




COMPUTE



STORE



ANALYZE

The background of the slide is a dramatic, high-contrast image of a storm. Dark, heavy clouds are illuminated from below, creating a bright, glowing effect. Several bright white lightning bolts are visible, striking downwards from the clouds. The overall color palette is dominated by dark blues, greys, and bright whites, with a hint of yellow/orange at the bottom right corner.

Emerging Data Analysis Technologies for the Earth Sciences

iCAS 2017

Dr. Phil Brown

Earth Sciences Segment Leader

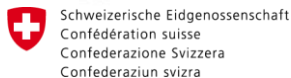
Topics

- **Brief update on Cray in the Earth Sciences**
- **Emerging Data Analysis technologies**
 - Enabling new data analysis environments
 - New analysis techniques

Cray Growth in Weather, Climate and Oceanography



Australian Government
Bureau of Meteorology



FINNISH METEOROLOGICAL INSTITUTE



Global NWP Centre



Global NWP Centre



COMPUTE

STORE

ANALYZE

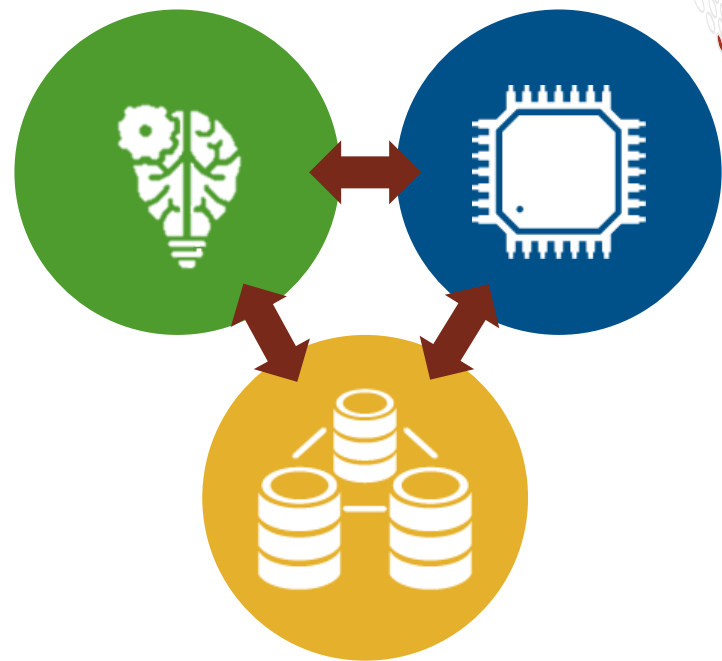
Flexible, scalable analysis environments

- **Challenges:**

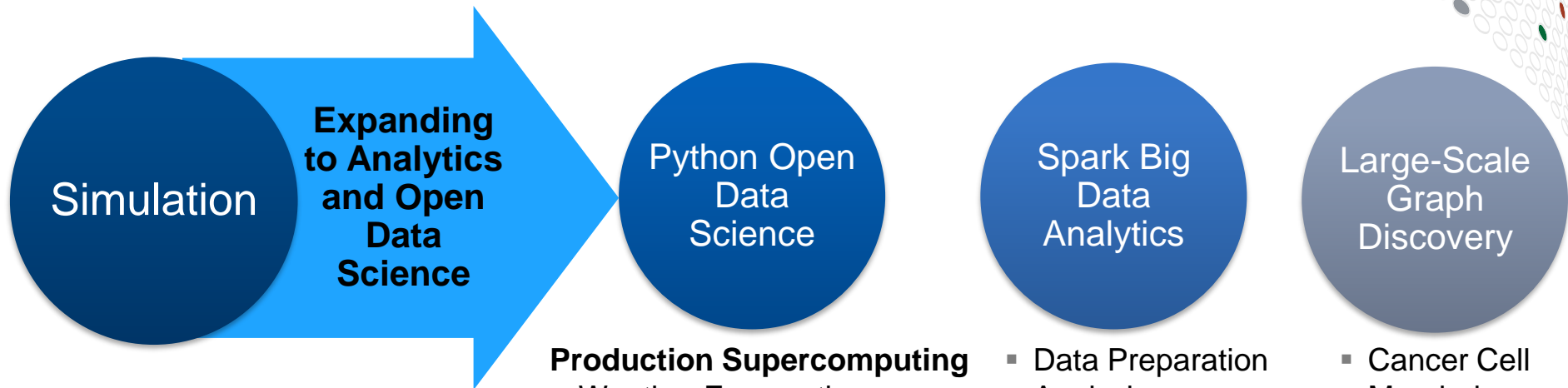
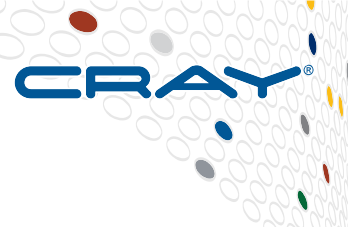
- Data Gravity
- Portability
- Ease of Use

- **Solution:**

- Enable scalable, flexible, user-friendly data analysis leveraging existing infrastructure



