|  |
| --- |
| OpenPAI搭建指南 |
| 东南大学数据科学实验室 |
| 时间：2019年12月10日 |
| 参与人员：唐元博、杨煜涵、陈靖仪 |
|  |

[1 项目简介 1](#_Toc13397)

[2 具体搭建步骤 2](#_Toc5555)

[索引 2](#_Toc4121)

[2.1 准备部署环境 2](#_Toc1692)

[2.2 修改配置文件 3](#_Toc28261)

[2.3 启动Kubenete集群服务 4](#_Toc8127)

[2.4 更新 Kubernetes 的集群配置 5](#_Toc12669)

[2.5 启动所有 OpenPAI 服务 5](#_Toc11028)

[3 计算平台验证 6](#_Toc4427)

[3.1 Hadoop服务： 6](#_Toc1889)

# **项目简介**

* OpenPAI架构与功能简介

OpenPAI是由微软亚洲研究院和微软（亚洲）互联网工程院联合研发的，支持多种深度学习、机器学习及大数据任务，可提供大规模GPU集群调度、集群监控、任务监控、分布式存储等功能，且用户界面友好，易于操作。

OpenPAI的架构如下图所示，用户通过Web Portal调用REST Server的API提交作业（Job）和监控集群，其他第三方工具也可通过该API进行任务管理。随后REST Server与Launcher交互，以执行各种作业，再由Launcher Server处理作业请求并将其提交至HadoopYARN 进行资源分配与调度。可以看到，OpenPAI给YARN添加了GPU支持，使其能将GPU作为可计算资源调度，助力深度学习。其中，YARN负责作业的管理，其他静态资源（下图蓝色方框所示）则由Kubernetes进行管理。

图 1 OpenPAI架构

# 具体搭建步骤

## 索引

[第一步: 准备部署环境](https://github.com/microsoft/pai/blob/master/docs/zh_CN/pai-management/doc/distributed-deploy.md" \l "c-step-1)

[第二步: 准备配置](https://github.com/microsoft/pai/blob/master/docs/zh_CN/pai-management/doc/distributed-deploy.md" \l "c-step-2)文件

[第三步: 部署Kubernetes](https://github.com/microsoft/pai/blob/master/docs/zh_CN/pai-management/doc/distributed-deploy.md" \l "c-step-3)

[第四步: 更新 Kubernetes 的集群配置](https://github.com/microsoft/pai/blob/master/docs/zh_CN/pai-management/doc/distributed-deploy.md" \l "c-step-4)

[第五步: 启动所有 OpenPAI 服务](https://github.com/microsoft/pai/blob/master/docs/zh_CN/pai-management/doc/distributed-deploy.md" \l "c-step-5)

## [准备部署环境](https://github.com/microsoft/pai/blob/master/docs/zh_CN/pai-management/doc/distributed-deploy.md" \l "c-step-1)

本地环境：

* 操作系统：ubuntu18.04
* Nvidia-driver：418.56
* Docker-version：Docker Engine - Community19.03.5

OpenPAI对于集群的配置主要是通过一个容器完成的，在配置集群中所需要用到的所有代码文件、代码运行的依赖环境、配置文件都被封装在这个容器中，做到与集群配置分离。首先，从docker-hub上拉取OpenPAI的官方配置镜像并运行：

sudo docker pull docker.io/openpai/dev-box:v0.14.0

sudo docker run -itd \

-e COLUMNS=$COLUMNS -e LINES=$LINES -e TERM=$TERM \

-v /var/run/docker.sock:/var/run/docker.sock \

-v /pathConfiguration:/cluster-configuration \

-v /hadoop-binary:/hadoop-binary \

--pid=host \

--privileged=true \

--net=host \

--name=dev-box \

docker.io/openpai/dev-box:v0.14.0

# Working in your dev-box

sudo docker exec -it dev-box /bin/bash

cd /pai

至此，我们已经进入dev-box这个容器内部，接下来所有的操作都会在其中进行。

## 修改配置文件

集群的信息在配置文件中进行修改，首先，在/pai/development/quick-start文件夹中，我们 使用vim命令修改quick-start.yaml文件。

# quick-start.yaml

# (Required) Please fill in the IP address of the server you would like to deploy OpenPAI

machines:

- 192.168.1.11

- 192.168.1.12

- 192.168.1.13

# (Required) Log-in info of all machines. System administrator should guarantee

# that the username/password pair or username/key-filename is valid and has sudo privilege.

ssh-username: pai

ssh-password: pai-password

# (Optional, default=None) the key file that ssh client uses, that has higher priority then password.

#ssh-keyfile-path: <keyfile-path>

# (Optional, default=22) Port number of ssh service on each machine.

#ssh-port: 22

深色区域指示了需要修改的位置，分别代表着在局域网内节点的ip地址，以及ssh连 接的账户密码。通过这些数据，配置程序可以知晓集群节点的位置并且获取访问权限。 下一步我们用这个文件生成其余配置文件。

python paictl.py config generate -i /pai/deployment/quick-start/quick-start.yaml -o ~/pai-config -f

