SOSCON

Method of NUMA-Aware Resource Management for Kubernetes 5G NFV Cluster

Samsung Electronics | Samsung Research | Byonggon Chun
10.16, 2019

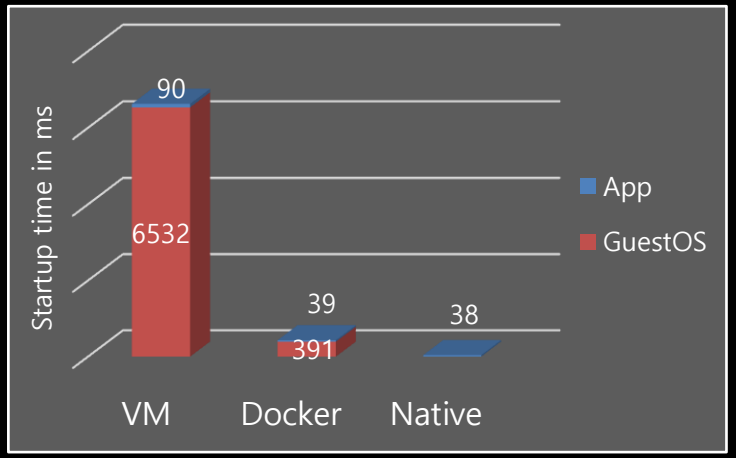
Byonggon Chun

Projects	Details
Tizen (2015~2016)	Tizen Web-Device API development
Iotivity (2016~2017)	Iotivity development based on OCF 1.0 spec *(Endpoint, Smarthome, etc)*
Edge Computing (2017~2018)	Factory Edge Computing PoC *(Based on EdgeX, DDS)*
	FaaS based Home Edge Computing PoC *(Based on Greengrass Core, OSS FaaS, etc)*
5G MEC (2018~2019)	5G MEC PoC ***(Based on LF Akraino, Openstak-helm, ETSI MEC Standard)***
Container-based NFV Infra (2019~)	NUMA-aware Resource Manager for CNF(PoC) *(CPU, Memory, Hugepages)*
	Opensource Contribution~ *(Kubernetes, Docker, Containerd)*

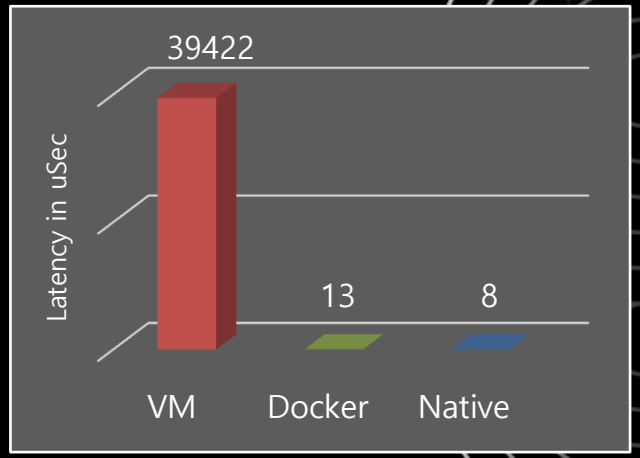
Agenda

Major benefits of Network Function Containerization

- Faster startup speed(quick to deploy)
- Lower performance overhead(no overhead from guest kernel)



Startup Benchmark with generic kernel

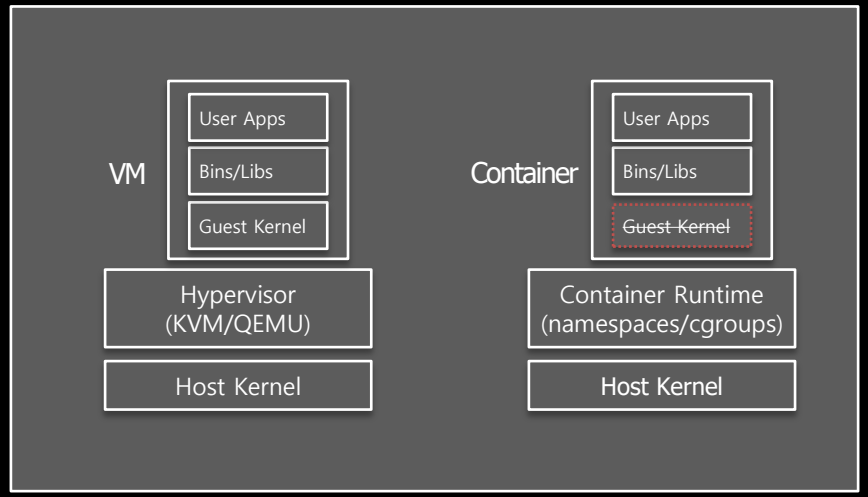


Cyclictest Benchmark with generic kernel

Cyclictest Benchmark source: Minimizing Latency of Real-Time Container Cloud for Software Radio Access Networks, IEEE CloudCom, 2015

Virtual Machine vs Container

- Q. So…is Container a new kind of Virtual Machine without kernel emulating?
- A. Nope, you should know about "Linux namespaces" and "Linux control groups".

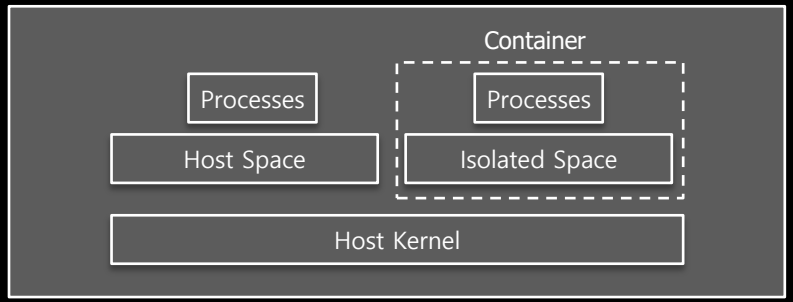


Difference between VM and Container

What is Container?

