High-Performance Data-Intensive Computing Systems Laboratory

With Big Data Comes Big Compute: Scaling Machine Learning onto Public and Commercial Clouds with Kubernetes – Part 2

Introduction to Containers and Kubernetes

IEEE International Conference on Big Data

December 2023

University of Missouri

Part 2 – Learning Objectives

- Deploying Scikit Learn ML Jobs to Kubernetes
- Deploying GPU Jobs to Kubernetes for Training Computer Vision Models
- Scaling Computer Vision Models on Kubernetes with Job Automation

MU-HPDI/Nautilus

- Sample Dockerfiles
- Sample Kubernetes YAML File
- Wiki with detailed walkthroughs for:
  - Getting Started
  - Creating PVC
  - Creating Pods
  - Creating Jobs

- Nautilus Portal:

  https://portal.nrp-nautilus.io
- JupyterHub Instance:

  https://bigdata-2023.nrp-nautilus.io/
- Tutorial Repository of Jupyter Project Pages, Code Samples, YAML, etc.

  https://github.com/MU-HPDI/bigdata-2023
- Git Clone Command:

  git clone https://github.com/MU-HPDI/bigdata-2023.git

Over a short break, we will ensure everyone has cloned this Repo into their JupyterLab environment

High-Performance Data-Intensive Computing Systems Laboratory

Part 2-A

National Research Platform Nautilus Research Cluster

Introduction to Containers and Kubernetes

IEEE International Conference on Big Data

December 2023

University of Missouri

Motivation

- Shallow ML modules often require extensive hyperparameter optimization to find optimal performance
- Deep Learning models require incredible amounts of compute to effectively train
- Using single developer machines or local on-prem resources often fail to scale effectively
- Kubernetes Clusters provide an efficient and scalable solution to training ML and deep learning models at scale
- This workshop will address the building and deployment of ML and deep learning containers to Nautilus to train deep networks

Part-2 Tutorial Outline

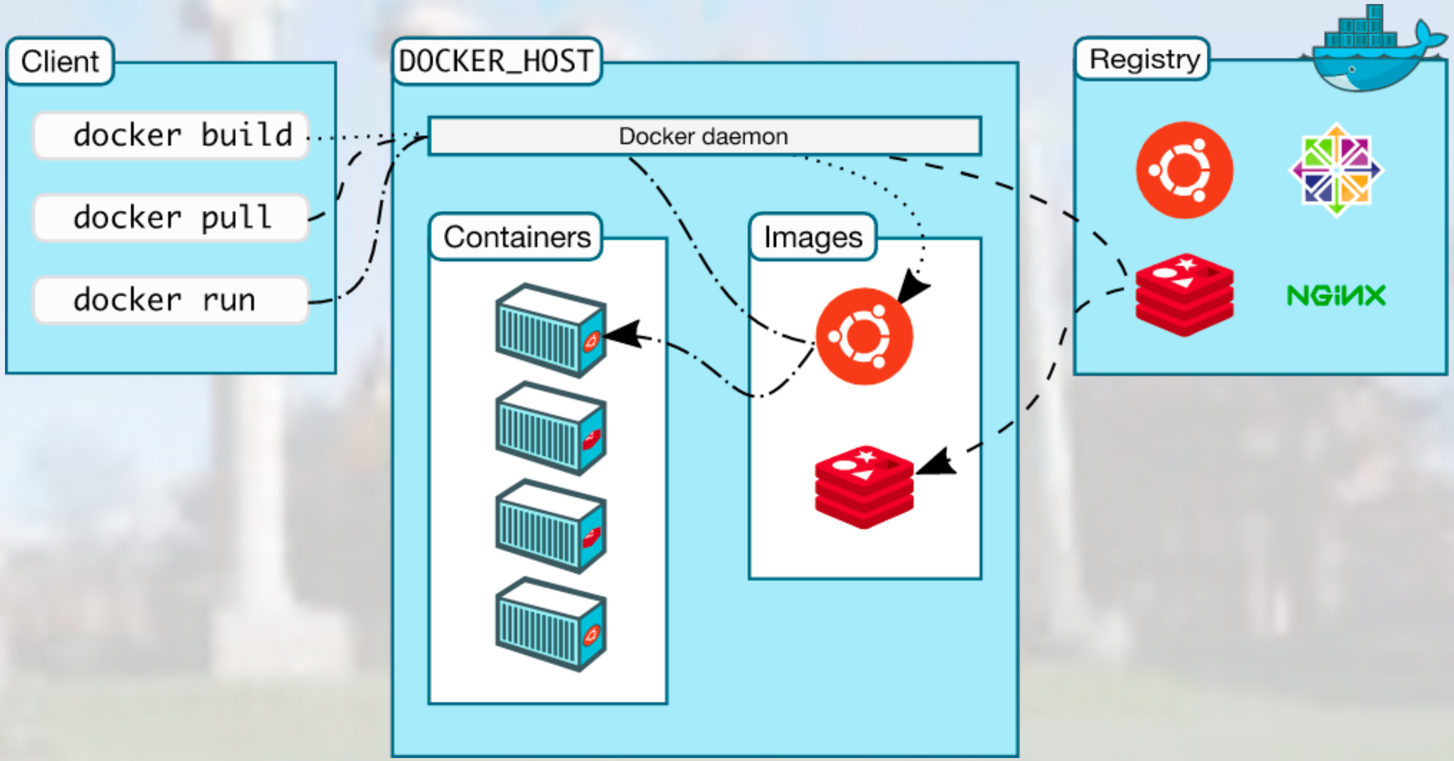
- Quick Containerization Review and Advanced:
  - Review of containerization
  - Building ML containers for Scikit-Learn
  - Building Optimized Containers for Deep Learning with PyTorch
  - Using Common Frameworks: Detectron2, MMDetection, and Ultralytics
- Kubernetes ML / Deep Learning:
  - Review of Kubernetes Architecture and Key Concepts
  - Introduction to NRP Nautilus HyperCluster
  - Migrating Data to NRP with S3
  - Deploying Scikit-Learn ML Jobs to Nautilus
  - Deploying GPU Jobs to Nautilus for Training Computer Vision Models

