CERN

Monitoring HTCondor: A common one-stop solution?

Iain Steers - CERN IT

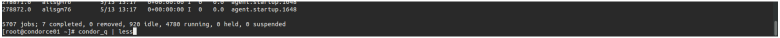
Introduction

This presentation is going to cover the various different ways of monitoring a HTCondor pool.

In an effort to solidify the best practices and build up a community around a common monitoring solution.

A Brief History of Monitoring HTCondor

We’ve all sat down and thought - ”What about the clis and unix cron?”.



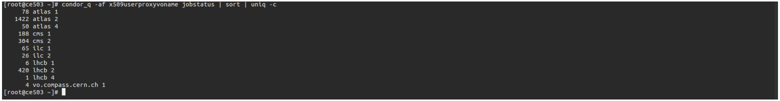
```
278871.0   alisgm76        5/13 13:17   0+00:00:00 I  0   0.0  agent.startup.1648
278872.0   alisgm76        5/13 13:17   0+00:00:00 I  0   0.0  agent.startup.1648

5707 jobs; 7 completed, 0 removed, 920 idle, 4780 running, 0 held, 0 suspended
[root@condorce01 ~]# condor_q | less
```

Figure: *One Hour Later*: Why is RecentDaemonDutyCycle so high?

A Slight Improvement

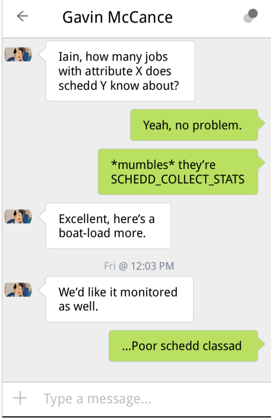
Autoformat to the rescue!



```
[root@ce503 ~]# condor_q -af x509userproxyvoname jobstatus | sort | uniq -c
     78 atlas 1
   1422 atlas 2
     50 atlas 4
    188 cms 1
    304 cms 2
     65 ilc 1
     26 ilc 2
      6 lhcb 1
    420 lhcb 2
      1 lhcb 4
      4 vo.compass.cern.ch 1
[root@ce503 ~]# 
```

Figure: A bit better, but only something you want a human to use.

Requests from Management



Gavin McCance

Iain, how many jobs with attribute X does schedd Y know about?

Yeah, no problem.

mumbles they're SCHEDD_COLLECT_STATS

Excellent, here's a boat-load more.

Fri @ 12:03 PM

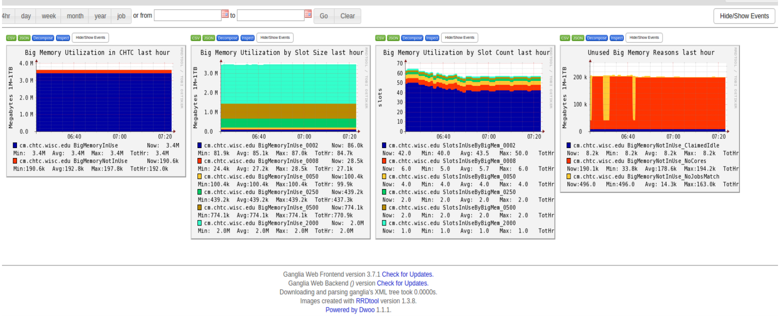
We'd like it monitored as well.

...Poor schedd classad

Figure: SCHEDD_COLLECT_STATS_BY_{BY,FOR} aggregate attributes from job matching expressions.

