@JeffryMolanus
@openEBS

https://openebs.io

mountpoint.io - 2018

OpenEBS

Containerized Storage for Containers

Latest (storage IO) patterns for cloud-native applications in a k8s environment

OpenEBS

People

Software

&

Hardware

Applications have changed;

&

somebody forgot to tell **storage**

Cloud native software architecture

- Cloud native ***apps*** that ***are distributed systems*** themselves
  - Let us use Paxos, RAFT, nobody flinches
- They want it to scale by default — batteries included
  - HaProxy, Envoy — no more storage scaling
- Apps are designed to fail across DC's, regions and providers
  - Should be multi-cloud, multi-hypervisor and multi-platform
- Databases provide distributed scale out; or one can use vitess for existing SQL (no-noSQL) databases
- Datasets of individual containers are relatively **small**
  - **The sum of the parts is greater then the whole**

OpenEBS

Data availability and performance is **not** (anymore) exclusively controlled at the storage layer

OpenEBS

DevOps (the people)

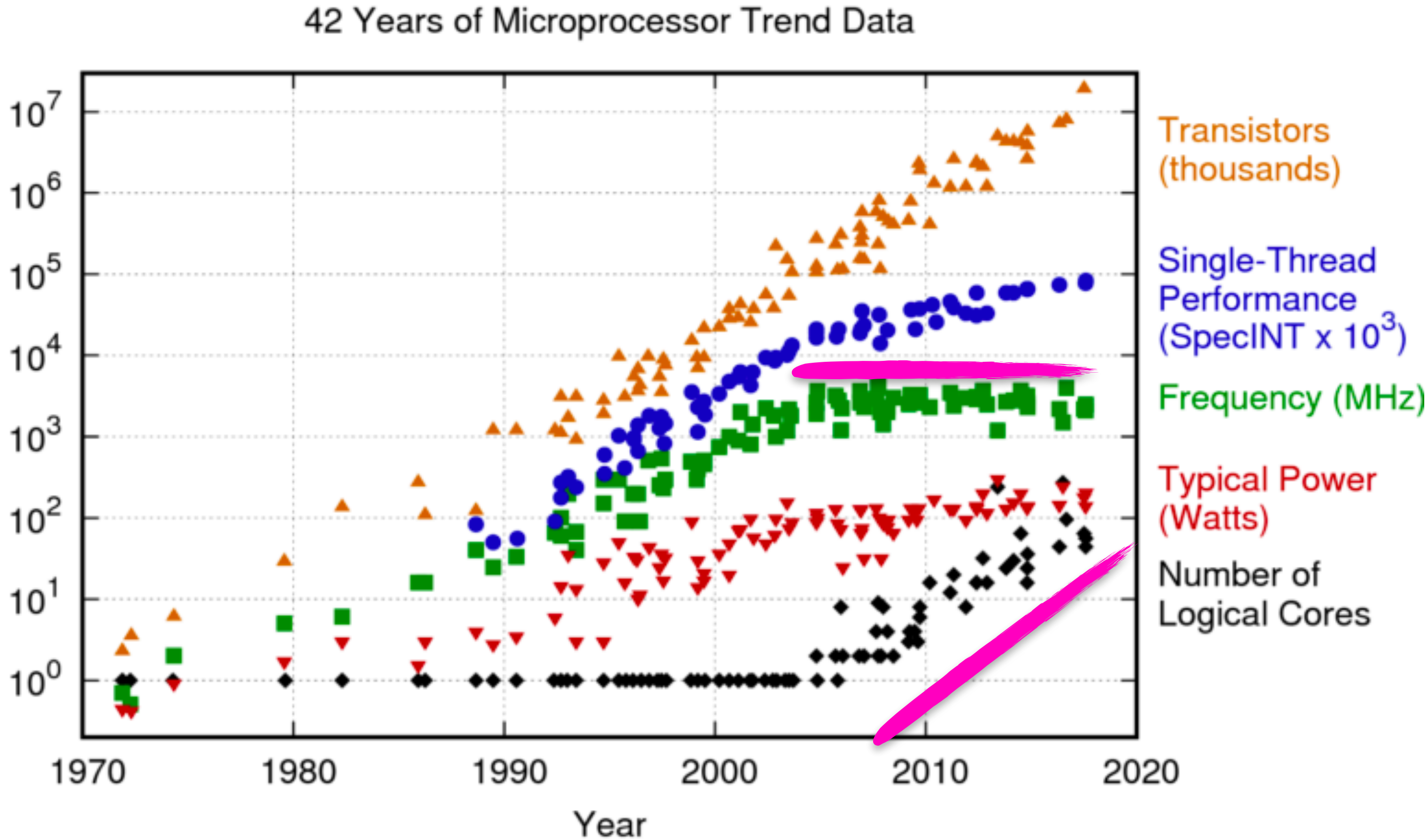
- Deliver **fast** and **frequently**
  - A deployment per day keeps the ….. away
- Small teams with specific goals. Shadow IT is where the innovation happens — born in the cloud
- **CI/CD** pipelines — blue-green or cannery deployment
- Make install has been upgraded to make push
  - Software delivery has changed, **tarball on steroids**
- **Declarative intent**, gitOps, chatOps
- K8s as the unified **cross cloud** control plane (control loop)
  - Everything in containers either bare metal or lkvm

HW / Storage trends

- Storage appliance peculiarities **bubble up** in apps
  - Don't do this because… don't do that because….
  - Makes it hard to write code that uses the full stack optimal when moving from c2c, private or public
  - Some vendors — simply put their appliances in the cloud
- Friction; “Do **not** run your CI while I do backups!”
  - You need LU’s again? Gave you 100 yesterday!
- ***“We simply use DAS as nothing is faster than that”***
  - NVMe and PDIMs **enforce** a **change** in the way we do things
- Increasing core counts create new challenges
  - caches, migrations, NUMA and yes — not fully utilised cores

