Using Kubernetes as an ATLAS computing site

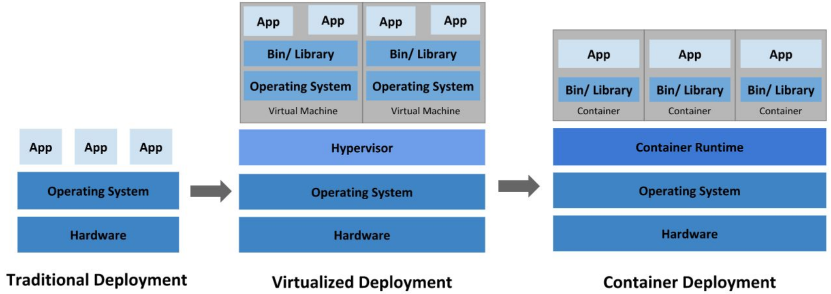
Fernando Barreiro Megino, Jeffrey Ryan Albert, Danika MacDonell, Tadashi Maeno, Ricardo Brito Da Rocha, Rolf Seuster, Ryan P. Taylor, Ming-Jyuan Yang
on behalf of the ATLAS experiment

CHEP 2019, Adelaide, Australia

Introduction. Rationale

- Idea born during ATLAS-Google Data Ocean project
  - Containers as lightweight alternative to VMs in cloud environments or on bare metal
  - Kubernetes as an open-source replacement for traditional WLCG CE+batch infrastructure
    - No HEP-specific middleware
    - Container-native environment, no additional container setups (e.g. singularity at sites)
    - Potential expansion/federation with commercial cloud providers
  - Integration in new ATLAS pilot submitter: Harvester
- ATLAS investigating in exploratory capacity with help from students and qualification tasks
  - Harvester linking experiment WMS and Kubernetes
  - Kubernetes testgrounds available at different sites
    - CERN: 2k core cluster available in Jan-Feb 2019
    - University of Victoria: 136 cores for initial testing, production cluster ramping up
- Initial working model available and ideas to improve the system

Kubernetes basics



- Kubernetes (K8s): container orchestration for automating application deployment, scaling and management
- Nomenclature
  - “Nodes”: the physical or virtual machines participating in the K8s cluster
  - “Pods”: basic scheduling unit, can contain one or multiple containers.
  - The “Job” controller can be used to run containers as batch workloads
    - Jobs create pods until a specified number of them successfully terminate
- Advantages: extensibility, flexibility, declarative API
- Available on all major cloud providers
- Various installation options, depending on available infrastructure
  - CERN: [on demand clusters through Openstack Magnum]
  - University of Victoria: Terraform, Kubespray+Ansible