Containerization

Building Optimized Containers for Deep Learning with PyTorch

Containerization
Docker

- **Docker**
  - You can think of Docker containers as mini-VMs that contain all the packages, both at the OS and language-specific level, necessary to run your software.
- **Nautilus Gitlab**
  - Offers the ability to create repo for your code
  - Offers the ability to create easily maintained and developed custom images using CI/CD feature



Using Common Deep Learning Frameworks

Open MMLab, FAIR, and Ultralytics

Deep Learning Frameworks

- Many companies and research labs are developing open source Deep Neural Network frameworks to perform Computer Vision tasks
  - Open MMLab
  - Facebook AI Research
  - Ultralytics
  - Google
- We can build optimized Docker containers with these frameworks and use them as a baseline for research
- To use a given framework on Nautilus, we only need a Dockerfile → From the Dockerfile, we can build a container image that can be deployed on the cluster

MMLab Frameworks: MMDetection, MMClassification, MMSegmentation

- Open MMLab has developed a set of extensible libraries for performing key Computer Vision tasks:
  - Classification: MMClassification
  - Object Detection: MMDetection
  - Semantic Segmentation: MMSegmentation
- These libraries can serve as an excellent starting point for many CV applications
- Very large model zoo with community trained models and benchmarks
- Highly extensible and configurable frameworks

MMDetection Dockerfile

- Dockerfile contains all steps to create a fully functional MMDetection Python environment
- Add your code to the container or use this Dockerfile as a base in a multi-stage build

MU-HPDI/Nautilus

University of Missouri

```
FROM nvcr.io/nvidia/pytorch:22.06-py3

# arg
ARG MMDET_VERSION=2.25.0
ARG MMCV_VERSION=1.5.2
ARG MMCLS_VERSION=0.23.1
ARG BUILD_DIR="/build"

# env variables
ENV MPLCONFIGDIR /tmp
ENV TORCH_HOME /tmp
ENV MMCV_WITH_OPS 1
ENV FORCE_CUDA 1
ENV CUDA_HOME /usr/local/cuda

# create the build dir
RUN mkdir -p ${BUILD_DIR}

RUN apt update -y && apt install -y libpng-dev libjpeg-dev libgl-dev wget && rm -rf /var/lib/apt/lists/*
# install mm cv
# https://mmcv.readthedocs.io/en/latest/get_started/build.html#build-on-linux-or-macos
RUN wget https://github.com/open-mmlab/mmcv/archive/refs/tags/v${MMCV_VERSION}.tar.gz -O /tmp/mmcv.tar.gz
RUN tar -xzf /tmp/mmcv.tar.gz
RUN mv ./mmcv-${MMCV_VERSION} ${BUILD_DIR}/mmcv
# install it
RUN pip install -r ${BUILD_DIR}/mmcv/requirements/optional.txt
RUN pip install -v ${BUILD_DIR}/mmcv


# install mm detection
# https://github.com/open-mmlab/mmdetection/blob/v2.25.0/docs/en/get_started.md#customize-installation
RUN wget https://github.com/open-mmlab/mmdetection/archive/refs/tags/v${MMDET_VERSION}.tar.gz  -O /tmp/mmdet.tar.gz
RUN tar -xzf /tmp/mmdet.tar.gz
RUN mv ./mmdetection-${MMDET_VERSION} ${BUILD_DIR}/mmdet
# install it
RUN pip install ${BUILD_DIR}/mmdet
```

