Comprehensive Grid and Job Monitoring with Fifemon

Kevin Retzke
User Support for Distributed Computing @ Fermilab
HTCondor Week 2016
FIFE Project

FabrIc for Frontier Experiments:

Common computing for “not CMS” experiments at Fermilab

- O(10) experiments
- O(100) users
- O(10 000) simultaneous jobs
- O(1 000 000) jobs per week
- O(1 PB) data collected per month
- One global HTCondor pool (via GlideinWMS)
  - ~⅔ jobs run on dedicated local cluster
  - ~⅓ opportunistic through Open Science Grid
Why Do We Need Monitoring?

Grid admins want to know:

- Overall health of the batch system
- Worker node status and availability
- Efficiency in matching jobs to resources
- Identify and fix problems quickly (before users and stakeholders notice... and open tickets)

Users want to know:

- State of their jobs
- Availability of resources
- WHY ISN’T MY JOB RUNNING?

Stakeholders want to know:

- Each group is getting the resources it needs
- Resources are being used effectively
Fermigrid Monitor (ca. 2004)

Monitoring for local HTCondor cluster (GPGrid).

- Aggregate metrics for grid and VOs.
- No offsite information, no user job information.
- Difficult to alter or expand.

OK for grid admins, good for stakeholders, bad for users.
Fifemon v1 (ca. 2014)

Growing usage of offsite resources through OSG; needed new monitoring.

- Aggregate metrics for users and VOs.
- No cluster information.
- Cumbersome to maintain and expand.

OK for grid admins, bad for stakeholders, good for users.
Fifemon v2+ (ca. 2015)

Landscape Program: develop comprehensive monitoring for FIFE, HEP Cloud, and beyond.

- Leverage open-source monitoring technology
- Focus on incorporating new data sources and new dashboards
- Rapid development and iteration of tailored views for each target audience.

Good for grid admins, stakeholders, and users alike.
Fifemon Backend

Data collection:

- Generic HTCondor probe collecting daemon, machine, and job status
- Logstash collecting live HTCondor Events
- Several other centrally-run probes querying other resources
- Some services directly reporting to Graphite

- Most probes report stats every five minutes
- Graphite:
  - 250K individual metrics
  - ~80GB
  - 10 year history
- Elasticsearch: ~8GB per day

Graphite:

- Time-series database, stores data in files similar to RRD with caching layer.
- Simple line protocol
- Powerful query manipulations and aggregations

Elasticsearch:

- “NoSQL” document database, powered by Apache Lucene.
- Store full details on current jobs, batch slots, and logs.
Fifemon Frontend

Grafana:

- Time-series (primarily) visualization dashboard platform.
- Supports numerous data sources (Graphite, InfluxDB, Elasticsearch, etc).
- Several auth methods (LDAP, OAuth, proxy).
- Rich user interface for graphing metrics and composing dashboards.
- Scripted and templated dashboards and raw HTML panels allow extensive customization.
- V3 (released last week) introduces new plugin system to support custom datasources and panels.

Kibana:

- Elasticsearch data only
  - Current jobs and machine status
  - Event logs
- Explore data, create ad-hoc visualizations, combine into dashboards
- Used for analytics and troubleshooting
- Access limited to grid admins and power users
Fifemon Architecture

Logstash & Probes
Collection:
- Job Details
- Slot Details
- System Metrics
- Event Logs

Elasticsearch

Graphite

Grafana

Kibana

Fifebatch
GPGrid
CMS Tier 1
CMS LPC
HEP Cloud

Data handling
- dCache
- BlueArc
- Postgres
- more...

Time-series Aggregations

Raw Documents
Next Steps

Fifemon is constantly evolving:

- Adding new data sources and metrics
- New dashboards:
  - Tailored views based on user request
  - Discovering new ways of looking at the data
- New Grafana panels
- Further leverage HTCondor event logs & gangliad/metricsd for true real-time monitoring
Case Studies

“There’s a dashboard for that...”
Case Study: Grid Admin

“Is the batch system healthy?”

