@JeffryMolanus
@openEBS
www.openebs.io

FOSDEM Feb 2018

OpenEBS

Containerised Storage for Containers

What if storage for container native applications was itself container native?

OpenEBS

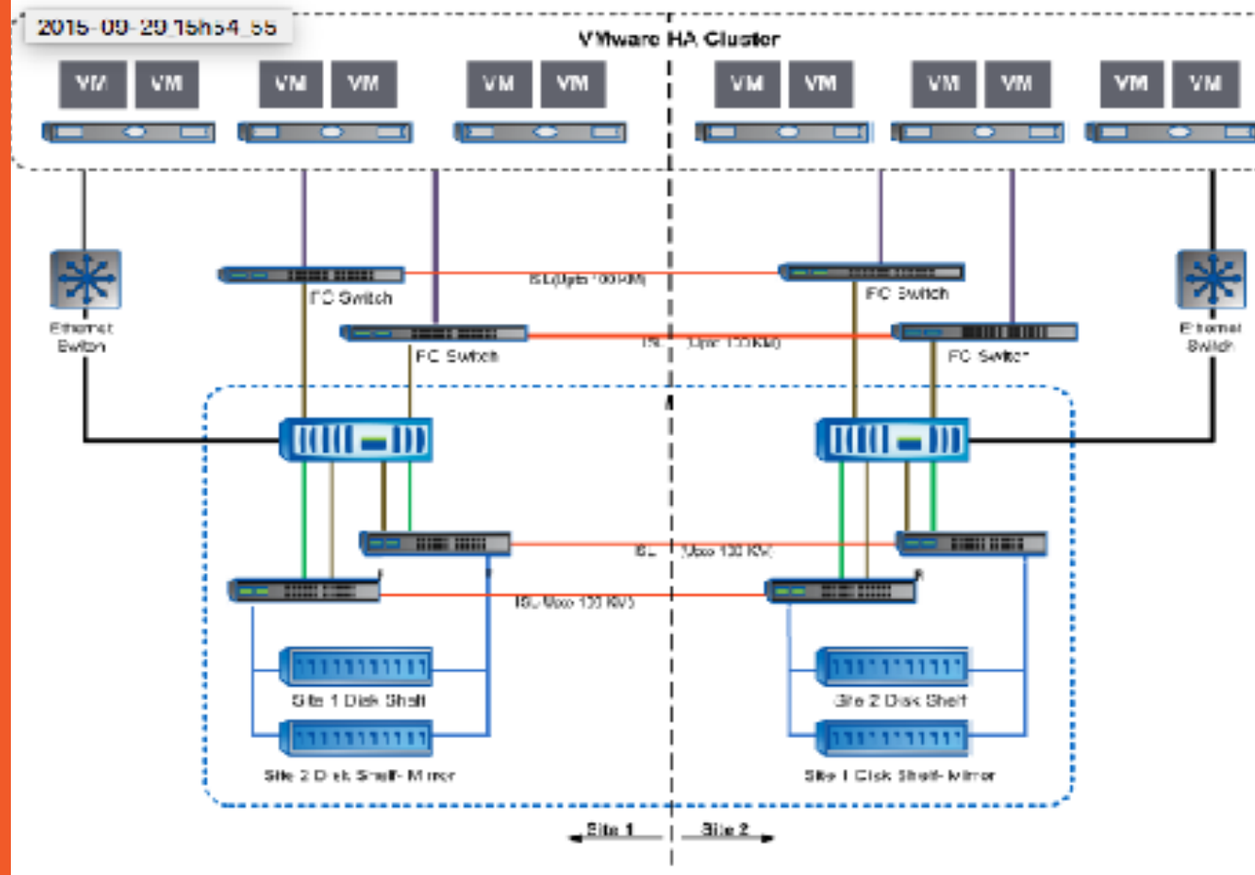
How Did We Get Here?

- Originally — storage consolidation was done for efficiency reasons in terms of performance and capacity
  - Easier data management
- Originating in the LAN (Novell, 80s),
  - Special MS-DOS TSRs to "login in" to the network
  - Novell Application Launcher, IPX/SPX
- Quickly became a dedicated network (00's)
  - Specialised HW — appliances
  - Separate network (FC) switches and cross DC interconnects

OpenEBS

Storage Network

- Typical HA SAN/NAS
  - shared storage
- Front-end back-end
  - HW & SW
  - NVRAM, PCI cache
  - Relatively small CPUs
- HA failover with in 180s
  - Fine tuning of OS-es
- Unified storage
  - Block and file services
  - virtual tape (VTL, B2D)
- Storage features
  - snapshots, clones, dedup, replication



