Performance and Energy Savings Trade-Off with Uncertainty-Aware Cloud Workload Forecasting

Diego Carraro, Andrea Rossi, Andrea Visentin, Steven Prestwich and Kenneth N. Brown
Insight Centre for Data Analytics and Centre for Research Training in Artificial Intelligence
School of Computer Science, University College Cork, Ireland
diego.carraro@insight-centre.org, a.rossi@cs.ucc.ie, andrea.visentin@ucc.ie, s.prestwich@cs.ucc.ie, k.brown@cs.ucc.ie

***Abstract*—Cloud managers typically leverage future workload predictions to make informed decisions on resource allocation, where the ultimate goal of the allocation is to meet customers' demands while reducing the provisioning cost. Among several workload forecasting approaches proposed in the literature, uncertainty-aware time series analysis solutions are desirable in cloud scenarios because they can predict the distribution of future demand and provide bounds associated with a given service level set by the resource manager. The effectiveness of uncertainty-based workload predictions is normally assessed in terms of accuracy metrics (e.g. MAE) and service level (e.g. Success Rate), but the effect on the resource provisioning cost is under-investigated. We propose an evaluation framework to assess the impact of uncertainty-aware predictions on the performance vs cost trade-off, where we express the cost in terms of energy savings. We illustrate the framework's effectiveness by simulating two real-world cloud scenarios where an optimizer leverages workload predictions to allocate resources to satisfy a desired service level while minimizing energy waste. Offline experiments compare representative uncertainty-aware models and a new model (HBNN++) that we propose, which predict a cluster trace's GPU demand. We show that more effective uncertainty modelling can save energy without violating desired service level targets and that model performance varies depending on the specific details of the allocation scheme, server and GPU energy costs.**

***Index Terms*—Cloud Computing, Energy Saving, Deep Learning, Workload Prediction, Uncertainty, Time Series Forecasting**

I. Introduction

Cloud computing has seen widespread adoption because it increases the productivity and efficiency of industries and allows for effective scalability of their business [1]. Guaranteeing performance levels is at the core of cloud services and requires huge computational resources, especially with the latest advances in technologies such as Artificial Intelligence and the Internet of Things [2]. Typically, customers subscribe to agreements where cloud providers ensure specific levels of reliability, availability and responsiveness to systems and applications and describe penalties if the service levels are not met. At the same time, massive computational resources are a cost for providers and have a significant environmental impact, which will increase in the future. It is estimated that the energy consumption of data centres (which host cloud services) will grow from 292 TWh in 2016 to 353 TWh in 2030 [3], and greenhouse gas emissions will increase over 14% in 2040, compared to a 1-1.6% increase in the 2007-2016 [4].

979-8-3503-0322-3/23/

This scenario poses new challenges for cloud providers because they must optimise a trade-off between serving an increasing number of diverse customers while reducing the environmental impact and maintenance cost. Accurately predicting cloud workload is an essential precondition to meet such optimization goals. Indeed, based on predictions, cloud managers can allocate resources dynamically and set up advanced configurations on the available computing machines. For example, they can intelligently plan in advance workload execution in low-carbon electricity time shifts or spatial regions. Another option is activating specific power management policies for resources such as CPU and GPU [5], [6], or simply increasing/decreasing the number of resources available.

Workload forecasting is challenging due to workload patterns being highly irregular [7]. The literature tackles the problem as a time series analysis task and has proposed state-of-the-art solutions using statistical, machine learning and deep learning approaches. These models often provide point estimate predictions for average future workload demand within a limited time window (typically five or ten minutes). However, recent work (e.g. [8], [9]) has proposed models that can provide uncertainty measures for such predictions, which tell how confident a model is about a prediction. Uncertainty can help resource managers make more informed decisions when optimizing the performance vs energy trade-off, approaching the problem based on the degree of a prediction's confidence. For example, when a workload demand with low uncertainty is predicted, a proportion of the extra resources allocated to meet the predicted demand (i.e. the safety margin) can be placed in a low power state with minimal energy consumption (e.g. sleep state). When the uncertainty of predictions is high, the safety margin resources might be set to a more responsive state (e.g. idle state), sacrificing energy efficiency.

