TECHNOLOGY GUIDE

Intel Corporation

Topology Management – Implementation in Kubernetes* Technology Guide

Authors

Louise Daly

Conor Nolan

Przemyslaw Lal

David Lu

1 Introduction

An increasing number of systems leverage a combination of CPUs and hardware accelerators to support latency-critical execution and high-throughput parallel computation. These include workloads in fields such as telecommunications, scientific computing, machine learning, financial services and data analytics. Such hybrid systems comprise a high-performance environment.

To extract the best performance, the required optimizations related to CPU isolation, memory, and device locality must be made. However, in Kubernetes*, these optimizations are handled by a disjointed set of components.

Topology Manager is a kubelet component that aims to coordinate the set of components that are responsible for these optimizations.

This technology guide is intended for Service Providers, Telecom Equipment Manufacturers (TEMs), Network Monitoring and Management solution providers or those planning to deploy closed loop automation solutions running on Intel® Xeon® Processors.

Topology Manager is an alpha feature in Kubernetes v1.16, graduating to beta in Kubernetes v1.18. The software is available at: https://github.com/kubernetes/kubernetes/tree/release-1.18/pkg/kubelet/cm/topologymanager.

Documentation for Topology Manager on the Kubernetes website is available at: https://kubernetes.io/docs/tasks/administer-cluster/topology-manager/.

A blog post for Topology Manager on the Kubernetes website is available at: https://kubernetes.io/blog/2020/04/01/kubernetes-1-18-feature-topoloy-manager-beta/.

This document is part of the Network Transformation Experience Kit, which is available at: https://networkbuilders.intel.com/network-technologies/network-transformation-exp-kits

Table of Contents

Tables

Figures

1.1 Terminology

Table 1. Terminology

ABBREVIATION	DESCRIPTION
CPU	Central Processing Unit
DPDK	Data Plane Development Kit
NUMA	Non-Uniform Memory Access
RAM	Random Access Memory
QoS	Quality of Service
TEM	Telecom Equipment Manufacturers

1.2 Reference Documents

Table 2. Reference Documents

REFERENCE	SOURCE
Kubernetes References: Standardized Glossary	https://kubernetes.io/docs/reference/glossary/?all=true#term-qos-class
Kubernetes Tasks: Control CPU Management Policies on the Node	https://kubernetes.io/docs/tasks/administer-cluster/cpu-management-policies/
Kubernetes Tasks: Control Topology Management Policies on a Node	https://kubernetes.io/docs/tasks/administer-cluster/topology-manager/

2 Overview

Prior to the introduction of the Topology Manager, the CPU Manager and the Device Manager in Kubernetes* made resource allocation decisions independently of each other, which resulted in undesirable allocations on multiple-socketed systems. Performance/latency sensitive applications suffered due to these undesirable allocations. Undesirable, in this case, meaning CPUs and devices being allocated from different Non-Uniform Memory Access (NUMA) Nodes, thus incurring additional latency.

The Topology Manager is a kubelet component, which acts as a source of truth so that other kubelet components can make topology aligned resource allocation choices.

