What’s new in HTCondor?
What’s coming?

HTCondor Week 2014

Todd Tannenbaum
Center for High Throughput Computing
Department of Computer Sciences
University of Wisconsin-Madison
Release Situation

› Development Series
  • HTCondor v8.1.6 frozen (release candidate for v8.2.0), in beta test, release to web 5/20/14.
  • Development on series v8.1.x over, v8.3.x release scheduled for 7/1/14.

› Stable Series
  • June 5th: HTCondor v8.2.0
  • v8.0.6 will likely be the last v8.0.x released
  • Last Year: Condor v8.0.0 (June 6th 2013)

› Since HTCondor Week 2012: 14 releases, 3375 commits by 32 contributors, resolved tickets: 170 stable series, 397 dev series
<table>
<thead>
<tr>
<th>Platforms tested with v8.2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>32bit Debian 6</td>
</tr>
<tr>
<td>32bit Scientific Linux 5</td>
</tr>
<tr>
<td>32bit Scientific Linux 6</td>
</tr>
<tr>
<td>64Bit Debian 6 (squeeze),</td>
</tr>
<tr>
<td>64Bit Debian 7 (wheezy),</td>
</tr>
<tr>
<td>64Bit Fedora 19</td>
</tr>
<tr>
<td>64Bit Fedora 20</td>
</tr>
<tr>
<td>64Bit MacOSX 8</td>
</tr>
<tr>
<td>64Bit MacOSX 9</td>
</tr>
<tr>
<td>64Bit Red Hat 5</td>
</tr>
<tr>
<td>64Bit Red Hat 6</td>
</tr>
<tr>
<td>64Bit Red Hat 7</td>
</tr>
<tr>
<td>64Bit Scientific Linux 5</td>
</tr>
<tr>
<td>64Bit Scientific Linux 6</td>
</tr>
<tr>
<td>(and 7 when it becomes</td>
</tr>
<tr>
<td>available)</td>
</tr>
<tr>
<td>64Bit Ubuntu 12.04 LTS</td>
</tr>
<tr>
<td>64Bit Ubuntu 14.04 LTS</td>
</tr>
<tr>
<td>64Bit Windows 7 SP1</td>
</tr>
<tr>
<td>64Bit Windows 8.1</td>
</tr>
</tbody>
</table>

› Continue to push into distro repositories
New goodies with v8.0

› HTCondor-CE
› Bosco
› DAGMan additions
› EC2 Spot, OpenStack
› Several new tools
› ClassAd Compression
› Generic Slot Resources
› Python Interfaces
› Job Sandboxing

Interactive jobs
Open development process across the entire project
Security policy maturation
Many more…

Last Year's News
“...we have identified six key challenge areas that we believe will drive HTC technologies innovation in the next five years.”

• Evolving resource acquisition models
• Hardware complexity
• Widely disparate use cases
• Data intensive computing
• Black-box applications
• Scalability
EC2 Grid job improvements

› Hardening, better failure handling
  • Especially for spot instances
  • Errors used to result in orphaned instances
  • Reduced calls to service EC2 jobs

› Better support for OpenStack
  • Recognize and handle protocol changes
  • Recognize API differences from Amazon

› condor_ssh_to_job supports EC2 jobs
Google Compute Engine Jobs

› New grid-type “gce”
› Similar to EC2 support
› Basic instance parameters
   • GCE zone
   • Machine type
   • VM Image
   • Instance-specific data
A Brief History of BOINC

- Berkeley Open Infrastructure for Network Computing
- Grew out of SETI@Home, began in 2002
- Middleware system for *volunteer computing*
Previous Work

› Previous work
  • Backfill state and Work-Fetch Hooks
  • HTCondor execute machine becomes a BOINC client when otherwise idle

› Now, we’re doing the reverse…
HTCondor submitting jobs to BOINC

› New grid universe type: boinc
  • Submit file format very similar to other job types
  • Application must be described to BOINC server first
    • Manual step at present

› Why?
  • Easy accessibility for campus users, OSG VOs
  • “Submit locally, run globally”
  • Use in workflows
Scalability

"You Won’t Find a Broader Line of Microcomputers Than

—Isaac Asimov
Renowned Science and Science-Fiction Author
Improve matchmaking protocol esp over slow links

“…CMS has stood up a new submit machine in Switzerland, trying to connect to a central manager in the US. Round trip ping time is on the order of 200 ms, and they see that the negotiation is much slower to those schedds, topping at 5Hz”
Improve matchmaking protocol esp over slow links, cont

The new patch helped CMS quite a bit. See the attached picture; you can see the difference before and after the patch was applied ("week 33"). Now we just need it in the official binaries.