Motivated by these considerations, in this paper, we extend the scope of [8] by investigating the trade-off between performance and energy savings in cloud computing when using uncertainty-aware workload forecasting models. In particular, we design an evaluation framework that tailors uncertainty-aware workload predictions with pre-defined target service levels and uses the predictions to allocate resources to cover the workload. To assess the effectiveness of our framework, we implement two simulations that resemble real-world cloud scenarios. These simulations compare representative state-of-the-art uncertainty-aware models for predicting a GPU work-

load trace and a new model (HBNN++) proposed here. Results show that more effective uncertainty modelling saves energy without violating the desired service level targets and that model performance varies depending on the specific details of the allocation scheme, the server and GPU energy costs.

In the remainder of the paper, section II reviews related work; section III describes our evaluation framework; section IV presents the models, and section V reports the experiments and their results based on our framework. Finally, we draw conclusions and outline future directions in section VI.

II. Related Work

Predicting future workload in cloud computing has been widely studied for more than twenty years, based mainly on time series modelling [10]–[14]. More recently, probabilistic forecasts have been proposed (e.g. [15], [16]) to help resource scheduler decisions, by modelling uncertainty around a point prediction while still offering high accuracy. In machine learning, uncertainty is often categorised as either epistemic or aleatoric [17]. Epistemic refers to the uncertainty of a model caused by insufficient training data, while aleatoric relates to the inherent stochasticity of observations due to noise and randomness. Attempts have been made to capture such uncertainties in the workload prediction task (e.g. [15], [18]), and specific layers in Deep Learning (DL) models have been designed to capture these uncertainty aspects [8], [9], but the topic is still under-investigated. Previously [8], we explored state-of-the-art deep learning approaches for CPU and memory demand prediction, including epistemic and aleatoric uncertainty, using an LSTM architecture with a Bayesian layer.

Focusing on the aim of reducing energy consumption, [9] designs a framework to predict GPU cluster workloads and uses the uncertainty of predictions to optimise a trade-off between performance and energy using quantile regression. The investigation in this paper is inspired by [9] and built on our previous work [8]. In particular, in [8], we introduce several elements that we use in this paper: (i) we propose HBNN and C-LSTM to forecast future workload; (ii) we provide a way of estimating uncertainty in their prediction; (iii) and we preprocess the datasets and the methodology we use in our experiments.

III. Evaluation framework

Our evaluation framework assumes the existence of N GPUs (of different specifications) distributed across several server machines I as resources available to cover workload demand. In particular, we consider *running* and *sleeping* states for a server machine. When sleeping, the server does not incur any energy cost, but its GPUs are not available for use. When running, its GPUs are available to cover the incoming workload demand, but there is an energy cost to keep the server awake. There is also the possibility of waking up a server, again with an energy cost. We consider three states for a GPU: *sleeping*, *idling* and *running*. When sleeping, GPUs do not incur any energy cost and are available for incoming workload with a negligible delay. When idling, GPUs consume a reduced amount of energy and are immediately available to cover the incoming workload. When running, GPUs cover a specific amount of workload and consume an energy cost depending on their specification.

We assume a workload predictor which predicts the average workload in a given time period $t \in T$ of length 5 minutes (T is the set of periods where the workload is predicted). The prediction is provided 10 minutes in advance to allow for resource allocation planning. An independent allocator then plans the resource provisioning to meet a given service level target (SLT) in the period t and decides which servers and GPUs to run within the time window to cover the demand. If a required server is already running, there is no additional energy cost; if it is sleeping, it is woken up one minute before the period t starts and incurs the energy cost. When allocating the workload within t (based on the real demand), it is unlikely that all GPUs will run at maximum capacity (individual tasks can not be broken down, there might be memory or data bottlenecks, etc.). Therefore, we consider an α utilization level that is the average utilization of GPUs when running (which also affects the energy consumption accordingly). Once the server runs, GPUs can be assigned to the incoming workload demand. Within t, we assume we cannot change the state of servers and GPUs based on real-time demand because the only prediction available is an average over the entire 5 minutes window; we can only prepare them for the next period. Other parameters of the framework are:

- c_i: energy consumption required to run server machine $i \in I$ for the entire period.
- b_i: energy consumption required to wake up server machine i (previously sleeping).
- e_n: energy consumption required to run the GPU $n \in N$ for the entire period.
- w_n: maximum GPU units per period offered by GPU n.
- h_{ni}: binary parameter that takes value 1 if the GPU n is deployed in server machine i, 0 otherwise.

