Energy and Expenditure Aware Data Replication Strategy

Morgan Séguéla, Riad Mokadem, Jean-Marc Pierson

Institut de Recherche en Informatique
de Toulouse

UMR CNRS 5505

Energy and Expenditure Aware Data Replication Strategy

Version 5

Morgan Séguéla — Riad Mokadem — Jean-Marc Pierson

Research report n° IRIT/RR–2021–07–FR

DÉPARTEMENT ASR – October 14, 2021

Institut de Recherche
en Informatique de Toulouse
CNRS - INP - UT3 - UT1 - UT2J

Energy and Expenditure Aware Data Replication Strategy

Version 5

Morgan Séguéla , Riad Mokadem , Jean-Marc Pierson

Département ASR

Research report no IRIT/RR–2021–07–FR October 14, 2021 (21 pages)

Abstract: Energy saving is a major challenge for Information Technology (IT) companies that aim to reduce their carbon footprint while providing large scale cloud services. These companies often rely on data replication technique in order to satisfy tenant's objectives, e.g., performance, especially with the increasing volume of data distributed throughout the world. In this paper, we propose a static and multi objective data replication strategy (E2ARS) that aims to reduce both energy consumption and expenditure of the provider. E2ARS leverages on cloud heterogeneity and energy efficient technologies. We first compare different policies of our strategy, from only taking energy consumption into account to only taking expenditure into account. Unsurprisingly, the more you want to reduce the energy consumption, the less you replicate. Then, we compare E2ARS with strategies from the literature. E2ARS reduces both energy consumption and expenditure where those strategies satisfy only one of the two objectives.

Key-words: Cloud, Data replication, Provider expenditure, Energy consumption, SLA violation

IRIT, Université Toulouse 3 Paul Sabatier (UT3)
118 Route de Narbonne - F-31062 TOULOUSE Cedex 9 FRANCE
Tel: (33) 5 61 55 67 65 – mail: contact@irit.fr

Stratégie de Réplication de données qui prend en compte la consommation énergétique et la dépense

Version 5

Résumé : La consommation énergétique est un défi majeur pour les sociétés d'informatique qui cherchent à réduire leurs émissions de gaz à effet de serre tout en fournissant des services cloud à large échelle. Ces sociétés s'appuient souvent sur la réplication de données pour satisfaire les objectifs des locataires, comme la performance, surtout avec une augmentation du volume de données distribuées autour du monde. Dans ce rapport, nous proposons une stratégie statique de réplication de données qui est multiobjectif (E2ARS) qui cherche à diminuer à la fois la consommation énergétique et la dépense pour le fournisseur. E2ARS s'appuie sur l'hétérogénéité dans le cloud et des technologies d'efficacité énergétique. Dans un premier temps, nous comparerons les différentes politiques de notre stratégie, d'uniquement prendre en compte la consommation énergétique à uniquement prendre en compte la dépense. Les résultats montrent qu'au plus l'utilisateur veut réduire la consommation énergétique, moins la stratégie va répliquer. Puis, nous comparons E2ARS avec des stratégies venant de la littérature. E2ARS réduit la consommation énergétique et la dépense alors que d'autres stratégies réduisent uniquement l'un ou l'autre des objectifs.

Mots-clés : Cloud, Réplication de données, Dépense du fournisseur, Consommation énergétique, Violation de SLA

IRIT, Université Toulouse 3 Paul Sabatier (UT3)
118 Route de Narbonne - F-31062 TOULOUSE Cedex 9 FRANCE
Tel: (33) 5 61 55 67 65 – mail: contact@irit.fr

Energy and Expenditure Aware Data Replication Strategy

Morgan Séguéla , Riad Mokadem , Jean-Marc Pierson

October 14, 2021

Abstract

Energy saving is a major challenge for Information Technology (IT) companies that aim to reduce their carbon footprint while providing large scale cloud services. These companies often rely on data replication technique in order to satisfy tenant's objectives, e.g., performance, especially with the increasing volume of data distributed throughout the world. In this paper, we propose a static and multi objective data replication strategy (E2ARS) that aims to reduce both energy consumption and expenditure of the provider. E2ARS leverages on cloud heterogeneity and energy efficient technologies. We first compare different policies of our strategy, from only taking energy consumption into account to only taking expenditure into account. Unsurprisingly, the more you want to reduce the energy consumption, the less you replicate. Then, we compare E2ARS with strategies from the literature. E2ARS reduces both energy consumption and expenditure where those strategies satisfy only one of the two objectives.

1 Introduction

Since the spread of information technologies, data produced by humanity as a whole increased sharply. These data are accessed by people and companies, for entertainment and debates, to fuel economic and scientific productions. Data have to be accessed at anytime and from everywhere as some data are accessed a lot in a short period of time. This became more relevant during the pandemic lockdown as everyone had to stay home even for working, implying a high raise of data access for IT companies, and for entertainment providers [1].

One way to answer this increasing number of accesses is to replicate data. Data replication is a well studied technique for several years and contexts. It has been commonly used in traditional systems like parallel databases systems [2] and more recently Cloud systems [3] on which we are focusing in this paper. Data replication aims to satisfy different objectives, from performance and availability objectives [4] to reducing energy consumption [5] and, costs [6] such as bandwidth network consumption. In fact, most giant IT companies like Google, Amazon and Microsoft provide cloud services for economic profit while investing in more powerful and greener components.

Elastic resource management is critical to minimize operating cost while ensuring tenant objectives [7]. In fact, the provider automatically adapts its resources as closely as possible to the tenant needs. Then, tenants only pay for what they consume. This corresponds to a specific economic model called the Pay-as-you-go model. The price to rent resources is specified in an agreement [8] signed by the provider and the tenant. This agreement, called the Service Level Agreement (SLA), essentially contains Service Level Objectives (SLO) that the provider has to achieve otherwise there will be penalties, mostly applied by refunding a part of the rent to the tenant. These penalties can be seen as costs for the provider amongst different kind of expenditure. [9] studied ways to model costs as most of them are split into capital expense (CAPEX) and operational expense (OPEX). CAPEX includes data center and server

depreciation over a certain time-frame. OPEX includes all others costs like resource costs, salaries, maintenance and so on.

Reducing the carbon footprint is becoming a more and more important objective for companies. In this context, companies communicate on this objective and Greenpeace made a report to state their evolution in terms of energy consumption and production [10]. In order to achieve this objective, several studies have been done to reduce energy consumption [11] [12] [13]. These studies focus on technologies like sleep state which permits putting in sleep mode some servers to reduce their energy consumption [14], and techniques like consolidation that avoid over-provisioning and put useless resources in sleep mode for instance.

In this paper, we propose a static data replication strategy E2ARS that aims to reduce both expenditure and energy consumption of the provider while considering performance for tenants. For this aim, we introduce models to estimate energy consumption and expenditure, e.g., storage and replication. In this context, the heterogeneity of prices and components in different data centers are taken into account. Based on these models, we propose an optimization algorithm in order to reduce both energy consumption and expenditure, through a 2 steps decision process. The reduction of energy consumption is leveraged by technologies like sleep state and consolidation. Note that we consider read-only data that will not be updated.

We compared the proposed static strategy to other static and dynamic data replication strategies. Compared to static strategies, the proposed strategy E2ARS is able to significantly reduce both energy consumption and expenditure when others reduce only one side. On the other hand, despite E2ARS is a static strategy, its energy consumption and the number of violations are only slightly higher compared to dynamic strategies, depending on the workload.

The rest of this paper is organized as follows: we begin in Section II by a state of the art of data replication strategies that takes into account energy consumption and/or expenditure of the provider. Then, in Section III, we describe our data replication strategy. After that, in Section IV, we validate the proposed strategy through the analysis of experimental results. Finally, we conclude and draw some lines for future works.