OpenEBS

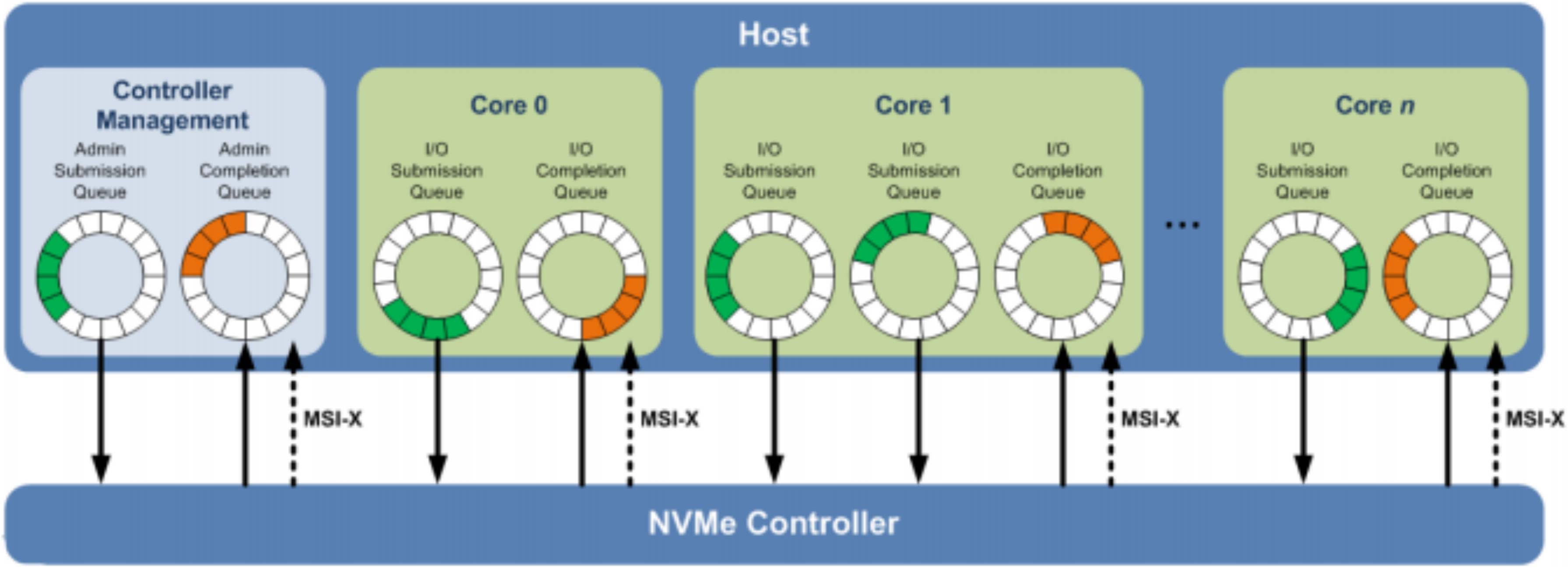
HW / Storage trends



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

OpenEBS

HW / Storage trends




ONE DOES NOT SIMPLY
CREATE NEW A STORAGE SYSTEM
imgflip.com

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

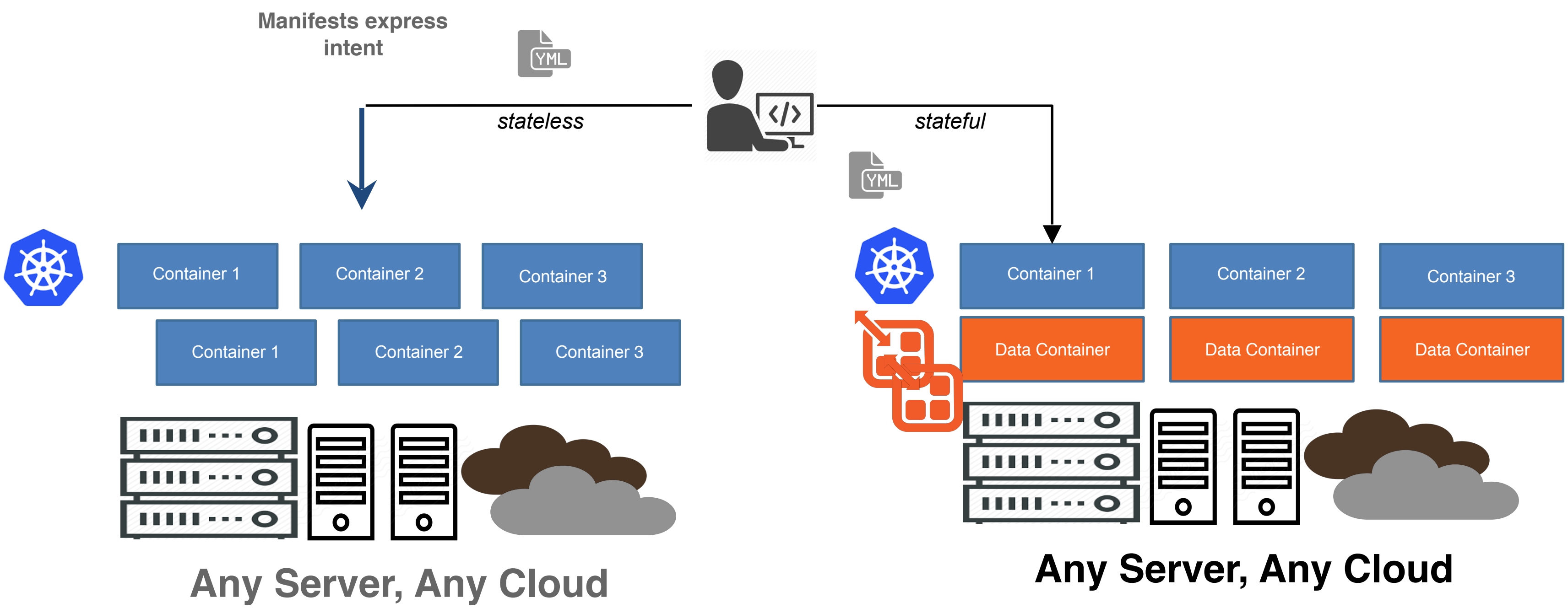
OpenEBS

Design constraints

- Not yet another distributed storage system; small is the new big
- Cloud native (not washed) applications are, inherently distributed applications
    - One on top of the other, an operational nightmare?
- Per workload storage system, using **declarative intent** defined by the **developer**
    - Applications defined storage
- Reduce blast radius and no IO blender
- Runs in containers for containers — **in user space**
- Not a clustered storage instance rather a cluster of storage instances

