OpenFlightHPC-Learn Documentation

OpenFlightHPC

May 05, 2022

Overview

This site contains various workflows and examples of preparing and running complex HPC workloads with the assistance of the Flight User Suite.

CHAPTER 1

Acknowledgements

We recognise the respect the trademarks of all third-party providers referenced in this documentation. Please see the respective EULAs for software packages used in configuring your own environment based on this knowledgebase.

1.1 License

CHAPTER 2

Table of Contents

2.1 Purpose of this Documentation

The breadth of application of HPC environments can be intimidating. There are many different users and workflows for every application.

This documentation aims to provide some suitable examples to demonstrate the ease of establishing consistent workflows and the flexibility provided by the flight tools in assisting with workflow creation.

Traditionally, administrators handle a lot of workflow creation. From manually compiling applications all the way through to assisting users launching their workflows. With this in mind, these documents should be able to assist both admins and users to establish their own customised solutions with minimal disruption,

2.2 Bowtie

2.2.1 About

Bowtie is a sequencing toolset for aligning sets of DNA into large genomes.

2.2.2 Workflow

Installing Bowtie with Spack

Note: The flight environment will need to be activated before the environments can be created so be sure to run `flight start` or setup your environment to automatically activate the flight environment.

- Create a spack software environment:

```
[flight@gateway1 (scooby) ~]$ flight env create spack
```

- Activate the environment:

```
[flight@gateway1 (scooby) ~]$ flight env activate spack
```

- Install bowtie:

```
<spack> [flight@gateway1 (scooby) ~]$ spack install bowtie
```

Running a Bowtie Job

- Download example ecoli data:

```
<spack> [flight@gateway1 (scooby) ~]$ wget -O ecoli.fq http://tiny.cc/ecoli
```

- Create a job script in the current working directory:

```
<spack> [flight@gateway1 (scooby) ~]$ cat << EOF > mybowtiejob.sh
#!/bin/bash -l
#SBATCH -N 1
flight env activate spack
spack load bowtie
bowtie-build ecoli.fq e_coli
EOF
```

- Submit the job to the queue:

```
<spack> [flight@gateway1 (scooby) ~]$ sbatch mybowtiejob.sh
```

The results can then be reviewed from the slurm output file for the job in the current working directory.

2.3 Hadoop

2.3.1 About

Hadoop is a scalable, distributed computing solution provided by Apache. Similar to queuing systems, Hadoop allows for distributed processing of large data sets.

2.3.2 Workflow

Installing Hadoop Manually to Shared Filesystem

- Install dependencies for Hadoop (press 'y' to confirm the installation when prompted):

```
[flight@gateway1 (scooby) ~]$ sudo yum install java-1.8.0-openjdk.x86_64 java-1.8.
↪0-openjdk-devel.x86_64
```

- Download Hadoop v3.2.1:

```
[flight@gateway1 (scooby) ~]$ wget -O /tmp/hadoop.tgz http://tiny.cc/hadoop321
```

- Decompress the Hadoop installation to shared storage:

```
[flight@gateway1 (scooby) ~]$ cd /opt/apps
[flight@gateway1 (scooby) ~]$ tar xzf /tmp/hadoop.tgz
```

- Edit line 54 in `/opt/apps/hadoop-3.2.1/etc/hadoop/hadoop-env.sh` to point to the Java installation as follows:

```
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.242.b08-0.el7_7.x86_64/jre
```

Downloading the Hadoop Job

These steps help setup the Hadoop environment and download a spreadsheet of data which will Hadoop will sort into sales units per region.

- Download and source Hadoop environment variables:

```
[flight@gateway1 (scooby) ~]$ wget https://tinyurl.com/hadoopenv
[flight@gateway1 (scooby) ~]$ source hadoopenv
```

- Create job directory:

```
[flight@gateway1 (scooby) ~]$ mkdir MapReduceTutorial
[flight@gateway1 (scooby) ~]$ chmod 777 MapReduceTutorial
```

- Download job data:

```
[flight@gateway1 (scooby) ~]$ cd MapReduceTutorial
[flight@gateway1 (scooby) MapReduceTutorial]$ wget -O hdfiles.zip https://tinyurl.
↪com/hdinput1
[flight@gateway1 (scooby) MapReduceTutorial]$ unzip -j hdfiles.zip
```

