Monitoring Primer

European HTCondor Workshop 2017

Todd Tannenbaum
Center for High Throughput Computing
University of Wisconsin-Madison

Ad types in the condor_collector

- *startd* ads
  - An ad for each slot on each machine
  - *State* = Unclaimed | Claimed | Drained …
  - *Activity* = Busy | Idle | Retiring | Vacating …
  - CPUs, Memory, Disk, GPUs, …
- *submitter ads*
  - An ad for each unique user
  - RunningJobs, IdleJobs, HeldJobs, …
- *schedd, master, negotiator, collector* ads
  - One ad from each daemon instance

Q: How many slots are running a job?

A: Count slots where State == Claimed (and Activity != Idle)

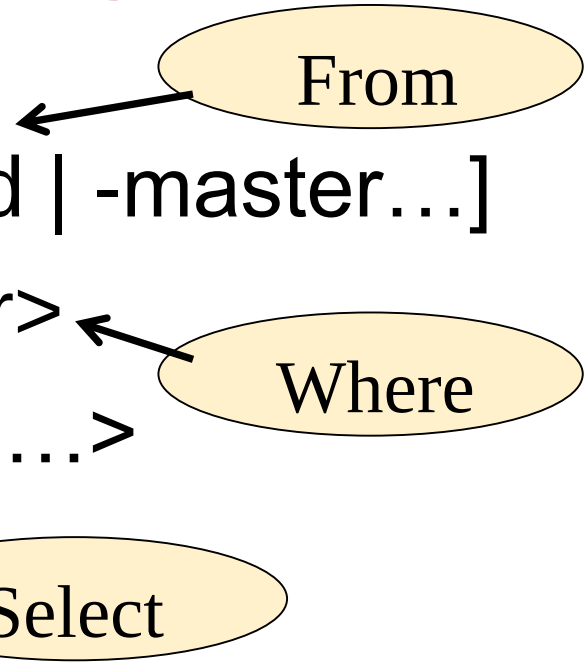
How?

Obvious solutions aren't the best

```
% condor_status
slot7@ale-22.cs.wi LINUX      X86_64 Claimed   Busy      0.990 3002  0+00:28:24
slot8@ale-22.cs.wi LINUX      X86_64 Claimed   Busy      1.000 3002  0+00:14:13
slot1@ale-23.cs.wi LINUX      X86_64 Unclaimed Idle      0.920 3002  0+00:00:04
…
```

- condor_status | grep Claimed | grep –v Idle | wc –l
  - Output subject to change, wrong answers, slow
- condor_status –l | grep Claimed | wc –l
  - Wrong answers, really slow

Use constraints and projections



- condor_status [-startd | -schedd | -master…]
  - –constraint <classad-expr>
  - –autoformat <attr1, attr2, …>

```
condor_status -startd      \
   -cons 'State=="Claimed" && Activity!="Idle"'    \
   -af name  | wc -l
```

Q: Which slots are running on machines where NFS is broken?

- Ask startd to run a script/program to test health of NFS

```
STARTD_CRON_JOB_LIST = tag
STARTD_CRON_tag_EXECUTABLE = detect.sh
```

  - Script returns a ClassAd with attribute NFS_Broken = True | False
- condor_status -cons 'NFS_Broken==True'
- Could specify customized output (i.e. a report) for condor_status to display broken machines

https://htcondor-wiki.cs.wisc.edu/index.cgi/wiki?p=ExperimentalCustomPrintFormats

Q: How many CPU cores are being utilized?

- Sum the Cpus attribute for each slot that is Claimed and Busy:

```
% condor_status -startd      \
    -cons 'State=="Claimed" && Activity!="Idle"'    \
    -af Cpus | less
1
1
4
4
…
% condor_status -startd      \
    -cons 'State=="Claimed" && Activity!="Idle"'    \
    -af Cpus | st
N        min      max      sum      mean     stddev
9053     1        40       10410    1.1499   1.39239
```

Simple Statistics from command line
https://github.com/nferraz/st

Graph of CPU utilization over time

- Could have a cron job run every minute…

```
#!/bin/sh
echo `date`, ; condor_status \
-cons 'State=="Claimed" && Activity!="Idle"'   \
-af Cpus | st --sum
```

- What if you have hundreds or thousands of metrics?
- Better idea: How about query the collector just once per minute for all attributes needed to compute all metrics?

Ganglia and condor_gangliad

- condor_gangliad queries the condor_collector once per minute
  - DAEMON_LIST = MASTER, GANGLIAD,…
- condor_gangliad has config file to *filter and aggregate* attributes from the ads in the condor_collector in order to form *metrics*
- Forwards these metrics to Ganglia, which stores these values in a database and provides graphs over the web