OpenEBS

Containers & k8s



OpenEBS

Challenge

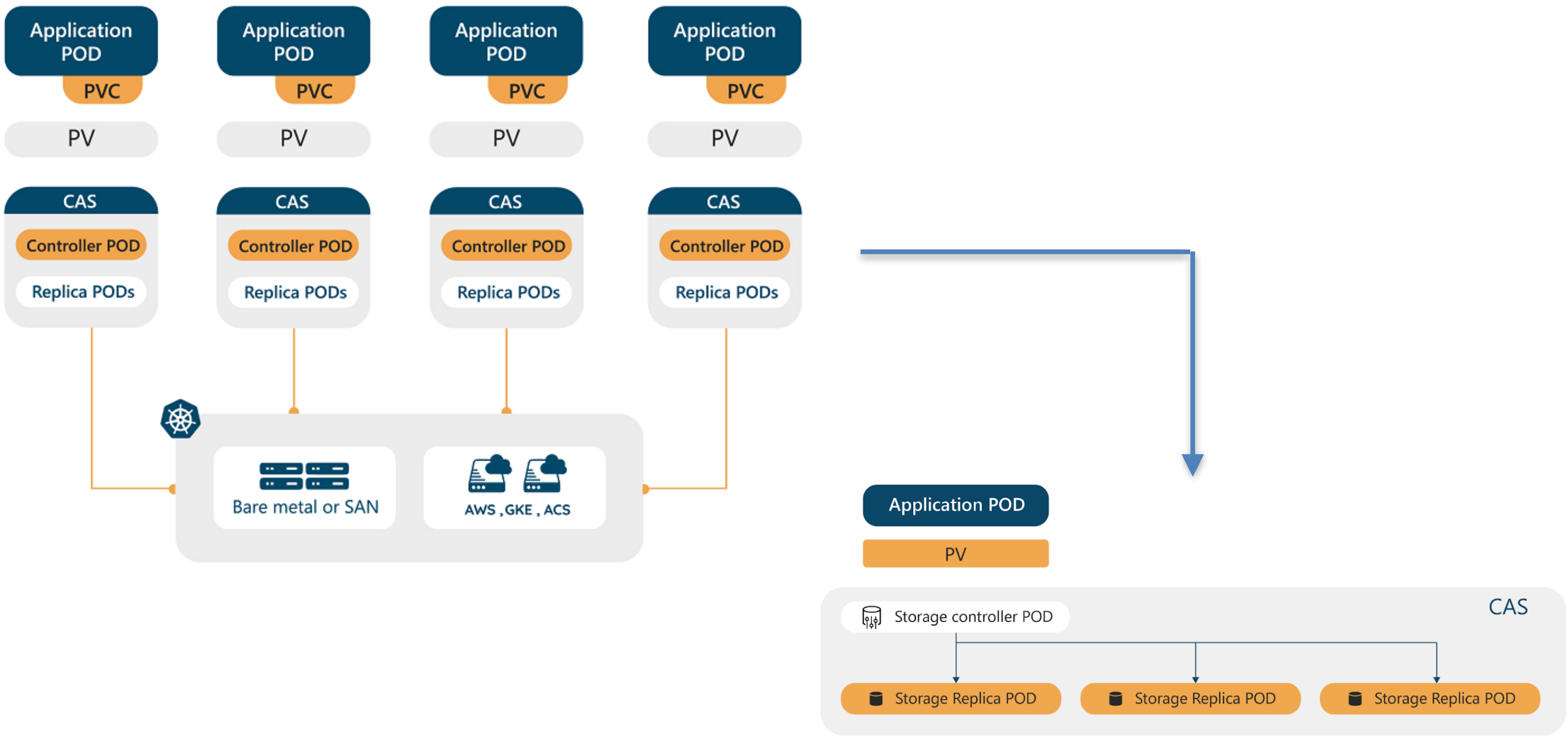
1. Small working sets
2. Ephemeral
3. Scale by N
4. Mobile workloads
5. DevOps responsible for operations
6. Cloud lock-in
7. Per workload optimisation

Solution

1. Keep data local
2. Controller in POD
3. N more containers
4. Follow the workload
5. Just another micro service
6. Workload migration
7. Declarative intent

