Using Quantile Regression for Reclaiming Unused Cloud Resources while achieving SLA

Jean-Emile Dartois, Anas Knefati, Jalil Boukhobza, Olivier Barais

Using Quantile Regression for Reclaiming Unused Cloud Resources while achieving SLA

Jean-Emile Dartois*†, Anas Knefati*, Jalil Boukhobza*‡ and Olivier Barais*†
*b<>com Institute of Research and Technology, †Univ Rennes, Inria, CNRS, IRISA, ‡Univ. Bretagne Occidentale
Email: jean-emile.dartois@b-com.com, anas.knefati@b-com.com, boukhobza@univ-brest.fr, barais@irisa.fr

***Abstract*—Although Cloud computing techniques have reduced the total cost of ownership thanks to virtualization, the average usage of resources (*e.g.,* CPU, RAM, Network, I/O) remains low. To address such issue, one may sell unused resources. Such a solution requires the Cloud provider to determine the resources available and estimate their future use to provide availability guarantees. This paper proposes a technique that uses machine learning algorithms (Random Forest, Gradient Boosting Decision Tree, and Long Short Term Memory) to forecast 24-hour of available resources at the host level. Our technique relies on the use of quantile regression to provide a flexible trade-off between the potential amount of resources to reclaim and the risk of SLA violations. In addition, several metrics (*e.g.,* CPU, RAM, disk, network) were predicted to provide exhaustive availability guarantees. Our methodology was evaluated by relying on four in production data center traces and our results show that quantile regression is relevant to reclaim unused resources. Our approach may increase the amount of savings up to 20% compared to traditional approaches.**

I. Introduction

One of the main objectives of Cloud providers (*CPs*) is to ensure a good quality of service (QoS) for customers while reducing the operating costs [1]. To achieve this goal, *CPs* have built large scale data centers and massively adopted virtualization technologies to share resources between customers.

The number of hosts within data centers is often increased to handle load peaks, hardware failures, customer QoS demands, and to give the illusion of infinite resources to customers. However, this illusion comes with a price. Although the use of virtualization improved the utilization of computing resources in data centers [28], several studies have demonstrated that the average usage of resources remains low, between about 20% to 50% in case of CPU [8].

One way to improve Cloud data center resource utilization and thus reduce the total cost of ownership (TCO) is to reclaim unused resources [26] and sell them. However, reclaiming resources needs to be done without impacting customers' requested QoS. QoS is usually defined in terms of Service Level Agreements (SLA). In case of violations of these agreements, penalties are applied. The goal of CPs is to maximize the amount of reclaimed resources while avoiding the risks of violations due to resources overcommitment.

In a Cloud infrastructure, unused resources for some metric (mtr, *e.g.,* CPU usage) at time t are defined as follows:

$$Unused_{(t,mtr)} = Cap_{(t,mtr)} - Used_{(t,mtr)} \quad (1)$$

where $Cap_{(t,mtr)}$is the maximum performance reachable by the system for mtr at time t; and $Used_{(t,mtr)}$ is the used capacity for mtr at time t.

Google and Amazon proposed to take advantage of unused resources by leasing them at a lower price compared to regular ones (*e.g.,* dedicated resources). In [1], the authors proposed a similar approach. However, reclaimed resources are coming with limited to no SLA guarantees, which reduces the number of applications that can be deployed [8].

In order to achieve SLA on top of unused reclaimed resources, several challenges have to be tackled:

1) Estimating $Cap_{(t,mtr)}$ to determine the real system capacity (*e.g.,* considering inter-workload interference)
2) Estimating $Used_{(t,mtr)}$ can be described with the following properties:
   - **Granularity**: The level at which the estimation is performed (*e.g.,* data center, cluster, or host level). The finer is the granularity, the more complex is the estimation, and the better the usability of the model.
   - **Flexibility**: The estimation needs to be flexible enough to give the CP the opportunity to find the best trade-off between the amount of resources to reclaim and the risk of SLA violations.
   - **Exhaustivity**: An estimation should be based on several resource metrics to achieve SLA requirements for reclaiming the maximum amount of unused resources. For example, if there are free CPU resources without available memory, this may lead to SLA violations.
   - **Robustness**: estimations should be robust to workload change as deployed workloads have vastly different runtime characteristics [29].
   - **Applicability**: estimation techniques should not have high overheads in terms of time and computing resource requirements as compared to the potential reclaimable resources.
3) Designing a scheduling strategy that takes into account the estimation of unused resources, the estimation of errors, QoS requirements, hardware/software constraints, affinity and anti-affinity [9].

In this paper, we focus on the second challenge by investigating how to provide an accurate estimation of the future used resources $Used_{(t,mtr)}$ (see eq.1). Our goal is to maximize the leasing of unused resources which, in turn, will maximize potential cost savings for the CP. Most state-of-the-art resource prediction studies [1] have focused on the estimation of the mean load. This makes those solutions poorly **flexible** as the mean load highlights only one aspect of the distribution of a variable (*e.g.,* CPU) without considering peak values (*i.e.,*

SLA violations). In addition, most studies [1] do not rely on an **exhaustive** set of metrics. Indeed, they are mostly based on two resource metrics: CPU and/or RAM consumption. Unfortunately, relying on one or two single metrics in a data center is not realistic since applications often require multiple computing resources. In effect, network and storage resources play a major role in SLA violation avoidance [1]. The authors of [8] provided an interesting investigation about forecasting the unused capacity in order to provide SLA over spare resources. Even though they provide a **robust** model, the **granularity** was too large. The cluster level (*i.e.,* aggregation of all hosts) was chosen. As it will be detailed further in the paper, the resource usage distribution among hosts in a given data center is not homogeneous, which makes it hard to design a scheduler strategy able to deploy applications among the hosts given the cluster level spare resource prediction.

In this paper, we investigate how to design the model for future unused resource while achieving SLA guarantees. We used three learning algorithms: Gradient boosting decision tree (*GBDT*), Random Forest (*RF*) and *LSTM* (Long short term Memory). We chose the **host level granularity** to make the model suitable and easily usable for deployment. One of the key contributions of our study is the use of quantile regression to make our model **flexible** for the CP, rather than using the simple mean regression of resource usage. This makes it possible for a CP to make relevant and accurate trade-off between the volume of resources that can be leased and the risk in SLA violations. We use six resource metrics (*i.e.,* CPU, RAM, disk read/write throughput, network receive/transmit bandwidth) for the forecast to be **exhaustive** enough and allow more accurate allocations.