2013-Aug-22 10:41:30 by tannenba:
^^^^ It is being released in HTCondor v8.1.1.
Currently in production with ~50k slots

Testing now for a target of 250k slots

Glidein nodes around the world
  • Network latency
  • CCB / shared_port
  • Strong security (GSI)

Knocking over bottlenecks
  • “Collector only handling 100 updates/sec!”
How to configure multi-tier collectors

Known to work in HTCondor version: 7.0.1

This is a technique for increasing the scalability of the HTCondor collector. This has been found to help scale up glidein pools using GSI authentication in order to scale beyond ~5000 slots to ~20000 slots. Other strong authentication methods are similarly CPU intensive, so they should also benefit from this technique. When authenticating across the wide area network, network latency is actually more of a problem for the collector than CPU usage during authentication. The multi-tier collector approach helps distribute the latency and CPU usage across
Collector Update Rate

udp 1k
Update w/ jumbo UDP datagrams

![Graph showing performance of udp 1k and udp 16k]
More scalability changes

- Schedd restart time improved
- Reduced max time to detect a disappeared node now 6 minutes, was 2 hours.
- Non-blocking I/O
- Do not fork schedd to process condor_q
  - Why? Observed schedd child process grow at ~500MB/sec
- Do not fork to process incoming connections to condor_shared_port
Defaults are all most users care about!
Defaults are all most users care about!
Defaults are all most users care about!
Defaults are all most users care about!
Defaults are all most users care about!
Defaults are all most users care about!
Defaults are all most users care about!
Defaults are all most users care about!
Defaults are all most users care about!
Defaults are all most users care about!
More power

New configuration language constructs

- \$(<\text{knob}>:<\text{default}>)
- include
- if, else,elif, endif
- use <category>:<option>

Use meta-knobs

- See categories and options with \texttt{condor\_config\_val}
- Examples:
  - use role:execute
  - use policy:always\_run\_jobs
  - use feature:gpus
GPU Support

› HTCondor can now automatically
  • Discover, Bind, Sandbox, Monitor, and Manage GPUs

› CUDA and OpenCL

› Both static slots and partitionable slots supported!

› How?
  • Add to config file
    use feature:gpus
  • Sample submit file
    request_cpus = 1
    request_gpus = 1
    executable = hola_gpus.exe
    queue
Data Intensive Computing
What is it?

- Allows the execution of a job to be overlaid with the transfer of output files of a different job

Conditions apply:

- Both jobs must be from the same user
- Submitted from the same submit point
- Jobs must explicitly choose to participate
Normal HTCondor operation for a single compute slot has three phases. Several consecutive jobs look like this:
The “input/execute” phase and the “output” phase can be done concurrently, also known as “pipelining”:
How does it work?

The schedd can now tell a startd to move a job from one slot to another.

The execute node is configured to create dedicated “transfer slots” for each traditional execute slot.

- The transfer slot will not run jobs
- Jobs are moved into the transfer slot when they signal the phase transition, but only if the transfer slot is idle.
- Typically, transfer slots are not visible, but like everything in HTCondor, there’s a knob for that!
A single core machine (with the transfer slot configured to be visible):

```
% condor_status
Name     OpSys     Arch     State         Activity LoadAv Mem   ActvtyTime
slot1@ingwe.cs.wis LINUX X86_64 Unclaimed Idle 0.250 15816 0+00:00:11
xfer2@ingwe.cs.wis LINUX X86_64 Unclaimed Idle 0.000  124  0+00:00:10

Total Owner Claimed Unclaimed Matched Preempting Backfill
   X86_64/LINUX     2     0       0         2       0          0        0
   Total            2     0       0         2       0          0        0
```
To work, the submit node, execute node, and the job itself must participate.

