Managing Energy Efficiency and Quality of Service in Cloud Applications Using a Distributed Monitoring System

Monica Vitali[1]

Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy

Abstract. Energy efficiency and quality of service are two important aspects in cloud applications. Their management requires the collection of a large amount of data through a monitoring system which monitors the behaviour of the application. In order to provide an effective approach towards efficiency and quality management, it is important to investigate relations and mutual influences between the monitored metrics. This analysis can be time and computational expensive. In this paper we propose to monitor the system in a distributed manner. We also propose to use a map-reduce like approach to analyse relations among the monitored variables by executing the computation locally and then sending the results to a coordination node. In this way we are able to create a dynamic model of the relations among monitored variables, here represented as a Bayesian Network.

1 Introduction

Energy consumption in modern data centers is becoming a relevant issue that can not be ignored by data center owners. Improving the efficiency of the hardware is an important step but it is not enough. Also applications plays an important role in energy consumption and an efficient management of applications running in clouds can substantially improve Energy Efficiency (EE) and reduce CO_2 emissions [4][5]. To reach this goal, applications need to be monitored and evaluated. The evaluation enables the enactment of proper strategies to improve the EE and the quality of service (QoS) of the application itself.

The considered scenario takes into account a service oriented architecture where an application can be deployed in a distributed way in the cloud. The application is decomposed in separated tasks, each one hosted in a dedicated Virtual Machine (VM). VMs are allocated to physical servers in the cloud but they can be migrated at any time if needed. In such an environment, a typical monitoring system will require to collect data in a centralized way by monitoring every relevant component through a monitoring agent and collecting all the data in a unique storage device in the cloud. This methodology allows a coordinated management of the monitoring system but can be inefficient when the data to be collected are copious. In fact, the centralized system can meet with performance issues which result in lower precision. Also, when the monitored data are used to raise alarms about the behaviour of the application, the delay in the detection of a non desired behaviour can provoke a delay in fixing the arisen issue.

In this work we face the issue of efficiently monitoring and managing applications in a cloud environment. We propose a goal oriented model where adaptation strategies can be addressed when an inefficiency is discovered. We state that understanding relations between the variables of the system is an important step to enable adaptation. To discover these relations, we propose to use an algorithm able to automatically learn a Bayesian Network (BN) from monitoring data. Each node of the BN is a variable of the system and a relation between two nodes represents a causal relation among two variables. We propose a distributed monitoring system which monitors some system and application related metrics able to represent the efficiency and QoS of applications. The distributed approach reduces the cost and the time needed to transfer the monitoring data from the monitored component to a central collection storage. We exploit the distributed monitoring system by running a distributed analysis of the application behaviour to learn the BN. This approach considerably reduces computation time and increases scalability.

The rest of the paper is organized as follows. In Sect. 2 we analyse the state of the art. Sect. 3 introduces the goal-oriented approach and the algorithm for learning the BN. Sect. 4 proposes the implementation of a distributed monitoring system to enable the distributed learning algorithm described in Sect. 5. Finally, results are discussed in Sect. 6 and some final remarks are presented in Sect. 7.

2 State of the art

Energy consumption in ICT is growing annually by 4% despite efficiency gains in technology, and its carbon impact is now comparable to air travel [12]. For these reasons, also the application perspective should be considered in order to reduce energy consumption in data centers and clouds. This topic has been widely studied in recent years and a wide range of methodologies to manage EE in ICT has been proposed [17][2][14]. However, EE comes at a cost and it is therefore important to study ways to reduce energy consumption while preserving acceptable levels of QoS. These two aspects are usually in contrast as discussed in [7]. The first step to manage these two perspectives consists in the evaluation of the system that is obtained through monitoring relevant metrics. In [6] a set of Green Performance Indicators (GPIs) and Key Performance Indicators (KPIs) are proposed to state the EE and the QoS in data centers from an application point of view. These metrics have been enriched in the framework of the ECO$_2$Clouds project[1] with new carbon-aware metrics for cloud applications in [16]. After the assessment, retrieved information can be used to adapt the system towards a desired state as proposed in [9] where collected data are used to mine relevant events, and in [5] where a smart deployment algorithm is proposed to reduce CO_2 emissions in clouds.

