Towards Source Selection in Transfer Learning for Cloud Performance Prediction

Hannes Larsson*, Jalil Taghia*, Farnaz Moradi*, and Andreas Johnsson*†
* Ericsson Research, Sweden, Email: {*hannes.larsson,jalil.taghia,farnaz.moradi,andreas.a.johnsson*}*@ericsson.com*
† Uppsala University, Department of Information Technology, Sweden, Email: *andreas.johnsson@it.uu.se*

***Abstract*—Learning performance models for cloud services is challenging due to the dynamics of the operational environment stemming from scaling and migration decisions. This requires exchange or adaptation of the models in order to maintain prediction accuracy over time. Approaches that incorporates previously acquired knowledge using transfer learning is a viable technique for timely and robust model adaptation, especially when the training data is limited.**

In this paper we quantify the impact of different source domains on the accuracy of a target model in another domain. The evaluation is performed using data traces collected from a testbed that runs a Video-on-Demand service and a Key-Value Store under various load conditions. We find that the choice of source domain can yield a transfer gain, and sometimes a substantial transfer penalty.

***Index Terms*—Service Management, Performance Prediction, Machine Learning, Transfer Learning, Source Selection.**

I. Introduction

A promising approach enabling intelligent service management is the use of performance models that can predict and forecast the service quality at the client side based on available observations in the network infrastructure. The ability to learn performance models from observations, using machine learning, simplify management tasks such as service on-boarding, proactive service assurance, and root-cause analysis.

In our recent work [1], we addressed a key challenge in data-driven modelling for dynamic network and cloud environments, namely the difficulty to maintain the model accuracy over time. Cloud services generally rely on a virtualization layer that allows service components to migrate between physical execution environments, and the resources assigned to the service can dynamically be scaled up or down based on operator policies or user requirements. This is exemplified in Figure 1 where the service execution environment changes from one configuration to a second (source and target domains). Such changes reduce the accuracy of a performance model, which has been trained for a specific system configuration and environmental condition. As a consequence, management functions that rely on a performance model are negatively affected, unless the model is appropriately updated. Further, extensive measurements and data collection is usually required for training machine-learning models. The data collection process takes time and the overhead associated with measurements and data collection can adversely affect the service itself and potentially co-located services as well. For certain services, such as short-lived Virtual Network Functions (VNFs), there is often not enough time to gather the data required for training a model.

978-3-903176-32-4 © 2021 IFIP

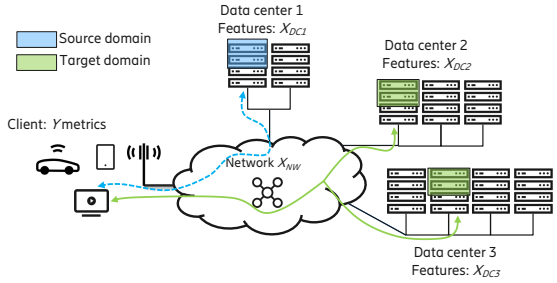

Fig. 1. Prediction of service-quality metrics Y at the client given observations of network and cloud infrastructure state and utilization $\{X_{NW}, X_{DC1..3}\}$. The execution environment changes from source to target domain.

To overcome these challenges, we have proposed and evaluated an approach using transfer learning where the knowledge embedded in a model and learned for one environment (source domain) is transferred to facilitate timely predictions in a changed execution environment (target domain) [1]. Since transfer learning is an approach that aims at incorporating knowledge gained in a source domain into the target domain, *the transferred knowledge from the sources must be relevant to the target domain* in order to avoid negative transfer [2].

This paper addresses this challenge and contributes by providing insights from experimental results quantifying the impact of different source domains on the target domain, with a varying number of available samples. A source domain can either yield improved accuracy in the target domain, or result in *negative transfer*.

II. Problem Setting

Figure 1 outlines the system under consideration, where clients are interacting, over a network, with services that are executing in a data center. For the purpose of this paper, we consider experiments and data traces where clients access two networked services executing in one data center; a Video-on-Demand (VoD) service and a Key-Value Store (KVS) service.

In a series of works [3], [4], and [5], we predicted the service-level metrics Y_t at time t on the VoD and KVS clients, based on knowing the infrastructure metrics X_t. Using supervised machine learning, we developed and evaluated

models $M : X_t \rightarrow \hat{Y}_t$, such that $\hat{Y}_t$ closely approximate Y_t for a given X_t.

In another work [1], we showed that the accuracy of a static performance model M decreases over time if the service execution environment changes, for example, due to the scaling of resources, service migration, changed hardware platform, or other infrastructure dynamics.

