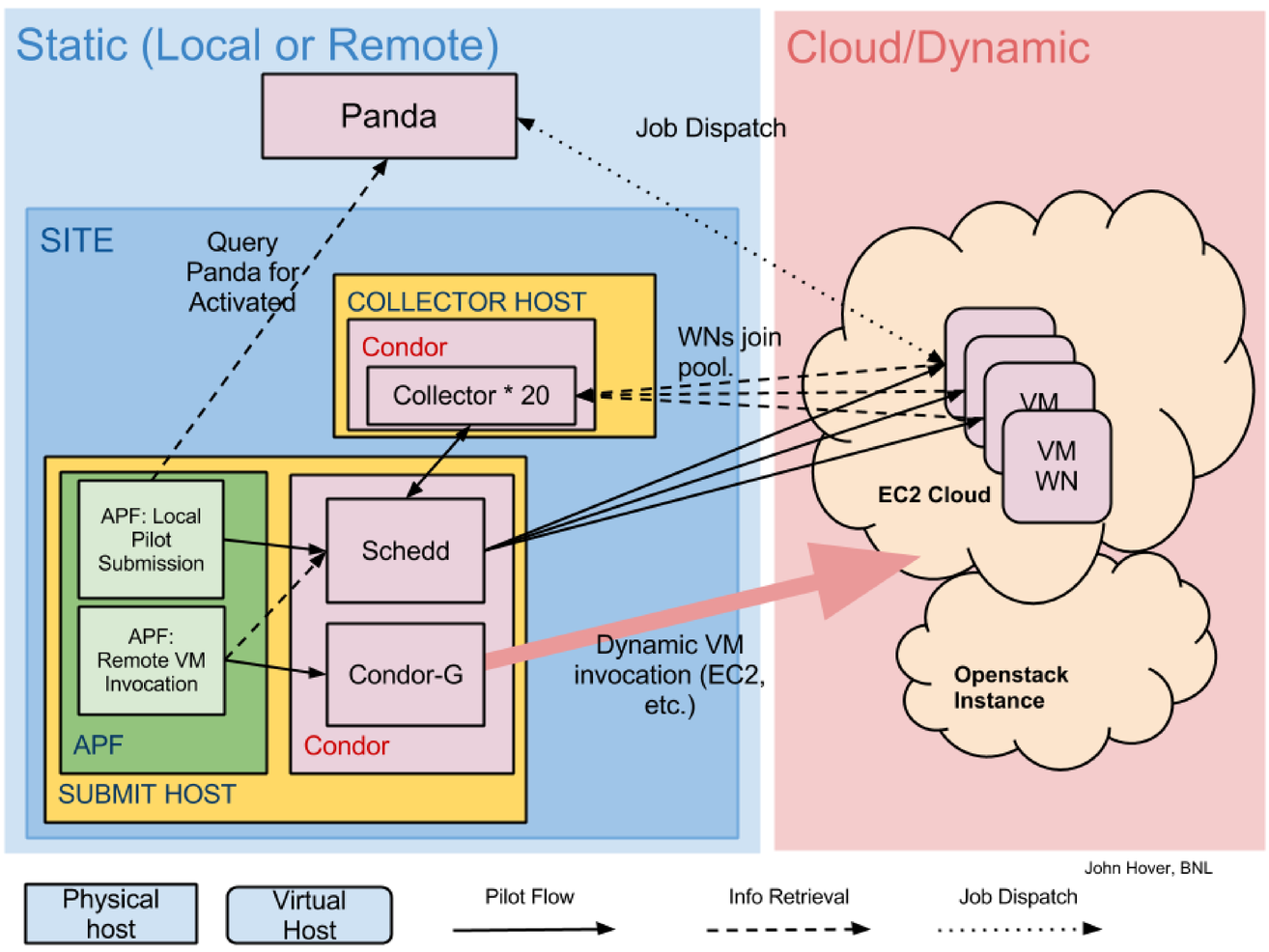
AutoPyFactory and the Cloud: Flexible and scalable management of virtual resources for ATLAS

J. Caballero [a], J. Hover [a], P. Love [b]
on behalf of the ATLAS collaboration.
[a]Brookhaven National Laboratory, [b]University of Lancaster

AutoPyFactory (APF) is a next-generation pilot submission framework that has been used as part of the ATLAS workload management system (PanDA) for two years. APF is reliable, scalable, and offers easy and flexible configuration. Using a plugin-based architecture, APF polls for information from configured information and batch systems (including grid sites), decides how many additional pilot jobs are needed, and submits them.

With the advent of cloud computing, providing resources goes beyond submitting pilots to grid sites. Now, the resources on which the pilot will run also need to be managed. Handling both pilot submission and controlling the virtual machine life cycle (creation, retirement, and termination) from the same framework allows robust and efficient management of the process. .



Although we have described the use of APF to manage cloud-based resources, this could be used equally well to manage a facility's virtualized, or even physical, resources. As long as the desired creation/retirement/termination actions can be placed in scripts, one could create an APF submit plugin to invoke them.

Below:

The diagram describes how pro-active retirement is managed. The key functionality is enabled by APF associating a VM job with a particular execute node that has joined the cluster by way of the VM *instance ID*. The ID must be explicitly queried from within the VM and advertised via the execute hosts Condor ClassAd mechanism.

