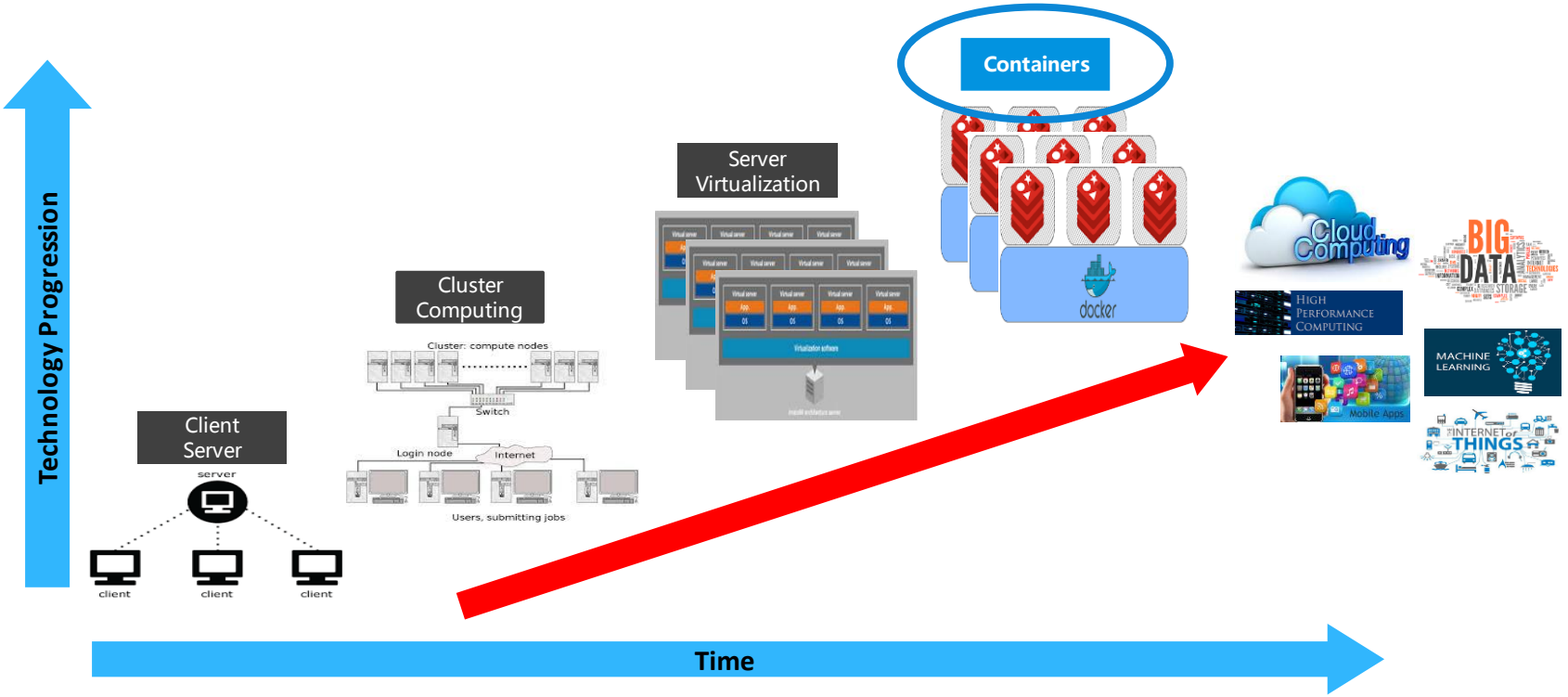
kycloud 天云软件 Software

Running Legacy Applications with Containers

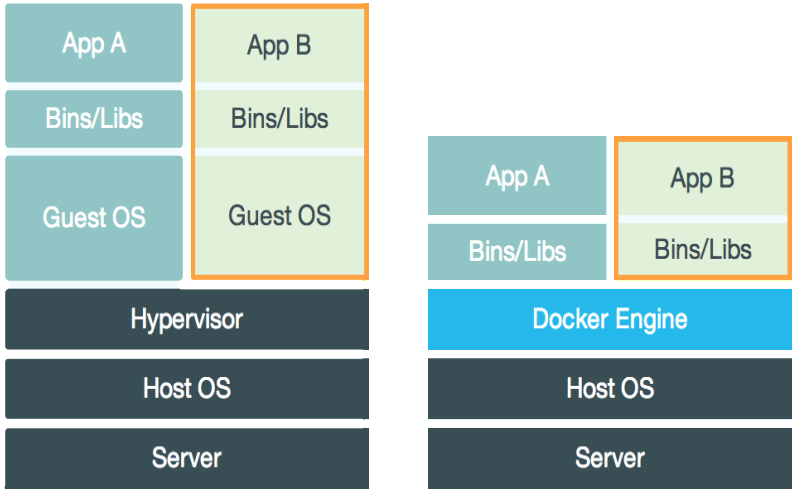
Niu, Jibin

Evolution of Distributed Computing



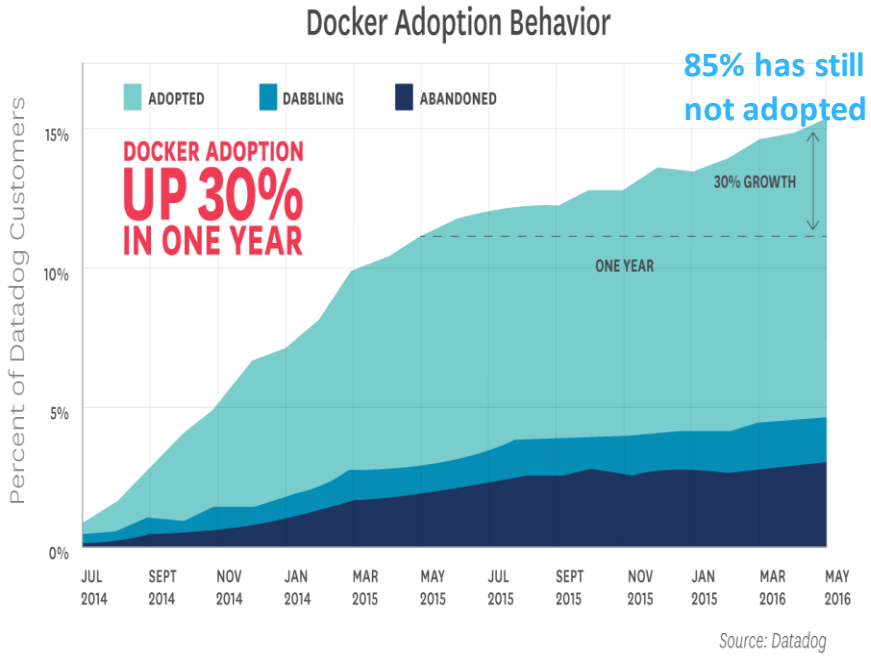
Docker Benefits

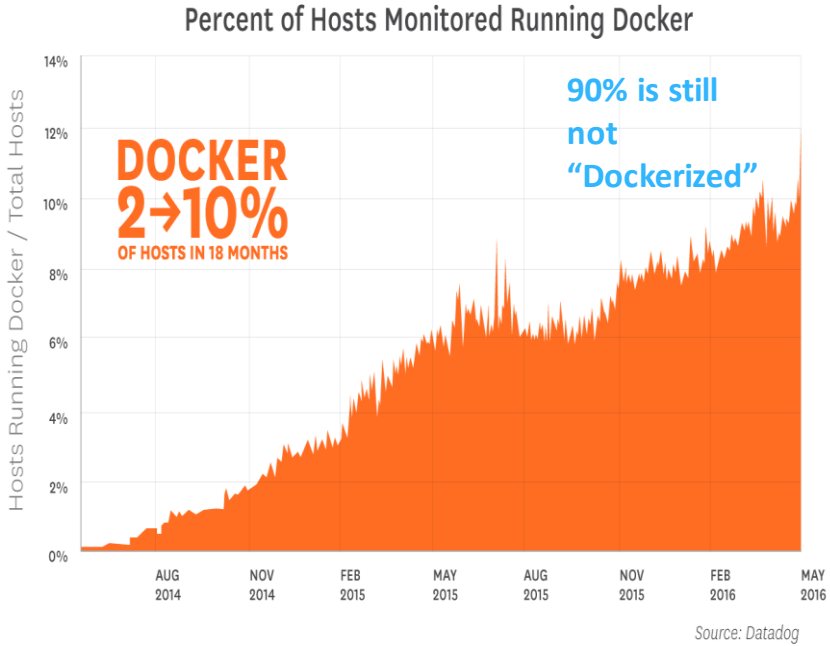
Virtual Machine vs Docker Container



- Empowers Developers
- Smaller Footprint
- High Performance
- Designed for Fastest Growing Workload Types
- “Good Enough” Isolation

Docker Adoption Growth





Sample of 10,000 companies tracking real usage (Ref: https://www.datadoghq.com/docker-adoption/)