Urika-XC make Analytics and Graph “first class” citizens on XC Series



Simulation

Expanding to Analytics and Open Data Science

Python Open Data Science

Spark Big Data Analytics

Large-Scale Graph Discovery



Production Supercomputing

- Weather Forecasting
- Seismic Imaging
- Manufacturing CAE

Scientific Supercomputing

- Climate Science
- Chemistry & Materials Science

- Data Preparation
- Analysis
- Visualization
- Machine Learning
- Deep Learning

- Cancer Cell Morphology
- Fraud and Insider Threat Detection

COMPUTE

STORE

ANALYZE



Example

- **Met Office “JADE” Data Analysis platform**
 - Goal to replace powerful desktops with analysis environment accessed via web browser
 - Leverages :
 - DASK parallel python engine
 - Jupyter interactive notebooks
 - IRIS Python Library for Meteorology and Climatology
 - Developed/prototyped on AWS
- **Desirable to support on Cray XC supercomputers**
 - First landing-point for data, access to highest I/O performance
- **Leverage Urika-XC software**
 - Support for Python, DASK, Jupyter, integrated with XC scheduler
 - Containerized for easy, repeatable deployment
- **Collaboration with Met Office Informatics Lab**

Example Use-cases

- **Persist recent ensemble data to allow interactive query/analysis**
 - 50GB from Met Office Global & Regional Ensemble Prediction System
 - Lazily load data of interest
- **Examine Wind-Speeds**
 - Select 10m wind speeds (x + y) for each member
 - Calculate real wind speed
 - Average across ensemble members
 - Plot winds for forecast timesteps
 - Create animation
- **Compare ensemble mean to single member**
- **Repeat for precipitation/rainfall**

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 so create a new bag.
precip_data = extract_precip_dask_bag(ds_name, merge_and_collapse)
```

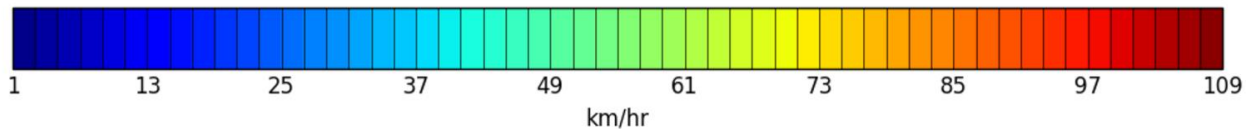
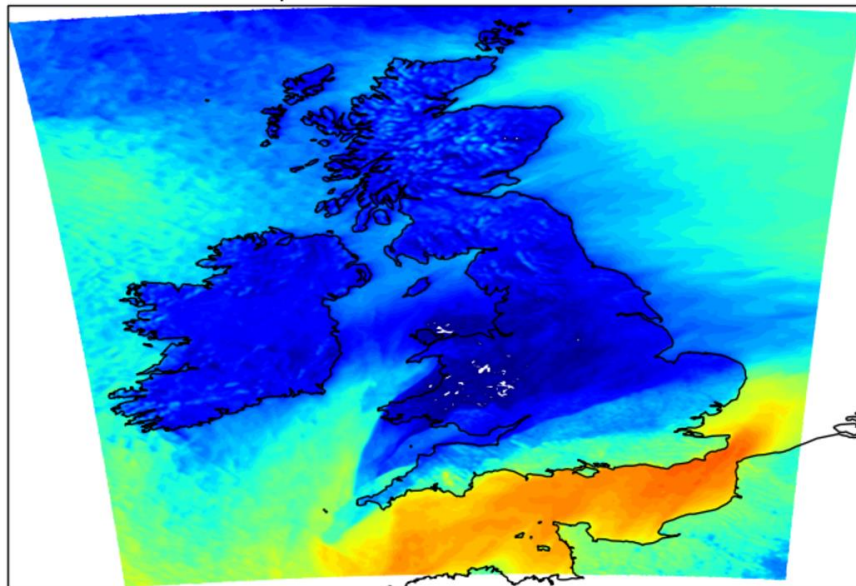
```
In [5]: list(precip_data)
```