在/pai目录下我们运行以上代码，则会在~/pai-config目录中生成四个配置文件，他们 分别代表着几个方面的集群信息。首先我们修改layout.yaml文件，这个文件用以设置 集群节点的cpu核数、gpu数量以及型号、节点内存等重要信息。

machine-sku:

k80-node:

mem: 40G

gpu:

type: Tesla K80

count: 4

cpu:

vcore: 24

os: ubuntu16.04

machine-list:

- hostname: xxx

hostip: yyy

machine-type: k80-node

- hostname: xxx

hostip: yyy

machine-type: p100-node

上边的文件列举了一些重要的信息以及修改的示例，至于其他信息的修改则相对没有那么重要，OpenPAI可供修改的集群参数包括集群内DNS、暴露端口号、显卡驱动版本等。

## 启动Kubenete集群服务

使用容器dev-box内的程序部署kubenete集群是一件很方便的事情，只需在终端运行如下代码：

cd pai

python paictl.py cluster k8s-bootup -p ~/pai-config

这段代码从/pai-config路径中读取配置文件（也就是我们在上一个步骤中修改的文件）进行kubenete集群的配置，关于kubenete集群的基础知识和使用方式详情可参见[[1]](#footnote-0)

至此kubenete的服务都已经成功启动，为了验证我们步骤的正确性，我们从web界面中访问http://<master-ip>:8080和http://<master-ip>:9090来分别观察集群的信息：



图 2 kubenete的webUI

从webUI中我们可以只管的看到kubenete集群内所有运行任务的直观情况，亦可以从此站点跳转到容器日志，还可以直接提交用户代码，具体使用方式请参见OpenPAI使用手册文件。

## 更新 Kubernetes 的集群配置

使用python命令将集群配置文件更新，这里主要针对的的是OpenPAI中的service.yaml文件中的配置信息。当对其中的信息进行改动之后，运行如下命令，将会使得改动生效。

python paictl.py config push -p /path/to/config/dir [-c ~/.kube/config]

## 启动所有 OpenPAI 服务

使用如下命令开启OpenPAI服务，需要注意的是由于我们已经预先装好了显卡驱动，所以我们不需要开启drivers服务，可以跳过它。

cd pai

# cmd should be executed under /pai directory in the environment.

python paictl.py service start [ -c ~/.kube/config] [ -n service-list ]

至此，OpenPAI的服务也完全启动。

# 计算平台验证

OpenPAI服务包含以下几大组件：

* Kubenete服务，负责所有容器、服务的调度，具有自恢复特性，其webUI界面已经在上文中有所展示故不赘述。
* Hadoop服务，其中包含yarn资源调度以及HDFS分布式文件系统。
* OpenPAI任务调度组件，它依赖以上两种服务，并提供了一套提交用户任务的接口，包括webUI界面、VS接口等。

## Hadoop服务：

通过http://<master-ip>:50070端口我们可以访问分布式文件系统的webUI界面，为了保证集群的数据安全性，我们设置了用户权限和登录账户密码系统。

