HPC and AI convergence

Session IV

Alessandro Rigazzi, Cray Inc.

HPC and AI convergence

- **Why does AI need HPC?**
  - Problem size is getting larger and larger
    - Larger samples, e.g. high-resolution images
    - Larger datasets, e.g. full days of HD camera footage
    - Larger networks, e.g.
      - More layers: Resnet 152, Tiramisu
      - More computational demanding: Densenet
  - Need for speed
    - How fast can you train a network?
    - How often do you want to re-train?

DL challenges

- **Fast training**
  - Training of NNs usually happens in mini-batches
    - The larger the mini-batch, the faster the training
      - If learning rate is correctly set
    - Memory sets a hard upper bound on mini-batch size
      - Especially on GPUs
- **More accurate training**
  - NNs have lots of Hyper-Parameters
    - What is the best setting?
      - Learning rate, momentum, …
      - Number of layers or building blocks

Faster Training at Cray

The Benchmark: KITTI

- **KITTI is a project of Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago**
  - Focusing on autonomous driving (tasks: stereo, optical flow, visual odometry, 3D object detection and 3D tracking)
  - Aims to develop novel challenging real-world computer vision benchmarks

2D Object Detection

- **This track focuses on 2D object detection and object orientation estimation**
- **Dataset**
  - 7481 training images (7000 for training and 481 for validation) – 5.9GB
    - These images are labeled with bounding boxes for objects
    - Relevant object classes are: Car, Pedestrian, Cyclist
    - Any other object can be labeled with DontCare
  - 7518 test images – 6GB
  - A total of 80,256 labeled objects
  - All images are color and saved as png

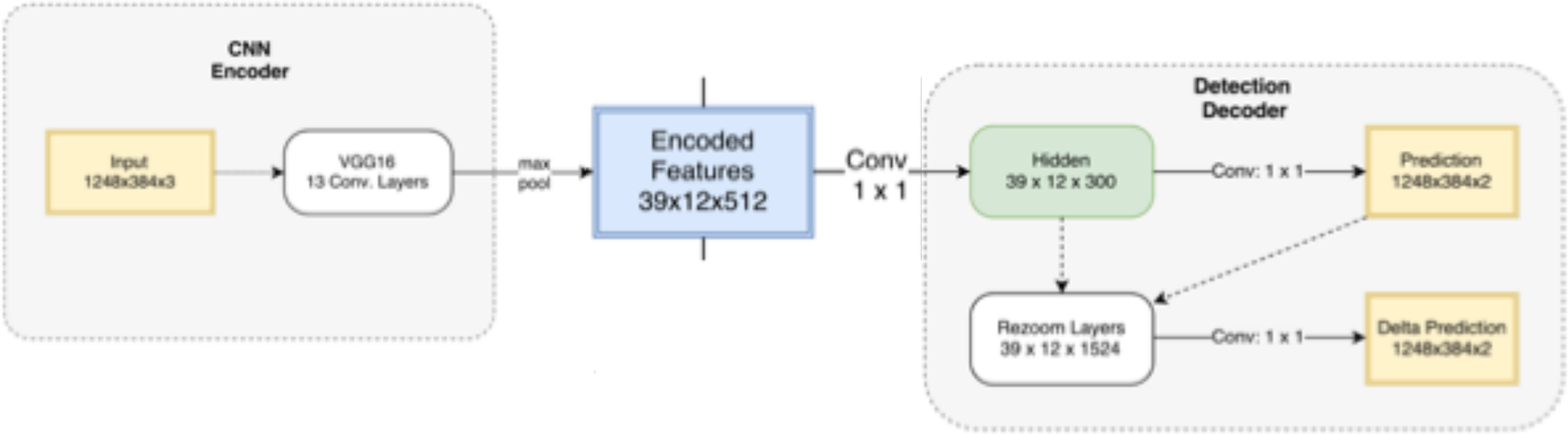
Object Detection Methods

- **Traditional deep learning approaches to object detection follow a two step process**
  - First, region proposals are generated
  - Then scored using a convolutional network
- **Some methods use CNNs for the proposal generation**
- **Recent methods use a single deep NN that is trainable end-to-end to directly perform detection**

