Noname manuscript No.
(will be inserted by the editor)

A Clustering-based Knowledge Discovery Process for Data Center Infrastructure Management

Diego García-Saiz · Marta Zorrilla · José Luis Bosque

Received: date / Accepted: date

Abstract Data Center Infrastructure Management (DCIM) is the integration of information technology and facility management disciplines to centralise monitoring and management in data centers. One of the most important problems of DCIM tools is the analysis of the huge amount of data obtained from the real time monitoring of thousands of resources. In this paper, an adaptation of the Knowledge Discovery Process (KDD) for dealing with the data analysis in DCIM tools is proposed. A case of study based on monitoring and labelling of nodes of a High Performance Computing (HPC) data center in real time is presented. This shows that characterising the state of the nodes according to a reduced and relevant set of metrics is feasible and its outcome directly usable, simplifying consequently the decision-making process in these complex infrastructures.

Keywords Data Mining · Data Centers · DCIM · Monitoring

1 Introduction

In recent decades, the process of delocalisation and internationalization of large companies, coupled with the explosion in the use of information technology (IT) and data processing, have caused that the computing needs of these organizations have grown dramatically. This development has led to the creation of very large *data centers* [1]. Data centers have lots of resources, often heterogeneous, to be administered and managed. The main objective of the data centers administrators is to achieve the maximum performance, with the lowest cost and energy consumption [14,5]. In this sense, *Data Center Infrastructure Management (DCIM)* is the integration of information technology

Diego García-Saiz · Marta Zorrilla · José Luis Bosque
Department of Ingeniería Informátimca y Electrónica
University de Cantabria
E-mail: diego.garcia@unican.es,marta.zorrila@unican.es,joseluis.bosque@unican.es

and facility management disciplines to centralise monitoring, management and intelligent capacity planning of critical systems in a data center [10].

Monitoring is comprised of two main stages: *data capture* and *analysis and interpretation*. While the capture can be carried out automatically, in fact, there are many tools to do it [11], the analysis phase is a much more complex process that requires data science techniques to deal with it. It is mainly due to the huge quantity of data that must be analysed in a very short time so as to make well informed decisions. Currently, this decision-making is performed with no nearly real-time information, and hence, the analysis of the situation might not be the most accurate.

Additionally, data mining techniques have proven to be very useful in order to uncover the hidden and relevant information from data which may help in the decision-making process of any kind of business [9,12,17]. However, this is only one of the steps in the more general process of Knowledge Discovery in Databases (KDD).

This paper proposes an adaptation of the KDD process to enrich the DCIM tools with the aim of helping in the analysis and making the interpretation of the huge amount of data from the monitoring of a data center easier. In such a way, the paper defines a series of steps that a data center manager should carry out to apply a KDD process for the extraction of useful and valuable knowledge addressed to the management and administration of data centers.

Clustering techniques are generally the first ones to be applied on a dataset since these extract information about the regularity of the data. Furthermore, they enable end-user to perform a multidimensional analysis of the dataset from a reduced but relevant subset of features selected for the purpose in hand. Thus, administrators can gain insight of the current situation of the nodes with a much higher level of abstraction, which makes their decision making easier. The case of study presented confirms that the use of our approach helps in the decision-making process.

The rest of the paper is structured as follows. Section 2 briefly explains the KDD process. Then, section 3 describes a KDD process adapted to a data center and a possible use case. The empirical evaluation and discussion of the experimental results are presented in section 4. Finally, some conclusions and future work are drawn in the last section.

2 Knowledge Discovery in Databases: a brief overview

Data mining is the process of applying data analysis and discovery algorithms to find knowledge patterns over a collection of data [4]. Importantly, data mining is only a step of an overall process named Knowledge Discovery in Databases. KDD consists of the application of data mining algorithms on data sources previously processed and prepared to meet the user's requirements. The overall process involves the repeated application of the following steps:

– Understanding the application domain and the goals of the end-user and translating this into a data mining problem.

- Creating a target dataset, i.e., selecting the set of variables suitable for the discovery process. This step is very important because it allows data miner to describe the dataset and assess its quality, and consequently, carry out preprocessing tasks if needed.
- Choosing the data mining task, this means, deciding whether the goal of the KDD process is descriptive or predictive, and based on this decision, choosing the data mining algorithm(s) and the most appropriate parameters for its execution.
- Executing the algorithm on the dataset.
- Interpreting mined patterns and assessing their quality and validity.
- Consolidating discovered knowledge.

