Power Consumption Optimization in High-Performance Computing

Sepideh Goodarzy
Reshma Morampudi
Maziyar Nazari
Department of Computer Science
University of Colorado Boulder
Boulder, CO 80302

1 Motivation

Why this project is a good (or bad) idea?

Power is one of the limiting factors in High Performance Computing (HPC) at Exascale and will continue to influence future advancements in super computing. Although controlling the energy consumption in the HPC platform is not the only way to control the associated costs in super computing but it can be considered as an important thing to be focused on. In this project, we replicated a paper[1] and extended their work and we aimed at applying different regression models, ensemble methods like Random Forest and Adaboost and sequential models to predict the mean power consumption for the actual scientific applications running on a super computer named Peregrine, at NREL facilities. We hope that a better job scheduling algorithm can in fact increase the energy efficiency in data ceters through these results.

2 Background

The Energy Systems Integration Facility (ESIF) at the Natural Renewable Energy Lab (NREL) in Golden, Colorado houses one of the most energy-efficient HPC data centers in the world through an innovative integration of the HPC system with the building and campus infrastructure. The heavy component of this integeration is the availability of data on a majority of HPC systems, including performance, scheduling, cooling, and energy usage. This provides a great testbed to explore trade offs (for instance in terms of cost and efficiency) with respect to the power and energy constraints.

Peregrine is the principle HPC machine at ESIF data center which is a Hewlett-Packard (HP) system composed of 1440 Intel Xeon nodes incorporating Sandybridge and Ivybridge microarchitectures with a peak performance of 1.19 PetaFLOPs. The informatics and data capture system collects real time information about jobs, power and performance data. The system uses Torque resource manager and adaptive computing Moab scheduler for monitoring and scheduling all jobs. Therefore, data related to jobs is collected by parsing these Torque and Moab logs every 15 minutes and storing available data in a relational database for future use. Time series data related to nodes power and performances are captured using a sensor polling technique every 10 seconds on each node and storing the data within a time series cluster. Therefore, the whole archived data can be accessed for later analysis using a combination of relational database and costume time-series cluster

Furthermore, what is interesting to do on this kind of data which is provided by the authors, and has not been done in the paper is to try sequential model with LSTM layer, because in many previous papers, scientist used sequential models and LSTM to model their time series data and in many cases it performs pretty well on time series data.[5, 6] It should be also noted that the code for this project is not publicly available and we have done all of the things either in the paper or as an extension from the scratch.

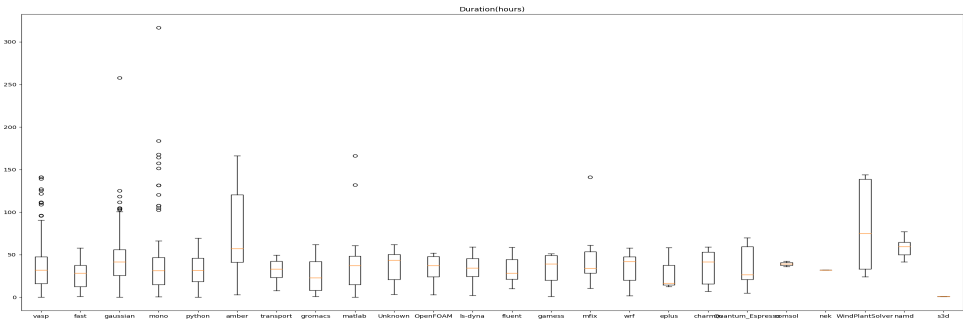

Figure 1: Duration statistics grouped by application name

3 Data and Analysis

What you did?

We considered a dataset in [3], which consists randomly sampled 10,000 jobs for a year long, continuous data which reflects the overall nature of the jobs on Peregrine. Each job corresponds to at least one time series of node level power usage observed at 10s intervals. Jobs that operate on multiple nodes are associated with multiple power time series. This leads to a full dataset of 18510 uniquely identified power time series. For each unique time series, various metadata including hardware characteristics, user requests, run time, and application type are available. A sample data object is provided at [7]

An important part of any machine learning project is feature selection and engineering for the training dataset. For our NREL dataset, a 12.5 gigabyte file with 10,000 job submissions to the Peregrine cluster, we had several different features organized in a single JSON file, with each key being a feature and the value for that key the feature's value. As preprocessing steps, we did data cleaning, data Analysis, feature selection, feature scaling and centralizing.