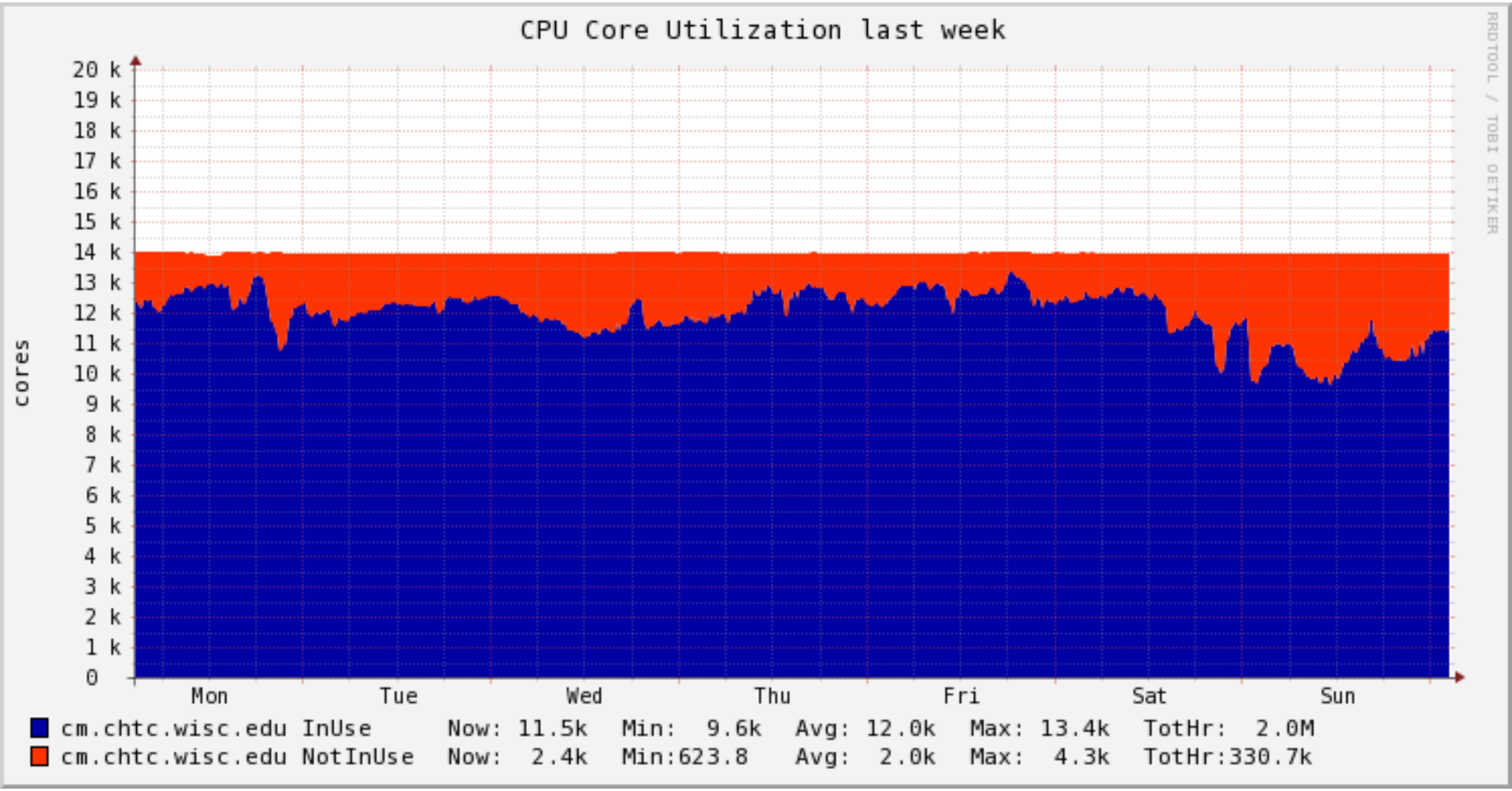
Example metric definitions in condor_gangliad

```
[
  Name   = "CpusInUse";
  Aggregate = "SUM";
  Value  = Cpus;
  Requirements = State=="Claimed" && Activity!="Idle";
  TargetType = "Machine";
]
[
  Name   = "CpusNotInUse";
  Aggregate = "SUM";
  Value  = Cpus;
  Requirements = State!="Claimed" || Activity=="Idle";
  TargetType = "Machine";
]
```

Add a graph to a view dashboard: /var/lib/ganglia/view_Miron.json

```
{ "aggregate_graph":"true",
  "host_regex":[
     {"regex":"cm.chtc.wisc.edu"}
  ],
  "metric_regex":[
     {"regex":"(Cpus(InUse|NotInUse)"}
  ],
  "graph_type":"stack",
  "vertical_label":"cores",
  "title":"CPU Core Utilization"
}
```

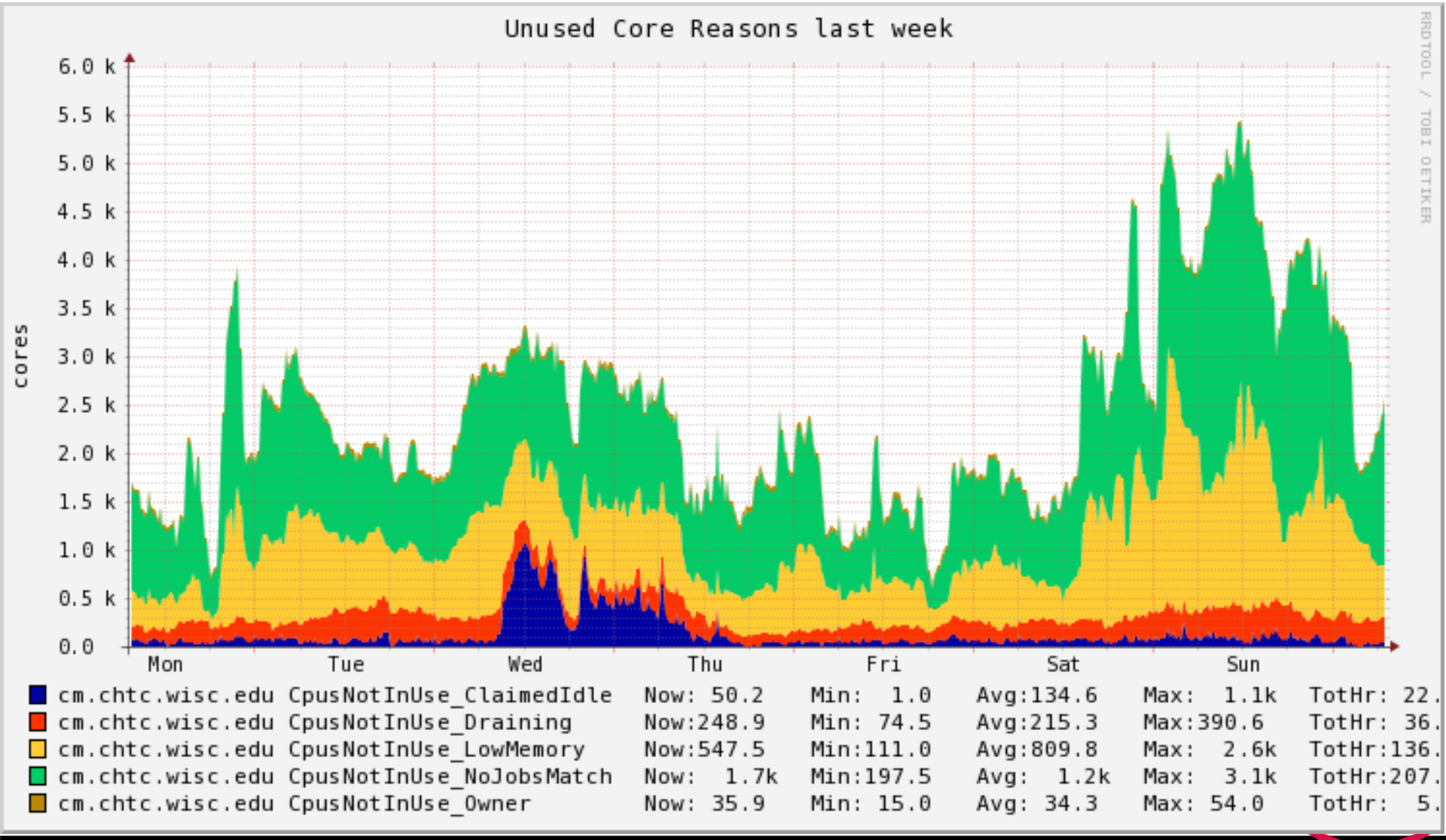
Voila!