In this paper, we use the definition of transfer learning from [6] to formalize the challenges of source selection in transfer learning for dynamic clouds. A domain $D = \{X, P(X)\}$ consists of two components: (1) a feature space X, and (2) a marginal probability distribution $P(X)$, where X corresponds to the infrastructure metrics. Further, a task $T = \{Y, M\}$ consists of two components: a target space Y corresponding to service-level metrics, and an objective predictive model M.

Transfer learning is then defined as follows. Given a source domain D_S and learning task T_S, a target domain D_T and learning task T_T, transfer learning aims at reducing the cost of learning the predictive model M in D_T using the knowledge in D_S and T_S, where $D_S \neq D_T$, or $T_S \neq T_T$. A model that is transferred from D_S to D_T is denoted $M_{S \rightarrow T}$, to separate it from a model M_S or M_T that is trained in isolation either in the source or target domain.

This paper, which builds upon and extends our findings in [1], aims at quantifying how the performance of a target model $M_{S \rightarrow T}$ varies with the choice of source domain D_S, and the number of available samples N_t at time t in D_T.

III. Approach

In this paper, the transfer learning approach builds upon re-training of a neural-network model M, that is transferred from a source domain, inspired by the work in [7]. Such a model consists of an input layer, corresponding to samples X_t, $n-1$ hidden layers $L_1, ..., L_{n-1}$, weights $w_1, ..., w_n$, and an output layer L_n corresponding to Y_t.

The weights w_i for a neural-network model M_S, called the source model, are trained using backpropagation [8] with samples from the source domain D_S. In this paper, the knowledge transfer corresponds to training a model initialized with the weights from the source model M_S and fine tuned using available samples from the target domain D_T . After knowledge transfer, the resulting neural network is denoted $M_{S \rightarrow T}$ and is thus used for service-level metric prediction in D_T. For the purpose of this paper we limit the study to transfer scenarios where $D_S \neq D_T$ while the tasks are identical, i.e. $T_S = T_T$. This corresponds to a transfer scenario where a cloud service is scaled or migrated, while the prediction task remains the same.

We assess the choice of source domain D_S and its impact on model accuracy in D_T by training target models $M_{S \rightarrow T}$ for each source model M_S. For each combination of D_S and D_T, we evaluate the model accuracy given the number of available samples N_t at time t in D_T.

As a baseline, we train a neural network from scratch, with randomly initialized weights, solely using samples from D_T that is available at time t, and this model is denoted by M_T.

TABLE I
Traces used for evaluation.

Trace ID	Service(s)	Load pattern	Target Y	# samples
K1P	KVS	Periodic	*ReadsAvg*	28962
K1F	KVS	Flashcrowd	*ReadsAvg*	19444
K2P	KVS + VoD	Periodic	*ReadsAvg*	26488
K2F	KVS + VoD	Flashcrowd	*ReadsAvg*	24225
V1P	VoD	Periodic	*FrameRate*	37036
V1F	VoD	Flashcrowd	*FrameRate*	36633
V2P	VoD + KVS	Periodic	*FrameRate*	27699
V2F	VoD + KVS	Flashcrowd	*FrameRate*	29151

We use the concept of *transfer gain* $\mathcal{G}$, inspired from [2], to quantify the impact of D_S on $M_{S \rightarrow T}$, which we define as $\mathcal{G} = e_T - e_{S \rightarrow T}$, where e_T and $e_{S \rightarrow T}$ are the model errors for M_T and $M_{S \rightarrow T}$, respectively.

IV. Testbed and data traces

A. Testbed and services

The results in this paper are based on traces obtained from executing experiments in a testbed [5]. This section provides an overview of the experimental infrastructure, and the structure of the data traces.

The testbed consists of a server cluster and a set of clients. The server cluster is deployed on a rack with ten high-performance machines interconnected by a Gigabit Ethernet. On these machines a Video-on-Demand (VoD) service and a Key-Value Store (KVS) service are installed, and can execute in parallel. The *VoD service* uses a modified VLC media player software [9], which provides single-representation streaming with a varying frame rate. The *KVS service* uses the Voldemort software [10].

Two load generators are running in parallel in the testbed, one for the VoD application and another for the KVS application. The VoD load generator dynamically controls the number of active VoD sessions, spawning and terminating VLC clients. The KVS load generator controls the rate of KVS operations issued per second. Both generators produce load according to two distinct load patterns; either a periodic-load or a flashcrowd-load pattern [11].

B. Collected data and traces

This subsection provides a description of the data collected on the testbed, namely the input feature set X as well as the specific service-level metrics Y_{VoD} and Y_{KVS}.