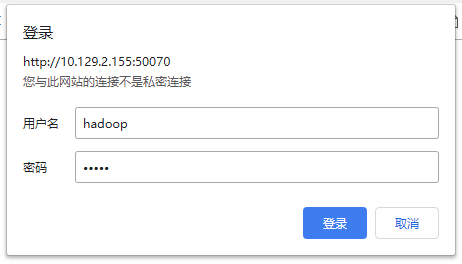


图 3-1 HDFS用户登录界面

进入系统之后我们可以从webUI界面中看到集群的基本信息，包括但不限于分布式文件系统的节点总数、系统总容量等信息。

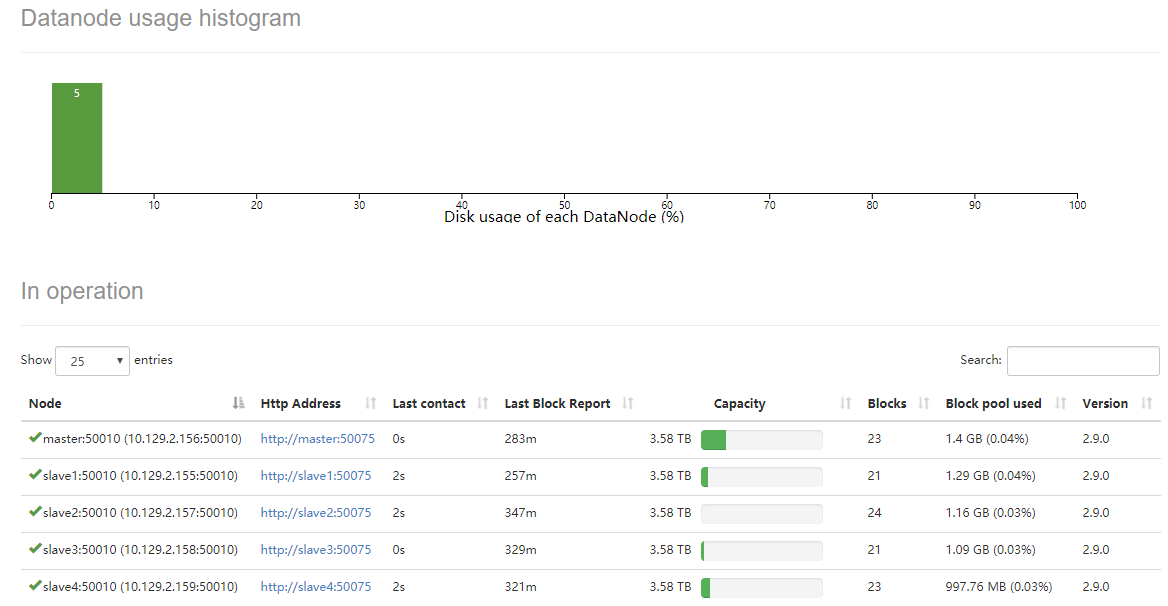


图 3-2 HDFS系统集群概况

这表明HDFS系统已经搭建成功。Apache Hadoop YARN （Yet Another Resource Negotiator，另一种资源协调者）是一种新的 Hadoop 资源管理器，它是一个通用资源管理系统，可为上层应用提供统一的资源管理和调度，它的引入为集群在利用率、资源统一管理和数据共享等方面带来了巨大好处。而在yarn资源调度界面中我们可以看到如下信息：

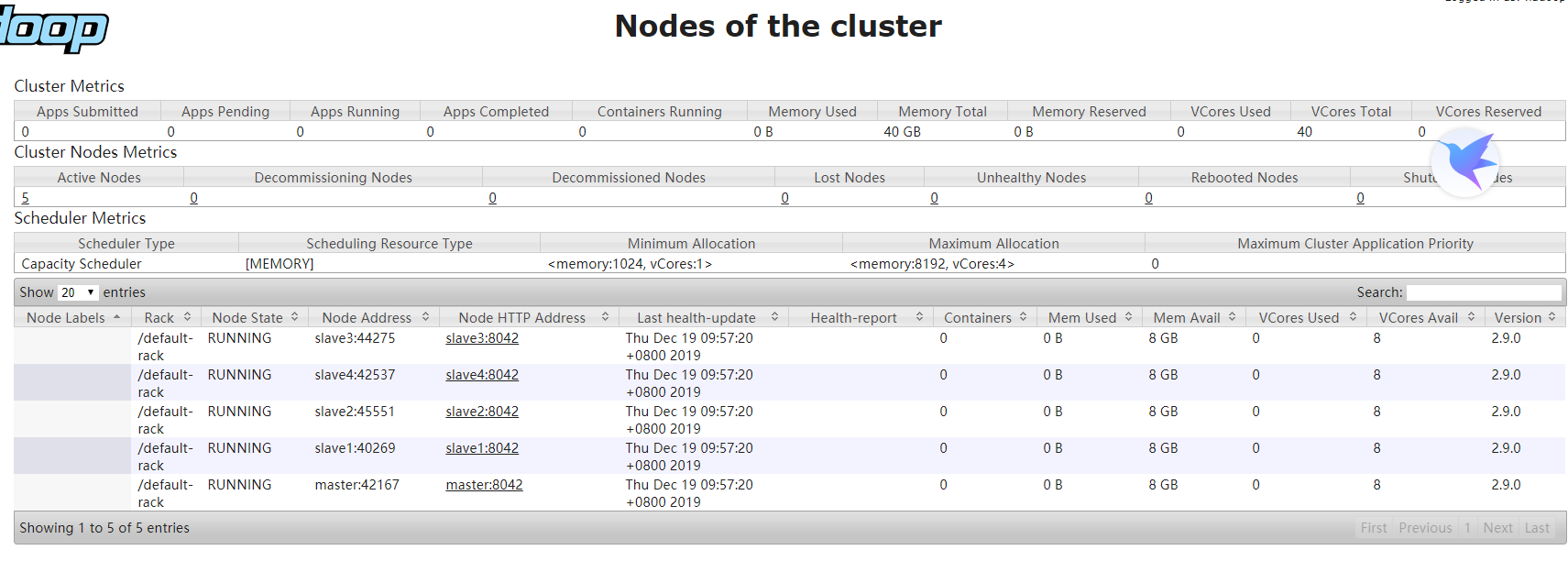


图 3-3 yarn资源调度界面

# 参考资料

这里我列举了一些几个平台服务的官方指导教程，在集群的搭建和使用中会遇到很多不明所以的问题，此时去官方网站上寻找答案和提问是最快最安全的途径，其次是Google，再其次是国内网站，因为每个人遇到的问题都有所不同，很多时候需要自己看log日志进行错误排查，保持耐心和信心，一定会成功的。

* OpenPAI的官方网站：<https://github.com/microsoft/pai>
* OpenPAI搭建指南<https://github.com/microsoft/pai/blob/master/docs/pai-management/doc/distributed-deploy.md>
* Kubenete网站：<https://kubernetes.io/>
* Kubenete基础概念与使用方法：<https://kubernetes.io/docs/tutorials/kubernetes-basics/>
* Hadoop官方网站：<https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html>

1. https://kubernetes.io/docs/tutorials/kubernetes-basics/ [↑](#footnote-ref-0)