An Approach to Reduce the Energy Cost of the Arbitrary Tree Replication Protocol

Robert Basmadjian
University of Passau
Innstrasse 43
Passau, Germany
Robert.Basmadjian@uni-passau.de

Hermann de Meer
University of Passau
Innstrasse 43
Passau, Germany
demeer@uni-passau.de

ABSTRACT

Until recently, there have been no efforts of devising energy-efficient replication protocols for large-scale distributed systems. In this paper, we introduce an approach that reduces the energy cost of a particular tree-structured replication protocol. We show that, by shutting down some replicas and by a simple logical structural transformation (rearrangement), our approach achieves comparable characteristics as the original protocol, yet with much reduced energy cost as well as overall energy consumption. The logical transformation does not necessitate the reconfiguration of the protocol whenever energy efficiency requirements change.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Performance attributes, Fault tolerance, Reliability, availability, and serviceability, Modeling techniques; H.3.4 [**Systems and Software**]: Distributed systems.

General Terms

Reliability, Algorithms, Model.

Keywords

Energy efficiency, quorum systems, replica control protocols, load, availability, energy cost.

1. INTRODUCTION

Replication involves creating and maintaining duplicates of data in order to provide *fault tolerance*, to improve the *system performance* and to increase the *service availability*. However, when replication is used, data becomes susceptible to inconsistency problems. Therefore, *synchronization* protocols, also known as *replica control protocols* (RCP), are required in order to maintain *data consistency* among replicas of the system.

Basically, such replica control protocols implement two operations; *read* (query) and *write* (update). To ensure *one-copy equivalence*, which is a common requirement for replication transparency, a read and write operations to two different copies of the data should not be allowed to execute concurrently. *Quorum systems*[1] are used by these protocols which serve as a basic tool of achieving one-copy equivalence.

Given the importance of the topic, several replica control protocols that enforce a specified semantics of accessing data have been described in the literature [17]. In general, these protocols can be classified into two categories: *structured* and *non-structured* RCPs. The difference is that only the former assumes that replicas of the system are arranged logically into a particular structure.

On the other hand, the concept of building *energy-efficient* distributed systems has attracted a great deal of attention lately due to the increase of *energy costs* (fuel) and the world wide desire to reduce CO_2 *emissions*. For instance, it was stated in [9] that storage centers can consume as much energy as a whole city if the number of servers reaches a certain level. Furthermore, in order to address to the problem of overheating, these storage centers are forced to be equipped with cooling systems, which as a matter of fact, increase the overall energy consumption. Therefore, minimizing and balancing the system's energy consumption becomes as important as the more traditional quality-of-service concerns, such as reliability, security, fault tolerance, and so forth.

One of the techniques to save energy costs, is to force some of the machines to enter into a sleep mode or even to be turned off. However, it is always desirable to keep the same overall performance level while minimizing the energy consumption of the system.

In this paper, we propose an approach to diminish the overall *energy consumption* of the *Arbitrary Tree* [3] protocol, where we compute the *energy cost* of its read and write operations. For this purpose, we adapt the energy cost model of [16] to the general approach of quorum systems. We show that, by shutting down significant amount of replicas of the system and by logically reorganizing the others into a new logical structure, it is possible to reduce the energy cost as well as consumption of [3] while preserving most of its characteristics. The tradeoff is that, in the worst case, the new approach has a worse load of 14% for its write operations than that of the original approach, however with much reduced energy cost as well as overall energy consumption.

[1] A quorum system is defined as a set of subsets of replicas called quorums having pair-wise non-empty intersections.

The rest of this paper is organized as follows. Section 2 discusses related approaches of replication and energy efficiency. After representing in section 3 the energy cost model, we present in section 4 the Arbitrary Tree protocol. Then, we introduce in section 5 our energy efficient approach by applying logical transformation. A comparison is given in section 6 by taking into account energy cost factor. The paper is concluded in section 7.