- Check that job data files are present:

```
[flight@gateway1 (scooby) MapReduceTutorial]$ ls
desktop.ini  hdfiles.zip  SalesCountryDriver.java  SalesCountryReducer.java ␣
↪SalesJan2009.csv  SalesMapper.java
```

Preparing the Hadoop Job

- Compile java for job:

```
[flight@gateway1 (scooby) MapReduceTutorial]$ javac -d . SalesMapper.java␣
↪SalesCountryReducer.java SalesCountryDriver.java
```

- Create a manifest file:

```
[flight@gateway1 (scooby) MapReduceTutorial]$ echo "Main-Class: SalesCountry.
↪SalesCountryDriver" >> Manifest.txt
```

- Compile the final java file for job:

```
[flight@gateway1 (scooby) MapReduceTutorial]$ jar cfm ProductSalePerCountry.jar␣
↪Manifest.txt SalesCountry/*.class
```

Starting the Hadoop Environment

- Start the Hadoop distributed file system service:

```
[flight@gateway1 (scooby) MapReduceTutorial]$ $HADOOP_HOME/sbin/start-dfs.sh
```

- Start the resource manager, node manager and app manager service:

```
[flight@gateway1 (scooby) MapReduceTutorial]$ $HADOOP_HOME/sbin/start-yarn.sh
```

- Create directory for processing data and copy sales results in:

```
[flight@gateway1 (scooby) MapReduceTutorial]$ mkdir ~/inputMapReduce
[flight@gateway1 (scooby) MapReduceTutorial]$ cp SalesJan2009.csv ~/
↪inputMapReduce/
```

- Load the data into the distributed file system:

```
[flight@gateway1 (scooby) MapReduceTutorial]$ $HADOOP_HOME/bin/hdfs dfs -ls ~/
↪inputMapReduce
```

Running the Hadoop Job

- Execute the MapReduce job:

```
[flight@gateway1 (scooby) MapReduceTutorial]$ $HADOOP_HOME/bin/hadoop jar␣
↪ProductSalePerCountry.jar ~/inputMapReduce ~/mapreduce_output_sales
```

- View the job results:

```
[flight@gateway1 (scooby) MapReduceTutorial]$ $HADOOP_HOME/bin/hdfs dfs -cat ~/
↪mapreduce_output_sales/part-00000 | more
```

2.4 Jupyter Notebook

2.4.1 About

Jupyter Notebooks are a web-based development and visualisation environment. It provides flexible integration of notebooks, code and data in a portable and secure manner.

2.4.2 Workflow

Installing Jupyter with Conda

Note: The flight environment will need to be activated before the environments can be created so be sure to run `flight start` or setup your environment to automatically activate the flight environment.

- Create a conda software environment:

```
[flight@gateway1 (scooby) ~]$ flight env create conda
```

- Activate the environment:

```
[flight@gateway1 (scooby) ~]$ flight env activate conda
```

- Install Jupyter:

```
<conda> [flight@gateway1 (scooby) ~]$ conda install jupyter
```

- Install Notebook dependencies:

```
<conda> [flight@gateway1 (scooby) ~]$ conda install matplotlib
<conda> [flight@gateway1 (scooby) ~]$ conda install folium -c conda-forge
```

Launch a Jupyter Notebook

Note: These commands will need to be run from a graphical desktop session as it will launch a web browser. This is out of scope for this documentation, for more information on launching desktops, see the use documentation

