THE UNIVERSITY OF UTAH

Adaptable Profile-Driven TestBed ‘Apt’

Brian Haymore (brian.haymore@utah.edu)

Slurm User Group Meeting
September 15th-16th, 2015

What is Apt?

- Apt (the Adaptable Profile-Driven Testbed) is a new type of facility: a meta-testbed that is adaptable through "profiles” to support a wide range of computing-based research domains. Apt focuses on getting the infrastructure out of the way, giving researchers the ability to create testbed environments ("profiles") that are tailored to their own domains, and to share those environments with their colleagues. Apt targets both researchers in computer science and researchers from other compute-intensive fields. (http://www.flux.utah.edu/project/apt)
    - My definition goes something like this. Apt is a framework for provisioning bare metal, storage and network resources into new and or predefined configurations.
- Apt is based on the Emulab software stack. Emulab is used as the base for other testbeds including GENI, PRObE, and DETER. Emulab is open source released under the GNU Affero General Public License, version 3 (AGPLv3).
- Apt and the Emulab software stack are also foundations for CloudLab. CloudLab is one of the two Cloud Computing Testbeds funded by the NSF, Chameleon is the other.

Apt Experiment Details

- The unit of work in Apt is known as an ‘Experiment’. Experiments are based on a ‘profile’ that defines many things including:
    - Node count and type.
    - Networking & topology.
    - Max time limit per instance of the ‘Profile’.
    - OS Provisioning details.

- When you are ready to start up an ‘Experiment’ you simply instruct the Apt portal, CLI, or make a call to the API asking to have it started. At this point there is essentially a binary result. Either it starts to ‘Swap in’ the ‘Experiment’ or it reports there were not enough resources to accommodate the ‘Experiment’.
    - There is a queue of sorts, within the emulab stack, but at this point it is both very immature and essentially not used.

- Once a job has been ‘Swapped in’ it will continue to run until it reaches the defined time limit or until we proactively ‘Swap out’ the experiment. Other important things to note about running ‘Experiments’ are:
    - Experiments can be modified for resource count, configuration, etc…. HOWEVER all resources are “reprovisioned” in the process…. All your nodes get rebooted…
    - Modifying an experiment to add additional resources does not restart the max time limit so the new resource (the whole experiment) only has the remainder left to live.

Tangent Project Info & Goals

- Collaboration between Flux group of the School of Computing and CHPC at the University of Utah using Flux's 'Adaptable Profile-Driven Testbed' (Apt).
  - 'Tangent' is a Slurm based HPC cluster environment that uses Apt resources dynamically to meet researcher's needs.
  - It is important to note that Apt does not represent an unlimited set of resources nor does Tangent represent the only point of resource consumption.
- Goals & Points of observation:
  - Impact on job turnaround.
  - Reliability/robustness of scheduling and Apt systems.
  - Job prioritization with uncertain resource availability.
  - "Max" job size that has reasonable expectation to run.
  - Job start failure handling when dynamic resources don't come online.
  - Node Health Checking 'just in time' for a job start.
  - Hardware health tracking & resolution.

Slurm’s Elastic Computing & Power Saving Support

- Slurm has built in support for ‘Elastic Computing’ and ‘Power Saving’.
  - Both share common features and functionality that allow a site to control the state of resources dynamically.
  - The current state of this support is somewhat basic in that some assumptions are made and the process for provisioning/deprovisioning is left for the individual site to implement.
  - Core Slurm settings:
    - SuspendProgram: Site provided program that provisions resources.
    - SuspendTime: Time after which an idle resource is to be deprovisioned.
    - SuspendRate: Number of nodes to be deprovisioned per minute max.
    - SuspendTimeout: Amount of time after a suspend call is made before the resource is expected to be able to be resumed.
    - ResumeProgram: Site provided program that deprovisions resources.
    - ResumeRate: Number of nodes to be provisioned per minute max.
    - ResumeTimeout: Amount of time after a resume call is made and when it comes online ready to receive the job that has been allocated to it.
    - BatchStartTimeout: Amount of time after a job start has been issued before we expect the job to be running on the allocated nodes.

Additional things we needed to handle...

- Apt’s experiment model and Slurm’s elastic computing and power savings model don’t see eye to eye…
  - Experiments are effectively a single unit with a single lease timer.
  - Node lists sent to Slurm’s SuspendProgram and ResumeProgram call out are not aligned to jobs, they are simply the nodes to be transitioned that iteration.
  - Common denominator then ends up being single nodes to each Apt experiment.

