HTCondor at

Collin Mehring
Using HTCondor Since 2011
Animation Studio Background

- Productions are our customers
  - Artists are the end users
- Production stages and their teams
  - Layout -> Animation -> Lighting / FX -> Finaling
- The production hierarchy - Production -> Sequence -> Shot -> Frames
  - Frames are composed of many steps composited together
  - Each frame has a left- and right-eye version for 3D effect
  - ~260k frames in a movie
- Support many different applications
- Hard deadlines
  - Leads to large amounts of work during crunch time
Who interacts with HTCondor and how?

- **Artists**
  - Submit to the farm and expect frames back
  - Focus on the art, no technical knowledge of HTCondor required

- **Technical Directors**
  - Configure artists’ software to use submission tools
  - Debug issues on the shot setup side

- **TRAs (Technical Resource Admins / Render Wranglers)**
  - Manage the HTCondor farm jobs
  - Answer artists’ questions about the farm, and provide help

- **JoSE (Job Submission and Execution, R&D team)**
  - Configure HTCondor
  - Develop and maintain tools to help the TRAs manage the farm
  - Developing submission tools
Why do we configure HTCondor the way we do?

- End users shouldn’t require any technical knowledge of the scheduling system
  - Available settings should be things they care about, everything else is automatic
- The scheduling system should not noticeably impact the end users
- Admins should be able to easily manage large amounts of jobs
- Admins should have easy access to all relevant information and statistics
  - Easier troubleshooting, helps establish causation, and present information to productions
- Prioritize throughput, but consider turnaround time as well
  - Minimize wasted compute hours
  - New renderer scales very well with cores, prioritize scheduling large jobs
- Accounting groups should always get their minimum allocation
- Help productions meet deadlines anyway possible
How do we have HTCondor configured?

- All DAG jobs
  - Many steps involved in rendering a frame
- GroupId.NodeId.JobId instead of ClusterId
  - Easier communication between departments
- No preemption (yet)
  - Deadlines are important - No lost work
  - Checkpointing coming soon in new renderer
- Heavy use of group accounting
  - Render Units (RU), the scaled core-hour
  - Productions pay for their share of the farm
- Execution host configuration profiles
  - e.g. Desktops only run jobs at night
  - Easy deployment and profile switching
- Load data from JobLog/Spool files into Postgres, Influx, and analytics databases

Quick Facts

- Central Manager and backup (HA)
  - On separate physical servers
- One Schedd per show, scaling up to ten
  - Split across two physical servers
- About 1400 execution hosts
  - ~45k server cores, ~15k desktop cores
  - Almost all partitionable slots
- Complete an average of 160k jobs daily
- An average frame takes 1200 core hours over its lifecycle
- Trolls took ~60 million core-hours
What additional configuration have we added?

- Lots of additional ClassAd attributes (~50)
- Concurrency limits
  - Each group has their own limit
  - Software limits can be per host, and can be released early
- Error & Production Error status
  - Differentiating between held and errored jobs
- Subway - Python submission API
  - In terms of studio specific constructs
  - Deferred submissions, v4 provides a REST API
- Job Policy
  - Predefined templates of several job attributes
- Heavy use of pre- and post-priorities
How do we manage our HTCondor pool?

The Farm Manager (WebApp)

- GUI for managing the HTCondor pool
  - Used by TRAs, TDs, Artists, etc.
- See specific details
  - Group progress
  - Job stats and information
    - Logs, charts, etc.
  - Finished and Canceled jobs
- Perform actions on jobs
  - Supports batched actions on nodes & groups
  - Can modify jobs that haven’t been submitted yet by the DAG
- Filter your view
  - Only see the groups relevant to you
- Hides most low-level HTCondor data
  - ClassAds, DAGs, SDFs, etc.
- Allocate resources between shares
  - Separate allocations for day and night
- Monitor execution hosts
  - Data and charts, just like jobs
- Links to other monitoring tools
### Allocation Manager