At each period t, the allocator must allocate resources to cover the given workload demand W_t, while minimizing its energy cost. The decision variables of the allocator are:

- x_{it}: binary variable that takes value 1 if the server machine i is running on period t, 0 otherwise.
- z_{nt}: binary variable that takes value 1 if the GPU n is running on period t, 0 otherwise, i.e. whether some workload is assigned to n.

We also need an auxiliary binary variable y_{it} that takes value 1 if the server machine i sleeps in period $t-1$ and must be woken up to run in period t, 0 otherwise. The optimization problem can be described as follows:

$$\min \quad \sum_{t\in T}(\sum_{n\in N}\alpha z_{nt}e_n + \sum_{i\in I}(x_{it}c_i + y_{it}b_i)) \tag{1}$$

$$\text{s.t.} \quad W_t \leq \sum_{n\in N}\alpha w_n z_{nt} \quad \forall t \in T \tag{2}$$

$$y_{it} + x_{i(t-1)} \geq x_{it} \quad \forall i \in I, \forall t \in T \tag{3}$$

$$z_{nt} \leq \sum_{i\in I} h_{ni}x_{it} \quad \forall n \in N, \forall t \in T \tag{4}$$

where Eq. 2 guarantees enough GPU computational power is allocated to meet demand W_t. Eq. 3 ensures that if a server i runs in period t, i.e. $x_{it} = 1$, either it was already running in $t-1$, i.e. $x_{i(t-1)} = 1$, or it wakes up to run in the current period, i.e. $y_{it} = 1$. Lastly, Eq. 4 checks that a GPU is running only if its server machine is running too. The model can be solved optimally using a Mixed Integer Linear Programming (MILP) solver, which finds the expected energy cost if W_t is a predicted workload or an approximation of the actual energy cost if W_t is the real demand. In practice, the allocator's must find a set of GPUs that are enough to service the upcoming request using the minimum amount of energy. The cost parameters and the constraints can be adapted to approximate the specific cluster system.

While this general formulation models a multi-period horizon, the forecasting approaches we consider do not provide multi-step ahead predictions but focus on single-period predictions instead (section IV). Therefore, we optimize allocation in each period t independently and consider the previous states $z_{i(t-1)}$ as problem parameters instead of decision variables.

The high number of binary variables (due to numerous GPUs and servers) makes it expensive for MILP to solve the problem. Comparing different models on thousands of predictions requires considerable computational effort. To overcome this issue, we deployed a heuristic based on Dantzig knapsack problem algorithm [19] that fixes which servers should be running to satisfy the predicted demand. This approach makes the assumption that all the GPUs in a server machine are running workload, or none of them is, so there is no server using only part of their GPUs. The computational power a server provides can be seen as the volume, while the cost of waking and running it is the value/reward efficiency. The goal is to fill the knapsack while minimizing the total cost/reward. Dantzig heuristic packs at each time the most efficient item; in our case, the efficiency of server i is computed as:

$$\text{efficiency}_i = \frac{c_i + (1 - x_{i(t-1)})b_i}{\sum_{n\in N} h_{ni}w_n} \tag{5}$$

that is the average energy the server requires to provide one GPU unit. We allocate the servers based on ascending efficiency. This heuristic has $O(I \log I)$ time and also provides bounds on the optimal energy. The solution found is an upper bound; if each server contains only GPU of one type, the energy cost before the last server is assigned is the lower bound [19]. So, this heuristic overallocates at most one server compared to the optimal solution; in instances with thousands of servers (where the computational cost of the MILP is excessive), the difference is negligible.

We evaluate the allocation in terms of a trade-off between performance and energy cost. We use Success Rate (SR), i.e. the percentage of times the upcoming workload is successfully covered by the allocation [8], to measure performance. To measure energy cost, we use the energy required to allocate the real demand with the servers allocated according to the predicted one: when underpredicting demand, the unmet demand is simply dropped (the energy cost will be rewarded, and the SR will be penalised); when overpredicting, the SR will be rewarded and the energy cost penalised.