For **robustness** concerns, we evaluated our approach using six months of traces about resource usage from four different data centers (*i.e.,* two private companies, one public administration and one university) composed of more than fifty hosts. We evaluated several metrics such as the prediction accuracy per host and for a collection of quantiles. In addition, for **applicability** issue, we measured the training and forecast time to determine the overheads.

We have also evaluated the economic impact of our contribution for comparison. Our results show that the use of quantile regression may provide an increase of the potential cost savings by up to 20% with LSTM and about 8% for *GBDT* and *RF* as compared to traditional approaches (see Section IV).

Our contributions can be summarized as follows:

- A technique that relies on machine learning and quantile regression that makes it possible to trade-off between the amount of reclaimable resources and SLA violations.
- A comparative study of three machine learning algorithms (*RF*, *GBDT* and *LSTM*) with six quantile levels.
- An evaluation on real traces of more than one hundred hosts from four data centers for a six-month time period.

Outline of paper. Motivations based on real production data hosts are given in Section II.Our methodology is described in Section III. Section IV details the experimental evaluation performed. Section V reviews related work. Finally, we conclude in Section VI.

II. Motivation: Datasets Analysis

This section motivates the work in this paper by providing some analysis about four in-production data center traces. These traces were collected between 2015 and 2017 from various types of organizations (*i.e.,* one University, one public administration and two private companies).

First, we focus on one data center at the host level, and then we give an overview of the resources for all data centers.

1) Private Company 1: Table I shows the hardware characteristics of the hosts for *private Company 1*. A first observation one may draw is that its hosts are heterogeneous (proportion between CPU and RAM). Thus, reclaiming resources on some hosts would be more effective than on others.

Table I
Hosts characteristics of Private Company 1

HostID	CPU Cores	RAM [GB]	CPU MODEL
12.0.0.1	20	300	Intel(R) Xeon(R) 2.20GHz
12.0.0.2	20	130	Intel(R) Xeon(R) 2.20GHz
12.0.0.3	12	130	Intel(R) Xeon(R) 2.30GHz
12.0.0.4	8	130	Intel(R) Xeon(R) 2.40GHz
12.0.0.5	12	130	Intel(R) Xeon(R) 2.30GHz
12.0.0.6	12	130	Intel(R) Xeon(R) 2.30GHz
12.0.0.7	12	130	Intel(R) Xeon(R) 2.30GHz
12.0.0.8	12	130	Intel(R) Xeon(R) 2.30GHz
12.0.0.9	12	130	Intel(R) Xeon(R) 2.30GHz

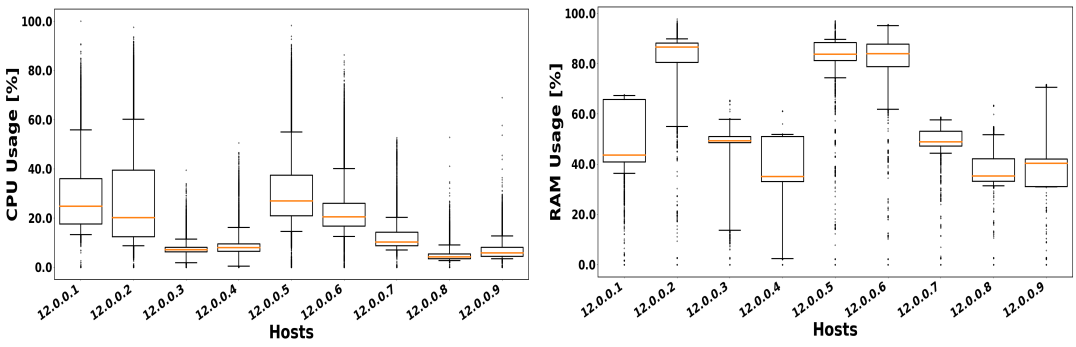
Let us focus on CPU and RAM in this section. Fig. 1a shows the box plots of CPU usage for the nine hosts. We observed that 75% of the time, the CPU median usage is under 40% for the hosts 12.0.0.1, 12.0.0.2 and 12.0.0.5. For the other hosts, the CPU median usage is even less than 20% during 75% of the time. We notice in the box-plots of Fig. 1b that the median usage of *RAM* is higher (about 50%) compared to *CPU*. This may be explained by the fact that in a virtualized environment the *RAM* is progressively allocated to the virtual machines but never released except when a memory management technique, such as memory ballooning, is enabled.

2) Potential Reclaimable Resources: Table II shows the overall capacity of the data centers used in this study. The *Private Company 2* data center is the largest one with 356 cores and 3.8 TB of memory provided by 27 hosts.

Table II
Available aggregated $Cap(t, mtr)$ of the data centers

Name	Number of Hosts	Duration [months]	CPU cores	RAM [TB]
University	10	22	116	1.5
Public Administration	7	35	240	2.5
Private Company 1	9	12	120	1.2
Private Company 2	27	17	356	3.8

Table III shows the average usage of the data centers for CPU, RAM, storage and network resources. One can notice that the four data centers have a maximum average CPU usage



(a) (b)

Figure 1. Box plots of (a) *CPU* and (b) *RAM* usage for each host with *Private Company 1*

of 17 % at the host level. This motivates our study as one can reclaim large amounts of resources to reduce the CP costs.

Table III
AVERAGE USAGE OF RESOURCES CALCULATED AT THE HOST LEVEL

Name	CPU Usage [%]	RAM Usage [%]	Disk R/W [Mb/s]	Network In/Out [Mb/s]
University	9.7	55.2	7.9/2.9	9.3/4.7
Public Administration	14.4	54.1	12/7.5	2/6.4
Private Company 1	17	57	10.6/3	7.9/2.1
Private Company 2	10.9	48	1/0.3	7.1/7.7

To conclude, from the four data centers investigated, all of them have a low resource usage. This encourages the use of reclaiming techniques. Secondly, in a given data center, configurations appear to be heterogeneous, and so resource usage is not balanced among hosts. This motivates the design of reclaiming technique at the host level granularity.

III. METHODOLOGY

Our goal is to provide a solution that maximizes the leasing of unused resources on a set of heterogeneous Cloud infrastructures (*e.g.,* VMware, OpenStack). Among the challenges discussed in the introduction, predicting the future use of hosts resources is an important issue. This forecasting has to be robust, fine-grained, flexible and exhaustive. In this section, we present our methodology that aims in building such a prediction model based on different learning machine algorithms. We introduce quantile regression as a technique to limit the risks of SLA violation at the cost of limiting the resources to be sold. We applied our methodology by replaying six months of four real data centers traces.

