
KubeCon
CloudNativeCon
Europe 2023
TiKV

Experience With “Hard Multi-Tenancy” in Kubernetes Using Kata Container

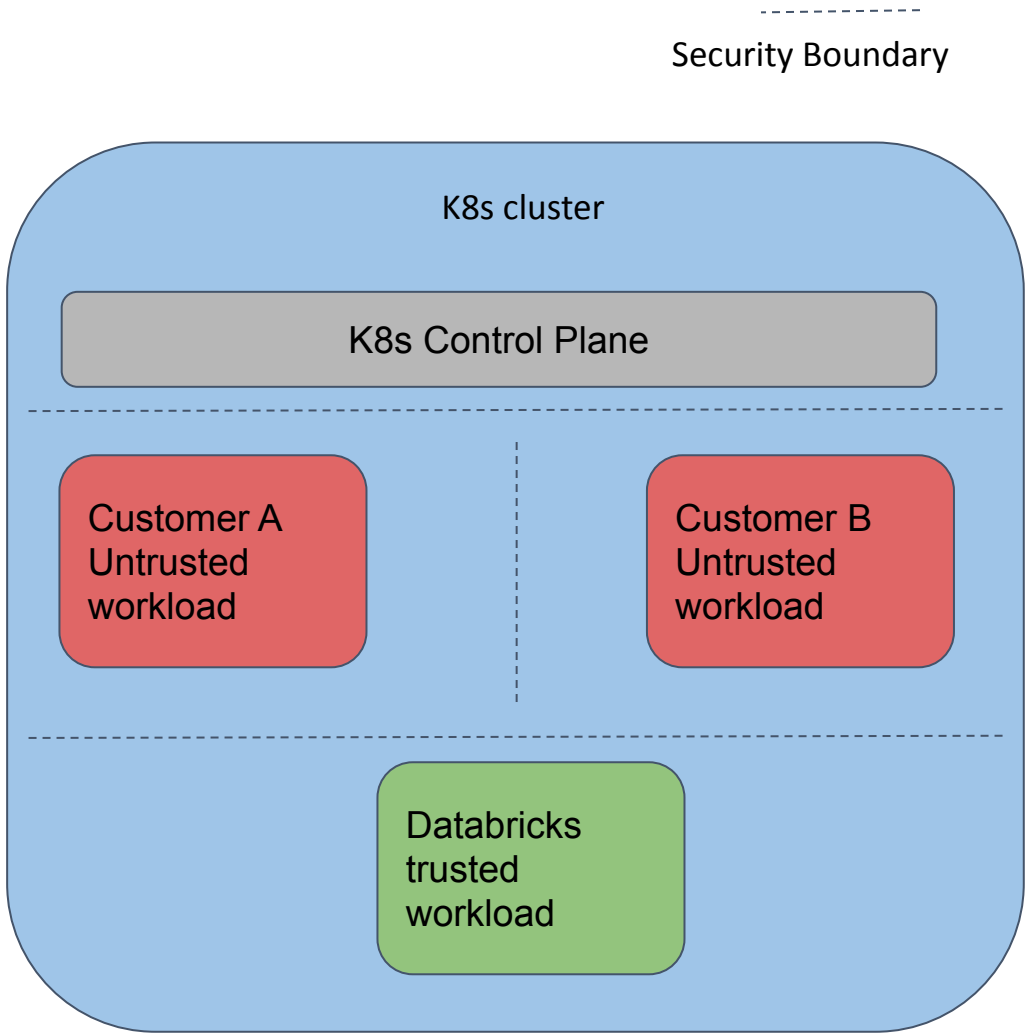
Shuo Chen @Databricks

Introduction

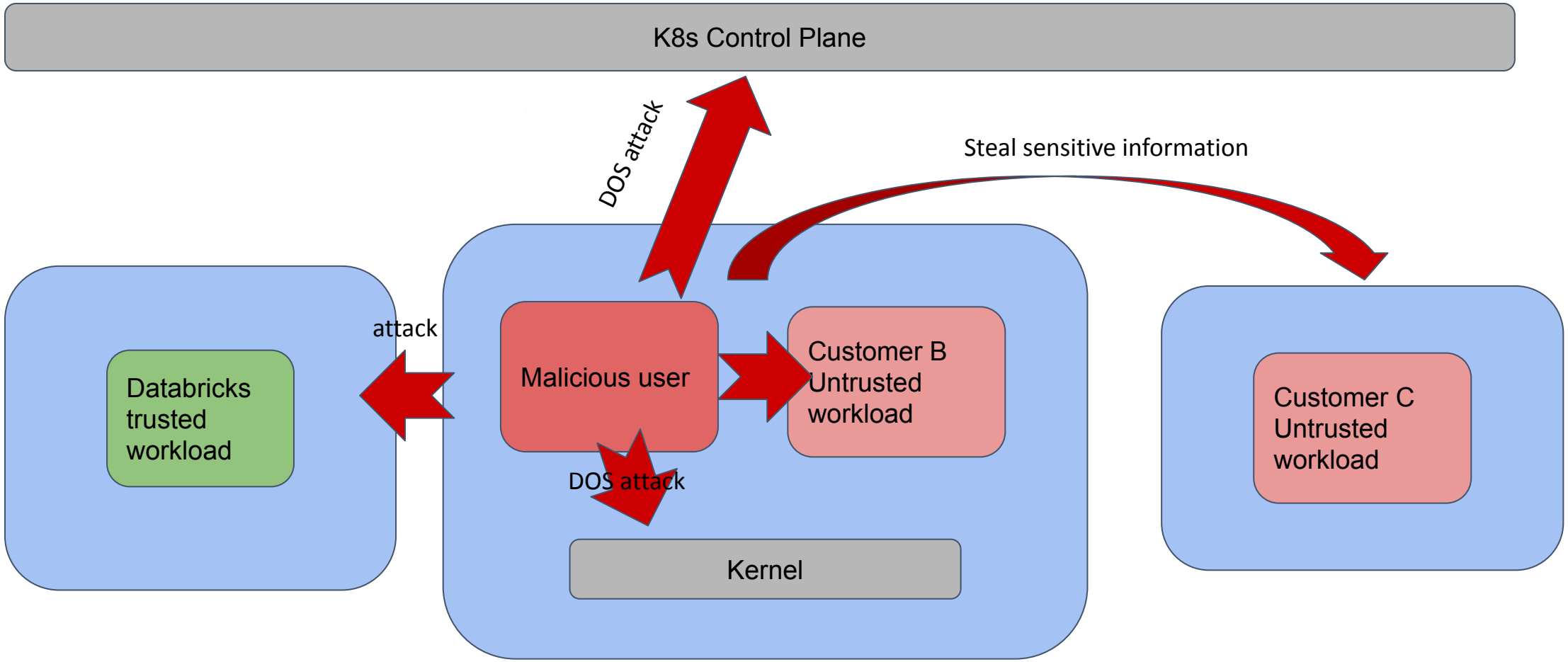
- **What is Databricks**
  - Unified & open data and analytics platform
  - Lakehouse
  - Data Engineering, machine learning, AI
  - Multi-cloud platform
- **Classic Databricks Infrastructure Model**
  - Integrates with cloud storage and security in customers' cloud account
  - Manage and deploy cloud resources in customers' account
- **New Infra Model: Serverless**
  - Run compute resources in Databricks' account
  - Eliminate management overhead
  - Instant and elastic
  - Lower infrastructure costs

Databricks Serverless Multi-tenant Environment

- **We want to use Kubernetes as our infrastructure**
  - Great portability, extensibility for containerized workload
  - Cloud agnostic
  - Rapid growing ecosystem
- **Run trusted and untrusted services at same cluster**
  - Customers' workloads are containerized into untrusted workloads
  - Databricks intra cluster control services from the same k8s cluster are trusted workloads
- **Hard multi-tenancy**
  - Tenants do not trust each other
  - Infrastructure do not trust tenant
  - Isolation of tenants within data plane & control plane are critical



