

# 深度学习 + 大数据 TensorFlow on Yarn

李远策  
2017年4月17日



- TensorFlow使用现状及痛点
- TensorFlow on Yarn设计
- TensorFlow on Yarn技术细节揭秘
- 深度学习平台演进及SparkFlow介绍

坐标：360-系统部-大数据团队

专业：Yarn、Spark、MR、HDFS ...

挑战：深度学习空前火爆，各种深度学习框架层出不穷，业务部门  
拥抱新兴技术。平台怎么应对？

机遇：Maybe 深度学习 + 大数据



Caffe



PYTORCH

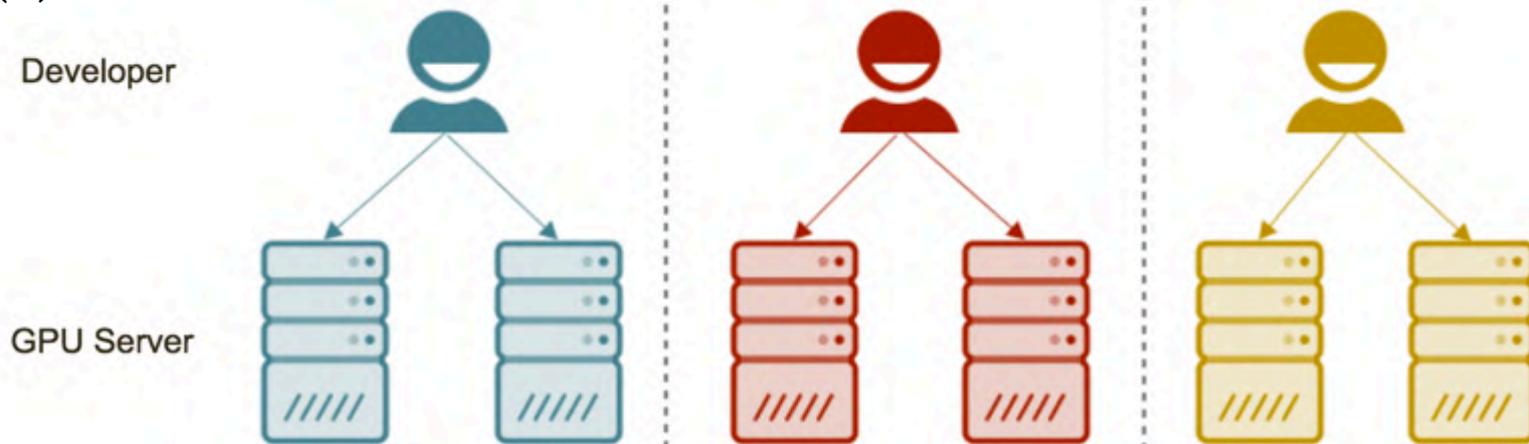
+

Spark

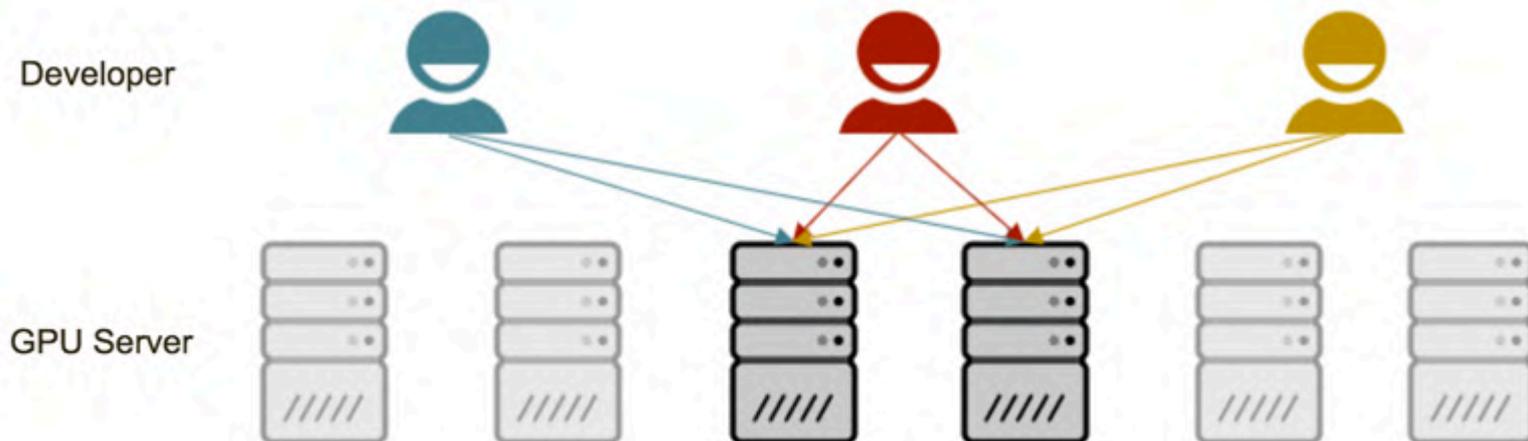


# TensorFlow使用现状及痛点

## 场景 (1)



## 场景 (2)



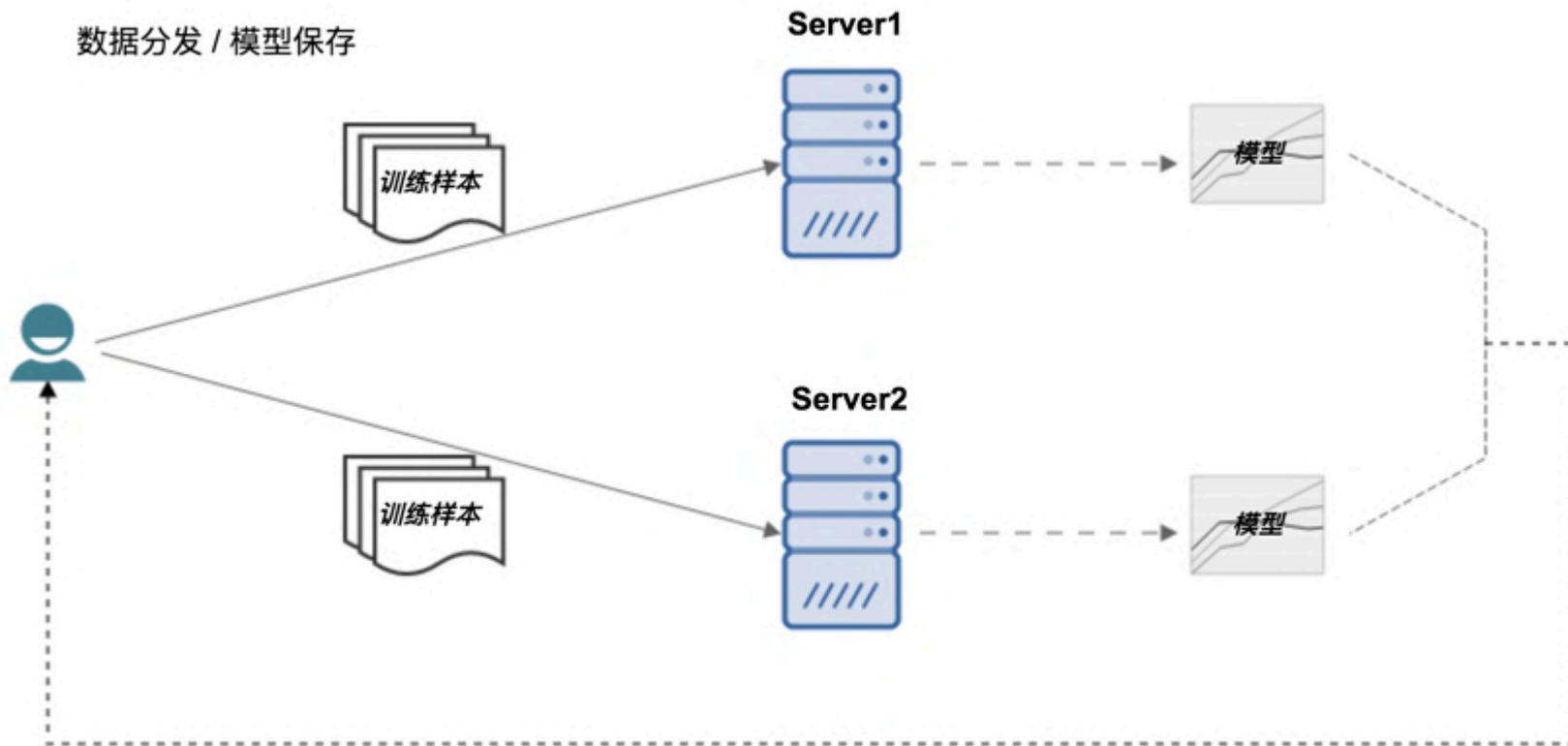
分布式版本ClusterSpec定义:

```
tf.train.ClusterSpec({  
    "worker": [  
        "worker0.example.com:2222",  
        "worker1.example.com:2222",  
        "worker2.example.com:2222"  
    ],  
    "ps": [  
        "ps0.example.com:2222",  
        "ps1.example.com:2222"  
    ]  
})
```

带来的问题:

- 手动指定机器很繁琐
- 端口冲突
- 机器负载不均

# TensorFlow使用现状及痛点



- 手动分发训练样本
- 手动拉取训练模型

## 总结:

- 多人多服务器使用混乱，计算资源如何划分?
- 没有GPU集群资源管理和调度（内存、CPU、GPU、端口），集群资源负载不均
- 训练数据手动分发，训练模型手动保存
- 进程遗留问题，需要手动杀死
- 缺乏作业统一管理，不便对作业运行状态跟踪
- 日志查看不方便

Yarn能解决什么问题：

- 集群资源的管理（目前支持CPU、内存，需要扩展GPU资源管理）
- 作业的统一管理、状态跟踪
- 资源组（Schedule Pool）的划分
- 作业进程的资源隔离

## 基本目标:

- 同时支持单机和分布式TensorFlow程序
- 支持GPU资源管理和调度
- 不再需要手动配置CluserSpec信息，仅需要设置work和ps的数量
- 训练数据和训练模型基于HDFS统一存储
- 作业训练结束自动回收work、ps和Tensorboard进程
- 训练效果和性能没有损失

## 扩展目标:

- 支持GPU亲和性调度（提高通信效率）
- Web的方式查看作业的运行状况和作业日志
- 在线查看Tensorboard
- HistoryServer支持查看结束作业的日志和状态信息
- 控制已有的TensorFlow作业的迁移成本（最多改三行代码）

## 提交脚本示例（分布式版本）：

```
tensorflow-submit \  
  --app-name "tfdemo" \  
  --files tfTestDemo.py,dataDeal.py \  
  --tfcmd "python tfTestDemo.py --training_epochs=20" \  
  --input /home/xitong/tf-test/data \  
  --output /home/xitong/tf-test/outputTest \  
  --worker-num 3 \  
  --worker-memory 8192M \  
  --worker-cores 1 \  
  --worker-gpus 2 \  
  --ps-num 2 \  
  --ps-memory 1024M \  
  --ps-cores 1 \  
  --priority VERY_LOW \  
  --board-enable true \  
  --conf tf.file.download.thread.num=10
```

*注释：*

- `#作业名`
- `#依赖的本地文件`
- `#TF运行指令`
- `#训练样本HDFS路径`
- `#保存模型的HDFS路径`
