FOSDEM 2022

This is the way: A Crash Course on the Intricacies of Managing CPUs in K8s

Marlow Weston: Cloud Native Architect

Swati Sehgal: Cloud Native Architect

Use Cases: How to optimize performance and maximize capability

Why should you care about CPU Management?

Performance

- Core Pinning
  - Context switching is expensive
  - Want to not be swapping memory in or out during a data transfer

Why should you care about CPU Management?

Security

- Core Pinning
  - You don’t want to share CPUs with others

Why should you care about CPU Management?

Maximize Resources Available

- Some cores in your processors may be more performant than others. You may want to classify these, choose class A for performant and class B for less performant cases and allocate appropriately

Why should you care about CPU Management?

Maximize Resources Available

- Power per Watt
  - Intelligently managing power on your node can save you both power (and money) and wear and your processors

Why should you care about CPU Management?

High Performance CPUs vs Low Performance CPUs on the same node

- Power savings
  - Choose to have the CPU at the performance you care about

Why should you care about CPU Management?

High Performance CPUs vs Low Performance CPUs on the same node

- Power savings
  - Choose to have the CPU at the performance you care about
- Monitoring job vs. performance jobs on same node
  - Save the turbo voltage for the CPUs you want performance from

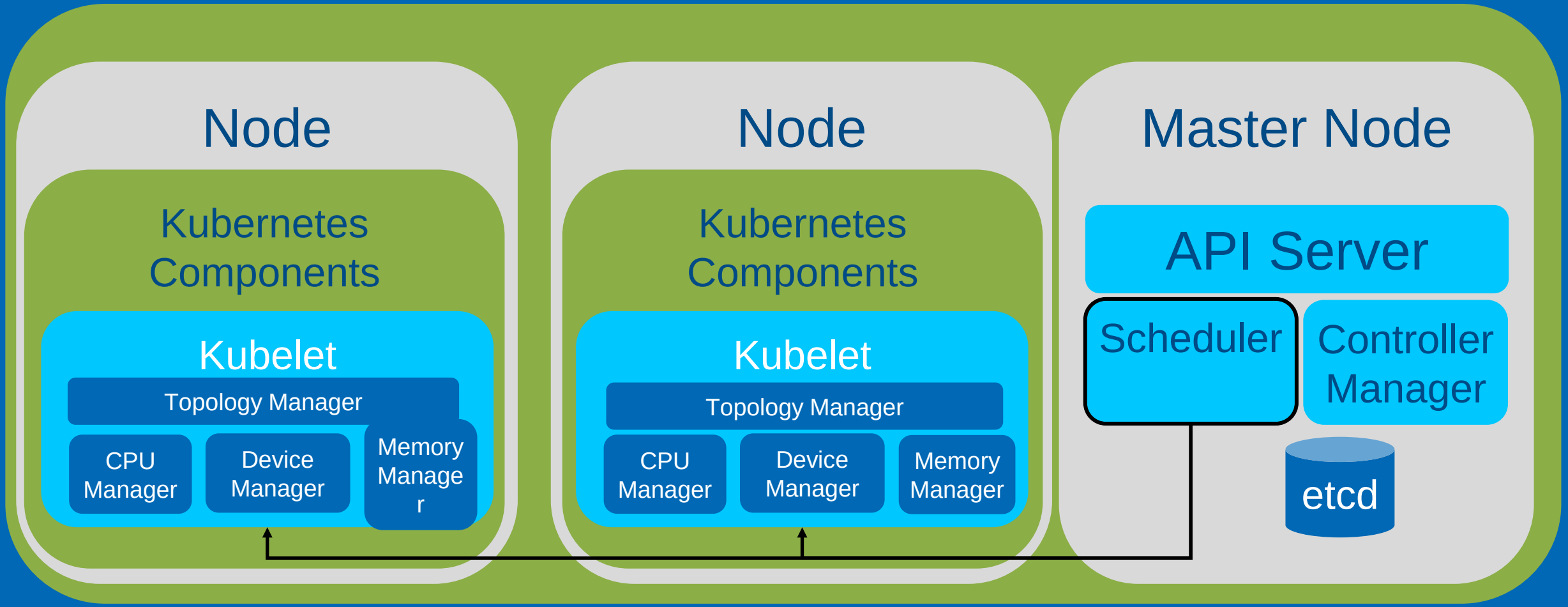
Why should you care about CPU Management?

