KubeCon | CloudNativeCon

North America 2020

Virtual

Tune it up!
Enabling low- latency workloads in Kubernetes clusters

Yanir Quinn - Senior software engineer, Red Hat
Francesco Romani - Principal software engineer, Red Hat

kubernetes

Telco/CNF

Workload Challenges

Real Time

Guaranteed response within specified time constraints

Low Latency

Process work as soon as possible

Modeling

CPU

Special devices

Feature use case

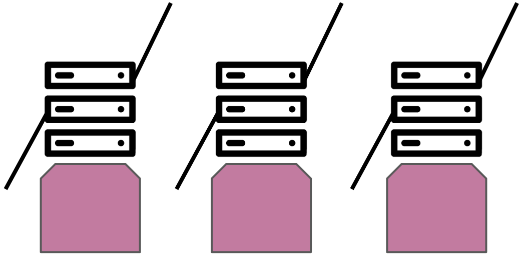
As a cluster administrator, I want to configure certain nodes for low latency, real-time or DPDK purposes without having to understand all the interactions of kernel, OS and Kubernetes components

- Certain workloads need to be isolated from all interruptions as the latency of their responses matter (packet latency <20us)
- This is specifically needed for Telco customers both for their Web-scale and Edge deployments
- Doing this manually possible but tedious

Source:
https://github.com/openshift-kni/performance-addon-operators

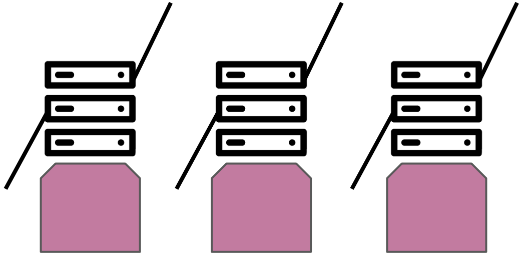
WHAT DO WE WANT?

LOW LATENCY!

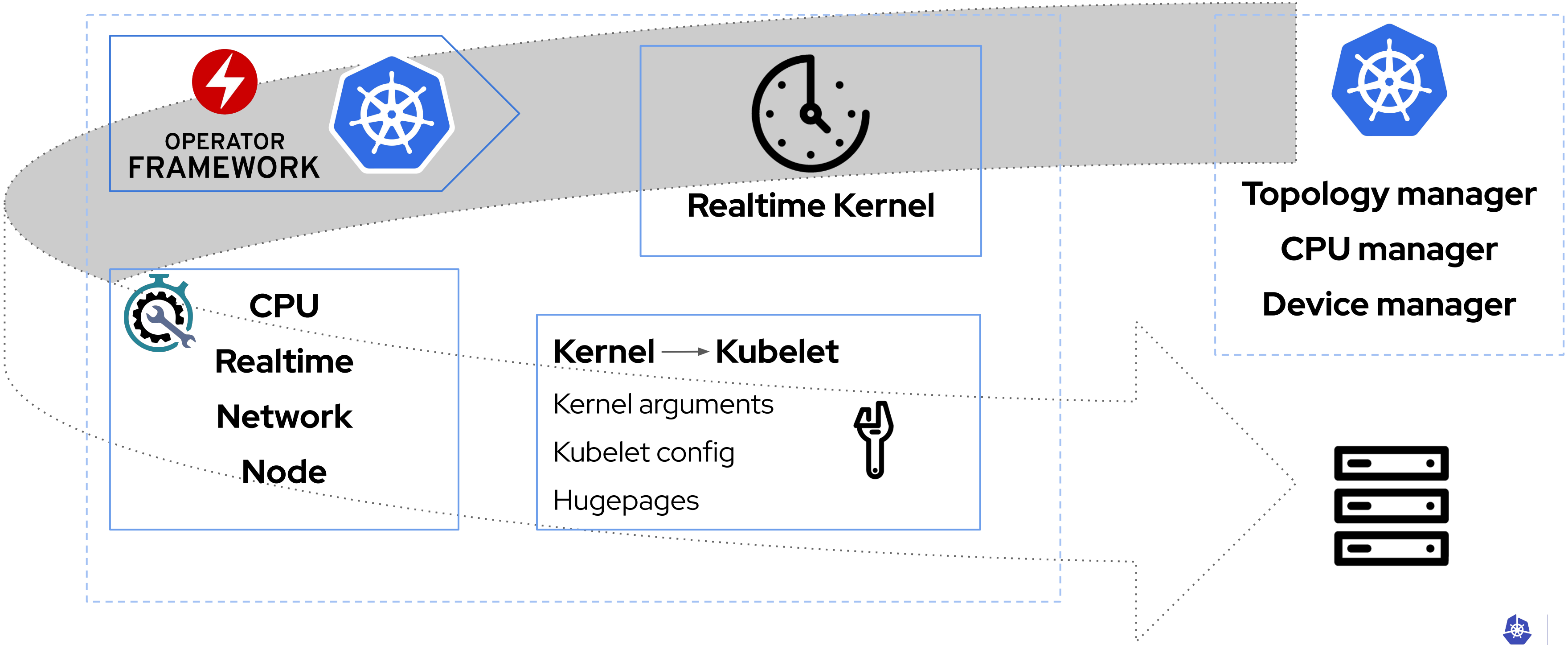


HOW DO WE DO IT?

