

超大规模训练调度优化实践

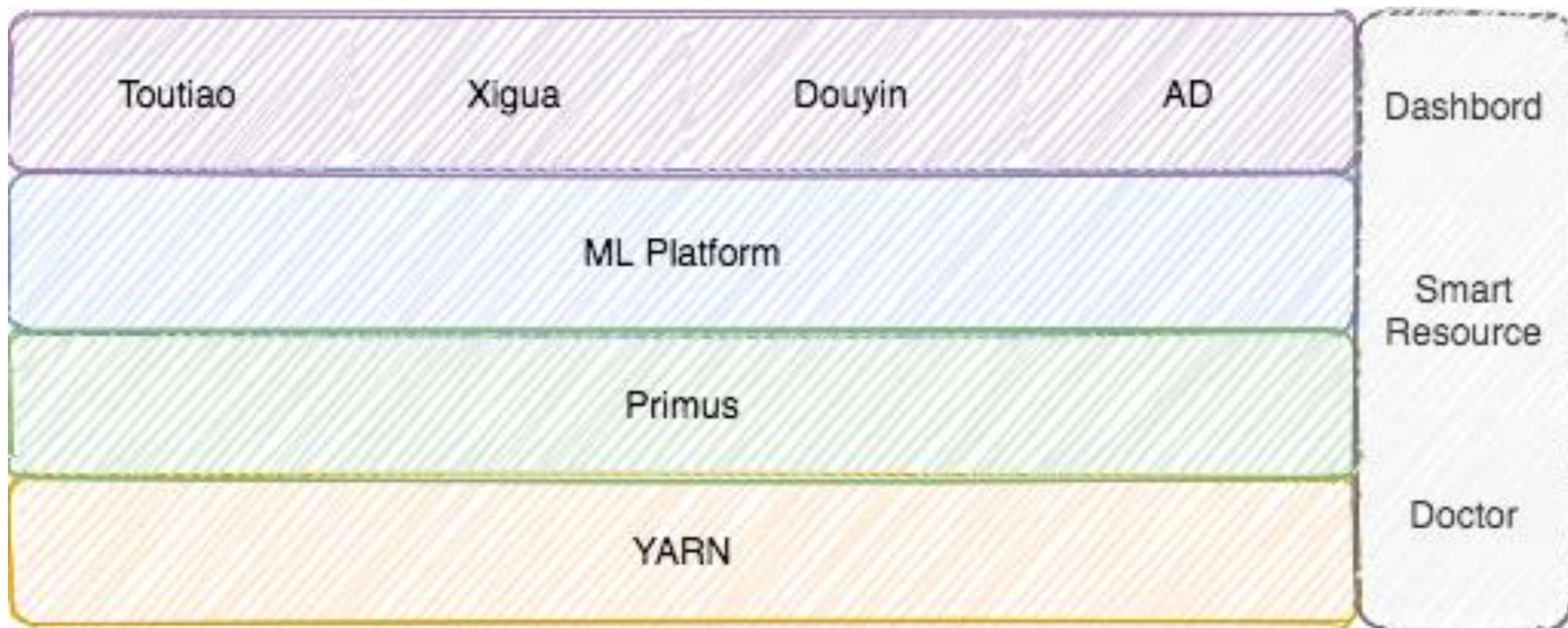
目录

- 背景
- YARN 针对训练场景的优化
 - 资源类型
 - 调度器
 - 隔离及性能
 - 稳定性
 - 混部
- PRIMUS 机器学习训练调度框架
 - 多框架支持
 - 多角色支持
 - 编排调度
 - 混部优化

背景

背景

整体架构



挑战

- 编排调度需求复杂
- 性能、稳定性要求高
- 部署环境复杂
- 资源严重短缺
- 多种深度学习框架需求各不相同
- 单作业存在多种角色，不同角色间的资源配置和调度需求差距很大

YARN 针对训练场景的优化

- 资源类型
- 调度器
- 隔离及性能
- 稳定性
- 混部

YARN 训练规模

YARN 训练规模

物理资源
上百万核心

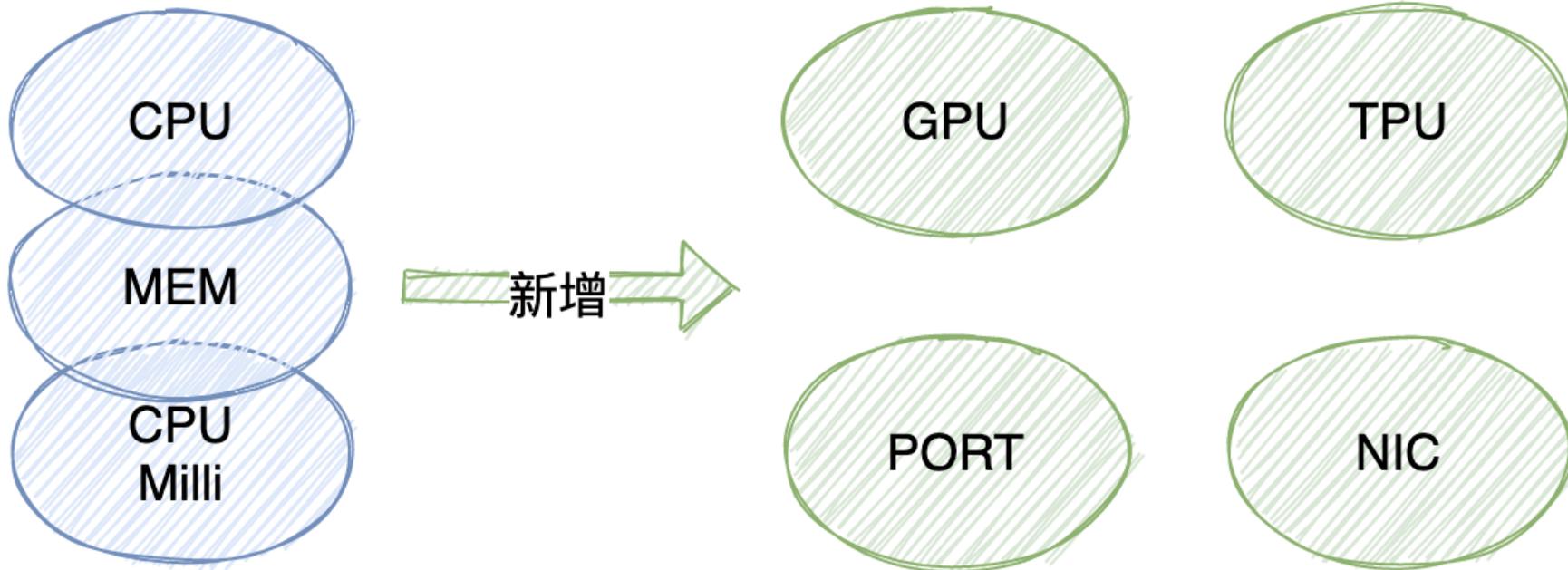
作业数量
每日上千个

最大单集群
5W 节点

YARN 资源类型

YARN 新增资源类型:

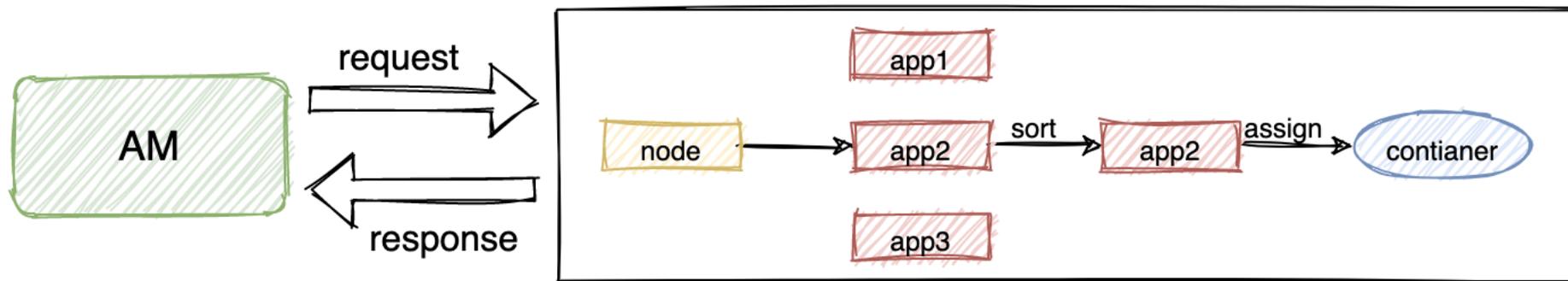
- GPU
- TPU
- PORT
- NIC



YARN 调度器

Fair Scheduler

- 调度由节点心跳触发
- 每次调度分配有限个 container



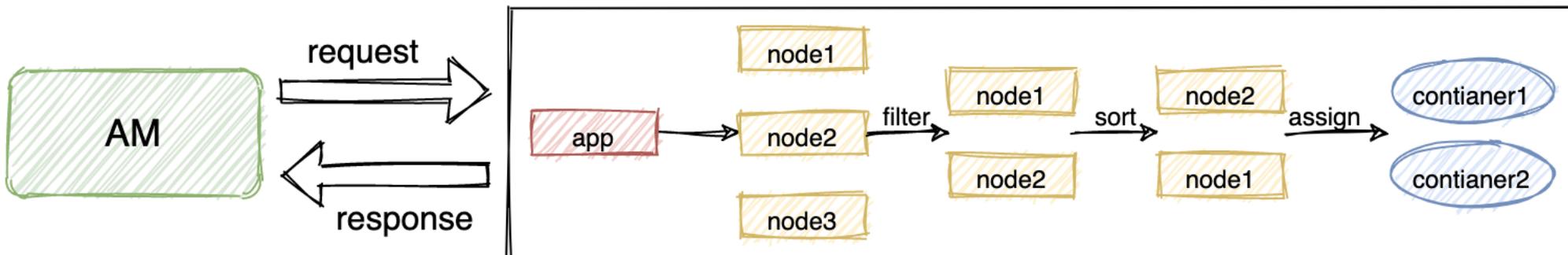
问题:

- 资源不足时，训练作业 pending 不如直接失败
- 没有全局视角，不能满足各种特定调度需求

YARN 调度器

Unified Scheduler

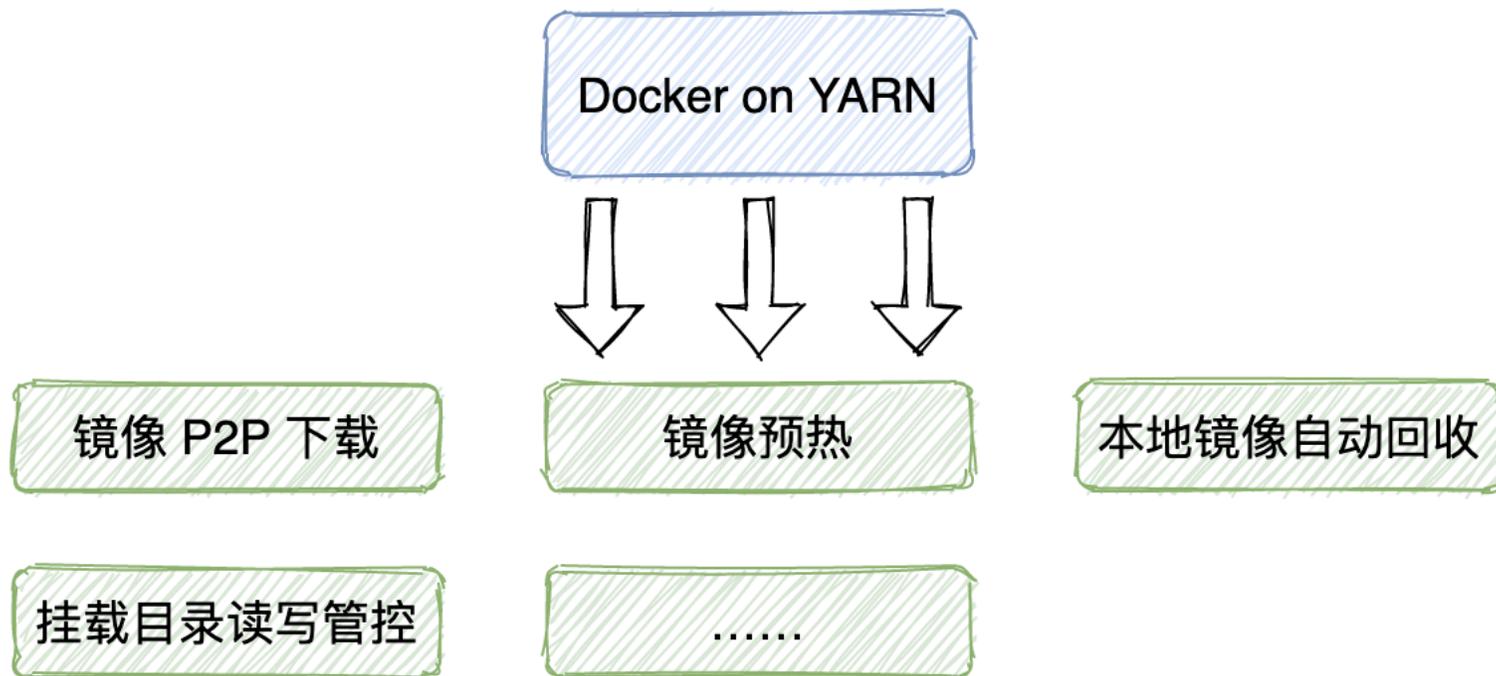
- 提供 All or Nothing 调度语义
- 全局视角
 - 丰富的调度约束
 - 约束类型：强约束（必须满足） & 弱约束（尽量满足）
 - 全局：quota 平均、quota 集中
 - 作业：container 打散、container 集中、host unique、整机调度
 - 节点：节点属性、高 load 跳过