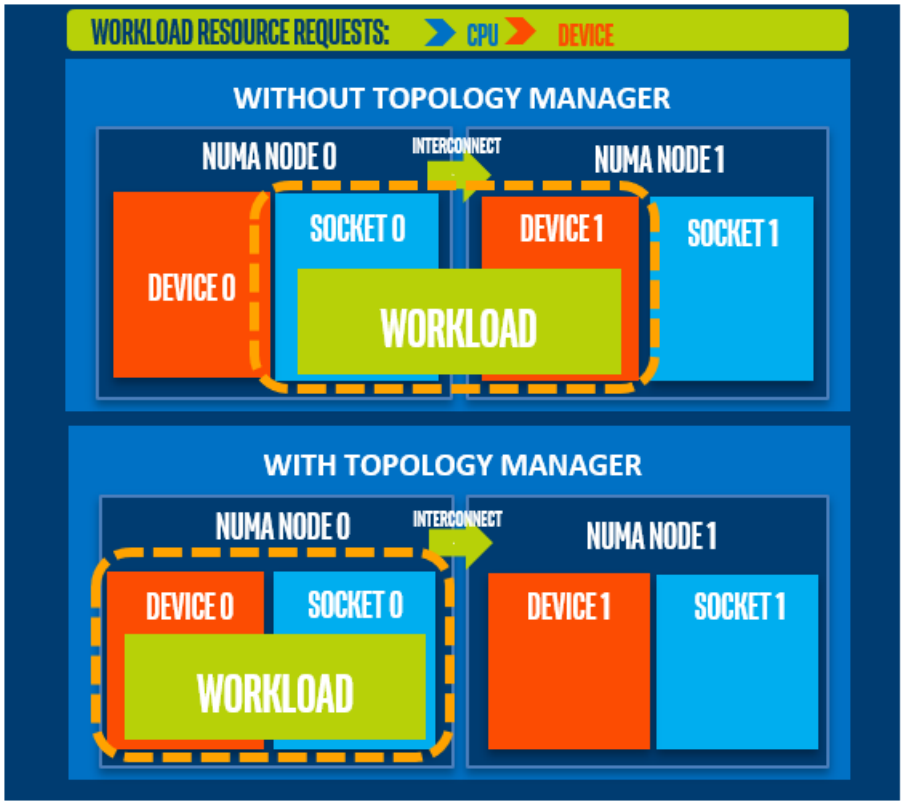

Figure 1. Overview with and without Topology Manager

The Topology Manager provides an interface for components, called *Hint Providers*, to send and receive topology information. Topology Manager has a set of node level policies which are explained below.

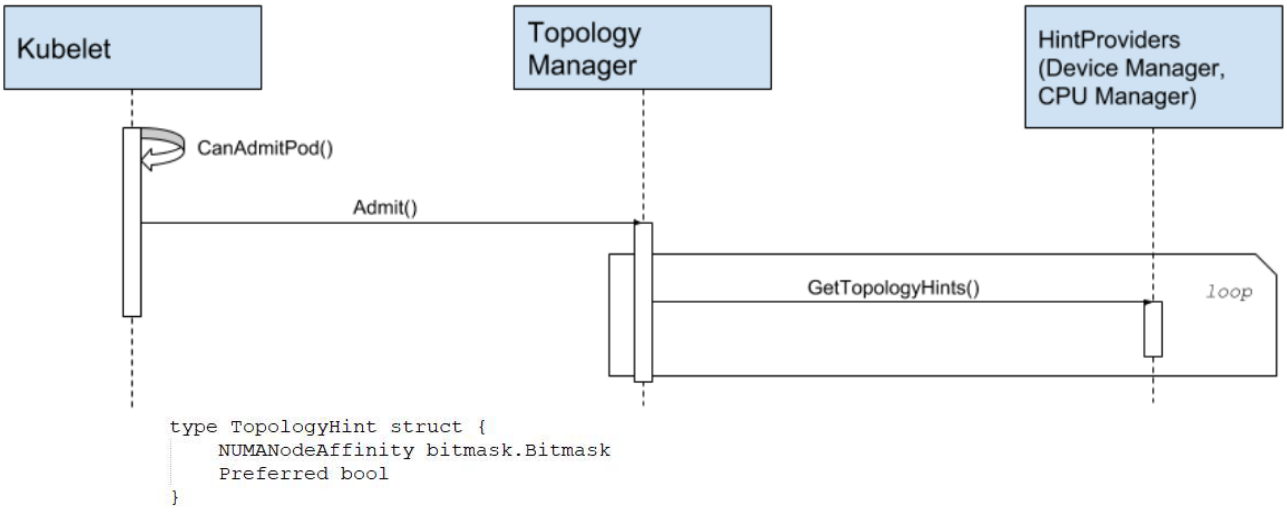

Figure 2. Node Level Policies

The Topology Manager receives Topology information from the *Hint Providers* as a bitmask denoting NUMA Nodes available and a preferred allocation indication. The Topology Manager policies perform a set of operations on the hints provided and converge on the hint determined by the policy to give the optimal result. If an undesirable hint is stored, the preferred field for the hint will be set to **false**. In the current policies, preferred is the narrowest preferred mask. The selected hint is stored as part of the Topology Manager. Depending on the policy configured, the Pod can be accepted or rejected from the node based on the selected hint. The hint is then stored in the Topology Manager for use by the *Hint Providers* when making the resource allocation decision.