Features X are extracted from the Linux kernels that run on the machines. To access the kernel data, we use System Activity Report (SAR), a popular open-source Linux library [12], which provides approximately 1700 features per server. Examples of such statistics are CPU utilization per core, memory utilization, and disk I/O.

The Y_{VoD} service-level metrics are measured on the client. During an experiment, we capture multiple metrics, but for the purpose of this paper we focus on the number of displayed video frames per second ($FrameRate$).

The Y_{KVS} service-level metrics are also measured on the clients. During an experiment, we capture multiple metrics, but here the focus is on Read Response Time as the average read latency for obtaining responses over a set of operations performed per second ($ReadsAvg.$).

A trace is generated by executing testbed experiments with different configurations where statistics are collected every second; specifically it includes features X, and service-level metrics Y_{VoD} and Y_{KVS}.

The 8 traces used in this paper are summarized in Table I. The trace ID is encoded according to service under investigation (KVS or VoD), number of concurrent services (1 or 2), and load pattern (periodic or flashcrowd load).

V. Evaluation and Results

We apply the evaluation methodology described in Section III to a set of transfer scenarios for VoD and KVS, respectively. The evaluation aims at providing evidence for the need of selecting a good source domain in transfer learning for cloud performance predictions.

A source domain D_S, and its corresponding model M_S, is created for each trace in Table I. A target domain corresponds to a change in the number of services executing on the platform, and/or the distribution of samples $P(X)$ that corresponds to the load pattern for a given trace. We define 8 transfer scenarios, summarized in Table II, based on the traces in Table I. One trace, separately for KVS and VoD, is used to define the target domain D_T while the other three traces constitute 3 separate source domains D_{S1}, D_{S2}, D_{S3}. Note that we limit the evaluation in this paper to scenarios where $T_S = T_T$, for KVS and VoD.

Further, the *Normalized Mean Absolute Error* is used as the metric for quantifying model performance, and is defined as $e = \frac{1}{\bar{y}}(\frac{1}{m}\sum_{t=1}^{m}|y_t - \hat{y}_t|)$, where $\hat{y}_t$ is the model prediction for the measured performance metric y_t, and $\bar{y}$ is the average of the samples y_t of the test set of size m.

We manually reduced the number of features to 18 using domain knowledge as proposed in [13]. Manual feature selection across traces ensures an identical feature space, enabling homogeneous transfer [14], for source and target domains.

For the purpose of this evaluation, we have selected a neural-network model architecture with an input layer with 18 nodes corresponding to the features, 3 hidden layers with 256 nodes each, and one output layer. The same architecture is applied for both VoD and KVS traces. Note that the ambition is not to find the optimal neural-network architecture, but rather to evaluate the impact of source domains.

To obtain the source models $\{M_{S1}, M_{S2}, M_{S3}\}$ for each scenario, the neural networks are initialized with random weights and are trained on samples from the source domains. For training, the source domains are reduced to the same size of 16000 samples randomly split into training (80%) and validation (20%) sets. The model and its weights are then transferred for tuning and predictions in the target domain. Similarly, in the target domain the samples are split into training, validation, and test sets. The training set is varied between 10 and 100 samples to emulate the shortage of samples in the target domain, and is used for either re-training the transferred source model $M_{S\to T}$ or to train a new target model M_T. The test set always corresponds to 20% of the samples in the trace, and the remaining samples are used for the validation set. Note that the validation set is used for early stopping and the test set is used for evaluation.

For the implementation of the neural networks, Keras library [15] running on top of TensorFlow [16] was used. We used the rectified linear unit (ReLU) activation function for all the layers, Adam optimizer [17] with a learning rate starting at 0.001 using exponential decay with decay rate 85, 1000 decay steps and staircase function, and mean absolute error (MAE) as the loss function. Each experiment was run for a maximum of 200 epochs, with early stopping using a patience of 10 epochs and weight restoring, with a batch size of 32.

We quantify the impact of source domains on the error of $M_{S\to T}$ for each scenario described in Table II. For each scenario, we select 10, 25, 50, 75 and 100 samples from the target domain. Using the selected target samples, we train a baseline model M_T and finetune the transferred models $M_{S_i\to T}$ for each source D_{S_i}. The data from the source domain is normalized, and the target domain data used for the fine tuning is scaled using the same transformation. For training the baseline model M_T, the selected target samples are normalized according to the full target data set. Once the models are trained, the transfer gain $\mathcal{G}$ can be computed by evaluating $M_{S_i\to T}$ and M_T on the same test set. This process is repeated 25 times with different randomly chosen target samples for each target sample size.