OpenEBS

High level CAS architecture



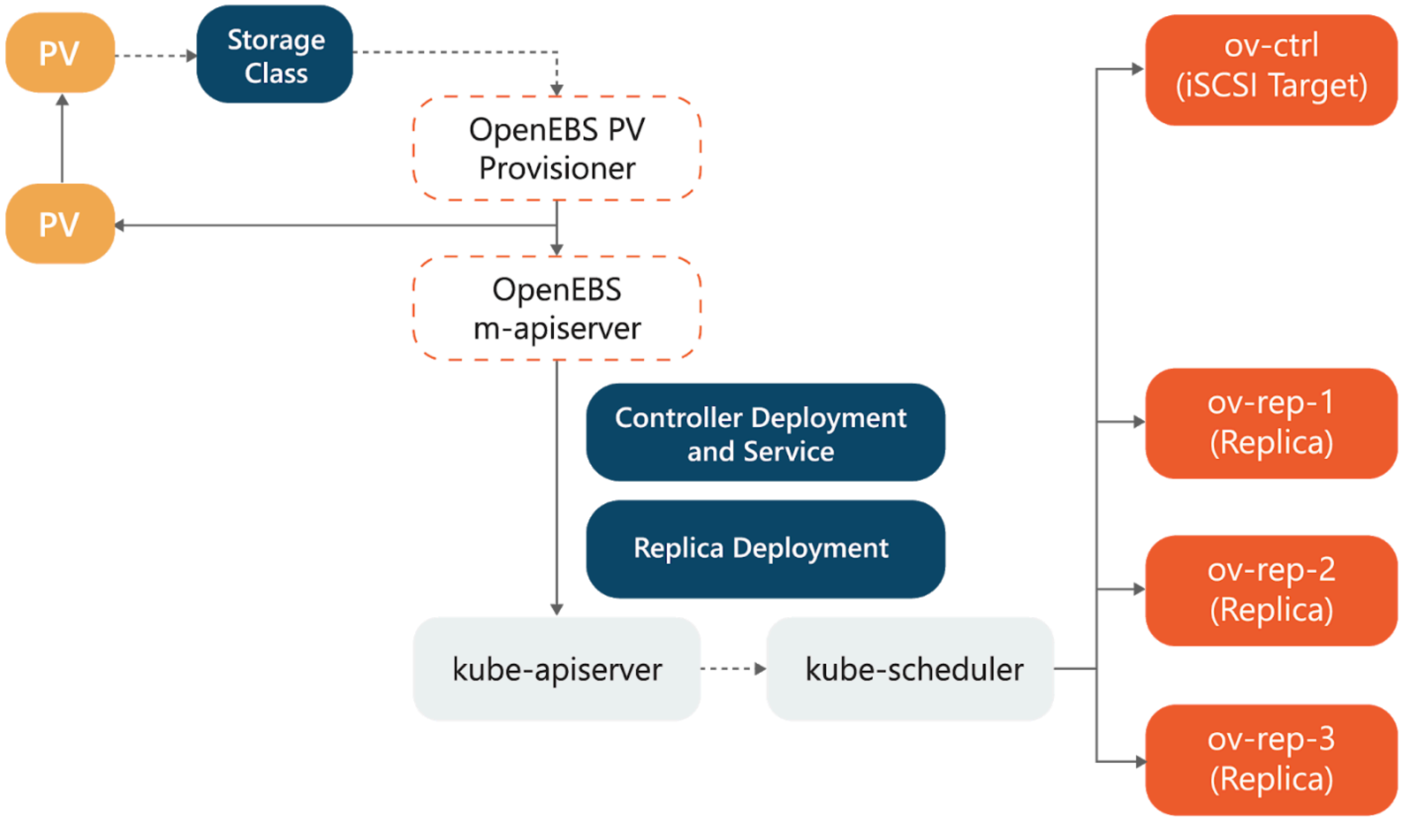
OpenEBS


OPENEBS BE LIKE:
AND YOU GET A CONTAINERIZED STORAGE CONTROLLER
imgflip.com

Using the k8s substrate

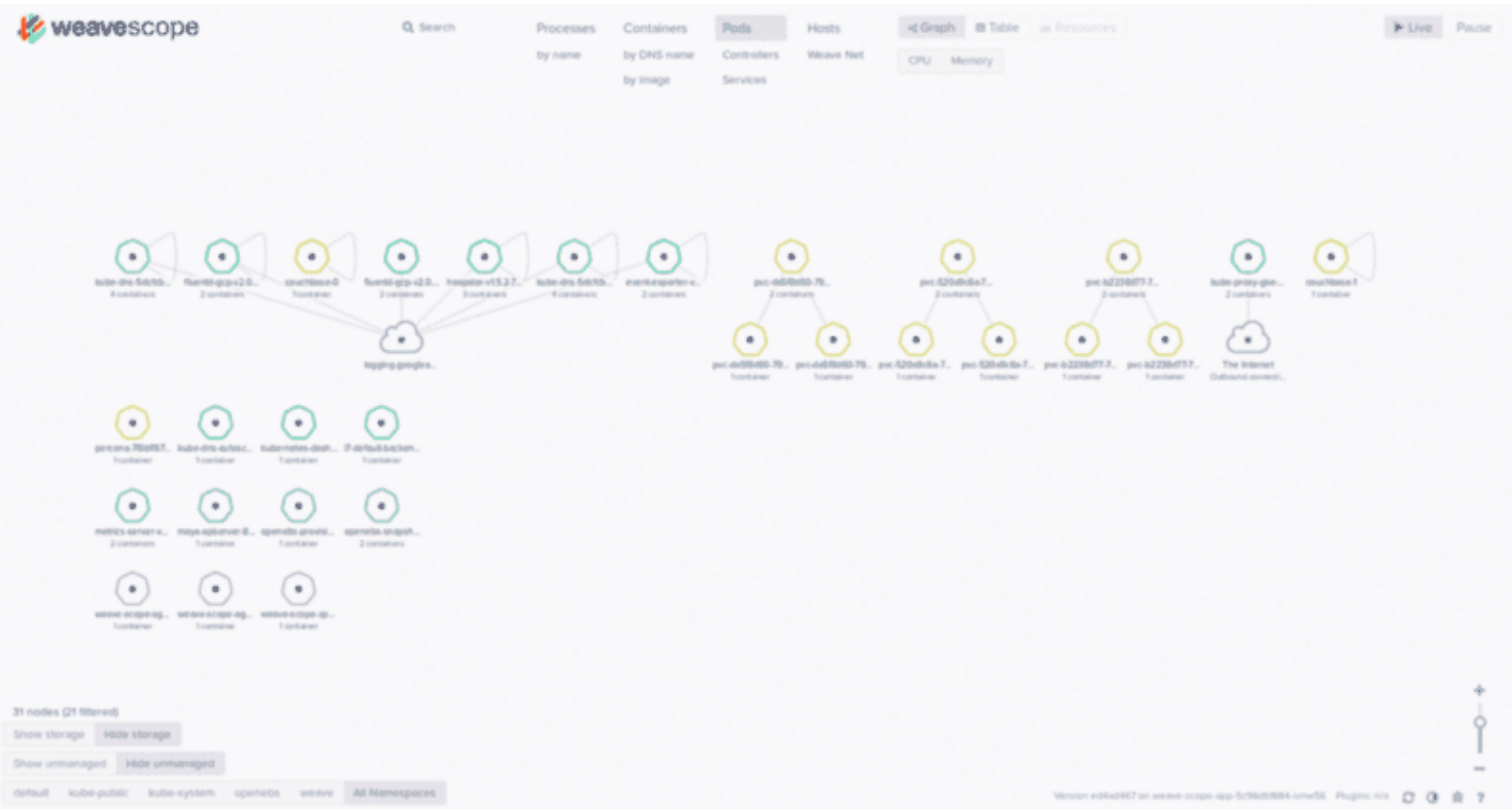
- Betting on k8s; don't integrate with plugins actually build on-top of it
  - CSI plugin standardised API to communicate with external storage (controller and agent)
- Implement dynamic provisioner to construct “volumes” **(openEBS operator)**
- Using the operator framework to construct storage topology and reflect storage systems state (kubectl describe)
  - watchers and CRDs to invoke logic to reconcile desired state
- Again, using the operator framework to discover local devices and their properties to create storage pools dynamically **(NDM)**
- Fully operated by kubectl i.e no external tools required (*)
- Visualise topology and EE testing **(Litmus)**

k8s control plane for storage



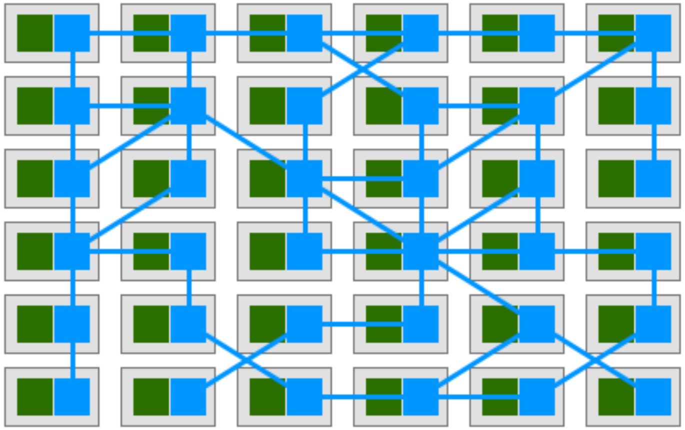
Provisoning flow in OpenEBS

OpenEBS


weavescope
Search
Processes
by name
Containers
by DNS name
by image
Pods
Controllers
Services
Hosts
Weave Net
Graph
Table
Resources
CPU
Memory
Live
Pause
The Internet
Show storage
Hide storage
Show unmanaged
Hide unmanaged
default
kube-public
kube-system
openebs
weave
All Namespaces

