KubeCon | CloudNativeCon

North America 2019

KubeCon | CloudNativeCon

North America 2019

Improving Performance of Deep Learning Workloads With Volcano

Ti Zhou, Baidu @tizhou86
Da Ma, Huawei @k82cn

Agenda

What Is
Baidu DLP

What Is
Volcano Project

Why Baidu DLP
Needs Volcano

How Baidu DLP
Leveraging Volcano

Baidu Deep Learning Platform

Unified Platform for ML/DL

- Data Processing
- Model Training/Serving
- 20+ Thousands Machines

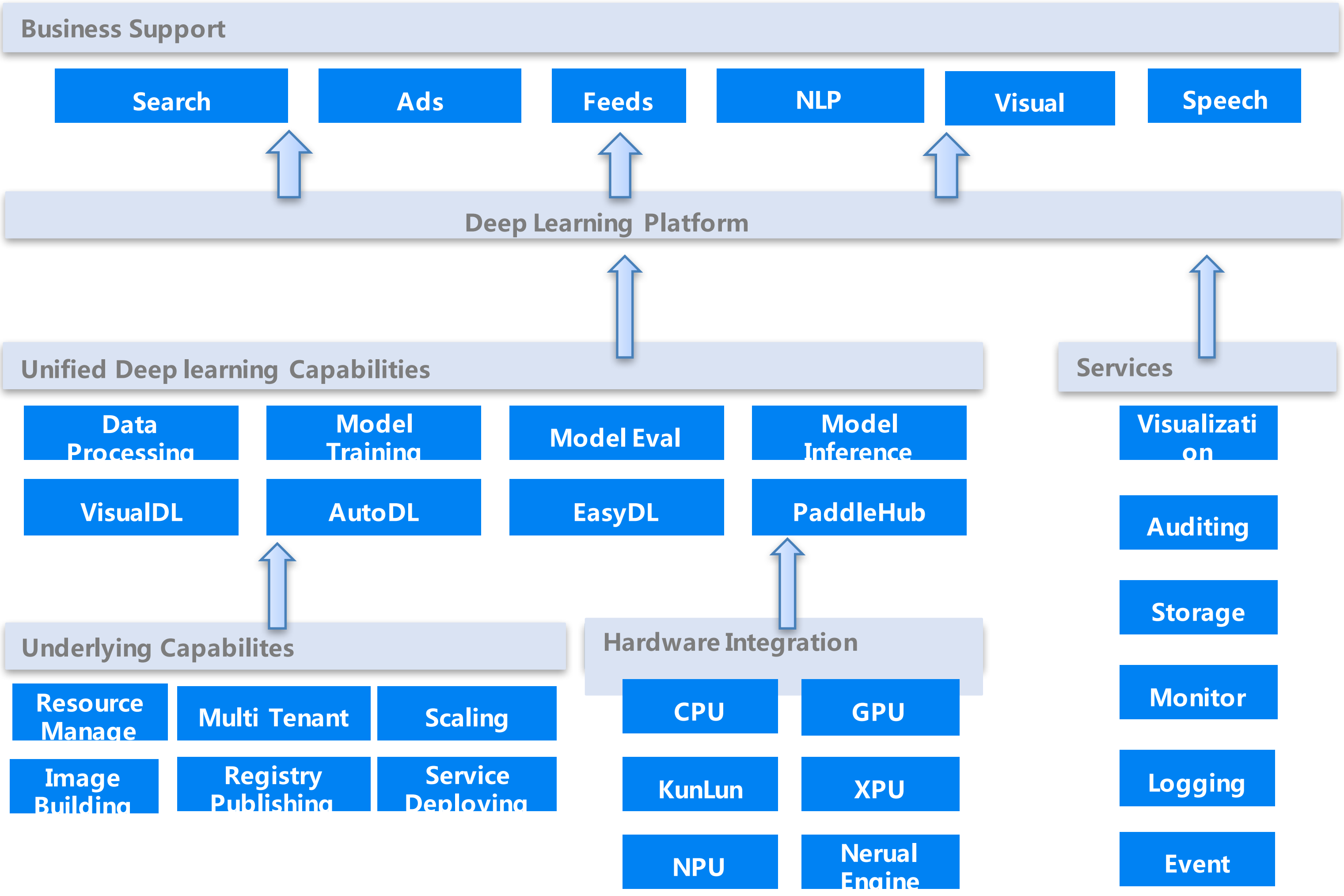
Advantage

- Resources Usage Optimization
- Model Pipeline
- Multi Tenant/Security

Business Support

- Search/Feeds
- Ads
- Autonomous Driving
- NLP/Visual/Speech

PaddlePaddle



Business Support

Search, Ads, Feeds, NLP, Visual, Speech

Deep Learning Platform

Unified Deep learning Capabilities

Data Processing, Model Training, Model Eval, Model Inference

VisualDL, AutoDL, EasyDL, PaddleHub

Services

Visualizati on, Auditing, Storage, Monitor, Logging, Event

Underlying Capabilites

Resource Manage, Multi Tenant, Scaling

Image Building, Registry Publishing, Service Deploying

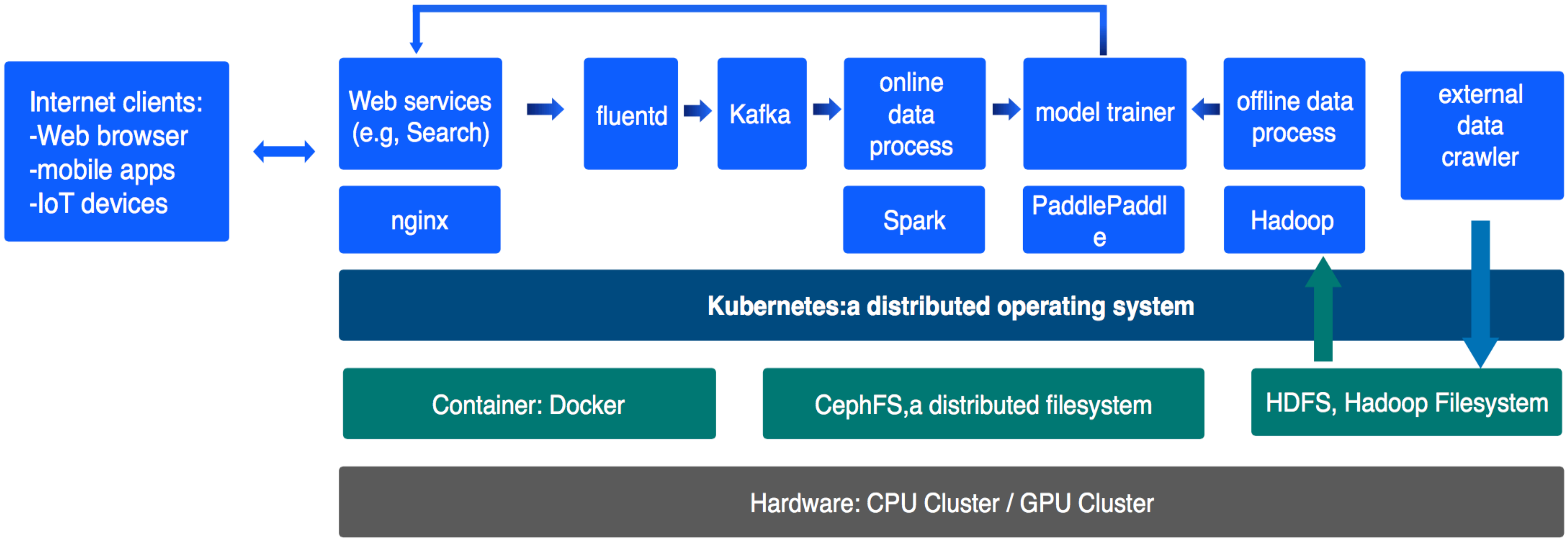