CPU Core Utilization last week
RRDTOOL / TOBI OETIKER
cores
20 k
19 k
18 k
17 k
16 k
15 k
14 k
13 k
12 k
11 k
10 k
9 k
8 k
7 k
6 k
5 k
4 k
3 k
2 k
1 k
0
Mon
Tue
Wed
Thu
Fri
Sat
Sun
cm.chtc.wisc.edu InUse Now: 11.5k Min: 9.6k Avg: 12.0k Max: 13.4k TotHr: 2.0M
cm.chtc.wisc.edu NotInUse Now: 2.4k Min:623.8 Avg: 2.0k Max: 4.3k TotHr:330.7k

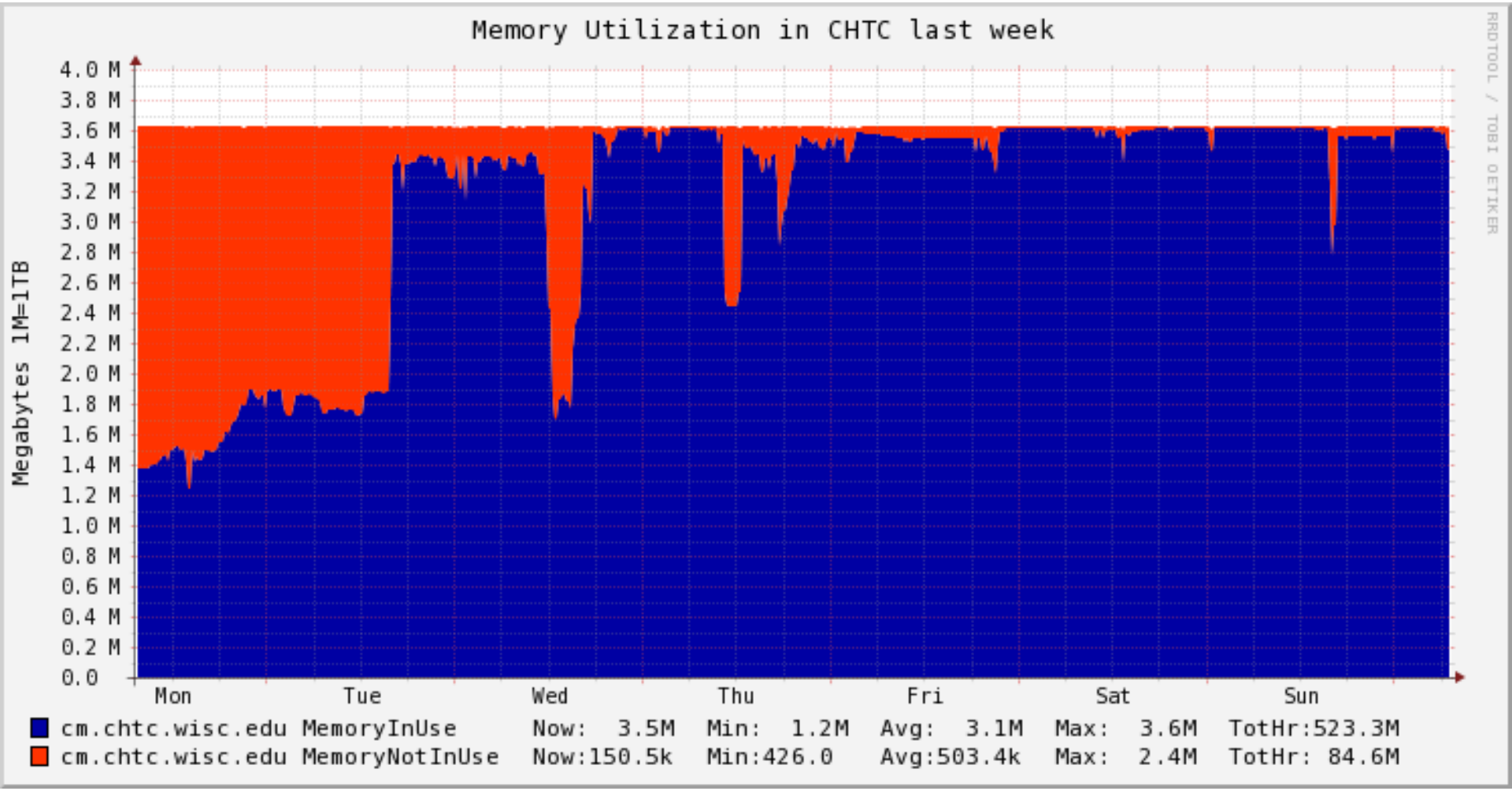
Why are cores not in use?

```
[
  Name   = "CpusNotInUse_LowMemory";
  Aggregate = "SUM";
  Value  = Cpus;
  Requirements = State=="Unclaimed" && Memory < 1024;
  TargetType = "Machine";
]
[
  Name   = "CpusNotInUse_Draining";
  Aggregate = "SUM";
  Value  = Cpus;
  Requirements = State=="Drained";
  TargetType = "Machine";
]
```