OpenEBS

Software Defined Storage

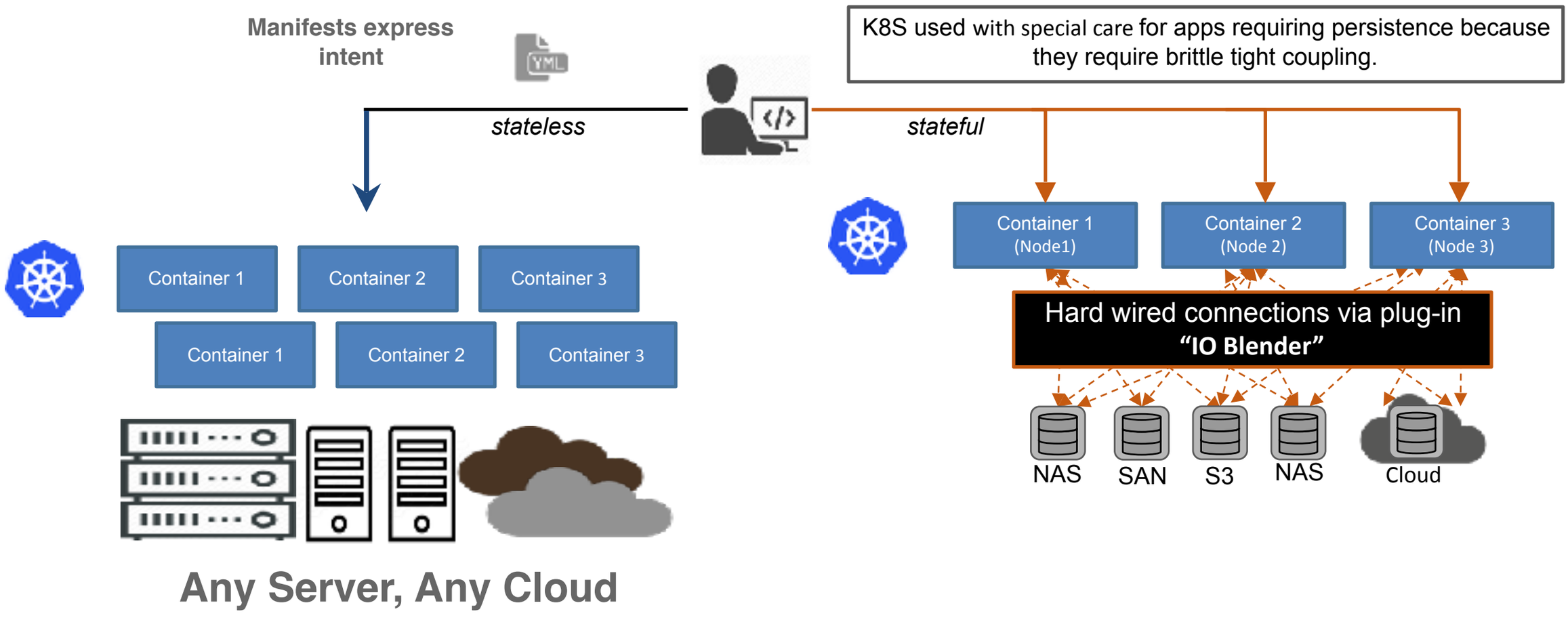
- As hardware evolved, the hardware differentiator became less of of thing compared to open source systems
  - Gluster, Sheepdog, Ceph, FreeNAS
- Commodity hardware, off the shelf
  - FC almost died, FCoE born dead
  - SAS is the new FC; JBODs
- Faster devices (SSD) removed the need for specialised NVRAM and PCI caches
- **Architecture did not change**

Then Containers (re)happend

- Apps have changed somebody forgot to tell storage
- Modern apps have native scalability features
  - load balancers, database sharding, Paxos and RAFT
- Apps deployment; scale up and down on demand
  - K8s based on the google borg paper
  - How to make that work across different storage flavours?
- Apps are designed to fail across DC's and even regions
  - Data availability is not exclusively controlled at the storage layer
- Friction between teams
  - Attempt to duck tape with "volume plugins" as the arch is the same
  - Typically does not allow for storage management capabilities