```
Out[5]: [(datetime.datetime(2013, 10, 29, 0, 45),
          [<iris 'Cube' of stratiform_rainfall_rate / (kg m-2 d-1) (grid_latitude: 548; grid_longitude: 421)>]),
         (datetime.datetime(2013, 10, 30, 9, 50),
          [<iris 'Cube' of stratiform_rainfall_rate / (kg m-2 d-1) (grid_latitude: 548; grid_longitude: 421)>]),
         (datetime.datetime(2013, 10, 28, 6, 0),
          [<iris 'Cube' of stratiform_rainfall_rate / (kg m-2 d-1) (grid_latitude: 548; grid_longitude: 421)>]),
         (datetime.datetime(2013, 10, 28, 3, 20),
          [<iris 'Cube' of stratiform_rainfall_rate / (kg m-2 d-1) (grid_latitude: 548; grid_longitude: 421)>]),
         (datetime.datetime(2013, 10, 28, 20, 55),
          [<iris 'Cube' of stratiform_rainfall_rate / (kg m-2 d-1) (grid_latitude: 548; grid_longitude: 421)>]),
         (datetime.datetime(2013, 10, 26, 23, 35),
          [<iris 'Cube' of stratiform_rainfall_rate / (kg m-2 d-1) (grid_latitude: 548; grid_longitude: 421)>]),
         (datetime.datetime(2013, 10, 26, 20, 10),
```


St Jude Storm



Windspeed at: 2013-10-28 02:00:00

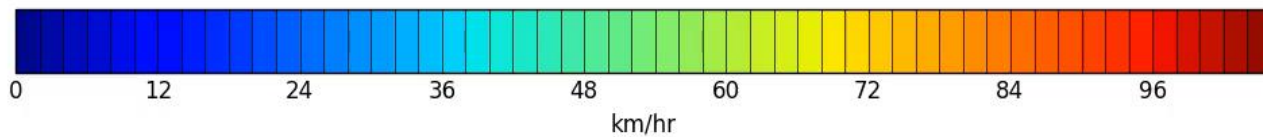
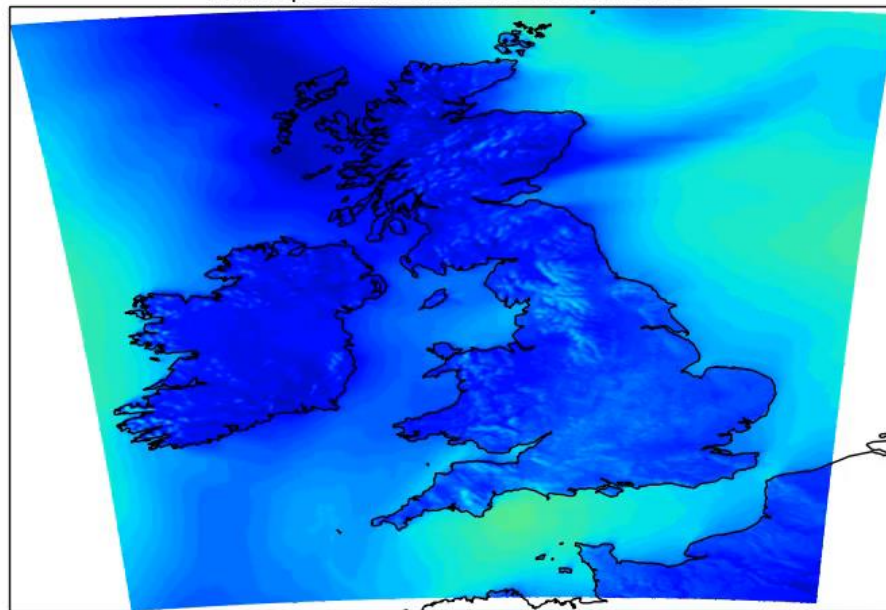


COMPUTE

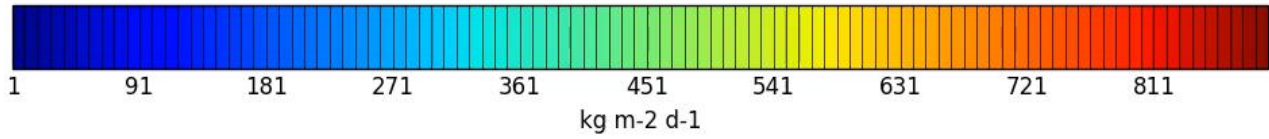
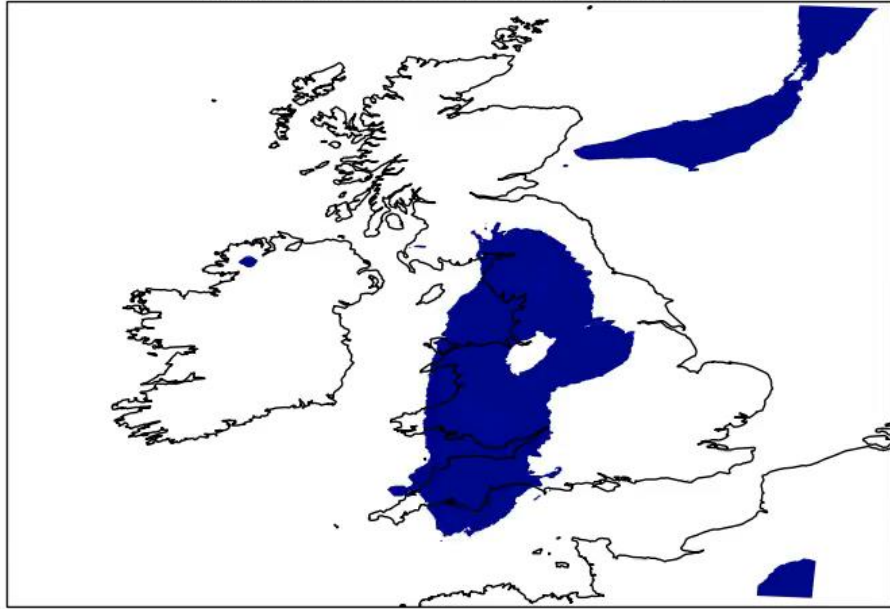
STORE

ANALYZE

Windspeed at: 2013-10-26 04:00:00