What a Malicious User Can Do



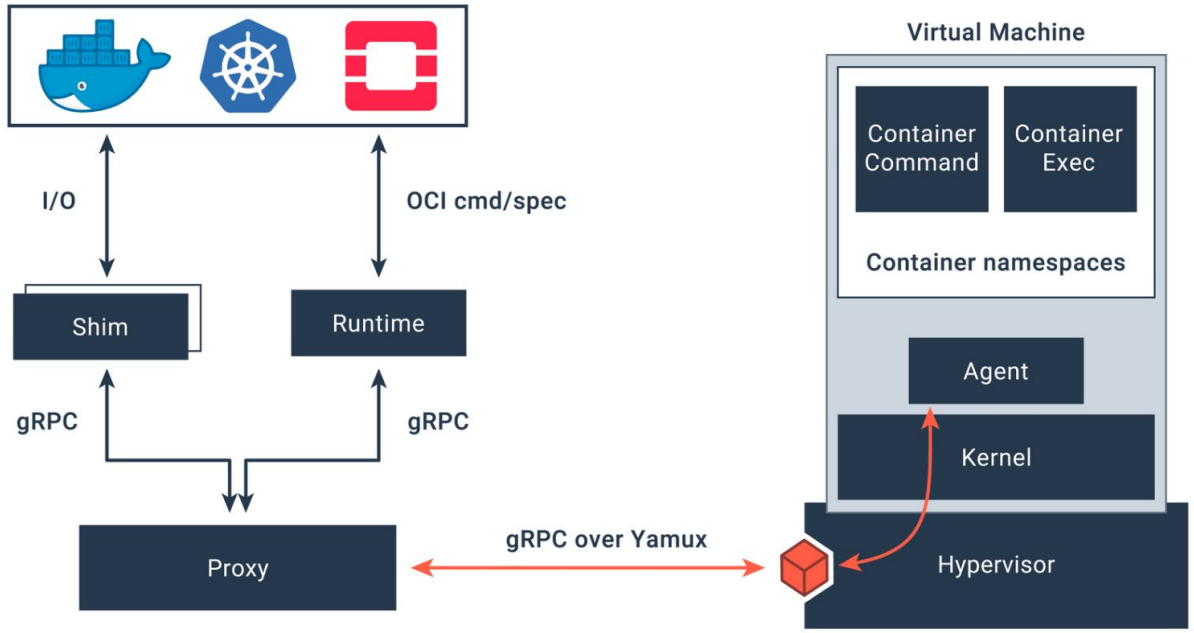
Exploration of the hard multi-tenancy solution

Kata Containers

Kata Containers is a secure container runtime with lightweight virtual machines that feel and perform like containers, but provides stronger workload isolation using hardware virtualization technology as a second layer of defense

Security Advantages:

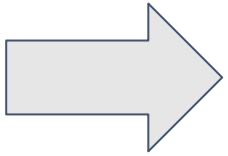
- VM boundary instead of container boundary



However, kata container requires virtualization capabilities from the host

How to Ensure the Security?

- **Provide hard boundary for customer's container**
  - Cpu & memory is fully isolated
  - Dedicated disks & file systems
  - Dedicated kernel
  - Hard to breakout

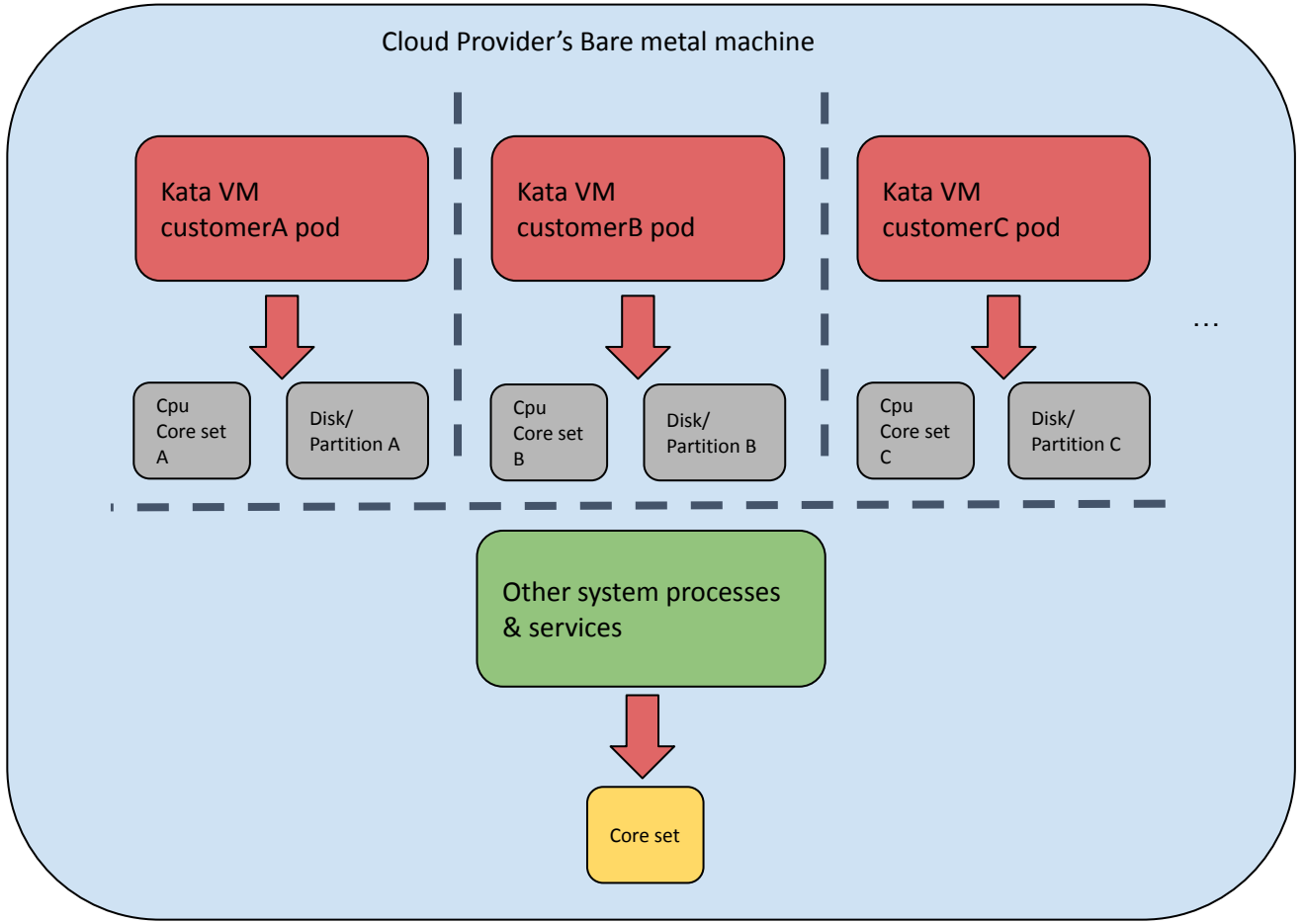


+

- **Secure network access control mechanism**
  - Container network policy enforcement
  - Prevent tenants from communicating with each other
  - Cloud provider's native firewall solution (network security group)