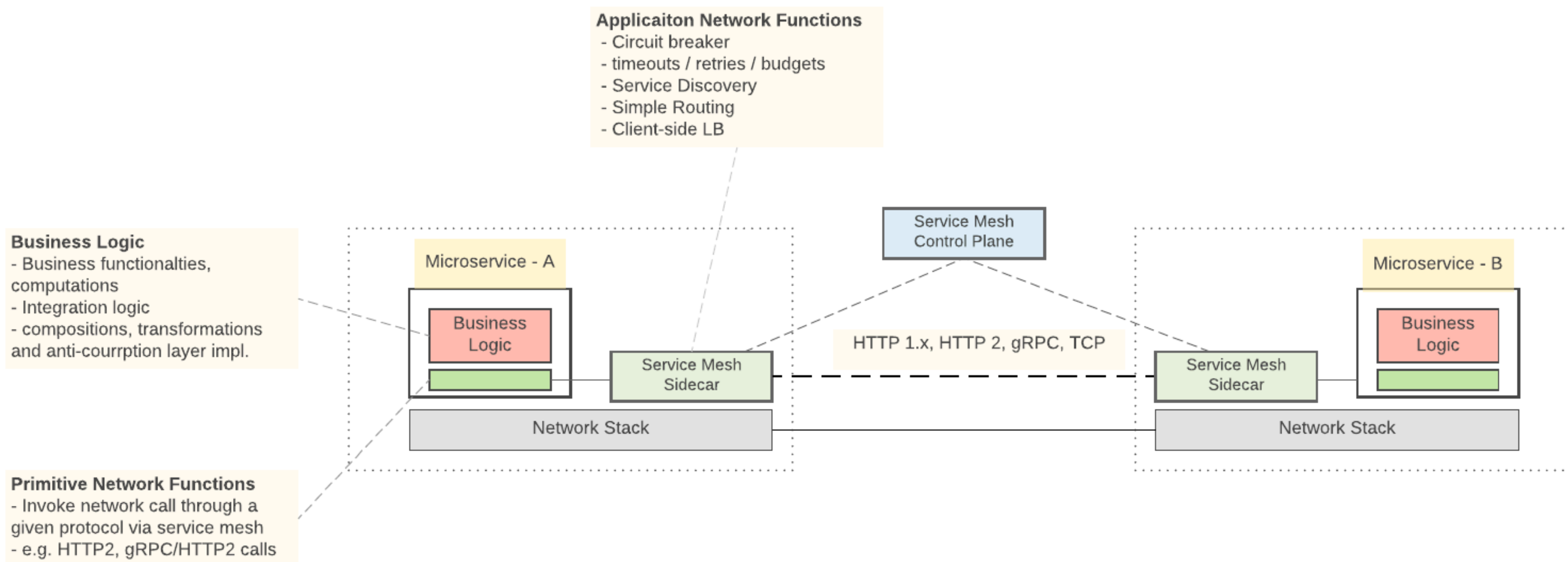
Patterns; sidecars & meshes

- A **CAS** volume consists out of **controller** and a **replica** and lives **somewhere**
- The fallacies of distributed computing (L. Peter Deutsch)
  - The only constant is **change**
- How do we **dynamic** (re) configure?
  - Optimal transport/path
  - Rescheduling
  - Different (virtual) machines
- **Data mesh** for dynamic IO reconfiguration



OpenEBS

K8s pattern; service mesh



[Kasun Indrasiri]()

OpenEBS

Data mesh negotiated transport

```yaml
kind: DataFrabricConnection
apiVersion: V1
metadata:
  labels:
    - ....
spec:
  name: my-iospec
  ioChannel: my-first-channel
  request:
    type: block
    - nvmeof
    - nbd
    - iscsi
    - .....
    properties:
      compress: false
      encrypt : true
  ....
  ....
```

- Controller and replica need to find optimal path — but also the app to the controller
- Virtual “HBA” uses negotiated transport and features (VHCI)
  - Capable of using different transport types
- Connection types and state reflected in custom resources
  - kubectl edit or update -f xyz.yaml
- Opportunity to innovate for application optimised IO patterns: **smart endpoints dumb pipes**

OpenEBS

Storage just fades away as a concern

OpenEBS

Implementation details

- JIVA, the primordial soup to determine feasibility
    - Praised for its ease of use by **users**
- Instrumental to use to find and explore **uses case** for the **cloud native** landscape
- Does not provide **efficient** "enterprise" storage features
- Swapping out JIVA with something else is just a matter of adding storage classes so we are evolving (pluggable)
    - Yay for the micro service approach
- The biggest problem to solve however is user space IO
    - Different kernels on different clouds — tainting
- Performance in public cloud **not yet** the biggest concern