Experience at UVic

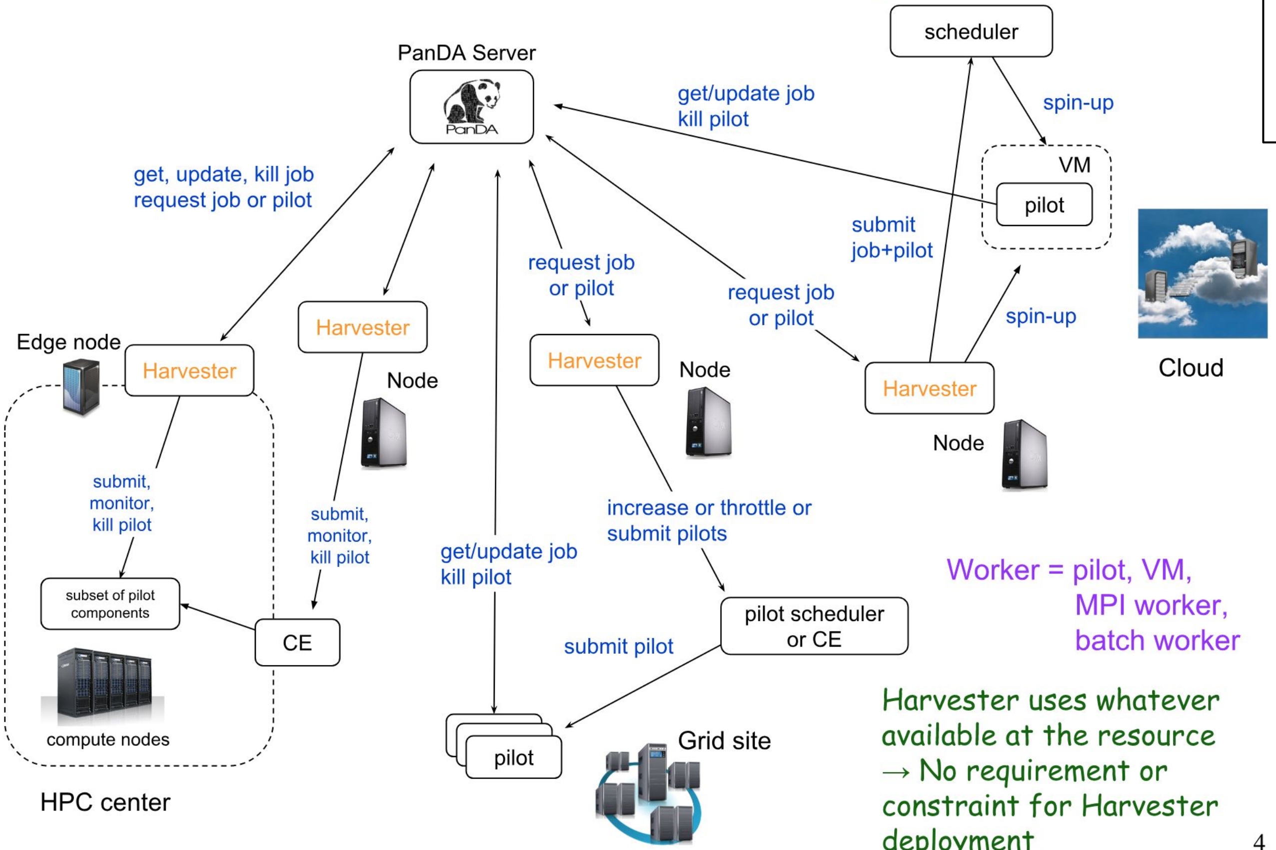
- Kubespray for Infrastructure as code (IaC) deployment on UVic Arbutus cloud
  - Extension of “cattle not pets”
- Terraform and Ansible, with integration of additional roles/inventory
  - Fixes/improvements for our use-case contributed upstream: 4934, 4912, 4435, 4022
- Cluster can be deployed, or rebuilt from scratch, in ~ 5-10 minutes
- CVMFS installed on hosts, mounted to pods via hostPath volume
- atlas-grid-centos7 image published to `/cvmfs/unpacked.cern.ch` with DUCC
- Containers launched with CVMFS Docker Graph Driver plugin for scalability
  - Significant startup acceleration and bandwidth savings

Kubernetes Security & Monitoring

- Cluster API is exposed over the internet to Harvester @ CERN
- Role Based Access Control restricts which API actions are allowed
- PSP configuration allows secure hostPath volume mount for CVMFS
  - Privileged pods forbidden, unprivileged pods have read-only access to /cvmfs
- IaC facilitates frequent updates for security patches
- Prometheus Helm operator provides real-time, self-configuring observability of containers and infrastructure