**Production:** dragon3  
**Total RU:** 28000

<table>
<thead>
<tr>
<th>Name</th>
<th>Production</th>
<th>Batch Alloc</th>
<th>Day Alloc</th>
</tr>
</thead>
<tbody>
<tr>
<td>dragon3_anim</td>
<td>dragon3</td>
<td>500</td>
<td>500</td>
</tr>
<tr>
<td>dragon3_blade_runner</td>
<td>dragon3</td>
<td>2300</td>
<td>2300</td>
</tr>
<tr>
<td>dragon3_cfx</td>
<td>dragon3</td>
<td>1370</td>
<td>1370</td>
</tr>
<tr>
<td>dragon3_cfx_mcrt</td>
<td>dragon3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>dragon3_char_td</td>
<td>dragon3</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>dragon3_crowd</td>
<td>dragon3</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>dragon3_cycles</td>
<td>dragon3</td>
<td>500</td>
<td>500</td>
</tr>
<tr>
<td>dragon3_dwa_logo</td>
<td>dragon3</td>
<td>5000</td>
<td>5000</td>
</tr>
<tr>
<td>dragon3_fargo</td>
<td>dragon3</td>
<td>2300</td>
<td>2300</td>
</tr>
<tr>
<td>dragon3_flo</td>
<td>dragon3</td>
<td>1400</td>
<td>1400</td>
</tr>
<tr>
<td>dragon3_fx</td>
<td>dragon3</td>
<td>3327</td>
<td>3328</td>
</tr>
<tr>
<td>dragon3_fx_mcrt</td>
<td>dragon3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>dragon3_handoff</td>
<td>dragon3</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>dragon3_iml</td>
<td>dragon3</td>
<td>400</td>
<td>400</td>
</tr>
<tr>
<td>dragon3_light_mcrt</td>
<td>dragon3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>dragon3_lod</td>
<td>dragon3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>dragon3_mickey</td>
<td>dragon3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>dragon3_misc</td>
<td>dragon3</td>
<td>367</td>
<td>367</td>
</tr>
<tr>
<td>dragon3_modeling</td>
<td>dragon3</td>
<td>20</td>
<td>15</td>
</tr>
<tr>
<td>dragon3_m_paint</td>
<td>dragon3</td>
<td>469</td>
<td>469</td>
</tr>
<tr>
<td>dragon3_m_paint_mcrt</td>
<td>dragon3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>dragon3_previz</td>
<td>dragon3</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>dragon3_rgb</td>
<td>dragon3</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>dragon3_sicaro</td>
<td>dragon3</td>
<td>4025</td>
<td>4025</td>
</tr>
<tr>
<td>dragon3_simfarm</td>
<td>dragon3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>dragon3_skyfall</td>
<td>dragon3</td>
<td>2875</td>
<td>2875</td>
</tr>
<tr>
<td>dragon3_surf</td>
<td>dragon3</td>
<td>299</td>
<td>298</td>
</tr>
<tr>
<td>dragon3_surf_mcrt</td>
<td>dragon3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>dragon3_team1</td>
<td>dragon3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>dragon3_train</td>
<td>dragon3</td>
<td>1709</td>
<td>1709</td>
</tr>
</tbody>
</table>

**Totals:**  
**28000**  
**28000**

### Share Charts

- **Batch Allocation**
- **Daytime Allocation**
How do we monitor pool stats in real-time?

Grafana

- Primarily used by the TRAs / Render Wranglers
- Quickly detect issues and receive alerts
- At-a-glance overview of the render farm
- Diagnose problems
  - Correlate events between metrics
- More dashboards for specific use cases
  - Software license usage, HTCondor negotiator stats, etc.
Viewing Historical Data

Tableau

- **Big Picture**
  - Trends over time
  - Comparison between productions

- **Used primarily for scheduling**
  - Can we fit all of the rendering we’re planning on doing into the render farm concurrently?
  - How do we move things around to make it all fit?
  - Are there areas we can optimize to better use the existing farm resources?
  - Are we still on schedule?

- **Historical data stored in a separate database**
RU Per Frame

- Shows historically how much compute is being used for each sequence
- Tracks overall trends and identifies complex sequences
- Useful for scheduling production work, allocating resources between teams
Sequence-Shot Details

- Shows RU usage for every farm job, broken down by sequence and shot
- Useful for identifying outliers and specific issues
Overnight Rendering Summary

- Tracks nightly render farm performance
- Number of jobs submitted by each production
  - Grouped by priority, with percent completed
- Amount of RU used by each production compared to their allocations, broken down by team
- Total RU used compared to capacity, broken down by production
- Proportion of capacity allocated to each production compared to what they actually used
- Memory usage compared to capacity
Question Time