A. Background on quantiles

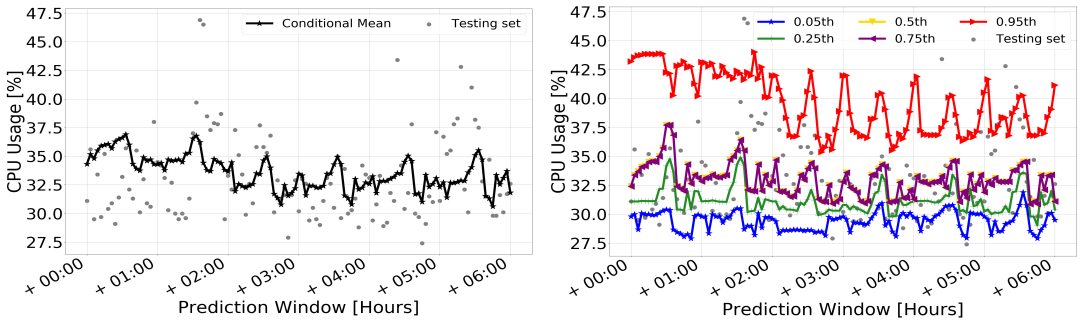
Quantiles are data values that divide a given dataset into adjacent intervals containing the same number of data samples [4]. They are useful to gain insight about the distribution of a random value (*e.g.,* CPU utilization) noted Y as compared to its mean value. **Conditional quantiles** investigate the behavior of Y by considering another vector of variables noted X that provides additional information. For example, the time (hour, minute) or historical values of CPU are variables that may be useful to describe CPU behavior (some VMs may be switched off during some period of time each day). The main advantage of conditional quantiles is to give a more comprehensive analysis of the relationship between X and Y at different points in the conditional distribution of Y given $X = x$. **Quantile regression** [24] seeks to estimate conditional quantiles. Rather than estimating the mean value of the CPU at a given time stamp, this regression method allows to estimate the τth quantile (*e.g.,* the $0.75th$ quantile or the CPU utilization value for which 75% of the values are lower).

In Fig. 2, we have estimated the conditional mean of the CPU usage (see Fig. 2a) and a collection of quantiles (*i.e.,* 0.05, 0.25, 0.5, 0.75 and 0.95) (see Fig. 2b) for a given time window. Quantile regression offers several levels of quantiles that gives us the opportunity to select the one that finds the best trade-off between SLA violations and available unused resources as compared to the conditional mean. As the quantile level increases, the amount of spare (reclaimed) resources decreases, the lower the risk of SLA violation. This is the main reason for the use of quantile regression for reclaiming unused resources in our study.

B. Approach overview

Our approach is composed of three steps as shown in Fig 3:

- A forecast strategy step: we chose to investigate three machine learning algorithms with their corresponding quantile approach. Our objective is to build a forecast model that infers future responses (*e.g.,* CPU, disk) from a set of past traces with different quantile levels. As a consequence, our problem fits in the supervised learning category. Since we want to forecast six metrics (*e.g.,* CPU, RAM), we used regression-based algorithms. We have evaluated RF, GBDT and LSTM (see Section III-C).
- A data pre-processing step: we prepared the extracted datasets from the data centers by applying the following operations: down-sampling, normalization, missing value handling, and features extraction (see Section III-D).
- An evaluation step: we replayed six months traces from four data centers by extracting all test windows of 24



(a) (b)

Figure 2. Forecasting of six hours of CPU with: (a) The conditional Mean curve in black, (b) Five different quantile regression curves.

hours and their associated training set per host. Then, we built prediction models with six quantile levels (*i.e.,* 0.5, 0.6, 0.7, 0.8, 0.9, 0.99*th*). We evaluated the accuracy, the training time and the potential economic savings induced by reclaiming resources (see Section IV).

C. Forecast Strategy step

In a Cloud infrastructure, forecasting future resources demands is a challenging task, especially for long periods of time [1]. Many variables may influence resources usage such as the deployed applications, the user behavior and the period of the day [29].

1) time series: Most CPs store their cluster resource usage indicators in time series. A time series is a sequence of N measurements $\{y_1, y_2, ..., y_N\}$ of an observable metric (*e.g.,* CPU, RAM), where each measurement is associated with a *time stamp*. As confirmed in [32] time series forecasting methods can reliably be used for cloud resource demand prediction. In this study, we use two strategies to forecast time series: (1) **a static strategy** that seeks to find a relationship between values of different time series. **a dynamic strategy called Multiple-input and Multiple-output (MIMO)** which can predict the whole sequence of values [5].

These strategies were used in the context of quantile regression. Indeed, quantile regression may provide the administrator with more flexibility to decide about the quantity of resources to rent as compared to the traditional conditional regressors.

2) Conditional quantile: There are two approaches to estimate the τth conditional quantile. To summarize these two approaches, we used the following notation:

- X a vector of p features (*e.g.,* working hours)
- Y an output variable (*e.g.,* CPU usage)
- $y_1, y_2, ..., y_n$ sampled values from Y
- x one observation of X
- $F(.|x)$ the conditional Cumulative Distribution Function (CDF) of Y given $X = x$
- $\mathbb{E}$ the mathematical expectation

The **direct** approach consists in minimizing a sum of asymmetrically weighted absolute residuals [17] based on:

$$q_\tau(x) = \arg\min_{\mu(x)} \mathbb{E}\left(\rho_\tau(Y - \mu(x))|X = x\right)$$

where ρ_τ is the following loss function introduced by Koenker and Basset [24] and τ is the quantile level:

$$\rho_\tau(u) = \begin{cases} \tau u & u \geq 0 \\ (\tau - 1)u & u < 0 \end{cases} \quad (2)$$

This loss function is asymmetric, except for $\tau = 0.5$ (median).

The **indirect** approach is performed in two steps, the first one estimates the conditional CDF. Then, the τth conditional quantile of Y given $X = x$ is obtained via inversion of the estimated conditional CDF [23] based on:

$$q_\tau(x) = F^{-1}(\tau|x)$$

3) Machine learning algorithms: We have investigated three algorithms:

- **RF** and **GBDT** for the static forecasting strategy, these algorithms was recognized to be the best potential choice according to [13], [1] and [7]
- **LSTM** algorithm for the *MIMO* forecasting strategy, where it proved its efficiency in the context of workload prediction as in [25], [30]

The interesting characteristics of these algorithms are summarized in Table IV. based on [13].

