CMS Tier-2 Computing Tutorial

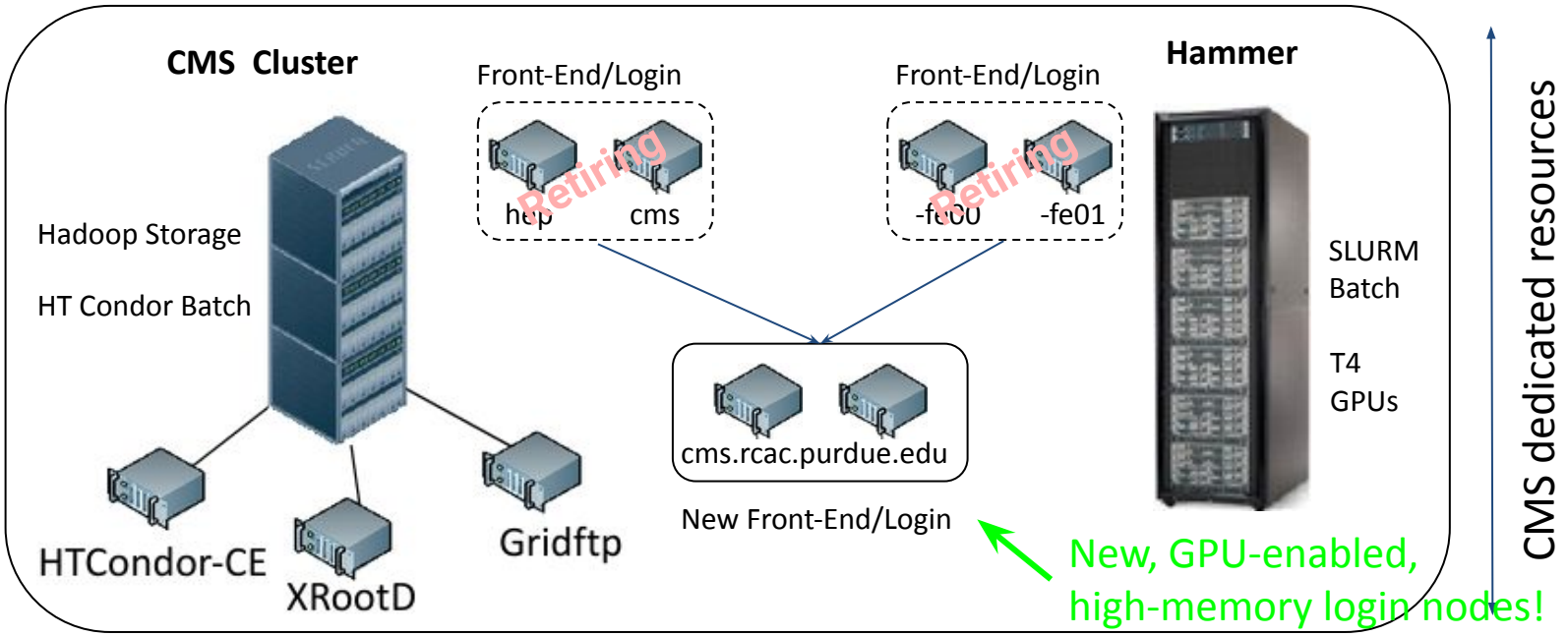
September 24, 2021

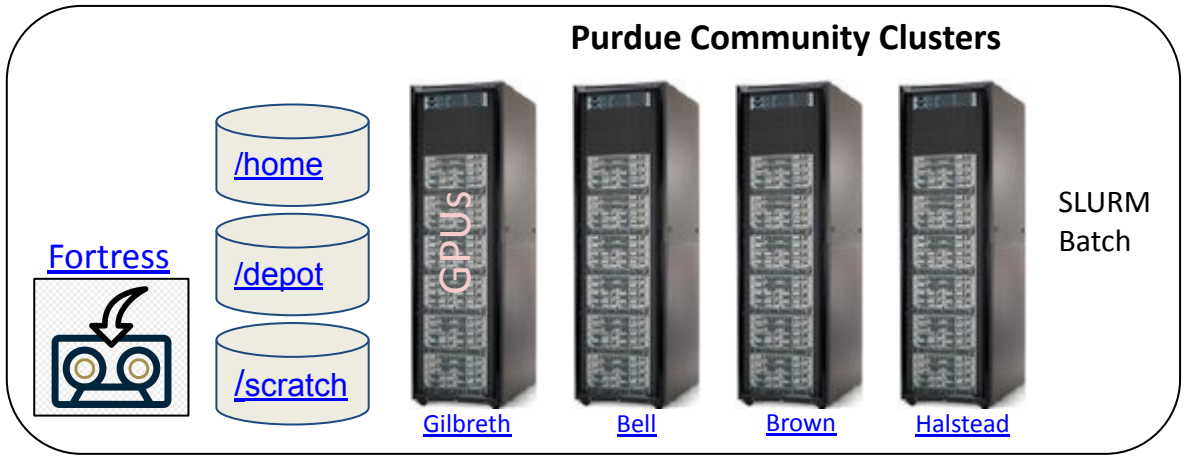
Stefan Piperov

spiperov@purdue.edu