Reserve CPUs outside of the Kubelet

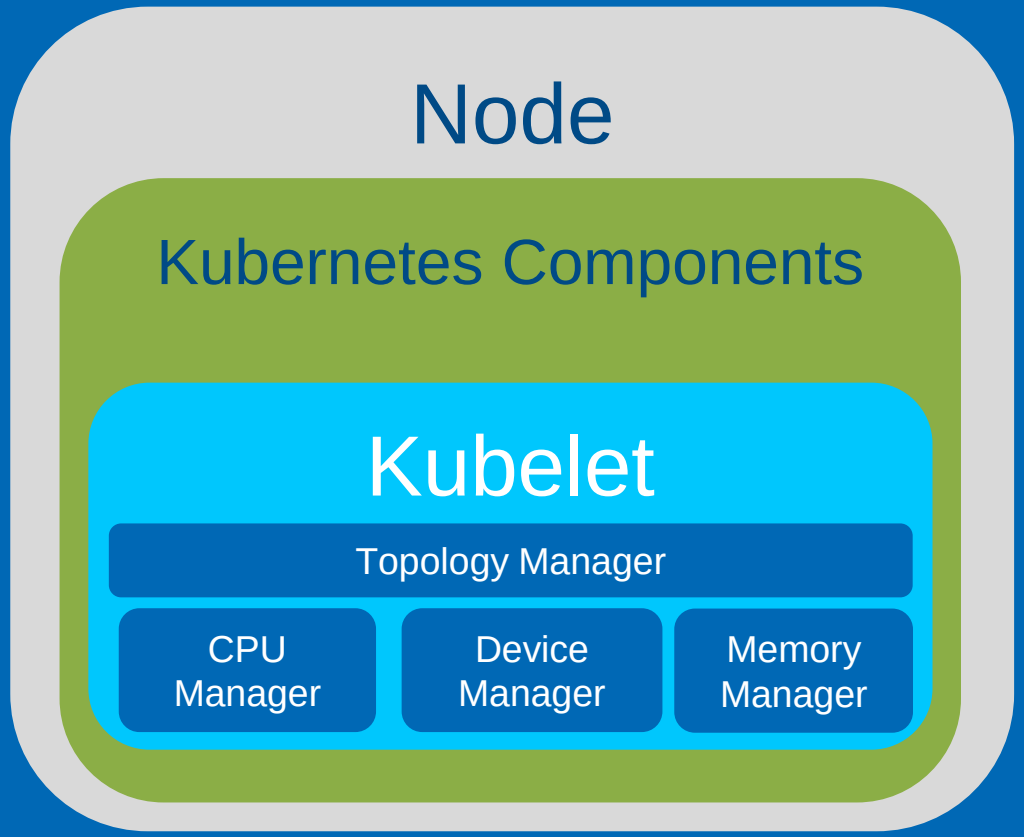
- You may have per-node processes you want running outside of Kubernetes – monitoring, security checks, et cetera

What can we do today?