In what follows, we describe each of the three learning

Table IV
LEARNING ALGORITHMS PROPERTIES :✓= GOOD AND ▼=POOR.

Characteristics	GBDT	RF	LSTM
Robustness to outliers	✓	✓	▼
Natural handling of data of "mixed" type	✓	✓	▼
Handling of missing values	✓	✓	▼
Computational complexity	▼	▼	▼
Forecasting accuracy	✓	✓	✓
Approach	direct	indirect	direct

algorithms used that seek to estimate the τth conditional quantile of $Used_{(t,mtr)}$ given $X = x$. The accuracy of a model depends strongly on the dataset and the learning algorithm used. It also depends on the algorithm tuning parameters, called hyperparameters. These hyperparameters impact the complexity of the learning model, and they are estimated so

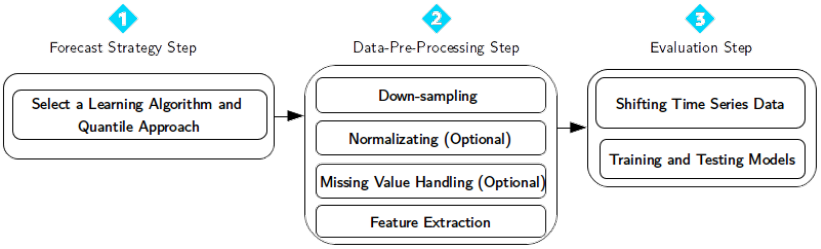


Figure 3. Overall Approach

as to minimize the error. This is why in this section we give some elements about their configuration.

Random Forests (RF), introduced in [6], they enhance decision trees by building a large collection of de-correlated trees, and then averaging them. RF are a combination of CART (Classification and Regression Trees) models, which are binary trees, such that each model depends on the values of a random vector sampled from training data independently with the same distribution for all trees in the forest. In *CART*, the split aims to maximize the accuracy score by splitting the training data with the best feature on each node of the trees. RF are very accurate and their hyperparameters are simple to tune [19]. In [27] authors have proposed a variant of RF that proved to be able to work with the quantile indirect approach.
RF have three hyper-parameters: the number of trees T and the following hyper-parameters for each tree:

- m : the number of features to consider when looking for the best split
- n_{min} : the minimum number of samples required to split an internal node

In our study, the hyperparameters were set to $T = 300$, $m = \lfloor \frac{\text{number of features}}{3} \rfloor$ and $n_{\text{min}} = 5$, as recommended by [6].

Gradient Boosting Decision Trees (GBDT) The main idea of GBDT is to train iteratively a decision tree such that the ensemble of these decision trees may be more accurate than any decision tree. In our study, we used *GBDT* proposed by Friedman [14]. GBDT has three main hyperparameters:

- M : the number of regression tree models
- ℓ : the size of trees
- ν : the learning rate

We set the hyperparameters of GBDT to $\ell = 6$, $\nu = 0.09$ and $M = 300$ as recommended by [13]. In addition, the loss function used, ρ_τ see eq. 2, was configured to estimate the quantile regression (*i.e.,* direct approach).

Long Short Term Memory (LSTM) Recurrent Neural networks (RNN) [2] were designed to capture dependencies within an input sequence and not only a single feature vector compared to RF and GBDT. To achieve that, RNN use hidden states that act as internal memory to keep information about previous inputs. In this way, RNN are useful for capturing temporal dependencies by tracing previous information.
Traditional RNN suffer from vanishing or exploding problem during the back-propagation of the gradient weights on long sequences [2]. In [20], the authors have proposed LSTM to address this issue.
The direct quantile approach was used with LSTM with the loss function ρ_τ. The architecture used is composed of one input layer, one LSTM hidden layer, and one output layer fully connected. LSTM parameters are:

- *Input layer* : the number of time steps the model will be looking at.
- *Hidden layer* : the number of neurons per step
- *Output layer* : the number of steps to forecast
- *Batch size* : the number of samples that are going to be propagated through the network
- *Epochs* : the number iteration over the entire dataset

As discussed in Section III-D, we used a sampling rate of 3 minutes, this means that one LSTM step is equivalent to 3 minutes. So, in order to forecast 24 hours we set the output layer to 480. The following hyperparameters are respectively set to, *input layer size*=20, *hidden layer size*=20, *learning rate*=0.09 , *batch size*=23, and *epochs*=90n see [30]

D. Data-pre-processing step

The goal of the pre-processing step is to create the matrix for input vector features, noted $\mathbf{X}$, and the vector of the observed responses, noted $\mathbf{Y}$ from past traces. To achieve that, three operations have to be done: standardization/normalization, handling of missing values, and preparing the data for the learning.

The first step is standardization/normalization of the datasets. It turns out that depending on the dataset and thus the company, the sampling rate of the metric collection was not the same. The sampling rate has an impact on the accuracy and the processing time. A too low frequency would provoke the loss of system dynamism and thus may lead to SLA violations, but a too high frequency would cause an increase of the processing time. We down-sampled the measurements in order to aggregate a time range into a single value at an aligned *timestamp*. We chose a data sampling rate of 3 minutes as a good trade-off.

In addition, as recommended for LSTM we scaled the input features between zero and one.

The second step handles the missing values that are common in real deployments, the data can be corrupted or unavailable. To achieve that, we filled the missing values by propagating the last valid measurement.

The third step consists in preparing the data by extracting the features $\mathbf{X}$ and the output response $\mathbf{Y}$ from the datasets. This extraction has to be done according to the characteristics of the learning algorithms.

Concerning RF and GBDT, for each y_i (*i.e,* $Used_{(t,mtr)}$), we extracted the row of x_i as follows:

- We extracted the day, hours and minutes features to investigate the *timestamp* information. We selected these features to allow learning algorithms to find the relationship between these features and resource usage. The feature month and year were not used since we trained our models using a one-month data.
- We extracted, from the datasets, the holidays and working hours features (*i.e.,* the feature is set to 1 for working hours, and 0 for hours of week-ends or holidays).

For LSTM, the training data required to use a sliding window in order to transform the time series (*i.e.,* $Used_{(t,mtr)}$) into a supervised learning problem.

E. Evaluation step

We evaluated our approach by replaying the six months traces from four data centers. One requirement to consider in order to evaluate a time series forecast compared to traditional supervised learning is to split the training and testing set sequentially in order to maintain the temporal dimension.

