



**PURDUE USCMS Tier-2**  
Compact Muon Solenoid Experiment

# CMS Tier-2 Computing Tutorial

September 24, 2021

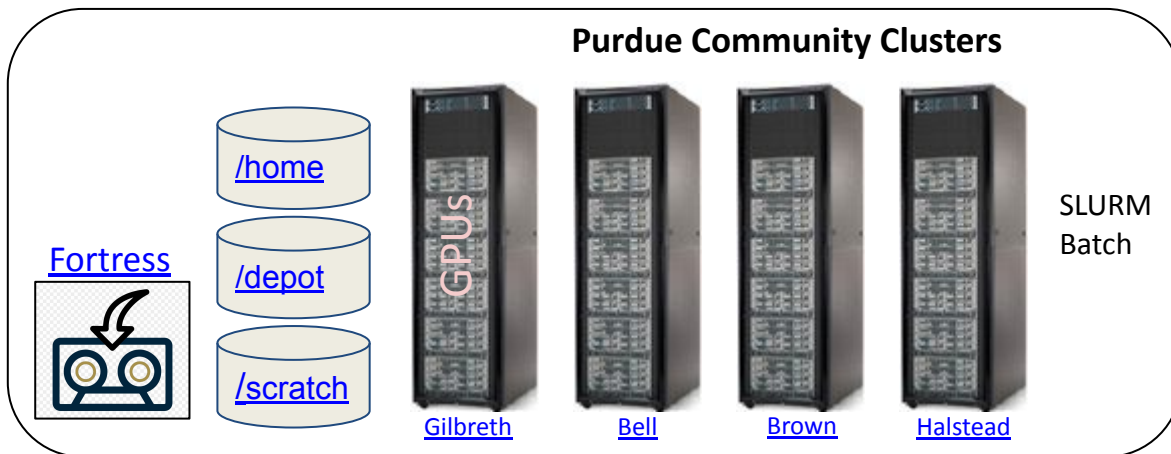
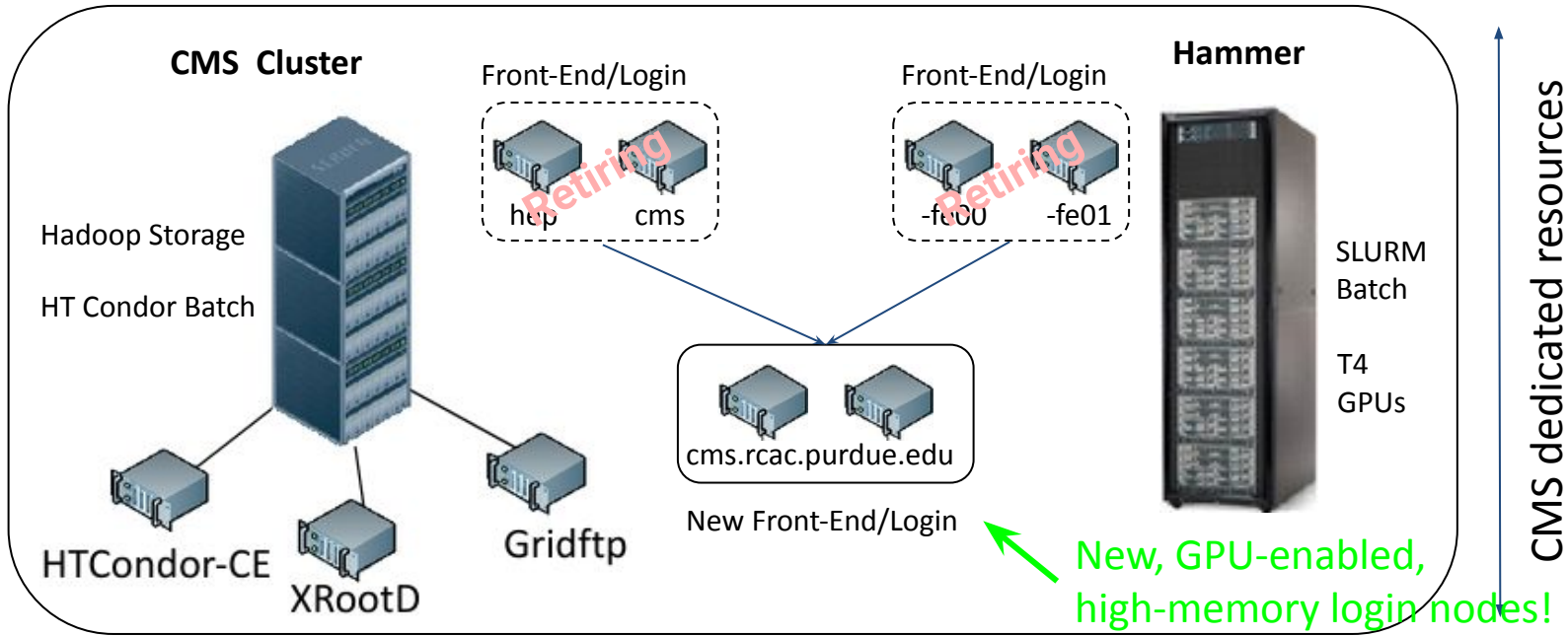
**Stefan Piperov**