- The Apt framework, as we discovered, could only handle so many experiments being swapped in and out at the same time.
  - We handle this in the short term by building in some additional rate limiting and retries into the SuspendProgram and ResumeProgram.
  - Longer term discussion to address how to fix things right...

- Apt imposes a Max Duration before an experiment is swapped out.
  - Initially we chose to have nodes allocated to a job turned back into Apt at the end of every job.
  - Later we decided to change things so that Slurm would hang onto the nodes if there were idle jobs that could still be run and finish before the Apt Max Duration was reached.
  - If there were not jobs that could fit in the remaining time then the node(s) were still deprovisioned to avoid waste.

Additional things we needed to handle...

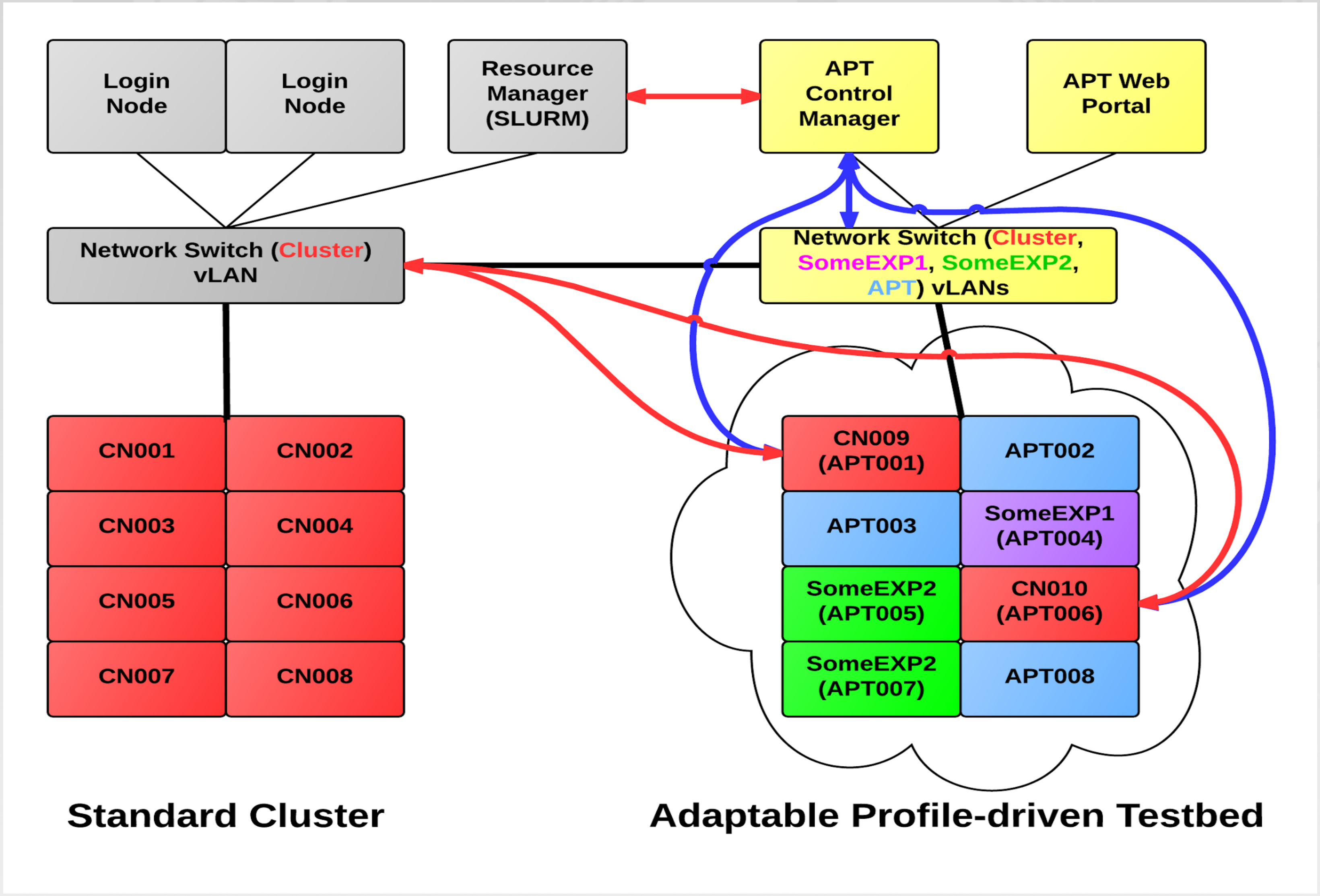
- Slurm does not check for resource availability first. It simply assumes it is available if it has been deprovisioned.
    - We need to let Slurm know when Apt resources are not available (in use by other Apt users) so unavailable nodes are not selected and allocated.
    - We also need to clear the unavailable status when they become available.