To achieve that, the six months of data were shifted into multiple sequential 24 hours windows. Each window is composed of a training and a testing part (*i.e.,* forecast window). The test window is starting after the end of a training window. As we fixed the forecast window to 24 hours on six months, this gives 183 windows per host. Then, each window was evaluated with *Normalized Mean Quantile Errors* (NMQE).

IV. Evaluation

This section describes the results of our experiments, by which we try to answer four research questions (RQ):

- **RQ1** (**Flexibility**): What are the potential cost savings for CP with regard to different quantile levels ?
- **RQ2** (**Exhaustivity**): What differences can we observe in SLA violations when considering several resource metrics as compared to only CPU as in state-of-the-art work.
- **RQ3** (**Robustness**): What is the accuracy of the tested algorithms and how does the accuracy change according to the evaluated workloads ?
- **RQ4** (**Applicability**): What is the training overhead of the learning algorithms and its impact on the reclaimed resources ?

A. Experimental setup

To answer these four research questions, we led four experiments, each using production traces from four data centers. In this section, we introduce the elements used in common within these experiments:

- the experimental scenario used to calculate the potential cost savings for CPs in particular the leasing model, the pricing model and the penalty model.
- the metrics used to evaluate the learning phase,
- the experimental environment.

1) Experimental Scenario:

a) Potential cost Savings: To calculate the potential cost savings for CPs, we defined three models. First, a leasing model to determine the period during which the customer rents the unused resources and their amount (resource granularity). Second, a pricing model to determine the fee that the CPs wold receive from the customer for the provided service. Finally, a penalty model that fixes the amount of discount on the customer bill in case of SLA violation. We assume that all reclaimed resources are leased. The cost savings estimations do not take into account the cost generated by the leasing such as the wear out of the hardware and energy consumption.

b) Leasing Model: For simplicity, we used a unique model based on the declared capacity of the hosts in the datasets. The leasing granularity is a container runtime provisioned for a period of 24 hours with 2 virtual CPU cores,8 GB Memory, and 100 Mbp/s of network bandwidth and the same for storage throughput.

c) Pricing Model: We used a fixed price based on a pay-as-you-go model since it is the dominant schema according to [29]. The price was fixed to 0.0317$ per hour for one leasing model as used by Google Preemptible VMs [18].

d) Penalty Model: There are three types of penalties [15]: (1) a *fixed penalty* where each time the SLA is violated a discount is applied, (2) a *delay-dependent penalty* for which the discount is relative to the CP response delay, and (3) a *proportional penalty* where the discount is proportional to the difference between the agreed upon and the measured capacity.

Public Cloud such as OVH, Amazon and Google use a hybrid approach (*Fixed penalty* and *Delay-dependent penalty*). Table V shows the discount applied when SLA are not met.

Table V
Discount applies in case violations for a 24-hour window

Violation Duration [Minutes]	Discount
$>$ 15 to $\leq$ 120	10%
$>$ 120 to $\leq$ 720	15%
$>$720	30%

2) Evaluation Metric: To evaluate the robustness of the learning algorithms and the potential cost savings, we used the NMQE and *Interquartile Range* IQR metrics.

a) NMQE: A common metric to evaluate the quality of quantile regression is the Mean Quantile Error (MQE):

$$\text{MQE} = \frac{1}{n}\sum_{i=1}^{n} \rho_\tau(y_{i,mtr} - \hat{q}_{mtr}(\tau, x_i))$$

In order to be able to compare accuracy with different metrics, we used a Normalized MQE (NMQE), given by:

$$\text{NMQE} = \frac{\text{MQE}}{y_{\max} - y_{\min}}$$

b) IQR: The interquartile range (IQR) is a measure of statistical dispersion (*i.e.,* NMQE's) given by:

$$IQR = Q_3 - Q_1$$

where Q_3 and Q_1 are respectively the 0.75 and 0.25 quantiles.

3) Experimental environment: We made use of Python with the following packages: *scikit-garden* [1] in case of *RF* with version 0.1 and *scikit-learn* [2] for GBDT version 0.18 and *keras* [3] for *LSTM* libraries which provide state-of-the-art machine learning.We also used Apache Spark [31] version 2.0.2. Beside, all training and forecasts were performed on servers with an Intel(R) Xeon(R) E5-2630 v2 CPU clocked at 2.60GHz with 130GB of RAM.

B. Flexibility: potential cost savings (RQ1)

To evaluate the benefits of quantile regression for reclaiming unused Cloud resources while achieving the SLA, we compared the potential cost savings for CP with regard to different quantile levels. We conduct three experiments:

- **Exp1**. Using the *Private Company 1* dataset, we compute the reclaimable resources amount using the three different learning algorithms. Based on these three models, *i.e.,* the leasing model, the pricing model and the penalty model introduced in the previous subsection, we then calculate the potential cost saving in dollars according to the quantile level.
- **Exp2**. We compared the behavior of the quantile regression depending on the host resources usage profile to investigate if the optimal quantile level changes according to the host resource usage.
- **Exp3**. We generalize to the other data centers.

1) Exp1: Fig 4a shows the potential cost savings in dollars according to the quantile levels for *Private Company 1* and for the three machine learning algorithms.

A first observation one may draw is that the potential cost savings increase with the increase of the quantile level up to τ=0.99 for both *GBDT* and *RF* and up to τ=0.9 for *LSTM*.

A second observation is that for each learning algorithm, there is an optimal τ level, which corresponds to the trade-off between SLA violations and the amount of reclaimable resources (*i.e.,* τ=0.99 for *GBDT* and *RF* and τ=0.9 for *LSTM*). This shows that with *GBDT* and *RF*, the decrease of reclaimable resources (increase of τ) is compensated by the reduction of SLA violations. However, in case of *LSTM* when τ>0.9, this is not the case anymore: the reduction of unused resources is higher than the decrease of SLA violations.

A third observation is that the best amount of potential cost savings is obtained with *LSTM* with 3166$.

[1]https://github.com/scikit-garden/scikit-garden

[2]https://github.com/scikit-learn/scikit-learn

[3]https://keras.io

We conclude that for all learning algorithms studied, quantile regression brings a clear added value: (1) improvement in cost savings as compared to a median-estimation based approach (τ=0.5), and (2) a flexibility to adapt to the optimal level of τ according to the selected algorithm. This result can be generalized to all tested data centers as discussed farther.

