caicloud

才云

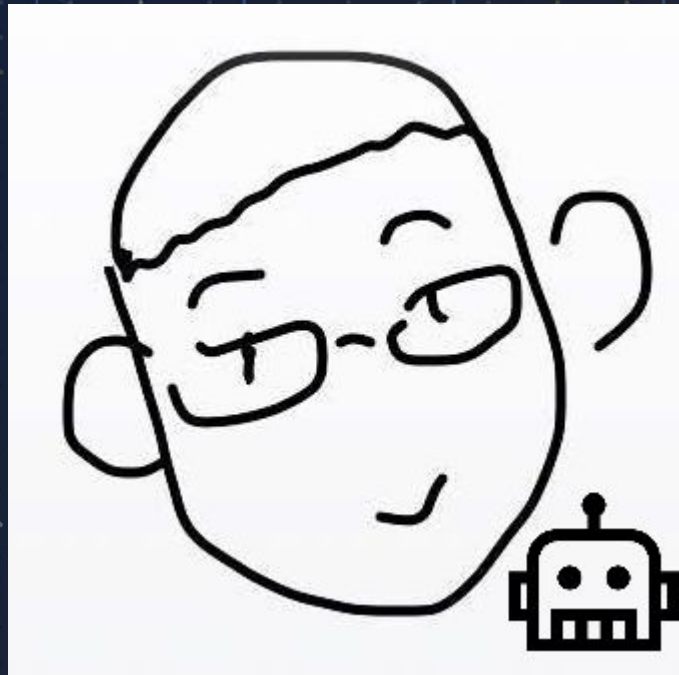
在云上运行机器学习工作负载

高策

技术分享

关于我

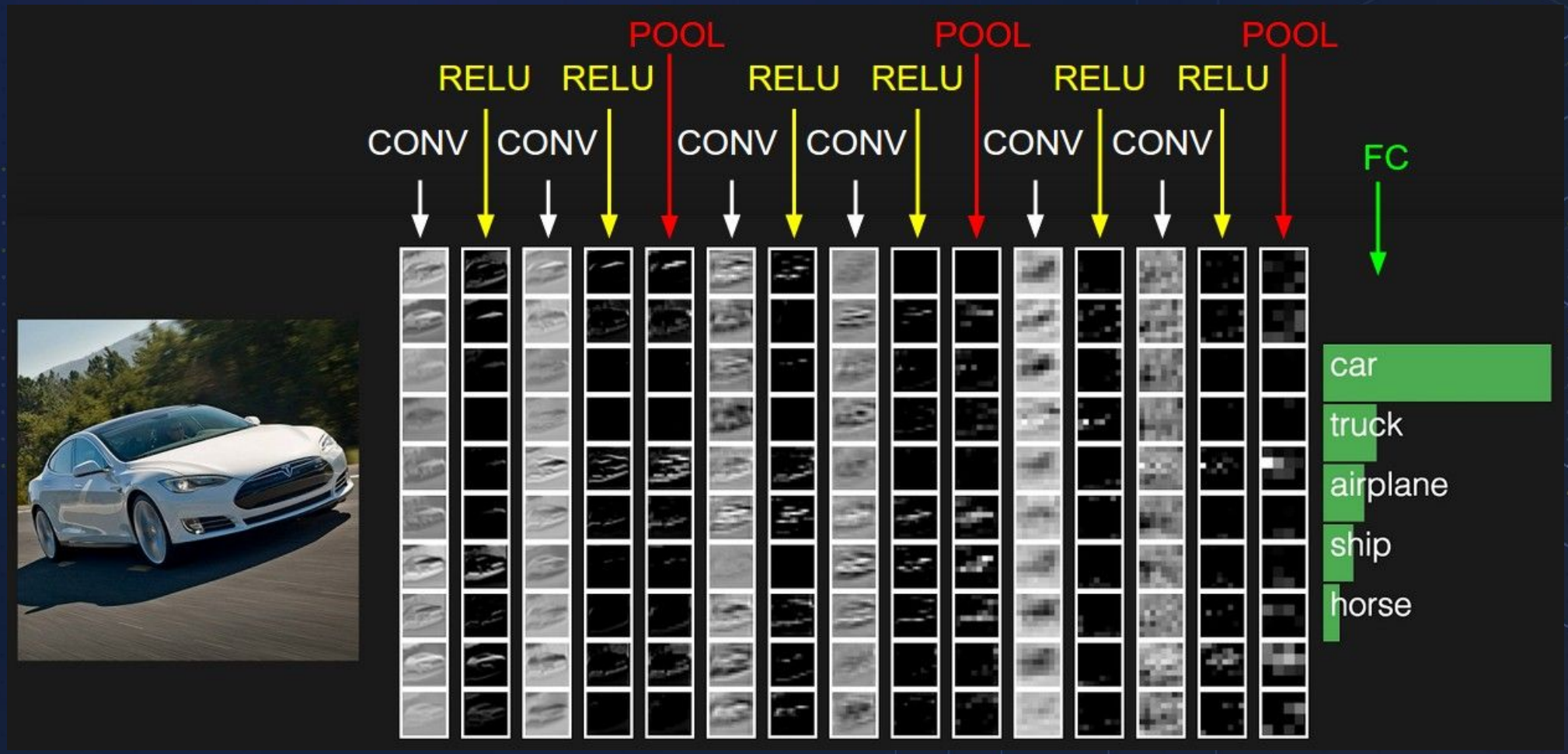
- 高策，就职于才云科技



纲要

- 背景
- 问题
- 云上的资源管理问题
- 大规模分布式训练
- 可容错的模型训练

背景-VGG16



Credits: http://cs231n.github.io/convolutional-networks/

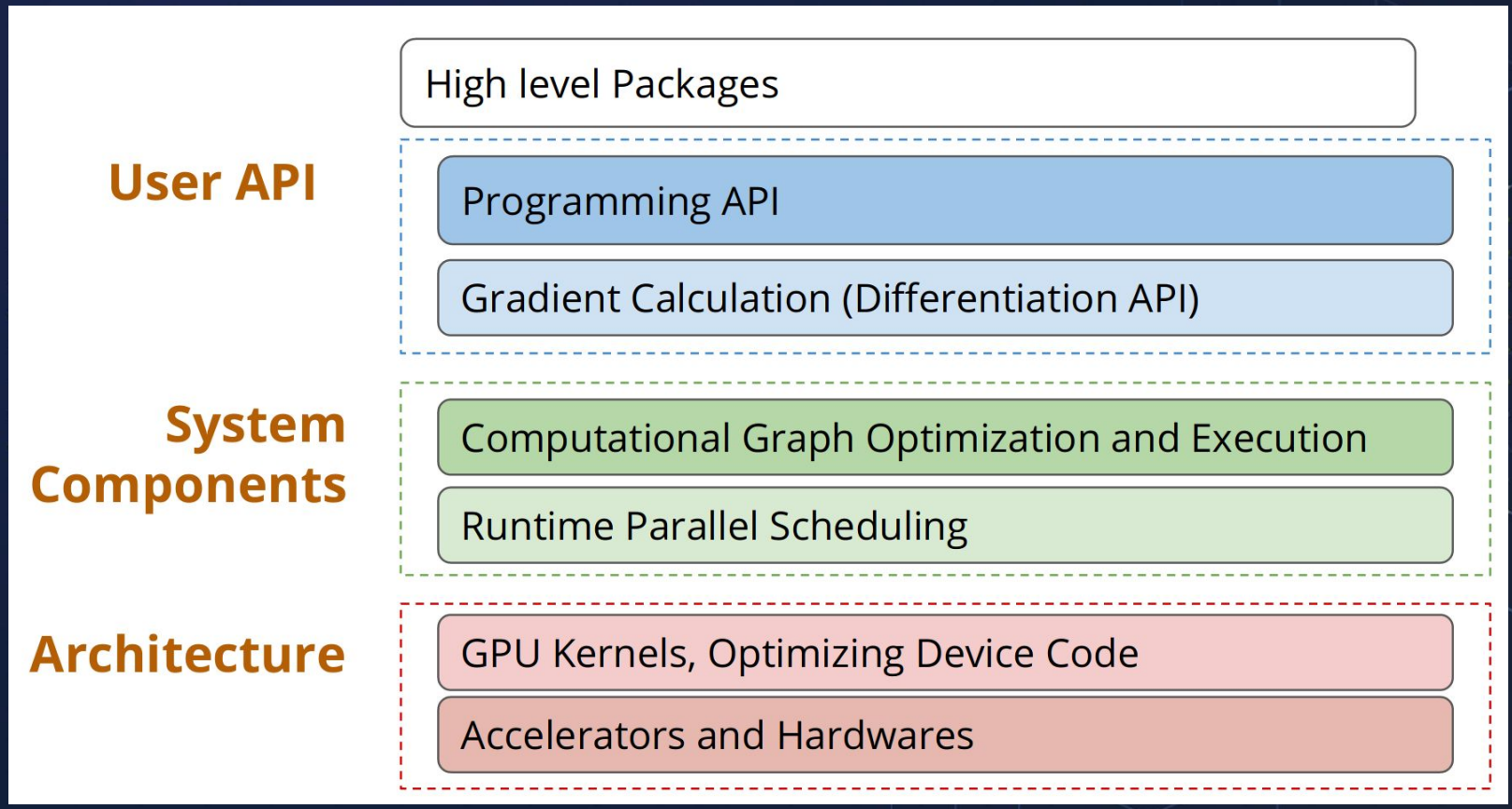
背景-VGG16

```
# Block 1
x = layers.Conv2D(64, (3, 3),
                  activation='relu',
                  padding='same',
                  name='block1_conv1')(img_input)
x = layers.Conv2D(64, (3, 3),
                  activation='relu',
                  padding='same',
                  name='block1_conv2')(x)
x = layers.MaxPooling2D((2, 2), strides=(2, 2), name='block1_pool')(x)

# Block 2
x = layers.Conv2D(128, (3, 3),
                  activation='relu',
                  padding='same',
                  name='block2_conv1')(x)
x = layers.Conv2D(128, (3, 3),
                  activation='relu',
                  padding='same',
                  name='block2_conv2')(x)
x = layers.MaxPooling2D((2, 2), strides=(2, 2), name='block2_pool')(x)

...
```