[1] ECO$_2$Clouds: http://eco2clouds.eu/

When dealing with a big amount of data travelling from a location to another as in cloud computing, the cost of the transportation can not be ignored [1]. The efficiency in data transmission has been discussed in the literature (e.g. [3]). Decentralization in the monitoring system and in data analysis can considerably reduce this issue by lowering the amount of data travelling in the network. First steps towards a distributed monitoring system have been discussed in the literature. In [18] authors state the inefficiency of using a centralized monitoring approach when dealing with large amounts of data and propose to use a decentralized approach to efficiently detect and filter events, and discover their temporal correlations. Events are aggregated in a set of databases rather than a centralized database, where the event correlation mining can be executed in parallel. A different scenario is depicted in [13] where a distributed monitoring approach is applied to wireless monitoring of energy consumption in smart buildings in order to learn the user usual behaviour and to reduce energy waste. Also in [10] the problem is faced. Here, the authors propose self-organized computational structures that can pool together their computational resources to collect and analyse data in real-time for decision making in the field of robotic swarms.

In this paper we propose to apply the concept of distributed monitoring and analysis to a cloud environment in order to improve EE and performance of applications deployed on the cloud.

3 The proposed approach

Managing EE and QoS of applications in a cloud environment is challenging. In fact, a lot of variables can have an impact on both these aspects. Also, since these two perspectives are in conflict in most of the cases, it is necessary to balance them in order to fulfil as much as possible the requirements of the service owner organization.

In this paper we propose to manage EE and QoS of an application using a goal-oriented model. The model, described in [15], is composed of two layers:

- The Goal Layer - In the upper layer we define the goals of the organization providing the service. Goals are expressed through variables that are able to properly express the behaviour of the system both in terms of EE and in terms of quality of the service provided. According to this, in the goal layer there are two kinds of variables: GPIs (Green Performance Indicators) used to measure the efficiency in terms of energy, and KPIs (Key Performance Indicators) used to measure quality. Several GPIs and KPIs have been defined in the literature . The selection of a subset of them depends on the goals of the application provider and users. A set of thresholds is assigned to each of these indicators. Thresholds allow the customer to define which are the acceptable range of values. The selected thresholds characterize the strategy of the customer who can be more sensitive towards some specific aspect and less towards others.
- The Treatment Layer - In the lower layer we represent possible adaptation strategies that can be applied to fix undesired states for the goals. When

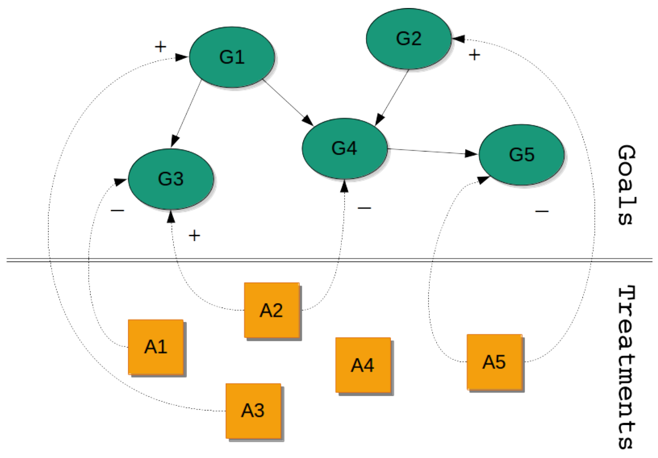


Fig. 1. A goal-oriented model for managing EE and QoS of a cloud application

indicators are violated a strategy can be selected in order to improve the state of the indicators and bring the system back into the desired situation. Actions and goals are linked together. A link between these two components of the model means that the action has an effect on the goal and can be used to improve its state. Treatments and their selection strategy are not of interest in this work, but a description can be found in [15].