2 State of the art

We focus on strategies that take into account the provider expenditures or its energy consumption or both. Dealing with strategies that consider economic issues, one of the first data replication strategies that take into account this issue in cloud systems is [15]. The replication is based on the failing probability of a node and a reliability requirement. This permits to reduce the storage cost by delaying the replication while holding the reliability requirement. The strategy [16] uses a multi-tier cloud to reduce the operating cost while keeping a high level of performance based on heterogeneity. It supposes that there is a top-tier data center that is very effective but very expensive and low-tier data centers that are cheaper but less reliable. In the strategy proposed by [6], replication is triggered by a SLO violation and is applied only if it makes profit for the provider. The profit estimation is done during the replication choice based on the provider's revenue minus the operating cost from the start of the tenant's rent. [17] proposed a resilient and cost-effective data replication strategy. Their idea is to classify data based on their popularity. The servers are classified into 2 groups, primary servers and backup ones. Each data item has 2 replicas in the primary group and a third replica in the backup. Based on their popularity, replicas are compressed in the backup group or not. [18] considers a workflow application model that contains information about which task needs which data. This permits to regroup data based on their dependencies, and know which data are requested too many times. If there is an overlap between the ones that are highly dependent, and the ones that are accessed a lot, they become eligible and are replicated if it reduces the cost compared

to doing nothing. [19] proposed a strategy that considers time frames and for each time frame estimates the benefit made from a new replica or data migration compared to letting data stored in their precedent location. They also take into account a minimum number of replicas based on a latency constraint such as all data center can request data with a maximum latency under a given threshold.

The other scope of this paper is the reduction of energy consumption. The authors in [20] proposed a static Multi-Objective Replica Management. The proposed strategy tries to answer availability, service time, load balancing, energy consumption and latency issues through an evolutionary algorithm where each objective is weighted. A dynamic strategy is proposed by [5], that aims to reduce both energy and bandwidth consumption. This strategy considers a 3-tier fat tree data center architecture with a database at each level and one across the internet. A replication is triggered when the number of data accesses is higher than a threshold and occurs only if the energy and the bandwidth consumed by the database at a lower level are lower than the ones consumed by the database at the current level. Another data replication proposed in [21] takes into account availability and performance issues through the energy consumption prism. For this aim, it classifies both data and servers, the first based on their popularity and the second based on their power consumption. Then, hot data are replicated in hot servers (that consumes more) and cold data are replicated in cold servers. [22] tries to reduce providers carbon footprint by reducing energy consumption and choosing the replication placement based on the energy mix of each data center. However, most of these strategies neglect the economic costs of the resources consumed by the provider.

Authors in [23] state that there are very few strategies that consider both expenditure and energy consumption. Among those strategies, [24] aims to reduce energy consumption in order to maximize the provider profit. The profit is estimated by the energy cost from an absence of replication minus the energy cost under this strategy. This strategy does not model all the other costs that can occur while transferring files for instance. To reduce both, [12] considers a linear combination of energy consumption and expenditure and then use a Particle Swarm Optimization mixed with a Tabu Search Algorithm between each iteration. However, this strategy cannot let the administrator choose its policy with a clear view, and does not allow a high range of choices. The proposed strategy in this paper aims, through a static replica placement, to reduce both energy consumption and costs to store, read and replicate data.

3 Data placement strategy

3.1 Notation

Let F be a set of files stored on a set of nodes N. Each file $f_i \in F, 1 \leq i \leq z$ has a size $s(f_i)$.

Each node $n_j \in N, 1 \leq j \leq m$ has a storage capacity cp_j.

ϕ is a matrix of size (z, m) and denotes the placement of the files on the nodes: $\phi(f_i, n_j)$ is equal to 1 if f_i is on n_j, and 0 otherwise.

E_a denotes the energy consumed to perform the action a, $a \in A$. Pr_b represents the price of a node action b, $b \in B$.

The sets of actions depend on the state of the node. The static state represents the node when it is up but inactive (*static*). The dynamic state contains actions such as reading a file from a node (*read*), writing a file to a node (*write*), transferring data between nodes through the network (*net*) and storing a file on a node (*storage*). A contains *read*, *write*, *net* actions of the dynamic state and also contains the *static* state. B contains only the *storage* and *net* actions.

Notation	Description
F	Set of files, $F = \{f_1, ..., f_i, ..., f_z\}$
N	Set of nodes, $N = \{n_1, ..., n_j, ..., n_m\}$
ϕ	Matrix of files F stored on node N
$s(f_i)$	Size of file i (MB)
cp_j	Storage capacity of node j
Pw_a	Energy consumption of node action (in Joules)
	$a \in \{read, write, net, static\}$
Pr_b	Price of a node action (in \$/MB or \$/MB per second)
	$b \in \{net, storage\}$
t	Renting time parameter (in seconds)
$nbRds$	Number of reads parameter

Table 1: Notation table

t is the time the user will rent the provider's nodes, and $nbRds$ is the number of reads the user will do. To summarize those parameters, it is possible to refer to Table 1.

In the following part, models used in our data placement strategy are described. At first, cost and energy consumption of the storage for a fixed number of seconds is detailed. Then, the writing process for each replica and the average energy consumption for a fixed number of reads are given.

3.2 Models and Objective functions

In this part, models are proposed to estimate the provider's expenditure and energy consumption used in the optimization algorithm. These models are divided into 3 kinds of events: replicating, storing, reading. The expenditure model used in the following estimations are based on the work of [6]. To estimate the energy consumption, we modeled each component that is used in the data management of a server. At first, disks energy consumption is modeled based on [25] that is also used in [26] to model Solid-State Drive. Then the memory is modeled, and more precisely the Random Access Memory, based on [27] which is regularly updated by technical reports of a founder like [28]. Finally, the network energy consumption is considered through the Network Interface Card for the nodes, based on [29], and switches, from [30]. In these models, processors are only considered through their idle energy consumption. However, it is worth to say that these models are only used to build the following strategy and are not meant to precisely describe reality. Note that they can be modified or updated as it will not impact the core of the proposed strategy.

3.2.1 Replication

First, the impact of replicating data from their original node to the chosen node is modeled. Let N_o be the set of nodes included in N which contains nodes that store the original file before the placement strategy occurs. n_{i_o} represents the original node where file f_i is stored. If the chosen node is the original node of the file, the energy consumption and the cost needed to replicate are equal to 0. Most of the time, the replication occurs on other nodes which implies costs for the network.

To model the energy consumption, transferring and writing energy consumption have to be considered. The writing part includes the following steps: writing the received file on the RAM,

and reading the file from the RAM to write it on the Disk. Note that we do not consider RDMA in this work.

Energy:

$$EC_{\text{replic}}(F, N, N_o, \phi) = \sum_{i=1}^{z} \sum_{j=1}^{m} \phi(f_i, n_j) * [E_{net}(n_{i_o}, n_j, f_i) + E_{write}(n_j, f_i)] \quad (1)$$

Replicating a file implies costs for provider which is mostly represented by the communication cost through the network. This cost is based on the size of the file $s(f_i)$ and the price per megabyte is represented by $Pr_{net}(n_{i_o}, n_j)$ and depends on the placement: if the files are in the same data center it will be cheaper than being in 2 different regions. The expenditure model to replicate files is considered as follow:

Cost:

$$Cost_{\text{replic}}(F, N, N_o, \phi) = \sum_{i=1}^{z} \sum_{j=1}^{m} \phi(f_i, n_j) * s(f_i) * Pr_{net}(n_{i_o}, n_j) \quad (2)$$

3.2.2 Storage

After writing all replicas, the impact of storing files for t seconds on nodes is modeled. We suppose here that nodes cannot be turned off if they store data, in order to retrieve requested data as quick as possible. Also, adding data to a node that already stores data does not increase its power consumption. This implies that storing a file on an empty node will consume more than placing it on a node that already store files. An empty node can be turned off and its power consumption will be equal to 0.