YARN 隔离及性能

Docker on YARN: 解决复杂依赖环境问题

- 提升作业启动速度: 镜像 P2P 下载、镜像预热
- 管控本地磁盘: 本地镜像自动回收, 挂载目录读写管控



训练作业已全量上线 Docker on YARN。

YARN 隔离及性能

CPU 隔离及性能优化



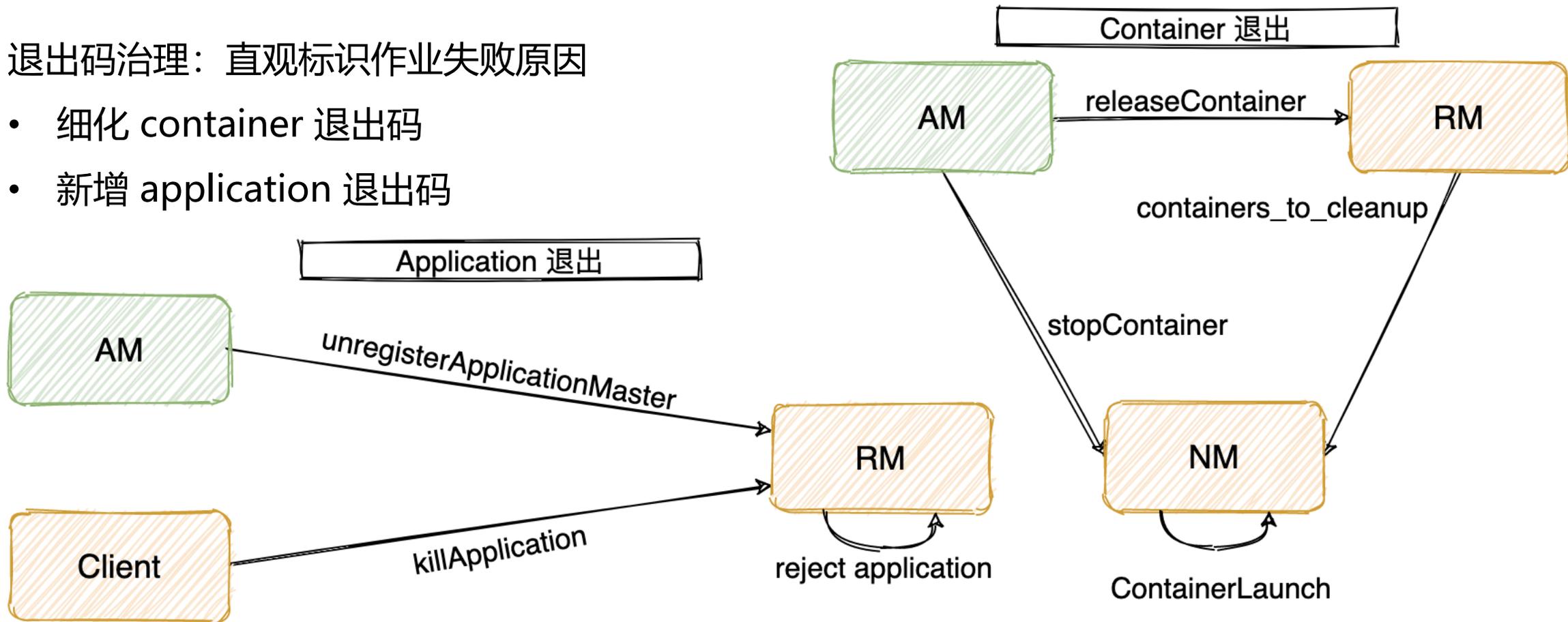
- CPU Strict: 保证每个容器的 CPU 时间片
- NUMA: 绑定 NUMA, 减少跨 socket 开销, 性能提升 20% 以上
- CPUSET: 绑核, 减少上下文切换开销和减少 cache miss 等, 性能提升 30% 以上

YARN 稳定性

Health Check、黑名单、机器巡检、.....

退出码治理：直观标识作业失败原因

- 细化 container 退出码
- 新增 application 退出码



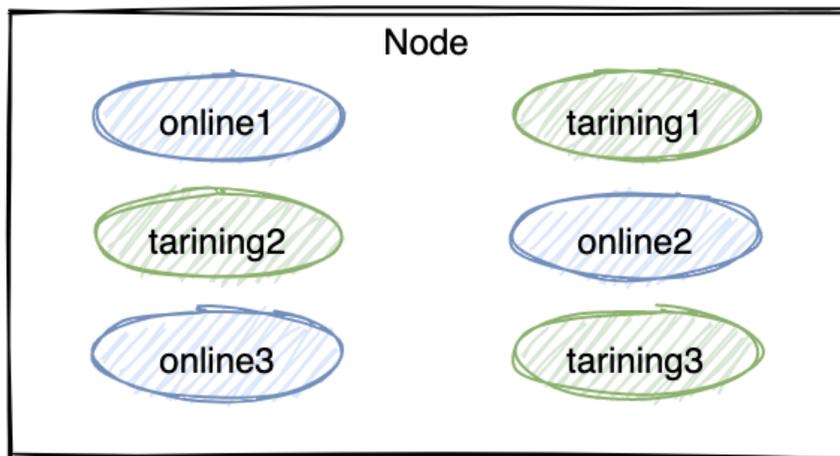
YARN 混部

如何解决资源短缺的问题?

通过大规模混部的方式, 使用其他业务 (在线/流式等) 的空闲资源。

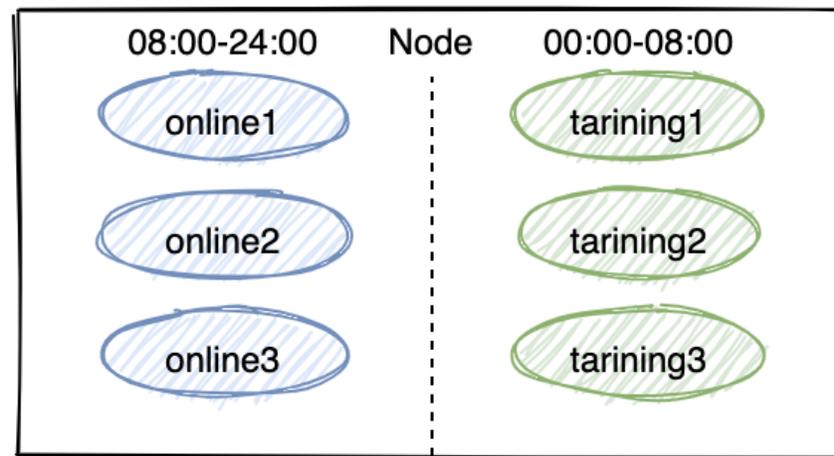
常态混部:

在线/流式和离线训练全天同时运行



错峰混部:

在线机器凌晨出让给离线训练



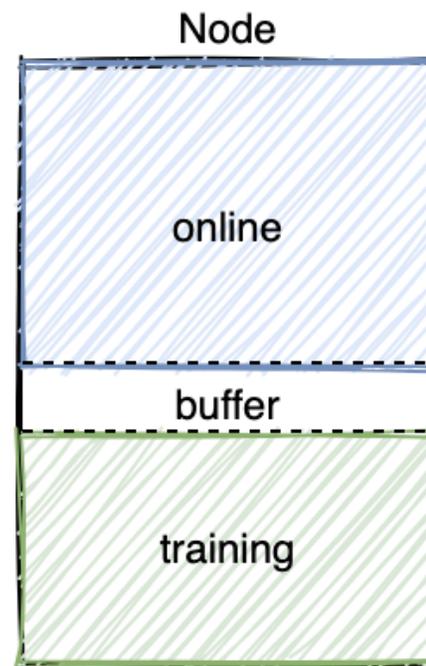
YARN 混部

混部主要工作：

- 单机多级资源隔离
- 动态资源 & 动态 quota
- 资源不足时驱逐混部 container
- 单集群 5W 节点
- 作业 pending 时间过长直接失败