背景-Behind the Scene



Credits: http://dlsys.cs.washington.edu/pdf/lecture3.pdf

背景-分布式训练

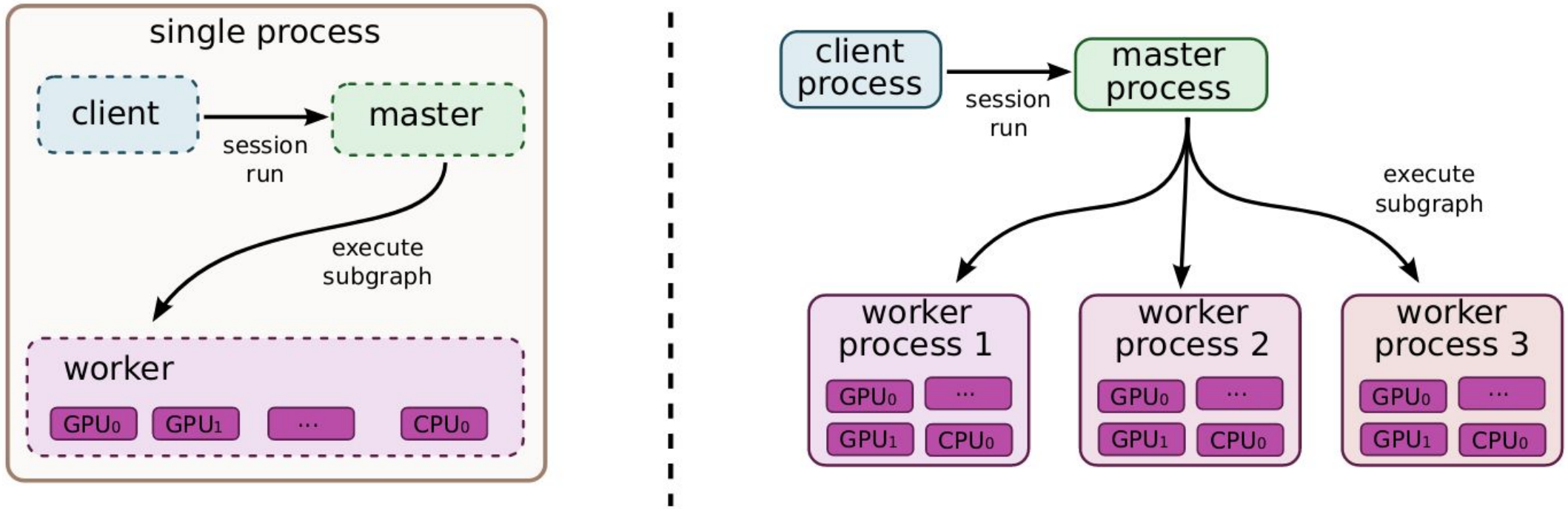


Figure 3: Single machine and distributed system structure

Credits: TensorFlow White Paper

机器学习落地的问题

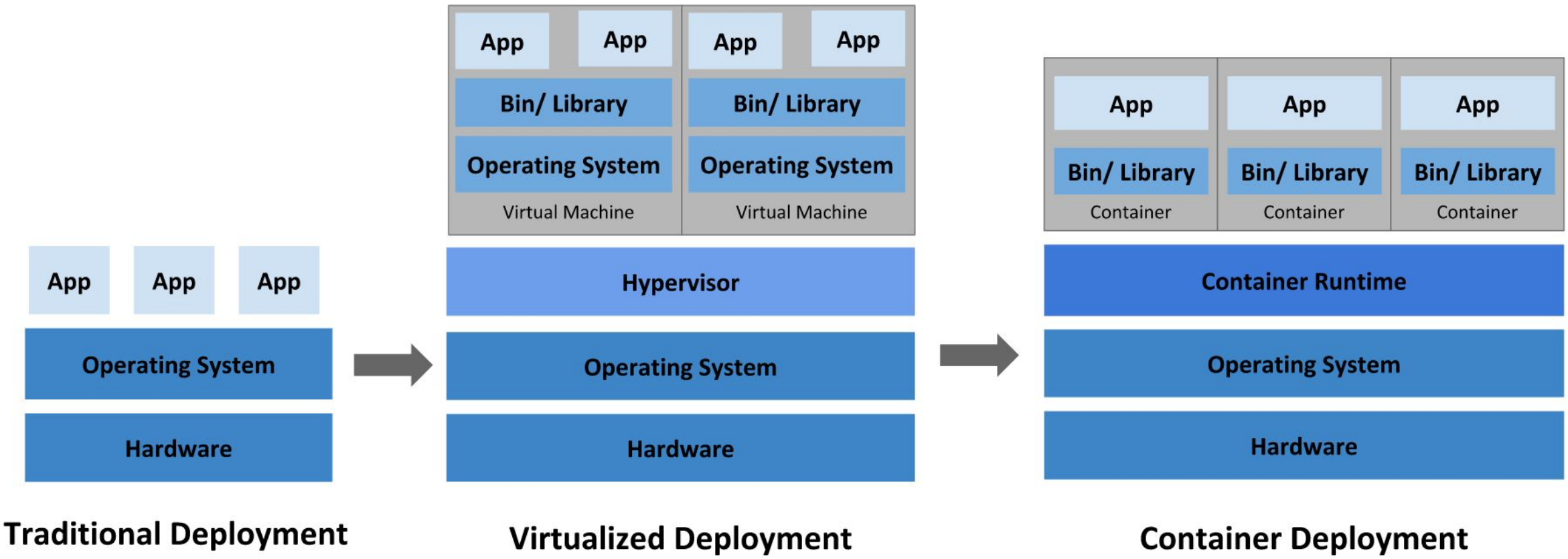
算法工程师的角度

- 如何做好数据的准备与预处理
- 如何做好模型训练(单机/分布式)

基础架构工程师的角度

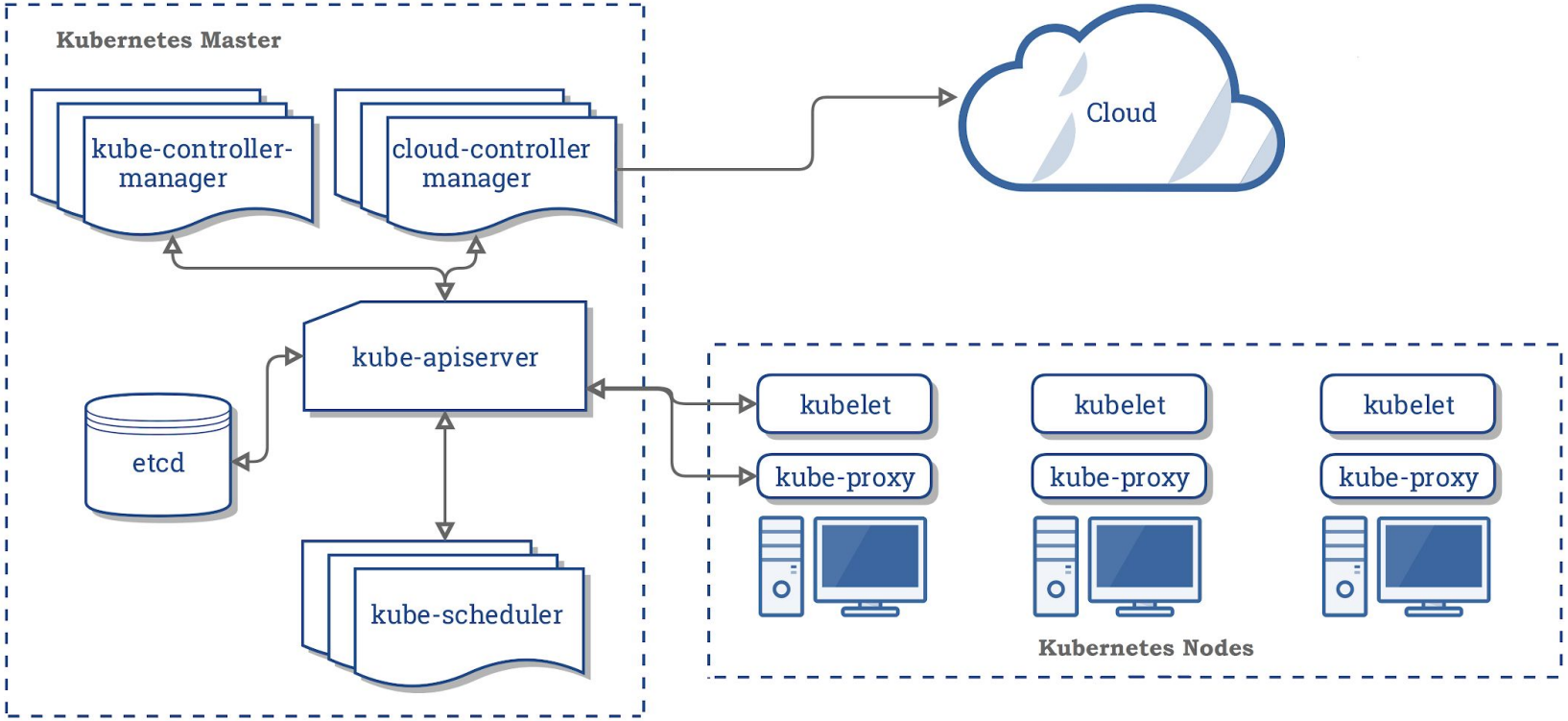
- 如何做好硬件资源的管理(隔离与共享)
- 如何做好大规模分布式训练的支持
- 如何保证集群资源的高利用率
- 如何容忍训练/推理过程中的错误

