Dynamic Energy and Expenditure Aware Data Replication Strategy

Morgan Séguéla, Riad Mokadem, Jean-Marc Pierson

Dynamic Energy and Expenditure Aware Data Replication Strategy

Morgan Séguéla, Riad Mokadem, Jean-Marc Pierson
Institut de Recherche en Informatique de Toulouse (IRIT)
Université de Toulouse, France
Email: {morgan.seguela, riad.mokadem, jean-marc.pierson}@irit.fr

***Abstract*—Nowadays, data are generated and accessed from all over the world. Applications and companies store these data on geo-distributed cloud providers that have to be profitable while reducing their environmental impact. As Cloud providers aim to satisfy their service level objectives in terms of availability and response time, they often rely on data replication. In this paper, we propose a dynamic data replication strategy (DE2ARS) that adapts the number of replicas according to the workload and addresses energy consumption and expenditure issues. It follows an initial placement and is triggered by a Control Chart. We first compare different parameter choices in order to provide better analysis for the proposed strategy. We compare DE2ARS with strategies from the literature. Results highlight the fact that DE2ARS reaches its goal to reduce both energy consumption and expenditure while having good performance in a strong workload context.**

***Index Terms*—Cloud, Data replication, Provider expenditure, Energy consumption, SLA violation**

I. Introduction

Since the spread of information technologies, companies and laboratory provide and request data from all over the world and some items are highly requested. This implies some availability and performance issues that can be addressed by replicating data, which aims to satisfy these issues. Other objectives can also be achieved such as reducing energy consumption or reducing expenditure [1].

This paper focuses data replication in Cloud systems, characterized by elasticity, that adapts resources automatically and allows the use of a *Pay-As-You-Go* economic model, where tenants pay what they consume. A Service Level Agreement (SLA), signed by both tenant and provider, specifies prices to rent resources, Service Level Objectives (SLO) the provider has to achieve, and penalties it must pay otherwise.

Reducing provider's environmental impact is getting more important and can be seen through the reduction of energy consumption [2]. Thus, multiple techniques and technologies have been developed, like sleep state and consolidation.

In this paper, we propose a dynamic data replication strategy, *DE2ARS* (Dynamic Energy and Expenditure Aware data Replication Strategy), which occurs following an initial placement [3]. DE2ARS aims to reduce both energy consumption and expenditure while improving performance for the tenant. This strategy is triggered with a control chart [4] which detects a workload evolution based on SLO violations. According to this workload evolution detected, one or several replicas is/are created or deleted, and decisions consider expenditure and energy consumption issues. Replication leverages on energy saving technologies like PowerSleep. For identifying data item to replicate, we use the Pareto principle supposing that 80 percent of violations come from 20 percent of data stored. We compared *DE2ARS* with strategies from the literature. And it achieves its objective to reduce both energy consumption and expenditure while improving performance. Note that we consider read-only data items that will not be updated.

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 expenditure. Section III describes the proposed strategy. In Section IV, we validate the proposed strategy.Finally, we conclude and draw some lines for future works in Section V.

II. State of the Art

We focus on data replication strategies that take into account energy consumption, expenditure, or both.

In terms of reducing energy consumption, [5] proposed *MORM*, a static data replication that answers to availability, service time, load balancing, energy consumption and latency issues where each objective is weighted. *Boru et al.* [6] proposed a strategy triggered by a threshold on the number of accesses and considers a hierarchical architecture, where data are replicated in lower level to reduce energy and bandwidth consumption. [7] used two *Least Recently Used* lists that permits to classify data item according to their heat (popularity). Then this strategy stores hot data items in hot nodes (energy consuming) and cold ones in cold nodes (reducing their power consumption).

On the side of taking expenditure into account, replication in *PEPR* [8] is triggered by an SLO violation, i.e. when the response time of a task is higher than a threshold, and replicates only if the provider is still profitable. [9] calculates 2 scores at each time frame: (i) an availability score, based on a file availability probability, and (ii) a replication factor, based on access frequency. As the proposed strategy considers a hierarchical architecture, it leverages on heterogeneity to create or remove replicas in order to reduce expenditure. [10] use an *Auto-Regressive Integrated Moving Average* to predict the number of accesses for a short period of time and then process a replication algorithm in order to minimize the global cost by minimizing the cost in each region.