为训练作业提供了充裕的资源，提升了机器利用率。

针对字节内部上线的多种混部方式，训练框架也做了相应优化。

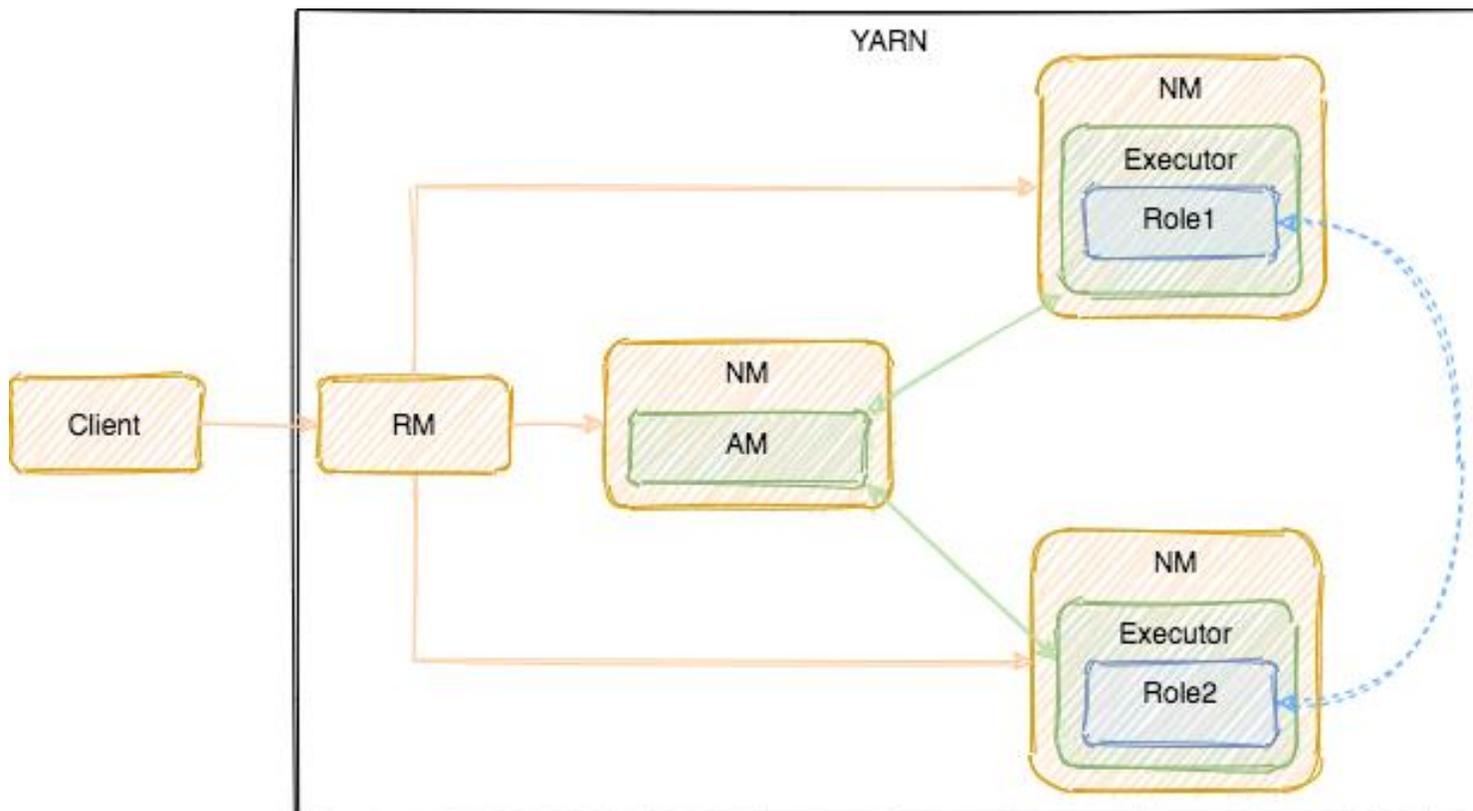


机器学习训练调度框架

- 多框架支持
- 多角色支持
- 编排调度
- 混部优化

Primus 简介

- 运行于Hadoop Yarn之上
- 负责机器学习框架的编排调度

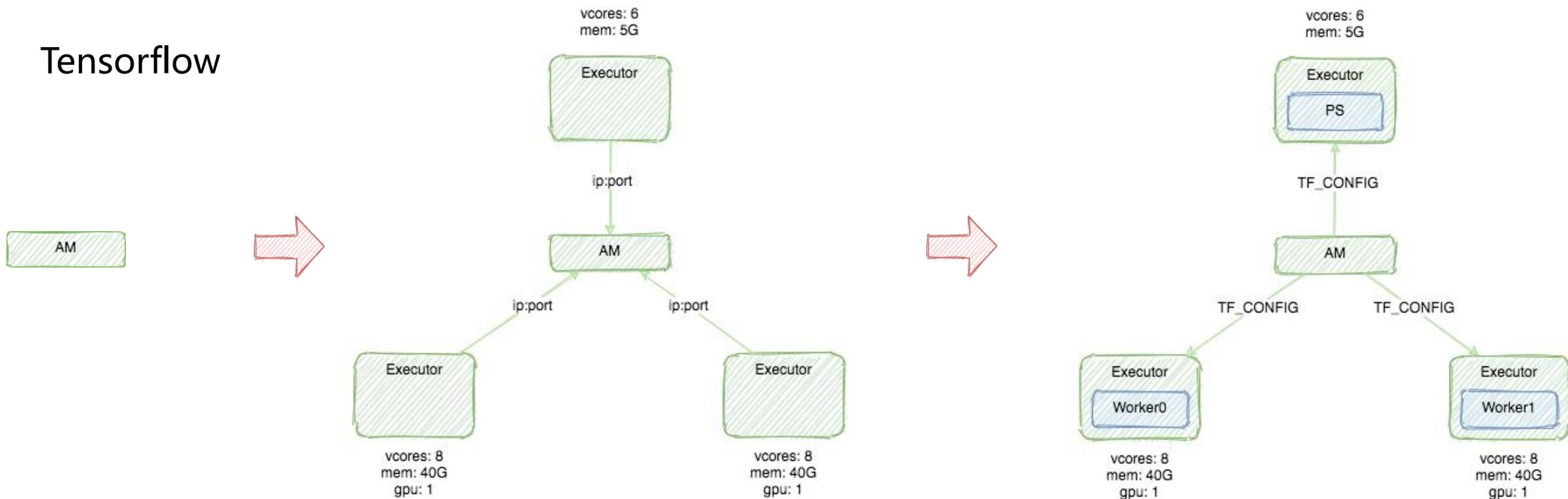


架构图

Primus 多框架

- 问题：如何用尽可能低的成本适配多种框架？
- 解决：
 - Plugin 方式，模块内有多个 hook 点

Tensorflow



Primus 自定义角色

- 问题：如何满足多种角色的需求？
- 解决：
 - Json 描述角色名、个数、资源、镜像、各种策略等
 - 根据不同角色使用不同的资源和调度配置

```
{
  "roleName": "chief",
  "num": 1,
  "vcores": 4,
  "memoryMb": 81472,
  "jvmMemoryMb": 3072,
  "command": "bash start.sh",
  "env": {
    "YARN_CONTAINER_RUNTIME_DOCKER_IMAGE": "xxxxxxxxxxxxxx",
    "YARN_CONTAINER_RUNTIME_TYPE": "docker"
  },
  "failover": {
    "commonFailoverPolicy": {
      "maxFailureTimes": 1,
      "maxFailurePolicy": "FAIL_ATTEMPT"
    }
  },
  "successPercent": 100
},
```

```
{
  "roleName": "worker",
  "num": 49,
  "vcores": 4,
  "memoryMb": 80960,
  "jvmMemoryMb": 3072,
  "command": "bash start.sh",
  "env": {
    "YARN_CONTAINER_RUNTIME_DOCKER_IMAGE": "xxxxxxxxxxxxxx",
    "YARN_CONTAINER_RUNTIME_TYPE": "docker"
  },
  "failover": {
    "commonFailoverPolicy": {
      "maxFailureTimes": 1,
      "maxFailurePolicy": "FAIL_ATTEMPT"
    }
  },
  "successPercent": 90
},
```

```
{
  "roleName": "ps",
  "num": 50,
  "vcores": 4,
  "memoryMb": 96592,
  "jvmMemoryMb": 3072,
  "command": "bash start.sh",
  "env": {
    "YARN_CONTAINER_RUNTIME_DOCKER_IMAGE": "xxxxxxxxxxxxxx",
    "YARN_CONTAINER_RUNTIME_TYPE": "docker"
  },
  "failover": {
    "commonFailoverPolicy": {
      "maxFailureTimes": 1,
      "maxFailurePolicy": "FAIL_ATTEMPT"
    }
  }
},
```

Primus 自定义角色

- 问题：如何满足多种角色的需求？
- 解决：
 - Json 描述角色名、个数、资源、镜像、各种策略等
 - 根据不同角色

```
{
  "roleName": "chief",
  "num": 1,
  "vcores": 4,
  "memoryMb": 81472,
  "jvmMemoryMb": 3072,
  "command": "bash start.sh",
  "env": {
    "YARN_CONTAINER_RUNTIME_DOCKER_IMAGE": "xxxxxxxx",
    "YARN_CONTAINER_RUNTIME_TYPE": "docker"
  },
  "failover": {
    "commonFailoverPolicy": {
      "maxFailureTimes": 1,
      "maxFailurePolicy": "FAIL_ATTEMPT"
    }
  },
  "successPercent": 100
},
```

```
{
  "roleName": "lagrange_worker",
  "num": 5,
  "vcores": 8,
  "memoryMb": 16384,
  "jvmMemoryMb": 1024,
  "command": "bash start.sh",
  "env": {
    "YARN_CONTAINER_RUNTIME_TYPE": "docker",
    "YARN_CONTAINER_RUNTIME_DOCKER_IMAGE": "xxxxxxxx",
  },
  "inputPolicy": "STREAMING",
  "failover": {
    "hybridDeploymentFailoverPolicy": {
      "commonFailover": {
        "restartType": "ALWAYS",
        "maxFailureTimes": 10,
        "maxFailurePolicy": "FAIL_ATTEMPT"
      }
    }
  },
  "successPercent": 100
},
```