- The concept of container is lightweight mechanism to provide isolated environment.
- Processes are "**isolated by linux namespaces**".
- The resource usage is "**restricted by linux cgroup**".
- So the most of containers share host kernel.
- Sometimes containers running on the isolated kernel similar to virtual machine.

  (kata-runtime, gvisor, etc)

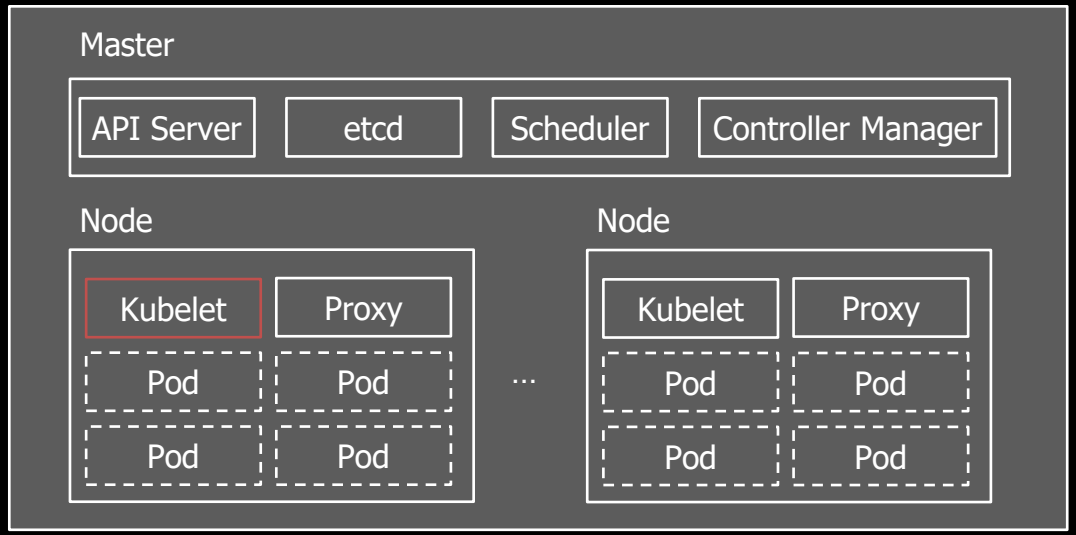


The fundamental concept of container

Deep dive into Kuburnetes at the node level

The structure of Kubernetes is straightforward.

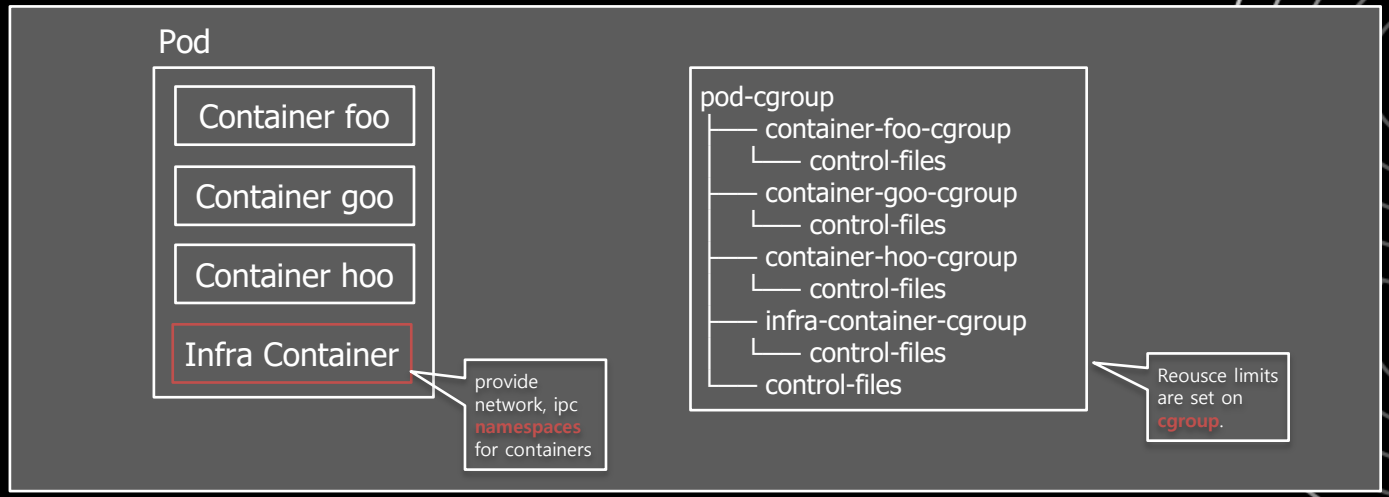
- Kubernetes consists of master components(APIs, scheduler, etc)
  and node components(kubelet, container-runtime, mandatory-services).



Overall architecture of Kubernetes

Let's tear down Pod.

- Pod is usually known as the basic execution unit or smallest deployable unit in Kubernetes.
- Let's see the Pod at the point of namespaces and cgroup.