Only few strategies consider both energy consumption and expenditure as objectives. In [11], authors propose a static data replication strategy which models reads and writes energy consumption and expenditure where each objective is weighted. The static data replication strategy proposed by [3], called *E2ARS*, replicates in a 2 steps decision process. The first step leverages on heterogeneity to reduce both energy consumption and expenditure while the second step uses sleep states to reduce power consumption of unused nodes.

Compared to the literature, our strategy, *Dynamic Energy and Expenditure Aware data Replication Strategy* (*DE2ARS*), addresses the reduction of both energy consumption and expenditure through a dynamic data replication strategy triggered by control charts. To the best of our knowledge, this is the first time that control charts are used in a data replication context.

III. Dynamic data replication strategy

We consider the following process of executing a query: A node receives a task, needed data items are transferred to the executing node. At the end of this process the provider verifies if there is an SLO violation according to the response time.

DE2ARS, which occurs after an initial static placement [3], aims to answer to an *increasing* and *decreasing* workload. To do so, we focus on the proportion of SLO violations in a sample.

In our context, this metric is expressed by a response time that is higher than a threshold, i.e., an SLO. We adapt here the SLO violations cost, from Google's SLOs[1]. If a node is unavailable more than 5% of the renting time, they refund 100% of the rent. So, if more than 5% of tasks have a response time higher than the SLO then, we refund 100% of the tasks price. To avoid reaching this proportion of violations, we use Control Charts [4].

A. Triggering the replication

1) Introducing Control Charts: Control Charts are tools from Statistical Process Control, a branch of Quality Management. They are based on probability laws where parameters are known or estimated. It permits to know the probability of an event and then take action when issues occur. Control charts are mostly used in computer science to monitor data quality [12] or to manage a process variability [13].

Control charts are defined by a target, limits around the target and a set of rules that trigger the control chart. Authors that introduced and described control charts [4], [14], used 3σ limits in a Normal Distribution.

In our case, an event corresponds to the proportion of SLO violations in a sample. This event is considered following a Binomial distribution, as each SLO violation follows a Bernoulli distribution, with a probability p of successes (i.e. probability for a task to generate an SLO violation), and a sample size n. Considering an event that follows a Binomial law, implies the use of a so-called p-chart.

As it was described before, we aim to consider an *Upper Control Limit* (UCL) that has to be lower than 5%. We have to consider a $p \in [0, 5\%[$ with $p + \frac{3\sigma}{n} < 5\%$, then we choose to use $p = 2.5\%$. Having a low value of p implies that n has to be high enough in order to have a more precise estimation.

To answer this kind of issues, authors of [15], compared multiple rules, and we use the following one:

$$n * p * (1 - p) > 9 \Leftrightarrow n > \frac{9}{0.025 * 0.975} \Leftrightarrow n > 369.2 \quad (1)$$

Based on this rule, we choose a sample size parameter $n = 370$. Setting parameters p and n permits to set control limits of the control chart as follows:

$$CL = p \pm 3 * \sqrt{\frac{p(1-p)}{n}} \quad (2)$$

Here *Upper Control Limit* equals 4.93%, and *Lower Limit Control* equals 0.07%. They correspond to 3σ limits. Between them, 3 zones are set at each side of the target, to apply rules: 0-1σ, 1-2σ and 2-3σ. Rules are introduced by [14] and then extended by [16]. We focus here on rules that detects a deviation from the target:

R1 - 1 point outside 3σ limits
R2 - 9 points in a row in the same side of the target
R3 - 6 points in a row steadily increasing of decreasing
R4 - 2 out of 3 points being beyond 2σ in the same side of the target
R5 - 4 out of 5 points being beyond 1σ in the same side of the target

2) Detecting Workload Evolution: As it was described before, control charts are triggered by rules applied to samples. Each rule permits to detect a deviation from the mean, thus multiple rules can be triggered detecting contradictory evolution. To avoid this issue, we give a score to the event, which start at 0. Then, if a triggered rule highlights a workload increasing, it adds +1 to the score. At the opposite, if this highlights a workload decrease, it decreases the score by 1.