In the following model all files in F can be stored on all nodes in N based on ϕ which represents nodes that store files. $E_{static}(n_j, t)$ is the static energy consumption to keep n_j up for t seconds. To estimate this energy consumption we consider the summation of the static power consumption of each component multiplied by t seconds. The model is described as follows:

Energy:

$$EC_{\text{storage}}(F, N, t, \phi) = \sum_{j=1}^{m} E_{static}(n_j, t) * [1 - \prod_{i=1}^{z} (1 - \phi(f_i, n_j))] \quad (3)$$

On the other hand, storing files imply a cost for the provider. This cost depends on the size of the file $s(f_i)$ and on the price $Pr_{storage}(n_j)$ to store data on the chosen node and for t seconds. The model is considered as follows:

Cost:

$$Cost_{\text{storage}}(F, N, t, \phi) = \sum_{i=1}^{z} \sum_{j=1}^{m} \phi(f_i, n_j) * s(f_i) * Pr_{storage}(n_j) * t \quad (4)$$

Where $Pr_{storage}(n_j)$ is the storage price per second and per megabyte of n_j.

3.2.3 Read

Finally, the energy consumption and costs linked to the number of reads is modeled.

Let N_i be a set of nodes that contains f_i and $N_{\bar{i}}$ the set of nodes that do not contain this file. These sets are different for each file and are built based on $\phi(f_i, n_j)$. In fact, N_i is the set of nodes where $\phi(f_i, n_j) = 1$ for a specific file f_i, and $N_{\bar{i}}$ contains the rest. Each node in N_i is written as n_j with $1 \leq j \leq m_i$ and m_i being equal to the number of replicas of f_i. In $N_{\bar{i}}$, nodes are written $n_{j'}$ with $1 \leq j' \leq m_{\bar{i}}$. One last notation specific to this part is $shortest(n_j, n_{j'})$ which is equal to 1 if the requestor node $n_{j'}$ is the closest to the storage node n_j in terms of transfer time and 0 otherwise.

The energy consumption model for this event includes the energy consumption to read a file f_i from the closest node n_j to the node $n_{j'}$ that requested this file ($E_{read}(n_j, f_i)$), and the energy consumed by the network to transfer this file between those nodes ($E_{net}(n_j, n_{j'}, f_i)$). The energy consumption to read the file considers the following steps: reading from the disk to write on the RAM, and then reading from the RAM to get transferred through the network.

Energy:

$$EC_{\text{read}}(F, N, \phi) = \frac{1}{z} \sum_{i=1}^{z} [\sum_{j=1}^{m_i} \frac{1}{m_{\bar{i}}} \sum_{j'=1}^{m_{\bar{i}}} shortest(n_j, n_{j'}) * [E_{net}(n_j, n_{j'}, f_i) + E_{read}(n_j, f_i)]] \quad (5)$$

$s.t.\ N_i \cup N_{\bar{i}} = N$

Like the writing part, the cost is mainly based on the cost to transfer data to the node $n_{j'}$ that requested the file f_i from the closest node n_j that stores this file. This cost is based on the size of f_i and the price to transfer this file $Pr_{net}(n_j, n_{j'})$ per megabyte.

Cost:

$$Cost_{\text{read}}(F, N, \phi) = \frac{1}{z} \sum_{i=1}^{z} [\sum_{j=1}^{m_i} \frac{1}{m_{\bar{i}}} \sum_{j'=1}^{m_{\bar{i}}} shortest(n_j, n_{j'}) * s(f_i) * Pr_{net}(n_j, n'_j)] \quad (6)$$

$s.t.\ N_i \cup N_{\bar{i}} = N$

3.2.4 Multi-Objective Function

In this last part, global models and the objective function used in the optimization algorithm are introduced. As it was detailed before, t represents the renting time of the user and $nbRds$ is the number of reads the user will do. These parameters are hypothesis made to balance the number of replicas. In fact, the lower the number of reads and the higher the renting time, the lower the number of replicas to reduce long term energy consumption and cost.

Based on previously defined functions (3), (1), (5), the global energy consumption model is described as follows:

Energy:

$$ECG(F, N, N_0, t, nbRds, \phi) = EC_{replic}(F, N, N_o, \phi) + EC_{storage}(F, N, t, \phi) + nbRds * EC_{read}(F, N, \phi) \quad (7)$$

The following global expenditure model is based on (4), (2), (6):

Cost:

$$ExpG(F, N, N_0, t, nbRds, \phi) = Cost_{replic}(F, N, N_o, \phi) + Cost_{storage}(F, N, t, \phi) + nbRds * Cost_{read}(F, N, \phi) \quad (8)$$

Let cp_j be the storage capacity of node n_j, the size of all files stored on this node cannot be higher than cp_j. This constraint is applied to nodes only, as it will be discussed later it does not occur if the node is considered as a representative of one data center. Based on [15], we choose to create a minimum of 2 replicas per file to assure a data availability within a year of more than 99.99%.

The following objective function is built upon 7 and 8. It searches the minimum value of both global models depending on ϕ.

$$\begin{cases} \min\limits_{\phi} \begin{pmatrix} ECG(F, N, N_0, t, nbRds, \phi), \\ ExpG(F, N, N_0, t, nbRds, \phi) \end{pmatrix} \\ \sum\limits_{i=1}^{z} * \phi(f_i, n_j) * s(f_i) \leq cp_j \quad , \forall j \\ \sum\limits_{j=1}^{z} \phi(f_i, n_j) \geq 3 \quad , \forall i \end{cases} \quad (9)$$

3.3 Energy and Expenditure Aware Strategy

We propose an *Energy and Expenditure Aware Replication Strategy* (*E2ARS*). This static data replication strategy can be considered as an initial data placement for a dynamic data replication strategy. The optimization algorithm used in this static data replication strategy is described below.

3.3.1 Overview on the static data replication algorithm

We consider a transnational cloud provider, like Amazon Web Services for instance, which provides services through different regions with several data centers in each region. All data centers have different prices and parameters. Nonetheless, we suppose that all nodes can store data, and for the sake of simplicity these nodes are homogeneous inside a data center. A multi-objective optimization algorithm is used to reduce both energy consumption and expenditure. However, including all nodes in the search space would make the processing very long. As the objective function considers the reading expenditure and energy consumption for each node between all other nodes, considering a subset of nodes permits to highly reduce the time to compute this part of the objective function. In order to make the algorithm works in a decent amount of time, a two steps decision process is proposed. The first optimization algorithm chooses on which data center replicas are stored. To do so, a node is chosen to represent the data center (referred to as getRepresentatives(N) in Algorithm 1). The second optimization algorithm chooses where the data will be stored inside the chosen data center. The whole process is given in algorithm 1.

3.3.2 Data replication between data centers (lines 1-5)

The first step is the most important because it has to choose between highly different data centers. It has to replicate and place data where the trade-off between expenditure and energy consumption is the most balanced from the database administrators (and all the hierarchy above them) perspective. To consider both objectives, we could use one objective as a constraint and minimize the other or use a weighted objective to mix them into one. However, we choose

Algorithm 1 Static Data Replication Algorithm

Input: N: List of nodes
Input: F: List of Files
Input: t: renting time
Input: NbRds: number of reads
Input: listOfDC: List of DC
Output: Final_Individual: matrix of files and nodes that represents replica placement

```
1: Final_Individual = zeros(F,N)
   // matrix (number of files, number of nodes) of zeros
2: DCrep = getRepresentativesDC(N)
   // retrieve one node for each data center
3: OriginFiles = getOriginalNodeFile(F, N)
   // nodes where files are stored at first
4: DC_individuals = getSPEA2Result(F, DCrep, OriginFiles, t, NbRds)
5: DC_individual = chooseIndividual(DC_individuals)
6: for all DC in listOfDC do
7:    tempNodesInDCList = getNodesFromDC(N, DC)
8:    tempFileInDCList = getFileOnDC(DC_individual, F)
9:    tempDCIndividual = FirstFitNoBottleneck(F, tempFileInDCList, tempNodesInDCList)
10:   Final_Individual[tempNodesInDCList] = tempDCIndividual
      // replace in the subset of nodes from the current DC in Final_Individual the result of
      FirstFitNoBottleneck
11: end for
```

to find the Pareto front to let the administrator choose if this is more important to reduce energy consumption, reduce expenditure, or make a more balanced choice. Thus, their choice could be supported by having an idea on their future costs and energy consumption. The Improved Strength Pareto Evolutionary Algorithm, also known as SPEA2 [31] (referred to as getSPEA2Result in algorithm 1), is efficient to find this Pareto front. SPEA2 returns a group of individuals from the Pareto front that are as far as possible from each other, the administrator can then choose the policy they want to apply (referred to as chooseIndividual in algorithm 1). This algorithm works with a population of individuals that can be dominated or not by other individuals, an archive that stores mostly non-dominated individuals. SPEA2 has 6 steps:

1. Generate an initial population
2. Calculate the fitness for each individual
3. Select individuals based on domination and truncation
4. End the algorithm if the maximum number of generations is reached
5. Selection with replacement within the archive to fill the mating pool
6. Recombination and mutation operators to the mating pool, go to step 2

Expenditure and energy consumption models are used as objective functions in the fitness calculation. For this first optimization algorithm, we suppose that each data center can store all the data. However, we are under the constraint of creating at least 2 replicas. [15] is a data replication strategy that tries to delay as much as possible replication in order to save costs, but it considers an availability issue. It highlights the fact that having independently stored

replicas increases the probability of not loosing data by node failure. Here, independently means reducing dependence of nodes where our data are stored. For instance, if 2 replicas are stored in the same data center and if there is an electrical shortage in this data center. Then, both replicas will not be available anymore. In a performance context, this will reduce a bottleneck induced by an overload of reading and replication requests.

3.3.3 Data placement inside chosen data centers (lines 6-10)

The second step chooses on which node a replica is stored inside each data center. Before processing this optimization algorithm, it has to gather nodes from the data center (referred to as getNodesFromDC in algorithm 1) and files that will be stored in the data center (getFileOnDC in algorithm 1). At first, we tried genetic and greedy algorithms. But, we found out that placing data on a very few nodes could highly reduce the energy consumption and there are not many things that could be done to reduce the expenditure. This kind of algorithm might not be optimal but the solution provided is enough considering the processing time. However, getting the minimum number of nodes storing data, would introduce a bottleneck in a performance context as a high amount of requests will be done on these nodes. Also, the more there is data on a node, the longer the response time would be if the workload is very high. To consider this issue, we choose to wake and store those replicas on a small proportion of nodes in the data center so that those nodes are not fully loaded. This small proportion of nodes includes nodes that store the original files if those files are planned to be replicated in this data center. Otherwise, these files will be deleted. Based on our experiments, we chose to use 6.25% of the data center nodes as it is a good trade off between performance and energy consumption. If this number of nodes cannot hold all the data, we increase this proportion by 6.25% steps.

The second optimization algorithm (Algorithm 2) takes as input, the set of files F, the set of files that will be stored in the chosen data center FilesInDC and the set of nodes in the current data center Nodes_DC. This algorithm at first sort chosen files by their size and return their id from the set F. The function used in line 3 is described in the algorithm 3. It permits to find the right number of nodes based on the 6.25% policy discussed before. Algorithm 3 starts by getting the size of all data that has to be stored (line 2) and verify if there is enough space for each increment by 6.25% of the number of woken up nodes. If this is the case, it returns this number of nodes. Then we place each file with a round-robin method between each node (lines 10-18). If the node cannot store the file due to its lack of capacity, it seeks for the next node (lines 21-29). If the file is not stored after trying all nodes chosen from the algorithm 3, it stores the file on another node, outside the ones considered by the algorithm 3. In fact, it bases its estimation of the number of nodes needed on the size of all the data that has to be stored and the capacity of these chosen nodes.

4 Experiments

4.1 Experimental Environment

4.1.1 Strategies Parameters

First, we highlight the differences between three kinds of policies based on our optimization algorithm. Those policies represent a range on our objective spectrum based on the Pareto front, from only considering energy consumption (*E2ARSEC*) to only considering expenditure (*E2ARSEX*). We also tried a policy that is more balanced toward energy consumption and expenditure which will be considered on the following experiments (*E2ARS*). To choose this balanced policy, we ordered all the proposed individuals by energy consumption, and we chose

Algorithm 2 FirstFitNoBottleneck

Input: F: List of all files
Input: FilesInDC: List of files that will be stored in the current DC
Input: Nodes_DC: List of nodes in DC
Output: DCIndividual : matrix of File stored on Node

```
1: DCIndividual = zeros(z, Node_DC)
   // matrix (z, nbNodesInDc) of zeros
2: orderedFileListId = sort(FilesInDC)
   // sort files based on their size and return their id
3: totalNbNode = getTotalNbNode(Node_DC, FilesInDC)
4: capacity = getAllCapacity(Node_DC)
   // get a list of capacity for all nodes in DC
5: for all idFile in ordderFilesListId do
6:    idNode = idFile % totalNbNode
      // First chosen node
7:    tempNode = Node_DC[idNode]
8:    fileStored = false
9:    nbNodeTested = 0
10:   repeat
11:      nbNodeTested += 1
12:      if capacity[tempNode] > s(F[idFile]) then
13:         DCIndividual[idFile][idNode] = 1
14:         fileStored = true
15:         capacity[tempNode] -= s(F[idFile])
16:      else
17:         idNode = (idFile + nbNodeTested) % totalNbNode
18:         tempNode = Node_DC[idNode]
19:      end if
20:   until fileStored ∨ nbNodeTested ≥ totalNbNode
21:   if ¬ fileStored then
22:      idNode = totalNbNode
23:      while ¬ fileStored do
24:         tempNode = Node_DC[idNode]
25:         if capacity(tempNode) > size(F[idFile]) then
26:            DCIndividual[idFile][idNode] = 1
27:            fileStored = true
28:         end if
29:         idNode += 1
30:      end while
31:   end if
32: end for
```

Algorithm 3 getTotalNbNode

Input: Nodes_DC: Set of nodes in DC
Input: FilesInDC: Set of files that will be stored in DC
Output: Number of Nodes

```
1: prop = 0.0625
2: totalFileSize = getTotalSize(FilesInDC)
   // return the size of all stored data in the chosen DC
3: allFileFit = false
4: repeat
5:    totalStorage = 0
6:    totalNbNodes = round(length(Node_DC) * prop)
7:    for i = 0 to totalNbNode do
8:       totalStorage += capacity(Node_DC[i])
9:    end for
10:   allFileFit = totalFileSize < totalStorage
11:   prop += 0.0625
12: until allFileFit
```

the middle one. In following experiments, t is set at 30 days, and $nbRds$ at 100. Then we compare our strategy with other data replication strategies proposed in the literature and control strategies. As control replication strategy, we used a strategy that places 3 replicas randomly called 3Rand. From the literature, we compare our strategy with a static replication strategy called MORM [20] that occurs before the experiment starts, with a multi-objective algorithm that considers availability, latency, load balancing, service time and energy consumption. We also compare our strategy with 2 dynamic replication strategies: (i) PEPR [6] and (ii) the one proposed by Boru et al. [5]. PEPR is a strategy that takes into account the provider profit (with an income of 0.0205$ per cloudlet) and performance. It is triggered by each violation, replicates if it is still profitable. Boru et al. considers a hierarchical topology with 3 level of databases (central, data center, rack). In this strategy, the replication is triggered by the number of reads reaching a threshold, and the replication occurs on lower level databases (data center, rack) if they consume less energy and bandwidth than the higher level databases (central, data center).