Legacy Applications in Containers

Challenges

Network dependencies

- Static IP address requirement
- Hostname/MAC address dependent

Storage dependencies

- Persistent data on local file system

Security dependencies

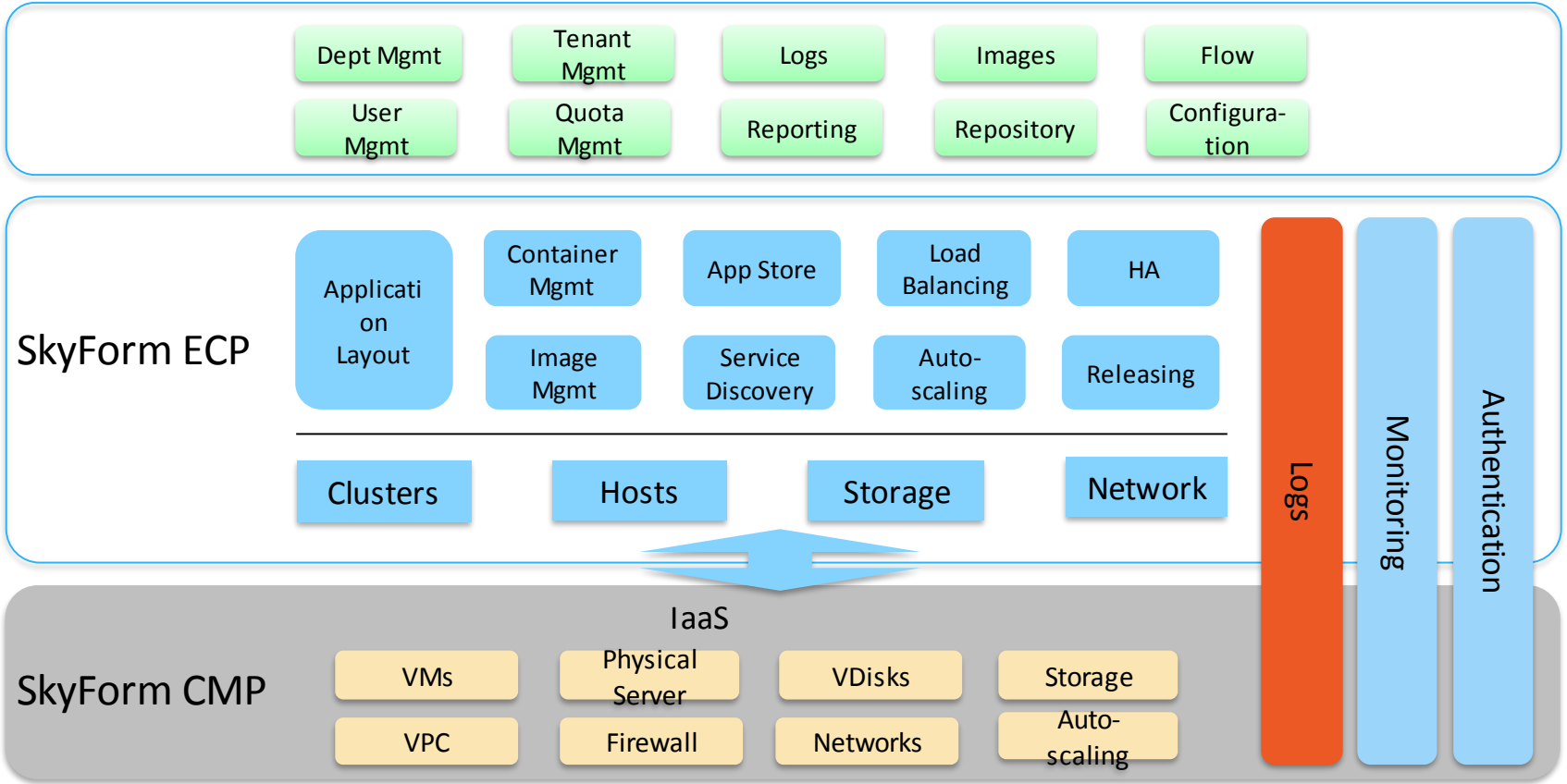
- Data security across tenants

Legacy Linux kernel requirements

Other OS, e.g. Windows

VM is still ideal to run legacy applications

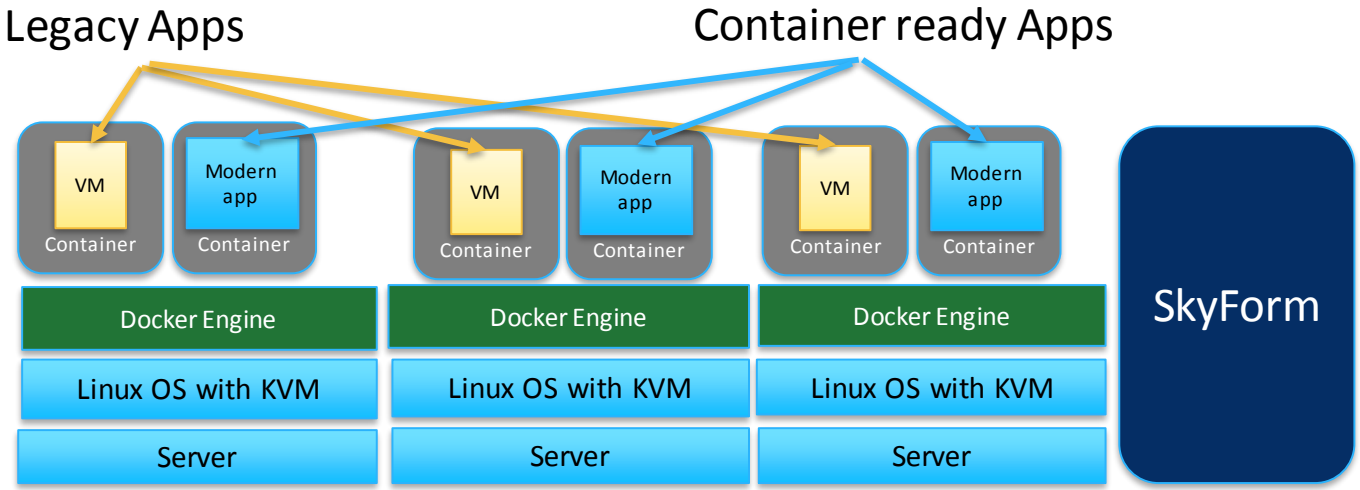
Introducing SkyForm



Fine Grain Shared Docker & VM Management