Hardware Integration

CPU, GPU

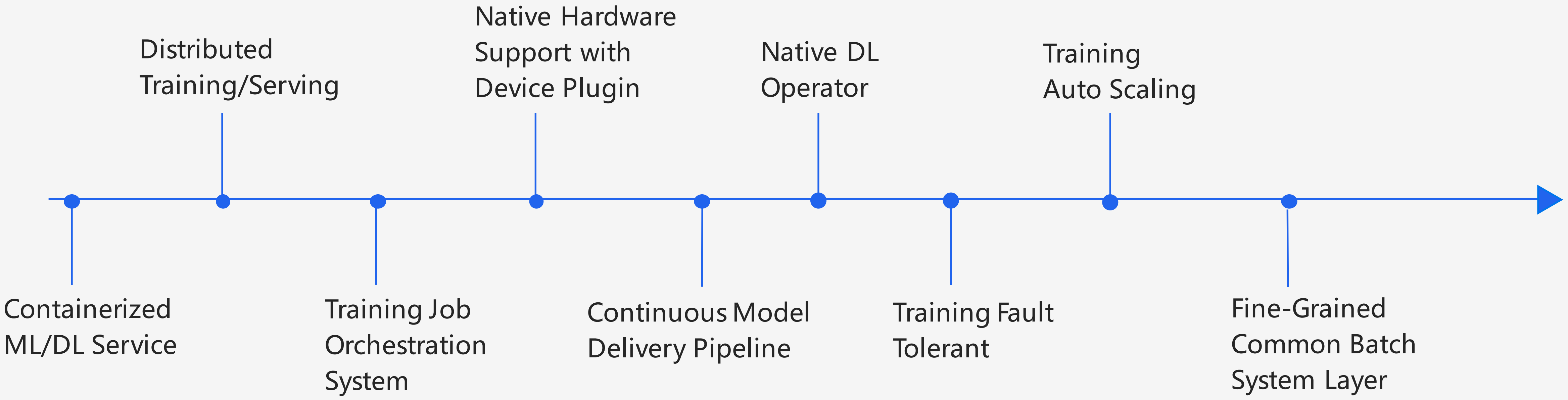
KunLun, XPU

NPU, Nerual Engine

Solution for modern AI Cluster



What’s Changed for DL Cluster



Containerized ML/DL Service

Distributed Training/Serving

Training Job Orchestration System

Native Hardware Support with Device Plugin

Continuous Model Delivery Pipeline

Native DL Operator

Training Fault Tolerant

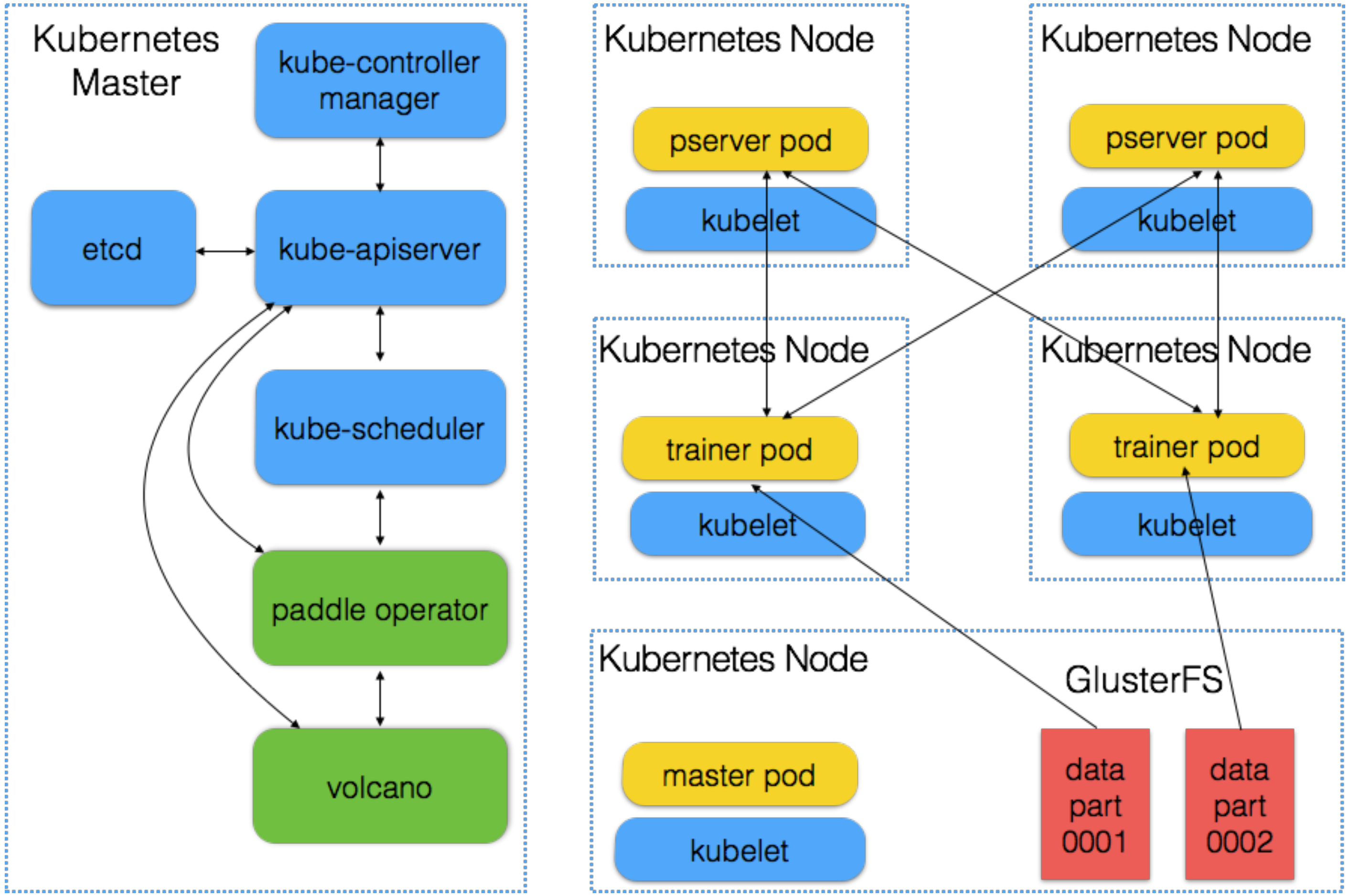
Training Auto Scaling

Fine-Grained Common Batch System Layer

PServer on K8s

Job Training Component:

- Paddle Operator (17-18)
  - DL Domain Technologies
  - DL Developer Self-Serving
  - Paddle Job CRD
  - Job Fault Tolerant
  - Job Auto Scaling
  - Modern Hardware Support
- Volcano (19)
  - K8s native batch system
  - Caring about HPC workloads
  - Ease to use



Paddle Operator Fault Tolerant

Master

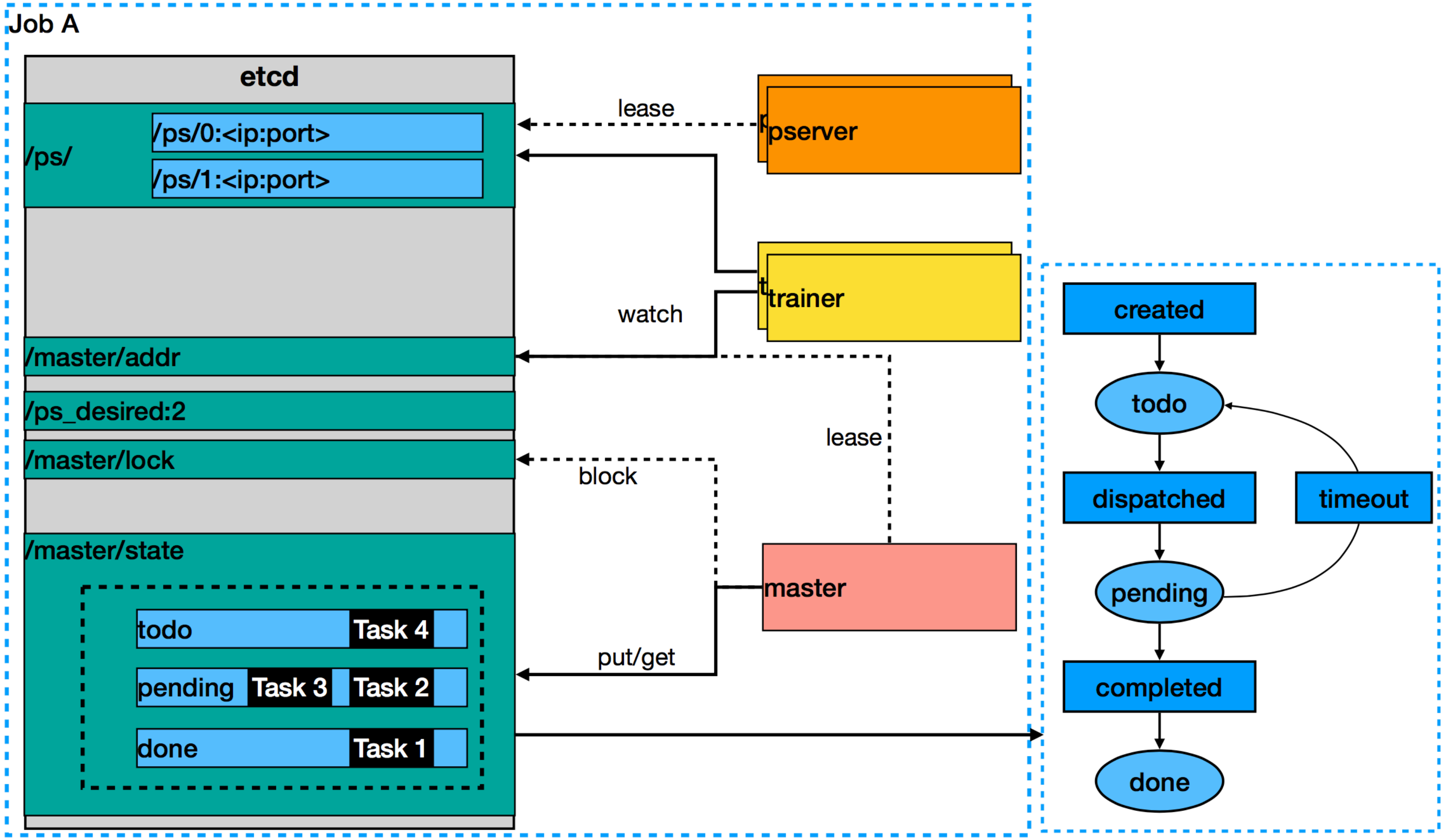
- Divide data
- Wrap up data to be task
- Assign the task to the trainer
- Check trainer health

Trainer

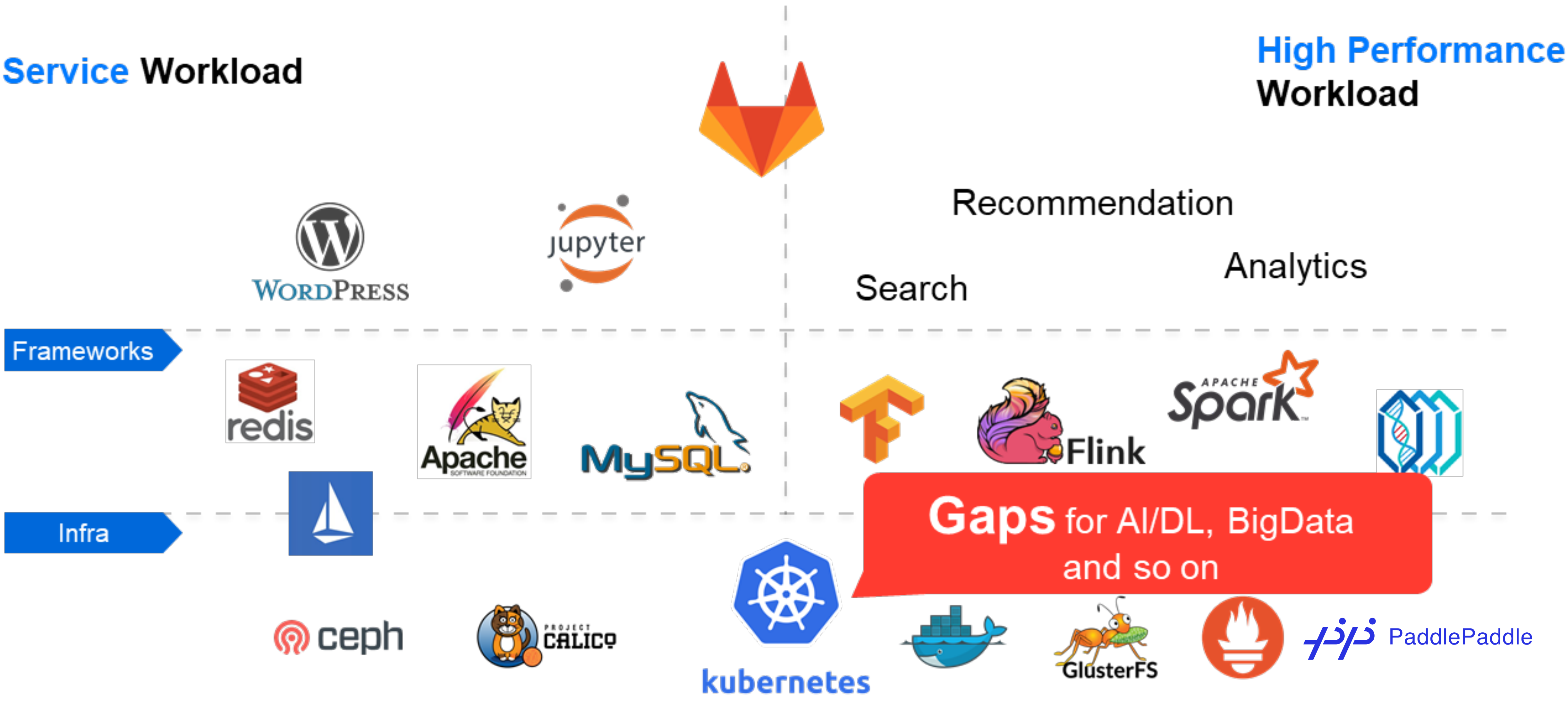
- Get task from master
- Calculate gradients