2. RELATED WORK

As mentioned above, several replication protocols have been proposed in literature which make use of *quorum systems* to achieve data consistency among the replicas. They differ according to various parameters of their read and write operations such as the *quorum size* (the number of replicas involved in a given operation), the *availability* (the probability of a given operation to terminate successfully), as well as the *load* induced on the system.

The general fault-tolerance (availability) properties of quorum systems were examined in [14]. Also, the load of these systems was studied in [13] and it was shown that the *optimal load* (both for read and write operations) of any quorum system of n replicas is $\frac{1}{\sqrt{n}}$ (highest is 1) if the smallest quorum is of size $\sqrt{n}$.

The well known *ReadOneWriteAll* (*ROWA*) [4] and *Majority Quorum Consensus* (MQC) [18] protocols have quorum sizes of $\mathcal{O}(n)$: the size thus increases linearly with the number of replicas of the system. By imposing a *logical structure* on the replicas of the system, it is possible to reduce the quorum sizes further. Several protocols have been introduced which also make use of quorum systems and assume that the replicas are organized logically into a specific structure: *finite projective plane* [12], a *grid structure* [6] and [13], or a *tree structure* [1], [2], [11], [10], [7] and [3].

In general, the *tree-structured* RCPs have a tight trade-off between the quorum size and the system load: a small size results in inducing a high load and vice versa. In [3], the *Arbitrary Tree* protocol was proposed and it was shown that its write operations induce an optimal system load of $\frac{1}{\sqrt{n}}$ with a quorum size of $\sqrt{n}$, which are lower than the state-of-the-art *tree-structured* RCPs, while preserving comparable write availability. On the other hand, it was proven that its read operations induce a quorum size of $\sqrt{n}$ which is smaller than previously proposed tree-structured RCPs with comparable system load and availability.

Recently, the need for saving the energy consumption of distributed systems composed of a large set of machines has attracted a great deal of research due to various ecological and economical concerns. Several techniques such as [5, 15, 8] have been proposed that deal with turning off some machines and shifting their load to other machines which have low thermic metrics. However, as it was stated in [19], these approaches are not appropriate for replication, because they assume that any request can be achieved by a number of currently active machines (replicas) which might have a stale copy of data. Our approach is also based on the idea of turning off replicas, however unlike [5, 15, 8], the overall data consistency is preserved.

In [16], a generic energy cost model was proposed which gives for each different architectural style (e.g., *client-server*, *Peer2Peer, Publish-Subscribe*) its corresponding energy cost model. Inspired by [16], we propose a model to compute the energy cost of read and write operations of replica control protocols based on quorum systems.

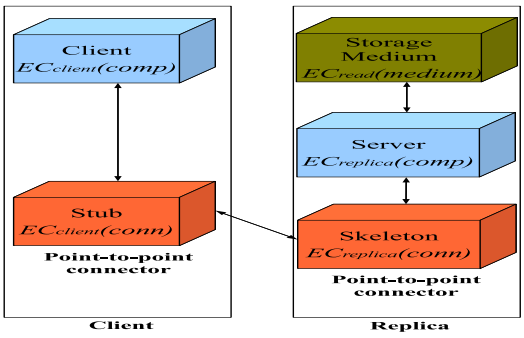


Figure 1: Interactions of performing a read operation on a single replica.

3. ENERGY COST MODEL

In order to compute the *energy cost* of read and write operations of the replica control protocols, we propose a general *quorum-based* model motivated by the *client-server* energy cost model of [16].

Typically, each operation of a replica control protocol is carried out in two phases: (1) sending the request by a *client* to all the *members* (replicas) of a given quorum, (2) receiving the response by a *client* from all the *members* (replicas) of the quorum. Note that upon receiving a request, every replica accesses its local storage medium to perform the desired operation (read a data or write to a data).