A generic representation of the proposed goal-oriented model is shown in Fig. 1. In this work we focus on the upper level and in particular on the issue related to the evaluation of the values of the indicators and the exploitation of this knowledge to improve EE and QoS. As can be observed from the model, goals in the upper level are interconnected. These connections express interdependencies existing among the satisfaction of indicators. When a relation exists between two goals, then violations or satisfactions of these variables are not independent and so, a modification in the state of the first has an impact (positive or negative) in the state of the other one. These relations are sometimes obvious but in other cases hidden. Also, in a cloud environment where a lot of variables are monitored, drawing these relations can be a very complex and time expensive task. For this reason we propose to automatically learn relations using a BN representation, where each node in the network is the representation of an indicator while a link expresses influences between goals. In [15] we proposed an algorithm for automatically learn a BN from the monitored data. The algorithm follows three main steps. First of all, a large set of monitored data collected for a long period of time are analysed to discover strong relations through correlations. Then, starting from this knowledge, the BN structure is built by selecting relevant edges and directing them using the Max-Min Hill Climbing (MMHC) algorithm [11]. Finally, the conditional probability table for each of the links is computed using well known techniques such as Maximum a Posteriori Estimation (MAP). That approach was suitable for a data center where dimensions can be relevant but limited. In a cloud environment, an approach requiring to store a big amount of monitored data and to process it all at once for computing the BN describing all the relations among goals is not scalable. Also, since applications deployment in the cloud is dynamic and can change over time, the BN needs to be recomputed in order to reflect the real structure of the system. From experimentation, we observed that the computation time increases exponentially with the number of variables, so each computation requires an amount of time that is not feasible with the needs of a dynamic environment.

In this paper we propose to use a distributed monitoring system and a distributed BN computation algorithm in order to increase the scalability of the approach and to reduce the latency needed for the computation of the BN. First of all we analyse the monitoring system, introducing a set of indicators and explaining how they can be collected in a distributed way (Sect. 4). Then we adapt the algorithm described in [15] for learning interconnections among goals to a distributed environment, making it more dynamic and scalable (Sect. 5).

4 Distributed monitoring of energy and quality related metrics

Monitoring EE and QoS in a cloud application requires the collection of a consistent amount of data during all the lifetime of the application. Indicators are computed starting from the data collected by a monitoring system. Several works proposed a set of indicators valuable for assessing EE, QoS, and sustainability of applications in data centres and clouds [8][16][6]. Starting from this set of indicators we selected a subset that will be used as an example in the rest of the paper. However a different subset could be used without affecting the approach. We classify metrics using two different perspectives. First of all we distinguish between:

– Performance metrics: used to measure the performance of a service or system;
– Energy metrics: used to assess the efficiency in terms of energy consumption of a service or system.

The other perspective is related to the component of the system that is monitored by the metric. According to this we distinguish between:

– Application level metrics: used to monitor the behaviour of the application or part of it;
– Server/VM level metrics: used to monitor a physical or a virtual machine hosting the application;
– Data center level metrics: used to monitor more in general a data center which contains the physical machines hosting the application.

In Table 1 we show a set of metrics that we are going to use as an example in the rest of the paper and the category to which they belong for each of the two perspectives.

Table 1. Classification of the considered indicators

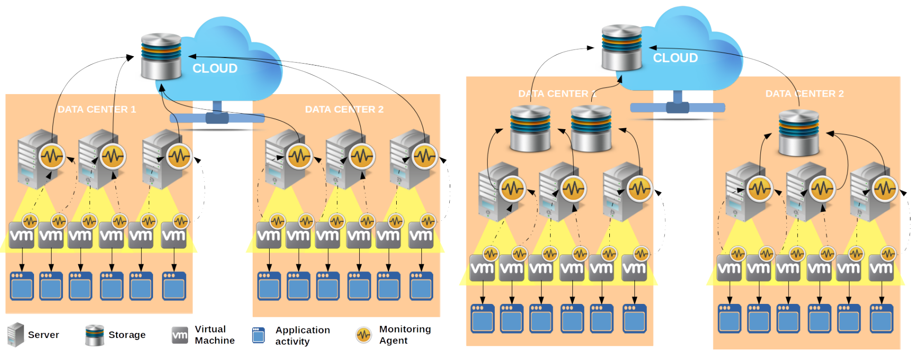
Indicator	**Levels**	**P/E**
Energy	all levels	E
CPU Usage	VM, host	P, E
Memory Usage	VM, host	P, E
Response Time	application	P
I/O throughput	application	P
Performance per Energy	application	E
Throughput	application	P
Storage Usage	application, host	P, E
CO_2 Emissions	application	E
Energy mix factor	data center	E

Metrics are described with a formula and a support. The formula express how the metric can be computed starting from raw data collected by the monitoring system. The support allows the user to define the range of desired values for each of the metrics. When the value of the metrics is inside this range the goal is considered satisfied, otherwise this is a trigger for enacting adaptation.

The most extensively used monitoring systems (e.g. zabbix[2] and nagios[3]) allow the monitoring of some basic information and the customization of the monitored data by creating a set of scripts that are executed by a monitoring agent (Fig. 2(a)). Samples are collected at different rates which can be specified by the user. Once retrieved, the data are moved in a centralized database which keeps the information for further analysis. In such an environment, the size of the collected information increases quickly and the transmission of such an amount of data to the collecting point can be problematic, forcing to increase the sampling interval loosing in precision. The collected data are only the first step of the monitoring system. In fact, they need to be manipulated to compute the indicators selected by the client of the cloud infrastructure. Usually, also this computation is performed in a single node executing queries to the centralized database. Another disadvantage of a centralized collection of monitored data is due to the fact that to perform analysis on the behaviour of the indicators this big amount of data needs to be manipulated. If these computations are complex, they will require a lot of time to be all executed.

In our approach we propose to use a distributed monitoring system as shown in Fig. 2(b). Data are retrieved hierarchically by the monitoring agents running on the servers and on the VMs. One or more storage devices in each data center are in charge of storing local data obtained from monitoring the servers, the VMs and the specific tasks of the application deployed on them. Data needed for monitoring the whole application are sent to a global storage device in the

[2] Zabbix: http://www.zabbix.com/
[3] Nagios: http://www.nagios.org/



(a) Centralized monitoring management (b) Distributed monitoring management

Fig. 2. Comparison between the centralized monitoring system management and the proposed distributed approach

cloud. Given this structure, also the analysis of the data can be performed in a distributed way. We propose to analyse data in two steps:

- *local analysis*: computations are executed on each host involved in the cloud application to compute local indicators and to perform local analysis. In this step, the computation of indicators at the VM, activity, and host granularity level is executed;
- *global analysis*: some of the collected or computed data are moved to a centralized node which is in charge of computing and analysing indicators at the application and cloud level.

The distributed monitoring system keeps local indicators inside a local storage and global indicators in a storage device in the cloud. In the following section we exploits the advantages of a distributed monitoring system for performing the analysis of the relations between goals in a scalable and efficient manner.

5 A distributed algorithm for learning metrics relations

The distributed monitoring system approach represented in Fig. 2(b) is the starting point for performing a distributed analysis over the collected data. Given the described architecture, data are stored in several storage devices and each of them can be accessed by a local server to perform analysis. In this section we exploit the proposed architecture for improving the BN learning algorithm described in [15] using a map-reduce inspired approach.

As introduced in Sect. 3, the original algorithm is performed in three steps:

- Correlation computation: in this step all the historical values for the indicators are used to discover correlations;
- BN structure learning: causal relations between the states of goals are discovered applying the MMHC algorithm which deletes or directs edges;

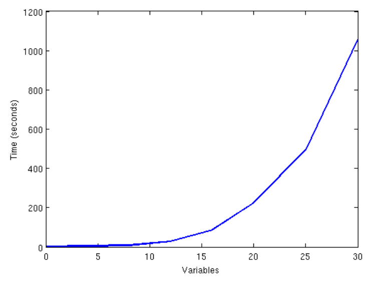


Fig. 3. Computation time exponential growth of the centralized learning algorithm [15]

– CPT learning: given the structure, conditional probability tables are learned using existing approaches, such as MAP.

The discovered BN can be used to perform "what if" analysis and to predict future states of indicators. The algorithm has been proven to be effective, with an average success rate in prediction of 87.5%. However, the computations needed to build the network are complex and require a significant amount of time to be executed. Also, computation time increases exponentially with the number of variables composing the BN, as shown in Fig. 3. With 30 variables, the computation of the network takes about 18 minutes to be executed. In a cloud environment we expect the number of monitored variables to be much higher than 30, making the solution not feasible in a dynamic environment.