Parameter Server

- Get gradients from trainer
- Optimize and give feedback



What needs to be Improved



Service Workload

High Performance Workload

Recommendation

Search

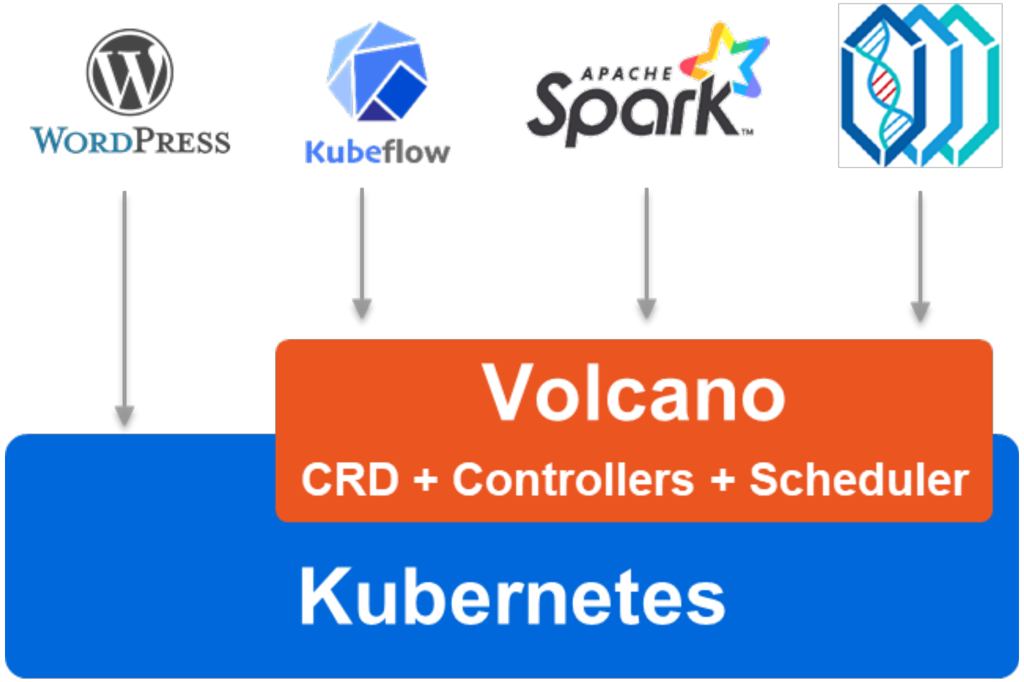
Analytics

Frameworks

Infra

Gaps for AI/DL, BigData and so on

Volcano Introduction



Volcano: A K8s Native Common Batch System

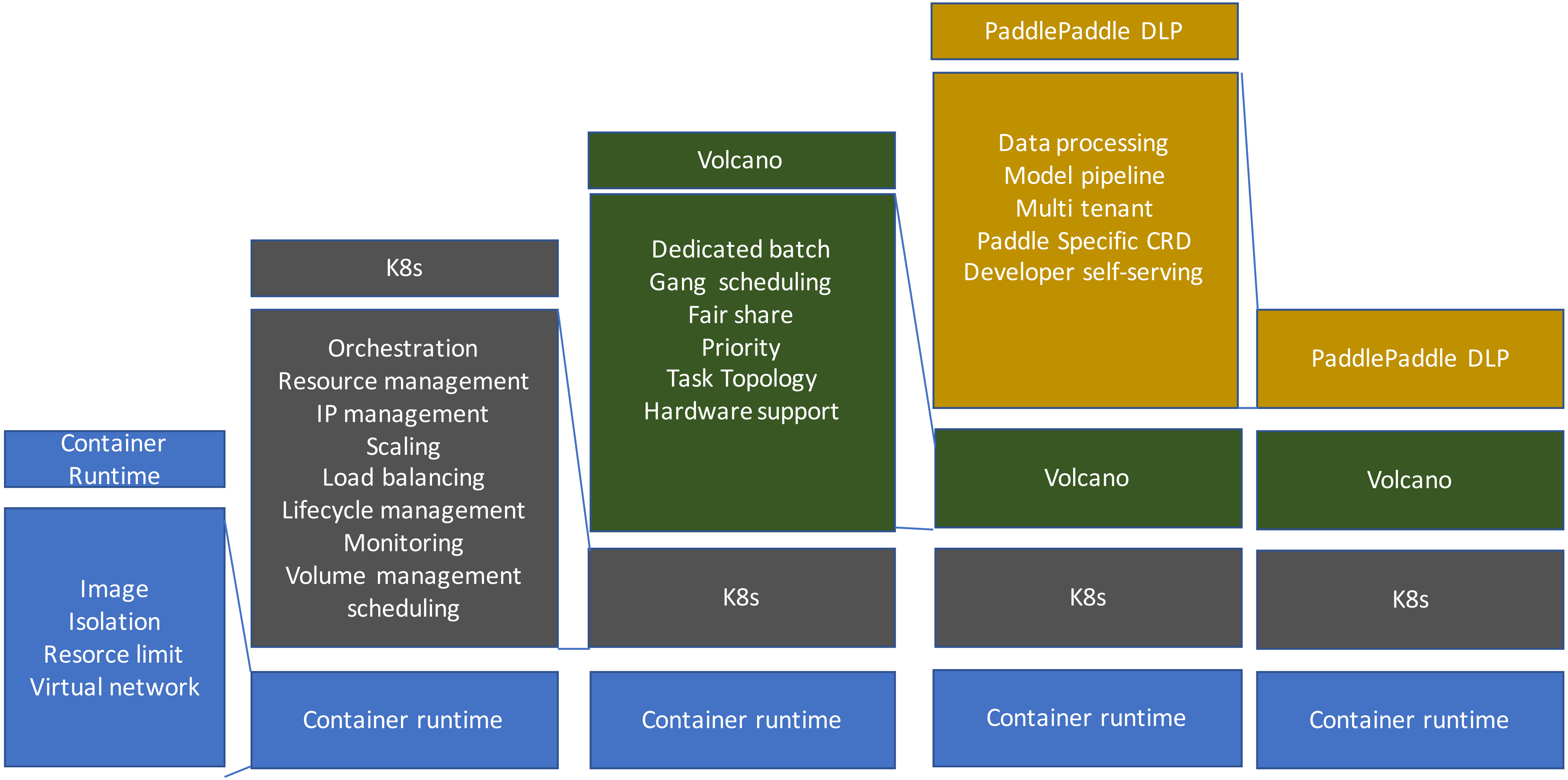
Domain frameworks:

- Provide specific DL/ML framework installation in K8s
- Map framework' s concepts into k8s native terms

Common Service for high performance workload:

- Batch scheduling, e.g. fair-share, gang-scheduling
- Enhanced job management, e.g. multiple pod template, error handling
- Common hardware accelerator, e.g. GPU, FPGA
- kubectl plugins, e.g. show Job/Queue information

How DLP leveraging Volcano



Container Runtime

Image
Isolation
Resorce limit
Virtual network

K8s

Orchestration
Resource management
IP management
Scaling
Load balancing
Lifecycle management
Monitoring
Volume management
scheduling

Container runtime

Volcano

Dedicated batch
Gang scheduling
Fair share
Priority
Task Topology
Hardware support

K8s

Container runtime

PaddlePaddle DLP

Data processing
Model pipeline
Multi tenant
Paddle Specific CRD
Developer self-serving