Running VM inside container

Limitations : Lack support for migration, snapshotting, etc.



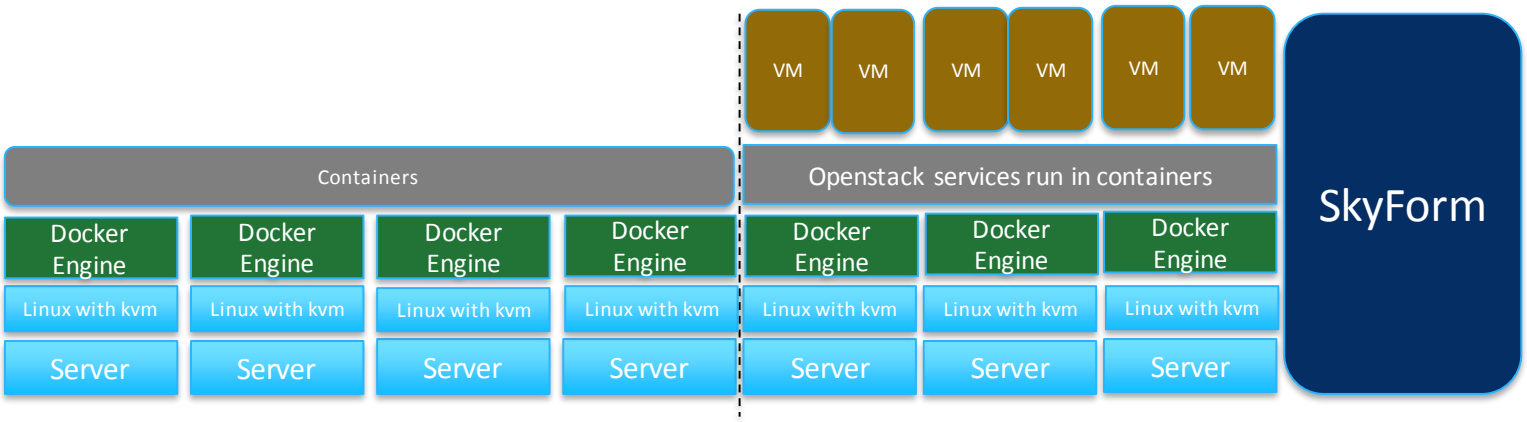
Model 2: Course Grain Shared Docker & VM Management:

The customer wants to keep their existing VM management framework, in particular, assume the existing VM environment is managed by OpenStack using KVM

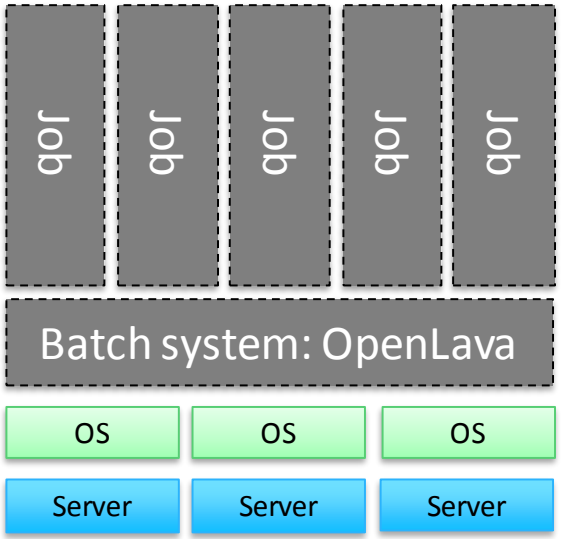
- Objective becomes how to improve utilization and QoS in a shared infrastructure

