Machine Learning Data Center Workloads Using Generative Adversarial Networks*

Boudewijn R. Haverkort
Tilburg University
The Netherlands
B.R.H.M.Haverkort@uvt.nl

Felix Finkbeiner
SeMI Technologies
The Netherlands
fe_fi@gmx.net

Pieter-Tjerk de Boer
University of Twente
The Netherlands
p.t.deboer@utwente.nl

ABSTRACT

In this paper we study the applicability of generative adversarial networks (GANs) for the description and generation of workloads for data centers. GANs are advanced neural networks that can learn complex likelihood functions and can sample from them. The field of workload modeling is concerned with describing and generating realistic workloads for performance evaluation of computer systems, in this paper, specifically for data centers. The characterization of the workload of modern data centers is crucial in order to study the effect of changing workloads on the performance of such data centers.

Previously, a number of statistical fitting techniques have been used to characterize data center workloads. This paper explores whether GANs are sufficiently capable to automatically learn such characterisations from multidimensional data sets. We describe the design and evaluation of a GAN, thereby using real-world data center traces. The learned model is evaluated by comparison to previously proposed fitting techniques.

We find that the resulting GAN is very well able to reproduce a realistic data center workload. Furthermore, the approach does not require (a priori) knowledge or assumptions about the underlying models themselves, which can be seen as an advantage. It is shown that the learning approach does reach comparable quality to other fitting techniques, although still at much higher computational costs.

Keywords

Data centres, Generative adversarial networks, Machine learning, Performance evaluation, Workload modelling

1. INTRODUCTION

Workload modelling has been an important topic in computer performance evaluation since the 1960's [1]. The idea of workload modelling is to find a compact (abstract) representation of the workload issued to a computer or communication system, in order to be able to use such representation in the validation of the performance of the system under study, given that workload. Typical initial workload models described the distribution of time between arrivals of jobs (or packets) and the required computational requirements (or packet lengths) of the arriving jobs; in such initial models the inter-arrival times as well as the job lengths were assumed to be mutually independent. For analytical performance analysis methods, a preference has long existed for models with negative exponential inter-arrival as well as job length distributions, as these lead naturally to Markov models, for which often nice closed-form solutions for the system performance can be derived. Next to analytical performance models, also for simulation purposes workload models are important to generate artificial workload traces. Over the years, workload models have become more intricate, that is, more detailed for particular purposes, e.g., by including correlations between successive arrival times and/or job service times, or including multi-mode behavior.

A challenge in the characterisation of workloads lies in the fact that effective fitting procedures require an *a priori* choice of the type of model, e.g., an Erlang or a Normal distribution, so that the appropriate parameters can be estimated. To overcome that challenge, we have turned our attention to machine learning to come up with effective descriptive models based on sample data. In particular, in this paper we propose to use so-called *generative adversarial networks*, a class of neural networks, to describe the service time distribution of a data centre. We use a number of data traces from an operational small-scale data center; in previous work we have used the same data and proposed a mixture distribution to describe it. In this paper we compare our previous approach with the new approach based on machine learning, both in terms of computational costs and in terms of quality.

This paper is further organised as follows. In Section 2 we describe the system we are addressing and sketch the data set considered. We then introduce generative adversarial networks (GANs) and briefly describe apprpriate learning algorithms Section 3. We then present a small selection of experimental results in Section 4, before concluding in Section 5

2. SYSTEM AND DATA SET

The main data set considered in this thesis was provided by BetterBe.[1] This SME provides software-as-a-service solutions for the automotive leasing industry. For this service, the company runs its own, small-scale redundant data center. The data set consists of HTTP(S) requests received at the company's data center. Features captured for every request include: (i) the time the request was received, (ii) a customer ID, (iii) an internal cluster host identifier, (iv) the HTTP request method, (v) the response size, (vi) the HTTP status code, (vii) the time elapsed from the arrival of the request until the response was send, and (viii) the time spend on processing

*This work was performed while Felix Finkbeiner was a visiting master student from Karlsruhe Institute of Technology at the University of Twente, September 2018–May 2019. At that time, Boudewijn Haverkort was also still employed at the University of Twente. The visit from Felix Finkbeiner was supported through the Erasmus+ program.

MAMA 2020 Boston, MA USA

[1] `https://www.betterbe.com/`; last viewed May 15, 2020.

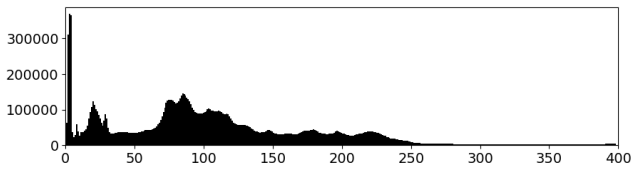


Figure 1: Histogram of service time using the cleaned data set and 400 buckets.

Measure	Response size	Service time	Interarrival time
mean	181273.45	106.34	89.94
σ	614917.41	74.40	287.34
c_v^2	11.507081	0.489	10.206
Minimum	0.00	0.00	0.00
Q25%	12438.00	56.00	36.00
Q50%	40232.00	95.00	94.00
Q75%	152766.00	147.00	8.00
Q97.5%	876884.00	269.00	425.00
Maximum	7872283.00	399.00	33313.00

Table 1: Moments of all ration scale features

the request.

