HTCondor LXBatch

Ben Jones IT-CM
Agenda

• LSF to HTCondor Status
• Priorities & Preemption
• Multicore & Draining
• Cloud Resources
• BEER
Background

• HTCondor introduced as new production batch service
• Replaces LSF, a proprietary product, with an Open Source product
• HTCondor now has more than double the capacity of LSF
  • 100k+ cores in HTCondor
  • 33k cores in LSF share, ~17k in ATLAS T0
• We haven’t really started reducing LSF in anger (yet!)
Scale

- LSF has a fixed maximum capacity of ~5k hosts
  - Due to limitation, LSF worker nodes are bigger (typically 16 core)
  - We know that "virtualisation overhead" for 16 core is ~3% whereas it’s negligible for 8 core
- HTCondor 100k+ cores with 8/10 core machines would be impossible with LSF
- CMS global pool bigger HTCondor scale, but we have different requirements (local kerberos submissions)
Production Experience

- Still very happy with htcondor-ce
  - Routing ability very helpful (and highly used)
  - Some issues around memory bloat resolved upstream
- CGroups used for memory (soft) limits
- Scaling service requiring work on Collectors & Negotiator
  - We are currently profiting from the work colleagues in CMS & Upstream have put into scale
Symmetric job matching

- **Schedd**
  - Send jobs to reserved pslot
  - Submit side

- **Collector**
  - Pull list of idle jobs

- **Negotiator**
  - Broker

- **Startd**
  - Send machine properties (ClassAds)
  - Execute Side
Why query the Name if it’s the same as the Schedd fqdn?
• Because it isn’t always – HA schedds publish names
Collector Bottleneck

Schedd → Collector

Pull list of idle jobs

Negotiator

→ Startd

Send machine properties (ClassAds)

Execute Side

Submit side

Send jobs to reserved pslot

Broker
Split the Collectors

Schedd

Submit side

Send jobs to reserved pslot

Collector

Negotiator

Pull list of idle jobs

Broker

Sub Collector

Send machine properties (ClassAds)

Startd

Execute Side
Splitting infrastructure

• Moving to sub collectors has reduced the times when the Collector is too busy to reply with the name of the schedd

• Still work to do! The next step is to scale out the Negotiator
  • Negotiator does the matching of jobs to machines
  • Long negotiation cycle also affects the Collector
  • Splitting pool between two negotiators
Priorities & Preemption

• Currently in production we have no preemption
  • Historically the majority of experiment use cases don’t allow for preemption
• Job priorities used for scheduling / negotiation but not for preemption
• Equal priority for Accounting Groups
  • Fairshare changes effective priority
• What could we do with priorities / preemption?
Nice User jobs

```
[  
    MaxJobs = 500;
    MaxIdleJobs = 100;
    TargetUniverse = 5;
    name = "AtlasT0";
    Requirements = (regexp("atlas", x509UserProxyVoName)) && (TARGET.queue == "AtlasT0");
    set_Requirements = (TARGET.Hostgroup == "bi/condor/gridworker/atlast0");
    set_AtlasGridJob = True;
    set_NiceUser = True;
]
```
Nice User jobs

• Via direct submission:
  • `condor_submit --nice_user`

• Via ClassAd:
  • `NiceUser = True`

• Sets priority to a level that means all other jobs will have higher priority

• In T0 this (should) mean that nice jobs only run when there are no idle T0 jobs

• Can use this principle to fill other resources
  • Effectiveness depends on runtime!
Preemption

- Can also preempt
- Job priority vs User priority
  - User priority might make sense for dedicated resources
- Preemption might be useful in similar cases to Nice Jobs (or in addition)
- ClassAds for “special” resources to take Nice & Preemptible jobs?
  - Send to (for example) machines draining for mcore?
Multicore & draining

- Use condor_defrag
  - Decide which machines to drain
  - must not be cloud machines
  - must be healthy
  - must be configured to actually start new jobs
  - must not be an xbatch node
- We only drain 8 cores
- We don’t look ahead into queue and change
- Currently humans & monitoring change amounts
- Steady queue of mcore better for us
Drain monitoring

Whole Machines: 3446
Draining % of Pool: 0%
Peak Draining (All-Time): 143
Peak Draining (This Window): 0
Recently Started Draining: 0

Drain Wastage

Avg Time Unclaimed
Avg Time Badput
Cloud Resources

- Continue to take advantage of Public Cloud Resources
- Oracle Bare Metal Cloud
  - Docker Universe on Bare Metal
  - Limited PoC of 9k cores
- HNScienceCloud
  - Current phase allows for some additional cores
- HTCondor-CE routes
  - remote_queue = externalcloud
  - +Xbatch = True
BEER (Batch on Eos Extra Resources)

- PoC / Pilot to investigate using additional CPUs
  disk servers don’t use
- Goal: ensure that jobs can’t interfere with EOS file server
- HTCondor service in CGroup with memory limit
- Limit cores available to HTCondor
  - Investigating pinning etc but might not be needed
- Docker universe for jobs
  - Decouple EOS host OS from execution
Questions?