The concept of Pod

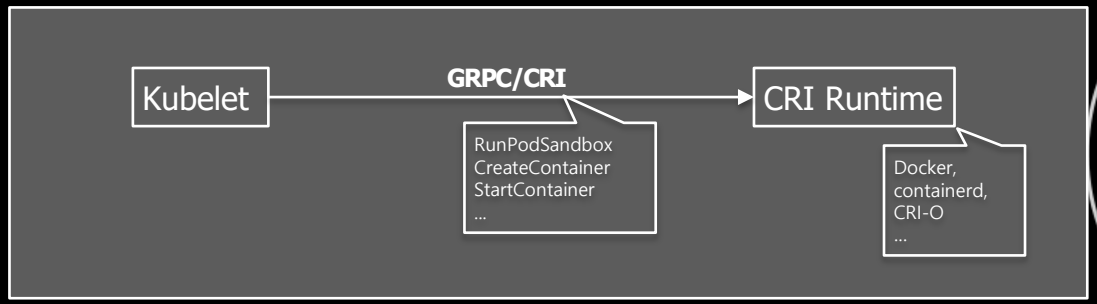
Deep dive into Kuburnetes at the node level

Let's talk about kubelet and container runtime.

- Kubelet communicates with container runtimes over CRI.
- CRI is developed for loosely coupled structure between kubelet and container runtimes.

  (But Kubelet still communicates with docker over dockershim which is part of kubelet)
- CRI offers set of gRPC APIs and protobuf messages for pod/container lifecycle management.

  (CRI runtime runs CRI runtime service server, kubelet is client)



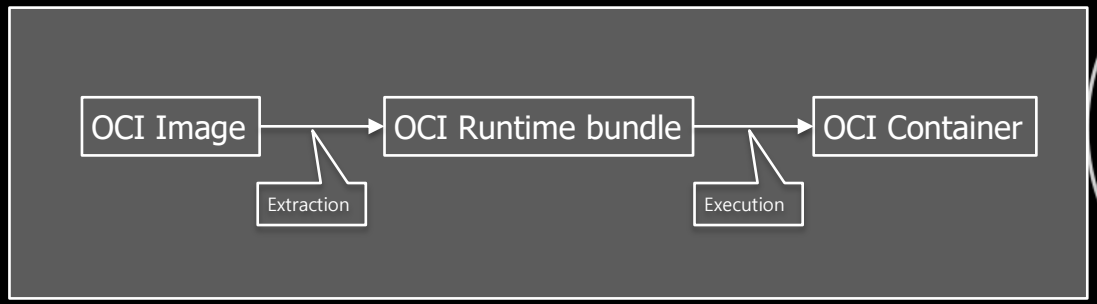
The concept of CRI

source: Kubernetes Blog, Introducing Container Runtime Interface (CRI) in Kubernetes

What is OCI and OCI compliant runtime?

- OCI(Open Container Initiative) offers "**image-spec**" and "**runtime-spec**" as open industry standards.
- Image-spec specifies image format for "**OCI Runtime bundle**" which is set of files.
- Runtime-spec defines the concept of runtime bundle and configuration & lifecycle of a container.
- OCI compliant runtime means runtime which can run **"OCI Runtime bundle"**.

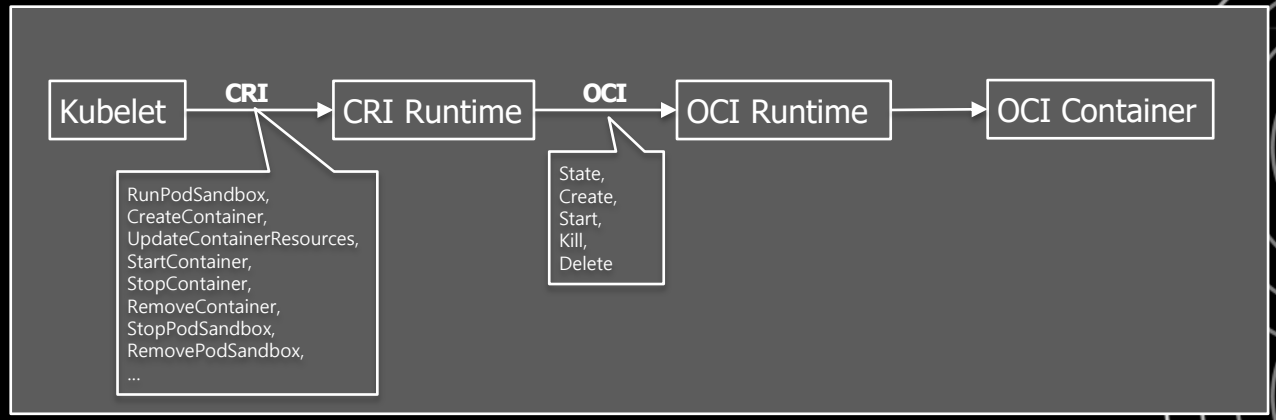
  *(**opencontainers/runc** is known as the iconic OCI runtime and reference implementation.)*