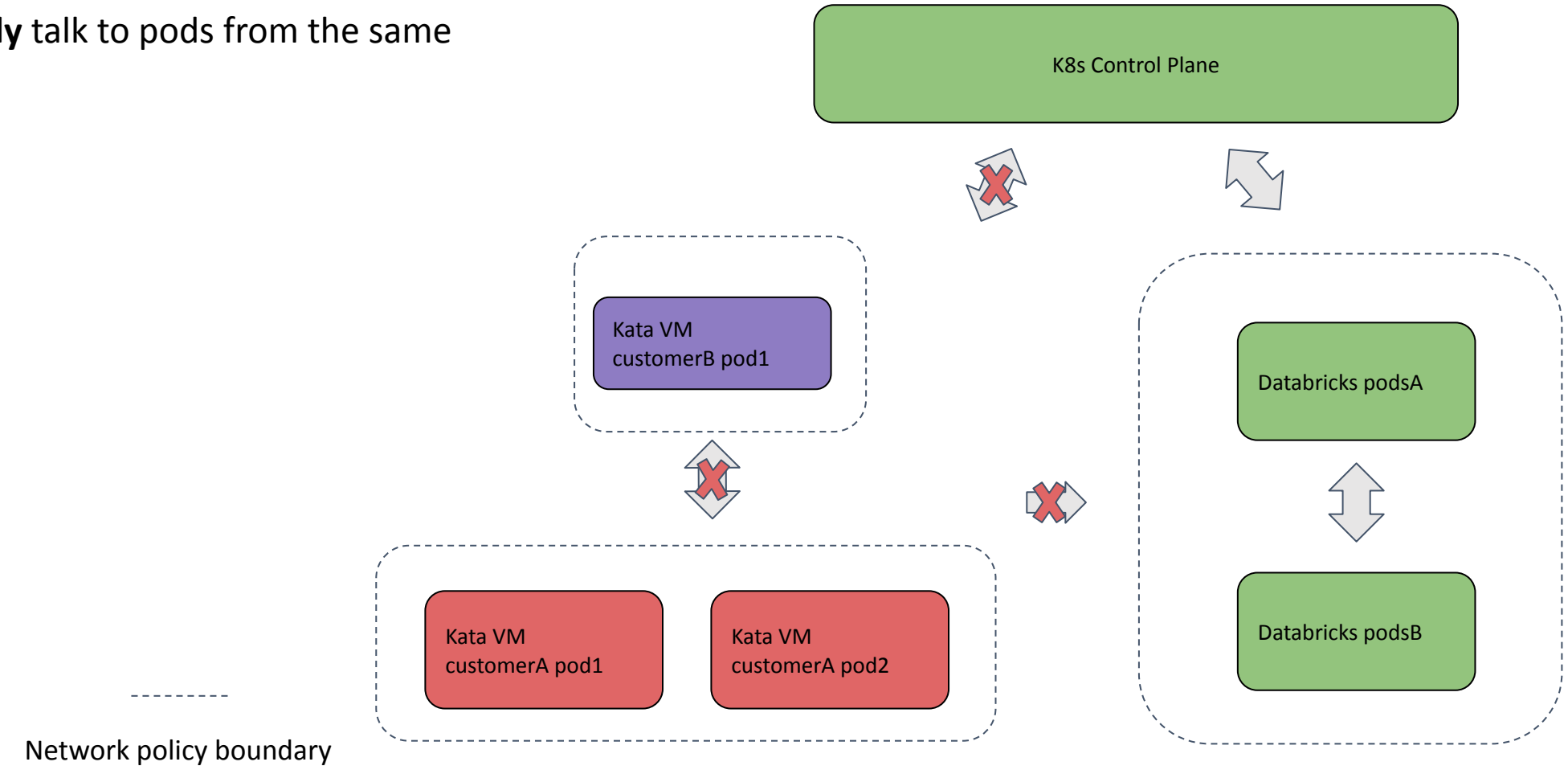
Kata Container Integration - Single Node View

- Use large machines to hold multiple kata VMs from multiple customers
- Each kata VM has its own dedicated physical resources including CPU cores, memory, disk(partitions), virtual network devices and container rootfs
- Reserve cores and memories for system services such as kubelet, containerd, etc.



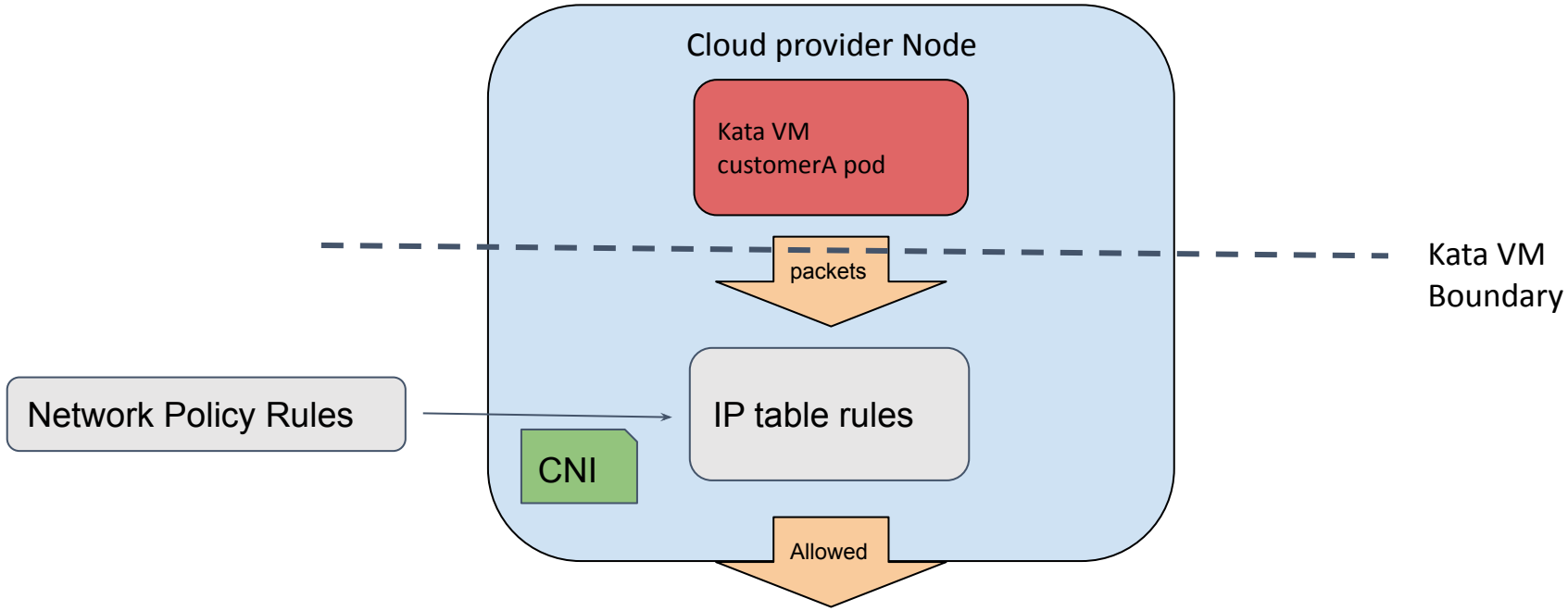
Network Security - Network Policy

- A pod can **only** talk to pods from the same customer.



Network Security - Network Policy

Kata container makes network policy more secure for multi-tenancy environment by isolating the host IP table per tenant



Onboarding Kata is Simple


```
priority: 100000
priorityClassName: dbr-executor-v1
restartPolicy: Always
runtimeClassName: kata-qemu
```

```
[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.kata-clh.options]
  ConfigPath = "/opt/kata/share/defaults/kata-containers/configuration-clh.toml"
[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.kata-qemu
  snapshotter = "overlaybd"
  runtime_type = "io.containerd.kata-qemu.v2"
  privileged_without_host_devices = true
  pod_annotations = ["io.katacontainers.*"]
  container_annotations = ["io.katacontainers.*"]
  [plugins."io.containerd.grpc.v1.cri".containerd.runtimes.kata-qemu-overlaybd.options]
    ConfigPath = "/opt/kata/share/defaults/kata-containers/configuration-qemu.toml"
```

Is That Good Enough for Productionisation?