High-Performance Data-Intensive Computing Systems Laboratory
https://scottgs.mufaculty.umsystem.edu/

Detectron2

- Detectron2 is an open source framework for Object Detection and Instance Segmentation developed by FAIR
- The original Mask R-CNN model was developed and published via this framework
- Highly extensible with custom components and highly configurable
- High quality pretrained models and dataset benchmarks

Meta AI

University of Missouri

Detectron2 Dockerfile

- Dockerfile contains all steps to create a fully functional Detectron2 Python environment
- Add your code to the container or use this Dockerfile as a base in a multi-stage build

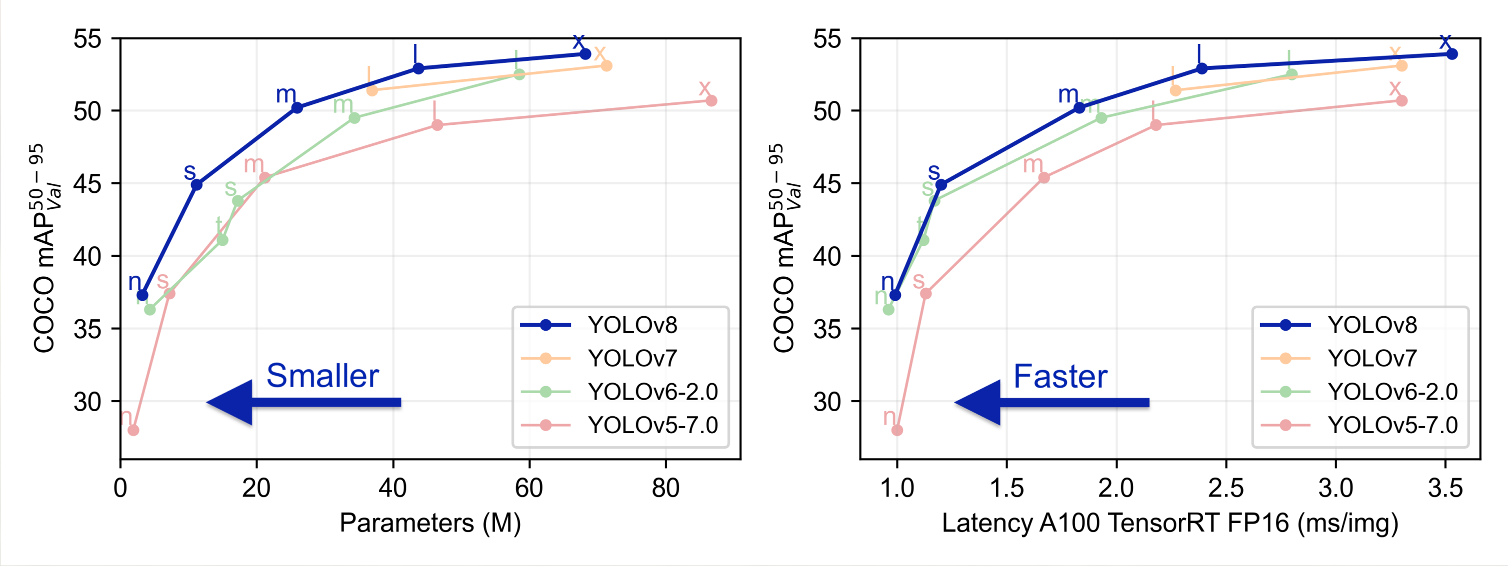
```
FROM nvcr.io/nvidia/pytorch:21.06-py3

# install system reqs
RUN apt update && apt install -y vim libgl-dev
RUN apt-get install --reinstall ca-certificates # for git

# env variables
ENV MPLCONFIGDIR /tmp
ENV TORCH_HOME /tmp
ENV FVCORE_CACHE /tmp


##########
# Detectron 2
#########
RUN git clone -b 'v0.4' --single-branch --depth 1 --recursive https://github.com/facebookresearch/detectron2.git /workspace/detectron2
RUN pip install -v  /workspace/detectron2/
```

