So you want to build the ultimate submit machine?

- While you can get HTCondor to run on your toaster in a weekend, providing a high-quality scalable submit service can take significant planning and effort.

- In this talk, we’ll walk through the process of putting together the service, noting special requirements for scalability and customization hooks.

- I focus on the non-obvious parts of this task; this is not “how to build your first submit machine”.

- Roughly, three portions:
  - Spec’ing out the service.
  - Installing and Configure HTCondor.
  - Customizing user environments.
Roadmap - Where Are We?

HTCondor Pool

Central Manager
- collector
- negotiator

Submit Node
- schedd

Worker Node
- startd

YOU ARE HERE
Spec’ing out the Service - Setting Expectations

• Before we even get to hardware, you need to work with users to understand what kind of service is needed:

  • **Job Scale:**
    • What is the maximum number of jobs this schedd will need to run? The average?
    • How many jobs are expected to be in queue?

  • **Job Rates:** What is the expected job start and stop rates? What does the distribution look like?

  • **IO requirements:** What, if anything, do you know about your per-job input and output transfer requirements?

• In general, it’s really hard to determine what the distributions look like. HTCondor keeps only rough statistics itself. I prefer to do the *highly scientific* “multiply everything by two” to determine peak scale.
Spec’ing out the Service - Hardware Considerations

• Next, I outline the hardware considerations from most important to least.

• **IO**: The schedd is a single-threaded daemon which blocks on disk IO and frequently calls fsync() on its job database.

  • Therefore, your overall scalability is limited by the latency of your storage system.

  • To maintain a stable service of >10k running jobs, you will want to keep the spool directory on an SSD.

• A typical setup has:

  • A dedicated, small, low-latency storage target for spool, AND

  • A large (TBs), high-throughput storage target for user home/working directories.
TL;DR:
Buy a SSD, Live Happy
Spec’ing out the Service - Hardware Considerations 2

- **Memory**: As a rule of thumb, plan on 1MB RAM per running job and 50KB per idle job.

  - In the last two years, this was reduced to 300-400KB per running job. I still prefer the above number to include a bit of a safety factor.

- **CPU**: The schedd has no CPU-bound component (the process is single-threaded anyway).

  - Base your CPU decisions on the needs of the logged-in users (i.e., compiling or running test jobs).

- **Network connectivity**: Unless you are aware of specific needs from your user base, 1Gbps is sufficient.
To shared filesystem or not?

• How do you move files between the submit and execute machines?

• **With a shared file system**: These can be expensive and finicky, but users often love the simplicity. They don’t need to know what files they use.
  
  • It’s often difficult to carefully control usage of the shared file system - life can be chaotic!

• **With HTCondor file transfer**: Forces users to *think* and express their file requirements inside the job.
  
  • Requires more work from the user - *however*, it typically results in a more “IO friendly” job. No user hammering AFS!

  • HTCondor can throttle new transfers (future: not match machines) if the schedd is spending too much time on IO. Shared file systems typically have no concept of queueing and performance degrades massively!

  • When using file transfers, it is simpler to run jobs offsite.
DANGER! WARNING!

• While we recommend using HTCondor file transfer, we understand this is not always possible.

• **NOTE** the `condor_schedd` writes user logs in-process. If the user has this file on the shared file system and the filesystem stops responding, then the schedd will stop responding.

• HTCondor relies on a few obscure POSIX semantics for user logs. **No funny business** such as FUSE filesystems. Even NFS was finicky until the last 3-5 years.
OS Tweaks
(for schedds with >10k jobs)

- Memory overcommit: In /etc/sysctl.conf, `sys.vm.overcommit_memory=1`
- Max socket backlog: In /etc/sysctl.conf, `net.core.somaxconn=1024`
- Max file descriptors: Set `sys.fs.file-max` to be greater than 500k (already is on most OSes!)
- Max per-process file descriptors: Set `nofile` in /etc/security/limits.d.
  - Not done commonly (see scaling talk).
- Maximum number of processes: Set `nprocs` in /etc/security/limits.d.
  - Only for hosts which do lots of DAGMan / Local universe.
- Beware of iptables conntrack module: Consider blacklisting the conntrack module if you need many TCP connections (see scaling talk).

Still relevant for some sites
OS Tweaks - 8.4.x

• Starting in the latest series, HTCondor will now perform developer-recommended reasonable kernel tunings on startup.
  
  • These are selected so they should be safe for “anyone,” but do touch some global settings.