Why Kubernetes



Why Kubernetes

- 容器化，方便管理
- 扩展性好，支持不同的存储方式，不同的运行时，不同的硬件
- 横向扩展性好，节点可横向扩容
- 集群管理领域的事实标准



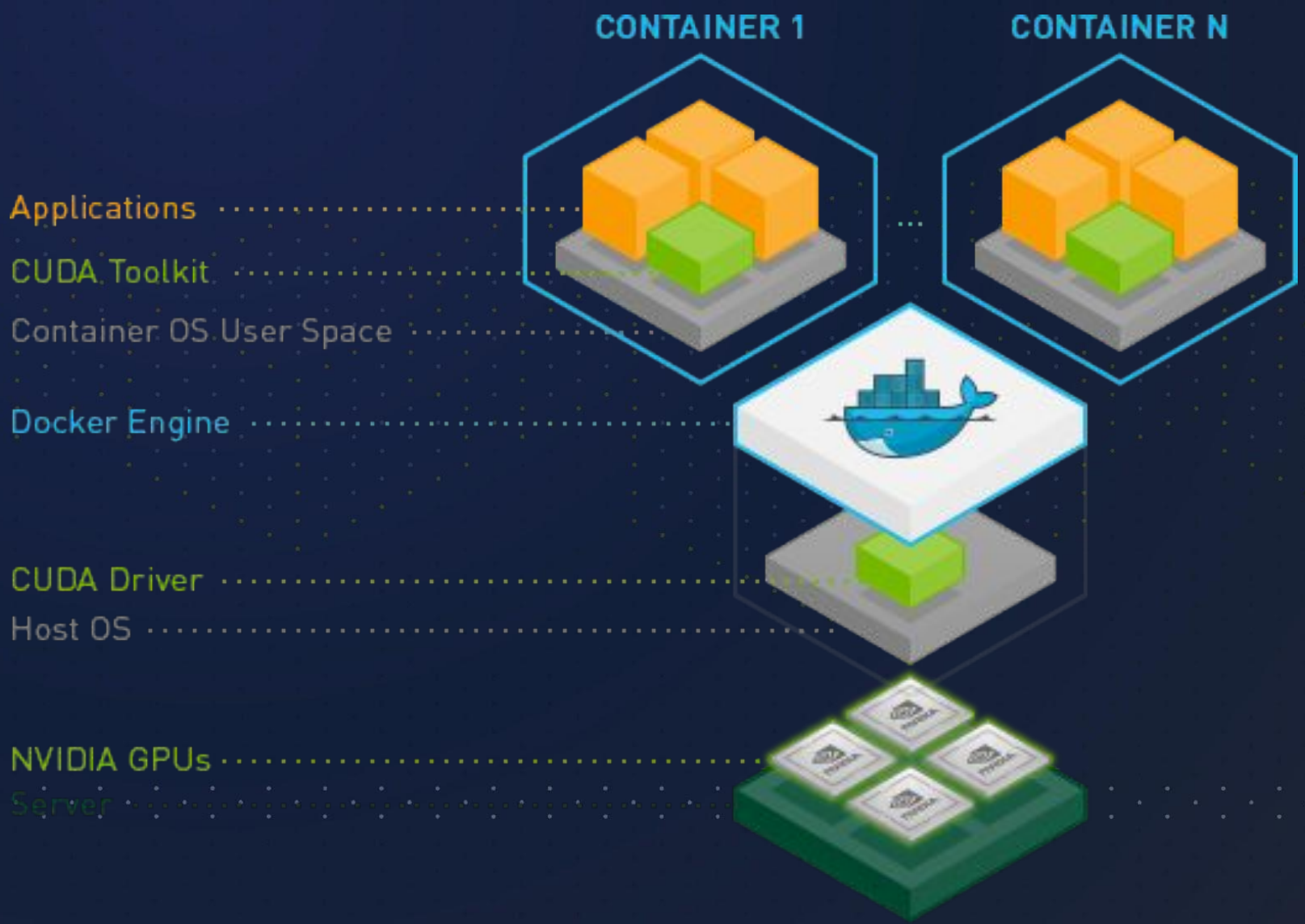
Why Kubernetes

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', 'echo Hello Kubernetes! && sleep 3600']
```

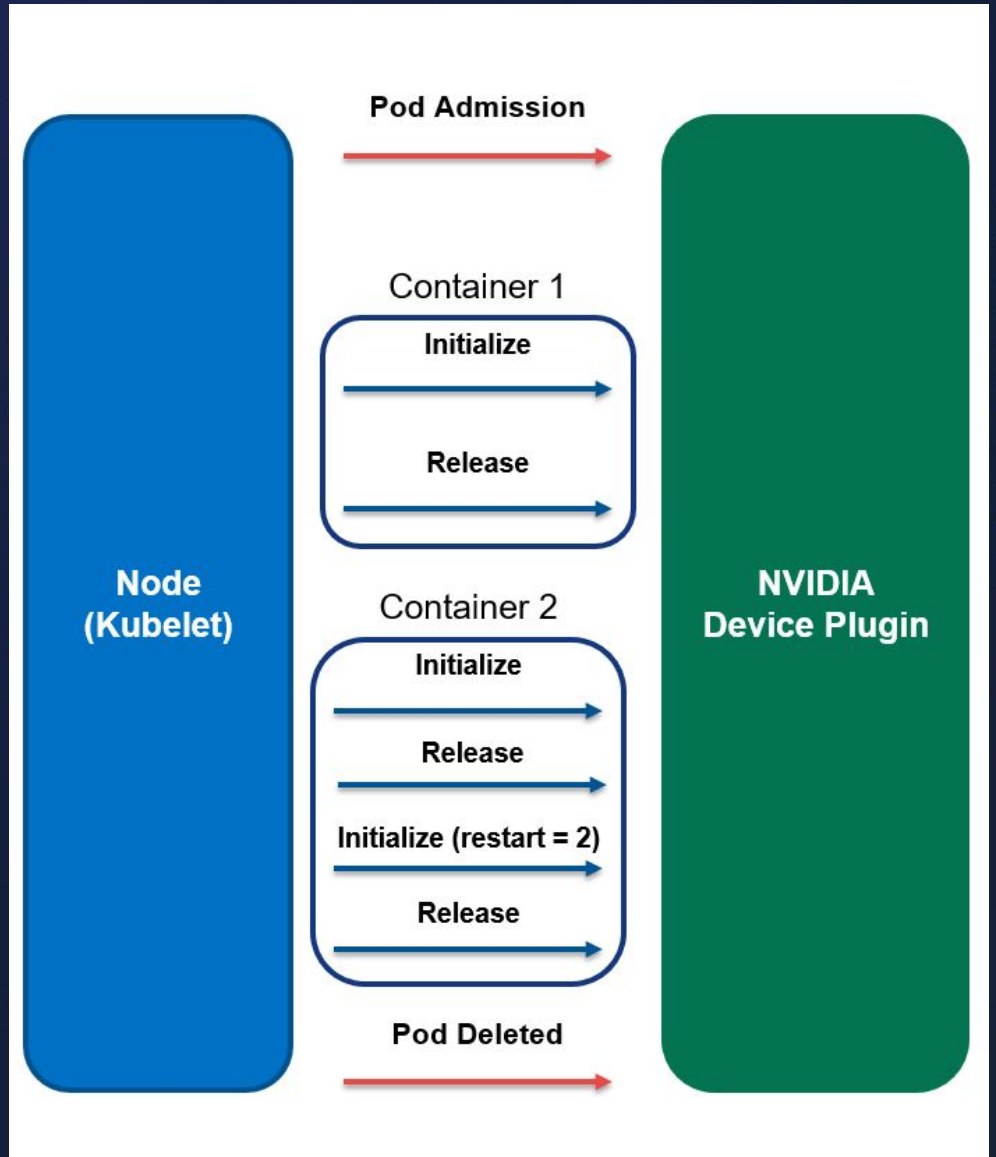
为什么需要对硬件资源进行管理

- 隔离（CPU, Memory, GPU 等资源）
  - 保证训练与训练之间不会互相影响
  - 保证训练失败不会影响其他训练
  - 保证训练不会影响推理服务
- 共享
  - 在推理等单个服务用不完单个显卡算力的情况下，共享来提高资源利用率，节约成本

硬件资源的管理:隔离

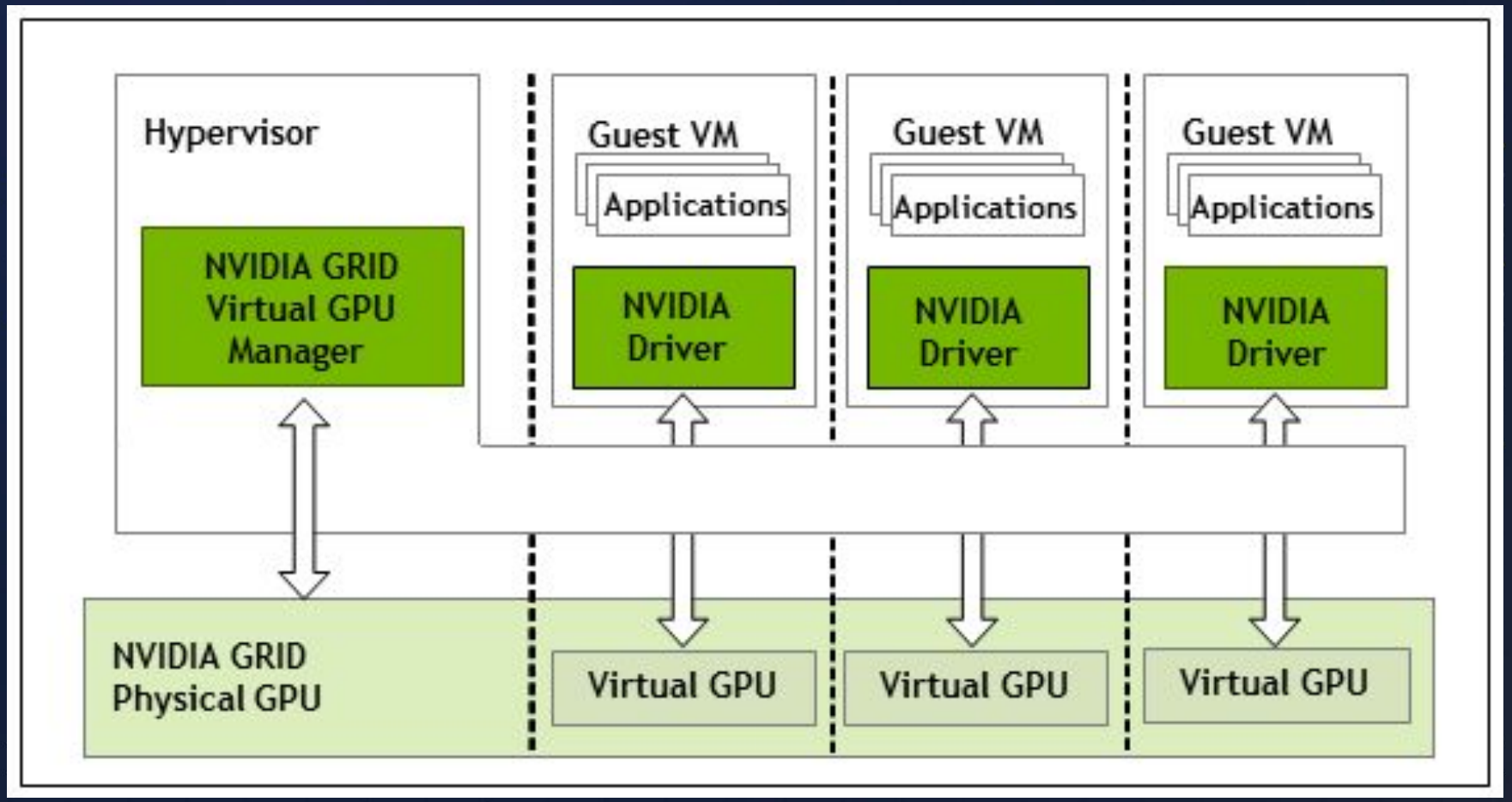

CONTAINER 1