NO

Performance

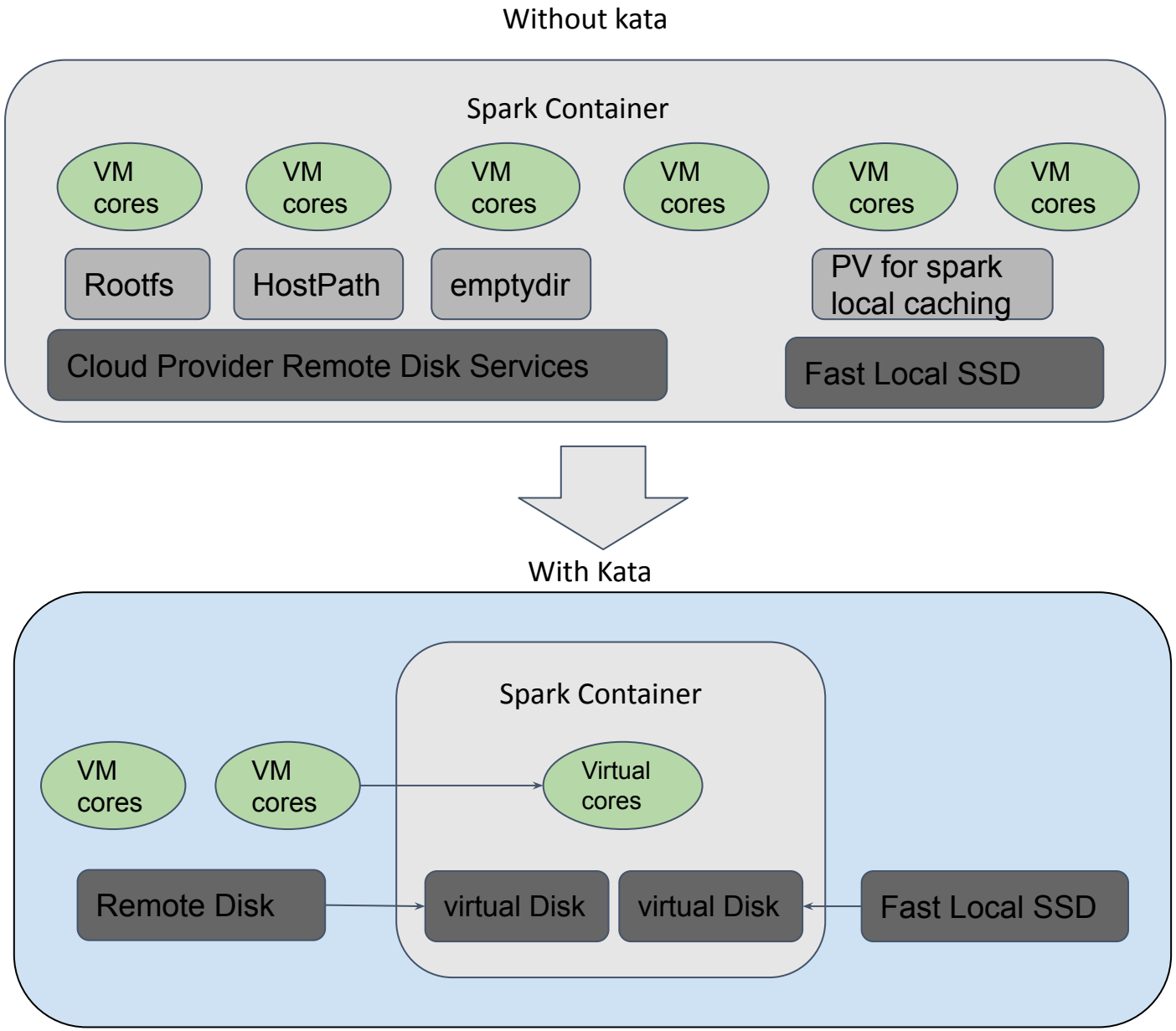
With vanilla kata, our workload has 3x ~ 6x slowdown

Kata Container Integration - Performance Problem

Spark as one of our workload

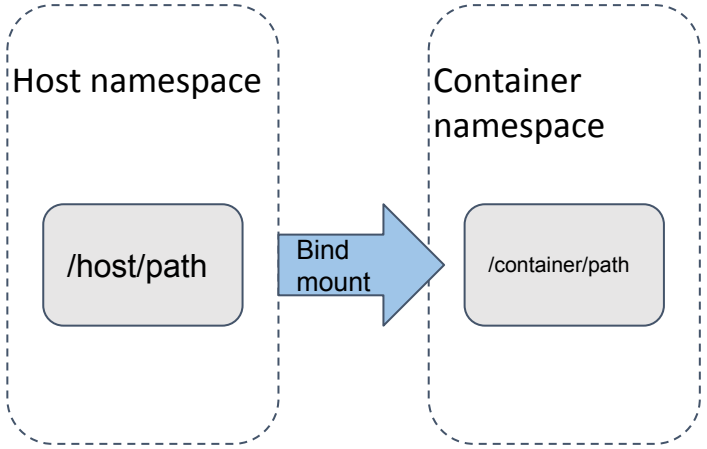
- Compute intensive
- Memory intensive
- Disk/Network IO Intensive

- VM_EXIT
- VCPU scheduling
- The virtualization layer add additional cost for memory access
- The virt-io protocol add additional latency and throughput overhead for virtual disks

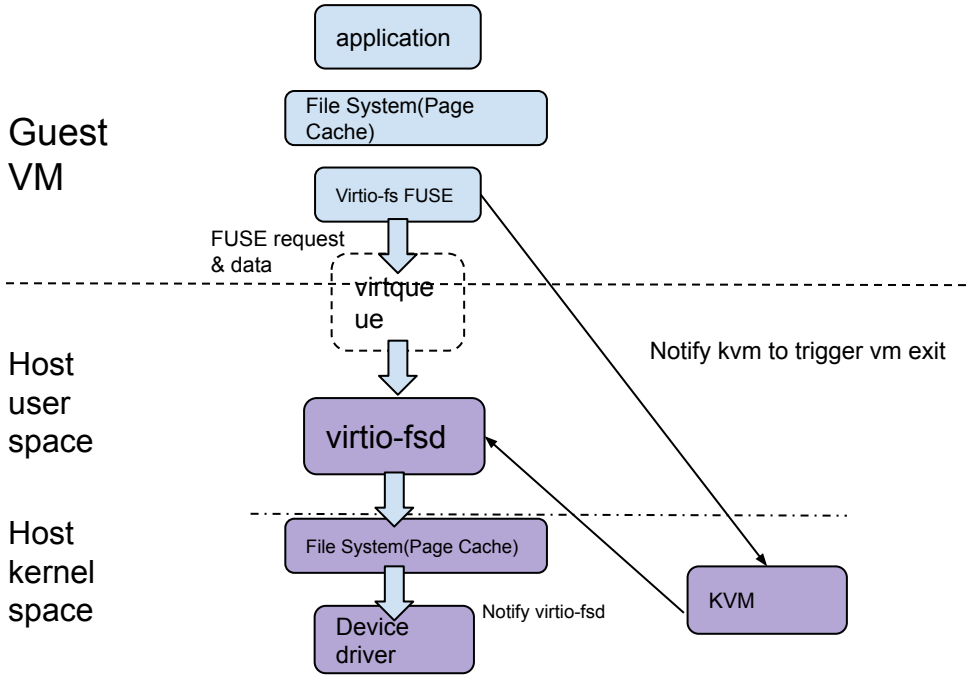


Storage Performance: Default PV Support by Virtiofs

- Spark requires a fast disk for IO intensive workloads, which is supported by a local SSD
- PV/PVC is used to announce this space inside our pod spec
- Vanilla storage support for PV/PVC is through bind mount a shared file system