?????????????



The ecosystem



OPERATOR FRAMEWORK

Realtime Kernel

Topology manager

CPU manager

Device manager

CPU

Realtime

Network

Node

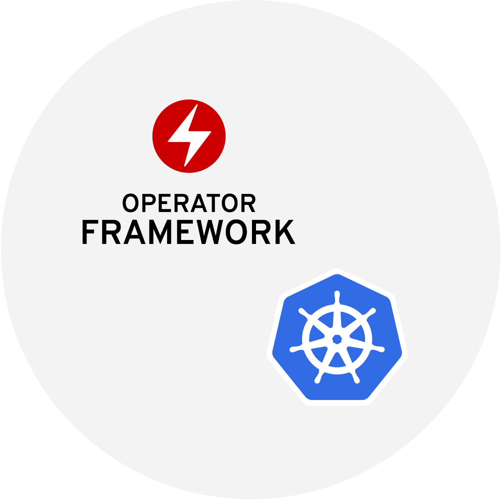
Kernel → Kubelet

Kernel arguments

Kubelet config

Hugepages

Operators



What is it?

Method of packaging, deploying and managing a Kubernetes-native application

A Kubernetes-native application is an application that is both deployed on Kubernetes and managed using the Kubernetes APIs and kubectl tooling

Kubernetes pattern

Extending K8s control plane with a custom Controller and Custom Resource Definitions that add additional operational knowledge of an application

Source:
https://www.openshift.com/learn/topics/operators

Real Time Kernel

Optimized kernel

Designed to maintain low latency, consistent response time and determinism

Not for everyone

A well tuned system fitted to requirements

CPU time is the principal resource

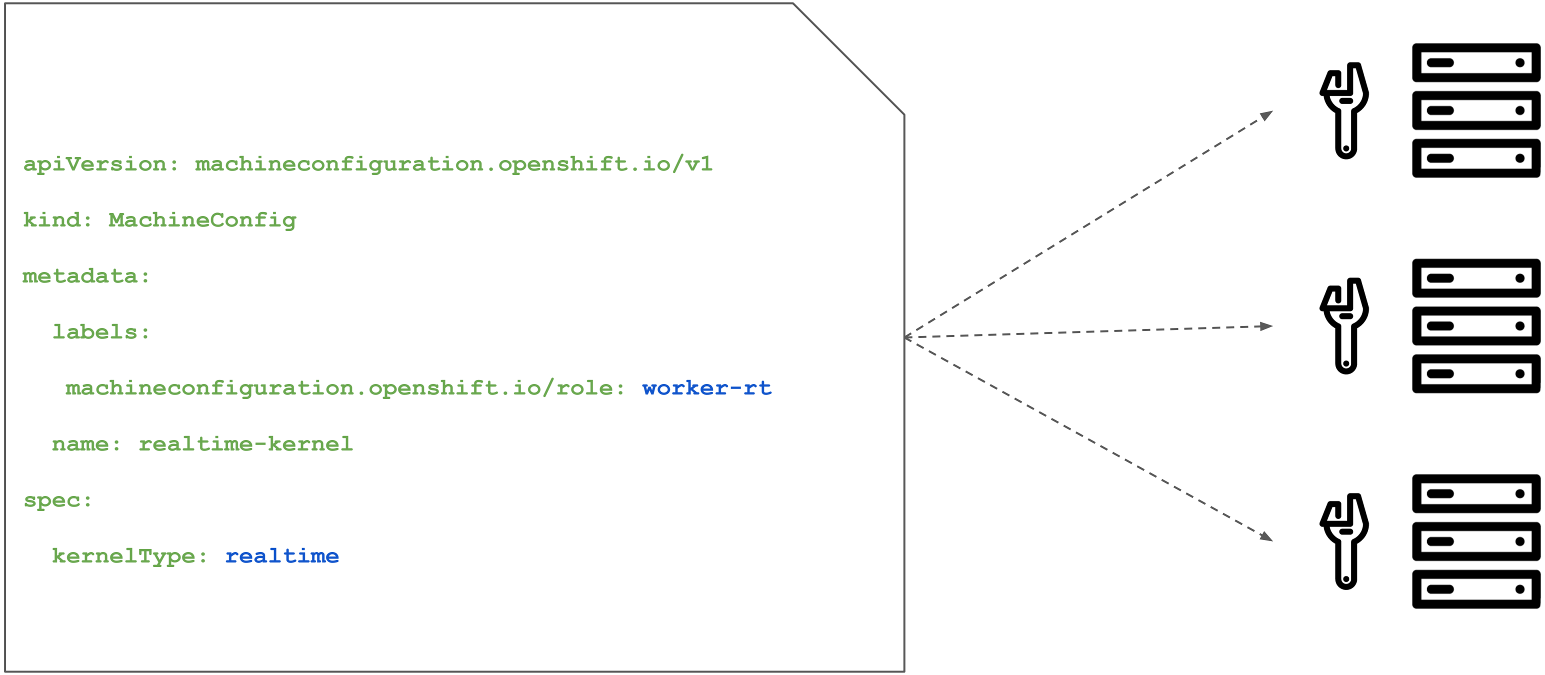
Kernel RT mechanism to support:

- preference to higher supported tasks
- Non CFS Scheduling policies
- Maintain low latency execution time

Tunings: at the cluster level

By NASA, ESA, AURA/Caltech, Palomar ObservatoryThe science team consists of: D. Soderblom and E. Nelan (STScI), F. Benedict and B. Arthur (U. Texas), and B. Jones (Lick Obs.)