Harvester: universal resource interface

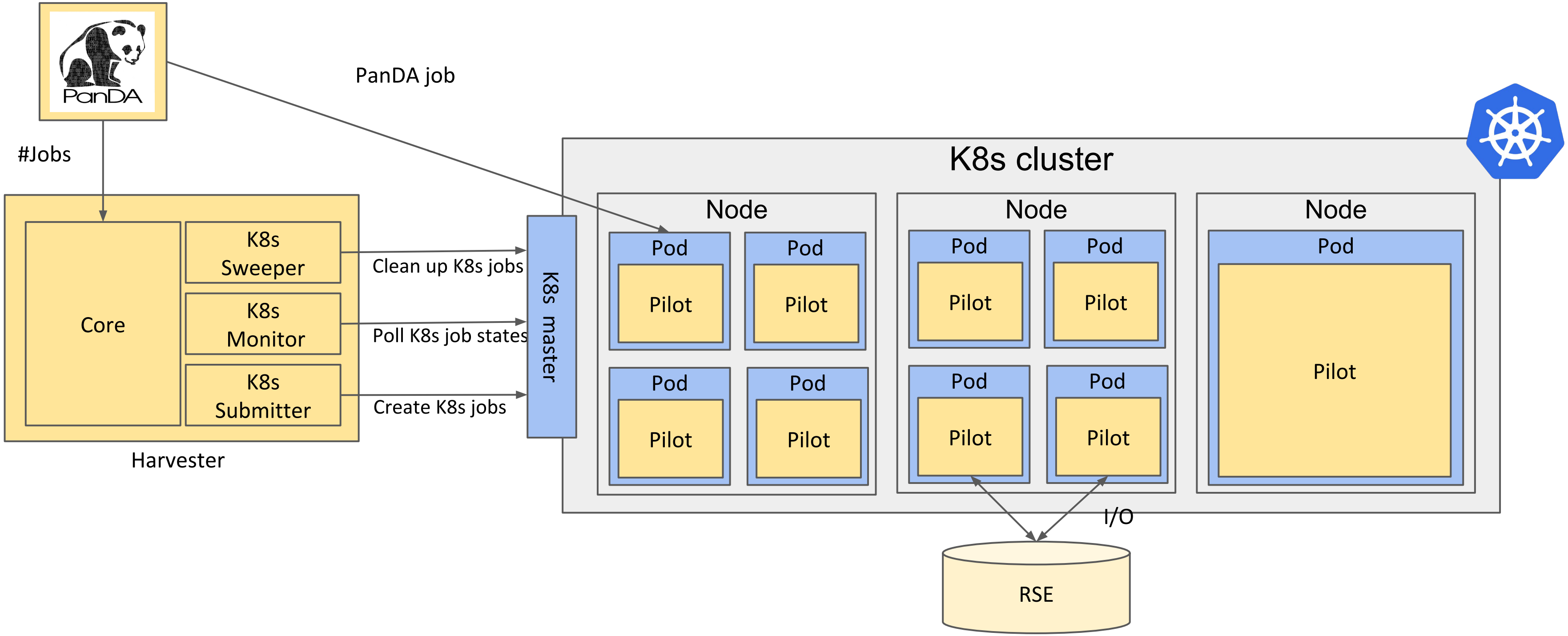
Other Harvester presentations in this conference:
[Harvester and the Grid]
[Harvester and HPC]



Worker = pilot, VM,
MPI worker,
batch worker

Harvester uses whatever available at the resource
→ No requirement or constraint for Harvester deployment

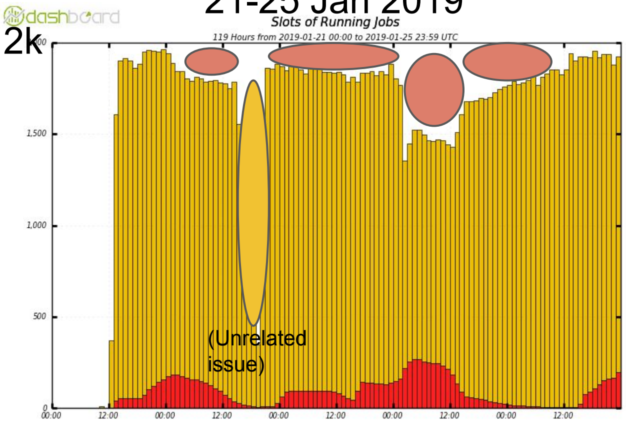
Harvester - Kubernetes integration



Container image configured per queue (for ATLAS: CentOS7 image)