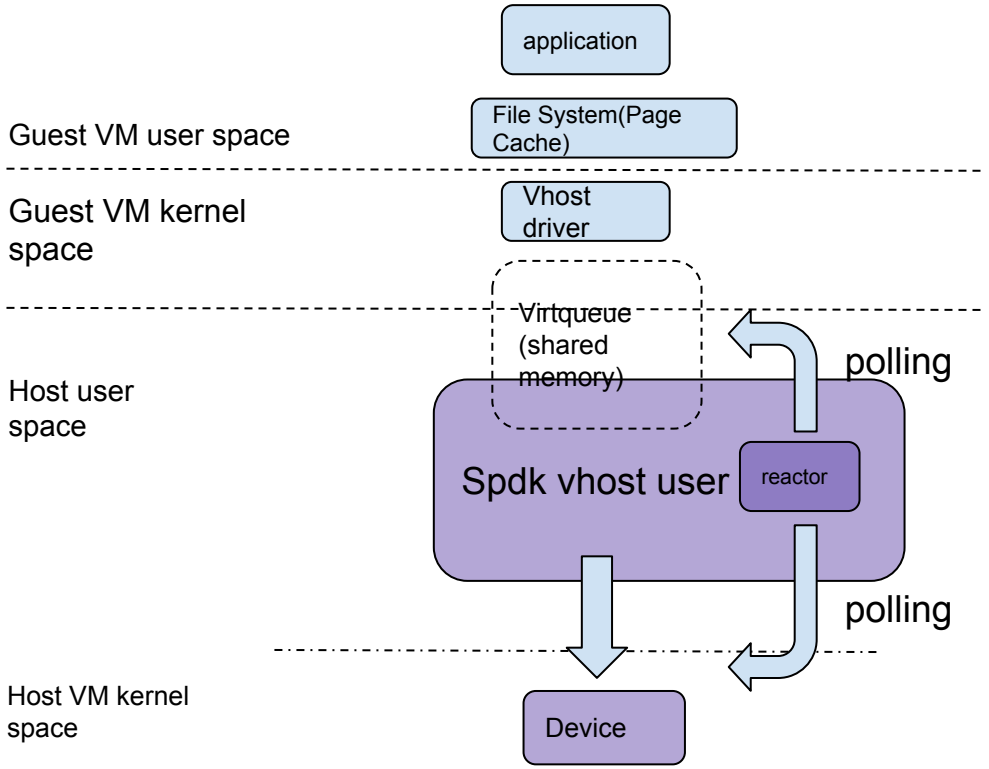
- Virtio-fs is used to support the shared file system between host and kata VM
- Performance is poor due to the frequent context switch between user/kernel spaces



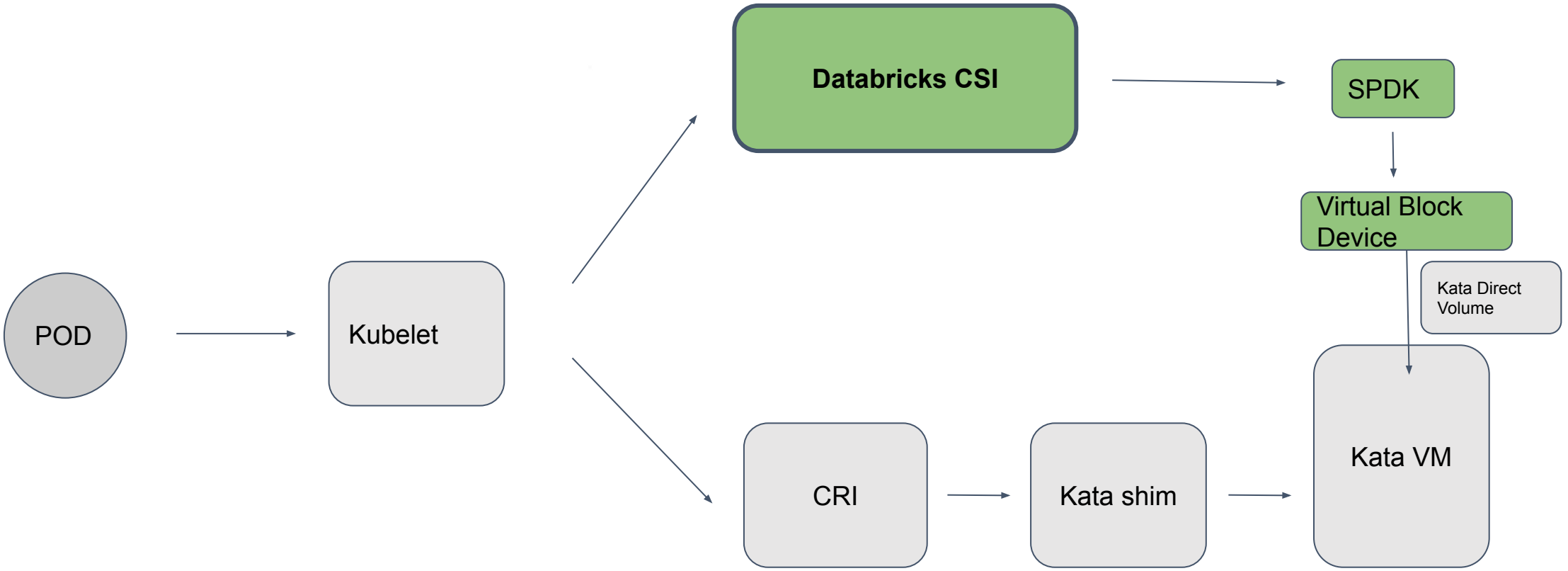
Storage Performance: SPDK + Kata Direct Volume

- What is SPDK?
  - The storage performance development Kit
  - Provide a set of tools and libs for writing high performance, scalable, user-mode storage application

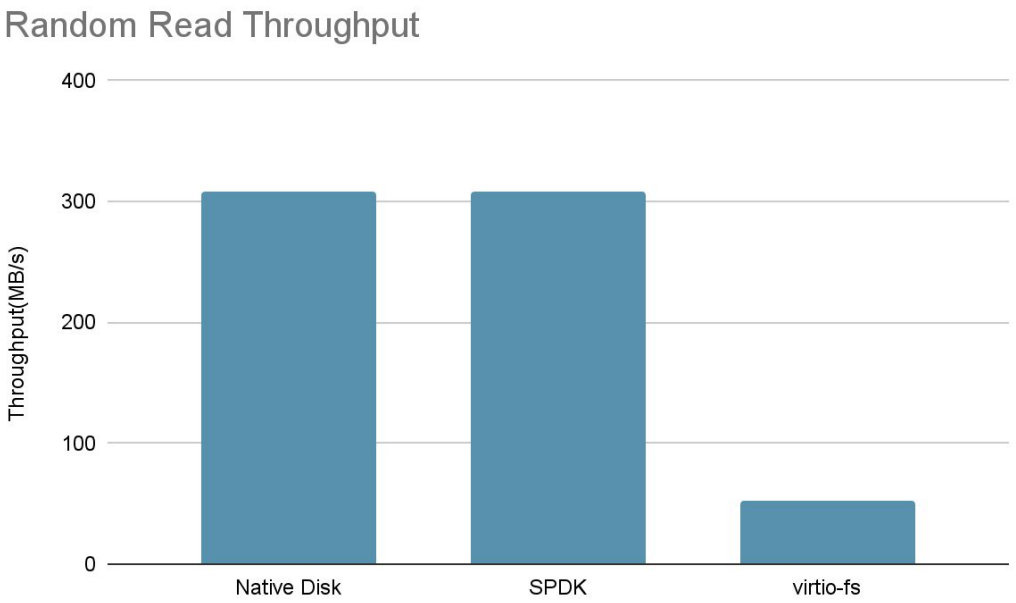
- Simplify the IO path:
  - Kata Direct Volume Project pass a block device into kata guest VM
  - SPDK introduces polling mode instead of interrupt mode to shorten latency
  - Avoid jumping between kernel and host namespace