For data cleaning, we omitted zero power values (as outlier), discarded jobs with near zero variance in their time series power values because they do not add useful information to our models. we considered jobs running on phi processors with power consumption greater than 700W and less than 90W as outliers because Intel Phi processors generally consume a maximum of 700W on high load and a minimum of 90W of power consumption. Non phi-processors generally consume 300W of power on high load and 90W in ideal consumption, so for non Phi processors we considered power consumption greater than 300W and less than 90W as outliers. We eliminated jobs with more than 50% of outliers in their time series power values from our dataset. We considered only those jobs having Moab and Torque exit codes equal to 0 which indicates successful job termination.

Data Analysis: To know the power consumption trends for different applications running on Peregrine, we plotted box plots to understand the differences in job duration(hours), Total Energy(KWh), Median Power(KW) and power range(KW) for selected applications (amber, python etc.)

Feature Importance: Based on the OLS(Ordinary Least Squares)summary, we considered p value and t-stats to know which features are more important. We kept only those features with (P < |t-stats|) < 5%. The p-value for each term tests the null hypothesis that the coefficient is equal to zero (no effect). A low p-value (< 0.05) indicates that you can reject the null hypothesis. In other words, a predictor that has a low p-value is likely to be a meaningful addition to our model because changes in the predictor's value are related to changes in the response variable.

We encoded any categorical data like queue type, application name, feature requests and any true false values to numbers.

Feature Scaling and Centralizing: We tried StandardScalar, MaxAbsScalar, RobustScalar, MinMaxScalar. Although there weren't so much difference between them, we got slightly better results for MinMaxScalar, so we used it for normalizing the data. Normalizing data is very important while

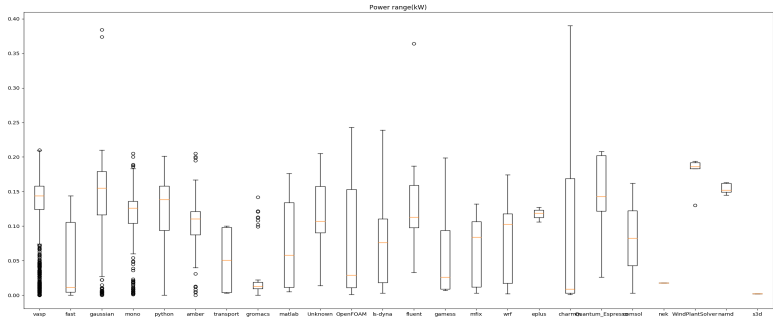

Figure 2: Power range statistics grouped by application name

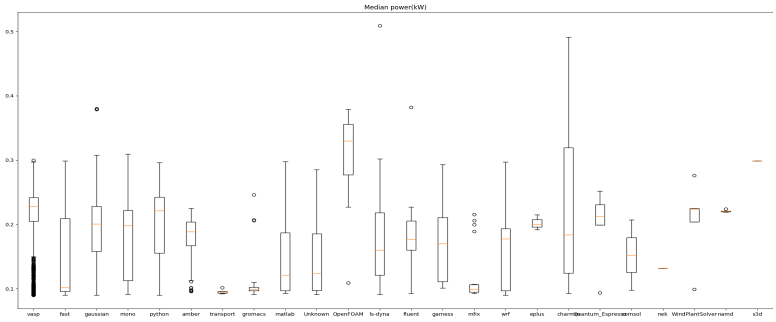

Figure 3: Median power statistics grouped by application name

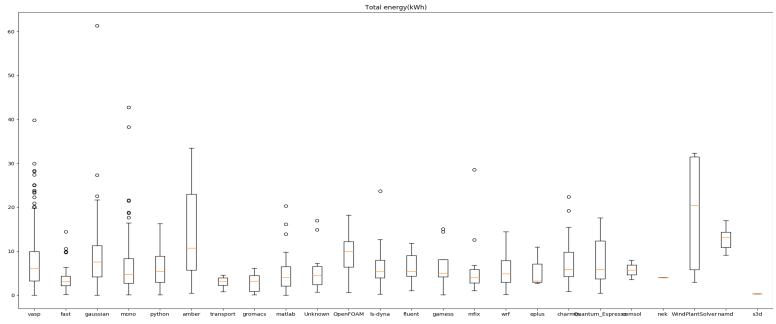

Figure 4: Total Energy statistics grouped by application name

using LSTM, so this is an important step in our analysis. It should be noted that feature importance, encoding, normalizing and centralizing have not been done in the paper and should be considered as an extension.