The concept of OCI image spec and runtime spec

source: OCI Runtime Specification

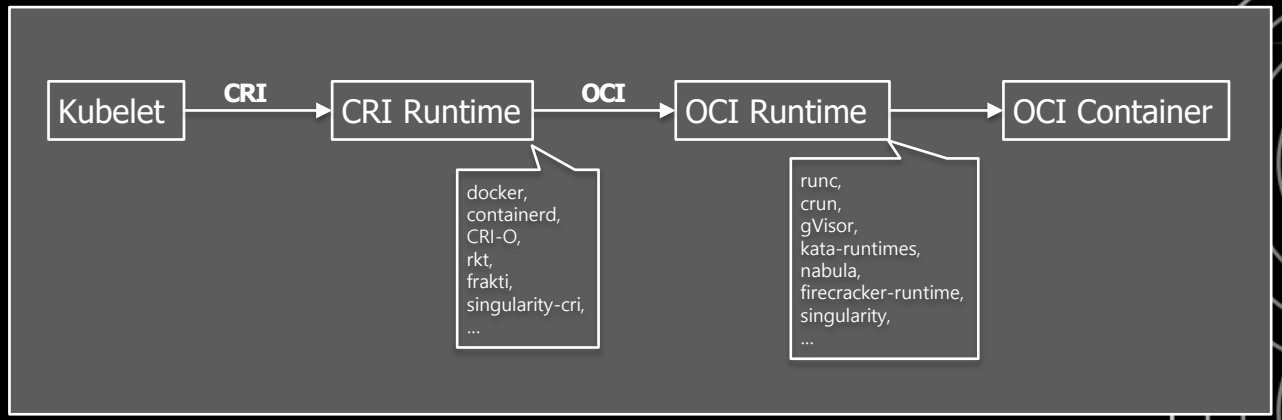
Now we can draw clear picture with CRI and OCI runtime.



Lifecycle related CRI APIs and OCI Runtime event

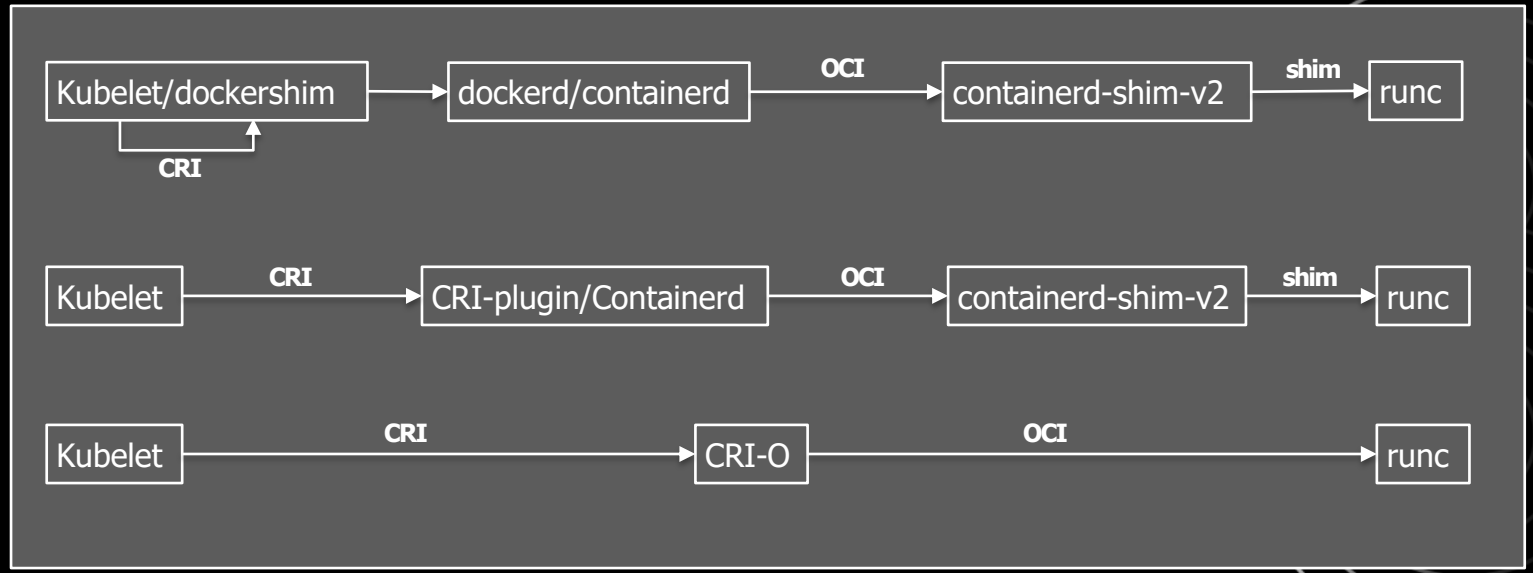
source: Container Runtime Interface, OCI Runtime Specification

Now we can draw clear picture with CRI and OCI runtime.



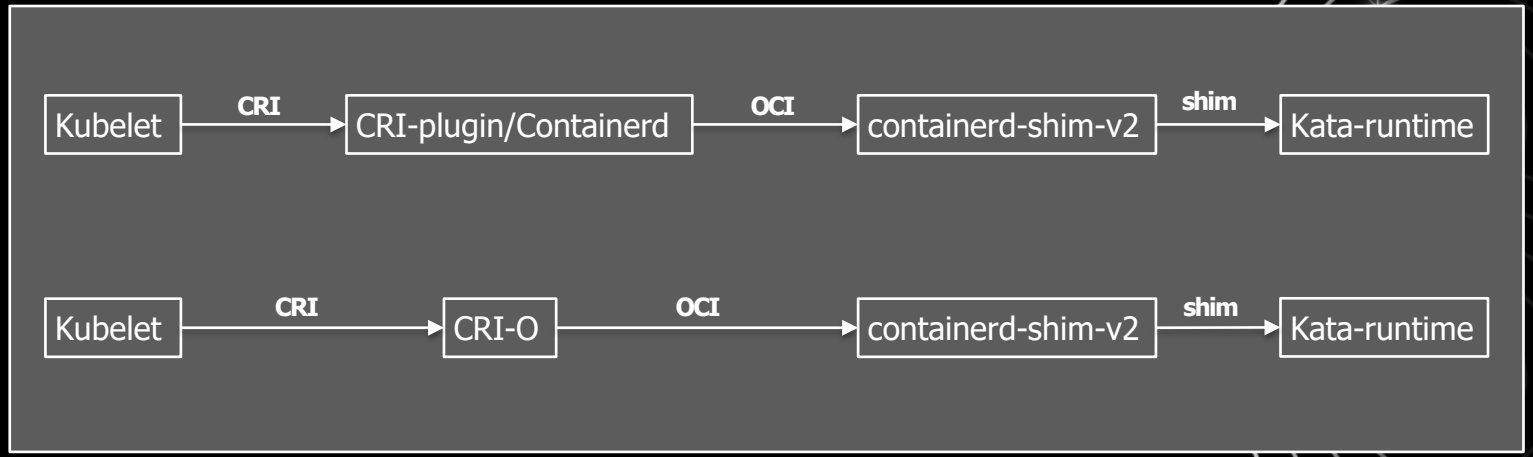
List of CRI and OCI Runtimes

But in the real world, there is a "shim".