Cinnamon

Photo: Public Domain
Woof, all green!
Throughput is Good.
Grid utilization is OK.
Let's check anyways... what happened here?
Hmm, we couldn’t query a CE for a few minutes. I’ll check the probe logs.
Case Study: Stakeholder

“Is my experiment getting the resources it needs and using them effectively?”

Photo: Claudio Gennari (CC-BY-2.0)
We’re well above our quota, but efficiency could be better.
Grrr… time to send some emails! Let’s check on resource requests.
Disk and Memory requests look good, lots of users exceeding request time though.

<table>
<thead>
<tr>
<th>User</th>
<th>I</th>
<th>R</th>
<th>C</th>
<th>X</th>
<th>H</th>
<th>Max Memory/Request</th>
<th>Max Disk/Request</th>
<th>Max Time/Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>anorman</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9</td>
<td>0.78</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>arrita1</td>
<td>100</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>bianimin</td>
<td>825</td>
<td>2506</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0.37</td>
<td>0.00</td>
<td>0.73</td>
</tr>
<tr>
<td>boyd</td>
<td>50</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>brebel</td>
<td>0</td>
<td>1</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>3.27</td>
</tr>
<tr>
<td>crisprin</td>
<td>0</td>
<td>3</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0.01</td>
<td>0.00</td>
<td>8.55</td>
</tr>
<tr>
<td>dmerdez</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>6</td>
<td>1.00</td>
<td>0.01</td>
<td>0.00</td>
</tr>
<tr>
<td>khamer</td>
<td>4</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>kratzke</td>
<td>1</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>kudeepm</td>
<td>0</td>
<td>10</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0.34</td>
<td>0.00</td>
<td>6.07</td>
</tr>
<tr>
<td>loremone</td>
<td>0</td>
<td>2</td>
<td></td>
<td>0</td>
<td></td>
<td>0.29</td>
<td>0.13</td>
<td>5.54</td>
</tr>
<tr>
<td>novapro</td>
<td>22154</td>
<td>3464</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>1.05</td>
<td>1.01</td>
<td>12.04</td>
</tr>
<tr>
<td>painer219</td>
<td>0</td>
<td>11</td>
<td></td>
<td>0</td>
<td></td>
<td>0.95</td>
<td>0.11</td>
<td>0.00</td>
</tr>
<tr>
<td>siva1987</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.66</td>
<td>0.00</td>
<td>3.39</td>
</tr>
</tbody>
</table>
Otherwise job throughput is good both onsite and offsite.
Case Study: User

“What’s the status of my jobs?”

Cocoa
Yay my Jobs are starting, but my efficiency is dropping!
This cluster has poor efficiency, let's take a look at it.

