

The logo for Storage Developer Conference 2018 (SDC 18) features the letters 'SDC' in a large, bold, white sans-serif font, followed by '18' inside a white circle. The background of the slide is dark blue with a complex, light-colored geometric pattern of overlapping lines and rectangles on the left side.

SDC 18

September 24-27, 2018
Santa Clara, CA

www.storagedeveloper.org

Container Attached Storage

@JeffryMolanus

<https://openefs.io>

Trends in general that fuel the cloud

- ❑ People
 - ❑ Devops
- ❑ Software
 - ❑ How this is build and deployed
- ❑ Hardware
 - ❑ NVMe, Fabrics



Applications have *changed* and someone forgot to tell *storage*

Cloud native SW architecture

- ❑ Cloud native apps 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

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

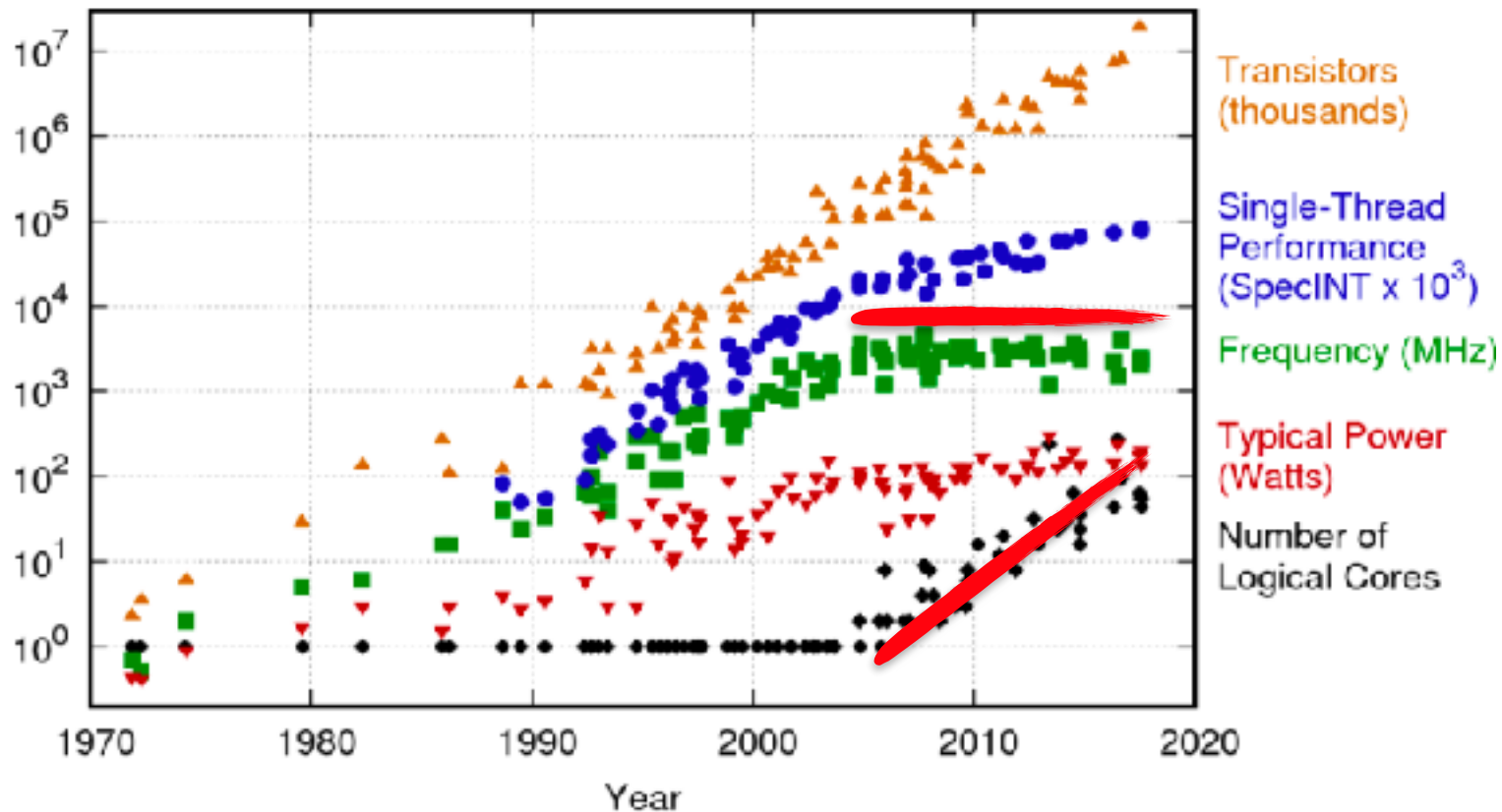
The people

- ❑ Deliver **fast** and **frequent**
- ❑ 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