[spiperov@purdue.edu](mailto:spiperov@purdue.edu)



Research Computing  
INFORMATION TECHNOLOGY

# PURDUE T2 Overview



# COMPUTE

- **Dedicated compute resources**
  - **CMS storage/batch cluster**
    - Provides ~10PB of distributed storage (HDFS)
    - A limited number (240) of batch slots via HT-Condor
    - Read-only access to HDFS storage
    - File access protocols (XRootD/Gridftp/WebDAV)
  - **Hammer cluster**
    - Provides a lot of (~10k) dedicated batch slots via SLURM
    - Provides some (20) GPU-enabled nodes (T4's)
    - Read-only access to HDFS storage
- **Purdue opportunistic resources**
  - A 'standby' queue on each Community Cluster provides short (4h) job slots. If you scale your jobs correctly (less than 4h run time), you get access to a lot of free job slots through CRAB and CMS-Connect
  - Gilbreth provides access to GPU-enabled nodes (V100's)

# STORAGE

- **Home area - private area, source code, development, *small!***
  - Available on all clusters
  - /home/<username>
- **Data Depot Group space - intermediate, read-write, medium size**
  - Available on all clusters. **Shared** by all CMS users!
  - /depot/cms/
  - Individual users may get dedicated sub-directories - e.g.:  
/depot/cms/users/spiperov/
  - Individual sub-groups have dedicated sub-directories, with *additional* quotas - e.g.: /depot/cms/top
- **Hadoop Distributed Filesystem (HDFS) - long term, large files/datasets**
  - Our main long-term storage space
  - /mnt/hadoop on Hammer and CMS cluster (read-only)
- /scratch - The Community Clusters provide large, fast local filesystems for *temporary* storage. Cleaned up regularly.
- /tmp - on CMS/Hammer, this is the main *temporary* space

# Interactive work

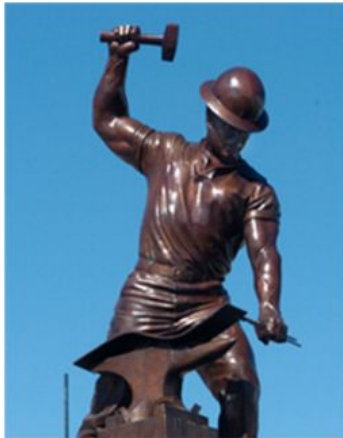
- **SSH to Log-in nodes:**
  - **CMS cluster:**
    - ssh <username>@hep.rcac.purdue.edu
    - ssh <username>@cms.rcac.purdue.edu
  - **Hammer cluster:**
    - ssh <username>@hammer.rcac.purdue.edu
  - **New CMS Front-Ends:**
    - ssh <username>@cms.rcac.purdue.edu
      - load-balanced, GPU-enabled, High-Memory (1TB)
  - **Other Community Clusters:**
    - ssh <username>@halstead.rcac.purdue.edu
    - ssh <username>@brown.rcac.purdue.edu
    - ssh <username>@gilbreth.rcac.purdue.edu
    - ssh <username>@bell.rcac.purdue.edu

Retiring!

# Interactive work - GUI

- Remote desktop on Community Clusters
  - <https://www.rcac.purdue.edu/compute/hammer/>

The screenshot shows a web browser at the URL <https://www.rcac.purdue.edu/compute/hammer/>. The page header includes the Purdue University logo, the text "Information Technology RESEARCH COMPUTING", and buttons for "Login" and "Get Help". A navigation menu lists: HOME, NEWS, EDUCATION, ACCOUNTS, COMPUTE, STORAGE, ENVISION, PURCHASE, SERVICES, ABOUT. Below the menu, a breadcrumb trail reads "Compute > Hammer".