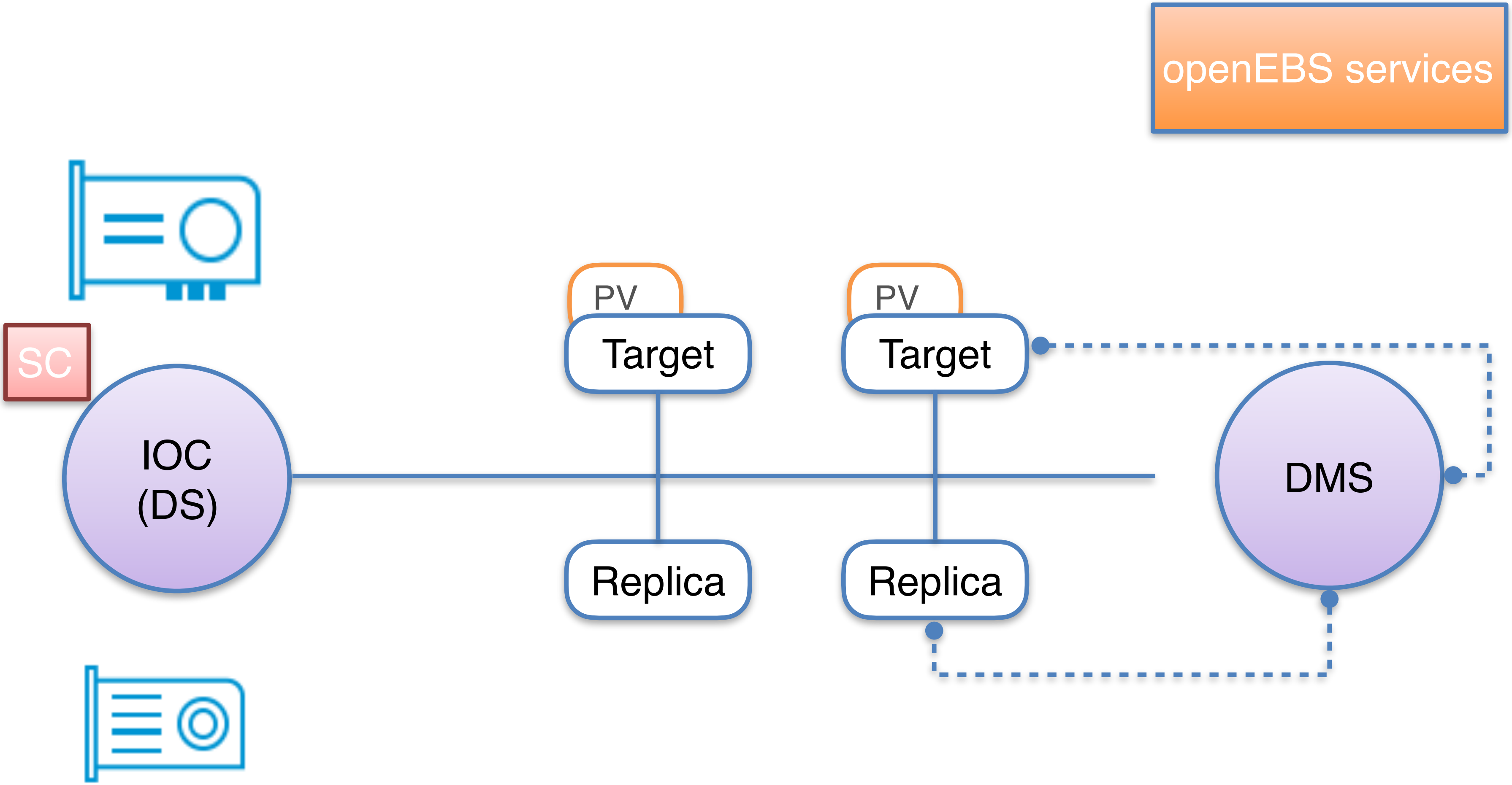
OpenEBS

Input output container (IOC)

- If containers perform (mostly) API request to one and other, why not have them do storage IO to each other?
- Select devices (NDM) and claim resources in the pod spec
  - DSM can handle this automatically as well
- IOC DaemonSet grabs the devices and exposes them through a variety of different protocols
- Leveraging Storage Plane Development Kit
  - There are other things available like UNVMe however
- Bypass the kernel using UIO/VFIO and DMA straight into the devices by leveraging huge pages (Poll Mode Drivers)

OpenEBS

IOC overview



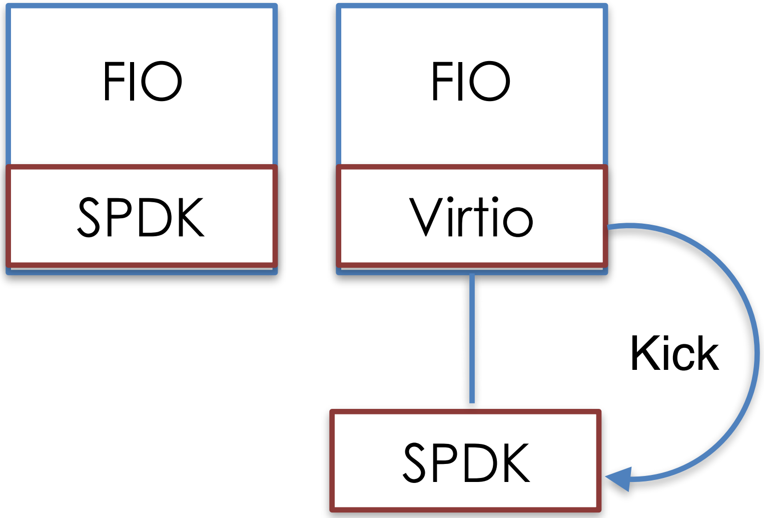
Reusing virtio for containers

- Widely used and actively developed protocol which uses shared memory to interact with several types of hardware
- In the case of openEBS — interested in user space virtio-{blk,nvme} **initiator**
- Primary reason for developing this was to have a loosely coupling with SPDK which use **Poll Mode Drivers** (PMD)
  - Perhaps also LIO's vhost support
- Even-though we have plenty of cores — having anything and everything attached to openEBS do polling is not acceptable
- There was no "libvirtio" unfortunately, so we created one