Architecture

- OpenStack itself is deployed and managed as one SkyForm ECP application where the OpenStack services run in docker containers
- SkyForm ECP allocates whole physical servers to OpenStack
- VMs are managed by OpenStack, and SkyForm ECP is not aware of VMs – VMs run on the hypervisor, not in containers



A Use Case: HPC Cluster



- HPC job could be serial or parallel. Parallel job could run across multiple hosts
- HPC cluster is typically used by multiple users.
- A supercomputing center may run jobs by multiple tenants. They share the same bare metal cluster environment
- The job management consists of
  - Job queues
  - Resource allocation policies
  - Job submission, monitoring and control

Docker-enabled HPC Cluster

Traditional HPC Cluster

HPC batch service & jobs run in processes

Docker-enabled HPC Cluster

- **Dockerized HPC compute nodes**
- **Run job from user supplied Docker image**
- **Run traditional "a.out" jobs**

Benefits

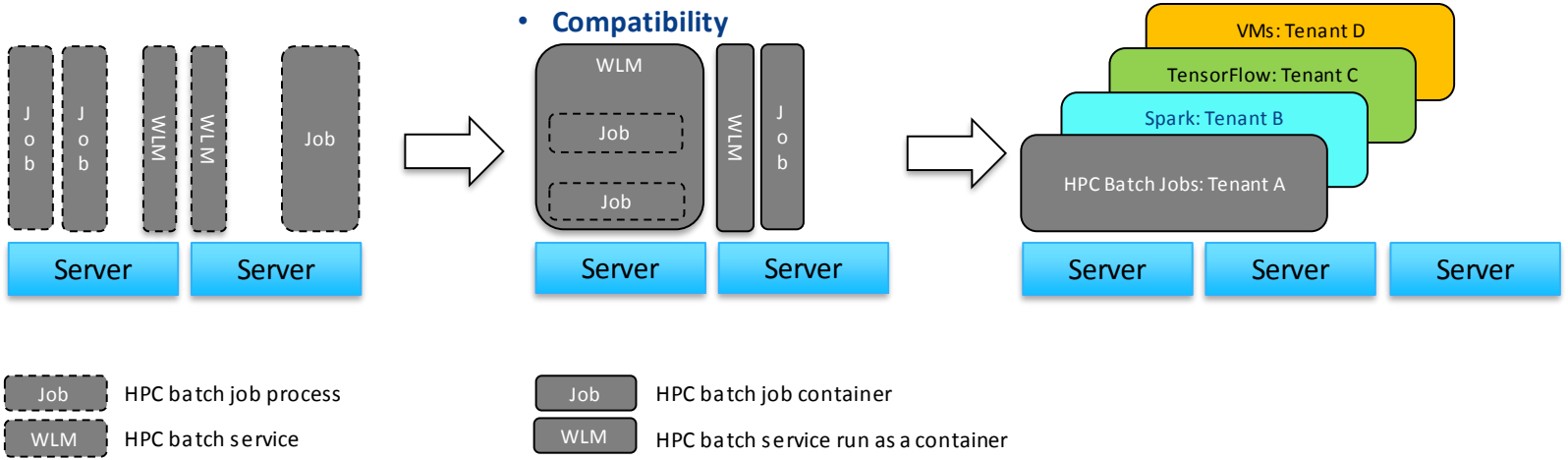
- **Application portability**
- **Isolation**
- **Compatibility**

Expansion of HPC Cluster

- **Big Data, ML, VM (legacy)**
- **Multi-tenant**
- **All workloads run in containers**

Benefits

- **Increased demand for HPC infrastructure**
- **Higher utilization**
- **Reduced IT costs**



Course Grain HPC Batch Sharing: Method 1

Sharing at the physical host boundary

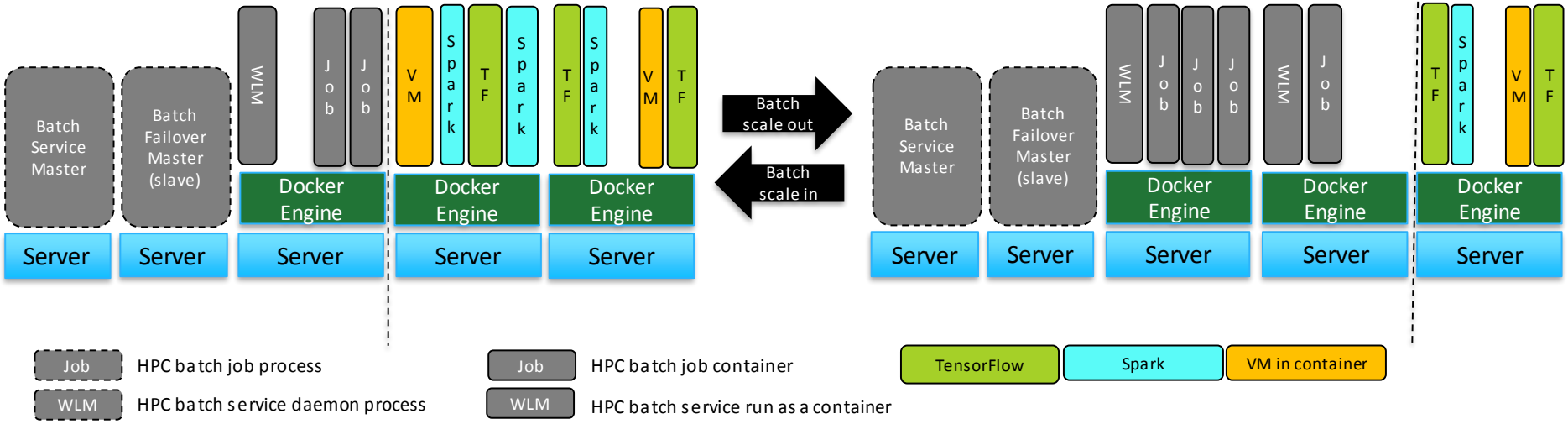
Batch jobs that are not provided as a container image are run inside a container created by the system

