Machine Learning Assisted System Monitoring and Diagnostics @Scale

Or: The Olympians Watchful Reign

Aaron Saxton, Michael Showerman, other Blue Waters Team members

What Is It

As part of its Holistic Measurement Driven System Assessment (HMDSA)[1] and quality assessment efforts, the Blue Waters Project[2] has an ongoing effort to architect and study ML models that process a system's monitoring and usage data to create a mixture of descriptive, predictive, and prescriptive analysis to assist systems managers and performance experts with near real time monitoring and diagnostics.

The team at NCSA, and Blue Waters staff in particular, has decades of problem solving experience on HPC systems. It is only natural that if useful machine learning models for system monitoring and diagnostics emerge it would be among such experienced individuals. Most recently, the extensive, but non-intrusive, instrumentation on Blue Waters gives an unprecedented source of data to convert this team's experienced practices into automated techniques. Blue Waters is producing on the order of 80GB per day of system monitoring data[3] that includes but not limited to 81 derived metrics per minute for each of almost 27,000 compute nodes and from each almost 14,000 high speed network interfaces. On a given day, Blue Waters runs thousands of jobs for upr to 48 hours. Those jobs can be very large up to the entire system and span all science disciplines and a range from traditional simulation to data

[1] Saurabh Jha, Jim Brandt, Ann Gentile, Zbigniew Kalbarczyk, Greg Bauer, Jeremy Enos, Michael Showerman, Larry Kaplan, Brett Bode, Annette Greiner, Amanda Bonnie, Mike Mason, William Kramer and Ravishankar Iyer: "Holistic Measurement Driven System Assessment", IEEE, 2017 IEEE International Conference on Cluster Computing (CLUSTER), pp 797-800, Honolulu, Hawai'i, U.S.A., 2017, doi:10.1109/CLUSTER.2017.124

[2] Brett Bode, Michelle Butler, Thom Dunning, William Gropp, Torsten Hoefler, Wen-mei Hwu, and William Kramer (alphabetical). *The Blue Waters Super-System for Super-Science*. Contemporary HPC Architectures, Jeffrey Vetter editor. Sitka Publications, November 2012.Edited by Jeffrey S . Vetter, Chapman and Hall/CRC 2013, Print ISBN: 978-1-4665-6834-1, eBook ISBN: 978-1-4665-6835-8

[3] Michael Showerman, Jeremy Enos, Joseph Fullop, Paul Cassella, Nichamon Naksinehaboon, Narate Taerat, Thomas Tucker, James Brandt, Ann Gentile, and Benjamin Allan. "Large Scale System Monitoring and Analysis on Blue Waters using OVIS." Proc. Cray Users Group, 2014.

analytics[4]. It is a challenge in itself to create any meaningful and actionable analysis in a timely manner.

The algorithmic abilities of machine learning have increased by leaps and bounds in recent years as it will be spelled out below, but part of that success is due to the efficient ML frameworks. Packages like Tensorflow and PyTorch have been optimized to take full advantage of both a CPUs and GPUs. This, mixed with our distributed training experience, gives us a nearly unbounded ability to process growing data sets.

Using TensorFlow to construct a model with a combination of convolutions, LSTM layers, and an unsupervised auto-encoding training process, the team is prototyping a model that can be used for anomaly detection and job classification. The model has been constructed and run on REAL metric data at scale. Preliminary training of the model showed that it will be computationally intensive, but with the resources available at BW, very possible. For prototyping, we use a 2016 commodity MBP laptop. The performance profile below shows that with its 8 cores encoding take 7-10 seconds, then decoding plus optimizing weights takes an additional 35 seconds to process metric data for 6 one hour long jobs. The input tensor is for data of Blue Waters 24x24x24 3D torus (the largest ever created in a Cray system), 60 times slices, and about 81 channels corresponding to each metric. A job mask is applied to the tensor to isolate data specific to each job. Since tensor operations are done with dense tensor arrays, in principe, this is the performance profile for analyzing any sized job on Blue Waters. Training these models will be a computational challenge at the system scale and duration, but once trained and online, they have the potential to process larger quantities of data.