The Ganglia Integration

Push standard and custom metrics into a ganglia instance using the daemon GANGLIAD.



```
■ cm.chtc.wisc.edu BigMemoryInUse          Now:  3.4M
Min:  3.4M  Avg:  3.4M  Max:  3.4M  TotHr:  3.4M
■ cm.chtc.wisc.edu BigMemoryNotInUse       Now:190.6k
Min:190.6k  Avg:192.8k  Max:197.8k  TotHr:192.0k
```

```
■ cm.chtc.wisc.edu BigMemoryInUse_0002    Now: 86.0k
Min: 81.9k  Avg: 85.1k  Max: 87.0k  TotHr: 84.7k
■ cm.chtc.wisc.edu BigMemoryInUse_0008    Now: 28.5k
Min: 24.4k  Avg: 27.2k  Max: 28.5k  TotHr: 27.1k
■ cm.chtc.wisc.edu BigMemoryInUse_0050    Now:100.4k
Min:100.4k  Avg:100.4k  Max:100.4k  TotHr: 99.9k
■ cm.chtc.wisc.edu BigMemoryInUse_0250    Now:439.2k
Min:439.2k  Avg:439.2k  Max:439.2k  TotHr:437.3k
■ cm.chtc.wisc.edu BigMemoryInUse_0500    Now:774.1k
Min:774.1k  Avg:774.1k  Max:774.1k  TotHr:770.9k
■ cm.chtc.wisc.edu BigMemoryInUse_2000    Now:  2.0M
Min:  2.0M  Avg:  2.0M  Max:  2.0M  TotHr:  2.0M
```

```
■ cm.chtc.wisc.edu SlotsInUseByBigMem_0002
Now: 42.0   Min: 40.0   Avg: 43.5   Max: 50.0   TotHr
■ cm.chtc.wisc.edu SlotsInUseByBigMem_0008
Now:  6.0   Min:  5.0   Avg:  5.7   Max:  6.0   TotHr
■ cm.chtc.wisc.edu SlotsInUseByBigMem_0050
Now:  4.0   Min:  4.0   Avg:  4.0   Max:  4.0   TotHr
■ cm.chtc.wisc.edu SlotsInUseByBigMem_0250
Now:  2.0   Min:  2.0   Avg:  2.0   Max:  2.0   TotHr
■ cm.chtc.wisc.edu SlotsInUseByBigMem_0500
Now:  2.0   Min:  2.0   Avg:  2.0   Max:  2.0   TotHr
■ cm.chtc.wisc.edu SlotsInUseByBigMem_2000
Now:  1.0   Min:  1.0   Avg:  1.0   Max:  1.0   TotHr
```

```
■ cm.chtc.wisc.edu BigMemoryNotInUse_ClaimedIdle
Now:  8.2k  Min:  8.2k  Avg:  8.2k  Max:  8.2k  TotHr
■ cm.chtc.wisc.edu BigMemoryNotInUse_NoCores
Now:190.1k  Min: 33.8k  Avg:178.6k  Max:194.2k  TotHr
■ cm.chtc.wisc.edu BigMemoryNotInUse_NoJobsMatch
Now:496.0   Min:496.0   Avg: 14.3k  Max:163.0k  TotHr
```

Ganglia Web Frontend version 3.7.1 Check for Updates.
Ganglia Web Backend () version Check for Updates.
Downloading and parsing ganglia's XML tree took 0.0000s.
Images created with RRDtool version 1.3.8.
Powered by Dwoo 1.1.1.

Figure: Ganglia Web Front-end

Ganglia Competition?

GangliaD is great!
But what about large sites which already have centralized monitoring solutions?

- ELK Stack
- Influxdb and Grafana
- sysdig

MetricsD

Todd's Talk of Lies from Barcelona mentioned an incoming MetricsD.

Same metric configuration language but instead of just sending to ganglia, publishes json blobs with the monitoring samples.

MetricsD Example

```
[
  Name   = "Availability";
  Value  = int(ifThenElse(IsCritical is undefined,
                      (RecentDaemonCoreDutyCycle < .95) ||
                      (FileTransferFileReadLoad_5m < 2.0) ||
                      (FileTransferFileWriteLoad_1m < 2.0),
                      !IsCritical));
  Desc   = "Average availability of CE";
  Scale  = 100;
  Units  = "%";
  TargetType = "Scheduler";
]
```

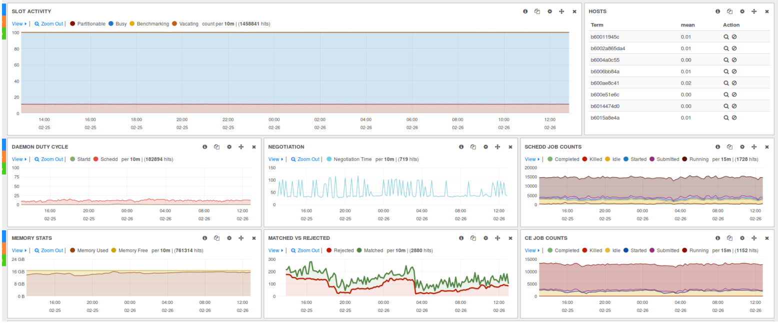
CERN

As a site, CERN already has a centralized monitoring set-up, based on ElasticSearch/Kibana. Leaving the old gangliad a bit redundant.

Pull out interesting metrics from classads/jobs using python-bindings.

So we've found ourselves using a mix of existing solutions.

Our Main Dashboard


SLOT ACTIVITY
View ▸ | Zoom Out | ● Partitionable ● Busy ● Benchmarking ● Vacating count per 10m | (1458841 hits)
100
80
60
40
20
0
14:00 02-25
16:00 02-25
18:00 02-25
20:00 02-25
22:00 02-25
00:00 02-26
02:00 02-26
04:00 02-26
06:00 02-26
08:00 02-26
10:00 02-26
12:00 02-26
HOSTS
DAEMON DUTY CYCLE
View ▸ | Zoom Out | ● Startd ● Schedd per 10m | (182894 hits)
100
75
50
25
0
NEGOTIATION
View ▸ | Zoom Out | ● Negotiation Time per 10m | (719 hits)
150
100
50
0
SCHEDD JOB COUNTS
View ▸ | Zoom Out | ● Completed ● Killed ● Idle ● Started ● Submitted ● Running per 15m | (1728 hits)
20000
15000
10000
5000
0
MEMORY STATS
View ▸ | Zoom Out | ● Memory Used ● Memory Free per 10m | (761314 hits)
24 GB
16 GB
8 GB
0 B
MATCHED VS REJECTED
View ▸ | Zoom Out | ● Rejected ● Matched per 10m | (2880 hits)
300
200
100
0
CE JOB COUNTS
View ▸ | Zoom Out | ● Completed ● Killed ● Idle ● Started ● Submitted ● Running per 15m | (1152 hits)
15000
10000
5000
0
16:00 02-25
20:00 02-25
00:00 02-26
04:00 02-26
08:00 02-26
12:00 02-26