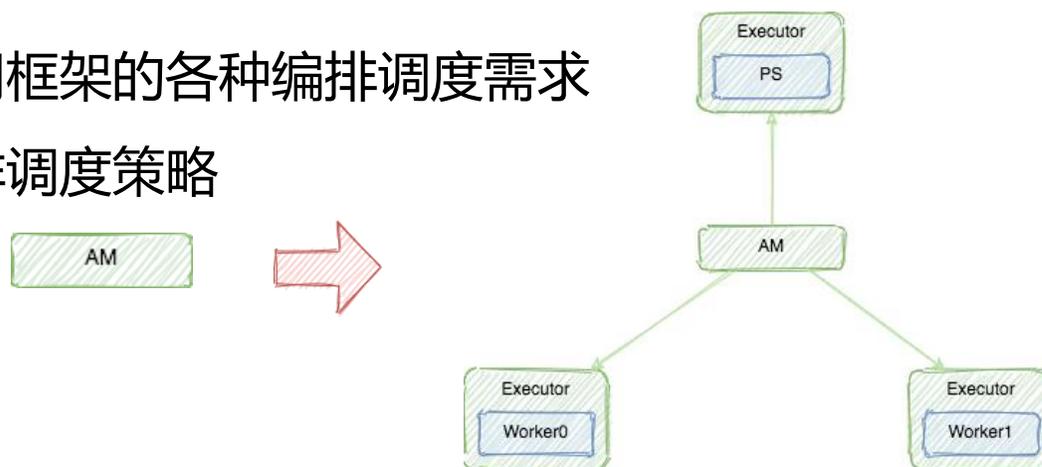
```
name": "ps",
: 50,
es": 4,
ryMb": 96592,
memoryMb": 3072,
and": "bash start.sh",
: {
RN_CONTAINER_RUNTIME_DOCKER_IMAGE": "xxxxxxxxxxxxxxxx",
RN_CONTAINER_RUNTIME_TYPE": "docker"
}
over": {
nmonFailoverPolicy": {
maxFailureTimes": 1,
maxFailurePolicy": "FAIL_ATTEMPT"
}
```