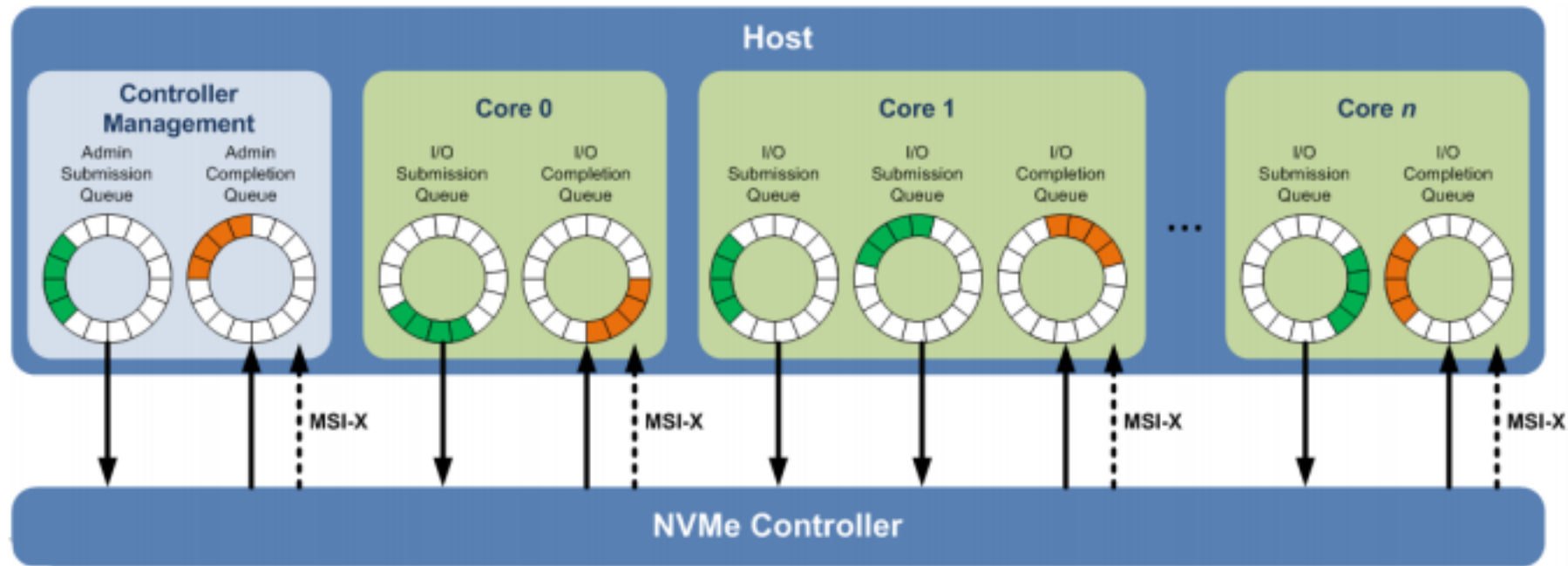
- ❑ Storage appliance peculiarities bubble up in apps
 - ❑ Don't do this because... don't do that because....
 - ❑ Makes it hard to write code that uses the full stack optimal when moving from c2c, private or public
- ❑ Friction; “Do not run your CI while I do backups!”
 - ❑ You need LU's again? Gave 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
 - ❑ Concurrency primitives built in to the languages