Experimental results have proven that relations are automatically organized in clusters which reflect the physical structure of the system. An example is shown in Fig. 4[4], representing goals relations in a data center running an application composed of three task deployed in three VMs and running on two different servers. As can be observed, relations are discovered between indicators monitored on the same VM (e.g. V1), and between indicators monitored on a server and the ones monitored on the VMs deployed on it (e.g. between S1 and V1 or V3). On the contrary, no relations are discovered between indicators of different VMs (e.g. V1 and V3) or of different servers.

This result can be exploited to learn the network in a distributed way, computing in parallel different parts of the network and then merging them together to build the complete network. This is allowed by the conditional independence property of BNs. The pseudocode of the algorithm for learning the BN of the application related indicators in a distributed environment is shown in Alg. 1. The algorithm decompose the original problem of discovering relations among all the indicators into sub-problems where a limited set of indicators are considered. First of all, the learning algorithm is executed for each VM involved in the application building the network for VM level and activity level indicators. This operation can be executed sequentially or in parallel for all the VMs, since

[4] Legend: V_i = VM, S_i = server, $U(x)$ = cpu usage, $R(x)$ = response time, $PE(x)$ = performance per energy, $E(x)$ = energy consumption

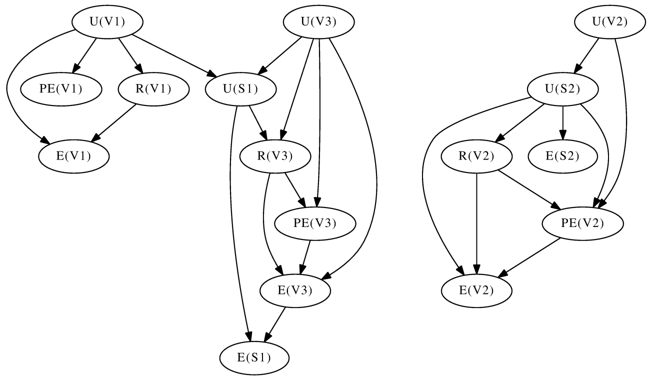


Fig. 4. An example of BN discovered by the algorithm: variables relations are automatically organized in clusters according to the physical structure of the system[15]

ALGORITHM 1: The distributed BN learning algorithm

```
**Input**: the monitored values at the different monitoring levels $\{I(x)\}$ for the application $App$
**Output**: a Bayesian Network describing relations among variables $BN_{App}$
$\forall$ virtual machine deployed on a server $S_j$ $VM_{ij}$ do:
    find the correlations $\mathbb{C}_{ij}$ inside the set $\{I(VM_{ij})\}$;
    find the BN structure $BN_{ij}$ for $\mathbb{C}_{ij}$;
    compute the CPT for $BN_{ij}$;
end;
$\forall$ server $S_j$ do:
    find the correlations $\mathbb{C}_j$ between $\{I(S_j)\}$ and $I(\{[VM_{ij}]\})$ representing the VMs deployed on
    the server $S_j$;
    find the BN structure $BN'_j$ for $\mathbb{C}_j$;
    compute the CPT for $BN'_j$;
    merge $BN'_j$ with $BN_{ij}$ in a unique network $BN_j$;
end;
find the correlations $\mathbb{C}_{\mathbb{A}ll}$ between $\{I(App)\}$ and $I(\{[S_j]\})$ representing all the servers hosting an
application activity;
find the BN structure $BN'_{App}$ for $\mathbb{C}_{App}$;
compute the CPT for $BN'_{App}$;
$\forall$ j merge $BN'_{App}$ with $BN_j$ in a unique network $BN_{App}$;
**Return** $BN_{App}$;
```

each sub-problem is independent from the others. The second step consists in computing the server level network. The computation is executed for each server involved in the application by looking for relations among the server variables and the VM and activity variables of all the VMs deployed on the server. Even if this set of variables can be big, we are not looking for relations among all the considered variables, but only between the few server variables (in our previous example only variables) and all the others, thus reducing the computation time. Finally, the application level metrics are considered and relations are discovered between them and all the other variables. As before, the number of variables to investigate is considerably reduced.

A graphical representation of the algorithm behaviour is shown in Fig. 5.