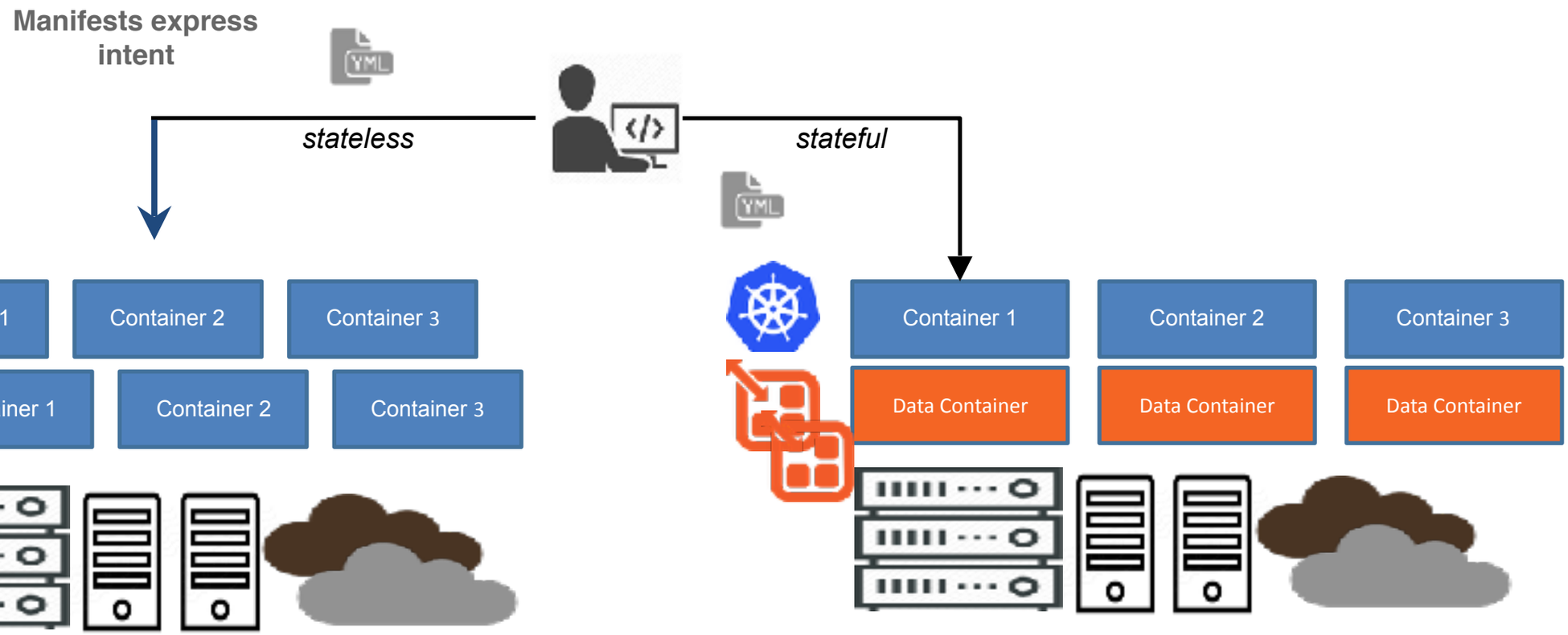
OpenEBS

Containers & k8s



OpenEBS

Containers & k8s



OpenEBS

Example

```
kubectl apply -f https://openebs.github.io/charts/openebs-operator.yaml
```

```
kubectl apply –f percona.yaml
```

```
kubectl get pods | grep pvc
```

```
pvc-8a9fc4b1-d838-11e7-9caa-42010a8000a7-ctrl-696530238-ngrgj   2/2   Running   0   36s
pvc-8a9fc4b1-d838-11e7-9caa-42010a8000a7-rep-3408218758-2ldzv   1/1   Running   0   36s
pvc-8a9fc4b1-d838-11e7-9caa-42010a8000a7-rep-3408218758-6mwj5   1/1   Running   0   36s
```