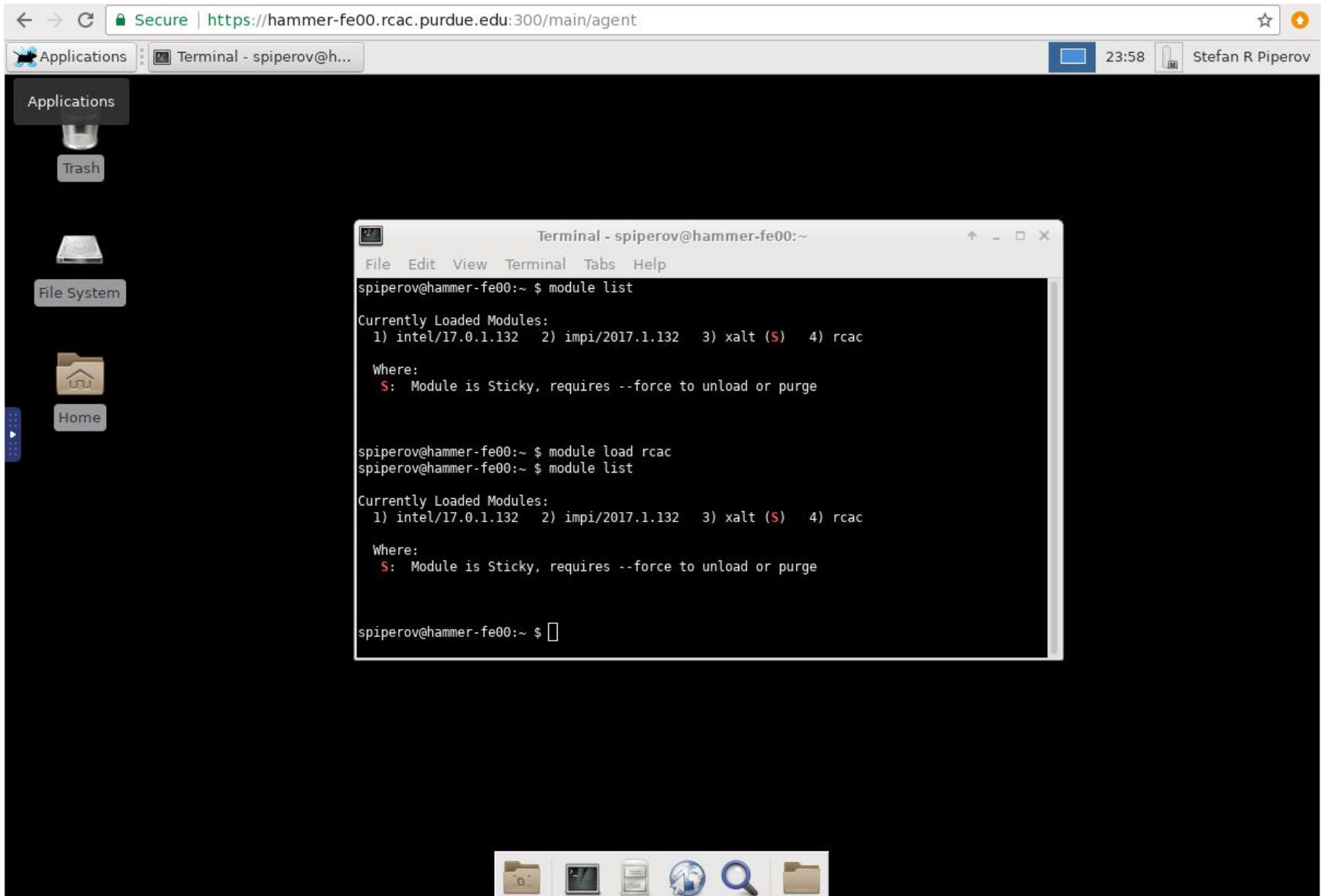
<b>Remote Desktop</b> <b>Launch</b>	<b>Jupyter Hub</b> <b>Launch</b>
--	-------------------------------------

## Overview of Hammer

Hammer is optimized for Purdue's communities utilizing loosely-coupled, high-throughput computing. Hammer was initially built through a partnership with HP and Intel in April 2015. Hammer was expanded again in late 2016. Hammer will be expanded annually, with each year's purchase of nodes to remain in production for 5 years from their initial purchase.

To purchase access to Hammer today, go to the [Cluster Access Purchase](#) page. Please subscribe to our Community Cluster Program Mailing List to stay informed on the latest purchasing developments or contact us via email at [rcac-](mailto:rcac-)

# Interactive work



The screenshot shows a web browser window with the address bar displaying `Secure | https://hammer-fe00.rcac.purdue.edu:300/main/agent`. The browser's top bar includes a search icon, the text "Applications", and a terminal icon labeled "Terminal - spiperov@h...". The system tray on the right shows the time "23:58" and the name "Stefan R Piperov".

The main content area displays a dark-themed desktop environment. On the left, there is a sidebar with icons for "Applications", "Trash", "File System", and "Home". A terminal window is open in the foreground, titled "Terminal - spiperov@hammer-fe00:~". The terminal output is as follows:

```
Terminal - spiperov@hammer-fe00:~
File Edit View Terminal Tabs Help
spiperov@hammer-fe00:~ $ module list

Currently Loaded Modules:
 1) intel/17.0.1.132  2) impi/2017.1.132  3) xalt (S)  4) rcac

Where:
 S: Module is Sticky, requires --force to unload or purge

spiperov@hammer-fe00:~ $ module load rcac
spiperov@hammer-fe00:~ $ module list

Currently Loaded Modules:
 1) intel/17.0.1.132  2) impi/2017.1.132  3) xalt (S)  4) rcac

Where:
 S: Module is Sticky, requires --force to unload or purge

spiperov@hammer-fe00:~ $
```

The terminal window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The desktop environment at the bottom features a taskbar with icons for a folder, a terminal, a document, a globe, a magnifying glass, and another folder.