- Occasionally Slurm and Apt have gotten out of sync. Apt shows resources allocated to Slurm. However Slurm does not think the nodes are available. To prevent waste we check for this every so often and clear the stale experiment from Apt if one is found.

ResumeProgram, SuspendProgram & 'NodeStateSync'

- ResumeProgram: APTEmulabSwapInExperiment.sh
  - Make experiment swap-In call to Apt for each node passed to 'ResumeProgram' (Swap-in up to 8 concurrently, wait, next 8, etc...).
  - Verify requested nodes are now swapped in (Reissue swap in as needed).
  - Loop through node-list from slurm and create 'tail-end' reservation for each node.
    - Ensures that node becomes idle before the Apt Max Duration limit such that Slurm will issue a 'SuspendProgram' call and swap out that experiment.
- SuspendProgram: APTEmulabSwapOutExperiment.sh
  - Make experiment swap-out calls to Apt for each node passed to 'SuspendProgram' (Swap-out up to 8 concurrently, wait, next 8, etc…).
  - Verify requested nodes are swapped out (Reissue swap out as needed).
  - Loop through node-list from slurm and delete the 'tail-end' reservation for each node.
- Extra Script: NodeStateSync.sh
  - Pull node state and experiment state from Apt
    - Mark all nodes not available AND not allocated to us as 'drained' in slurm.
    - Clear drained state from nodes that have become available.

Operational Diagram



Observations so far...

- The general response both from users and our own observations are that Tangent is working really quite well. Most bumps reported are due to a lack of patience vs any real problem.
- Job start tax is about 3-5 minutes. The swap out time takes a bit longer due to extra "clean up" steps Apt takes on a node before it is available again.
- The max job size that will likely get started has varied between the low 20s to the upper 40s of the 64 nodes in the pool we are using from Apt.
- Slurm is a bit too effective when it comes to competing against other users of Apt. We have the potential to dominate the resources. We need a better mechanism at the Apt level to help prioritize and throttle usage.
- We ran into some reliability issues when we tried to swap in or out nodes too fast from Apt. We have worked around this in our scripts and we also have been working with the Apt team to address this in a better way.
- Our current setup can attempt to swap in nodes that are not actually available. While we may not be able to eliminate this possibility entirely, we would like to narrow the chance.
    - First is in the time gap between 'NodeStateSync.sh' executions, about 15 seconds.
    - Second is when Slurm asks to turn on a large set of nodes. Here with how we currently have to stagger individual node swap in calls we have a chance to have another user pick up one of the nodes we have not yet made the swap in call for.