The energy consumed by a *replication-based* system, when performing a read or write operation, is modeled by summing (1) the energy consumed by the operation *initiator's* (client) component and connector as well as, (2) the energy consumed by the corresponding *quorum members'* (replicas) components, connectors and storage media.

In this paper, we assume that a client consists of a component (denoted by *Client*) and a connector (denoted by *Stub*) whereas a replica is composed of a component (*Server*), a connector (*Skeleton*) and a storage medium as shown in Figure 1, which illustrates the interactions between a client and a replica to perform a read operation. It is worthwhile to note that components and connectors are enforced as separate processes.

Next, we first introduce the energy costs at the client and server sides respectively, and then give the general *quorum-based* energy costs of read and write operations.

3.1 Client-side Energy Cost

In this section, we give the energy costs of the client-side *component* as well as *connector* induced when performing read and write operations of a replica control protocol.

3.1.1 Energy Cost of the Component

The energy cost of the *client-side component* when performing a read or write operation is modeled by summing the energy costs due to (1) executing some algorithm (E_{comp_logic}), (2) sending a read or write operation request to the connector (E_{toConn}) and (3) receiving its response from the connector ($E_{fromConn}$). These energy costs are represented by

the following equation:

$$EC_{client}(comp) = E_{comp\_logic} + E_{toConn} + E_{fromConn} \quad (1)$$

3.1.2 Energy Cost of the Connector

The energy cost of the *client-side connector* is modeled by summing the energy costs due to (1) receiving an operation request from the component ($E_{fromComp}$), (2) sending this request to all the members of a given quorum, (3) receiving the responses from all the members of the quorum, and (4) sending a response to the component (E_{toComp}). These energy costs are represented by the following equation:

$$EC_{client}(conn) = E_{conn\_logic} + E_{comm} \quad (2)$$

where E_{conn_logic} represents the energy cost of services that a connector can provide (marshal/unmarshal requests and responses, creation and management of connections), whereas E_{comm} represents the energy cost of exchanging data both locally and remotely such that:

$$\begin{aligned} E_{conn\_logic} &= E_{conversion} + E_{facilitation} \\ E_{comm} &= E_{commWithComp} + E_{remoteComm} \end{aligned}$$

Furthermore, the parameters of the above two equations are given by:

$$\begin{aligned} E_{conversion} &= (qSize) \times (E_{marshal} + E_{unmarshal}) \\ E_{facilitation} &= qSize \times E_{remoteConnect} \\ E_{commWithComp} &= E_{fromComp} + E_{toComp} \\ E_{remoteComm} &= (qSize) \times [\ (tSize \times tEC + tS) \\ &\quad +(rSize \times rEC + rS)\ ] \end{aligned}$$

such that $qSize$ denotes the size of the read or write quorum, $tSize$ ($rSize$) represents the size of the transmitted request (response), tEC (rEC) denotes the energy cost of transmitting a request (response), and tS (rS) represents constant energy overhead associated with channel acquisition. On the other hand, $E_{fromComp}$ and E_{toComp} represent the energy costs due to receiving a read or write operation request and sending its response to the component. Finally, $E_{remoteConnect}$ denotes the energy cost of creating and managing remote connections.

It is worthwhile to note that, if the component and connector are implemented as a single process, then E_{toConn}, $E_{fromConn}$, $E_{fromComp}$ and E_{toComp} have a value of zero.

3.2 Replica-side Energy Cost

In this section, we give respectively the energy costs of the replica-side *connector*, *component* as well as *storage medium* induced when performing read and write operations of a replica control protocol.

