Python based Data Science on Cray Platforms

Rob Vesse, Alex Heye, Mike Ringenburg - Cray Inc

Overview

- Supported Technologies
  - Cray PE Python Support
  - Shifter
  - Urika-XC
    - Anaconda Python
    - Spark
      - Intel BigDL machine learning library
    - Dask Distributed
    - TensorFlow
    - Jupyter Notebooks
- Use Cases
  - Met Office JADE Platform
  - Nowcasting

Supported Technologies Overview

- Python in Cray PE
- Shifter
  - Containers for XC
- Urika-XC - Analytics for Cray XC systems
  - Anaconda
  - Apache Spark
  - Dask Distributed
  - Tensorflow
  - Jupyter Notebooks

Python in Cray PE

```
> module load cray-python
> python example.py

> echo "Coming December…”
> module load cray-python/2.7.13.1.102
> python2 example2.py
> module load cray-python/3.6.1.1.102
> python3 example3.py
```

- Currently single module bundling Python 2 and 3
  - Stock Python distributions
  - Version numbering matches PE versions
- From December release separate Python 2 and 3 modules
  - Exact version numbers may change in release but will correspond to Python versions
  - Have corresponding python2 and python3 commands
  - Both contain a python command, most recently loaded module takes precedence
  - module load cray-python without explicit version will use Python 2
- Built against relevant Cray libraries (libsci and mpt)
- Includes common data science libraries:
  - numpy, scipy, mpi4py and dask

Shifter

- Container support for HPC
  - Originally developed at NERSC
  - Officially supported by Cray since CLE 6.0UP02
- Containers allow for encapsulating applications and their dependencies
- Supports common image repositories and formats
  - e.g. Dockerfile and Docker repositories
- Key features for HPC environments
  - Provides better security model than Docker
    - Users run as themselves inside the container
    - System admin can control mount points
  - Enables MPI and GPU support for containerized applications
  - Loopback Cache feature avoids metadata overheads

Urika-XC

- Provides a suite of analytics and data science software that runs on the XC platform
- Requirements:
  - CLE 6.0UP02
  - Shifter
- Provided as a module
  - module load analytics
- Dynamically creates analytics clusters in the context of an existing WLM reservation
  - Spark and/or Dask clusters
  - Added in 1.1 release:
    - Google TensorFlow and Intel BigDL for machine learning
    - Jupyter Notebooks for interactive development

```
module load analytics
salloc -N 10 start_analytics
spark-shell
```

Urika-XC - Python Dependency Management

```
> conda create -n example
> source activate example
(example) > conda install pandas
(example) > python --version
Python 3.6.2 :: Continuum Analytics, Inc.
> source deactivate example
> conda create -n py2 python=2.6
> source activate py2
(py2) > python --version
Python 2.6.9 :: Continuum Analytics, Inc.
```

- For analytics products we use Anaconda to manage dependencies
  - Provides users the ability to manage isolated Python environments with their desired package and Python versions
- Environments can be consumed by relevant applications e.g. Spark, Dask etc.
- As of 1.1 release:
  - Can optionally use Intel Distribution of Python if preferred
  - Add --idp flag to start_analytics

Urika-XC - Apache Spark

- Popular in-memory analytics package
  - http://spark.apache.org
  - Currently version 2.1.0
  - 1.1 release will upgrade to 2.2.0
- Always run as part of Urika-XC jobs
- Supports jobs written in multiple languages
  - Scala/Java
  - Python
  - R
- Interactive Scala, Python and R shells available
  - Can also batch submit
- Integrates with Anaconda environments for dependency management
- Also includes the Intel BigDL machine learning libraries
  - Currently version 0.3.0