IV. Uncertainty-aware models

In this section, we describe the forecasting models we compare in the experiments. Prophet is the only model that is not derived from an LSTM network. All models, except for LSTM-Q, predict a probability as a Gaussian distribution with parameters μ and σ. Following the methodology outlined in [8], we can then compute the confidence interval (CI) corresponding to the desired SLT and take its upper bound as the final workload prediction. The LSTM-Q provides predictions naturally associated with a specific SLT, i.e. quantile-based [20], which we take as the final workload prediction.

- HBNN: proposed in [8], is an LSTM-based neural network optimised via variational inference where the last layer is Bayesian and models the epistemic uncertainty. The network's output, i.e. the distributional layer, captures the aleatoric uncertainty.
- HBNN++: derived from the HBNN, this architecture presents two Bayesian layers. The first one replaces the 1DCNN layer of the HBNN, and it captures the aleatoric uncertainty. The second one is positioned as the last layer of the architecture, and its output provides a non-deterministic pointwise prediction. We run the inference step 30 times and compute μ and σ of the Gaussian distribution. This is a further step towards a fully Bayesian neural network. However, it comes with the drawback of being slower and requiring more data for training.
- Monte Carlo Dropout LSTM (MCD-LSTM): a traditional LSTM network whose output is made non-deterministic by means of the dropout approach [21] (applied both during training and inference). We run the inference step 30 times to estimate μ and σ of the Gaussian distribution.
- LSTM Quantile (LSTM-Q): this LSTM's variation outputs quantiles, thus making the model distribution independent from the Gaussianity assumption. The benefit of quantile optimisation is to make a model more robust to outliers (common in workload forecasting) and to capture the aleatoric uncertainty [20]. We train different models for different SLTs (see Section V).
- C-LSTM: from [8] this traditional LSTM-based model outputs μ (deterministically), while σ is inferred from the training set as the standard deviation that a confidence interval should have to meet a specific SLT (and this value is constant for all the predictions).

- Prophet: [22] uses an additive model to best capture trends and seasonalities in the data. It has already been used for cloud workload prediction [14], but its performance in probabilistic predictions is still under-investigated in this domain.

V. Experiments

We implemented two scenarios from the evaluation framework designed in section III on real-world data. In this section, we give their details and present the results obtained.

TABLE I
GPUs specifications and server machines details composing the Alibaba cluster. Details in this table are manually crafted from [23]. The maximum computational power is expressed in GPU units per second (GPU units/s), and the number reported is the amount of workload processed in a 5 minutes time window. Maximum power consumption also refers to the energy consumed in a 5 minutes time window.

Name	# GPUs	# servers	# GPUs per server	Max. comp. power (units/s)	Max. power cons (kWh)
P100	1596	798	2	5821.61	0.0208
T4	994	497	2	4158.29	0.0058
V100	1912	239	8	8316.59	0.0233
MISC	2240	280	8	6513.58	0.0208
Total	6742	1814			

A. Setup

We preprocess Alibaba GPU cluster data [23] following the same procedure in [8]. The resulting dataset is a time series where each data point represents the aggregated average GPU workload demand appearing in the entire cluster in a 5 minutes period t (expressed in GPU units per second). The task of each model from the previous section is to predict that workload.

From [23], we also craft specifications of the GPUs in the cluster and their distribution across server machines (see Table I). From that, we calculate energy consumption e_n and the maximum GPU units provided for the 5 minutes period (w_n) for each GPU. For the server machine consumption, we rely on expert-provided estimates of a fixed 5-minute energy cost of $c_i = 0.0017$ KWh, $\forall i \in I$, and a booting cost $b_i = 0.2 * c_i, \forall i \in I$ that corresponds to the energy spent to wake up the server one minute before period t. As noted in Section III, we consider an average GPU utilization of $\alpha = 0.7$.