4 Power Forecast

In [2], authors used two approaches for predicting the mean power consumption. In a Priori prediction, mean power consumption is predicted based on the type of application, memory usage, nodes, hardware platform, memory configuration, user characteristics and etc. For example for our model for a Priori prediction, features used are type of queue, application name, processors used, CPU used, wallclock requested, features requested, nodes requested, type of job(interactive). In in Situ power prediction along with some of factors considered in a Priori like type of queue, application name, wall clock requested, we also incorporated small period of observed power consumption data (which indicates the first 5 minutes of job runtime). Here are the models we considered for a Priori and in Situ mean power prediction. Models are Linear Regression, Ridge Regression, Lasso Regression, PLS Regression, Multiple Adaptive Regression Splines (MARS), Random Forest, Adaboost and Sequential model with a LSTM layer and 6 dense layers. For Ridge regression we considered alpha value to be 1. Generally Lasso Regression not only helps in reducing over-fitting but also in feature selection. For Lasso Regression we considered alpha=1 but did not see much change compared to linear regression. For Partial Least Squares Regression(PLS), the number of components are 2. For Random Forest model we considered number of estimators$= 100$ and max_depth $= 8$. For Adaboost model, we considered base estimator as Decision tree with max_depth $= 8$, number of estimators $= 100$ and learning rate $= 0.3$. In Sequential model with an LSTM layer and 6 dense layers, the dimensions of these dense layers are 256, 128,64, 32,16 and 1. The output dimension of LSTM is 100. The reason for considering dimensions, number of layers in our sequential model or parameters like max-depth, number of estimators and etc is just deciding on hyperparameters and bias-variance trade off. Thus, we came up with these numbers which gave us the best results. Ridge and Lasso regression are techniques to reduce model complexity and prevent over fitting which may result from Linear Regression. Based on the prior work, we have an idea that Sequential model with LSTM layer would work good for time series data. We observed the same results for our problem space.

5 Results

For these models we considered the same parameters for both a Priori and in Situ predictions. Figure 1 shows the Root Mean Squared Error(RMSE) for the models we considered. We can observe that LSTM gave the lowest RMSE in both a Priori and in Situ predictions among all the models considered. Lower the RMSE, better the model is. Table shows the values predicted by all the models in both a Priori and in Situ conditions. We can observe that all the Regression models predicted almost similar mean power values, Adaboost and Random Forest are next best. Finally using Sequential model with LSTM, we got better results, in fact in the paper, authors just did the Linear regression, MARS and Random Forest. Whereas, we added all other regression models Adaboost and Sequential as extensions to the paper. With respect to results, our LSTM model works the best amongst all models. Thus it is better than the paper results.

Model	A Priori	In situ
Linear Regression	41.9	17.2
Ridge Regression	41.9	17.2
Lasso Regression	42.1	17.3
PLS Regression	42	17.2
MARS	38.4	17.2
Random Forest	32.1	15.5
Adaboost	34	16
Sequential (LSTM, 6 Dense Layers)	**31.2**	**12.9**

Figure 6 and Figure 7 show the minimization of loss function RMSE for mean power consumption for A priori and in situ sequential models respectively. We chose 50 epochs because after 50 epochs we didn't observe much change in the RMSE so we decided up on using 50 epochs for the model.

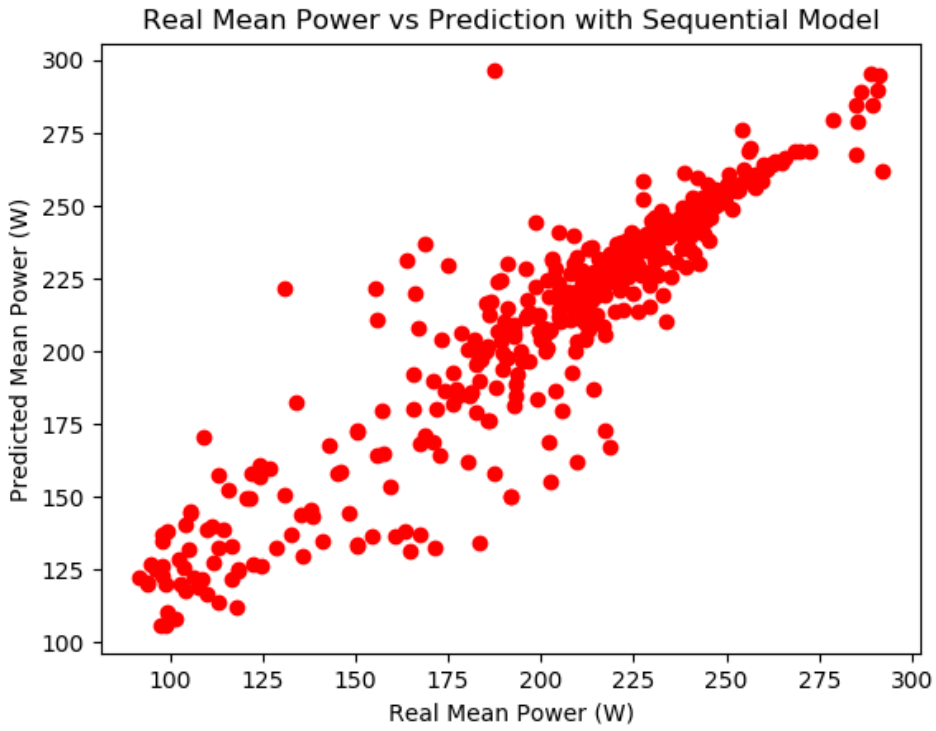

Figure 5: Sequential In situ, Predicted mean power vs Real mean power

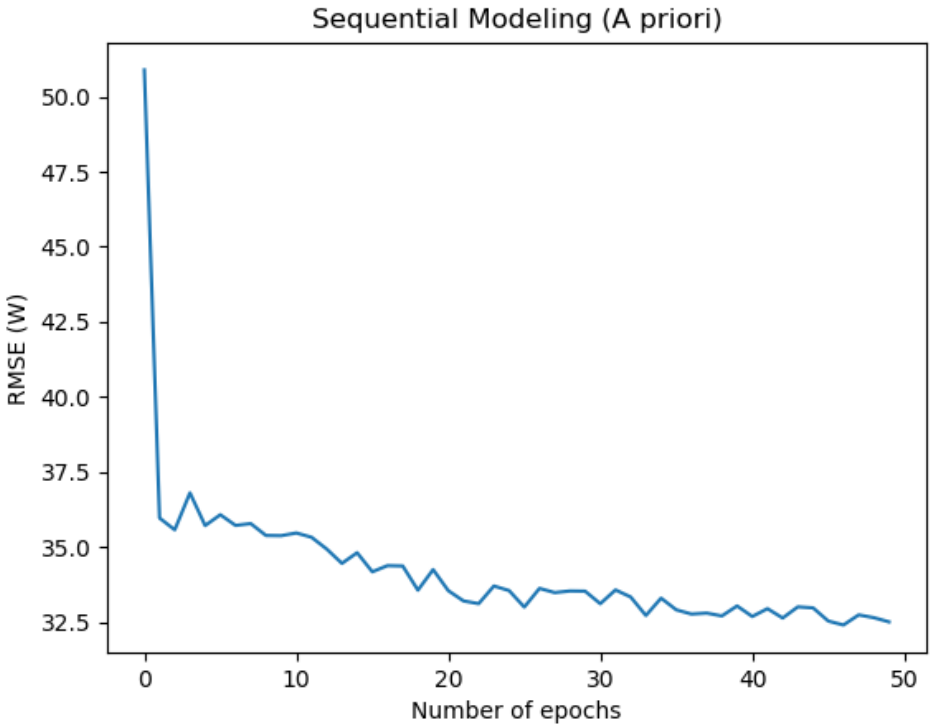

Figure 6: Sequential a priori RMSE over 50 epochs

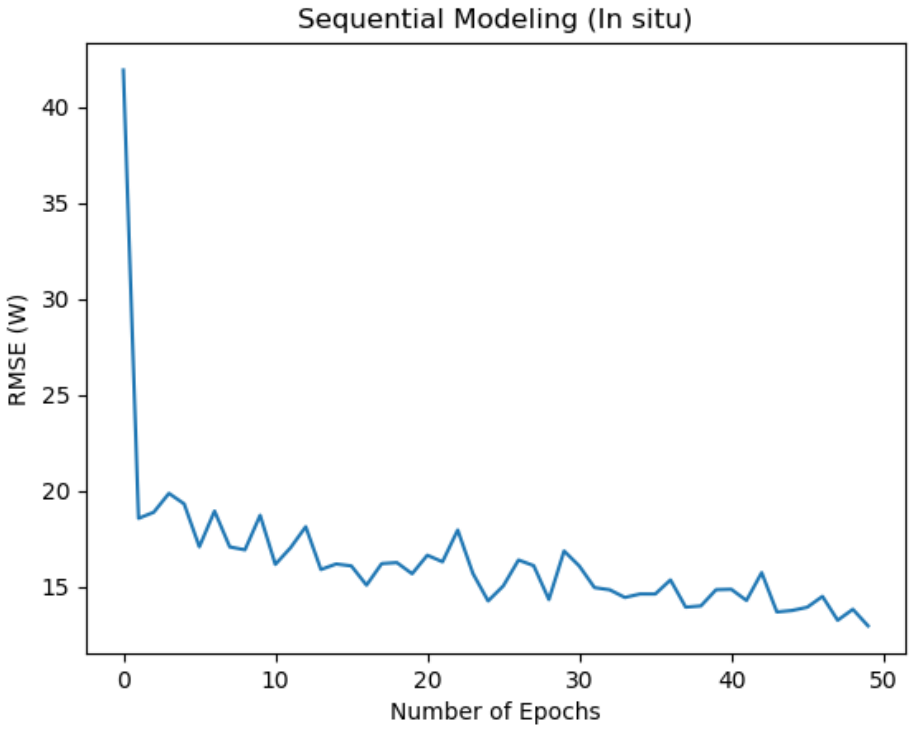

Figure 7: Sequential in situ RMSE over 50 epochs

From these two figures we can see that in situ model performs better than a priori model. Figure 5 shows the results for in situ LSTM model, Predicted mean power vs Real mean power. The results are along the y=x line except for some exceptions.

6 Evaluation

We leveraged RMSE to rate the model. For our problem, we are not looking for either True/False value so measures like accuracy and F1 don't seem like good options. If the real mean power is 39W and if two models predicts 40W and 49W then the model which predicts 40W is the better one, which deciding upon the best model, RMSE is very useful. We used RMSE on 10 fold cross validation data. Based on the results we can conclude that our technique worked for predicting the mean power consumption using Sequential model with one LSTM and 6 dense layers. We achieved results with considerably less RMSE value compared to the one given in the paper.

7 Contribution of team members