2) Exp2: Fig. 4b shows the potential costs savings at a host level using LSTM for *Private company 1*. We notice two behaviors of the cost saving according to the quantile level: (1) the hosts where cost savings increase up to τ=0.99; and (2) those where savings decrease starting from τ=0.9. We notice that all the the hosts obeying the first behavior are those with a low usage, such as 12.0.0.3, 12.0.0.4, 12.0.0.8. This can be explained by the fact that an increase in the quantile level does not imply a strong decrease of reclaimable resources, as the peak utilization of resources reaches a maximum of 40% for the CPU and 45% for the RAM. Even when τ increases, the loss of cost savings is less significant as compared to hosts with a high utilization (peaks that reach 100%) and with larger resource usage dispersions.

As expected, when comparing the cost savings with the measured resource usage (see Fig. 1a and 1b) we notice that hosts with a high usage, such as 12.0.0.2, 12.0.0.5 and 12.0.0.6 generate less savings, except for the host 12.0.0.1 which has extra memory of 170GB compared to the others.

We conclude that: (1) quantile regression makes it possible to adapt to resource usage heterogeneity in data centers and (2) the host **granularity** is relevant for reclaiming resources in a data center.

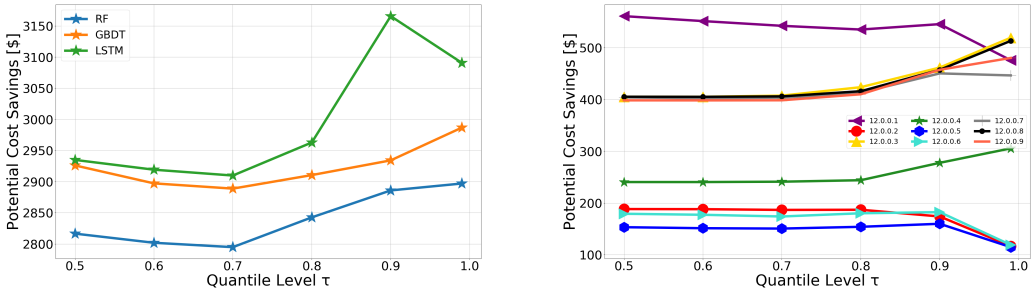
Table VI
POTENTIAL COST SAVINGS WITH REGARDS TO τ FOR ALL DATASETS

Dataset	τ	0.5	0.6	0.7	0.8	0.9	0.99
University	RF	2122	2124	2122	2155	2185	**2198**
	GBDT	2672	2651	2652	2568	2727	**2786**
	LSTM	2163	2134	2122	2155	**2259**	2236
Public Administration	RF	4628	4616	4635	4786	5024	**5034**
	GBDT	4715	4676	4670	4691	4794	**4926**
	LSTM	4708	4687	4698	4789	**5142**	4838
Private Company 1	RF	2816	2801	2795	2842	2885	**2897**
	GBDT	2926	2897	2889	2910	2934	**2987**
	LSTM	2935	2919	2910	2963	**3166**	3090
Private Company 2	RF	6659	6650	6655	6887	**7414**	6803
	GBDT	6670	6728	6763	6995	**7210**	7153
	LSTM	6428	6441	6528	6736	**7722**	6857

3) Exp3. Generalization on 4 data centers: Table VI shows the aggregated potential cost savings on the four data centers with regards to quantile levels and the three learning algorithms. We observe that for all datasets *LSTM* is the best choice except for the *University* where *GBDT* gives better cost savings.

Compared to the traditional approaches that use conditional mean (*i.e.,* equivalent to τ=0.5), our approach based on the use of quantile regression performs better with an increased amount of savings of 8% for *private company 1*, 20% for *private company 2*, 9% for *public administration* and 4% for the *university*.

Overall one can observe that the use of quantile regression is useful for the three algorithms and four datasets and provide the required **flexibility** to reduce SLA violations.



(a) Aggregated potential cost savings for *Private Company 1* with **exhaustive** SLA metrics awareness

(b) Potential saving with regard to the quantile level with LSTM and the nine hosts of *Private Company 1*

Figure 4. Aggregated Potential Cost Savings

C. Exhaustivity: impact of relying on a single resource (RQ2)

To illustrate the need to apply metrics-exhaustive models, we calculated the cost savings by taking into account only the CPU. We then subtracted the calculated savings to the results previously calculated by our six metrics model.

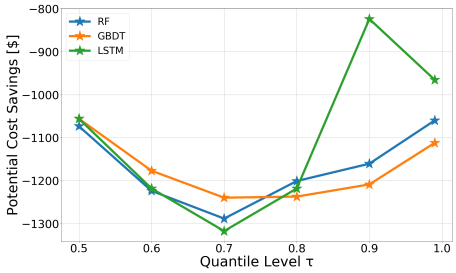


Figure 5. Aggregated cost violations for *Private Company 1* when there is no exhaustive SLA metrics awareness (*i.e.,* only CPU)

Fig 5 shows the cost of SLA violation when taking into account only *CPU*. With τ=0.5, one can observe that a non-exhaustive choice of metrics leads to a violation of about -1050$. Taking into account only CPU leads to an increase in SLA violations. Indeed, these violations get higher, up to -1317$ with τ=0.7 and then decrease down to τ=0.99 due to the reduction of reclaimable resources.

To conclude, we observe that a non-exhaustive choice of metrics leads to no savings as the penalties are higher compared to benefits. In addition, the use of quantile regression in a non-exhaustive way has increased the amount of violations.

D. Robustness: resilience to workload change (RQ3)

To evaluate the accuracy of the tested algorithms and observe its evolution along the various deployed workloads, we use $NMQE$ and IQR indicators. These are used on all forecasting models and all hosts of *private company 1* with a quantile equal to 0.9.

Table VII shows the resilience of the learning algorithms when facing various workloads for six months evaluated using $NMQE$ and IQR indicators.

Table VII
MEDIAN (M) AND INTERQUARTILE RANGE (IQR) OF $NMQE$ FOR ALL FORECAST MODELS AND ALL HOSTS WITH 0.9 QUANTILE LEVEL WITH *Private Company 1* DATASET.

Metric	Indicator	RF	GBDT	LSTM
CPU	M	**0.37**	0.48	0.57
	IQR	**0.71**	0.91	0.97
RAM	M	**0.00002**	0.14	0.15
	IQR	**0.09**	0.38	0.18
Disk Read	M	**0.13**	0.21	0.27
	IQR	**0.62**	0.68	0.91
Disk Write	M	**0.05**	0.1	0.12
	IQR	**0.11**	0.2	0.14
Netreceived	M	**0.01**	0.025	0.018
	IQR	**0.09**	0.138	0.136
Nettransmitted	M	**0.009**	0.014	0.011
	IQR	**0.04**	0.08	0.077

We observe that all the forecast models have a quite good accuracy. We observe that *RF* has the best accuracy regarding the median of $NMQE$. It also provides the smaller dispersion given by IQR compared to the other algorithms. This means that *RF* is more inclined to be resilient to workload pattern change, which is a very interesting property. In addition, we observed that *CPU* and *Disk Read* were the metrics with the highest dispersion with a IQR of 0.71 and 0.62.