Inspired by the Persistent Mode[1] (PM) option of real world GPU servers, we implemented and ran two scenarios (with shared code[2]). They differ based on the default status of the server's GPUs when the server is running:

- PM_{off}. When the persistence mode is *off*, we consider GPUs to be in a sleep state by default while the server runs. GPUs are wakened up accordingly to meet workload (with some negligible delay of a few seconds); when no demand is available, there is no energy cost to pay. This scenario simulates a medium-responsive system where applications do not require real-time computations. In this case, the GPUs do not have to be idling even if the server that contains them is on.
- PM_{on}. When the persistence mode is *on*, we consider GPUs idling by default while the server runs. When demand is assigned to a GPU n, the GPU is immediately available to meet workload (with no delay); when no demand is assigned, there is an idle energy cost to pay e_n^{idle}, i.e. 20% of the maximum computational power e_n in our experiments. To model this, we add the idle costs of all the GPUs in a server to the server energy consumption:

$$c_i^* = c_i + \sum_{n \in N} h_{ni} e_n^{idle} \tag{6}$$

and we consider as GPU energy utilization cost the difference from being idle or in use $e_n^* = e_n - e_n^{idle}$. With this adaptation, we can use model III to compute the optimal approximated energy cost. This scenario simulates a high-responsive system where applications require real-time computations, e.g. gaming.

For each scenario, we evaluated all six predictors described in section IV for $|T| = 2582$ periods for the ten different SLT in $\{0.9, 0.91, .., 0.99\}$. We also implemented two baselines:

- *oracle*, predicting the true demand – the allocator provides maximum SR with minimal energy consumption;
- *naive*, which always predicts the maximum workload that the whole system can cover. The allocator thus keeps all servers running for all periods, resulting in the maximum SR with maximum energy consumption.

B. Results

We first report in Table II the evaluation of the predictors for traditional accuracy metrics, i.e. Mean Squared Error (MSE) and Mean Average Error (MAE), widely used metrics in time-series analysis [24]. We calculate the metrics based on the real demand and where each workload prediction is the μ estimated by each model (i.e. so that predictions are independent of any specific SLT). Results show MCD-LSTM and C-LSTM are the most accurate predictors (and the former is the best model) because they have been trained to minimise the MSE. On the contrary, more complex models, such as Prophet and HBNN++, seem to struggle. However, accuracy results give a myopic view of the real performance of predictors under real-world cloud scenarios, as can be seen in in Figs. 1 and 2 from our evaluation framework. For each predictor, these figures visualise the trade-off between SR and the energy cost of the allocation for each model for the ten different SLTs. The cost reported is expressed as energy savings proportional to the maximum energy cost (as incurred by the naive predictor).

Fig. 1 shows results for the PM_{off} scenario, where the oracle (i.e. the reference predictor) achieves a 100% SR and a relatively small energy savings of 5.5%. Prophet is good at meeting its SLTs but is the worst regarding energy savings. Interestingly, all the other models can save more energy but at the expense of some SR percentage points. MCD-LSTM (not

[1]Persistence Mode is the term for a user-settable driver property that keeps a target GPU initialized even when no clients are connected to it.

[2]https://github.com/andreareds/GreenUncertaintyForecaster

TABLE II
ACCURACY AND SERVICE LEVEL METRICS. FOR EACH PREDICTOR, MSE AND MAE REPORTED ARE CALCULATED ON THE ESTIMATED μ. SUCCESS RATE (SR), UNDERPREDICTION (UP) AND OVERPREDICTION (OP) REPORTED ARE CALCULATED ON PREDICTIONS CALCULATED FOR AN SLT OF 95%.

Model	MSE	MAE	SR (%)	UP (%)	OP (%)
HBNN	0.0059	0.0551	96.86	0.15	13.54
HBNN++	0.0094	0.0755	95.23	0.33	17.59
MCD-LSTM	**0.0021**	**0.0339**	61.07	1.53	2.58
LSTM-Q	0.0071	0.0753	**94.89**	0.18	8.95
C-LSTM	0.0033	0.0423	86.13	0.53	5.85
PROPHET	0.0335	0.1498	96.59	0.23	32.96

reported in Fig. 1 to improve the visualisation) is the best at saving energy (around +7.5%), but its performance for SR is 61%, way below satisfying any SLT (and C-LSTM is next best at energy saving, but is consistently below the SLTs). All the other models show more balanced and interesting trade-offs. While they are all relatively close in energy savings (i.e. within 5.5% and 6.2%) HBNN++ saves the most energy (e.g. furthest right on the red horizontal line showing a 0.95 SLT). These results can be further interpreted by the last three columns in table II, which report SR, Underprediction (UP, the proportion of unmet workload) and Overprediction (OP, the proportion of the extra workload predicted) corresponding to an SLT of 0.95, (see [8] for a formal definition of UP and OP). Overprediction has little impact on energy cost in this scenario, because overallocation does not waste much energy (only a small fixed cost for running a server). Only Prophet, with a large overprediction, shows poor energy savings. Underprediction impacts the trade-off, as under-allocating resources penalises SR but saves energy (e.g. MCD-LSTM and C-LSTM).

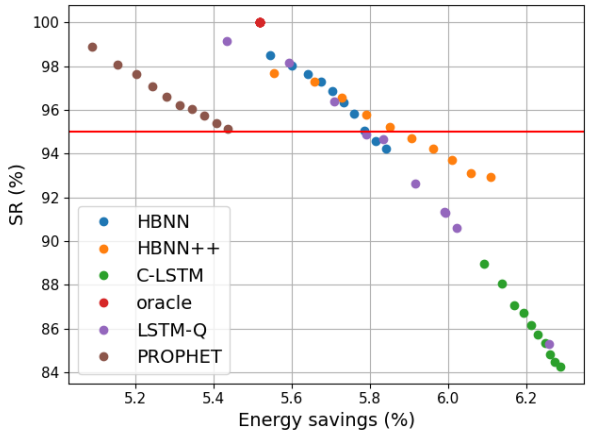

Fig. 1. Results for scenario PM_{off} (Persistent Mode off). We report the ten trade-off values corresponding to different SLT in $\{0.9, 0.91, .., 0.99\}$ (from left to right) for each predictor. The models with the best trade-off should be positioned on the top right hand of the plot. The red horizontal line helps visualising models that have the best trade-off at 0.95 SLT.

Fig. 2 shows results for the PM_{on} scenario (again, we do not plot MCD-LSTM to improve the visualisation), where the oracle achieves a 100% SR and energy savings of

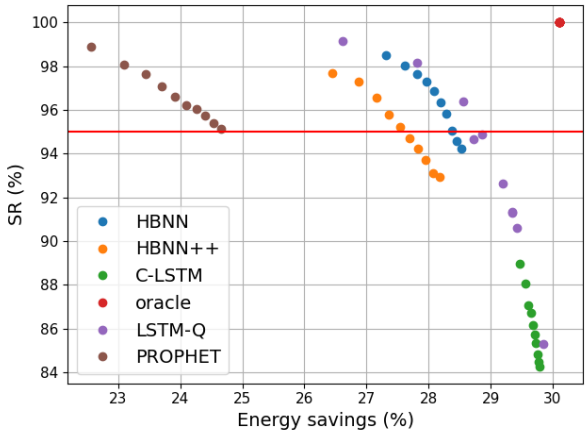

Fig. 2. Results for scenario PM_{on} (Persistent Mode off). We report the ten trade-off values corresponding to different SLT in $\{0.9, 0.91, .., 0.99\}$ (from left to right) for each predictor. The models with the best trade-off should be positioned on the top right hand of the plot. The red horizontal line helps visualising models that have the best trade-off at 0.95 SLT.

30.1% (and only MCD-LSTM achieved better energy savings, i.e. +31%). This is because overprediction (therefore, over-allocating servers) now has a higher energy impact due to idling GPUs. Prophet, MCD-LSTM and C-LSTM behave similarly to the previous scenario and show unbalanced trade-offs. Trade-offs for the other models are more subtle. LSTM-Q appears to provide greater energy savings, but it consistently undershoots its target SLTs. The original HBNN provides the best performance, as it consistently exceeds its SLTs while saving between 27% and 28% of the energy costs.

