

Towards Designing an Energy Aware Data Replication Strategy for Cloud Systems Using Reinforcement Learning

Amir Najjar, Riad Mokadem, Jean-Marc Pierson

Université de Toulouse

Institut de Recherche en Informatique de Toulouse - Campus Rangueil - 118 Route de Narbonne

31062 Toulouse - France

{amir.najjar, riad.mokadem, jean-marc.pierson}@irit.fr

Abstract

The rapid growth of global data volumes has created a demand for scalable distributed systems that can maintain a high quality of service. Data replication is a widely used technique that provides fault tolerance, improved performance and higher availability. Traditional implementations often rely on threshold-based activation mechanisms, which can vary depending on workload changes and system architecture. System administrators typically bear the responsibility of adjusting these thresholds. To address this challenge, reinforcement learning can be used to dynamically adapt to workload changes and different architectures. In this paper, we propose a novel data replication strategy for cloud systems that employs reinforcement learning to automatically learn system characteristics and adapt to workload changes. The strategy's aim is to provide satisfactory Quality of Service while optimizing a trade-off between provider profit and environmental impact. We present the architecture behind our solution and describe the reinforcement learning model by defining the states, actions and rewards.

Keywords : Cloud Systems, Data Replication, Reinforcement Learning, Economic, Energy Consumption.

1. Introduction

The growth of social media has generated large data volumes in the cloud. As such, assuring satisfactory Quality of Service (QoS) has become crucial. QoS can be measured using various metrics such as response time (RT), throughput, availability and fault tolerance [1]. Data replication provides improved RT, fault tolerance and availability through efficient placement of the replicas [8].

Previously proposed data replication strategies often utilize thresholds that require prior knowledge of the system and human intervention [11, 2, 28]. Statistical methods have been used to predict those thresholds automatically : Khatua et al. [9] use time series, Calheiros et al. [6] use an autoregressive integrated moving average (ARIMA), and Séguéla et al [27] use control charts to define such thresholds.

Machine learning (ML) is a promising technique for automatically adapting to system characteristics and adjusting the thresholds without prior knowledge of the underlying architecture or workload. This eliminates the need for human intervention. By using workload traces and

environmental state, an ML model can be trained to predict the necessary thresholds for data replication strategies. Additionally, it can use current resource utilization to predict future system performance, aiding in decisions related to activating the data replication mechanism.

Machine learning has been increasingly applied to data replication through supervised learning [5, 23], unsupervised learning [22, 26], and reinforcement learning [13, 30]. Among these methods, reinforcement learning appears to be the most suitable, as it can learn directly from a simulated or real environment. In contrast, supervised learning relies on labeled datasets, and unsupervised learning requires underlying patterns within the environment [18].

Current strategies, particularly those incorporating machine learning [18], often overlook the economic aspect, which is essential for ensuring profitability in commercial applications such as the cloud [17]. Additionally, they fail to sufficiently address the environmental impact of distributed systems [12]. In this paper, we introduce a reinforcement learning-based data replication strategy for cloud systems that balances provider profit with environmental impact in terms of energy consumption and purchase of machines. The initial replication aims to ensure availability and fault tolerance, while the dynamic replication ensures satisfactory response time while optimizing a trade-off between provider profit and energy consumption. VM aggregation is utilized to minimize purchase of additional machines.

The rest of the paper is organized as follows : Section 2 introduces data replication and reinforcement learning. Section 3 provides an overview of the considered architecture and its specifications. It defines the necessary terms to introduce our proposed strategy, which is presented in Section 4. We describe the reinforcement learning technique by defining the state space, the action space, and the reward signal. Section 5 concludes and presents the next steps required to implement the strategy as well as future plans.

2. Background

In this section, we provide an overview of data replication and reinforcement learning.