Ultralytics

- Developer of most used Open-Source implementation of YOLOv3, YOLOv5, and more recently, YOLOv8
- You Only Look Once (YOLO) is a common real-time object detection architecture with many variations over the years
  - YOLO
  - YOLO9000 (a.k.a. YOLOv2)
  - YOLOv3
  - YOLOv5
  - YOLOv8

ultralytics YOLOv5

ultralytics YOLOv8



Ultralytics Dockerfile

- Dockerfile contains all steps to create a fully functional YOLOv3, YOLOv5, or YOLOv8 Python environment
- Add your code to the container or use this Dockerfile as a base in a multi-stage build

ultralytics/ultralytics

University of Misso

```
# Ultralytics YOLO 🚀, AGPL-3.0 license
# Builds ultralytics/ultralytics:latest image on DockerHub https://hub.docker.com/r/ultralytics/ultralytics
# Image is CUDA-optimized for YOLOv8 single/multi-GPU training and inference

# Start FROM PyTorch image https://hub.docker.com/r/pytorch/pytorch or nvcr.io/nvidia/pytorch:23.03-py3
FROM pytorch/pytorch:2.0.0-cuda11.7-cudnn8-runtime

# Downloads to user config dir
ADD https://ultralytics.com/assets/Arial.ttf https://ultralytics.com/assets/Arial.Unicode.ttf /root/.config/Ultralytics/

# Install linux packages
# g++ required to build 'tflite_support' package
RUN apt update \
    && apt install --no-install-recommends -y gcc git zip curl htop libgl1-mesa-glx libglib2.0-0 libpython3-dev gnupg g++
# RUN alias python=python3

# Security updates
# https://security.snyk.io/vuln/SNYK-UBUNTU1804-OPENSSL-3314796
RUN apt upgrade --no-install-recommends -y openssl tar

# Create working directory
RUN mkdir -p /usr/src/ultralytics
WORKDIR /usr/src/ultralytics

# Copy contents
# COPY . /usr/src/app  (issues as not a .git directory)
RUN git clone https://github.com/ultralytics/ultralytics /usr/src/ultralytics
ADD https://github.com/ultralytics/assets/releases/download/v0.0.0/yolov8n.pt /usr/src/ultralytics/

# Install pip packages
RUN python3 -m pip install --upgrade pip wheel
RUN pip install --no-cache -e . albumentations comet tensorboard thop pycocotools

# Set environment variables
ENV OMP_NUM_THREADS=1
```

Migrating Data to Nautilus

Deep Learning & Data

- Deep Learning algorithms require a vast amount of representative training data to effectively train
  - The more complex the network architecture, the more quality data is required
- Previous portions of this workshop have covered creating the containers to train the algorithms, but we have not yet covered how to stage data to Nautilus
  - All data for use on Nautilus will need to be moved to *persistent volumes* on the cluster
- Three ways to move data to a persistent volume on Nautilus:
  - Using KubeCTL
  - Using Commercial Cloud Storage
  - **Using Nautilus S3 Storage** *← Recommended*

Migrating Data to Nautilus: KubeCTL

- The command line Kubernetes tool, KubeCTL, has functionality to copy data to and from running pods
  - We can use this copy utility to move data to the cluster
- Advantages:
  - No additional installation or setup
  - No need to stage data in cloud storage
  - Requires only the KubeCTL command line tool
- Disadvantages:
  - Only small amounts (< 100 MB) of data can be moved per kubectl copy command
  - KubeCTL copy is *slow* at < 100 Mbps upload speeds
- How to:

```
kubectl cp localpath podname:/path/on/pod
```

Migrating Data to Nautilus: Using Commercial Cloud Storage

- We can use commercial cloud storage, such GCP Buckets or AWS S3 to move data to Nautilus
- Advantages:
  - Flexibility of cloud platform
  - Very fast transfer speeds
  - Capable of virtually any amount of data
- Disadvantages
  - Cost
  - Setup of Cloud Storage
  - Installation of Cloud Interface
  - Staging of data in cloud
- How to:
  - Create cloud bucket with Commercial Cloud Vendor (i.e., GCP)
  - Copy data to Bucket: `gsutil cp localpath gs://bucketName`
  - Create Google Cloud Pod on Nautilus
  - Copy data from Bucket: `gsutil cp gs://bucketName /path/on/pod`