Urika-XC - Dask Distributed Python

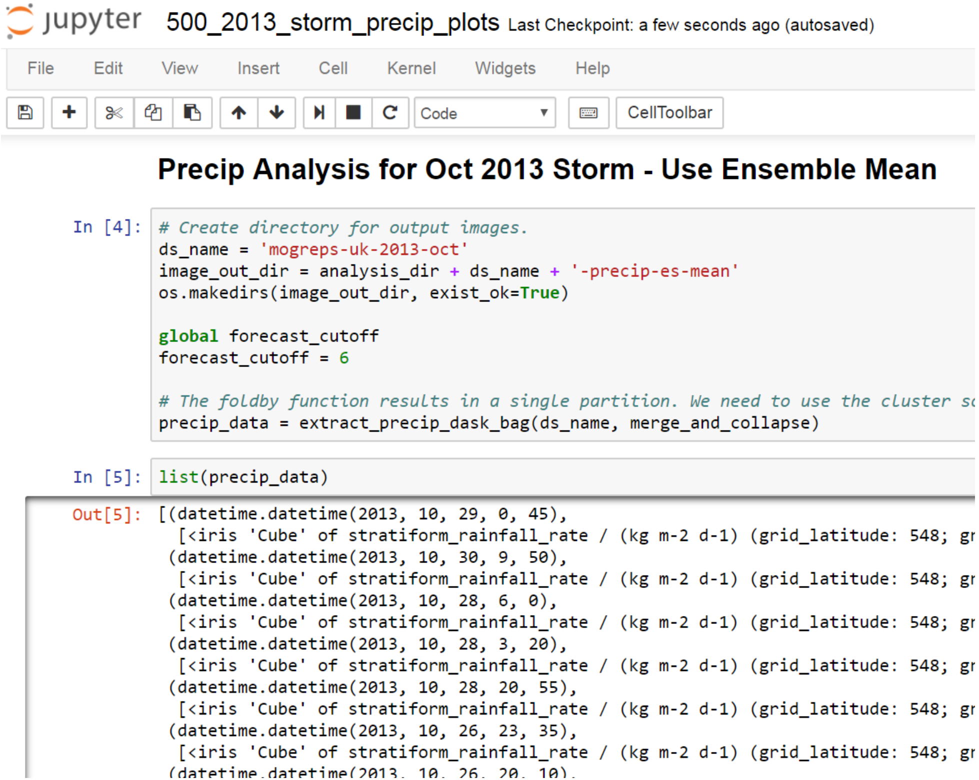
```
> salloc -N 10 start_analytics --dask --dask-env example --dask-workers 8 --dask-cores 2
```

- Popular distributed Python analytics package
  - https://distributed.readthedocs.io/en/latest/
- Optionally runs as part of Urika-XC jobs
  - Co-exists with Spark cluster
- Integrates with Anaconda for dependency management

Urika-XC - TensorFlow

- Popular machine learning framework
  - Originally developed at Google, now open source
  - Version 1.3
  - New feature in our 1.1 release
- Machine learning workflows can be written in Python
- Supports two modes of distribution
  - Google gRPC (TCP/IP)
  - Cray MPI (via Cray developed plugin)
- Supports both CPU and GPU nodes
- Our analytics module provides several helper scripts for launching distributed jobs

Urika-XC - Jupyter Notebooks



jupyter 500_2013_storm_precip_plots Last Checkpoint: a few seconds ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Code CellToolbar

Precip Analysis for Oct 2013 Storm - Use Ensemble Mean

In [4]:
```
# Create directory for output images.
ds_name = 'mogreps-uk-2013-oct'
image_out_dir = analysis_dir + ds_name + '-precip-es-mean'
os.makedirs(image_out_dir, exist_ok=True)

global forecast_cutoff
forecast_cutoff = 6

# The foldby function results in a single partition. We need to use the cluster sc
precip_data = extract_precip_dask_bag(ds_name, merge_and_collapse)
```

In [5]:
```
list(precip_data)
```

Out[5]:
```
[(datetime.datetime(2013, 10, 29, 0, 45),
  [<iris 'Cube' of stratiform_rainfall_rate / (kg m-2 d-1) (grid_latitude: 548; gr
 (datetime.datetime(2013, 10, 30, 9, 50),
  [<iris 'Cube' of stratiform_rainfall_rate / (kg m-2 d-1) (grid_latitude: 548; gr
 (datetime.datetime(2013, 10, 28, 6, 0),
  [<iris 'Cube' of stratiform_rainfall_rate / (kg m-2 d-1) (grid_latitude: 548; gr
 (datetime.datetime(2013, 10, 28, 3, 20),
  [<iris 'Cube' of stratiform_rainfall_rate / (kg m-2 d-1) (grid_latitude: 548; gr
 (datetime.datetime(2013, 10, 28, 20, 55),
  [<iris 'Cube' of stratiform_rainfall_rate / (kg m-2 d-1) (grid_latitude: 548; gr
 (datetime.datetime(2013, 10, 26, 23, 35),
  [<iris 'Cube' of stratiform_rainfall_rate / (kg m-2 d-1) (grid_latitude: 548; gr
 (datetime.datetime(2013, 10, 26, 20, 10),
```

- Provides UIs for data scientists to develop analytics workflows in
  - Mix code, prose, images etc in a single document
  - Can be shared for collaboration
- Offloads execution to underlying analytics framework
  - Spark/Dask/TensorFlow/BigDL etc
- SSH tunnels used to expose notebook server to outside world i.e. user laptops