2.1. Data Replication

In the context of distributed systems, data replication is the process of creating copies of data in different locations to provide fault tolerance [20], increased availability [24], and higher performance [14]. A data replication strategy is required to ensure efficiency of the replication mechanism and to prevent excessive replication. Data replication generally aims to answer the following questions [17] :

- Which data to replicate ?
- What is the activation condition for a replication to occur ?
- How many replicas (replication factor) are required ?
- What are the placement locations ?
- How to minimize the economic cost of replication ? This question is particularly relevant for strategies designed for cloud systems.

2.2. Reinforcement Learning

Machine learning utilizes samples or experience for inference. It can be divided into three methods :

- Supervised learning : Characterized by labelled data. The goal is to minimize the loss between the predicted outcome and the true labels. Common tasks are regression and classification.
- Unsupervised learning : Characterized by unlabelled data. Unsupervised learning is

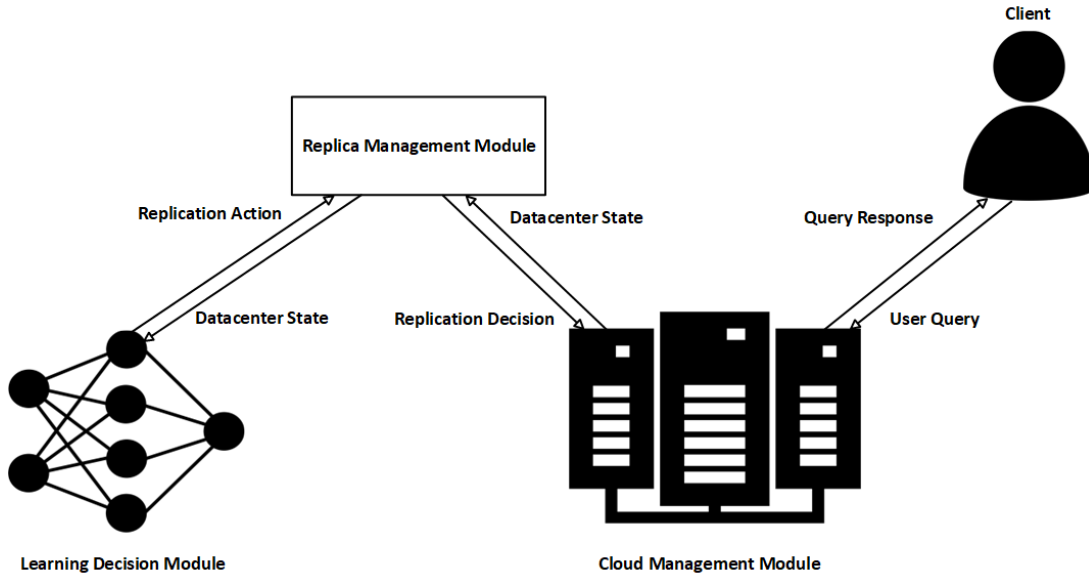


FIGURE 1 – Visual representation of the architecture

used to separate the data into groups, to reduce the dimensionality of the data or to learn the relationships in the data.

- Reinforcement learning : One or more agents interact with an environment through an action policy. A feedback mechanism known as a reward signal is provided through the agent's exploration of the action space instead of an explicit dataset or labels.

Reinforcement learning can be divided into model-based and model-free methods, based on whether the model is known [16]. Model-free methods can be used when modeling the environment is challenging. Q-Learning is one such method that utilizes the properties of Markov Decisions Processes (MDP) [19, 3] to estimate the reward given a certain state-action pair [25]. With the recent advancements in deep learning [4, 10], its use has been explored in reinforcement learning. Deep Q-Learning [15] is a method that employs neural networks to solve the core limitations of Q-Learning : Exploration space explosion and the tendency to overestimate the value of actions.

3. Architecture

The architecture is divided into three main components : The Replica Management Module (RMM), the Learning Decision Module (LDM) and the Cloud Manager Module (CMM). An overarching design of our platform is presented in Figure 1.