Volcano

K8s

Container runtime

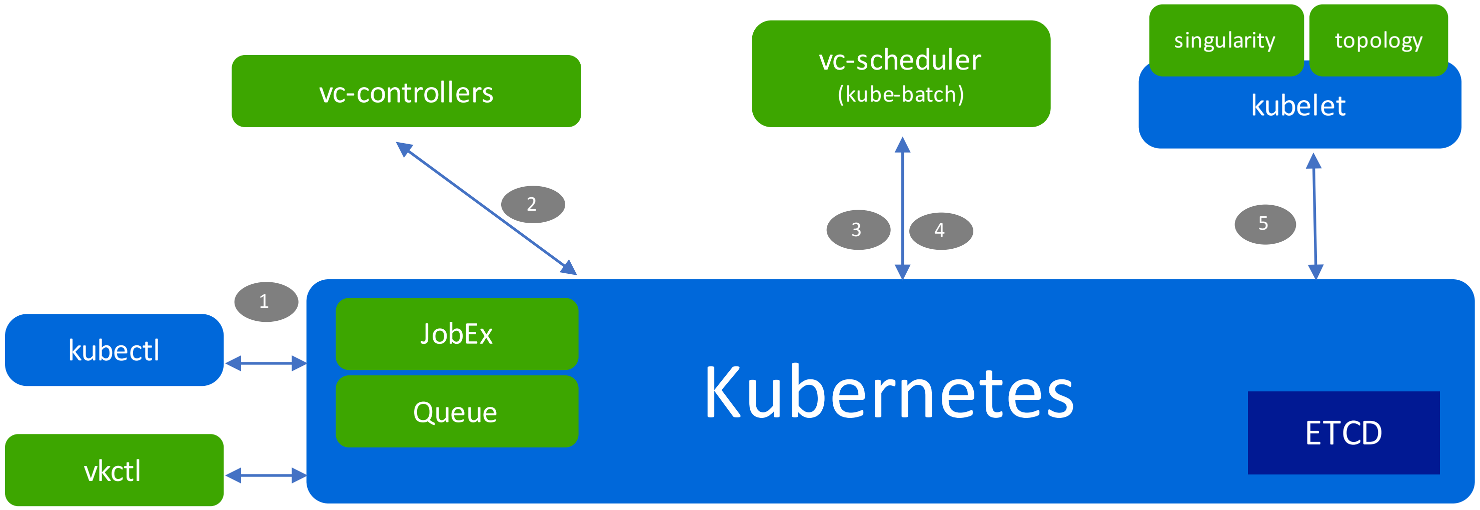
PaddlePaddle DLP

Volcano

K8s

Container runtime

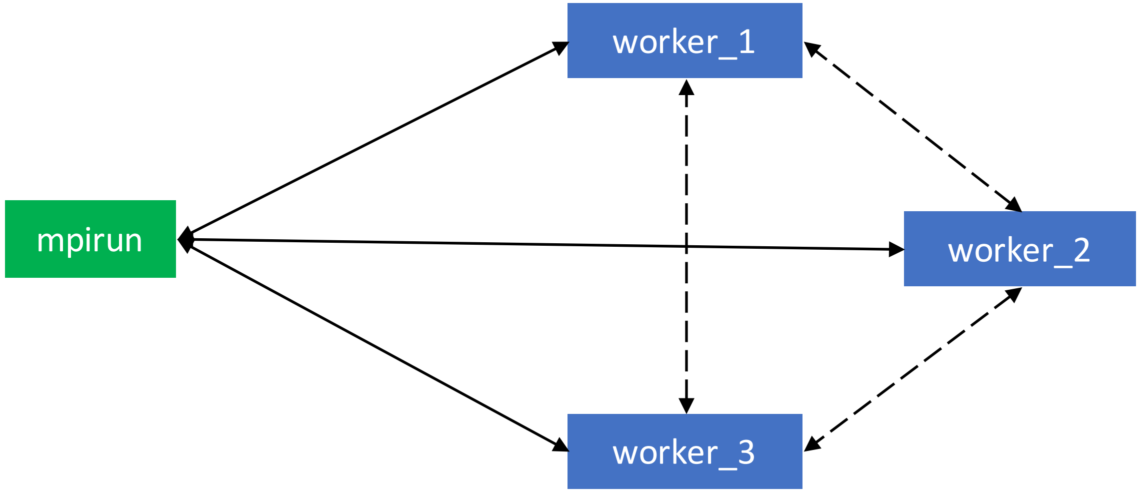
Volcano Architecture

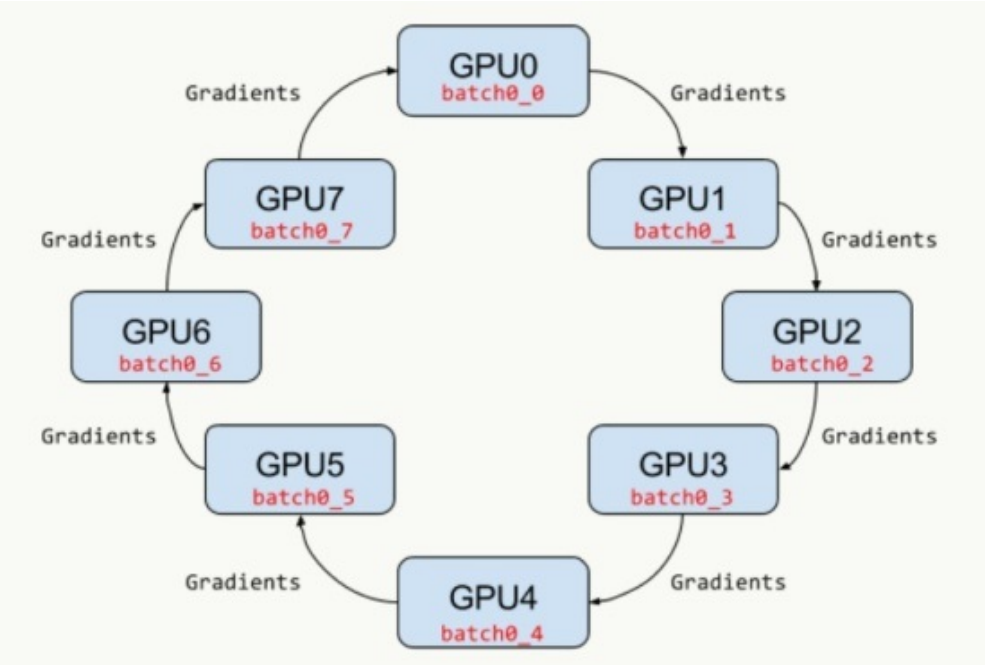


- The policy in **vc-scheduler** is pluggable, e.g. DRF, Priority, Gang
- **vc-controllers** includes **JobExController**, **QueueController**

- Kubectl creates a ***JobEx*** object in apiserver if all admission passed
- ***JobExController*** *create Pods based on its replicas and templates*
- **vc-scheduler** get the notification of Pod from apiserver
- **vc-scheduler** chooses one host for the Pod of ***JobEx*** based on its policy
- kubelet gets the notification of Pod from apiserver; and then start the container

Scenarios: MPI

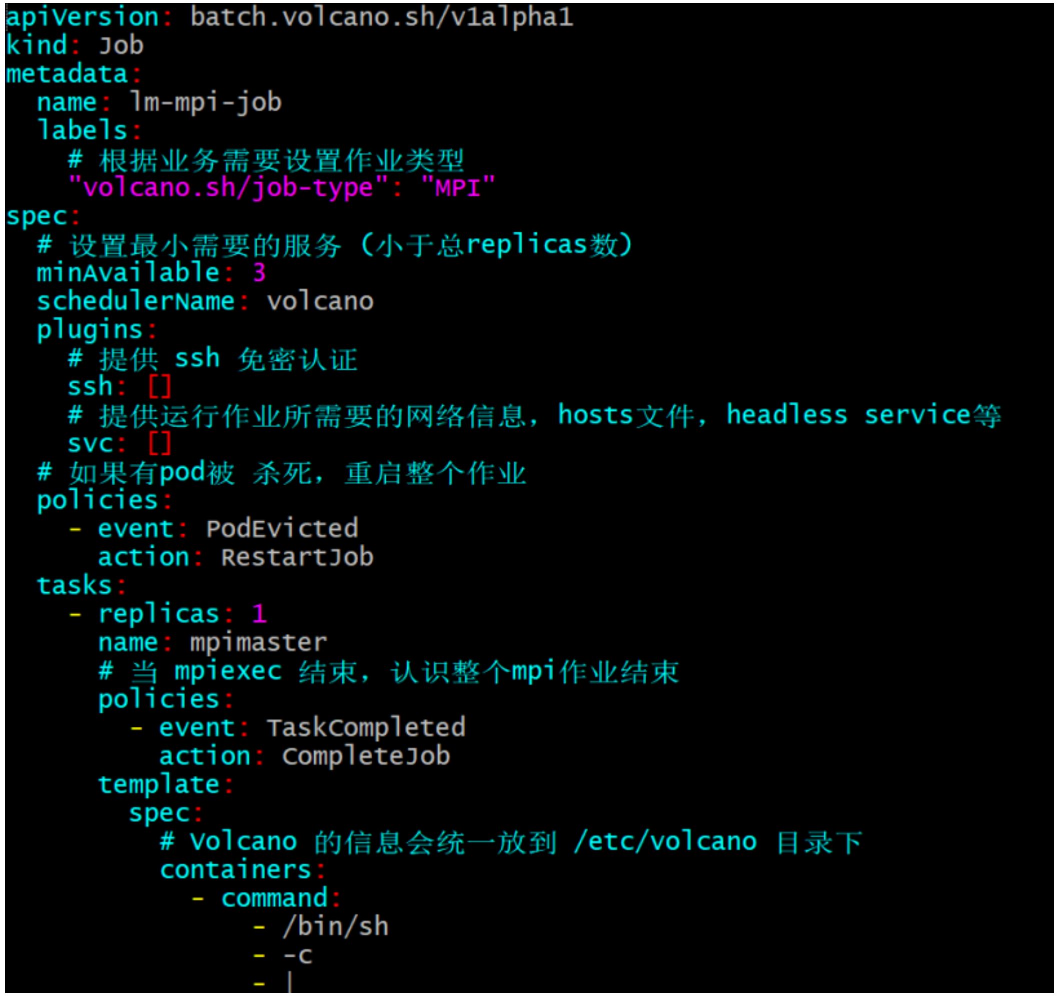




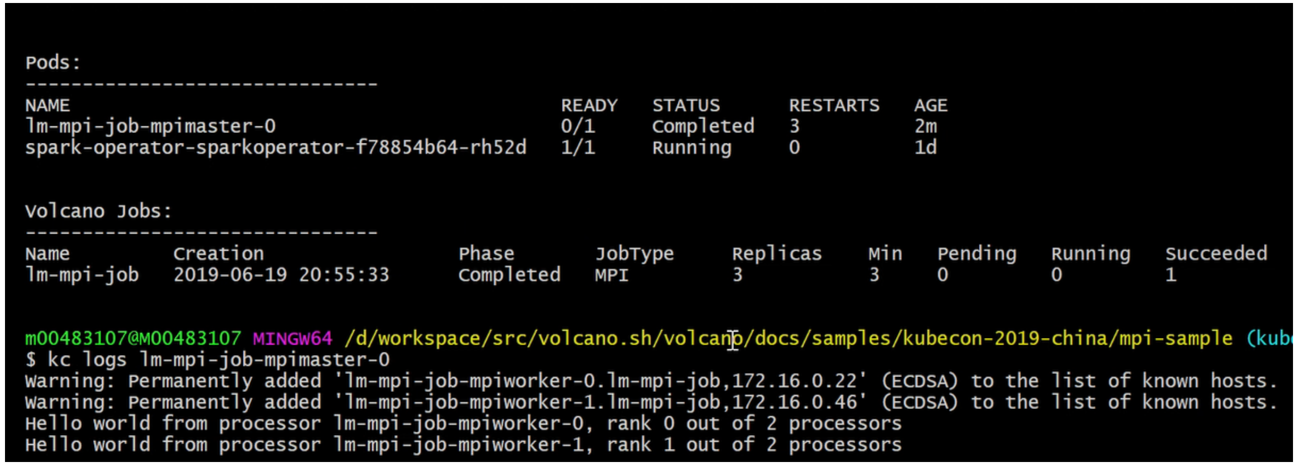
- Multiple Pod Template
- Lifecycle Policy
- Gang-scheduling