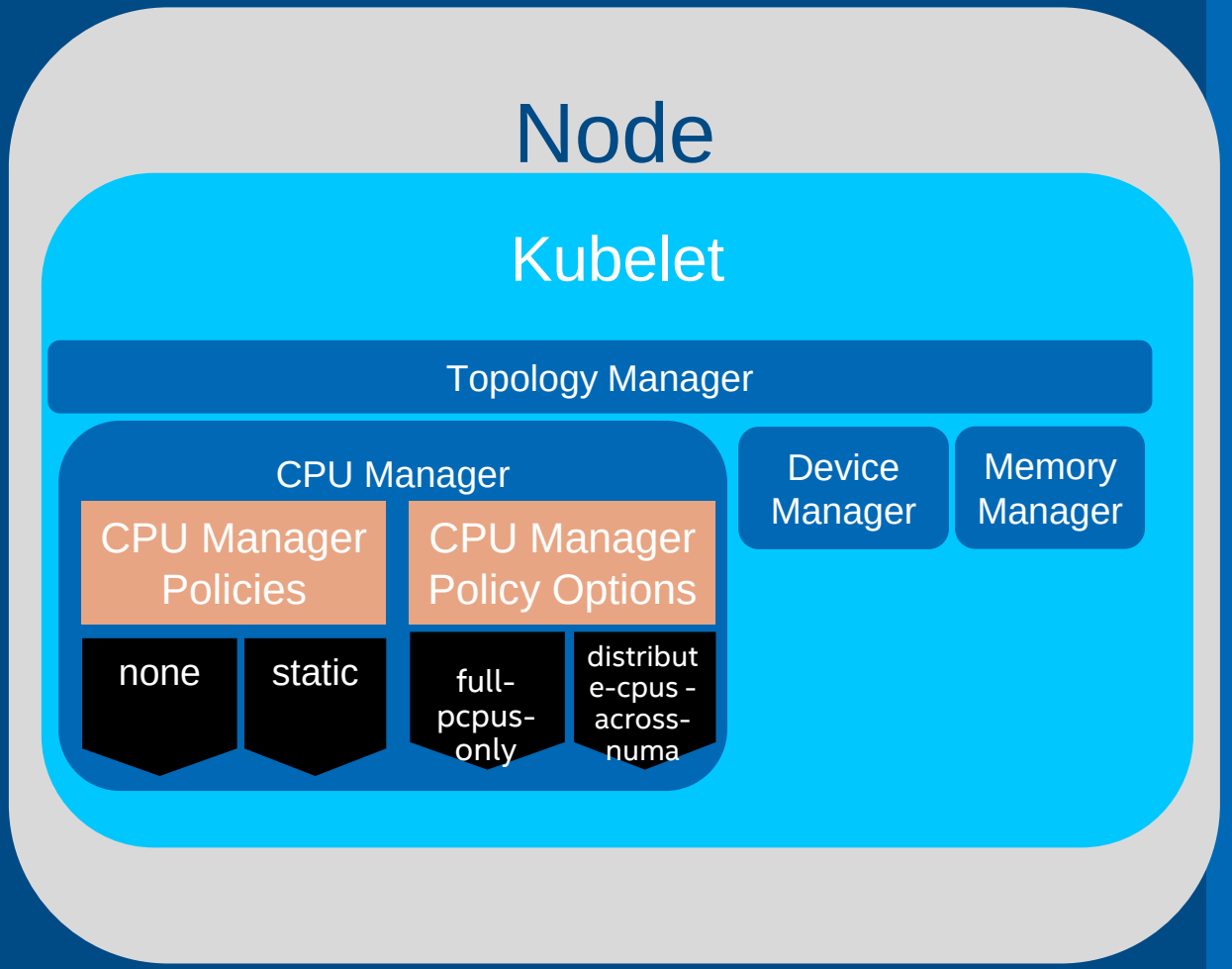
Kubernetes General Architecture



Resource Management and Kubelet components



CPU Manager



CPU Manager when enabled with static policy allocates CPUs exclusiv ely for a container if

- pod QoS is **`Guaranteed`**
- has **a positive integer** CPU request

The static policy maintains the following sets of logical CPUs:

- Exclusive
- Shared
- Reserved
- Assignable

Pinning CPUs to a Container
- Use CPU Manager Policies: Static vs. Default

--cpu-manager-policy kubelet flag used to specify the policy

None

- Default
- Provides no affinity beyond what the OS scheduler does automatically
- Can handle partial CPUs

Pinning CPUs to a Container
- Use CPU Manager Policies: Static vs. Default

--cpu-manager-policy kubelet flag used to specify the policy

Static

- allows containers access to exclusive CPUs on the node
- does not change CPU assignments for exclusively pinned guaranteed containers after the main container process starts
- Only uses whole CPUs, so increases perceived CPU utilization
- Only by Pod, not by container

Pinning CPUs according to a particular core: Use CPU Manager Policy Options

CPU Manager PolicyOptions can be **used to** fine-tune **the** behavior of static **policy** -- cpu-manager-policy-options kubelet flag

full-pcpus-only

- Beta option, visible by **default**
- **the** static policy will always allocate full physical cores, so guarantee same NUMA zone.
- Fails for partial core allocation.