<table>
<thead>
<tr>
<th>Cluster</th>
<th>I</th>
<th>R</th>
<th>H</th>
<th>Submit Time/Command</th>
<th>Memory (MB)</th>
<th>Disk (MB)</th>
<th>Time (hr)</th>
<th>Max Eff.</th>
<th>Starts</th>
</tr>
</thead>
<tbody>
<tr>
<td>7898120</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>2016-03-06T02:22:51.000Z qhuang-qhuang-reco-koppin-Orfite-3000-S16-03-04-neardet-BNL-28_days_ago-20160308_022251_3281177_0_1_wrap.sh</td>
<td>0 / 3000</td>
<td>0 / 10240</td>
<td>0 / 11</td>
<td>----</td>
<td>0</td>
</tr>
<tr>
<td>7898126</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>2016-03-06T02:23:06.000Z qhuang-qhuang-reco-koppin-Orfite-3000-S16-03-04-neardet-BNL-27_days_ago-20160308_022306_3282245_0_1_wrap.sh</td>
<td>0 / 3000</td>
<td>0 / 10240</td>
<td>0 / 11</td>
<td>----</td>
<td>0</td>
</tr>
<tr>
<td>7898131</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>2016-03-06T02:23:18.000Z qhuang-qhuang-reco-koppin-Orfite-3000-S16-03-04-neardet-BNL-29_days_ago-20160308_023118_3293095_0_1_wrap.sh</td>
<td>0 / 3000</td>
<td>0 / 10240</td>
<td>0 / 11</td>
<td>----</td>
<td>0</td>
</tr>
<tr>
<td>7898137</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>2016-03-06T02:23:30.000Z qhuang-qhuang-reco-koppin-Orfite-3000-S16-03-04-neardet-BNL-31_days_ago-20160308_023300_3294093_0_1_wrap.sh</td>
<td>0 / 3000</td>
<td>0 / 10240</td>
<td>0 / 11</td>
<td>----</td>
<td>0</td>
</tr>
<tr>
<td>7998802</td>
<td>0</td>
<td>244</td>
<td>0</td>
<td>2016-03-08T04:58:24.000Z brazzom-med_full_chain_1R18-03-03-prodGenie_a_ND_numl_epoch30-20160308_0488.sh_20160308_048824_3734286_0_1_wrap.sh</td>
<td>1952 / 2000</td>
<td>451 / 34180</td>
<td>0 / 6</td>
<td>62.3%</td>
<td>2</td>
</tr>
<tr>
<td>7993210</td>
<td>1719</td>
<td>0</td>
<td>0</td>
<td>2016-03-08T08:05:35.000Z tghosh-prod_daq_1R16-02-11-prodGenie_b_fd_genie nonswap_fho_nova_v10_full_batch1_v1_1b4amod3-20160308_0806.sh_20160308_080638_4016786_0_1_wrap.sh</td>
<td>13533 / 2000</td>
<td>2178 / 4000</td>
<td>2 / 3</td>
<td>36.6%</td>
<td>1</td>
</tr>
<tr>
<td>4293778</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2016-03-08T12:32:00.000Z brazzom-med_full_chain_1R18-03-03-prodGenie_a_ND_numl_epoch30-20160308_022251_3281177_0_1_wrap.sh</td>
<td>1925 / 2000</td>
<td>114 / 34180</td>
<td>1 / 6</td>
<td>57.5%</td>
<td>1</td>
</tr>
<tr>
<td>4937313</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2016-03-08T12:32:00.000Z brazzom-med_full_chain_1R18-03-03-prodGenie_a_ND_numl_epoch30-20160308_022251_3281177_0_1_wrap.sh</td>
<td>1929 / 2000</td>
<td>79 / 34180</td>
<td>1 / 6</td>
<td>55.2%</td>
<td>1</td>
</tr>
<tr>
<td>4937313</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2016-03-08T12:32:00.000Z brazzom-med_full_chain_1R18-03-03-prodGenie_a_ND_numl_epoch30-20160308_022251_3281177_0_1_wrap.sh</td>
<td>1920 / 2000</td>
<td>85 / 34180</td>
<td>1 / 6</td>
<td>53.2%</td>
<td>1</td>
</tr>
</tbody>
</table>
A few failed processes, and a bunch are disconnected.
Some of my jobs are held… I need to increase memory request.
Case Study: Upper Management

“What does the computing division do again?”

Sage (and minions)
Fermilab Scientific Computing Summary
FIFE

Average Number of Jobs Running Concurrently
13899

Total Jobs Run
818008

Running Jobs by Experiment (includes Onsite, OSG & Cloud)

Percent Jobs Run Onsite 92.4%
Percent Jobs Run on HEP Cloud 0.0%
Percent Jobs Run on OSG 7.6%
Fermilab Scientific Computing Summary
FIFE (continued)

Average Number of Jobs Waiting in Queue
77565

Average Time Spent Waiting in Queue
13.06 hour

Time Spent Waiting in Queue

New Data Cataloged
314.9 TB
NOvA Computing Summary

- Average Jobs Running Concurrently: 4093
- Total Jobs Run: 152018
- Average Time Spent Waiting in Queue (Production): 3.440 hour
- Total Jobs Failed (nonzero exit code): 15132
- Average CPU Efficiency: 43.6%
- New Data Cataloged: 116.4 TB
- Total Data Cataloged: 6.8 PB
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