• Sysadmins can turn this off (not recommended) or provide their own overrides / additions (recommended).

• This was a contentious feature internally: the need for simplicity versus reluctance to touch system settings.
  
  • I suspect there is tuning of the approach left to do.

• Would love to hear feedback!
Host Firewalls and Networking

- **DNS**: DNS is a mixed bag! HTCondor can work fine with- or without DNS; in fact, DNS failures (or slow name resolution) often cause problems for submit services. Recommendations:
  
  - Go all-in or all-out: don’t try to mix use of IP addresses in some cases and DNS in others.
  
  - It is the *host name*. There should be one per host; if you use DNS, the hostname should match the public DNS name for simplicity. If you need a more complex setup, the `NETWORK_HOSTNAME` config option overrides the hostname detection logic.
  
  - Consider your cluster’s dynamics: if there’s a small number (<50) of nodes and they won’t come in and out of the cluster frequently, you may not need DNS.

- The worker nodes, central manager, and schedd need to be able to contact each other via the network.

- I *highly* recommend setting `USE_SHARED_PORT=true` (in fact, the plan is to make this the future default) throughout your pool. This will allow all HTCondor daemons to use the same inbound port, TCP 9618.

- HTCondor has the ability to rewrite addresses (for TCP port-forwarding setups) and intelligently manage multiple private and public networks. While this means HTCondor can work with very adverse networking conditions, *think twice before using; they can be extremely difficult to debug.*
Host Firewalls and Networking

- With shared port enabled, the firewall configuration becomes:

  - **Inbound connections**: TCP 9618 from client hosts, the central manager, and worker nodes.
  
  - **Outbound connections**: Outbound connections are necessary to the central manager and worker nodes.

    - HTCondor phone home: By default, the HTCondor daemons report simple usage statistics to UW via UDP. This is a requirement from the funding agencies; consider leaving this on if you wish continued support of the software. For more, see [http://research.cs.wisc.edu/htcondor/privacy.html](http://research.cs.wisc.edu/htcondor/privacy.html).

    - By default, UDP updates are sent to the central manager; if desired, switch them to TCP using `UPDATE_COLLECTOR_WITH_TCP=true`. All other outgoing communication uses TCP.

  - The CCB allows the worker nodes to be behind a separate stateful firewall or NAT (i.e., no inbound connectivity from the schedd). This is not typically used in site setups.
Installing and Configuring

• **Basics:**
  
  • Always install via RPM (debs); I strongly discourage use of tarballs.
  
  • Always maintain your configurations with configuration management software such as Puppet or Chef.
  
  • *Never* edit condor_config or condor_config.local. *Always* use the config.d directory.
Logging Considerations

• Consider enabling the AuditLog; this contains a concise log of who used the schedd, what they did, and how they authenticated.
  • Essential for security incidents!

• Explicitly determine your log retention policy; default is 10MB x 2 files per log.
  • Most large sites will want to retain more. I use 100MB x 10 files.

• Set the logfile name to SYSLOG to forward a HTCondor log to /dev/log. Useful for sites that have an existing centralized log management scheme and/or strict retention policies.
  • In particular, sites should consider forwarding the AuditLog to syslog.
Monitoring - Host

• Host-level monitoring and alerting is critical, especially if users have a login to the submit host.

• This is not HTCondor-specific; apply the security protections you believe needed for a generic login host.

• Users are quicker than your alert system; typically, monitoring is best for post-crash telemetry.
Monitoring - HTCondor

- All HTCondor daemons export 5-20 critical metrics in their ClassAds.

- Recently, HTCondor delivered native integration with Ganglia. This allows you to turn the above metrics into time series.

  - When combined with host metrics (CPU usage, memory, network activity), these are a powerful mechanism for debugging problems.

  - If your site doesn’t use Ganglia for monitoring, the daemon can integrate with your system by invoking a “gmetric” compatible command-line utility.
Accounting

• While condor_history is great, the logs do rotate eventually.
  • Don’t wait until your boss asks about accounting usage to discover this fact!

• If you set `PER_JOB_HISTORY_DIR`, then the schedd records the job ClassAd into a unique file when it leaves the queue.
  • Accounting can be done by reading each of these files and uploading to a DB.
  • Alternately, the `PER_JOB_HISTORY_DIR` captures the job execution instances on the remote startds. Further, this can be queried centrally (if you have admin privileges).
Accounting

- Recall `condor_history` can be invoked remotely.
  
