Characterize and Classify HPC Applications Using Machine Learning Models

Zhengchun Liu[1], Rajkumar Kettimuthu[1], Ian Foster[1], Nageswara Rao[2]
[1]Argonne National Laboratory; [2]Oak Ridge National Laboratory

1 ModSim components

Our research falls under the ModSim 2019 topic area "*Modeling and Simulation of Artificial Intelligence and Machine Learning as a Method of ModSim*". Specifically, for all the jobs submitted to Mira in 2018 at ALCF, we combined and correlated scheduler logs (recorded resource allocation), Darshan logs (recorded file I/O), and Autoperf logs (recorded MPI calls and hardware performance counters). Our comprehensive analysis broadens understanding of factors that influence HPC application and system performance. We believe these insights are valuable for guiding current system and application optimization, and designing future supercomputers. Based on our analysis, we engineer features to classify HPC jobs. We trained a machine learning model based on our features can get a classification accuracy of 99.6%.

2 Target application

This work was focused on HPC applications run at supercomputer center, e.g., DOE leadership computing facilities. The big picture here is using machine learning to enhance HPC applications and HPC systems.

3 Modeling techniques

We extract interpretable features from logs to represent tasks in order to study the commonality of HPC applications. The features we used so far to represent a task in the machine learning model space are as follows: (1) Fraction of run time spent on communication (i.e., time spent on MPI routines); (2) Fraction of run time on file I/O; (3) Operations per second and floating point operations per second; (4) Number of processes (i.e., MPI rank) per node; (5) Average RAM fetch/store per CPU cycle. As one can see, all the features are raw (not derivative) and interpretable. This makes the machine learning model trained with these features and its inference are possibly interpretable. Machine learning models will be trained to classify applications so that to (1) justify if the HPC resource is actually used as the initial allocation proposal; (2) detect anomalous application behave because of either user's misconfiguration (e.g., scalable parameters) or system exceptions.

4 Preliminary results

The t-SNE is a technique developed for the visualization of high-dimensional data. It converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data. In order to get an overview of the representative capability of proposed features, Figure 1 plots the two-dimensional t-SNE embedding of the feature representation in which we label points using color for top 10 executable names and the rest executable names are marked as red. Here we assume that different jobs with the same executable name are runs of the same application but configured differently. As one can see from Figure 1 that jobs' executable name are represented by features pretty accurately. We note that t-SNE is used for visualization purpose only, which we believe can somehow demonstrate the representative capability of the proposed features.

To classify if a given job belongs to any of the existing applications. We trained a classifier using gradient boosting tree (XGBoost). By using 70% of the data (320k jobs) to train the machine learning model and testing on the rest 30% (135k jobs). The model can identify the application with an accuracy of 99.6%.

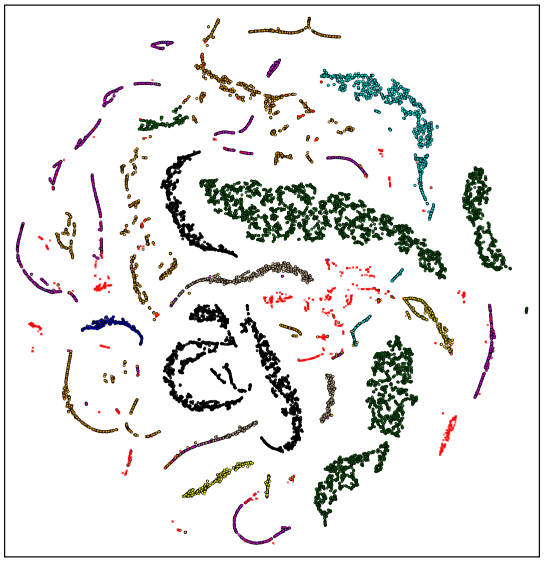

Figure 1: 2D t-SNE embedding of the task representation. Dots (tasks) with the same color share the same executable name.