Pros

- Evolutionary extension of existing HPC cluster architecture
- Physical isolation between HPC batch tenant and non-HPC

Cons

- Lower utilization
- Dedicated master hosts
- Longer time to scale out – entire host must be vacated before providing to new tenant



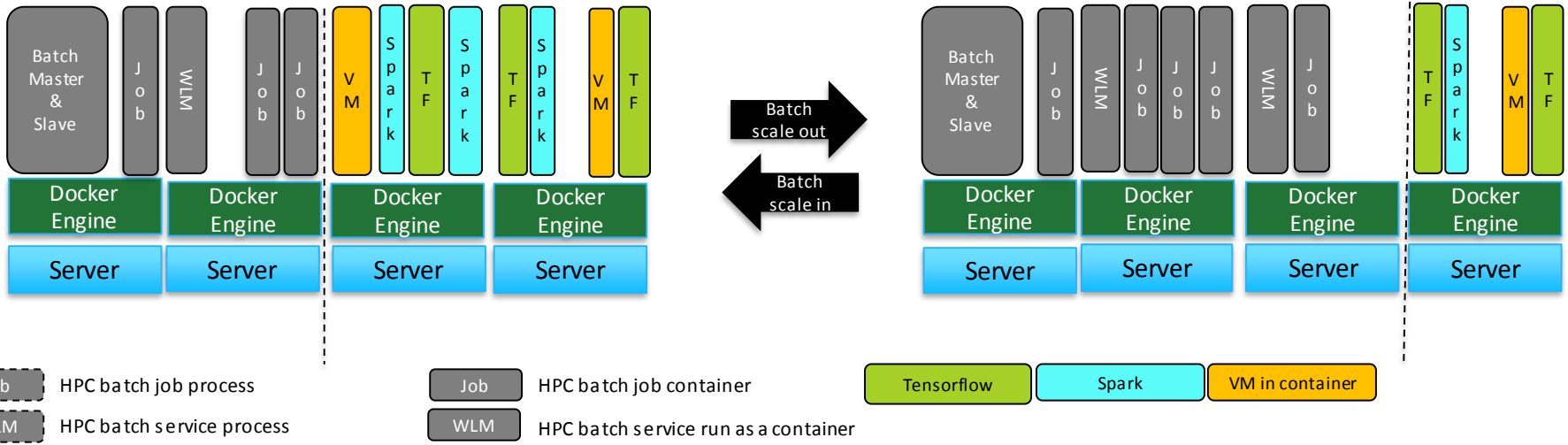
Course Grain HPC Batch Sharing: Method 2

Run Master & slave in a container
Leverages ECP for master HA
Pro

- Remove dedicated hosts
- Improved utilization compared with Model 1
- Can support multiple HPC Batch Service tenants

Con

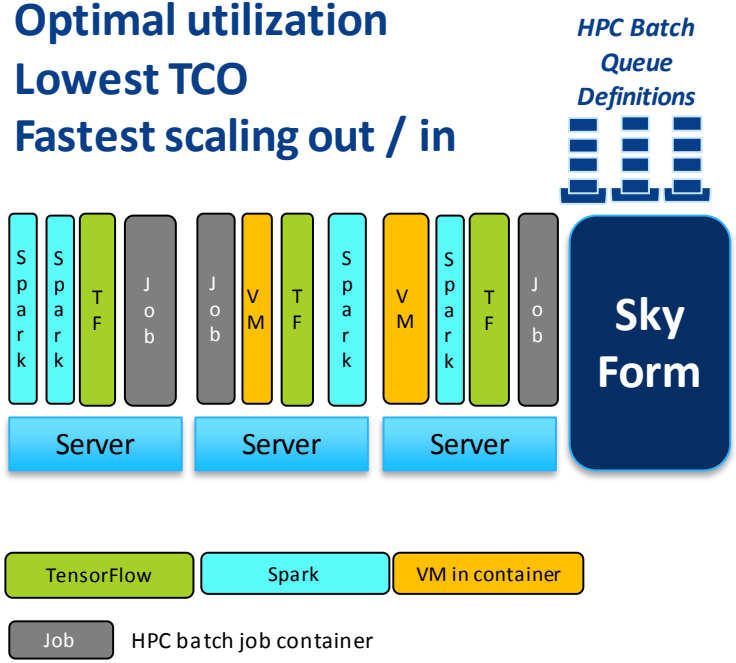
- Utilization is not optimal due to sharing at the physical host boundary