3 ways to runc

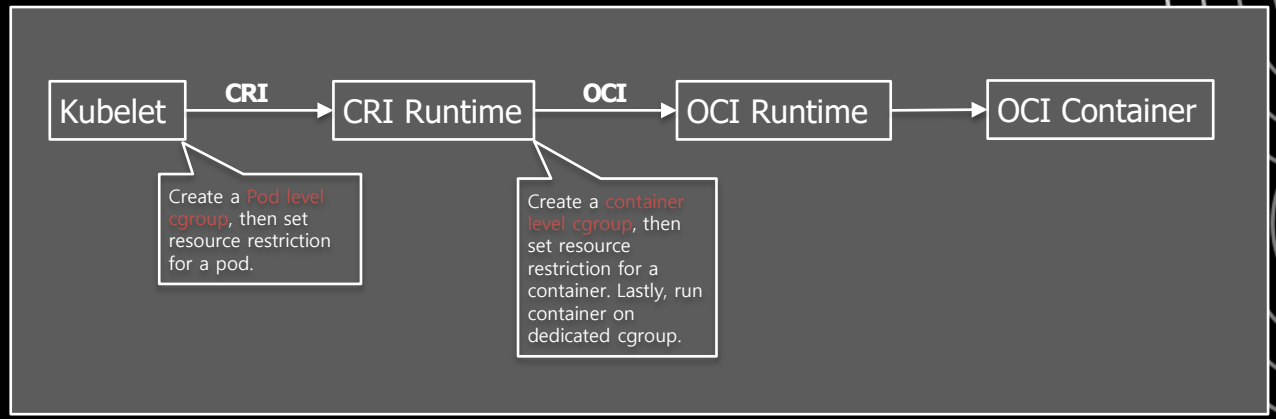
But in the real world, there is a "shim".



2 ways to Kata-runtime

Do we have to know all of this for resource management?

- It is required to know how to manage resources at the low level.
  (to use Node Allocatable Feature, and Resource Managers in Kubernetes like CPU manager.)
- It is required to know to run hardware accelerated application like DPDK with low level resource management.
- In the case of kata-container with KVM/QEMU, the way to manage resources is little bit different.



Sequence of pod and container creation

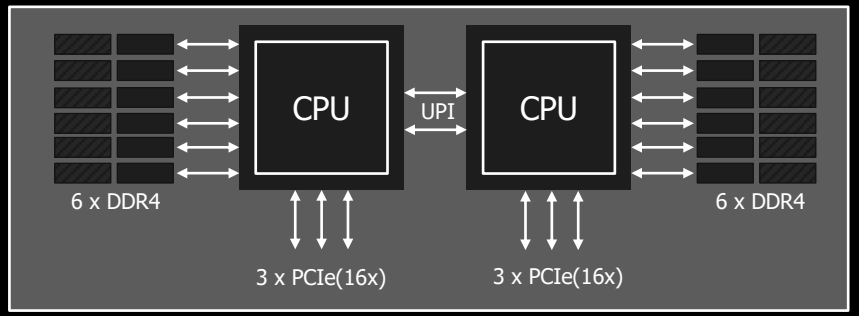
How Kubernetes supports NUMA

What is NUMA?

- NUMA(Non-Uniform Memory Access) is modern style architecture for multi processors.
- Each socket(NUMA node) has own CPU Processor, Memory, PCI Devices.

  (Typically, one socket equal to one NUMA node.)
- Processor is able to access remote memory and I/O devices on other sockets.

  (But the remote access of resources shows performance decrement)

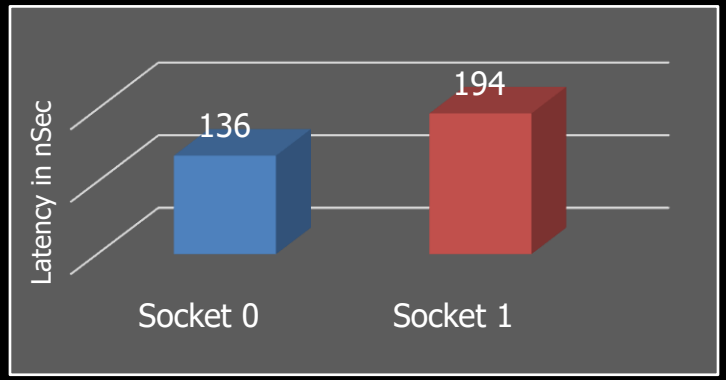


Typical 2 sockets configuration of Intel Xeon

How Kubernetes supports NUMA

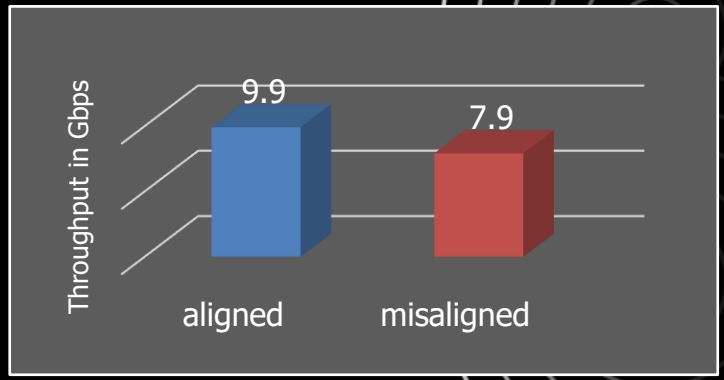
When NUMA aware resource allocation is required?