3 Deployment

3.1 Enabling Topology Manager

In Kubernetes* v1.16 and v1.17, Topology Manager is supported as an alpha feature and is disabled by default behind a kubelet feature gate. To enable Topology Manager, the Topology Manager kubelet feature gate must be set to **true**, **`TopologyManager=True`** through the kubelet arguments.

In Kubernetes v1.18, Topology Manager is a beta feature. For this version and future versions, the Topology Manager feature gate is enabled by default.

Topology Manager works on Nodes with the **static** CPU Manager Policy enabled. For details, see Control CPU Management Policies.

If these conditions are met, Topology Manager will align CPU and device requests.

3.2 Topology Manager Policies

Topology Manager supports four allocation policies. You can set a policy through a kubelet flag, --**topology-manager-policy**. There are four supported policies:

1. **none policy**
   This is the default policy and does not perform any topology alignment.
2. **best-effort policy**
   For each container in a Pod, a kubelet, with the **best-effort** topology management policy, calls each Hint Provider to discover their resource availability. Using this information, the Topology Manager stores the preferred NUMA Node affinity for that container. If the affinity is not preferred, the Topology Manager will store this and admit the Pod to the node anyway.
   The *Hint Providers* can then use this information when making the resource allocation decision.
3. **restricted policy**
   For each container in a Pod, kubelet, with the **restricted** topology management policy, calls each Hint Provider to discover their resource availability. Using this information, the Topology Manager stores the preferred NUMA Node affinity for that container. If the affinity is not preferred, Topology Manager will reject this Pod from the node. This will result in a Pod in a **Terminated** state with a Pod admission failure.
   If the Pod is admitted, the *Hint Providers* can then use this information when making the resource allocation decision.
4. **single-numa-node policy**
   For each container in a Pod, kubelet, with the **single-numa-node** topology management policy, calls each Hint Provider to discover their resource availability. Using this information, the Topology Manager determines if a single NUMA Node affinity is possible. If it is, the Topology Manager will store this, and the *Hint Providers* can then use this information when making the resource allocation decision. If, however, this is not possible, the Topology Manager will reject the Pod from the node. This will result in a Pod in a **Terminated** state with a Pod admission failure.

3.3 Pod Interactions with Policies

Consider the containers in the following Pod specs:

```
spec:
  containers:
  - name: nginx
    image: nginx
```

This Pod runs in the `BestEffort` Quality of Service (QoS) class because no resource `requests` or `limits` are specified.

```
spec:
  containers:
  - name: nginx
    image: nginx
    resources:
      limits:
        memory: "200Mi"
      requests:
        memory: "100Mi"
```

This Pod runs in the Burstable QoS class because requests are less than limits.
Topology Manager would not consider either of these Pod specifications as they do not request any CPU or device resources.

```
spec:
  containers:
  - name: nginx
    image: nginx
    resources:
      limits:
        memory: "200Mi"
        cpu: "2"
        example.com/device: "1"
      requests:
        memory: "200Mi"
        cpu: "2"
        example.com/device: "1"
```

This Pod runs in the `Guaranteed` QoS class because requests are equal to limits.

```
spec:
  containers:
  - name: nginx
    image: nginx
    resources:
      limits:
        example.com/deviceA: "1"
        example.com/deviceB: "1"
      requests:
        example.com/deviceA: "1"
        example.com/deviceB: "1"
```

This Pod runs in the `BestEffort` QoS class because there are no CPU and memory requests.

The Topology Manager would consider the Pods running in the `Guaranteed` and the `BestEffort` classes and would consult the *Hint Providers*.

In the case of the `Guaranteed` Pod, the static CPU Manager policy would return hints relating to the CPU request and the Device Manager would return hints relating to the requested device.

In the case of the `BestEffort` Pod, the CPU Manager would return the default hint as there is no CPU request and the Device Manager would return the hints for each of the requested devices. This example Pod interaction is applicable to Topology Manager in Kubernetes v1.17 and newer. Topology Manager in Kubernetes v1.16 does not consider Pods running in the `BestEffort` or the `Burstable` classes under any circumstances.