Term	mean	Action
b60011945c	0.01	🔍 ⊘
b6002a865da4	0.01	🔍 ⊘
b6004a0c55	0.00	🔍 ⊘
b6006bb84a	0.01	🔍 ⊘
b600ae8c41	0.02	🔍 ⊘
b600e51e6c	0.00	🔍 ⊘
b6014474d0	0.00	🔍 ⊘
b6015a8e4a	0.01	🔍 ⊘

Figure: The main pool health dashboard

Our Draining Dashboard

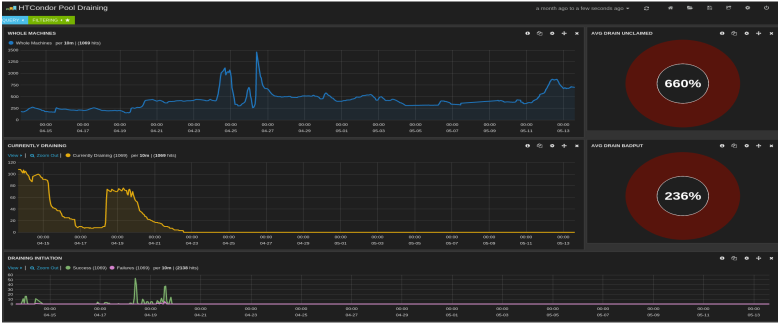


Figure: Track the multi-core draining of the pool and wasted cpu over time.

Our Cgroups Monitoring

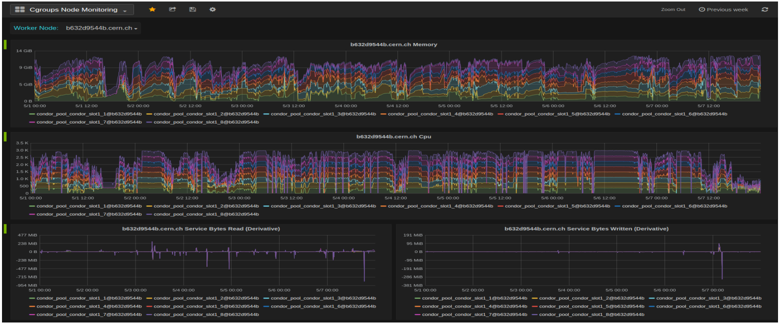

Cgroups Node Monitoring
Zoom Out
Previous week
Worker Node: b632d9544b.cern.ch
b632d9544b.cern.ch Memory
14 GiB
9 GiB
5 GiB
0 B
5/1 00:00
5/1 12:00
5/2 00:00
5/2 12:00
5/3 00:00
5/3 12:00
5/4 00:00
5/4 12:00
5/5 00:00
5/5 12:00
5/6 00:00
5/6 12:00
5/7 00:00
5/7 12:00
condor_pool_condor_slot1_1@b632d9544b
condor_pool_condor_slot1_2@b632d9544b
condor_pool_condor_slot1_3@b632d9544b
condor_pool_condor_slot1_4@b632d9544b
condor_pool_condor_slot1_5@b632d9544b
condor_pool_condor_slot1_6@b632d9544b
condor_pool_condor_slot1_7@b632d9544b
condor_pool_condor_slot1_8@b632d9544b
b632d9544b.cern.ch Cpu
3.5 K
3.0 K
2.5 K
2.0 K
1.5 K
1.0 K
500
0
b632d9544b.cern.ch Service Bytes Read (Derivative)
477 MiB
238 MiB
0 B
-238 MiB
-477 MiB
-715 MiB
-954 MiB
b632d9544b.cern.ch Service Bytes Written (Derivative)
191 MiB
95 MiB
0 B
-95 MiB
-191 MiB
-286 MiB
-381 MiB

Figure: Cgroups Monitoring of Jobs

Too Many Dashboards

Too Many Dashboards and different Systems.
(ELK/Influx/Grafana/Spark/Jupyter)

#MonitoringSucks

The Problem

This doesn't seem to be a problem just at CERN.

Looking at the HTCondor community it seems everyone has done their own thing for monitoring.

Siloing knowledge, implementations and best practices.

Conclusion

Could we do better?

Could we bring together a community around monitoring HTCondor?

Get the Python, HTCondor experts and data junkies in a (virtual-)room together and come up with a common platform, to really reveal the health of your pool?

Questions?

Any Questions?

www.cern.ch