Pinning CPUs by a particular core: Use CPU Manager Policy Options

CPU Manager PolicyOptions can be **used to** fine-tune **the** behavior of static **policy** --cpu-manager-policy-options kubelet flag

distribute-cpus-across-numa

- alpha, hidden by default
- the static policy will evenly distribute CPUs across NUMA nodes
- **Still per container**

Having Cores Outside of the Kubelet:

Explicitly specify Reserved CPU List

--reserved-cpus=n-n+x

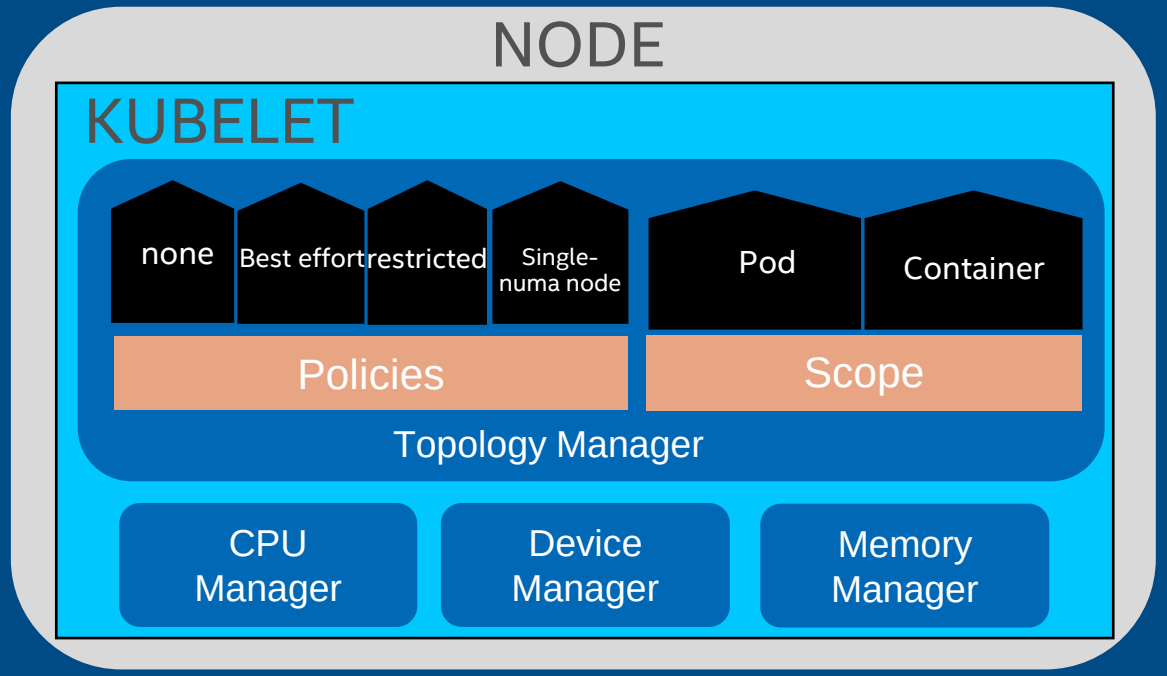
- Explicitly specify **CPU set for OS system daemons and Kubernetes system daemons**

Use the Native Kubernetes CPU Manager

Performance

- Core Pinning
  - Can choose “static” and “none” policies for containers.
  - Security is not guaranteed in this case at this time
  - ALL of this pinning is per container, NOT per pod
  - Provides NUMA hints to the Topology Manager

Topology Manager and Resource Managers



Topology Manager is a Kubelet components for topology alignment of resources.

Aligns resource of pods of

- All QoS classes

Resource alignment can be performed at a pod or container level.

- Kubelet flag **--topology-manager-scope** used to configure this

Topology Manager policies (set via Kubelet flag **--topology-manager-policy**):

