Future Generation Computer Systems 116 (2021) 190–199

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

DISCERNER: Dynamic selection of resource manager in hyper-scale cloud-computing data centres

Damián Fernández-Cerero [a,*], F. Javier Ortega [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 9 July 2020
Received in revised form 1 October 2020
Accepted 31 October 2020
Available online 4 November 2020

Keywords:
Data centre
Decision theory
Machine learning
Cloud computing

ABSTRACT

Data centres constitute the *engine* of the Internet, and run a major portion of large web and mobile applications, content delivery and sharing platforms, and Cloud-computing business models. The high performance of such infrastructures is therefore critical for their correct functioning. This work focuses on the improvement of data-centre performance by dynamically switching the main data-centre governance software system: the resource manager. Instead of focusing on the development of new resource-managing models as soon as new workloads and patterns appear, we propose DISCERNER, a decision-theory model that can learn from numerous data-centre execution logs to determine which existing resource-managing model may optimise the overall performance for a given time period. Such a decision-theory system employs a classic machine-learning classifier to make real-time decisions based on past execution logs and on the current data-centre operational situation. A set of extensive and industry-guided experiments has been simulated by a validated data-centre simulation tool. The results obtained show that the values of key performance indicators may be improved by at least 20% in realistic scenarios.

1. Introduction

Data centres are considered the *engine* of the Internet. From websites and services to mobile apps, everything is supported by some kind of service hosted on these huge infrastructures. Thus, data centres are required to be effective, reliable, efficient and performant. Moreover, data centres are also used for performing back-end and computing-intensive tasks that are requested from stakeholders for various purposes.

Data centres are governed by software solutions that deal with the management of their computing, storage and networking resources. These software solutions are called resource managers, which constitute the main agents responsible for the operation of data centres. Resource-managing models can be grouped in terms of their architecture: centralised, distributed, or hybrid solutions.

Nowadays, centralised solutions are the most popular and widespread in the majority of data centres in production. These centralised approaches propose a single point of decision-making in order to decide which resources should be selected for the deployment of tasks and services. Within centralised resource managers, however, there are three main alternatives: the so-called monolithic model, the two-level model, and the shared-state model. Each alternative is shown to be optimal for one particular kind of workload, as explained in detail in [1].

However, the larger the data centre is, the more heterogeneous the data-centre resources and the workload requirements and patterns become [2]. In such environments of mixed-workloads, current resource managers struggle to perform optimally for each type of workload. Although, resource managers have been proposed and developed in response to the new kinds of workloads and requirements that data centres address, such tightly coupled models are insufficiently flexible and suitable for ever-evolving scenarios, such as those that are present in Cloud Computing and run by hyper-scale data centres.

Therefore, it would undoubtedly be useful if data centres could rely on any of the resource managers proposed in these dynamic environments to date, since they are each useful for their specific type of workload. In scenarios where the workload is heterogeneous and workload patterns and characteristics are mutating during operation time, it appears logical to employ a decision-support system that could apply such resource-managing models depending on the characteristics of the incoming workload and the data-centre situation, as an alternative to the development of new fine-tuned specialised resource managers. Therefore, this paper presents the following innovations:

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

https://doi.org/10.1016/j.future.2020.10.031
0167-739X/

- A decision-support model for the dynamic switch of data-centre resource-managing models at run-time that is in accordance with the characteristics and purpose of the incoming workload so that the data-centre performance is improved.
- A classic machine-learning classifier for the detection of the optimal resource-managing model to be applied based on past workloads.
- An in depth and realistic analysis of the performance of the proposed model through extensive simulation based on industry environments, and comparison between the proposed model and the most commonly used industry solutions.

As a result, it is shown that the dynamic switching of resource managers may be provided by a model that significantly improves both the behaviour in terms of homogeneity and also the performance compared to the isolated utilisation of the most popular resource managers. Through the simulation of various realistic workloads of very unclear patterns, including extreme scenarios, we show that performance is improved by 20% at the very least for average scenarios, according to the widely accepted key performance indicators, such as job queue times and makespan.

As a potential application of the proposed model, an innovative type of resource management system, built on top of the current systems, such as that used in Kubernetes, could be developed to dynamically apply the optimal resource-managing model in accordance with the incoming workload.

The rest of the paper is organised as follows: the related work is described in Section 2. In Section 3, we propose the theoretical decision-theory model to be employed as the foundation of the implemented model. Section 4 presents the research methodology and experimentation framework. Empirical results are presented and discussed in Section 5, where we consider 3 incremental workloads and 3 workload-arrival patterns in order to compare the performance results of the resource-managing models. Conclusions and future work are laid out in Section 6.

2. Related work

Resource managers can be classified as centralised, distributed, or hybrid [1,3]. The majority of the actual industry data centres rely on centralised approaches, in particular:

1. Kubernetes and Docker swarm [4,5] as monolithic solutions;
2. DC/OS (Mesos), YARN (which can also act as monolithic), and Tupperware as two-level approaches [6–8];
3. Google Omega and Google Borg as a shared-state approaches [9,10].

From these approaches, this work is based on the dynamic switch of the most commonly used resource-managing models: the two-level and shared-state models represented by Mesos and Omega, respectively. On one hand, the Mesos resource manager follows a pessimistic approach where the centralised coordinator blocks the whole cluster every time a scheduling decision is made by any scheduler. This coordinator offers resources to the various frameworks, such as Hadoop and MPI. Scheduling decisions may therefore be suboptimal, since the total cluster state is not made available to the framework scheduler involved; only the resources offered are made available.

On the other hand, the Omega scheduler follows an optimistic approach where the centralised coordinator manages several concurrent schedulers that are able to operate simultaneously. Each scheduler makes scheduling decisions by using a stale copy of the whole cluster state. These schedulers then commit atomic scheduling transactions to the centralised cluster. If these transactions result in a conflict, then the local copy of the cluster state used by the scheduler is updated and the scheduling process is retried.

With respect to the dynamic selection of schedulers, in this paper, the use of a supervised machine-learning system is proposed which builds a model intended to decide which type of scheduler is the best with each job-arrival to the data centre. Nowadays, there are a number of classification techniques, from the first Naive Bayes [11] and Hidden Markov models [12] to more recent approaches, such as Support Vector Machine [13], Random Forest [14] and XGBoost [15]. These supervised algorithms take a dataset of feature vectors as their input, each corresponding to an instance or example and tagged with a category. These algorithms then process each instance in order to build a model capable of inferring the category of other new, unseen instances. The performance of those algorithms depends on many factors, such as the size and representativeness of the dataset, the type of data of the features (quantitative, ordinal, or categorical), the number of categories, and the quality of the feature set in relation to the categories.

Regarding the problem at hand, once the data from past workloads of previous simulations had been collected, we realised that our system could benefit from the application of a feature-selection algorithm in order to determine which features are useful and which are noisy for our classification problem. There are also a number of approaches to this problem, grouped into two main groups [16,17]: filter methods, that strive to maintain the most relevant features, and only addresses correlations between the features and the class in the dataset; and wrapper methods, which use an external classifier to evaluate the goodness of the chosen features. In this work, the well-known Correlation Feature-Selection method (CFS) [18] has been applied, which is based on the assumption that "Good feature subsets contain features highly correlated with the classification, yet uncorrelated to each other".

In summary, in this work the current state of the art is extended by:

- Proposition of a model based on decision theory focused on the dynamic selection of a resource manager depending on the data-centre operation performance indicators and the characteristics of the workload.
- Evaluation of multiple classic machine-learning models to classify the optimal resource-managing model to be employed in a given time period.
- Deep and extensive analysis and comparison of the proposed model against the most commonly used resource-managing models for various realistic data-centre scenarios and workloads.

3. Theoretical framework

This work is focused on the dynamic switch of the data-centre resource-managing model whose choice depends on the data centre and workload characteristics. The final goal involves the minimisation of the performance impact of each approach on inconvenient scenarios by utilising the best-fit selection. The proposed model is superior to other proposals which implement ad hoc algorithms to match a scheduling path with a given task. Instead, in our approach, the dynamic resource-management system learns from past executions and virtually any resource-management model can be added to the catalogue of available resource-management models. The first stage of the proposed solution involves the formulation of the decision-theory models that describe the problem under consideration. The second stage includes selecting the variables that need to be simulated in order

to find the numerical solution to the problem. The next step of presented method involves performing necessary experiments and processing the results in order to obtain the values that cannot be calculated using analytical tools.

The main objective is the selection of one action from among a finite set of known actions, whose consequences depend on some unknown state of the external environment. The impact of the external environment is represented by the incoming workload to be processed by the data centre. Such a workload comprises a set of jobs which has to be deployed and executed by the data-centre resources [19]. Let us denote the single workload as W and the set of all possible workloads as $\mathbf{W}$. Each workload is composed of a number of jobs $W = \{J_j\}_{j=1}^{n}$, and each job is composed of tasks $\mathcal{T}_j = \{t_{ji}\}_{i=1}^{n_j}$.

Let A represent a set of actions, from which a single set item a is be chosen. Given a set of states of the data-centre and the incoming workloads, the main objective involves ascertaining which action is the most beneficial. The reason behind the selection from among the set of actions is linked to their related quantitative performance impact, measured by the regret loss function L. The consequences depend both on the unknown state of the world, and the actions taken, therefore:

$$L(W, a) = \sup_{a'(W)} u(a'(W)) - u(a(W)), (W, a) \in (\mathbf{W} \otimes A) \quad (1)$$

where u represents the utility function, which depends on the actions taken in the W state: $a(W)$ and $a'(W) \in A$. The regret loss function measures the inappropriateness of a given action a for a state W, and it is non-negative.

In this paper, the two most commonly employed centralised resource-management models are considered: two-level resource managers, such as Mesos; and shared-state resource managers, such as Google Omega. The decision to be made is whether the data centre should switch between resource-management models (a_1 action) or not (a_2 action). Such decisions may be denoted as $A = \{a_1, a_2\}$. The workload is characterised by inter-arrival times λ_W: $W = (\{J_j\}_{j=1}^{n}, \lambda_W)$. Jobs are composed of tasks $\mathcal{T}_j = \{t_{ji}\}_{i=1}^{n_j}$, whose attributes are explained in Section 4.2. The loss function, which represents the negative performance impact, is modelled as follows:

- Let $\overline{R_{util}}(W, a)$ denote the percentage of data-centre resources utilised on average in state W under action a. Therefore, $ua(W) = 100 - \overline{R_{util}}(W, a)$, and $\sup_{a'(W)} u(a'(W))$ denote the best possible scenario in terms of efficiency of data-centre utilisation. The $L(W, a)$ describes the deterioration in the performance of the data-centre in the case of choosing action a instead of the most beneficial action $a_0 = argmax_{a'(W)} u(a'(W))$.
- Let $\overline{q_1}(W, a)$ represent the average job queue time until the first task is scheduled. Therefore, $u(a(W)) = 0 + \overline{q_1}((W, a))$ and $\inf_{a'(W)} \overline{q_1}((W, a'))$ denote the best possible scenario, which minimises the job queue time. $L(W, a)$ is equal to the deterioration in the performance of the data-centre in the case of choosing action a instead of the most beneficial action: $a_0 = argmin_{a'(W)} u(a'(W))$.

The main aim of the selection from among the possible actions is to find the action that minimises the performance deterioration given the known data-centre load W:

$$a_0 = argmin_a L(W, a) \quad (2)$$

The next task is to find the action that minimises the maximum performance deterioration only:

$$a_M = argmin_a max_W L(W, a) \quad (3)$$

Subsequent to the application of the expected utility principle [20], the prior expected deterioration may be calculated as follows:

$$L_{\Pi}(a) = \int^{W} L(W, a)\Pi(W)dW \quad (4)$$

where $\Pi(W)$ is the prior probability of W, which indicates the density if W is continuous. Function L described above does not present an analytical form and needs to be found using simulations.

Previous work demonstrates that the Batch workload can achieve good performance levels in terms of scheduling even if we rely on the expected task-processing time according to the computational resources on which it is deployed (see [21–23]). Due to the difficulty of performing the computation needed to solve Eq. (4) in real time, which may be even more complex if a multi-objective function is added to cover multiple workload types (e.g. Service jobs), the Random Forest (RF) supervised machine-learning algorithm is therefore employed to predict the values of a_0 and a_M, L_{Π}.

4. Experimental design

4.1. Methodology

In this work, a standard Design Science methodology [24] is followed, as shown in Fig. 1, which is based on the research question:

May data-centre performance be improved by dynamically switching the resource-management model in heterogeneous and extreme scenarios?

The research question may be formulated as:

$$L(W, a_1) < L(W, a_2), \quad (5)$$

Given a chosen $a_1, a_2 \in A$, a_1 represents the action of switching the resource-management model whilst a_2 denotes the opposite. Such an experiment design enables us to find the form of the optimised function L. Additionally, it enables us to determine the solution of problems (1)–(4) by using the numerical optimisation methods and by providing the tool for the execution of the comparison of function values. In order to answer the research question, an innovative dynamic data-centre resource-management model is proposed, which selects from among a set of existing resource-managing models. The steps outlined below have been followed in order to validate the proposed model:

1. Problem investigation is performed by clearly identifying certain environments where the current resource managers present serious limitations that influence in the overall data-centre performance, and that could be alleviated by equipping the proposed dynamic resource-management model.
2. A formal theoretical framework based on decision theory is constructed to implement the decision-support system.
3. A strong dataset of data-centre execution logs is then built as a knowledge base for the decision-support system.
4. The proposed dynamic data-centre resource-management model is then implemented in an extensively validated large-scale data-centre simulation tool.
5. An extensive set of experiments are then designed and executed to test the proposed model.
6. The significance of the results are validated through a solid statistical framework.
7. The results are analysed and conclusions drawn.

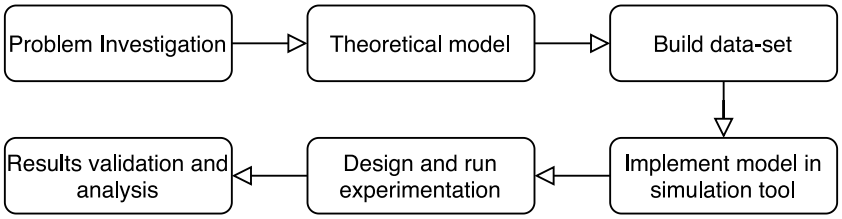


Fig. 1. Design Science process followed in this work.

4.2. Generation of the training dataset

In this work, we created a synthetically-generated set of experiments in order to empirically analyse the KPIs for a vast range of heterogeneous workloads.

Jobs are modelled by the following attributes:

- **Inter-arrival time** $\lambda_{\mathcal{W}}$ represents the average time between two consecutive jobs J_j and J_{j-1} for a given workload. Therefore, it also influences the amount of jobs executed in a specific time window. The inter-arrival time between two *Batch* jobs is usually shorter than that of two *Service* jobs. According to the queue theory, the inter-arrival time usually follows an exponential distribution. In this work, we employ the Exponential distribution to generate the training dataset, whilst more realistic parameterised Weibull distributions [25] with α values of $\alpha = 0.3$, $\alpha = 0.5$, and $\alpha = 0.7$ are employed in order to add realistic unclear patterns and extreme behaviour depending on the value of α (the lower the value, the more extreme the behaviour).
- **Number of tasks** $n_j \sim Exp(\lambda_t)$ represents the number of tasks that comprise a job. The number of tasks of a particular job J_j is generated by following an Exponential distribution with a given mean value of $1/\lambda_t$.
- **Job duration** $d_j \sim Exp(\lambda_d)$ represents the period of time a given job J_j consumes resources in the data centre. The duration of all tasks of a particular job J_j is generated by means of the Exponential distribution with the given expected value $1/\lambda_d$.
- **Resource usage** u_j is the amount of CPU K_{CPU} and RAM K_{RAM} that all the tasks of each particular job in a workload consumes.

Due to the large number of parameters involved in the generation of the workload, we decided to employ a range of values in two key workload-generation parameters to create the workload set:

- **Inter-arrival time** $\lambda_{\mathcal{W}}$: A range of average inter-arrival times of [2000–10] s in steps of 50 s was employed as $\overline{\lambda_{\mathcal{B}}}$ for the Batch workload to simulate various data-centre utilisation rates. The inter-arrival time for Service workload $\overline{\lambda_{\mathcal{S}}}$ is approximately ten times that of the Batch workload.
- **Operation time** T_o: A set of operation simulation times of 6 and 12 h, as well as 1, 3, 5, 7, 10, and 15 days, are executed to create the training set, in order to weigh the impact of the accumulative workload on the performance behaviour of the data centre.

The above set of parameters results in formulating the following theoretical model:

$$W = \{J_j(d_j, u_j), \lambda_W\}_{j=1}^{n} \tag{6}$$

$$\mathcal{T}_j = \{t_{ji}\}_{i=1}^{n_j} \tag{7}$$

All the workloads used in this work follow a day/night pattern. Fig. 2 shows the day/night pattern in terms of resource consumed by the workloads, whose inter-arrival time is generated by means of $\lambda_W \sim$ Weibull $\alpha = 0.7$, $\lambda_W \sim$ Weibull $\alpha = 0.5$, and $\lambda_W \sim$ Weibull $\alpha = 0.3$ distributions, respectively.

The remaining workload parameters are kept constant for the sake of clarity of the results and conclusions. Such workload parameterisation is then used to generate the workload to be submitted to the simulated data centre, composed of 1000 homogeneous machines, each of which equips four batch-scheduler agents and one services-scheduler agent. The performance results of such experimentation constitute the core of the training set with which the decision-support classifier learns.

4.3. Experimentation framework

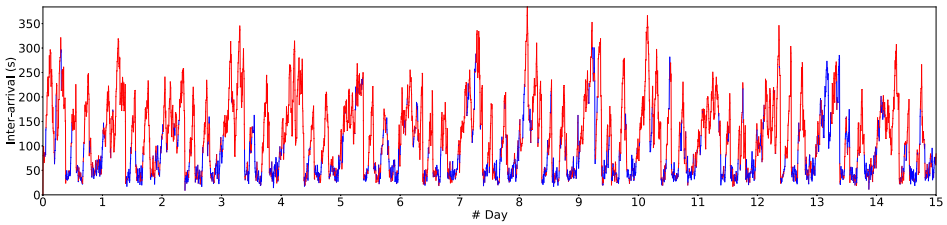
In this work, the previously developed SCORE tool [21] is employed, which has been used in various studies in the literature [1,22,23,25–29], and is able to simulate large-scale infrastructures, often composed of thousands of servers.

In order to evaluate the performance results, a set of indicators for each resource manager is shown. The two main performance-related results deserve special attention:

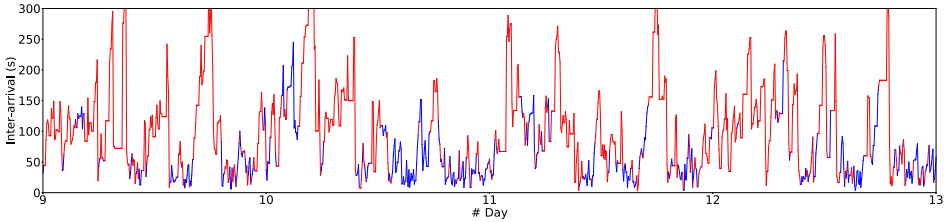
- **Queue times**: How long jobs J_j, composed of n_j tasks, are waiting in the queue until their first task $q_{(1)j}$ and their last task $q_{(n_j)j}$ are deployed.
- **Makespan**: How long it takes from the submission of the job until its completion. The makespan of job J_j may be described as $\mathcal{C}_j = q_{(n_j)j} + d_j$.

For a full comprehension of the behaviour of each resource manager, other key performance indicators in the following tables are presented. The table headings use the following abbreviations:

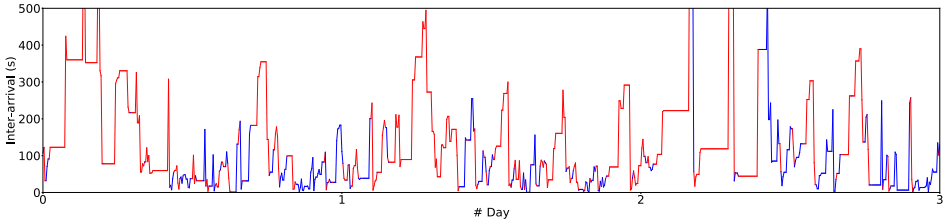
- $\overline{\boldsymbol{\lambda_{\mathcal{B}}}}$ denotes the average inter-arrival mean between Batch jobs. The Batch workload represents almost 90% of scheduled jobs, as our workload is modelled on industry patterns [30,31]. Due to the main impact of the Batch workload in terms of scheduling and for the sake of clarity, only these results are presented in the tables. The results for the Service workload can be found as supplementary material. $\boldsymbol{R_{util}}$ denotes the percentage of data-centre resources utilised on average. $\overline{\boldsymbol{J_{to}}}$ is the number of jobs that could not be executed due to the scheduling process timing out, which means 1000 consecutive tasks or 100 consecutive unsuccessful job scheduling attempts. $\boldsymbol{q_1\mathcal{B}}$ represents the (Batch) job queue time until the first task is scheduled (the lower, the better the user experience), while $\boldsymbol{q_{n\mathcal{B}}}$ denotes the (Batch) job queue time until the last task is scheduled (the lower, the better the user experience). Both the average and the 90 percentile results are shown. $\boldsymbol{\mathcal{C}_{\mathcal{B}}}$ is the (Batch) job makespan (the lower, the better the user experience). Both the average and the 90 percentile results are shown.
- Both the Two-level and Dynamic resource manager model results employ the previous abbreviations plus: $\boldsymbol{R_{lock}}$, which denotes the percentage of resources that are blocked due



(a) Decision evolution for a workload generated by $\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.7$ and job inter-arrival time ($\overline{\lambda_{\mathcal{B}}}$) = 100s for Batch jobs and ($\overline{\lambda_{\mathcal{S}}}$) = 1000s for Service jobs.



(b) Decision evolution for a workload generated by $\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.5$ and job inter-arrival time ($\overline{\lambda_{\mathcal{B}}}$) = 80s for Batch jobs and ($\overline{\lambda_{\mathcal{S}}}$) = 800s for Service jobs.



(c) Decision evolution for a workload generated by $\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.3$ and job inter-arrival time ($\overline{\lambda_{\mathcal{B}}}$) = 100s for Batch jobs and ($\overline{\lambda_{\mathcal{S}}}$) = 1000s for Service jobs.

Fig. 2. Evolution of the utilisation of the two possible resource-managing strategies of the dynamic resource-managing model according to the inter-arrival job time ($\lambda_{\mathcal{W}}$) of the recent past. Red segments represent the periods where the two-level strategy (Mesos) is employed, while blue segments represent those periods where the shared-state (Omega) approach is employed. It can be seen that the shared-state strategy (Omega) is usually preferred when the inter-arrival time ($\overline{\lambda_{\mathcal{B}}}$) is shorter than 150 s. However, it becomes evident that the unpredictability of the generated workload inherent to the Weibull distribution behaviour blurs the decision pattern. This can be stated if the decision pattern shown in 2(a) is compared to those of 2(b) and 2(c), which progressively present a less clear pattern.

to the pessimistic-blocking strategy of the Two-level resource managers; $\boldsymbol{S_{ret}}$ refers to the number of scheduling operations retried due to lack of resources in the copy of the cluster state (the lower, the better to avoid scheduling bottlenecks).

- Both the Shared-state and Dynamic resource manager results employ the previous abbreviations, plus: $\boldsymbol{J_{conf}}$ $\boldsymbol{T_{conf}}$, which represent the total number of job conflicts and of tasks, respectively (the lower, the better for the overall performance).

All the experiment configurations are simulated for 15 days of operation time with a data-centre composed of 1000 machines. Each experiment configuration is run 20 times. Tables 4, 5, and 6 show the average results. The following statistical tests were performed in SPSS in order to check the statistical significance at $\alpha = 0.05$ of the results based on the queue times:

1. Iglewicz and Hoaglin's robust test for multiple outliers (two-sided test) with $Z = 3.5$.
2. One-way ANOVA Levene test to check the non-homogeneity of variances with a significance level of $\alpha = 0.05$.
3. Multiple post-hoc tests, such as the Tamhane, Dunnett T3, Games-Howell, and Dunnett C tests, to analyse the difference between pairs of resource schedulers with a significance level of $\alpha = 0.05$, thereby proving the statistical significance of the results presented.

The results of the SPSS statistical tests confirmed their statistical significance and they are provided as supplementary material.

4.4. Classification algorithms and feature selection

The proposal for the dynamic switch of resource-managing models in accordance with the data-centre workload and status relies on a supervised classifier, which, whenever a new job arrives to the data centre, decides whether it is better to switch to another scheduler or not.

To this end, a training dataset has been built from the job attributes explained in Section 4.2 by evaluating the total workload executed as a whole, and the data has also been disaggregated by workload (Batch and Service). This results in a dataset composed of 11 columns, in addition to the class column corresponding to the scheduler that achieved the best performance for those workloads according to the aspects explained in Section 4.3.

It should be borne in mind that, in order to avoid training bias, the inter-arrival time of all workloads used for the training dataset is generated by means of the Exponential distribution, whilst the inter-arrival time of the workloads generated for the evaluation of the proposal follow extreme-value Weibull distributions. The raw dataset can be found as supplementary material.

Once the training dataset is built, the next step in our proposal is the execution of a feature-selection algorithm in order to retain only those features of the dataset that are relevant to our problem. In this case, we have opted for the use of the implementation

Table 1
Accuracy of classifiers, using a 10-fold cross validation with the refined 4-column dataset.

Naive Bayes	Linear SVC	XGBoost	Random Forest
0.45	0.74	0.73	0.76

Table 2
Mean, standard deviation, mode and median values of the results obtained from the execution of the Random-Forest-based classification algorithm ∼500 times.

μ (ms.)	σ (ms.)	*Mo* (ms.)	$\tilde{\mu}$ (ms.)
8.60	2.68	6.52	7.73

Table 3
Time complexity of the training and classification phases for each classification algorithm. n is the number of training samples, c is the number of classes (in our case, $c = 2$), d is the number of features considered (in our case, $d = 4$), k is the number of vectors, and t is the number of trees ($t = 100$ by default in the implementations tested).

Algorithm	Training complexity	Runtime complexity
Naive Bayes	$O(n \cdot d)$	$O(c \cdot d)$
Linear SVC	$O(n^2)$	$O(k \cdot d)$
Random Forests	$O(t \cdot d \cdot n \cdot log(n))$	$O(d \cdot t)$
XGBoost	$O(t \cdot d \cdot n \cdot log(n))$	$O(d \cdot t)$

of the CFS algorithm [18] included in Weka [32], since it is a well-known and widely-used workbench for machine learning. The output of this algorithm is a subset of features formed by the resource utilisation, $\overline{R_{util}}$, inter-arrival (Batch), $\overline{\lambda_{\mathcal{B}}}$, the number of Service tasks, $n_j \sim Exp(\lambda_t)$, and the memory task (Service), K_{RAM}.

This refined dataset with four columns (five including the class) has been passed as input to four different classification algorithms: Naive Bayes [11], LinearSVC as an SVM [13] implementation, Random Forest [14], and XGBoost [15]. These are recent implementations of a number of the most representative algorithms in supervised statistical machine-learning classification (Bayesian networks, support vector machines, and decision trees).

In our preliminary experimentation (a 10-fold cross validation with the same training dataset and the default values for all the parameters of the methods), the algorithm that achieved the best accuracy results was Random Forest, as can be observed in Table 1, which is the method integrated in our proposal for the dynamic selection of the scheduler. However, the design of our experimentation framework enables various classification models to be easily evaluated in future works.

The performance of the selected classification algorithm was measured on the machine employed to run the simulation experiments (Macbook Pro 13", 2,3 GHz Intel Core i5, 16 GB 2133 MHz LPDDR3 and 256GB of SSD HD) in order to evaluate its suitability for real-time scenarios. We executed approximately 500 classification runs. The results are shown in Table 2 and the raw data can be found as supplementary material.

Regarding the cost effectiveness of these algorithms on the overall performance of the data centre, it is worth noting that the training phase (usually the most time-consuming task) is executed off-line, just once, in order to generate each classification model beforehand. In this way, the only addition to the normal behaviour of a resource manager is the classification of each job arriving to the data centre, which is a very economical task in terms of time. However, in Table 3, the time complexity of training and classification for each of the methods discussed in this section is shown.

We can see that the complexity of the most critical task in our case, the run-time classification of new jobs, in all the cases is $O(1)$, and it should therefore inflict no significant harm on the performance of the resource manager.

5. Evaluation

This section discusses the behaviour of the dynamic, two-level and shared-state resource managers when executing three realistic workloads, each of which has a higher level of extreme peaks and unpredictable patterns, to check the utility of the proposed model. Such extreme behaviour of the arrival of jobs can easily be seen in Fig. 2, where the dynamic behaviour of the Dynamic resource manager is depicted according to the arrival of jobs. Red periods represent the time where the two-level model is employed, and blue periods represent the time where the shared-state model is used. It should borne in mind that a relatively stable day/night pattern can be seen for $\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.7$ in Fig. 2(a). This clear pattern gradually fades away, and ends up in the unclear pattern and frequent job-arrival peaks for $\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.3$ (very extreme) in Fig. 2(c). An intermediate state may be found for Weibull $\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.5$ (extreme) in Fig. 2(b).

A second insight shown in Fig. 2 involves the coherency in the decision-making process of the decision-support model of the Dynamic resource manager proposed, which leads to a predictable pattern of utilisation according to the arrival of jobs, even for the extreme workloads generated by means of the Weibull distribution with $\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.3$.

Tables 4, 5, and 6 present the performance results for three resource mangers: our proposal (DISCERNER); two-level (Mesos); and shared-state (Omega). Each table also presents three workloads: $\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.3$ (very extreme); $\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.5$ (extreme); and $\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.7$ (slightly extreme), together with three utilisation rates, represented by the average job inter-arrival time: $\overline{\lambda_{\mathcal{B}}} =\sim$100 s (lesser utilisation); $\overline{\lambda_{\mathcal{B}}} =\sim$85 s (medium utilisation); and $\overline{\lambda_{\mathcal{B}}} =\sim$70 s (high utilisation), in order to evaluate the impact in terms of performance of both the job arrival pattern and the number of executed jobs.

A first glance at the data-centre utilisation rate ($\overline{R_{util}}$) unveils the main pattern present in these results: DISCERNER, the Dynamic resource manager, will match the results of the other resource-managing models (or even impose a low deterioration due to classification and forecast errors) in low-pressure scenarios, improving (especially if compared to the two-level resource-managing model) gradually as the workload pattern becomes more extreme and the arrival rate increases. That is exactly what ($\overline{R_{util}}$) results show: the three resource-managing models provide the same utilisation results for slightly extreme ($\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.7$) patterns and low arrival rates ($\overline{\lambda_{\mathcal{B}}} =\sim$100 s), until a very extreme pattern arrives ($\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.3$), where the Dynamic model (Table 4) significantly outperforms (∼10%) the two-level results (Table 5), thereby preventing the drop-out of jobs (J_{to}) experienced by the two-level model. It should also be borne in mind that the same improvement is achieved for extreme workloads ($\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.5$) when the arrival rate is high ($\overline{\lambda_{\mathcal{B}}} =\sim$70 s). A direct correlation with the rate of resource that is locked due to the pessimistic approach of the two-level model ($\overline{R_{lock}}$) becomes clear: the greater the difference between the dynamic and two-level models in terms of $\overline{R_{lock}}$, the higher the difference in terms of resource utilisation. In this particular indicator, the Dynamic resource-managing model achieves similar results compared to those provided by the shared-state model (Table 6).

In terms of the key performance indicators, $q_{1\mathcal{B}}$, $q_{n\mathcal{B}}$, and $\mathcal{C}_{\mathcal{B}}$, all three follow the aforementioned pattern. Hence, a detailed explanation is now given of the results of $\overline{q_{1\mathcal{B}}}$ as representative for all these indicators in order to avoid redundant explanations. On one hand, $\overline{q_1}$ shows a lesser improvement (∼20%, 27.77 s vs. 35.84 s) and an even lesser deterioration (∼10%, 27.77 s vs. 24.49 s) for low-arrival-rate ($\overline{\lambda_{\mathcal{B}}} =\sim$ 100 s) and slightly-extreme

Table 4
Performance results of DISCERNER (the Dynamic resource-managing model) for Batch Workloads.

$\overline{\lambda_{\mathcal{B}}}$ (s)	$\overline{R_{util}}$ (%)	J_{to}	S_{ret}	R_{lock} (%)	J_{conf}	T_{conf}	$\overline{q_{1\mathcal{B}}}$ (s)	$q_{1-90\%\mathcal{B}}$ (s)	$\overline{q_{n\mathcal{B}}}$ (s)	$q_{n90\%\mathcal{B}}$ (s)	$\overline{\mathcal{C}_{\mathcal{B}}}$ (s)	$\mathcal{C}_{\exists r\%\mathcal{B}}$ (s)
$\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.3$												
109.43	28.51	0	393	10.87	1081	41821	97.56	267.53	173.09	448.89	250.76	579.14
89.64	29.19	0	971	12.60	1426	52586	117.30	312.63	229.50	574.35	305.78	724.09
70.74	29.84	0	2297	16.15	1914	69623	164.25	425.59	396.90	971.80	450.13	1095
$\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.5$												
105.23	28.65	0	223	12.5	961	38844	42.00	134.29	64.91	187.48	165.04	382.27
87.23	29.24	0	319	13.79	1335	53114	53.89	164.20	90.33	249.20	188.78	434.92
68.88	30.15	0	693	15.64	1966	74234	74.77	220.11	143.26	387.00	236.61	566.73
$\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.7$												
102.59	28.72	0	190	12.91	878	35241	27.77	95.14	42.01	128.24	150.14	347.76
85.16	29.32	0	286	13.92	1314	50983	37.86	124.02	65.31	184.51	171.59	396.67
68.15	30.24	0	534	15.59	1979	77125	59.22	184.48	114.91	315.13	216.96	504.49

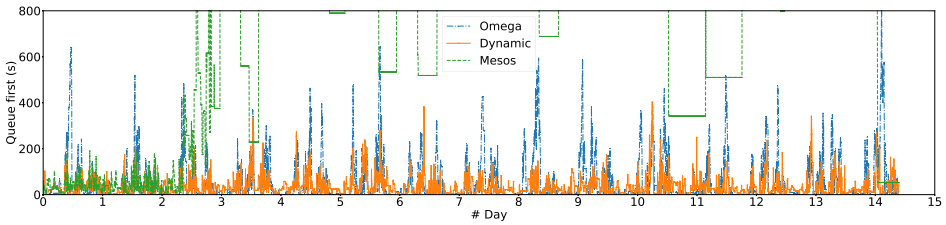
Table 5
Performance results of the two-level resource-managing model for Batch Workloads.

$\overline{\lambda_{\mathcal{B}}}$ (s)	$\overline{R_{util}}$ (%)	J_{to}	S_{ret}	R_{lock} (%)	$\overline{q_{1\mathcal{B}}}$ (s)	$q_{1-90\%\mathcal{B}}$ (s)	$\overline{q_{n\mathcal{B}}}$ (s)	$q_{n90\%\mathcal{B}}$ (s)	$\overline{\mathcal{C}_{\mathcal{B}}}$ (s)	$\mathcal{C}_{\exists r\%\mathcal{B}}$ (s)
$\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.3$										
110.96	26.78	2	6814	52.63	7960	27799	79275	362614	638481	1296000
88.09	26.12	486	96521	68.64	31176	102823	232980	695903	1129181	1296000
71.52	26.03	0	1729	71.13	35875	131479	264077	831729	1162026	1296000
$\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.5$										
105.30	28.63	0	57	19.64	55.03	162.98	60.77	179.39	146.99	329.38
86.91	29.18	0	184	23.27	62.21	180.24	71.55	206.22	156.14	349.27
68.53	26.67	0	2443	64.41	19108	58021	108044	386205	754762	1296000
$\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.7$										
103.14	28.67	0	18	20.88	35.84	111.64	38.51	120.55	135.53	305.45
85.46	29.25	0	43	24.04	41.98	128.24	46.18	140.85	140.15	315.93
68.14	30.14	0	186	28.17	50.15	149.40	58.47	174.01	150.65	337.96

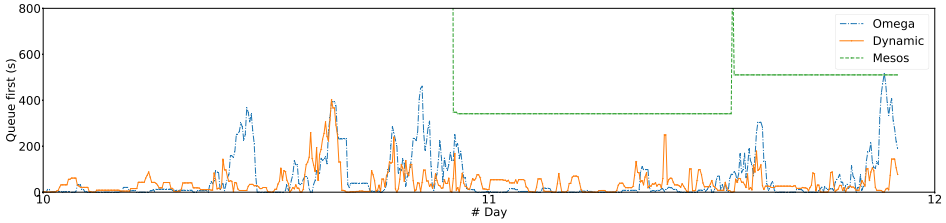
Table 6
Performance results of the shared-state resource-managing model for Batch Workloads.

$\overline{\lambda_{\mathcal{B}}}$ (s)	$\overline{R_{util}}$ (%)	J_{to}	J_{conf}	T_{conf}	$\overline{q_{1\mathcal{B}}}$ (s)	$q_{1-90\%\mathcal{B}}$ (s)	$\overline{q_{n\mathcal{B}}}$ (s)	$q_{n90\%\mathcal{B}}$ (s)	$\overline{\mathcal{C}_{\mathcal{B}}}$ (s)	$\mathcal{C}_{\exists r\%\mathcal{B}}$ (s)
$\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.3$										
110.36	28.56	0	2551	121274	115.76	358.11	275.75	723.60	322.22	787.15
91.51	28.99	0	3229	154765	177.18	547.43	413.16	1072.18	405.13	1014.22
73.06	29.84	0	4423	210212	305.70	917.42	722.48	1800.01	595.39	1518.74
$\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.5$										
105.25	28.61	0	2034	92566	36.75	124.52	84.90	236.77	191.93	446.21
87.11	29.22	0	2733	125856	69.14	235.85	161.06	426.48	242.69	558.19
69.09	30.09	0	4083	186690	167.24	546.20	395.18	1019.65	398.18	929.41
$\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.7$										
102.74	28.69	0	1729	79219	24.49	82.95	54.68	152.53	164.21	373.18
85.67	29.31	0	2523	113377	54.87	189.44	123.25	333.56	215.42	473.66
67.78	30.24	0	3885	175264	145.56	477.91	336.85	884.21	356.95	812.66

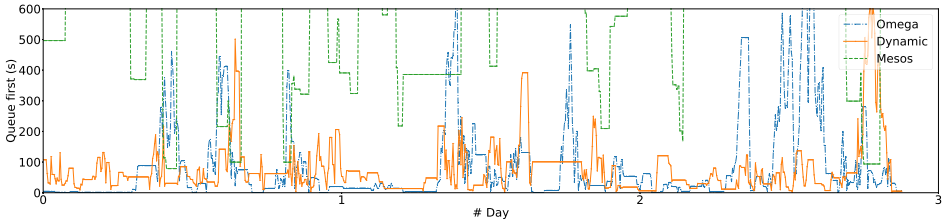
workloads ($\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.7$) compared to the two-level and shared-state models, respectively. On the other hand, once the arrival pattern becomes more extreme ($\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.5$) and the amount of processed workload increases ($\overline{\lambda_{\mathcal{B}}} =\sim$ 85 s) then the benefits of the application of the Dynamic model become clearer, and $\overline{q_{1\mathcal{B}}}$ is reduced by ∼20% (53.89 s vs. 62.21 s) and 30% (53.89 s vs. 69.14 s) compared to the two-level and shared-state models, respectively. This pattern is also confirmed in Fig. 3, where it is shown that, the lower the arrival rate and the more extreme the workload, the better the results for the Dynamic model. Furthermore, an even better performance behaviour is shown in this figure: the dynamic model is the only resource-managing model that remains stable even for very extreme workloads and low arrival rates (high-pressure workloads), especially when compared with the two-level model, which faces a performance bottleneck due to its pessimistic blocking approach. This behaviour is confirmed by the results presented in Tables 4, 5, and 6 for very extreme workloads ($\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.7$) and low arrival rates ($\overline{\lambda_{\mathcal{B}}} =\sim$ 70 s), where the two-level model is unable to process the incoming workload, as shown in Fig. 3, and reduces the queue time by approximately half as compared to the shared-state model (164.25 s vs. 305.70 s).



(a) Decision evolution for a 15-day period and a workload generated by $\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.5$ and job inter-arrival time $(\overline{\lambda_{\mathcal{B}}}) = 80$s for Batch jobs and $(\overline{\lambda_{\mathcal{S}}}) = 800$s for Service jobs.



(b) Detail of decision evolution for a 2-day period and a workload generated by $\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.5$ and job inter-arrival time $(\overline{\lambda_{\mathcal{B}}}) = 80$s for Batch jobs and $(\overline{\lambda_{\mathcal{S}}}) = 800$s for Service jobs.



(c) Decision evolution for 4-day period and a workload generated by $\lambda_{\mathcal{W}} \sim$ Weibull $\alpha = 0.3$ and job inter-arrival time $(\overline{\lambda_{\mathcal{B}}}) = 100$s. for Batch jobs and $(\overline{\lambda_{\mathcal{S}}})$ 1000s. for Service jobs.

Fig. 3. Comparison of the evolution of job queue time until their first task is scheduled ($q_{1\mathcal{B}}$) over time between shared-state, two-level, and the proposed dynamic model. In Fig. 3(a), it is shown that the two-level approach (Mesos) hits its scheduling-processing limit on day 3 due to extremity and a low inter-arrival period. If we compare the aforementioned queue times with those shown in Fig. 3(c), it becomes clear that the number of incoming jobs is more decisive in achieving good performance results than the extremity of the arrival pattern. In contrast, both Omega (shared-state) and the proposed Dynamic model are able to continue serving incoming jobs. However, it should be borne in mind that the proposed Dynamic resource-managing model shows a more stable scheduling behaviour when facing the majority of peak loads as repeatedly shown in Fig. 3(a), and more in detail, as can be seen between days 10 and 11, and at the end of day 11 in Fig. 3(b).

5.1. Summary of results

From the analysis of the results presented herein, the advantages and limitations of the proposed resource manager alternative, DISCERNER can be summarised. The main advantages include:

- DISCERNER improves performance of the most popular resource managers, especially in that it decreases queue times and maintains a more homogeneous behaviour by reducing peaks in waiting times, which are present in the other alternatives (Mesos and Omega).
- The percentage of resource utilisation ($\overline{\boldsymbol{R_{util}}}$) for DISCERNER is similar to those of Mesos and Omega: DISCERNER even presents a slight improvement therein.

Main limitations of the presented alternative includes:

- In the case of stable, well-known, and non-changing workloads (i.e., workloads that do not follow patterns such as day/night patterns), DISCERNER may not be very useful, as fine-tuning strategies could provide better results. However, current data centres are executing workloads that are more and more heterogeneous, and DISCERNER could prove highly useful in these cases.
- As technical limitations, the implementation of a dynamic resource manager could be a challenge in the real world. It is not trivial to include this kind of resource managers in a production data centre, and it is as even more complex challenge if the dynamic selection of resource managers include resource managers from different strategies (e.g., from centralised to distributed). The complexity of this implementation underlies the fact that certain decisions have to be made in relation to:
  - the moment of exchange of the resource manager. There could be several alternatives that include: a daemon that queries whether the swap has to be made periodically (e.g., every 5 min of runtime, which is the alternative selected in the experiments run in this work) or whether this is performed only when certain events arise (e.g., when a new job arrives).
  - the queue management. The queues where new jobs are waiting to be scheduled could be shared by resource managers, and therefore a hot-swap could be made. Another alternative would be to have independent queues for each type of resource manager, so that when a switch is ordered, new jobs arrive in the new queue, and the swap is not effective until the queue of last resource manager used becomes empty.

- The performance of the classification methods, which is crucial in our proposal, should be improved, mainly because once they are trained they cannot adapt well with new workload patterns. The fact that DISCERNER outperforms other resource managers, in spite of the accuracy of the classifiers, encourages us to further our work in this direction, and to include schemes of reinforcement learning, for example, in order to enable DISCERNER to adapt to highly varying situations.

6. Conclusions and future work

In this work we proposed a new decision-support model for data-centre resource managing, which, by means of a machine-learning classifier, dynamically switches between existing resource-managing models according to the data-centre operational situation and incoming workload.

The presented model is based on a theoretical decision theory framework supported by the experiments to find the form of the optimised function. Such an approach is applied due to the fact that the considered function does not have an analytical form in Cloud Computing scenarios, where numerous heterogeneous workloads are executed on the same hardware, which leads to higher utilisation rates and unpredictable and extreme patterns. The selected theoretical framework was chosen due to its simplicity over other alternatives, such as strategic game-based solutions, usage of rationality principles, and advanced artificial intelligence methods. The considered problem needs to be solved on demand for a highly dynamic Cloud environment without unnecessary delays. In order to accelerate the time taken to find solutions, the system may run dedicated simulations in advance.

It has been shown that this new model presents a great opportunity in terms of performance. In addition, since new resource-managing models could be included, it provides a flexible model that could easily adapt to the ever-evolving environment of Cloud Computing.

On one hand, in the worst scenarios with moderate workloads and predictable patterns, this new model achieves minor performance gains, and even may cause a low negative impact depending on the scenario. This is mainly due to the errors inherent to the classification and forecasting models employed, which can be reduced or even avoided if classification and forecasting models of a more complex nature are considered. On the other hand, major performance gains in the range of [20%–50%] in terms of queue times and makespan can be achieved in more realistic and extreme scenarios.

As future work, we intend to improve this model through several actions:

- Extension of the proposed model for the dynamic utilisation of various resource managers in parallel, as well as dynamic combinations thereof to serve different groups of workloads depending on their characteristics and requirements.
- Modification of the simulation tool to provide real-time job performance results instead of temporal aggregations.
- Improvement of the classification and forecasting models by employing artificial neural networks.
- Performance of a deep comparative analysis against other resource-managing models seldom employed in industry, such as parallel and hybrid models.
- Validation of the results by modifying existing data-centre resource managers and applying them to real data centres in production.

CRediT authorship contribution statement

Damián Fernández-Cerero: Conceptualization, Methodology, Investigation, Formal analysis, Data curation, Software, Writing - original draft, Resources, Validation, Writing - review & editing. **F. Javier Ortega:** Project administration, Investigation, Data curation, Writing - original draft, Writing - review & editing, Funding acquisition. **Agnieszka Jakóbik:** Conceptualization, Investigation, Formal analysis, Writing - original draft. **Alejandro Fernández-Montes:** Project administration, Writing - original draft, Writing - review & editing, Validation, Supervision, Funding acquisition.

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.

Acknowledgement

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

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.future.2020.10.031.

References

[1] 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, 2018, pp. 210–219.

[2] 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, 2020, pp. 1–4.

[3] I. Kilanioti, A. Fernández-Montes, D. Fernández-Cerero, A. Karageorgos, C. Mettouris, V. Nejkovic, N. Albanis, R. Bashroush, G.A. Papadopoulos, Towards efficient and scalable data-intensive content delivery: State-of-the-art, issues and challenges, in: High-Performance Modelling and Simulation for Big Data Applications, Springer, 2019, pp. 88–137.

[4] B. Burns, J. Beda, K. Hightower, Kubernetes, Dpunkt, 2018.

[5] J. Turnbull, The Docker Book: Containerization is the New Virtualization, James Turnbull, 2014.

[6] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A.D. Joseph, R.H. Katz, S. Shenker, I. Stoica, Mesos: A platform for fine-grained resource sharing in the data center, in: NSDI, Vol. 11, 2011, p. 22.

[7] V.K. Vavilapalli, A.C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, et al., Apache hadoop yarn: Yet another resource negotiator, in: Proceedings of the 4th Annual Symposium on Cloud Computing, ACM, 2013, p. 5.

[8] A. Narayanan, Tupperware: Containerized Deployment at Facebook, DockerCon, 2014.

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

[10] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, J. Wilkes, Large-scale cluster management at google with borg, in: Proceedings of the Tenth European Conference on Computer Systems, ACM, 2015, p. 18.

[11] M.E. Maron, Automatic indexing: an experimental inquiry, J. ACM 8 (3) (1961) 404–417.

[12] L.E. Baum, T. Petrie, Statistical inference for probabilistic functions of finite state Markov chains, Ann Math. Statist. 37 (6) (1966) 1554–1563.

[13] C. Cortes, V. Vapnik, Support-vector networks, in: Machine Learning, 1995, pp. 273–297.

[14] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32, http://dx.doi.org/10.1023/A:1010933404324.

[15] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794.
[16] A survey on feature selection methods, Comput. Electr. Eng. 40 (1) (2014) 16–28.
[17] A. Jović, K. Brkić, N. Bogunović, A review of feature selection methods with applications, in: 38th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO, 2015, pp. 1200–1205.
[18] M.A. Hall, Correlation-based Feature Subset Selection for Machine Learning, University of Waikato, Hamilton, New Zealand, 1998.
[19] D. Fernández-Cerero, A. Fernández-Montes, A. Jakóbik, Limiting global warming by improving data-centre software, IEEE Access 8 (2020) 44048–44062.
[20] G. Parmigiani, L. Inoue, Decision Theory: Principles and Approaches, Vol. 812, John Wiley & Sons, 2009.
[21] D. Fernández-Cerero, A. Fernández-Montes, A. Jakóbik, J. Kołodziej, M. Toro, SCORE: Simulator for cloud optimization of resources and energy consumption, Simul. Model. Pract. Theory 82 (2018) 160–173.
[22] D. Fernández-Cerero, A. Fernández-Montes, A. Jakobik, J. Kolodziej, Stackelberg game-based models in energy-aware cloud scheduling, in: ECMS, 2018, pp. 460–467.
[23] D. Fernández-Cerero, A. Jakobik, A. Fernández-Montes, J. Kołodziej, GAME-SCORE: Game-based energy-aware cloud scheduler and simulator for computational clouds, Simul. Model. Pract. Theory 93 (2019) 3–20.
[24] R.J. Wieringa, Design Science Methodology for Information Systems and Software Engineering, Springer, 2014.
[25] D. Fernández-Cerero, F.J. Ortega-Irizo, A. Fernández-Montes, F. Velasco-Morente, Bullfighting extreme scenarios in efficient hyper-scale cluster computing, Cluster Comput. (2020) 1–17.
[26] D. Fernández-Cerero, A. Fernández-Montes, J.A. Ortega, Energy policies for data-center monolithic schedulers, Expert Syst. Appl. 110 (2018) 170–181.
[27] D. Fernández-Cerero, A. Fernández-Montes, F. Velasco, Productive efficiency of energy-aware data centers, Energies 11 (8) (2018) 2053.
[28] D. Fernández-Cerero, A. Fernández-Montes, F.J. Ortega, A. Jakóbik, A. Widlak, Sphere: Simulator of edge infrastructures for the optimization of performance and resources energy consumption, Simul. Model. Pract. Theory 101 (2020) 101966.
[29] D. Fernández-Cerero, A. Fernández-Montes, A. Jakóbik, Limiting global warming by improving data-centre software, IEEE Access 8 (2020) 44048–44062.
[30] 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.
[31] C. Lu, K. Ye, G. Xu, C. Xu, T. Bai, Imbalance in the cloud: An analysis on Alibaba cluster trace, in: 2017 IEEE International Conference on Big Data, Big Data, 2017, pp. 2884–2892.
[32] M.A.H. Eibe Frank, I.H. Witten, The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, 2016.

Damián Fernández-Cerero received the B.E. degree and the M.Tech. degrees in Computer Science from the University of Sevilla. In 2014, he joined the Department of Computer Languages and Systems, University of Seville, as a PhD. Student. During his Ph.D. period he established European collaborations in France, Poland, Italy, and Ireland through several research stays. After the obtention of his Ph.D, he was granted with a Marie Curie post-doctoral position at Dublin City University. Currently he both lectures and conducts research at University of Sevilla. He has worked on several research projects supported by the Spanish government and the European Union. His research interests include performance and energy optimisation models for cloud and edge-computing systems.

F. Javier Ortega is a Ph.D. in Computer Science from the University of Seville. His research is focused on machine learning and graph-based algorithms and its application to Social Network analysis and Natural Language Processing. His most recent works are related to opinion mining, social network analysis, web spam detection and trust and reputation analysis.

Agnieszka Jakóbik (Krok) received her M.Sc. in the field of Stochastic Processes at the Jagiellonian University, Poland and a Ph.D. degree in Artificial Neural Networks at the Tadeusz Kosciuszko Cracow University of Technology, Poland. Since 2009 she is an Assistant Professor at the Tadeusz Kosciuszko Cracow University of Technology. Her e- mail address is: ajakobik@pk.edu.pl.

Alejandro Fernández-Montes received the B.E. degree in computer science and the M.Tech. and international Ph.D. degrees in software engineering from the University of Seville, Spain. In 2006, he joined the Department of Computer Languages and Systems, University of Seville, as an Intern, and he became a Lecturer and an Assistant Professor, in 2013. In 2018, he became an Associate Professor (Professor Titular de Universidad). In 2008 and 2009, he was invited to work with the École Normale Supérieure de Lyon (ENS de Lyon) in saving energy solutions for Grid'5000 infrastructure. In 2012, he was invited to work at the Universitat Politécnica de Barcelona to share experiences in saving energy and doctoral methodologies. In 2016, he was invited to work with Shanghai Jiao Tong University, China, to initiate collaborations in distributed computing. He currently teaches and conducts research with the University of Seville, where he is also the Principal Investigator of several projects. His research interests include energy efficiency in distributed computing, applying prediction models to balance load, and applying on–off policies to the Internet data centres.