Condor Week 2009

Condor WAN scalability improvements

A needed evolution to support the CMS compute model

by Dan Bradley, Igor Sfiligoi and Todd Tannenbaum
Condor and CMS

- CMS relies on Grids to manage the $O(10k)$ cores needed to analyze its data
- CMS uses Condor in many different ways
  - Burt H. had a talk about this yesterday
- One use of Condor is to use it for creating a virtual-private Condor pool on top of the Grid
  - Condor glide-ins
Condor glide-ins

- Negotiator
- Collector
- Schedd

1. Negotiator → Schedd
2. Schedd → Grid API
3. Grid API → Collector
4. Collector → Negotiator

- Glidein Factory
- Glidein Startd
- Glidein Startd
- Glidein Startd

1. Negotiator → Glidein Factory
2. Glidein Factory → Grid API
3. Grid API → Glidein Startd
4. Collector → Glidein Startd
5. Glidein Startd → Negotiator
Glidein scalability at CMS

• Spring 2007
  • GCB is unreliable
    – Although OK with a few hundred of glideins
    – But breaks easily
  ➔ Glidein usable on LAN but not on WAN (firewalls)

Igor: Glideins are the way to go!
But I need your help.
Miron: We have to fix GCB!
Alan/Jaime/Todd/Derek: We will make it work, trust us.
GCB get fixed

- Fall 2007
  - GCB code has been polished
    - Reduce port use
    - Use only TCP (before UDP was used as well)
    - Fix many bugs
  - GCB now scales over 5k glideins

Igor is happy.
Glidein scalability at CMS

- Winter 2008
  - CMS tested at Fermilab
    - One schedd node + 3 GCBs (for test purposes)
    - 10k running jobs & 200k queued jobs
  - Life is good

Igor: We are ready for production

Frank/Sanjay: We will run CCRC08* with glideins!

* CMS scalability challenge
CMS starts CCRC08

- CCRC08 (Spring 2008)
  - CMS sets up a glidein factory to all CMS Tier-2s
    - ... and the whole hell breaks loose...
  - Condor is not scaling as expected!
    - Difficult to sustain O(1k) running jobs
    - Many glideins are sitting underutilized without work

**Frank**: This thing is broken!

**Igor**: Don't worry, we will find out what is wrong.

**Igor**: Why is it not scaling as in my last tests at Fermilab?

**Dan**: Must be the network latencies.
Why are latencies hurting so much?

- In one word: Secure Authentication
  - Glideins require strong, mutual authentication
- Secure authentication requires multiple message exchanges
  - But just once, then use session key
- Condor daemons are single-threaded

<table>
<thead>
<tr>
<th>Collector</th>
<th>Startd</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK, let’s use GSI</td>
<td>Here is my server ID</td>
</tr>
<tr>
<td>Here is my client ID</td>
<td>Welcome</td>
</tr>
<tr>
<td>Use this session key in the future</td>
<td>Here is my ClassAd</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WAN</th>
<th>LAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.4s</td>
<td>0.15s</td>
</tr>
</tbody>
</table>
Where is the major bottleneck?

- The collector handles all the daemons
  - With 1.4s per daemon, it takes a long time to register $O(10k)$ of them

Igor: Why don't we create a tree of collectors? Like you do with CondorView?

Dan: Requires minor changes, but should work!
Collector tree at CCRC08

- CMS deploys a tree of 1+20 collectors
  - All on the same node, each using a different port, CONDOR_VIEW_HOST points to the main one

- The collector now handles 5k glideins with ease
  Life looks good.
More troubles with CCRC08

• Efficiencies are still very low
  • Most of the glideins are just sitting there idle!

  Sanjay: The system is still broken!
  Igor: Let me have a look

• The schedd is very slow at claiming the glideins, and is hitting timeouts left and right

  Igor: Looks like we have problems with latencies again!
  But why?
  Dan: Let's analyze what is going on.
Schedd talks a lot, too

- Schedd handles many connections
  - Each connection is a new secure handshake
    - Glideins come and go
  - All the above block 1.4s on WAN

  **Igor**: What can we do?
  I don't want to use many schedds!

  **Dan**: I will make all connections nonblocking.
A couple months later...

- Just in time for the last round of CCRC08
- Many Condor libraries have been modified to be non-blocking
  - Bringing WAN blocking time to 1.0s
  - System behaves better, but O(10k) still just a dream

Dan: Give me a few more months and I will make all connections non-blocking.
Todd: Don’t unwind all our code; use cooperative threads.
Igor: I trust you, but I need a solution soon
Miron: Guys, think! Is there no better solution?
Miron was right!

```
condor_config.local:
c = 3000000 km/s
```
The better solution

- Instead of trying to brute force the problem, we found a better solution
- Use the Collector as trust manager
  - Welcome “(security) match sessions”
    (enabled via SEC_ENABLE_MATCH_PASSWORD_AUTHENTICATION)
Glidein scalability at CMS

• Winter 2009
  • CMS tested across the ocean
    - 1+70 collectors (and using CCB)
    - Using the “match sessions”
  • 23k running jobs & 400k queued jobs
    - Limited by port usage (2 ports x running job)
    - But way above the target of 10k+ running jobs
      • 200k jobs processed in a day!
  • Life is good again
Glideins in production at CMS

- Winter 2009:
  - CMS uses glideins for worldwide data processing

![Graph showing running glideins - last month](image-url)
A comment on CCB

• Since Fall 2007 GCB scaled fine, but
  • Used a lot of ports
    (5-6 per glidein → max 8k glideins x GCB )
  • Was not fault tolerant
    (could not restart GCB without losing the pool)
• CCB was designed based on GCB experience
  • Uses just one port
  • It can be restarted without harm
  • Scales just as much as GCB
Conclusion

- Major progress made in WAN setups
  - GCB fixed $\rightarrow$ experience inspired CCB
  - Tree of collectors to distribute authentication load
  - “Match sessions” for smarter security
- CMS heavy user of glideins
  - Via glideinWMS
  - Could not have used them without the effort invested by the Condor team
Acknowledgments

• This work was supported by
  • U.S. DoE under contract No. DE-AC02-07CH11359
  • U.S. NSF grants PHY-0427113 (RACE) and PHY-0533280 (DISUN)