WP2 Update

Sergey Panitkin

BNL

BigPanDA TIM in Washington, DC

September 23, 2008

Outline

- Containers on Titan
  - Introduction
  - Current state on Titan
  - Container builder
- Study of CSC108 impact on other projects on Titan.
  - Wait time analysis
- ML for backfill project
- Summary

Containers on Titan

- Singularity containers were introduced on Titan in late fall 2017
- First results were reported at the BigPanDA TIM in Arlington and at ATLAS meetings
  - Build recipes developed for containers with single ATLAS software release and corresponding databases
  - OLCF specific container configuration
  - Performance comparisons were done with ATLAS detector simulation jobs
    - Different container form factors (EXT4, SquashFS)
    - Placements on different filesystems at OLCF (NFS, Lustre, RAM disk)
- Containers showed good IO properties. Low load on Lustre MDS
  - Due to switch in IO pattern: *access to one large file instead of access to multiple small files*
- Scalability tests were performed and confirmed good IO properties
  - Splunk and strace analyses
  - Up to 1000 nodes per job
- Results reported at the BigPanDA meeting at BNL in April 2018
- Plans for introduction of container into ATLAS production on Titan were discussed

Singularity Issues and Updates

- In April 2018 security vulnerability was uncovered in Singularity 2.4
- As a result OLCF removed Singularity from Titan
- A new Singularity version 2.5 was released with fixes for the vulnerability
  - Also requires certain features to be present in Linux kernel
- Plans for Singularity updates on Titan were uncertain due to the old Linux kernel on worker nodes that do not support features in v. 2.5
- Cray was consulted but no plans for kernel upgrade were announced at the time
- In August 2018 a new Singularity module 2.5.0 became available for (unofficial) testing on Titan.
  - OLCF's container expert Adam Simpson left for Nvidia
  - Jack Morrison took over
- No documentation is available about the new set up
- Our old containers did not work with this version
- A prototype container builder facility was introduced at OLCF and made available for testing.
  - The facility is expected to serve both Titan and Summit

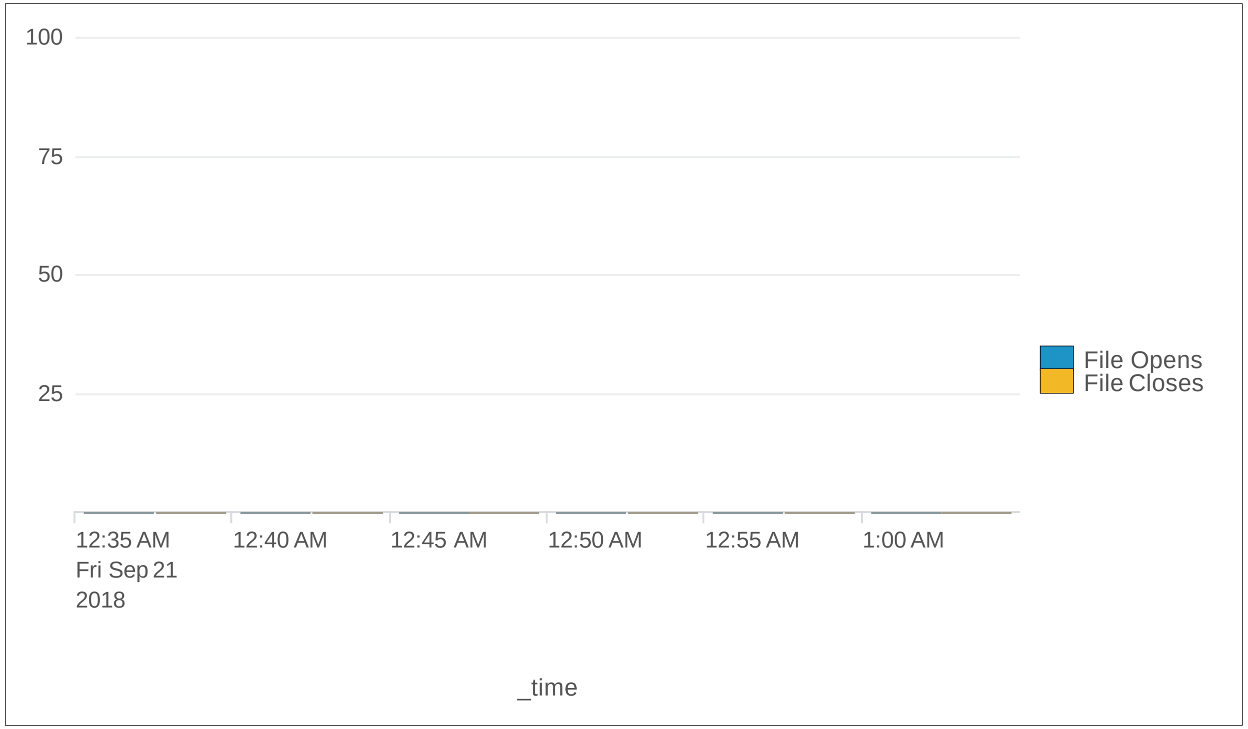
Container builder

- In August 2018 a new container building setup was introduced at OLCF and we were invited to test it
- Employs VMs for container building
- Special module on Titan for build tasks submission
- None of Titan’s shared filesystems is visible from the build VM
  - Imposes constraints on container build recipe
    - access only to remote files (wget, git, etc)
    - That can slow down build process and introduce instabilities due to remote access latencies
- Limited documentation or examples
- Build process was unstable in early on in August