Primus 编排调度

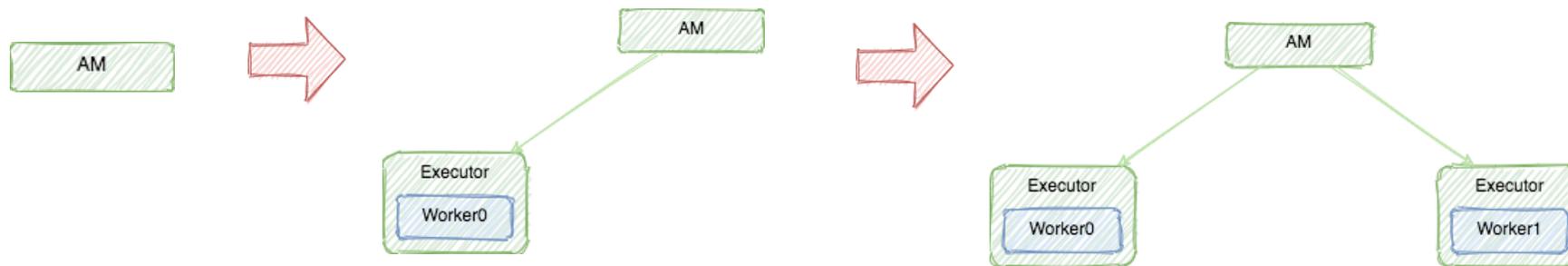
问题：如何满足不同框架的各种编排调度需求

解决：定义多种编排调度策略

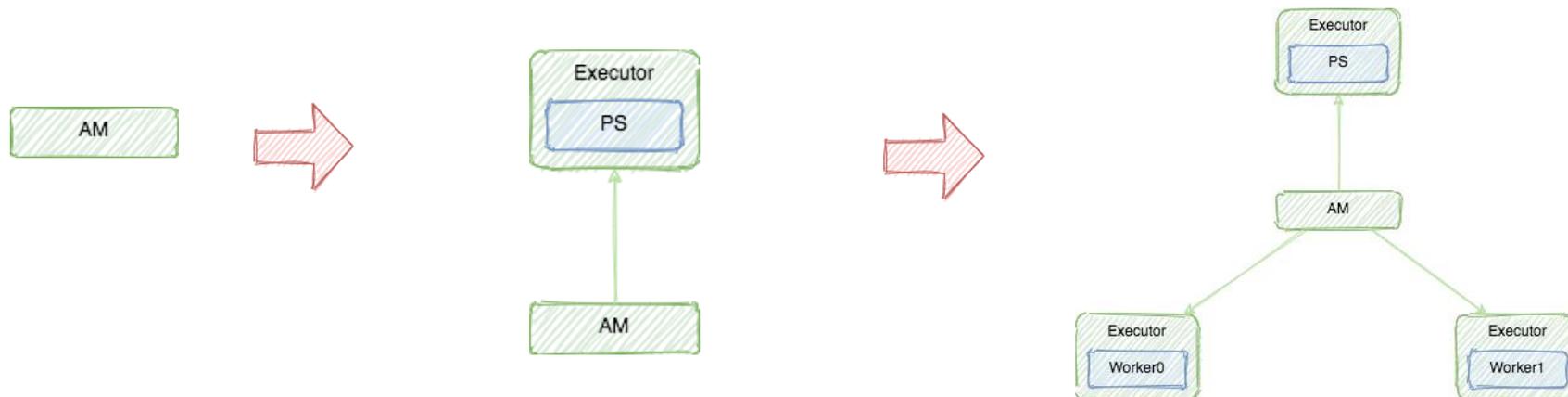
Gang



Dynamic



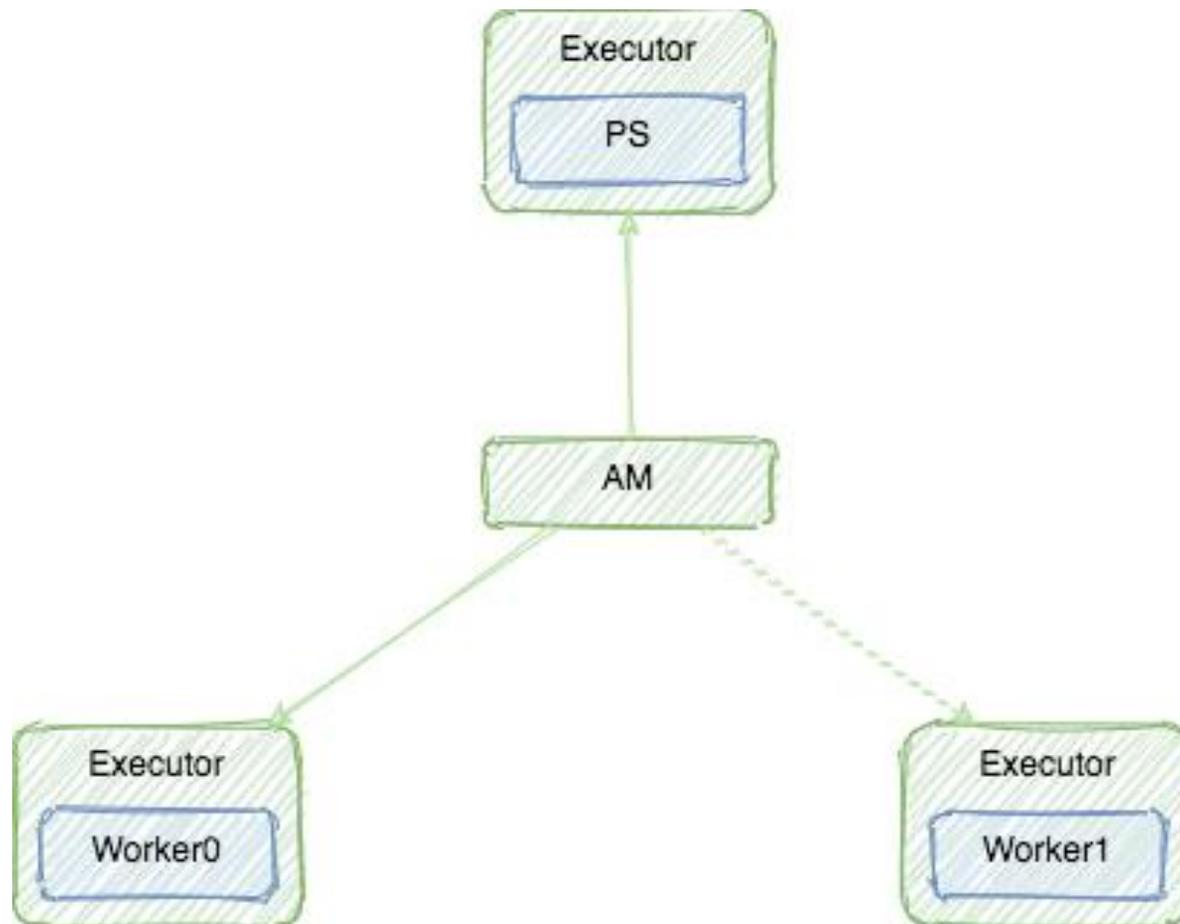
Order



Primus 编排调度

Failover策略

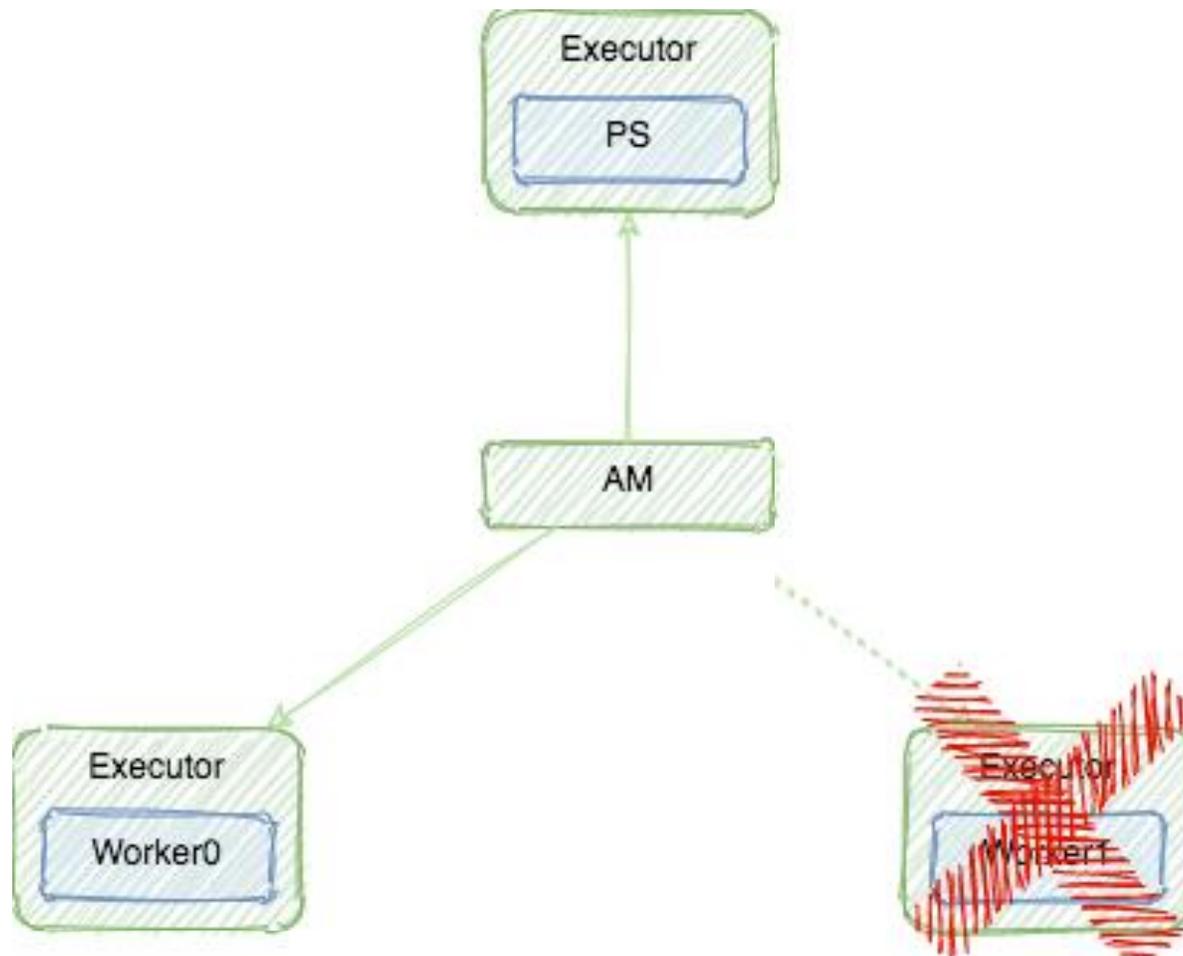
- **Never**
- Always
- OnFailure



Primus 编排调度

Failover策略

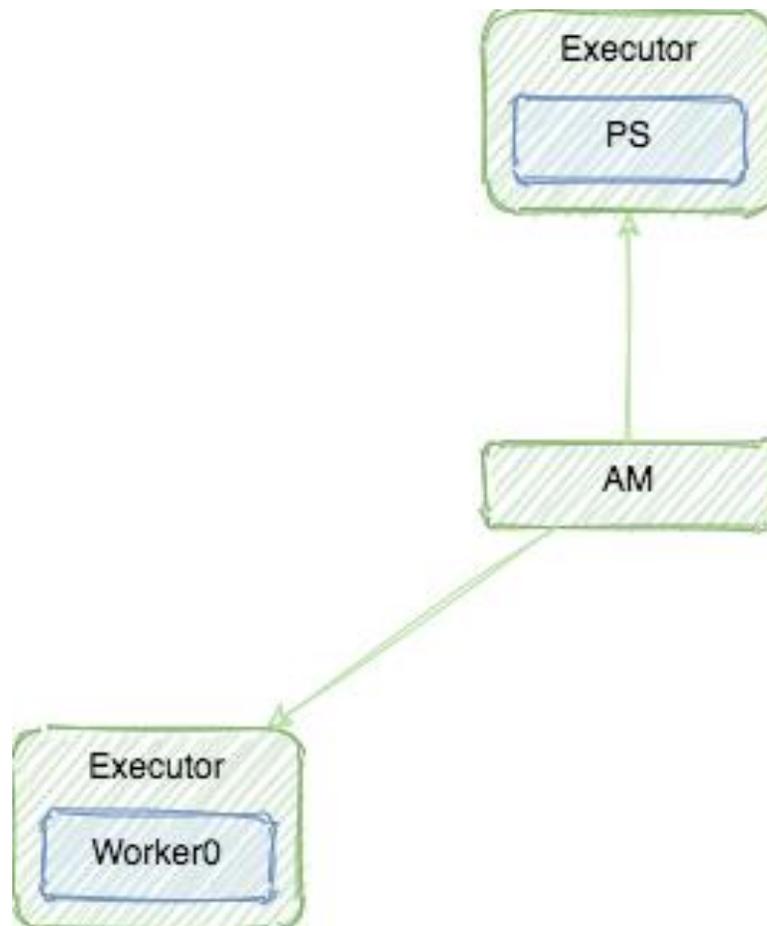