Experience at CERN: Jan-Feb 2019

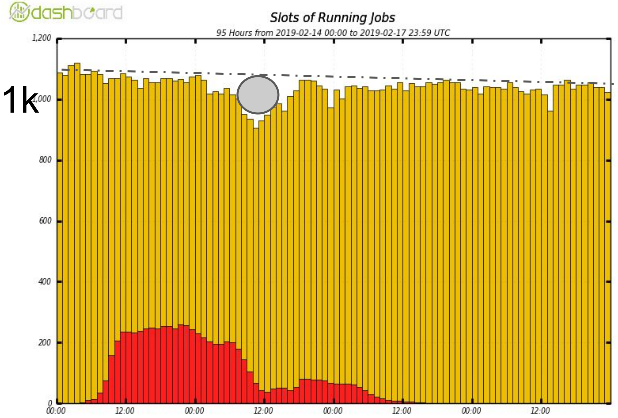
Beginning of the exercise: default K8s scheduling
21-25 Jan 2019



8 core 1 core

Kubernetes default scheduling spreads the 1 core pods across nodes (8 cores), blocking out 8 core jobs on those nodes

Later in the exercise: policy tuning to pack nodes
14-17 Feb 2019



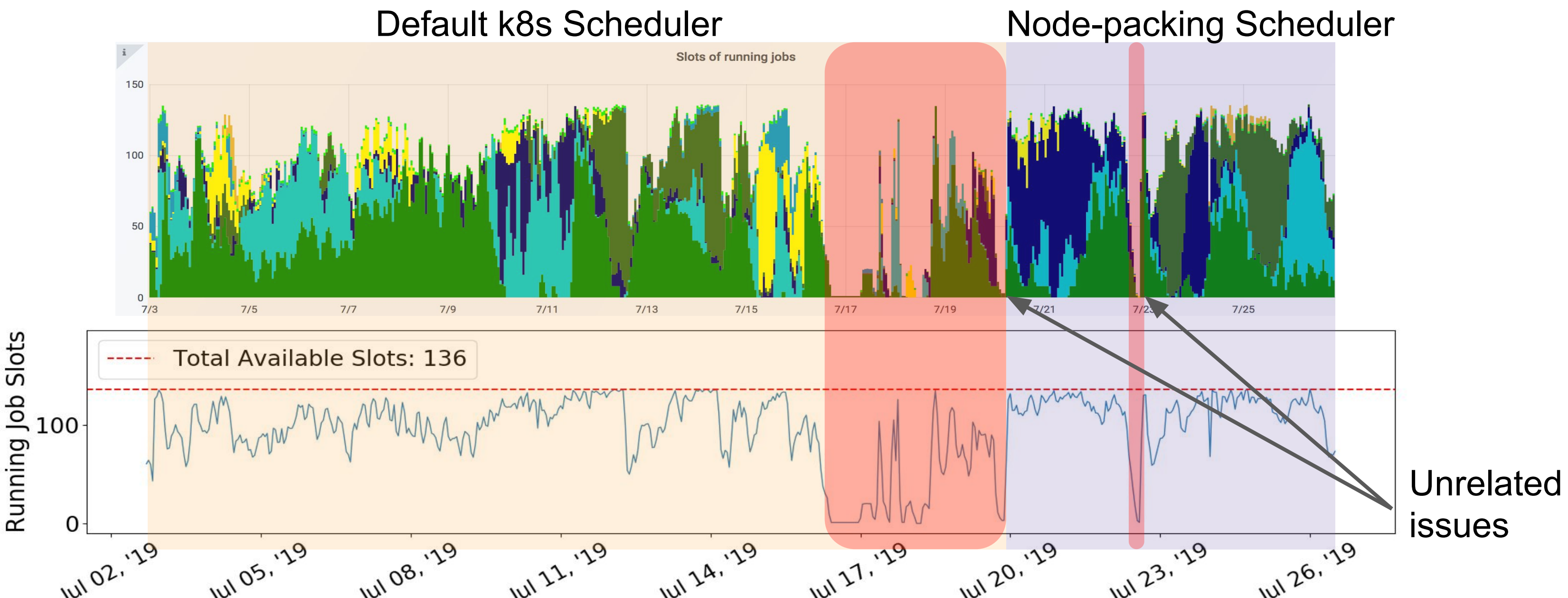
8 core 1 core

Policy scheduling packs multiple 1 core pods in the nodes(8 cores). Utilization is much more efficient, except for node draining when 1 core jobs vacate (same as in batch systems)