The data set consists of two traces, captured in August 2016 and September 2016, contain 7638309 and 19687179 records, respectively; these data sets were also used in [8] to demonstrate a new Maximum Likelihood Estimation with Refinement Algorithm for fitting the processing duration (feature (viii) above), which will also serve as reference in our evaluation in Section 4.

The analysis process starts with the processing, cleaning and understanding of the data. Just like most data collections it is not unusual for workloads to require some kind of cleanup before they can be used, see e.g., [2]; think of corrupted or incomplete measurements, outliers, or otherwise unusable measurements, e.g., due to a server failure or wrongly used URL [3].

The BetterBe data set contains a few very long running jobs which are not representative for the vast majority of the requests. These very long service times are caused by ongoing garbage collection and maintenance activities and are thus omitted. It should be pointed out that these activities are real and should be considered for an overall system performance evaluation, however, due to their extreme nature they skew the data set so massively that it is best to consider them in a separate model. Note that also in [8] these jobs were omitted; this also makes it easier to compare results. The neglected jobs account for about 7% of the so far refined data set. Figure 1 depicts a histogram of the cleaned workload, showing a clear multimodal distribution.

3. MACHINE LEARNING & GANS

Neural networks have been known since the 1960s, however, their popularity increased since 2000, when it became possible to train large neural networks within reasonable time. Scientists explored the properties of so-called deep neural networks, which contain many neuron and layers between input and output layer. Neural networks have been applied in many fields but we are not aware of extensive use of neural networks for computer workload characterisation.

The goal within machine learning is to find a hypothesis function $H(x)$ that produces some prediction vector $\hat{y}$, that lies as close as possible to the real value y, given an input vector x; Figure 2b visualizes this. In a regression task, x contains several features

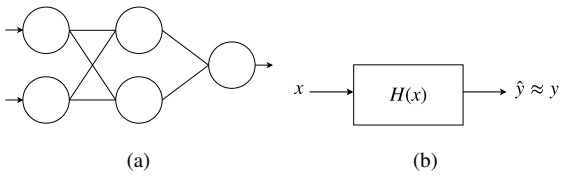


Figure 2: A neural network with five perceptrons (left); it can be used as Hypothesis function $H(x)$ (right), in order to find a good approximation of y.

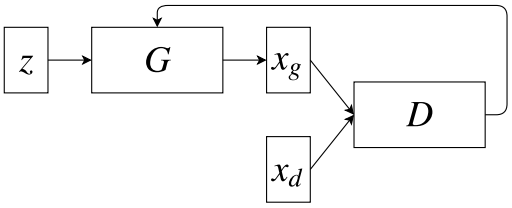


Figure 3: A Generative Adversarial Network: two neural networks "playing against each other"

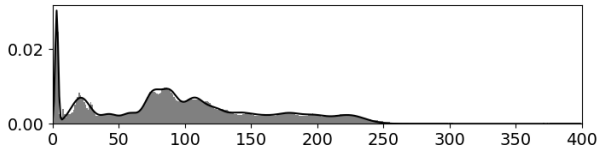
whose combination can predict a value close to y.

Neural networks can approximate $H(x)$ by using a learning algorithm, such as stochastic gradient descent. The result is a model that can quickly predict y after the training is complete. The speed of learning depends on the network structure but in a plain feed forward network the speed is roughly $O(n^l)$ for a network with n neurons and l layers. Neural networks have proven very successful for approximating complex functions in high-dimensional spaces. The attempts to use neural networks for data generation, however, have been less successful for a long time. This problem is tackled by generative adversarial networks (GANs) [6]. GANs combine two neural networks that play against each other. The idea of using programs to train each other goes back to the absolute beginnings of machine learning; Samuel let two programs learn Checkers by letting them play against each other [5] already in 1951. Figure 3 visualizes a GAN: the generator neural network G, is receiving random noise z and maps it in the data space X_{g} (for generated). It tries to learn the distribution function of the data and generates new synthetic data. The generated data from G and real training data X_{d} serve as input for the second network, the so-called discriminator D. D then learns to distinguish between real data, X_{d}, and synthetic data X_{g}. The generator learns to manipulate random noise from z into data that fools the discriminator into classifying it as real data. The two networks create a feedback loop between each other. The learning process is iterative; we used PyTorch, a library built with C++, but fully integrated in Python, cf. `https://pytorch.org/`. For the parameter fitting in the learning process, the gradient-based optimization algorithm ADAM has been used [7].

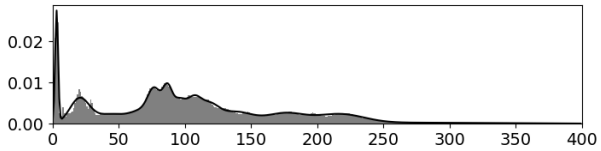
4. EXPERIMENTAL RESULTS

We have performed a large number of experiments, both using well-structured synthetic data traces, and the BetterBe data set. We focus on the service time distribution here. We found, in general, by using only the smallest 97.5% of the data set, thus omitting extreme values. In Figure 4 we show the results obtained with the GAN (bottom) as well as the result obtained with maximum likelihood estimation with refinement (top, cf. [8]) and a model-based clustering algorithm (middle, cf. [9]). As can be observed, the re-

Maximum likelihood estimation (MLE) with refinement.