- NUMA aware resource allocation should be made for following applications.
- Latency-sensitive applications such as real-time AR/VR and game streaming.
- Hardware acceleration based applications such as DPDK and CUDA.



Memory access latency from socket 0

(Intel Xeon E7-4800)



DPDK l2fwd Throughput with 10Gbps NIC

(Intel Xeon Scalable Gold 6148)

Latency test source: Memory Latencies on Intel® Xeon® Processor E5-4600 and E7-4800 product families, Intel

CPU Pinning in Kubernetes

- CPU pinning allows exclusive usage of CPUs for process or thread.
- CPU Manager in Kubernetes responsible for allocating logical threads(SMT) to containers.

  (CPU Manager attempts to allocate sibling threads to containers, when siblings are available.)
- CPU Manager allocates exclusive CPUs using CPUSET cgroup controller.

  (It is possible to adjust container's cpu affinity at thread level by "sched_setaffintiy".)
- Alternative(Intel CMK) also available.

  (Both solutions and NTM are contributed by Intel.)

Solution	Part of Kubelet	Approach	Allowed CPUSET	NUMA Support	Node Allocatable Feature	Node Topology Manager
CPU Manager	Yes	cgroup (CPUSET)	Allocated CPUs only	CPU, I/O Devices, etc over NTM	supported	supported
Intel CMK	No(Plugin)	sched_setaffinity subprocess	Entire CPUs on machine	CPU Only	Not supported	Not supported

Comparison between CPU Manager and Intel CMK

How Kubernetes supports NUMA

How it Works: CPU Manager

```
apiVersion: v1
kind: Pod
metadata:
  name: dpdk-sample
spec:
  containers:
  - image: dpdk-sample
    name: simple-l2fwd
    resources:
      requests:
        cpu: "4"
        memory: "1Gi"
        hugepages-1Gi: "2Gi"
      limits:
        cpu: "4"
        memory: "1Gi"
        hugepages-1Gi: "2Gi"
```

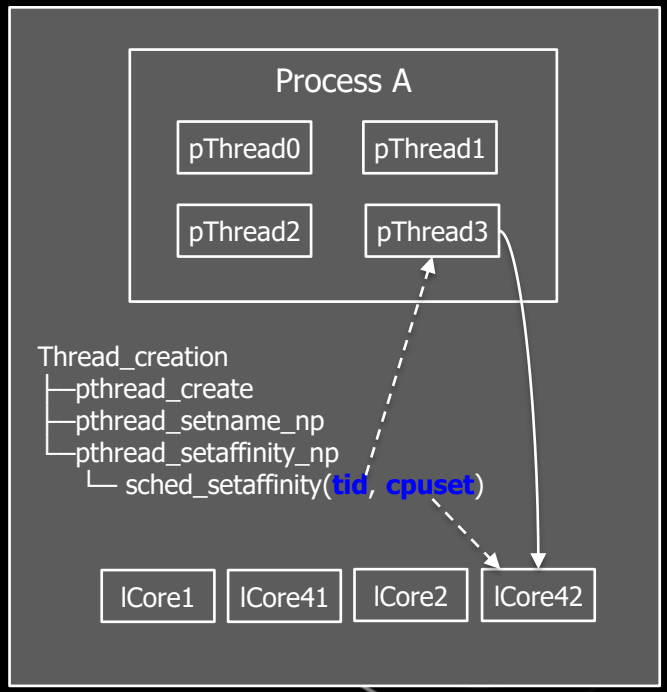
Yaml example for CPU pinning

```
cat <container-cgroup>.cpuset.cpus
1-2,41-42
```

Allocated CPUs for container

	Socket 0	Socket 1
Core 0	[0, 40]	[20, 60]
Core 1	[1, 41]	[21, 61]
Core 2	[2, 42]	[22, 62]
Core 3	[3, 43]	[23, 63]
Core 4	[4, 44]	[24, 64]
Core 5	[5, 45]	[25, 65]
Core 6	[6, 46]	[26, 66]
Core 7	[7, 47]	[27, 67]
Core 8	[8, 48]	[28, 68]
…		

CPU Layout(2socket, SMT enabled)



"sched_setaffinity" usage in DPDK

(pin pThread3 to lCore42)