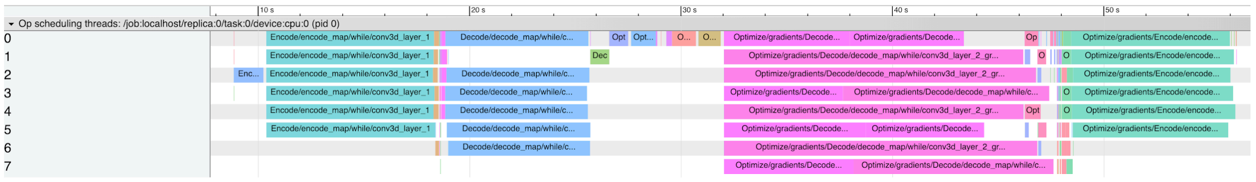

Fig 1. Total time of one training step. Integers on left column is core index. Timeline in seconds is across the top.

Where Did It Come From

In math science and engineering there are many common motifs in problem solving. Two problem motifs stand out. The first can be thought of as a reductionist approach. That is, reduce the misbehavings of a system to its most basic components then study how each of the components contributes to a system's anomalous behaviour. In a systems engineering

[4] *Workload Analysis Of Blue Waters* (ACI 1650758), Matthew D. Jones, Joseph P. White, Martins Innus, Robert L. DeLeon, Nikolay Simakov, Jeffrey T. Palmer, Steven M. Gallo, and Thomas R. Furlani (furlani@buffalo.edu), Center for Computational Research, University at Buffalo, SUNY, Michael Showerman, Robert Brunner, Andry Kot, Gregory Bauer, Brett Bode, Jeremy Enos, and William Kramer (wtkramer@illinois.edu), National Center for Supercomputing Applications (NCSA), University of Illinois at Urbana Champaign, January 13, 2017
https://bluewaters.ncsa.illinois.edu/liferay-content/document-library/Technical%20Reports/XDMoD-BW-FinalReport.pdf

approach, this often involves combing through log data looking for an important warning or error message from a specific component. When “Eureka!” finally comes, the solution of this approach is often something like “Oh, this system variable was set wrong”.

The second motif can be thought of as a global approach. In this case, one would look at trends in a system's behavior and identify a taxonomy of “good” system states and “bad” system states. In this situation, an administrator or performance analyst may look at histograms and regressions of system data. With the analysis they could modify a systems policy or architecture to better service “good” or and reduce the likelihood of “bad” states. Clearly these approaches are not mutually exclusive and a proficient practitioner will balance their efforts across all the variations of these approaches. Herein lies the problem. An experienced practitioner will choose the right details to focus on and keep the right trends in the back of their head while architecting solutions. But the amount of data and uses of a system is ever growing which places more pressure on a practitioners experience and the resulting tribal knowledge.

With all the success of machine learning in many disciplines three themes emerge. Methods that are “translation invariant”, methods that “remember”, and methods that “encode”.

Convolutional Neural Networks (CNNs) are the breakout hit success that cracked the computer vision problem. One of the reasons is that convolutions are translationally equivalent. That mixed with max-pool’ing allows a CNN to classify an image no matter where the defining feature is located in the image.

Recurrent Neural Networks (RNNs) were a promising mechanism to create very deep models with potentially unbounded *memory* capacity. Unfortunately, they suffered from the *vanishing gradient problem* if they are used without care. Long Short Term Memory (LSTM) networks introduced gating in RNN’s thus allowing the state of a LSTM to selectively “remember” features as it was exposed to ongoing streams of data. This has become useful in natural language processing for semantic labeling and document summarization. In their raw form, RNN’s map a sequence of data to another sequence of data and are able to remember earlier features to understand context of current features.

Auto-Encoding models are created by architecting an encoding model that reduce the dimension of the input data and then reverse the process for the decoder. The training process optimizes the weights of the model so that the input is the same as the output of the encoder-decoder pair. As a result, the output of the encoder by itself represents a classification of the input. This classification when fed into the decoder will produce the original input image. Auto-Encoders have become popular with image synthesis[5]. That is, say you want to synthesize a zebra standing in a horse pasture. One would train an auto-encoder with an image data set of zebras and horses. The output of the encoder represents all the various learned features of the training set. If one chooses an arbitrary vector to feed the trained decoder, its output will be an image with a mixture of horses, zebras, and pasture features. If one chooses the right vector,

[5] https://arxiv.org/pdf/1703.10593.pdf

you will get the desired horse in a pasture. In other words, auto-encoders do a good job of characterizing complex data AND they do it unsupervised.

