第二届
低空经济智能飞行管理挑战赛 性能赛
规则说明

2024.8.19

1. 概述

本文档以初赛为例，介绍比赛规则、流程。复赛流程与初赛相似。

初赛（8 月 26 日-9 月 30 日）分为两个阶段：单机调试，在线竞赛。第一阶段供参赛选手学习比赛环境、调试算法；第二阶段正式计分。

2. 单机调试（8.26-9.30）

- 选手将单机仿真环境 docker 镜像、SDK 及相关工具下载到本地，具体步骤如下：

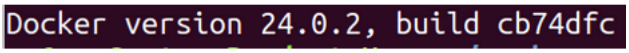
  1）配置本地 docker 环境

  a. 建议用官方脚本自动配置所需环境

  ```
  curl -fsSL https://test.docker.com -o test-docker.sh
  sudo sh test-docker.sh
  ```

  b. 验证是否成功

  ```
  docker --version  # 检查 Docker 版本
  ```

  2）下载 SDK 镜像

  a. 新建脚本 **race_images.sh**，添加以下内容到文件中

```
#!/bin/bash

# 要拉取的镜像列表
images=(
  "registryonline-hulk.sankuai.com/custom_prod/com.sankuai.udm.udss/race:race_user_sdk"
  "registryonline-hulk.sankuai.com/custom_prod/com.sankuai.udm.udss/race:race_drone_sdk"
  "registryonline-hulk.sankuai.com/custom_prod/com.sankuai.udm.udss/race:race_car_sdk"
  "registryonline-hulk.sankuai.com/custom_prod/com.sankuai.udm.udss/race:race_scene_sdk"
)
```

```
# 循环拉取每个镜像
for image in "${images[@]}"; do
    echo "Pulling $image..."
    docker pull "$image"
    if [ $? -ne 0 ]; then
        echo "Failed to pull $image"
        exit 1
    fi
done

echo "All images pulled successfully!"
```

b. 运行脚本

```
chmod +x race_images.sh
./race_images.sh
```

3）创建子网络

a. 创建子网络

```
docker network create --subnet=192.168.100.0/24
```

b. 查看子网络详细信息

```
docker network inspect race_net
```

4）启动 SDK

a. 新建启动脚本 start_race.sh，添加以下内容到文件中

```
#!/bin/bash

# Function to check the last command status and exit if it failed
check_status() {
    if [ $? -ne 0 ]; then
        echo "Error: $1 failed"
        exit 1
    fi
}

# Run the first container
docker run -d --entrypoint /manager/run.sh --name
```

```
race_scene_sdk_container registryonline-hulk.sankuai.com/custom_prod/com.sankuai.udm.udss/race:race_scene_sdk
check_status "docker run race_scene_sdk_container"

# Connect the first container to the network
docker network connect race_net race_scene_sdk_container --ip 192.168.100.5
check_status "docker network connect race_scene_sdk_container"

# Run the second container
docker run -d --name race_car_sdk_container registryonline-hulk.sankuai.com/custom_prod/com.sankuai.udm.udss/race:race_car_sdk
check_status "docker run race_car_sdk_container"

# Connect the second container to the network
docker network connect race_net race_car_sdk_container --ip 192.168.100.2
check_status "docker network connect race_car_sdk_container"

# Run the third container
docker run -d --name race_drone_sdk_container registryonline-hulk.sankuai.com/custom_prod/com.sankuai.udm.udss/race:race_drone_sdk
check_status "docker run race_drone_sdk_container"

# Connect the third container to the network
docker network connect race_net race_drone_sdk_container --ip 192.168.100.3
check_status "docker network connect race_drone_sdk_container"

# Run the fourth container
docker run -d --name race_user_sdk_container -v /etc/localtime:/etc/localtime:ro -v /etc/timezone:/etc/timezone:ro registryonline-hulk.sankuai.com/custom_prod/com.sankuai.udm.udss/race:race_user_sdk
check_status "docker run race_user_sdk_container"
```

```
# Connect the fourth container to the network
docker network connect race_net race_user_sdk_container --ip
192.168.100.4
check_status "docker network connect race_user_sdk_container"

echo "All commands executed successfully"
```

b. 执行启动脚本

```
sudo chmod +x start_race.sh
./start_race.sh

#可以使用 docker ps 查看当前启用的容器
```