- **Frameworks used for training**
  - Caffe, Pytorch, scikit-learn, MatLab, TensorFlow
- **All of them train on a single node (some multi-GPU)**

FastBox – The Network Architecture

- FastBox
  - Based on a encoder-decoder architecture



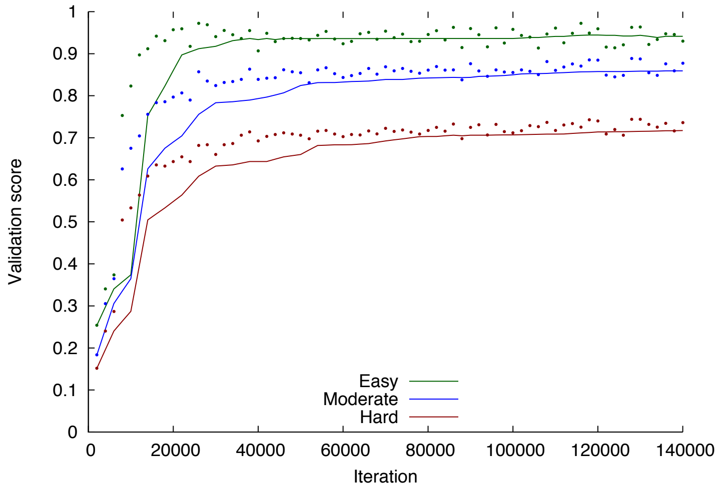
- KittiBox
  - A collection of scripts to train the FastBox model on the KITTI Object Detection dataset

KittiBox Implementation

- **Uses TensorVision**
  - A library to build, train and evaluate neural networks in TensorFlow
  - A json file is used to specify
    - The dataset
    - The architecture
    - The training schedule (optimizer, learning rate, loss, etc)
- **Runs on a single GPU**
- **Trains the model for 140,000 iterations where each iteration processes a mini batch of 5 images**

KittiBox Training on P100

- **Full run on 1 P100 node on an XC50 Cray system**
- **Run time is 27h30min**
  - Validation time is **3h**
    - Includes evaluation every 2000 iterations
  - Training time is **24h30min**
    - On average 0.63 sec (per Batch); 7.93 imgs/sec
- **Evaluation results (end of training)**
  - easy : 0.93
  - medium : 0.87
  - hard : 0.73
  - Speed (msec per image) : 37.89



Source: Diana Moise, Cray Performance team

Computational Scaling of the Training

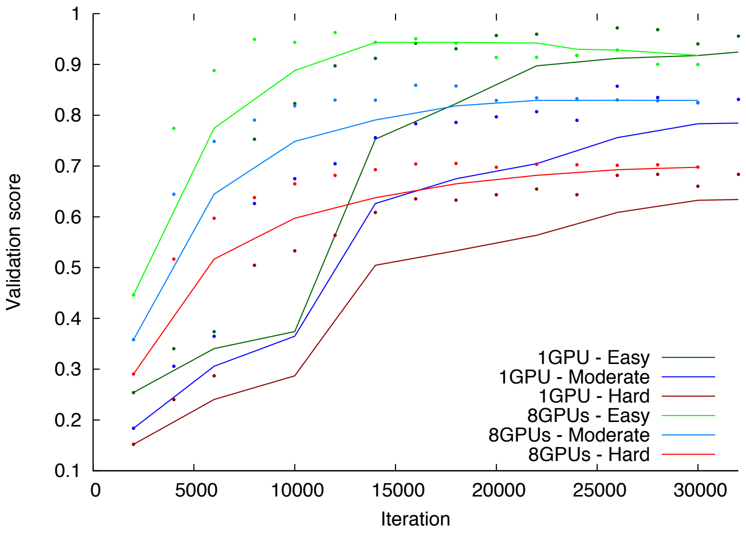
- **Cray's ML plugin allows to easily scale the training to multiple nodes**
- **Original scripts were slightly modified to use the plugin**
- **Data parallelism is already supported**
  - Implemented via a shuffling of the file names in the input queue