For the rules R1, R2, R4, R5, a proportion that is higher than the target, correspond to an increase and vice versa. And for the rule R3, if proportions are increasing, it means a workload increase, and vice versa.

At the end of the control chart evaluation, if the score is positive, the strategy will create new replicas. If the score is negative, the strategy will delete replicas. Otherwise, the number of replicas is not changed.

3) Information Retrieval: During the lifetime of the system, the strategy gathers 2 kinds of information from tasks. First, each task that breaks the SLO is assigned the value of 1, or 0 otherwise, in the sample given to the control chart. Then, if a task breaks the SLO, we gather information about on which node the task was processed and which data items was needed.

B. Notation

Before describing replicas creation and removal, we introduce the following notations:

Let F be a set of items stored on a set of nodes N.

Each **item** $f_i \in F, 1 \leq i \leq z$ has a size s(f_i) (in MB).

Each **node** $n_j \in N, 1 \leq j \leq m$ with a storage capacity cp_j (in MB). Nodes have a static power consumption

[1]https://cloud.google.com/compute/sla (03/07/2022)

$Pow_{static}(n_j)$ (in Watt), that depends on the server components. Similarly, each node has a storage cost $Cost_{storage}(n_j)$ (in \$/MB per second), that mostly depends on the region, hardware and so on.

ϕ is a matrix of size (z, m) which represents the placement of data items on nodes: $\phi(f_i, n_j)$ is equal to 1 when f_i is stored on n_j and 0 otherwise.

$diff_{time}(f_i, n_j)$ (in seconds) corresponds to the duration between the latest query of f_i on n_j and the time when the score is calculated.

$propEC$ which is the weight given for the energy consumption part. Its values goes from 0 for not considering energy consumption to 1 for only considering energy consumption.

C. Creating Replicas

When the number of violations is getting high, it means that the system is being overloaded. Replicating data may balance workload among multiple nodes and avoid bottleneck.

Replication is done in two steps: Plan and Replicate.

1) Replication Plan: At first, the strategy plans which data item will be replicated and on which node.

a) Which items: Based on the Pareto Principle, we sort data items according to their number of violations, and select data items that accumulate 80% of violations, supposing that they come from 20% of data items.

b) Which data center: Based on the same principle, for each data item, the strategy sorts regions according to their number of violations, then selects regions that correspond to 80% of the data item violations. This is repeated for data center for each chosen regions for each item, where data centers of a region are sorted according to their number of violations, and selected to reach 80% of SLO violations of the region.

c) Which node: When choosing on which node to store the replica, a trade-off has to be considered between (i) storing the replica on a new node to balance the workload but increasing its energy consumption, or (ii) storing the replica on a node that already stores data to keep a low number of activated nodes with a possibility to create a bottleneck.

To handle this trade-off, we consider a score corresponding to the transfer time (seconds) per Megabytes (MB) between two nodes n_j and $n_{j'}$ for a data item f_i. Here, n_j corresponds to a possible node on which the strategy will replicate, and $n_{j'}$ the closest node that stores a replica of f_i.

If this score goes above a threshold for a node, we consider this node as a bottleneck. This threshold can be set by an administrator in order to apply a policy more or less consuming in terms of energy consumption.

DE2ARS considers at first nodes that stores items, and if any of them is not considered as a bottleneck, it selects the one with the lowest score. If all of them are considered as bottleneck, it will wakes up and chooses a new node if this is possible, or DE2ARS chooses the node with the lowest score if they are all already up. These choices are under the constraint of available storage, that has to be higher than the item size.

2) Replicating: The main idea of these two steps, is to avoid huge overhead by transferring data only for replication.

When data centers receive the replication plan for their nodes, they check within current tasks which data items are used, and if they have to be replicated according to the replication plan. If this is the case, it transfers the item to the chosen node. If some items are not replicated, DE2ARS checks new coming tasks and when one requires data items that have to be replicated, then it takes advantage of the transfer to compute the task and replicate at the same time.