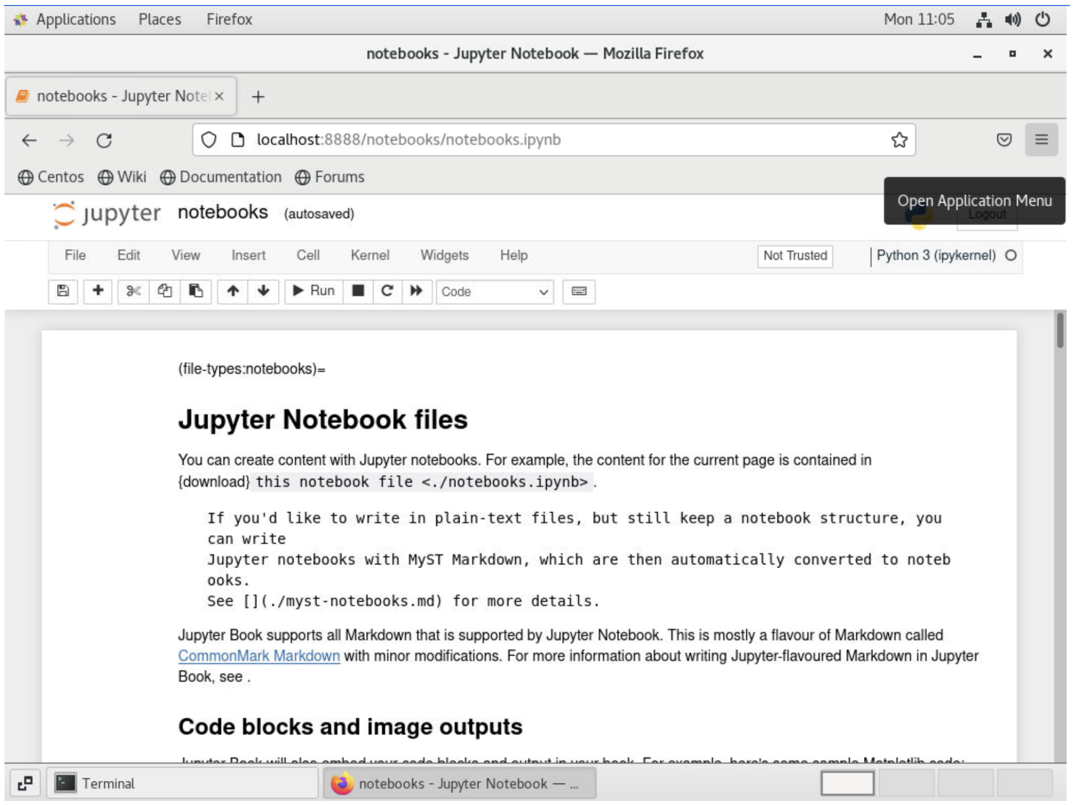
- Download the example notebook:

```
<conda> [flight@gateway1 (scooby) ~]$ curl -L https://jupyterbook.org/en/stable/_
↪downloads/12e9fb0f1c062494259ce630607cfc87/notebooks.ipynb > notebooks.ipynb
```

- Launch the notebook:

```
<conda> [flight@gateway1 (scooby) ~]$ jupyter notebook notebooks.ipynb
```

- A web browser will launch with the notebook displayed



(file-types:notebooks)=

Jupyter Notebook files

You can create content with Jupyter notebooks. For example, the content for the current page is contained in {download} `this notebook file <./notebooks.ipynb>`.

```
If you'd like to write in plain-text files, but still keep a notebook structure, you
can write
Jupyter notebooks with MyST Markdown, which are then automatically converted to noteb
ooks.
See [](./myst-notebooks.md) for more details.
```