Unused Core Reasons



Memory Provisioned



Memory Used vs Provisioned

In condor_config.local :

```
STARTD_JOB_EXPRS =
        $(START_JOB_EXPRS) MemoryUsage
```

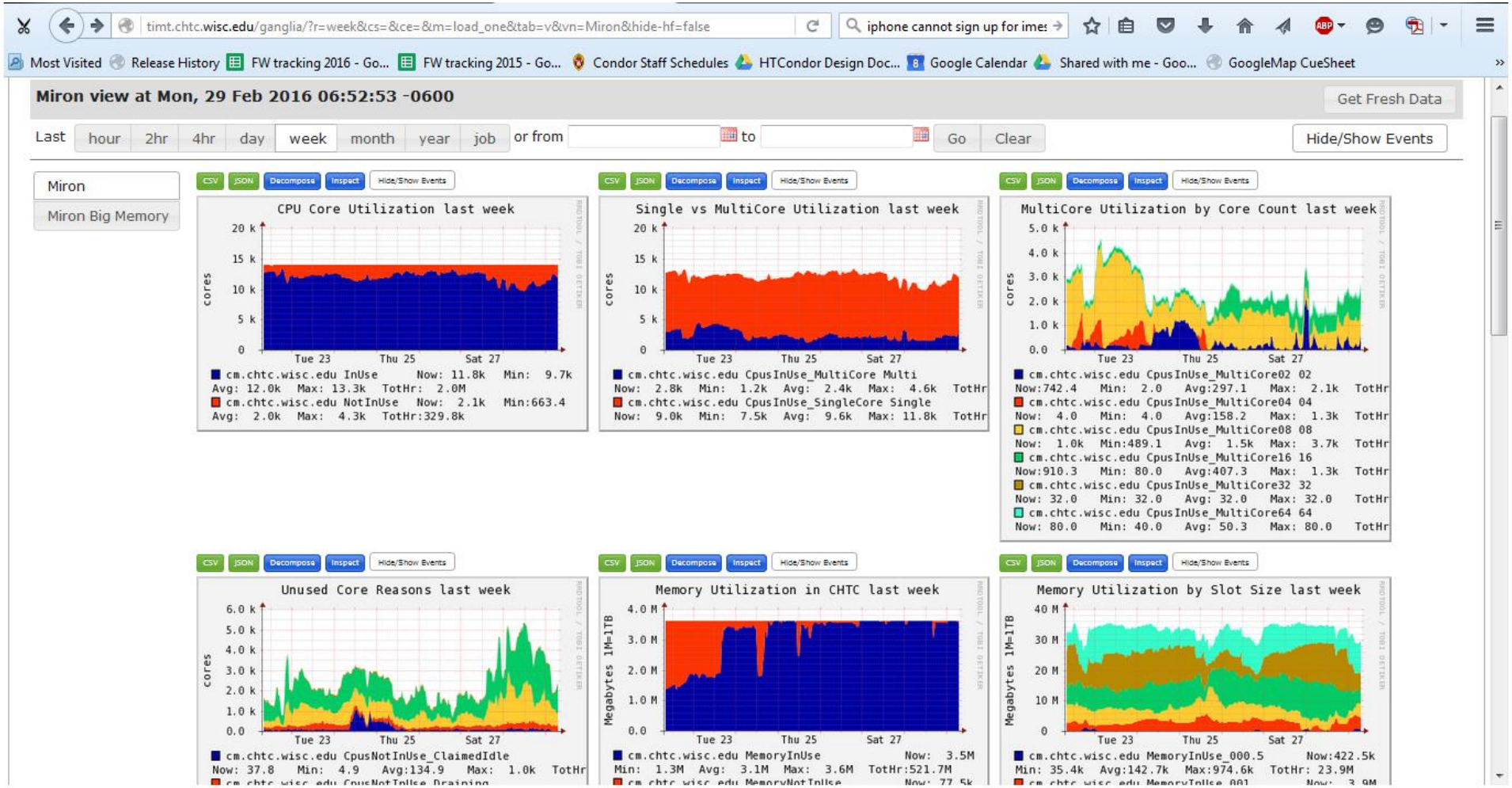
Then define MemoryEfficiency metric as:

```
[
  Name   = "MemoryEfficiency";
  Aggregate = "AVG";
  Value  = real(MemoryUsage)/Memory*100;
  Requirements = MemoryUsage > 0.0;
  TargetType = "Machine";
]
```

Example: Metrics Per User

```
[
  Name   = strcat(RemoteUser,"-UserMemoryEfficiency");
  Title  = strcat(RemoteUser," Memory Efficiency");
  Aggregate = "AVG";
  Value  = real(MemoryUsage)/Memory*100;
  Requirements = MemoryUsage > 0.0;
  TargetType = "Machine";
]
```