OpenEBS

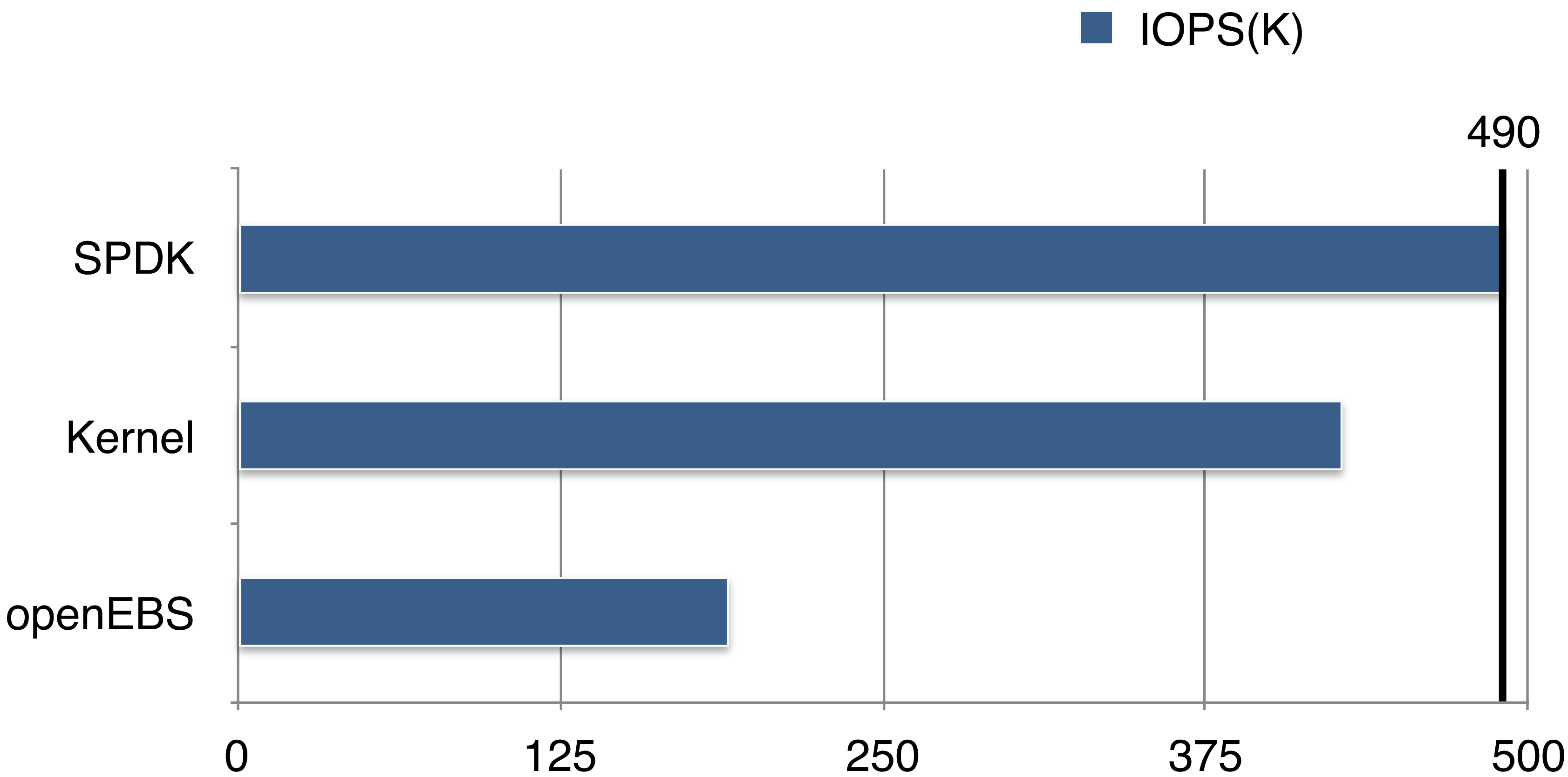
Feasibility test

- SPDK in full polling mode using the SPDK provided plugin
- Virtio plugin using SHM, to issue IO to SPDKs
- Experiment expectations:
  - Due to non polling mode performance will drop
  - Due to eventfd() performance will drop (context switches)
- Desired result: ~kernel
  - Quid pro quo