4.1.2 Simulation Parameters

In order to compare those strategies, we implemented them on CloudSim [32]. This simulator has been extended by [6] in order to take into account data replication while considering heterogeneous network bandwidth, and monetary cost of resources. Then, it has been extended to estimate the energy consumption [23]. Finally, we added the capacity for all nodes to be turned into sleep mode based on [14] in order to reduce the energy consumption. However, we have supposed that storing data block the node from sleeping to keep access to the data it stores. This implies some differences compared to results in [23] in terms of energy consumption.

Networks parameters come from different sources. Bandwidth is based on [5] and latency values are from Wikipedia[1]. Pricing characteristics are coming from Google Cloud Pricing, assuming a 20% margin. The response time threshold is coming from [33] where a 15 seconds wait without feedback implies a loss of 25% of users. This threshold corresponds to our SLO, and if the response time is higher than 15 seconds, we consider that this cloudlet violates the SLO implying a refund to the tenant. Experiment parameters are summarized in the Table

[1] https://wikitech.wikimedia.org/wiki/Network_design (08/28/2020)

2. We consider 2 kinds of experiments. Large scale ones with 1024 files with 128 nodes per data center, and smaller scale ones with 30 files and 32 nodes per data center. Smaller scale experiments are kept as MORM needed too much memory and could not be processed for larger scale ones. Then, when the number of nodes and files is higher, we also increased the number of cloudlets from 75,000 to 150,000.

In our experiments, we have to take care of the given architecture for each data replication strategy. In fact, 3Rand, E2ARS, MORM and PEPR are considered using an architecture following a peer to peer topology. In this case, all nodes can process tasks and store data. Boru et al.'s architecture is based on a three tier fat tree topology where each level has a database. In order to take its architecture into account inside a peer to peer one, we add a data center per region which corresponds to the data center database, and a region which represents the central database. However, these added structures are not taken into account to the global energy consumption, because those nodes can not process tasks, and only store data. This permits us to compare those strategies for a given amount of storage and processing resources.

Parameters	Values	Parameters	Values
Number of files	30 / 1,024	File size	[0.2, 5, 10] GB
Simulation Duration	6h	Number of cloudlets	75,000 / 150,000
Minimum cloudlet size	1,000 MI	Maximum cloudlet size	7,500 MI
Node processing capacity	1,600 MIPS	Number of Nodes per DC	32 / 128
Number of DCs per region	5	Number of region per Cloud	4
BW between regions	100 Gbit/s	Latency between regions	160 ms
BW within a region	10 Gbit/s	Latency within a region	30 ms
BW within a DC	10 Gbit/s	Latency within a DC	1 μs
Response time SLO	15s	Penalty Cost	0.00205$
Cloudlet execution cost	3.8-4.4*$10^{-9}$$/MI	Storage cost	2.2-7.7*$10^{-8}$$/GB
Transfer cost between regions	0.094$/GB	Transfer cost within a region	0.0078$/GB
Transfer cost within a DC	7.8*$10^{-4}$$/GB		

Table 2: Simulation parameters (Smaller scale / Larger scale)

For the workload, [34] shows that when there is a post on a social media with a link to Wikipedia, there is an increasing interest about the topic, increasing the number of views on the linked Wikipedia page. Then the interest fades away, decreasing the number of access to the topic page on Wikipedia. [35] also highlights the fact that this interest can fade at different speed. Based on this information, we simulate 2 different workload. The first one is represented by *experiment 1* and *experiment 3* for the smaller and larger scale ones respectively. The second workload is represented by *experiment 2* and *experiment 4*. These workload model a short interest for a content. To do so, we choose the gamma probabilistic distribution to model the arrival rate of requests with parameters $\alpha = 4$ and $\beta = 600$. The difference between the first workload and the second is that the increase comes at 4h and 1h30, respectively. All differences between experiments are summarized in Table 3. Also, experiments are processed 25 times as there is some randomness that may impact the results. Results are shown through means and standard deviations.

To compare these policies and strategies, we used 4 metrics: (i) the number of created replicas, (ii) the proportion of violations which represents the number of time the response time is higher than the SLO divided by the number of request. (iii) The energy consumed (in MJ) by the Cloud at the end of the experiment and (iv) the total cost for the provider (in $).

A GitHub repository is available here [2] to let everyone run their own experiments with their parameter on our environment.

[2] https://github.com/MorganSeguela/Cloud_2021_XPS

Experiments	1	2	3	4
Number of files	30	30	1024	1024
Number of cloudlets	75k	75k	150k	150k
Number of nodes	32	32	128	128
Times of increase	4h	1h30	4h	1h30

Table 3: Differences between experiments

4.2 Results

4.2.1 Comparison between different policies

First, we compare different policies from our static data replication strategy. As a reminder, we compare *E2ARSEX* and *E2ARSEC* which are policies that only take respectively expenditure and energy consumption exclusively. There is a 3rd policy that is a balance between those 2 objectives called E2ARS. The following figures and analysis are first based on experiment 3 (large scale experiment with an increase after 4h).

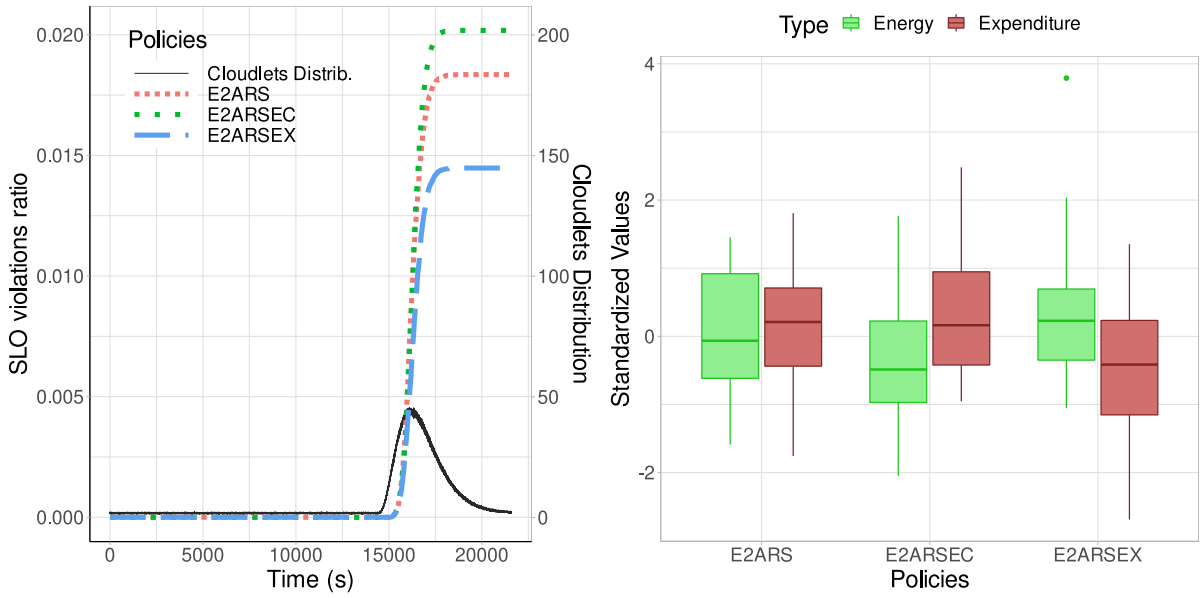
At first, it can be highlighted the fact that the more energy consumption is considered, the less the strategy creates replicas. In fact, E2ARSEC is the one having the lowest number of replicas with an average of 2915.1 ($\sigma = 13.12$) replicas, followed by E2ARS and E2ARSEX with an average of 2,923.2 ($\sigma = 9.61$) and 2,937.5 ($\sigma = 9.26$) replicas respectively. These differences have an impact on the number of violations. Figure 1a represents the cloudlet arrival rate of experiment 3 and the SLO violations ratio. This figure shows that violations are sorted according to the number of replicas which mean that E2ARSEC, with an average of 3,027.3 ($\sigma = 1,184.3$), has more violations than E2ARS, with an average of 2,752.4 ($\sigma = 1,634.6$). Then E2ARS has more violations than E2ARSEX with an average of 2,171.6 ($\sigma = 786.7$). Yet, only the difference between E2ARSEC and E2ARSEX are statistically different (based on Student tests to compare means).