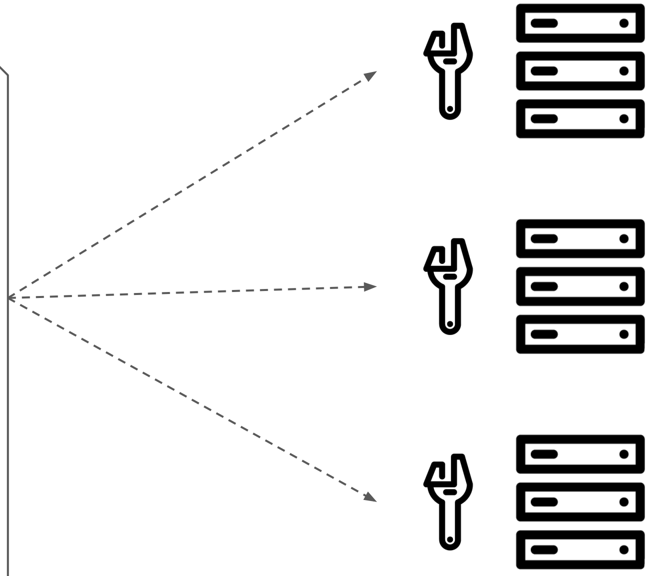
Machine configuration



```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
   machineconfiguration.openshift.io/role: worker-rt
  name: realtime-kernel
spec:
  kernelType: realtime
```

Machine configuration

```
apiVersion: machineconfiguration.openshift.io/v1
kind: KubeletConfig
spec:
  machineConfigPoolSelector:
    matchLabels:
     machineconfiguration.openshift.io/role: worker-rt
  kubeletConfig:
       reservedSystemCPUs: 1-3
       systemReserved:
       cpu: 1000m
       memory: 500Mi
       topologyManagerPolicy: best-effort
```