PURDUE UNIVERSITY | Research Computing INFORMATION TECHNOLOGY

PURDUE T2 Overview





PURDUE UNIVERSITY

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

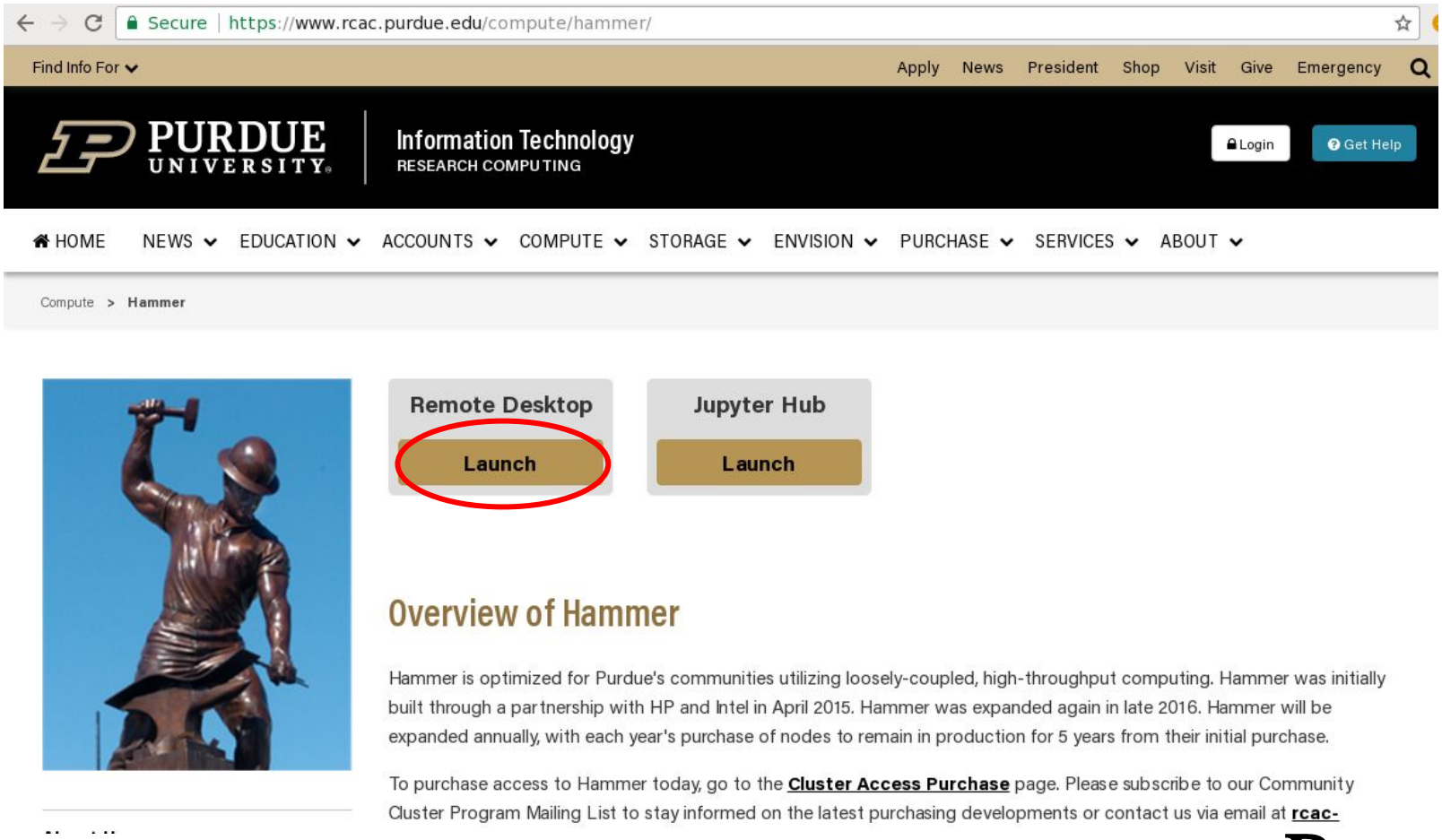
Retiring!

