Movie Rendering Service

Samantha Rack
Initial Plan - Maya Animation Rendering

- Straightforward web interface for students in Visual FX
  - Renderings of final projects for the class take hours to days
Obstacles

- Limited number of licenses for Maya software
  - Only 30 Notre Dame machines can run the software simultaneously
- Animation dependencies prevent parallel rendering
- Optimal job length hard to achieve
  - Different frames of the same project can have drastically different render times
- Multiple versions of Maya that are not compatible
- Huge number of render options
Resulting Project - Upgraded POVRay Rendering

- Straightforward web interface for submission instead of terminal access
- Management of many job submissions from different clients
- Length of jobs tailored to Condor’s “ideal” job running time
- Awareness of Condor cluster’s status
System Overview
while (1):

    // get current status of condor pool
    avail = condor_status(LINUX, avail)

    // check how many jobs are running
    running = db.jobs_running_count()

    num_start = min(avail / factor, max_ps - running)

    for i = 0 to num_start:
        db.update(job, RUNNING)
        fork('condor_povray', job)

    // wait to check db again
    sleep(timeout)
condor_povray processes

// locally render one frame
one_frame_time = time('povray file.pov -K0')

// calculate optimal number of frames per condor job
frames_per_job = (30*60) / one_frame_time

// build submit files to render all frames
generate_and_submit(frames_per_job, tot_frames)

// wait for completion, the build movie
condor_wait()

db.update(job, DONE)

ffmpeg(frames)
Summary

- Greater overall throughput from resources through designing Condor jobs carefully
- Gatekeeping in system accepts many jobs but is not overwhelmed