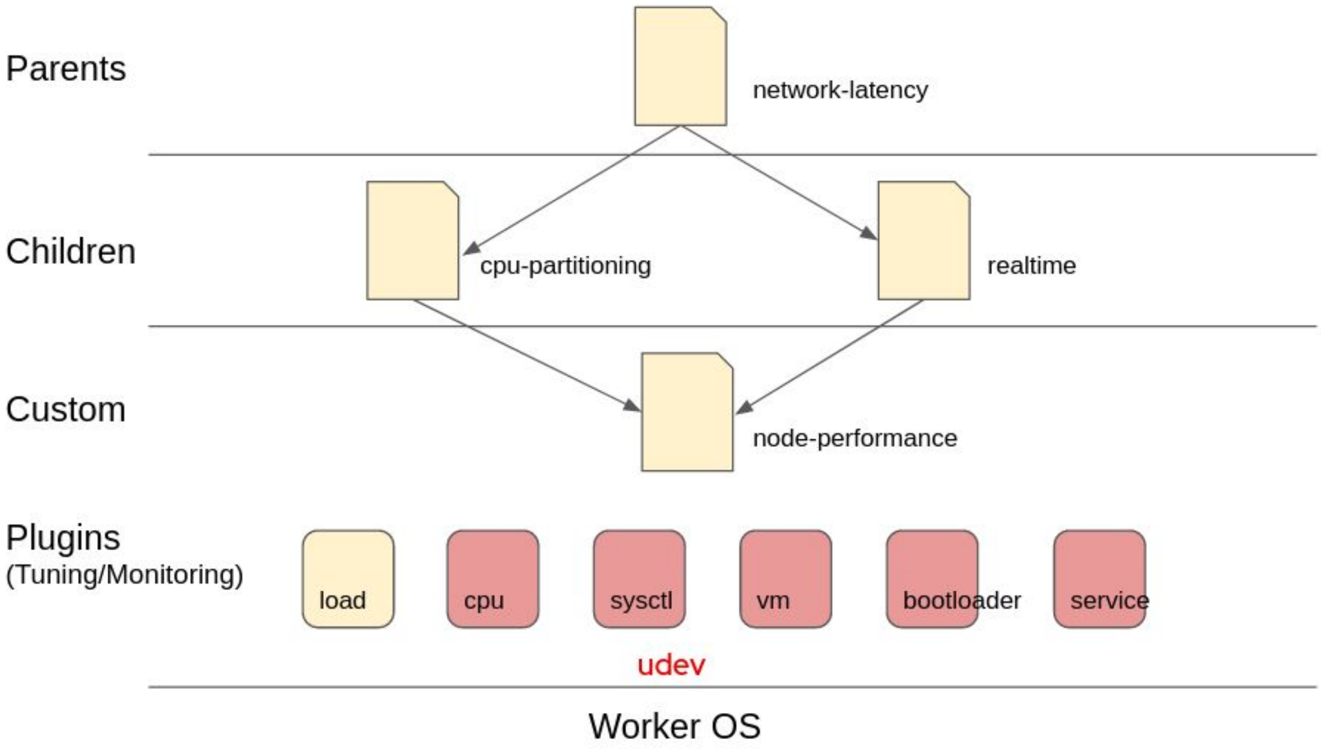
Micro Demo

Machine configuration setting:

- Realtime kernel
- Kubelet arguments

Play Demo

Tuned daemon



Source:
https://tuned-project.org/

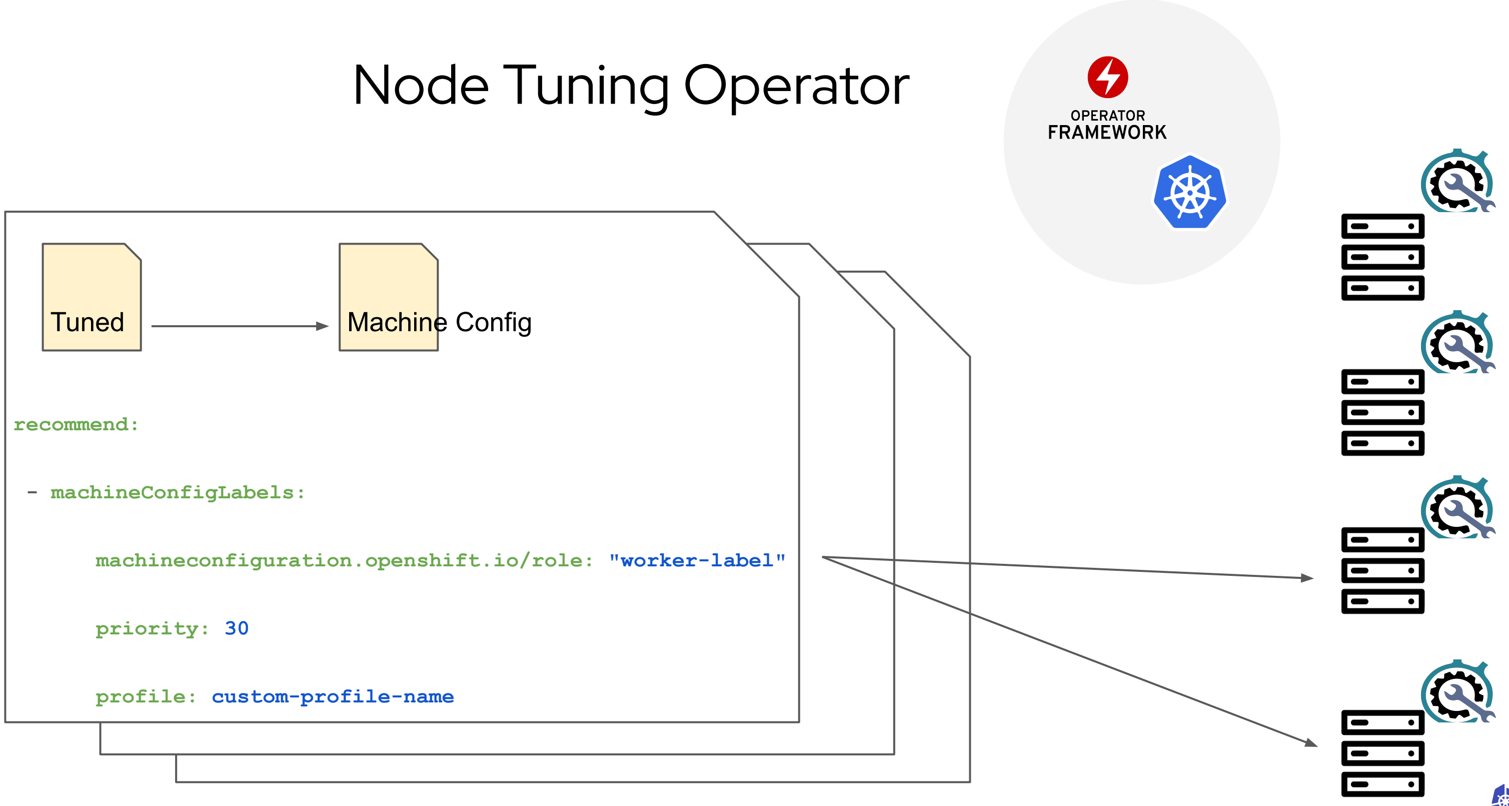
Node Tuning Operator

The majority of high-performance applications require some level of kernel tuning. The operator provides a unified management interface to users of node-level sysctls and more flexibility to add custom tuning specified by user needs