Rainfall rate at: 2013-10-26 03:10:00



Opportunities for Machine/Deep Learning in Weather/Climate

- **Almost the opposite of a physics/dynamics based model**
 - Arduous to train, but comparatively quick to run
 - Data Consumer vs Data Producer
- **Use-cases likely to be complementary**
- **Some ideas:**
 - Rapid classifiers or predictors for radar/observations
 - Pattern recognition in model outputs
 - Infilling/smoothing model outputs
 - Replacements for expensive parameterizations

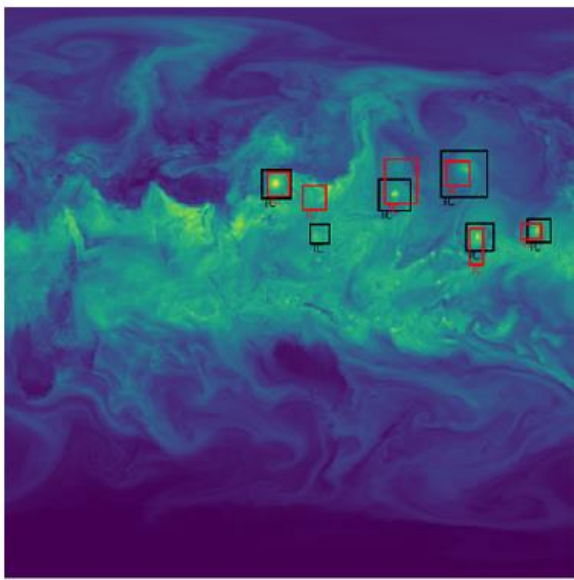


Figure 6: Results from plotting the network's most confident (>95%) box predictions on an image for integrated water vapor (TMQ) from the test set for the climate problem.



August 28, 2017

A collaborative effort between Intel, NERSC and Stanford has delivered the first 15-petaflops deep learning software running on HPC platforms and is, according to the authors of the paper (and to the best of their knowledge), currently the most scalable deep-learning implementation in the world. The work described in the paper, *Deep Learning at 15PF: Supervised and Semi-Supervised Classification for Scientific Data*¹, reported that a Cray XC40 system with a configuration of 9,600 self-hosted 1.4GHz Intel Xeon Phi Processor 7250 based nodes achieved a peak rate between 