- `#work数量`
- `#每个worker需要的内存`
- `#每个worker需要的CPU核数`
- `#每个worker需要的GPU卡数`
- `#ps数量`
- `#每个ps需要的内存`
- `#每个ps需要的CPU核数`
- `#作业优先级`
- `#是否开启Tensorboard服务`
- `#其他参数设置`

# TensorFlow on Yarn设计



Yarn首页作业信息:

集群GPU资源概况

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	GCores Used	GCores Total	GCores Reserved
31	0	6	25	20	75 GB	800 GB	0 B	21	480	0	20	80	0

User Metrics for dr.who

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used	Memory Pending	Memory Reserved	VCores Used
0	0	0	0	0	0	0	0 B	0 B	0 B	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation
Fair Scheduler	[MEMORY, CPU, GPU]	<memory:1024, vCores:1, gCores:0>

Show 20 entries

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated VCores	Allocated GCores
<a href="#">application_1491884189069_0032</a>	liyance	tensorflow-demo2	TensorFlow	root.default	NORMAL	Tue Apr 11 15:50:46 +0800 2017	N/A	RUNNING	UNDEFINED	6	6	9
<a href="#">application_1491884189069_0031</a>	liyance	tensorflow-demo	TensorFlow	root.default	VERY_LOW	Tue Apr 11 15:47:11 +0800	N/A	RUNNING	UNDEFINED	6	6	9

作业类型

作业分配到的GPU数量

# TensorFlow on Yarn设计



TensorFlow作业AM页面:

查看work、ps日志

Container所在的机器

分配到的GPU物理设备号



## TensorFlow Application application\_1491884189069\_0031

All Containers:

Container ID	Container Host	GPU Device ID	Container Role	Container Status
container_e05_1491884189069_0031_01_000004	hadoop-102-102-qihoo.net	0,1,2	worker	RUNNING
container_e05_1491884189069_0031_01_000005	hadoop-102-102-qihoo.net	0,1,2	worker	RUNNING
container_e05_1491884189069_0031_01_000006	hadoop-102-102-qihoo.net	0,1,2	worker	RUNNING
container_e05_1491884189069_0031_01_000002	hadoop-102-102-qihoo.net	-	ps	RUNNING
container_e05_1491884189069_0031_01_000003	hadoop-102-102-qihoo.net	-	ps	RUNNING

View TensorBoard:

Tensorboard Info
<a href="http://hadoop-102-102-qihoo.net:57614">http://hadoop-102-102-qihoo.net:57614</a>

tensorboard url链接

Container角色

Container当前状态

Save Model

saved the model completed!