- ssh or kubectl
- Complete job when mpirun completed
- Headless service

Scenarios: MPI



```
apiVersion: batch.volcano.sh/v1alpha1
kind: Job
metadata:
  name: lm-mpi-job
  labels:
    # 根据业务需要设置作业类型
    "volcano.sh/job-type": "MPI"
spec:
  # 设置最小需要的服务 (小于总replicas数)
  minAvailable: 3
  schedulerName: volcano
  plugins:
    # 提供 ssh 免密认证
    ssh: []
    # 提供运行作业所需要的网络信息, hosts文件, headless service等
    svc: []
  # 如果有pod被 杀死, 重启整个作业
  policies:
    - event: PodEvicted
      action: RestartJob
  tasks:
    - replicas: 1
      name: mpimaster
      # 当 mpiexec 结束, 认识整个mpi作业结束
      policies:
        - event: TaskCompleted
          action: CompleteJob
      template:
        spec:
          # Volcano 的信息会统一放到 /etc/volcano 目录下
          containers:
            - command:
                - /bin/sh
                - -c
                - |
```

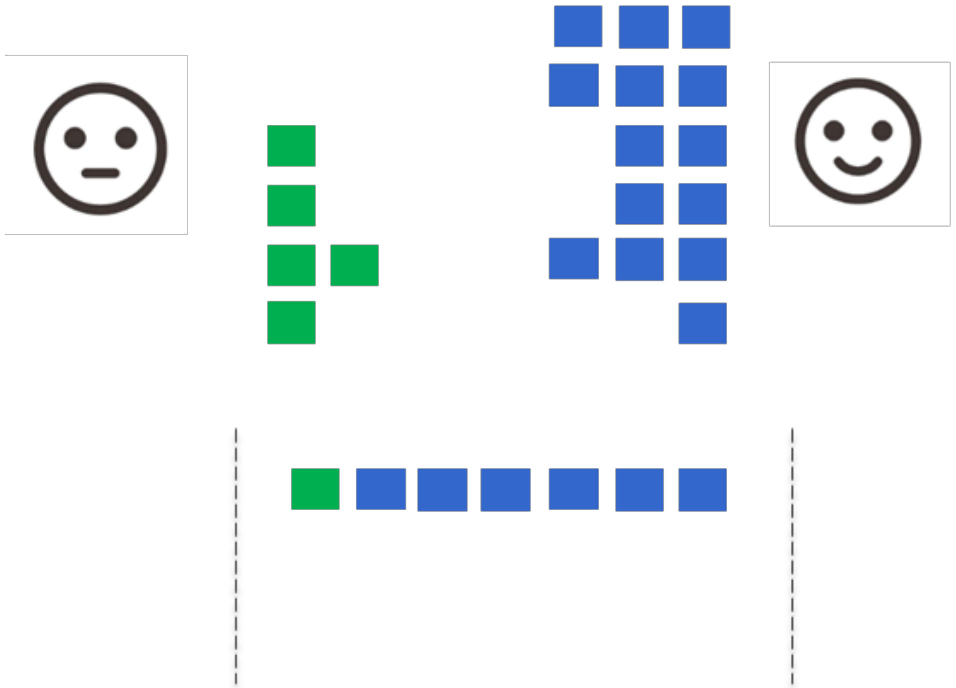


```
Pods:
--------------------------------
NAME                                     READY   STATUS      RESTARTS   AGE
lm-mpi-job-mpimaster-0                   0/1     Completed   3          2m
spark-operator-sparkoperator-f78854b64-rh52d   1/1     Running     0          1d


Volcano Jobs:
--------------------------------
Name         Creation               Phase       JobType     Replicas    Min    Pending    Running    Succeeded
lm-mpi-job   2019-06-19 20:55:33    Completed   MPI         3           3      0          0          1

m00483107@M00483107 MINGW64 /d/workspace/src/volcano.sh/volcano/docs/samples/kubecon-2019-china/mpi-sample (kub
$ kc logs lm-mpi-job-mpimaster-0
Warning: Permanently added 'lm-mpi-job-mpiworker-0.lm-mpi-job,172.16.0.22' (ECDSA) to the list of known hosts.
Warning: Permanently added 'lm-mpi-job-mpiworker-1.lm-mpi-job,172.16.0.46' (ECDSA) to the list of known hosts.
Hello world from processor lm-mpi-job-mpiworker-0, rank 0 out of 2 processors
Hello world from processor lm-mpi-job-mpiworker-1, rank 1 out of 2 processors
```

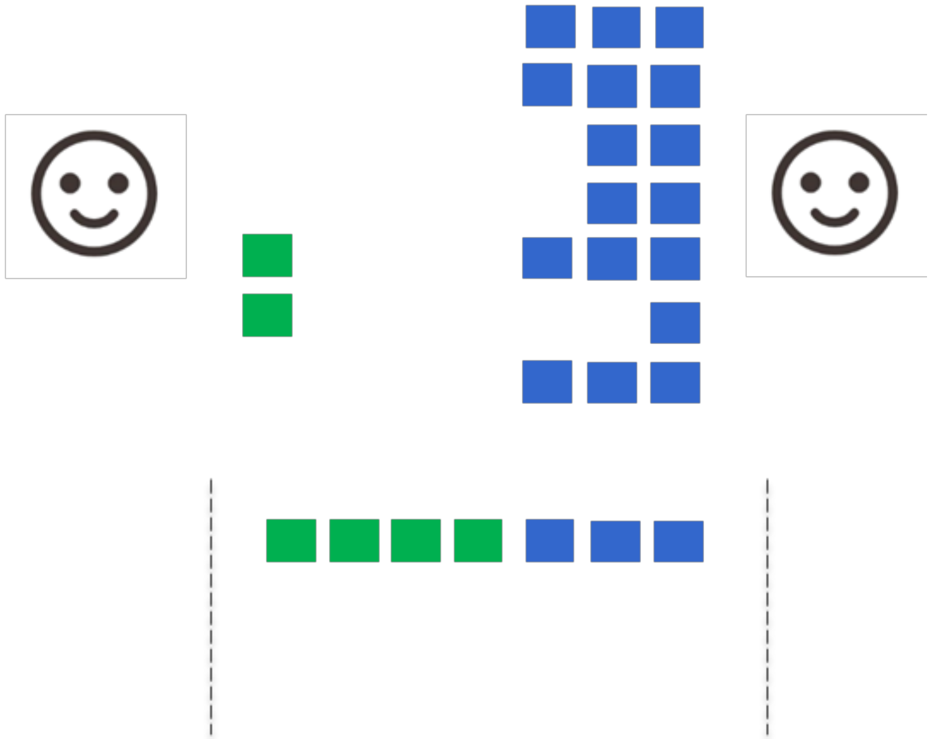
- The worker pods are deleted by job controller after job get finished

- The pod of mpirun will not be deleted for output

Scenarios: Fair Share

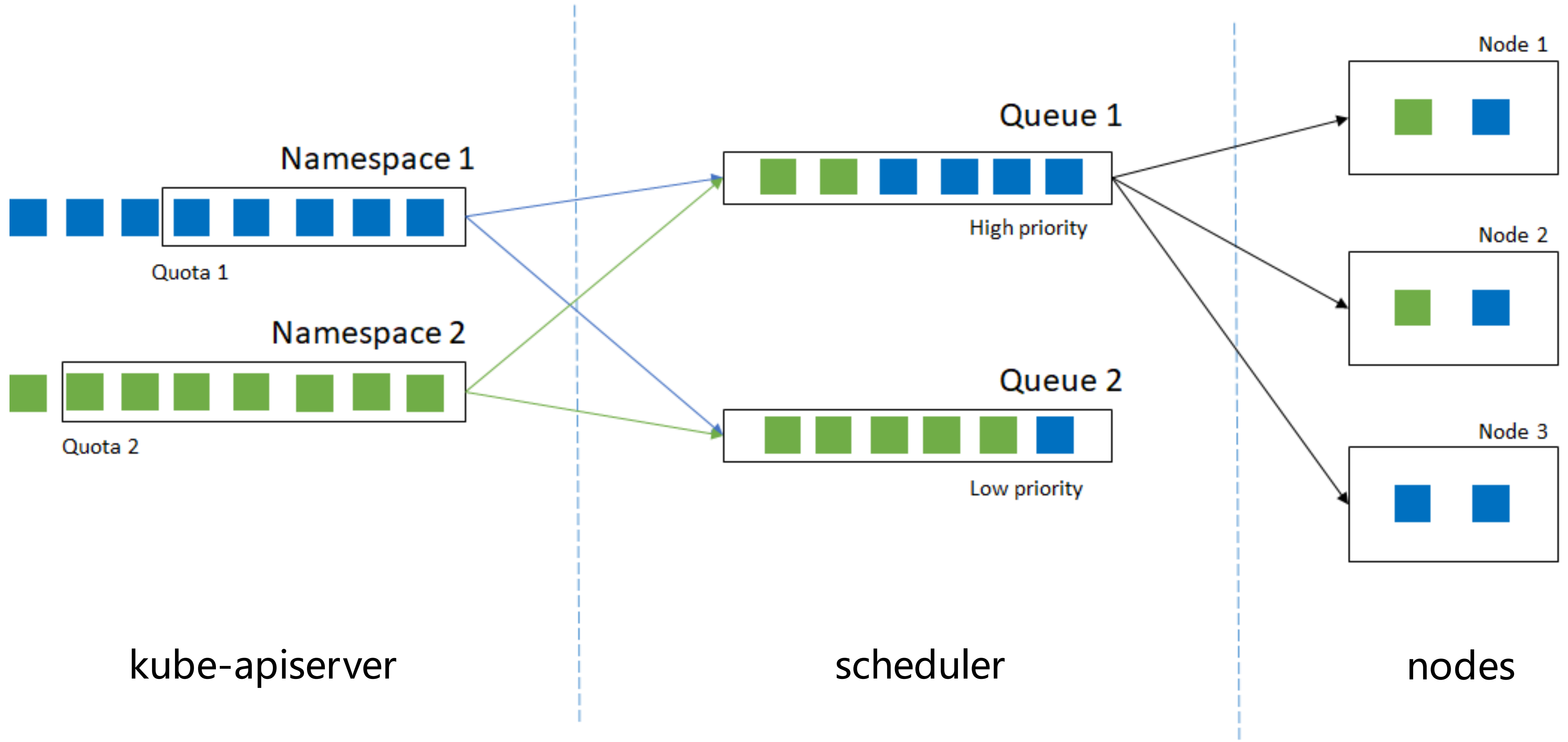


The more workload, the more resources???