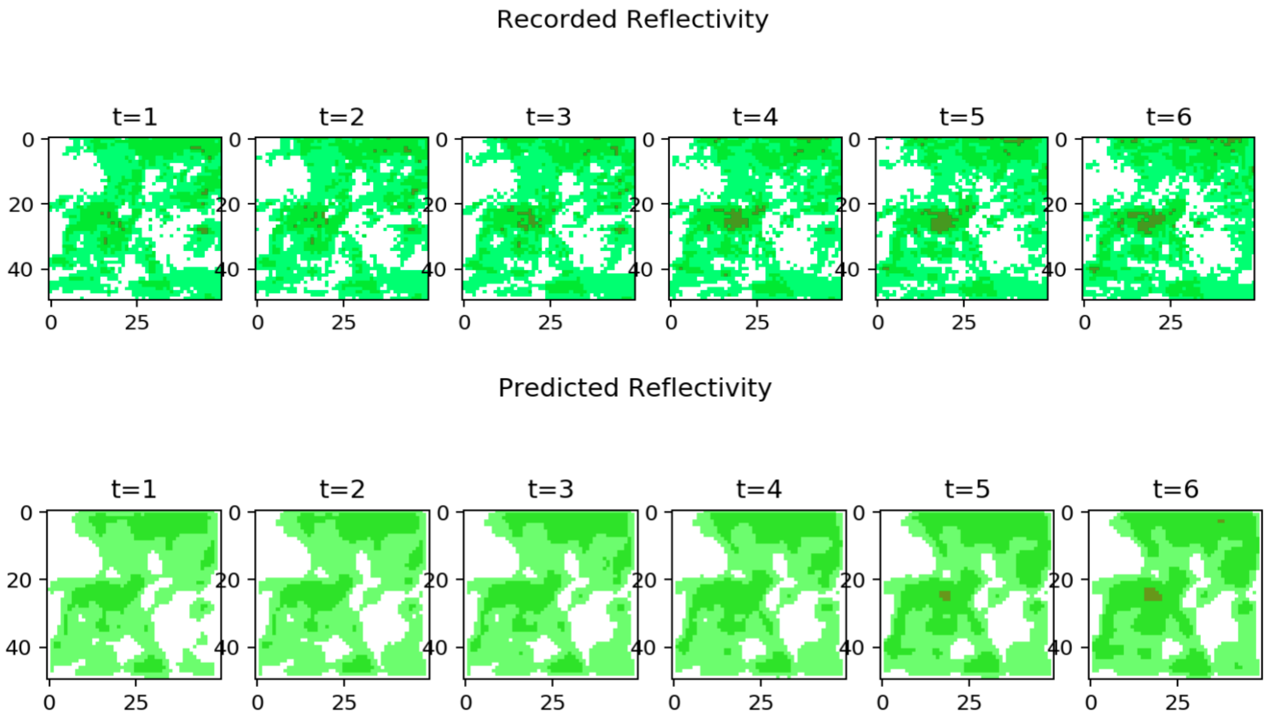
Use Cases - Met Office JADE Platform

- Met Office “JADE” Data Analysis platform
  - Collaboration with Met Office Informatics Lab
  - Goal to replace powerful desktops with analysis environment accessed via web browser
  - Leverages :
    - DASK distributed python engine
    - Jupyter interactive notebooks
    - IRIS Python Library for Meteorology and Climatology
  - Developed/prototyped on AWS
- Will be able to run on their XC systems once Urika-XC 1.1 is released

Use Cases - Nowcasting

- Create very short term forecasts on demand
- Goals:
  - Investigate the utility of machine learning for very short term (0-1 hour) precipitation forecasts
  - Gain insights into the full deep learning workflow to drive product roadmap
- Approach
  - Machine learning used to train a convolutional neural network on historical observational data for a region
  - Once trained model can be used to generate a very short-term forecast based upon current/past observations
    - Past observations useful as we can compare predictions with subsequent observations for evaluation
    - i.e. Train once, Predict many
  - Python implementation of workflow encapsulated in a Jupyter Notebook

