NUMA Awareness in Kubernetes with Topology Manager

Louise Daly, Intel
Feng Pan, Red Hat
Agenda

• Intro
• CPU Manager and Device Manager in Kubernetes
• Topology Manager
• What’s next
• How to get involved
The Need For NUMA Awareness

- Workloads in fields such as telecommunications, scientific computing, machine learning, financial services and data analytics often have NUMA alignment as a requirement to run.
- A DPDK based networking application may require dedicated CPUs, huge pages and SR-IOV VFs in a single NUMA node to achieve desired results.
A Broader Context

• An increasing number of systems require direct access to CPUs and hardware accelerators to support performance sensitive applications.
• Hardware resources need to be co-ordinated or performance will be negatively impacted
CPU Manager

• “Static” CPU Manager policy
• Allocate exclusive CPUs to certain pod containers
• Node level config

```yaml
apiVersion: v1
class: Pod
metadata:
  name: guaranteed-pod
spec:
  containers:
    - name: g-container
      image: nginx
      resources:
        requests:
          memory: "64Mi"
          cpu: 2
        limits:
          memory: "64Mi"
          cpu: 2
```
Device Plugins

- Advertise system hardware resources to the Kubelet
- Enables vendor specific initialization and setup
- API for Device Plugins to communicate with Device Manager
Introducing: Topology Manager

• **Alpha as of Kubernetes 1.16**
• CPU and Device Manager assign resources independently which can result in sub-optimal placement
• Topology Manager, a new kubelet component, provides an interface to co-ordinate resource assignment on node level
• CPU Manager and Device Manager are the first components to implement the Topology Manager interface
Continued: Topology Manager

apiVersion: v1
class: Pod
metadata:
  name: guaranteed-pod
spec:
  containers:
  - name: g-container
    image: nginx
    resources:
      requests:
        memory: "64Mi"
        cpu: 2
    limits:
      memory: "64Mi"
      cpu: 2
    vendor/device-a: 1
Node Level Policies

- **None**: Default policy that does not perform any Topology Alignment
- **Best-effort**: Tries to align resources on the optimal chosen alignment
- **Restricted**: Enforces optimal chosen alignment or fails the pod admission
- **Single-numa-node**: Aligns all resources on a single NUMA node or fails the pods admission
Policy Examples

Policy:
• Best Effort, Restricted or Single NUMA Node:
Pod Admitted with Topology Locality.
Policy Examples

Policy:

• Best Effort:
  Pod Admitted across NUMA nodes.

• Restricted:
  Topology Affinity Error - Preferred is False.

• Single NUMA Node:
  Topology Affinity Error - Request cannot be satisfied on a single NUMA Node.
Policy Examples

Policy:

- **Best Effort:**
  Pod Admitted across NUMA nodes.

- **Restricted:**
  Pod Admitted across NUMA nodes.

- **Single NUMA Node:**
  Topology Affinity Error - Request cannot be satisfied on a single NUMA Node.
What’s Next

- Memory support - HugePages
- Device interconnect discovery - Multi GPU
- PCIE Root Complex detection - GPU Direct with RDMA
- Topology Aware Scheduling
How to get involved

• Try out Topology Manager!
  • kubectl feature gate:
    ```shell
    --feature-gates=TopologyManager=true
    ```
  • kubectl flag:
    ```shell
    --topology-manager-policy=best-effort|restricted|single-numa-node
    ```

• Topology Manager Documentation

• SIG Node:
  • Slack: Kubernetes/SIG-Node
  • Google Group
  • Weekly Meeting: Every Tuesday 17:00 UTC

• Topology Management Github issue
Thank you!
Q & A
Backup slides
Backup slides

- CPU Manager diagram

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-a
spec:
  containers:
  - name: container-a
    image: nginx
    resources:
      requests:
        memory: "64Mi"
        cpu: 2
      limits:
        memory: "64Mi"
        cpu: 2
```