Saved timeStamp	Saved path
2017-04-11 15:47:51	/home/liyuance/outputdemo/interResult/interResult_2017_04_11_15_47_51

训练中保存的中间模型

# TensorFlow on Yarn设计



TensorFlow作业Tensorboard页面:

The screenshot displays the TensorBoard interface for a TensorFlow training job. The browser address bar shows the URL `qiuhoo.net:57614`. The main navigation bar includes tabs for SCALARS, IMAGES, AUDIO, GRAPHS, DISTRIBUTIONS, HISTOGRAMS, and EMBEDDINGS. The left sidebar contains controls for Fit to screen, Download PNG, Run (1), Session runs (0), Upload (Choose File), Trace inputs (toggle), Color (Structure selected), and Graph (\* = expandable). The main area shows the 'Main Graph' with nodes for global\_step, weights, biases, input, softmax, cross\_entropy, save, and train. The graph illustrates the flow of data and operations during training, including initialization (init) and saving (save) operations. An 'Auxiliary Graph' on the right shows details for the 'init' operation, including global\_step, weights, and biases.

# TensorFlow on Yarn设计



TensorFlow作业history页面：

oap

Tensorflow Application application\_1491884189069\_0031

Logged in as: dr.who

All Containers:

Container ID	Container Host	GPU Device ID	Container Role	Container Status
container_e05_1491884189069_0031_01_000002	hadoop-102-102-qihoo.net	-	ps	SUCCEEDED
container_e05_1491884189069_0031_01_000003	hadoop-102-102-qihoo.net	-	ps	SUCCEEDED
container_e05_1491884189069_0031_01_000004	hadoop-102-102-qihoo.net	0,1,2	worker	SUCCEEDED
container_e05_1491884189069_0031_01_000005	hadoop-102-102-qihoo.net	0,1,2	worker	SUCCEEDED
container_e05_1491884189069_0031_01_000006	hadoop-102-102-qihoo.net	0,1,2	worker	SUCCEEDED

查看历史日志

View TensorBoard:

Tensorboard Info	
tensorboard	--logdir=hdfs://hadoop-102-102-qihoo.net:9000/home/yarn/tensorflow/eventLog/application_1491884189069_0031

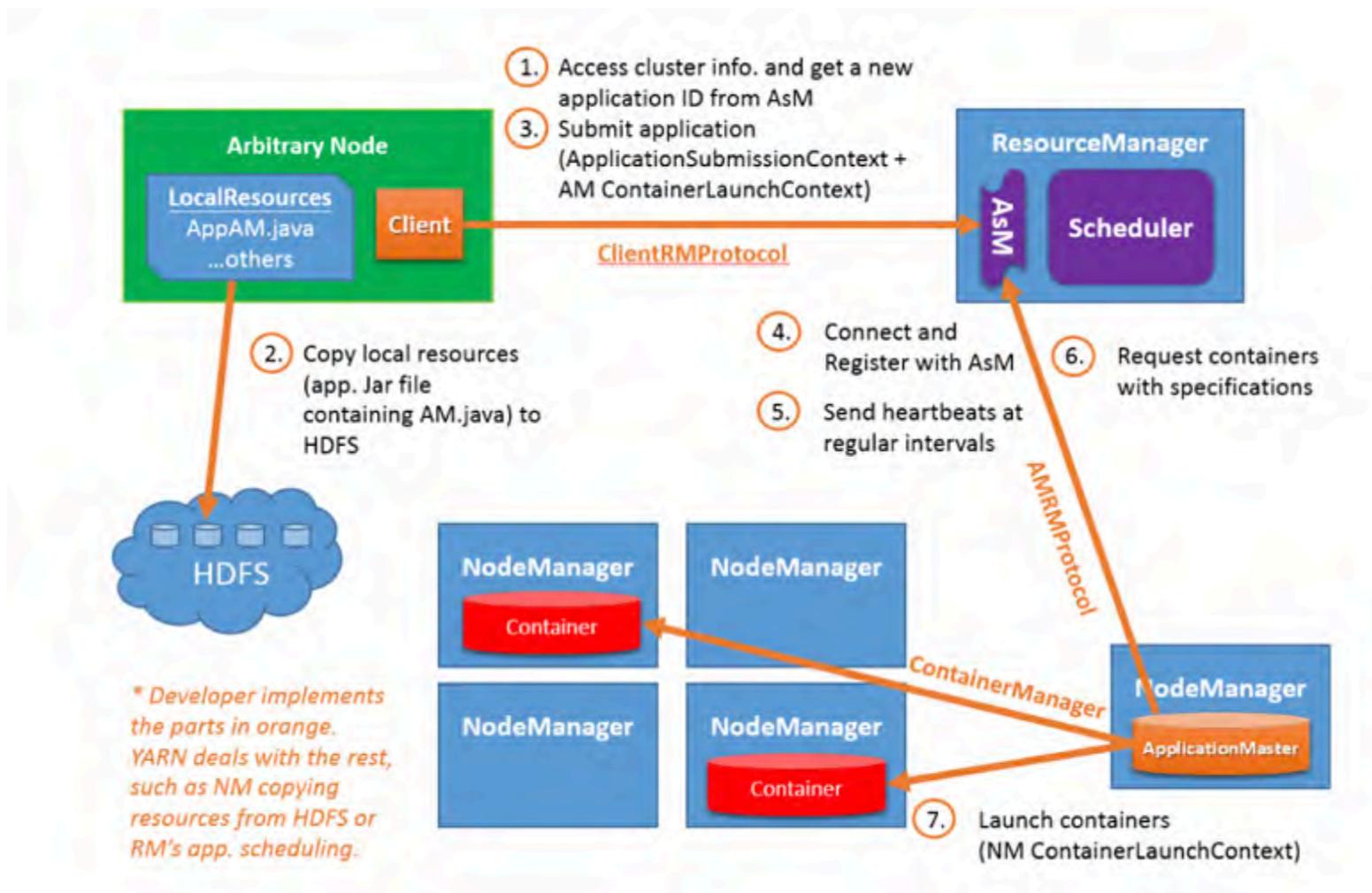
Saved Model

Saved timeStamp	Saved path
2017-04-11 15:47:51	/home/liyuance/outputdemo/interResult/interResult_2017_04_11_15_47_51

Event log上传到了HDFS

# TensorFlow on Yarn技术细节揭秘

实现Yarn Application的标准流程:



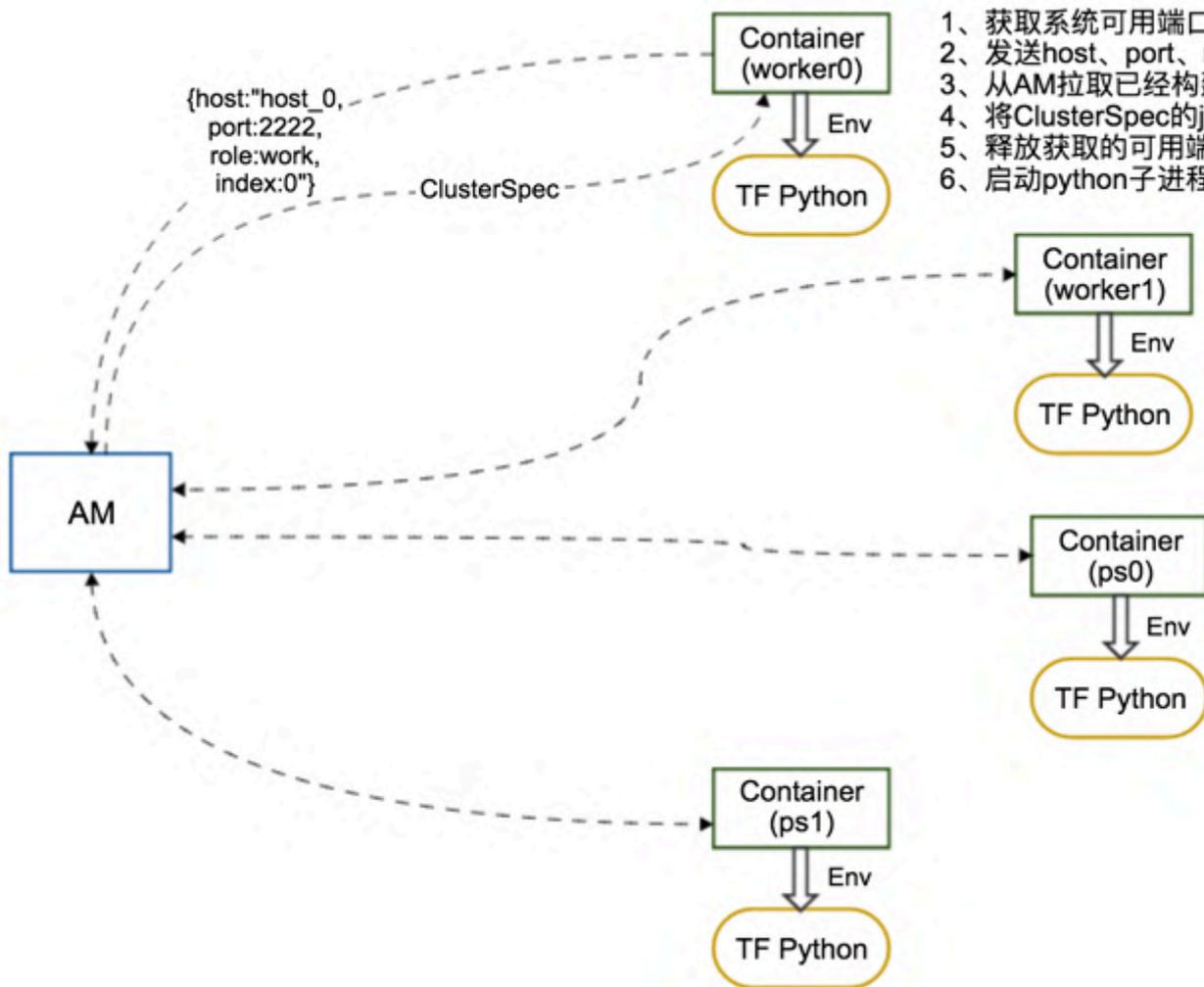
集成TensorFlow到Yarn面临的特定问题：

- 如何自组织ClusterSpec信息
- 训练数据的划分
- 如何启动Tensorboard服务
- 如何降低迁移成本
- 已分配的物理GPU设备号到用户态GPU设备号的映射