# Best Practices

- Setup CMS environment on the local machine:
  - FrontEnd nodes (cms, hep, hammer, ...)
  - laptop/desktop
  - LXPLUS
- Develop and test analysis code locally
- Test analysis code on a small local dataset
- When convinced that analysis runs correctly locally - Submit multiple batch jobs:
  - either to local clusters via Condor and SLURM
  - or remotely via CRAB and CMS-Connect
- NB. As shared resources, the Front-End (login) nodes deliberately limit the resources available per individual user (20% RAM, 80% CPU)
- Full-size, long, production jobs should never be run there.
- It is much better to start an interactive whole-node SLURM job and have all the memory and CPUs to yourself, than to compete/interfere with everyone else on the FE



# Setup CMS environment

- After logging into a FrontEnd machine

```
$ export SCRAM_ARCH=slc7_amd64_gcc700
$ source /cvmfs/cms.cern.ch/cmsset_default.sh
$ export CMSSW_GIT_REFERENCE=/cvmfs/cms.cern.ch/cmssw.git.daily
$ mkdir MyAnalysis       #(only if creating it for a first time)
$ cd MyAnalysis
$ cmsrel CMSSW_10_6_18   #(only if setting up for a first time)
$ cd CMSSW_10_6_18/src
$ cmsenv
$ git cms-init
$ echo $CMSSW_BASE

$ export X509_USER_PROXY=~/.x509up_u`id -u`
$ voms-proxy-init -voms cms -valid 168:00
```

# Setup Python environment

- **On Community Clusters - multiple named environments**

```
$ module spider anaconda
```

```
  Versions:
```

```
  anaconda/5.1.0-py27
```

```
  anaconda/5.3.1-py37
```

```
$ module load anaconda/5.3.1-py37
```

```
$ conda create --name test_coffea python=3.7
```

```
$ source activate test_coffea
```

```
$ pip install --upgrade coffea
```

```
$ conda install -c conda-forge xrootd
```

```
$ conda install nb_conda
```

```
$ source deactivate #to exit the environment
```

```
$ module load anaconda/5.1.0-py27
```

```
$ conda create --prefix ~/test/ml/uprootenv python=2.7
```

```
$ source activate ~/test/ml/uprootenv
```

```
$ conda install -c conda-forge uproot
```

```
$ conda install -c conda-forge tensorflow keras numpy pandas
```

See more examples of managing [packages](#) and [environments](#)

# Copy small dataset locally

- To develop/test your analysis code, you only need one (or a few) input files, and not the complete dataset.
- You can use these commands to copy such files locally:

```
$ voms-proxy-init -voms cms -valid 168:00   #(in case you have not done so already)
```

```
$ xrdcp Note: redirector - no need to know the exact location!  
root://cmsxrootd.fnal.gov/store/relval/CMSSW_10_6_4/RelValZMM_13/MINIAODSIM/PUpmx25ns_106X_upgrade2018_realistic_v9-v1/10000/EC35B5C1-A0A7-574F-8D7A-FCB4C3FBECBE.root ./
```

```
$ xrdcp 'xrdcp' can also do recursion now!  
-r ..
```

```
$ gfal-copy Note: Precise locations needed!  
gsiftp://cms-gridftp.rcac.purdue.edu/store/relval/CMSSW_10_6_1/RelValZEE_13/MINIAODSIM/PU25ns_106X_mc2017_realistic_forECAL_v6_HS-v3/20000/83E6A167-6CD2-BF48-8937-AC79791ACC72.root  
file:///home/spiperov/DAS2020/CMSSW_10_6_4/src/
```

```
$ gfal-copy can do recursion!  
-r  
gsiftp://cms-gridftp.rcac.purdue.edu/store/relval/CMSSW_10_6_1/RelValZEE_13/MINIAODSIM/PU25ns_106X_mc2017_realistic_forECAL_v6_HS-v3 file:///home/spiperov/DAS2020/CMSSW_10_6_4/src/
```

**NB.** Sometimes the OSG tools are conflicting with CMSSW tools, and gfal-copy starts crashing. To fix that, execute:

```
$ source /cvmfs/oasis.opensciencegrid.org/osg-software/osg-wn-client/3.5/current/e17-x86_64/setup.sh  
$ source /cvmfs/oasis.opensciencegrid.org/osg-software/osg-wn-client/3.4/current/e16-x86_64/setup.sh
```