Figure 1b shows the standardized values of energy consumption and expenditure for experiment 3. On this plot, E2ARSEC is the policy that consume less with an average of 595.4 MJ ($\sigma = 16.64$) compared to E2ARSEX which consumes 606.1 MJ ($\sigma = 17.3$) and E2ARS that consumes 601.7 MJ ($\sigma = 14.1$). E2ARSEX is the cheapest one with an average cost of 8,713.8$ ($\sigma = 234.6$) compared to E2ARS which costs 8,848.6$ ($\sigma = 215$) and E2ARSEC 8,867.3$ ($\sigma = 210.1$). The ordering of those policies still seems to make sense as they represent 3 different policies of our spectrum. E2ARS will then correspond to the selected strategy as it is in the middle of the spectrum. Just like the replication, E2ARSEC and E2ARSEX are significantly different for both energy consumption and expenditure.

However, it has to be highlighted that the number of files and nodes have an impact on these results. In fact, this has an impact on their range of possibility, implying that some results shows that E2ARS can be closer to E2ARSEX or E2ARSEC in terms of performance, energy consumption and expenditure.

4.2.2 Comparison between different strategies

To compare all the strategies, we analyze experiment 1 (small scale, increase after 4h), but other results will also be analyzed as they highlight some differences compared to this experiment. Figure 2a shows the number of replications during the experiment 1 when comparing our strategy with the static ones (MORM, 3Rand), and the dynamic ones (Boru et al., PEPR). Within the static strategies, 3Rand creates 2 replicas per files, MORM creates a lot of replicas (more than a thousand) and E2ARS created a bit less than 3 replicas per file. A first comparison we can focus on is the time needed to let MORM choose how many and where to place its replicas compared



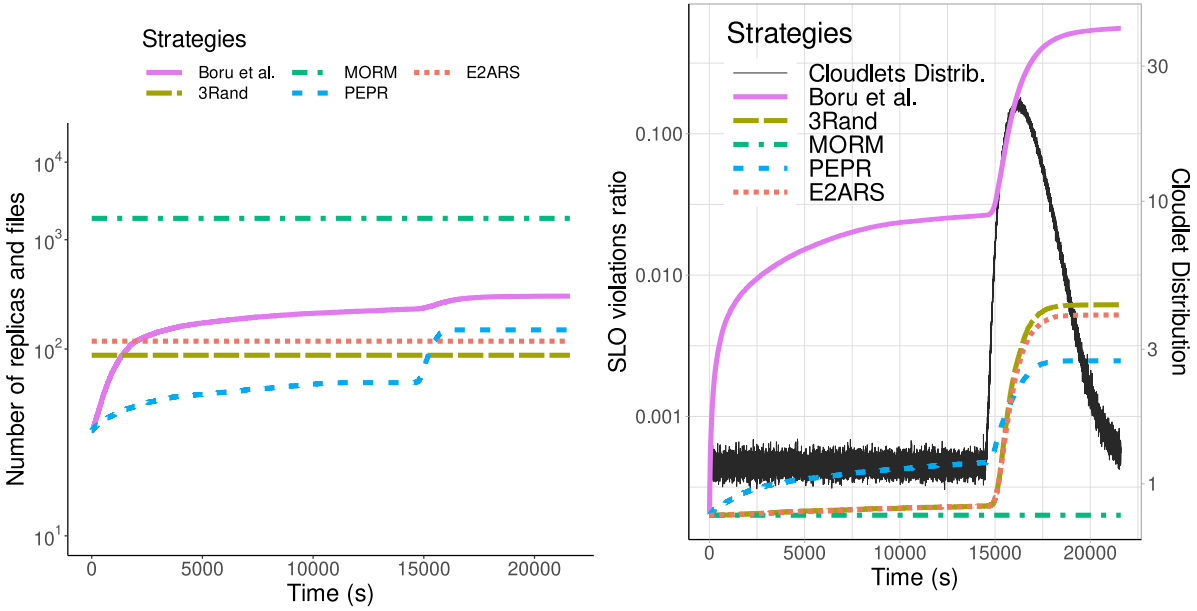
(a) Cloudlets distribution and Violations over execution between policies

(b) Standardized Value of energy consumption and expenditure between policies

Figure 1: Results from experiment 3

to E2ARS. In fact MORM takes an average of 1,164 seconds to compute its strategy while E2ARS takes around 9.6 seconds. In the dynamic group, PEPR starts without replicas, and it adapts dynamically based on violations and profit. Implying that the stop of the increasing number of replicas can be either due to the lack of violations or the lack of profit. Boru et al. starts without replicas either, as all data are stored in the central database, and then if the number of requests is higher than a threshold it triggers the replication, and it replicates if the energy consumption and the bandwidth of the level below is lower. This is why there is a sharp increase of data replication, then it gets slower, as long as most of the highly requested data in the central database are already replicated. However, on Figure 2b, we can see that even with the 2nd largest number of replicas, it as the highest number of violations (reaching up to 83% at the end. This is mostly due to the topology that makes a bottleneck where data are stored. It is followed by 3Rand and E2ARS which violated the SLO with an average of 700 times ($\sigma = 370$) corresponding to 0.9% of cloudlets and 510 times ($\sigma = 164$) which correponds to 0.7% of all cloudlets respectively. PEPR's number of violations follows the number of replicas created in this workload. In fact, as the number of violations began to remain constant, the number of replicas is also kept constant. Which means that the reason of stopping the creation of replicas in this workload is based on the number of violations. Finally, MORM is the strategy that has no violations at all as it created a huge amount of replicas. Table 4 and Table 5 represent the number of replicas created and the number of SLO violations respectively at the end of each workload.

Figure 3 corresponds to the standardized values of energy consumption and expenditure of experiment 1. We chose 3Rand as a control strategy, its average energy consumption is 228.47 MJ ($\sigma = 7.17$) and it costs 4,732$ ($\sigma = 304.89$). It is the baseline to which we compare others strategies. On this figure, MORM is the cheapest with a reduction of 87% of the cost. It is explained by the fact that it replicates a lot implying an absence of data transfer between regions which is very expensive, but this would be very expensive for a long term storage. But, this is the second most energy consuming strategy with an increase of 84% compared to 3Rand



(a) Replication over execution between strategies

(b) Cloudlets distribution and Violations over execution between strategies

Figure 2: Results from experiment 1

Experience	1	2	3	4
Boru et al.	241 (5.9)	240 (4.3)	126 (3.7)	127 (3.87)
3Rand	60 (0)	60 (0)	2,048 (0)	2,048 (0)
MORM	6.6k (3.2k)	6.3k (3.1k)	-	-
PEPR	108 (9.7)	159 (8.1)	184 (80.1)	2,075 (32.1)
E2ARS	85 (2.3)	85 (2.1)	2,923 (9.6)	2,927 (15.7)
E2ARSEC	83 (1.7)	84 (1.9)	2,915 (13)	2,914 (12)
E2ARSEX	86 (1.29)	86 (1.4)	2,937 (9.3)	2,933 (8.6)

Table 4: Number of replicas created per each experiment – Mean ($Standard - Deviation$)

Experience	1	2	3	4
Boru et al.	63k (451)	70k (190)	144k (160)	146k (171)
3Rand	700 (370)	610 (137)	55k (7.3k)	53k (6.9k)
MORM	0 (0)	0 (0)	-	-
PEPR	226 (34)	168 (10)	109k (5.8k)	2.1k (32)
E2ARS	510 (164)	483 (79)	2.8k (1635)	2.2k (807)
E2ARSEC	505 (118)	507 (106)	3k (1184)	3.7k (1477)
E2ARSEX	313 (74)	429 (92)	2.2k (787)	2k (781)