42 Years of Microprocessor Trend Data



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

Perfect match



ONE DOES NOT SIMPLY

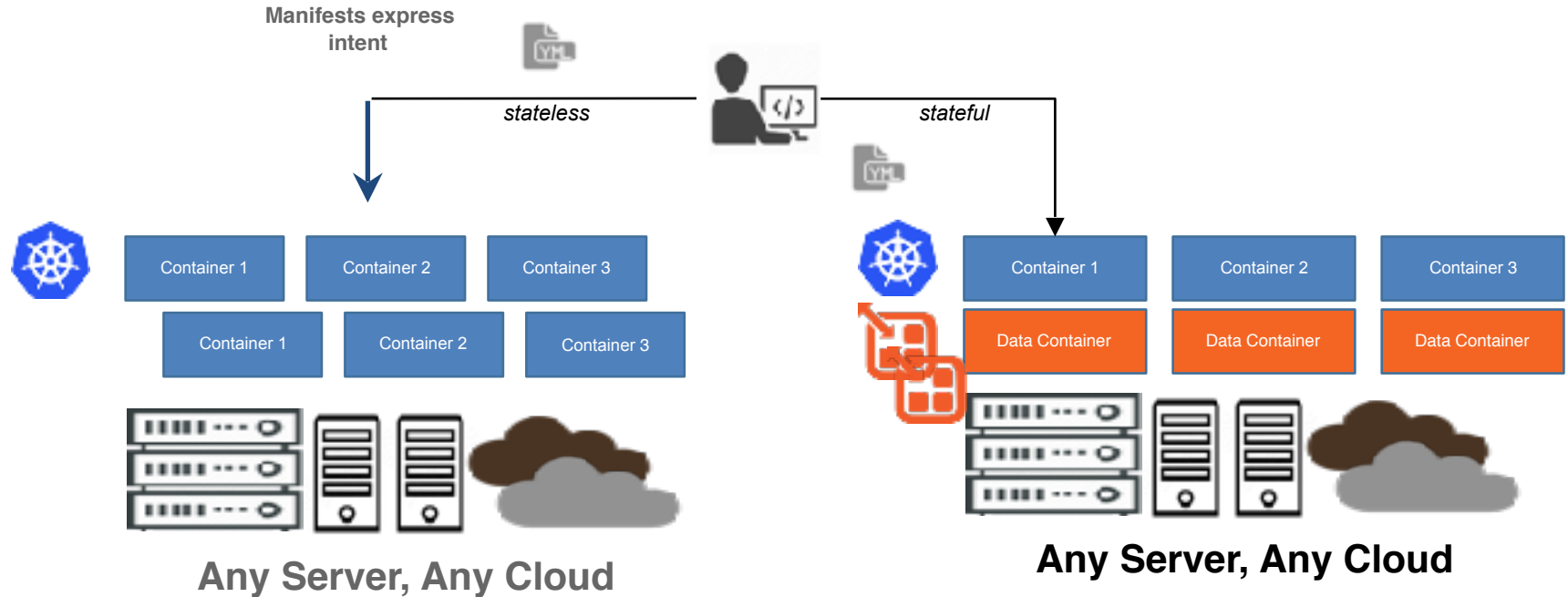
CREATE NEW A STORAGE SYSTEM

What if storage for containers native applications where was *itself* — container native?

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

State full workloads in the cloud

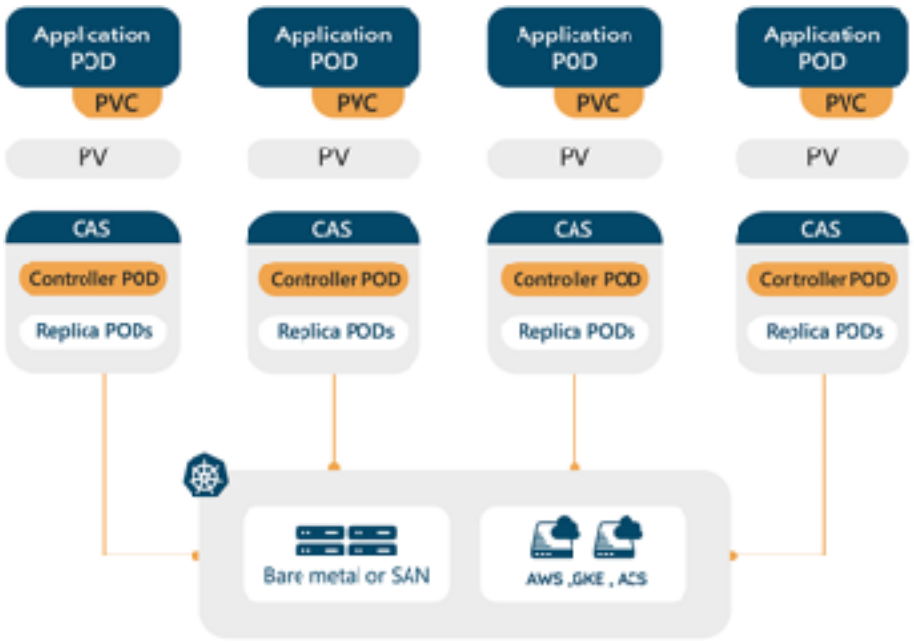


Challenge

- ❑ Small working sets
- ❑ Ephemeral
- ❑ Scale by N
- ❑ Mobile workloads
- ❑ DevOps responsible for operations
- ❑ Cloud lock-in
- ❑ Per workload optimisation