Use Case - Nowcasting - Sample Results



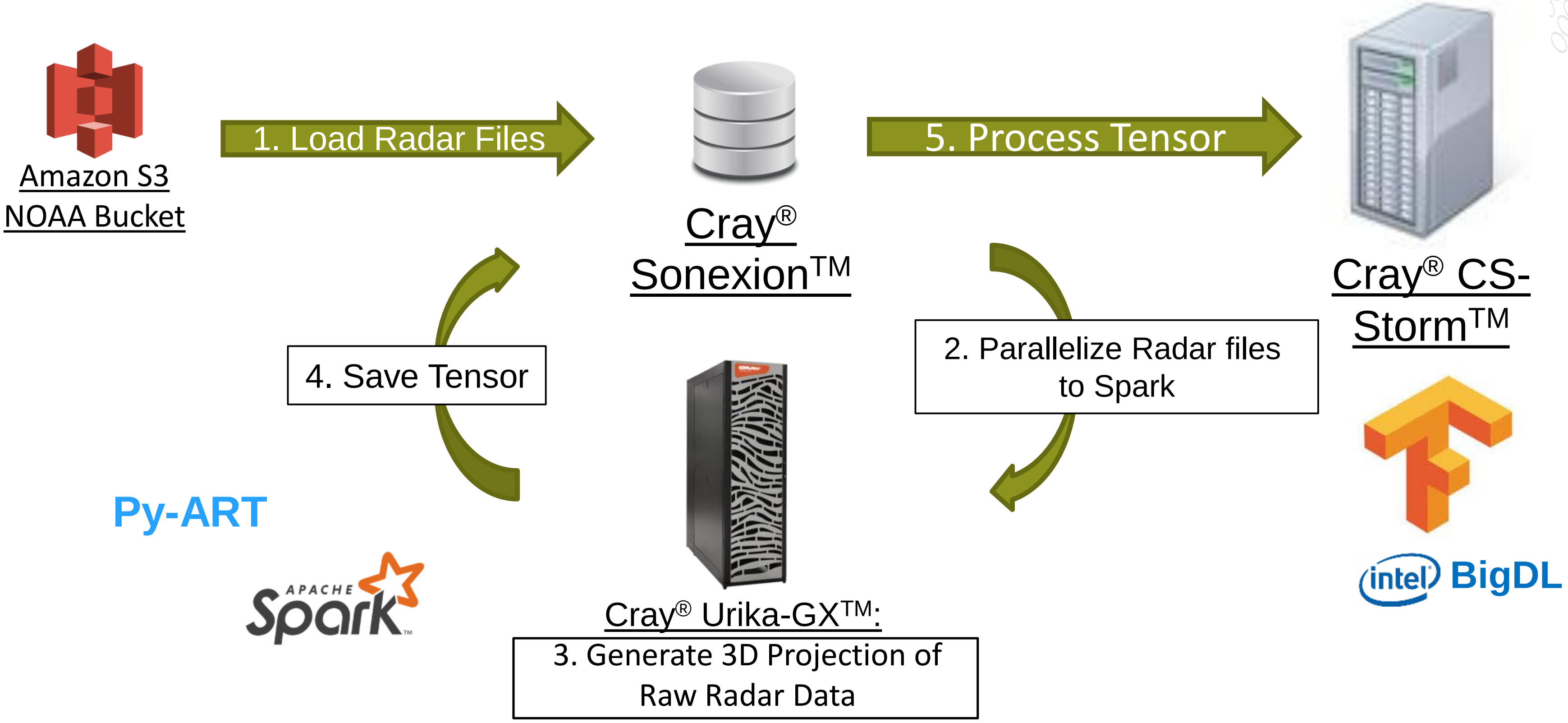
Questions?