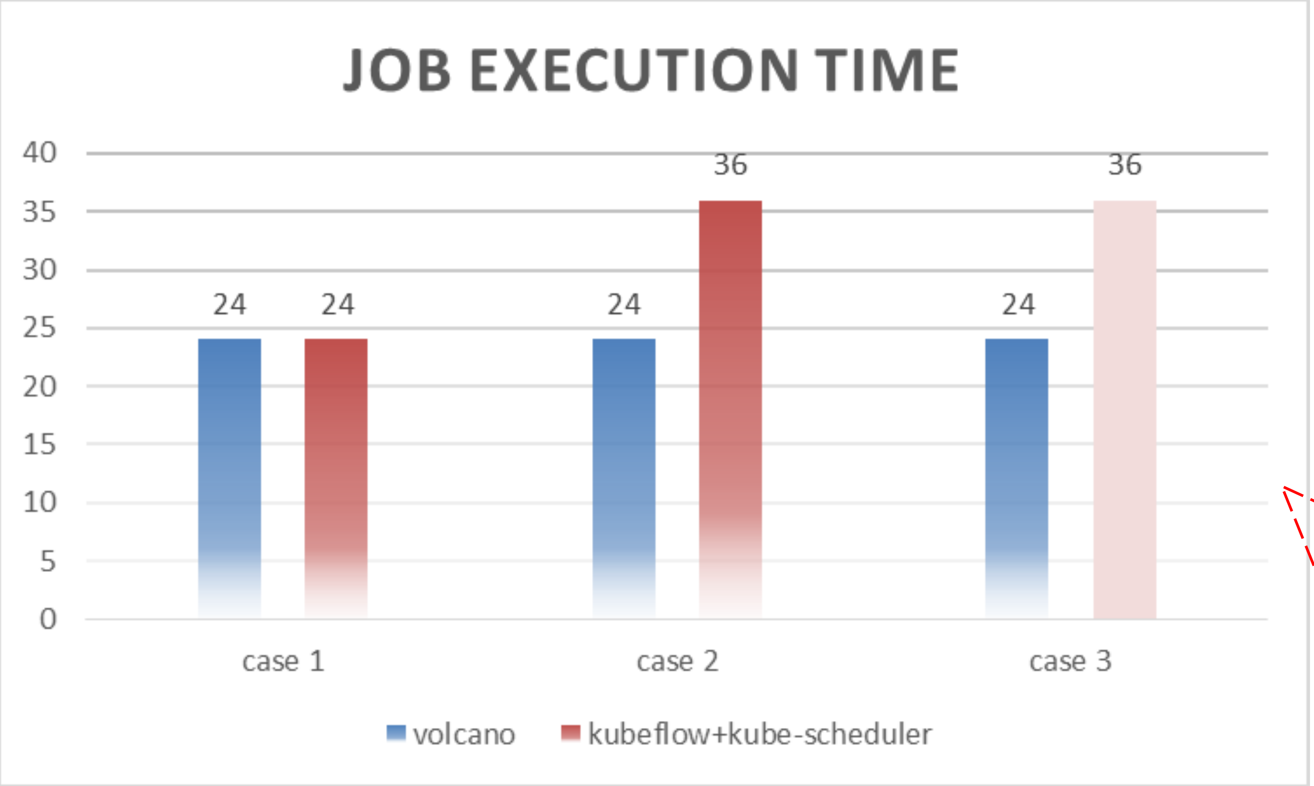


Share resources by weight !!!

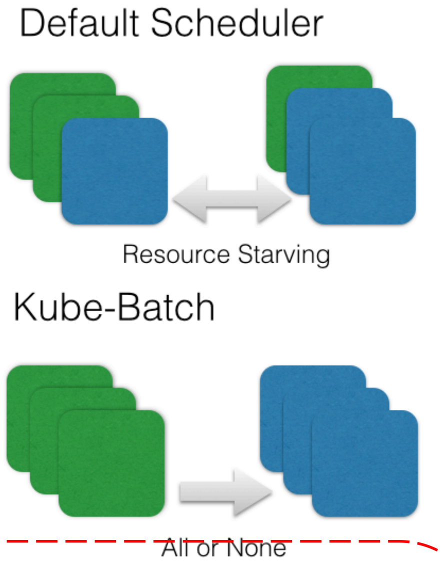
Scenarios: Job Priority



Gang Scheduling



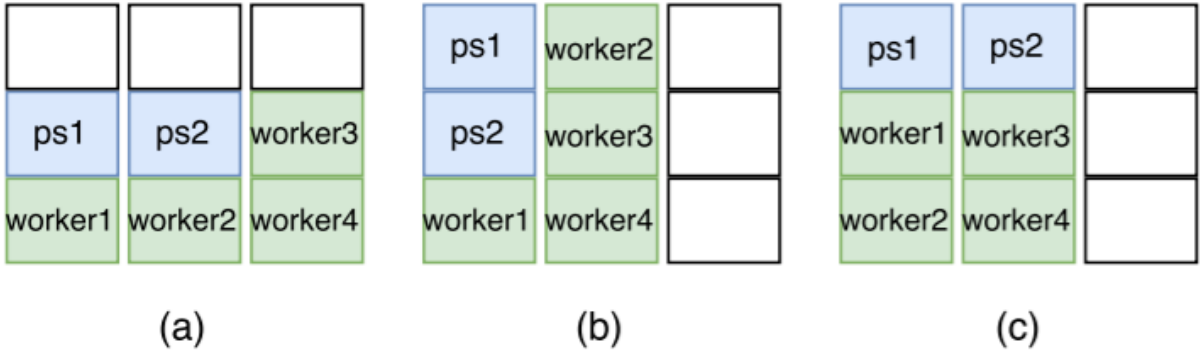
- Case 1: 1 job with 2ps + 4workers
- Case 2: 2 jobs with 2ps + 4workers
- Case 3: 5 jobs with 2ps + 4workers

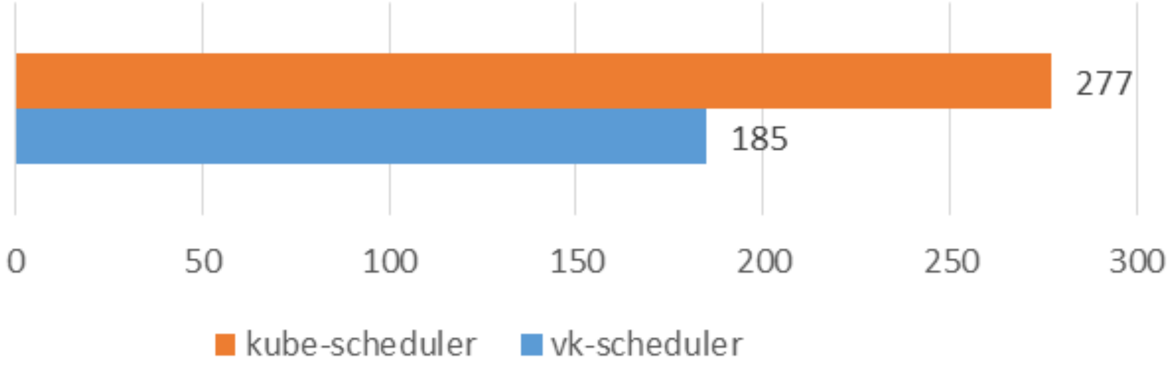


- No enough resource for 2 Jobs to run concurrently; one of them wasting resources without Gang-Scheduling !
- 2 of 5 jobs was finished because of deadlock (+20 hours)

http://status.openlabtesting.org/builds?project=theopenlab%2Fvolcano

Task Topology & Binpack

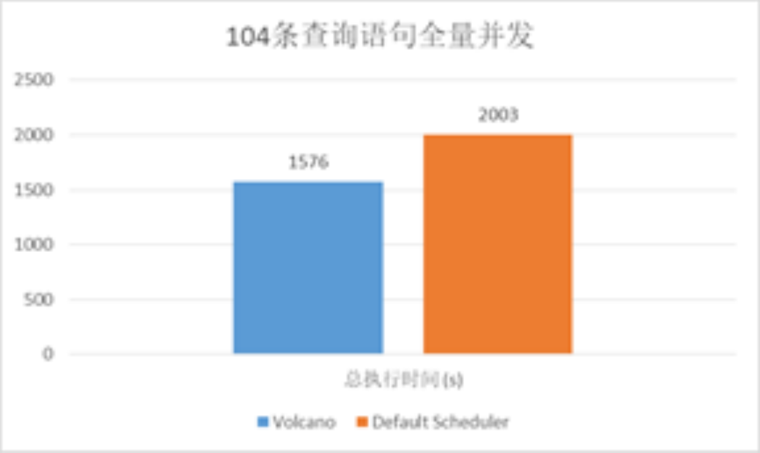


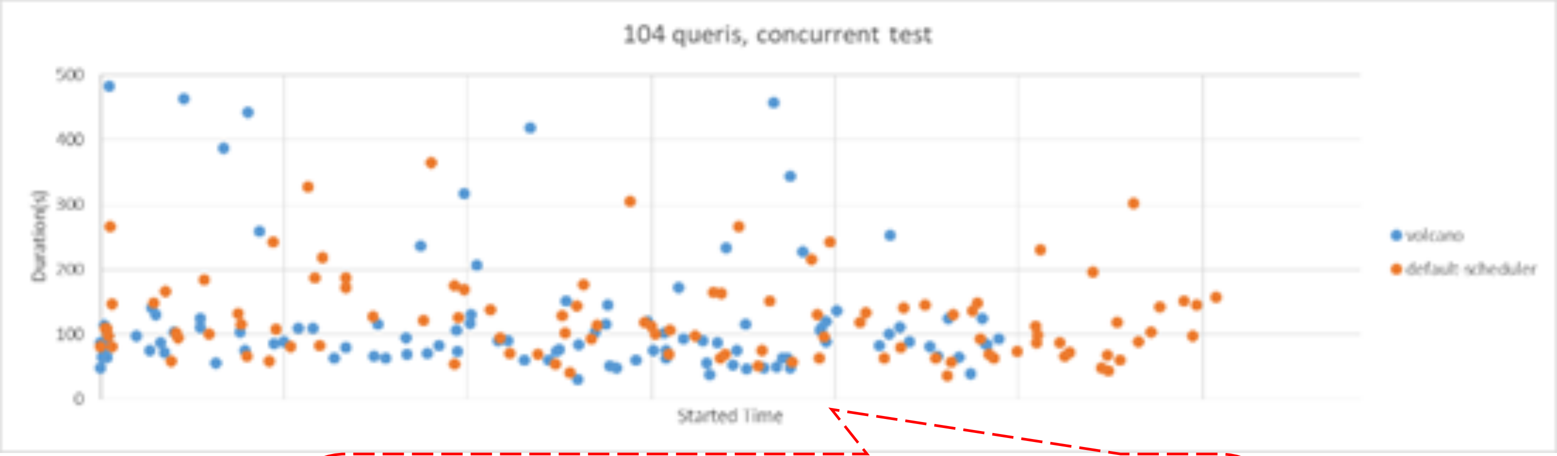


- The execution time of 3 jobs in total; 2ps + 4workers for each job
- The execution time is unstable when tested by default scheduler
- Default BinPack and Pod Affinity cannot always have worker and ps on the same host
- Task-topology make it happen