Current status and short term plans

- I updated recipes for container building to reflect new Singularity and container builder setup at OLCF
- I build new containers on my laptop and tested them on Titan with ATLAS detector simulation jobs.
- I used container builder at OLCF to build container with the new recipes
  - The builder works more stable now (2 out of 3)
  - Required to put ATLAS database elements on my Dropbox
- Got Splunk profiles for the new containers and confirmed that they looked similar to the old ones
- The updated container is available to the CSC108 project members along with an example how to configure and run it on Titan
- Wen Guan will test it with Event Service jobs
- Danila will test it with Harvester and ATLAS production jobs

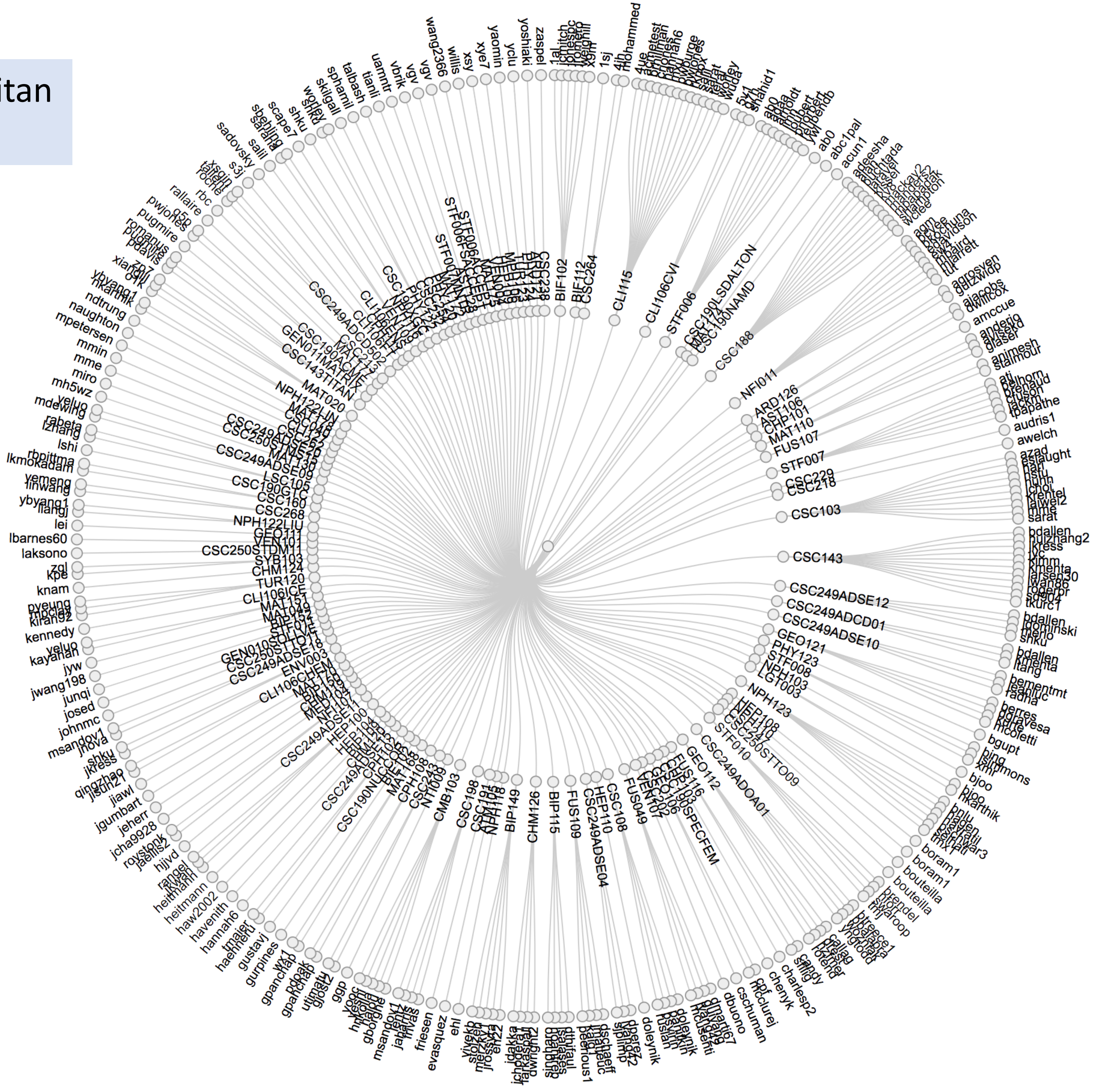
Splunk profile for a new container on Titan