As a consequence of the importance that these techniques are acquiring, the industrial sector developed a standard (currently de facto), named CRISP-DM [2], which gathers these steps and describes the tasks to be performed.

In general, data mining tasks are classified into two main categories: descriptive and predictive [9]. Descriptive mining tasks characterise the general properties of the data, whereas predictive ones perform inference on current data in order to make predictions. For instance, finding out whether a customer will contract a bank service given his demographic characteristics [12] is a predictive problem, whereas discovering the students' profile enrolled in an e-learning course [17] is a descriptive problem.

Inside each category, there are many algorithms following different approaches [16]. Therefore, choosing the correct technique for the target is highly important. Even more, it is essential that the outcome provides end users with meaningful insights from the original data, thus the output pattern must fulfil with quality criteria and a high degree of actionability [7].

3 Particularising a KDD process: a process model for clustering

One of the main challenges in the KDD process is to adapt the steps to be performed to a specific domain. This is why, next, we enumerate and briefly describe the indispensable tasks that should be carried out for a descriptive problem as the proposed one, that is to identify the current state of a set of nodes from a set of monitored parameters and describe these in a simpler, understandable and actionable way. From all the existing techniques, we will choose clustering because this is suitable for this purpose.

Clustering algorithms segment data into a certain number of clusters (groups, subsets, or categories) such that the data points in a cluster are more similar to each other than points in different clusters. Thus, their goal is to find a reduced number of prototypes as representatives of the set. The steps to apply clustering in DCIMs could be summarised in:

1. **Monitoring**: The first step is to monitor a number of independent metrics on all nodes of the data center. These metrics must be defined according to the needs of each particular case of study. This phase produces a huge

amount of data that is complex to analyse, but suitably managed, this allows decision makers to discover useful knowledge for the domain.

2. **Statistical Analysis**: A series of statistical analyses of the monitored variables must be performed. This step aims to assess the quality of the dataset to be processed. Among other tasks, the following ones should be done: 1) a correlation study between variables with the aim of reducing the dimensionality of the set; 2) an outlier detection process to check that all variables are in the range of permitted values; 3) an analysis of data distribution, i.e. there are values in all the allowed range; 4) and, finally, if there were null values in any of the variables, an assessment of its usefulness and convenience for the KDD goal.
3. **Clustering**: Once the analysis has been performed and the dataset built, the following step is to select a clustering technique. There are different kinds of algorithms according to how the clusters are built. A well-established classification is: partitional clustering, hierarchical clustering, density-based clustering and grid-based clustering. Algorithms belonging to the first class are the most frequently used due to their low computational complexity and be easily understandable and reasonably scalable. Inside this group, *kmeans* algorithm [9] is one of the most frequently used [15]. This algorithm is iterative and starts 1) selecting K cluster centers (or centroids) randomly. K is given by the user. Next, 2) it calculates the distance between each data point and centroids and assigns the data point to the nearest centroid. Then, 3) it recalculates the new cluster center as the average value of the points belonging to that cluster. Finally, 4) it stops when no data point is reassigned, otherwise it repeats from step 2).
4. **Evaluation and Labelling**: As previously mentioned, KDD process is iterative and then, several clustering models should be built and compared. There is not a technique better than another, but the success depends on choosing the best algorithm for the dataset under study. There are several indices to measure the quality of the models but these must be complemented with the degree of understandability of pattern as well as their actionability. Therefore, visual evaluation as well as based on indices must be performed. As a consequence of this evaluation, a clustering model will be chosen and its clusters labelled according to the metrics used. These labels should provide useful and valuable information regarding the specific decision-making process.
5. **Deployment**: Finally, if the outcome is new, useful and actionable could be directly used or be integrated as an input in another process.

For instance, it could be applied to achieve a better schedule of applications over the compute nodes so as to improve both performance and energy efficiency. In this way, the proposed KDD process could be applied as follows. Firstly, a set of metrics to describe the state of use of each element in a compute node, such as CPU or Memory utilization, must be monitored, storing this data in a persistent system. On this data, the steps 2 and 3 would be performed. Finally, a series of labels would be defined for the clusters found,

such as: HigCPU, LowCPU, HighMEM, LowMEM, and so on, which indicate the degree of use of each component. These labels would be assigned to the clusters, in such a way that a single cluster could have two or more labels. For instance, a label such as "LowCPU-HighCom" would be assigned to a cluster where all the nodes run applications that require little use of CPU but a heavy use of the communication network. On the other hand, users should classify their applications based on these labels and a job scheduler should perform the matching between requirements and compute nodes.

Additionally, clustering also allows us to identify those nodes that are underused. This information could be used to optimise energy consumption [6]. Processes or virtual machines running on the these little loaded nodes could be grouped and migrated to a reduced set of nodes, which would become more loaded, but this would allow that the leftovers nodes could be turned off, and thus saving the static power they consume. Furthermore, other important applications that can benefit from this methodology are, for example, the batch processing in background that can take advantage of the less loaded nodes to run applications that are not time-critical or the provisioning of both hardware and software services in cloud computing.

An advantage of the clustering algorithms is that they are very fast and can be frequently applied with an inappreciable computational cost, getting thus an updated insight of the current status of the resources. This means that the decision making process will be computational cost, getting thus an updated insight of the current status of available resources, and that the decisions can be made with real time information, allowing a more dynamic process, not fixed to some pre-established parameters. Even more, the set of labels can be kept since the meaning of these labels will depend on the current conditions of the nodes in the data center. This avoids the use of static thresholds that are very rigid to determine the states of the computer nodes. Instead, the information provided by clustering, dynamically adapts to the global system status. Thus, if the overall system is heavily loaded, the value of the Low label can be 60%, but if it is very low loaded this value can become 10%, since its value will be taken from the centroid which represents the group.

4 Case study

The study presented in this paper has been carried out on a particular case of a High Performance Computing cluster called "Altamira", which belongs to the University of Cantabria [1]. This platform consists of up to 158 IBM-idataplex dx360m4 nodes and four login servers. Each node has two Intel Sandybridge Xeon E5-2670 processors, 8 cores (16 cores per node) operating at 2.6 GHz per processor, 64 GB of main memory (DRAM) and 500 GB local disk. These nodes are interconnected by a folded Clos topology using Infiniband FDR10 network technology. The peak capacity exceeds 50 Tflops.

[1] https://grid.ifca.es/wiki/Supercomputing

Table 1 Average and standard deviation (in brackets) of the metrics selected

Dataset	CPU	MEM	HDD	NET
Dataset 1 (d1)	49.96(39.47)	35.93(31.59)	7422.09(22756.98)	47014.29(171040.45)
Dataset 2 (d2)	56.58(38.91)	28.15(27.03)	6571.77(21640.65)	53609(174806.102)
Dataset 3 (d3)	68.89(37.44)	34.66(28.33)	35897.27(45713-48)	819752.81(160218.51)

In order for the study to be representative enough, three different packages of applications were programmed to be executed in the data center and a monitor was developed to measure the metrics to be used to characterise the state of nodes. The monitor was compiled with GCC version 4.4.7 and the OpenMPI 1.8.2 library, which was the latest stable release version at the time. Two of the packages comprised a set of CPU-bound applications (d1 and d2 dataset) whereas the third one (d3) contained Memory-bound and Communication-bound programmes.

Regarding the metrics to be monitored, we must mention that their selection heavily depend on the particular purpose of the data center. For our goal that is to describe the state of the nodes of a HPC data center according to the degree of occupation of the resources, the following measures were chosen:

- **CPU Load**: This metric, labelled as CPU, measures the percentage of processor usage. It ranges from 0%, the CPU is not used at all, to 100%. It is worth remembering that each node comprises 16 cores. Hence a value of 50% might mean that all nodes are charged to 50% or the half of the nodes are fully occupied and the rest are empty.
- **Memory load**: This measurement, named MEM, determines the amount of occupied main memory in each computing node. It ranges from 0%, the memory is not used at all, to 100% where the memory is fully occupied.
- **Hard disk load**: This value, labelled as HDD, gathers the amount of data read/written from/to the local hard disk, expressed in a number of Kbytes per second. It ranges from 0, hard disk is not used, to about 120.000 KB/s, that is the maximum speed of the hard disk. A high value of this metric means that the applications are data intensive, and hence they barely use CPU.
- **Network load**: This metric, labelled NET, measures the bandwidth used in each computing node, i.e., the number of bytes sent and received per second in the network interface. This parameter gathers the level of congestion of a node with respect to communications. All nodes are connected by the same network (Infiniband FDR10). It ranges from 0 to 40 Gbps.

It must be highlighted that these metrics were read in a concrete time window. Five points were taken (with one second of difference between them) for each node and next, averaged. Table 1 shows the average value and standard deviation of the metrics in each dataset. Each dataset comprises 150 instances, one for each node.

Next, datasets were analysed and prepared for applying the clustering algorithm. As mentioned, *k-means* was selected , although other algorithms such

Table 2 Distribution of the four metrics in d1 dataset

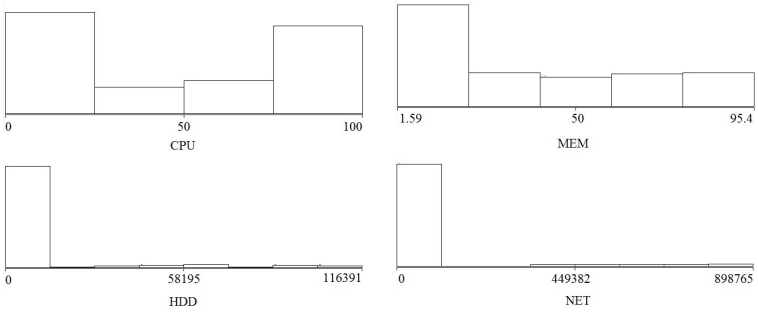
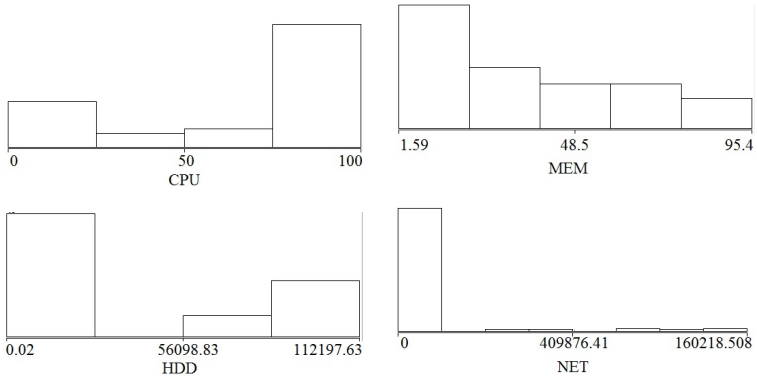


Table 3 Distribution of the four metrics in d3 dataset



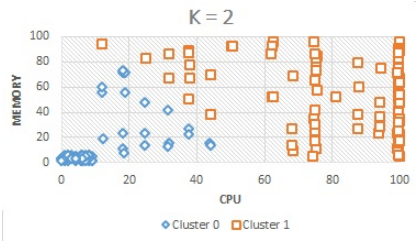
as k-medoids was also used. One of the main concerns by using any of both algorithms is selecting the value of the K parameter, that means, the number of clusters or groups to generate. Although there are some proposals which try to recommend a good K value, neither of them guarantees an optimal value. So, in this study, we executed *k-means* with K ranged from 2 to 6. We used the *SimpleKMeans* implementation of this algorithm offered by Weka [8].

As several models were built, the Davies Bouldin index (DB)[3] was used for assessing their quality. DB index measures the average of similarity between each cluster and its most similar one. As the clusters have to be compact and separated, the lower DB index means better cluster configuration. This index ranges from 0 to infinite. This measure was computed by means of *clusterSIM* library available in R software [13]. The value of this index along with the analysis of the textual and graphical results of the models built allow us to select which model is more informative and works better to our goal.

Next, we graphically show the distribution of values of each metric of the d1 and d3 datasets in Tables 2 and 3 respectively with the aim at easily observing how these change according to the kind of applications executed and ensuring that there were nodes with different degree of load. Since d2

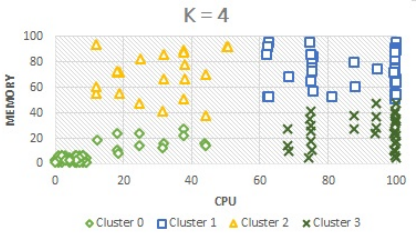
Table 4 Centroids, David Bouldin index and cluster distribution for d1 dataset

k=2	CPU	MEM
C0	10.81	13.05
C1	80.72	53.90
DB	0.71	

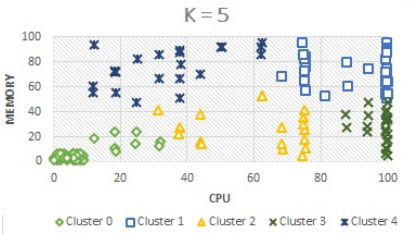


k=3	CPU	MEM
C0	10.97	10.68
C1	68.17	76.83
C2	91.39	29.29
DB	0.82	

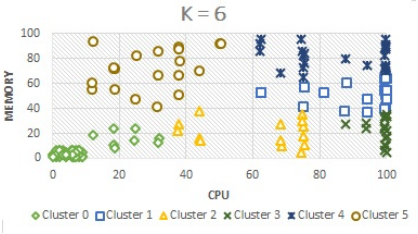
k=4	CPU	MEM
C0	9.65	6.63
C1	87.21	72.53
C2	30.08	70.03
C3	90.45	25.01
DB	0.70	



k=5	CPU	MEM
C0	7.34	5.65
C1	91.10	72.04
C2	60.70	26.67
C3	97.76	26.43
C4	33.77	75.66
DB	0.76	



k=6	CPU	MEM
C0	7.34	5.65
C1	91.32	52.09
C2	61.53	20.89
C3	98.40	21.28
C4	85.18	81.85
C5	29.38	71.62
DB	0.80	



dataset is similar to d1 and, due to space constraints, it is not depicted. As can be observed, CPU-intensive applications mainly use CPU and MEM, being HDD and NET close to 0, whereas data-intensive applications also make use of HDD. Neither of them require NET resources, so this metric was eliminated from the datasets.

Once datasets were ready, *k-means* was applied on them. Table 4 depicts the centroids of the five clustering models built (k=2 to k=6) with d1 dataset only using CPU and MEM metrics along with the percentage of nodes that belong to each cluster and their DB index. Likewise, the assignation of nodes to clusters is shown in a scatter plot so that the model can be visually evaluated.

Let us describe the first model in detail, i.e., k=2. Here, the main differences between the two obtained clusters, Cluster 0 (C0) and Cluster 1 (C1), is that the MEM average value in C1, 53.90%, is higher than in Cluster 0, 13.05%, and C1 has also a higher CPU, 80.72%, than C0, 10.81%. However, the range of values of CPU in C1 is between 20% and 100%, and MEM ranges from 0% to 100%. This clustering model therefore is not very informative. When the number of clusters is 3, the DB index worsens, and the Cluster 1 is visually inconclusive since CPU also takes values that range from 20% to 100%.

The clustering model with k=4 is the most informative. Its DB index is the lowest, thus the best, and furthermore, the clusters are very compact. Even more, they perfectly discriminate four quadrants. Cluster 0 gathers nodes with low CPU and MEM, Cluster 1 groups the nodes with high CPU and MEM, meanwhile Cluster 2 contains the nodes with low CPU and high MEM and Cluster 3 agglutinates the nodes with high CPU and low MEM.

The model with k=5 has a worse DB index and the Cluster 2 is little compact. Finally, increasing to 6 the number of clusters led to a worsening in the model, reaching again, as with k=3, a DB index of 0.80. The Cluster 0 is exactly the same that with k=5, and the only difference is that a new cluster can be observed, the Cluster 1, that contains nodes with high CPU and medium MEM.

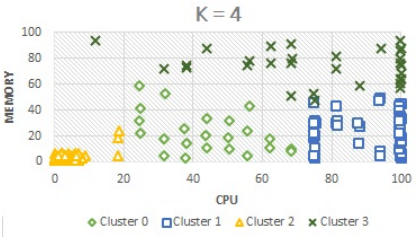
Next, we assess the models built with d2 dataset. As this is similar to d1 dataset, only we show, in Table 5, the results corresponding to k=4 and k=5. In this case, the setting with k=5 is quite better, since that the clusters are more compact, although the DB index is only slightly lower.

Table 6 illustrates the models achieved with d3 dataset, where HDD metric was considered for its relevance. As in previous models with k=2, the clusters have not got a good quality, high DB index and centroids with very near values in CPU and MEM attributes.

The model improves with k=3 but it is with k=4 when the clustering is really informative. As can be observed, this model groups the nodes in four well differentiated clusters. C0, with a low CPU load and MEM but high HDD usage; C2, with high CPU and HDD and higher MEM usage than C0; C1, with high CPU, medium MEM and low HDD; and C3, with low MEM, CPU and HDD. Moreover, its DB index also confirms that it is the best model. A higher value of k did not achieve a better result.

Table 5 Centroids, David Bouldin index and cluster distribution for dataset2

k=4	CPU	MEM
C0	44.29	22.39
C1	92.21	23.93
C2	5.16	4.40
C3	75.56	74.2
DB	0.76	



k=5	CPU	MEM
C0	45.82	19.23
C1	92.48	22.54
C2	5.16	4.40
C3	47.29	76.88
C4	91.60	66.85
DB	0.75	

The experimentation performed demonstrates that a high level description of the state of the nodes of a data center is feasible and its outcome is easily understandable for non data miner experts. The computation time is inappreciable, practically fractions of seconds. Thus, clustering is shown to be useful and effective for our purpose.

The same conclusions were drawn when applied *kmedoids* clustering algorithm, which generates clusters selecting real points, instead averaged as centroids. In Table 7 we shown a comparison of the DB index obtained with *kmedoids* and *kmeans* for d1 dataset, concluding that, ranging the k value from 3 to 6, the best clustering model is obtained in both cases with k=4.

However, *kMedoids* models are visually less informative due to this technique requires a higher number of instances to build more compact clusters.

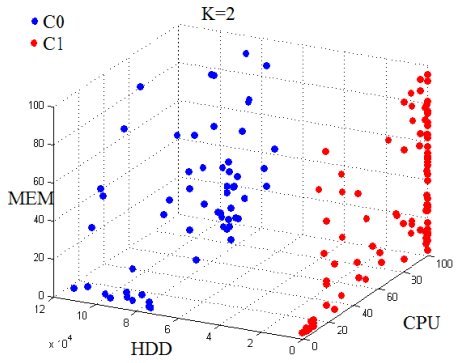
5 Conclusions and future work

This paper presents a methodology for dealing with the real time data analysis in DCIM tools. In particular, we propose clustering as a useful technique to describe the state of a set of nodes in a data center. This innovative approach might well be utilised so that users can describe their applications according to a limited set of simply and understandable labels that, in turn, the job manager would use to better schedule its jobs. This technique simplifies the decision-making process, while introduces dynamism in the description of the state of the nodes.

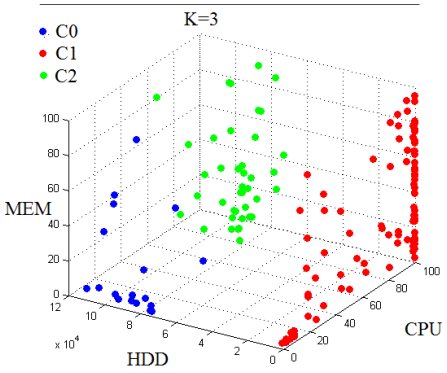
Our experimentation shows that different groups of nodes can be characterised according to the hardware requirements with which a new incoming

Table 6 Centroids, David Bouldin index and cluster distribution for d3 dataset

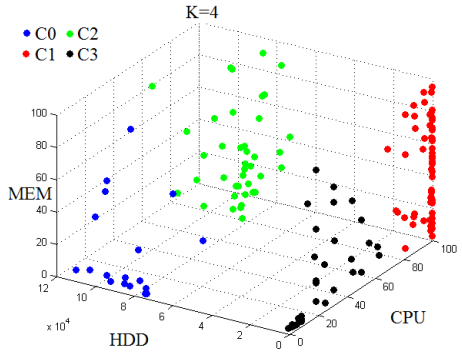
k=2	CPU	MEM	HDD
C0	66.18	27.61	92582.03
C1	70.60	39.11	161.23
DB	1.15		