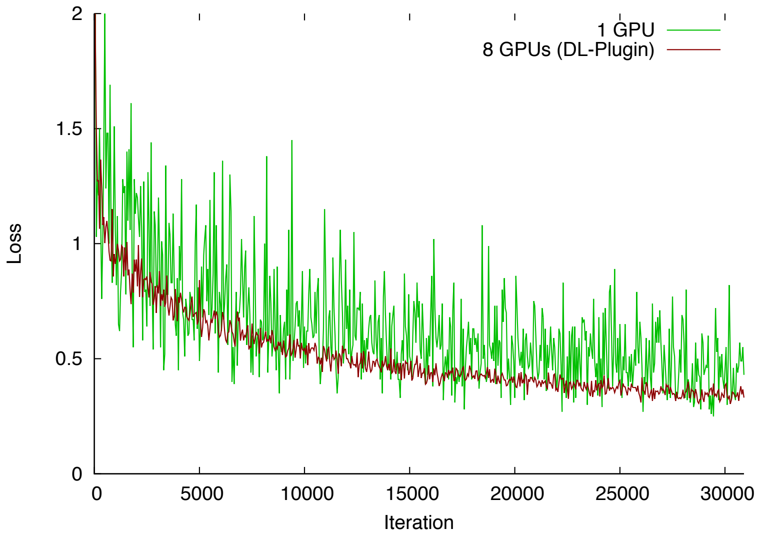
- **Several SGD schemas available with the plugin**
  - Preliminary tests use Synchronous SGD
    - This will speed up the learning process, without affecting the accuracy

- **When using multiple GPUs, training time can be reduced (by reducing the number of steps)**
  - Good scaling would reduce the training time by the same factor as the number of GPUs

Training on 1GPU vs 8GPUs (DL-plugin)

- **When using multiple GPUs, the validation precision increases earlier on, leading to faster convergence**
  - The noise in the loss curve is also greatly reduced





Scaling Results

#GPUs	#steps	training time (h)	validation accuracy (raw values)			performance (img/s)
Mini batch size = 5						
			easy	medium	hard	
1	140000	24.5	0.930	0.877	0.736	7.93
8	17500	3.2	0.958	0.894	0.753	60.8
16	8750	1.62	0.960	0.892	0.744	122.24
Mini batch size = 16						
1	43750	23.55	0.959	0.890	0.745	8.28
8	5470	3.02	0.936	0.866	0.735	64.64
16	2735	1.52	0.967	0.867	0.738	129.28

Scaling Results (cont.)

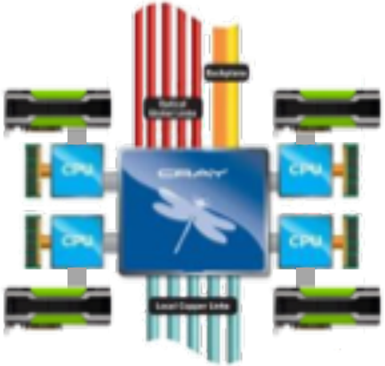
- **A larger mini-batch size enables**
  - Better GPU utilization, thus better performance and shorter training time, while reaching similar accuracy
- **With the DL-plugin, almost perfect scaling is achieved**
  - Small fraction of time spent on communication
  - Training time substantially reduced from 24.5 hours to 1.5 hours when using 16 GPUs
- **Better scaling and easier to use than Horovod**