3.1. Overview

The architecture for our solution is comprised of three main components :

- Replica Management Module (RMM) : Serves as an intermediary between the LDM and CMM. Replication actions are relayed from the LDM. They are checked for validity and sent as replication decisions to the CMM. Updated system state is relayed from the CMM. It is then batched and sent to the LDM.
- Learning Decision Module (LDM) : Using updates in the system state, a Deep Q-learning model takes a replication decision and relays it to the RMM.

- Cloud Manager Module (CMM) : relays static and dynamic information to the RMM. Static information includes datacenter specifications and economic costs (for example, in USD). Dynamic information includes current network and resource utilization, as well as performance metrics such as response time.

3.2. Specification

The cloud system services a set of L clients $U = \{u_1, \dots, u_L\}$ and consists of a set of N datacenters $DC = \{dc_1, \dots, dc_N\}$. Each datacenter dc_i , ($1 \leq dc_i \leq N$) is characterized by a carbon intensity factor of $CI = [ci_i]$. Each client u_l , ($1 \leq l \leq L$) has a certain latency to each of the datacenters $LAT_l = \{lat_{l1}, \dots, lat_{lN}\}$. Each datacenter has a set of M_i homogeneous hosts $H_i = \{h_{ij}\}$, $1 \leq j \leq M_i$. In case a datacenter has heterogeneous hosts, it can be considered as multiple homogeneous datacenters sharing a common backbone. Hosts of dc_i are characterized by a CPU with $C = [c_i]$ cores, a core frequency of $F = [f_i]$, and economic cost per execution cycle of $CC = [cc_i]$. Each host has $S = [s_i]$ bytes of storage capacity and a storage cost of $SC = [sc_i]$ per byte. Several VMs can be deployed on a host depending on the core count and the required resources. A VM on h_{ij} is denoted by v_{ijk} . A VM v_{ijk} can occupy one or more cores of the host h_{ij} . The number of cores occupied by v_{ijk} is referred to by nco_{ijk} .

Network links are established between all pairs of datacenters : for dc_i and $dc_{i'}$ $i \neq i'$, $b_{ii',t}$ denotes the bandwidth available at time t , and $bc_{ii'}$ denotes the bandwidth economic cost per byte. Similarly, $dc_b_{i,t}$ denotes the bandwidth available at time t for the backbone network of dc_i , and dc_bc_i denotes the bandwidth economic cost per byte. It is possible to represent the network properties in four matrices, $B_t = [b_{ii',t}]$, $BC = [bc_{ii'}]$, $DC_B_t = [dc_b_{i,t}]$, $DC_BC = [dc_bc_i]$, which will be relevant for the solution design. Figure 2 presents an example of a platform with two datacenters. dc_1 has two hosts, each of which has two VMs. dc_2 has two hosts with one VM each.

A Service Level Agreement (SLA) is established between the provider and a client u_l . The client pays a fixed rate per query $RATE_l$. Multiple Service Level Objectives (SLO) are defined :

- Availability objective AVO_l : Minimum replication factor to respect at all times,
- Response time objective RTO_l : Maximum allowed response time before incurring a penalty RT_PEN_l .

For each datum d_l of size sz_l associated to user u_l , we retain visibility of the replication factor in each datacenter $R_l = [r_{l1}, \dots, r_{lN}]$ as well as the total number of replicas across the platform $GR_l = \sum_{i=1}^N r_{li} \geq AVO_l$.

A client u_l communicates with the RMM to request d_l . The query arrival rate follows a Poisson process [7, 29]. A query Q at time t is characterized by an arrival time $t(Q)$ and an amount of execution cycles $ec(Q)$. At time t , we also calculate the average CPU load percentage of hosts containing d_l per datacenter $UTIL_{l,t} = [util_{l1,t}, \dots, util_{lN,t}]$. The definition of each term can be found in appendix A.

4. Proposed Strategy

To avoid the environmental impact incurred by purchasing more machines, we adopt VM aggregation; data is replicated on hosts that are already active to minimize the amount of active hosts. We describe the strategy's approach to initial placement as well as dynamic allocation.