# Working with Datasets

- Main navigational tool - Data Aggregation System ([DAS](#)) at CERN (<https://cmsweb.cern.ch/das/>)
  - Universal and (somewhat) intuitive
  - Slow!
- Command-line tool - [dasgoclient](#)
  - Much faster
  - Very flexible, when combined with other UNIX-shell tools

```
$ voms-proxy-init -voms cms -valid 168:00 -rfc
$ /cvmfs/cms.cern.ch/common/dasgoclient -examples
$ /cvmfs/cms.cern.ch/common/dasgoclient -query="dataset=/EG/Run2010A*/AOD"
```

- python APIs - [cmssw\\_das\\_client.py](#), DBS client [examples](#)
  - can be integrated in directly in your code

# Working with Datasets

- Do we have this dataset at Purdue?

results format: 50    dbs instance: prod/global    autocompletion: disable    Search    Reset

dataset=/EG/Run2010A-Apr21ReReco-v1/AOD

Show DAS keys description

Showing 1—1 records out of 1.    <first | prev | next | last>

Dataset: [/EG/Run2010A-Apr21ReReco-v1/AOD](#)  
Creation time: 2011-04-22 18:12:14 Physics group: NoGroup Status: **VALID** Type: data Dataset size: 4486583998147 (4.5TB)  
Number of blocks: 74 Number of events: 53163466 Number of files: 3503  
[Release](#), [Blocks](#), [Files](#), [Runs](#), [Configs](#), [Parents](#), [Children](#), [Sites](#), [Physics Groups](#) XSDB Sources: [dbs3](#) [show](#)

Showing 1—1 records out of 1.    <first | prev | next | last>

processing time: 6.960020008 sec

DAS version: git=04.06.05 go=go1.12.5 date=2020-03-04 18:29:29.046915631 +0100 CET m=+0.040029733

Ask DAS:

- Enter Dataset Name
- Click “Sites”

- Or ask the DAS GO Client:

```
$ dasgoclient -query="site dataset=/EG/Run2010A-Apr21ReReco-v1/AOD"
T1_UK_RAL_Buffer
T1_UK_RAL_MSS
T3_CH_CERN_OpenData
T3_TW_NTU_HEP
```

# Working with Datasets

Do we need this dataset at Purdue?

Well, it depends:

- If it exists at another Tier-2, and you only want to run over a few events/files, then - NO. You can access it remotely via [XRootD](#) (AAA):

- directly in ROOT:

```
TFile *f
=TFile::Open("root://cmsxrootd.fnal.gov//store/mc/SAM/GenericTTbar/GEN-SIM-RECO/CMSSW_5_3_1_START53_V5-v1/0013/CE4D66EB-5AAE-E111-96D6-003048D37524.root");
```

- or in CMSSW:

```
process.source = cms.Source("PoolSource",
    fileName = cms.untracked.vstring('root://cmsxrootd.fnal.gov//store/myfile.root')
)
```

*Just prepend the address of the XRootD redirector*

- But if you plan to run your analysis multiple times, on the entire dataset, then - YES, it's better to copy it here at Purdue.
  - Create a Rucio "[rule](#)" - very easy!
  - Copy the files yourself (tedious, but only option for privately produced datasets at other sites, not registered in Rucio)

E.g.:

```
xrdcp root://stormgf3.pi.infn.it:1094//store/user/PrivateProd/... root://xrootd.rcac.purdue.edu//store/user/piperov/
```

# Manage Datasets via Rucio

CMS has switched recently its data management systems from PhEDEx to [Rucio](#). For an end-user to manage their files and datasets, one first needs to

## Setup their Rucio environment

```
$ source /cvmfs/cms.cern.ch/cmsset_default.sh
$ export X509_USER_PROXY=~/.x509up_u`id -u`
$ voms-proxy-init -voms cms -valid 168:00
$ source /cvmfs/cms.cern.ch/rucio/setup-py3.sh
$ export RUCIO_ACCOUNT=piperov
$ rucio whoami #check that it all worked fine
```

Data placement/copy in Rucio is controlled via ‘rules’.

## Misc. Rucio commands

```
$ rucio list-rules --account $RUCIO_ACCOUNT
$ rucio rule-info [RULE_HASH]
$ rucio list-account-limits $RUCIO_ACCOUNT
```

## To copy a dataset, you need to create a new rule

```
$ rucio add-rule cms:/CMS/DATA/SET/NAME 1 T2_US_Purdue
or:
$ rucio add-rule cms:/CMS/DATA/SET/NAME#BLOCK-NAME 1 T2_US_Purdue
or even better:
$ rucio add-rule --ask-approval --lifetime 2592000 cms:/CMS/DATA/SET/NAME#BLOCK-NAME 1 T2_US_Purdue
```

# Rucio - advanced

**NOTE:** Rucio's concept of 'dataset' is what CMS historically calls a 'block'. What CMS normally calls a 'dataset' is called 'container' in Rucio!