D. Removing Replicas

Elasticity implies a replica management that can adapt to an increasing workload by replicating, but also when the workload decreases by removing replicas. It has to be noted that, dynamic data replication in DE2ARS follows a static data replication. For the sake of simplicity, during the removing process, we do not remove replicas created during this initial placement. We suppose here that other events might be more efficient to reconsider this initial placement.

1) Choosing the number of removal: During our research, we considered multiple possibilities when removing replicas.

First, we considered removing one replica at a time, which should avoid under-provisioning, but this might be too slow to put enough nodes in sleep state in order to reduce energy consumption. At the opposite, we tried to remove all created replicas during the dynamic replication, creating a cycle of replication/removal, as workload might stay high. One last possibility, is to remove a proportion of replicas, in order to make it faster to delete replicas without implying a cycle of creation and removal.

2) Choosing replicas to remove: Finally, we addressed the way of choosing replicas to remove through a removal score, based on energy consumption and expenditure to store each replica. Then replicas are sorted according to this score, and top ones will be removed according to the number of removal. To compute this score, we introduce the following notations.

a) Energy consumption: The energy consumption score of a replica f_i stored on a node n_j is calculated as follow:

$$EC_{score}(f_i, n_j) = \phi(f_i, n_j) * Pow_{static}(n_j) * diff_{time}(f_i, n_j) * \frac{s(f_i)}{\sum_{f'_i in F} \phi(f'_i, n_j) * s(f'_i)} \quad (3)$$

b) Expenditure: The expenditure score of a replica on a node is calculated as follow:

$$EX_{score}(f_i, n_j) = \phi(f_i, n_j) * s(f_i) * Cost_{storage}(n_j) * diff_{time}(f_i, n_j) \quad (4)$$

c) Removal score: In order to compare those scores, they are normalized. It means that scores are reduced by their mean and divided by their standard deviation of their category. This permits to keep the distribution of results and highlights scores that are particularly high or low. The normalized score

is written:$Norm_{AC}(f_i, n_j),\ AC \in \{EC, EX\}$. The removal score is then calculated as follows:

$$DelScore(f_i, n_j) = Norm_{EC} * propEC + Norm_{EX} * (1 - propEC) \quad (5)$$

With this score, the administrator has the possibility to balance between energy consumption and expenditure, then replicas with highest removal scores are immediately deleted.

IV. Evaluation

A. Experimental environment

1) Strategies parameters: DE2ARS uses [3] as an initial placement in our evaluations. In this evaluation section, all experiments associated with DE2ARS uses a value of $propEC$ of 0.5. We highlight different policies proposed by DE2ARS as different parameters have been set.

a) Bandwidth Threshold: In section III-C1, we choose to set the threshold to consider a node as a bottleneck or not and we wanted to highlight its impact. To do so, we used a *VLow* and *Low* values of 0.0001 and 0.001 s/Mo respectively, a threshold of 0.005 s/Mo, called *Mid*. And *High* and *VHigh* values of 0.01 s/Mo and 0.1 s/Mo respectively.

b) Other strategies: We compare our strategy with other data replication strategies from the literature (see Section II) and a control strategy used as a baseline. As a control strategy, we used a static strategy that creates 3 replicas and places them randomly, called *3Rand*. From the literature, we compare our strategy with 2 static replication strategies: *MORM* [5] and *E2ARS* [3]. We also compare our strategy with dynamic data replication strategies: *PEPR* [8] which takes provider's profit into account (with an income of 0.0081$ per task), and *Boru* [6] which considers energy and bandwidth consumption.