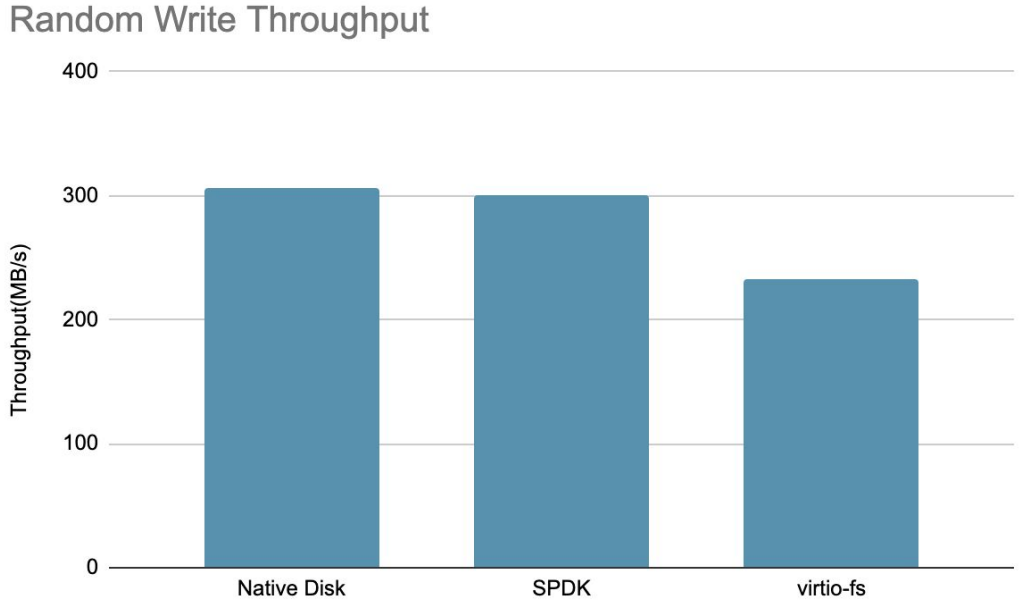


Storage Solution: Implementing Our Own CSI



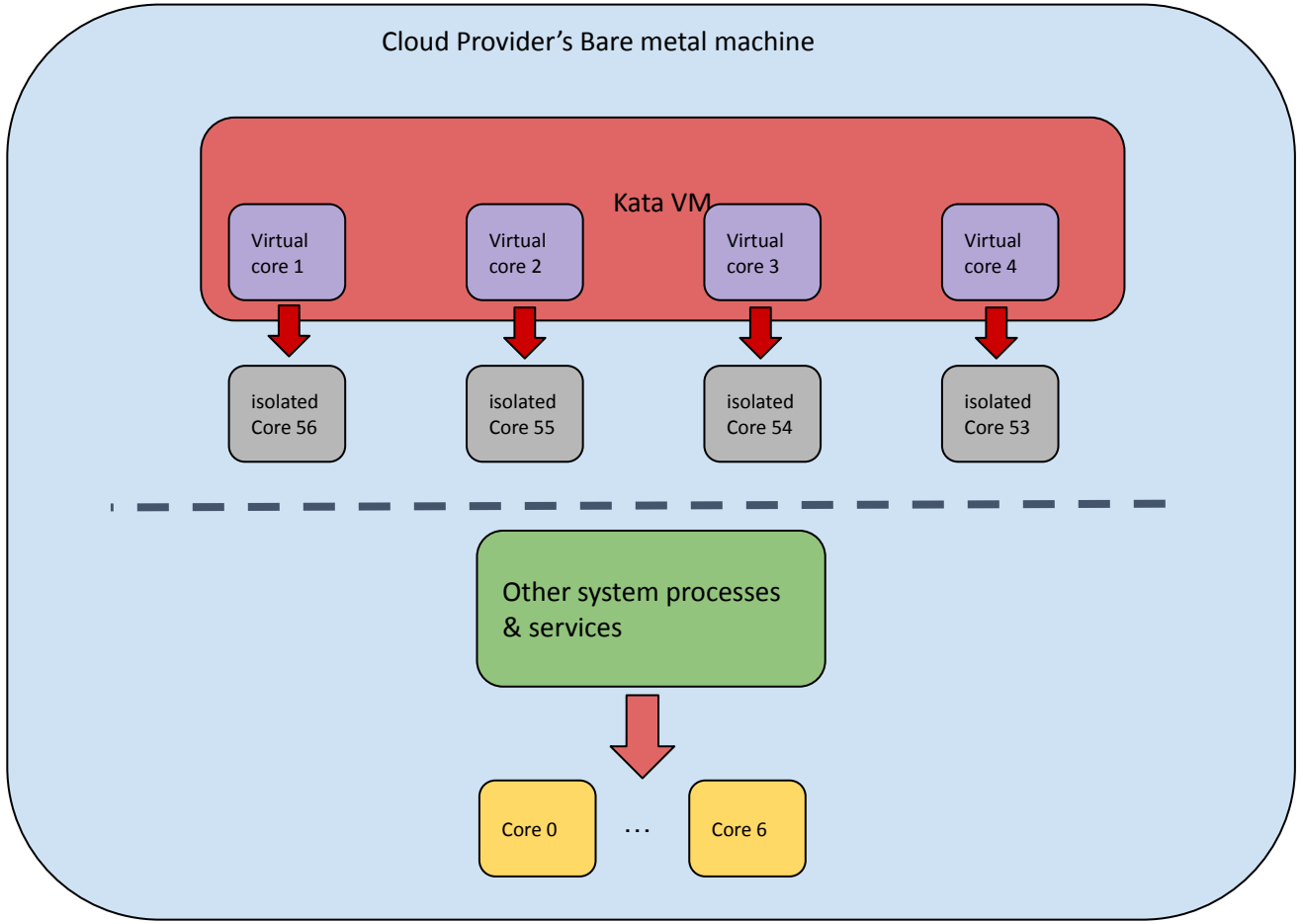
Disk Performance: SPDK vs virtio-fs





Compute Performance: Host CPU Tuning

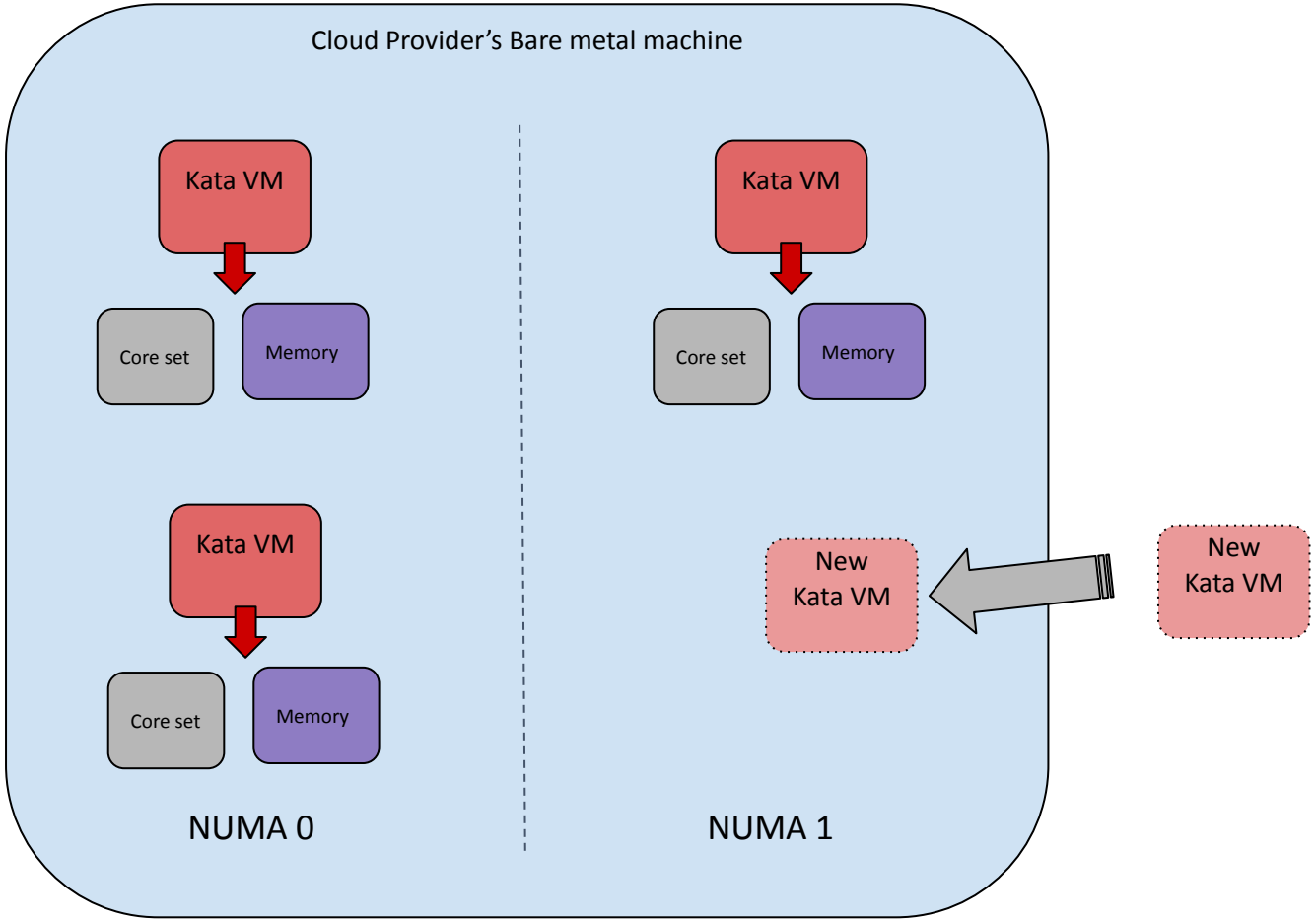
- Cpu Isolation
  - Isolate CPU cores from linux scheduler
  - Prevent linux scheduler to assign other host processes and interrupts to the customer's CPU
- Cpu pinning
  - Pin each kata vm's virtual CPU to a dedicated core
  - Prevent virtual CPU threads jumping between cores