## Managing groups of datasets via containers

Basic idea: Create one container for all datasets needed for your analysis, and then manage its copying to a site via single rule (instead of individual rules for each dataset).

```
$ rucio add-container user.piperov:/Analyses/Hmumu2020/USER #create the container
$ rucio attach user.piperov:/Analyses/Hmumu2020/USER cms:/SingleMuon/Run2018A-02Apr2020-v1/NANOAOB \
cms:/SingleMuon/Run2018B-02Apr2020-v1/NANOAOB #add two datasets to the container
$ rucio add-rule user.piperov:/Analyses/Hmumu2020/USER 1 T2_US_Purdue #create a rule to copy to Purdue
$ rucio attach user.piperov:/Analyses/Hmumu2020/USER cms:/SingleMuon/Run2017B-02Apr2020-v1/NANOAOB #add one more DS
$ rucio detach user.piperov:/Analyses/Hmumu2020/USER cms:/SingleMuon/Run2018A-02Apr2020-v1/NANOAOB #and remove one
```

```
$ rucio list-content user.piperov:/Analyses/Hmumu2020/USER #List contents of your container
```

```
+-----+-----+
| SCOPE:NAME | [DID TYPE] |
+-----+-----+
| cms:/SingleMuon/Run2016B-02Apr2020-v1/NANOAOB | CONTAINER |
| ... |
+-----+-----+
```

```
$ rucio list-dids --filter type=CONTAINER user.piperov:* #List all your containers
```

```
+-----+-----+
| SCOPE:NAME | [DID TYPE] |
+-----+-----+
| user.piperov:/Analyses/Tests/USER | CONTAINER |
+-----+-----+
```

```
$ rucio delete-rule [RULE_HASH] #delete the container from a site. (RULE_HASH from the add-rule command earlier)
$ rucio erase user.piperov:/Analyses/Hmumu2020/USER #delete container permanently from all sites. Note: this is final!
```



# Submitting Jobs

- Distributed:
  - CRAB
    - [Send CMSSW jobs to many CMS sites](#)
  - CMS-Connect
    - [Send Condor jobs to many CMS sites](#)
- Local:
  - Condor
    - [Send jobs to our local CMS cluster](#)
      - `condor_submit sleep-condor.jdl`
  - SLURM
    - [Send jobs to Hammer cluster](#)
    - Send a job to a GPU-enabled Hammer node
      - `sbatch --account=cms --partition=hammer-gpu --time=0:05:00 nvidia-smi.sub`
      - `sinteractive --account=cms --partition=hammer-gpu`
    - Send a job to Gilbreth (GPU) node
      - `(ssh gilbreth.rcac.purdue.edu)`
      - `sbatch --account=cms --gres=gpu:2 --constraint=V100 --time=0:05:00 nvidia-smi.sub`
      - `sinteractive --account=cms --gres=gpu:1 --constraint=V100`

# Storing and Sharing Data

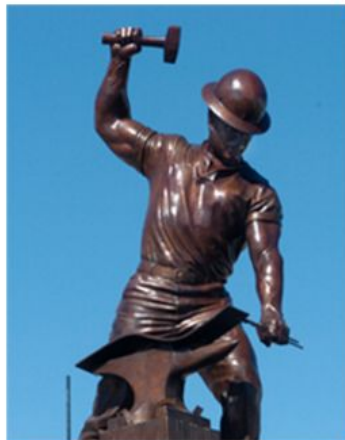
- Once your jobs are finished, where to store the output files?
  - **NOT** in /tmp or /scratch - for sure! That's only for temporary storage
    - those get cleaned - frequently
  - Home directory - probably too small
  - Data Depot - for R/W access and sharing with the group
  - HDFS - best for long-term storage (R/O) and sharing with the collaboration worldwide
    - just point them to your /store/user or /store/group directory
  - ```
$ xrdcp outfile.root root://xrootd.rcac.purdue.edu//store/user/piperov/
```

    - HDFS is already the default stage-out location for your CRAB jobs
    - for local jobs - add a bunch of xrdcp commands at the end of the job
  - [Fortress](#) - for archival storage on tape
    - HTAR/HSI [commands](#)

# Jupyter Notebooks

- Jupyter Hubs on Community Clusters
  - <https://www.rcac.purdue.edu/compute/hammer/>

The screenshot shows a web browser at the URL <https://www.rcac.purdue.edu/compute/hammer/>. The page header includes the Purdue University logo, the text "Information Technology RESEARCH COMPUTING", and buttons for "Login" and "Get Help". A navigation menu contains links for HOME, NEWS, EDUCATION, ACCOUNTS, COMPUTE, STORAGE, ENVISION, PURCHASE, SERVICES, and ABOUT. Below the menu, a breadcrumb trail reads "Compute > Hammer". The main content area features a bronze statue of a hammer-wielding worker on the left. To the right are two buttons: "Remote Desktop" and "Jupyter Hub". The "Jupyter Hub" button has a "Launch" sub-button that is circled in red.