4.1. Initial Replication

To satisfy the availability objective for each user, AVO_l replicas need to be created. The RMM places the initial copies in different datacenters to ensure fault tolerance. The initial datum is

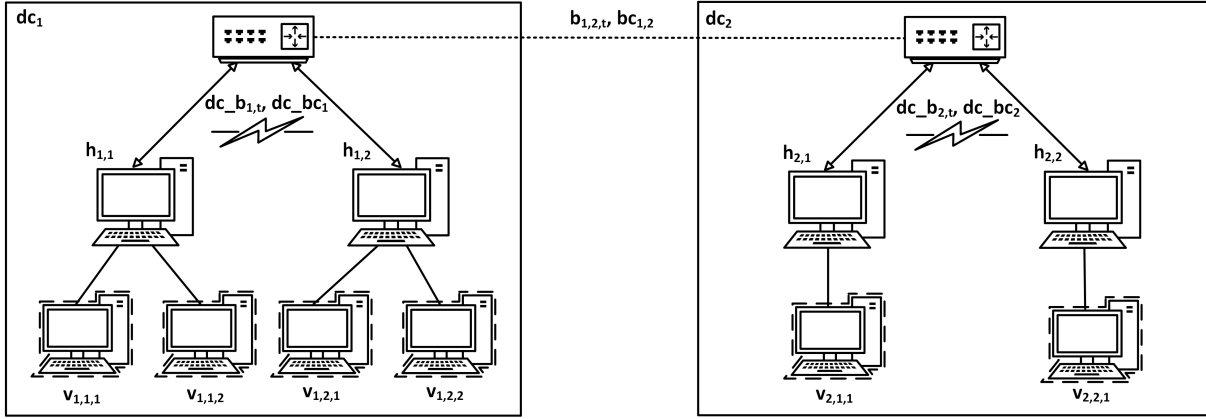


FIGURE 2 – Architecture example

placed in the datacenter with the lowest latency to the user to ensure fast response time. The rest of the copies are placed in the datacenters with the lowest carbon intensity ci_i . Response time is not as crucial for the remaining copies as they are used as backup in case a fault related to the VM containing the initial datum occurs.

4.2. Dynamic Replication

The LDM uses a Deep Q-Learning neural network to take actions based on a state. The state is encoded in the input layer and the value of actions is encoded in the output layer. The state consists of :

- Datacenter information : F core frequencies, C core counts, CC costs per execution cycle, SC storage costs, carbon intensities CI ,
- Network information : B_t inter-DC bandwidth available at time t , BC inter-DC bandwidth costs, DC_B_t intra-DC bandwidth available at time t , DC_BC intra-DC bandwidth costs,
- User information : LAT_l latencies, $UTIL_{l,t}$ average utilization, sz_l data size, RTO_l RT objective.

A preliminary check is performed before providing the state space. For example, in case of insufficient cores or disk space in a certain data center, the frequency of said data center is set to 0 in the state space. Furthermore, replication action is verified for validity and transformed into a replication decision. State-action-reward tuples are batched periodically. However, upon incurring penalties, batching is invoked and a replication decision is taken to remedy the response time loss.

The action space consists of :

- replicating in a datacenter dc_i ,
- not performing any replication

In case of an invalid action, the action with the next best value is taken into consideration until a valid action is reached.