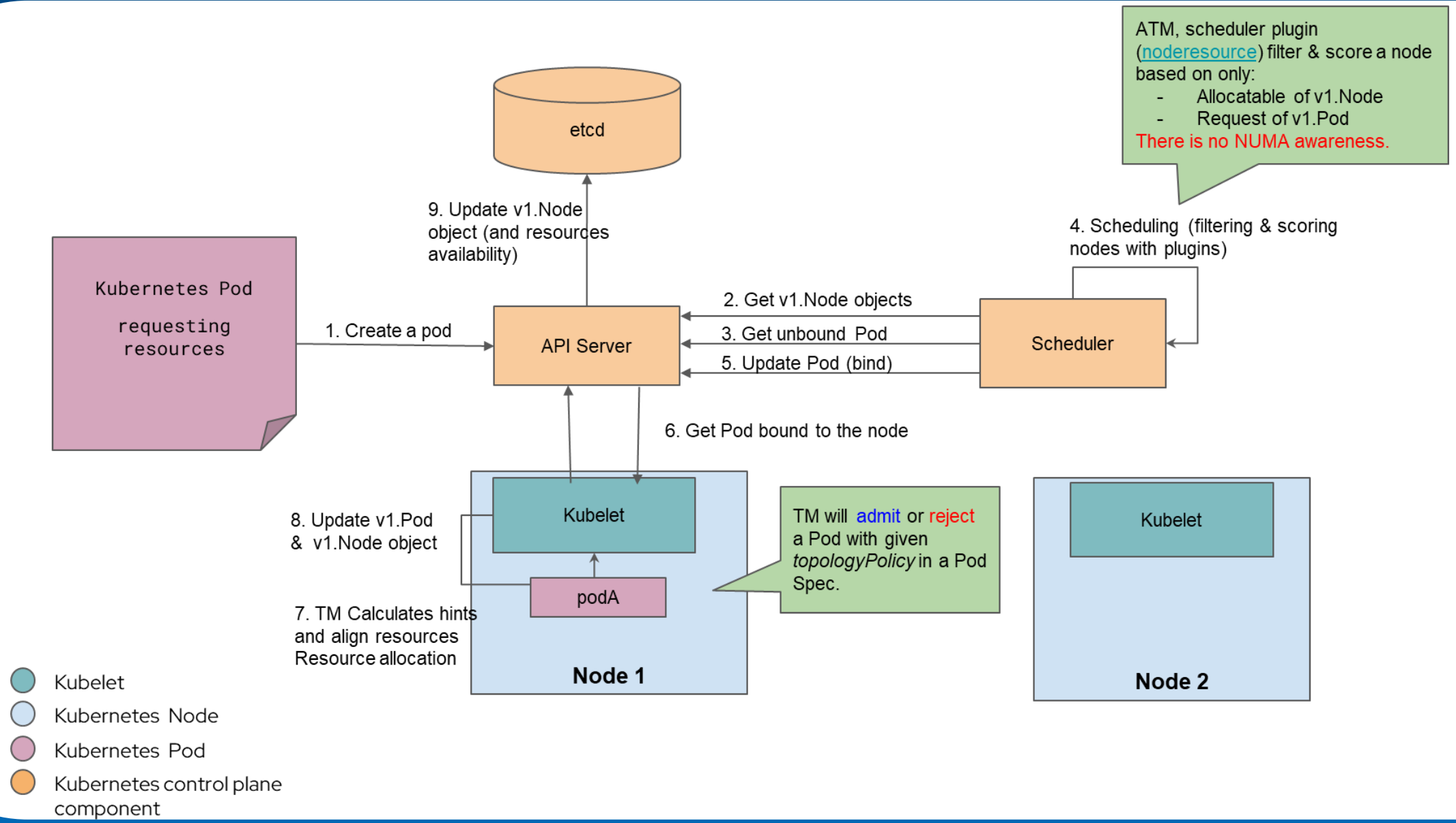
- none (default)
- best-effort
- restricted
- single-numa-node