Fine Grain HPC Cluster Sharing Architecture

Built-in HPC Batch Service

- **Optimal utilization**
- **Lowest TCO**
- **Fastest scaling out / in**

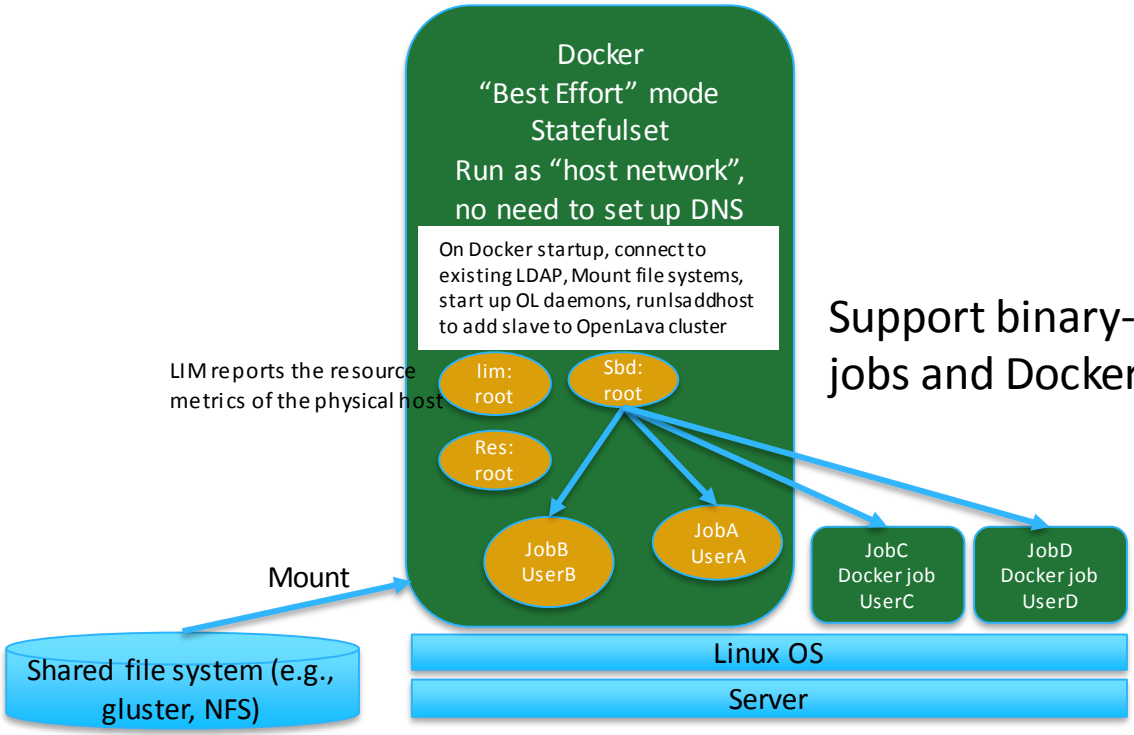


Each of HPC Batch Service, Spark, TensorFlow, all VMs in aggregate is a separate tenant

Sample Resource Plan (Algorithm) for Batch Service tenant

- Own: 50 CPUs, 100 GB memory
- Borrow: As many CPUs/memory as available with the understanding that a job running on borrowed resources must finished within ½ hour
- Lend: Lend up to 20 CPUs, 50 GB memory with the understanding that the loan resources will be returned by within 15 minutes after a reclaim request is made
- The number of pending jobs and the job characteristics / historical patterns are used to trigger scale up (reclaim, borrow), scale down (lend, return borrowed resources)

OpenLava Compute Server



Support binary-based (process) jobs and Docker based jobs

Using Kubernetes

SkyForm ECP (Enterprise Container Platform) leverages Kubernetes to manage container clusters

Challenge: How to share resources to ensure QoS among multiple tenants when there is contention for resources (i.e., all resources all used/reserved)?