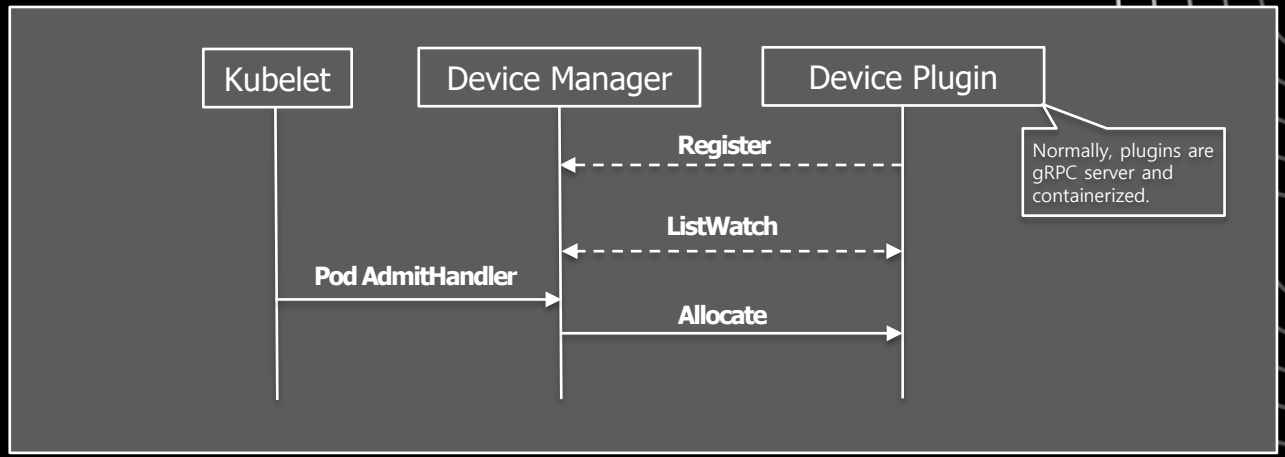
Resource Manager and Plugins in Kubernetes

- Device Manager

  (Component of Kubelet, advertises/allocates extended resources.)

- Device Plugins

  (nvidia-gpu-plugin, amd-gpu-plugin, gpu-sharing-plugin, sr-iov-plugin, rdma-device-plugin, etc)



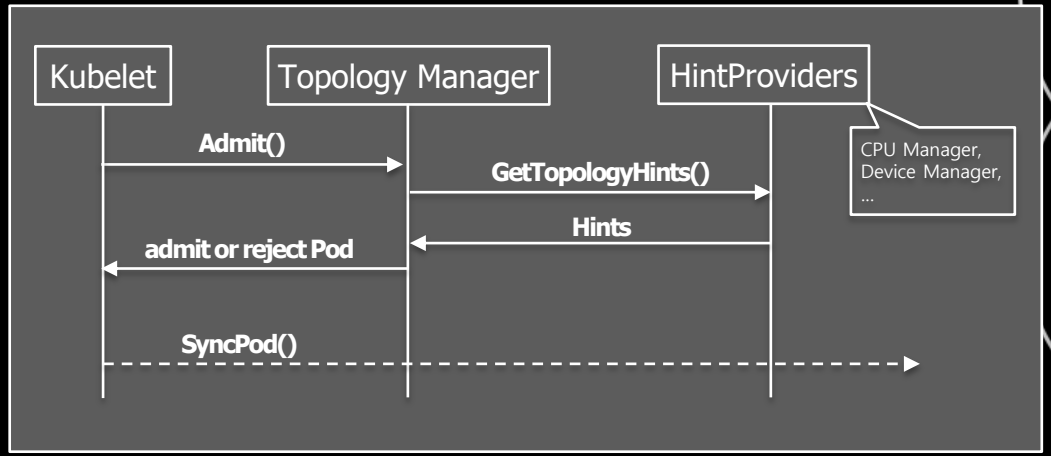
Sequence of extended resource allocation in Kubernetes

How Kubernetes supports NUMA

The concept of Topology Manager

- Topology Manager provides the way of NUMA-aware resource allocation for containers at the node level.
- Topology Manager retrieves Topology Hint from Hint Providers
- Topology Manager calculates NUMA node affinity then judges whether admit pod or not by given policy.

  (pod admission will be rejected, if chosen policy cannot be satisfied.)



Sequence of Pod admission with Topology Manager

How Kubernetes supports NUMA

What is Topology Hint and Topology Policy?

- Topology Hint is data structure to represent NUMA nodes of allocable resources as bits.
- Topology Manager collects hints then merges the hints to find best hint.

  (Policies share the same merging algorithm in 1.16, each policy will have own one in future release)
- Each policy has own pod admission criteria.

```
//TopologyHint is a struct containing the NUMANodeAffinity for a Container
type TopologyHint struct {
        NUMANodeAffinity bitmask.BitMask
        // Preferred is set to true when the NUMANodeAffinity encodes a preferred
        // allocation for the Container. It is set to false otherwise.
        Preferred bool
}
```

Topology Hint Structure

Policy	Description
none	Do nothing, Topology Manager will not working.
best-effort	Calculate best hint then just use it whatever it is
restricted*	Reject pod admission if best hint is not preferred hint
single-numa*	Reject pod admission if best hint does not fit to single NUMA node

Topology Policies

How Kubernetes supports NUMA

How it Works: Topology Manager (w/single-numa policy)

```
apiVersion: v1
kind: Pod
metadata:
  name: ntm-sample
spec:
  containers:
  - image: simple-sample
    name: simple-sample
    resources:
      requests:
        cpu: "4"
        memory: "1Gi"
        nvidia.com/gpu: 1
      limits:
        cpu: "4"
        memory: "1Gi"
        nvidia.com/gpu: 1
```

Yaml example for NTM

Test Case	Resource availability at scheduler level	Available Resources on Socket 0	Available Resources on Socket 1	Expected Result
Positive Case 1	CPU: 20, GPU: 4	CPU: 10, GPU: 2	CPU: 10, GPU: 2	Socket0, *Socket1*
Positive Case 2	CPU: 20, GPU: 2	CPU: 10, GPU: 2	CPU: 10, GPU: 0	Socket0
Positive Case 3	CPU: 7, GPU: 3	CPU: 3, GPU: 2	CPU: 4, GPU: 1	Socket1
Negative Case 1	CPU: 13, GPU: 2	CPU: 3 , GPU 2:	CPU: 10, GPU: 0	Admit Rejected
Negative Case 2	CPU: 6, GPU: 4	CPU: 3 , GPU 2:	CPU: 3 , GPU 2:	Admit Rejected