2) Simulation Parameters: In order to compare those strategies, we implemented them on CloudSim [17] extended by [8] to implement an economic model and a large scale architecture. We added to this extension an estimation of energy consumption and a heterogeneity of components, power consumption and costs. We also implemented the Sleep State technique [18], [19] which permits to set nodes in sleep state to reduce their energy consumption. The use of this technique implies an implementation of a task scheduler that takes this technique into account.

a) Task Scheduler: We implemented a naive task scheduler that knows the state of all tasks processed in its data center. It places tasks on awakened nodes and places a task on a sleeping node only if the response time to process this task is higher than the SLO on running nodes. Then an empty node that is idle for 15 will be turned back in sleep state.

b) Provider: We consider a Cloud provider with a large scale architecture divided as follows. There are 4 regions with 5 data centers per region and 64 nodes inside each data center. Each region is associated with a profile in a cyclic manner. We consider an SLO of having a response time lower than 15 seconds.

c) Components and Energy parameters: To simulate a real architecture, we have used components from Grid5000 [20]. This testbed is geographically distributed and components of each nodes are available here[2]. We consider 3 different profiles, associated with 3 different grid5000 cluster: 1) Paranoïa cluster in Rennes, 2) Gros cluster in Nancy, 3) Dahu cluster in Grenoble.

d) Prices: Prices are derived from Google Cloud[3], where prices depend on locations. Cost profiles correspond to regions drawn amongst all available regions: (i) London, (ii) Taiwan, (iii) Zurich et (iv) Hong Kong. However, to set an unique SLO violation cost, we suppose a unique price per task.

e) Network: Network parameters comes from different sources. Bandwidth parameters are based on [6] and latency parameters from [6] and Wikipedia[4] technical documents.

f) Workload: In our experiments, we consider having 256 items with a size between 200Mo and 8Go. We consider 2 different kinds of workload: i) Short term and ii) Long term. The first experiment (*XP1*) is a 4 hours experiment with 150k tasks, with a workload based on [21] which shows that when there is a post with a link to a Wikipedia article, the number of accesses increases and the loss of interest reduces at different speed. We suppose that the workload follows a Gamma distribution, with $\alpha = 4$ and $\beta = 600$. The second experiment (*XP2*) is a 4 days experiment with 2 millions tasks, with a workload based on [22] that shows a correlation between Google queries and Wikipedia queries. We chose to use the opening of Vaccination of COVID-19 to the public in France in May 29th 2021, that permits to generate a peak in terms of accesses.

To compare these policies and strategies, we used 4 metrics: (i) the energy consumed (in MJ) by the Cloud at the end of the experiment, (ii) the total cost for the provider (in $), (iii) the number of created replicas, (iv) the proportion of violations.

A summary of simulation parameter are provided in table I. A Github repository is available[5] to replicate the work and explore different experiments and parameters.

B. Results

1) Comparison between bandwidth threshold: We first compare the number of replicas created and the number of violations, summarized in the first part of the table II. In XP1, we can see that *VLow* and *Low* have higher SLO violations, thus a higher number of replicas created compared to *High* and *VHigh*. This can be explained by the fact that when the workload increases, the number of items that are being transferred on the data center network increases. In this context, waking up a node to store a data item on it, implies to transfer this item, increasing the bandwidth consumption, and response time, implying more violations and replicas and so on.

[2]https://www.grid5000.fr/w/Hardware
[3]https://cloud.google.com/ (02/28/2020)
[4]https://wikitech.wikimedia.org/wiki/Network_design (08/28/2020)
[5]https://github.com/MorganSeguela/cloud-2022-XPS

Parameters	Values	Parameters	Values
Number of files	256	File size	[0.2, 4, 8] GB
Number of region per Cloud	4	Number of DCs per region	5
Number of nodes per DC	64	Response time SLO	< 15s
Components Profile	[Paranoïa, Gros	Prices Profile	[London, Taiwan
each region	Dahu, Paranoïa]	*each region*	Zurich, Hong Kong]

