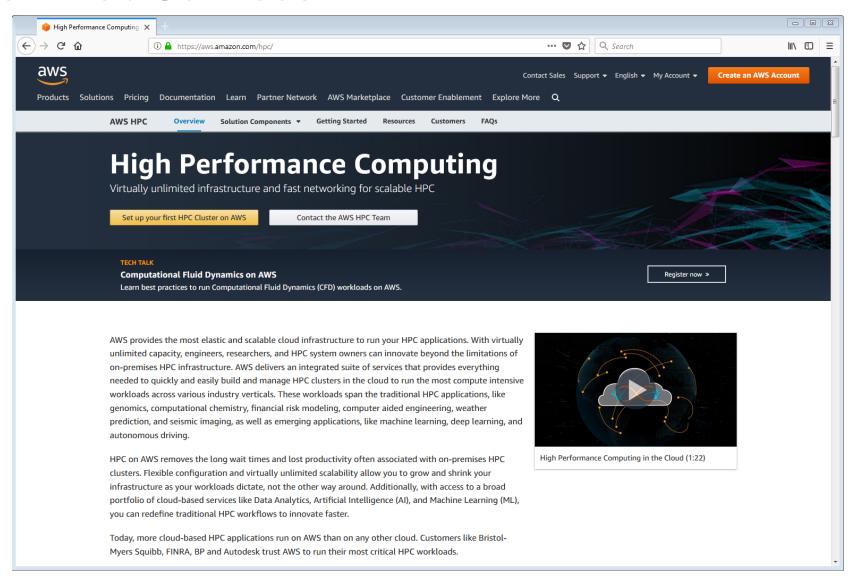
N8 CIR COMPUTATIONALLY INTENSIVE RESEARCH

Success in the Cloud Using HTCondor on AWS

Dr Christopher Paul Research IT, University of Manchester

Amazon Web Services





HTCondor @ University of Manchester



- Established as a service to researchers in Nov. 2009.
- Consists of
 - 24/7 rack servers ("backbone nodes") 1200 cores
 - Student cluster PCs (1300 PCs) overnight, weekends, vacation.
- Available to staff, PhD students and undergraduate project students.

- Currently 190 registered users
- Demand is very "lumpy"
- Jobs run for 5-10 minutes to several months

- Open-source C++ code,
 Geant4, Julia, Python, R.
- Commercial software –
 Matlab, Mathematica.

Integrating AWS with UoM HTCondor service



- Seamless Integration
 - Use existing UoM HTCondor infrastructure separate matchmaker ("admin") and job submitter ("user") nodes.
 - Minimal changes to jobs for "bursting into the cloud".
 - Maximize use of available local HTCondor nodes before "bursting into the cloud".

- Replicate available software applications (dependent on licensing conditions).
- Provision of accounting information for UoM usage and AWS usage.

HTCondor on AWS



- 7 HTCondor AMI images on AWS Marketplace
- HTCondor 8.6.10 on Amazon Linux 2016.09 (v3)
- Instructions at https://research.cs.wisc.edu/ht condor/manual/v8.9.0/HTCond orAnnexUsersGuide.html

- Challenges
 - Personal HTCondor pool
 - Single matchmaker/submitter node
 - Matchmaker/submitter node in AWS
 - AMI uses static HTCondor slots

Adapting the HTCondor AMI image for UoM



Solutions

- Setup a personal Condor pool for user "condor"
- Configure matchmaker node to act as a Condor Connection Broker
- Change the HTCondor slot type to use dynamic slots

- Copy pool password file and use SEC_PASSWORD_FILE in .condor/user_config file
- Add CCB_ADDRESS line to HTCondor clients condor_config.local
- NUM_SLOTS = 1SLOT_TYPE_1_PARTITIONABLE = True

Additional challenges when providing an HTCondor service



- Prevent "large" VMs being "hijacked" by small jobs
- Use local HTCondor clients in preference over AWS.
- Minimize cost by using SpotFleet VMs in cheapest AWS region
- Prevent HTCondor jobs requesting applications not present in AMI

- Tag each HTCondor job cluster and corresponding VMs with unique AnnexName ID
- Only consider queued jobs for AWS if (ClassAD) QueuedTime exceeds 20 minutes
- Calculate 28 days SpotFleet price range and use diversified SpotFleet allocation strategy
- Check job HTCondor ClassADs against ClassADs in AMI

Additional challenges when providing an HTCondor service



- Prevent "failing" jobs from restarting indefinitely
- Need to account for CPU usage on local nodes, "always on" Reserved Instance nodes and "on demand" SpotFleet VMs
- Use condor_hold command if (job ClassAD) NumJobStarts exceeds limit.
- Set CumulativeSlotTime ClassAD to 0 when tagging job.
- Also use ClassADs CommittedTime, CommittedSuspensionTime, RemoteWallClockTime and CumulativeSuspensionTime

OWNER UOM_CPU_USAGE AWS_RI_CPU_USAGE AWS_CPU_USAGE TOTAL_CPU_USAGE xxxxxxxx 0000+00:00:00 0000+21:16:36 0000+00:00:00 0000+21:16:36 yyyyyyyy 0174+18:55:12 0000+00:00:00 0000+00:00:00 0174+18:55:12

Additional challenges when providing an HTCondor service



 Rate limit on ec2-describeinstances/describe-spot-fleetrequests calls in 49ec2instance.sh

 # If you launch as few as 100 instances at a time, it's possible to exceed AWS' rate limit on describe-spot-fleetrequests.

- Put each AWS call in a "repeat until successful" loop
- Reduce risk of hitting rate limit by adding --query filter to AWS calls
- These enchancements will be included in future versions of HTCondor AMI
- Spin-up VMs in batches (50 VMs with 30 seconds gap)

Conclusions / Future Developments



- Seamless integration with existing HTCondor service
- Need to use diversified
 SpotFleet allocation strategy
 (rather than cheapest) to
 maximize number of VMs.
- Reserved Instance VMs overcome issue of limited SpotFleet capacity
- Ideal for handling "lumpy" demand

- Can quickly "deploy" new application - just update AMI
- Ability to target AWS region based on SpotFleet capacity
- Drs Nick Chilton/Daniel Reta -47,000+ jobs ~ 5-7 CPU hours (28-38 CPU years) in 2
 WEEKS for around \$1,500.

Success in the Cloud – using HTCondor on AWS



Acknowledgements:

Dr Daniel Corbett (UoM)
Phil Edwards (AWS)
Charlotte Spiteri (AWS)