PodAdmit TestCase

How Kubernetes supports NUMA

Issues(in 1.16)

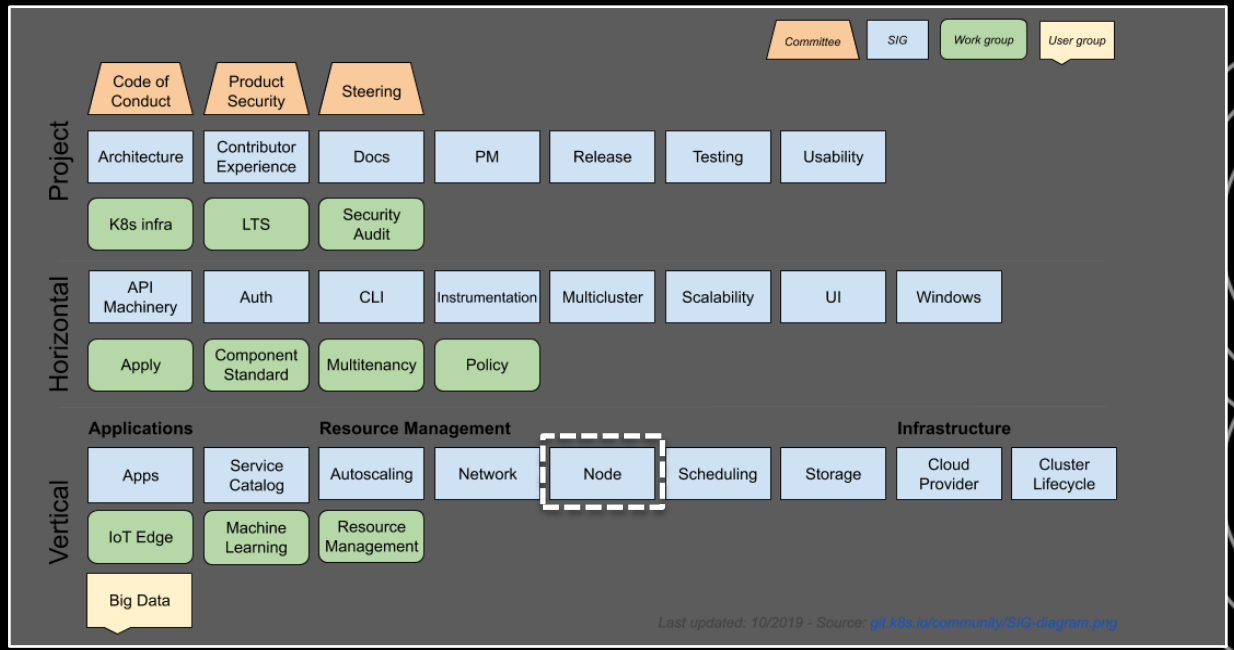
Issue	Description
Kubernetes/Issues/#83476*	Unreliable Topology Hint generation when multiple containers in the same pod require alignment.
Kubernetes/PR/#83697	Topology Manager wouldn't allow pod admit with single-numa policy when any of hint providers had no NUMA preferences. *(Merged)*
Kubernetes/PR/#83492	Topology Manager supports only guaranteed QoS class. *(Merged)*
Kubernetes/Issue/#83483	To support "inter-device" topology contstraints(i.e. GPU-direct, Nvlink, RDMA)
Kubernetes/Issues/#83478	Same affinity calculation algorithm for various policies. *(Refactoring has been already started.)*
TBD	Alignment is limited at the container level, Topology Manager doesn't support Pod level alignment.

Helpful Links

Title	Link
Cgroup	https://www.kernel.org/doc/Documentation/cgroup-v1/cgroups.txt
CPU Manager KEP	https://github.com/kubernetes/community/blob/master/contributors/design-proposals/node/cpu-manager.md
Device Manager KEP	https://github.com/kubernetes/community/blob/master/contributors/design-proposals/resource-management/device-plugin.md
Topology Manager KEP	https://github.com/kubernetes/enhancements/blob/master/keps/sig-node/0035-20190130-topology-manager.md
CPU Manager Guide	https://kubernetes.io/docs/tasks/administer-cluster/cpu-management-policies/
Topology Manager Guide	https://kubernetes.io/docs/tasks/administer-cluster/topology-manager/
Kubelet (Container Manager)	https://github.com/kubernetes/kubernetes/tree/master/pkg/kubelet/cm

Kubernetes Contribution

Special Interest Groups(SIGs) are open to new contributors



source: https://github.com/kubernetes/community

Hugepages Enhancement

But…What is hugepages?

- Hugepages are literally page which has huge size, typical Linux machine supports two page sizes(2MB, 1GB).
  (Default page size is 4kb)
- The concept of hugepages is reducing TLB miss to reduce memory access latency.
  (Hugepages also allow high utilization of hardware cache by reducing PageTable Entries.)
- DPDK and Database are usually known as applications which consumes hugepages.
  (DPDK is Data Plane Development Kit for packet processing.)
- Kubernetes supports to consume pre-allocated hugepages but it does not support NUMA and container isolation of hugepages.

Hugepages Enhancement

What is the goal of hugepages enhancement?

- Support container isolation of hugepages
- Support multi size hugepages at host and container level.
- Support NUMA aware hugepages management.

THANK YOU

SOSCON 2019

SAMSUNG OPEN SOURCE CONFERENCE 2019