The Collider Detector at Fermilab (CDF) Scientific Collaboration's experience with Condor

Doug Benjamin
Duke University
Collider Detector at Fermilab (CDF)

CDF:
- Large multipurpose Particle Physics Experiment at Fermi National Accelerator Lab (Fermilab)
- Started collecting data in 1988
- Data taking will continue until at least Oct. 2009 (with desire to extend another year)
Computing Model

- CAF: production/User analysis
  - FNAL hosts data: farm used to analyze them
- Decentralized CAF (dCAF): mostly Monte Carlo production

Robotic tape storage

Robotic tape storage

7MHz beam Xing

0.75 Million channels

TEVATRON

RECYCLER

Data

Disk

Grid

User jobs

Data

Data

dCAF

Grid
CAF : CDF Analysis Farm

✓ develop, debug and submit from the desktop
✓ users authentication via kerberos v5
✓ pseudo-interactive monitoring available
✓ check the jobs status over the web-interface
✓ no need to stay connected
✓ notification and summary of the end of jobs via email

output to any place
Initial Condor implementation

• Dedicated pool of machines - Very successful
• In production since 2004

Not fully grid compliant
Remote sites used dedicated computer farms
Physicist view vs Condor’s view of Computing job.

- We think of job as task with the related parallel tasks (Job sections)
- Condor – each job section – independent Job
- DAGMan – allows CDF Users – Condor to work together

- A Condor “Scheduler Universe” job
  - This is jobs that manage other jobs
  - Runs on head node
- DAGman submits individual job sections automatically
  - Submission can be conditional
    - E.g. SAM start section first, other sections only if start works
- DAGman job finishes when all sections are done
Interactive User Job Monitoring

• Condor Computing on Demand (COD) is used to allow users to monitor/interact with their job.

CDF Code and Condor Code together give the users tools needed to get their jobs efficiently.

w/ Condor COD Users can:
- look their working directories and files
- check their running jobs (debug if needed)
CDF’s Dedicated Condor Pools very successful

Condor has made it easier for us to allocate resources (Quotas)

Condor priority mechanism Dynamically adjusts for usage

Sections by Accounting Group (ordered by AcctGroup)

<table>
<thead>
<tr>
<th>Accounting Group</th>
<th>Quota</th>
<th>Running</th>
<th>Assigned</th>
<th>Idle</th>
<th>Wait</th>
<th>Held</th>
<th>Completed</th>
<th>Removed</th>
<th>Total</th>
<th>Jobs</th>
</tr>
</thead>
<tbody>
<tr>
<td>common</td>
<td>203*</td>
<td>354</td>
<td>0</td>
<td>2</td>
<td>2316</td>
<td>0</td>
<td>1612</td>
<td>768</td>
<td>5052</td>
<td>44</td>
</tr>
<tr>
<td>group_highprio</td>
<td>1822</td>
<td>1566</td>
<td>0</td>
<td>0</td>
<td>19</td>
<td>0</td>
<td>898</td>
<td>11</td>
<td>2494</td>
<td>15</td>
</tr>
<tr>
<td>group_italy</td>
<td>0</td>
<td>34</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>351</td>
<td>2</td>
<td>387</td>
<td>3</td>
</tr>
<tr>
<td>group_uk</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>35</td>
<td>40</td>
<td>1</td>
</tr>
<tr>
<td>Total (4)</td>
<td>1954</td>
<td>0</td>
<td>2</td>
<td>2335</td>
<td>0</td>
<td>2866</td>
<td>816</td>
<td>7973</td>
<td>63</td>
<td></td>
</tr>
</tbody>
</table>

* - Estimated, no real quota

Accounting User Sections (ordered by AcctUser)