Solution

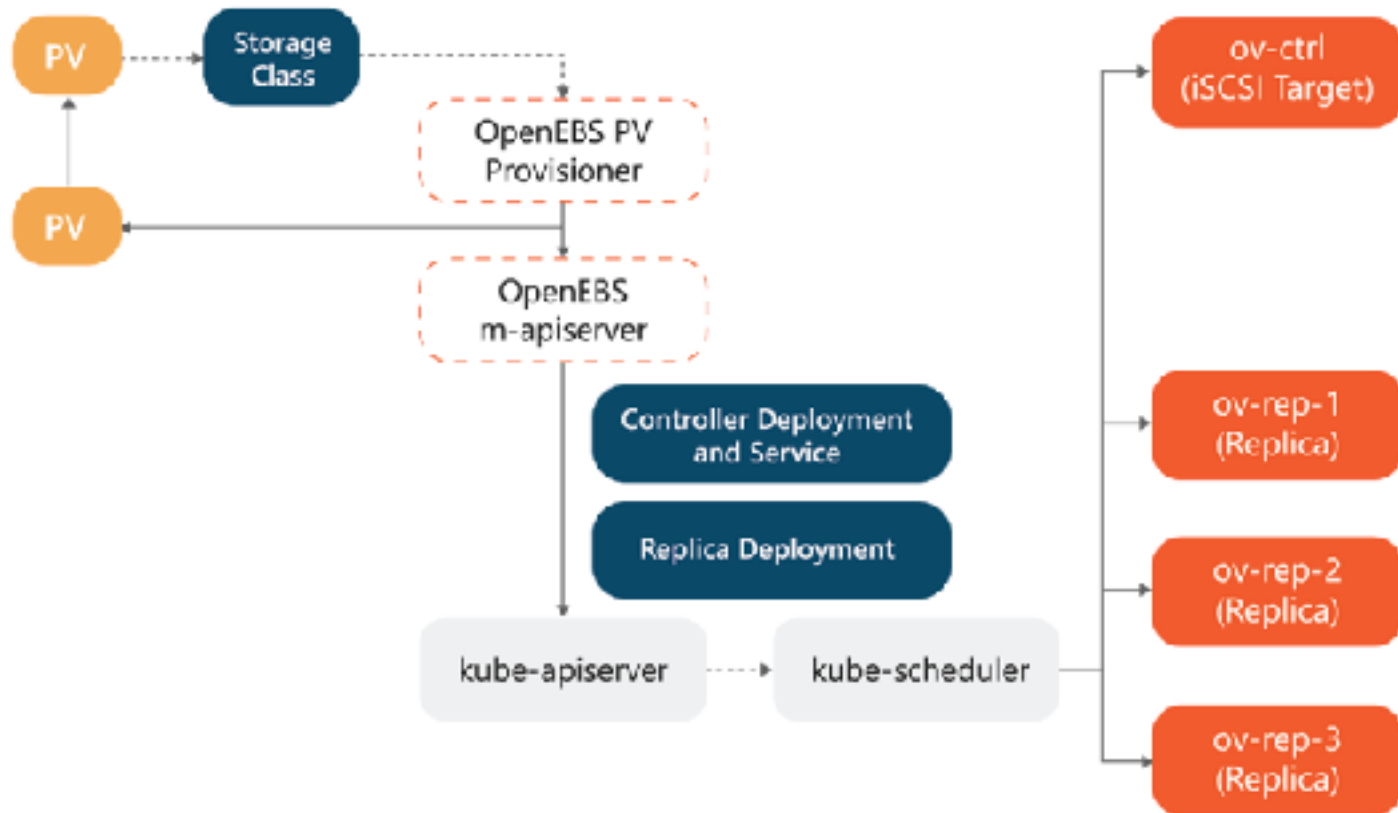
- ❑ Keep data local
- ❑ Controller in POD
- ❑ N more containers
- ❑ Follow the workload
- ❑ Just another micro service
- ❑ Workload migration
- ❑ Declarative intent





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) — what about SNIA swordfish?
- ❑ 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)



Provisioning flow in OpenEBS



38 nodes (34 filtered)