Two-queue coordination: Local batch system and remote VMs

At left:

One APF queue is configured to watch the Panda WMS for ready work. It submits pilot jobs to a local Condor batch queue. This queue has no execute nodes, so the jobs remain idle.

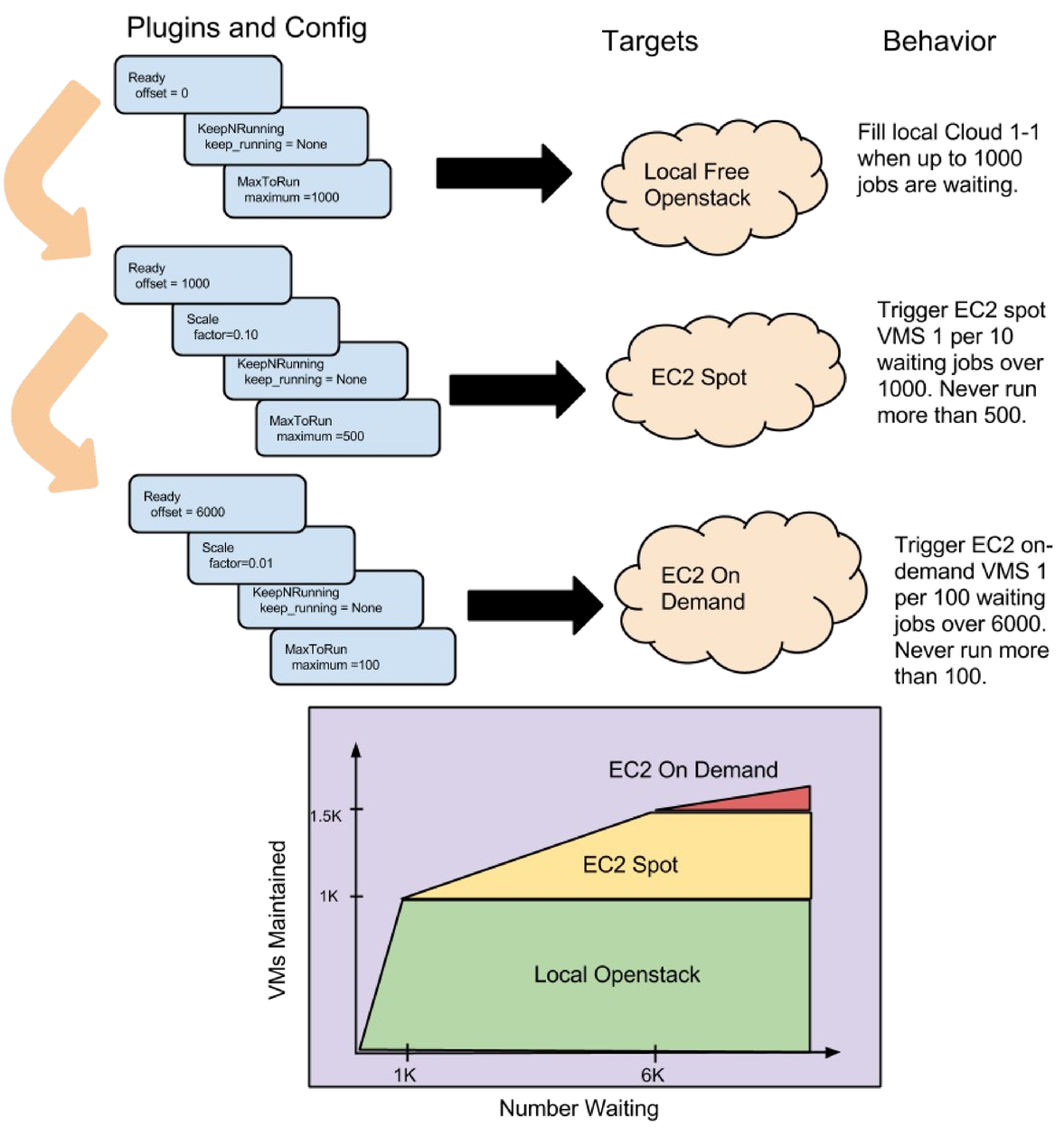
A second APF queue is configured to watch the local Condor batch queue (playing the role of WMS in this case). When Idle jobs are seen, it submits Virtual Machine invocation requests to the appropriate Cloud platform (EC2, Openstack, etc.).

These VMs are configured to connect back to the local Condor Central Manager and join the local pool of execute nodes.

The local Condor batch system runs the Idle jobs on the now-available worker nodes.

Below:

By creating multiple APF Queues targetting different cloud platforms, and configuring the scheduling plugins with appropriate offsets and scaling factors, we can preferentially fill one before “spilling over” to the other. These can be arranged in order from less efficient/least costly to more efficient/most costly, depending on circumstance.



References:
1. Maeno T., *Overview of ATLAS PanDA Workload Management*, J. Phys. Conf. Ser. 331 (2011)
2. Douglas Thain, Todd Tannenbaum, and Miron Livny, "*Distributed Computing in Practice: The Condor Experience*" Concurrency and Computation: Practice and Experience, Vol. 17, No. 2-4, pages 323-356, February-April, 2005.

For more information contact: autopyfactory-l @ lists.bnl.gov