Brookhaven Laboratory Cloud Activities Update

John Hover, Jose Caballero

US ATLAS T2/3 Workshop

Indianapolis, Indiana

Outline

Addendum to November Santa Cruz Status Report

http://indico.cern.ch/conferenceProgram.py?confId=201788

Current BNL Status

- Condor Scaling on EC2
- EC2 Spot Pricing and Condor
- VM Lifecycle with APF
- Cascading multi-target clusters

Next Steps and Plans

Discussion

Condor Scaling 1

RACF recieved a $50K grant from Amazon: Great opportunity to test:

- Condor scaling to thousands of nodes over WAN
- Empirically determine costs

Naive Approach:

- Single Condor host (schedd, collector, etc.)
- Single process for each daemon
- Password authentication
- Condor Connection Broker (CCB)

Result: **Maxed out at ~3000 nodes**

- Collector load causing timeouts of schedd daemon.
- CCB overload?
- Network connections exceeding open file limits
- Collector duty cycle -> .99.

Condor Scaling 2

Refined approach:

- Split schedd from collector,negotiator,CCB
- Run 20 collector processes. Configure startds to randomly choose one. Enable collector reporting, so that all sub-collectors report to one top-level collector (which is not public).
- Tune OS limits: 1M open files,
- Enable shared port daemon on all nodes: multiplexes TCP connections. Results in dozens of connections rather than thousands.
- Enable session auth, so that each connection after the first bypasses password auth check.

Result:

- Smooth operation up to 5000 startds, even with large bursts.
- No disruption of schedd operation on other host.
- Collector duty cycle ~.35. Substantial headroom left. Switching to 7-slot startds would get us to 35000 slots, with marginal additional load.

Condor Scaling 3

Overall results:

- Ran ~5000 nodes for several weeks.
- Production simulation jobs. Stageout to BNL.
- Spent approximately $13K. Only $750 was for data transfer.
- Moderate failure rate due to spot terminations.
- Actual spot price paid very close to baseline, e.g. still less than .$.01/hr for m1.small.
- No solid statistics on efficiency/cost yet, beyond a rough appearance of “competitive.”

EC2 Spot Pricing and Condor

On-demand vs. Spot

- On-Demand: You pay standard price. Never terminates.
- Spot: You declare *maximum* price. You pay current,variable spot price. If/when spot price exceeds your maximum, instance is terminated without warning. Note: NOT like priceline.com, where you pay what you bid.

Problems:

- Memory provided in units of 1.7GB, less than ATLAS standard.
- More memory than needed per “virtual core”
- NOTE: On our private Openstack, we created a 1-core, 2GB RAM instance type--avoiding this problem.

Condor now supports submission of spot-price instance jobs.

- Handles it by making one-time spot request, then cancelling it when fulfilled.

EC2 Types

Type	Memory	VCores	“CUs”	CU/Core	$Spot/hr Typical	$On-Demand/hr	Slots?
m1.small	1.7G	1	1	1	.007	.06	-
m1.medium	3.75G	1	2	2	.013	.12	1
m1.large	7.5G	2	4	2	.026	.24	3
m1.xlarge	15G	4	8	2	.052	.48	7

Issues/Observations:

- We currently bid 3 * <baseline>. Is this optimal?
- Spot is ~1/10th the cost of on-demand. Nodes are ~1/2 as powerful as our dedicated hardware. **Based on estimates of Tier 1 costs, this is competitive.**
- Amazon provides 1.7G memory per CU, not “CPU”. Insufficient for ATLAS work (tested).
- Do 7 slots on m1.xlarge perform economically?

EC2 Spot Considerations

Service and Pricing

- Nodes terminated without warning. (No signal.)
- Partial hours are *not charged*.

Therefore, ATLAS needs to consider:

- Shorter jobs. Simplest approach. Originally ATLAS worked to ensure jobs were *at least* a couple hours, to avoid pilot flow congestion. Now we have the opposite need.
- Checkpointing. Some work in Condor world providing the ability to checkpoint without linking to special libraries. (But not promising.)
- Per-event stageout (event server).

If ATLAS provides sub-hour units of work, we could get significant free time!

VM Lifecycle with APF

Condor scaling test used manually started EC2/Openstack VMs. Now we want APF to manage this:

2 AutoPyFactory (APF) Queues

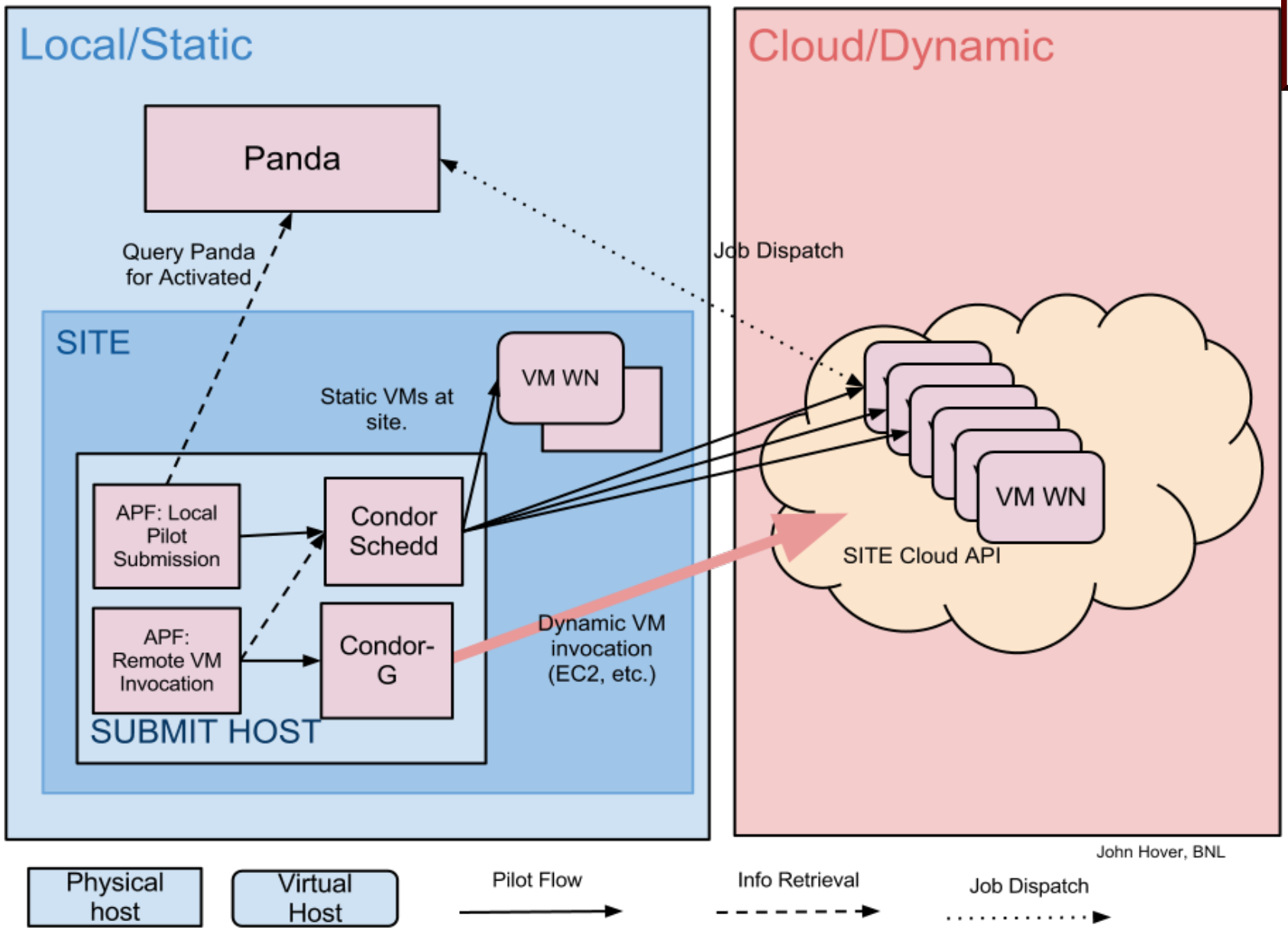
- First (standard) observes a Panda queue, submits pilots to local Condor pool.
- Second observes a local Condor pool, when jobs are Idle, submits WN VMs to IaaS (up to some limit).

Worker Node VMs

- Condor startds join back to local Condor cluster. VMs are identical, don't need public IPs, and don't need to know about each other.

Panda site (BNL_CLOUD)

- Associated with BNL SE, LFC, CVMFS-based releases.
- But no site-internal configuration (NFS, file transfer, etc).


Local/Static
Cloud/Dynamic
Panda
Query Panda for Activated
Job Dispatch
SITE
Static VMs at site.
VM WN
APF: Local Pilot Submission
Condor Schedd
APF: Remote VM Invocation
Condor-G
SUBMIT HOST
Dynamic VM invocation (EC2, etc.)
SITE Cloud API
VM WN
John Hover, BNL
Physical host
Virtual Host
Pilot Flow
Info Retrieval
Job Dispatch

VM Lifecycle 2

Current status:

- **Automatic ramp-up working properly.**
- Submits properly to EC2 and Openstack via separate APF queues.
- Passive draining when Panda queue work completes.
- Out-of-band shutdown and termination via command line tool: Required configuration to allow APF user to retire nodes.

Next step:

- Active ramp-down via retirement from within APF.
- Adds in tricky issue of “un-retirement” during alternation between ramp-up and ramp-down.
- APF issues *condor_off -peaceful -daemon startd -name <host>*
- APF uses *condor_q* and *condor_status* to associate startds with VM jobs. Adds in startd status to VM job info and aggregate statistics.

Next step 2:

- Automatic termination of retired startd VMs. Accomplished by comparing *condor_status* and *condor_status -master* output.

Ultimate Capabilities

APF's intrinsic queue/plugin architecture, and code in development, will allow:

- Multiple targets
  - E.g., EC2 us-east-1, us-west-1, us-west-2 all submitted to in a load-balanced fashion.
- Cascading targets, e.g.:
  - We can preferentially utilize free site clouds (e.g. local Openstack or other academic clouds)
  - Once that is full we submit to EC2 spot-priced nodes.
  - During particularly high demand, submit EC on-demand nodes.
  - Retire and terminate in reverse order.

The various pieces exist and have been tested, but final integration in APF is in progress.

Next Steps/Plans

APF Development

- Complete APF VM Lifecycle Management feature.
- Simplify/refactor Condor-related plugins to reduce repeated code. Fault tolerance.
- Run multi-target workflows. Is more between-queue coordination is necessary in practice.

Controlled Performance/Efficiency/Cost Measurements

- Test m1.xlarge (4 “cores”, 15GB RAM) with 4, 5, 6, and 7 slots.
- Measure “goodput” under various spot pricing schemes. Is 3*<baseline> sensible?
- Google Compute Engine?

Other concerns

- Return to refinement of VM images. Contextualization.

Questions/Discussion