This can also be explained by Table II, where HBNN, HBNN++ and LSTM-Q have a better trade-off for UP and OP to Prophet, C-LSTM and MCD-LSTM. These findings illustrate the distinctions between the various components utilized in modelling uncertainty for time series forecasting. Despite high pointwise performance, MCD-LSTM is not good at uncertainty modelling, as indicated by its low standard deviation. Its overconfident predictions make it more suitable for larger networks such as the ones used in Computer Vision or Natural Language Processing. Prophet performs poorly when dealing with time series lacking seasonality. HBNN++ outperforms HBNN in terms of SR accuracy, likely due to its additional Bayesian convolutional layer; improving its performance would necessitate a significant increase in data and training time. HBNN, conversely, strikes a balance between accuracy and runtime performance. Finally, the LSTM-Q model's quantile regression approach offers a more straightforward yet effective solution, thanks to the lack of assumptions regarding Gaussian output. Moreover, all the models achieve runtime performance suitable for real-world applications.

VI. CONCLUSION

This paper proposes an evaluation framework assessing the impact of uncertainty-aware predictions for workload demand. The evaluation goes beyond the traditional accuracy and

service metrics proposed in the literature and tailors a simple yet effective resource allocation scheme that leverages the predicted demand to satisfy a desired service level while minimizing resource cost. We assess the effectiveness of the framework with offline experiments inspired by real-world cloud scenarios that compare a set of models on a GPU workload prediction task. We showed that the proposed model can be adapted to fit different data centre settings. We provided a heuristic that quickly computes a near-optimal policy, making it usable for comparing prediction models on large datasets.

Results show that our framework provides insights into the trade-off between service performance and energy cost of different predictors. Our analysis reveals that different predictions (with uncertainty) lead to different trade-offs under different scenarios (although they are quite consistent across different SLTs). While HBNN++ shows the best trade-off for PM_{off}, HBNN offers the best trade-off for PM_{on}.

In the future, we aim to extend the model to represent more aspects of real data centres, making the approximated energy cost closer to the real one. We will extend experiments to two different datasets, i.e. Microsoft Philly [25], and Helios [26] GPU traces, to confirm the usefulness of our framework. Moreover, several other directions can be explored from this work. For example, we will leverage the framework to evaluate uncertainty-aware multi-step-ahead forecasting models, using the full MIP model. Also, multi-step-ahead prediction could feed the allocator so that resource optimization would benefit from more sophisticated allocation, as uncertainty can be used to model inter-period workload trends. Finally, richer and different cost models (e.g. related to business metrics) could be integrated into our framework, tailored to the allocation task or the uncertainty of the predictions (or both).

ACKNOWLEDGMENT

This publication has emanated from research supported by Science Foundation Ireland under Grant Nos. 18/CRT/6223 and 12/RC/2289-P2 at Insight the SFI Research Centre for Data Analytics, which is co-funded under the European Regional Development Fund, and by the EU Horizon projects Glaciation (101070141) and TAILOR (952215). We thank Peter MacHale for helping to specify the GPU server performance models.

REFERENCES

[1] S. D. Müller, S. R. Holm, and J. Søndergaard, "Benefits of cloud computing: literature review in a maturity model perspective," *Communications of the Association for Information Systems*, vol. 37, no. 1, p. 42, 2015.

[2] M. M. Nair and A. K. Tyagi, "Ai, iot, blockchain, and cloud computing: The necessity of the future," in *Distributed Computing to Blockchain*. Elsevier, 2023, pp. 189–206.

[3] M. Koot and F. Wijnhoven, "Usage impact on data center electricity needs: A system dynamic forecasting model," *Applied Energy*, vol. 291, p. 116798, 2021.

[4] L. Belkhir and A. Elmeligi, "Assessing ict global emissions footprint: Trends to 2040 & recommendations," *Journal of cleaner production*, vol. 177, pp. 448–463, 2018.

[5] D. Patterson, J. Gonzalez, U. Hölzle, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. R. So, M. Texier, and J. Dean, "The carbon footprint of machine learning training will plateau, then shrink," *Computer*, vol. 55, no. 7, pp. 18–28, 2022.

[6] A. Souza, N. Bashir, J. Murillo, W. Hanafy, Q. Liang, D. Irwin, and P. Shenoy, "Ecovisor: A virtual energy system for carbon-efficient applications," in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ser. ASPLOS 2023. New York, NY, USA: Association for Computing Machinery, 2023, p. 252–265.