TABLE I
SIMULATION PARAMETERS

	Number of Replicas $\times$ *1000*		SLO violations *in permille*	
Strategy	XP1	XP2	XP1	XP2
VLow	5.54 (0.2)	2.12 (0.1)	41.2‰ (2‰)	1.66‰ (0.2‰)
Low	5.51 (0.2)	2.1 (0.1)	40.9‰ (3‰)	1.71‰ (0.1‰)
Mid	5.55 (0.2)	2.07 (0.1)	41.4‰ (3‰)	1.66‰ (0.2‰)
High	5.47 (0.2)	2.05 (0.1)	40.4‰ (3‰)	1.84‰ (0.2‰)
VHigh	5.43 (0.3)	2.12 (0.1)	40.5‰ (3‰)	1.7‰ (0.2‰)
3Rand	0.51 (0)	0.51 (0)	632‰ (10‰)	230‰ (19‰)
MORM	38.3 (38)	29.9 (10)	0‰ (0‰)	0‰ (0‰)
E2ARS	0.89 (0.1)	0.92 (0.1)	584‰ (25‰)	167‰ (35‰)
Boru	–	–	905‰ (4‰)	363‰ (6‰)
PEPR	–	–	589‰ (37‰)	44‰ (6‰)
DE2ARS	–	–	56.6‰ (2.8‰)	19.5‰ (1.3‰)

TABLE II
RESULTS IN TERMS OF NUMBER OF REPLICAS AND PROPORTION OF SLO VIOLATIONS FOR EACH POLICY AND STRATEGY IN EACH EXPERIMENT
MEAN (STANDARD-DEVIATION)

	Energy (MJ)		Expenditure (k$)	
Strategy	XP1	XP2	XP1	XP2
VLow	218.2 (2.9)	4588 (84.2)	4.23 (0.17)	56.96 (4.23)
Low	214.8 (3.6)	4255 (65.3)	4.12 (0.25)	56.53 (4.03)
Mid	209.7 (3.9)	4159 (52.9)	4.23 (0.23)	56.59 (4.31)
High	202.9 (4)	4128 (70.3)	4.27 (0.23)	59.76 (5.24)
VHigh	186.5 (3.6)	3833 (28)	4.24 (0.27)	56.49 (4.88)
3Rand	594 (22.3)	7006 (63.8)	18.96 (0.62)	259.5 (6.5)
MORM	501 (18.2)	11570 (5)	10.25 (9.68)	14.8 (3.1)
E2ARS	233 (38)	3770 (2.2)	13.8 (1.8)	200.5 (22.4)
Boru	347 (0.3)	3877 (1.2)	4.48 (0.02)	53.2 (0.1)
PEPR	552 (59.1)	4768 (0.3)	22.9 (1.08)	155.6 (5.4)
DE2ARS	204 (4.2)	3959 (15.7)	5.9 (0.28)	145.8 (9.8)

TABLE III
RESULTS IN TERMS OF ENERGY CONSUMPTION AND EXPENDITURE FOR EACH POLICY AND STRATEGY IN EACH EXPERIMENT
MEAN (STANDARD-DEVIATION)

As it was foreseeable, in the first part of table III, the energy consumption decreases as the threshold increases due to fewer nodes that are woken up reducing the power needed. This reduction is about 14.5% between *VHigh* and *VLow* in XP1 and 16.5% in XP2. In this table, no trend can be highlighted in terms of expenditure, unless the fact that *High* threshold slightly increase expenditure. In the following evaluations, we considered the *Mid* value of the threshold to avoid bottleneck.

2) Comparison between different strategies: In this section, we are evaluating the whole data replication strategy, *DE2ARS*, considering a bandwidth threshold of 0.005 s/MB (*Mid*) and a 10% replicas removal. The last part of table II corresponds to the number of replicas created by static data replication strategy, and SLO violations of all compared strategies. Then, results in terms of expenditure and energy consumption are provided in the last part of table III.

a) Comparing replication and violations: A first interesting strategy is *MORM* which creates an average of more than 29,000 replicas implying mostly no violation, making it the strategy that as the lowest number of violations. The other static strategy *E2ARS*, creates 70% more replicas than *3Rand* but slightly reduces the number of violations.