Sepideh - Paper reading, Preprocessing, Linear Regression, Ridge Regression, PLS Regression, Lasso Regression, MARS, Random Forest, Adaboost, LSTM, report.

Maziyar - Paper reading, Preprocessing, Linear Regression, Ridge Regression, PLS Regression, Lasso Regression, Random Forest, Adaboost, LSTM, cross validation, poster, report.

Reshma - Paper reading, Preprocessing, Random Forest, poster, report

References

[1] https://onlinelibrary.wiley.com/doi/epdf/10.1002/sam.11339

[2] https://link.springer.com/chapter/10.1007/3-540-44668-0_93

[3] https://cscdata.nrel.gov/#/datasets/d332818f-ef57-4189-ba1d-beea291886eb

[4] S et. al. Bhalachandra. 2017. Improving Energy Efficiency in Memory-constrained Applications Using Core-specific Power Control.Proceedings of the 5th International Workshop on Energy Efficient Super computing(2017).

[5] Janardhanan, Deepak, and Enda Barrett. "CPU workload forecasting of machines in data centers using LSTM recurrent neural networks and ARIMA models." 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST). IEEE, 2017.

[6] Masouros, Dimosthenis, Sotirios Xydis, and Dimitrios Soudris. "Rusty: Runtime System Predictability Leveraging LSTM Neural Networks." IEEE Computer Architecture Letters 18.2 (2019): 103-106.

[7] https://github.com/Maziyar-Na/MachineLearningProjectFall2019/tree/master/Final