OpenEBS

Storage just fades away as a concern

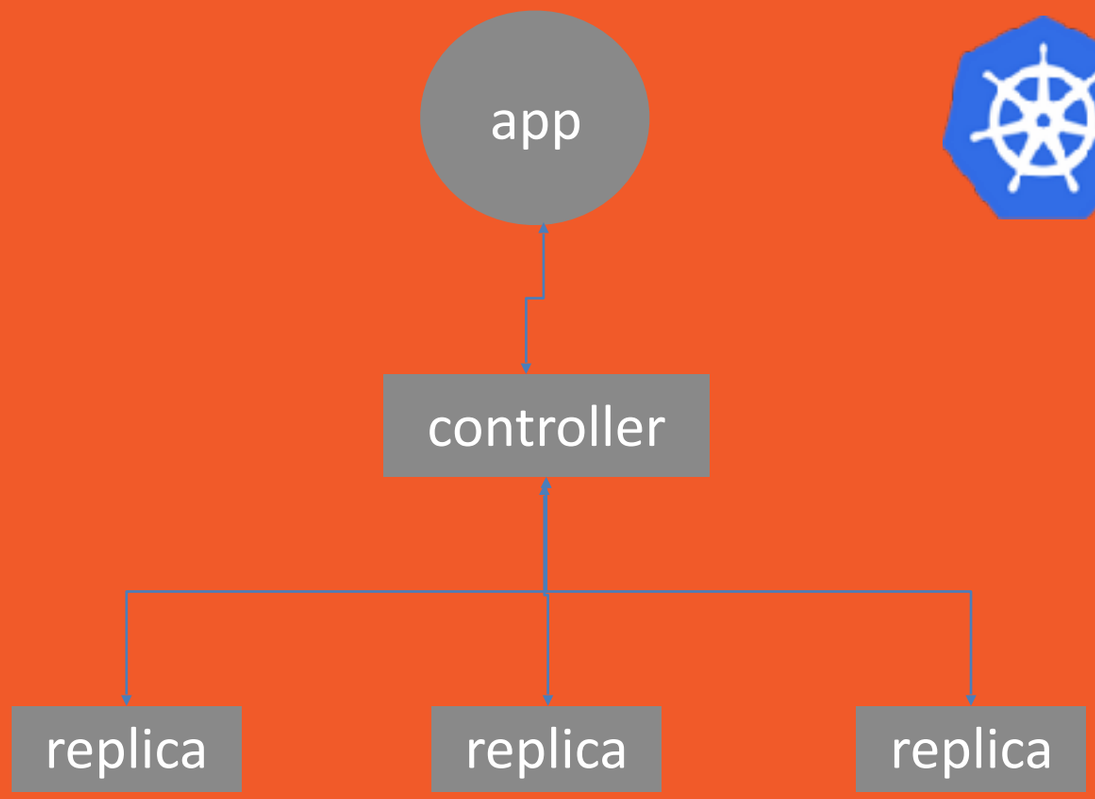
OpenEBS

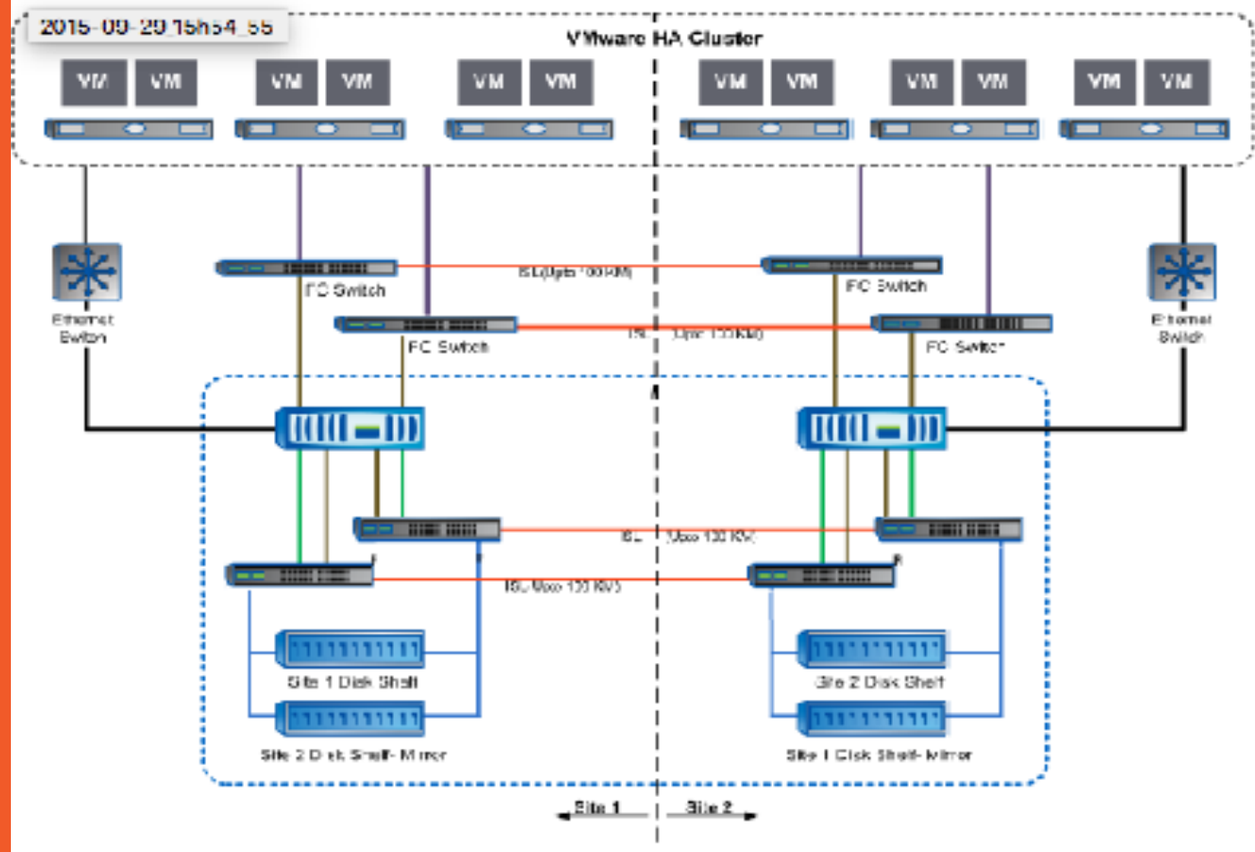
What OpenEBS is not

- It is not a distributed filesystem
  - Hard to manage in production and even harder to debug?
  - Non standard “drivers” to unleash full potential
- Containerised workloads are segregated and thus inherently small
  - Do we need to scale capacity to PBs, scale out?
  - Untangled datasets
- Hardware trends enforce a change in the way we do things
  - Architecture change, finally?
  - Single NMVe devices up to 400K IOPS, scale out?
- Typically container apps are already distributed by nature
  - App performance loosely coupled from storage and scale themselves

