB

Adapt batchspawner: `condor_ssh_to_job` for port forwarding.

⇒ Run JupyterLab in jobs on 'firewalled' / NATted worker nodes!

Working proof-of-concept exists (will upstream later).

Conclusions

- Huge flexibility of HTCondor allowed us to:
 - backfill draining nodes easily and get more cycles out
 - add second data centre with different I/O to the existing cluster
 - expose all node states transparently to our monitoring and users
- Knobs like the `STARTD_CRON` allow easy automation
- We and our users don't have to take manual care of:
 - security updates / reboots
 - draining for maintenance periods
- Reducing downtime comes with many components
- Users don't even notice maintenance / downtime (only parts of resources affected)

If you use HTCondor or consider it. . .

Join the HTCondor workshops!

Exchange of lots of experience and ideas (also in a virtual workshop), presentations for both beginners and experts!

Thank you
for your attention!



Example for a mail upon node reboot (1)

Dear BAF admins,

a reboot action was scheduled for wn000.baf.physik.uni-bonn.de in 30 minutes
for the following reason(s):
"PENDING_ACTIONS:1"

For your information, the machine has not yet been reboot-draining
(plan was to start at Wed May 8 01:02:20 CEST 2052), but it was empty and we made use of
↪ the chance.

In case you want to cancel this and prevent a future
reboot, login to wn000.baf.physik.uni-bonn.de within the next 30 minutes,
and execute:

```
shutdown -c  
condor_on
```

Quickly set the inactivate-flag on profile::wn_reboot_check
for this host to true, and execute Puppet.

Here's a summary of the 1/1 executed reboot actions:

1. Executed /etc/wn-reboot-actions/10-hello-world.sh, it had the following content:

```
-----  
#!/bin/bash  
echo "Hello world!"  
-----
```

It exited with exit code 0 and the following output:

```
-----  
Hello world!  
-----
```

All reboot actions executed successfully.

All the best,
the WN reboot automation script

Example for a mail upon node reboot (2)

Dear BAF admins,

a reboot action was scheduled for wn014.baf.physik.uni-bonn.de in 30 minutes
for the following reason(s):
"NEEDS_RESTARTING_REBOOTHINT"

For your information, the machine has been reboot-draining since Thu Aug 27 00:20:02 CEST
↔ 2020
(which is 337199 s ago, i.e. 3d_21h_39m_59s).

In case you want to cancel this and prevent a future
reboot, login to wn014.baf.physik.uni-bonn.de within the next 30 minutes,
and execute:

```
shutdown -c  
condor_on
```

Quickly set the inactivate-flag on profile::wn_reboot_check
for this host to true, and execute Puppet.

No reboot actions have been specified.

All the best,
the WN reboot automation script

Draining: How to implement in HTCondor

- Users set `+MaxRuntimeHours` in their jobs (added to `SIGNIFICANT_ATTRIBUTES`)

```
• START = ($(START)) && (MY.BackfillableMaxRuntimeHours >=  
  ↳ ifThenElse(isUndefined(TARGET.MaxRuntimeHours), 168,  
  ↳ TARGET.MaxRuntimeHours) )
```

⇒ Define `BackfillableMaxRuntimeHours` (variables are read in by `STARTD_CRON` jobs):

```
BackfillableMaxRuntimeHours_IMMEDIATE_DRAIN =  
↳ ifThenElse((MY.IMMEDIATE_DRAIN == True) &&  
↳ isInteger(MY.MaxLatestJobEndTime),  
↳ Max({(MY.MaxLatestJobEndTime-time())/60/60-1, 0}),168)  
BackfillableMaxRuntimeHours_TIMED_DRAIN =  
↳ ifThenElse((MY.TIMED_DRAIN == True) &&  
↳ isInteger(MY.DRAIN_TARGET_TIME),  
↳ Max({(MY.DRAIN_TARGET_TIME-time())/60/60-1, 0}),168)
```

Draining: How to implement in HTCondor

- Users set `+MaxRuntimeHours` in their jobs (added to `SIGNIFICANT_ATTRIBUTES`)

```
• START = ($(START)) && (MY.BackfillableMaxRuntimeHours >=  
  ↪ ifThenElse(isUndefined(TARGET.MaxRuntimeHours), 168,  
  ↪ TARGET.MaxRuntimeHours) )
```

Finally, merge both:

```
BackfillableMaxRuntimeHours =  
↪ Min({BackfillableMaxRuntimeHours_IMMEDIATE_DRAIN,  
↪ BackfillableMaxRuntimeHours_TIMED_DRAIN})
```

Node reboot handling in practice

- Hourly cronjob checks number of:
`condor_starter` , `jobs (condor_who)` , `condor_startd`
- No jobs and starters are found, but `condor_startd` running:
 - 1 Run `condor_off -peaceful`
 - 2 Wait up to two minutes until `condor_startd` is gone
- When 'all' are gone, check if `condor.service` is still active (might have failed!) and if it is not, alert admins
- If all is well, execute any executables from
`/etc/wn-reboot-actions`
Admins can use that to hook into the reboot process!
- Output is summarized into a mail sent to the admin mailing list, 30 min before the actual reboot / shutdown.
- Special reboot actions possible:
reboot, shutdown, notification