- Centralized Customization
- Modular & Dynamic
- OS level defaults for the control plane and worker nodes

Source:
https://github.com/openshift/cluster-node-tuning-operator

Node Tuning Operator



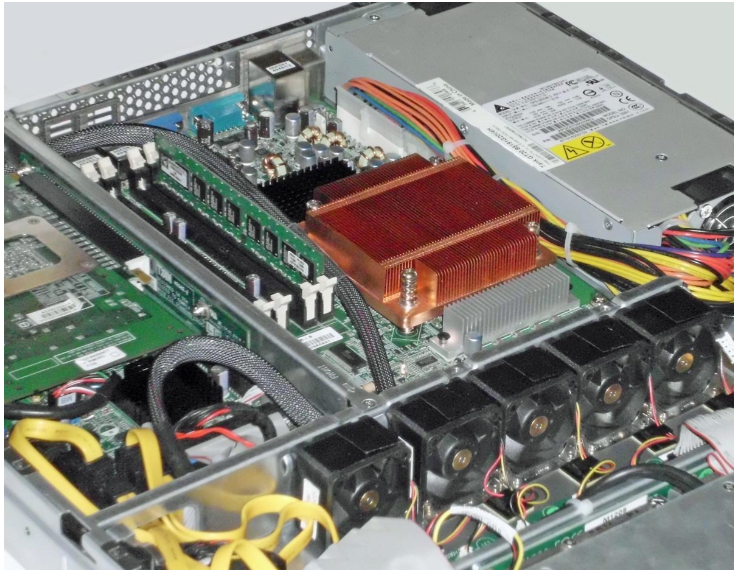
```
recommend:
 - machineConfigLabels:
     machineconfiguration.openshift.io/role: "worker-label"
     priority: 30
     profile: custom-profile-name
```

Micro Demo

Deploying a Tuned profile

Play Demo

Tunings: at the node level

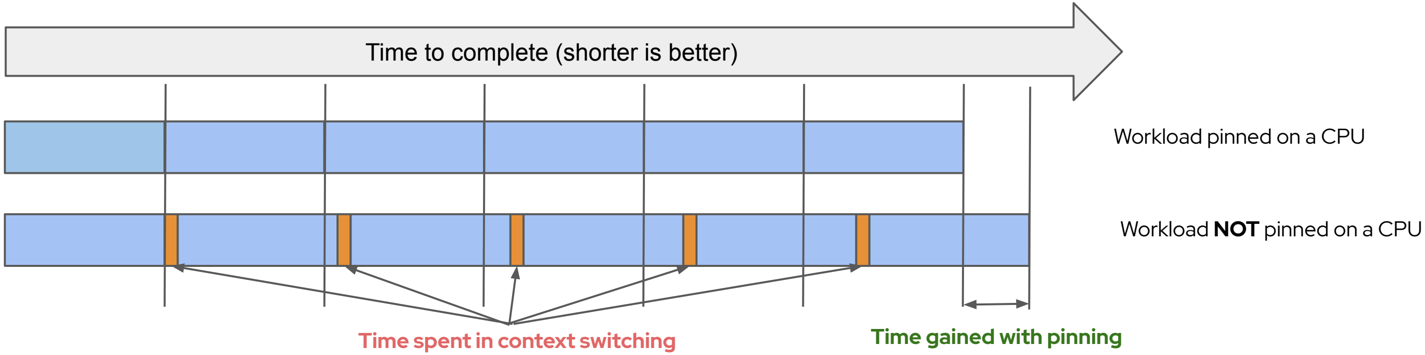


Tobias Maier, CC0, via Wikimedia Commons

CPU manager

CPU manager is a built-in kubernetes component which allows workload to require exclusive assignment of cpu cores

- Minimizes workload context switches
- Minimizes workload cpu-resource contention


Time to complete (shorter is better)
Workload pinned on a CPU
Workload **NOT** pinned on a CPU
Time spent in context switching
Time gained with pinning

https://kubernetes.io/docs/tasks/administer-cluster/cpu-management-policies/
https://docs.openshift.com/container-platform/4.5/scalability_and_performance/using-cpu-manager.html

Device manager

Device Manager is a built-in kubernetes component which enables node hardware resources and peripheral devices using third party plugins

Enables the workload to consume to specialized hardware resources:

- SR-IOV VFs
- Hardware Accelerators
- GPUs

https://kubernetes.io/docs/concepts/extend-kubernetes/compute-storage-net/device-plugins/
https://docs.openshift.com/container-platform/4.5/nodes/pods/nodes-pods-plugins.html

Topology manager

Topology Manager is a built-in kubernetes component which enables NUMA alignment of CPUs and peripheral devices

- Flexible policies for different resource alignment requirements
- Orchestrates CPU and Device Manager
- Allows workloads to run in an environment optimized for low-latency

https://kubernetes.io/docs/tasks/administer-cluster/topology-manager/
https://docs.openshift.com/container-platform/4.5/scalability_and_performance/using-topology-manager.html