Aligning resources in same NUMA node and making scheduler topology-aware

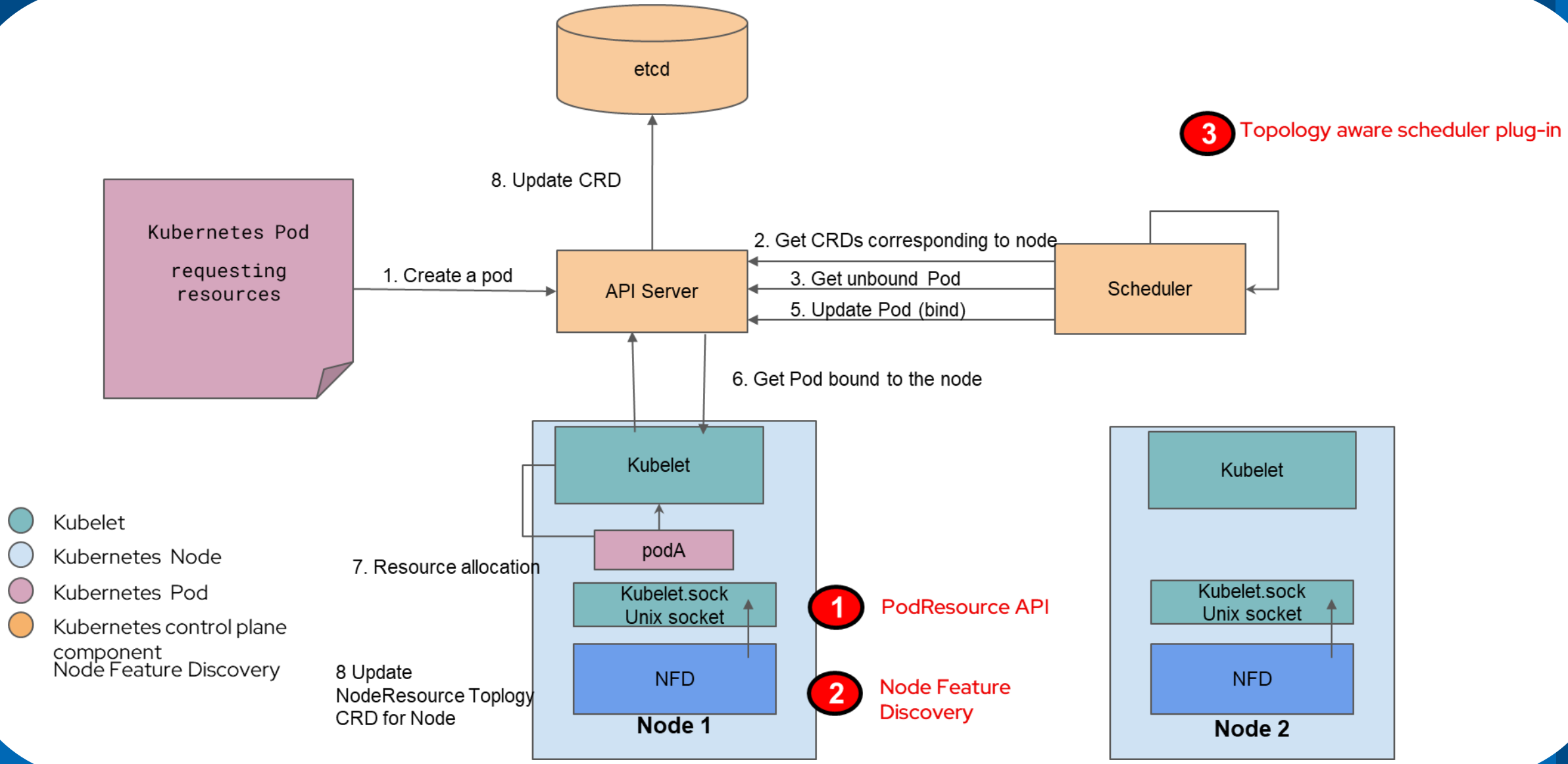
Performance

- Optimize scheduling behavior to make sure CPUs are in same NUMA region as other resources (CPUs, memory, xPUs, other devices)

Scheduling with Default Scheduler



Scheduling with Topology aware Scheduling



Use External Resource Managers

Power Management

- Use the Kubernetes Power Manager to change the performance according to profiles, either at a container level or at a node level

Write Your Own

Mixed CPU needs

- No current ability to handle this.
- Change the cpusets behind the scenes on the node directly

How Can You Get Involved In the Community?

Communities

For more information refer to:

- SIG Node: Handles more on-the-node cpu and resources https://github.com/kubernetes/community/tree/master/sig-node
- SIG Scheduling: Handles Scheduling across nodes: https://github.com/kubernetes/community/tree/master/sig-scheduling
- Topology aware Scheduling:

  https://github.com/k8stopologyawareschedwg

Slack Channels

For more information refer to:

- SIG Node Slack: https://kubernetes.slack.com/messages/sig-node
- SIG Scheduling Slack: https://kubernetes.slack.com/messages/sig-scheduling
- Topology aware Scheduling:

  https://kubernetes.slack.com/messages/topology-aware-scheduling

Questions?


intel