  - **New CMS Front-Ends:**
    - ssh <username>@cms.rcac.purdue.edu
      - load-balanced, GPU-enabled, High-Memory (1TB)

  - **Other Comunity 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

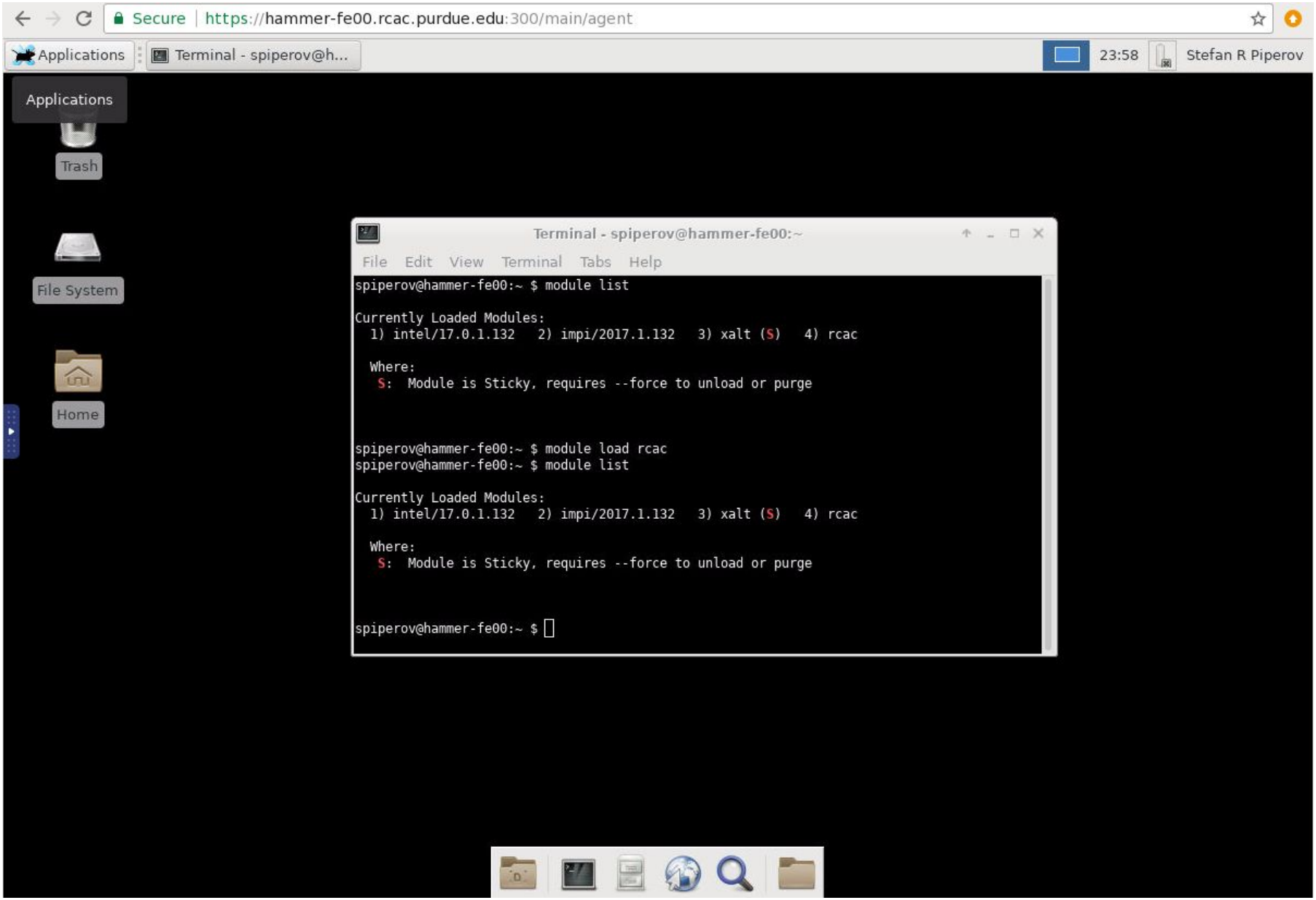
Interactive work - GUI

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



PURDUE UNIVERSITY

Interactive work



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 exnvironments

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
root://cmsxrootd.fnal.gov//store/relval/CMSSW_10_6_4/RelValZMM_13/MINIAODSIM/PUpmx25ns_106X_upgrade201
8_realistic_v9-v1/10000/EC35B5C1-A0A7-574F-8D7A-FCB4C3FBECBE.root ./

$ xrdcp -r ...

$ gfal-copy
gsiftp://cms-gridftp.rcac.purdue.edu//store/relval/CMSSW_10_6_1/RelValZEE_13/MINIAODSIM/PU25ns_106X_mc
2017_realistic_forECAL_v6_HS-v3/20000/83E6A167-6CD2-BF48-8937-AC79791ACC72.root
file:///home/spiperov/DAS2020/CMSSW_10_6_4/src/


$ gfal-copy -r
gsiftp://cms-gridftp.rcac.purdue.edu//store/relval/CMSSW_10_6_1/RelValZEE_13/MINIAODSIM/PU25ns_106X_mc
2017_realistic_forECAL_v6_HS-v3  file:///home/spiperov/DAS2020/CMSSW_10_6_4/src/
```

Note: redirector - no need to know the exact location!

*`xrdcp` can also do **recursion now**!*

Note: Precise locations needed!

*can do **recursion**!*

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/el7-x86_64/setup.sh
$ source /cvmfs/oasis.opensciencegrid.org/osg-software/osg-wn-client/3.4/current/el6-x86_64/setup.sh
```