# TensorFlow on Yarn技术细节揭秘



自动构建ClusterSpec的流程图：



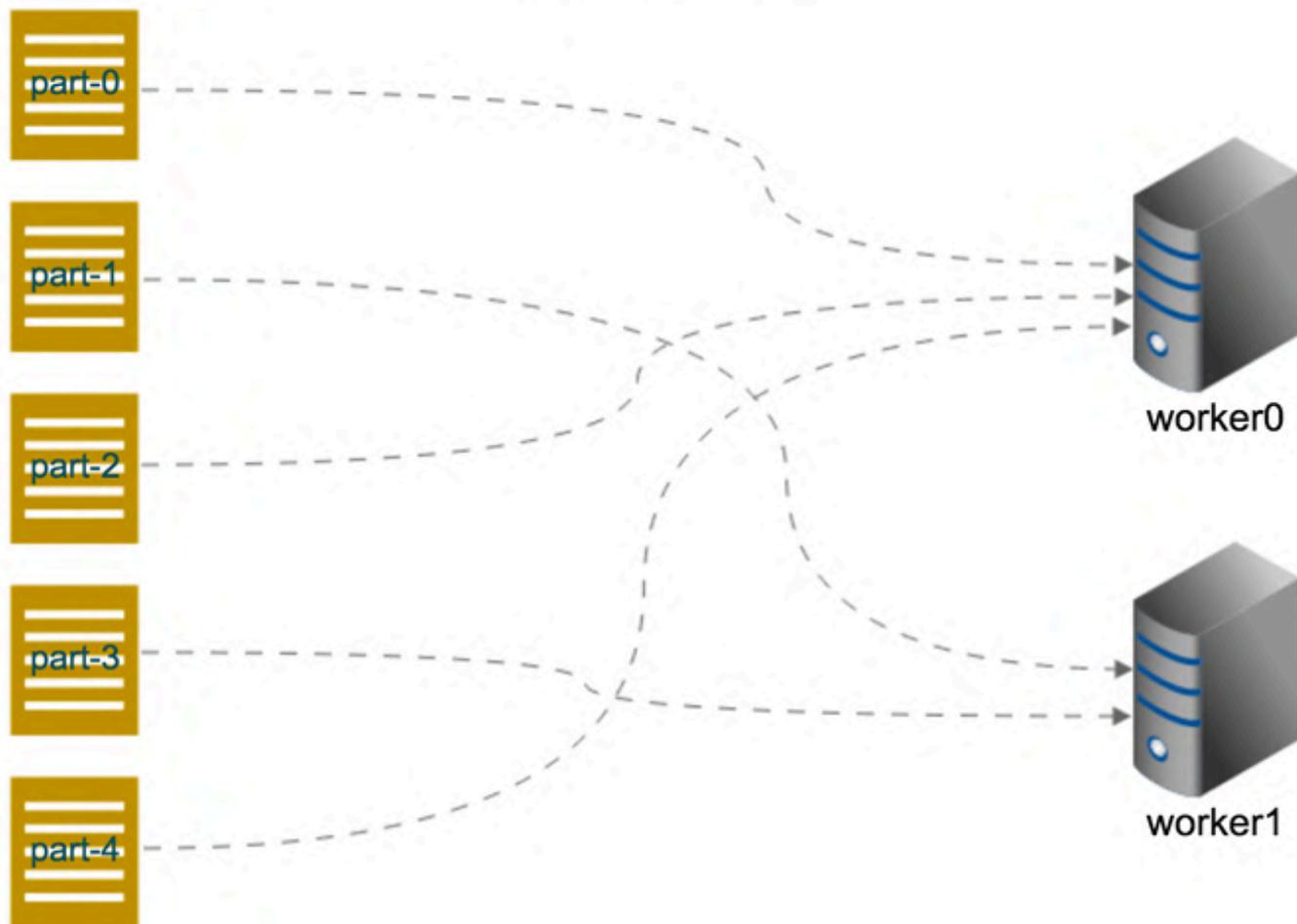
- 1、获取系统可用端口
- 2、发送host、port、role、index给AM
- 3、从AM拉取已经构建好的ClusterSpec信息
- 4、将ClusterSpec的json字符串作为环境变量传给TF的python子进程
- 5、释放获取的可用端口
- 6、启动python子进程

# TensorFlow on Yarn技术细节揭秘

训练数据的划分:

hdfs:///path/to/input/

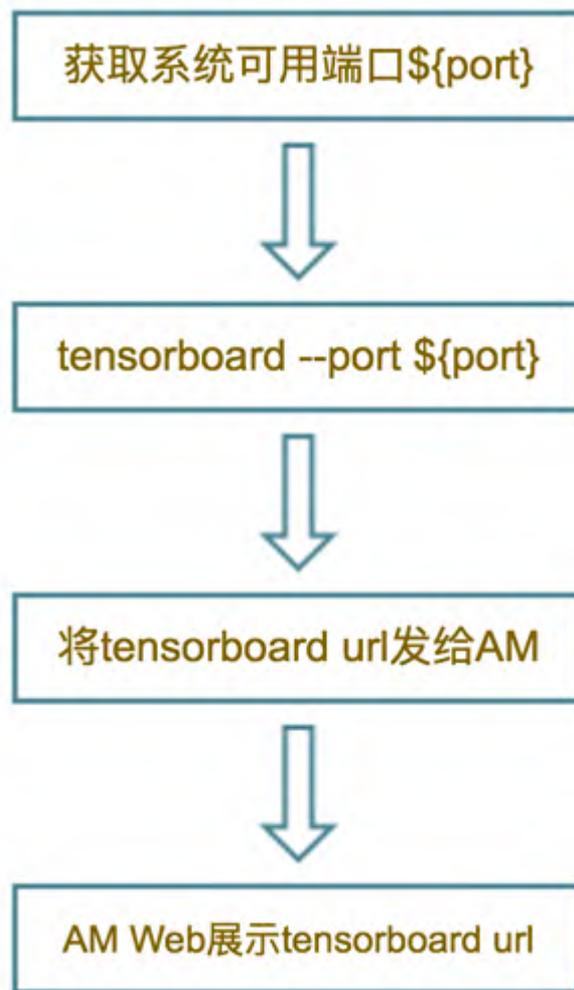
按文件轮流分配



# TensorFlow on Yarn技术细节揭秘



启动Tensorboard服务:



降低已有tensorflow程序迁移成本：

(1) 单机模式

*不需要修改代码*

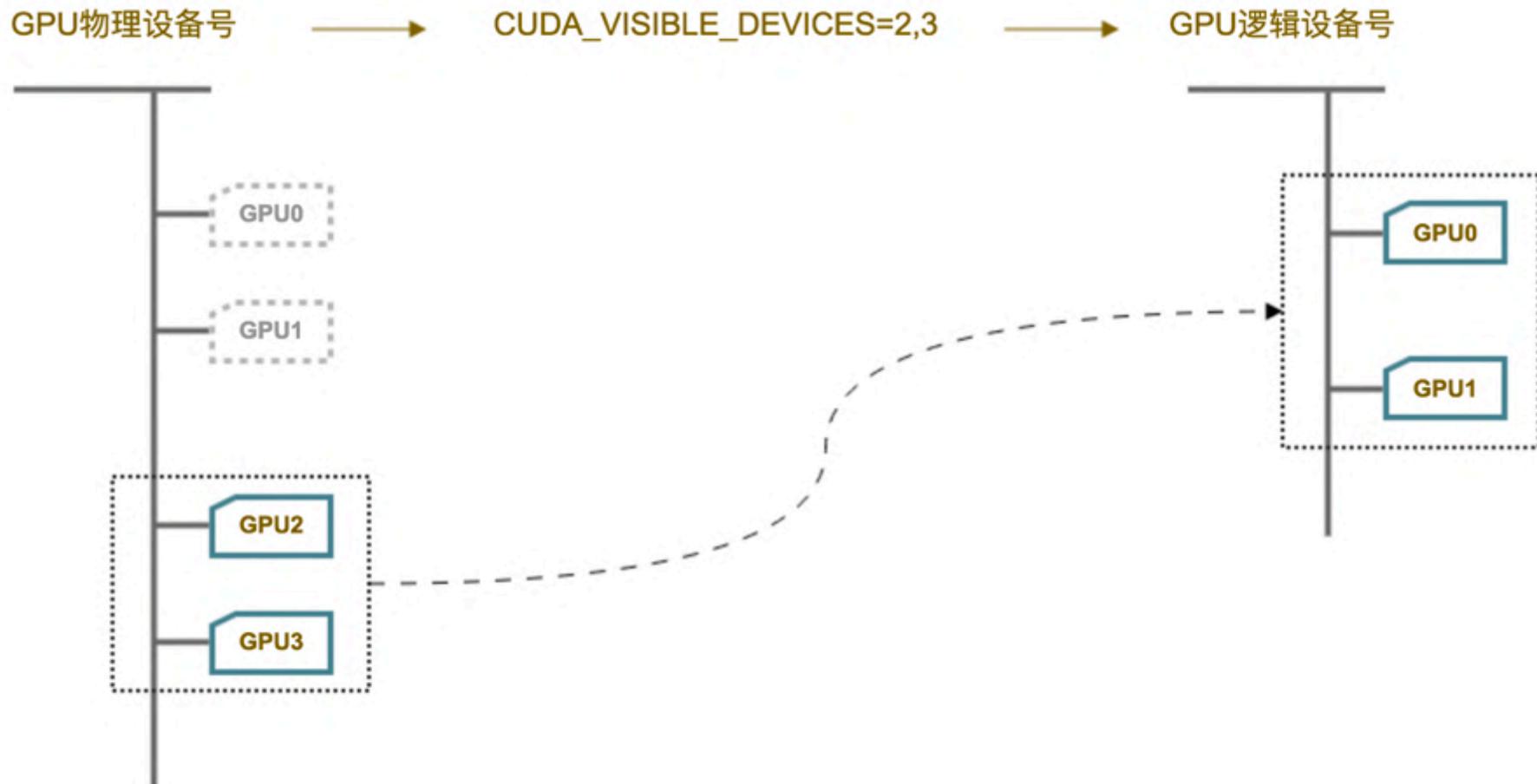
(2) 分布式模式 (*最多修改三行代码*)

```
cluster = tf.train.ClusterSpec(json.loads(os.environ["TF_CLUSTER_DEF"]))  
job_name = os.environ["TF_ROLE"]  
task_index = int(os.environ["TF_INDEX"])
```

# TensorFlow on Yarn技术细节揭秘

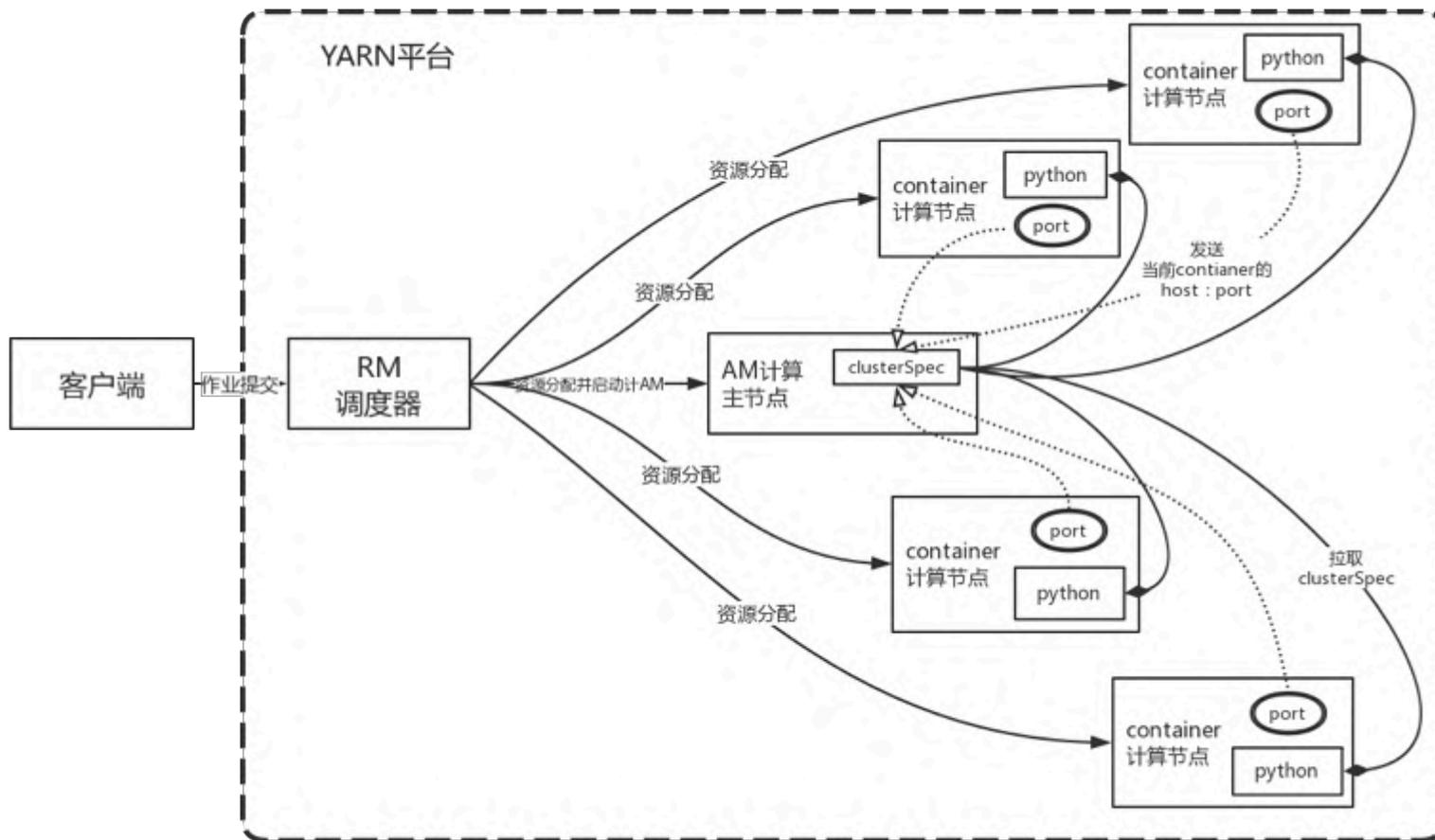


已分配的物理GPU设备号到用户态GPU设备号的映射：



# TensorFlow on Yarn技术细节揭秘

TensorFlow on Yarn系统架构图：



## Yarn支持CPU调度 vs GPU调度:

CPU	GPU
每个NodeManager配置可用CPU核心数量	每个NodeManager配置可用GPU卡数量
ResourceManager统计计数并按数量分配	ResourceManager统计计数并按数量分配
作业必须占用CPU资源	作业可以不需要GPU资源
系统自动分配物理CPU核心	需要知道具体GPU卡号, 代码分配计算任务到指定GPU设备
设备亲和性影响较小	设备亲和性影响较大

## Yarn支持GPU调度ResourceManager端实现:

扩展org.apache.hadoop.yarn.api.records.Resource抽象类及其实现, 增加:

```
public abstract int getGpuCores();
```

```
public abstract void setGpuCores(int gCores);
```

最终在ResourceManager端需要完成:

- 1、对NodeManager GPU卡数量的统计管理
- 2、调度器统计管理每个Pool的GPU设备数的分配情况

具体可以参考下面Patch的实现思路:

<https://issues.apache.org/jira/browse/YARN-5517>

## Yarn支持GPU调度NodeManager端实现:

NodeManager yarn-site.xml中添加配置:

```
<!-- K40 -->
<property>
  <name>yarn.nodemanager.resource.gpu-cores</name>
  <value>((2,2))</value>
</property>
```

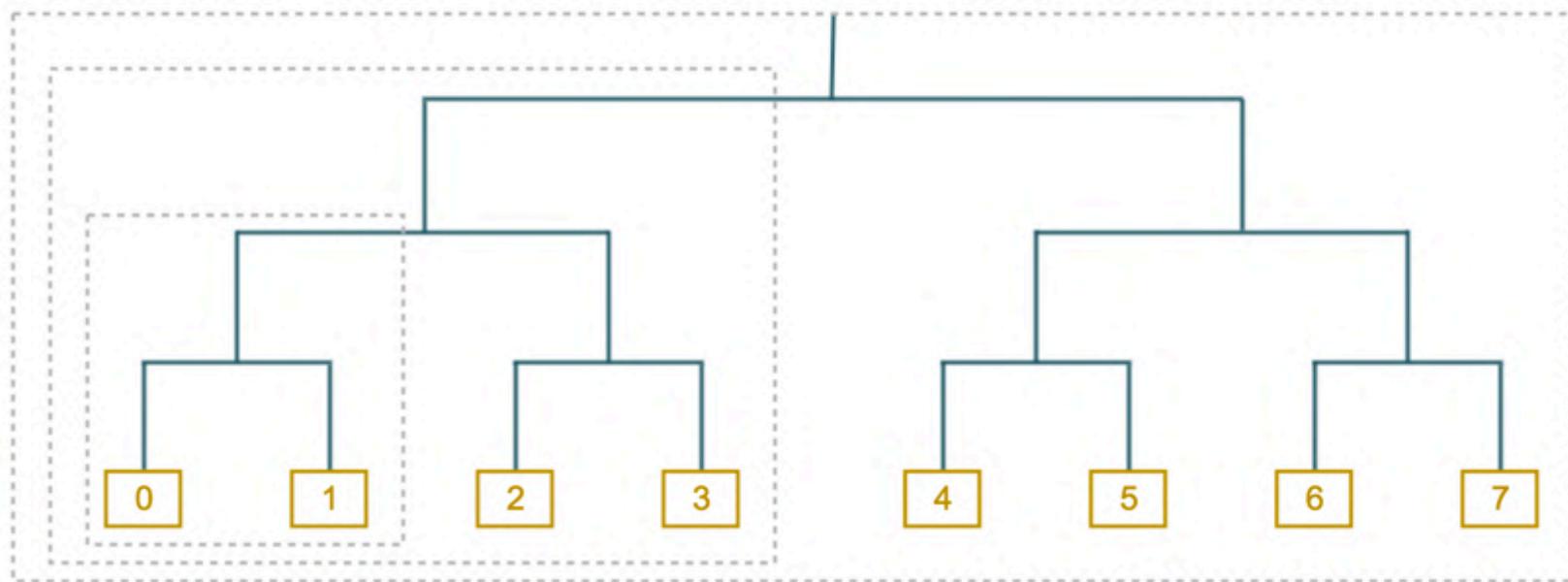
NodeManager上可用的GPU卡数是:  $2 + 2 = 4$

```
<!-- K80 -->
<property>
  <name>yarn.nodemanager.resource.gpu-cores</name>
  <value>((2,2), (2,2))</value>
</property>
```

NodeManager上可用的GPU卡数是:  $2 + 2 + 2 + 2 = 8$

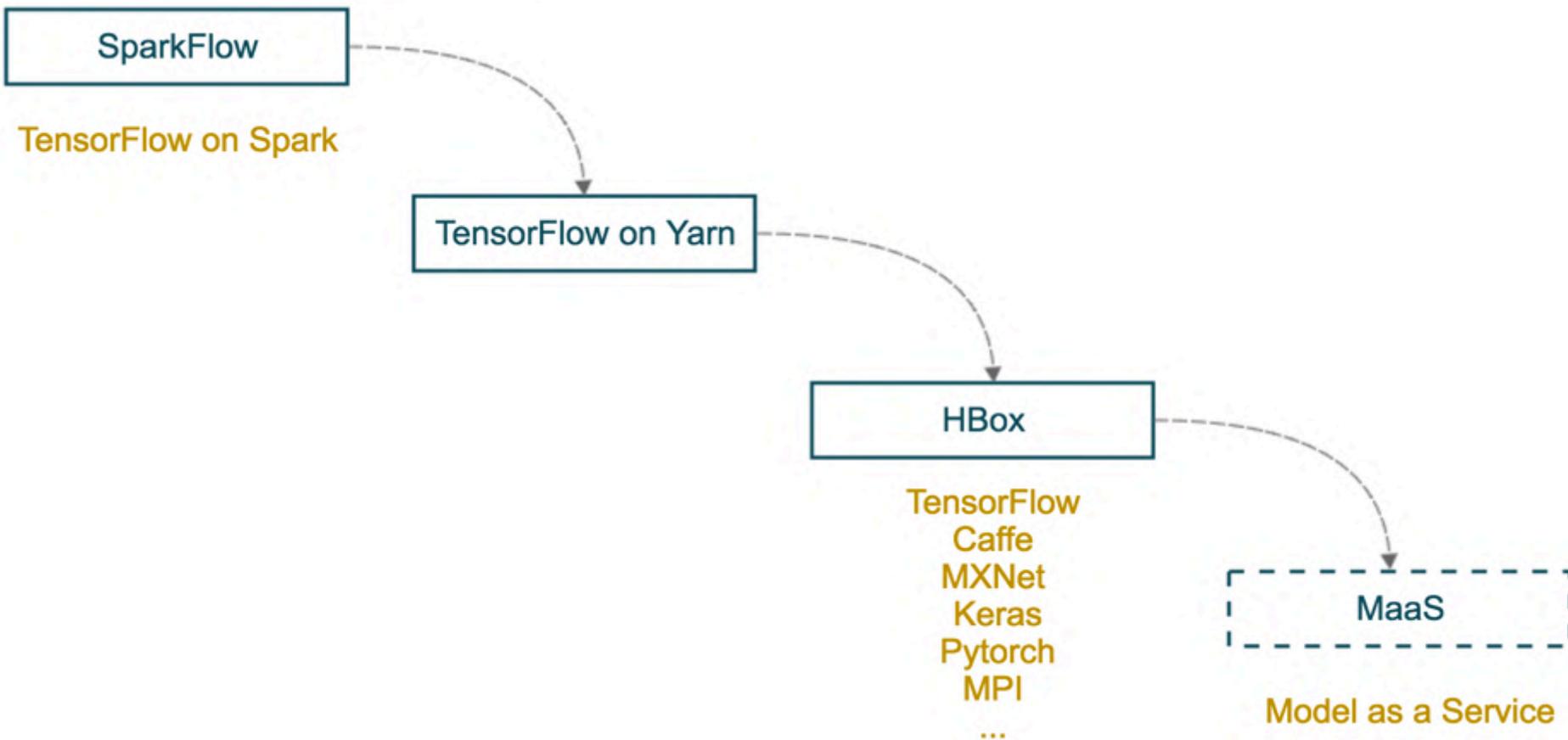
NodeManager端GPU亲和性调度:

$((2, 2), (2, 2))$

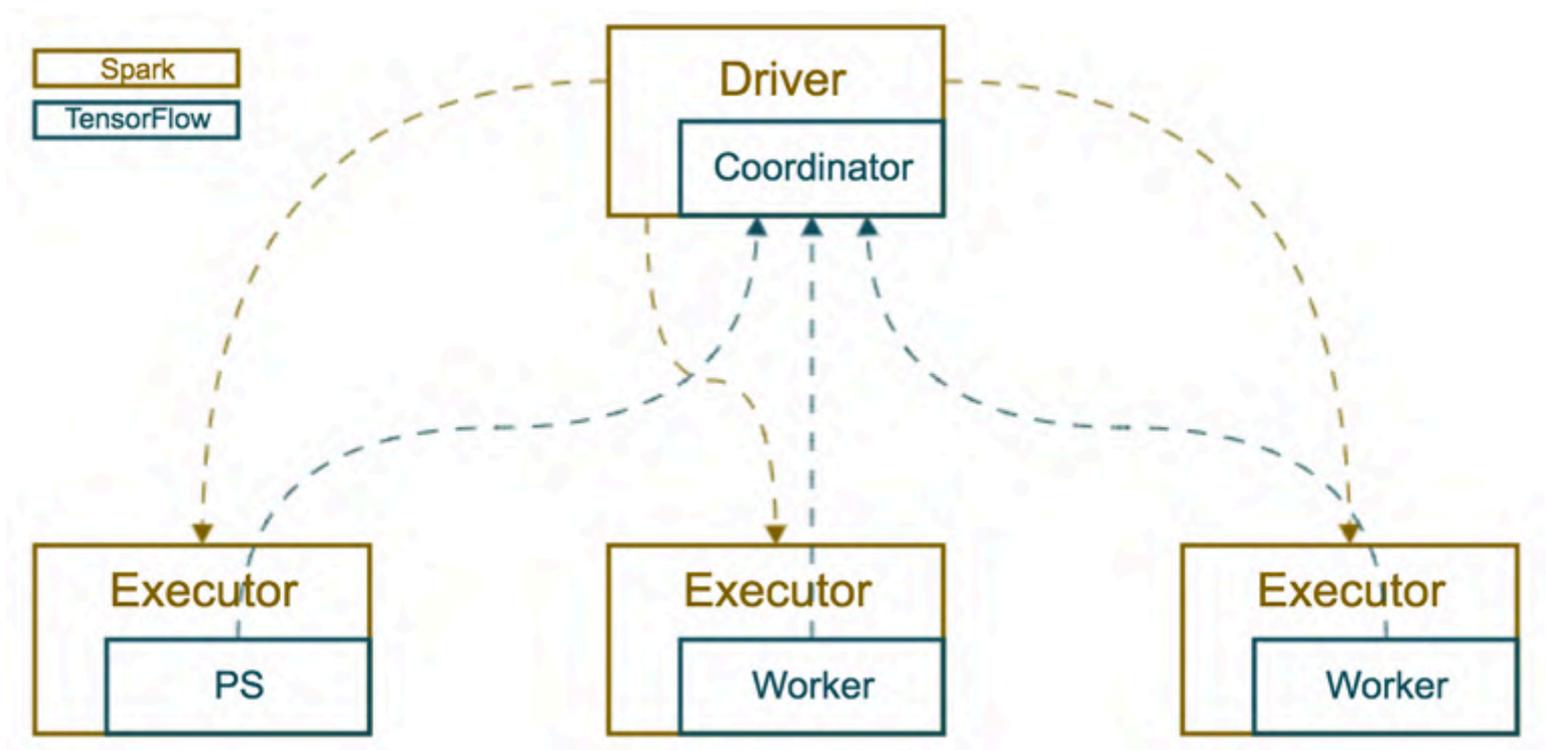


深度优先

后序、递归、深度优先分配算法



SparkFlow: 360系统部大数据团队设计的TensorFlow on Spark解决方案



- Coordinator负责协调生成ClusterSpec（扩展的TensorFlow gRPC server）
- Worker通过读取RDD获取训练样本
- RDD的数据cache到内存或者磁盘供多次迭代训练使用

## SparkFlow与TensorFlow on Yarn对比:

SparkFlow	TensorFlow on Yarn
通过RDD读取训练样本数据，关心文件存储格式	直接读取HDFS数据，不关心文件存储格式
Worker和PS的资源同构	Worker和PS可以各自配置资源
不支持GPU调度	支持GPU调度
迁移成本较高	迁移成本低
嵌入到Spark计算框架里，方便打通数据流	实现了一种新的Yarn Application，可以与TensorFlow灵活整合和功能定制
代码量几百行	代码量几千行

<card>

<name> *Yuance.Li* </name>

<phone> *15201453364* </phone>

<email> *liyuanca@{360.cn, gmail.com}* </email>

</card>

# 谢谢！



360 互联网安全中心

安全第一 就用360