CONTAINER N
Applications
CUDA Toolkit
Container OS User Space
Docker Engine
CUDA Driver
Host OS
NVIDIA GPUs
Server

硬件资源的管理:隔离


Pod Admission
Node
(Kubelet)
Container 1
Initialize
Release
Container 2
Initialize
Release
Initialize (restart = 2)
Release
NVIDIA
Device Plugin
Pod Deleted

硬件资源的管理:共享

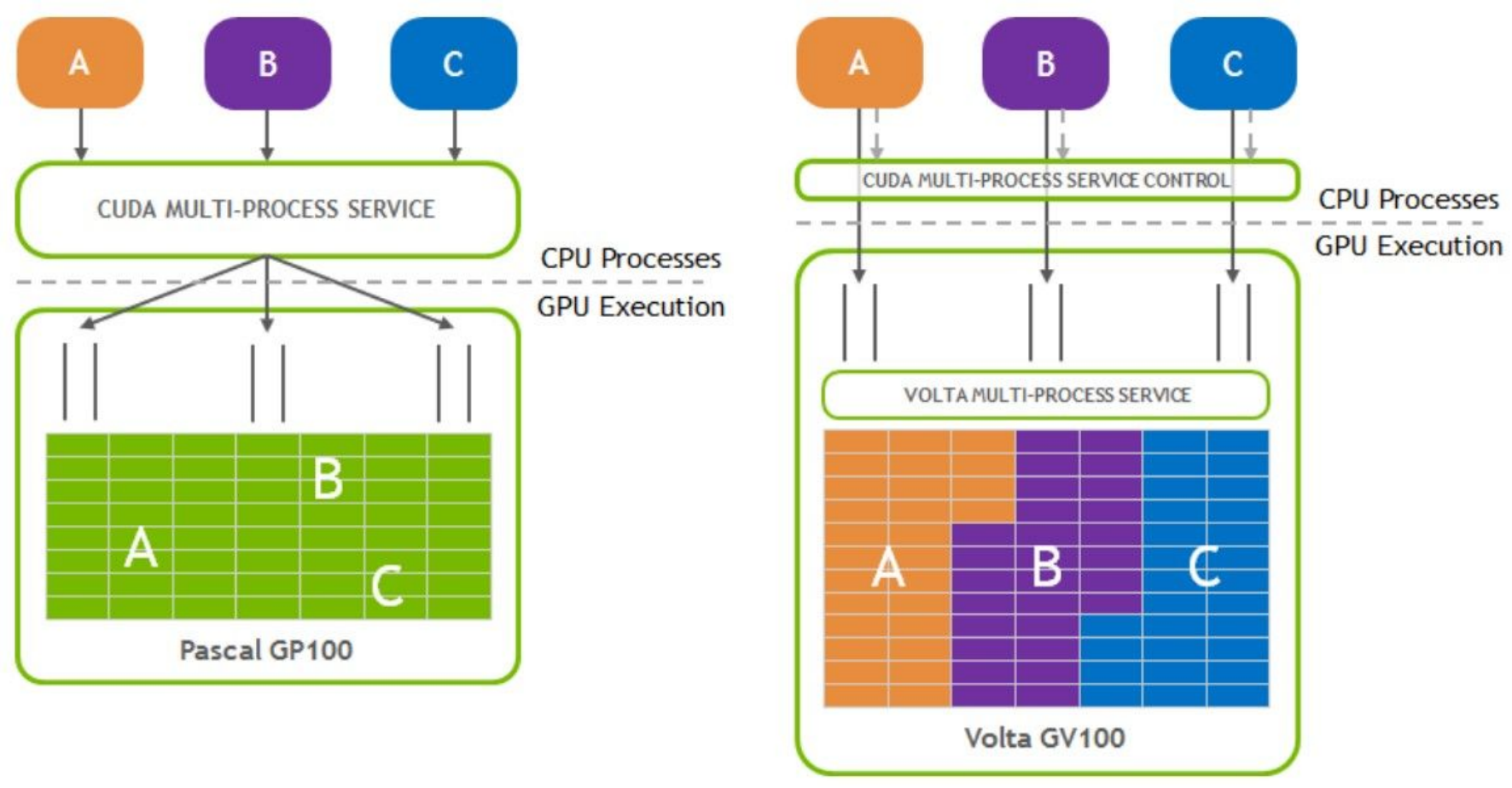
- 英伟达基于虚拟化的解决方案 Nvidia Grid



Credits: Nvidia Grid

硬件资源的管理:共享

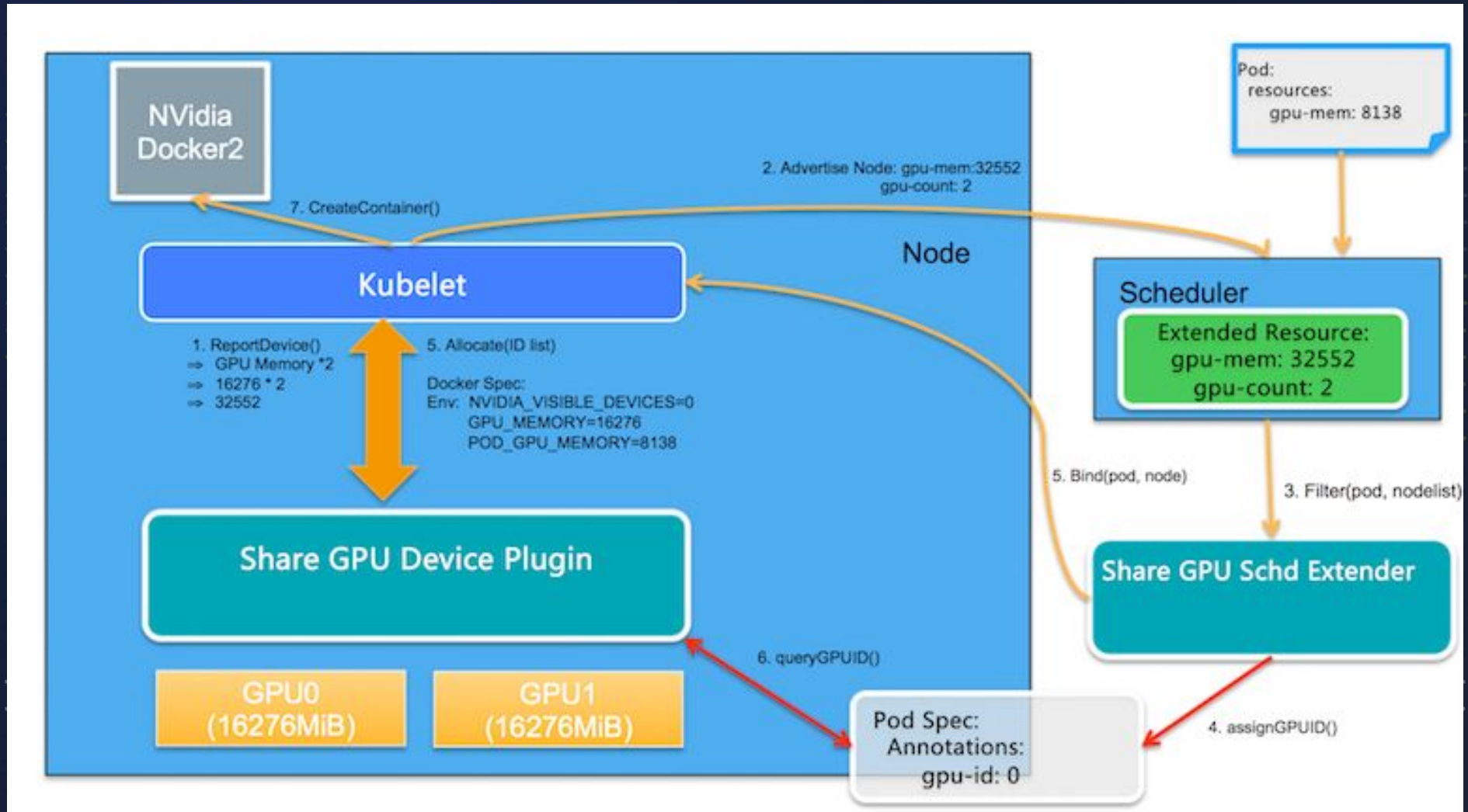
- 英伟达基于最新硬件(Volta)的解决方案 MPS



Credits: Volta 架构白皮书

硬件资源的管理:共享