Dashboard(s) of useful charts



New Hotness: Grafana

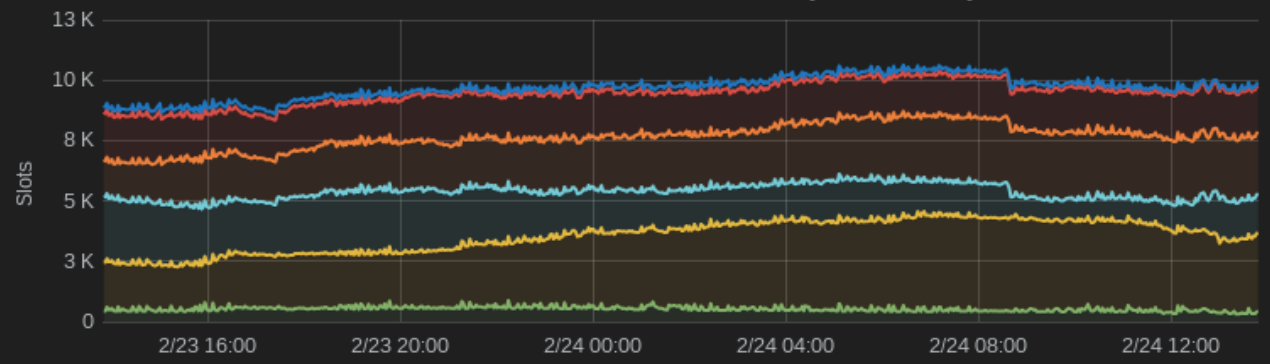
- Grafana
  - Open Source
  - Makes pretty and interactive dashboards from popular backends including *Graphite's Carbon*, Influxdb, and very recently ElasticSearch
  - Easy for individual users to create their own custom persistent graphs and dashboards
- condor_gangliad -> ganglia -> graphite

```
# gmetad.conf - Forward metrics to Carbon via UDP
carbon_server "mongodbtest.chtc.wisc.edu"
carbon_port 2003
carbon_protocol udp
graphite_prefix "ganglia"
```

Grafana - Miron

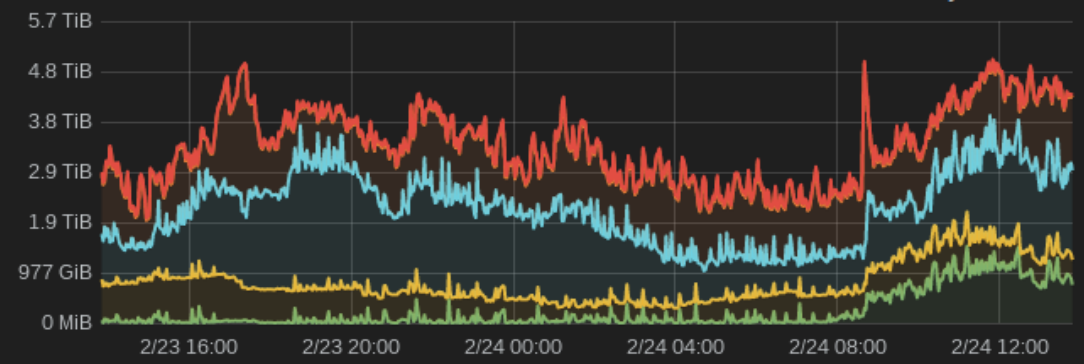
mongodbtest.chtc.wisc.edu/grafana/dashboard/db/miron

Memory Utilization by Slot Count



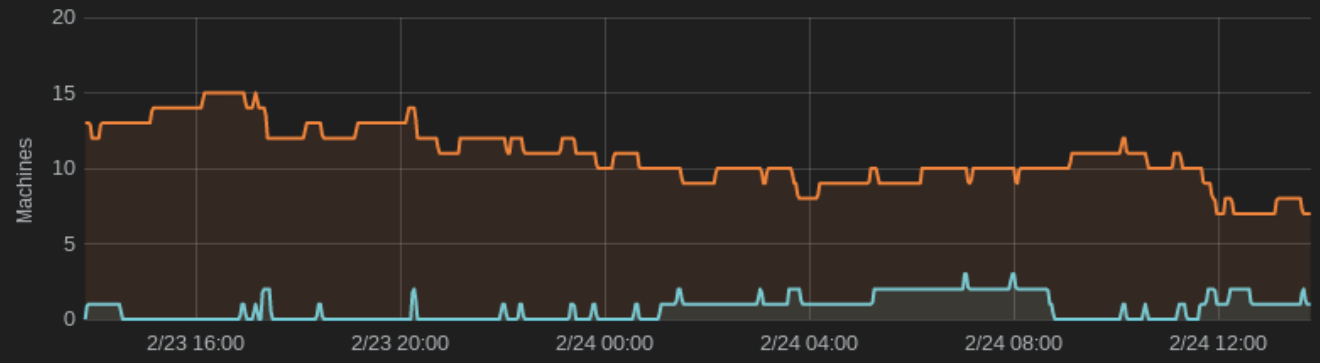
	min	max	avg	current
000.5	320	880	526	419
001	1.83 K	3.98 K	3.04 K	3.25 K
002	813	2.72 K	1.81 K	1.63 K
004	1.45 K	2.80 K	2.29 K	2.56 K
008	1.56 K	2.06 K	1.80 K	1.90 K
999	129	315	223	134