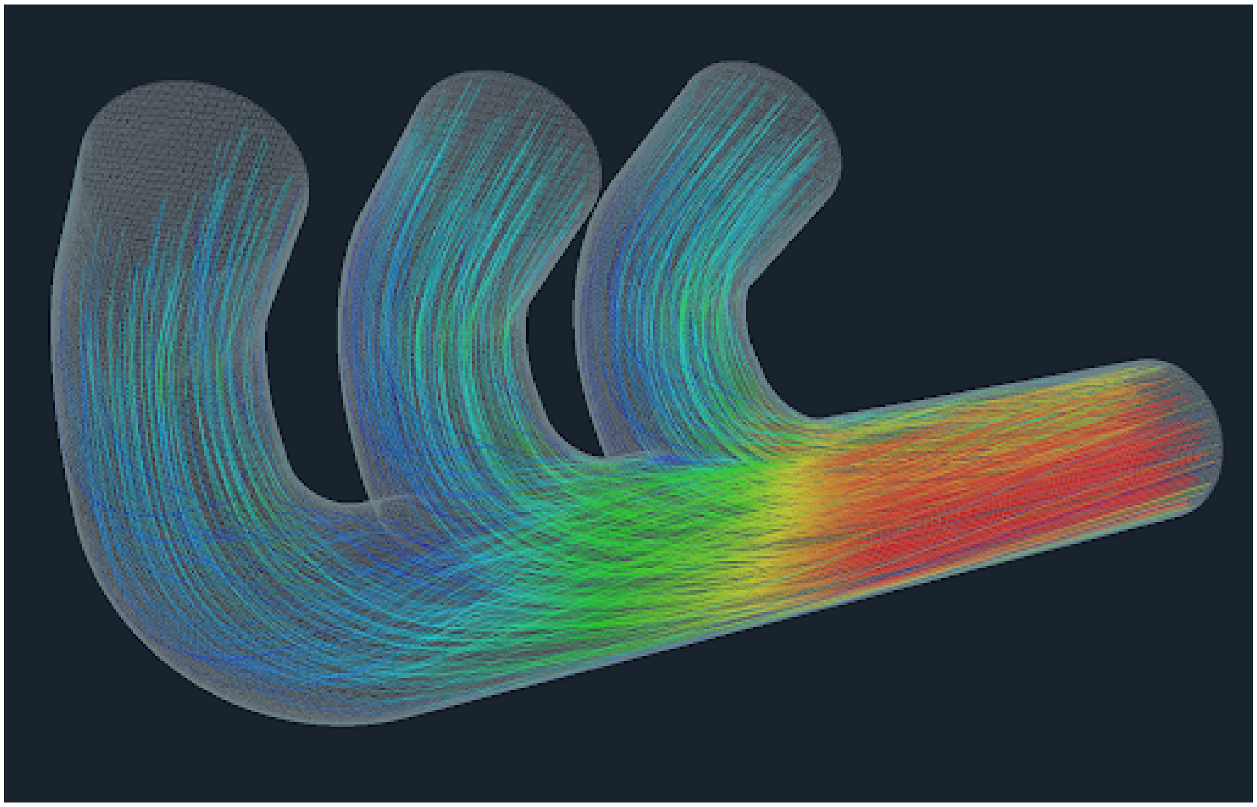
Jupyter Book supports all Markdown that is supported by Jupyter Notebook. This is mostly a flavour of Markdown called CommonMark Markdown with minor modifications. For more information about writing Jupyter-flavoured Markdown in Jupyter Book, see .

Code blocks and image outputs

2.5 OpenFOAM

2.5.1 About

OpenFOAM is a popular engineering application toolbox. It's open source and is used for simulating fluid flow around objects.



2.5.2 Workflow

Installing OpenFOAM with Gridware

Note: The flight environment will need to be activated before the environments can be created so be sure to run `flight start` or setup your environment to automatically activate the flight environment.

- Create a gridware software environment:

```
[flight@gateway1 (scooby) ~]$ flight env create gridware
```

- Activate the environment:

```
[flight@gateway1 (scooby) ~]$ flight env activate gridware
```

- Locate available OpenFOAM versions:

```
<gridware> [flight@gateway1 (scooby) ~]$ gridware search openfoam
```

- Install OpenFOAM 4.1:

```
<gridware> [flight@gateway1 (scooby) ~]$ gridware install apps/openfoam/4.1
```

Note: After pressing 'y' to accept the installation. Gridware will install various dependencies for OpenFOAM to ensure it runs

Checking OpenFOAM is Working

- Load the OpenFOAM module:

```
<gridware> [flight@gateway1 (scooby) ~]$ module load apps/openfoam
```

- Check an OpenFOAM command can be seen by viewing the help page:

```
<gridware> [flight@gateway1 (scooby) ~]$ icoFoam -help
```

Running a Simple OpenFOAM Job

- Create a job script in the current working directory:

```
<gridware> [flight@gateway1 (scooby) ~]$ cat << 'EOF' > myfoamjob.sh
#!/bin/bash
#SBATCH -N 1

# Activate environment and load OpenFOAM module
flight env activate gridware
module load apps/openfoam

# Create job directory from example job
cp -r $FOAM_TUTORIALS/incompressible/icoFoam/cavity/cavity $HOME/.
cd $HOME/cavity

# Calculate fluid pressure with OpenFOAM tools
blockMesh
checkMesh
icoFoam
EOF
```

- Submit the job to the queue system:

```
<gridware> [flight@gateway1 (scooby) ~]$ sbatch myfoamjob.sh
```

- Check that the job is running (iit will take 1-2 minutes to complete):

```
<gridware> [flight@gateway1 (scooby) ~]$ squeue
```

Viewing the Results

Once the cavity job has finished running, the results can be visualised through a desktop session.