PURDUE UNIVERSITY

Working with Datasets

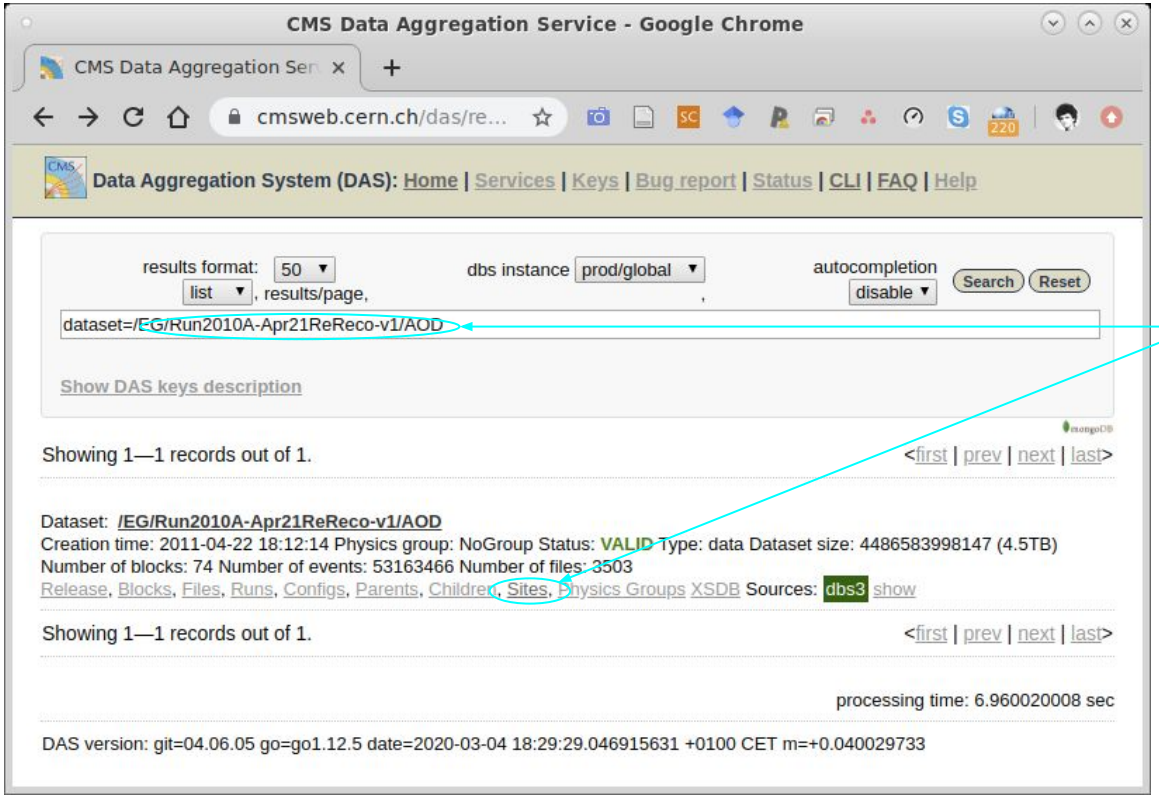
- Main navigational tool - Data Aggregation System (DAS) at CERN (https://cmsweb.cern.ch/das/)
  - Universal and (somewhat) intuitive
  - Slow!
- Command-line tool - dasgoclient
  - Much faster
  - Very flexible, when combined with other UNIX-shell tools

```
$ 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?**



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
```