Table 5: Number of SLO violations per experiment – Mean ($Standard - Deviation$)

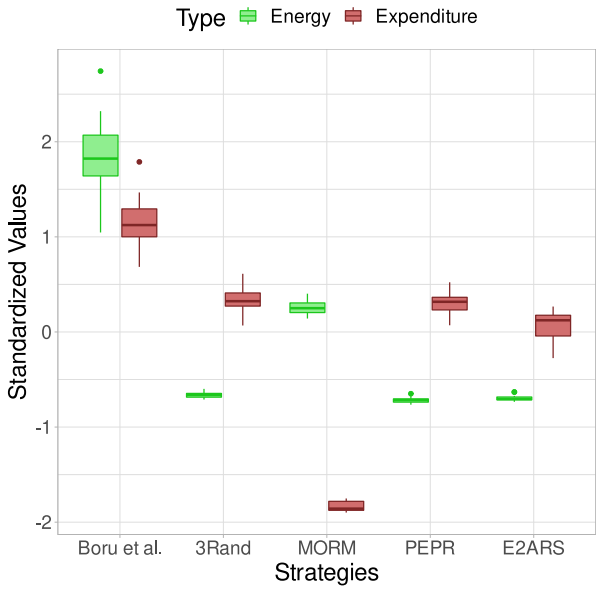


Figure 3: Standardized Value of energy consumption and expenditure of experiment 1 between strategies

as it needs to keep all nodes awaken while other strategies can put them in sleep mode. Boru et al. is the one that costs and consumes the most with expenditure increase of 33% and an energy consumption that is equal to more than 3 times 3Rand. This is mostly explained by the bottleneck created by the topology increasing the execution time, implying an increase in terms of energy consumption. Also, due to its topology, Boru et al. has to retrieve data from the central database, as it is considered as another region, it sharply increases the cost to transfer data. Compared to 3Rand, PEPR slightly reduces its energy consumption by 4% while keeping the same cost. E2ARS reduces significantly both energy consumption and expenditure by 3% and 10% respectively. As other experiments have been done on other workload, results are summarized on table 6 and 7 for energy consumption and expenditure respectively.

Based on all results (Table 4, Table 5, Table 6 and Table 7), experiments show differences in terms of results and behaviors. For instance, Boru et al. keeps a low number of replicas in larger scale experiments compared to smaller ones. This is mostly explained by the chosen threshold that has to be lowered as the number of file increases. However, we kept the same replication conditions between all experiments as it was for other strategies. It mostly has an impact on transfer time, implying an increase in terms of violations (up to 98% of all cloudlets), expenditure and energy consumption. PEPR handles the larger scale experiments and keeps reducing both expenditure and energy consumption in almost all experiments. Yet, in experiment 3 (large scale, increase after 4h), PEPR has a very low number of replicas linked with a high number of violations (73% of all cloudlets). This means that replications stopped due to the lack of profit as the expenditure went higher. Finally, E2ARS keep its performance in larger scale with a low number of violations (up to 1.8% in large scale) with a low processing time (around 9 seconds in small scale and 161 seconds for larger scale ones). Also, larger scale experiments highlight a higher reduction of both energy consumption and expenditure compared to 3Rand with a reduction of 36% and 18% respectively.

Experience	1	2	3	4
Boru et al.	747 (77)	722 (56)	8.9k (131)	8.8k (129)
3Rand	227 (6)	194 (1)	942 (37)	785 (5.54)
MORM	419 (14)	457 (9)	-	-
PEPR	216 (6)	192 (1)	896 (119)	498 (2.81)
E2ARS	220 (6)	187 (1)	602 (14)	503 (4.55)
E2ARSEC	220 (7)	187 (1)	595 (17)	501 (3.8)
E2ARSEX	228 (10)	188 (1)	606 (17)	505 (3.2)

Table 6: Energy consumption by experiment and policies in MJ – Mean ($Standard-Deviation$)

Experience	1	2	3	4
Boru et al.	6.3k (377)	6.4k (288)	40k (488)	40k (484)
3Rand	4.8k (305)	4.7k (195)	11k (230)	11k (262)
MORM	594 (102)	1.8k (1.1k)	-	-
PEPR	4.7k (350)	3.7k (230)	22k (2.1k)	6.8k (106)
E2ARS	4.3k (268)	4.1k (331)	8.8k (215)	8.8k (244)
E2ARSEC	4.3k (269)	4.2k (389)	8.8k (210)	8.9k (209)
E2ARSEX	3.8k (299)	3.7k (333)	8.7k (235)	8.7k (186)

Table 7: Expenditure by experiment and policies in $ – Mean ($Standard - Deviation$)

5 Conclusion

In this paper, we proposed a static data replication strategy called E2ARS that achieves its objective to reduce both energy consumption and expenditure compared to other strategies. The proposed strategy takes into account the trade-off between reducing energy consumption and reducing expenditure that is considered and decided by the administrator instead of finding an optimal solution for one issue or another. E2ARS uses a 2 steps decision process in order to reduce the search space. The first step uses an Evolutionary Algorithm named SPEA2 followed by a second step which is a heuristic that places data on a few amounts of nodes in order to leverage technologies like PowerSleep.

We compared our strategy alongside a control strategy that places 3 replicas placed randomly (3Rand). We also compared our strategy to other existing strategies proposed in the literature, both static (MORM) and dynamic (Boru et al., PEPR). Results show that E2ARS reduces the number of violations compared to 3Rand with only a few more replicas. Also, E2ARS fulfilled its objective to reduce both energy consumption and expenditure compared to the others strategies, that mostly reduced efficiently one objective. Compared to MORM, E2ARS significantly reduces both energy consumption and expenditure. On the other hand, despite E2ARS is a static strategy, its energy consumption and the number of violations are only slightly higher compared to the compared dynamic strategies. Furthermore, E2ARS leverages the cloud heterogeneity and technologies like PowerSleep while keeping a high level of performance without creating a large amount of replicas.

For future work, we will use the proposed static data replication strategy as an initial placement for a forthcoming dynamic data replication strategy. Also, it would be interesting to use some techniques like data compression in order to reduce storage cost and energy consumption. Another idea is to include the reduction of carbon footprint by knowing the energy mix of each data center. Finally, we are currently repeating these experiments on a real architecture.

References

[1] H. Reed, "Netflix Q1 Report 2020," Tech. Rep., Apr. 2020. [Online]. Available: https://www.netflixinvestor.com/financials/quarterly-earnings/default.aspx

[2] P. Valduriez, "Parallel database systems: Open problems and new issues," *Distributed and Parallel Databases*, vol. 1, no. 2, pp. 137–165, Apr. 1993.

[3] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, Jun. 2009.

[4] J. Janpet and Y.-F. Wen, "Reliable and Available Data Replication Planning for Cloud Storage," in *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*. Barcelona: IEEE, Mar. 2013, pp. 772–779. [Online]. Available: http://ieeexplore.ieee.org/document/6531832/

[5] D. Boru, D. Kliazovich, F. Granelli, P. Bouvry, and A. Y. Zomaya, "Energy-efficient data replication in cloud computing datacenters," *Cluster Computing*, vol. 18, no. 1, pp. 385–402, Mar. 2015.

[6] U. Tos, R. Mokadem, A. Hameurlain, T. Ayav, and S. Bora, "Ensuring performance and provider profit through data replication in cloud systems," *Cluster Computing*, Dec. 2017.

[7] N. R. Herbst, S. Kounev, and R. Reussner, "Elasticity in Cloud Computing: What It Is, and What It Is Not," *ICAC*, vol. 13, pp. 23–27, 2013.

[8] D. Serrano, S. Bouchenak, Y. Kouki, F. A. de Oliveira Jr., T. Ledoux, J. Lejeune, J. Sopena, L. Arantes, and P. Sens, "SLA guarantees for cloud services," *Future Generation Computer Systems*, vol. 54, pp. 233–246, Jan. 2016.