Deep Learning on Piz Daint at CSCS – Cray XC50

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

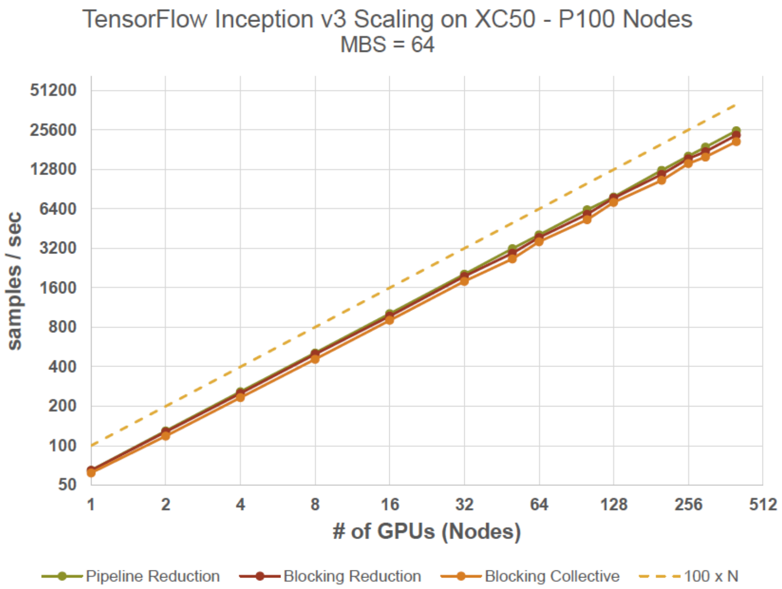
- 4500 NVIDIA P100 GPUs
- #8 on the November 2016 Top500 List
- #2 on the November 2016 Green500 List

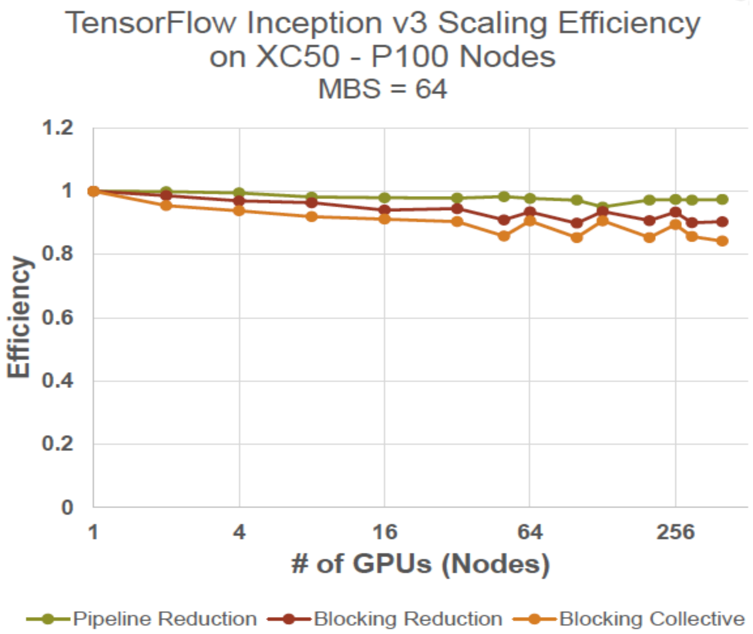
1CPU:1GPU node architecture



Differentiating Results: TensorFlow

HPC Thinking: Message-size, MPI-collective, Global all-reduce modifications





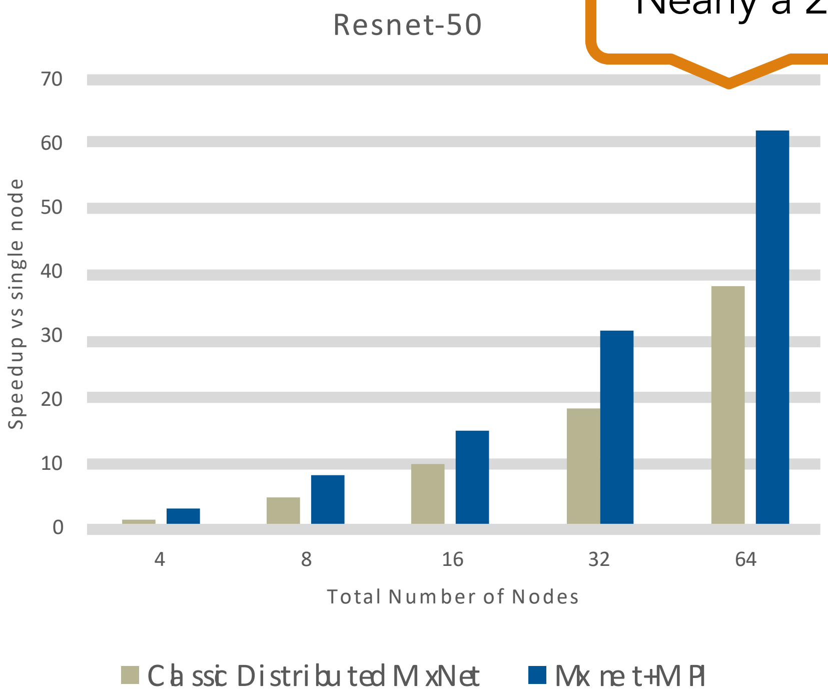
90%+ scalability efficiency that can reduce training time from days to hours