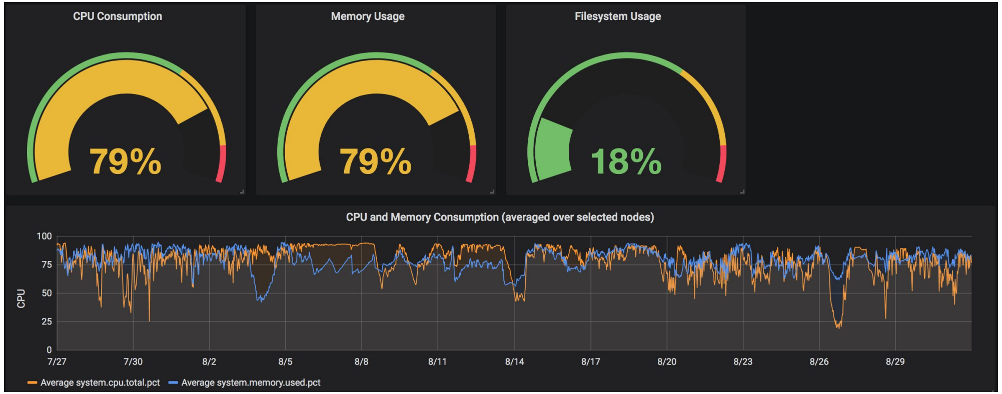
Throughout the exercise we kept losing nodes for multiple reasons (issues with the CVMFS driver, Kubernetes leftovers, etc.). Need detection and automated replacement of faulty nodes.

Experience at Uvic

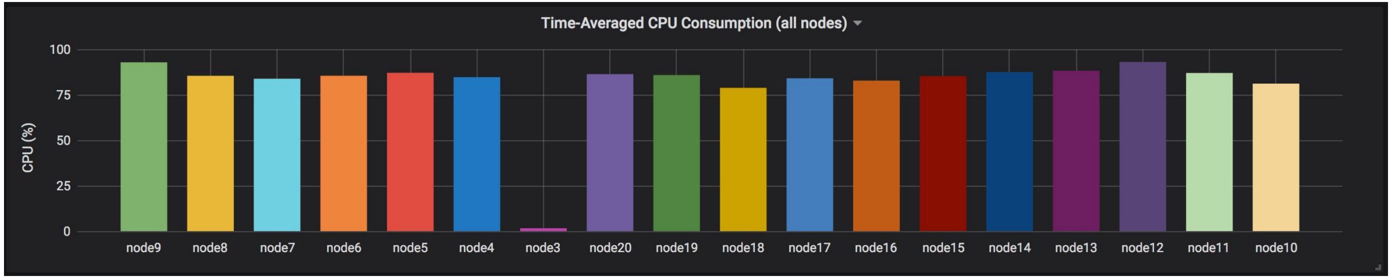
- Node-packing pod scheduler improves job occupancy of available cores compared with default round-robin scheduler
- Implemented as a custom kube-scheduler pod



UVic Kubernetes Cluster Results



- Kubespray cluster running jobs stably for several months
- Internal cluster resource metrics monitored and stored in Elasticsearch



Conclusions and future work

- We have integrated PanDA and Kubernetes through Harvester
- We have executed all types of ATLAS production workloads at various sites at T2 equivalent scale
- Initial model does not fully exploit Kubernetes philosophy
    - Ideally users should be able to specify arbitrary containers and executables
    - Pilot should not run in the container; instead Harvester should handle some pilot functionality (e.g. data stage-in/out, job monitoring, resource consumption)
        - Concept of Pilot 2 APIs
- T2 services: deploy Squids in Kubernetes for CVMFS and Frontier
- Kubernetes federations - expand into the cloud?
- Quotas: sites shared by multiple experiments need to be able to define a flexible quota/share system
- Infrastructure stability: automatic replacement of faulty nodes