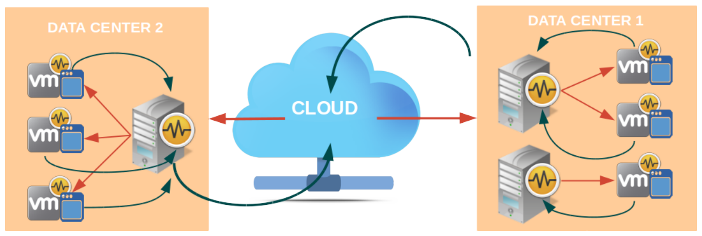


Fig. 5. A graphical representation of the steps of the distributed learning algorithm

6 Results

The results, obtained using the distributed algorithm for learning relations between goals described in Sect. 5 and enabled by the distributed monitoring approach described in Sect. 4, are compared with the centralized approach in terms of the computation time needed to build the complete BN. We start with the same configuration for both approaches, in which we have two servers hosting one VM each. We monitor four different metrics for each VM and associated activity: CPU usage, response time, performance per energy and energy consumption. For each server we consider only CPU usage and energy consumption. To evaluate the scalability of the algorithm, we observe how the system reacts to the deployment of new VMs while keeping fixed the set of monitored metrics and the number of servers. We compare also the sequential and parallel approach to evaluate the worst and the best case for each configuration.

Results are shown in Fig. 6 where, for each number of VMs, the third bar represents the centralized execution time, the second bar represents the distributed algorithm using sequential execution (worst case), and the first bar represents the results of the distributed algorithm with parallel execution (best case). As can be observed, the computation time of the distributed algorithm is considerably reduced and, more important, it grows linearly with the number of VMs instead of exponentially as in the centralized version. The data reported in Fig. 6 are valid for the first computation of the network. Since we are dealing with a dynamic environment in which the deployment of the application can change over time, the network needs to be updated at each modification. The distributed approach simplifies this step. In fact, every time a modification occurs, only the part of the network involved in the modification has to be recomputed, while the rest of the network remains fixed. As an example, in the BN shown in Fig. 4, if V1 is removed only relations between S1 and V3 are affected. If instead a new VM V4 is added to server S2, only the BN for V4 needs to be discovered and the relations between variables on S2 and variables related to V2 and V4 have to be updated. Other VMs (e.g. V1 and V3) or servers (e.g. S1) are not affected. The linear behaviour and the fast update are big improvements in comparison with the original centralized algorithm.

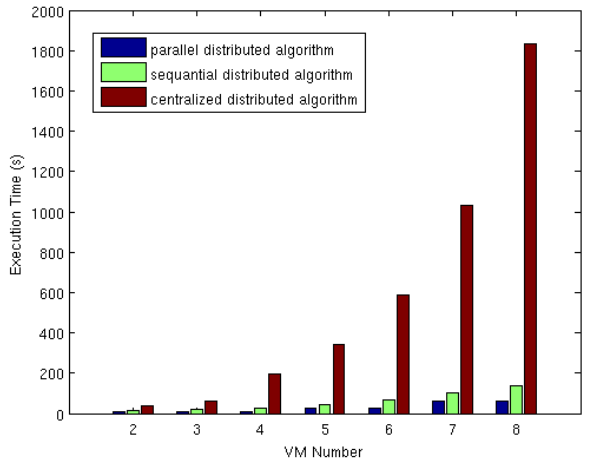


Fig. 6. Computation time comparison for the BN learning algorithm

7 Conclusion and future work

In this work we have proposed to manage EE and QoS in cloud applications using a goal-oriented model. To enable the model to work properly, relations among metrics need to be discovered. A centralized approach is not scalable for this purpose. We proposed a distributed monitoring system, allowing the retrieval and storage of monitoring information locally, to reduce costs and delays due to data transmission in a centralized data collector. Indicators related to EE and QoS of the monitored application can be computed and stored in a hierarchical way, starting from the VMs hosting the application tasks. The distributed monitoring system enables an efficient analysis of the indicators. Relations between indicators are discovered using advanced machine learning techniques while the time needed for the computation is optimized. The proposed distributed algorithm learns a BN representing causal relations existing among goals by decomposing the problem into smaller steps at different granularity levels: isolated networks for each VM involved in the application are learned and then they are composed together in a unique BN using a map-reduce inspired approach. Results have shown that the computation time of the proposed algorithm is reduced between the 62% and the 92% when all the networks are computed sequentially, and between the 81% and the 96% when parallel computation is enabled. The advantage of using the distributed algorithm keep growing with the number of VMs considered. This improvement can be obtained without affecting the precision of the learned BN. In future work we plan to further increase the scalability by classifying variables in the system through an ontology. This knowledge can be used to predict possible relations among similar variables on different machines. This temporary knowledge can be used when a new VM is added to the system until enough monitoring data to build the real network are collected.