Topology-aware resource allocation

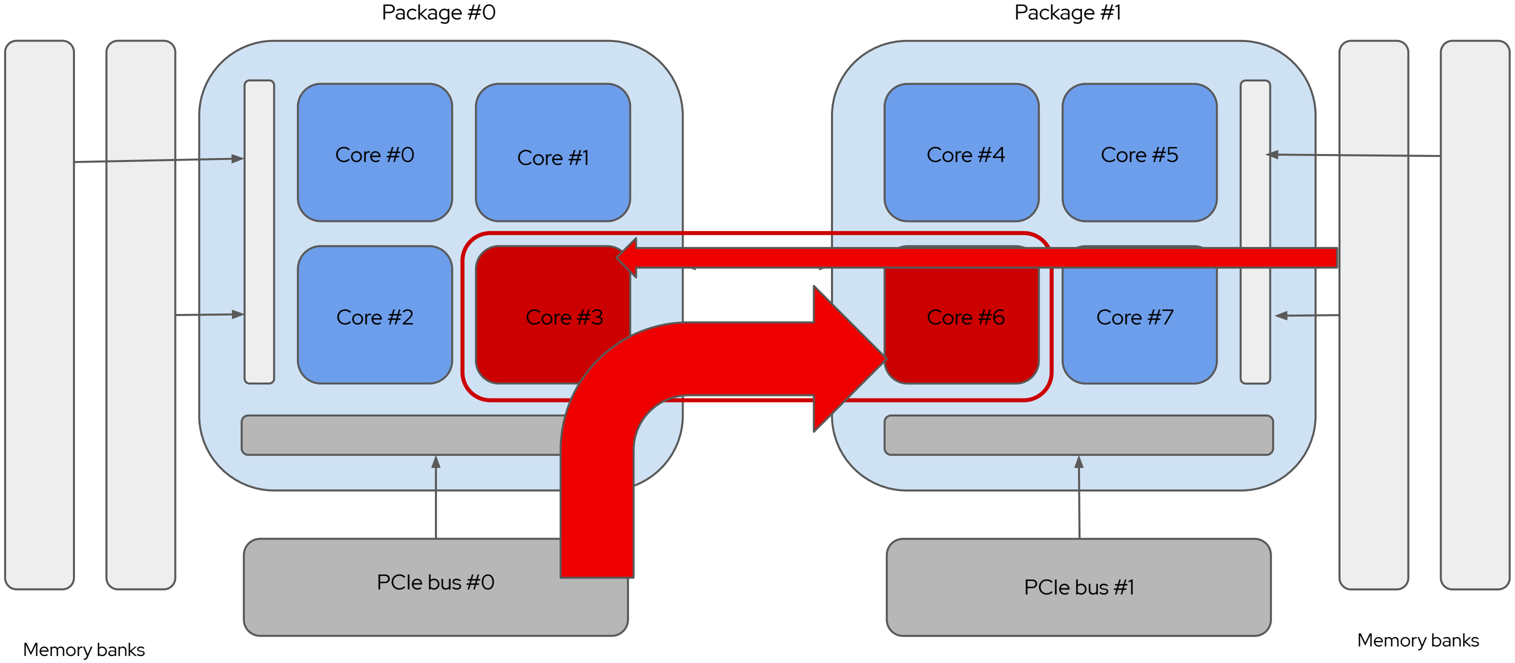
In order to illustrate aligned and not-aligned resources allocation, let's consider a very simple scenario:

A system with two numa cells

Let's consider a workload requesting

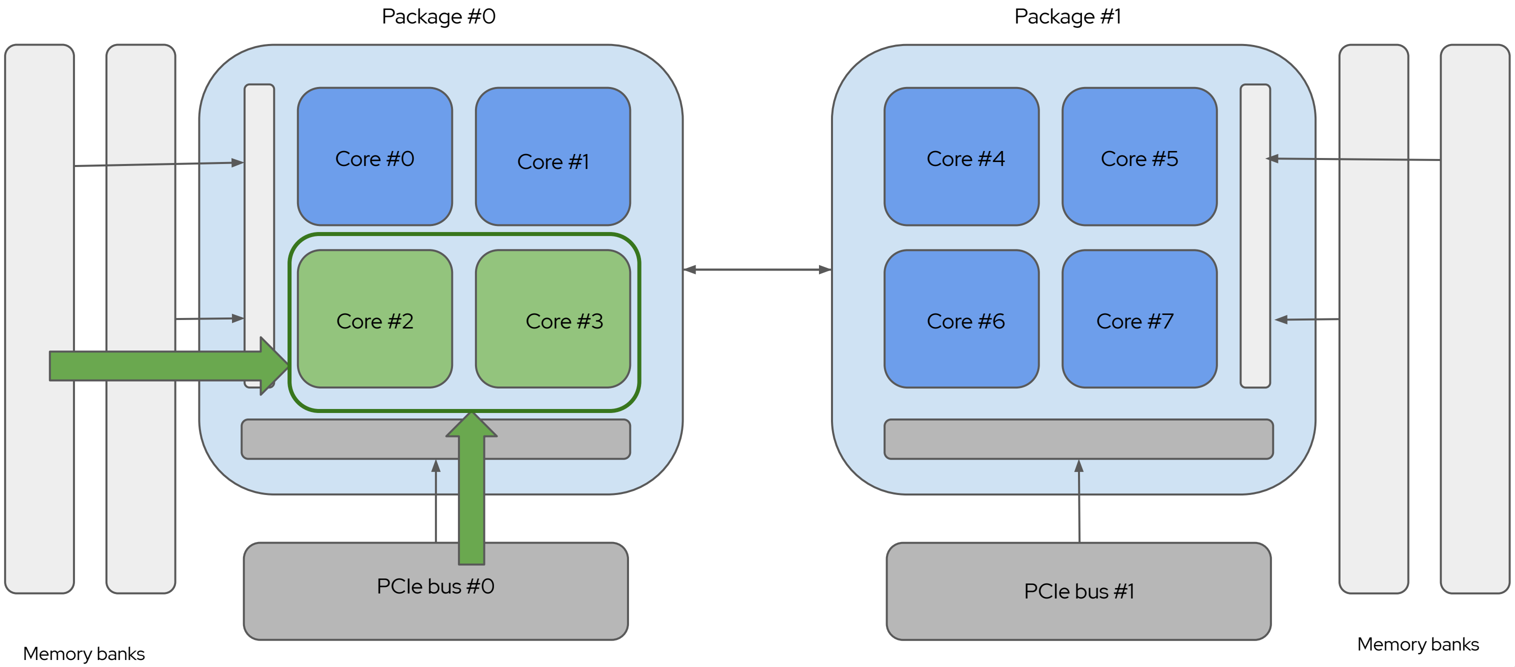
- One SR-IOV Virtual Function
- Some Huge pages
- Two CPU cores

Example: Not-NUMA-Aligned resources



KubeCon | CloudNativeCon
North America 2020
Virtual

Example: NUMA-Aligned resources



Micro Demo

NUMA-aligned resources with Topology Manager

Play Demo

Tunings: putting all together

Antares, CC0, via Wikimedia Commons

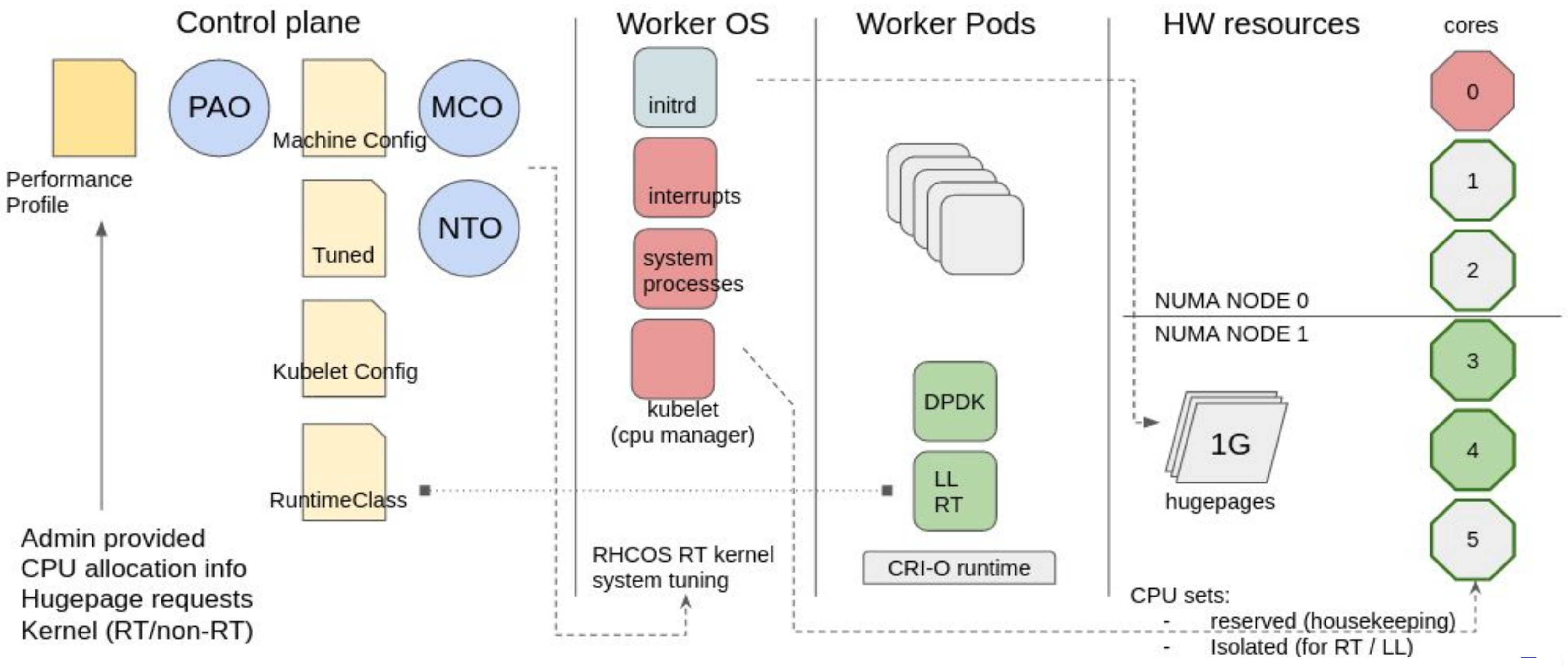
Performance Addon Operator

The performance addon operator (PAO) orchestrates all the moving parts by creating resources to utilize NTO, Kubelet and more complimentary settings that drive the system towards required tunings and maximum performance