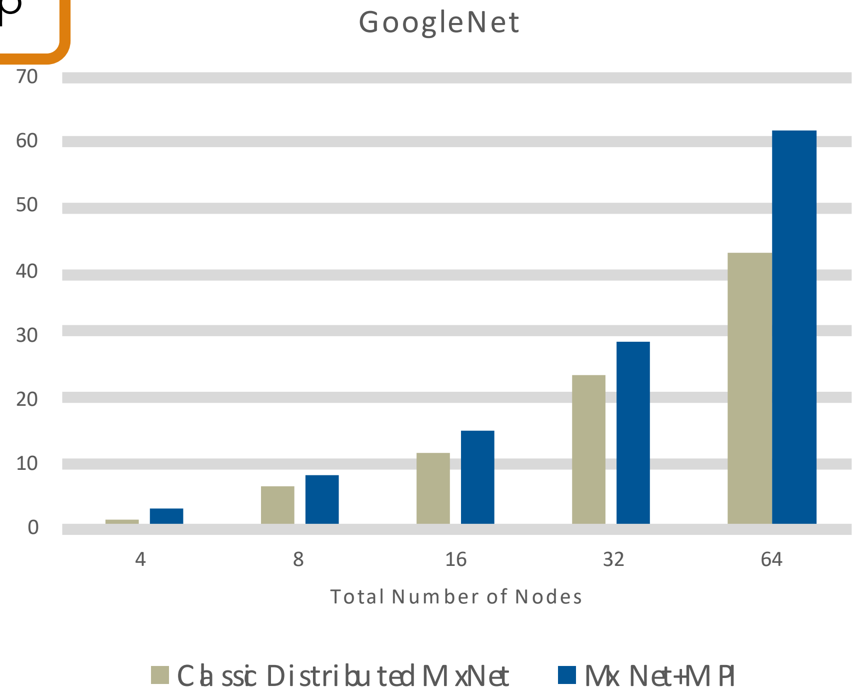
Source: Peter Mendygral and Jef Dawson, Cray PE and Performance

Differentiating Results: MXNet

Distributed vs. Cray MPI approach

Nearly a 2x speedup





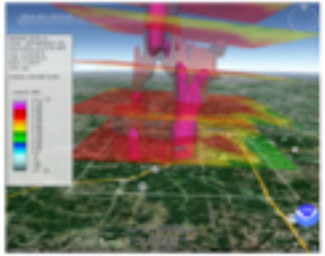
What is missing? End-to-end workflows!

- **A DL/AI problem is not only about training or inference**
- **Many other challenges**
  - Data preparation and ingestion
  - Handling of AI output
  - Interpretability and reliability of prediction
- **For example**
  - **Precipitation Nowcasting Leveraging Deep Learning and HPC Systems to Optimize the Data Pipeline**
    - **[Alex Heye (Cray), AMS 98th Annual Meeting, 2018]**

Dataset Processing

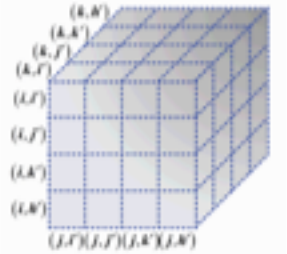
Data Collection

- Historical Radar Data (NETCDF)
- Geographical Region
- 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