Remote Desktop

Launch

Jupyter Hub

Launch

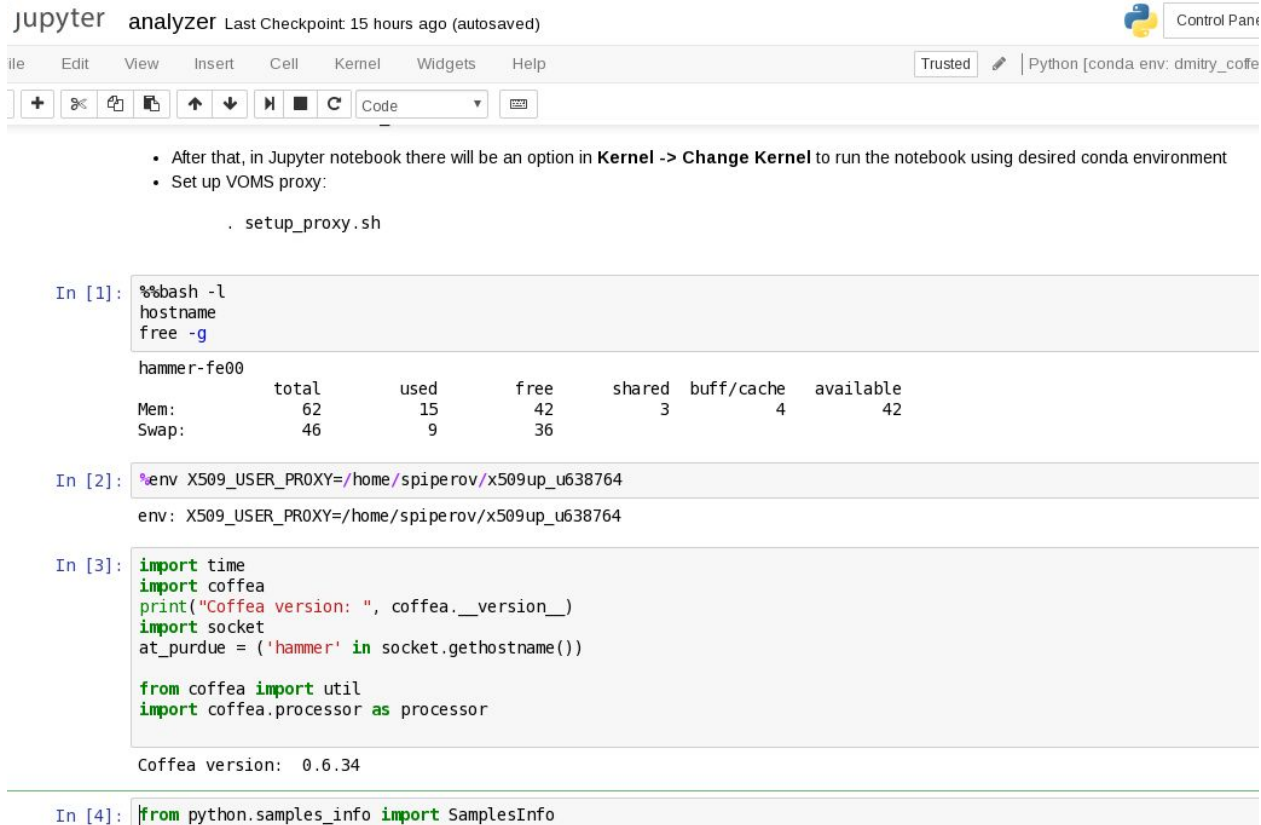
## Overview of Hammer

Hammer is optimized for Purdue's communities utilizing loosely-coupled, high-throughput computing. Hammer was initially built through a partnership with HP and Intel in April 2015. Hammer was expanded again in late 2016. Hammer will be expanded annually, with each year's purchase of nodes to remain in production for 5 years from their initial purchase.

To purchase access to Hammer today, go to the [Cluster Access Purchase](#) page. Please subscribe to our Community Cluster Program Mailing List to stay informed on the latest purchasing developments or contact us via email at [rcac-](mailto:rcac-)

# Jupyter Notebooks

- “...web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.”



