Status Report: NUMA-aware scheduling in kubernetes

Francesco Romani • 2023-09-14
github.com: ffromani | mail: fromani@redhat.com

Overview

Achievements

Highlights

- Novelty factors
- Challenges - areas for improvement

Future

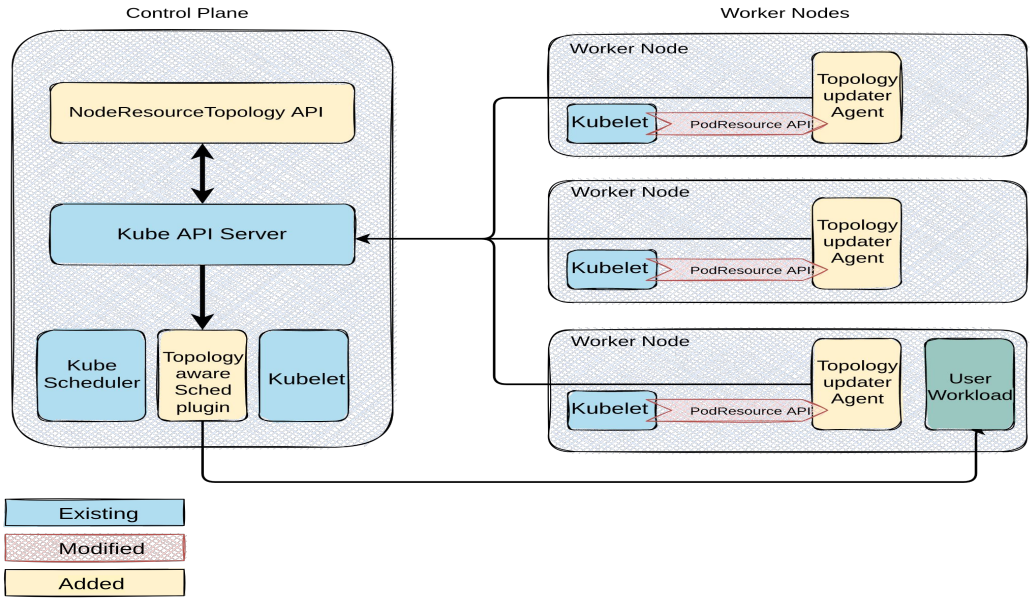
Recap

Recap

An out-of-tree solution.

Development started in 2020 (alongside Kubernetes v1.22)

1. Topology-aware Scheduler Plugin: (filter + scoring [+ reserve])
2. NodeResourceTopology API
3. Topology Updater Agent
   a. Node Feature Discovery
   b. Resource Topology Exporter



Details: Topology-aware Scheduling Kubecon EU 2022 Presentation

Achievements

Achievements

Feature completeness

- Caching support (overreserve, discardreserved)
    - Please check here to learn why do we need a cache at all
- Testsuite (integration tests + external suites)
- Red Hat active involvement and support

Achievements

Lower the entry barrier

- Continued effort to make easy to try and consume
    - NUMA-aware scheduling enablement is a stack of components, setup is nontrivial
- Deployer project: a set of go packages and a command line tool to setup all the components and settings needed to enable the topology-aware-scheduling on a kubernetes cluster.
- Minikube quickstart

Achievements

Demos

- https://asciinema.org/a/kiaRBqGVkHYwFTyp9WmkbjpaU
- https://asciinema.org/a/587245

Highlights

Novelty/Challenge

Split allocation logic

- Allocating resources is a node-level responsibility: the Topology Manager is the final arbiter
- The scheduler cannot drive the topology manager behavior
- We cannot even disable the Topology Manager
- The scheduler can only anticipate/second guess the Topology Manager

Novelty/Challenge

Canonical data representation

- The NodeResourceTopology API is flexible by design
- Multiple different topology representations are legal and valid
  - They merely reflect the growing HW complexity
- How to normalize them effectively?
- How to ensure compatibility producer/consumer?
  - WASM to the rescue?

Novelty/Challenge

Pod in terminal phase

- Root cause: the scheduler has to track state and reconcile frequently with the node state
- Need a concise node state representation
- Kubelet reports pod in terminal phase. The scheduler filters them out
- Just a kubelet bug?

Challenge

Observability

- Understanding scheduling decisions is harder in this case;users are keen to understand fine details of NUMA-aware scheduling
- Wishlist: more for sig-instrumentation probably :)
    - Tuning of logging verbosiness at runtime
    - Per-pod/per flow logging and verbosiness
    - Inspect internal state (without debugger!)

Challenge

Code sync/reuse

- Another consequence of the split allocation logic
- Less maintenance cost and better scheduling decisions if we can abstract and reuse Topology Manager logic
    - Massive refactoring required
    - Perhaps move logic proper into plugins, reuse them?
    - Do we have resources as community?

Future

Next steps

NodeResourceTopology API beta1

Possible compatibility break point (spec/status?)

Cleanups and polishing

Stabilization, bug fixes, performance enhancements

IOW the usual maintenance work

Merge into kubernetes core

Together with the NodeResourceTopology API

Thanks for listening!

1. #wg-batch on k8s slack
2. #topology-aware-scheduling on k8s slack
3. https://github.com/k8stopologyawareschedwg (code)