OpenEBS

Comparison





OpenEBS

Controller

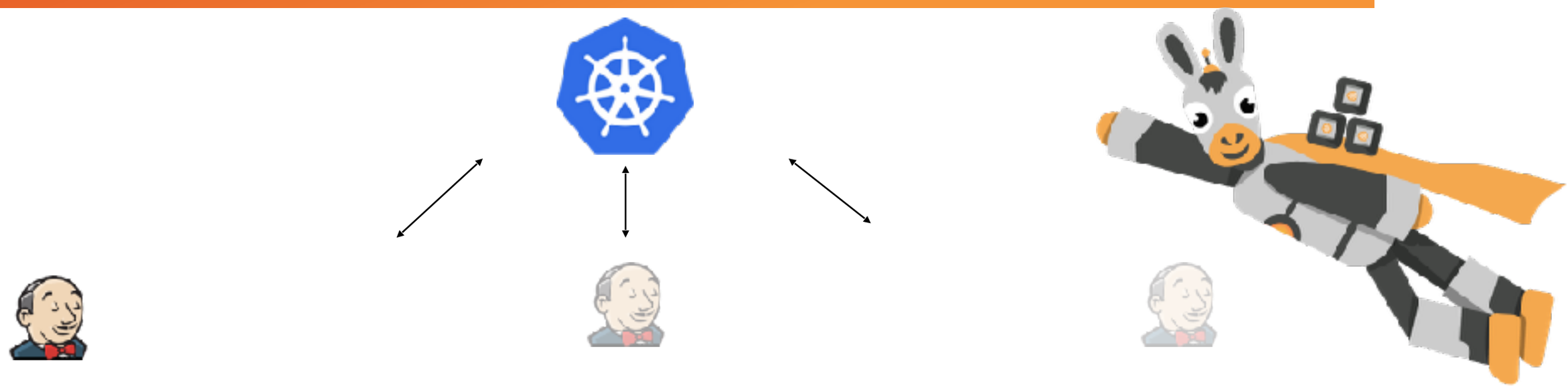
- The actual storage service component to which the client connects – the frontend
  - iSCSI, iSER, NMVeoF
  - Future ideas do include file bases access (NFS, SMB)
- Proxy to a Pluggable backend that allows us to change the the part that actually persists the data (replica)
- Declarative state: define the number or replicas, snapshot schedules, etc
- Consistency levels, one, quorum or all
  - DC aware

Replica engine(s)

- DMU layer of ZFS, well proven and battle tested FS
- Each write is assigned to a transaction
  - Transactions are batched in transaction groups
  - Transaction group numbers are updated atomically which means that all write either succeed or fail
- Pooled storage model
  - VMM type like allocation *(rampant layer violation, 2006)*
- End to end data integrity
- Runs in a container and hence in **user space**
  - No kernel dependencies or DKMS
  - Does not taint the kernel