The screenshot shows a Jupyter Notebook interface with the following elements:

- Header: "jupyter analyzer Last Checkpoint: 15 hours ago (autosaved)" and a "Control Panel" button.
- Menu: "File Edit View Insert Cell Kernel Widgets Help".
- Trust status: "Trusted" and environment: "Python [conda env: dmitry\_cofe]".
- Toolbar: Includes buttons for new, save, copy, paste, undo, redo, and a dropdown menu set to "Code".
- Code cells:
  - In [1]:

```
%%bash -l
hostname
free -g
```

Output:

```
hammer-fe00
total        used        free        shared  buff/cache   available
Mem:           62          15          42           3           4           42
Swap:          46           9           36
```
  - In [2]:

```
!env X509_USER_PROXY=/home/spipe ro/v/x509up_u638764
```

Output:

```
env: X509_USER_PROXY=/home/spipe ro/v/x509up_u638764
```
  - In [3]:

```
import time
import coffea
print("Coffea version: ", coffea.__version__)
import socket
at_purdue = ('hammer' in socket.gethostname())

from coffea import util
import coffea.processor as processor
```

Output:

```
Coffea version: 0.6.34
```
  - In [4]:

```
from python.samples_info import SamplesInfo
```

- Nice introductory [tutorial](#) and a gallery of interesting [examples](#)
- Best experienced *live!* Try the [CMSDAS pyROOT exercise](#).

# Speeding things up

When your analysis gets too big to fit in RAM, or on one CPU or node...

→ run it in parallel!

- ◆ Apache [Spark](#)
  - big, general purpose Big Data framework, written in Java
  - well integrated with the rest of Apache's ecosystem (e.g. Hadoop)
  - claimed to be best for Machine Learning
  - requires a cluster manager and a distributed storage system
  - <https://indico.cern.ch/event/726985/>
  
- ◆ [DASK](#)
  - purely Python library, designed for parallel computing - either on the laptop, or on a cluster
  - dynamic task scheduling
  - "Big Data" types of collections for distributed environments
  - smaller, lightweight, runs on your laptop
  
- ◆ (future tutorials)

# More Guides and Tutorials

## Useful External Tutorials

### [CMS Data Analysis School](#)

[Pre-exercises](#) (very useful for beginners!)

[Short exercises](#) (OFFLINE)

[Long exercises](#)

### [Hands-on Advanced Tutorial Sessions at the LPC \(HATS\)](#)

[Git/GitHub](#)

[Containers](#)

[Data and MC Processing](#)

[Machine Learning](#)

[Visualization](#)

[Jupyter and pyROOT](#)

[Condor/CMSConnect/CRAB3](#)

[Columnar Analysis Tools](#)

[Effective Scale Out Techniques](#)

[Uproot and Awkward Array for columnar analysis](#)

[Trigger](#) [Tau](#) [MET](#) [Muon](#) [Electron and Photon](#)

[Jet Algorithms and Substructure \(Jets I\)](#)

[Jet Energy Corrections and Pile-Up Mitigation \(Jets II\)](#)

[\(B,t,H,W,Z\) Tagging](#) [Generators](#)

# User Guides and Contact

- Main page of User's guide:  
<https://www.physics.purdue.edu/Tier2/user-info/>
- Community Clusters [docs](#)
- For most CMSSW related issues -
  - the CMS [WorkBook](#)
  - the [Framework and Event Data Model Offline Guide](#)
- Email us for support:
- [cms-support@lists.purdue.edu](mailto:cms-support@lists.purdue.edu)

# CMS Cluster (still) in MATH





# (Backup Slides)

- Backup slides

[pyROOT in Jupyter](#)

Other GPU-enabled machines:

- [lxplus-gpu.cern.ch](http://lxplus-gpu.cern.ch) - has mostly Nvidia T4's
- [cmslpcgpu\[1-3\].fnal.gov](http://cmslpcgpu[1-3].fnal.gov) - has Nvidia P100's

# Setup ML+pyROOT environment

- There is an example of setting up and using a complex Machine Learning environment with pyROOT and CMSSW here:  
<https://github.com/piperov/MLpyROOT>
- Try it out and see if you can add more packages to your environment.

# Legacy CMSSW, OS

- **There are still a small number of analyses done in the old ScientificLinux6 (SL6,RHEL6) OS**
  - **it its 'end-of-life', not supported anymore**
  - **replaced everywhere with CentOS7 (CC7,RHEL7).**
  - **However, some legacy CMSSW releases still need SL6 environment**
- **For compatibility, we provide it via Singularity containers:**
  - `/depot/itap/singularity/cms/cmssw-slc6`

**After starting the container, you should set up your CMSSW environment in the usual way, but for the older `SCRAM_ARCH=slc6...`**

```
$ export SCRAM_ARCH=slc6_amd64_gcc630
$ source /cvmfs/cms.cern.ch/cmsset_default.sh
$ cmsrel CMSSW_9_3_2 (if necessary)
$ cd CMSSW_9_3_2/src
$ cmsenv
$ git cms-init
$ source /cvmfs/cms.cern.ch/crab3/crab.sh
$ voms-proxy-init -voms cms -valid 168:00
..etc.
```