Unused Memory Reasons



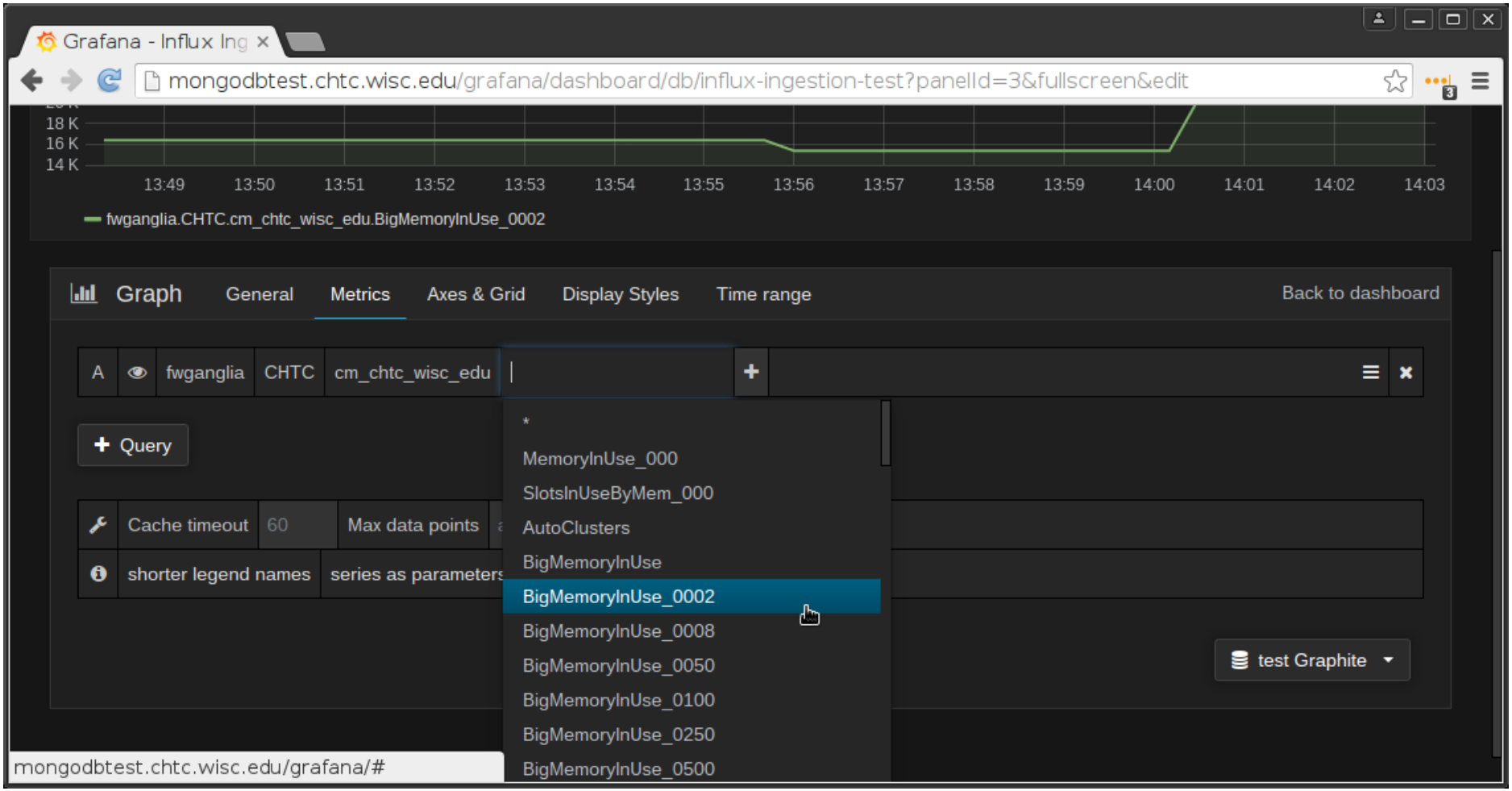
	min	max	avg	current
Claimed Idle	18 MiB	1.456 TiB	230 GiB	740 GiB
Draining	276 GiB	921 GiB	554 GiB	483 GiB
No Cores	542 GiB	3.016 TiB	1.403 TiB	1.702 TiB
No Jobs Match	203 GiB	3.476 TiB	1.166 TiB	1.415 TiB
Owner	35 GiB	51 GiB	44 GiB	47 GiB