References

1. Andersson, G., Ilic, M.D., Madani, V., Novosel, D.: Network Systems Engineering for Meeting the Energy and Environmental Dream. Proceedings of the IEEE 99(1), 7–14 (2011)
2. Beloglazov, A., Buyya, R., Lee, Y.C., Zomaya, A.: A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems. Advances in Computers 82(2), 47–111 (2011)
3. Berl, A., Gelenbe, E., Girolamo, M.D., Giuliani, G., Meer, H.D., Dang, M.Q., Pentikousis, K.: Energy-efficient cloud computing. The Computer Journal 53(7), 1045 (2010)
4. Cappiello, C., Plebani, P., Vitali, M.: Energy-Aware Process Design Optimization. In: Proceedings of the 3rd Int. Conference on Cloud and Green Computing (2013)
5. Cappiello, C., Melià, P., Pernici, B., Plebani, P., Vitali, M.: Sustainable choices for cloud applications: a focus on CO2 emissions. In: ICT for Sustainability (2014)
6. Chen, D., Henis, E., Kat, R.I., Sotnikov, D., Cappiello, C., Ferreira, A.M., Pernici, B., Vitali, M., Jiang, T., Liu, J., Kipp, A.: Usage Centric Green Performance Indicators. SIGMETRICS Performance Evaluation Review 39(3), 92–96 (dec 2011)
7. Gelenbe, E., Lent, R.: Trade-offs between energy and quality of service. In: Sustainable Internet and ICT for Sustainability (SustainIT), 2012. pp. 1–5. IEEE (2012)
8. Kipp, A., Jiang, T., Fugini, M., Salomie, I.: Layered Green Performance Indicators. Future Generation Computer Systems 28(2), 478–489 (2012)
9. Mello Ferreira, A., Pernici, B.: Managing the Complex Data Center Environment: an Integrated Energy-Aware Framework. Computing 96(not assigned), 1–41 (2014)
10. OGrady, R., Christensen, A., Groß, R., Dorigo, M.: Self-organised computational structures for real time analysis in highly distributed environmental monitoring. In: IROS 2012 Workshop on Robotics for Environmental Monitoring (2012)
11. Tsamardinos, I., Brown, L.E., Aliferis, C.F.: The Max-Min Hill-Climbing Bayesian Network Structure Learning Algorithm. Machine Learning 65(1), 31–78 (2006)
12. Van Heddeghem, W., Lambert, S., Lannoo, B., Colle, D., Pickavet, M., Demeester, P.: Trends in worldwide ict electricity consumption from 2007 to 2012. Computer Communications 50, 64–76 (2014)
13. Viani, F., Giarola, E., Robol, F., Oliveri, G., Massa, A.: Distributed monitoring for energy consumption optimization in smart buildings. In: Antenna Measurements & Applications (CAMA), 2014 IEEE Conference on. pp. 1–3. IEEE (2014)
14. Vitali, M., Pernici, B.: A Survey on Energy Efficiency in Information Systems. International Journal of Cooperative Information Systems (IJCIS) 23(3), 1–38 (2014)
15. Vitali, M., Pernici, B., OReilly, U.M.: Learning a goal-oriented model for energy efficient adaptive applications in data centers. Information Sciences (IN PRESS)
16. Wajid, U., Pernici, B., Francis, G.: Energy Efficient and CO2 Aware Cloud Computing: Requirements and Case Study. In: Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on. pp. 121–126 (2013)
17. Watson, R.T., Boudreau, M.C., Chen, A.J.: Information Systems and Environmentally Sustainable Development: Energy Informatics and New Directions for the IS Community. Management Information Systems Quarterly 34(1), 23–38 (2010)
18. Wu, G., Zhang, H., Qiu, M., Ming, Z., Li, J., Qin, X.: A decentralized approach for mining event correlations in distributed system monitoring. Journal of parallel and Distributed Computing 73(3), 330–340 (2013)