Job 430487

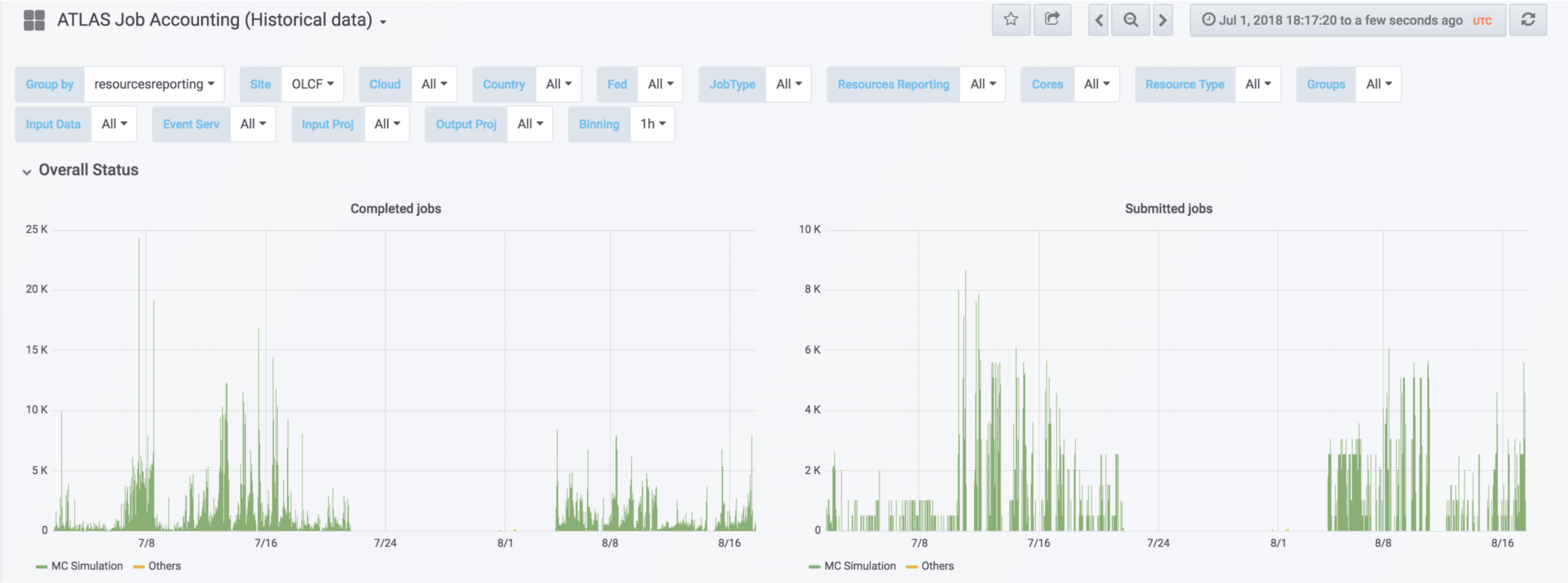
ATLAS detector simulation job
Singularity 2.5.0
Image built with OLCF's container builder
Good IO pattern, same as seen for ATLAS containers before

Projects and users on Titan
March 2018



Just a nice picture

ATLAS Backfill jobs July-August 2018



- From July 21 to August 4 no backfill jobs were submitted to Titan due to pause in ATLAS simulation campaign
- This gave an opportunity to evaluate influence of ATLAS backfill operations on wait time of other projects on Titan

Wait time for non-CSC108 jobs in June-August 2018
All bins

~ two weeks

Period	7/1 to 7/7	7/7 to 7/14	7/14 to 7/21	7/22 to 8/4	8/5 to 8/12	8/12 to 8/18	Mean
Wait Time, h	6.0	4.8	6.1	5.6	6.1	3.8	5.4 (5.3)
Jobs	10975	10925	20131	33090	11074	4157	

- Preliminary analysis shows significant (~1 hour) week-to-week variations in wait time of non-CSC108 projects
- No significant reduction in wait time during the pause in ATLAS backfill operations

Wait time for non-CSC108 jobs in June-August 2018
Bin 5 (requested nodes [1; 125])

Period	7/1 to 7/7	7/7 to 7/14	7/14 to 7/21	7/22 to 8/4	8/5 to 8/12	8/12 to 8/18	Mean
Wait Time, h	2.4	2.2	1.5	3.6	3.6	3.0	
Jobs	9706	9544	18545	29933	9613	3285	

- Preliminary analysis shows significant (~2 hour) week-to-week variations in wait time of non-CSC108 projects in this bin
- No evidence of influence of CSC108 activities on other projects

Wait time for non-CSC108 jobs in June-August 2018 Bin 4 (requested nodes [126; 312])

Period	7/1 to 7/7	7/7 to 7/14	7/14 to 7/21	7/22 to 8/4	8/5 to 8/12	8/12 to 8/18	Mean
Wait Time, h	12.5	20.3	45.4	13.4	27.1	4.4	
Jobs	780	823	788	1700	718	526	

- Preliminary analysis shows significant (~30 hour) week-to-week variations in wait time of non-CSC108 projects in this bin
-

Wait time for non-CSC108 jobs in June-August 2018
Bin 3 (requested nodes [313; 3749])

Period	7/1 to 7/7	7/7 to 7/14	7/14 to 7/21	7/22 to 8/4	8/5 to 8/12	8/12 to 8/18	Mean
Wait Time, h	67.7	24.2	78.0	38.4	18.6	10.0	
Jobs	439	513	753	1356	703	313	

- Preliminary analysis shows significant (~50 hour) week-to-week variations in wait time of non-CSC108 projects in this bin

Wait time for non-CSC108 jobs in June-August 2018
Bin 2 (requested nodes [3750; 11249])

Period	7/1 to 7/7	7/7 to 7/14	7/14 to 7/21	7/22 to 8/4	8/5 to 8/12	8/12 to 8/18	Mean
Wait Time, h	40.1	44.6	25.8	12.9	20.1	13.3	
Jobs	36	38	38	89	32	33	

- Preliminary analysis shows significant (~20 hour) week-to-week variations in wait time of non-CSC108 projects in this bin
-

Wait time for non-CSC108 jobs in June-August 2018
Bin 1 (requested nodes >11249)

Period	7/1 to 7/7	7/7 to 7/14	7/14 to 7/21	7/22 to 8/4	8/5 to 8/12	8/12 to 8/18	Mean
Wait Time, h	77.4	57.6	36.1	35.2	34.8	-	
Jobs	14	7	7	12	8	0	