  - Via python bindings, one can collect the poolwide history
  
  - Looking to make this more efficient in 8.5.x.
  
  - Similarly, python bindings can fetch `PER_JOB_HISTORY_DIR` from schedds and startds.
  
- Consider taking this centrally collected data and pushing it into ElasticSearch. Popular to do this + Kibana.

  Extensive CMS-specific example: https://github.com/bbockelm/cm-s-htcondor-es
Configuration Knobs to investigate

- **SYSTEM_PERIODIC_REMOVE / SYSTEM_PERIODIC_HOLD**: Expression to either remove or hold “malformed” jobs.
  
  - Check out **SYSTEM_PERIODIC_XXX_REASON** too!

- **MAX_JOBS_RUNNING / MAX_JOBS_SUBMITTED**: Limit the number of jobs running / submitted to prevent users from pushing the schedd into swap.

- **FILE_TRANSFER_DISK_LOAD_THROTTLE**: If you are using HTCondor transfer mechanisms, this limits the amount of disk load HTCondor places on the system (suggestion: set to N for a host with N spinning disks).

- **MAX_TRANSFER_{INPUT,OUTPUT}_MB**: Avoid transferring excessive amounts of data per job.
NEW - Managing User Job ClassAds

- Historically, the job ClassAd “belongs” to the user. All attributes except Owner could be modified by the user via condor_q. However,
  - Group accounting information is taken from ad.
  - Some attributes (X509 certificate DN) are used by admins for policy decisions.
  - In 8.3.x, we introduced SUBMIT_REQUIREMENTS: you can force jobs to match certain constraints.
  - In 8.5.2, we introduced protected attributes: once set, can only be changed by the sysadmin.
Managing User ClassAds

• Finally, the big hammer: custom ClassAd functions. These can be written in python (easy) or C++ (hard).

• Use sparingly (i.e., in SUBMIT_REQUIREMENTS but not job’s REQUIREMENTS).

• Must evaluate quickly; no side-effects, no state.

• If it must access a remote service, cache aggressively.
SUBMIT_REQUIREMENTS
Example

Config snippet:

SCHEDD_CLASSAD_USER_PYTHON_MODULES=my_utils
SCHEDD_ENVIRONMENT="PYTHONPATH=/path/to/my_modules"
SUBMIT_REQUIREMENT_NAMES = CHECKTODD
SUBMIT_REQUIREMENT_CHECKTODD =.isUserTodd(Owner)
SUBMIT_REQUIREMENT_CHECKTODD_REASON = \
    strcat("This is ", Owner, ", not Todd!")

Python code example:

import classad

def isUserTodd(user, state={}):
    return user == "todd"

classad.register(isUserTodd)
SUBMIT_REQUIREMENTS
Example

$ condor_run echo "Hello world"
Submitting job(s).
ERROR: Failed to commit job submission into the queue.
ERROR: This is bbockelm not Todd!
Failed to submit Condor job.
Setting up the User Environment

• How does a user submit a job? It's a bit of a religious argument.

• **School of thought #1**: Make users learn `condor_submit`. There's tons of documentation “on the internet”, allows users to fully unlock the power of `condor_submit`, and is no-maintenance.

• **School of thought #1.1**: Write a small wrapper around `condor_submit` to “helpfully” fix obvious errors in files or set a few site-specific defaults.
  
  • Alternately, can control some defaults from the user environment. I.e., add the following to `/etc/profile.d/condor.sh`:
    
    ```bash
    export _CONDOR_AccountingGroup="local.`id -gn`\`id -un`"
    ```
  
  • Periodically check schedd-side to see if a user is trying to game the system.

• **School of thought #2**: Any `condor_*` command is too damn hard to use. Replace it with a simpler site-specific interface and train them to use this.
  
  • *Alternately*, use `condor_qsub` because you like PBS-style scripts better!
  
  • **Note**: wrapper scripts require the users to play along. Do not be surprised to find they bypass your script when python bindings are used.

• **School of thought #2.1**: Any command line is too hard for users; they only access the system through a webapp.
User Environments - Automating attribute settings

• **Easy**: Utilize SUBMIT_ATTRS. Add to the config file:

  ```
  JobIsGrid = true
  SUBMIT_ATTRS = $(SUBMIT_ATTRS), JobIsGrid
  ```

• **Medium**: Use MODIFY_REQUEST_EXPR_* to modify a user’s request_* at the startd or JOB_DEFAULT_* to modify at condor_submit.