When comparing with the potential cost savings, we would expect *RF* to give the best results. However, as shown in Table VI LSTM did. This can be explained by the fact that the calculation of the potential savings only penalizes the negative errors (*i.e.,* when the available unused resources are overestimated). Underestimation is not penalized directly compared to the indicator NMQE, which penalizes both positive errors and negative errors.

E. Applicability: training overhead (RQ4)

Training time is important as it is directly related to the amount of reclaimable resources. Indeed, the resources used for training and forecast would not be available for leasing. In this experiment, we evaluate the overhead (median computation time) to train and forecast each of the three learning algorithms for six metrics and a forecast horizon of 24 hours.

Table VIII shows the raw results. It turns out that *LSTM* was the slowest with a training/forecast time of about 500 seconds, as it has a high number of parameters to optimize. Then, with a duration of 130 seconds *RF* is slower than *GBDT*. This could be due to the fact that *RF* is estimating the quantile with an indirect approach that requires two steps.

Table VIII
MEDIAN COMPUTATION TIME USED FOR THE TRAINING AND FORECAST 24 HOURS FOR ONE HOST.

Algorithm	Median Processing Time [Seconds]
GBDT	2
RF	157.11
LSTM	424.20

This means that for a data center composed of 100 hosts the learning phase would take about 12 hours each 24 hours with *LSTM* if we used a similar equipment to the one we experimented. In comparison, *RF* would take about 4 hours and *GBDT* 3 minutes. Note that the duration is highly related to the implementation of the learning algorithms and the choice of hyperparameters. When looking from the point of view of training/forecast computation time *GBDT* seems to be a good choice and LSTM the worst.

F. Threats to validity

Our experiments show the benefits of using quantile regression for reclaiming unused Cloud resources while achieving SLA. However, as in every experimental protocol, our evaluation has some bias which we have tried to mitigate. All our experiments were based on the same case study regarding the leasing model, the pricing model and the penalty model. We have tried to mitigate this issue by using models close to those of real Cloud providers.

One external threat to validity is our choice of data centers raw data. Further work is needed to reproduce our case study on other datasets, and we cannot guarantee that our results will apply to all data centers. We have tried to mitigate this issue by using datasets from different real CPs and different business cases.

Finally, there is a threat that the choice of hyperparameters are incorrectly set for the learning phases even if we relied on strong state-of-the-art work. If this happens to be the case, then all experiments introduce a similar level of imprecision, and a relative comparison of these may still be valid.

V. RELATED WORK

A. Reclaiming unused resources

Maximizing the utilization of Cloud computing resources can be achieved using various strategies. Some state-of-the-art studies [26] took advantage of unused resources by leasing them with limited SLA. Others proposed to use predictive models in order to achieve SLA [8] by forecasting mainly the CPU. In addition, in [1], the authors noticed that the literature has focused on two metrics (*i.e.,* CPU and RAM) while effective forecasting model should investigate all resources not only CPU and RAM in order to provide SLA. In [26] authors make available any underutilized resources in an opportunistic way to improve resource utilization. In the same way, iExec [21], Amazon Spot instances and Preemptible Virtual Machines propose similar services. In [8] authors have proposed to claim unused Cloud capacities to offer a cheaper class (*i.e.,* limited SLA) with long-term availability by forecasting available resource for the next 6 months. Compared to our work, the authors have focused on the forecast of aggregated CPU consumption (*i.e,* all hosts in the cluster). This does not provide the required level of granularity.

B. Time Series Forecast Algorithms

Many state-of-the-art studies such as [1] have discussed how to select the appropriate learning algorithm(s) to forecast time series with learning algorithms such as Autoregressive (AR), Integrated Moving average (ARIMA) and other more complex algorithms such as RNN, SVM, LSTM, and RF.

In [11] authors used *AR, MA, ARMA* and ARIMA models or other variants to forecast the load average from 1 to 30 seconds in the future. The main drawbacks are the linear character of the ARIMA models, the lack of explicit seasonal indices. It remains hard to interpret coefficients or explain "*how the model works*". Yang *et el.* proposed several homeostatic and tendency-based one-step-ahead forecasting methods. The tendency-based method forecasts the future CPU value under the assumption that the pattern is stable which is not our case. Beghdad *et al.* [3] proposed to use neuro-fuzzy and Bayesian inferences for the the problem of CPU load forecasting. Gmach *et al.* [16] studied the workload analysis for enterprise data center application. In this case, the workload analysis demonstrates the burstiness and repetitive nature of enterprise workloads. Song *et el.* [30] applied LSTM to forecast the mean host load in data centers of Google and other traditional distributed system. They compared LSTM method with the following methods: Autoregressive (AR) model [33], artificial neural networks(ANN) [12], Bayesian model [10], the PSR+EA-GMDH method [34] and the echo state networks(ESN) [35]. They have shown that their method achieves state-of-the-art performance with higher accuracy in both data centers. Kumar *et al.* [25] used LSTM networks to build a workload forecasting model. They showed that the accuracy of their forecasting model has reduced the mean square error up to 3.17×10^{-3}. Islam *et al.* [22] treated the case of resource provisioning in the Cloud. They used Neural networks, linear regression algorithms and a sliding window technique. Their approach supposes a linear fashion of the workload pattern which is not our case.

Our approach improves state-of-the-art solutions by applying quantile regression on the same machine learning algorithms but with a higher number of metrics. This is done to find the best trade-off between SLA violation and leased resources.

VI. CONCLUSION AND PERSPECTIVES

The use of quantile regression is a relevant approach to reclaim unused resources with SLA requirements. We described

our technique that makes it possible to select the quantile level that gives the best trade-off between the amount of reclaimable resources and the risk of SLA violations. We evaluated three machine learning algorithms with regards to five properties **granularity**, **flexibility**, **exhaustivity**, **robustness** and **applicability** by replaying six months of four data centers traces (*i.e.,* one *public administration*, two *private companies*, one *university*).

We drew four main conclusions. **First**, our results show that quantile regression provides the required **flexibility** that makes it possible to find the optimal quantile level that maximizes cost savings.