Reference: “Optimus: An Efficient Dynamic Resource Scheduler for Deep Learning Clusters”

Job minResource (Spark)





- Spark-sql-perf (TP-DCS, master)
- 104 queries concurrently
- (8cpu, 64G, 1600SSD) * 4nodes
- Kubernetes 1.13
- Driver: 1cpu,4G; Executor: (1cpu,4G)*5

- Max 26 concurrent queries if no dedicated driver nodes
- ~30% performance improvement because of job level reservation

- Volcano (min-res): 3.3cpu, 12G
- Kubernetes: 1 node for drivers

What get improved

Native Solution

- **Native** support **MPI** and **PServer**.
- **Native batch compute model** lifecycle management.
- **One click** installation on Kubernetes with CRD.

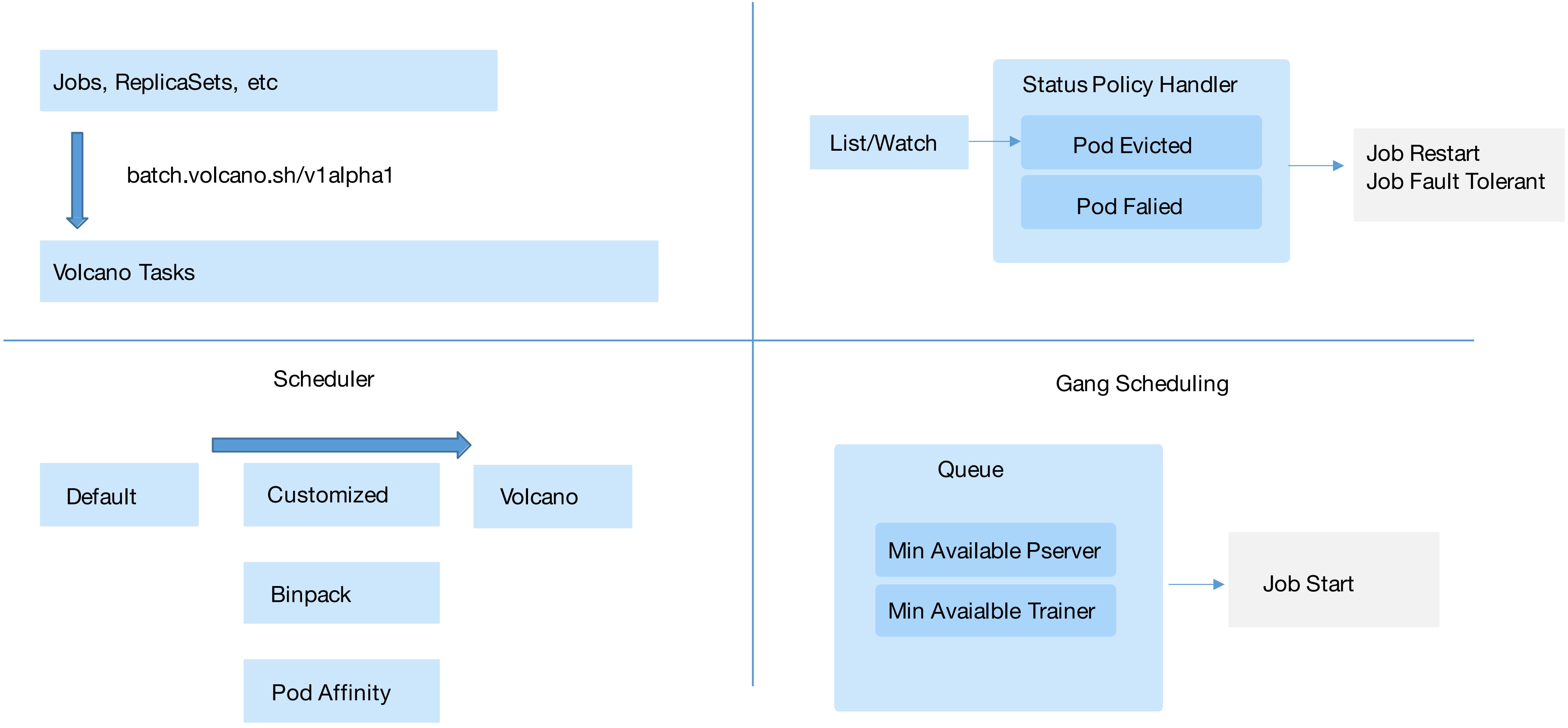
Accelerate Training Speed

- **Task topology** to reduce communication latency for PServer and Trainer.
- **MPI ring all-reduce** to resolve single point bottleneck.
- **Gang scheduling** to prevent jobs from resource starvation.

Optimize Resource Usage

- **Fair share** algorithm to optimize resource usage between tenants.
- **Priority queues** for urgent deep learning tasks to get executed in high priority.