3.2.1 Energy Cost of the Connector

The energy cost of the *replica-side connector* is modeled by summing the energy costs due to (1) receiving an operation request from the client-side connector, (2) sending this request to the component (E_{toComp}), (3) receiving the response from the component ($E_{fromComp}$), and (4) sending a response to the client-side connector. These energy costs are represented by the following equation:

$$EC_{replica}(conn) = E_{conn\_logic} + E_{comm} \quad (3)$$

where E_{conn_logic} and E_{comm} have the same definitions as in (2) such that:

$$\begin{aligned} E_{conn\_logic} &= E_{conversion} + E_{facilitation} \\ E_{comm} &= E_{commWithComp} + E_{remoteComm} \end{aligned}$$

Furthermore, the parameters of the above two equations are given by:

$$\begin{aligned} E_{conversion} &= E_{marshal} + E_{unmarshal} \\ E_{facilitation} &= E_{remoteConnect} \\ E_{commWithComp} &= E_{fromComp} + E_{toComp} \\ E_{remoteComm} &= [\ (tSize \times tEC + tS) \\ &\quad +(rSize \times rEC + rS)\ ] + E_{buffer} \end{aligned}$$

where E_{buffer} represents the energy cost of buffering the read or write request whereas the remaining other parameters have the same definitions as above.

3.2.2 Energy Cost of the Component

The energy cost of the *replica-side component* is modeled by summing the energy costs due to (1) receiving an operation request from the connector ($E_{fromConn}$), (2) executing some algorithm (E_{comp_logic}), (3) sending a read or write operation request to the storage medium (E_{toDisk}), (4) receiving its response from the storage medium ($E_{fromDisk}$) and (5) sending a response to the connector (E_{toConn}). These energy costs are represented by the following equation:

$$\begin{aligned} EC_{replica}(comp) &= E_{fromConn} + E_{comp\_logic} + E_{toDisc} \\ &\quad +E_{fromDisc} + E_{toConn} \end{aligned} \quad (4)$$

Finally, we use $EC_{replica}$ to denote:

$$EC_{replica} = EC_{replica}(comp) + EC_{replica}(conn) \quad (5)$$

3.2.3 Energy Cost of the Storage Medium

The energy cost of the *replica-side storage medium* is modeled by summing the energy costs due to (1) receiving an operation request from the component ($E_{fromComp}$), (2) executing the desired operation (reading from the disk or writing to the disk), and (3) sending a response to the component (E_{toComp}). Since writing to the disk might have different energy cost than reading from the disk, then the above energy costs are represented by the following two equations:

$$EC_{read}(medium) = E_{fromComp} + Size_{rd} \times EC_{rd} + E_{toComp} \quad (6)$$

$$EC_{write}(medium) = E_{fromComp} + Size_{wt} \times EC_{wt} + E_{toComp} \quad (7)$$

where $Size_{rd}$ ($Size_{wt}$) and EC_{rd} (EC_{wt}) denote respectively the size of read (write) operation and the energy cost of accessing the storage medium to perform a read (write) operation request.

3.3 General Quorum-based Energy Costs

In this section, we give the general *quorum-based* energy costs of read and write operations of a replica control protocol. These energy costs are modelled by summing (1) the energy costs of the client-side component and connector, and (2) the energy costs of the replica-side component, connector and storage medium.

3.3.1 Read Operation

The overall energy cost due to executing a read operation request on a read quorum of size $rqSize$ is given by:

$$\begin{aligned} EC_{Read} &= rqSize \times [EC_{replica} + EC_{read}(medium) + \\ &\quad E_{marshal} + E_{unmarshal} + E_{remoteConnect} + \\ &\quad (tSize \times tEC + tS) + (rSize \times rEC + rS)] + \\ &\quad EC_{client}(comp) + E_{commWithComp} \end{aligned} \quad (8)$$

We rewrite the above equation in the following form:

$$EC_{Read} = rqSize \times k_r + c \quad (9)$$

where k_r and c represent energy related constants.

3.3.2 Write Operation

The overall energy cost due to executing a write operation request on a write quorum of size $wqSize$ is given by:

$$\begin{aligned} EC_{Write} &= wqSize \times [EC_{replica} + EC_{write}(medium) + \\ &\quad E_{marshal} + E_{unmarshal} + E_{remoteConnect} + \\ &\quad (tSize \times tEC + tS) + (rSize \times rEC + rS)] + \\ &\quad EC_{client}(comp) + E_{commWithComp} \end{aligned} \quad (10)$$

We rewrite the above equation in the following form:

$$EC_{Write} = wqSize \times k_w + c \quad (11)$$

where k_w and c represent energy related constants.