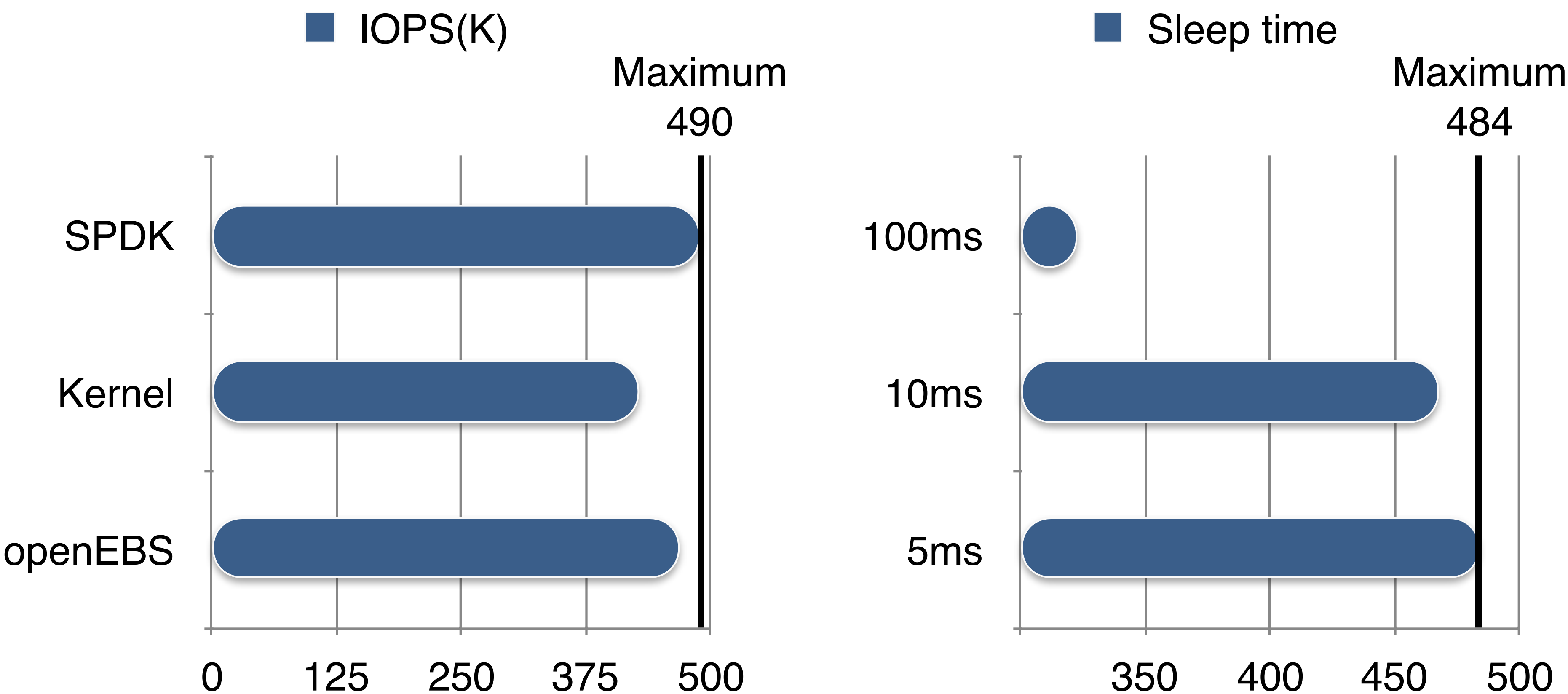
OpenEBS

Results of the first test



OpenEBS

Results using adaptive polling



OpenEBS

Initial observation

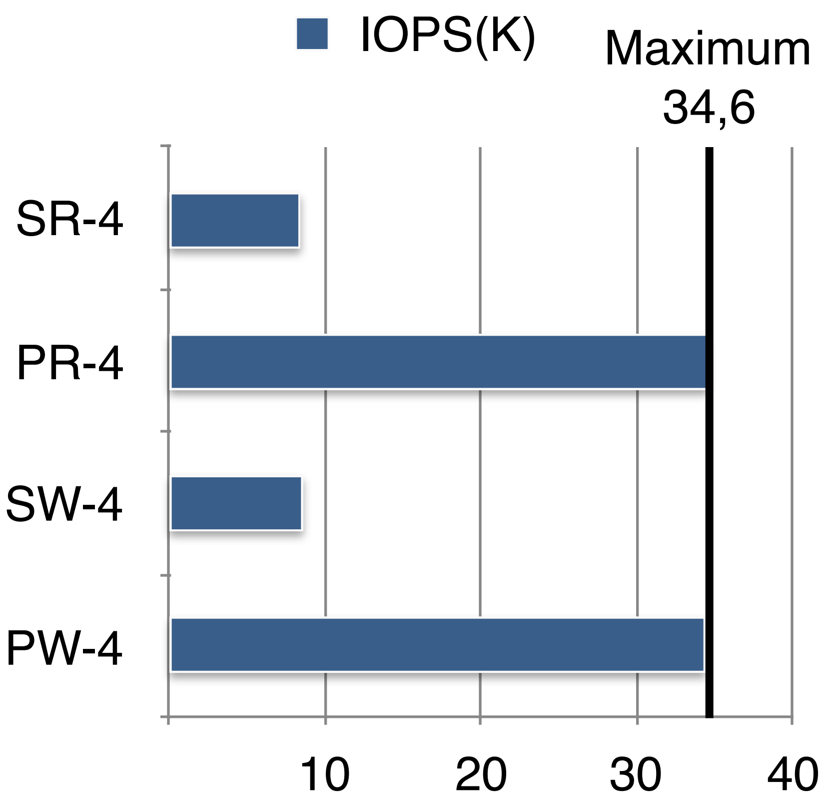
- SPDK can indeed out perform the kernel
  - Using it however has some ramifications but is IO processing in user space becoming a new trend?
- Using **virtio** as a user space library to do block IO is **feasible**
- Using **eventfd**() to kick the vhost controller has very high **negative** impact on performance (how much actually was surprising)
- Sleepy polling improves performance reaching (~0.82%) of direct NVMe with no virtio only a **6K** IOPS **drop**)
- Implement adaptive polling that dynamically updates the sleep interval based on measured throughput

OpenEBS

Go-virtio API

- Can we implement virtio in Go such that the read and write interfaces use virtio?
- Golang interfaces are like classes If you implement the methods you satisfy the interface
- Go uses go routines — user level threads, **should** provide less scheduling overhead when using multiple routines
  - Less context switches is good for performance
- Need to understand the “M”, “G” and “P” which is part of the go runtime
  - C functions are always executed on a separate M, implement it natively in go

Go virtio-blk (not the same HW)



- Cant use small sleeps in Golang to do polling
  - #25471
- Results shown use eventfd()
- Clearly more work needed but its a start
- Results obtained using go test -bench
- **Note; not the same HW as before!**

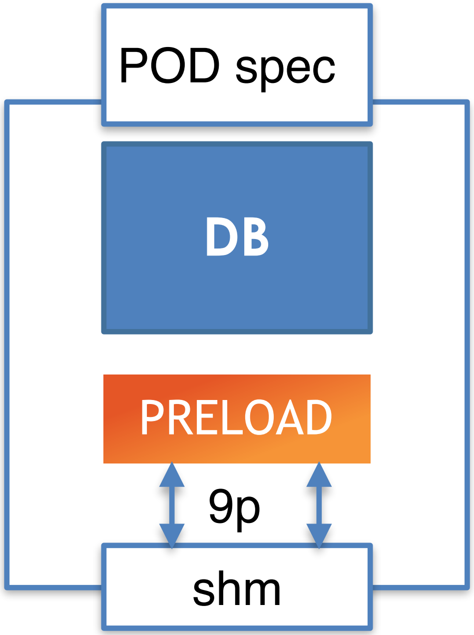
OpenEBS

Other protocols

- Most applications wont be able to connect directly with virtio
  - Support for iSCSI, NBD, TCMU,
- To really keep up with NVMe we need nvme-of to be more widely adopted
  - Should work over TCP as well as RNICs for transitions in particular for cloud based deployments (softroce and nvmeof-tcp)
- Add support for contiv which leverages VPP-VCL to accelerate network and **stay** in user space
  - At current requires TAP/TAP2 — to expose interface to the container
  - Microsoft FreeFlow also aimed at network acceleration
- Both implementations use LD_PRELOADs to intercept syscalls to avoid application changes

File based access

- Inject syscall interception library for applications that need basic file operations typically for databases that have data consistency models built in
  - Not targeted towards file servers
- DB have a very typical IO pattern, mostly append only as they typically have some form of WAL with compaction
- Library is mounted in the namespace configured based on developer intent
- Crucial to have proper testing and regression framework
  - CI/CD, devOps, End 2 End (litmus)



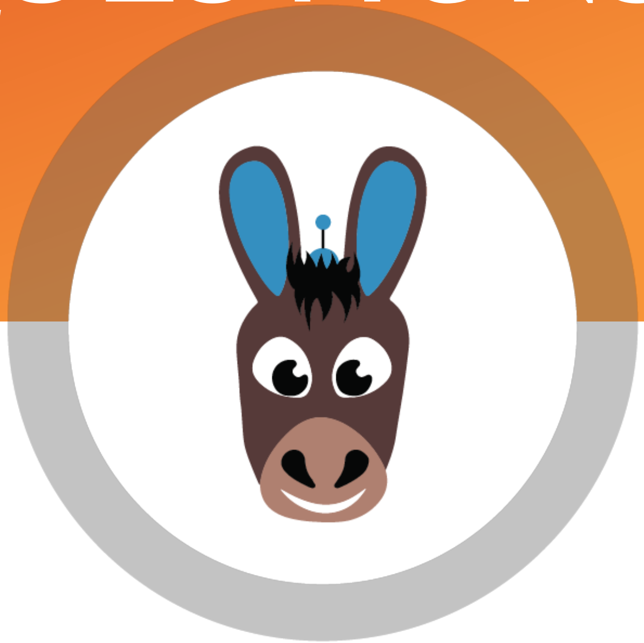
OpenEBS

Summary about OpenEBS

- Bring advanced storage feature to individual container workloads
- Cloud native; using the same software development paradigm
    - Build for containers in containers
- IO handling from the IOC implemented fully in user space
    - Control release cadence, extra performance is a bonus
- Declarative provisioning and protection policies
    - Remove friction between teams
- Multi cloud from public to private
- Not a clustered storage instance rather a cluster of storage instances

OpenEBS

QUESTIONS?



https://openebs.io

OpenEBS