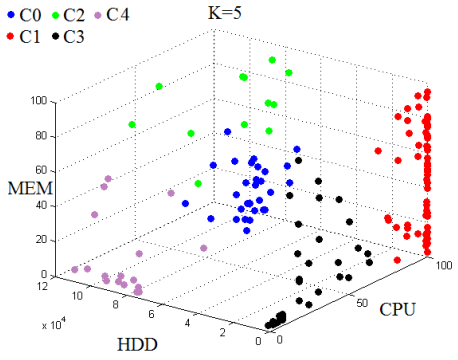
k=3	CPU	MEM	HDD
C0	12.44	17.49	92682.47
C1	70.60	39.11	161.23
C2	90.36	32.15	92536.83
DB	0.90		



k=4	CPU	MEM	HDD
C0	12.44	17.49	92682.47
C1	95.40	47.14	164.65
C2	90.36	32.15	92536.83
C3	24.08	24.05	154.83
DB	0.81		



k=5	CPU	MEM	HDD
C0	91.91	18.15	92491.63
C1	95.40	47.14	164.65
C2	81.68	69.83	93672.79
C3	24.08	24.05	154.83
C4	11.34	13.94	91966.28
DB	0.83		



application should be executed. This would be very useful in order to automatize and optimize the usage of the data center so the applications are not running in the same nodes, avoiding the overload by using free-nodes or low-occupied nodes for new applications. The current work is in an initial phase.

In a near future, we will extend the experimentation by using more metrics and considering other data mining techniques. Then, we will try to develop the software component to speak to both system administrator and end user.

Table 7 DB index values by applying k-means (mn) and k-medoids (md) on d1 dataset

k=3		k=4		k=5		k=6	
mn	md	mn	md	mn	md	mn	md
0.82	0.74	0.70	0.68	0.76	0.85	0.80	0.77

Acknowledgements This work has been supported by the Spanish Science and Technology Commission (CICYT) under contract TIN2013-46957-C2-2-P and CAPAP-H5 network TIN2014-53522, the European HiPEAC Network of Excellence.

References

1. Luiz André Barroso, Jimmy Clidaras, and Urs Hlzle. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, Second Edition*. Morgan and Claypool Publishers, 2013. Synthesis Lectures on Computer Architecture.
2. Pete Chapman, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer, and Rudiger Wirth. Crisp-dm 1.0 step-by-step data mining guide. Technical report, The CRISP-DM consortium, August 2000.
3. David L. Davies and Donald W. Bouldin. A cluster separation measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-1(2):224–227, April 1979.
4. Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. The kdd process for extracting useful knowledge from volumes of data. *Commun. ACM*, 39(11):27–34, 1996.
5. César Gómez-Martn, Miguel A. Vega-Rodríguez, and José-Luis González-Sánchez. Performance and energy aware scheduling simulator for hpc: evaluating different resource selection methods. *Concurrency and Computation: Practice and Experience*, 27(17):5436–5459, 2015. cpe.3607.
6. J.Octavio Gutierrez-Garcia and Adrian Ramirez-Nafarrate. Agent-based load balancing in cloud data centers. *Cluster Computing*, 18(3):1041–1062, 2015.
7. Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3):107–145, 2001.
8. Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1):10–18, November 2009.
9. Jiawei Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.
10. Mark Harris and Hwaiyu Geng. Data center infrastructure management. *Data Center Handbook*, pages 601–618, 2015.
11. Matt Massie, Bernard Li, Brad Nicholes, Vladimir Vuksan, Robert Alexander, Jeff Buchbinder, Frederiko Costa, Alex Dean, Dave Josephsen, Peter Phaal, and Daniel Pocock. *Monitoring with Ganglia*. O'Reilly Media, Inc., 1st edition, 2012.
12. Sergio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.
13. R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013. ISBN 3-900051-07-0.
14. Greg Schulz. *The Green and Virtual Data Center*. Auerbach Publications, Boston, MA, USA, 1st edition, 2009.
15. Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, GeoffreyJ. McLachlan, Angus Ng, Bing Liu, PhilipS. Yu, Zhi-Hua Zhou, Michael Steinbach, DavidJ. Hand, and Dan Steinberg. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008.
16. Rui Xu and II Wunsch, D. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678, May 2005.
17. Marta E. Zorrilla and Diego García-Saiz. A service oriented architecture to provide data mining services for non-expert data miners. *Decision Support Systems*, 55(1):399–411, 2013.