UNIVERSITY OF TEXAS ARLINGTON

Kubernetes Activities at UTA

Armen Vartapetian
University of Texas, Arlington

with thanks to Patrick McGuigan for providing and help with the compute infrastructure

US ATLAS Computing Facilities Face-to-Face at SLAC
December 1, 2022

Introduction

- The main goal of this exercise is to study ATLAS production running with Kubernetes
- Built a Kubernetes cluster at UTA, and setup to run ATLAS jobs on it
- Use the special instance of Harvester (CERN_central_k8s), interfacing ATLAS workload submission engine PanDA with Kubernetes to run containerized jobs as pods
- Learn from running a variety of ATLAS production workloads and workflows with Kubernetes - understand possible inefficiencies, bottlenecks, and work on optimization
- Carry out maximum possible tuning to optimize the cluster performance, and see whether the results indicate that we can benefit from the Kubernetes (K8S) model

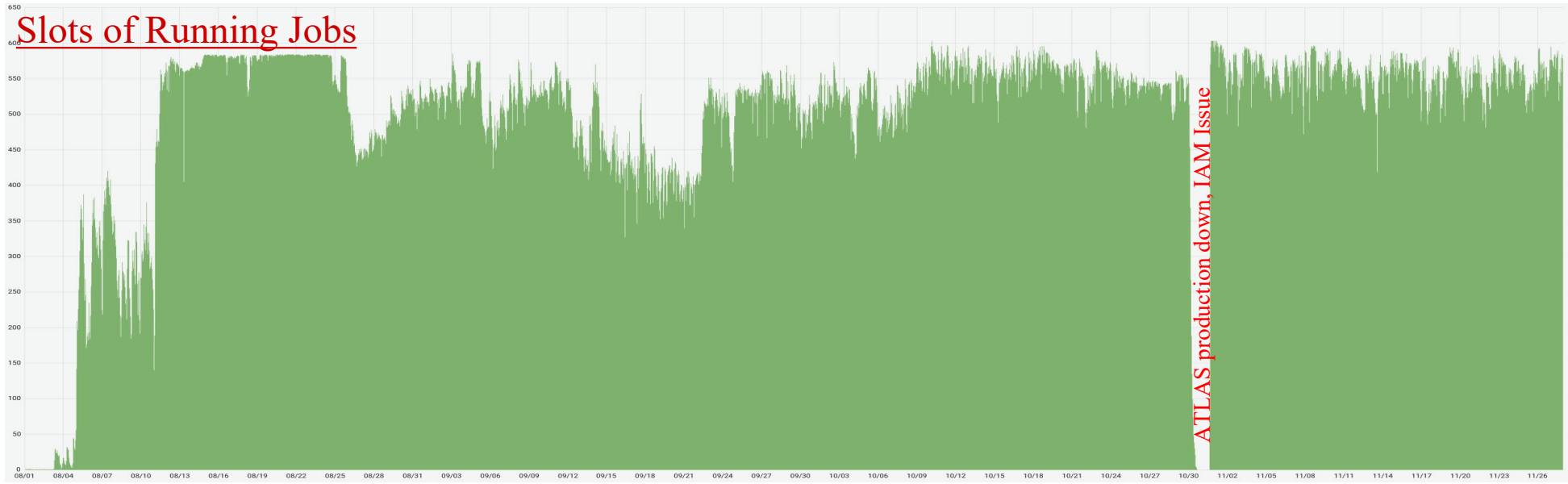
Kubernetes Cluster Hardware

- Started the UTA K8S cluster early this year with several nodes
- The size of the cluster was increased after the UTA_SWT2 site was moved to UTA campus, and some fraction of that hardware was utilized for K8S
- Racked in the same machine room with the SWT2_CPB site hardware, but separate from the main cluster, separate network
- The current overall size of the cluster 576 cores in worker nodes, with DELL PowerEdge server models R410, R620, R630
  - Respectively we have mixture of nodes with core count and memory size: 40core 3.2GB/core, 16core 2GB/core

Kubernetes Installation, Issues

- The UTA K8S cluster was deployed on bare metal using the kubeadm method
- The cluster is running the K8S version 1.24.3
- For container runtime docker was used in early test setup, and we moved to containerd for the current cluster setup
- Concerning the installations, with the gained experience, extending the cluster was easy and fast. And usually adding a new worker node takes just minutes.
- We had an event when we did a full reinstall of all the worker nodes (OS & K8S), and it took just a couple of hours.
- During debugging of the system, some issues with network configuration and hardware were found and resolved
- Ephemeral storage default location was moved to different partition with more available space
- CVMFS autofs mount was not working correctly, not allowing re-mount for a fraction of pods, with temporary stuck state in ContainerCreating. Moving to hard mount fixed the issue.

Going Online, Current Status (I)



- The UTA K8S cluster queue is SWT2_CPB_K8S
- The queue was set online at the beginning of August this year
- Regular ATLAS production - no limitation on type or resource request (SCORE, MCORE, *_HIMEM) of jobs submitted/run in the cluster
- After some optimization (see the next slide), the production during the past month was overall stable, at the expected level of running job slots

Going Online, Current Status (II)

- We had to limit the number of running SCORE jobs (set in CRIC), as at some point SCORE jobs gradually occupied most of the cores. After the limit was in place we were back to reasonable mixture of SCORE/MCORE jobs.
- Noticed some inefficiency in completely filling the cores in older machines (worker nodes with 32GB memory). Appeared to be an issue with pod scheduling, when overall available node memory was slightly lower (~0.3%) than the requested memory (2000MiB/16000MiB for SCORE/MCORE jobs). After discussion with Fernando, he introduced a new parameter in the Harvester: k8s.resources.requests.memory_scheduling_ratio, and it was set to 98% for SWT2_CPB_K8S site in CRIC. That resolved the occupancy issue for those worker nodes.
- As the older nodes are operating close to memory limit, they are more susceptible to memory pressure. I noticed some cases, when out of memory event makes the node unresponsive, and for kubelet reclaiming the node back may take time.
- Following the scheduling at the node level, trying to understand dynamics, and various scenarios. Though overall it look fine and reasonable.
- Trying to understand if we may benefit from any possible configuration tuning of the kube-scheduler. See if in some scenarios, the inter-pod affinity or anti-affinity, tainting, etc, may help with better packing, and more efficient resource utilization.

Next Steps

- During the first quarter of next year increase the cluster size to at least 1000 cores
  - Include also newer generation of hardware
  - May need an upgrade of the setup in the machine room
- Improve the monitoring of the cluster, collecting the metrics, adding a graphical interface. Analyze and understand the collected data.
- Continue to gain experience analyzing various scenarios of production jobs running in the cluster, the details of pod scheduling and running, possible ways of optimization, and mitigation of factors which may create inefficiencies
- Evaluate and compare the performance of the optimized K8S cluster.