- CPU State tuning
  - Enable CPU Performance mode
  - Tune CPU power management options for lower latency



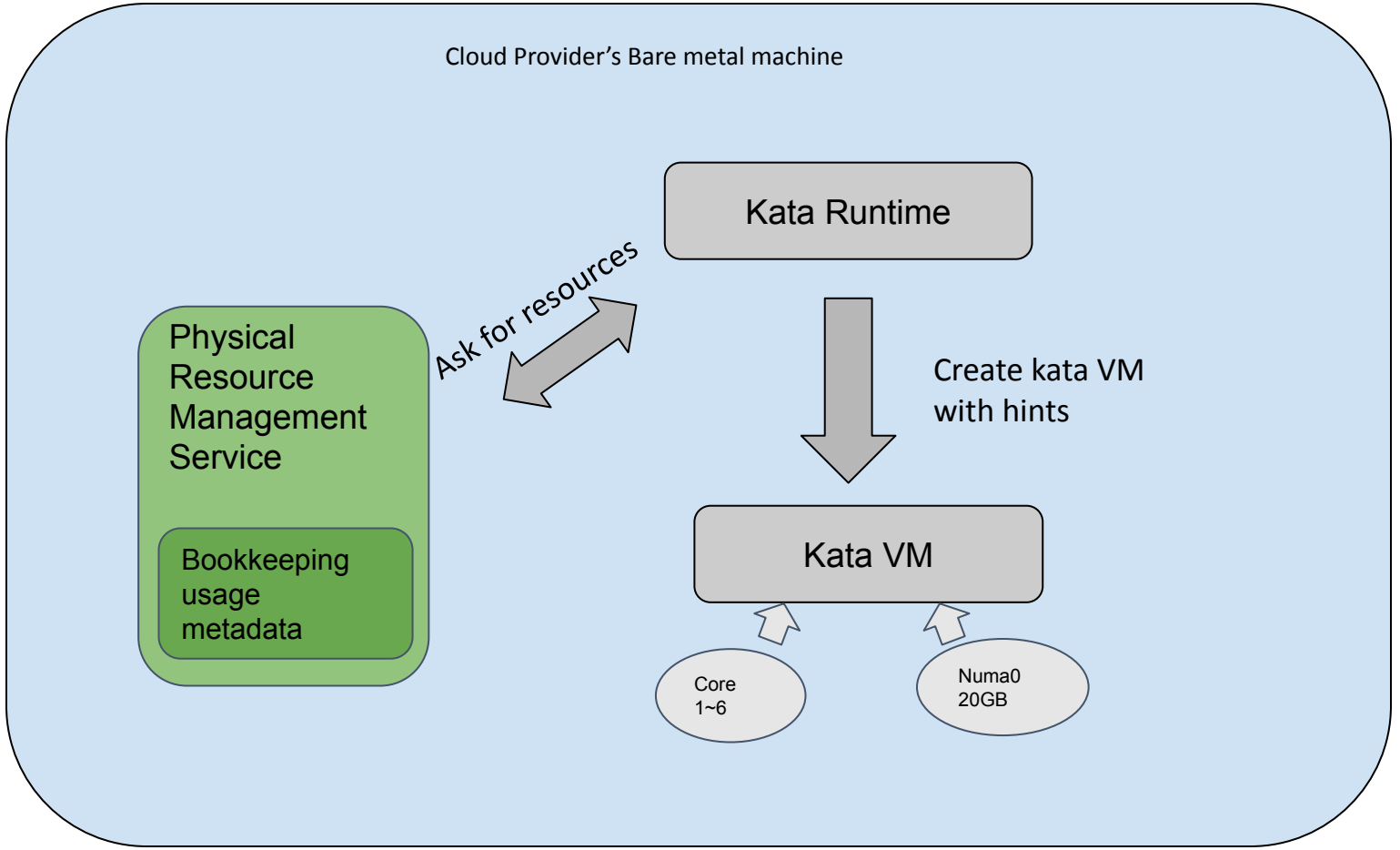
Compute Performance: Numa Control

- What is NUMA?
  - Non-uniform memory access
  - Different cores access different memory slot will have different performance

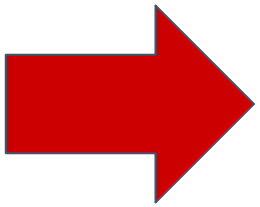
- Prevent a single kata VM using cross NUMA memories
- Auto balancing the load for different numa nodes on the same host