11.73 and 15.07 petaflops (single-precision) and an average sustained performance of 11.41 to 13.47 petaflops when training on physics and climate based data sets using Lawrence Berkeley National Laboratory's (Berkeley Lab) NERSC (National Energy Research Scientific Computing Center) Cori Phase-II supercomputer. The group utilized an amalgamation of Intel Caffe, Intel Math Kernel Library (Intel MKL), and Intel *Machine Learning Scaling Library* (Intel MLSL) software to achieve this scalability and performance.²

Sources:

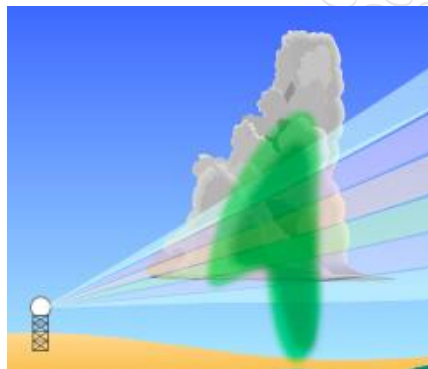
<https://www.hpcwire.com/2017/08/28/nersc-scales-deep-learning15-pflops/>

<https://arxiv.org/abs/1708.05256>

DL-driven Nowcasting

- **Goals:**

- Investigate the utility of Deep Learning for very short term (0-1 hour) precipitation forecasts
- Gain insights into the full deep learning workflow



Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting

Xingjian Shi Zhouong Chen Hao Wang Dit-Yan Yeung
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
{xshiab, zchenbb, hwangaz, dyyeung}@cse.ust.hk

Wai-kin Wong Wang-chun Woo
Hong Kong Observatory
Hong Kong, China
{wkwong, wwoo}@hko.gov.hk

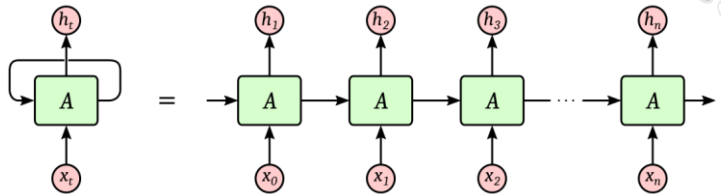
Abstract