The evaluation results are shown in Figure 2, where each graph plots the transfer gain versus number of available samples in the target domain, for different source domains.

The results for transfer scenarios 1 - 4, corresponding to the task of predicting *ReadAvgs* for KVS, are shown in the graphs of the upper row in Figure 2. In scenario 1 and 2, corresponding to single-service target domains, transfer learning always give a transfer gain $\mathcal{G} > 0$. The transfer gain is reduced when the number of target-domain samples increases, which corroborates our observations in [1]. In scenarios 3 and 4, where the target domain corresponds to a co-located service execution environment, the choice of source domain becomes more important. Selecting a source domain with co-located services, that is D_{S3} in scenario 3 and D_{S3} in scenario 4, gives $\mathcal{G} > 0$, whereas a negative transfer gain is observed for the other sources. The difference in transfer gain in the worst case corresponds to an NMAE of approximately 0.5. For source domains yielding $\mathcal{G} < 0$, we note that an increase in target-domain samples reduces the negative transfer gain.

The evaluation results for transfer scenarios 5 - 8, corresponding to the task of predicting *FrameRate* for VoD, are shown in the graphs of the lower row in Figure 2. The results for single-service target domains in scenario 5 and 6 are similar to what was observed in scenario 1 and 2. However, the penalty of selecting the wrong source for the target domains with co-located services, corresponding to scenarios 7 and

TABLE II

THE TRANSFER SCENARIOS STUDIED IN THIS WORK. D_{Si} CORRESPONDS TO THE SOURCE DOMAINS EVALUATED FOR THE TARGET DOMAIN D_T. THE TASK FOR SCENARIOS 1 - 4 IS READSAVG AND FOR SCENARIOS 5 - 8 FRAMERATE. NOTE THAT $T_S = T_T$.

Scenario	$D_{Si}, i = 1..3$									D_T		
	D_{S1}			D_{S2}			D_{S3}					
	Trace	Service	$P(X_S)$	Trace	Service	$P(X_S)$	Trace	Service	$P(X_S)$	Trace	Service	$P(X_T)$
1	K1F	KVS	Flash	K2P	KVS+VoD	Periodic	K2F	KVS+VoD	Flash	K1P	KVS	Periodic
2	K1P	KVS	Periodic	K2P	KVS+VoD	Periodic	K2F	KVS+VoD	Flash	K1F	KVS	Flash
3	K1P	KVS	Periodic	K1F	KVS	Flash	K2P	KVS+VoD	Periodic	K2F	KVS+VoD	Flash
4	K1P	KVS	Periodic	K1F	KVS	Flash	K2F	KVS+VoD	Flash	K2P	KVS+VoD	Periodic
5	V1F	VoD	Flash	V2P	VoD+KVS	Periodic	V2F	VoD+KVS	Flash	V1P	VoD	Periodic
6	V1P	VoD	Periodic	V2P	VoD+KVS	Periodic	V2F	VoD+KVS	Flash	V1F	VoD	Flash
7	V1P	VoD	Periodic	V1F	VoD	Flash	V2P	VoD+KVS	Periodic	V2F	VoD+KVS	Flash
8	V1P	VoD	Periodic	V1F	VoD	Flash	V2F	VoD+KVS	Flash	V2P	VoD+KVS	Periodic

