Deep Domain Adaptation for Runtime Prediction in Dynamic Workload Scheduler

Hoang H. Nguyen *
University of Illinois at Chicago
Chicago, IL, USA
hnguy7@uic.edu

Ben Matthews
National Center for Atmospheric Research (NCAR)
Boulder, CO, USA
matthews@ucar.edu

Irfan Elahi
National Center for Atmospheric Research (NCAR)
Boulder, CO, USA
irfan@ucar.edu

ABSTRACT

In HPC systems, users' requested runtime for submitted jobs plays a crucial role in efficiency. While underestimation of job runtime could terminate jobs before completion, overestimation could result in long queuing of other jobs in HPC systems. In reality, runtime prediction in HPC is challenging due to the complexity and dynamics of running workloads. Most of the current predictive runtime models are trained on static workloads. This poses a risk of over-fitting the predictions with bias from the learned workload distribution. In this work, we propose an adaptation of Correlation Alignment method in our deep neural network architecture (DCORAL) to alleviate the domain shift between workloads for better runtime predictions. Experiments on both Benchmark workloads and NCAR real-time production workloads reveal that our proposed method results in a more stable training model across different workloads with low accuracy variance as compared to the other state-of-the-art methods.

KEYWORDS

domain adaptation, scheduling, high performance computing, deep learning

ACM Reference Format:
Hoang H. Nguyen, Ben Matthews, and Irfan Elahi. 2019. Deep Domain Adaptation for Runtime Prediction in Dynamic Workload Scheduler .

1 INTRODUCTION

In most High Performance Computing (HPC) systems, users are required to estimate runtime and other resources for their submitted jobs before execution. This helps schedulers allocate adequate resources for the running jobs and backfill other queuing jobs in the systems. These estimations require experience in running jobs on HPC. Both underestimation and overestimation have negative effects on HPC systems. Underestimation which occurs when the job's actual runtime exceeds what users estimate can waste resources and time of the HPC systems since they get killed without generating any results. On the other hand, overestimation can increase queuing time and idleness of the system due to inability of backfilling. Therefore, better runtime predictions cam save time and resources for HPC systems tremendously.

Although there exist machine learning algorithms to tackle this runtime prediction problem, most classifiers are trained on a static dataset. However, workloads on HPC systems are dynamic across different systems and periods of time, requiring robust machine learning models for reliable runtime predictions.

In the meantime, Domain Adaptation (DA) and Transfer Learning (TL) have been growing rapidly. DA is defined as a special case of TL in which labeled data in one or more related source domains are leveraged to learn a classifier for unseen or unlabeled target domain. The main distinction between DA and TL is the assumption of similar category space [2]. In other words, DA assumes both different marginal distribution between source and target data ($P(X_{source}) \neq P(X_{target})$) and similar category space $P(Y_{source}|X_{source}) = P(Y_{target}|X_{target})$ [13]. Despite recent advances in DA with deep networks, they have not been applied in HPC scheduling problems.

2 OUR APPROACH

2.1 Problem Formulation

In this work, we focus on minimizing users' mispredictions of actual running time using Deep Neural Network Architecture. We divide runtime mispredictions into different classes depending on their time range. Specifically, six time periods are taken into considerations: under-prediction (<0m), over-prediction within 15 minutes (<=15m), over-prediction between 15 minutes and 1 hour (15m-1h), over-prediction between 1 hour and 3 hours (1h-3h), over-prediction between 3 hours and 7hours (3h-7h), and over-prediction more than 7 hours (>7h). This becomes a 6-class classification problem.

2.2 Domain Adaptation Formulation

To tackle DA problem, we adapt the successful approach of CORAL [10] for domain alignment in our work, named as DCORAL. This work is novel in HPC community due to its initial attempt in utilizing DA methods for runtime predictions among different workloads. In domain adaptation settings, we denote source labeled data as $D^s = \{X^s, Y^s\} = \{(x_i^s, y_i^s)\}_{i=1}^{n_s}$ and unlabeled target data as $D^t = \{X^t\} = \{x_i^t\}_{i=1}^{n_t}$ where n_s and n_t represent the number of source examples and target examples respectively. Both source and target examples share the same set of features. In other words, x_i^s and $x_i^t \in R^L$ where L represents the learned deep feature dimensions from the bottleneck layers [1]. The overall networks can be

*This work is carried out when the first author is a visiting researcher at NCAR

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

, ,
© 2019 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

trained by minimizing the overall loss function as below:

$$\mathcal{L}(\theta|X_s, Y_s, X_t) = L_s + L_d \quad (1)$$

where L_s, L_d denote source domain loss and domain discrepancy loss respectively

2.2.1 *Backbone architecture.*

We re-implement AlexNet architecture [5] from Image Recognition tasks. The architecture contains 5 convolutional layers (Conv1-Conv5) for feature extraction from source data (bottleneck layers) and 3 fully-connected layers (FC1-FC3) to adapt classification task from source to target domains.

2.2.2 *Source Domain Loss.*

Source domain loss defines the classification loss function in labeled source domain data. c(.) can be cross-entropy loss in our problem.

$$\mathcal{L}_s = \frac{1}{n_s} \sum_{i=1}^{n_s} c(\theta|x_i^s, y_i^s) \quad (2)$$

2.2.3 *Domain Discrepancy Loss.*

Domain Discrepancy Loss measures the discrepancy between source and target distributions which can be learned by either correlation alignment (CORAL), DeepCORAL [11], or Maximum Mean Discrepancy (MMD) [6]. In our experiments, we implement CORAL as a preliminary domain discrepancy measure as follows:

$$\mathcal{L}_d = CORAL(H_s, H_t) = \frac{1}{4L^2} ||Cov(H_s) - Cov(H_t)||_F^2 \quad (3)$$

where Cov(.) denotes co-variance matrix for the source or target output from bottleneck layers. $||.||_F^2$ is the standard Frobenius normalization. H_s and H_t are the feature representations of source and target domains from the bottleneck layers.

3 EXPERIMENTS

3.1 Dataset and Feature Augmentation

We conduct testing our proposed model on real-time production Portable Batch System (PBS) workload [7] from National Center for Atmospheric Research (NCAR) and standard Benchmark datasets. For standard Benchmark dataset, we identify Cornell Theory Center IBM SP2 (CTC) [1] [8], Swedish Royal Institute of Technology IBM SP2 (KTH)[2] [8], and San Diego Supercomputer CenterBlue Horizon (SDSC) [3] [8] as experimental workloads. These workloads are recorded in Standard Workload Format (SWF) [4] [8].

Regarding NCAR dataset, we identify three different historical workloads to conduct adaptation learning method. To avoid shortened number of days in the first and last week of each month, we use Week 2 (M2), Week 3 (M3) and Week 4 (M4) of April 2019.

As NCAR workloads are extracted from PBS logs, all features from PBS accounting logs are extracted since they are rich in representation of submitted jobs. SWF features are extracted for Benchmark dataset. Both of these features are augmented with additional historical features proposed by [3], resulting in 79 features for NCAR dataset and 34 features for Benchmark dataset.

[1] http://www.cs.huji.ac.il/labs/parallel/workload/l$_c tc_s p2/index.html$
[2] http://www.cs.huji.ac.il/labs/parallel/workload/l$_k th_s p2/index.html$
[3] http://www.cs.huji.ac.il/labs/parallel/workload/l$_s dsc_b lue/index.html$
[4] http://www.cs.huji.ac.il/labs/parallel/workload/swf.html

3.2 Evaluation Result

We use Accuracy as DA evaluation metric due to its direct correlation with errors of model predictions on target domains. The state-of-the-art methods include Random Forest (RF), Gradient Boosting Classifier(XGB) proposed by [4], Feed Forward Neural Network (FF) [12], Bidirectional Long Short-term Memory Network (Bi-LSTM)[9] and Convolutional Neural Network (CNN) with AlexNet architecture. Table 1 and 2 report accuracy on the target data through adaptation from source data (Noted as source → *target*)

Table 1: Benchmark Dataset Result

Methods	RF	XGB	FF	Bi-LSTM	CNN	DCORAL
CTC→KTH	0.254	0.089	0.191	0.245	0.193	**0.385**
KTH→CTC	0.462	0.118	**0.528**	0.303	0.276	0.426
CTC→SDSC	**0.292**	0.058	0.092	0.161	0.143	0.241
SDSC→CTC	0.256	**0.611**	0.204	0.21	0.285	0.395
KTH→SDSC	0.563	0.273	0.695	**0.886**	0.287	0.384
SDSC→KTH	0.614	0.159	0.749	**0.799**	0.212	0.511
Average	0.407	0.218	0.410	**0.434**	0.233	0.390

Table 2: NCAR Dataset Result

Methods	RF	XGB	FF	Bi-LSTM	CNN	DCORAL
M2→M3	0.467	0.054	0.389	0.357	0.412	**0.875**
M3→M2	0.396	0.295	0.591	**0.610**	0.211	0.365
M2→M4	0.258	0.041	0.105	0.129	0.295	**0.398**
M4→ M2	0.372	0.326	**0.498**	0.456	0.349	0.383
M3→M4	0.849	0.377	0.772	**0.882**	0.129	0.698
M4→M3	0.747	0.277	0.835	**0.873**	0.505	0.735
Average	0.515	0.228	0.532	0.551	0.317	**0.576**

As indicated in Table 2, DCORAL outperforms other methods in average accuracy across different workloads (2.5% above the second best model in NCAR dataset). From both Table 1 and 2, We could also determine that CORAL loss benefits adaptation from source to target domains as DCORAL performs better than CNN with the same AlexNet architecture in all tasks with improvement gain of 15.7% on Benchmark dataset, 25.9% on NCAR dataset. Based on our result, Bi-LSTM architecture is the best option for backbone architecture as compared to FF and AlexNet.

4 DISCUSSION AND FUTURE WORK

We can further improve the performance of DA methods and reduce computational cost by pre-training backbone architecture from larger workload datasets which are similar to MNIST dataset in Computer Vision. In addition, further discriminative feature learning on the source domain [1] could be elevated to further minimize the discrepancy between target and source domains. Further simulation with more diverse datasets is also needed to verify the effectiveness of these runtime predictions.

ACKNOWLEDGMENTS

We would like to thank you support from Computational and Information Systems Laboratory (CISL) Visitor Program, High-End Services Section (HSS), Supercomputing Services Group (SSG), and Consulting Services Group (CSG) at National Center for Atmospheric Research (NCAR)

REFERENCES

[1] Chao Chen, Zhihong Chen, Boyuan Jiang, and Xinyu Jin. 2019. Joint domain alignment and discriminative feature learning for unsupervised deep domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3296–3303.

[2] Gabriela Csurka. 2017. A comprehensive survey on domain adaptation for visual applications. In *Domain Adaptation in Computer Vision Applications*. Springer, 1–35.

[3] Eric Gaussier, David Glesser, Valentin Reis, and Denis Trystram. 2015. Improving backfilling by using machine learning to predict running times. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 64:1–64:10.

[4] Jian Guo, Akihiro Nomura, Ryan Barton, Haoyu Zhang, and Satoshi Matsuoka. 2018. Machine learning predictions for underestimation of job runtime on HPC system. In *Asian Conference on Supercomputing Frontiers*. Springer, Cham, 179–198.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[6] Mingsheng Long, Yue Cao, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. 2018. Transferable representation learning with deep adaptation networks. *IEEE transactions on pattern analysis and machine intelligence* (2018).

[7] PBSWorks 2019. PBSWorks. https://www.pbsworks.com/

[8] Real Parallel System Workloads Logs 2006. Logs of Real Parallel Workloads from Production Systems. http://www.cs.huji.ac.il/labs/parallel/workload/logs.html

[9] Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45, 11 (1997), 2673–2681.

[10] Baochen Sun, Jiashi Feng, and Kate Saenko. 2016. Return of frustratingly easy domain adaptation. In *Thirtieth AAAI Conference on Artificial Intelligence*. 2058–2065.

[11] Baochen Sun and Kate Saenko. 2016. Deep coral: Correlation alignment for deep domain adaptation. In *European Conference on Computer Vision*. Springer, 443–450.

[12] Daniel Svozil, Vladimir Kvasnicka, and Jiri Pospichal. 1997. Introduction to multilayer feed-forward neural networks. *Chemometrics and intelligent laboratory systems* 39, 1 (1997), 43–62.

[13] Lei Zhang. 2019. Transfer Adaptation Learning: A Decade Survey. *arXiv preprint arXiv:1903.04687* (2019).