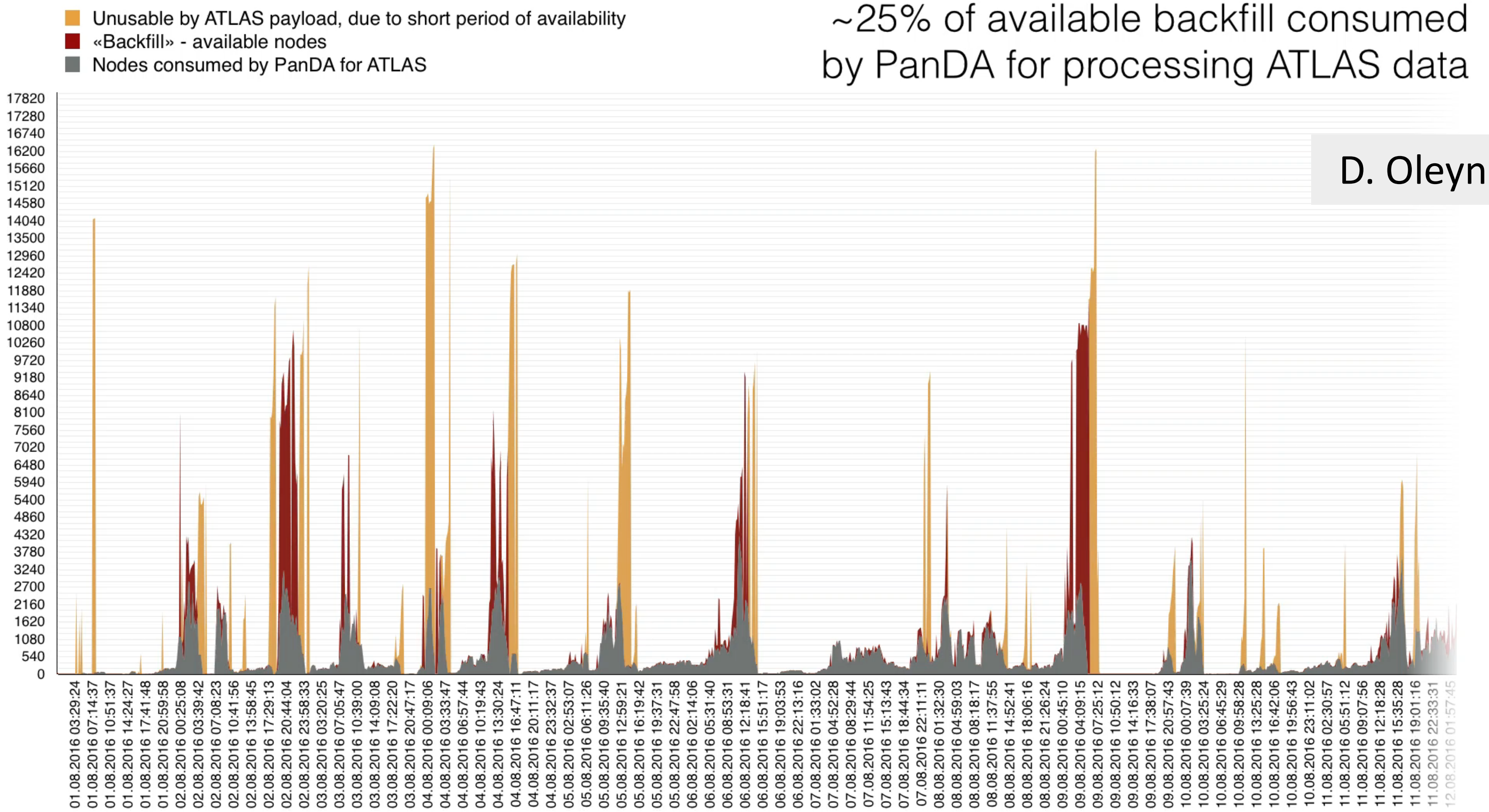
- Preliminary analysis shows significant (~40 hour) week-to-week variations in wait time of non-CSC108 projects in this bin
- No indication of influence of CSC108 on other projects on Titan

Machine Learning for Titan backfill

- BigPanDA developed a system for efficient utilization of “backfill” on Titan
- The system monitors currently available resources and dynamically shapes ATLAS jobs, both in size and duration, to fit them
- This system operates since 2016 and collected more than 250M core hours with monthly peaks exceeding 20M core hours and backfill utilization efficiency sometimes exceeding 50%.
- It helped to increase Titan’s utilization by ~3%
- How do we improve backfill utilization efficiency and increase ATLAS job throuput on Titan?
- One idea is to try to Machine Learning (ML) tools