The reward signal is constructed from economic profit and a penalty on energy consumption.

$$\text{Reward}(Q) = \alpha \cdot \text{Economic}(Q) - \beta \cdot \text{Energy}(Q) \quad (1)$$

Where $\alpha, \beta \geq 0$ are parameters set by the provider based on how much they're willing to compromise profit for saving energy. The economic part is the difference between the rate per

user query and possible penalty cost in addition to the cost of utilized resources which are comprised of the CPU cost, the storage cost, bandwidth cost, and the replication cost. It is defined as follows :

$$\text{Economic}(Q) = \text{RATE}_l - (\text{CPU}(Q) + \text{STOR}(Q) + \text{BW}(Q) + \text{PEN}(Q)) \quad (2)$$

The CPU cost can be calculated using the execution cycles $ec(Q)$, the core count nco_{ijk} and the cost per execution cycle cc_i . For simplicity, we assume that task performance scales linearly with core count :

$$\text{CPU}(Q) = \frac{ec(Q)}{nco_{ijk}} \cdot cc_i \cdot nco_{ijk} = ec(Q) \cdot cc_i \quad (3)$$

The storage is the size of the datum sz_l multiplied by the sum of storage costs incurred in each datacenter, which can be calculated using the product of the replication factor r_{li} and the storage cost sc_i :

$$\text{STOR}(Q) = sz_l \cdot \sum_{i=1}^N sc_i \cdot r_{li} \quad (4)$$

Replication can occur between two hosts in the same datacenter ($h_{ij} \rightarrow h_{ij'}, j \neq j'$) or between two hosts in different datacenters ($h_{ij} \rightarrow v_{i'j'}, i \neq i'$), the bandwidth cost associated is calculated accordingly :

$$\text{BW}(Q) = \begin{cases} sz_l \cdot dc_bc_i & \text{in same DC} \\ sz_l \cdot bc_{ii'} & \text{in different DCs} \\ 0 & \text{no replication} \end{cases} \quad (5)$$

The final part of equation 2 is the penalty RT_PEN_l incurred if RTO_l is not respected :

$$\text{PEN}(Q) = \begin{cases} \text{RT_PEN}_l & \text{RT}(Q) > \text{RTO}_l \\ 0 & \text{RT}(Q) \leq \text{RTO}_l \end{cases} \quad (6)$$

The energy part of the reward signal in equation 1 is defined as follows :

$$\text{Energy}(Q) = p_{ijk,F(Q)} - p_{ijk,t(Q)} + p_{\text{REPL},Q} \quad (7)$$

Where :

- $t(Q), F(Q)$ are the arrival time and the finish time of query Q respectively
- $p_{ijk,t}$ is the power consumption of v_{ijk} executing Q at time t
- $p_{\text{REPL},Q}$ is the energy consumed to create a replica if a replication occurred

5. Conclusion and Future Work

We have explored the possibility of designing a data replication strategy based on reinforcement learning in order to respond to the general lack in the literature for strategies that take into account the economic and environmental aspect. Reinforcement learning also allows us to eliminate human intervention commonly required in strategy design. We have presented the architecture behind our solution and defined the solution space as well as the reward signal.

Our next steps are : designing a suitable neural network layout used, implementing further optimizations for Deep Q-Learning such as Experience Replay [21] and normalizing the components of the reward signal to facilitate the choice of the weighted parameters for the provider. In addition, we plan to consider the energy consumption of the learning decision module and the cost of its usage. Further in the future, we plan to test our strategy in both simulated environments and testbeds. In addition, we plan to compare against other state of the art strategies, notably ones that utilize machine learning in their design.