Draining

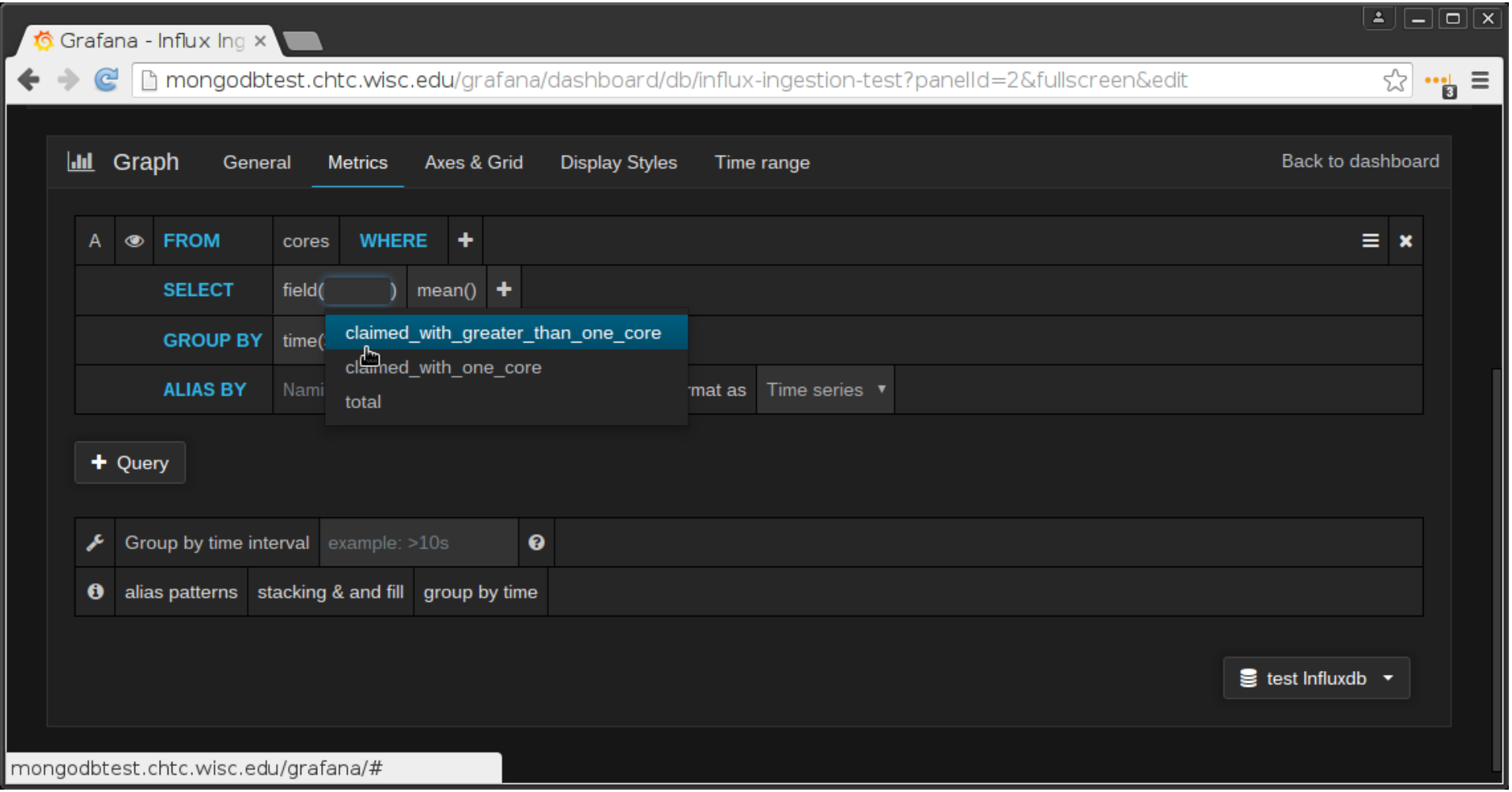


	min	max	avg	current
Arrival Mean				
Arrival Std Dev				
Whole	0	3	1	1
Draining	7	15	11	7

Adding Grafana graph (Graphite)



Adding Grafana graph (Influxdb)



What sort of attributes are avail?

- Lots of good attributes in the collector by default; browse via
  - condor_status -schedd -l,
  - condor_status -submitter -l
  - condor_status -startd -l
- Lots more available via HTCondor "Statistics"
  - Especially in the schedd, collector
  - condor_status –direct –schedd –statistics all:2 <name>
  - Send to the collector via knobs STATISTICS_TO_PUBLISH and STATISTICS_TO_PUBLISH_LIST
  - All kinds of output, mostly aggregated
  - See TJ or Manual for details

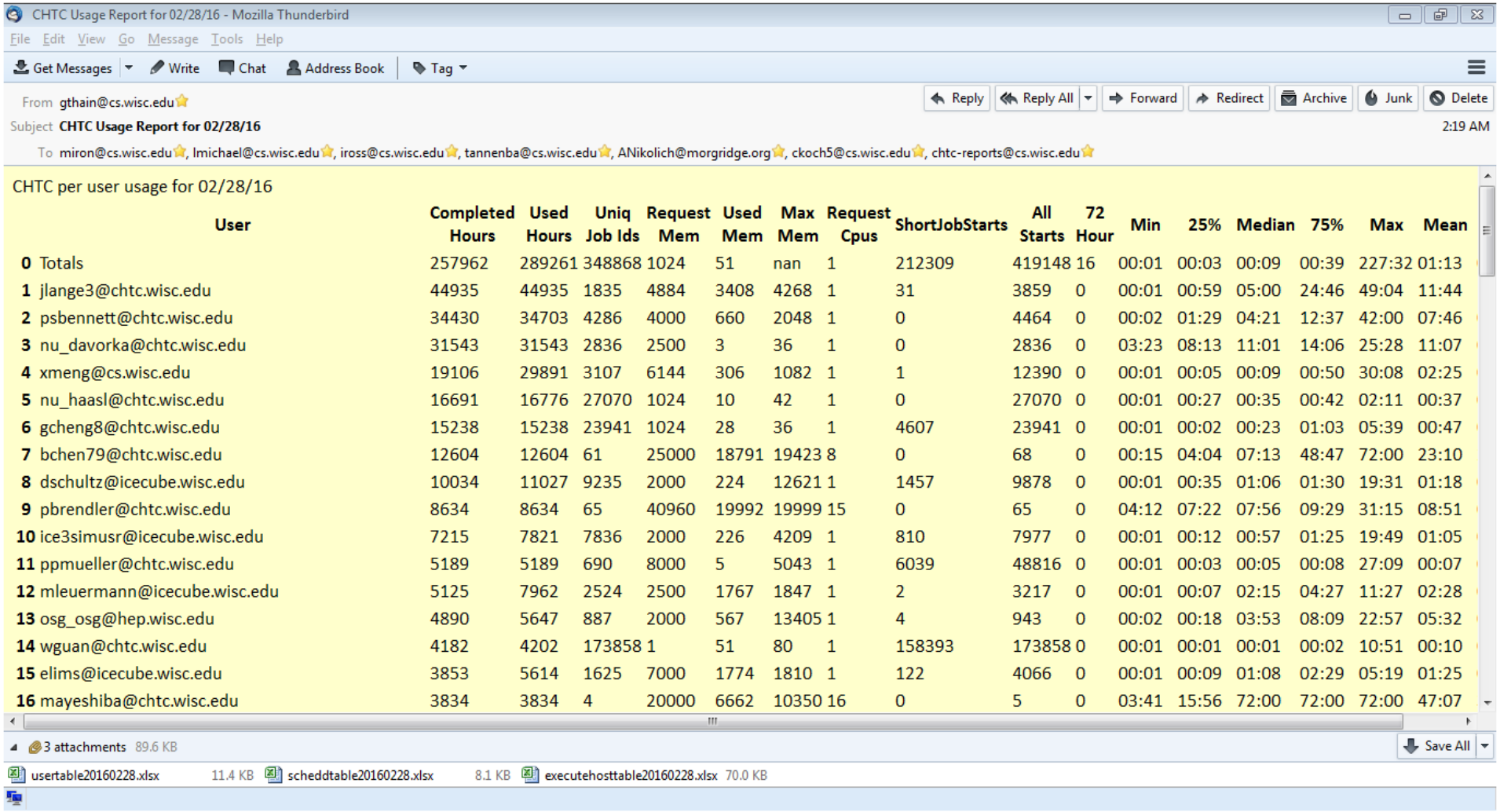
RecentDaemonCoreDutyCycle

- Todd's favorite statistic for watching the health of submit points (schedds) and central manager (collector)
- Measures time not idle
- If goes 98%, your schedd or collector is saturated