5）停止 SDK

a. 新建停止脚本 stop_rach.sh，添加以下内容到脚本中

```
#!/bin/bash

# Function to check the last command status and exit if it failed
check_status() {
    if [ $? -ne 0 ]; then
        echo "Error: $1 failed"
        exit 1
    fi
}

# Stop and remove the first container
docker stop race_scene_sdk_container
check_status "docker stop race_scene_sdk_container"

docker rm race_scene_sdk_container
check_status "docker rm race_scene_sdk_container"

# Stop and remove the second container
docker stop race_car_sdk_container
check_status "docker stop race_car_sdk_container"
```

```
docker rm race_car_sdk_container
check_status "docker rm race_car_sdk_container"

# Stop and remove the third container
docker stop race_drone_sdk_container
check_status "docker stop race_drone_sdk_container"

docker rm race_drone_sdk_container
check_status "docker rm race_drone_sdk_container"

# Stop and remove the fourth container
docker stop race_user_sdk_container
check_status "docker stop race_user_sdk_container"

docker rm race_user_sdk_container
check_status "docker rm race_user_sdk_container"

echo "All containers have been stopped and removed successfully."
```

b. 执行脚本

```
sudo chmod +x stop_race.sh
./stop_race.sh

#可以自行设置是否删除容器，正常来讲 race_user_sdk_container 镜像可以重复使用，其他三个镜像每次使用都需要新建进行初始化
```

6）注意事项

a. 查看所有 SDK 服务是否启动

```
docker ps
```

b. 查看选手使用的 SDK 服务是否启动

```
# 进入选手使用的容器
docker exec -it race_user_sdk_container bash
# 查看 SDK 服务节点
rosnode list
```

```
root@14f88a643550:/home/sim_competition_sdk# rosnode list
/competition_msg_handler_node
/map_client_node
/rosout
```

c. 选手需要将代码在 docker 容器 race_user_sdk_container 中调试执行比赛代码，并打包为一个新的镜像，镜像名为 race_user:队伍简称（队伍简称会通过邮件的方式告知各位选手），确认全部调试完毕之后，提交到选手自己的 docker hub。

- Docker 镜像中包含完整仿真环境，地图，简单测试例等。（更多技术细节请见 SDK 使用文档。）
- 选手可熟悉比赛环境，并尝试调度算法，观察算法运行结果等等。
- 单机版完全线下运行（选手个人电脑），全程可用。单机运行不计分。
- 单机版推荐配置：6 核及以上 CPU，专用 Nvidia 或 AMD GPU，16G 内存，500G 硬盘。详情参考 SDK 使用说明

3. 在线提交（9.10-9.30）

- 使用单机版本调试好代码后，根据文档打包成 Docker 镜像，然后使用工具提交给比赛系统。步骤如下：

1）创建 Docker Hub 账号

a. 访问 https://hub.docker.com/

b. 点击 Sign Up 按钮，按照提示完成注册

c. 注册完成后，会收到一封验证邮件，点击邮件中的链接完成邮箱地址验证

2）安装 Docker（前文已经提供）

3）创建 Docker 镜像

保证 race_user_sdk_container 正在运行，然后执行以下指令

```
docker commit race_user_sdk_container race_user:队伍简称
```

race_user 是统一的名称

4）登录 Docker Hub

```
docker login
```

输入你的 Docker hub 用户名和密码

5）提交 Docker 镜像到 Docker Hub

```
docker tag race_user:队伍简称 yourdockerhubusername/race_user:队伍简称
docker push yourdockerhubusername/race_user:队伍简称
```

yourdockerhubusername 是用户的 Docker Hub 用户名

6) 验证是否提交成功

a. 可以在 Docker Hub 官网查看镜像是否已经推送。

b. 可以从 Docker Hub 拉取镜像并运行。

- 比赛系统会根据提交顺序运行镜像，并且计分。
- 运行算法的机器为固定配置（详细细节参考技术文档），经过验证，该配置能够满足绝大多数算法的计算。注：过于复杂的或者需要超高算力的算法可能无法顺利运行。
- 比赛服务器将配送任务**实时下发**到选手镜像；规划代码根据任务，及时生成规划方案（无人机、无人车轨迹点序列）发送至比赛服务器；比赛服务器进行仿真，并最终打分。
- 本阶段开放在线榜单，榜单日更，选手可看到成绩

4. 成绩确认

- 在初赛结束后，排名最高的 30 支队伍需提交代码，接受代码检查。
- 如有疑似不当行为，我们会联系参赛队求证。如确认不当行为，对应队伍将被取消成绩，后续排名队伍依次补位。
- 最终，通过检查后的 30 支队伍将晋级复赛。

5. 其他说明

- 参赛队在比赛过程中应保持诚信，遵守学术规范，以自身知识技能解决比赛问题，严禁抄袭、尝试破解比赛系统等行为。
- 如直接引用现有算法或使用开源代码，应在注释中予以说明。
- 参赛队之间可以就技术问题进行讨论，但禁止直接共享解决方案；如多支队伍的提交过于相似，可能会被判定抄袭。
- 比赛过程中，主办方提供指导培训、在线技术支持。参赛队应服从主办方安排与指示，如遇问题应及时联系主办方，充分沟通并协调解决。
- 赛事细节可能发生变更，以主办方发新发布信息为准。