Future Directions

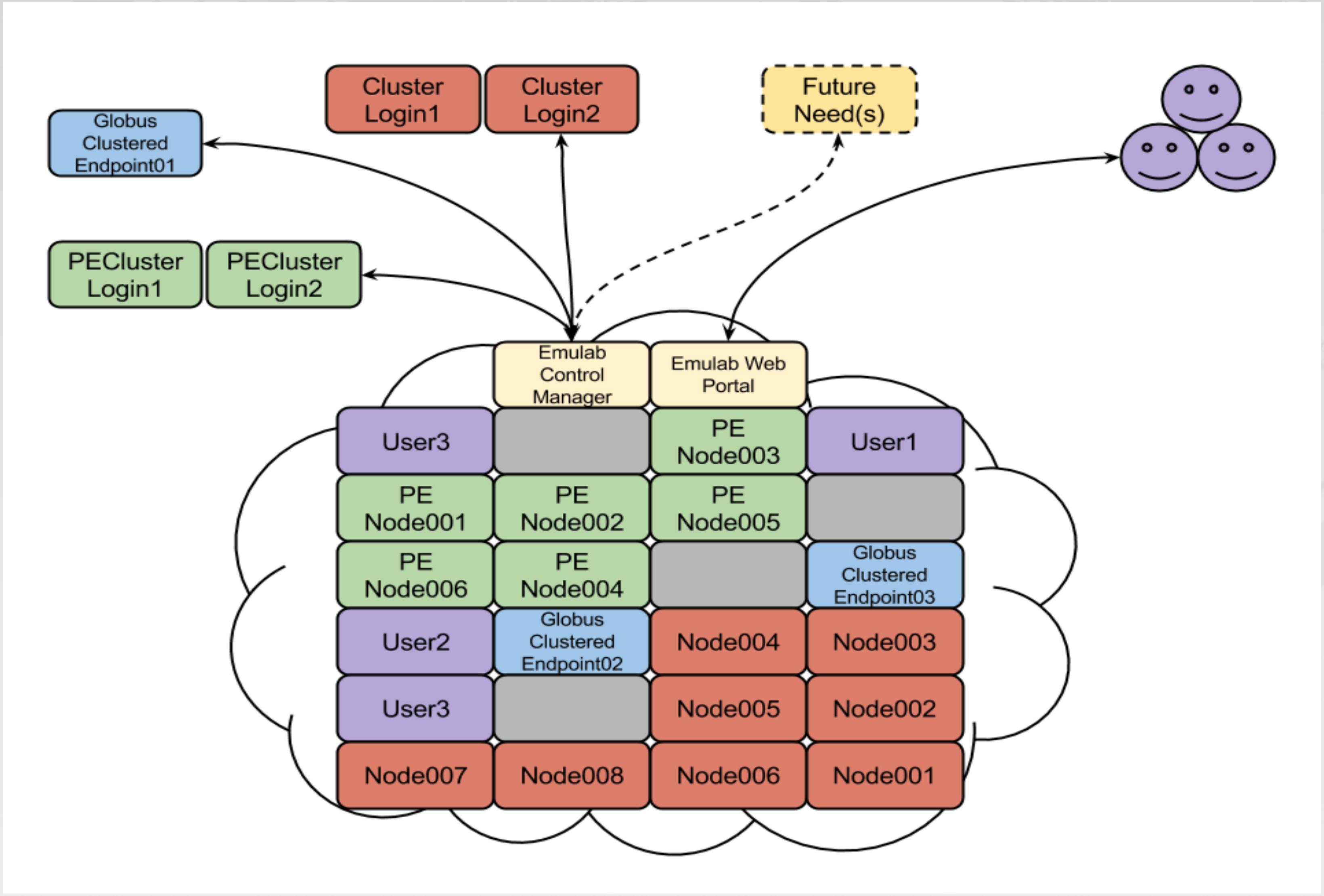
- Realizing that currently there are not reasonable means for having Slurm check on the status of each node during the node selection process with Apt… Are there things we can do to help minimize the times that an incorrect node is selected while NOT creating an unreasonable load on Apt?

- Can the Apt framework satisfy the needs for support a mixed usage HPC & research computing environment?
  - Apt layer manages the physical systems and network access.
  - Multiple clusters could pull from a common pool of resources.
  - Potentially supporting regulated and non-regulated data environments.
  - Must satisfy any ‘sanitization’ needed on a resource that moves from a regulated to a non-regulated role.

- Somewhat related, but can Software Defined Networking be included into the mix to help better isolate systems involved in regulated work and limit network access to just meet the needs of the running jobs.

THE UNIVERSITY OF UTAH

Possible Resource Model



Future Directions

- The Flux group has been very focused on improving the Emulab software layers both to meet current users needs as well as to address the larger scale up required to support the new NSF funded CloudLab. As part of our relationship with them on Apt, together we have identified a few areas that they are working on improving:
  - Address scaling issues we encountered when too many transactions per unit time.
  - Modify aspects of their experiment model so that we can look at a single larger experiment vs many single node experiments.
    - Allow an experiment to be modified (nodes added/removed) without rebooting the rest of the nodes.
    - Create a per resource time limit instead of a single per experiment.
  - Ability to “call home” resources they want back (Crude prioritization/preemption).
    - Implemented both as a soft request and a hard reclaim.
    - Related to this… Wish list item for slurm. Allow us to include a short message when an admin issues a scancel that is conveyed to the user.

- Can Apt gain some additional abilities such as:
  - meaningful queue.
  - prioritization of resource requests in the queue.
  - advanced reservations of resources.
  - fairshare and allocation limits.
  - … looks like a scheduler… can’t we just use one?

Demo & Questions

Time permitting we can do a quick demo…

Any Questions?

Thank you for your attention.

Contact Info/Links

- Brian Haymore: - brian.haymore@utah.edu
- University of Utah’s Center for High Performance Computing: - www.chpc.utah.edu
- Univeristy of Utah’s Flux Group: - www.flux.utah.edu
- Adaptable Profile-Driven Testbed: - www.flux.utah.edu/project/apt
- Emulab: - www.flux.utah.edu/project/emulab
- Geni: - www.geni.net
- PRObE: - www.nmc-probe.org
- Deter: - www.isi.deterlab.net
- CloudLab: - cloudlab.us
- Chameleon: - www.chameleoncloud.org