PanDA backfill on Titan in action

~25% of available backfill consumed
by PanDA for processing ATLAS data

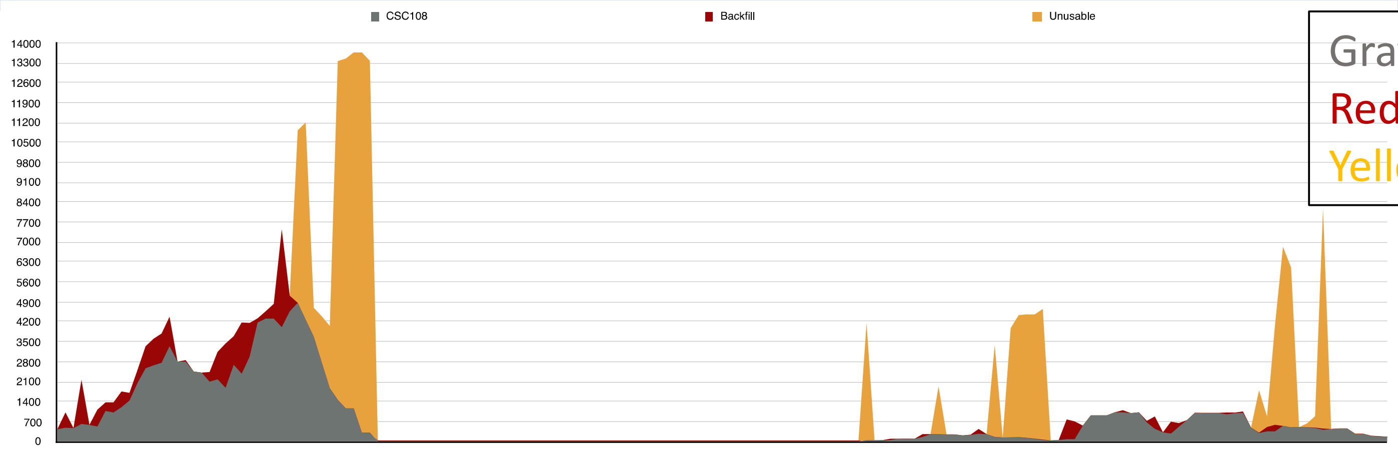


D. Oleynik

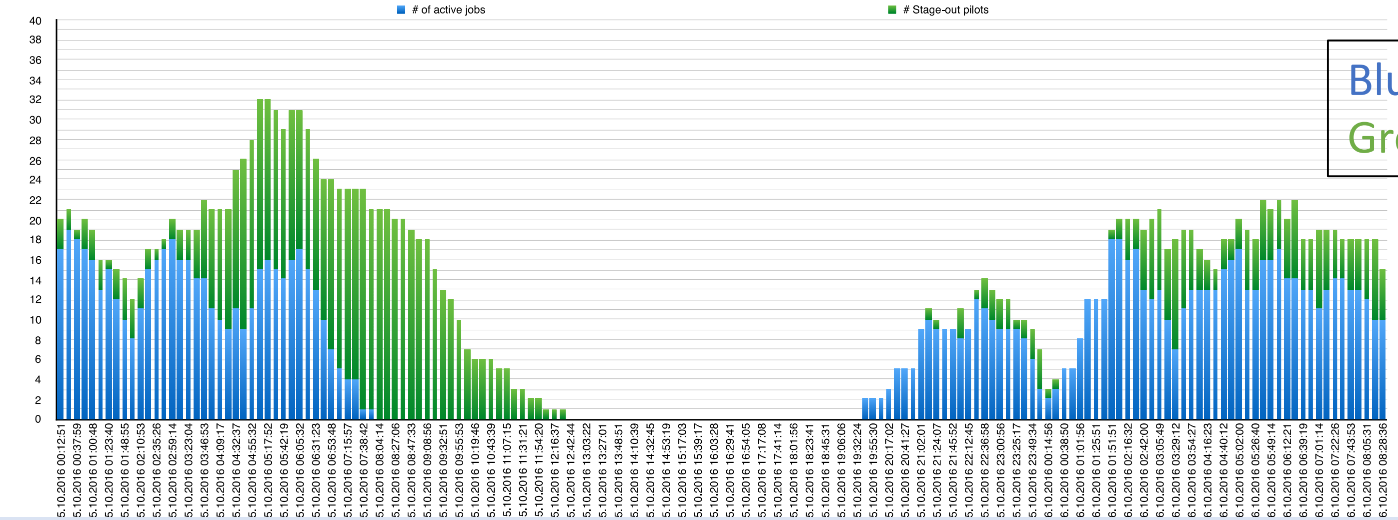
- “Good” backfill peaks are not always used completely
- Backfill often comes in “waves”
- Number of pilots is limited by policy and can be exhausted

PanDA backfill operations on Titan

D. Oleynik



Gray – used backfill
Red – unused backfill
Yellow – “bad” backfill



Blue – running jobs
Green – pilots stage-out

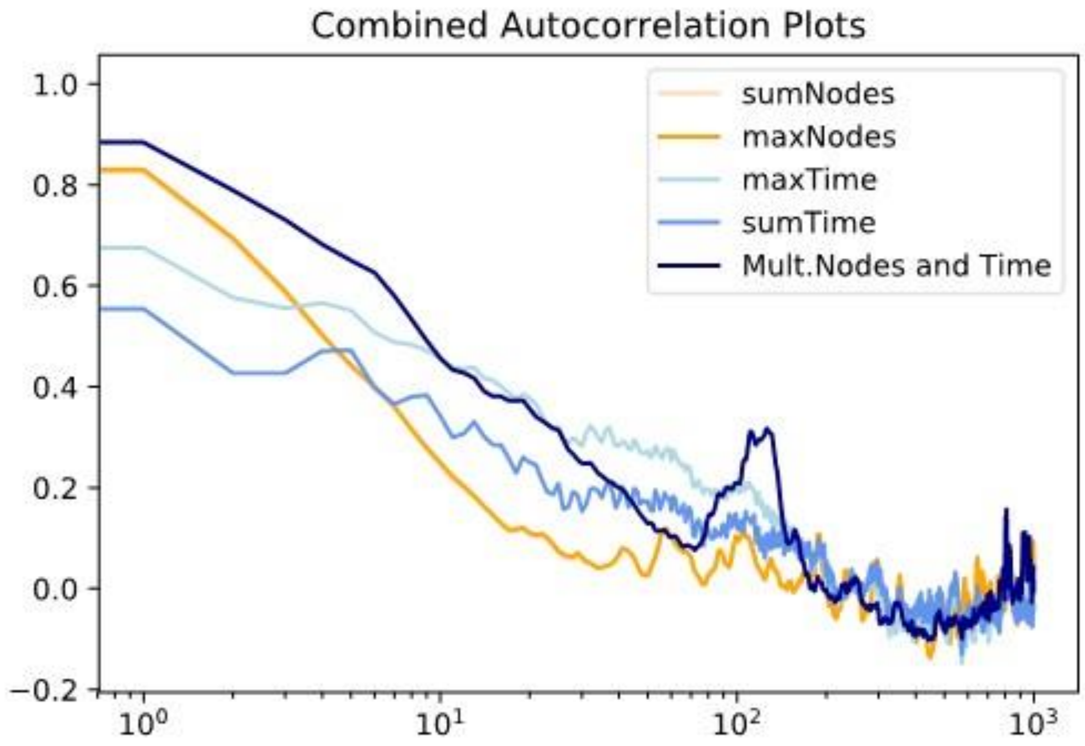
- Usually (not always) we can run up to 20 active pilots on 4 DTN nodes, with up to 32 pilots doing data stage-out
- Greedy algorithm for resource capture– we see backfill - we try to grab it
- Number of nodes per pilot/job Is limited to 350 due to stage in IO concerns (may change with Harvester)
- A possibility to ran out of pilots for large “peaks” in resource availability
- Can we do better than greedy? Can we predict “peak arrivals” and change job submission pattern to use peaks more efficiently