With the problems of systems engineering casted in the above light and all the promises that machine learning has given us in recent years, it is still ambitious to create a system that can monitor, diagnose, and fix itself. But with the glut of data pouring out of HPC systems, the ability of systems engineer and/or application performance analysts to have a balanced problem solving strategy becomes more challenging.

Where Will It Go

Scale Up!

Scaling the data on which code runs by one order of magnitude often reveals new problems. While it is certain there will be technical problems such as load balance with scaling up these techniques, the capability of the model to learn useful information from larger amounts of data will be another challenge. In machine learning, one hot topic is *Hyper Parameter Optimization*. Hyper parameters are a loosely defined feature of ML. They can include parameters such as learning rate and batch size used for training. But the model architecture can also be parameterized. For example, number of convolution layers, size of convolution kernels, size of LSTM cells, number and size of hidden layers are all features of a model' utility that will become more or less apparent as the scale of training and testing data grows. Many of these parameters will adversely affect a model's computational profile. Fortunately, preliminary investigation has shown we still have a lot of latitude varying these parameters as we scale training data. These are two performance profiles comparing a model with 2 and 6 fully connected perceptron layers. Aside from the initial convolution and the final convolution_transpose, changing the middle layers seems to have little effect on performance.

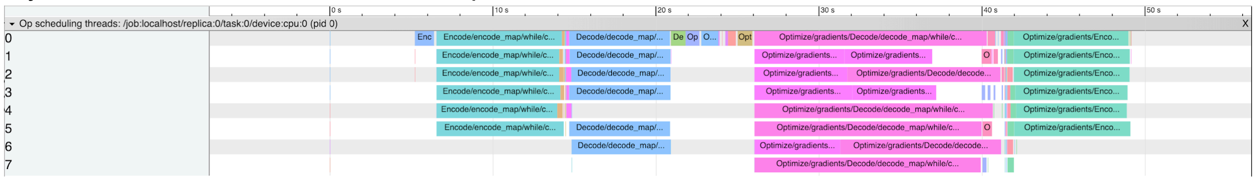

Fig 2. Total time of one training step. Integers on left column is core index. Timeline in seconds is across the top.
2 hidden fully connected layers

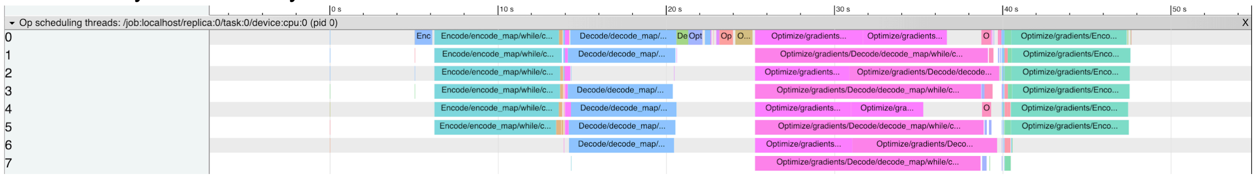

Fig 3. Total time of one training step. Integers on left column is core index. Timeline in seconds is across the top.
6 hidden fully connected layers

Improve Model Architecture

One large drawback of using machine learning is that there is no a priori way of guaranteeing the accuracy profile of a model. The Blue Waters team has started with the most basic combination of convolutions, LSTMs, and auto encoding. These choices were made by drawing on success from other disciplines that had similar features as the system monitoring problem. However, it is still unclear how robust each component must be. Convolutions can be mixed with max pooling, concatenation, or residual functions. LSTMs can be layered in novel ways or even generalized to gated recurrent units. Auto encoding can be augmented with adversarial networks. Future work includes studying the effects of some of these features as they relate to usable system analysis.

Results Analysis and Visualization

Results of machine learning models can sometimes feel as if it's speaking in semaphore. What is more, many technical communities complain it is hard to diagnose *why* or *how* a model arrived at a result. That is, traceability is important to users. One major goal of this project is to augment a human with tools to help them describe, diagnose, and predict misbehaving systems. Once the model produces results, there are several ways we can slice the data so that it is useful to the widest range of follow up analysis. The visualizing features of the convolutional layers can give us insight to locality of system behavior. Augmenting and analyzing the hidden layers of the LSTM can give insight to causality. A user will also have their own ideas for analysis and understand.

Acknowledgements: This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications.