Journal of King Saud University – Computer and Information Sciences 34 (2022) 3191–3203

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

Machine learning regression to boost scheduling performance in hyper-scale cloud-computing data centres

Damián Fernández-Cerero [a,*], José A. Troyano [a], Agnieszka Jakóbik [b], Alejandro Fernández-Montes [a]

[a] Department of Computer Languages and Systems, University of Seville, Avda. Reina Mercedes s/n., 41012 Seville, Spain
[b] Department of Computer Science, Cracow University of Technology, Cracow, Poland

ARTICLE INFO

Article history:
Received 2 November 2021
Revised 13 April 2022
Accepted 14 April 2022
Available online 22 April 2022

Keywords:
Data centre
Cloud computing
Scheduling optimisation
Machine learning
Gradient boosting

ABSTRACT

Data centres increase their size and complexity due to the increasing amount of heterogeneous workloads and patterns to be served. Such a mix of various purpose workloads makes the optimisation of resource management systems according to temporal or application-level patterns difficult. Data-centre operators have developed multiple resource-management models to improve scheduling performance in controlled scenarios. However, the constant evolution of the workloads makes the utilisation of only one resource-management model sub-optimal in some scenarios.

In this work, we propose: (a) a machine learning regression model based on gradient boosting to predict the time a resource manager needs to schedule incoming jobs for a given period; and (b) a resource management model, Boost, that takes advantage of this regression model to predict the scheduling time of a catalogue of resource managers so that the most performant can be used for a time span.

The benefits of the proposed resource-management model are analysed by comparing its scheduling performance KPIs to those provided by the two most popular resource-management models: two-level, used by Apache Mesos, and shared-state, employed by Google Borg. Such gains are empirically evaluated by simulating a hyper-scale data centre that executes a realistic synthetically generated workload that follows real-world trace patterns.

© 2022 The Authors. Published by Elsevier B.V. on behalf of King Saud University. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

1. Introduction

Data centres constitute the core infrastructure for current Internet services, from world-wide web and mobile applications to Big-data and High-Performance-Computing workloads. Therefore, the performance of such infrastructures is crucial to enable services that require computational and storage resources to be available, scalable, and ready to serve the incoming workloads, which reduces operational costs and offers the most competitive services to the final users.

In the past, it was more common for medium and large companies to manage their own clusters, which served a fully controlled workload. In such scenarios, the computing and storage resources can be fine-tuned by the operation teams. Currently, hyper-scale data centres serve a wide range of users and workload requirements. The mix of continental or even world-wide workloads causes difficulties in optimising the computational resources according to temporal or usage patterns, since the job-arrival patterns fade away, and unknown events may cause unexpected demand peaks that may degrade the performance of the data centre notably. In summary, the workloads related to Cloud Computing have mutated from controlled workloads to workloads that present arrival and resource-demand patterns that evolve rapidly and really hard to predict.

Data-centre resource managers are considered as the data-centre operating systems, as they are responsible for the management and monitoring of the scheduling process and the monitoring of the computational resources and workload at the highest level. Resource managers usually coordinate one or several scheduling agents, which, in turn, are in charge of the application of scheduling policies that determine the placement of each particular task on the data-centre resources. Fine-tuned monolithic resource managers were typically used when companies used to operate their own small-sized data centres. However, such monolithic resource managers, which usually employ scheduling algorithms designed for the specific workload under consideration, cannot keep up to the massive-scale workloads related to hyper-scale data centres. Even though data-centre operators used to leave some data-centre resource capacity idle to absorb moderate demand peaks without seriously degrading the data-centre performance, the

* Corresponding author.
E-mail addresses: damiancerero@us.es (D. Fernández-Cerero), troyano@us.es (J.A. Troyano), ajakobik@pk.edu.pl (A. Jakóbik), afdez@us.es (A. Fernández-Montes).

https://doi.org/10.1016/j.jksuci.2022.04.008
1319-1578/© 2022 The Authors. Published by Elsevier B.V. on behalf of King Saud University.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

resource managers could not keep up with the high number of scheduling operations present in new Cloud-Computing scenarios (Gog et al., 2016).

Various resource managers were developed to overcome the aforementioned limitations: centralised frameworks such as Omega (Schwarzkopf et al., 2013) and Mesos (Hindman et al., 2011), distributed solutions such as Pigeon (Wang et al., 2019), and even fine-tuned hybrid approaches, such as Mercury (Karanasos et al., 2015). The two-level (Mesos) and shared-state (Omega) centralised strategies are the most used currently. However, such resource managers focus on serving a common workload pattern present in hyper-scale data centres (Schwarzkopf et al., 2013), where approximately 90% of batch jobs consume few resources for a short time, whilst the 10% of jobs consume a large amount of resources for a longer time. However, such workload patterns change over time, making those models suboptimal in comparison to other resource-managing models for some use cases, or even only for a period of the same use case (D. Fernández-Cerero et al., 2018).

This work focuses on improving the scheduling performance of the data centre by proposing a novel resource manager selector that determines the most performant resource management model for each period. This proposal leverages the industry data-centre resource management architecture, which equips only a single resource management model, by enabling the dynamic selection of a resource management model from a catalogue of existing resource managers.

A gradient boosting regression model is responsible for the estimation of the scheduling performance of each resource management model in the catalogue for a given data-centre operating situation and period. According to the estimation of scheduling performance, the most beneficial resource management model is applied during that particular period.

The benefit of the proposed resource management model is then illustrated by comparing three alternatives: (a) the utilisation of the two-level resource management model used by Mesos or YARN; (b) the utilisation of the shared-state resource management model used by Google Borg and Google Omega; and (c) the utilisation of the proposed resource management model, Boost, which estimates the scheduling performance of the two resource management models above to apply them dynamically. The centralised two-level and shared-state resource management models have been selected as they constitute the main resource managers used in industry. The performance and energy efficiency results of the three alternatives above were evaluated through an extensive simulation employing a trustworthy simulation tool as well as a realistic hyper-scale data-centre composed of 1,000 machines and synthetic traces that follow the patterns present in Google and Alibaba data-centre traces.

In this work, we do not improve data-centre performance by proposing any scheduling algorithm, but a new data-centre resource management model. Any scheduling algorithm can be applied by scheduling agents coordinated by resource managers. The scheduling algorithm employed in the experimental analysis tries to maximise resource utilisation while preventing resource contention by following Google's latest improvements in this area (Lo et al., 2016).

The following contributions are presented in this work:

- Gradient boosting regression estimator for the prediction of the scheduling performance of the most popular industry-level resource managers.
- Transformation rules that enable the successful estimation of the scheduling performance of resource managers.
- Resource management model that leverages the estimator developed to select the most performant resource management model for a period and predicted incoming workload.

The benefits of the aforementioned contributions are backed by:

- Analysis of the impact of each feature of the regression models on the quality of the predictions for the catalogue of resource management models employed to train the regression models used in this work: two level (Mesos) and shared state (Omega).
- Analysis of the behaviour of our proposed resource management model, Boost, vs. the most utilised resource managers in extreme situations related to hyper-scale data centres, where the workload arrival pattern follows an extreme-value distribution. All the experimentation is based on realistic workload traces, which enables Boost to be applied in production environments.
- Analysis of the full scheduling time Key Performance Indicator (KPI), in addition to the traditional scheduling queue times evaluated in the literature (Schwarzkopf et al., 2013; Tirmazi et al., 2020).
- Evaluation of the performance impact of estimating and selecting the resource manager synchronously, that is, every time a job is submitted, vs. carrying it out asynchronously in the background every given period. Three time periods are analysed for the background process.
- Evaluation of the energy consumption of the proposed resource management model and the comparison with current resource managers.

This paper is organised as follows: Section 2 examines the related work, while Section 3 presents the theoretical basis that supports this study. The machine learning regression model is illustrated in Section 4. The results of the empirical analysis performed are evaluated in Section 5. Finally, Section 6 presents the conclusions of this work and future work.

2. Related work

Most research on data centres has focused on the increase of performance, reduction of costs, and improvement of computational and storage services. Over the last ten years, the focus has shifted to the application of various Artificial Intelligence techniques to various goals.

Workload classification and workload prediction have attracted considerable attention from the research community, as they are essential for the improvement of data-centre performance, the reduction of energy consumption, compliance with the required quality of service (QoS) levels, and the improvement of the scalability of cloud service providers.

Concerning workload classification, the authors of Dewangan et al. (YYYY) proposed a modified KNN algorithm for the classification of jobs into four categories, so the scheduling process takes into account both the user priority and the operating costs. The aim of Wu et al. (2018) is to classify jobs according to the required resources, such as CPU, memory, network, and storage, to develop a scheduling algorithm that includes this classification among the heuristic rules used to determine the optimal resources. The authors of Elrotub and Gherbi (2018) propose a deterministic method for the classification of tasks according to their size. Three groups are considered in the proposed scheduling algorithm to select optimal resources for each task: heavy, medium, and light tasks. Iqbal et al. (2018) shows an analysis of a web application workload based on URI requests to auto-scale the needed resources. The authors employ artificial intelligence to generate the so-called probabilistic workload pattern for the prediction of the incoming workload. Such prediction is then used to provision the estimated resource demand. Other works focus only on the analysis of various workloads to identify, classify, and cluster the jobs in differentiated groups. In these works, the authors do not directly employ such classification and clustering models to improve the data-centre KPIs. Genkin et al. (2019) focuses on the classification of Spark and Hadoop workloads by taking into account the container performance. A recent analysis of the Google traces presented in Patel and Kushwaha (2020) reveals that the Gaussian Mixture Model results in a better workload clustering in comparison with other methods.

Regarding to workload prediction, the majority of proposals are based on machine learning and artificial intelligence techniques, from Neural Networks to regression-based models. A sample of some representative proposals is presented below.

The authors of Singh et al. (2019) propose a Support Vector Machine model for the estimation of the characteristics of the incoming jobs in a given time window so that the resource provisioning process can be optimised. Li et al. (2021) proposes a hybrid model based on Autoregressive Integrated Moving Average model (ARIMA) and Back Propagation (BP) neural networks to predict SLA violations and VM migration times. The authors of Zhang et al. (2018) propose a new Deep Learning model based on canonical polyadic decomposition to efficiently predict the incoming workload. A trace included in the CloudSim simulator is used to validate the efficiency and accuracy of the proposed model. Wamba et al. (2017) compares constraint programming and neural network models to predict and generate traces, and reached the conclusion that neural network models offer better prediction results while constraint programming is more suitable for trace generation.

The authors of Kumar et al. (2018) present an interesting combination of three-layered neural networks and a self-adaptive differential evolution algorithm, and claim very good results in terms of workload prediction accuracy to improve resource provisioning. The training data set is based on the requests done to a very specific service (a Canadian university web server and a NASA Kennedy Space Center web server as well). The results reported by Gao et al. (2020) suggest that it is important to perform the prediction some time before it is needed, so that the scheduling algorithm has enough time to adapt. These authors compare various techniques and conclude that clustering methods for prediction can achieve up to 90% of prediction accuracy for CPU and memory. The authors of Tang et al. (2018) performed an analysis to determine suitable techniques for workload prediction to develop energy-aware scheduling algorithms. They combine linear regression and wavelet neural networks to this end and conclude that these techniques can be useful for low-utilisation cloud data centres. In Amiri et al. (2018), the authors propose a prediction model that outperforms the performance of current predictors. Their model is based on episode mining and is able to adapt to the workload changes rapidly, which is then used to improve the resource provisioning process.

In summary, as shown in Table 1, various research studies employ artificial intelligence models to analyse, classify, and even apply this knowledge to improve existing scheduling algorithms. However, to the best of our knowledge, no progress has been made at the resource-management level. This work presents a prediction model based on gradient-boosting regression that predicts the scheduling performance of a catalogue of resource management models for a given data-centre operating environment and period, which enables the selection of the resource management strategy that minimises the scheduling time for that period and incoming workload from such catalogue. The proposed resource manage-

Table 1
Classification of the related work considered. Boost works at the data-centre resource-managing level.

	Dewangan et al. (YYYY)	Wu et al. (2018)	Elrotub and Gherbi (2018)	Iqbal et al. (2018)	Genkin et al. (2019)	Patel and Kushwaha (2020)	Singh et al. (2019)	Li et al. (2021)	Zhang et al. (2018)	Wamba et al. (2017)	Kumar et al. (2018)	Gao et al. (2020)	Tang et al. (2018)	Amiri et al. (2018)	**Boost**
Target layer															
Scheduling algorithms	X		X									X	X		
Resource provisioning		X		X			X	X		X	X			X	
Workload analysis					X	X			X						
Resource manager															**X**
Environment															
Simulation	X		X						X		X		X		X
Benchmark		X		X	X			X							
Real world						X	X	X		X	X	X		X	
Supervised learning category															
Classification	X	X	X	X	X	X	X			X		X		X	
Regression							X	X	X	X	X	X	X		X

ment model does not conflict with the state of the art, since resource managers could coordinate the scheduling agents that equip the scheduling algorithms considered, as well as the provisioning and migration techniques.

3. Theoretical framework

Let us denote a single workload as $\mathscr{W}$, whilst $\mathscr{W}$ represents all workloads to be served. In this work, the values of the following parameters characterise each workload:

- **Inter-arrival time** $\Delta t_{j,sub} \sim Wei(0.5, \lambda_W)$ represents the time between two consecutive job submissions ($t_{j,sub}$ and $t_{j-1,sub}$ for the job J_j) of a particular workload for $j = 1, 2, \ldots$. We employ an extreme-value Weibull distribution with a shape value of α = 0.5 to generate the inter-arrival times with a scale value of $1/\lambda_{\mathscr{W}}$ to represent the hardly predictable workload inter-arrival patterns present in hyper-scale data-centres.
- **Number of tasks** $n_j \sim Exp(\lambda_{j,n})$ represents the number of tasks that a job is composed of, following an Exponential distribution with a given mean value of $1/\lambda_{j,n}$.
- **Job duration** $l_j \sim Exp(\lambda_{j,l})$ represents the period of time a given job J_j consumes resources in the data centre, generated by means of the Exponential distribution with the given expected value $1/\lambda_{j,l}$.
- **Resource usage** u_j is the amount of CPU (denoted by K_{CPU}) and RAM (denoted by K_{RAM}) that all the tasks of each particular job J_j in a workload consumes.

Each workload is composed of a set of jobs $\mathscr{W} = \{J_j\}_{j=1}^{n}$ for $n \in \mathbb{N}$, and each jth job is composed of tasks $\mathscr{T}_j = \{t_{jn}\}_{n=1}^{n_j}$ for $n_j \in \mathbb{N}$. Every time a job J_j is submitted to the data centre at a particular operation time, considered as the job submission time $t_{j,sub}$ of the job J_j, the scheduling process graphically represented in Fig. 1 starts.

Every time a job arrives, one scheduling agent performs a job scheduling action if it is available or puts the job in a queue until it becomes available. A scheduling action can be defined as the process that employs a scheduling algorithm to determine the set of resources where the tasks of a job are to be executed. When a scheduling action is unable to schedule all the tasks in a job, new scheduling actions will be performed until all the tasks of the job are assigned to computing resources or a time-out is reached. Once this process is over, the job is considered fully scheduled at $t_{j,sched}$. The time required by the scheduling agents to fully schedule the job J_j is denoted as $\mathscr{S}_j$, which represents the sum of times $\mathscr{S}_{ji}$ needed for each scheduling action performed on the job J_j, as follows: $\mathscr{S}_j = \sum_{i=1}^{n_{j\mathscr{A}}} \mathscr{S}_{ji}$, where $n_{j\mathscr{A}}$ denotes the number of scheduling actions needed to fully schedule the job J_j. Hence, $t_{j,sched} = t_{j,sub} + \mathscr{S}_j$. The completion time $t_{j,com}$ of job J_j may be denoted, then, as follows: $t_{j,com} = t_{j,sched} + l_j$, i.e: $t_{j,com} = t_{j,sub} + \mathscr{S}_j + l_j$.

Let us denote by $\mathscr{A}$ the set of all $\mathscr{A}_W$ actions, that is:

$$\mathscr{A} = \{\mathscr{A}_W, W \in \mathscr{W}\} = \{\mathscr{A}_{j,i}, j = 1, 2, .., n, i = 1, 2, .., n_{j\mathscr{A}}\} \tag{1}$$

Each scheduling action $\mathscr{A}_{ji}$ takes some time to compute the algorithms needed to make the deployment decisions, as follows:

$$\mathscr{S}_{\mathscr{A}}(i,j) = \mathscr{K}_J(j) + \sum_{n=1}^{n_{j,uns}} \mathscr{K}_T(n,j) \tag{2}$$

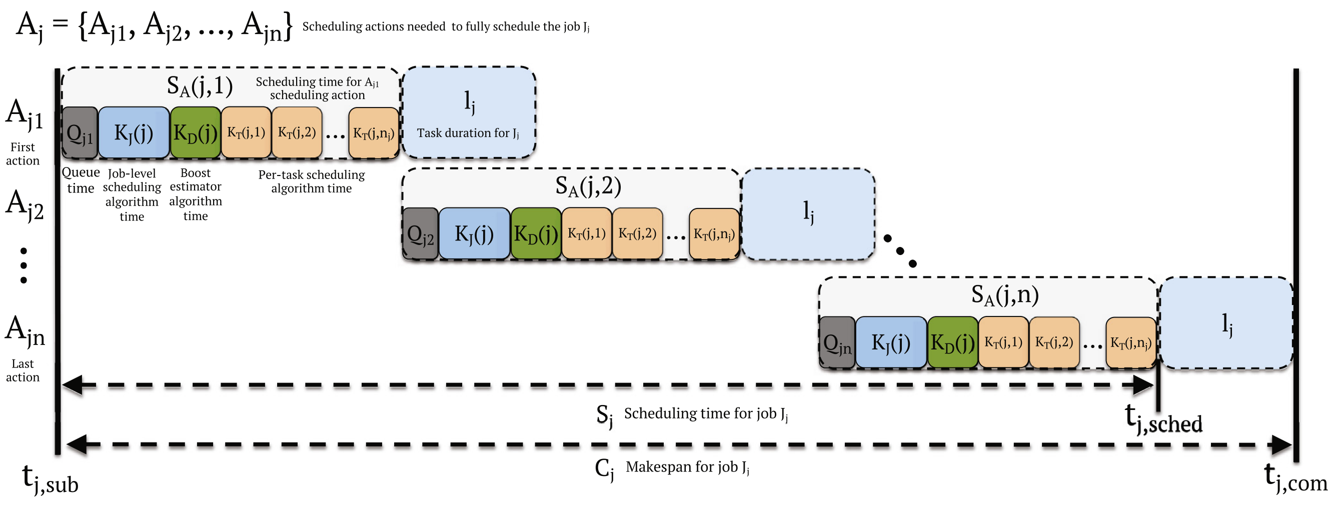
$\mathscr{S}_{\mathscr{A}}(i,j)$ denotes the time required to perform the ith scheduling action for the jth job, $\mathscr{K}_J(j)$ represents the time the scheduling algorithm executed at the job level, $\mathscr{K}_T(n,j)$ designates the time of the scheduling algorithm applied to nth task in the jth job, while $n_{j,uns}$ represents the remaining number of tasks to be scheduled. The equation above assumes the scheduling agent is free when the job arrives. However, this may not happen when previous jobs in the queue are not fully scheduled. Then, the time that a job waits in the queue must be added to the equation as follows:

$$\mathscr{S}_{\mathscr{A}}(i,j) = \mathscr{Q}_{ji} + \mathscr{K}_J(j) + \sum_{n=1}^{n_{j,uns}} \mathscr{K}_T(n,j) \tag{3}$$

$\mathscr{Q}_{ji}$ represents the time the jth needed to wait in queue when the ith scheduling action started.