Using this information, the Topology Manager calculates the optimal hint for the Pod and stores this information, which will be used by the Hint Providers when they are making their resource assignments.

4 Benefits

Performance benchmarking on the Topology Manager was conducted by Intel using the `testpmd` application that is shipped as part of the Data Plane Development Kit. Throughput figures are compared for a setup with no NUMA alignment of Pod resources (refer to Figure 3) and a setup using the Topology Manager to achieve NUMA alignment of the Pod resources (refer to Figure 4).

4.1 Test Details

Table 3. Hardware Components for Performance Benchmark Tests

ITEM	DESCRIPTION	NOTES
Platform	Intel® Server Board S2600WT2	Version: H21573-360 Serial: BQWL44850120
Processor	2x Intel® Xeon® Gold 6252N	@ 2.30 Ghz
Memory	192GiB RAM	24x 8GiB Kingston 9965589-001.E00G DIMM DDR4 Synchronous 2666MHz (64GiB total)
NIC	2x Intel® E810-CQDA2 100GbE QSFP28 Network Adapter 2x Mellanox* MT28800 Family (ConnectX-5 Ex)	2 x 100 GbE ports per NIC. Mellanox NIC used for TRex traffic generation only.
BIOS	Intel Corporation SE5C620.86B.02.01.1008.041420190659 Release Date: 04/14/2019	P-state, Intel® HT Technology, Intel® VT-x, Intel® VT-d, Intel® Turbo Boost enabled

Table 4. Software Components for Performance Benchmark Tests

SOFTWARE COMPONENT	DESCRIPTION
Linux* Kernel	Kernel: v3.10.0-514.21.el7.x86_64
Data Plane Development Kit (DPDK)	DPDK v19.11
Docker	v1.19
Kubernetes*	V1.16.2
Multus CNI	V3.3
Flannel CNI	V0.11
SR-IOV Device Plugin	V3.1
Workloads: SR-IOV device plugin, hugepages and testpmd	testpmd is a Data Plane Development Kit (DPDK)-based application. It was configured in I/O forwarding mode.

4.2 Test Results

These tests were completed by Intel in March 2020.[1]

Table 5. Performance Benchmark Test Results

PACKET SIZE (B)	DPDK THROUGHPUT WITH NUMA ALIGNMENT (GBPS)	DPDK THROUGHPUT WITHOUT NUMA ALIGNMENT (GBPS)	PERFORMANCE IMPROVEMENT (%)
64	58.81	27.97	110
128	102.36	48.46	111
256	190.60	86.59	120
512	198.09	161.58	22
1024	200.00	199.99	0
1280	200.00	199.98	0
1518	200.01	199.99	0

5 Summary

In summary, Topology Manager facilitates enhanced resource management in Kubernetes* through NUMA alignment of workload resources. This results in increased throughput and improved performance for workloads.

The adoption of 5G and edge computing sees workloads move out of core data centers and into edge compute nodes. This shift has presented a new challenge: workloads are now running on a smaller footprint with less resources, whilst still maintaining their demand for low latency and high throughput.

Kubernetes, with its cloud native approach, is seen as the orchestrator of choice in these environments. However, Kubernetes is still lacking in some resource management capabilities required to maximize resource potential. For instance, optimal workload placement based on the underlying system topology is not considered during container resource allocation.

Topology Manager enhances resource management in Kubernetes by coordinating a set of previously disjointed individual resource managers to achieve NUMA alignment of CPUs and hardware accelerators for a workload.

This document describes the usage and benefits of Topology Manager in Kubernetes, including:

- Enabling Topology Manager
- Topology Manager Policies
- Workload behavior and placement about Topology Manager Policies

For more information on what Intel is doing with containers, see:
https://networkbuilders.intel.com/network-technologies/intel-container-experience-kits

[1] Refer to Section 4.1 for configuration. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks
Refer to http://software.intel.com/en-us/articles/optimization-notice for more information about optimization.