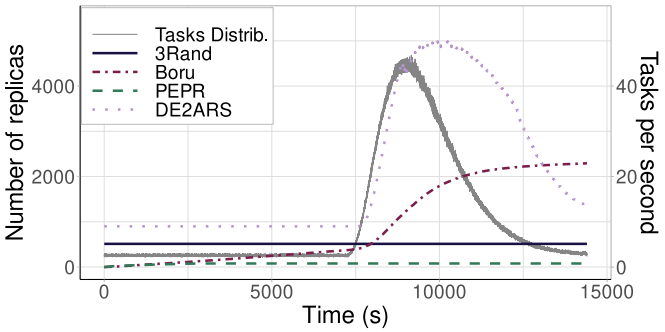

Fig. 1. Tasks distribution and Number of replicas over time for each strategy of the first experiment (XP1)

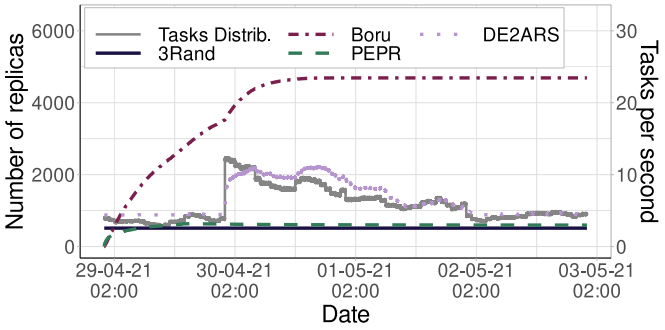

Fig. 2. Tasks distribution and Number of replicas over time for each strategy of the second experiment (XP2)

On figure 1 and figure 2, it can be seen that *Boru* adapts its number of replicas according to the workload when it increases, as it does not have any replica removal mechanism. At the end of each experiment, it has the most number of replicas, compared to other dynamic strategy, yet it also has the highest number of violations. It is explained by its hierarchical architecture that creates bottlenecks.

PEPR is the second strategy in terms of number of violation in XP1, with an important reduction of SLO violations in XP2 compared to *3Rand*. However, in terms of replication, *PEPR* stops replicating even if the number of violations keeps increasing, due to the fact that when the expenditure is higher than revenue, *PEPR* stops replicating.

The proposed *DE2ARS* strategy creates and removes replicas according to the workload. Furthermore, it can be seen that this strategy is the second lowest in terms of SLO violations with a proportion of 5.66% and 1.95% for the first and second experiment respectively.

b) Comparing energy consumption and expenditure: *MORM* reduces expenditure by 46% compared to *3Rand* with a high replicating cost, but lower cost in long term experiment as no data transfer are needed. As it has the highest number of replicas, it is also the most energy consuming strategy.

E2ARS is able to reduce its energy consumption by 61% (XP1) and 47.2% (XP2) and its expenditure by 27.5% (XP1) and 22.7% (XP2) compared to *3Rand*. It succeeds to reduce them, by replicating region-wide, while consolidating data on a few number of nodes.

Boru et al. is the cheapest strategy, and also reduces its energy consumption by 42% (XP1) and 44% (XP2) compared to *3Rand*. These results can be explained by the fact that only a few number of nodes can store data, and replication are done inside each region at first, reducing transfer cost.

PEPR permits to reduce its energy consumption by 7% (XP1) and 32% (XP2) compared to *3Rand*, as it leverages on the task scheduler to replicates as closely as possible to node that broke the SLO. However, it increases its expenditure in XP1 and reduces it by 40% in XP2 compared to *3Rand*, as it has a very low number of replicas in XP1.

Finally, the proposed strategy *DE2ARS* reduces its expenditure by 69% and 44% compared to *3Rand* on XP1 and XP2 respectively, while also reducing its energy consumption by 66% on XP1 and 43% on XP2 compared to *3Rand*. In this context, *DE2ARS* outperforms other strategies by achieving its objective to reduce both the energy consumption and the expenditure while keeping a low proportion of violations.

V. Conclusion

We proposed DE2ARS (*Dynamic Energy and Expenditure Aware data Replication Strategy*) that follows an initial placement. it aims to reduce both the energy consumption and expenditure while taking into account workload variations. replication is triggered by a control chart, a tools from Statistical Process Control, that permits to adapt the number of replicas to the workload with low computing power. The proposed strategy can be tuned in different ways, and removal can be tuned to focus more on reducing storage energy consumption or reducing storage expenditure.

We compared our strategy with a control strategy that creates and places 2 replicas randomly (3Rand). DE2ARS was also compared with other strategies from the literature, where some of them takes into account energy consumption (MORM, Boru et al.) and expenditure (PEPR). Results show that DE2ARS achieves its goal to adapt its number of replicas according to the workload. It also reaches its objective of reducing both energy consumption and expenditure while reducing the number of violations in a strong workload context.