<table>
<thead>
<tr>
<th>Accounting User</th>
<th>Priority</th>
<th>Running</th>
<th>Pending</th>
<th>Completed</th>
<th>Removed</th>
<th>Total</th>
<th>Jobs</th>
</tr>
</thead>
<tbody>
<tr>
<td>common.balvarez</td>
<td>1627.55</td>
<td>25</td>
<td>0</td>
<td>663</td>
<td>0</td>
<td>688</td>
<td>12</td>
</tr>
<tr>
<td>group_MCprod.benjamin</td>
<td>5167.37</td>
<td>41</td>
<td>0</td>
<td>780</td>
<td>1</td>
<td>822</td>
<td>7</td>
</tr>
<tr>
<td>group_MCprod.njg</td>
<td>466.20</td>
<td>3</td>
<td>0</td>
<td>723</td>
<td>118</td>
<td>844</td>
<td>2</td>
</tr>
<tr>
<td>group_MCprod.strolog</td>
<td>576.08</td>
<td>3</td>
<td>0</td>
<td>731</td>
<td>0</td>
<td>734</td>
<td>1</td>
</tr>
<tr>
<td>Total (4)</td>
<td>72</td>
<td>0</td>
<td>2897</td>
<td>119</td>
<td>3088</td>
<td>22</td>
<td></td>
</tr>
</tbody>
</table>
CDF Run II computing transition to Grid

- Initially distributed computing in dedicated CAF’s (North America, Asia and Europe)
- dedicated resources $\Rightarrow$ shared grid resources
  - all new computers at FNAL in grid based farms (2006)
- Data volume will grow 2.5 times
- Need more CPU power than can be found at FNAL
  - Missing resources have to be found on the grid

CDF CPU requirements by Fiscal Year – and by activity

**All Monte Carlo activities on the Grid**
- Simulated data
- Additional calculations
  - due to advanced analysis techniques
GRID Transition requirements and solution

• Requirements
  – Minimize impact on CDF user – so user can focus on the science not the computing
  – Small changes to CDF middleware
  – Continue to use the Kerberos V5 authentication for the users
    • X509 Authentication used across grid components
  – users needs to be able to run over data @ FNAL
  – submission and access must be centralized
  – Run via Globus at each grid site

• Solution
  – Condor glideins using Condor startd’s run at remote sites
  – A Virtual Condor pool is setup
    • User jobs run like a dedicated Condor pool
GlideCAF overview

- **GlideCAF** (Portal)
  - Collector
  - Negotiator
  - Main Schedd
  - Submitter Daemon
  - Monitoring Daemons
  - Glidekeeper Daemon
  - Glide-in Schedd

- **Startd** registers
- **Real job goes to slot**
- **Glide-ins to Grid**
- **New addition for grid**
- **Globus**
- **Grid Pool**
- **Available Batch**
  - Glide-in
  - Startd
- **Globus**
- **Grid Pool**
- **Glide-ins to grid-local batch**

Monitoring Daemons check to see if jobs are queued. If jobs are queued, a glide-in is submitted to a second schedd. Batch queue is used to manage tasks.
Firewall challenge - Condor GCB solution

(Generic Connection Brokering)

Works well for CDF - < 1000 connections spread across 3 GCB machines

1) Establish a persistent TCP connection to a GCB

2) Advertise itself and the GCB connection

3) To communicate, schedd sends own address to GCB

4) Schedd address forwarded

5) Establish a TCP connection to the schedd

6) A job can be sent to the startd

GCB must be on a public network

Just a proxy

Slide from Igor Sfiligoi – CDF week- Elba 2006
Glidein Security Considerations

• Glidein submitted to Globus resource starts with pilot job credentials
  – CDF end users credentials are transferred to glidein when it receives the user job
  – Some sites require the user job to run with its own credentials not the pilot job

• gLExec program is used run the user job with the users credential. (see previous Europe Condor week 2006 for more details)

• FNAL requires that CDF jobs use gLExec as part of the authentication chain.
GlideCAF performance on Fermigrid

- In past year or so installed > 6 different Condor versions. Had to use 6.9.1-6.9.5 then 7.0.0, 7.0.1.
  - Upgrades required for bug fixes that we often found after deployment
- In Oct had to split collector/negotiator to separate node we could not fill Fermigrid compute element with glideIns
  (N.B.– dedicated Condor w/o Glidein all on one node)
GlideCaf experiences