The goal of precipitation nowcasting is to predict the future rainfall intensity in a local region over a relatively short period of time. Very few previous studies have examined this crucial and challenging weather forecasting problem from the machine learning perspective. In this paper, we formulate precipitation nowcasting as a spatiotemporal sequence forecasting problem in which both the input and the

Neural Network Architecture

- **Recurrent Neural Network**

- Recurrent connection and mechanism for retaining state.
- Extracts temporal relationships



- **Convolution Long Short-Term Memory (ConvLSTM)**

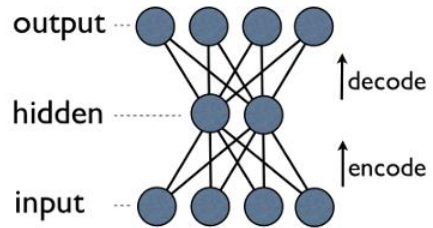
- RNN with defined cell-state representing encoded version of sequential history
- Cell-State is updated through “gating functions” that control information retention, loss and acquisition
- Ability to retain and apply long-term dependencies of a sequence

- **Nowcasting is a sequence to sequence problem**

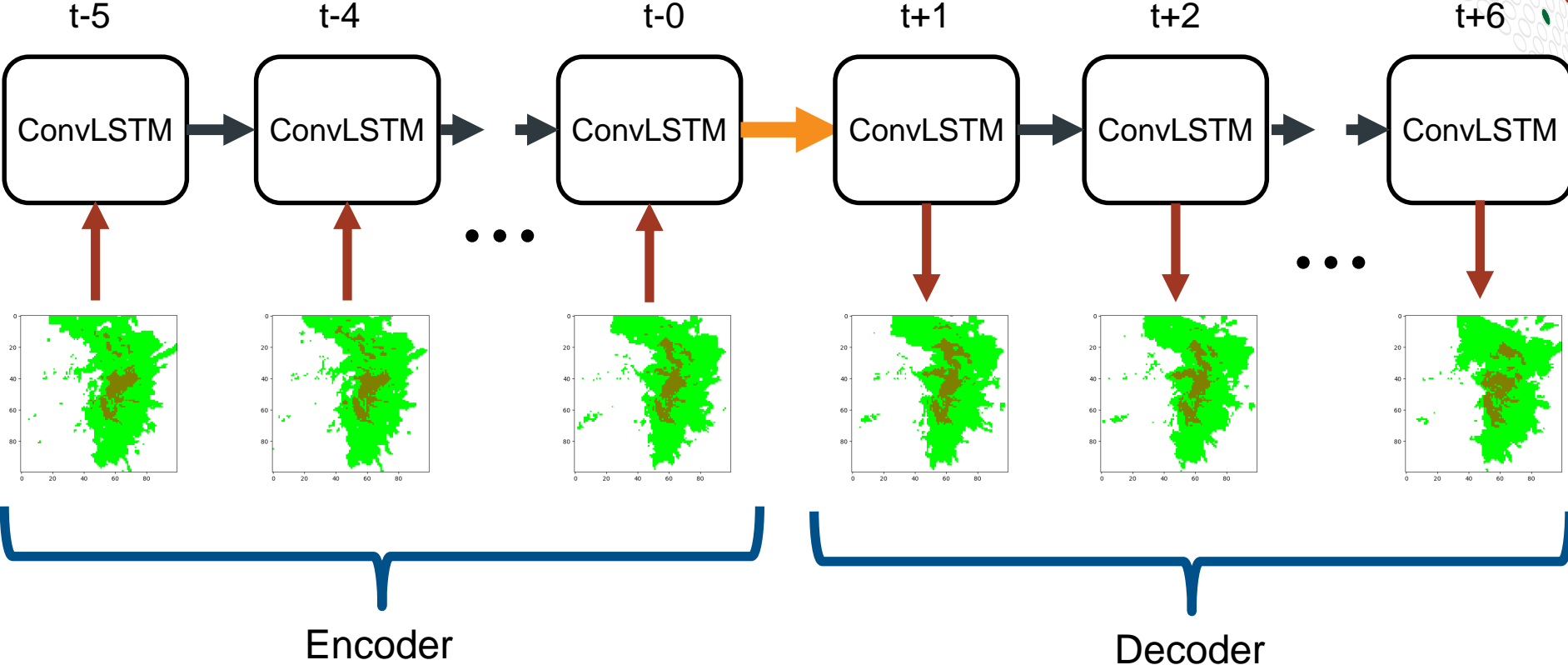
- Input: Sequence of radar images leading up to the current time
- Output: Sequence of predicted radar images arbitrarily far in the future

- **Solution: Encoder-Decoder Networks**

- Encoder digests the input sequence and produces a single tensor representation
- This tensor is used to initialize the decoder
- Decoder takes previous images as input and produces predictions of the next image.