Compute Performance: Numa Control

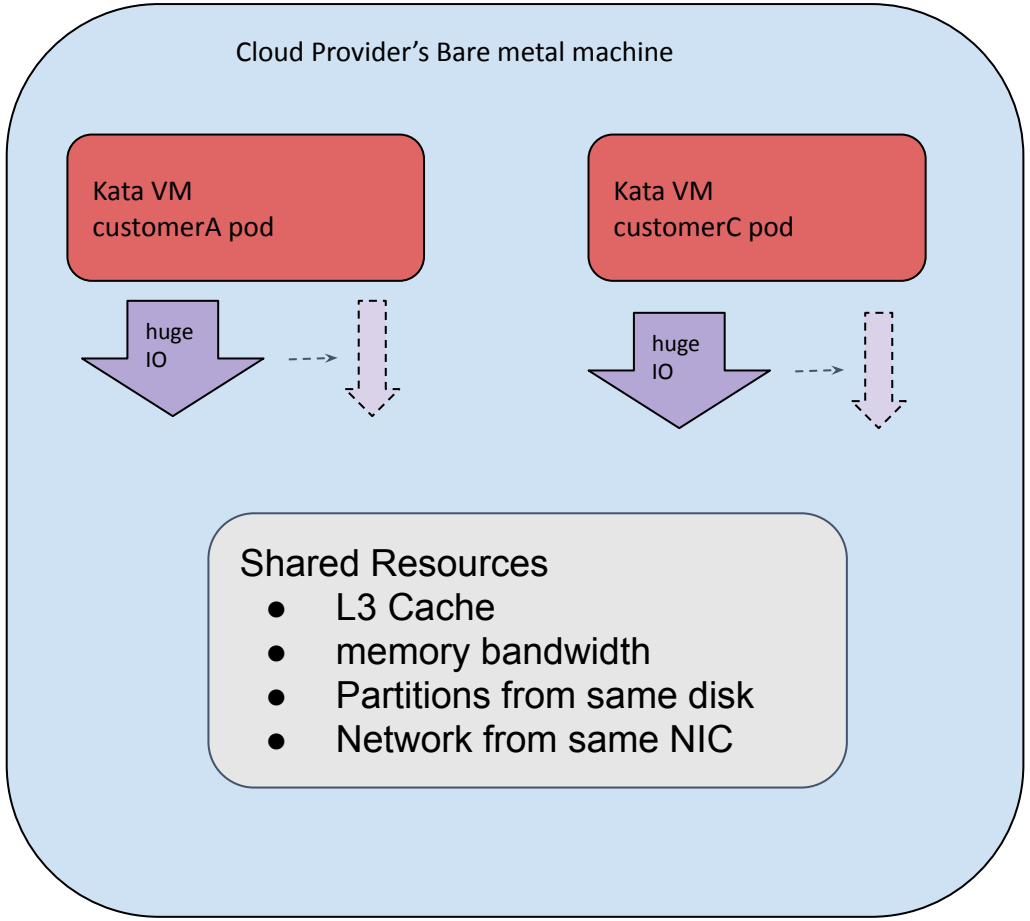


Performance Result

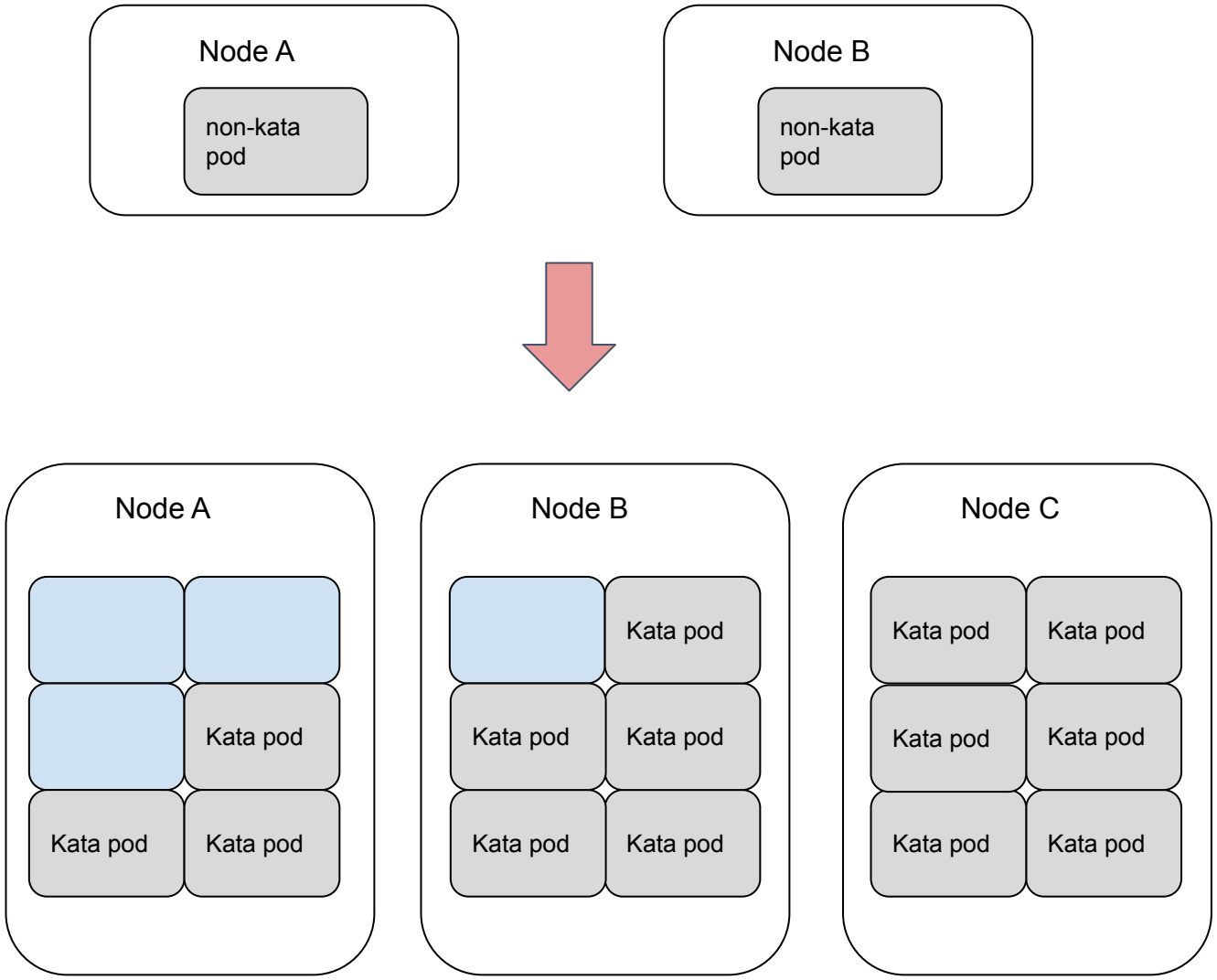
3x ~ 6x slow down  < 5% slow down

Kata can be one of the option to support our infrastructure!!

Potential Risks - Noisy Neighbor



Potential Risks - Resource Fragmentation



Other Problems

- Instance type capacity might not large enough
- Allocate additional resource in the node to cover the virtualization cost
- Some products scenario is not well supported such as passing GPU into a kata VM, etc.

Conclusion

- Kata container is a great project to support hard multi-tenancy in kubernetes
- By fine tuning the performance, Kata container can reach similar performance level to native container technology
- Kata container brings its own complexity to the infrastructure management

Databricks Serverless SQL is now in public preview for both AWS and Azure
welcome to shoot a try
https://docs.databricks.com/serverless-compute/index.html

Questions

Useful links:

- Kata container - https://katacontainers.io/
- Network policy - https://kubernetes.io/docs/concepts/services-networking/network-policies/#prerequisites
- Virtiofs - https://virtio-fs.gitlab.io/
- SPDK - https://spdk.io/
- NUMA - https://en.wikipedia.org/wiki/Non-uniform_memory_access
- Container storage interface - https://github.com/container-storage-interface/spec/blob/master/spec.md
- Databricks: https://www.databricks.com/

KubeCon CloudNativeCon Europe 2023

KubeCon CloudNativeCon North America 2021

Please scan the QR Code above
to leave feedback on this session