Each scheduling action $\mathscr{A}_{ji}$ may provide two outcomes:

(a) The scheduling action $\mathscr{A}_{ji}$ succeeds in the deployment of all tasks $\mathscr{T}_j$ of the job J_j. In such scenario, $t_{j,sched} = t_{j,sub} + \mathscr{S}_j$, where $\mathscr{S}_j = \mathscr{S}_{\mathscr{A}}(0,j)$. In this equation, $t_{j,sched}$ denotes the operation time where the job is fully scheduled and $\mathscr{S}_{\mathscr{A}}(0,j)$ the time needed to perform the first scheduling action.



A_j = {A_j1, A_j2, ..., A_jn} Scheduling actions needed to fully schedule the job J_j
A_j1
First action
S_A(j,1)
Scheduling time for A_j1 scheduling action
Q_j1
K_J(j)
K_D(j)
K_T(j,1)
K_T(j,2)
K_T(j,n_j)
l_j
Task duration for J_j
Queue time
Job-level scheduling algorithm time
Boost estimator algorithm time
Per-task scheduling algorithm time
A_j2
S_A(j,2)
Q_j2
A_jn
Last action
S_A(j,n)
Q_jn
S_j Scheduling time for job J_j
t_j,sched
C_j Makespan for job J_j
t_j,sub
t_j,com

Fig. 1. General incremental job scheduling and execution process.

(b) The scheduling action $\mathscr{A}_{ji}$ can't fully deploy all $\mathscr{T}_j$ tasks of the job J_j, due to two main reasons: the data centre has not enough free resources to execute them, or the transaction that commits the scheduling action raises a conflict in optimistic-locking strategies. In such scenarios, the scheduling action $\mathscr{A}_{ji}$ needs to be retried until the number of scheduling actions $n_{j\mathscr{A}}$ reaches the limit of scheduling actions $n^{max}_{j\mathscr{A}}$, increasing the number of timed-out jobs J_{to}; or until the job J_j is fully deployed in the successive scheduling actions, when the number of required scheduling actions $n_{j\mathscr{A}} < n^{max}_{j\mathscr{A}}$. In such scenarios, $\mathscr{S}_j = \sum_{i=1}^{n_{j\mathscr{A}}} \mathscr{S}_{ji}$.

Boost employs a Machine Learning algorithm to estimate the performance of the resource manager for a certain data-centre operation period $\Delta_{\mathscr{D}}$ to apply the optimum resource-management model for that particular period. In this work, we consider two versions of the dynamic resource-management agent:

1. Background agent which performs the resource-management-model selection decision $\mathscr{D}$ every fixed time period $\Delta_{\mathscr{D}}$, independently from the scheduling actions $\mathscr{A}$ to avoid scheduling overhead in each job scheduling action $\mathscr{A}_{ji}$ and the related scalability issues.
2. Online agent which performs the resource-management model selection decision $\mathscr{D}$ for every job scheduling action $\mathscr{A}_{ji}$, thus, $\Delta_{\mathscr{D}} = f(\mathscr{A}_{ji}), \mathscr{A}_{ji} \in \mathscr{A}$ therefore increases the time needed for each job scheduling action. In this online approach, the estimation algorithm time $K_{\mathscr{D}}$ must be taken into account to compute each scheduling action, which may cause scalabilty issues in the scheduling process:

$$\mathscr{S}_{\mathscr{A}}(i,j) = \mathscr{Q}_{ji} + \mathscr{K}_J(j) + \mathscr{K}_{\mathscr{D}}(j) + \sum_{n=1}^{n_{j,uns}} \mathscr{K}_T(n,j) \quad (4)$$

The completion time (makespan) $\mathscr{C}_j$ is the final performance indicator that represents the end of the execution process for the job J_j, and may be denoted as follows: $\mathscr{C}_j = \mathscr{S}_j + l_j$.

The following energy measurement model is employed in this work:

$$\mathscr{E}(T) = \sum_{t\in\Delta}\sum_{r\in\mathscr{R}} \mathscr{P}(r,t)\,\delta \quad (5)$$

Let $t \in \Delta = \{\delta, 2\delta, \ldots, T\}$, where T denotes the total operation time. The power state for each computing resource in every machine $\mathscr{P}(r,t)$ is measured every δ time period. Two different power levels are considered in this work: $\mathscr{P} \in \{\text{Idle}, \text{Executing}\}$. The Idle power level denotes the power consumption of the computing resource when no tasks are being executed, while the Executing power level may be defined as follows: $\mathscr{P}_{Executing} = \mathscr{P}_{Max} - \mathscr{P}_{Idle}$, where $\mathscr{P}_{Max}$ represents the maximum power consumption when the computing resource is used at a maximum level. Hence, the power consumption $\mathscr{P}(r) = \mathscr{P}_{Idle} + \frac{r_{exec}}{r_{idle}} \cdot \mathscr{P}_{Executing}$, where $\frac{r_{exec}}{r_{idle}}$ represents the resource utilisation rate.

The proposed resource management model cannot impact negatively in terms of data-centre availability and robustness, since its main task, the prediction and dynamic application of the optimal resource-managing model for a given period, would not interfere in the workload execution. Even in a crash of the service executing the proposed resource management model, the resource manager that was running would keep serving the incoming workload as usual.

4. Gradient boosting regression model

4.1. Generation of the training data set

In this work, we will employ a regression model, which will be responsible for the determination of a numeric output from an input vector, to estimate the scheduling performance of two resource-managing strategies every certain period. A set of instances, each one composed of a pair of one input vector and one numeric output, shapes the data set our regression model is trained with. A 15-day data-centre execution trace, provided by the simulation tool SCORE Fernández-Cerero et al. (2018) is taken as the raw training data.

Synthetic data-centre execution traces for each resource manager considered are employed to train the estimation model. Such traces follow the workload patterns present in Google cluster 2011 and 2018 real data-centre traces (C. Reiss et al., 2012; Tirmazi et al., 2020). We employed realistic synthetic workload traces for several reasons:

- Normalisation and obfuscation of the workload. The aforementioned data-centre workload traces from Google 2011 and 2018 hide the real low-level characteristics of both the computational resources, such as the number of CPUs and GBs of RAM of each machine, as well as the actual workload resource consumption. Only a normalised value of such resource consumption can be obtained, whereas other attributes are directly obscured. Due to this, the actual reproduction of the physical data center with the real workload is unattainable - mainly due to the need of big industry players, such as Google and Microsoft, to keep the shape of their infrastructure confidential as a part of the industrial secrets-. Thus, only similar synthetic workloads can be generated, which can reproduce the behaviour of real data centres to some extent.
- Rapid evolution of cloud computing workload. The study of the traces above (C. Reiss et al., 2012; Tirmazi et al., 2020; Lu et al., 2017) shows a clear pattern where the workload has changed its shape over time: from the inter-arrival time to resource consumption, as well as the heterogeneity and extreme events of the workload. This evolution can make any resource management model developed only for a particular scenario useful only for a short time.
- Broader applicability of the proposal. Given the evolution of the Cloud-computing workloads, as well as the heterogeneity of the big Internet Data Centers present in the industry, we consider that testing the validity of a resource management model only for one short period of a particular environment could fall short to show its behaviour. Thus, the generation of realistic workload traces is relevant since it enables the evaluation of broader scenarios where industry players could check the benefits and drawbacks of the proposed resource management model and how it fits to their current situation.
- Validity of the synthetic workload. The fidelity of the synthetic workload to reality was first validated by the Google research team that developed both Google Omega and the first version of the employed simulator Schwarzkopf et al. (2013), and then such workloads have been shown to be highly useful in many previous articles published in prestigious journals, such as (Fernández-Cerero et al., 2018; Fernández-Cerero et al., 2018; Fernández-Cerero et al., 2018; Fernández-Cerero et al., 2021).

Some of the features in the raw data-centre execution trace may require some transformations to improve the usefulness of the raw data. The raw trace and the transformed training data set can be

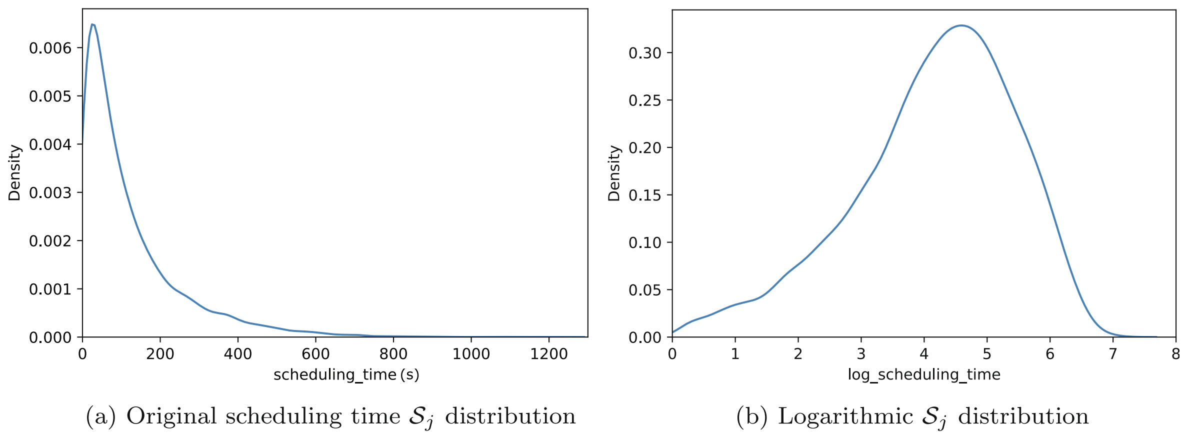


Fig. 2. Original vs. logarithmic Kernel density estimation (KDE) distribution for the class feature scheduling time $\mathscr{S}_j$.

found as supplementary material. In this section, the following main data transformations applied to build our training data set are explained: 1. Logarithmic transformation of the output feature. 2. Creation of a delta submission time feature $\Delta t_{j,sub}$. 3. Framing by time windows. 4. Uncertainty management for unfinished jobs.

The performance of the regression model is strongly related to the wideness of the range of values of the output feature. Fig. 2 shows the range of values for the output feature to be predicted, that is, the scheduling time $\mathscr{S}_j$. It must be noticed in Fig. 2a that the values are in a fairly wide range of $\sim$[0–1200] seconds, also presenting a long-tailed distribution. Both the wide range and the distribution skewness of $\mathscr{S}_j$ are challenging for the successful training of the regression model, since the estimation results must be precise enough in the lower extreme of the distribution while wide enough to consider the opposite extreme of the distribution.

A logarithmic transformation is applied to the values of the output parameter $\mathscr{S}_j$. Fig. 2 shows the resulting Kernel density estimation (KDE) distribution of $\mathscr{S}_j$ values after the application of a natural logarithm. As a result, the range is reduced to $\sim$[0–7], and the distribution is much closer to a normal distribution, as can be shown in Fig. 2b.

Even though the job submission time $t_{j,sub}$ contains useful information, the raw values of this feature are not enough for a predictor to make precise decisions, since the constant increase of its value makes extremely hard the establishment of patterns. $\Delta t_{j,sub}$, explained above, is taken to overcome this limitation and extract valuable patterns.

The estimation problem must be modelled as a temporal series that requires the past knowledge to be included in the regression model through the training data set, since a set of jobs submitted in the recent past (window frame) needs to be considered to improve the training process.

Fig. 3 shows an example of the possible status of the scheduling process of the previous jobs when J_j is considered. Let us use only the scheduling status, that is, the $t_{j,sched}$ parameter, for the coming explanations for the sake of clarity; however, the same explanation applies to the rest of features. At any given submission time $t_{j,sub}$ for the job J_j, any previous job $J_p, p = 1, 2, \dots, j-1$ may be in the following scheduling state:

(a) The previous job was fully scheduled. In this case, the scheduling time of the previous job, $t_{p,sched}$ occurred before $t_{j,sub}$, and therefore is known. This is the status of the jobs J_{j-4},J_{j-3}, and J_{j-1} in Fig. 3.
(b) The previous job was not fully scheduled. In this case, the scheduling time of the previous job, $t_{p,sched}$ will eventually occur in an unknown future. In Fig. 3, J_{j-2} represents this state.

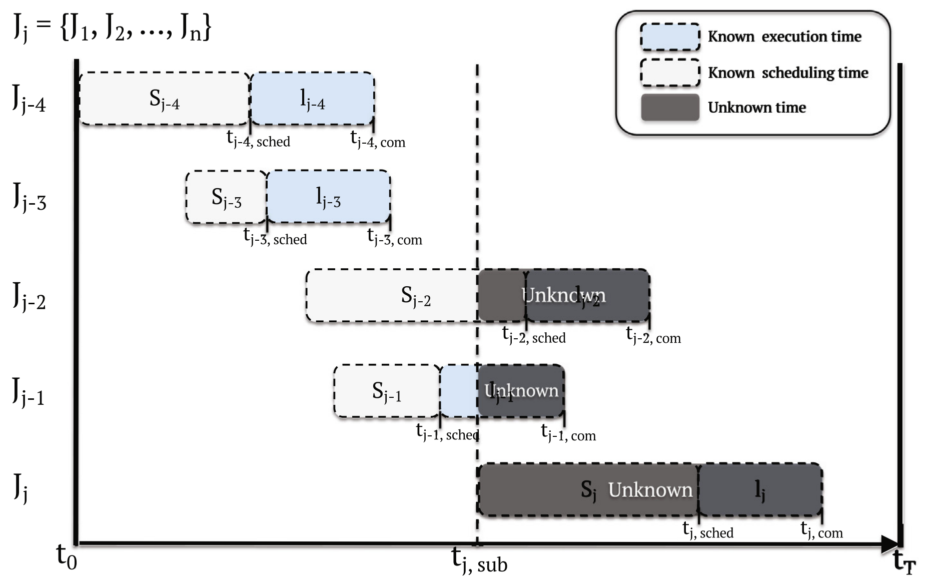


Fig. 3. Scheduling status and related times for previous jobs when J_j is considered.

In this work, we only employ known information to build our training data set. Due to this, the following two features are incorporated into every row in our training data set, which represent each J_j job, to manage the scenarios where unknown information regarding previous jobs is present: (a) $\mathscr{F}_{p,sched}$, that denotes whether the immediate past job is fully scheduled or not. $\mathscr{F}_{p,sched}$ is computed as follows to generate the training data set: $\mathscr{F}_{p,sched} = \begin{cases} 1 & \text{if } t_{j-1,sched} \leqslant t_{j,sub} \\ 0 & \text{otherwise} \end{cases}$ (b) $t_{p,sched}$, that represents the known period of scheduling time of the previous job. In our training data set, $t_{p,sched} = min\{t_{j-1,sched}, t_{j,sub}\}$. This very same process is used for job queue and completion times.

4.2. Training and evaluation of the regression model

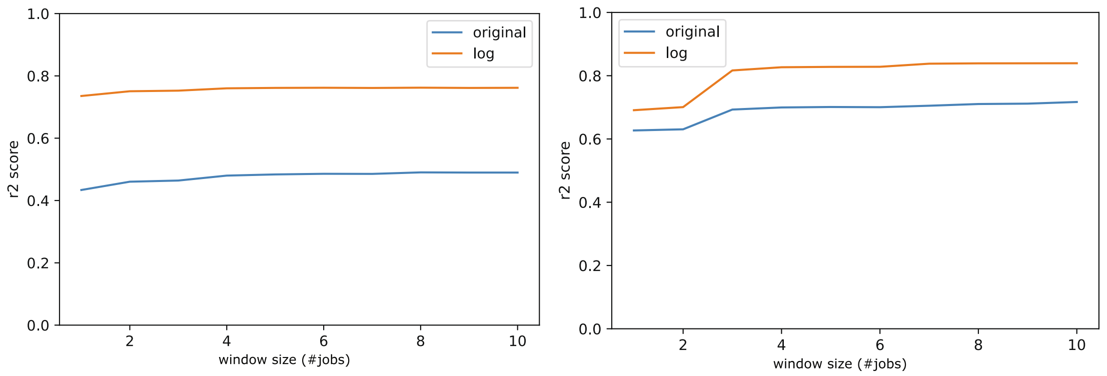
Gradient-boosting regression (Friedman, 2002), which has been used in the past with successful results in many regression problems (Zhang and Haghani, 2015; Cai et al., 2020), is an ensemble learning method that combines the estimates provided by a set of simple regressors to produce a joint prediction. Decision trees are used in this work as base estimators.

The R^2 metric is used as the main cross-validation KPI for the evaluation of the developed regression models. The metric R^2 computes a correlation coefficient between the regressor output and the expected real value, with 1 being the best possible result.

Fig. 4 shows the evolution of the values of the R^2 metric for the regression models trained with the scheduling performance traces provided by Mesos (two-level) and Omega (shared-state) resource-managing models according to the size of the past time window. Values in the range [1–10] are considered to analyse the impact of the size of the past window. The results provided by employing the original scheduling time $\mathscr{S}_j$ and its logarithmic transformation for model training are shown. The analysis performed provides the following conclusions:

- The estimation results for both the Mesos (two-level) and Omega (shared-state) resource management models are good, resulting in R^2 values higher than 0.8.
- The logarithmic transformation is the key to achieving successful results, increasing the value of R^2 by approximately 0.3 for the two-level resource management model (see Fig. 4 and by more than 0.1 for the shared-state resource management model (see Fig. 4b).
- The size of the past window time is also a key to improving the estimation results, especially for Omega, where the value of R^2 increases by approximately 0.1 when using a window size of at least 3, as shown in Fig. 4b.

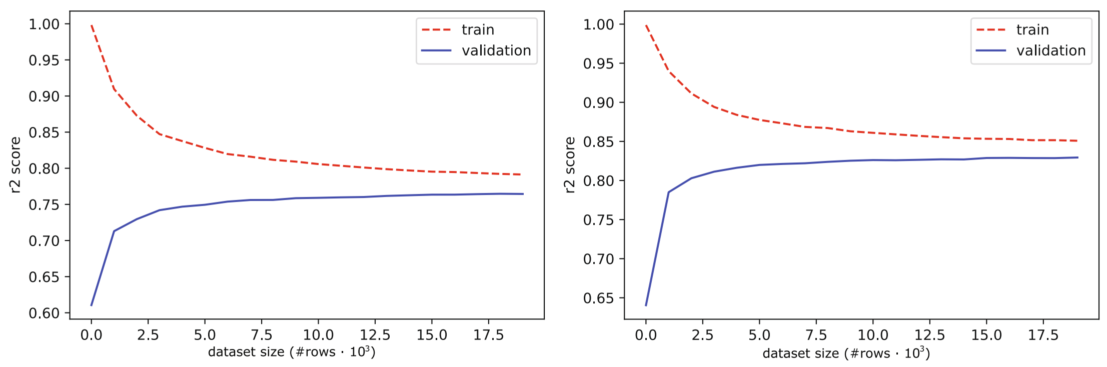
Fig. 5 shows the learning curves for the two-level and shared-state resource management models. The analysis of learning



(a) Regression model trained with the scheduling performance results provided by the two-way resource-managing model (Mesos)

(b) Regression model trained with the scheduling performance results provided by the shared-state resource-managing model (Omega)

Fig. 4. R^2 score evolution according to the size of the time window.