Machine Learning for Titan backfill

- This experimental project was started in August 2018 as a collaboration between BigPanDA and ML experts from the Center for Data Driven Discovery (C3D) at Computational Science Initiative (CSI) at BNL
  - Shinjae Yoo (CSI)
  - Chenxiao Xu (SUNYSB and CSI)
  - Peggy Yin (summer student at CSI)
  - Sergey Panitkin (BNL)

- Objective: Design improved high throughput backfill algorithm using statistical analysis or machine learning of backfill timeseries in real time.
- Approach: Using both statistical and deep machine learning techniques (i.e. autoregressive model, deep neural network, recurrent neural network, etc), predict compute resource availability on Titan at OLCF. By leveraging this predictive models, design more efficient job submission algorithm than the currently used greedy resource capture method.

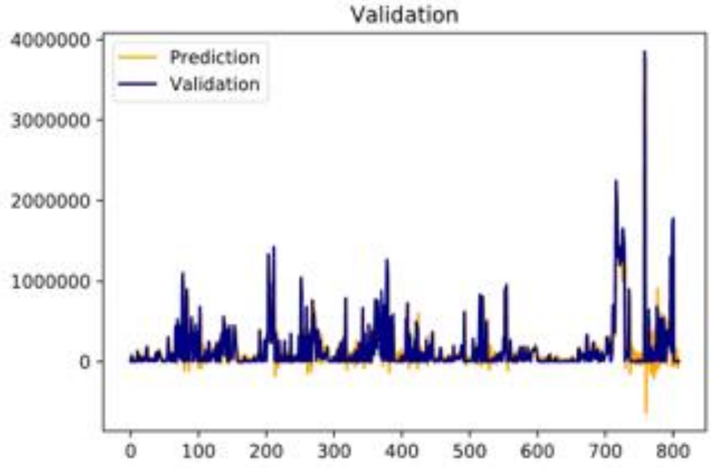
Statistical approach. Autoregression model

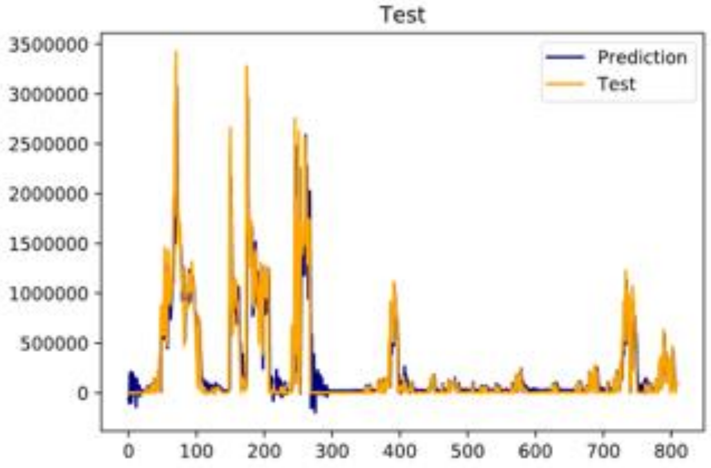


X-axis : 10 minutes log scale

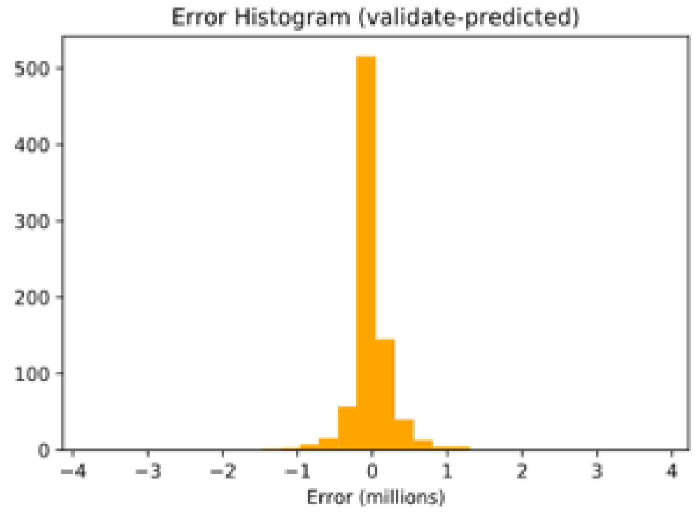
- Backfill data from PanDA edge service on Titan. (T; N_avail ,T_avail). ~ 1 month worth of backfill data
- 50% training set , 25% validation set, final 25% - test data set
- Different variables were tried. NxT product seems to be the best metrics
- Autocorrelation coefficient is positive over a wide range
- Good short term predictability ~0.9 at 10 minutes, ~0.7 at 30 minute interval
- Peaks at 1 day and 1 week