[9] L. A. Barroso, U. Hölzle, and P. Ranganathan, "The Datacenter as a Computer: Designing Warehouse-Scale Machines, Third Edition," *Synthesis Lectures on Computer Architecture*, vol. 13, no. 3, pp. i–189, Oct. 2018. [Online]. Available: https://www.morganclaypool.com/doi/abs/10.2200/S00874ED3V01Y201809CAC046

[10] G. Cook, J. Lee, T. Tsai, A. Kong, J. Deans, B. Johnson, and E. Jardim, "Clicking Clean: Who is Winning the Race to Build a Green Internet?" Greenpeace Inc., Washington, DC, Tech. Rep., 2017. [Online]. Available: http://www.clickclean.org/

[11] T. Mastelic, A. Oleksiak, H. Claussen, I. Brandic, J.-M. Pierson, and A. V. Vasilakos, "Cloud Computing: Survey on Energy Efficiency," *ACM Computing Surveys*, vol. 47, no. 2, pp. 1–36, Dec. 2014.

[12] Y. Ebadi and N. J. Navimipour, "An energy-aware method for data replication in the cloud environments using a Tabu search and particle swarm optimization algorithm," *Concurrency and Computation: Practice and Experience*, vol. 31, no. 1, p. e4757, 2019, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.4757. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4757

[13] I. Hamzaoui, B. Duthil, V. Courboulay, and H. Medromi, "A Survey on the Current Challenges of Energy-Efficient Cloud Resources Management," *SN Computer Science*, vol. 1, no. 2, p. 73, Feb. 2020. [Online]. Available: https://doi.org/10.1007/s42979-020-0078-9

[14] S. Wang, J. Liu, J.-J. Chen, and X. Liu, "PowerSleep: A Smart Power-Saving Scheme With Sleep for Servers Under Response Time Constraint," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 3, pp. 289–298, Sep. 2011. [Online]. Available: http://ieeexplore.ieee.org/document/6042345/

[15] W. Li, Y. Yang, and D. Yuan, "A Novel Cost-Effective Dynamic Data Replication Strategy for Reliability in Cloud Data Centres," in *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*. Sydney, Australia: IEEE, Dec. 2011, pp. 496–502.

[16] N. K. Gill and S. Singh, "A dynamic, cost-aware, optimized data replication strategy for heterogeneous cloud data centers," *Future Generation Computer Systems*, vol. 65, pp. 10–32, Dec. 2016.

[17] J. Liu, H. Shen, H. S. Narman, Z. Lin, and Z. Li, "Popularity-aware Multi-failure Resilient and Cost-effective Replication for High Data Durability in Cloud Storage," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1–1, Oct. 2018.

[18] F. Xie, J. Yan, and J. Shen, "A Data Dependency and Access Threshold Based Replication Strategy for Multi-cloud Workflow Applications," in *Service-Oriented Computing – ICSOC 2018 Workshops*, ser. Lecture Notes in Computer Science, X. Liu, M. Mrissa, L. Zhang, D. Benslimane, A. Ghose, Z. Wang, A. Bucchiarone, W. Zhang, Y. Zou, and Q. Yu, Eds. Springer International Publishing, Apr. 2019, pp. 281–293.

[19] Y. Mansouri and R. Buyya, "Dynamic replication and migration of data objects with hot-spot and cold-spot statuses across storage data centers," *Journal of Parallel and Distributed Computing*, vol. 126, pp. 121–133, Apr. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731518309092

[20] S.-Q. Long, Y.-L. Zhao, and W. Chen, "MORM: A Multi-objective Optimized Replication Management strategy for cloud storage cluster," *Journal of Systems Architecture*, vol. 60, no. 2, pp. 234–244, Feb. 2014.

[21] Y. Lin and H. Shen, "EAFR: An Energy-Efficient Adaptive File Replication System in Data-Intensive Clusters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1017–1030, Apr. 2017, conference Name: IEEE Transactions on Parallel and Distributed Systems.

[22] Z. Xu, N. Deng, C. Stewart, and X. Wang, "CADRE: Carbon-Aware Data Replication for Geo-Diverse Services," in *2015 IEEE International Conference on Autonomic Computing*. Grenoble, France: IEEE, Jul. 2015, pp. 177–186.

[23] M. Séguéla, R. Mokadem, and J.-M. Pierson, "Comparing energy-aware vs. cost-aware data replication strategy," in *2019 Tenth International Green and Sustainable Computing Conference (IGSC)*, Oct. 2019, pp. 1–8.

[24] M. Alghamdi, B. Tang, and Y. Chen, "Profit-based file replication in data intensive cloud data centers," in *2017 IEEE International Conference on Communications (ICC)*. Paris, France: IEEE, May 2017, pp. 1–7.

[25] A. Hylick and R. Sohan, "A methodology for generating disk drive energy models using performance data," *Energy (Joules)*, vol. 80, p. 100, 2009.

[26] M. Song, "Minimizing Power Consumption in Video Servers by the Combined Use of Solid-State Disks and Multi-Speed Disks," *IEEE Access*, vol. 6, pp. 25 737–25 746, Jun. 2018.

[27] M. Rhu, M. Sullivan, J. Leng, and M. Erez, "A locality-aware memory hierarchy for energy-efficient GPU architectures," in *2013 46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. Davis, California: ACM Press, Dec. 2013, pp. 86–98.

[28] Micron, "TN-40-07: Calculating Memory Power for DDR4 SDRAM," 2017. [Online]. Available: https://www.micron.com/support

[29] R. Basmadjian, H. D. Meer, R. Lent, and G. Giuliani, "Cloud computing and its interest in saving energy: the use case of a private cloud," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 1, no. 1, p. 5, Jun. 2012. [Online]. Available: https://doi.org/10.1186/2192-113X-1-5

[30] A. Vishwanath, K. Hinton, R. W. Ayre, and R. S. Tucker, "Modeling energy consumption in high-capacity routers and switches," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 8, pp. 1524–1532, 2014, publisher: IEEE.

[31] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," *TIK-report*, vol. 103, 2001.

[32] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, Jan. 2011.

[33] F. F.-H. Nah, "A study on tolerable waiting time: how long are Web users willing to wait?" *Behaviour & Information Technology*, vol. 23, no. 3, pp. 153–163, May 2004.

[34] D. Moyer, S. L. Carson, T. K. Dye, R. T. Carson, and D. Goldbaum, "Determining the influence of Reddit posts on Wikipedia pageviews," in *Ninth international AAAI conference on web and social media*. AAAI Press Oxford, UK, 2015, pp. 75–82.

[35] K. Lerman and R. Ghosh, "Information Contagion: an Empirical Study of the Spread of News on Digg and Twitter Social Networks," *arXiv:1003.2664 [physics]*, Mar. 2010, arXiv: 1003.2664. [Online]. Available: http://arxiv.org/abs/1003.2664

Institut de Recherche
en Informatique de Toulouse
CNRS - INP - UT3 - UT1 - UT2J

IRIT, Université Toulouse 3 Paul Sabatier (UT3)
118 Route de Narbonne - F-31062 TOULOUSE Cedex 9 FRANCE
Tel: (33) 5 61 55 67 65 – mail: contact@irit.fr

IRIT, INP - ENSEEIHT
2, Rue Camichel, BP 7122 - 31071 TOULOUSE cedex 7 FRANCE
IRIT, Université Toulouse 1 Capitole (UT1)
Place Anatole-France 31042 TOULOUSE cedex 9 FRANCE
IRIT, Université Toulouse 2 Jean Jaurès (UT2J)
Maison de la Recherche, 5 allées Antonio Machado - 31058 TOULOUSE cedex 9 FRANCE
IRIT, IUT de Blagnac – Université Toulouse 2 Jean Jaurès (UT2J)
1 Place Georges Brassens, BP 60073 - 31703 BLAGNAC cedex FRANCE

http://www.irit.fr