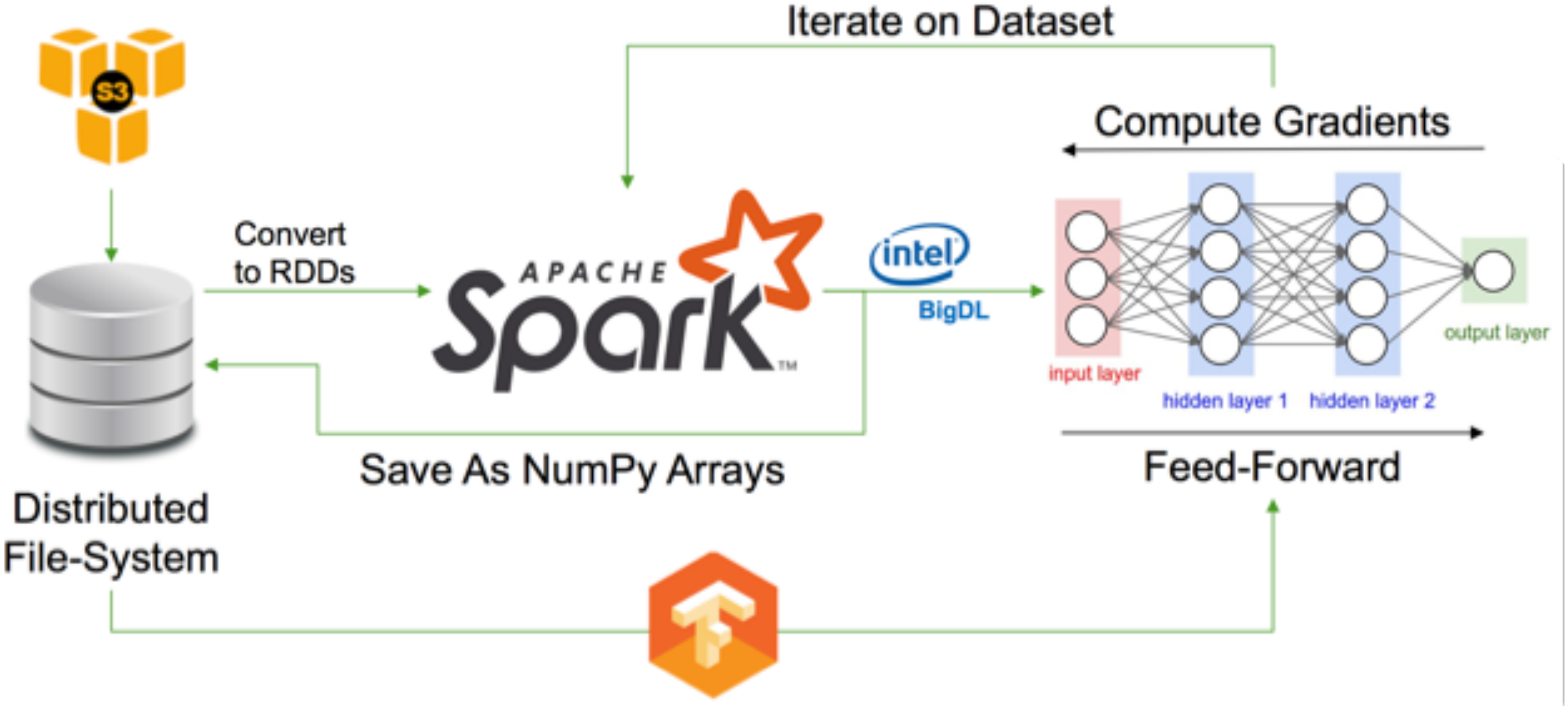
intel

BigDL

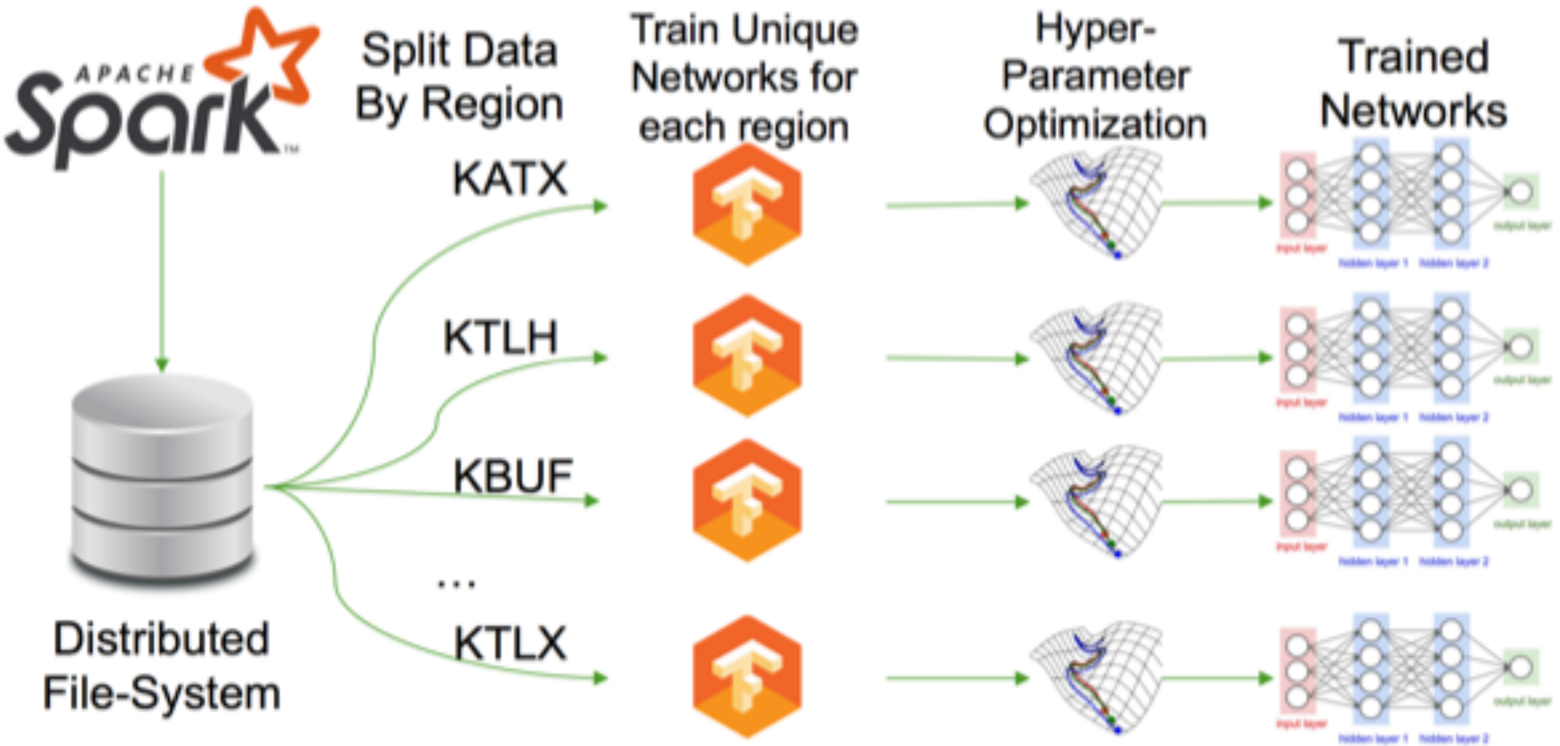
Framework

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

Data Processing Pipeline



Distributed Training



Idealized Timeline

- **Station: KATX**
- **Dataset size:**
  - 118,342 Sequences
  - 101GB
- **Parameters**
- **Systems:**
  - Data processing: Cray Urika-GX – 1024 cores
  - Training: Cray CS-Storm – 8 Nvidia P100 GPUs

Process	Wall-Time	Proportion
Download	13 hours	32%
Spark	4 hours	10%
Training	24 hours	58%
Inference	10 seconds	0%

Scaling

- Tensorflow via Cray MPI Com. Plugin
- Nvidia Tesla P100 GPUs
- Batchsize of 4 samples per device
- Throughput in Samples/Second

Device Count	Throughput	Scaling Efficiency
1	25.8	1.0
2	51.6	1.0
4	102.7	.995
8	205.4	.995
16	410.5	.994

$$\frac{\text{Training } 58\%}{(16*0.994)} + \text{Rest } 42\% = \cancel{100\%}\ \ 45.65\%$$

>2x speedup

Legal Disclaimer

Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

Cray Inc. may make changes to specifications and product descriptions at any time, without notice.

All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Cray uses codenames internally to identify products that are in development and not yet publicly announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

Q & A

Overall Session