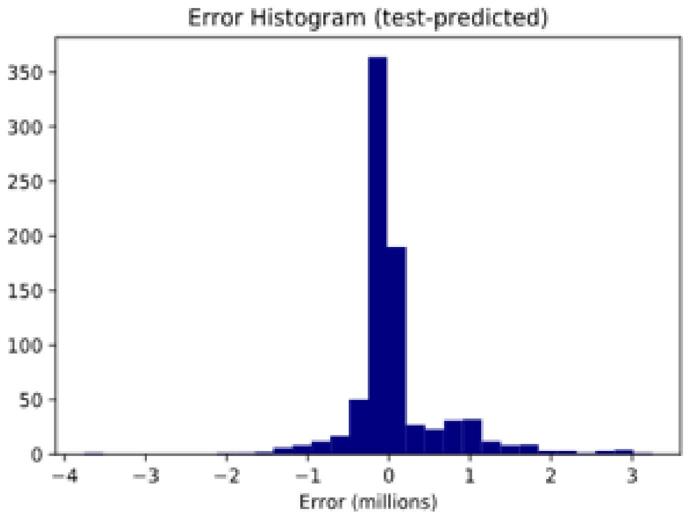
Statistical approach. Autoregression model





Model vs Data





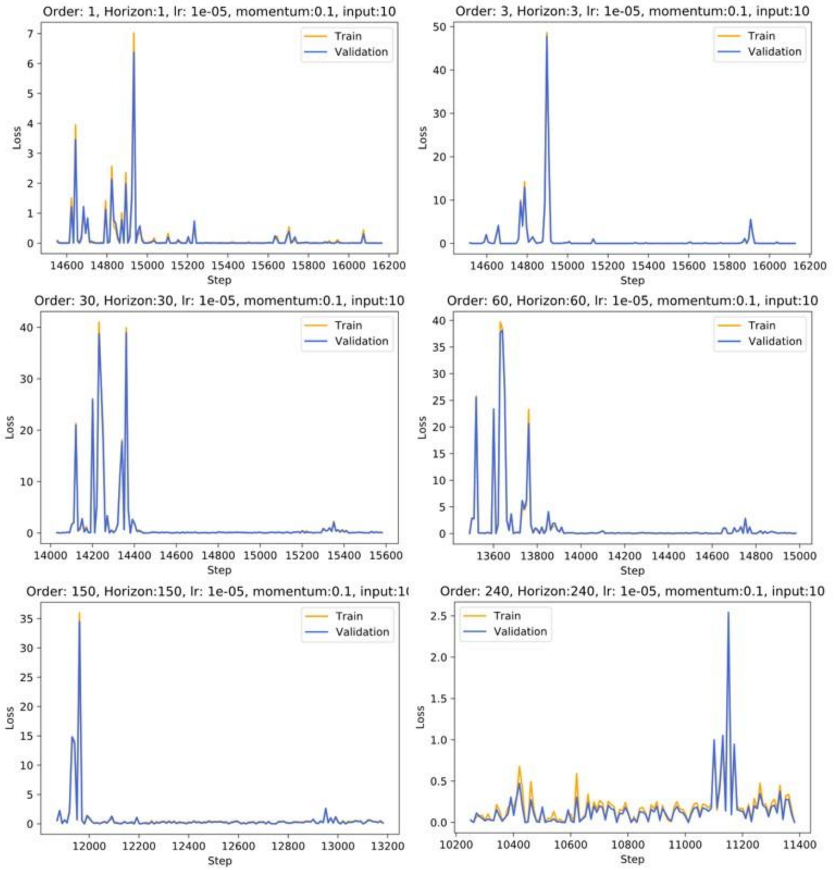
Errors

Errors centered around zero. Large tails (poor peak description?)

Multilayer Perceptron Model (MLP)

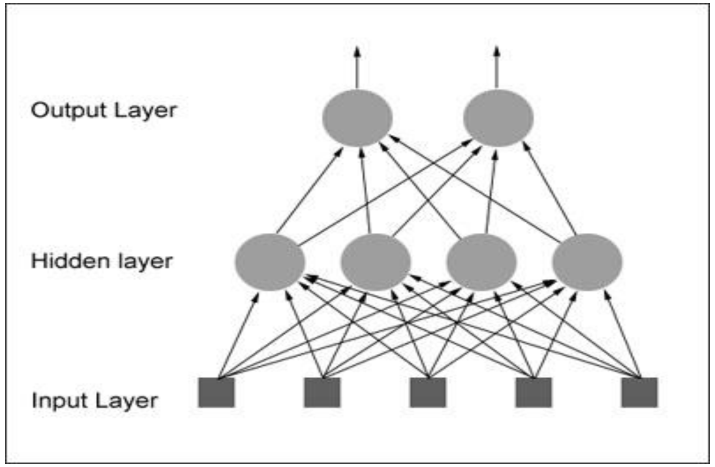
- MLP is a type of feed forward neural network
- More sophisticated than AR, not well suited for time series
- Implemented in PyTorch. single hidden layer with 10 nodes and ReLU activation function
- Varied: order, horizon, number of hidden layers, learning rate (lr), and momentum.
- In general MLP did not show good results