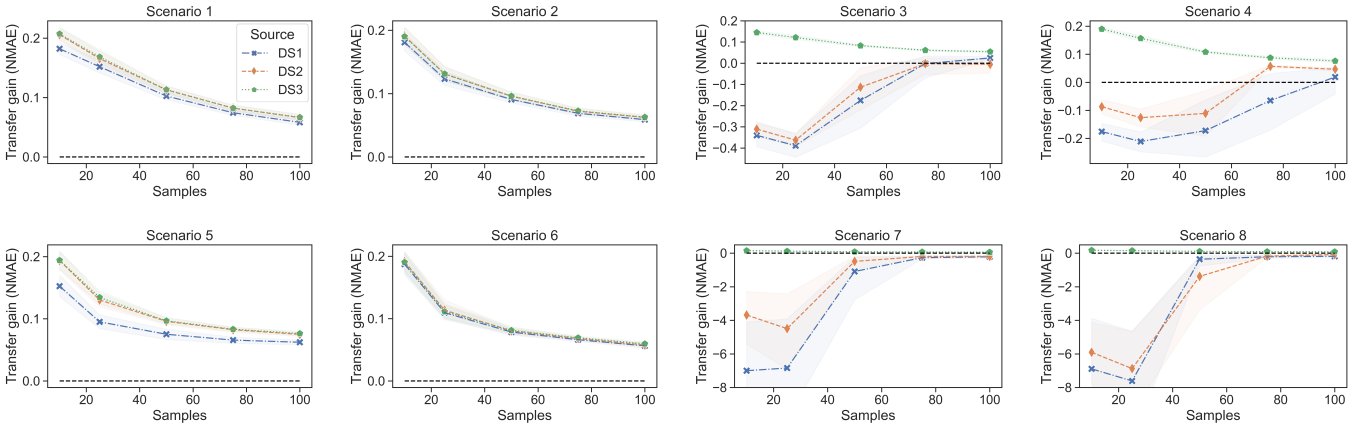

Fig. 2. Transfer gain $\mathcal{G}$ (NMAE) given source domains D_{S1}, D_{S2}, D_{S3} versus sample size in D_T for 8 different target domains. See Table II for details regarding source and target domains for each scenario. The legend in Scenario 1 applies for all scenarios.

8, is striking. The difference in $\mathcal{G}$, in terms of NMAE, is in the worst case almost 7.5. Also, for these scenarios, the benefits and penalties of transfer learning are reduced with an increasing number of samples.

In summary, the benefit of choosing a good source is notable especially when few samples are available in the target domain. Further, the results show that the risk of obtaining a significant penalty in performance also decreases with a growing number of target-domain samples. It is evident that intelligent soure selection in transfer learning is essential for performance modeling of services executing in a dynamic cloud environment. Proposals and evaluation of approaches for source selection is part of future work.

VI. CONCLUSIONS

In this paper, we identified the need for automated source selection in transfer learning for improved performance modeling of dynamic cloud services. We evaluated the impact on the target domain given different source domains in multiple scenarios with realistic data sets obtained from a testbed for two different services under varying load, namely a Video-on-Demand and a Key-Value Store service.

We provided empirical evidence showing that the transfer gain is highly dependent on the source domain. In scenarios where the target domain constitutes a complex shared service environment we observe a significant penalty from selecting the wrong source. The positive impact of transfer learning, and also the penalty, is reduced with an increased number of target-domain samples.

Future work will focus on more challenging transfer scenarios, and novel approaches for automated source selection in transfer learning.

ACKNOWLEDGMENT

The authors are grateful to Rolf Stadler, Forough Shahab Samani, and Masoumeh (Azin) Ebrahimi at KTH for fruitful discussions. This research has been supported by the Swedish Governmental Agency for Innovation Systems, VINNOVA, through projects ITEA3 AutoDC and Celtic ANIARA.

REFERENCES

[1] F. Moradi, R. Stadler, and A. Johnsson, "Performance prediction in dynamic clouds using transfer learning," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2019, pp. 242–250.

[2] Z. Wang, Z. Dai, B. Póczos, and J. Carbonell, "Characterizing and avoiding negative transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 293–11 302.

[3] R. Yanggratoke, J. Ahmed, J. Ardelius, C. Flinta, A. Johnsson, D. Gillblad, and R. Stadler, "Predicting real-time service-level metrics from device statistics," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2015, pp. 414–422.

[4] ——, "Predicting service metrics for cluster-based services using real-time analytics," in *2015 11th International Conference on Network and Service Management (CNSM)*. IEEE, 2015, pp. 135–143.

[5] ——, "A service-agnostic method for predicting service metrics in real time," *International Journal of Network Management*, vol. 28, no. 2, p. e1991, 2018.

[6] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[7] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[8] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics New York, 2001, vol. 1, no. 10.

[9] "Vlc, 2016. [online]. available: http://www.videolan.org/vlc/."

[10] "Voldemort, 2016. [online]. available: http://www.project-voldemort.com/voldemort/."

[11] I. Ari, B. Hong, E. L. Miller, S. A. Brandt, and D. D. Long, "Managing flash crowds on the internet," in *11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems, 2003. MASCOTS 2003.* IEEE, 2003, pp. 246–249.

[12] "Sar, 2016. [online]. available: http://linux.die.net/man/1/sar."

[13] J. Ahmed, T. Josefsson, A. Johnsson, C. Flinta, F. Moradi, R. Pasquini, and R. Stadler, "Automated diagnostic of virtualized service performance degradation," in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2018, pp. 1–9.

[14] C. Shi, Y. Li, J. Zhang, Y. Sun, and S. Y. Philip, "A survey of heterogeneous information network analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 1, pp. 17–37, 2016.

[15] "Keras, 2018. [online]. available: https://keras.io/."

[16] "Tensorflow, 2018. [online]. available: https://github.com/tensorflow/tensorflow."

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.