Model based clustering (MBC) with thirteen clusters, based on the amount of clusters MLE with refinement found.



GAN trained on 97.5% quantile.

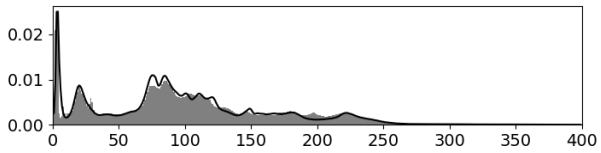


Figure 4: Fitting of the service time (in msec) with multiple normal distributions; the normalized histogram of the original data is plotted in grey, the fitted function is plotted as a black line.

Method	N=400	N=1000
MLE with refinement	3.3964E-4	3.7401E-3
MBC 13	3.0531E-4	3.7216E-3
GAN all	6.9638E-4	3.8479E-3
GAN 97.5%	3.5580E-4	3.7595E-3

Table 2: The square error SE for different bin sizes and fitting techniques.

sults all look very similar.

To compare the results we also used distance metrics, which work well with big data sets. Following [8], we use the square error distance metric $\text{SE} = \sum_{i=1}^{N}(\text{generated}_i - \text{expected}_i)^2$. It is calculated by generating a histogram with N bins from the generated data and the original data. Table 2 shows the results for the different algorithms; the MBC with 13 clusters gives the best results. The GAN trained on all the data produces the worst results, that based on 97.5% performs better.

To give an idea of the complexity of the trained GANs, both the generator and the discriminator are neural networks with 5 layers, with as many as 64 to 512 neurons per layer, hence, the number of weights to be estimated/learned is enormous. The learning process required several million iterations.

So far, all experiments assumed the workload was constant, not varying in time, while in reality there are, e.g., diurnal patterns. GANs offer an interesting way of including this in the model, by adding clock time as an extra feature into both the generator and the discriminator. The result is a so-called conditional GAN, see Figure 5. Initial experiments with this on synthetic data traces are promising [4], but doing this on the real data is currently computationally prohibitive.

5. CONCLUSIONS

Looking back at all our experiments, we conclude that the learning of GANs can compete with more traditional statistical fitting techniques in terms of achieved statistical quality of the generated workloads. However, the more specialized statistical techniques are much faster (a factor 100-1000) than the generic learning techniques for GANs. On the other hand, GANs allow for the fitting of multiple features of the unknown workload at the same time, e.g., including correlations or joint fitting of inter-arrival and service time (we did not report on this here), which makes them more versatile. Also, the learning process of the GAN does not use a priori knowledge about the distribution; it just uses the data available and learns. It is especially these generic properties that make GANs promising candidates for future workload modeling studies. Furthermore, at this point, the parameters estimated with the classical fitting techniques, allow for a direct (statistical) interpretation, thus give more insight in what is actually going on. Further study is needed to combine the advantages of both methods. Many more results using GANs are reported in [4].

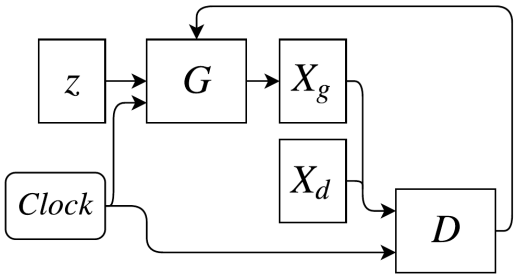


Figure 5: Using a Conditional GAN for modelling a time-varying workload

6. REFERENCES

[1] M. Calzarossa and G. Serazzi. Workload characterization: a survey. *Proceedings of the IEEE*, 81(8):1136–1150, 1993.

[2] D. G. Feitelson. Workload modeling for performance evaluation. In M. C. Calzarossa and S. Tucci, editors, *Performance Evaluation of Complex Systems: Techniques and Tools*, pages 114–141. Springer, 2002.

[3] D. G. Feitelson, D. Tsafrir, and D. Krakov. Experience with the parallel workloads archive, 2012.

[4] F. Finkbeiner. Modeling of Datacenter Workloads using Generative Adverserial Networks. Master's thesis, Karlsruhe Institute of Technology & University of Twente, 2019.

[5] E. F. Gio Wiederhold, John McCarthy. Memorial resolution Arthur L. Samuel, 1990.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[7] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[8] B. F. Postema, N. J. Geuze, and B. R. Haverkort. Fitting realistic data centre workloads: A data science approach. In *Proceedings ACM e-Energy*, pages 486–491. ACM, 2018.

[9] L. Scrucca, M. Fop, T. Murphy, and A. Raftery. MCLUST: 5 Clustering, Classification and Density Estimation Using Gaussian Finite Mixture Models. *The R Journal*, pages 205–233, 08 2016.