Second, the most **robust** learning algorithm was given by *RF* with a median NMQE of 0.37 for *Private Company 1* hosts. However, traditional accuracy metrics used in machine learning fail to determine the best algorithm that maximizes the potential cost savings while limiting SLA violations. Using our approach, it turned out that *LSTM* performs better on robustness for three data centers, with potential cost savings increasing up to 20 %.

Third, as expected we need to be as **exhaustive** as possible to avoid SLA violations by taking into account a higher number of metrics. We measured that considering only CPU and omitting disk read/write, network reception and RAM leads to no savings, as the violation amount reaches -1317$ in the worst case.

Fourth, for **applicability** concerns, GBDT was the one with the smaller computational overhead while LSTM had the highest overheads.

As for future work, we will use these results to design a scheduling strategy to deploy applications among the predicted unused resources. Finally, we plan to integrate more metrics in our model such as GPU usage or network latency.

ACKNOWLEDGMENT

This work was supported by the Institute of Research and Technology b-com, dedicated to digital technologies, funded by the French government through the ANR Investment referenced ANR-A0-AIRT-07.

REFERENCES

[1] M. Amiri and L. Mohammad-Khanli. Survey on prediction models of applications for resources provisioning in cloud. *Journal of Network and Computer Applications*, 82:93–113, 2017.

[2] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[3] F. Benhammadi, Z. Gessoum, A. Mokhtari, et al. Cpu load prediction using neuro-fuzzy and bayesian inferences. *Neurocomputing*, 74(10):1606–1616, 2011.

[4] F. Benson. A note on the estimation of mean and standard deviation from quantiles. *Journal of the Royal Statistical Society. Series B (Methodological)*, 11(1):91–100, 1949.

[5] G. Bontempi. Long term time series prediction with multi-input multi-output local learning. *Proc. 2nd ESTSP*, pages 145–154, 2008.

[6] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[7] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM, 2006.

[8] M. Carvalho, W. Cirne, F. Brasileiro, and J. Wilkes. Long-term slos for reclaimed cloud computing resources. In *Proceedings of the ACM Symposium on Cloud Computing*, pages 1–13. ACM, 2014.

[9] C. Delimitrou and C. Kozyrakis. Hcloud: Resource-efficient provisioning in shared cloud systems. *ACM SIGOPS Operating Systems Review*, 50(2):473–488, 2016.

[10] S. Di, D. Kondo, and W. Cirne. Host load prediction in a google compute cloud with a bayesian model. In *High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for*, pages 1–11. IEEE, 2012.

[11] P. A. Dinda and D. R. O'Hallaron. An evaluation of linear models for host load prediction. In *High Performance Distributed Computing, 1999.*, pages 87–96. IEEE, 1999.

[12] T. V. T. Duy, Y. Sato, and Y. Inoguchi. Improving accuracy of host load predictions on computational grids by artificial neural networks. *International Journal of Parallel, Emergent and Distributed Systems*, 26(4):275–290, 2011.

[13] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[14] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[15] R. Garg, H. Saran, R. S. Randhawa, and M. Singh. A sla framework for qos provisioning and dynamic capacity allocation. In *Quality of Service, 2002. Tenth IEEE International Workshop on*, pages 129–137. IEEE, 2002.

[16] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper. Workload analysis and demand prediction of enterprise data center applications. In *Workload Characterization, 2007. IISWC 2007. IEEE 10th International Symposium on*, pages 171–180. IEEE, 2007.

[17] S. Goh. Design-adaptive nonparametric estimation of conditional quantile derivatives. *Journal of Nonparametric Statistics*, 2012.

[18] Preemptible virtual machines. https://goo.gl/zoqP1x, 2018.

[19] P. Han, X. Zhang, R. S. Norton, and Z.-P. Feng. Large-scale prediction of long disordered regions in proteins using random forests. *BMC bioinformatics*, 10(1):1, 2009.

[20] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[21] iexec white paper. https://iex.ec/whitepaper/iExec-WPv3.0-English.pdf, 2018.

[22] S. Islam, J. Keung, K. Lee, and A. Liu. Empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Computer Systems*, 28(1):155–162, 2012.

[23] M. A. Knefati, A. Oulidi, and B. Abdous. Local linear double and asymmetric kernel estimation of conditional quantiles. *Communications in Statistics-Theory and Methods*, 45(12):3473–3488, 2016.

[24] R. Koenker and G. Bassett Jr. Regression quantiles. *Econometrica: journal of the Econometric Society*, pages 33–50, 1978.

[25] J. Kumar, R. Goomer, and A. K. Singh. Long short term memory recurrent neural network (lstm-rnn) based workload forecasting model for cloud datacenters. *Procedia Computer Science*, 125:676–682, 2018.

[26] P. Marshall, K. Keahey, and T. Freeman. Improving utilization of infrastructure clouds. In *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, pages 205–214, 2011.

[27] N. Meinshausen. Quantile regression forests. *Journal of Machine Learning Research*, 7(Jun):983–999, 2006.

[28] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis. Efficient resource provisioning in compute clouds via vm multiplexing. In *Proceedings of the 7th international conference on Autonomic computing*, pages 11–20. ACM, 2010.

[29] M. Shaari, T. F. Ang, L. Y. Por, C. S. Liew, et al. Dynamic pricing scheme for resource allocation in multi-cloud environment. *Malaysian Journal of Computer Science*, 30(1), 2017.

[30] 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*, pages 1–15, 2017.

[31] A. Spark. Apache spark™-lightning-fast cluster computing, 2014.

[32] S. B. Taieb and A. F. Atiya. A bias and variance analysis for multistep-ahead time series forecasting. *IEEE transactions on neural networks and learning systems*, 27(1):62–76, 2016.

[33] Y. Wu, Y. Yuan, G. Yang, and W. Zheng. Load prediction using hybrid model for computational grid. In *Grid Computing, 2007 8th IEEE/ACM International Conference on*, pages 235–242. IEEE, 2007.

[34] Q. Yang, C. Peng, H. Zhao, Y. Yu, Y. Zhou, Z. Wang, and S. Du. A new method based on psr and ea-gmdh for host load prediction in cloud computing system. *The Journal of Supercomputing*, 68(3):1402–1417, 2014.

[35] Q. Yang, Y. Zhou, Y. Yu, J. Yuan, X. Xing, and S. Du. Multi-step-ahead host load prediction using autoencoder and echo state networks in cloud computing. *The Journal of Supercomputing*, 71(8):3037–3053, 2015.