Google Cloud

Amazon S3

MU-HPDI/Nautilus

Migrating Data to Nautilus: Using Nautilus S3 Storage

MU-HPDI/Nautilus

- NRP provides S3 bucket storage for free to Nautilus users upon request
- Advantages:
  - Free
  - High-throughput link to Nautilus cluster
  - Integration with S3-compatible software
  - Capability to handle large amounts of data
- Disadvantages:
  - Must request access
  - Setup of cloud integration
  - Staging of data into Nautilus S3
- How To:
  - Request access to cloud storage to receive Access ID and Keys
  - Install `rclone` or similar tool at data source and copy data from source to Nautilus S3
  - Create `rclone` or similar tool pod on Nautilus cluster and copy from S3 to Persistent Volume

High-Performance Data-Intensive Computing Systems Laboratory

Part 2-B

Deploying Scikit-Learn Jobs to Nautilus for Machine Learning

Introduction to Containers and Kubernetes

IEEE International Conference on Big Data

December 2023

University of Missouri

Prerequisites

- To run jobs on Nautilus, the following prerequisites must be met:
  - You have access to Nautilus and have been assigned a namespace
  - You have a container published to a public registry with the necessary code to perform the ML task
  - You have the data for the ML task on a persistent volume in the cluster

Access

ML Container

Data

Access the MachineLearning_K8s Notebook

Follow along activity

High-Performance Data-Intensive Computing Systems Laboratory

Part 2-C

Deploying GPU Jobs to Nautilus for Computer Vision Applications

Introduction to Containers and Kubernetes

IEEE International Conference on Big Data

December 2023

University of Missouri

Prerequisites – Review + GPU differences

- To run GPU jobs on Nautilus, the following prerequisites must be met:
  - You have access to Nautilus and have been assigned a namespace
  - You have a **GPU enabled** container published to a public registry with the necessary code to perform the CV task
  - You have the data for the CV task on a persistent volume in the cluster

Access

GPU
Container

Data

Access the DeepLearning_K8s Notebook

Follow along activity

High-Performance Data-Intensive Computing Systems Laboratory

Part 2-D
Automating Jobs on Nautilus using Bash and Python

Introduction to Containers and Kubernetes

IEEE International Conference on Big Data

December 2023

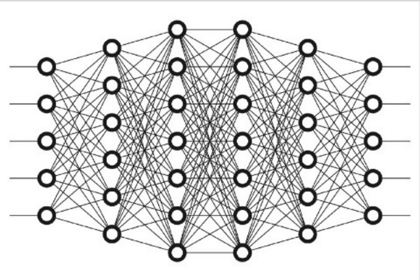
University of Missouri

Deep Learning on Nautilus: Semantic segmentation

- Iterations of Training Completed: 515,550
- Number of images Processed: 7,070,400
- Trainable Parameters Optimized: 23 millions per model
- The time it took to prepare the experimental set up and to run all the training sessions in parallel is 12 hours
- The actual time it would have take to train is 21 days 12 hours 45 minutes

Deep Learning on Nautilus: Transformer Research

Nautilus

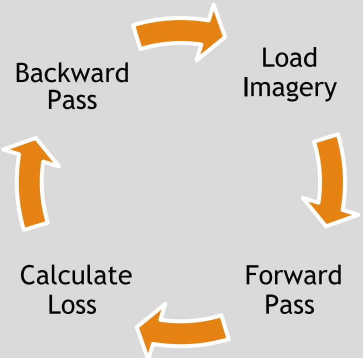


9 Deep Neural Architectures

27 Trained Models

124.74 TB Imagery Processed

3 Open Source HR-RSI Datasets



- 8,100 Epochs
- 30M iterations
- 1.7B parameters

Wall Clock Time: 76 days, 10 hours

https://www.freecodecamp.org/news/want-to-know-how-deep-learning-works-heres-a-quick-guide-for-everyone-1aedeca88076/
https://commons.wikimedia.org/wiki/File:PyTorch_logo_icon.svg
https://www.clipartmax.com/download/m2K9A0m2b1m2i8i8_4-o-clock-clip-art-clock-face-clip-art/

MU-HPDI/Nautilus – A resource for exploration

- Sample Dockerfiles
- Sample Kubernetes YAML File
- Wiki with detailed walkthroughs for:
  - Getting Started
  - Creating PVC
  - Creating Pods
  - Creating Jobs

Check back occasionally, we are adding more content and recipes for scaling

Machine Learning and Deep Learning