Appendix A Test Setup

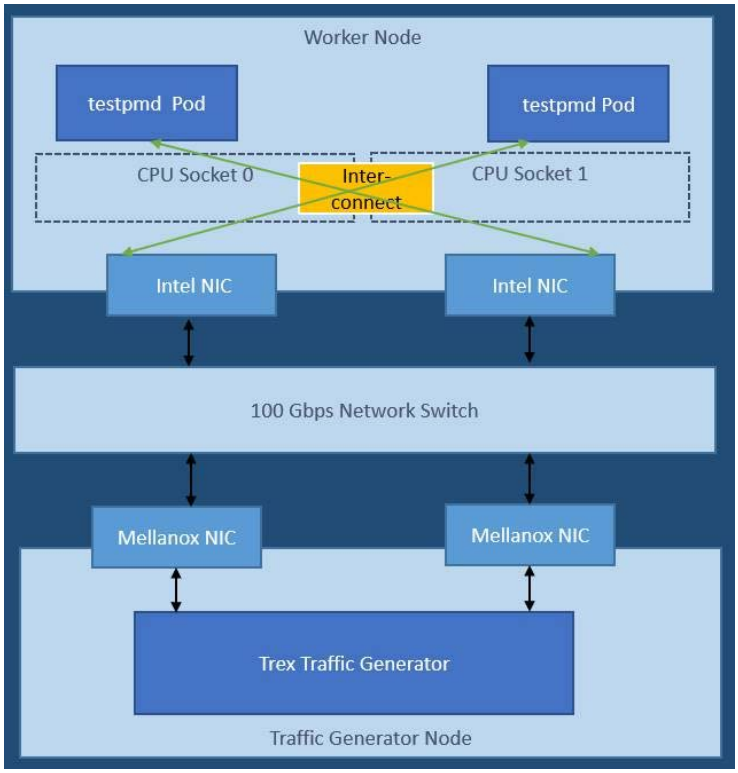

Figure 3. Test Setup without NUMA Alignment

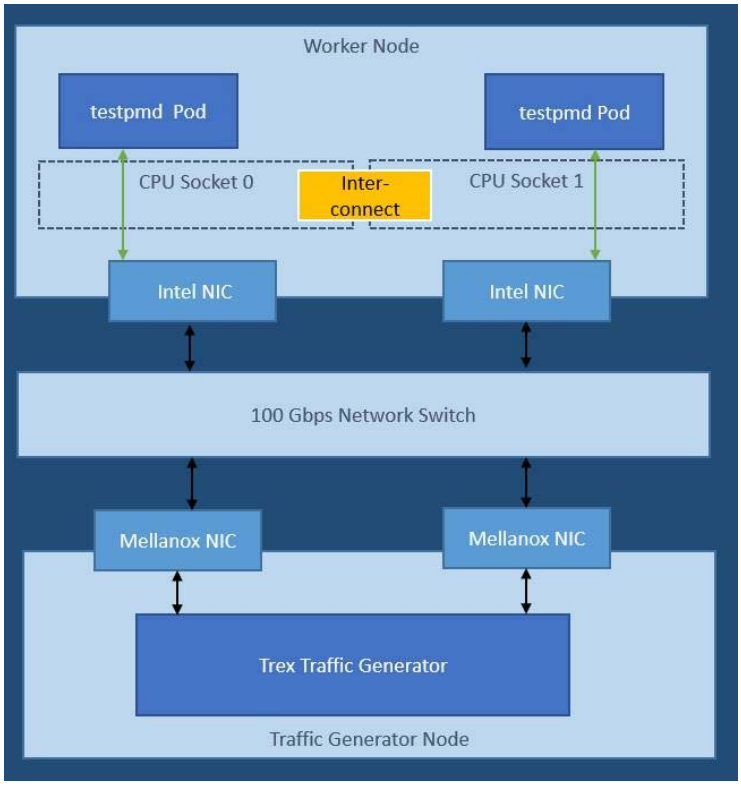

Figure 4. Test Setup with NUMA Alignment by the Topology Manager

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors.

Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See configuration disclosure for details. No product or component can be absolutely secure.

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or visiting www.intel.com/design/literature.htm.

0420/DN/PTI/PDF 621830-001US