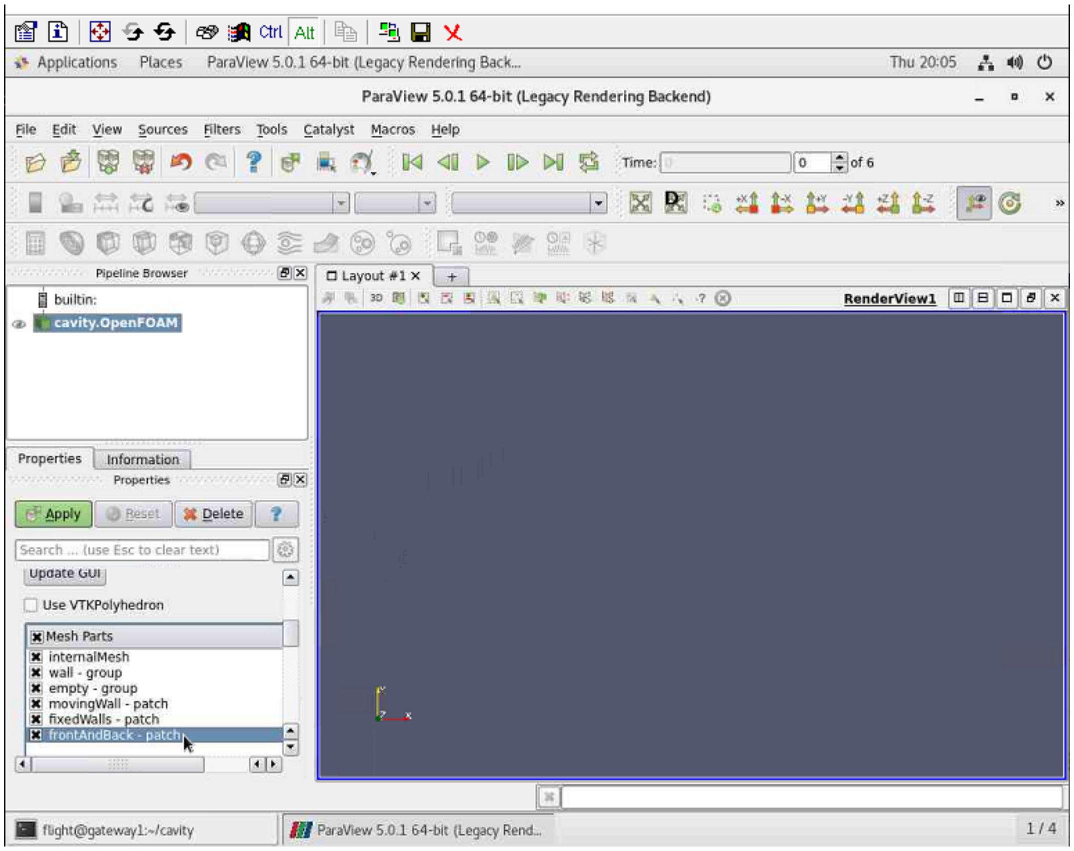
- Connect to VNC desktop (this is out of scope for this documentation, for more information on launching desktops, see the use documentation)
- In a terminal on the desktop session, ensure that the OpenFOAM module is loaded:

```
[flight@gateway1 (scooby) ~]$ flight env activate gridware
<gridware> [flight@gateway1 (scooby) ~]$ module load apps/openfoam
```