Individual Job Monitoring

- Schedd Event Log (rotates)
  - Union of all job event logs for all jobs on a schedd
  - Config Knob: `EVENT_LOG = /some/file`
- Audit Log (rotates)
  - Provenance of modifications to any job
  - Config Knob: `SCHEDD_AUDIT_LOG = /file`
- History File (rotates)
  - Schedd history : all job ads that left the queue
    - `HISTORY = /file`
  - Startd history : all job ads that used a slot
    - `STARTD_HISTORY = /file`
  - View with condor_history (local or remote)

condor_pool_job_report



CHTC Usage Report for 02/28/16 - Mozilla Thunderbird

File Edit View Go Message Tools Help

Get Messages | Write | Chat | Address Book | Tag

From gthain@cs.wisc.edu

Reply | Reply All | Forward | Redirect | Archive | Junk | Delete

Subject **CHTC Usage Report for 02/28/16** 2:19 AM

To miron@cs.wisc.edu, lmichael@cs.wisc.edu, iross@cs.wisc.edu, tannenba@cs.wisc.edu, ANikolich@morgridge.org, ckoch5@cs.wisc.edu, chtc-reports@cs.wisc.edu

CHTC per user usage for 02/28/16

	User	Completed Hours	Used Hours	Uniq Job Ids	Request Mem	Used Mem	Max Mem	Request Cpus	ShortJobStarts	All Starts	72 Hour	Min	25%	Median	75%	Max	Mean
0	Totals	257962	289261	348868	1024	51	nan	1	212309	419148	16	00:01	00:03	00:09	00:39	227:32	01:13
1	jlange3@chtc.wisc.edu	44935	44935	1835	4884	3408	4268	1	31	3859	0	00:01	00:59	05:00	24:46	49:04	11:44
2	psbennett@chtc.wisc.edu	34430	34703	4286	4000	660	2048	1	0	4464	0	00:02	01:29	04:21	12:37	42:00	07:46
3	nu_davorka@chtc.wisc.edu	31543	31543	2836	2500	3	36	1	0	2836	0	03:23	08:13	11:01	14:06	25:28	11:07
4	xmeng@cs.wisc.edu	19106	29891	3107	6144	306	1082	1	1	12390	0	00:01	00:05	00:09	00:50	30:08	02:25
5	nu_haasl@chtc.wisc.edu	16691	16776	27070	1024	10	42	1	0	27070	0	00:01	00:27	00:35	00:42	02:11	00:37
6	gcheng8@chtc.wisc.edu	15238	15238	23941	1024	28	36	1	4607	23941	0	00:01	00:02	00:23	01:03	05:39	00:47
7	bchen79@chtc.wisc.edu	12604	12604	61	25000	18791	19423	8	0	68	0	00:15	04:04	07:13	48:47	72:00	23:10
8	dschultz@icecube.wisc.edu	10034	11027	9235	2000	224	12621	1	1457	9878	0	00:01	00:35	01:06	01:30	19:31	01:18
9	pbrendler@chtc.wisc.edu	8634	8634	65	40960	19992	19999	15	0	65	0	04:12	07:22	07:56	09:29	31:15	08:51
10	ice3simusr@icecube.wisc.edu	7215	7821	7836	2000	226	4209	1	810	7977	0	00:01	00:12	00:57	01:25	19:49	01:05
11	ppmueller@chtc.wisc.edu	5189	5189	690	8000	5	5043	1	6039	48816	0	00:01	00:03	00:05	00:08	27:09	00:07
12	mleuermann@icecube.wisc.edu	5125	7962	2524	2500	1767	1847	1	2	3217	0	00:01	00:07	02:15	04:27	11:27	02:28
13	osg_osg@hep.wisc.edu	4890	5647	887	2000	567	13405	1	4	943	0	00:02	00:18	03:53	08:09	22:57	05:32
14	wguan@chtc.wisc.edu	4182	4202	173858	1	51	80	1	158393	173858	0	00:01	00:01	00:01	00:02	10:51	00:10
15	elims@icecube.wisc.edu	3853	5614	1625	7000	1774	1810	1	122	4066	0	00:01	00:09	01:08	02:29	05:19	01:25
16	mayeshiba@chtc.wisc.edu	3834	3834	4	20000	6662	10350	16	0	5	0	03:41	15:56	72:00	72:00	72:00	47:07

3 attachments 89.6 KB Save All

usertable20160228.xlsx 11.4 KB scheddtable20160228.xlsx 8.1 KB executehosttable20160228.xlsx 70.0 KB

Upcoming

- condor_gangliad → condor_metricd
  - Send aggregated metrics to Ganglia
  - Write out aggregated metrics to rotating JSON files
  - Send aggregated metrics to Graphite / Influx
- A new "HTCondor View" tool
  - Some basic utilization graphs out-of-the-box

Check out Fifemon!

"Comprehensive grid monitoring with Fifemon has improved resource utilization, job throughput, and computing visibility at Fermilab"

- Probes, dashboards, and docs at:

  https://github.com/fifemon

- Fifemon Overview talk from HTCondor Week 2016:

https://research.cs.wisc.edu/htcondor/HTCondorWeek2016/presentations/ThuRetzke_Fifemon.pdf

Thank you!