If any party does not participate, normal job execution results. (i.e. no overlay)

The job explicitly signals the transition to output phase using a chirp command:

```
condor_chirp phase output
```
What does it look like from the user perspective?

Example executable file:

```
#!/bin/sh

echo "Executing..."
sleep 120

echo "Transferring..."
condor_chirp phase output
sleep 120
```
ASYNC_STAGEOUT

What does it look like from the user perspective?

Normal job execution for first 120 seconds:

```
% condor_q

-- Submitter: ingwe.cs.wisc.edu : <128.105.121.64:56450> : ingwe.cs.wisc.edu

<table>
<thead>
<tr>
<th>ID</th>
<th>OWNER</th>
<th>SUBMITTED</th>
<th>RUN_TIME</th>
<th>ST</th>
<th>PRI</th>
<th>SIZE</th>
<th>CMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1255.0</td>
<td>zmiller</td>
<td>4/29 11:26</td>
<td>0+00:01:29</td>
<td>R</td>
<td>0</td>
<td>0.0</td>
<td>async.sh 120</td>
</tr>
<tr>
<td>1255.1</td>
<td>zmiller</td>
<td>4/29 11:26</td>
<td>0+00:00:00</td>
<td>I</td>
<td>0</td>
<td>0.0</td>
<td>async.sh 120</td>
</tr>
<tr>
<td>1255.2</td>
<td>zmiller</td>
<td>4/29 11:26</td>
<td>0+00:00:00</td>
<td>I</td>
<td>0</td>
<td>0.0</td>
<td>async.sh 120</td>
</tr>
<tr>
<td>1255.3</td>
<td>zmiller</td>
<td>4/29 11:26</td>
<td>0+00:00:00</td>
<td>I</td>
<td>0</td>
<td>0.0</td>
<td>async.sh 120</td>
</tr>
<tr>
<td>1255.4</td>
<td>zmiller</td>
<td>4/29 11:26</td>
<td>0+00:00:00</td>
<td>I</td>
<td>0</td>
<td>0.0</td>
<td>async.sh 120</td>
</tr>
</tbody>
</table>

5 jobs; 0 completed, 0 removed, 4 idle, 1 running, 0 held, 0 suspended
```
What does it look like from the user perspective?

After 120 seconds another job starts:

```
% condor_q

-- Submitter: ingwe.cs.wisc.edu : <128.105.121.64:56450> : ingwe.cs.wisc.edu

<table>
<thead>
<tr>
<th>ID</th>
<th>OWNER</th>
<th>SUBMITTED</th>
<th>RUN_TIME</th>
<th>ST</th>
<th>PRI</th>
<th>SIZE</th>
<th>CMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1255.0</td>
<td>zmiller</td>
<td>4/29 11:26</td>
<td>0:00:02:09</td>
<td>R</td>
<td>0</td>
<td>0.0</td>
<td>async.sh 120</td>
</tr>
<tr>
<td>1255.1</td>
<td>zmiller</td>
<td>4/29 11:26</td>
<td>0:00:00:03</td>
<td>R</td>
<td>0</td>
<td>0.0</td>
<td>async.sh 120</td>
</tr>
<tr>
<td>1255.2</td>
<td>zmiller</td>
<td>4/29 11:26</td>
<td>0:00:00:00</td>
<td>I</td>
<td>0</td>
<td>0.0</td>
<td>async.sh 120</td>
</tr>
<tr>
<td>1255.3</td>
<td>zmiller</td>
<td>4/29 11:26</td>
<td>0:00:00:00</td>
<td>I</td>
<td>0</td>
<td>0.0</td>
<td>async.sh 120</td>
</tr>
<tr>
<td>1255.4</td>
<td>zmiller</td>
<td>4/29 11:26</td>
<td>0:00:00:00</td>
<td>I</td>
<td>0</td>
<td>0.0</td>
<td>async.sh 120</td>
</tr>
</tbody>
</table>

5 jobs: 0 completed, 0 removed, 3 idle, 2 running, 0 held, 0 suspended
```
What does it look like from the user perspective?