One limitation of our work is that all nodes and data items are static although the proposed strategy is dynamic. As a future work, we aim to consider the addition or removal of data items and/or nodes. Further, we plan to implement the proposed DE2ARS on a real cloud environment.

References

[1] 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)*, Alexandria, VA, US, Oct. 2019, pp. 1–8.

[2] 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.

[3] M. Séguéla, R. Mokadem, and J.-M. Pierson, "Energy and Expenditure Aware Data Replication Strategy," in *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*. Chicago, IL, USA: IEEE, Sep. 2021, pp. 421–426.

[4] W. A. Shewhart, *Economic control of quality of manufactured product*. Macmillan And Co Ltd, London, 1931.

[5] 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.

[6] 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.

[7] L. Zhang, Y. Deng, W. Zhu, J. Zhou, and F. Wang, "Skewly replicating hot data to construct a power-efficient storage cluster," *Journal of Network and Computer Applications*, vol. 50, pp. 168–179, Apr. 2015.

[8] U. Tos, R. Mokadem, A. Hameurlain, T. Ayav, and S. Bora, "Ensuring performance and provider profit through data replication in cloud systems," *Cluster Computing*, vol. 21, pp. 1479–1492, Dec. 2017.

[9] 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.

[10] T.-Y. Hsu and A. D. Kshemkalyani, "A Proactive, Cost-aware, Optimized Data Replication Strategy in Geo-distributed Cloud Datastores," in *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*, ser. UCC'19. New York, NY, USA: Association for Computing Machinery, Dec. 2019, pp. 143–153.

[11] 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, Jan. 2019.

[12] L. A. Jones-Farmer, J. D. Ezell, and B. T. Hazen, "Applying Control Chart Methods to Enhance Data Quality," *Technometrics*, vol. 56, no. 1, pp. 29–41, Jan. 2014.

[13] J. Kim, G. M. Abdella, S. Kim, K. N. Al-Khalifa, and A. M. Hamouda, "Control charts for variability monitoring in high-dimensional processes," *Computers & Industrial Engineering*, vol. 130, pp. 309–316, Apr. 2019.

[14] C. Western Electric, "Western Electric - Statistical Quality Control Handbook," 1956.

[15] M. Schader and F. Schmid, "Two Rules of Thumb for the Approximation of the Binomial Distribution by the Normal Distribution," *The American Statistician*, vol. 43, no. 1, pp. 23–24, Feb. 1989, publisher: Taylor & Francis Group.

[16] L. S. Nelson, "The Shewhart Control Chart—Tests for Special Causes," *Journal of Quality Technology*, vol. 16, no. 4, pp. 237–239, Oct. 1984.

[17] 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.

[18] D. Meisner, B. T. Gold, and T. F. Wenisch, "PowerNap: eliminating server idle power," *ACM SIGARCH Computer Architecture News*, vol. 37, no. 1, pp. 205–216, Mar. 2009.

[19] 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.

[20] D. Balouek, A. C. Amarie, G. Charrier, F. Desprez, E. Jeannot, E. Jeanvoine, A. Lèbre, D. Margery, N. Niclausse, L. Nussbaum, O. Richard, C. Perez, F. Quesnel, C. Rohr, and L. Sarzyniec, "Adding Virtualization Capabilities to the Grid'5000 Testbed," in *Cloud Computing and Services Science*, ser. Communications in Computer and Information Science, I. I. Ivanov, M. van Sinderen, F. Leymann, and T. Shan, Eds. Cham: Springer International Publishing, 2013, pp. 3–20.

[21] 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*. Oxford, UK: AAAI Publications, 2015, pp. 75–82.

[22] M. Yoshida, Y. Arase, T. Tsunoda, and M. Yamamoto, "Wikipedia Page View Reflects Web Search Trend," in *Proceedings of the ACM Web Science Conference*, ser. WebSci '15. New York, NY, USA: Association for Computing Machinery, Jun. 2015, pp. 1–2.