- 阿里云开源的 Kubernetes Native 的解决方案



硬件资源的管理：共享

- Intercept CUDA 调用的解决方案：以腾讯云开源的解决方案为例

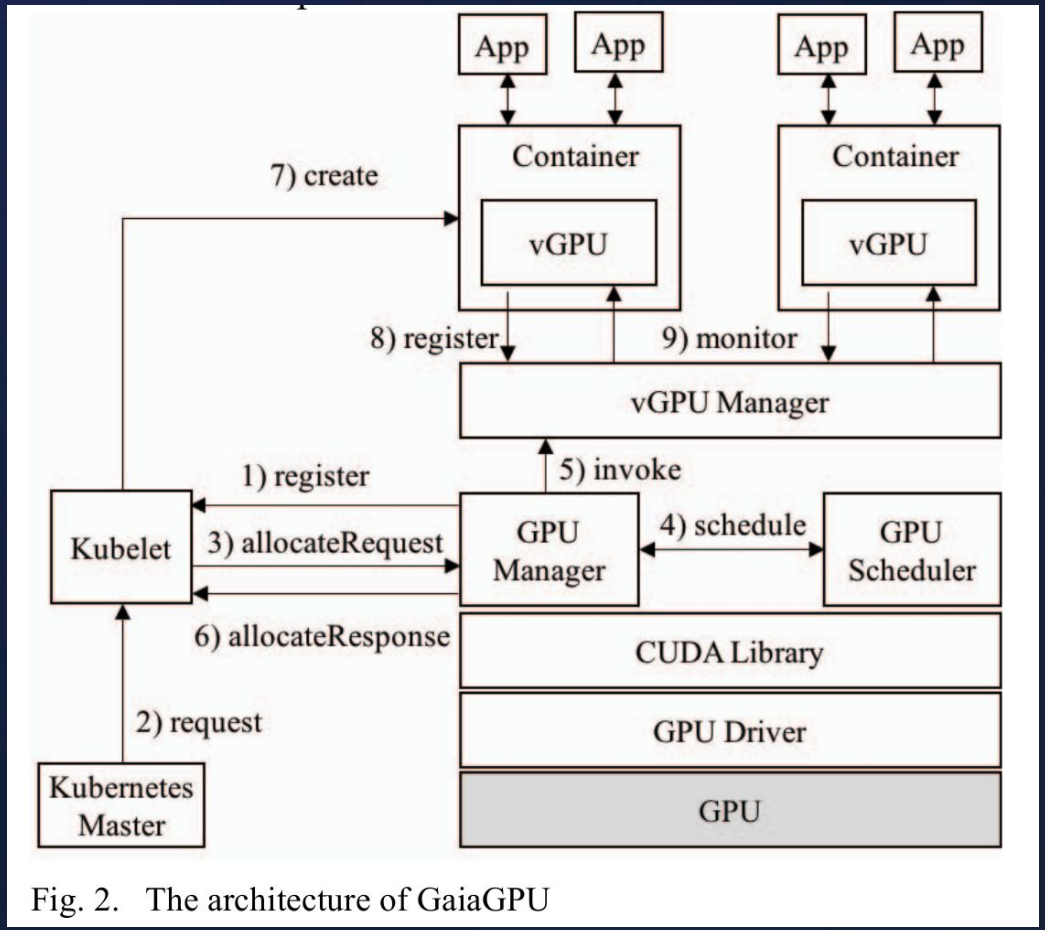


Fig. 2. The architecture of GaiaGPU

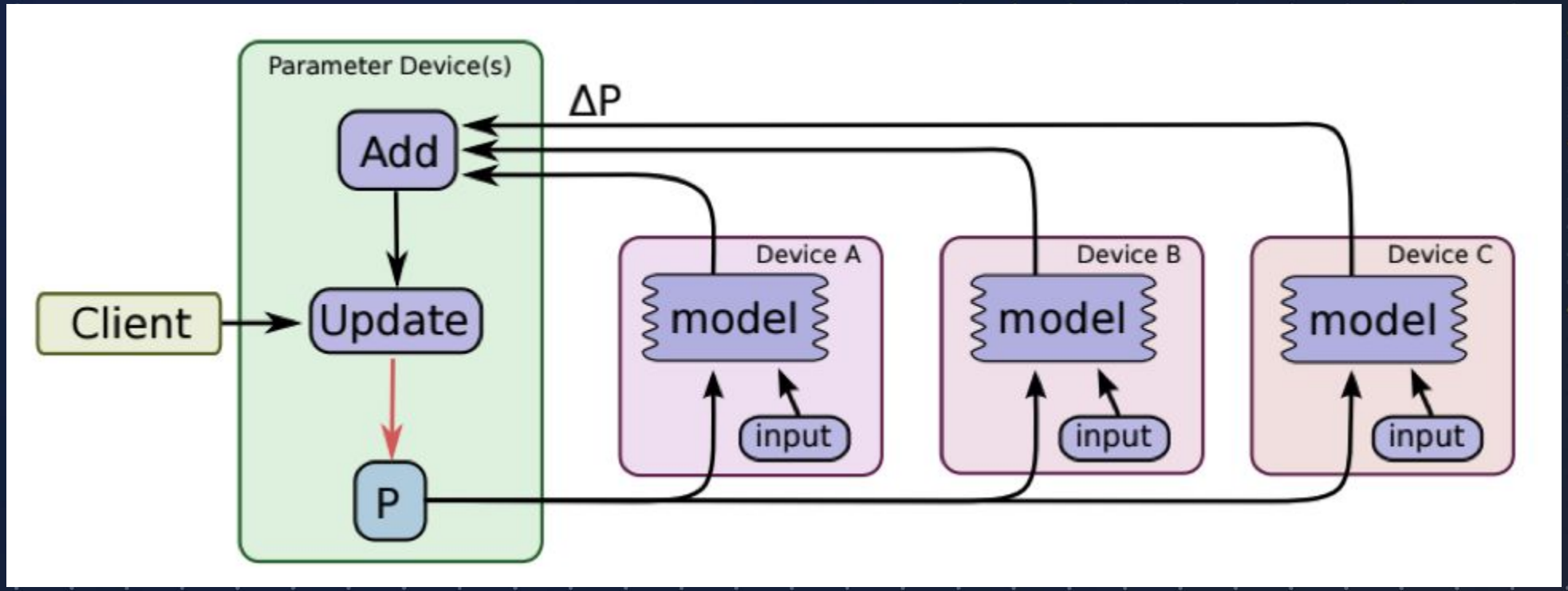
Credits: GaiaGPU: Sharing GPUs in Container Clouds

硬件资源的管理-总结

- 各有利弊
  - Nvidia Grid：基于虚拟机的解决方案
  - MPS：需要新硬件
  - Kubernetes Native 方案：没有真正的隔离
  - Intercept CUDA：error prone

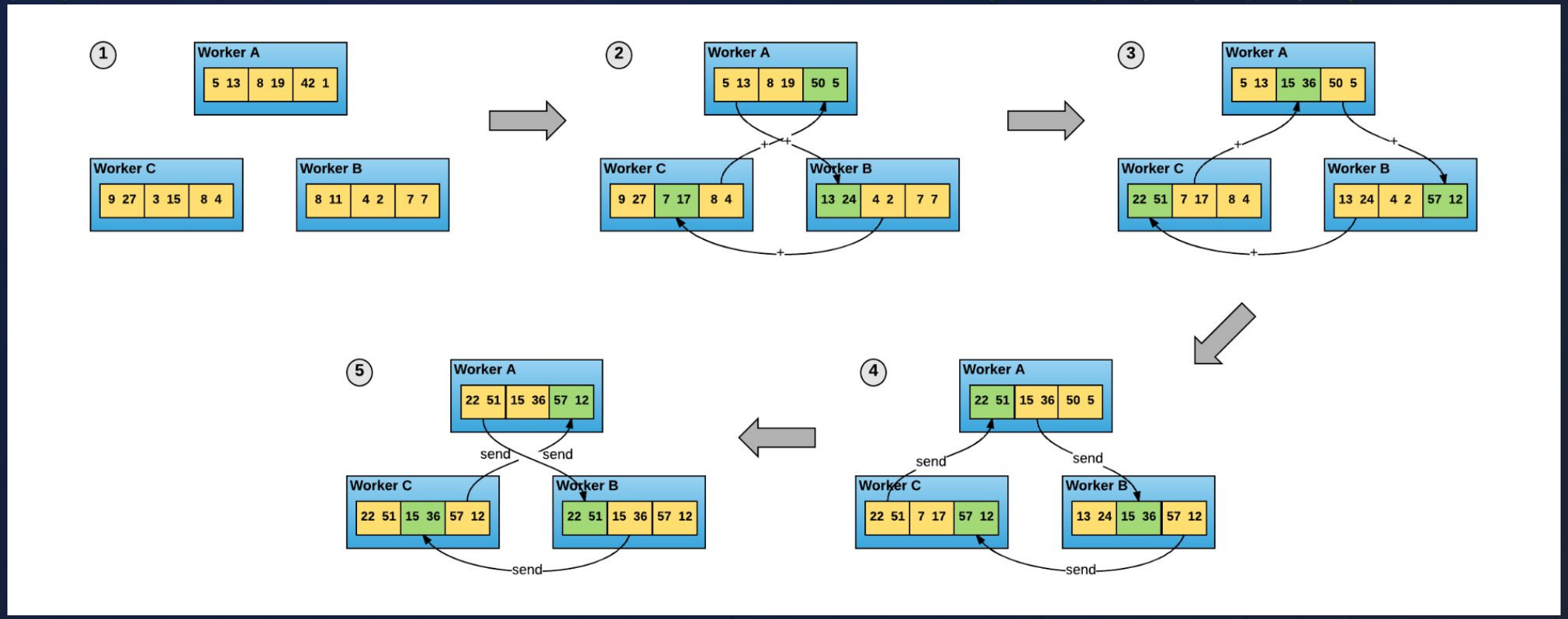
分布式模型训练

- 加快训练速度
  - PS-Worker
  - RingAllReduce



分布式模型训练

- 加快训练速度
  - PS-Worker
  - RingAllReduce



分布式模型训练

```
# On ps0.example.com:
$ python trainer.py \
     --ps_hosts=ps0.example.com:2222,ps1.example.com:2222 \
     --worker_hosts=worker0.example.com:2222,worker1.example.com:2222
\    --job_name=ps --task_index=0
# On ps1.example.com:
$ python trainer.py \
     --ps_hosts=ps0.example.com:2222,ps1.example.com:2222 \
     --worker_hosts=worker0.example.com:2222,worker1.example.com:2222
\    --job_name=ps --task_index=1
# On worker0.example.com:
$ python trainer.py \
     --ps_hosts=ps0.example.com:2222,ps1.example.com:2222 \
     --worker_hosts=worker0.example.com:2222,worker1.example.com:2222
\    --job_name=worker --task_index=0
# On worker1.example.com:
$ python trainer.py \
     --ps_hosts=ps0.example.com:2222,ps1.example.com:2222 \
     --worker_hosts=worker0.example.com:2222,worker1.example.com:2222
\    --job_name=worker --task_index=1
```

分布式模型训练

```yaml
apiVersion: "kubeflow.org/v1beta1"
kind: "TFJob"
metadata:
  name: "dist-mnist"
spec:
  tfReplicaSpecs:
    PS:
      replicas: 2
      restartPolicy: Never
      template:
        spec:
          containers:
            - name: tensorflow
              image: kubeflow/mnist:1.0
    Worker:
      replicas: 4
      restartPolicy: Never
      template:
        spec:
          containers:
            - name: tensorflow
              image: kubeflow/mnist:1.0
```

分布式模型训练

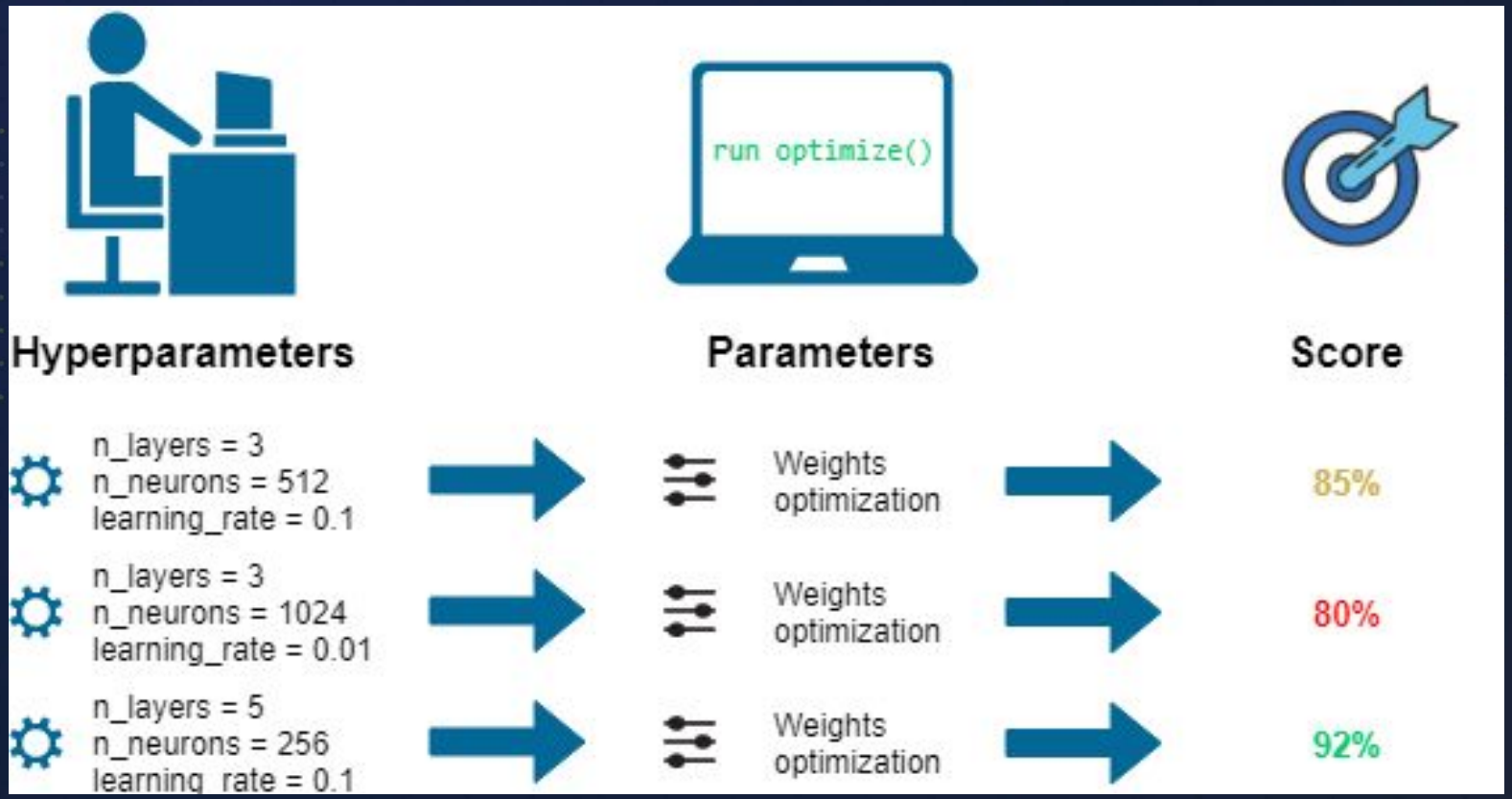
TF_CONFIG 的格式

`TF_CONFIG` 环境变量是一个 JSON 字符串，格式如下：

键	说明
"cluster"	TensorFlow 集群描述。此对象的格式设置为 TensorFlow 集群规范，并且可以传递给 tf.train.ClusterSpec 的构造函数。
"task"	描述运行代码的特定节点的任务。您可以使用此信息为分布式作业中的特定工作器编写代码。此条目是一个含有以下键的字典：
	"type" 此节点执行的任务类型。可能的值包括 **master**、**worker** 和 **ps**。
	"index" 任务的索引（从零开始）。大多数分布式训练作业都有一个主任务、一个或多个参数服务器，以及一个或多个工作器。
	"trial" 当前运行的超参数调节试验的标识符。为作业配置超参数调节时，可以设置多个要训练的试验。此值为您提供了一种区分代码中正在运行的试验的方法。标识符是一个包含试验编号的字符串值，从 1 开始。
"job"	启动作业时使用的作业参数。在大多数情况下，您可以忽略此条目，因为其中包含的数据与通过命令行参数传递给应用的数据完全一致。

超参数搜索

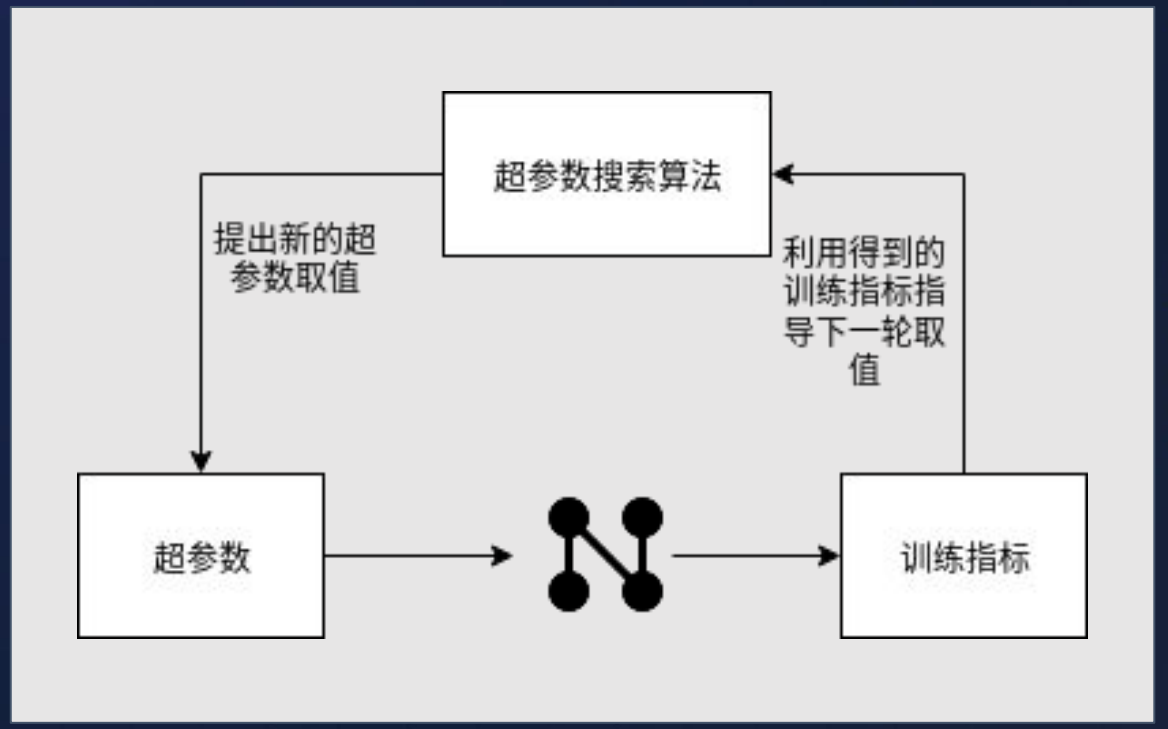
超参数：由用户在训练之前指定的参数，如神经网络中的 Learning Rate，Optimizer，Batch Size 等



Credits: https://towardsdatascience.com/hyperparameters-optimization-526348bb8e2d

超参数搜索

经典的黑盒优化问题

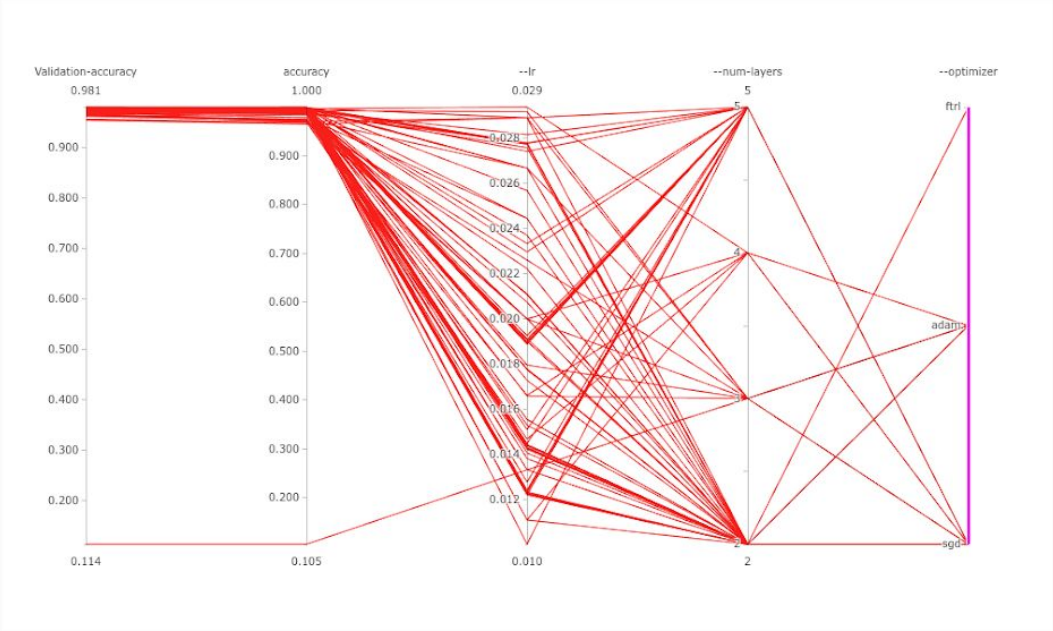


超参数搜索

```yaml
apiVersion: "kubeflow.org/v1alpha3"
kind: Experiment
metadata:
  name: example
spec:
  objective:
    type: maximize
    goal: 0.99
    objectiveMetricName: accuracy
    additionalMetricNames:
      - accuracy
  algorithm:
    algorithmName: bayesianoptimization
  parallelTrialCount: 3
  maxTrialCount: 12
  maxFailedTrialCount: 3
  parameters:
    - name: --num-layers
      parameterType: int
      feasibleSpace:
        min: "2"
        max: "5"
    - name: --optimizer
      parameterType: categorical
      feasibleSpace:
        list:
        - sgd
        - adam
        - ftrl
  trialTemplate:
    ...
```

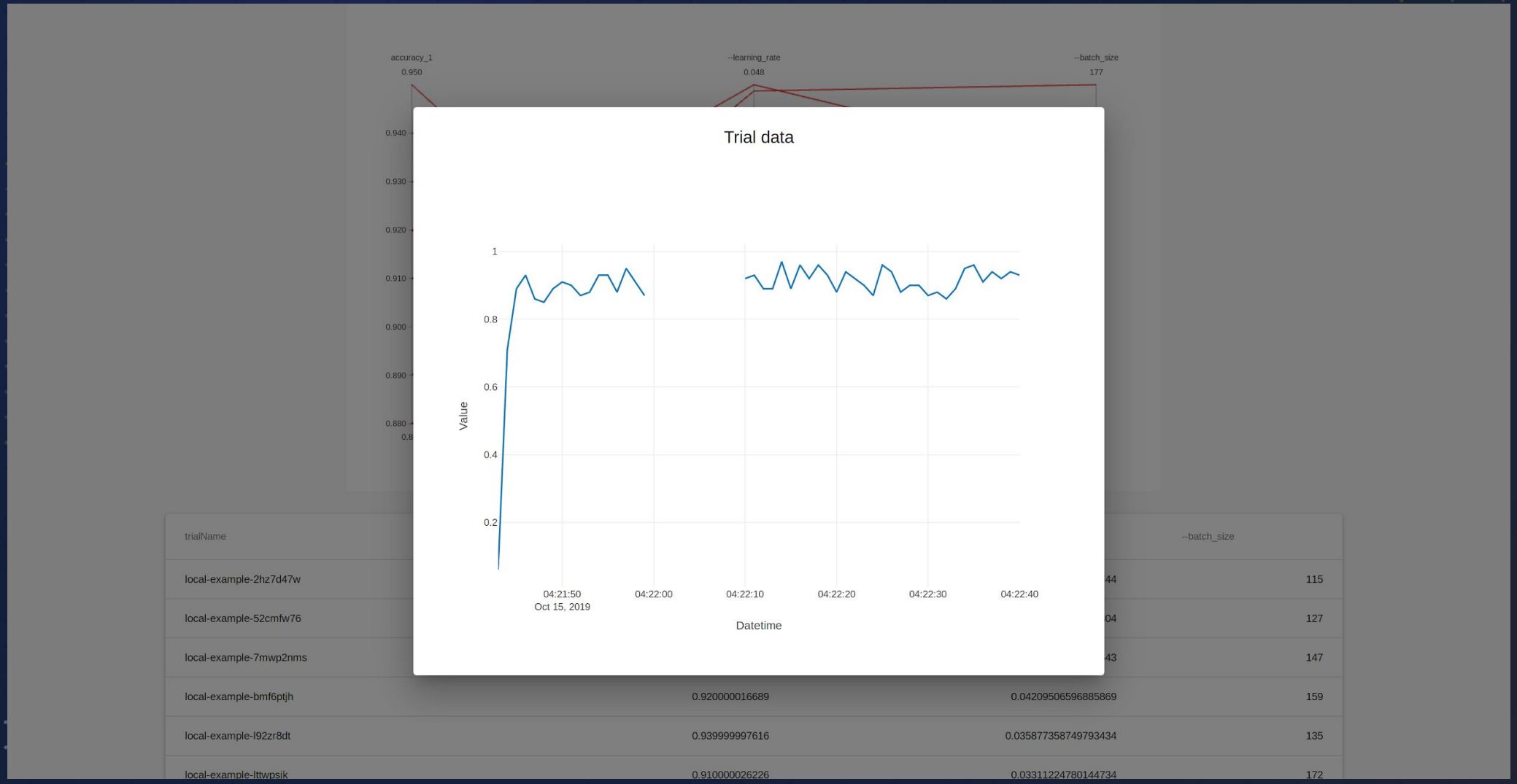
超参数搜索

Experiment Name: mnist-experiment



trialName	Validation-accuracy	accuracy	--lr	--num-layers	--optimizer
mnist-experiment-5p7tvdgn	0.979299	0.992031	0.0179615419877716 9	3	sgd
mnist-experiment-ccdgdp5v	0.981190	0.998906	0.014491188430762471	5	sgd
mnist-experiment-n24h6lwd	0.976712	0.992656	0.02915797314372763	3	sgd
mnist-experiment-lwf8dq49	0.962878	0.984062	0.017791967922146663	2	adam
mnist-experiment-2qv4mtsf	0.964869	0.974688	0.014689376618494194	4	adam
mnist-experiment-msdlqrjp	0.978603	0.994687	0.01775280397488328	2	sgd
mnist-experiment-pcr92pt2	0.957404	0.967969	0.022601601543350176	3	adam
mnist-experiment-njwhd2tr	0.977309	0.996719	0.029364679387401435	4	sgd

超参数搜索


accuracy_1
0.950
--learning_rate
0.048
--batch_size
177
0.940
0.930
0.920
0.910
0.900
0.890
0.880
Trial data
Value
1
0.8
0.6
0.4
0.2
04:21:50
Oct 15, 2019
04:22:00
04:22:10
04:22:20
04:22:30
04:22:40
Datetime
trialName
--batch_size
local-example-2hz7d47w
115
local-example-52cmfw76
127
local-example-7mwp2nms
147
local-example-bmf6ptjh
0.920000016689
0.04209506596885869
159
local-example-l92zr8dt
0.939999997616
0.035877358749793434
135
local-example-lttwpsjk
0.910000026226
0.03311224780144734
172

分布式训练总结

- 基于 Kubernetes 的大规模训练和超参数搜索可行，但资源利用率等问题仍有很大优化空间

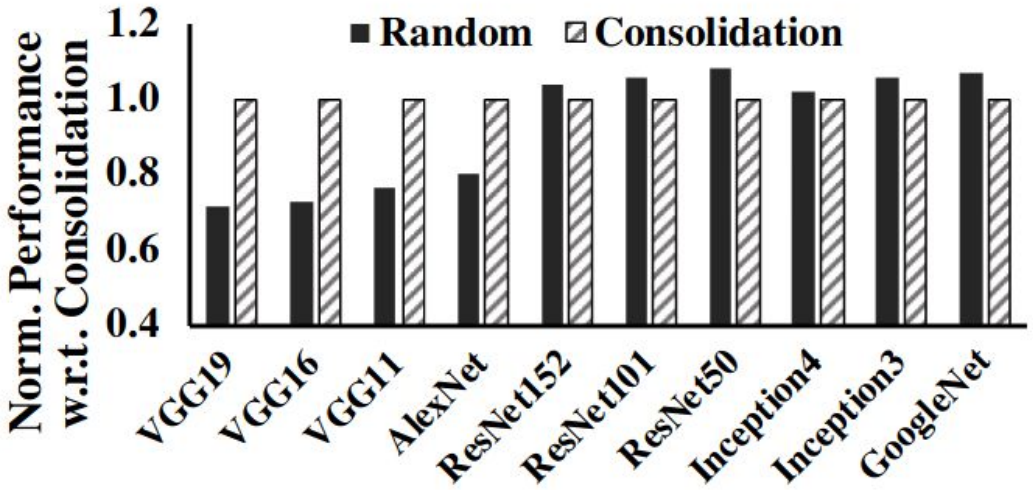
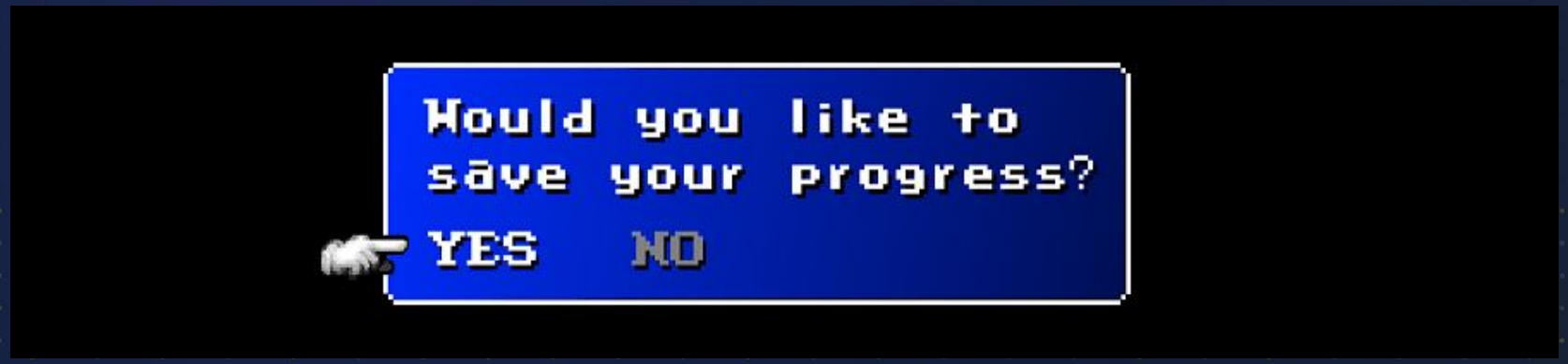


Figure 3: 4 concurrent 8-worker jobs with different placement schemes. The performance values are normalized by the value of the consolidation scheme. We use the median value from 10 (20) runs for consolidation (random) scheme.

Credits:Tiresias: A GPU Cluster Manager for Distributed Deep Learning

可容错的训练

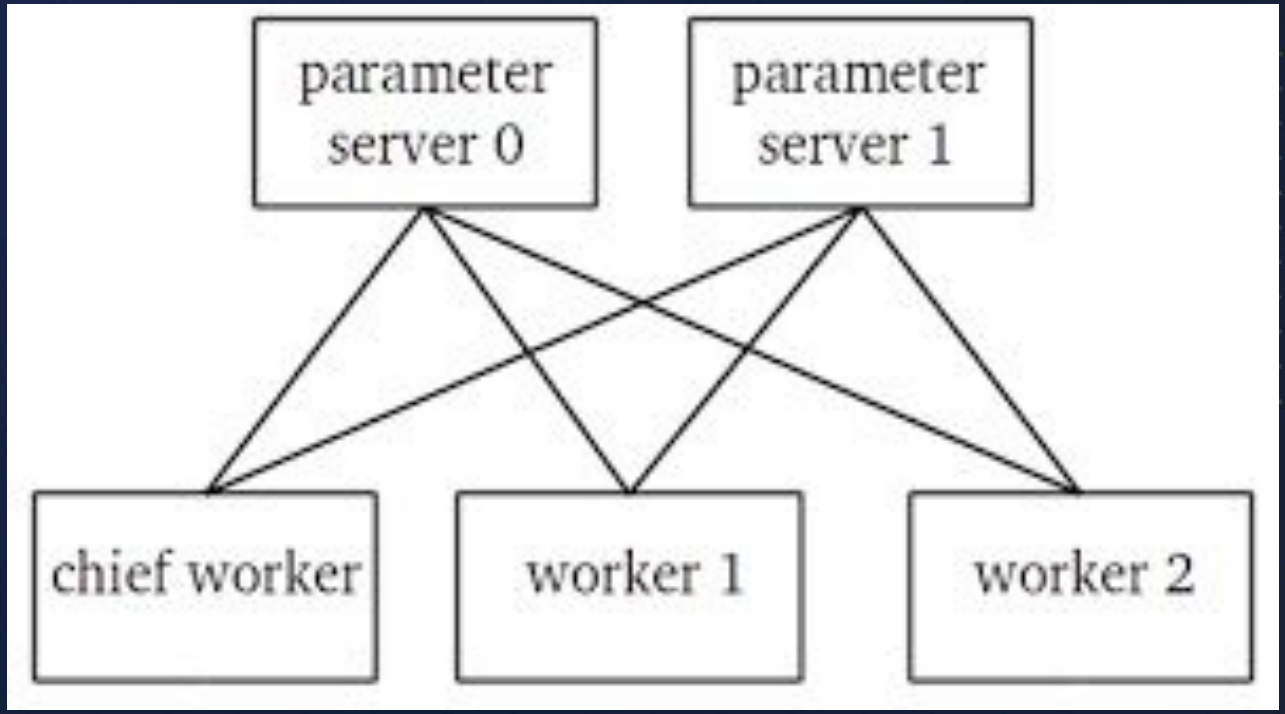