PURDUE UNIVERSITY

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",
   fileNames = 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/NANOAOD \
cms:/SingleMuon/Run2018B-02Apr2020-v1/NANOAOD   #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/NANOAOD #add one more DS
$ rucio detach user.piperov:/Analyses/Hmumu2020/USER cms:/SingleMuon/Run2018A-02Apr2020-v1/NANOAOD #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/NANOAOD | 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!
```


PURDUE UNIVERSITY

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

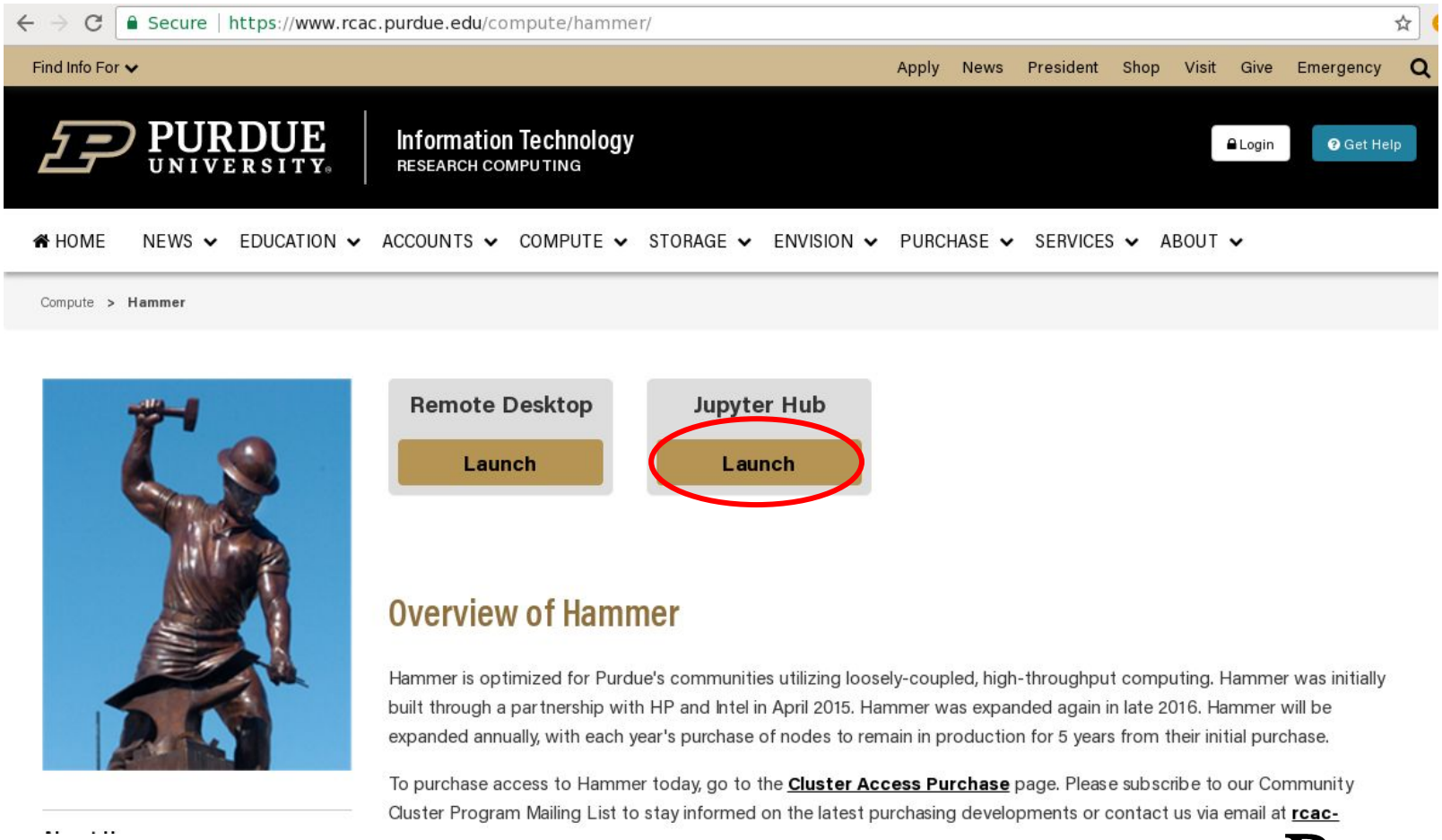
- Once your jobs are finished, where to store the output files?
  - NOT in `/tmp` or `/scratch` - for sure! That's only for temporary storage
    - those get cleaned - frequently
  - Home directory - probably too small
  - Data Depot - for R/W access and sharing with the group
  - HDFS - best for long-term 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

