glideinWMS Training

Known HTCondor break points

by Igor Sfiligoi, US Can Diego

Overview

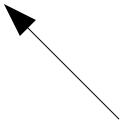
- These slides present a set of known scenarios, where HTCondor breaks down under stress

Blocking connections

- HTCondor still has many places where it blocks reading from a network connection
  - It has a timeout, but it is ~20s
- Most HTCondor daemons very busy
  - A blocking of more than a couple seconds can push the system over the wall
  - Critical for both schedds and CCBs
- Can be monitored by looking for time holes in the Condor logs, and timeout messages

Out of file descriptors

- If a process hits the fd limit, all kind of problems ensue
- CCB the most critical here
  - Since it keeps many long lived TCP connections
  - Others usually OK unless under malicious attack
- CCB/Collector by the numbers
  - Current situation: 5 fds per glidein
  - With shared port: 3 fds per glidein



Assuming system limits high enough

Really long negotiation times

- Negotiation times must be under 15 mins
  - Or a glidein pool gets into a self destructing mode
- Typical culprits
  - Autocluster explosion
  - One (or more) sick schedd, replying very slowly
- Several knobs in HTCondor to reduce damage
  - NEGOTIATOR_TIMEOUT, NEGOTIATOR_MAX_TIME_PER_CYCLE, NEGOTIATOR_MAX_TIME_PER_SUBMITTER, NEGOTIATOR_MAX_TIME_PER_PIESPIN, NEGOTIATOR_TRIM_SHUTDOWN_THRESHOLD
  - But fixing the root cause better

Side effect is either unclaimed slots or jobs never matched

Disk IO limits

- Disks have limited IO capabilities
  - If IOWait% gets high, things start to fall apart
- Two main culprits
  - Logging
  - Job file transfer
- HTCondor can cap job file transfers
  - MAX_CONCURRENT_UPLOADS, MAX_CONCURRENT_DOWNLOADS, FILE_TRANSFER_DISK_LOAD_THROTTLE
  - Can be monitored in schedd classads

Side effect is slots in Claimed/Idle

Insufficient CPU power

- All major HTCondor daemons are CPU hungry
  - Assuming not limited by Network
- If not enough CPU power available, they may get behind and melt down (i.e. hit the wall) Possible reasons:
  - Too small HW for HTCondor alone
  - Non-HTCondor related CPU use (e.g. users)
- System should always have significant Idle%
- All daemons also single threaded
  - Fast single core performance important

Memory exhaustion

- Condor very sensitive
  - Due to heavy memory fragmentation
- Once a node starts to swap, things slow down drastically
  - Usually leading to a complete meltdown of the HTCondor daemons on the node
- Schedd node the most affected
  - Each shadow uses about 1 MB of memory (varies by HTCondor version and job type)
  - Most active during job startup and termination

Firewalls

- Most HTCondor daemons manage lots of network connections
  - Easy to overwhelm firewalls, if they are used
- Ideally, operate without firewalls
  - Unless you have other services that you need protected
- Else, make sure you tune the firewall
  - ip_conntrack_max

Glidein reconnects

- After a network problem, a glidein will try to contact the Schedd to re-establish the connection
- Sometimes it attempts a full x509 handshake
  - Most of the time, it (correctly) uses the shared secret
- If many glideins choose to do this, they can effectively kill the Schedd

Excessive querying

- Every new authenticated communication to the HTCondor daemons is expensive
  - This includes queries (condor_q and condor_status)
- glideinWMS daemons are careful to not overdo it
  - But users may
- Too many queries can effectively kill a daemon
  - Something to monitor and act on if needed

Hitting process limits

- HTCondor spawns many processes
  - Especially on the schedd nodes
  - Make sure system limits are high enough
- RHEL6 comes with very low per-user limits
  - That may hurt both schedd and collector nodes
  - Should be raised to 16k+ on any busy system
    - /etc/security/limits.d/*nproc*.conf

Open Science Grid

University of California San Diego

The end