rvesse@cray.com
mikeri@cray.com
aheye@cray.com

Nowcasting Case Study

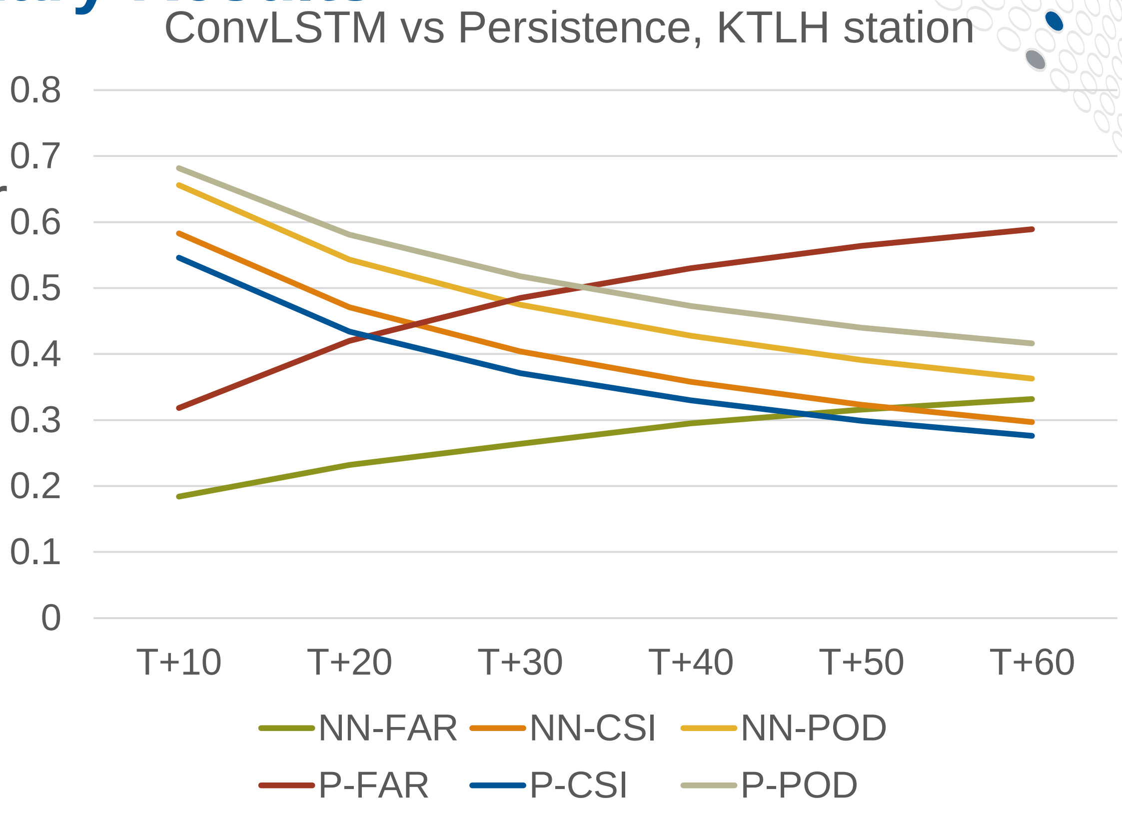
Additional Details and Results

Nowcasting Data Pipeline



Nowcasting Preliminary Results

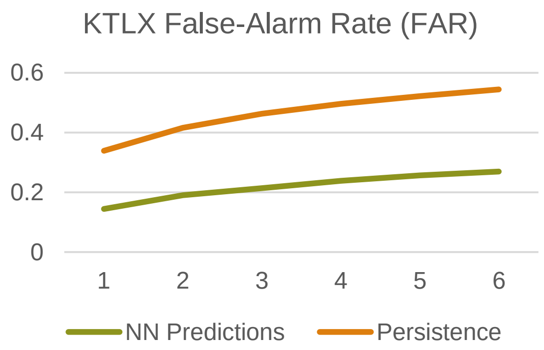
- FAR: False Alarm Rate – lower is better
- CSI: Critical Success Index – higher is better
- POD: Probability of Detection – higher is better

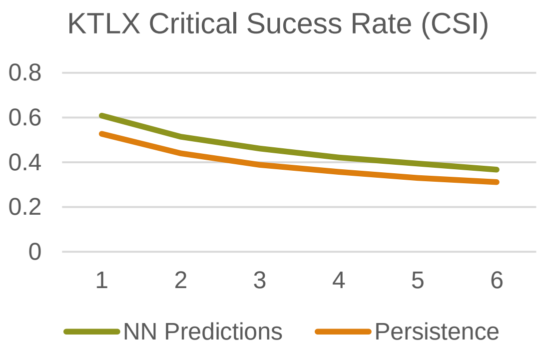


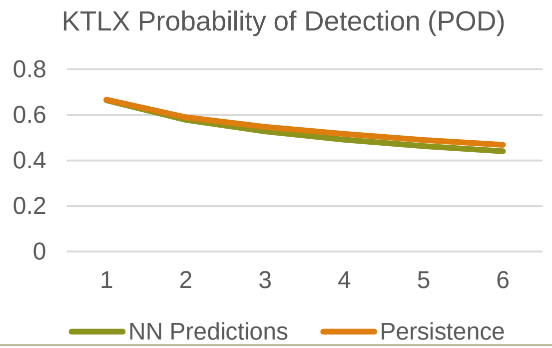
NN = ConvLSTM P = Persistence

Nowcasting - Further Results

- **Station: KTLX, Oklahoma City**
- **1 Timestep = 10 minutes**
- **Blue: Predictions made by DL**
- **Red: Constant Prediction of Persistence**







Nowcasting - Further Results