- **Never**
- Always
- OnFailure



Primus 编排调度

Failover策略

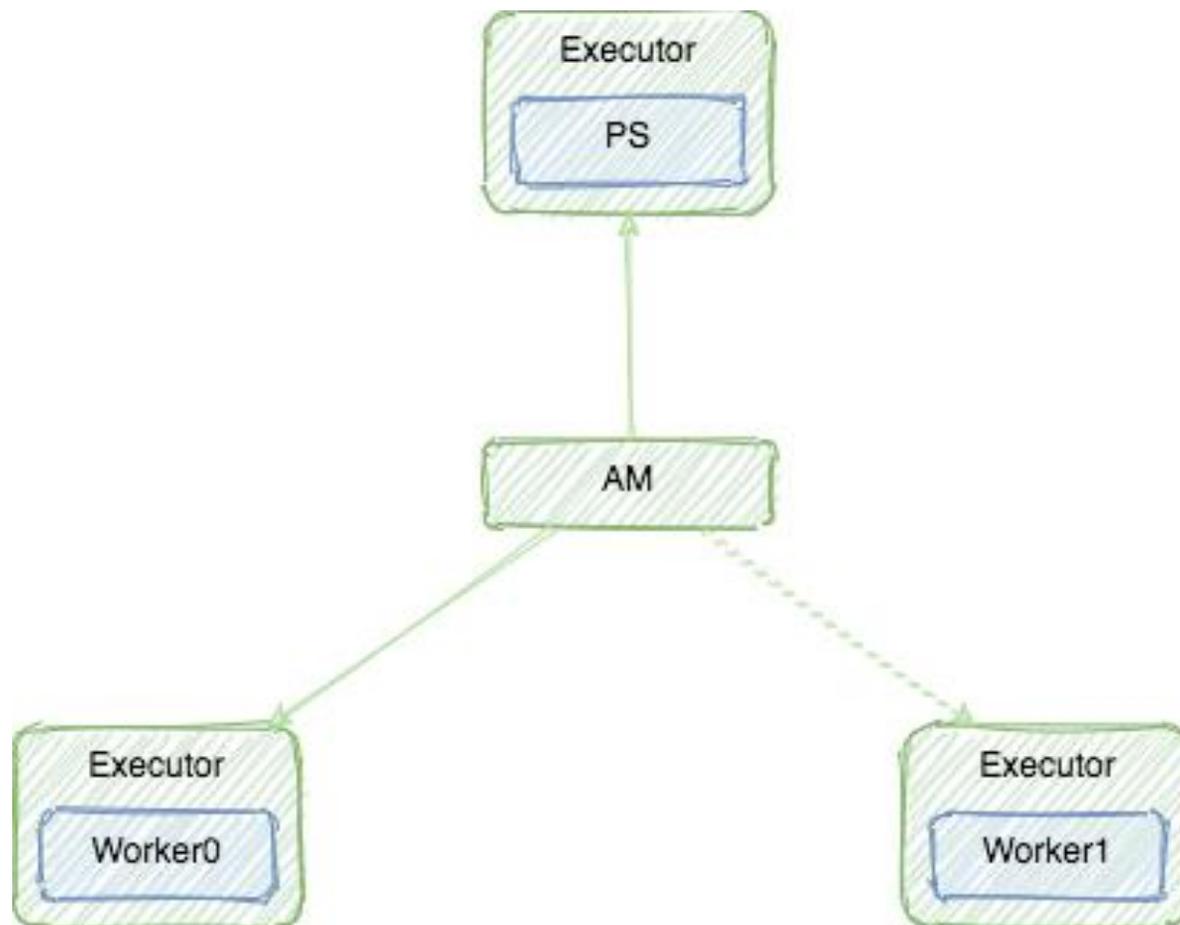
- **Never**
- Always
- OnFailure



Primus 编排调度

Failover策略

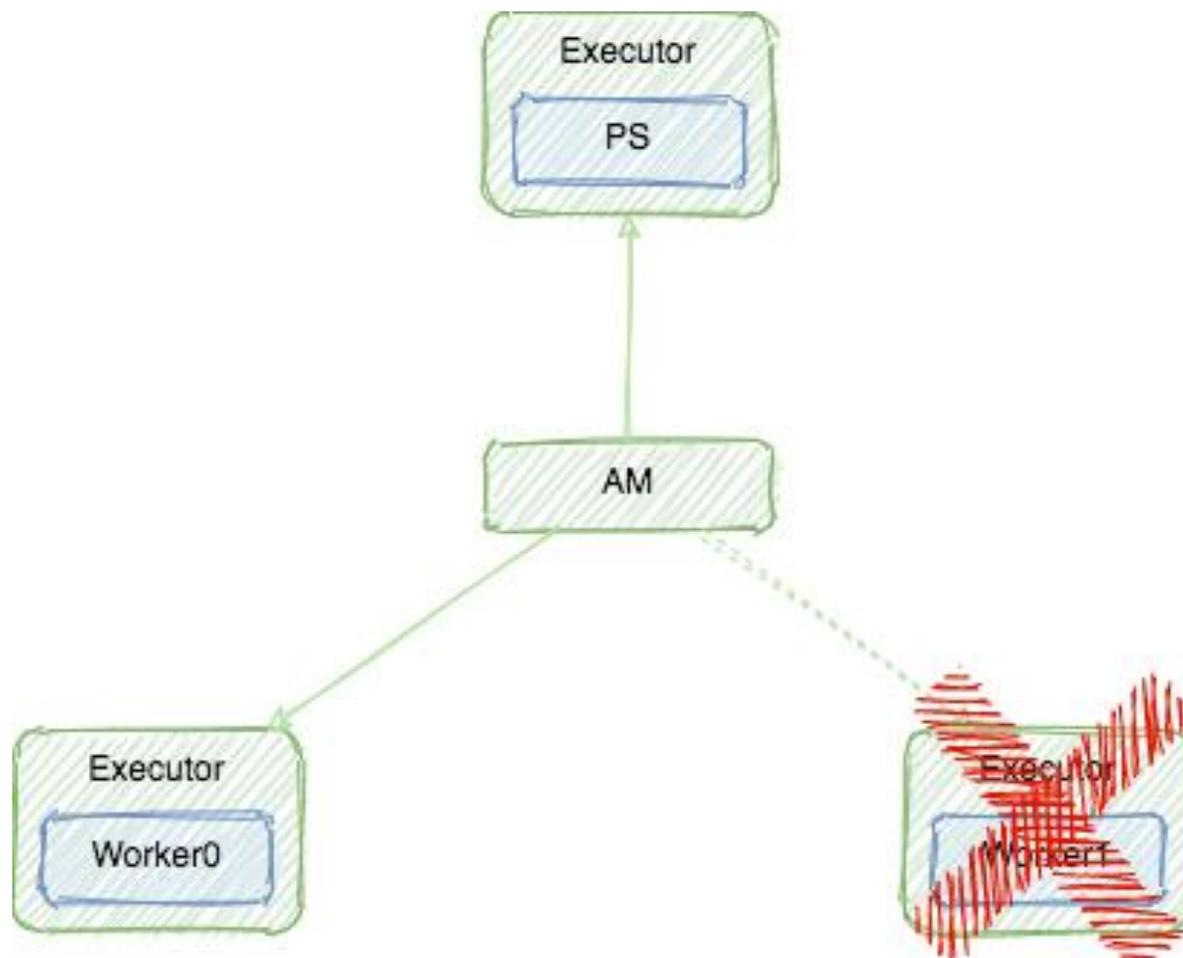
- Never
- **Always**
- OnFailure



Primus 编排调度

Failover策略

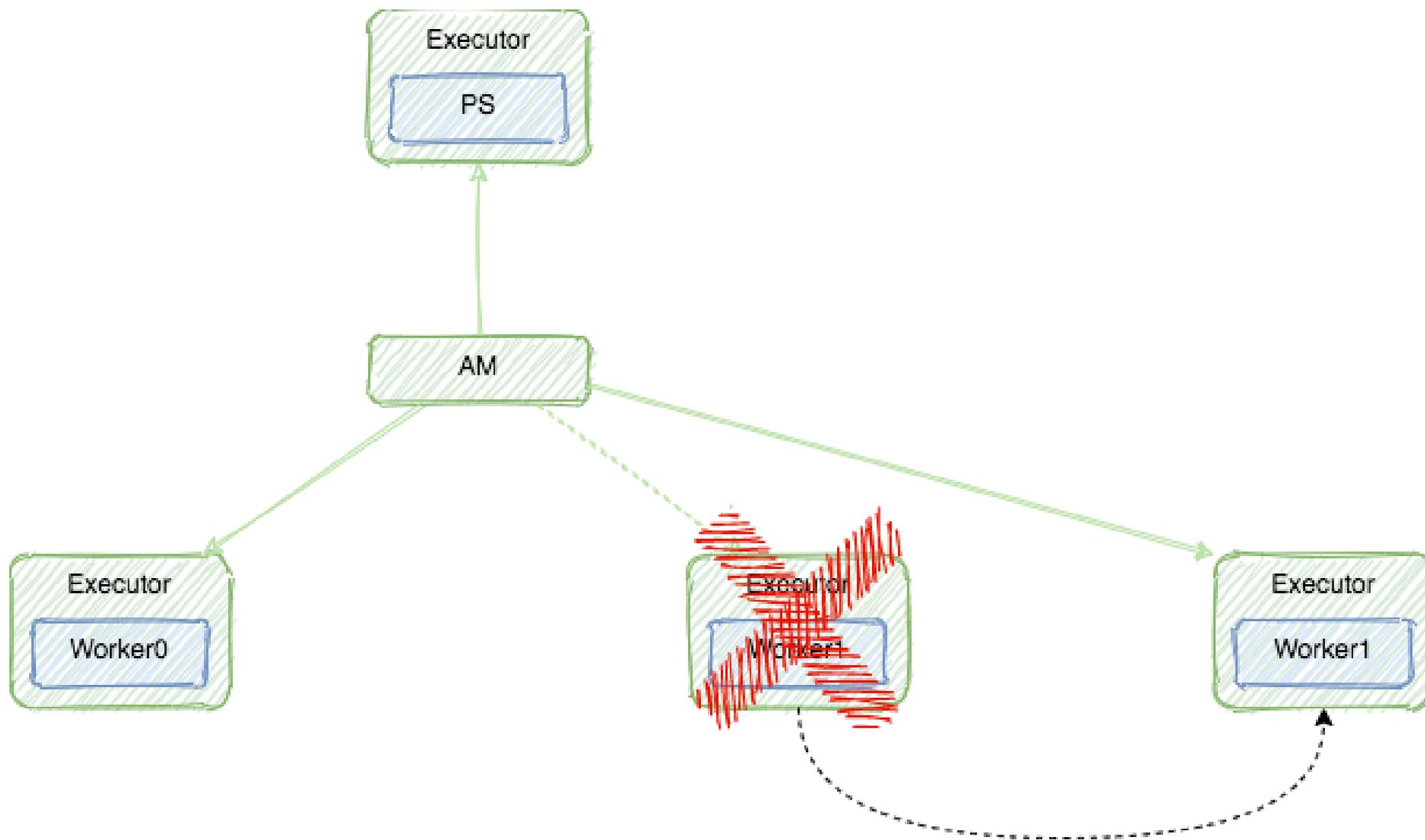
- Never
- **Always**
- OnFailure



Primus 编排调度

Failover策略

