glideinWMS Training

HTCondor scalability testing hints

by Igor Sfiligoi, UC San Diego

Overview

- These slides provide a few hints on how to do a HTCondor scalability test of a glideinWMS instance

What to test?

- There are several potential choke points in HTCondor
  - Central Collector
  - Negotiator
  - CCB
  - Schedd
  - HA
- A single test will not cover them all

Number of idle jobs

- Pushing a large number of jobs in the queue(s) will stress the Schedd and the Negotiator
  - Especially if done with many users, and
  - Having many autoclusters
- To be tested with at least moderate running job nr
- Things to monitor
  - Memory consumption
  - Blocking behavior (e.g. internal housekeeping)
  - Response times (submit and query)
  - Negotiation times

Number of idle jobs

- Does not need a complex setup
  - A few dedicated, but high memory and high CPU nodes should be enough
- Adding network latency desirable
  - http://www.linuxfoundation.org/collaborate/workgroups/networking/netem

Number of slots/running jobs

- Getting as many slots as possible, and run jobs on them
  - This will stress all parts of the system, but in different ways
- Central collector may become either CPU or memory starved
  - Has to keep track of all the slots
  - Especially if complex/big attributes are present in ClassAds

Number of slots/running jobs

- Secondary collectors/CCB
  - There will be a large number of connections to handle
  - May run into FD or networking issues
  - May need to either increase the number of CCBs or tweak system parameters (e.g. more FDs per proc.)
- Negotiator
  - High churn of glideins starting and ending will likely put significant load on the Negotiator
  - Monitor Negotiator times, in various stages

Number of slots/running jobs

- Schedd nodes
  - Schedds are way less scalable than Collectors, will likely need to use many of them
- Single schedd node config
  - Purely test how the schedd+shadows handle the load in idealized conditions
  - Push until it breaks
  - Monitor for blocking behavior, and response times

Number of slots/running jobs

- Multi schedd test run, three main reasons for it
  - Only way to test Collector and Negotiator scalability
  - Negotiator & CCB load may affect Schedd performance
  - One bad schedd can significantly affect other good schedds
- Two test modes needed
  - Approx. equal distribution of load among all nodes
  - Push a couple schedds close to limit, others low load
    - Manage high load with MAX_RUNNING

Number of slots/running jobs

- Test setup will require significant number of execute nodes
- Two options:
  - Sleeper pools, with “vanilla” glideins
  - Real resources, with “inflated” glideins
    - i.e. configure glideins to advertise N slots per real CPU
    - Factor 16 should not be unreasonable
- Assuming pure sleep jobs
  - If/when file transfers need to be tested, one has to be significantly more conservative

Slight inflation could be possible here, depending on sleep setup

Number of glideins

- This is a variation of previous test
  - But requires the actual number of glideins to be very high
- This will increase load on the CCBs
  - More connections
- And allows for testing of partitionable slots
  - Needed when there is a mix of multi-core and single-core jobs

Number of glideins

- Two options to get there
  - Bigger sleeper pools
  - Hacking the glidein_startup script to start multiple HTCondor instances in a single glidein
- Hacking is not that hard
  - But does not come out of the box

Recovery on outage

- This test requires a node to be killed, and restarted some time later
- Two useful tests
  - Within minutes
  - After 1+ hour

Mimics maintenance

Mimics HW problem

- Want to monitor how this affect the system
  - Ideally performed while having lots of slots in the system
- Both Schedd and CM nodes should be tested

HA recovery

- A variation of the test is to have the Central Manager in HA mode
  - Same tests, including CM and Schedd outage
- Monitor if HA actually helps or not
  - Both when one HA nodes goes down, and
  - When a “dead” HA node comes back

HA + large number of slots

- HA introduces more complexity in the system
- Repeat all the scalability tests in this mode to measure if it has any adverse affect at all

Open Science Grid

University of California San Diego

The end