You can see the first job has moved:

```
% condor_q -run

-- Submitter: ingwe.cs.wisc.edu : <128.105.121.64:56450> : ingwe.cs.wisc.edu
ID   OWNER     SUBMITTED       RUN_TIME     HOST(S)
1255.0 zmiller  4/29 11:26  0+00:02:44   xfer2@ingwe.cs.wisc.edu
1255.1 zmiller  4/29 11:26  0+00:00:38   slot1@ingwe.cs.wisc.edu
```
What does it look like from the user perspective?

Eventually first job finishes:

```
% condor_q

-- Submitter: ingwe.cs.wisc.edu : <128.105.121.64:56450> : ingwe.cs.wisc.edu

<table>
<thead>
<tr>
<th>ID</th>
<th>OWNER</th>
<th>SUBMITTED</th>
<th>RUN_TIME</th>
<th>ST</th>
<th>PRI</th>
<th>SIZE</th>
<th>CMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1255.1</td>
<td>zmiller</td>
<td>4/29 11:26</td>
<td>0+00:02:01 R 0 0.0 async.sh 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1255.2</td>
<td>zmiller</td>
<td>4/29 11:26</td>
<td>0+00:00:00 I 0 0.0 async.sh 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1255.3</td>
<td>zmiller</td>
<td>4/29 11:26</td>
<td>0+00:00:00 I 0 0.0 async.sh 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1255.4</td>
<td>zmiller</td>
<td>4/29 11:26</td>
<td>0+00:00:00 I 0 0.0 async.sh 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4 jobs; 0 completed, 0 removed, 3 idle, 1 running, 0 held, 0 suspended
```
What does it look like from the user perspective?

And another job starts... and so on...

```
% condor_q

-- Submitter: ingwe.cs.wisc.edu : <128.105.121.64:56450> : ingwe.cs.wisc.edu
   ID   OWNER   SUBMITTED       RUN_TIME ST PRI SIZE CMD
1255.1 zmiller 4/29 11:26 0+00:02:07 R  0   17.1 async.sh 120
1255.2 zmiller 4/29 11:26 0+00:00:05 R  0   0.0 async.sh 120
1255.3 zmiller 4/29 11:26 0+00:00:00 I  0   0.0 async.sh 120
1255.4 zmiller 4/29 11:26 0+00:00:00 I  0   0.0 async.sh 120

4 jobs; 0 completed, 0 removed, 2 idle, 2 running, 0 held, 0 suspended
```
To enable on your cluster, just use the meta-knob:

```
use EXPERIMENTAL:ASYNC_STAGEOUT
```

See ticket #4291
Is my pool healthy?

We added a lot of metrics in HTCondor v7.9.x, but how to spot trends?
Operational monitoring over time with condor_gangliad

› Monitor your cluster with familiar tools
› Measure utilization
› Diagnose problems
› View usage over time
› Easy to configure
  • Esp if already using ganglia
HTCondor File Transfer metrics (6)

- **FileTransferDownloadBytesPerSecond_5m**
  - Bytes/s vs. time for processes:
    - Wednesday 23rd
    - Friday 25th
    - Sunday 27th
  - Last week: submit-1.chtc.wisc.edu, now 2.55

- **FileTransferFileReadLoad_5m**
  - File read load vs. time for processes:
    - Wednesday 23rd
    - Friday 25th
    - Sunday 27th
  - Last week: submit-1.chtc.wisc.edu, now 0.00

- **FileTransferFileWriteLoad_5m**
  - File write load vs. time for processes:
    - Wednesday 23rd
    - Friday 25th
    - Sunday 27th
  - Last week: submit-1.chtc.wisc.edu, now 0.02

- **FileTransferNetReadLoad_5m**
  - Network read load vs. time for processes:
    - Wednesday 23rd
    - Friday 25th
    - Sunday 27th
  - Last week: submit-1.chtc.wisc.edu, now 0.01

- **FileTransferNetWriteLoad_5m**
  - Network write load vs. time for processes:
    - Wednesday 23rd
    - Friday 25th
    - Sunday 27th
  - Last week: submit-1.chtc.wisc.edu, now 0.01

- **FileTransferUploadBytesPerSecond_5m**
  - Upload bytes per second vs. time for processes:
    - Wednesday 23rd
    - Friday 25th
    - Sunday 27th
  - Last week: submit-1.chtc.wisc.edu, now 1.27
Could always configure max # of concurrent file uploads and downloads.