Encoder Decoder using Convolution LSTM



Data Pipeline



Amazon S3
NOAA Bucket

1. Load Radar Files



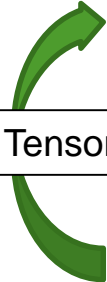
Cray[®]
Sonexion[™]

5. Process Tensor



Cray[®]
CS-Storm[™]

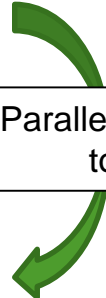
4. Save Tensor



Py-ART



2. Parallelize Radar files
to Spark



Cray[®]
Urika-GX[™]
3. Generate 3D Projection of
Raw Radar Data

COMPUTE

STORE

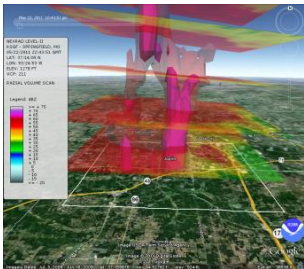
ANALYZE

ETL Pipeline



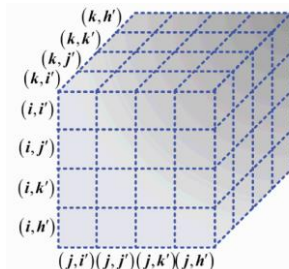
Data Collection

- Historical Radar Data (NETCDF)
- Geographical Region (Eg:- Seattle)
- Days with over 0.1 inches of precipitation, info from NOAA – NCDC
- Radar scans every 5-10 minutes throughout the day



Transformation

- Raw radial data structure converted to evenly spaced Cartesian grid (Tensors with float 32)
- Resolution scaling and clipping
- Configure dimensionality
- Sequencing
- 2 channels – Reflectivity, Velocity
- Uses Py-ART package



Sampling

- Time-series
- Inputs and Labels
- Random sampling

Framework

- Apache Spark on Urika-GX
- Implemented in Jupyter notebooks and Python



Processing Times

Process	Wall Time
Download from S3	13:31:00
Preprocess – 128 cores	07:46:18
Preprocess – 256 cores	04:36:30
Preprocess – 512 cores	03:58:56

- **Hardware: Urika-GX**