References

1. Armstrong (D.) et Djemame (K.). – Towards quality of service in the cloud. – In *Proc. of the 25th UK Performance Engineering Workshop*, 2009.
2. Bai (X.), Jin (H.), Liao (X.), Shi (X.) et Shao (Z.). – RTRM : A response time-based replica management strategy for cloud storage system. – In *Grid and Pervasive Computing : 8th International Conference, GPC 2013 and Colocated Workshops, Seoul, Korea, May 9-11, 2013. Proceedings 8*, pp. 124–133. Springer, 2013.
3. Bellman (R.). – On the theory of dynamic programming. *Proceedings of the national Academy of Sciences*, vol. 38, n8, 1952, pp. 716–719.
4. Bengio (Y.), Goodfellow (I.) et Courville (A.). – Deep learning (Vol. 1), 2017.
5. Bui (D.-M.), Hussain (S.), Huh (E.-N.) et Lee (S.). – Adaptive Replication Management in HDFS Based on Supervised Learning. *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, n6, 2016, pp. 1369–1382.
6. Calheiros (R. N.), Masoumi (E.), Ranjan (R.) et Buyya (R.). – Workload Prediction Using ARIMA Model and Its Impact on Cloud Applications' QoS. *IEEE Transactions on Cloud Computing*, vol. 3, n4, 2015, pp. 449–458.
7. Cao (J.), Cleveland (W. S.), Lin (D.) et Sun (D. X.). – Internet traffic tends toward Poisson and independent as the load increases. *Nonlinear estimation and classification*, 2003, pp. 83–109.
8. Goel (S.) et Buyya (R.). – Data replication strategies in wide-area distributed systems. In : *Enterprise service computing : from concept to deployment*, pp. 211–241. – IGI Global, 2007.
9. Khatua (S.), Ghosh (A.) et Mukherjee (N.). – Optimizing the utilization of virtual resources in Cloud environment. – In *2010 IEEE International Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems*, pp. 82–87, 2010.
10. LeCun (Y.), Bengio (Y.) et Hinton (G.). – Deep learning. *nature*, vol. 521, n7553, 2015, pp. 436–444.
11. Lee (M.-C.), Leu (F.-Y.) et Chen (Y.-p.). – PFRF : An adaptive data replication algorithm based on star-topology data grids. *Future generation computer systems*, vol. 28, n7, 2012, pp. 1045–1057.
12. Lindberg (P.), Leingang (J.), Lysaker (D.), Khan (S. U.) et Li (J.). – Comparison and analysis of eight scheduling heuristics for the optimization of energy consumption and makespan in large-scale distributed systems. *The Journal of Supercomputing*, vol. 59, 2012, pp. 323–360.
13. Lu (K.), Zhao (N.), Wan (J.), Fei (C.), Zhao (W.) et Deng (T.). – RLRLP : High-Efficient Data Placement with Reinforcement Learning for Modern Distributed Storage Systems. – In *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 595–605, 2022.
14. Mansouri (N.). – Adaptive data replication strategy in cloud computing for performance improvement. *Frontiers of Computer Science*, vol. 10, 2016, pp. 925–935.
15. Mnih (V.), Kavukcuoglu (K.), Silver (D.), Rusu (A. A.), Veness (J.), Bellemare (M. G.), Graves