```
net.save_weights('easy_checkpoint')
```

可容错的训练

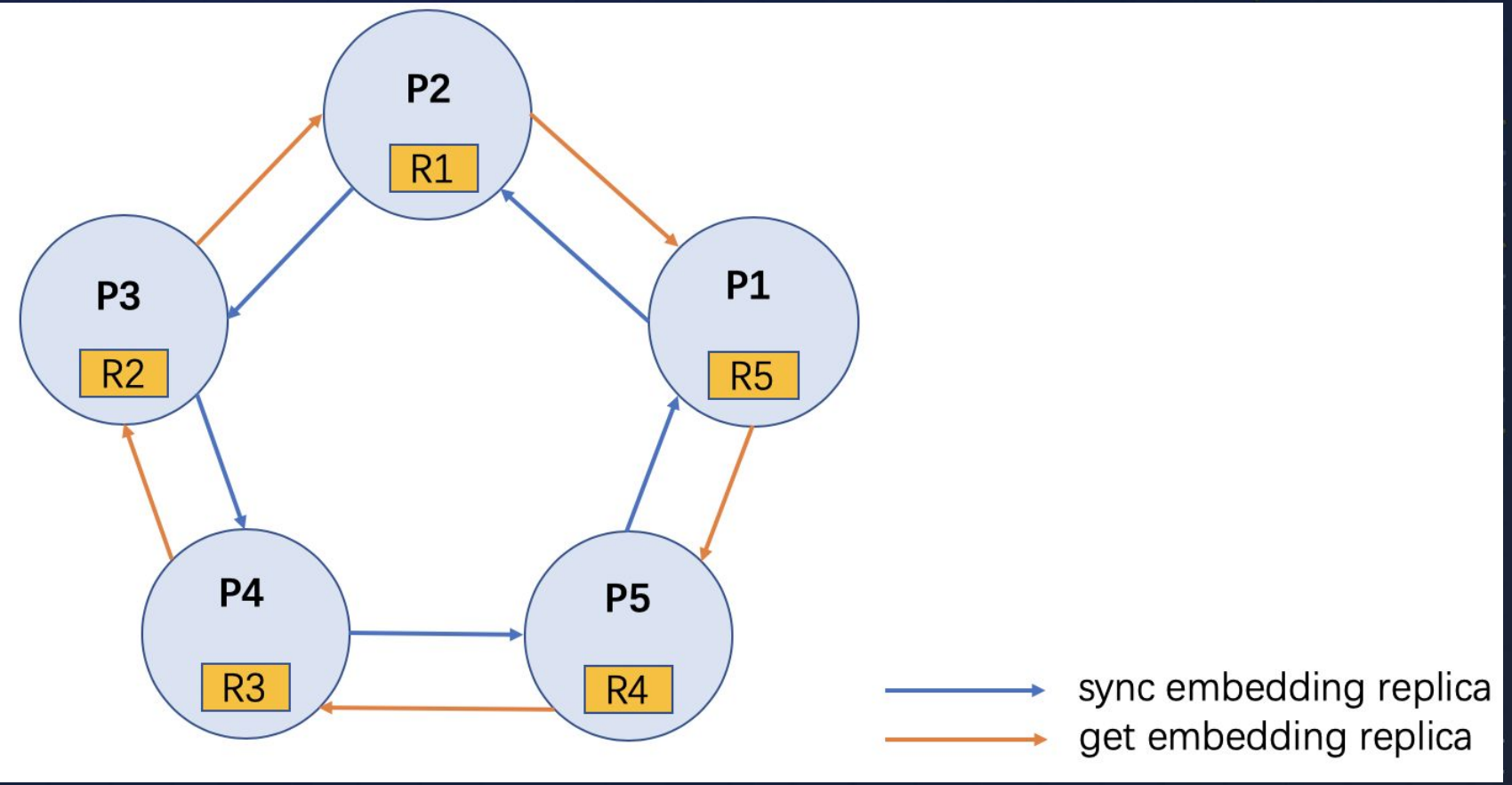
```
tf.compat.v1.train.MonitoredTrainingSession(
    master='',
    is_chief=True,
    checkpoint_dir=None,
    scaffold=None,
    hooks=None,
    chief_only_hooks=None,
    save_checkpoint_secs=USE_DEFAULT,
    save_summaries_steps=USE_DEFAULT,
    save_summaries_secs=USE_DEFAULT,
    config=None,
    stop_grace_period_secs=120,
    log_step_count_steps=100,
    max_wait_secs=7200,
    save_checkpoint_steps=USE_DEFAULT,
    summary_dir=None
)
```

可容错的训练


parameter server 0
parameter server 1
chief worker
worker 1
worker 2

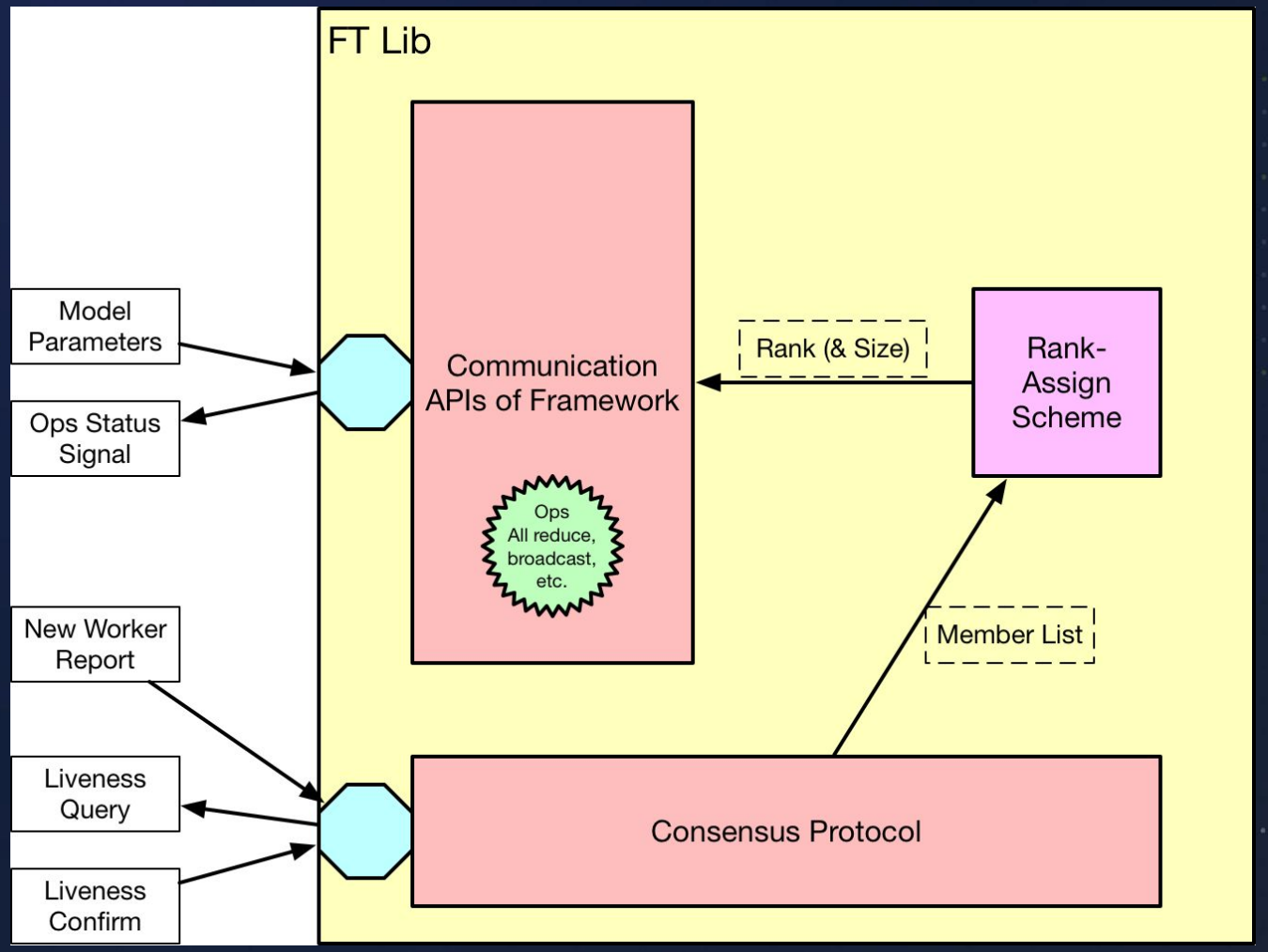
可容错的训练-PS/Worker

ElasticDL



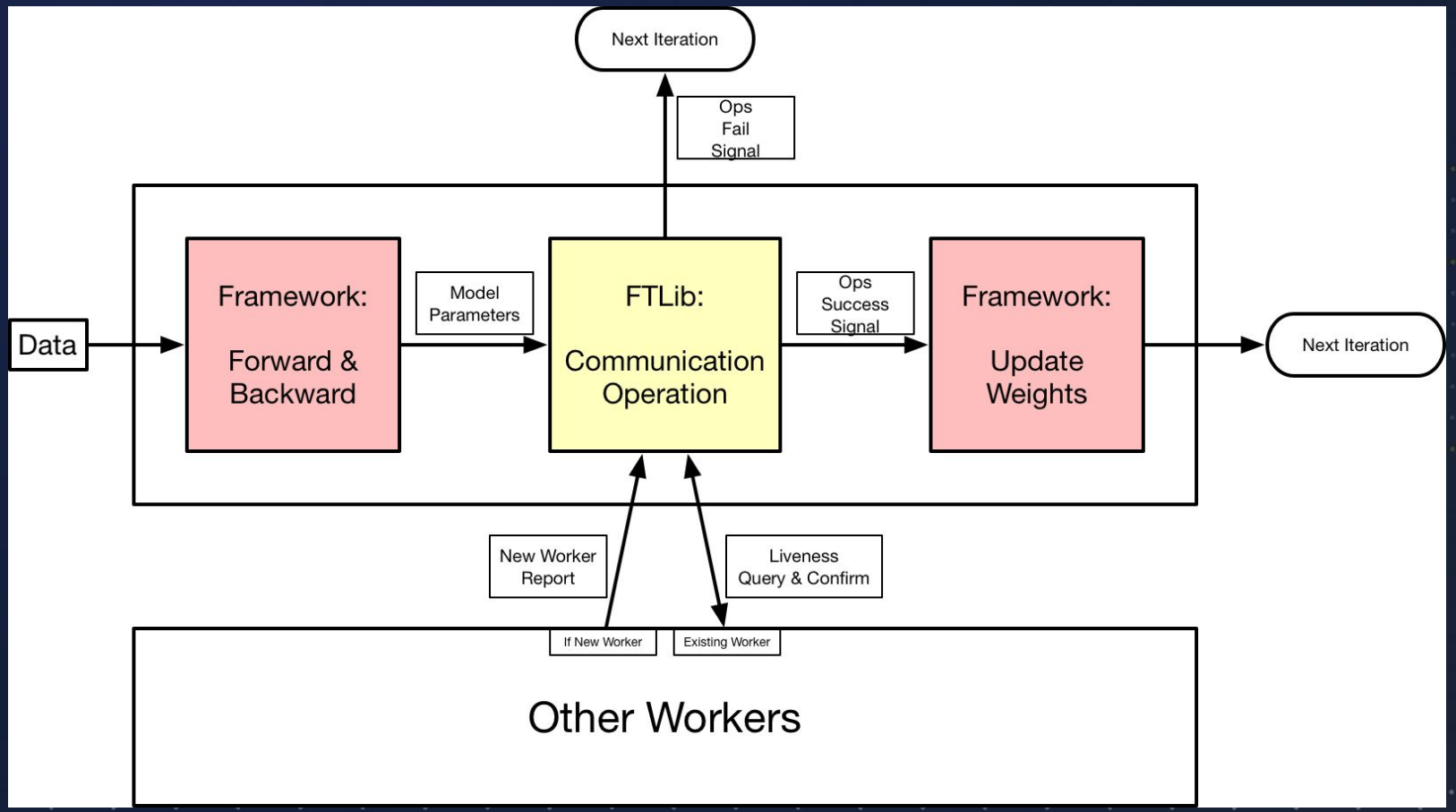
可容错的训练-RingAllReduce

https://github.com/caicloud/ftlib/



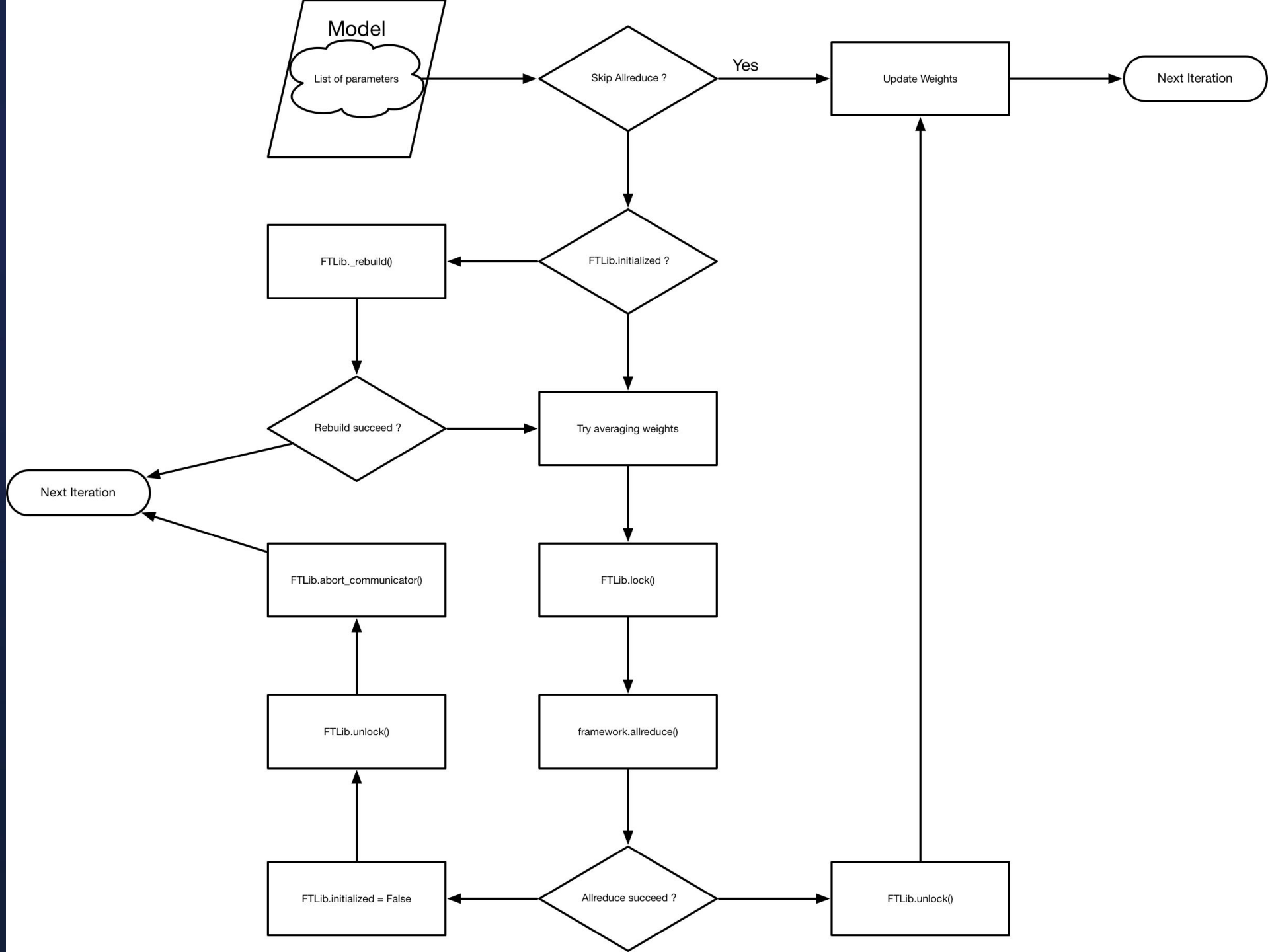
可容错的训练-RingAllReduce

https://github.com/caicloud/ftlib/



技术分享

可容错的训练


Model
List of parameters
Skip Allreduce ?
Yes
Update Weights
Next Iteration
FTLib._rebuild()
FTLib.initialized ?
Rebuild succeed ?
Try averaging weights
Next Iteration
FTLib.abort_communicator()
FTLib.lock()
FTLib.unlock()
framework.allreduce()
FTLib.initialized = False
Allreduce succeed ?
FTLib.unlock()

可容错的训练-RingAllReduce

一致性：SWIM-based Membership

A. Das, I. Gupta and A. Motivala, "SWIM: scalable weakly-consistent infection-style process group membership protocol," Proceedings International Conference on Dependable Systems and Networks, Washington, DC, USA, 2002, pp. 303-312.
doi: 10.1109/DSN.2002.1028914
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1028914&isnumber=22107

可容错的训练-总结

- 容错在大规模分布式训练中必不可少
- PS/Worker 和 RingAllReduce 需要不同的容错实现(?)
- 容错是弹性训练的前提

总结

- 硬件加速器资源的共享仍然是一个值得研究的领域
- 大规模，可容错的训练可行但具有侵入性
- 资源利用率的优化

THANK YOU!

Caicloud.io