OpenEBS

cMotion



US-Central

CentOS

EU-East

Ubuntu

US-East

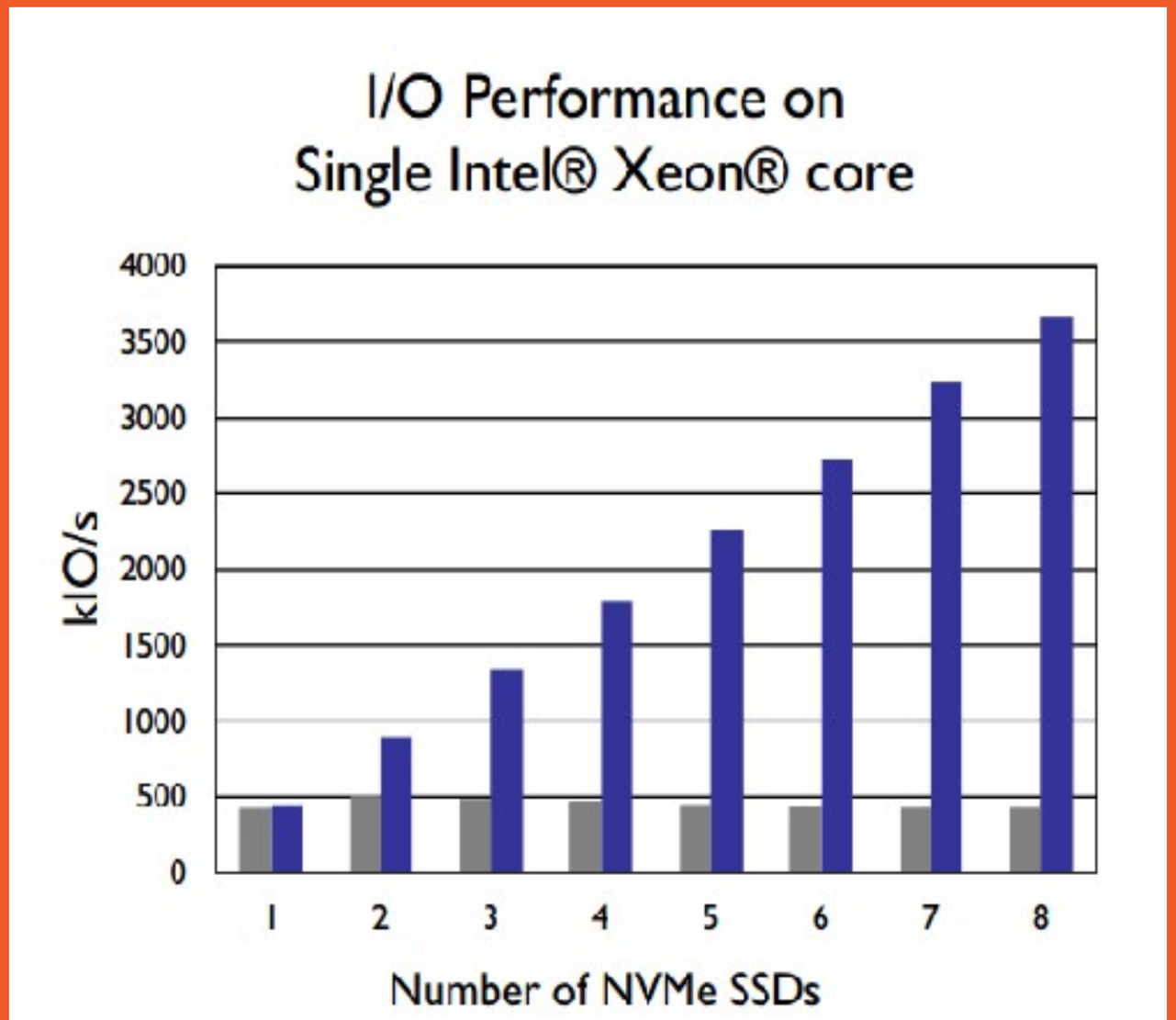
SUSE

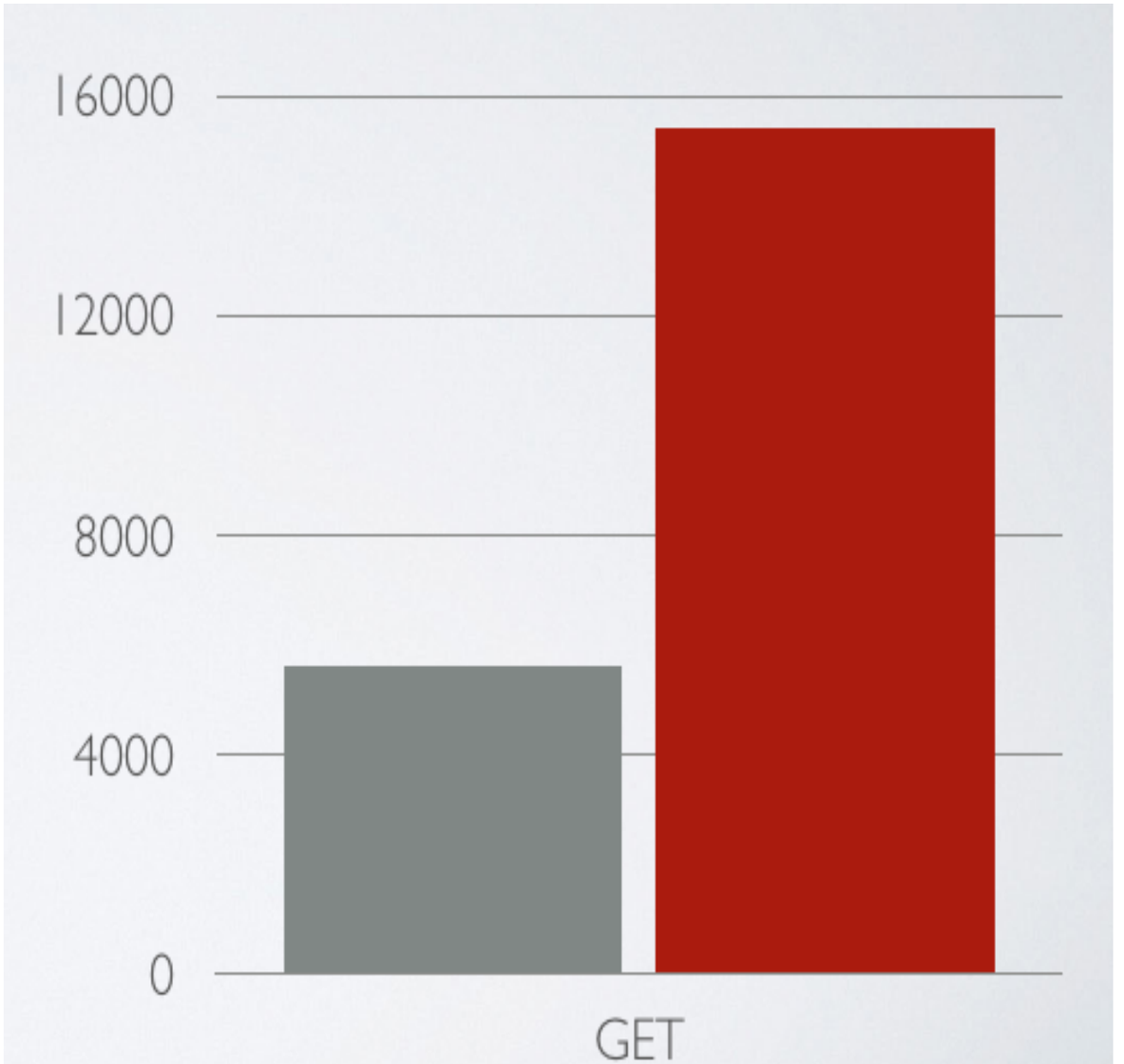
OpenEBS

Storage performance

- *People who think that user space filesystems are realistic for anything but toys are just misguided (2011)*
- How to achieve high performance numbers from user space
  - Context switches
  - Copying (in and out)
  - DMA transfers
- With current hardware trends the kernel actually becomes the bottle neck
  - 100GB networks
  - NVMe devices, 3D X-point
- Frequency remains relativity steady – core count goes up
  - Idle cores?