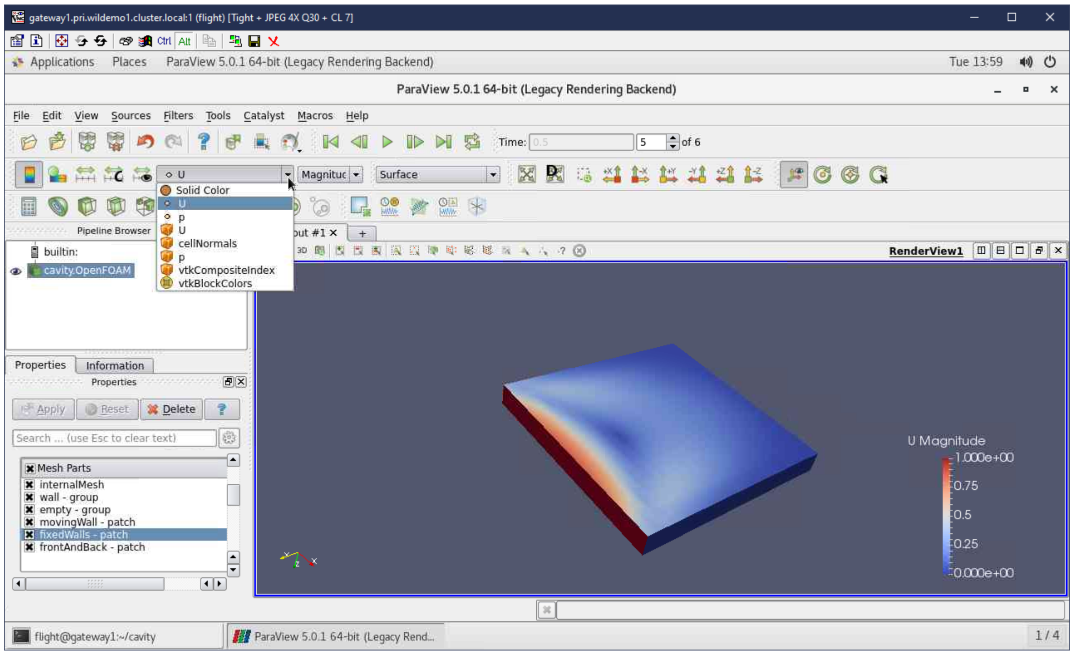
- Navigate to the cavity directory and launch the paraFoam viewer:

```
<gridware> [flight@gateway1 (scooby) ~]$ cd cavity
<gridware> [flight@gateway1 (scooby) cavity]$ paraFoam
```

- In the window that opens, scroll down to “Mesh parts” and select all the boxes, then click apply



- Next, change the display to ‘U’ in the dropdown menu, click play to change the timestamp and then click and drag to rotate the subject



2.6 Tensorflow

2.6.1 About

Tensorflow is an open-source machine learning platform. It provides an ecosystem of tools and libraries that allow researchers to build and deploy machine learning applications.

2.6.2 Workflow 1: Singularity

Installing Tensorflow with Singularity

Note: The flight environment will need to be activated before the environments can be created so be sure to run `flight start` or setup your environment to automatically activate the flight environment.

- Create a singularity software environment:

```
[flight@gateway1 (scooby) ~]$ flight env create singularity
```

- Activate the environment:

```
[flight@gateway1 (scooby) ~]$ flight env activate singularity
```

Running the Job

- Download the example job models:

```
<singularity> [flight@gateway1 (scooby) ~]$ git clone -b v1.13.0 https://github.
↪com/tensorflow/models.git
```

- Launch the tensorflow docker container with singularity to run the job:

```
<singularity> [flight@gateway1 (scooby) ~]$ singularity exec docker://tensorflow/
↪tensorflow:1.15.0 python ./models/tutorials/image/mnist/convolutional.py
```

2.6.3 Workflow 2: Conda

Note: The flight environment will need to be activated before the environments can be created so be sure to run `flight start` or setup your environment to automatically activate the flight environment.

Installing TensorFlow with Conda

- Create a conda software environment:

```
[flight@gateway1 (scooby) ~]$ flight env create conda
```

- Activate the environment:

```
[flight@gateway1 (scooby) ~]$ flight env activate conda
```

- Create a Python environment for tensorflow:

```
<conda> [flight@gateway1 (scooby) ~]$ conda create -n tensorflow python=3.6
```

- Activate the Python environment:

```
<conda> [flight@gateway1 (scooby) ~]$ source activate tensorflow
```

- Install the tensorflow package:

```
<conda> [flight@gateway1 (scooby) ~]$ pip install tensorflow==1.15
```

Running the Job

- Download the example job models:

```
<conda> [flight@gateway1 (scooby) ~]$ git clone -b v1.13.0 https://github.com/
↪tensorflow/models.git
```

- Execute the job with python:

```
<conda> [flight@gateway1 (scooby) ~]$ python ./models/tutorials/image/mnist/
↪convolutional.py
```