• **Medium**: Use SCHEDD_ROUND_ATTR_* to round up arbitrary attributes at the schedd.

• **Medium-hard**: Write a wrapper around your submit script.

• **Hard**: Use JobRouter to enforce policy schedd-side.
Upcoming Automation

• For automating attribute values, in 8.5.x, we hope to:

  • Make SUBMIT_ATTRS work schedd-side.

  • Allow attributes to be evaluated at submit time.

  • Re-introduce the “unexpanded” state. This causes the schedd to not consider the job until it has been transformed by the JobRouter.
Tweaks

• Ideas that make user’s life better:

  • Use the custom condor_q / condor_status print formats for your site.

  • Take advantage of ~/.condor/user_config (user-specific config file, like ~/.bashrc); for example, you can created this file on first login with a PAM module to lock the user to a specific schedd.

  • Customize MOTD to tell the user a summary of their jobs on login.
Print Formats

SELECT
  Name AS Name WIDTH -18
  OSG_Resource AS Resource WIDTH -18
  OSG_BatchSystems AS Batch WIDTH -8
  HTCondorCEVersion AS CEVer WIDTH -5
  split(condorversion)[1] AS CondorVer
  DaemonStartTime AS Uptime PRINTAS ACTIVITY_TIME
  grid_resource AS Resource

