Demand-driven provisioning of Kubernetes-like resource in OSG

Igor Sfiligoi, Frank Würthwein, Jeff Dost – UCSD
Brian Lin- University of Wisconsin Madison
HTCondor and glideins

- HTCondor is a popular science batch systems
  - Core technology in OSG
- HTCondor can be used to managed bare-metal nodes
  - But more often used in conjunction with other provisioning systems
  - i.e., The glidein model (also known as the pilot model)
No native glidein provisioning in HTCondor

- HTCondor **does not** natively provide any resource provisioning capabilities.
- Manual, e.g. backfill, deployment easy, but static
- OSG has been mostly relying on GlideinWMS for dynamic provisioning
No native k8s support in GlideinWMS

- Kubernetes is becoming a popular bare-metal resource management system
  - While batch-like, it is not a true batch system
  - GlideinWMS does not have native support for k8s
- We used backfill in the past
- But true dynamic provisioning highly desirable
Developed dedicated provisioner

- Takes inspiration from GlideinWMS
  - But drastically simplified
  - K8S easier+richer than Grid
- Based on constant pressure
  - Periodically query HTC and K8S
  - If HTC has jobs but there are no pending K8S pods, submit more
- Glideins auto-terminate in case of over-provisioning

HTCondor Scheduling

K8S Provisioner

HTCondor-managed Execution (glidein)

K8S is batch-like

https://github.com/sfiligoi/prp-htcondor-portal
Developed dedicated provisioner

• Takes inspiration from GlideinWMS
  • But drastically simplified
  • K8S easier+richer than Grid
• Based on constant pressure
  • Periodically query HTC and K8S
  • If HTC has jobs but there are no pending K8S pods, submit more
• Glideins auto-terminate in case of over-provisioning

Running the provisioner as a K8S pod makes life even easier.
(e.g. secret/token handling)

But not a hard requirement.

https://github.com/sfiligoi/prp-htcondor-portal
Only eligible HTC jobs

• Not all HTC jobs can/may run in every K8S resource
  • Policy restrictions
  • Technical limits
  • Data locality
• Provisioner fully configurable
  • Within limits of HTCondor ClassAd expressiveness

Simplified example

[HTCondor]
additional_requirements= \n  ((DESIRED_Sites is undefined)|| \n    stringListMember("SDSC-PRP",DESIRED_Sites,"",")&& \n    ((UNDESIRED_Sites is undefined)|| \n    !stringListMember("SDSC-PRP",UNDESIRED_Sites,"",")&& \n    (!isUndefined(ProjectName))&& \n    (!isUndefined(SingularityImage))

[k8s]
node_affinity_dict=\nautilus.io/low-power:true

https://github.com/sfiligoi/prp-htcondor-portal
Pushing scheduling to K8S

- Provisioner propagates pending HTC job requirements into glidein pods
  - Currently a tuple of (#CPU, memory, disk, #GPUs)
- Thus, many types of pods
  - K8S fully in control of scheduling
- Each glidein will only serve jobs that match its requirements

https://github.com/sfiligoi/prp-htcondor-portal
Preemption often used in K8S as backfill
• Great way to maximize resource utilization and lower cost
• HTCondor and K8S-based provisioner transparently handle it

Opportunistic usage of PRP/NRP CPU resources in 2nd half of 2022
Extending to Lancium

- Lancium is a green computing provider
  - Has a IaaS offering
  - Using a proprietary Cloud interface
- Was a major contributor of OSG resources in 2022
Kubernetes-like interface

• Lancium Cloud API proprietary
  • But Kubernetes-like in spirit
  • Based on containers, no-client-side-state

• Provisioner uses only Kubernetes basics
  • We like the KISS principle!
  • Lancium API rich enough to express all concepts we were relying on

• Integration required only a little glue
  • Replacing K8S invocations with Lancium invocations
Lancium provisioner

- Most code common with K8S provisioner
  - Mostly an interface change
- Still based on constant pressure
  - Periodically query HTC and Lancium
  - If HTC has jobs but there are no pending Lancium jobs, submit more
- Glideins auto-terminate in case of over-provisioning

HTCondor-managed Execution (glidein)

Lancium is K8S-like (and batch-like)
Lancium provisioner

• Most code common with K8S provisioner
  • Mostly an interface change
  • Still based on constant pressure
  • Periodically query HTC and Lancium
  • If HTC has pending jobs but there are no pending Lancium jobs, submit more
• Glideins auto-terminate in case of over-provisioning

OSG switch to Lancium provisioner in June 2022

Usage of Lancium CPU resources in 2nd half of 2022
Lancium provisioner

- Most code compatible with K8S provisioner
- Mostly an interface change
- Still based on constant pressure
- Periodically query HTC and Lancium
- If HTC has jobs but there are no pending Lancium jobs, submit more
- Glideins auto-terminate in case of over-provisioning

OSG switch to Lancium provisioner

In Spring 2023 Lancium decided to abandon its IaaS offering.

So, the Lancium provisioner will be retired soon.
Nevertheless, it proved the feasibility of extending the K8S provisioner to other platforms.
Summary and conclusions

• Kubernetes is becoming increasingly popular and we needed a way to integrate it with HTCondor glidein infrastructure

• We opted for a dedicated, but GlideinWMS-inspired solution
  • Keeping it as simple as possible
  • Available in github, under a BSD license

• Currently in use to make good use of opportunistic K8S resources on the PNRP on behalf of OSG, but also IceCube
  • We extended it to support Lancium, too

https://github.com/sfiligoi/prp-htcondor-portal
• This work has been partially funded by the US National Science Foundation (NSF) Grants OAC-2030508, OAC-2112167, OAC-1826967, CNS-1925001, OAC-1841530, CNS-1730158, CNS-2100237 and CNS-2120019.