Loss per step for various parameter combinations



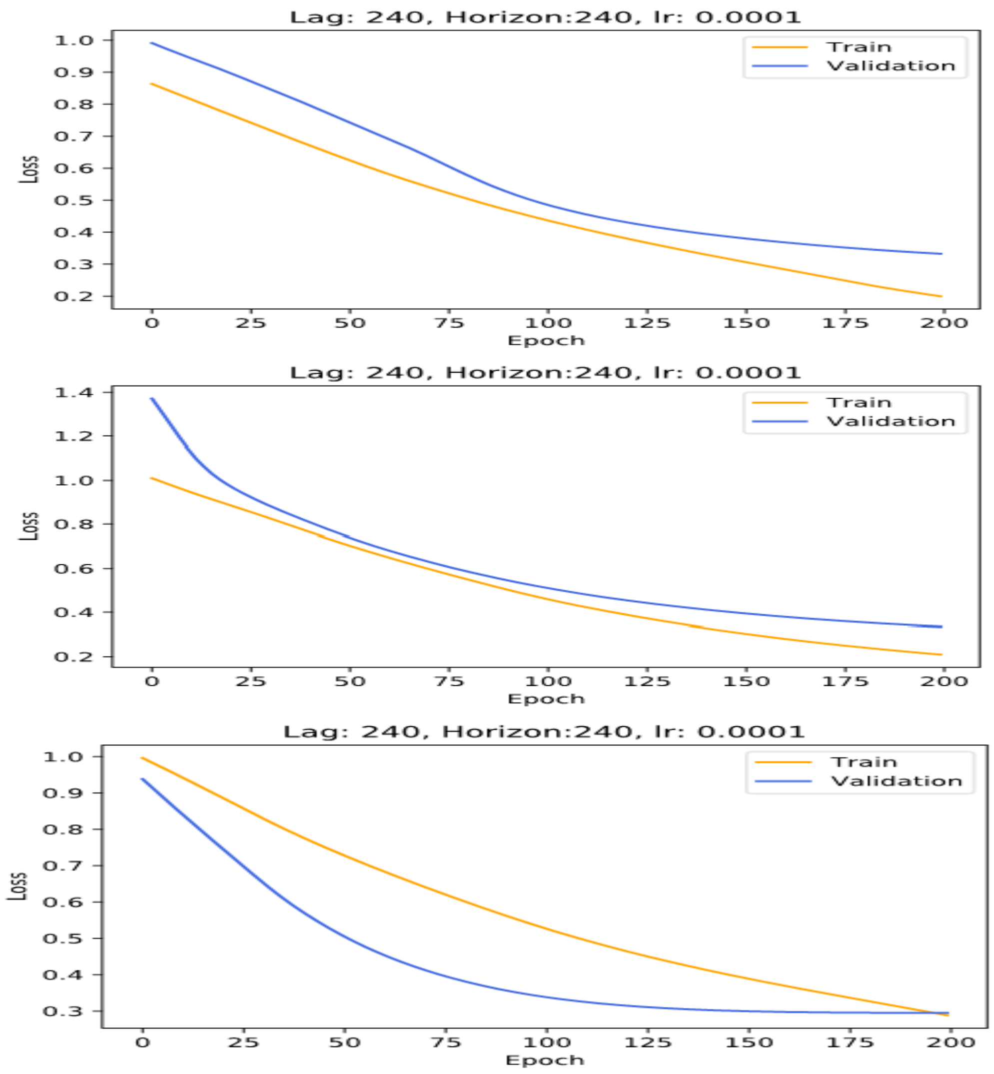
Order = p and horizon = q means that for a given time stamp t we use (t-p +1: t) time stamps to predict the t + q timestamp

MLP example

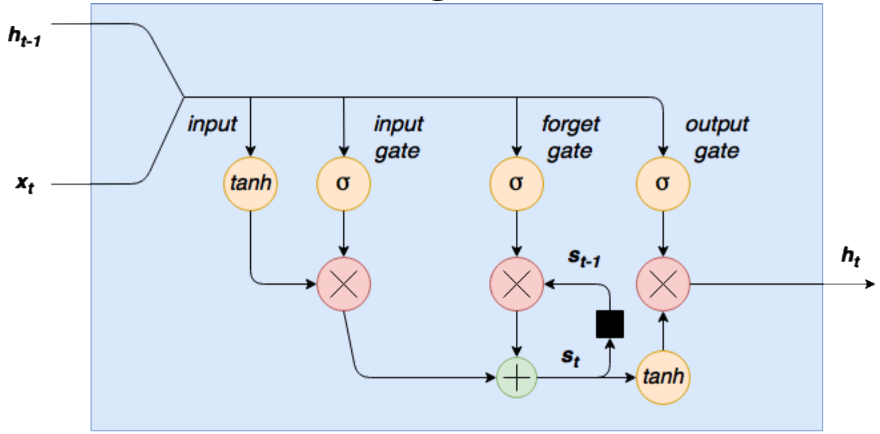


Long Short Term Memory Model (LSTM)

- LSTM is a special type of recurrent neural network
- Often used for time series analysis
- More sophisticated than MLP. Long term dependencies in data can be learned
- Implemented in PyTorch
- Showed very promising results even with our relatively small data sample and small number of epochs in training
- Model Loss function seems to converge quickly



LSTM cell diagram



Summary

- Containers on Titan
  - Container studies resumed on Titan after a pause related to Singularity security issues
  - New container building tools tested at OLCF
  - Updated containers built and tested
  - Container tests with Event Service (ES) and Harvester are planned
    - ES I/O pattern study is planned
- Study of CSC108 impact on other projects on Titan was performed
  - Wait time analysis indicates that there is no evidence that CSC108 activities impacted other projects on Titan
  - Large natural fluctuations in wait time were observed for all batch bins over the analyzed period
- Experimental ML for backfill project is was started
  - Several techniques were applied to Titan backfill data (AR, MLP, LSTM)
  - Preliminary results are encouraging
  - AR model shows wide positive autocorrelation
  - LSTM model promising predictive behavior
  - Training on larger dataset s planned

Plans for the next 3 month

- Containers on Titan
  - Container tests with Event Service (ES) and Harvester are planned
    - ES I/O pattern study is planned
- ML for backfill
  - Continue collaboration with C3D ML experts
    - DNN based models
    - refinement of the LSTM model
  - Training on larger datasets planned for LSTM model

The END

Long Short Term Memory Model

- MLP is a type of feed forward neural network
- single hidden layer with 10 nodes and ReLU activation function
- Implemented in PyTorch