Scaling IO

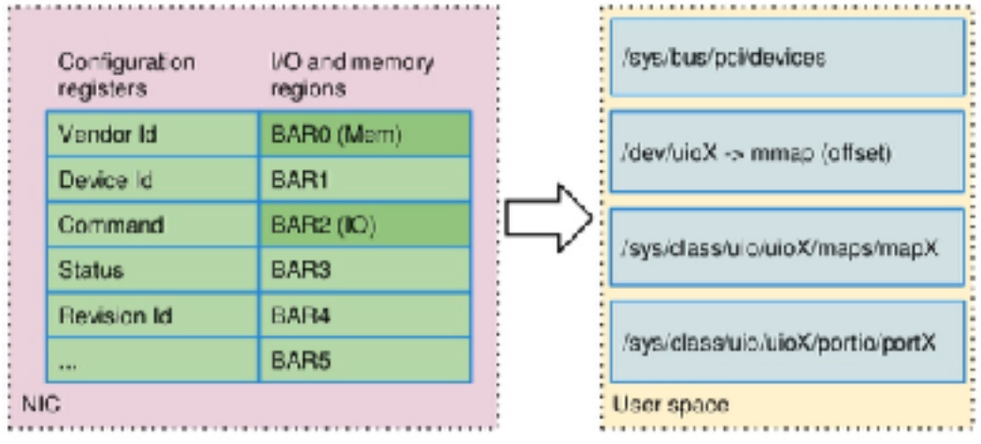




OpenEBS

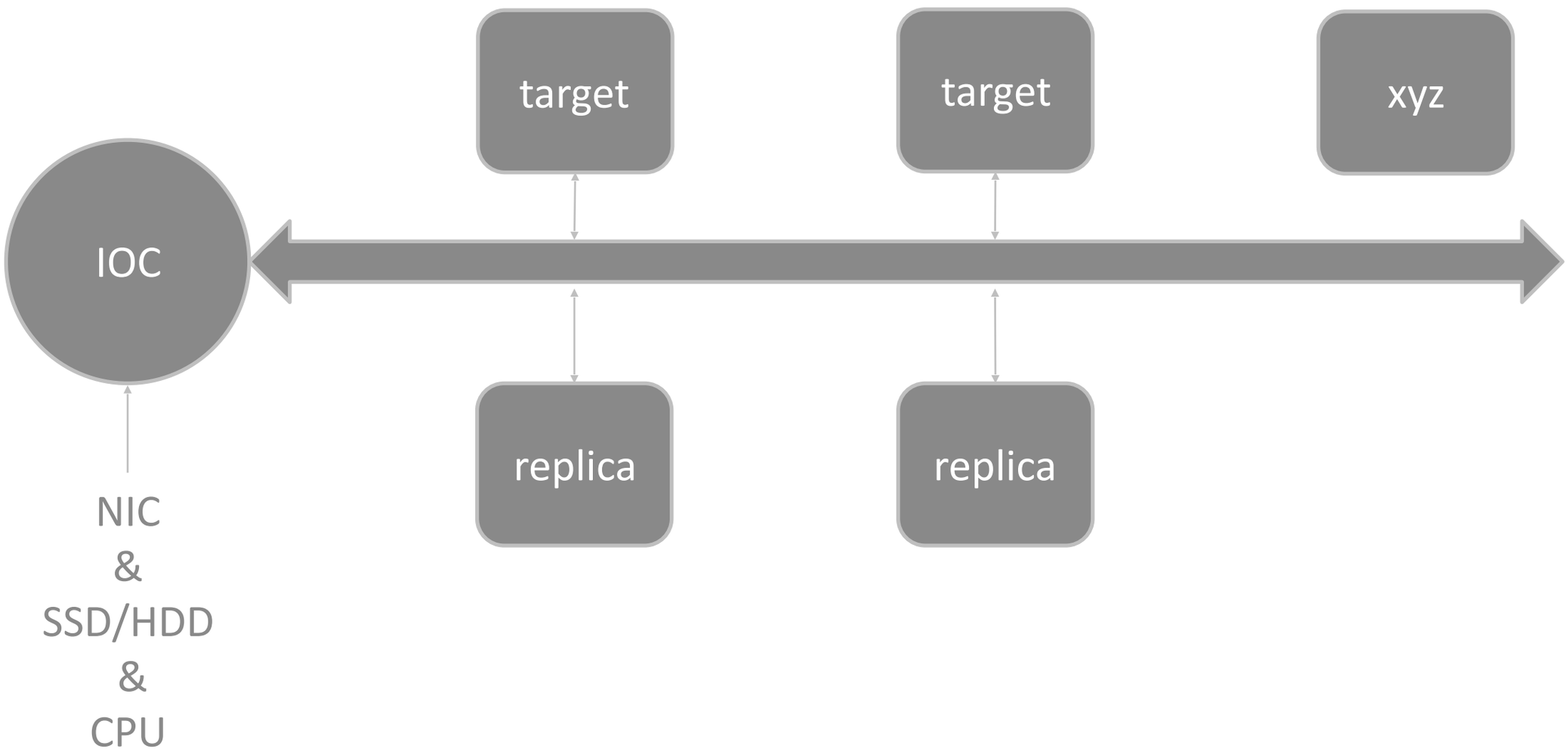
Storage performance

- Bypass the kernel altogether in the IO path, running everything in user space (UIO and SPDK)
- Dedicate core(s) that does IO polling instead of interrupt driven
  - Poll Mode Drivers (PMD) with different implementations, container unfriendly
- Containerise all IO devices – IOC
  - Consumers submit IO to the IOC instead of the kernel
  - A user space "devfs"
- But how to access the devices and sharing them (potentially) with one or more processes?
  - Typically done by the kernel