- (A.), Riedmiller (M.), Fiedjeland (A. K.), Ostrovski (G.) et al. – Human-level control through deep reinforcement learning. *nature*, vol. 518, n7540, 2015, pp. 529–533.
16. Moerland (T. M.), Broekens (J.), Plaat (A.), Jonker (C. M.) et al. – Model-based reinforcement learning : A survey. *Foundations and Trends® in Machine Learning*, vol. 16, n1, 2023, pp. 1–118.
 17. Mokadem (R.) et Hameurlain (A.). – A data replication strategy with tenant performance and provider economic profit guarantees in Cloud data centers. *Journal of Systems and Software*, vol. 159, 2020, p. 110447.
 18. Najjar (A.), Mokadem (R.) et Pierson (J.-M.). – A Review of Data Placement and Replication Strategies Based on Machine Learning. – In *2024 IEEE 30th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 278–285. IEEE, 2024.
 19. Puterman (M. L.). – *Markov decision processes : discrete stochastic dynamic programming*. – John Wiley & Sons, 2014.
 20. Qu (Y.) et Xiong (N.). – RFH : A resilient, fault-tolerant and high-efficient replication algorithm for distributed cloud storage. – In *2012 41st International Conference on Parallel Processing*, pp. 520–529. IEEE, 2012.
 21. Schaul (T.), Quan (J.), Antonoglou (I.) et Silver (D.). – Prioritized experience replay. *arXiv preprint arXiv :1511.05952*, 2015.
 22. Sellami (M.), Mezni (H.), Hacid (M. S.) et Gammoudi (M. M.). – Clustering-based data placement in cloud computing : a predictive approach. *Cluster Computing*, vol. 24, n4, 2021, pp. 3311–3336.
 23. Shwe (T.) et Aritsugi (M.). – Proactive Re-replication Strategy in HDFS based Cloud Data Center. – In *Proceedings of The10th International Conference on Utility and Cloud Computing, UCC '17, UCC '17*, p. 121–130, New York, NY, USA, 2017. Association for Computing Machinery.
 24. Sun (D.-W.), Chang (G.-R.), Gao (S.), Jin (L.-Z.) et Wang (X.-W.). – Modeling a dynamic data replication strategy to increase system availability in cloud computing environments. *Journal of computer science and technology*, vol. 27, n2, 2012, pp. 256–272.
 25. Sutton (R. S.) et Barto (A. G.). – *Reinforcement learning : An introduction*. – MIT press, 2018.
 26. Symvoulidis (C.), Kiourtis (A.), Marinos (G.), Totow Tom-Ata (J.-D.), Manias (G.), Mavrogiorgou (A.) et Kyriazis (D.). – A User Mobility-Based Data Placement Strategy in a Hybrid Cloud/Edge Environment Using a Causal-Aware Deep Learning Network. *IEEE Transactions on Computers*, vol. 72, n12, 2023, pp. 3603–3616.
 27. Séguéla (M.), Mokadem (R.) et Pierson (J.-M.). – Dynamic Energy and Expenditure Aware Data Replication Strategy. – In *2022 IEEE 15th International Conference on Cloud Computing (CLOUD)*, pp. 97–102, 2022.
 28. Tos (U.), Mokadem (R.), Hameurlain (A.), Ayav (T.) et Bora (S.). – A performance and profit oriented data replication strategy for cloud systems. – In *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCOM/IoP/SmartWorld)*, pp. 780–787. Ieee, 2016.
 29. Xiong (P.), Chi (Y.), Zhu (S.), Moon (H. J.), Pu (C.) et Hacgümüş (H.). – SmartSLA : Cost-Sensitive Management of Virtualized Resources for CPU-Bound Database Services. *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, n5, 2015, pp. 1441–1451.
 30. Zhang (T.), Gupta (A.), Rodríguez (M. A. F.), Spjuth (O.), Hellander (A.) et Toor (S.). – Data management of scientific applications in a reinforcement learning-based hierarchical storage system. *Expert Systems with Applications*, vol. 237, 2024, p. 121443.

Appendix A

Term	Definition
L	Number of clients
U, u_l	Client
d_l	Datum
N	Number of datacenters
DC, dc_i	Datacenter
CI, ci_i	Carbon intensity
LAT_l, lat_{li}	User latency
M_i	Number of hosts of dc_i
H_i, h_{ij}	Host
C, c_i	CPU core count
CC, cc_i	CPU cost per unit of time
S, s_i	Storage capacity
SC, sc_i	Storage cost per byte
v_{ijk}	VM
nco_{ijk}	Number of cores occupied by VM
$B_t, b_{ii'}, t$	Inter-DC bandwidth
$BC, bc_{ii'}$	Inter-DC bandwidth cost per byte
$DC_B_t, dc_b_{i,t}$	Intra-DC bandwidth
DC_BC, dc_bc_{ij}	Intra-DC bandwidth cost per byte
$RATE_l$	Rate per query
AVO_l	Availability objective
RTO_l	Response time objective
RT_PEN_l	Penalty incurred upon failing to meet RTO_l
sz_l	Size of d_l in bytes
R_l, r_{li}	Replication factors
GR_l	Global replication factor of d_l
Q	Query
$t(Q)$	Time of arrival
$ec(Q)$	Amount of execution cycles
$UTIL_{l,t}, util_{li,t}$	Average utility in dc_i

TABLE 1 – Terms and definitions