Easy-to use high level profile: tune the cluster with opinionated optimal settings

Source:
https://github.com/openshift-kni/performance-addon-operators
Image credits: wikipedia CC BY-SA 3.0 nl

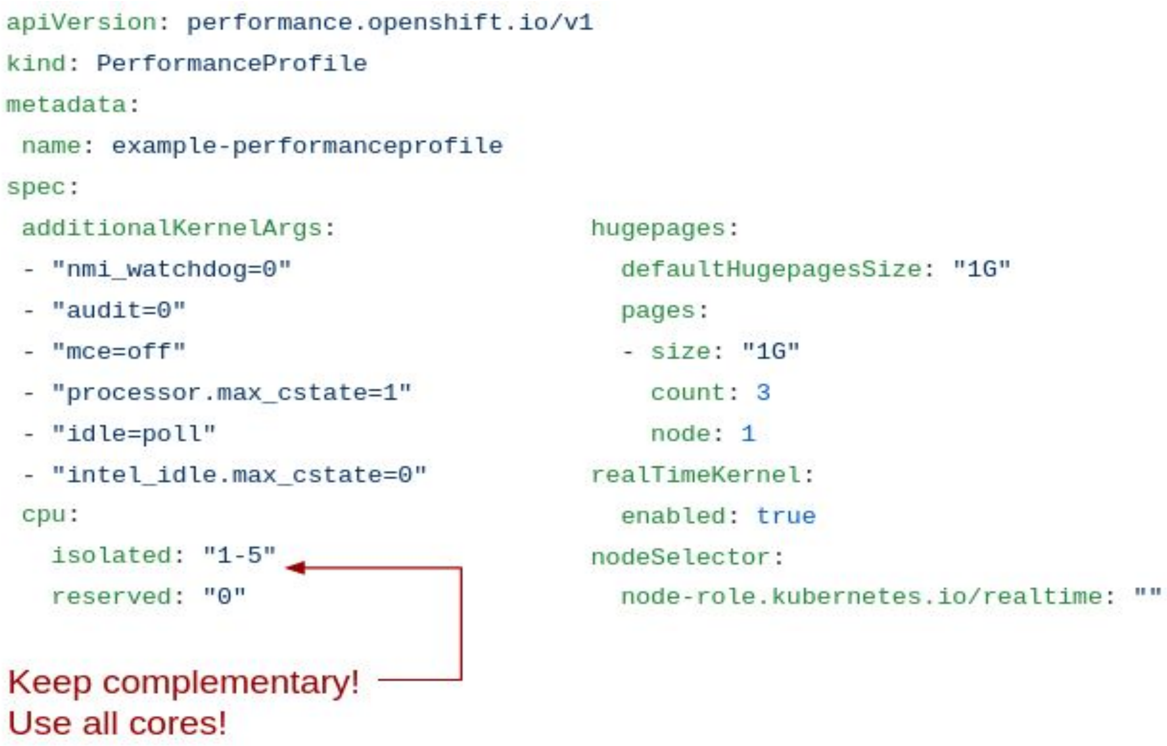
Performance Addon Operator as orchestrator



Admin provided
CPU allocation info
Hugepage requests
Kernel (RT/non-RT)

CPU sets:
- reserved (housekeeping)
- Isolated (for RT / LL)

Performance Addon Operator



```
apiVersion: performance.openshift.io/v1
kind: PerformanceProfile
metadata:
 name: example-performanceprofile
spec:
 additionalKernelArgs:
 - "nmi_watchdog=0"
 - "audit=0"
 - "mce=off"
 - "processor.max_cstate=1"
 - "idle=poll"
 - "intel_idle.max_cstate=0"
 cpu:
   isolated: "1-5"
   reserved: "0"
 hugepages:
   defaultHugepagesSize: "1G"
   pages:
   - size: "1G"
     count: 3
     node: 1
 realTimeKernel:
   enabled: true
 nodeSelector:
   node-role.kubernetes.io/realtime: ""
```

Keep complementary!
Use all cores!

Source:
https://github.com/openshift-kni/performance-addon-operators

Micro Demo

Performance profile deployment

[Play Demo]

Try it out!

- The performance-addon-operator is open source!
  - Explore the source code
  - Find the documentation
  - Installation instructions (Operator Lifecycle Manager suggested)
  - Future plans: operatorhub.io integration
- Pre-built container images on quay.io

KEEP CLOUD NATIVE EVERYWHERE

KubeCon | CloudNativeCon

North America 2020

Virtual