OpenEBS

Storage performance II

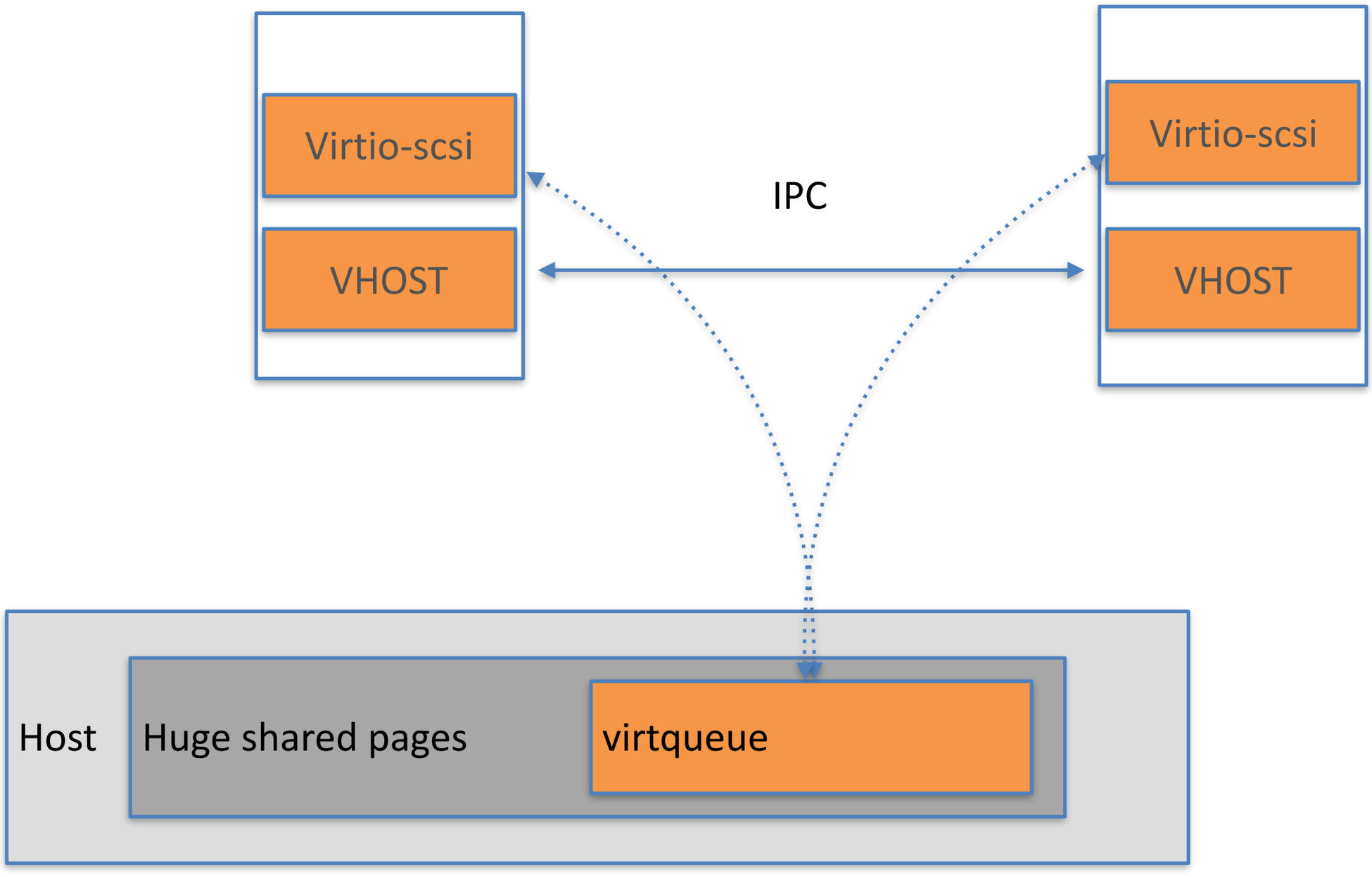


OpenEBS

Storage performance III

- Reuse proven technology VHOST and virtio-{scsi,blk}
  - In user space vhost-user (part of SPDK)
- IOC exposes VHOST interface to the "outside"
- Replica containers connect using virtio-scsi
  - Replica exposes sockets to target for read/write
  - Unfortunately, there is no virtio-scsi library (WIP)
- Allocate huge pages and pin them
  - Suitable for DMA transfers
- Future work
  - Explore further integration with FD.IO
  - VPP-VCL in particular

VHOST USER



OpenEBS

Summary about OpenEBS

- Bring advanced storage feature to individual container workloads
  - COW, data integrity, storage reduction, snapshotting and replication
- Cloud native; using the same software development paradigm
  - Build for containers in containers
- Implemented fully in user space
  - Avoid congestion in kernel
  - multi cloud
- Declarative provisioning and protection policies
  - Remove friction between teams
- **Not a clustered storage instance rather a cluster of storage instances**

OpenEBS

QUESTIONS?

www.openebs.io

OpenEBS