Docker 与 MPI

何文阳

Docker 简介

- 容器化工具
- 利用 Linux 内核里的 namespace、cgroup 等功能，实现一个隔离的 Linux 环境，容器内的文件系统、网络、进程列表等与主机隔离
- 功能类似虚拟机，但成本低得多
- 可用于 ICS 的 class machine、模拟多台 Linux 机器等

Docker 简介

- 主要命令：docker run，创建容器
  - 可以用 docker run --help 查看选项
  - **容器创建后选项无法更改，直到容器删除为止**
- 常用选项：
  - image：镜像名，例如 ubuntu，可以在 https://hub.docker.com 查看
  - -v：将主机上的容器挂载到容器内，https://docs.docker.com/storage/volumes/
  - -e：设置环境变量
  - -i、-t：连接标准输入、支持 tty
  - --rm：容器停止后立刻删除容器
  - -p：暴露端口

Docker 简介

- 可以用 docker-compose 来定义容器配置
  - 可以方便地修改容器配置、定义多个容器
  - 语法参见 https://docs.docker.com/compose/compose-file/

Docker 简介

常用命令：

- docker ps -a，查看所有容器
- docker run，创建容器
- docker start，启动容器；docker restart，重新启动容器
- docker stop，停止容器；docker kill，杀死容器
- docker stats，实时查看容器状况（CPU、内存占用等）

在 Docker 中使用 CUDA

- 需要安装 NVIDIA Container Toolkit
  - https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/install-guide.html
  - 不要与 nvidia-docker 混淆。nvidia-docker 是较旧的方法，直接代替 docker 运行含 CUDA 的容器，启动容器需要使用 nvidia-docker 而不是 docker。目前的 NVIDIA Container Toolkit 作为插件安装在 Docker 里，可以直接用原版 Docker 启动 CUDA.
- 启动容器时添加 --gpus all 参数即可
- 常用镜像
  - nvidia/cuda
  - pytorch/pytorch
  - tensorflow/tensorflow

构建 Docker 容器

- 需要使用 Dockerfile
  - 描述从一个源镜像开始安装整个容器的过程
  - 使用 Ubuntu、Debian 时注意 DEBIAN_FRONTEND=noninteractive apt-get install -y tzdata
- 可以先在一个容器内测试，成功后再将安装过程记录为 Dockerfile

在 Docker 中使用 MPI

- 成功运行 MPI 有两个要素
  - 在各机器上启动程序需要配置 ssh
  - 各 MPI 程序实例之间能互相连接
- 由于 MPI 傻逼导致以上配置出错时程序会直接挂住，没有任何错误提示，只能手动排查
  - 注意防火墙
- Docker 默认配置下无法运行 MPI
  - 容器虽然拥有独立的 IP，但只在本机上有效，没法跨机器通信
- 需要在容器内安装 sshd
- 需要用 --network host 直接使用主机网络栈

使用 Singularity

- 可参考 https://sylabs.io/guides/3.0/user-guide/installation.html
- 注意 Singularity >= 3.0 需要 Go >= 1.13 来编译，而文档里的例子使用了 Go 1.11
- 已测试成功的 Singularity 版本：v3.8.4

使用 Singularity

- Singularity 的容器是一个单独的文件，一般以 .sif 为后缀名
  - 用 singularity build 命令生成这个文件
  - 可以从 Docker 镜像转换，也可以用 Singularity 的 definition 文件生成镜像
- 该文件可执行
  - 可以直接套 mpi 壳