Show storage Hide storage

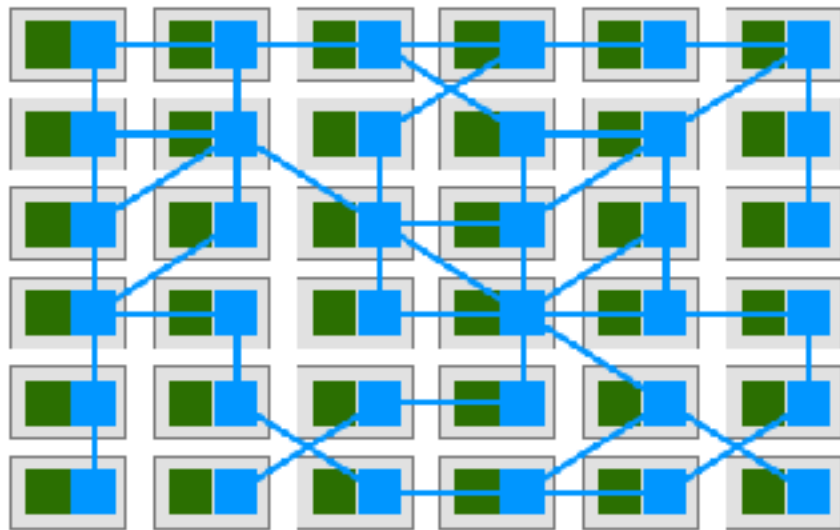
Show namespaces Hide namespaces

default kube-public kube-system openshift weaver AllNamespaces

WeaveScope v2.10.0 (2018-08-08) on Kubernetes v1.10.0 (2018-08-08) on Docker v1.13.0 (2018-08-08) on Linux v4.13.0 (2018-08-08) on x86_64

Data mesh

- ❑ 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



Data mesh negotiating

- ❑ 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

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

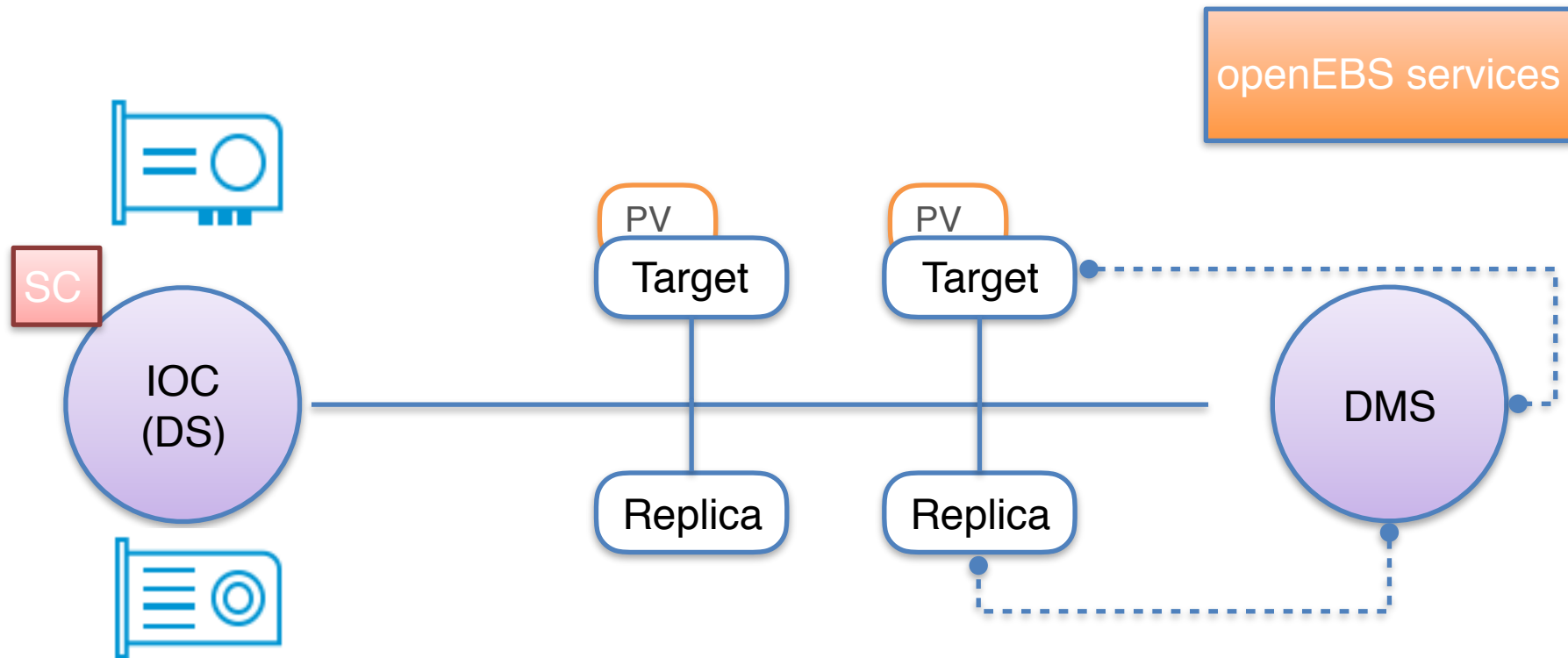
Storage just fades away as a concern

Implementation

- ❑ 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
- ❑ 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

- ❑ 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)