- **Software: PySpark, Jupyter Notebooks, PyART**
- **Raw Dataset Size: 938GB**
- **Processed Dataset Size: 101GB**
- **Number of radar files: 111,571**
- **Number of Days: 467**
- **Average Download Speed: 20MB/s**

Training Times

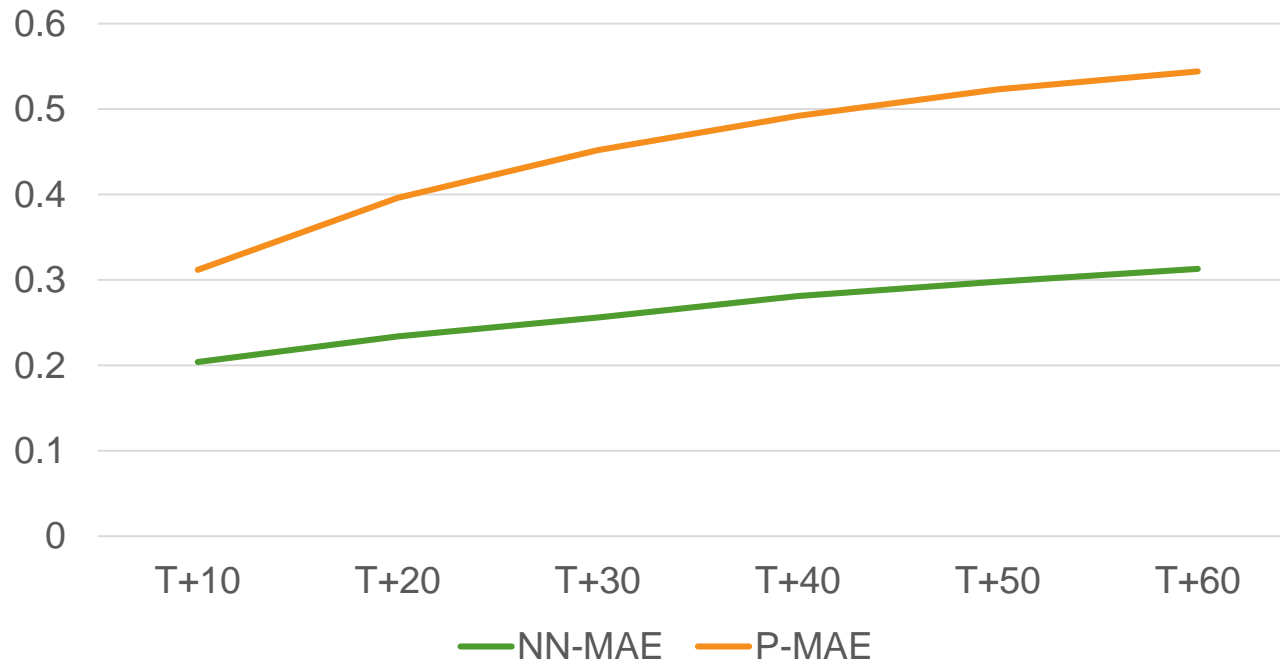
- CS-Storm with NVIDIA Tesla P100 GPUs
- 32 sample per device per batch
- Framework: Tensorflow

GPU Count	Batch-size	Time Per 10 Epochs	Sample /Sec	Final Accuracy (MAE)
1	32	8:47:38	29	0.0192
4	128	2:41:13	88 (3.0x)	0.0185
8	256	1:41:13	142 (4.9x)	0.0185

Early Results – Mean Absolute Error



ConvLSTM vs Persistence, KTLH station



NN = ConvLSTM

P = Persistence

COMPUTE

STORE

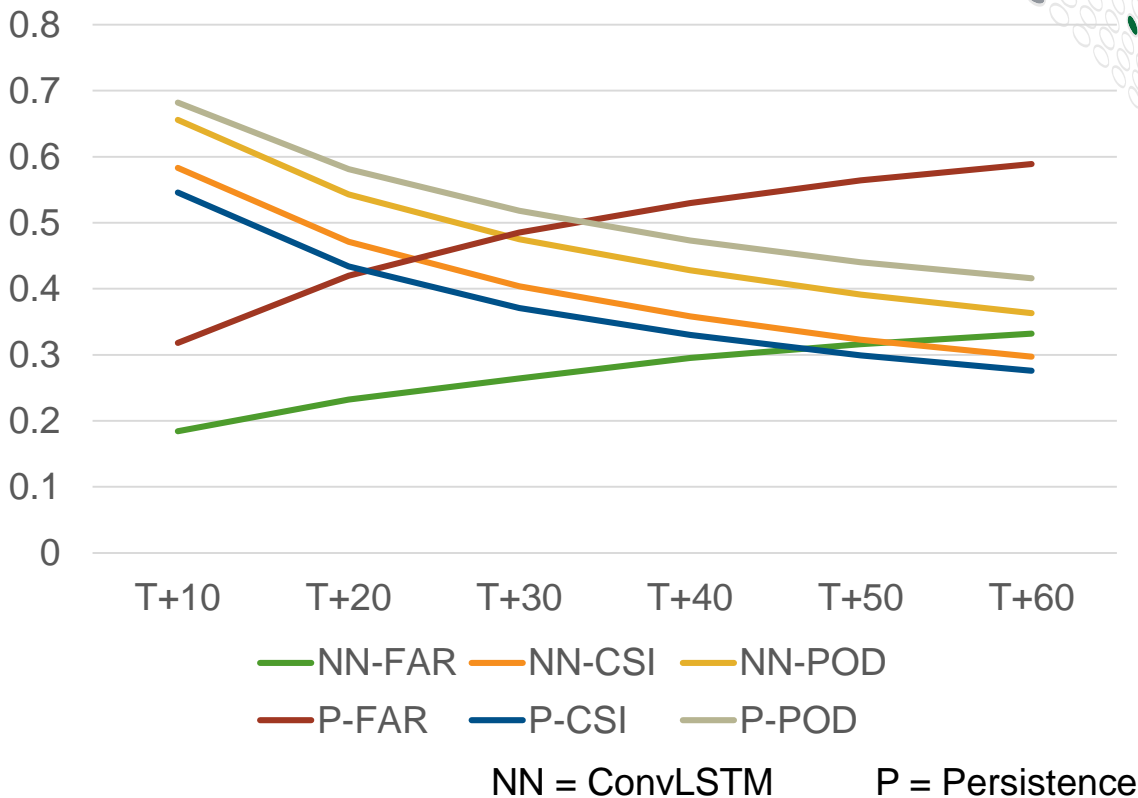
ANALYZE

Early Results – Skill scores

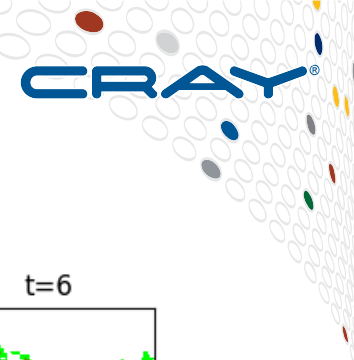


ConvLSTM vs Persistence, KTLH station

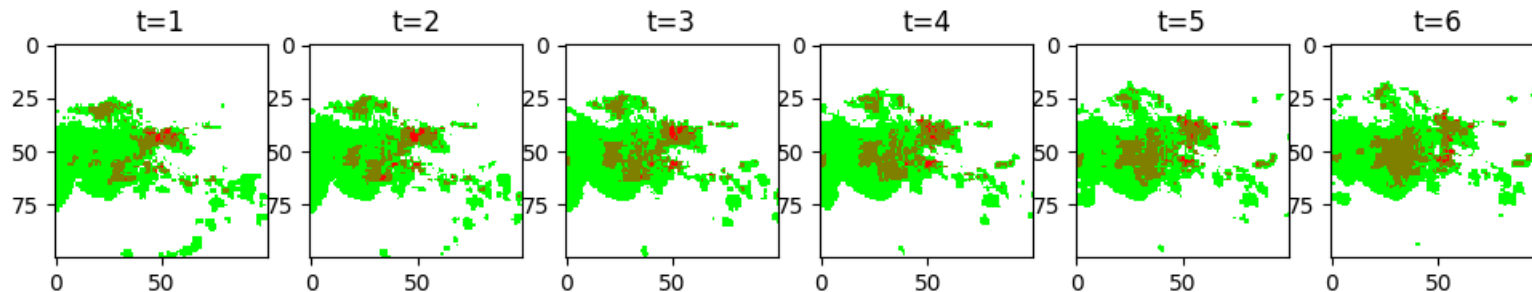
- **FAR: False Alarm Rate – lower is better**
- **CSI: Critical Success Index – higher is better**
- **POD: Probability of Detection – higher is better**



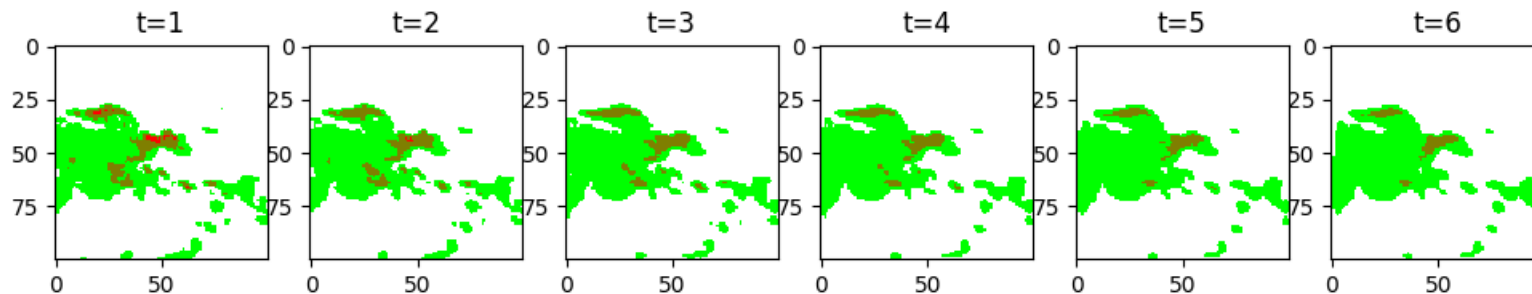
Actual Predictions



Recorded Reflectivity



Predicted Reflectivity

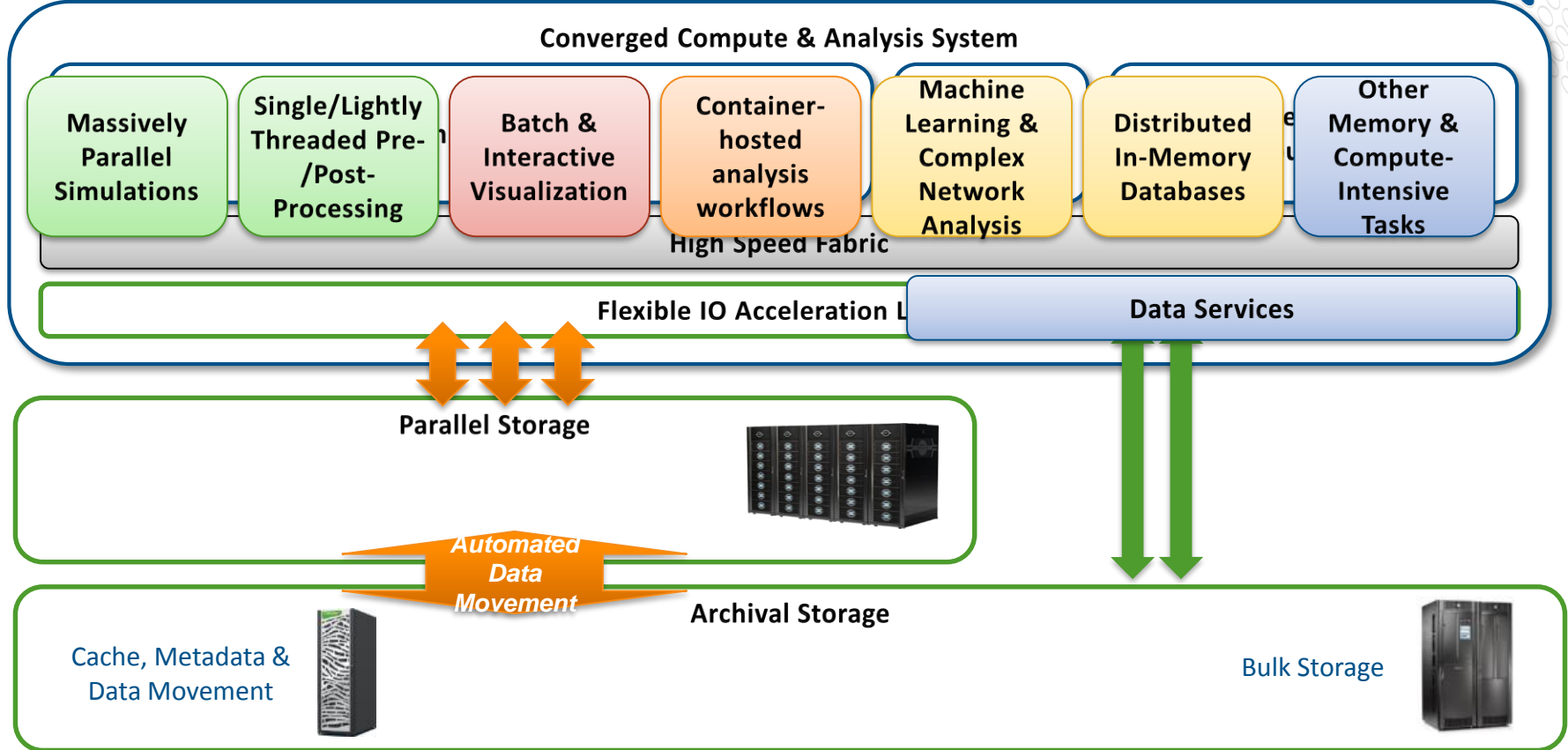


COMPUTE

STORE

ANALYZE

Future Converged Architecture



COMPUTE

STORE

ANALYZE

CRAY



Phil Brown

philipb@cray.com

Thanks for your attention!