• Overloaded head nodes susceptible to losing glideins
• Less efficient at using similar resources
  – need more hardware to provide same number of slots
• Added Grid layer makes debugging problems more challenging
  – Requires help from Grid Site admin.’s
• With each new release new “features” found
  – E.g. Condor 7.0.1, 7.1.0 – broke COD – glidein’s were killed after COD command – had to revert to 7.0.0 for glideins
  – Need to use the new releases (Condor team diligence in bug fixes)
    – Request that Condor prereleases be available for testing
      • CDF continues to find new features not seen by others
CDF GlideCAF performance in Europe, Asia and on Open Science Grid

Asian GlideCAF
500 slots 6 mon.

Europe GlideCAF – CNAF Italian T1 – 600 slots 3 mon.
Future – GlideCaf improvements

• Migrate to GlideinWMS
• Installation of more powerful head nodes
  – 8 cores, 2 GB/core
• Migrate Stand-alone Condor Pool into Glidein Pool
• Further Scaling improvements
  – Investigate moving secondary schedd’s from head node to additional hardware
  – Condor Team improvements in schedd performance helps to reduce the need for migration
Conclusions

- CDF Computing continues to be very successful as a result of our use of Condor
- Through the use of Condor glideins were able to transition to the Grid with little/no impact on the Physics analyzing data
- Continued collaboration with Condor team desired (Thanks for all your help)

Acknowledgements:

- Previous CDF CAF developers – Frank Wuerthwein, Mark Neubauer, Elliot Lipeles, Matt Norman and Igor Sfiligoi
- Current CDF CAF team – DB, Federica Moscato, Donatella Lucchesi, Marian Zvada, Simone Pagan Griso, Gabrielle Compostella, Krzysztof Genser
- Fermigrid Department – especially Igor Sfiligoi and Steve Timm
Backup Slides
Computing Model

CDF Level 3 trigger ~ 100 Hz

Data Handling service

Offline Reconstruction Farm

Archival Storage (tapes)

Data

Offline

DISK CACHE

0.75 Million channels
Passive Web Monitoring

- **Monitor** extracts job status information from log files using the standard library.
- **State_Monitor** uses `condor_status` to collect node status.
- Cron jobs put information in files and RRD databases on monitoring node.
- Web pages are the data files and RRDs served by CGI scripts.

- Provides for users to see the history of their jobs.
- Provides a mechanism – Data reconstruction teams to monitor their work and submit jobs as needed.

Condor daemons are queried to job status, user priorities.
CDF Dedicated Condor Pool Head
Node Configuration

• Hardware - Dell PowerEdge 2850
  2 - Xeon 3.4 GHz (hyperthreaded) memory - 8GB

• Software - Scientific Linux Fermilab – 4.5 – Condor 6.8.6
  – Tried Condor 7.0.1 but collector problems forced a roll back
  – Our use of Kerberos authentication makes us different from many others – We often discover new behaviors in Condor

• Condor Daemons
  – Collector, Negotiator, Primary Schedd (DAGman), 4 Secondary Schedd’s (User job sections), Shadows (up ~ 2400 user sections)

• CDF Daemons
  – Submitter, Monitor, Mailer
CDF Fermigrid Glidein Pool Head Nodes Configuration

• Collector Negotiator Node:
  – Worker node class hardware- dual HT Xeon – 8 GB RAM
  – Condor 7.0.1 - Condor Collector/Negotiator Daemons

• Job and Glidein Submission Node:
  – Hardware - Dell PowerEdge 2850
    2 - Xeon 3.4 GHz (hyperthreaded) memory - 8GB
  – Software - Scientific Linux Fermilab – 4.5 – Condor 7.0.1
    • Condor 7.0.0 for glideins Condor Daemons
  – Condor Daemons
    • Primary Schedd (DAGman), 4 Secondary Schedd’s (User job sections), Shadows (up ~ 2700 user sections), Glidein schedds and Shadows
  – CDF Daemons
    • Submitter, Monitor, Mailer, Glidein submitter