(a) Regression model trained with the scheduling performance results provided by the two-way resource-managing model (Mesos)

(b) Regression model trained with the scheduling performance results provided by the shared-state resource-managing model (Omega)

Fig. 5. Resulting learning curves from the regression model training process.

curves is an essential tool for the detection of overfitting. In this figure, where the R^2 score is presented versus the size of the data set, it becomes clear that the trained regression models do not overfit, since the gap between the training and validation curves strongly decreases as soon as the size of the data set increases. The stabilisation of the gap trend towards the end of the learning curves clearly shows that the size of the training data set is large enough.

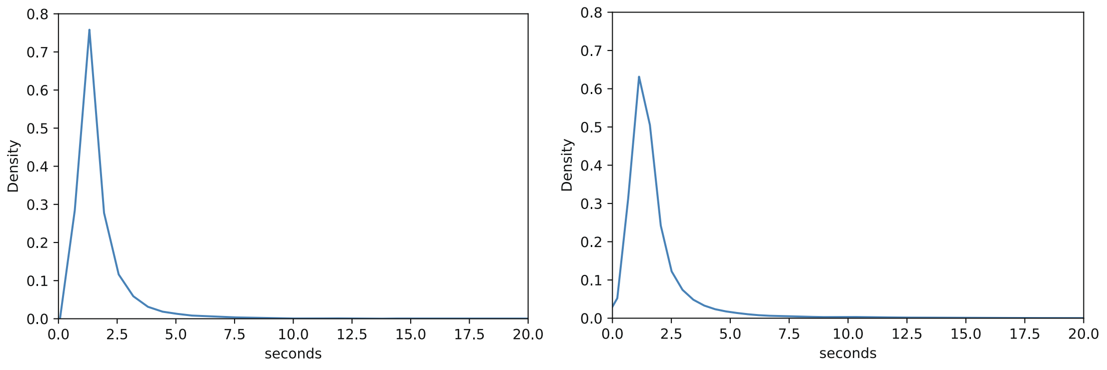
Fig. 6 shows the Kernel Density Estimation (KDE) error distribution for the regression models trained with the Omega and Mesos scheduling performance result traces, respectively. KDE is defined as the difference, in seconds, between the estimations provided by the regression models and the expected real values. These error values are computed through cross-validation, resulting in a distribution skewed toward a small value of approximately 2 s for Mesos and Omega resource managers.

The gradient boosting technique provides a set of quality indicators for each feature during the regression model training process. The feature importance must be highlighted among the aforementioned indicators, since it denotes the contribution of each feature to the predictive capacity of the regression model. In the gradient boosting model, the feature importance is calculated on the basis of the number of appearances of each feature in the set of decision trees trained for the base models, along with the position of each feature in those decision trees. The feature importance indicator may be used, in addition to the implementation of feature selection algorithms, as a first and simple step for model explanation, since some hints about how the model performs the predictions may be obtained.

Fig. 7 shows the 10 features with the highest feature importance for the prediction of the scheduling time S_j of a given job J_j for the regression models trained with the scheduling results of the two-level (see Fig. 7 and shared-state (see Fig. 7b) resource management models, respectively.

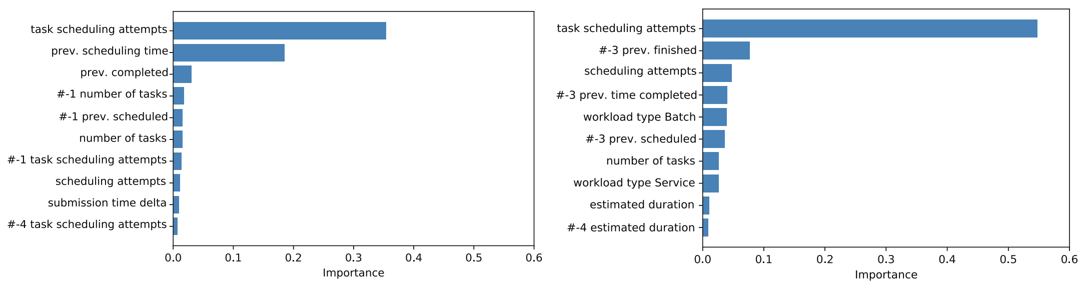
It should be noted that, even if similar, the feature importance values are different for both regression models. The following two hints may be highlighted since they represent that the model was successful in detecting the differences in terms of behaviour between the strategies of the considered models.

- The two-level resource management model (Mesos) gives more importance to the scheduling time of the previous job $\mathscr{S}_{j-1}$, while the shared-state model (Omega) takes into account only whether or not the previous job J_{j-1} has been completed. This is due to the pessimistic blocking of two-level resource manager agents (Mesos), which perform batch scheduling processes that prevent other scheduling agents from making scheduling decisions in parallel.
- The feature task scheduling attempts $\sum_{i=1}^{n_{j,\mathscr{A}}} n_{j,uns}$ is more critical for Omega (shared-state model) than for Mesos (two-level model). This is due to shared-state resource managers being more prone to perform conflicting scheduling actions, which



(a) Regression model trained with the scheduling performance results provided by the two-way resource-managing model (Mesos) (b) Regression model trained with the scheduling performance results provided by the shared-state resource-managing model (Omega)

Fig. 6. Kernel density estimation (KDE) error distribution.



(a) Regression model trained with the scheduling performance results provided by the two-way resource-managing model (Mesos) (b) Regression model trained with the scheduling performance results provided by the shared-state resource-managing model (Omega)

Fig. 7. Feature importance for the prediction of the scheduling time S_j of a given job J_j.

need to be retried and worsen the data-centre performance, when a higher number of task scheduling decisions are made in the same period of time.

5. Empirical Evaluation

5.1. Experimentation framework

In this work, we used the SCORE simulator Fernández-Cerero et al. (2018), which has been widely used (Fernández-Cerero et al., 2018; D. Fernández-Cerero et al., 2018; Fernández-Cerero et al., 2020; Fernández-Cerero et al., 2020; Fernández-Cerero et al., 2021) to analyse the behaviour of realistic hyper-scale data centres, usually composed of thousands of machines, both in terms of energy consumption and scheduling performance.

The following key performance indicators, previously explained in Section 3, are studied to analyse the performance impact of the proposed data-centre resource management model:

(a) **Measured average inter-arrival time $\overline{\Delta t_{sub,\mathscr{W}}}$**. In this work, two main workloads are considered, Batch ($\mathscr{B}$) and Service (*Service*) workloads, that is, $\mathscr{W} = \{\mathscr{B}, Service\}$. 1) **Batch workload**. Batch jobs are usually composed of a large number of tasks that consume relatively few resources for a short period and have a fixed end. Map-reduce jobs are a good example of this kind of workload. We analyse three different scale parameters $\lambda_{\mathscr{B}}$ for the Weibull distribution used to generate the inter-arrival times for the Batch workload $\frac{1}{\lambda_{\mathscr{B}}} = [120, 100, 80]s$. 2) **Service workload**. Service jobs are usually of a reduced number of tasks, which consume a high amount of resources for a longer time period and have no determined end. Web servers or DBMS jobs are a good example of this kind of workload. The arrival rate of this workload is set to be 10 times less frequent than that of the Batch workload, that is, $\lambda_{Service} = 10 \cdot \lambda_{\mathscr{B}}$. The Batch workload usually represents almost 90% of scheduled jobs, as the workload used is modelled following industry usage patterns (Tirmazi et al., 2020; Lu et al., 2017). Due to the prevalence and stronger impact on the scheduling system of this workload, the results for the Batch workload, such as $\overline{\Delta t_{sub,\mathscr{B}}}$ instead of global $\overline{\Delta t_{sub,\mathscr{W}}}$, are presented in the tables for the sake of clarity. The results for the Service workload can be found as supplementary material. (b) **Queue times**. Both $\overline{\mathscr{Q}_{1\mathscr{B}}}$, which denotes the average time that each J_j job in the Batch workload $\mathscr{W}_{\mathscr{B}}$ waits in queue until its first task is scheduled, and $\overline{\mathscr{Q}_{n\mathscr{B}}}$, which, in turn, represents the average time that the $J_{\mathscr{B}}$ jobs in the Batch workload wait in queue until its last task n_j is scheduled, can be found in the tables. The 90 percentiles of these values, $\mathscr{Q}_{1,90\%\mathscr{B}}$ and $\mathscr{Q}_{n,90\%\mathscr{B}}$, respectively, are also presented so that the long-tailed behaviour, if present, is shown. (c) **Scheduling time**. The average scheduling time of Batch jobs $\overline{\mathscr{S}_{\mathscr{B}}}$, and the 90 percentile value $\mathscr{S}_{90\%\mathscr{B}}$, are analysed. (d) **Makespan**. The average completion time of the jobs that make up the Batch workload $\overline{\mathscr{C}_{\mathscr{B}}}$, and the 90 percentile value $\mathscr{C}_{90\%\mathscr{B}}$ are shown.

In addition to these indicators, there is a set of useful indicators related to the behaviour of each resource management model studied. The following indicators are analysed and presented in the tables: (a) **Timed-out jobs**. $\overline{J_{to}}$ denotes the percentage of jobs that could not be completed due to a time-out in the scheduling process. This event occurs when scheduling bottlenecks appear of 1,000 consecutive unsuccessful task scheduling attempts or 100 consecutive unsuccessful job scheduling attempts. $\overline{J_{to}}$ is only shown in the tables when the percentage of timed-out jobs is greater than zero in any of the experiments presented. (b) **Locked resources**. Due to their pessimistic blocking strategy, both the two-level and Boost resource management models present the parameter $\overline{R_{lock}}$, which represents the average percentage of resources unavailable due to a scheduling agent performing a scheduling operation, which prevents any other scheduling agent from performing scheduling operations in parallel. (c) **Conflicted tasks**. Due to their optimistic blocking nature, both Shared-state and Boost resource management models make conflicting task scheduling attempts T_{conf}, which must be retried.