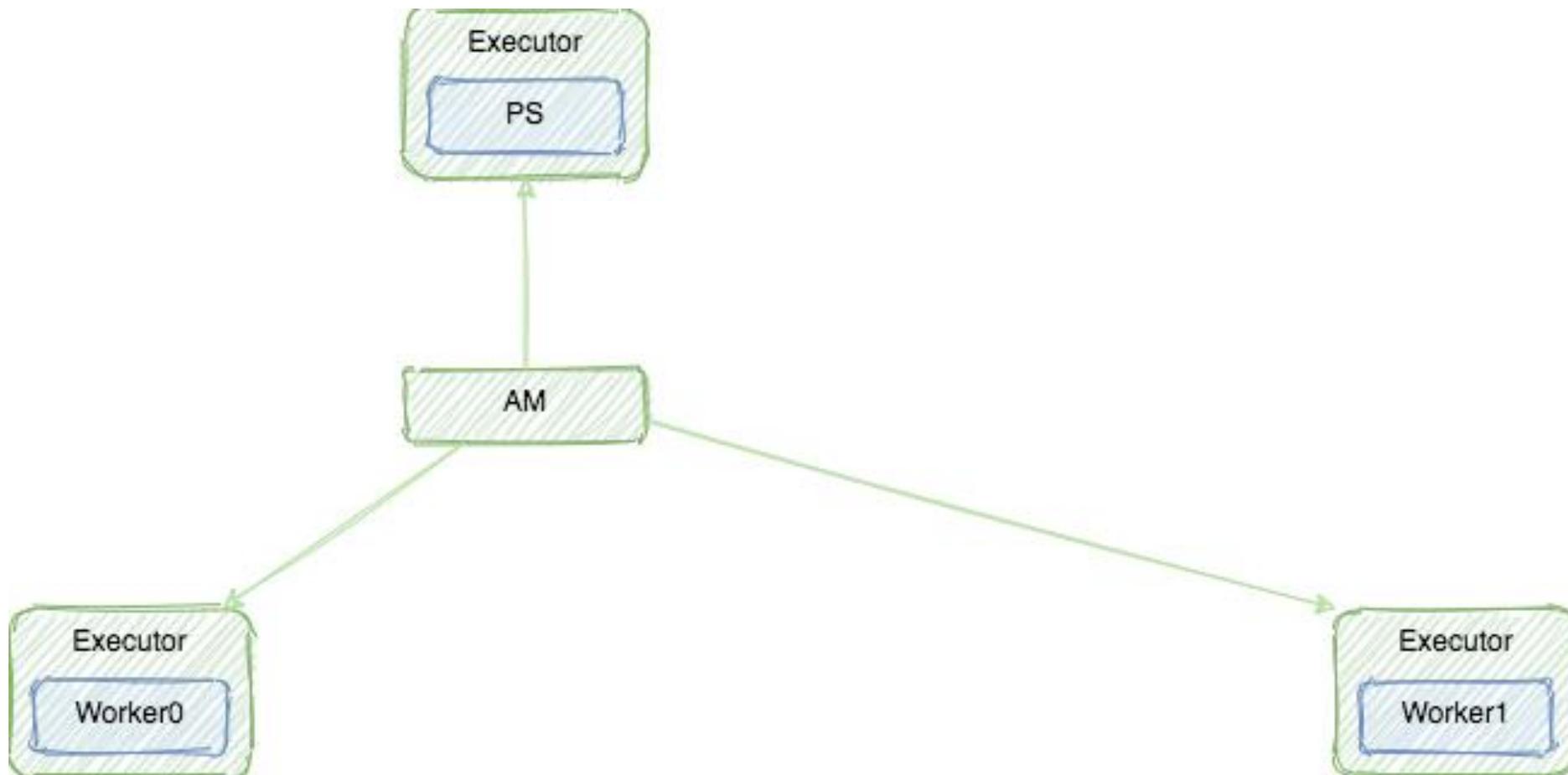
- Never
- **Always**
- OnFailure



Primus 编排调度

Failover策略

- Never
- **Always**
- OnFailure



Primus 常态混部支持

- 特点
 - 杀死率高
 - 不可预知
 - 明显潮汐特征
- 解决
 - 只调度支持 failover 角色, 优化 failover 逻辑
 - 细粒度 checkpoint
 - 低 worker 运行 & 跨 app 状态保存与恢复

Primus 错峰混部支持

- 特点
 - 可预知
 - 资源稳定, 但不连续
- 解决
 - 提前 checkpoint
 - 跨 app 状态保存与恢复

THANKS!