[7] M. Tirmazi, A. Barker, N. Deng, M. E. Haque, Z. G. Qin, S. Hand, M. Harchol-Balter, and J. Wilkes, "Borg: the next generation," in *Proceedings of the Fifteenth European Conference on Computer Systems*, 2020, pp. 1–14.

[8] A. Rossi, A. Visentin, S. Prestwich, and K. N. Brown, "Bayesian uncertainty modelling for cloud workload prediction," in *2022 IEEE 15th International Conference on Cloud Computing (CLOUD)*. IEEE, 2022, pp. 19–29.

[9] P. M. Mammen, N. Bashir, R. R. Kolluri, E. K. Lee, and P. Shenoy, "Cuff: A configurable uncertainty-driven forecasting framework for green ai clusters," in *Proceedings of the 14th ACM International Conference on Future Energy Systems*, 2023, pp. 266–270.

[10] P. A. Dinda and D. R. O'Hallaron, "Host load prediction using linear models," *Cluster Computing*, vol. 3, no. 4, pp. 265–280, 2000.

[11] S. Di, D. Kondo, and W. Cirne, "Host load prediction in a Google compute cloud with a Bayesian model," in *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE, 2012, pp. 1–11.

[12] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using ARIMA model and its impact on cloud applications' QoS," *IEEE transactions on cloud computing*, vol. 3, no. 4, pp. 449–458, 2014.

[13] B. Song, Y. Yu, Y. Zhou, Z. Wang, and S. Du, "Host load prediction with long short-term memory in cloud computing," *The Journal of Supercomputing*, vol. 74, no. 12, pp. 6554–6568, 2018.

[14] M. Daraghmeh, A. Agarwal, R. Manzano, and M. Zaman, "Time series forecasting using facebook prophet for cloud resource management," in *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2021, pp. 1–6.

[15] D. Minarolli, A. Mazrekaj, and B. Freisleben, "Tackling uncertainty in long-term predictions for host overload and underload detection in cloud computing," *Journal of Cloud Computing*, vol. 6, pp. 1–18, 2017.

[16] R. Mohammadi Bahram Abadi, A. M. Rahmani, and S. Hossein Alizadeh, "Self-adaptive architecture for virtual machines consolidation based on probabilistic model evaluation of data centers in cloud computing," *Cluster Computing*, vol. 21, pp. 1711–1733, 2018.

[17] E. Hüllermeier and W. Waegeman, "Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods," *Machine Learning*, vol. 110, pp. 457–506, 2021.

[18] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using arima model and its impact on cloud applications' qos," *IEEE Transactions on Cloud Computing*, vol. 3, no. 4, pp. 449–458, 2015.

[19] G. B. Dantzig, "Discrete-variable extremum problems," *Operations research*, vol. 5, no. 2, pp. 266–288, 1957.

[20] R. Wen, K. Torkkola, B. Narayanaswamy, and D. Madeka, "A multi-horizon quantile recurrent forecaster," *arXiv preprint arXiv:1711.11053*, 2017.

[21] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[22] S. J. Taylor and B. Letham, "Forecasting at scale," *The American Statistician*, vol. 72, no. 1, pp. 37–45, 2018.

[23] C. Jiang, Y. Qiu, W. Shi, Z. Ge, J. Wang, S. Chen, C. Cerin, Z. Ren, G. Xu, and J. Lin, "Characterizing co-located workloads in alibaba cloud datacenters," *IEEE Transactions on Cloud Computing*, 2020.

[24] F. X. Diebold and R. S. Mariano, "Comparing predictive accuracy," *Journal of Business & economic statistics*, vol. 20, no. 1, pp. 134–144, 2002.

[25] M. Jeon, S. Venkataraman, A. Phanishayee, J. Qian, W. Xiao, and F. Yang, "Analysis of {Large-Scale}{Multi-Tenant}{GPU} clusters for {DNN} training workloads," in *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, 2019, pp. 947–960.

[26] Q. Hu, P. Sun, S. Yan, Y. Wen, and T. Zhang, "Characterization and prediction of deep learning workloads in large-scale gpu datacenters," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021, pp. 1–15.