IOC overview

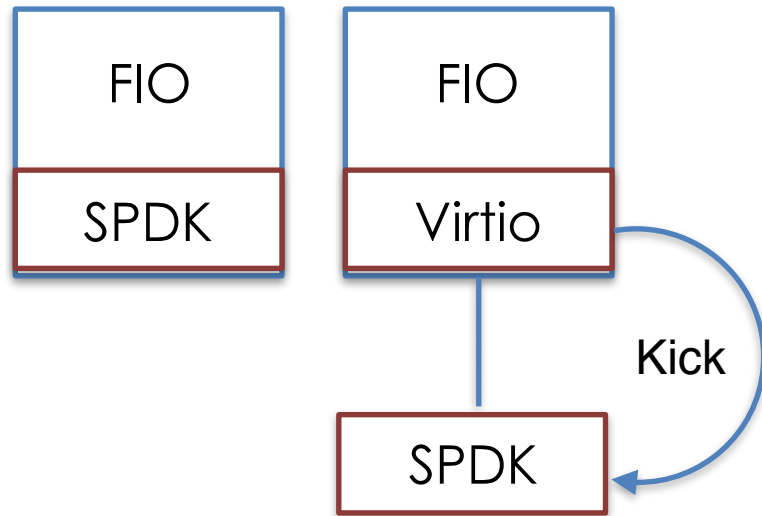


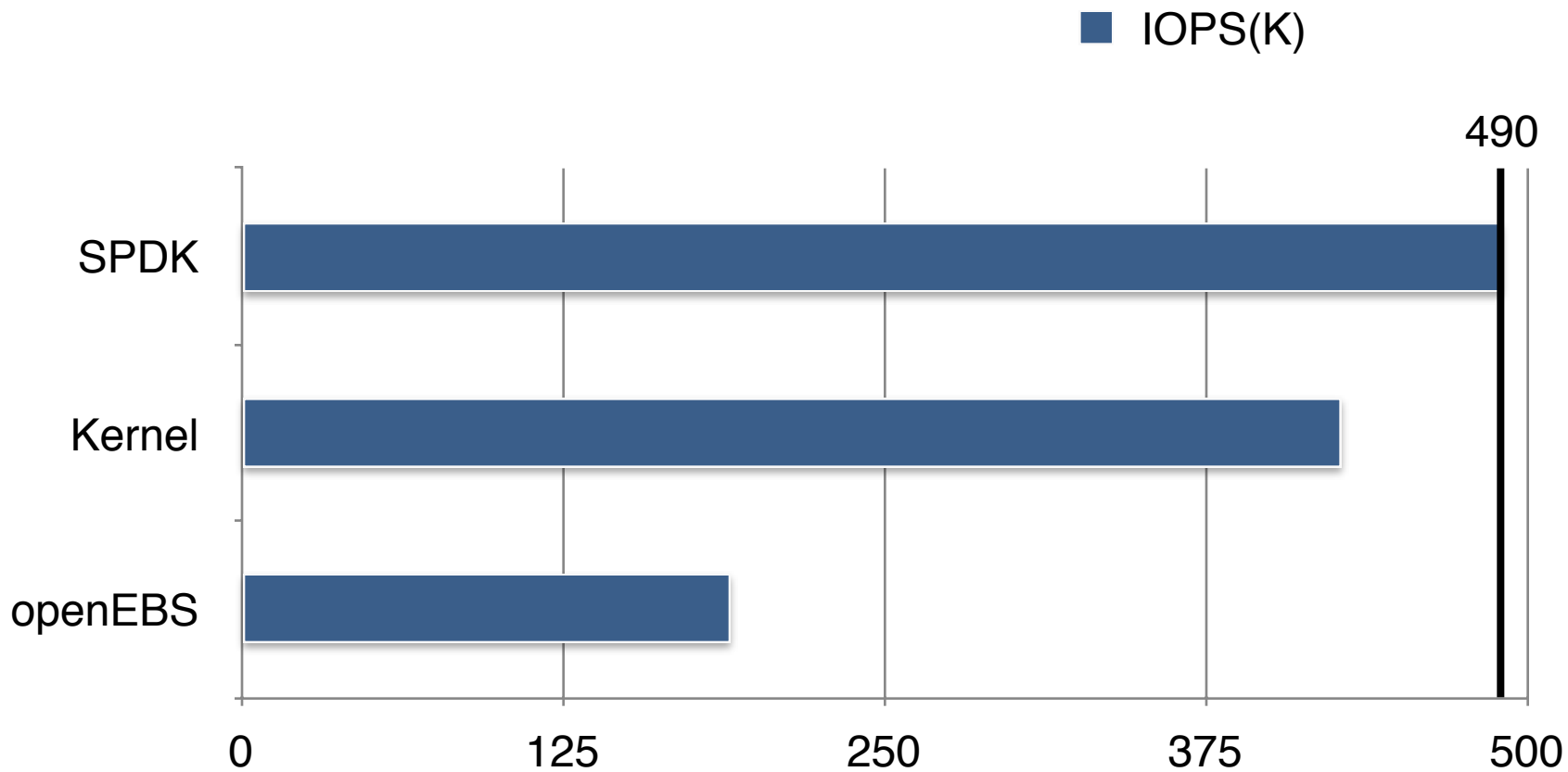
New old protocols

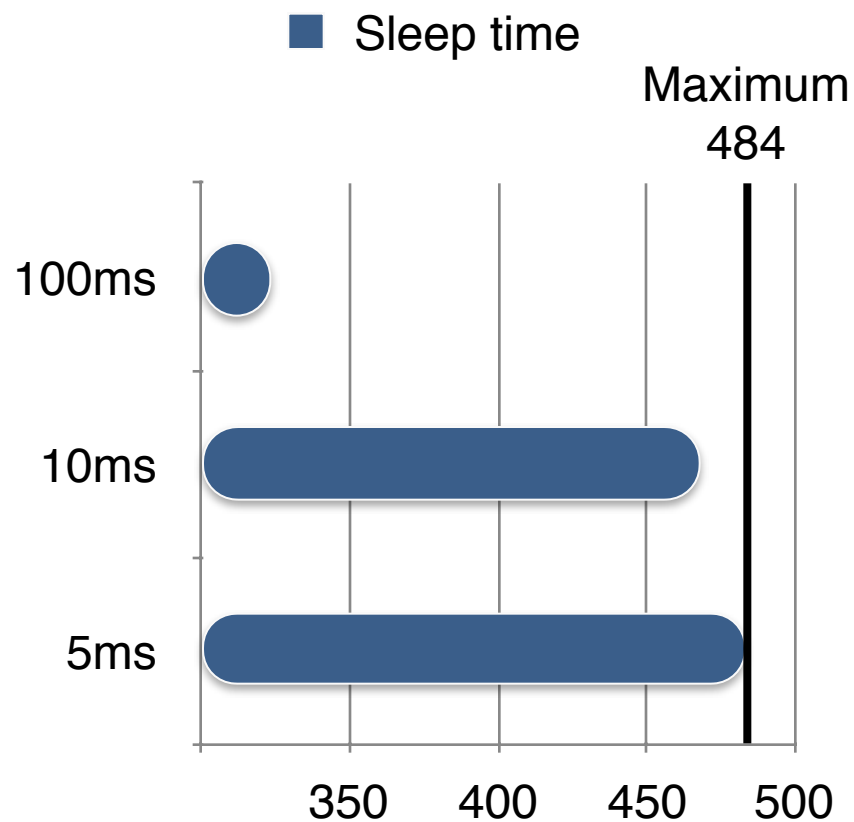
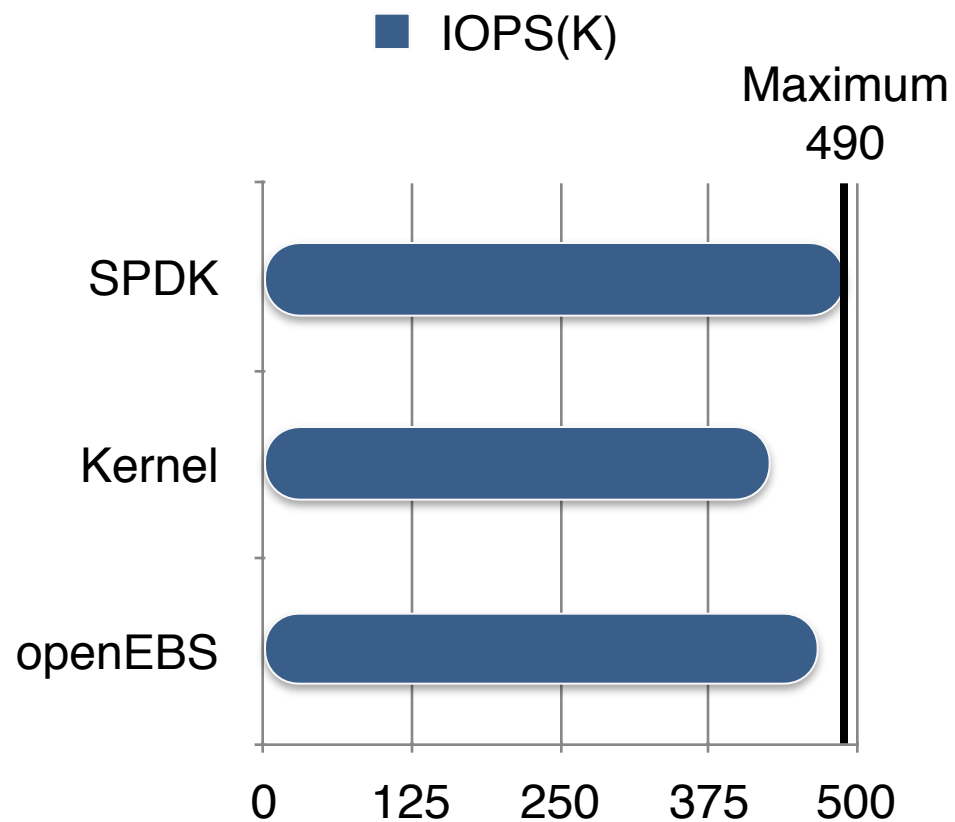
- ❑ Widely used and actively developed protocol which uses shared memory to interact with several types of hardware
- ❑ In the case of openEBS — interested in user space virtio-
{blk,nvme} initiator
- ❑ Primary reason for developing this was to have a loosely coupling with SPDK which use Poll Mode Drivers (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