Energy consumption $\mathscr{E}$ is also shown in the tables to analyse the impact of Boost in terms of energy efficiency. The value of the scheduling algorithm times, both at the job and task level, $\mathscr{K}_J$ and $\mathscr{K}_T$, is set following the trends of the industrial literature, in 1000 and 100 ms, respectively (Schwarzkopf et al., 2013). The execution time of the Boost estimation process $\mathscr{K}_{\mathscr{D}}$ has been established through empirical measurements performed on an M1 Macbook Pro, which, after 500 runs, results in an average time of 100 ms. Each experiment simulates 15 days of operation of a data centre composed of 1,000 homogeneous machines. All experiment configurations are run 25 times. Tables 2–4 show the average performance and energy results for Boost, Mesos, and Omega, respectively.

The following statistical tests were performed in SPSS to verify the statistical significance of $\alpha = 0.05$: (1) Iglewicz and Hoaglin robust test for multiple outliers (two-sided test) with $Z = 3.5$. (2) Levene test to check the non–homogeneity of variances with a significance level of $\alpha = 0.05$. (3) Kruskal–Wallis test for independent samples, to analyse the difference in the results between pairs of resource management models with a significance level of $\alpha = 0.05$.

The results of the tests performed are provided as supplementary material.

5.2. Simulation evaluation

Fig. 8 shows the evolution of the selection of the resource management strategy performed by Boost depending on the workload inter-arrival time. The red line denotes the period where the two-level strategy is used (the same strategy employed by Mesos), whilst the blue line represents the period where the shared-state strategy is used (the same strategy employed by Omega). Although the behaviour of Boost's selection process suggests homogeneity between Fig. 8a and 8b, a main trend is present: when the workload pressure of the data centre increases (see Fig. 8b), the two-level strategy is used more frequently, while in periods of lower utilisation rates, Boost employs the shared-state strategy more frequently (see Fig. 8a).

The results in terms of performance and energy consumption for Boost, Mesos, and Omega are shown in Tables 2–4, respectively.

It must be kept in mind that the value presented in each cell is the average of 25 runs of the same experiment configuration, without outliers. Each table shows 3 rows, corresponding to a workload whose inter-arrival time is generated by means of a Weibull $\alpha = 0.5$ with three different mean inter-arrival rates: $\lambda_{\mathscr{B}} = [120, 100, 80]s$, which represent low, medium, and high utilisation rates, respectively. The experimentally measured average inter-arrival time $\overline{\Delta t_{sub,\mathscr{B}}}$ for each workload is shown in the first column. Table 2 also shows four groups, each representing one of the configurations for the decision period of the Boost resource management selection process: The first shows the results when the resource management selection process is performed on each job scheduling action, $\Delta_{\mathscr{D}} = f(\mathscr{A}_{ji}), \mathscr{A}_{ji} \in \mathscr{A}$; the following three show the results of making resource management selection decisions on a background agent in every given period: $\Delta_{\mathscr{D}}$ = [30,60,120]s.

The results of the average scheduling time $\overline{\mathscr{S}_{\mathscr{B}}}$, the main scheduling performance indicator, can be analysed to expose the main trends in the Boost behaviour. Although our resource management model improves the performance of the data centre in all scenarios, Boost excels as soon as the decision period for the

Table 2
Performance results of Boost for Batch Workloads.

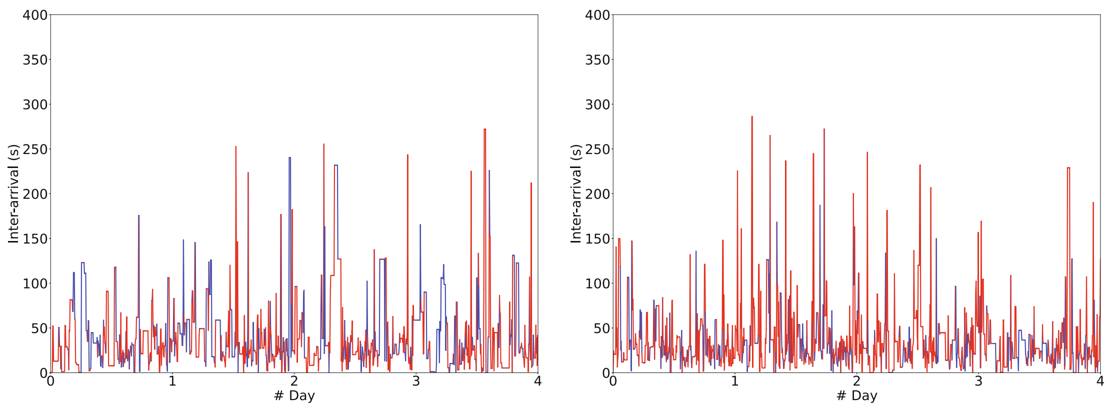
$\overline{\Delta t_{sub,\mathscr{B}}}$ (s)	$\overline{Q_{1\mathscr{B}}}$ (s)	$\overline{\mathscr{Q}_{n\mathscr{B}}}$ (s)	$\mathscr{Q}_{1,90\%\mathscr{B}}$ (s)	$\mathscr{Q}_{n,90\%\mathscr{B}}$ (s)	$\overline{\mathscr{S}_{\mathscr{B}}}$ (s)	$\mathscr{S}_{90\%\mathscr{B}}$ (s)	$\overline{\mathscr{C}_{\mathscr{B}}}$ (s)	$\mathscr{C}_{90\%\mathscr{B}}$ (s)	$\overline{R_{lock}}$ (%)	T_{conf}	$\mathscr{E}$ (MWh)
$\Delta_{\mathscr{D}} = f(\mathscr{A}_{ji}), \mathscr{A}_{ji} \in \mathscr{A}$											
104,47	8,65	22,27	28,37	68,67	101,17	246,25	190,97	426,53	15,89	922	95,77
86,48	9,30	26,51	30,13	80,71	110,91	270,76	200,45	455,77	18,43	1249	96,38
68,69	10,53	35,51	33,20	104,10	130,80	323,19	220,66	503,67	22,06	1843	97,17
$\Delta_{\mathscr{D}}$ = 30 s											
104,88	12,31	28,64	34,74	84,64	108,28	271,00	198,37	455,94	14,29	1051	95,85
86,27	14,99	37,11	42,66	108,86	123,02	308,89	212,92	489,67	16,94	1415	96,35
68,92	20,01	53,65	56,79	156,31	152,00	385,83	242,30	561,76	20,13	2030	97,15
$\Delta_{\mathscr{D}}$ = 60 s											
104,77	17,78	34,81	52,43	101,31	111,52	277,51	201,40	461,53	13,82	1025	95,86
86,75	21,63	45,39	60,48	131,38	129,05	324,97	219,02	504,76	16,05	1378	96,40
68,87	28,79	65,00	80,93	187,21	160,79	410,88	250,79	582,59	19,34	1959	97,15
$\Delta_{\mathscr{D}}$ = 120 s											
104,92	26,20	44,13	81,21	126,53	117,24	288,45	207,15	468,60	13,57	950	95,84
86,62	31,78	57,02	93,89	162,86	137,96	345,64	227,60	523,54	15,73	1306	96,38
69,20	41,73	82,22	117,22	229,64	177,16	449,54	267,32	618,41	18,93	1810	97,14

Table 3
Performance results of the two-level resource management model for Batch Workloads.

$\overline{\Delta t_{sub,\mathscr{B}}}$ (s)	$\overline{\mathscr{Q}_{1\mathscr{B}}}$ (s)	$\overline{\mathscr{Q}_{n\mathscr{B}}}$ (s)	$\mathscr{Q}_{1,90\%\mathscr{B}}$ (s)	$\mathscr{Q}_{n,90\%\mathscr{B}}$ (s)	$\overline{\mathscr{S}_{\mathscr{B}}}$ (s)	$\mathscr{S}_{90\%\mathscr{B}}$ (s)	$\overline{\mathscr{C}_{\mathscr{B}}}$ (s)	$\mathscr{C}_{90\%\mathscr{B}}$ (s)	J_{to} (%)	$\overline{R_{lock}}$ (%)	$\mathscr{E}$ (MWh)
104,55	67,71	72,31	198,80	211,36	128,49	312,15	218,53	469,87	0,00	20,82	95,84
86,63	77,37	85,27	222,61	244,14	148,99	365,10	238,92	516,08	0,00	23,82	96,40
68,78	19e3	118e3	68e3	341e3	246e3	622e3	251e3	615e3	14,57	54,50	95,06

Table 4
Performance results of the shared-state resource management model for Batch Workloads.

$\overline{\Delta t_{sub,\mathscr{B}}}$	$\overline{\mathscr{Q}_{1\mathscr{B}}}$	$\overline{\mathscr{Q}_{n\mathscr{B}}}$	$\mathscr{Q}_{1,90\%\mathscr{B}}$	$\mathscr{Q}_{n,90\%\mathscr{B}}$	$\overline{\mathscr{S}_{\mathscr{B}}}$	$\mathscr{S}_{90\%\mathscr{B}}$	$\overline{C_{\mathscr{B}}}$	$\mathscr{C}_{90\%\mathscr{B}}$	T_{conf}	$\mathscr{E}$
(s)	(s)	(s)	(s)	(s)	(s)	(s)	(s)	(s)		(MWh)
104,61	38,97	89,44	134,59	251,26	179,45	426,65	269,50	603,35	2024	95,84
86,50	73,75	170,58	252,51	460,27	270,97	640,41	360,91	800,18	2779	96,39
68,86	168,55	394,39	542,01	1017,99	510,32	1206,66	600,28	1348,31	4097	97,14



(a) Mean inter-arrival time $\lambda_{\mathcal{B}} = 120s$ (b) Mean inter-arrival time $\lambda_{\mathcal{B}} = 80s$

Fig. 8. Evolution of the Boost's selection of resource-managing strategy depending on the workload inter-arrival time. In red: two-level (Mesos), in blue: shared-state (Omega). It can be noticed that when data-centre workload pressure increases, more red periods are present, which means that the two-level strategy is employed more frequently.

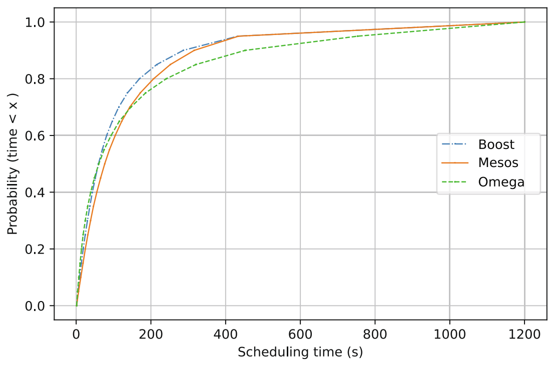
selection of the resource management model decreases. Table 2 perfectly shows this trend: For workloads with $\overline{\Delta t_{sub,\mathscr{B}}} \approx 85$ s, it can be observed that, for a less frequent decision period $\Delta_{\mathscr{D}}$ = 120 s, the average scheduling time $\overline{\mathscr{S}_{\mathscr{B}}}$ is reduced by $\sim -10\%$ compared to Mesos (137.96 vs. 148.99 s), and almost half compared to Omega (137.96 vs. 270.97 s). However, when the resource-management strategy selection decision is made in every scheduling action, the reduction increases to $\sim -35\%$ (110.91 vs. 148.99 s) and $\sim -60\%$ (110.91 vs. 270.97 s) compared to Mesos and Omega, respectively. Boost improves the performance of the data centre, especially when the data centre is under pressure. Let us analyse the increase in the average scheduling time $\overline{\mathcal{S}_{\mathcal{B}}}$ between the first two result rows of Tables 2–4, to confirm this behaviour. The average Mesos scheduling time $\overline{\mathcal{S}_{\mathcal{B}}}$ shown in Table 3 increases by almost 20% (128.49 vs. 148.99 s) when the measured average workload inter-arrival time $\overline{\Delta t_{sub,\mathcal{B}}}$ is reduced from $\sim$ 105 to $\sim$ 85 s, while Omega suffers a higher performance penalty of $\sim$ +50% (179.45 vs. 270.97 s), as shown in Table 4. This extra penalty is imposed by the $\sim$ 750 extra task scheduling actions re-tried due to the conflicts related to the higher frequency of scheduling actions, as can be seen in the results of T_{conf} (2,024 vs. 2,779). Boost overcomes the bottleneck related to the pessimistic blocking strategy, such as the two-level model used by Mesos. Even though four parallel Batch scheduling agents are working, when the data centre is under high workload pressure, Mesos cannot keep up with the incoming workload. This behaviour is shown in Table 3 in the third row, where all performance indicators are three orders of magnitude worse than those achieved in environments of lower use. It should be noted that this case is the only scenario in which almost 15% of the workload is left in the queue, as shown by the J_{to} parameter. The rest of the tables do not show the J_{to} parameter because its value is always 0.

Fig. 9 visually summarises the trends described above. It becomes evident that the increase in workload pressure has a lesser negative impact on Boost than the industry alternatives when the columns are compared. It is also shown in the comparison between rows that the increase in frequency in the selection of resource-management strategy makes Boost's CDF line more vertical, which means greater performance. In Fig. 9b it is shown that 80% of the jobs are fully scheduled in less than approximately 270 s, when decisions are made every 120 s. On the other hand, in Fig. 9d, 80% of the jobs are fully scheduled in less than 200 s when decisions are made in each job scheduling action.

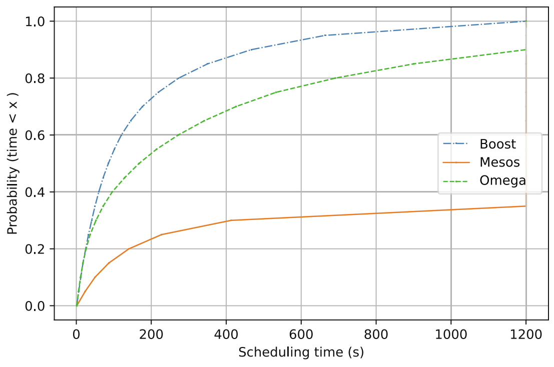
Boost performance gains are not related to the improvement of shared-state or two-level scheduling strategies, but just to the selection of the best resource management strategy for each period. This statement becomes evident when the rate of increment of conflicts T_{conf} of Boost is compared to that of Omega: the increment of both follows the same trend, increasing by $\sim$ +30% when

Table 5
Summary and comparison of the main performance results between Boost $\Delta_{\mathcal{D}} = f(\mathcal{A}_{ji}), \mathcal{A}_{ji} \in \mathcal{A}$, Mesos, and Omega. The improvement of Boost is computed as a comparison against the best of the competitors.

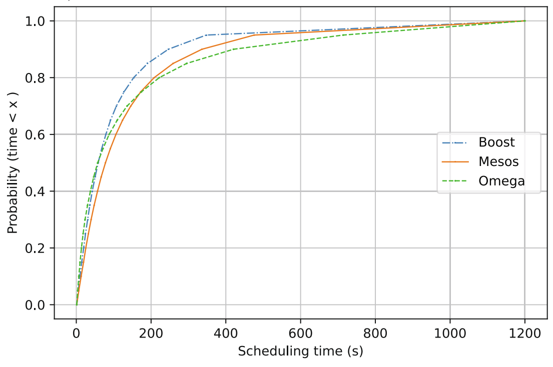
Resource manager	$\overline{\mathcal{Q}_{n\mathcal{B}}}$ (s)	$\overline{\mathcal{S}_{\mathcal{B}}}$ (s)	$\overline{\mathcal{C}_{\mathcal{B}}}$ (s)
	Low workload pressure. $\overline{\Delta t_{sub,\mathcal{B}}} \approx 105s$		
Boost	22,27	101,17	190,97
Mesos	72,31	128,48	218,53
Omega	89,44	179,45	269,5
Improvement	69,20%	21,26%	12,61%
	Medium workload pressure. $\overline{\Delta t_{sub,\mathcal{B}}} \approx 85s$		
Boost	26,51	110,91	200,45
Mesos	85,27	148,99	238,92
Omega	170,58	270,97	360,91
Improvement	68,91%	25,56%	16,10%
	High workload pressure. $\overline{\Delta t_{sub,\mathcal{B}}} \approx 70s$		
Boost	35,51	130,8	220,66
Mesos	118e3	246e3	251e3
Omega	394,39	510,32	600,28
Improvement	91,00%	74,37%	63,24%



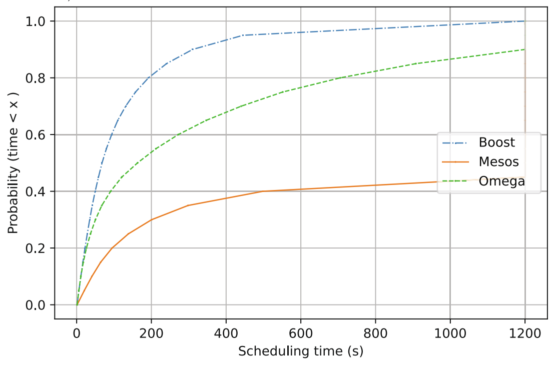
(a) Decision period $\Delta_{\mathcal{D}} = 120$ s, Inter-arrival time $\overline{\Delta t_{sub,\mathcal{B}}} \approx 105s$



(b) Decision period $\Delta_{\mathcal{D}} = 120$ s, Inter-arrival time $\overline{\Delta t_{sub,\mathcal{B}}} \approx 70s$



(c) Decision every job scheduling action $\Delta_{\mathcal{D}} = f(\mathcal{A}_{ji}), \mathcal{A}_{ji} \in \mathcal{A}$, Inter-arrival time $\overline{\Delta t_{sub,\mathcal{B}}} \approx 105s$



(d) Decision every job scheduling action $\Delta_{\mathcal{D}} = f(\mathcal{A}_{ji}), \mathcal{A}_{ji} \in \mathcal{A}$, Inter-arrival time $\overline{\Delta t_{sub,\mathcal{B}}} \approx 70s$

Fig. 9. Comparative of the cumulative distribution function (CDF) of the scheduling time according to the decision period and the workload inter-arrival time.

the mean inter-arrival time is reduced from $\sim$ 105 to $\sim$ 70 s, and by $\sim$ +50% when it is reduced from $\sim$ 85 to $\sim$ 70 s.

In terms of energy consumption, Boost does not have a negative impact on energy efficiency. The only difference in terms of energy consumption is the reduction shown in the last row of Table 3 and is due to Mesos being unable to execute the incoming workload, leaving the data centre more idle.

Table 5 presents a summary in which the proposed model shows its ability to improve the overall scheduling performance without negatively impacting energy efficiency by successfully selecting the best scheduling strategy for each period and operating environment, especially with a high frequency of selection decisions.

6. Conclusions

In this paper, we have presented a novel model that predicts the scheduling performance of a set of resource managers based on the gradient-boosting regression technique, so that the optimal one can be selected for a given operation period. The results reported here confirm that our proposal reduces the average scheduling time by approximately 20% compared to Mesos and by more than 50% compared to Omega. The usage of synthetic workload traces that represent real-world hyper-scale data centre operation environments and the analysis of the most popular industry-level resource managers enables the adoption of the proposed model in similar data centres.

We have devised a method for the transformation of a workload trace into a time-series data set suitable for the training of artificial intelligence models. The results of this study explain that some features are key to the quality of the predictions. Among them: the number of task-scheduling attempts and the scheduling state of previous jobs. In addition, the logarithmic transformation of the scheduling time is a key to the quality of the predictions. As a result, our trained regression model achieved a high R^2 score, higher than 0.8.

One potential application of our results could be the creation of a background agent that estimates and selects the best resource manager. Although the results clearly state that the real-time agent provides the best performance results, this strategy can impose higher stress on the scheduling agent and cause bottlenecks. However, the background agent does not impose any additional stress on the scheduling agents, while performance degradation is minor when short decision periods are considered.

For future work, the following limitations should be considered.

- No real-world traces have been used, but synthetically generated ones that follow the same realistic industry patterns have been used. As a consequence, workload-trace simplifications restrict the application of even more complex artificial intelligence models. In future work, performance techniques that provide confidence errors by nature should be employed for workload generation, in addition to the approximation through simulation.
- Other resource-managing models and data-centre configurations should be explored in future work.
- A logarithmic transformation was used to improve the estimation process, leading to a loss of precision for long-tailed distribution values. In future work, we could coordinate two versions of the model, with and without the logarithmic transformation, to fine-tune the estimations of such long-tailed values. In addition, more powerful machine learning algorithms should be explored so that no logarithmic transformation is needed to obtain high-precision estimates.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Project RTI2018-098062-A-I00, funded by FEDER/ Spanish Ministry of Science and Innovation - National Research Agency.

References

M. Amiri, L. Mohammad-Khanli, R. Mirandola, An online learning model based on episode mining for workload prediction in cloud, Future Generation Computer Systems 87 (2018) 83–101, ISSN 0167–739X, doi: 10.1016/j.future.2018.04.044, URL:https://www.sciencedirect.com/science/article/pii/S0167739X18300712.

Cai, J., Xu, K., Zhu, Y., Hu, F., Li, L., 2020. Prediction and analysis of net ecosystem carbon exchange based on gradient boosting regression and random forest. Applied energy 262, 114566.

C. Reiss, A. Tumanov, G.R. Ganger, R.H. Katz, M.A. Kozuch, Heterogeneity and dynamicity of clouds at scale: Google trace analysis, in: Proceedings of the Third ACM Symposium on Cloud Computing, ACM, 7, 2012.

B.K. Dewangan, A. Agarwal, M. Venkatadri, A. Pasricha&4, Workload Classification and Clustering through Modified KNN Algorithm in Cloud, International Journal of Scientific Research in Computer Science Applications and Management Studies 8, ISSN 2319–1953.

D. Fernández-Cerero, A. Fernández-Montes, J. Kolodziej, L. Lefèvre, Quality of cloud services determined by the dynamic management of scheduling models for complex heterogeneous workloads, in: 2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC), IEEE, 210–219, 2018.

Elrotub, M., Gherbi, A., 2018. Virtual machine classification-based approach to enhanced workload balancing for cloud computing applications. Procedia computer science 130, 683–688.

Fernández-Cerero, D., Fernández-Montes, A., Jakóbik, A., Kołodziej, J., Toro, M., 2018. SCORE: Simulator for cloud optimization of resources and energy consumption. Simul. Model. Pract. Theory 82, 160–173.

Fernández-Cerero, D., Jakóbik, A., Grzonka, D., Kołodziej, J., Fernández-Montes, A., 2018. Security supportive energy-aware scheduling and energy policies for cloud environments. Journal of Parallel and Distributed Computing 119, 191–202.

Fernández-Cerero, D., Fernández-Montes, A., Ortega, J.A., 2018. Energy policies for data-center monolithic schedulers. Expert Syst. Appl. 110, 170–181.

Fernández-Cerero, D., Fernández-Montes, A., Jakóbik, A., 2020. Limiting global warming by improving data-centre software. IEEE Access 8, 44048–44062.

Fernández-Cerero, D., Ortega-Irizo, F.J., Fernández-Montes, A., Velasco-Morente, F., 2020. Bullfighting extreme scenarios in efficient hyper-scale cluster computing. Cluster Computing, 1–17.

D. Fernández-Cerero, F.J. Ortega, A. Jakóbik, A. Fernández-Montes, DISCERNER: Dynamic selection of resource manager in hyper-scale cloud-computing data centres, Future Generation Computer Systems 116 (2021) 190–199, ISSN 0167–739X, doi: 10.1016/j.future.2020.10.031, URL:https://www.sciencedirect.com/science/article/pii/S0167739X20330156.

Friedman, J.H., 2002. Stochastic gradient boosting. Computational statistics & data analysis 38 (4), 367–378.

Gao, J., Wang, H., Shen, H., 2020. Machine Learning Based Workload Prediction in Cloud Computing. In: 2020 29th International Conference on Computer Communications and Networks (ICCCN), pp. 1–9. https://doi.org/10.1109/ICCCN49398.2020.9209730.

M. Genkin, F. Dehne, P. Navarro, S. Zhou, Machine-Learning Based Spark and Hadoop Workload Classification Using Container Performance Patterns, in: C. Zheng, J. Zhan (Eds.), Benchmarking, Measuring, and Optimizing, Springer International Publishing, Cham, 118–130, ISBN 978-3-030-32813-9, 2019.

Gog, I., Schwarzkopf, M., Gleave, A., Watson, R.N.M., Hand, S., 2016. Firmament: Fast, Centralized Cluster Scheduling at Scale, in: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), USENIX Association, Savannah, GA, 99–115, ISBN 978-1-931971-33-1, URL:https://www.usenix.org/conference/osdi16/technical-sessions/presentation/gog.

Hindman, B., Konwinski, A., Zaharia, M., Ghodsi, A., Joseph, A.D., Katz, R.H., Shenker, S., Stoica, I., 2011. Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center. NSDI 11, 22.

W. Iqbal, A. Erradi, A. Mahmood, Dynamic workload patterns prediction for proactive auto-scaling of web applications, Journal of Network and Computer Applications 124 (2018) 94–107, ISSN 1084-8045, doi: https://doi.org/10.1016/j.jnca.2018.09.023, URL:https://www.sciencedirect.com/science/article/pii/S1084804518303102.

K. Karanasos, S. Rao, C. Curino, C. Douglas, K. Chaliparambil, G.M. Fumarola, S. Heddaya, R. Ramakrishnan, S. Sakalanaga, Mercury: Hybrid centralized and distributed scheduling in large shared clusters, in: 2015 {*USENIX*} Annual Technical Conference ({*USENIX*}{*ATC*} 15), 485–497, 2015.

J. Kumar, A.K. Singh, Workload prediction in cloud using artificial neural network and adaptive differential evolution, Future Generation Computer Systems 81 (2018) 41–52, ISSN 0167–739X, doi: 10.1016/j.future.2017.10.047, URL:https://www.sciencedirect.com/science/article/pii/S0167739X17300444.

Li, C., Liu, J., Lu, B., Luo, Y., 2021. Cost-aware automatic scaling and workload-aware replica management for edge-cloud environment. J. Network Computer Appl. 180, 103017.

D. Lo, L. Cheng, R. Govindaraju, P. Ranganathan, C. Kozyrakis, Improving Resource Efficiency at Scale with Heracles, ACM Transactions on Computer Systems (TOCS) 34 (2016) 6:1–6:33, URL:http://dl.acm.org/citation.cfm?id=2882783.

Lu, C., Ye, K., Xu, G., Xu, C., Bai, T., 2017. Imbalance in the cloud: An analysis on Alibaba cluster trace. In: 2017 IEEE International Conference on Big Data (Big Data), pp. 2884–2892.

E. Patel, D.S. Kushwaha, Clustering Cloud Workloads: K-Means vs Gaussian Mixture Model, Procedia Computer Science 171 (2020) 158–167, ISSN 1877-0509, doi: https://doi.org/10.1016/j.procs.2020.04.017, URL:https://www.sciencedirect.com/science/article/pii/S1877050920309820, third International Conference on Computing and Network Communications (CoCoNet'19).

Schwarzkopf, M., Konwinski, A., Abd-El-Malek, M., Wilkes, J., 2013. Omega: flexible, scalable schedulers for large compute clusters. In: Proceedings of the 8th ACM European Conference on Computer Systems. ACM, pp. 351–364.

Singh, P., Gupta, P., Jyoti, K., 2019. Tasm: technocrat arima and svr model for workload prediction of web applications in cloud. Cluster Computing 22 (2), 619–633.

Tang, X., Liao, X., Zheng, J., Yang, X., 2018. Energy efficient job scheduling with workload prediction on cloud data center. Cluster Computing 21 (3), 1581–1593.

M. Tirmazi, A. Barker, N. Deng, M.E. Haque, Z.G. Qin, S. Hand, M. Harchol-Balter, J. Wilkes, Borg: the Next Generation, in: EuroSys'20, Heraklion, Crete, 2020.

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

G.M. Wamba, Y. Li, A.-C. Orgerie, N. Beldiceanu, J.-M. Menaud, Cloud Workload Prediction and Generation Models, in: 2017 29th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), 89–96, 2017, doi: 10.1109/SBAC-PAD.2017.19.

Wang, Z., Li, H., Li, Z., Sun, X., Rao, J., Che, H., Jiang, H., 2019. Pigeon: An effective distributed, hierarchical datacenter job scheduler. In: Proceedings of the ACM Symposium on Cloud Computing, pp. 246–258.

Wu, Y., Wu, H., Zhang, W., Xu, Y., Wei, J., Zhong, H., 2018. HW3C: a heuristic based workload classification and cloud configuration approach for big data analytics. In: Proceedings of the Tenth Asia-Pacific Symposium on Internetware, pp. 1–10.

Zhang, Y., Haghani, A., 2015. A gradient boosting method to improve travel time prediction. Transportation Research Part C: Emerging Technologies 58, 308–324.

Zhang, Q., Yang, L.T., Yan, Z., Chen, Z., Li, P., 2018. An Efficient Deep Learning Model to Predict Cloud Workload for Industry Informatics. IEEE Trans. Industr. Inf. 14 (7), 3170–3178. https://doi.org/10.1109/TII.2018.2808910.