Refactoring for Volcano



Jobs, ReplicaSets, etc

batch.volcano.sh/v1alpha1

Volcano Tasks

Status Policy Handler

List/Watch

Pod Evicted

Pod Falied

Job Restart
Job Fault Tolerant

Scheduler

Default

Customized

Volcano

Binpack

Pod Affinity

Gang Scheduling

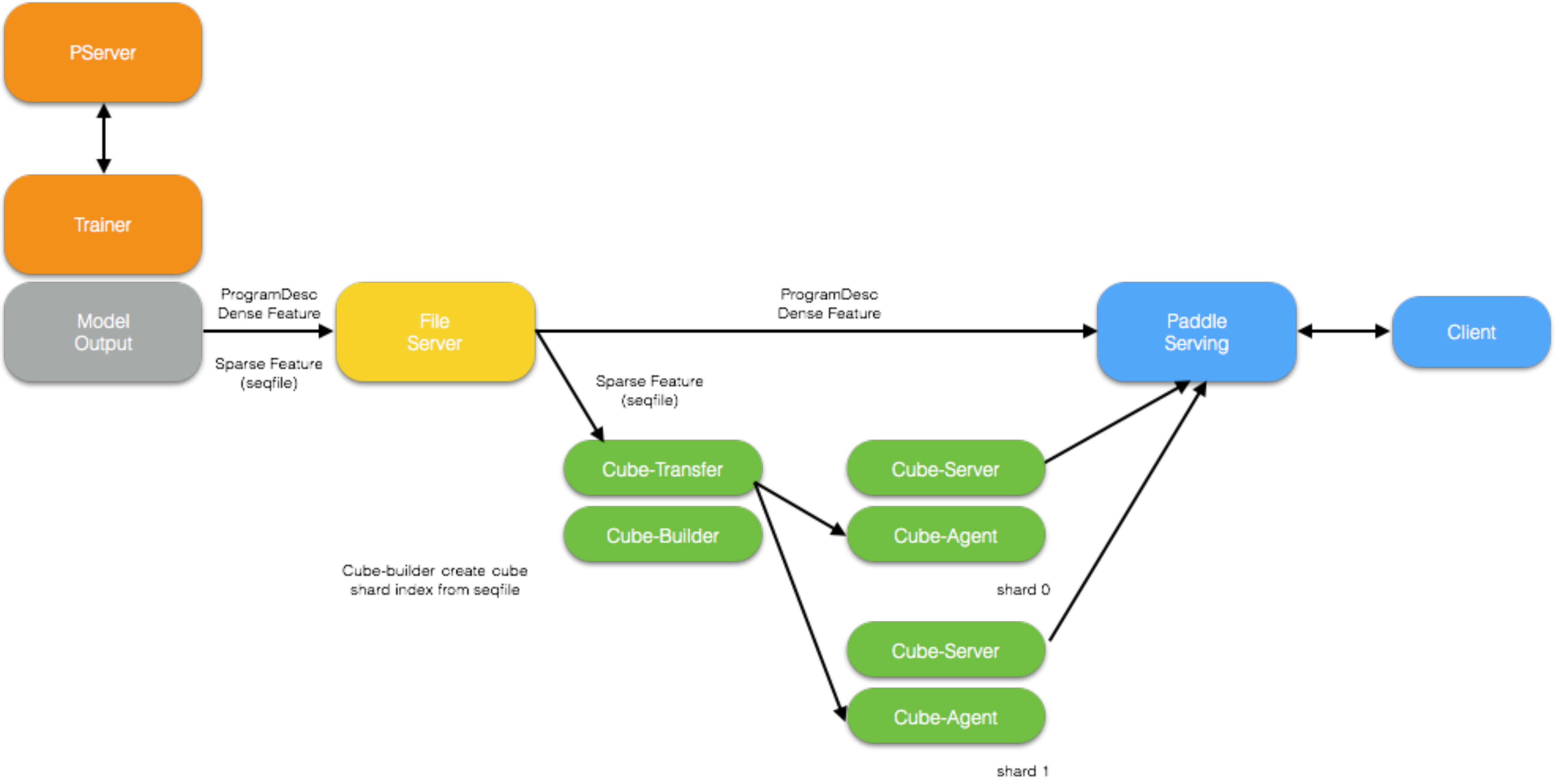
Queue

Min Available Pserver

Min Avaialble Trainer

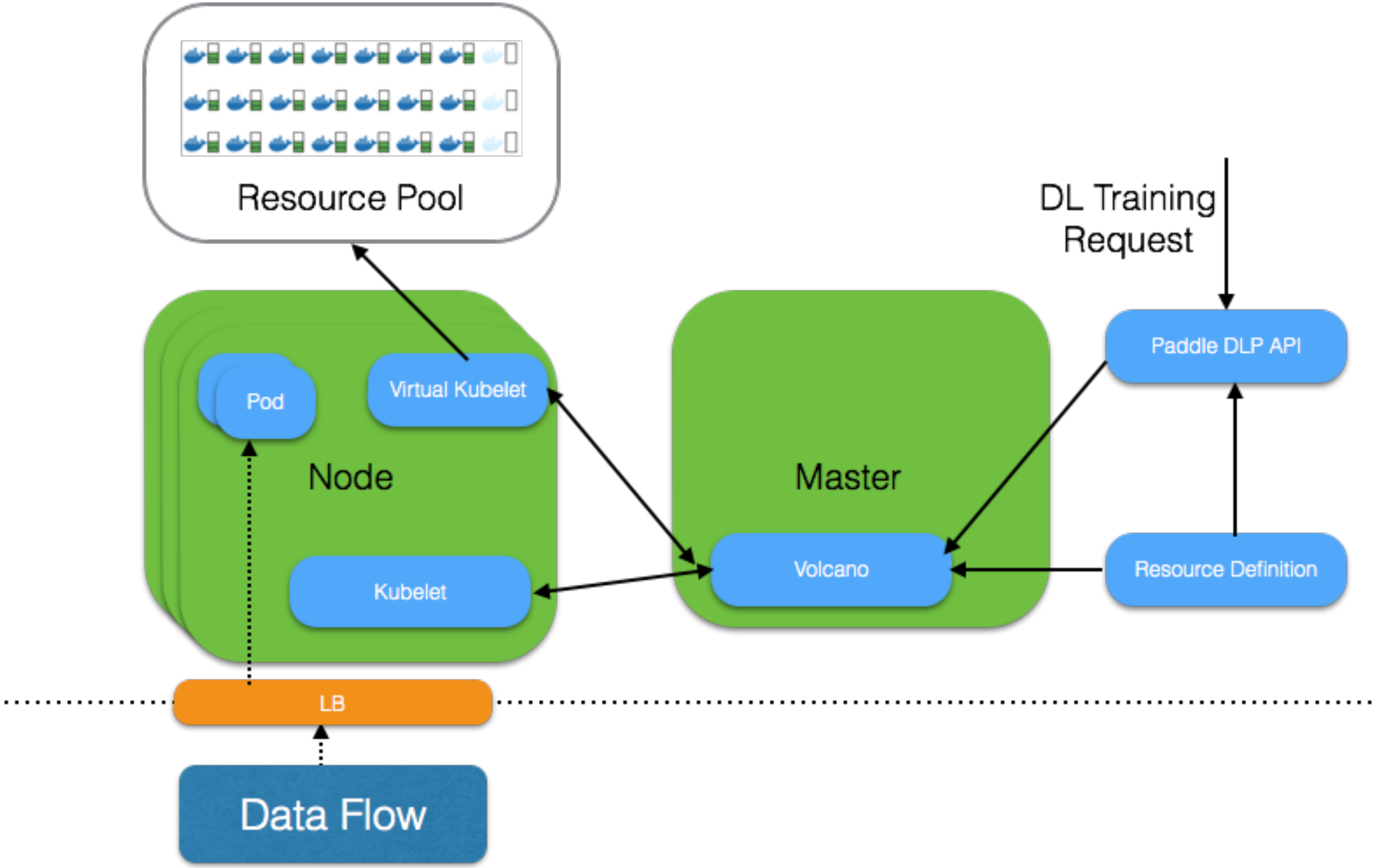
Job Start

Elastic CTR Estimation Demo

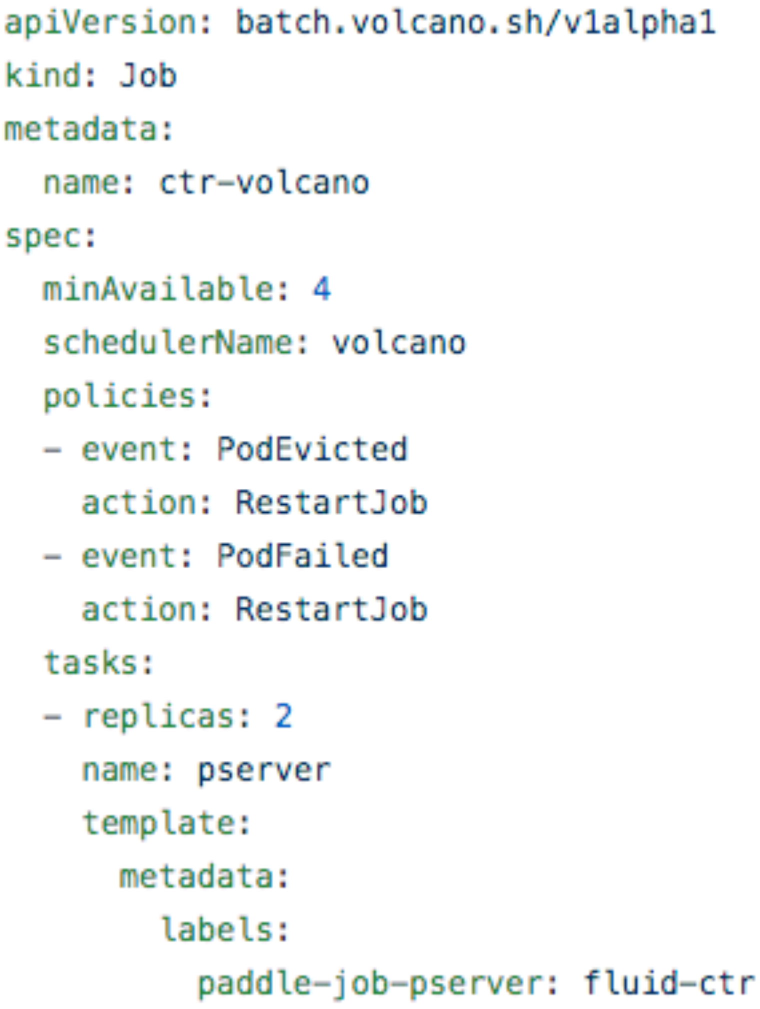


Cube-builder create cube shard index from seqfile

CTR with Volcano

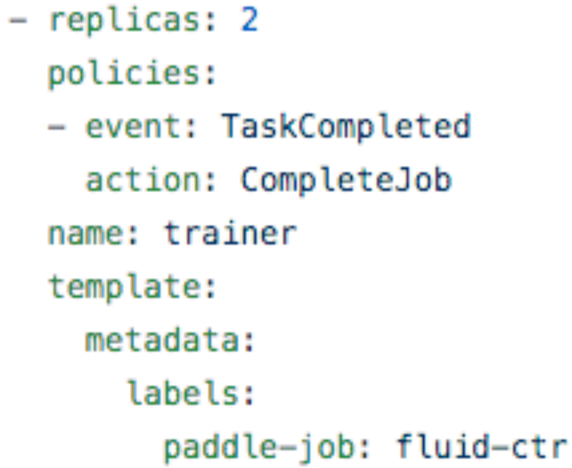


CTR yaml



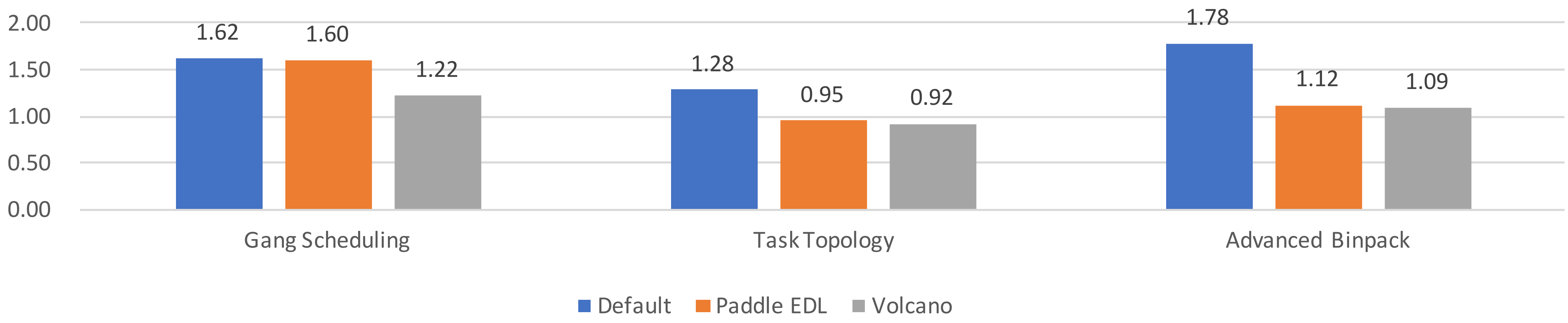
```
apiVersion: batch.volcano.sh/v1alpha1
kind: Job
metadata:
  name: ctr-volcano
spec:
  minAvailable: 4
  schedulerName: volcano
  policies:
  - event: PodEvicted
    action: RestartJob
  - event: PodFailed
    action: RestartJob
  tasks:
  - replicas: 2
    name: pserver
    template:
      metadata:
        labels:
          paddle-job-pserver: fluid-ctr
```

https://github.com/volcano-sh/volcano/blob/master/example/integrations/paddlepaddle/ctr-paddlepaddle-on-volcano.yaml



```
  - replicas: 2
    policies:
    - event: TaskCompleted
      action: CompleteJob
    name: trainer
    template:
      metadata:
        labels:
          paddle-job: fluid-ctr
```

CTR Performance Comparison



Gang Scheduling

- Prevent deadlock in cluster with high resource utilization.

Task Topology

- Reduce data transmission latency between trainer pod and worker pod.

Advanced BinPack

- Reduce network overloads between different hosts.

Pipeline

- GPU Share/Topology
- Job Management
- Queue Management
- Hierarchical Queue
- Preemption/Reclaim
- ……

Contact Info

Welcome to use and contribute!

Website: https://volcano.sh

Github: http://github.com/volcano-sh/volcano

Twitter: https://twitter.com/volcano_sh

Slack: http://volcano-sh.slack.com

Email: volcano-sh@googlegroups.com

Thanks!