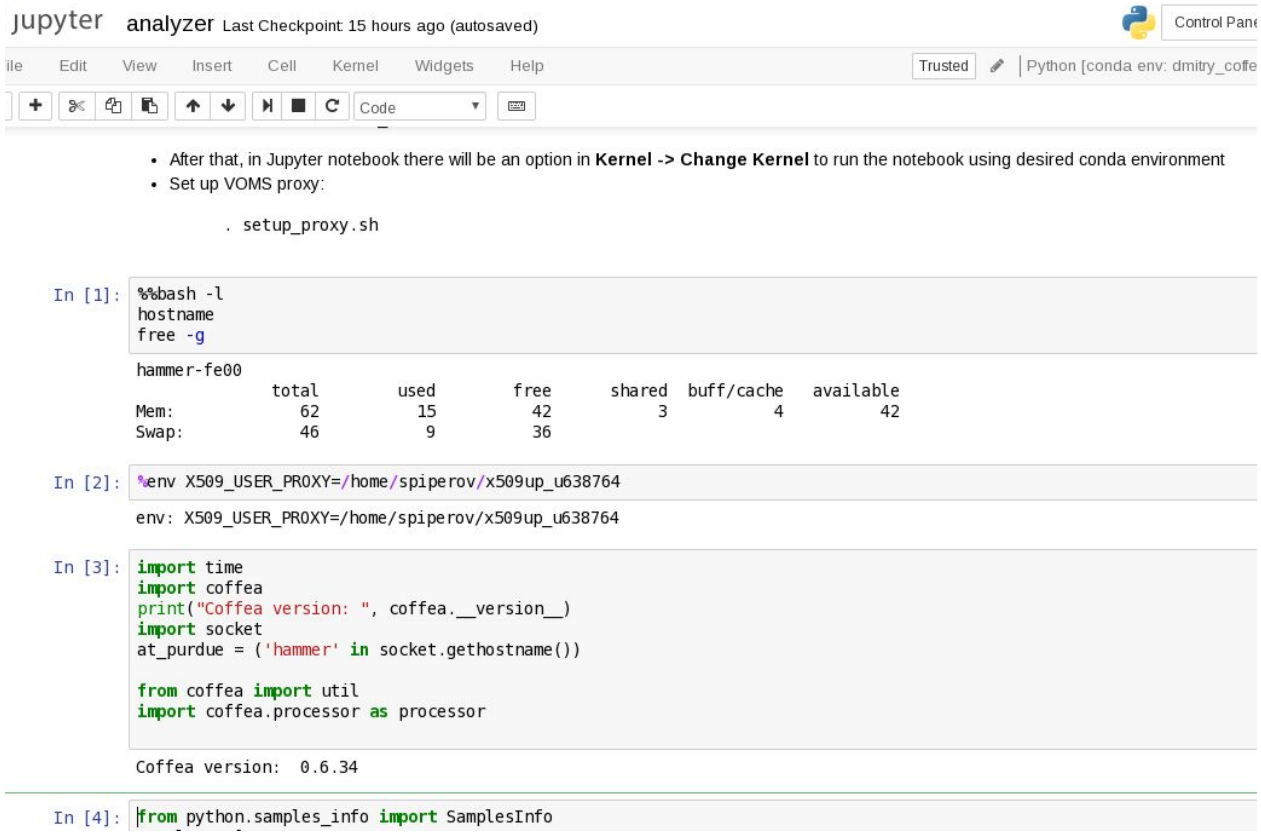
- **Jupyter Hubs on Community Clusters**
  - **https://www.rcac.purdue.edu/compute/hammer/**



PURDUE UNIVERSITY

Jupyter Notebooks

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



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

PURDUE UNIVERSITY

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	Columnar Analysis Tools
Containers	Effective Scale Out Techniques
Data and MC Processing	Uproot and Awkward Array for columnar analysis
Machine Learning	Trigger Tau MET Muon Electron and Photon
Visualization	Jet Algorithms and Substructure (Jets I)
Jupyter and pyROOT	Jet Energy Corrections and Pile-Up Mitigation (Jets II)
Condor/CMSConnect/CRAB3	(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

CMS Cluster (still) in MATH

PURDUE UNIVERSITY

(Backup Slides)

- Backup slides

pyROOT in Jupyter

Other GPU-enabled machines:

- **lxplus-gpu.cern.ch** - has mostly Nvidia T4's
- **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.
```