THE WINNER OF THE “NOT MY JOB” AWARD GOES TO.....
Could always configure max # of concurrent file uploads and downloads.

New configuration variable `FILE_TRANSFER_DISK_LOAD_THROTTLE` enables dynamic adjustment of the level of file transfer concurrency to keep the disk load generated by transfers below a specified level.
Schedd overloaded, now what?

› condor_sos <whatever….>, eg
  • condor_sos condor_q
  • condor_sos condor_hold toddt
  • condor_sos –schedd –direct my_schedd

› Every superuser command is an SOS

Save Our Schedd!
New script in ~condor/bin

Email report shows preempted jobs:
The following users have run vanilla jobs without +is_resumable = true, that have hit the MaxJobRetirementTime yesterday.

# of User Jobs
---- ----
3  wbrooks2@submit.chtc.wisc.edu
6  quefeng@submit.chtc.wisc.edu
7  davisa@submit.chtc.wisc.edu
44 asiahpirani@discovery.wisc.edu
condor_job_report script

Shows statistics about job runtimes

- Short jobs
- Quartiles
- Mean runtime
- Restarts of same job
More monitoring

- Monitoring jobs via BigPanDAmon
- Publication of useful metrics, such as
  - Number of job preemptions (by startd)
  - Number of autoclusters (by schedd)
- `condor_status -defrag`
New DAGMan features

- More info and improved structure in node status file
- Metrics reporting (used by Pegasus)
- DAG files can now be > 2.2 GB
- DAGMAN_DEFAULT_NODE_LOG has been made more powerful, so that it can be defined in HTCondor configuration files
- Non-workflow log mode is now deprecated
- Node retry number is now available as VARS macro
startd RANK for pslots

- Startd RANK expressions for pslots now works
- Pslot slot ads now have info about dslots
- In new classad arrays:
  - If startds has four dslots running, looks like
    ```
    childCpus = { 1,1,4,1 }
    childCurrentRank = { 0.0,0.0,0.0,0.0,0.0 } 
    childState = {"Claimed","Claimed","Claimed","Claimed""
    ```
  - But not user prio yet
- Future work: no dslots in collector!
VM Universe enhancements for admins

› Test a VM on start-up. (#3789)

› Test VM networking on start-up. (#3960)

› And periodically re-check both.

› Advertise health of VMU on a host. (#3976)
  • Include testing results and job attempts.
VM Universe enhancements for users

› Clean up submission problems
  • `vm_memory` vs `RequestMemory` (#3907)
  • file transfer
    • spurious (?) warnings (#3908)
    • doesn’t play well with file-transfer plugins (#4167)
    • interaction with `vm_no_output_vm` (#2556)

› Support an exit status code (#3961)
KVM-Enabled Checkpointing of vanilla universe jobs

› Swap standard universe’s constraints for the constraints of VMs
  • Moves some burdens from user to admin

› Transparent to application and user?
  • Admin could set default VM to one that looks like the host OS

› Combine vanilla and VM universe
Original Plan

VM

starter

shared folder

starter

user job
Native OpenStack Support

› Speak OpenStack’s NOVA protocol
  • Better supported than EC2 compatibility interface
  • Allows better error handling
  • Provides richer set of controls on instances
  • Potential to obtain and manage resources beyond servers
Continue to push on…

› Network enhancements
  • Zhe Zhang’s talk yesterday (LARK)
  • Alan DeSmet’s talk tomorrow (source routing, IPv6)

› First-class facility in HTCondor for caching job input data sets on execute nodes
  • Give feedback on our design! See http://goo.gl/8sxQJb

› HTPC scheduling mechanisms
  • Built-in vs expressions

› Less tuning out of the box (think Skype…)
  • shared_port and collector hierarchy on by default, …

› Grouping of jobs (and machines?)

› Horizontal scaling

› Provide operators more visibility into their pool
Thank You!