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



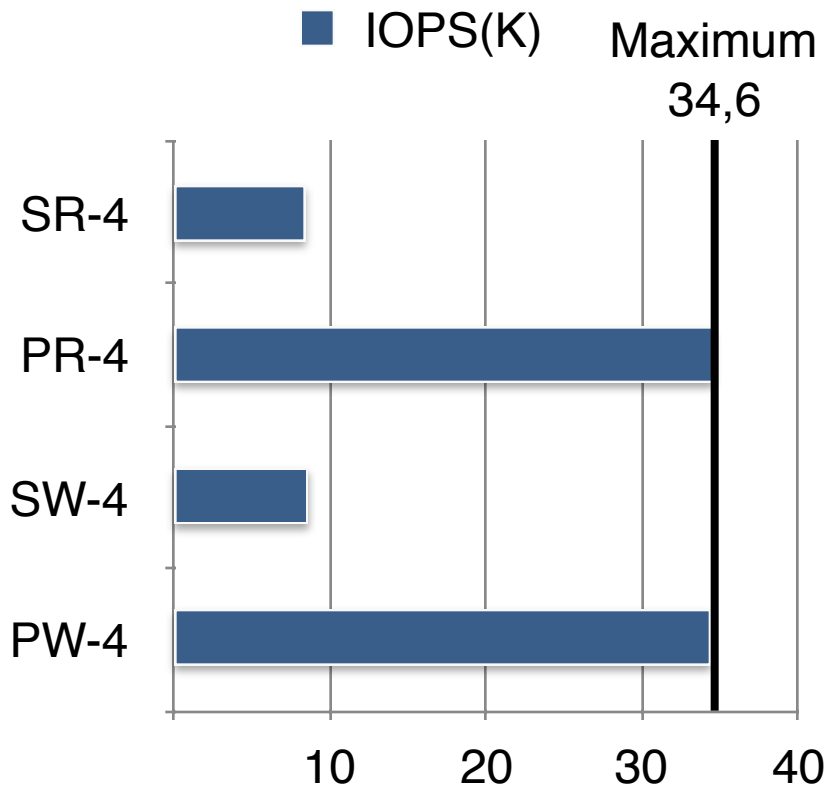




Observations

- ❑ 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
- ❑ Implement IO path — which **is single threaded and lockless**

Runtime abstraction gone wrong

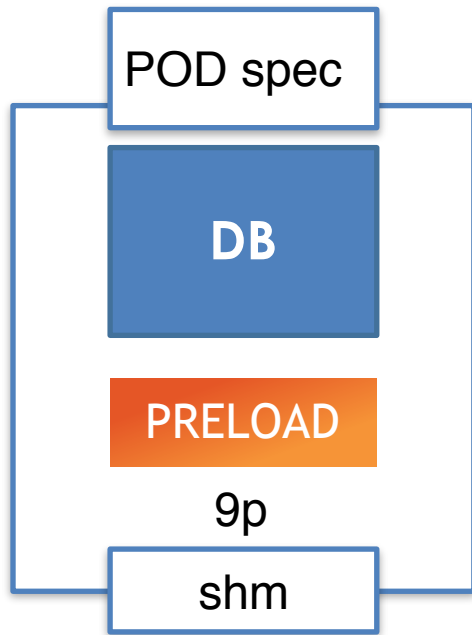


Other things we are looking into

- ❑ Most applications won't 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

- ❑ 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)



Summary

- ❑ 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

Questions