SUMMARY NONE
## Print Formats

<table>
<thead>
<tr>
<th>Name</th>
<th>Resource</th>
<th>Batch</th>
<th>CEVer</th>
<th>CondorVer</th>
<th>Uptime</th>
</tr>
</thead>
<tbody>
<tr>
<td>alt13gm.physics.arizona.edu</td>
<td>Arizona_CE</td>
<td>2.0.0</td>
<td>8.4.3</td>
<td>18:04:36:43</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>bonner06.rice.edu</td>
<td>OSG-Rice</td>
<td>2.0.0</td>
<td>8.2.10</td>
<td>5:21:15:05</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>byggvir.princeton.edu</td>
<td>UNAVAILABLE</td>
<td>2.0.0</td>
<td>8.2.10</td>
<td>8+20:41:16</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>calclab-ce.math.tamu.edu</td>
<td>TAMU_Calclab</td>
<td>2.0.0</td>
<td>8.2.10</td>
<td>10+17:56:05</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>carter-osg.rcac.purdue.edu</td>
<td>Purdue-Carter</td>
<td>1.20</td>
<td>8.2.10</td>
<td>5+15:16:53</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>ce01.brazos.tamu.edu</td>
<td>TAMU_BRAZOS_CE</td>
<td>1.20</td>
<td>8.2.9</td>
<td>5+14:39:22</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>ce01.cmsaf.mit.edu</td>
<td>MIT_CMS</td>
<td>1.16</td>
<td>8.4.0</td>
<td>3+23:23:01</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>ce02.cmsaf.mit.edu</td>
<td>MIT_CMS_2</td>
<td>1.16</td>
<td>8.4.0</td>
<td>2+17:10:16</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>ce02.cmsaf.mit.edu</td>
<td>MIT_CMS</td>
<td>2.0.0</td>
<td>8.4.0</td>
<td>2+22:50:44</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>cms-cel1-osg.rcac.purdue.edu</td>
<td>Purdue-Hadoop-HTCE</td>
<td>1.20</td>
<td>8.2.10</td>
<td>8+13:43:38</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>cms-cel2-osg.rcac.purdue.edu</td>
<td>Purdue-Hadoop-HT-PBS-CE</td>
<td>2.0.0</td>
<td>8.4.3</td>
<td>3+18:18:03</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>cms-grid00.hep.uprm.edu</td>
<td>uprm-cms-ce</td>
<td>1.14</td>
<td>8.2.8</td>
<td>73+49:07:04</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>cms.rc.ufl.edu</td>
<td>UF-Florida-CMS</td>
<td>1.20</td>
<td>8.4.2</td>
<td>5+10:58:50</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>cmsgrid01.hep.wisc.edu</td>
<td>GLOW</td>
<td>1.20</td>
<td>8.4.2</td>
<td>13+10:01:29</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>cmsgrid02.hep.wisc.edu</td>
<td>GLOW-CMS</td>
<td>1.20</td>
<td>8.4.2</td>
<td>14+04:30:03</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>cmsgrid03.hep.wisc.edu</td>
<td>GLOW-CONDOR-CE</td>
<td>1.20</td>
<td>8.4.2</td>
<td>23+17:30:32</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>cmsosgc.efnl.gov</td>
<td>cmsosgc.efnl.gov</td>
<td>2.0.0</td>
<td>8.2.8</td>
<td>3+21:47:21</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>cmsosgc2.efnl.gov</td>
<td>cmsosgc2.efnl.gov</td>
<td>2.0.0</td>
<td>8.2.8</td>
<td>3+21:42:39</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>cmsosgc3.efnl.gov</td>
<td>cmsosgc3.efnl.gov</td>
<td>2.0.0</td>
<td>8.2.8</td>
<td>3+21:33:19</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>cmsosgc4.efnl.gov</td>
<td>cmsosgc4.efnl.gov</td>
<td>2.0.0</td>
<td>8.2.8</td>
<td>3+21:32:39</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>cmstest1.rcac.purdue.edu</td>
<td>Purdue-Hadoop-TestCE</td>
<td>1.20</td>
<td>8.4.3</td>
<td>12+48:02:35</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>crane-gw1.unl.edu</td>
<td>Crane-CE1</td>
<td>2.0.0</td>
<td>8.3.5</td>
<td>11+15:10:02</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>gate02.grid.umich.edu</td>
<td>AGLT2_CE</td>
<td>2.0.0</td>
<td>8.4.3</td>
<td>4+22:31:20</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>gate03.aglt2.org</td>
<td>AGLT2_TEST_CE</td>
<td>2.0.0</td>
<td>8.4.3</td>
<td>4+22:14:54</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>gate04.aglt2.org</td>
<td>AGLT2_SL6</td>
<td>1.15</td>
<td>8.2.9</td>
<td>11+10:45:15</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>globus1.hyak.washington.edu</td>
<td>Hyak_CE</td>
<td>2.0.0</td>
<td>8.4.3</td>
<td>4+22:31:20</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>pge01.fnal.gov</td>
<td>pge01.fnal.gov</td>
<td>2.0.0</td>
<td>8.2.8</td>
<td>46+04:12:17</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>pge02.fnal.gov</td>
<td>pge02.fnal.gov</td>
<td>2.0.0</td>
<td>8.2.8</td>
<td>2+22:12:33</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>gridgk01.rcac.bnl.gov</td>
<td>BNL_ATLAS_1</td>
<td>1.10</td>
<td>8.2.7</td>
<td>24+01:25:15</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>gridgk08.rcac.bnl.gov</td>
<td>BNL_ATLAS_8</td>
<td>1.16</td>
<td>8.2.7</td>
<td>9+16:46:50</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>gridtest02.rcac.bnl.gov</td>
<td>BNL_T2_CE</td>
<td>2.0.0</td>
<td>8.2.8</td>
<td>8+16:49:42</td>
<td>3:40:36:43</td>
</tr>
<tr>
<td>hadoop-osg.rcac.purdue.edu</td>
<td>Purdue-Hadoop-CE</td>
<td>1.20</td>
<td>8.4.3</td>
<td>3+21:53:52</td>
<td>3:40:36:43</td>
</tr>
</tbody>
</table>
User education and training

• A little bit of user education goes a long way!
  • While we have dozens of “circuit breakers” in HTCondor to prevent more common mistakes, it helps if the user doesn’t make them in the first place.

• A handful of topics to make your life easier (beyond the “standard intro”):
  • How to avoid invoking condor_q?
  • How long to wait for a job to start / what to do when a job is idle?
  • What’s an “excessive” number of jobs in the queue?
User Education - Userlog files

• HTCondor users love to write the following code to submit or monitor jobs:

```bash
while true
  if [ `condor_q bbockelm -run | wc -l` -lt 100 ]; then
    condor_submit some_file
  fi
sleep 1
done
```

• This is unnecessarily wasteful of schedd resources; if enough users do the same thing, the schedd may become unresponsive.

• Instead, take advantage of the user logs which are typically available locally and record the job lifetime.

• Users don’t even need to parse them - utilize `condor_wait` instead!

• `condor_dagman` will do this automatically for you!
Parting Thoughts

• In the latest stable series, the best scalability tunings come out-of-the-box.

• Building a successful submit host is mostly about how users interact with condor - filesystems & IO, inserting appropriate default attributes.

• Make sure you have both accounting and monitoring in the planning from the beginning.
Questions?