Solution: Leverage traditional HPC job scheduling algorithm for scheduling containers

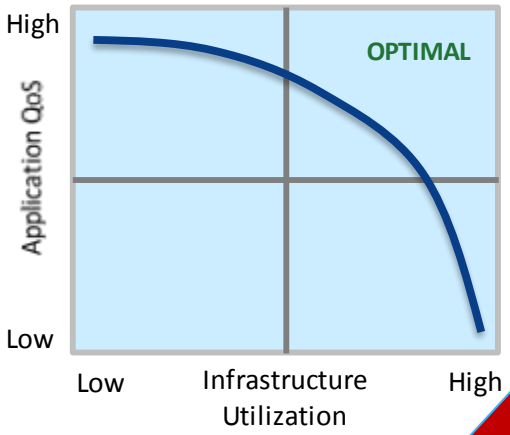
Scheduling Policies and QoS

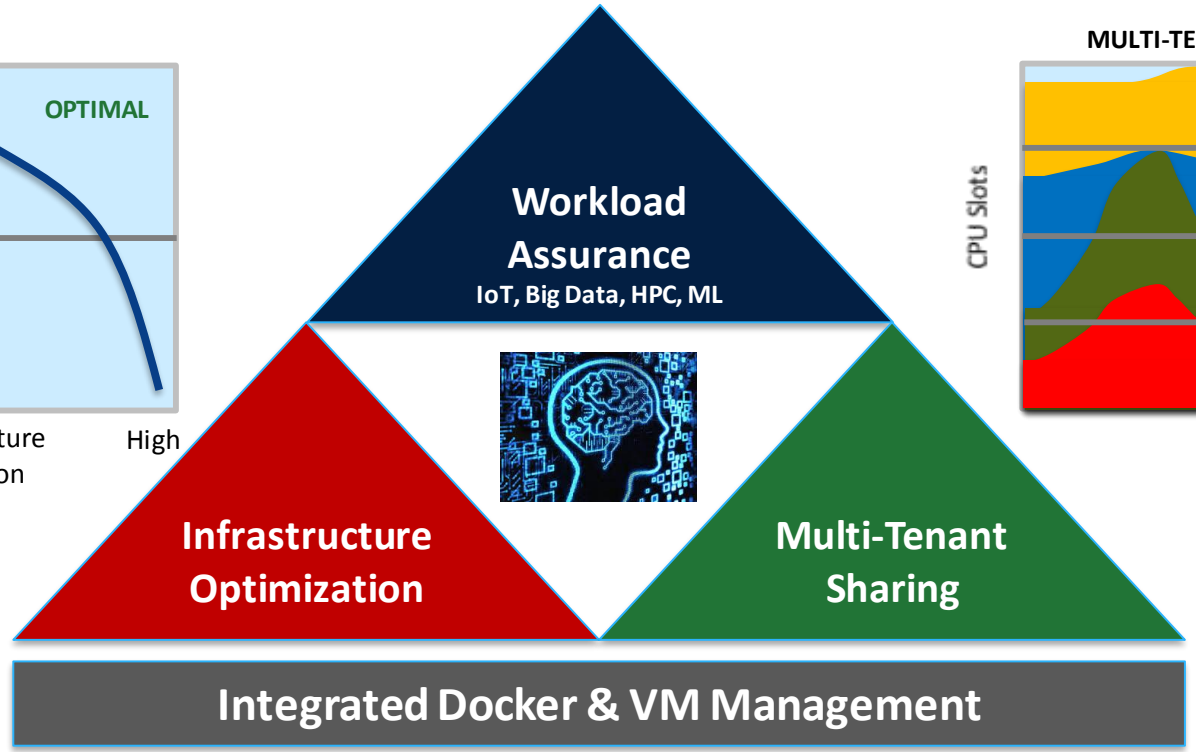
Policies	Priority
	Fairshare/Round Robin
	Preemption
	Ownership (private cloud)

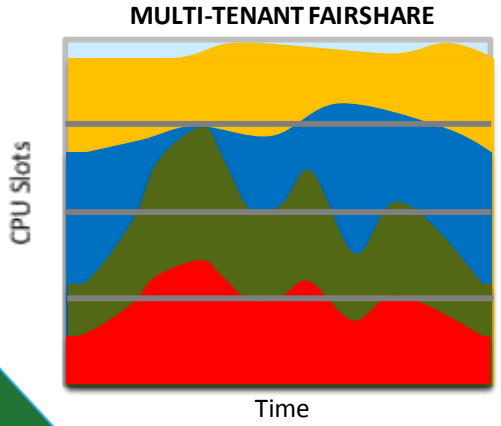
- Higher QoS users get
  - High priority
  - More shares
  - Being able to preempt low priority users' containers
  - More physical resource ownership
- Lower QoS users get
  - Low priority
  - Less shares
  - Containers could be preempted
  - Less or zero physical resource ownership

SkyForm Future Development

Algorithmic IT (AI) Enabled VM/Docker Cluster Manager







THANK YOU