3.4 Overview

As we can notice from equations (9) and (11) that the *quorum sizes* ($rqSize$ or $wqSize$) play a major role in the overall energy cost computation of replication-based systems.

4. THE ARBITRARY TREE PROTOCOL

The *Arbitrary Tree* protocol was proposed by [3], which assumes that replicas of the system are organized logically into *any tree* structure where its nodes can be either *logical* or *physical*. Unlike a physical node which corresponds to a replica of the system, a logical node is used only to preserve the tree structure. Two new notions of *physical and logical levels* were introduced such that a physical level consists of at least one physical node whereas a logical level has all of its nodes logical.

Basically, a write quorum is composed of all physical nodes of a single physical level of the tree whereas a read quorum consists of any single physical node of every physical level of the tree.

The structure of the tree can be configured based on the frequencies of read and write operations. For instance, if write operations dominate in the system, then as many physical levels are added to the tree as possible. On the other hand, if the system is *mostly-read*, then all replicas are arranged into one and only one physical level. Furthermore, when read and write operations happen in equiprobable frequencies, an algorithm was introduced which configures the tree structure in such a way that its write operations induce on the system a load of $\frac{1}{\sqrt{n}}$ with a quorum size of $\sqrt{n}$, which are lower than previously proposed tree replication protocols. It was shown that a comparable write availability was preserved with respect to the state-of-the-art tree replication protocols. On the other hand, its read operations have a quorum size of $\sqrt{n}$ which is lower than the previously proposed tree replication protocols with comparable load and availability.

It is worthwhile to note that definitions related to the load and system models can be found in the preliminaries section of [3].

4.1 The Proposed Algorithm

In order to obtain satisfactory results both for read and write operations in terms of the quorum size, availability and load induced on the system, the proposed Algorithm 1 of [3] constructs the tree structure in the following manner:

- Sets the root of the tree to be logical.
- Sets the number of physical levels as well as the height of the tree to be $\sqrt{n}$.
- Arranges 4 physical nodes (replicas) at the 1st seven physical levels of the tree.
- Arranges $\frac{n-28}{\sqrt{n}-7}$ physical nodes (replicas) at every remaining physical level of the tree by obeying the assumption 3.1 of [3].

5. ENERGY-EFFICIENT APPROACH

5.1 Motivation

In contrast to previous replication protocols, in this paper, we suggest to compute the *energy cost* of the read and write operations of replica control protocols using the model of section 3. Moreover, we make use of the protocol defined in [3] (see section 4) in order to study the case of the proposed *Algorithm 1* (see section 4.1) by taking into account the system's *energy consumption*.

In short, the read operation of [3] is carried out on any replica of every level of the structure whereas its write operations are performed on all replicas of a single level of the structure. It is worthwhile to note that the proposed algorithm of [3] takes into consideration large number of replicas ($n > 64$) and provides acceptable results for both read and write operations in terms of the quorum size, availability and system load.

In order to *reduce* the energy consumption of Algorithm 1 of [3], we propose to *turn off* a significant amount of replicas of the system and to *logically reorganize* the other replicas into a new logical rectangle structure. Note that, the logical transformation from tree structure into rectangle does not necessitate the reconfiguration of the protocol of [3] and ensures overall *data consistency* among replicas of the system.

Next, we introduce the new proposed rectangle structure and then show the way we transform from tree into this structure.

5.2 Rectangle Structure

We propose to *rearrange* the replicas, which are organized logically into an arbitrary tree structure in [3], into a rectangle structure of height $h > 1$ and width $w > 1$.

Given a set of replicas organized logically into a rectangle structure of height $h > 1$ and width $w > 1$, the read

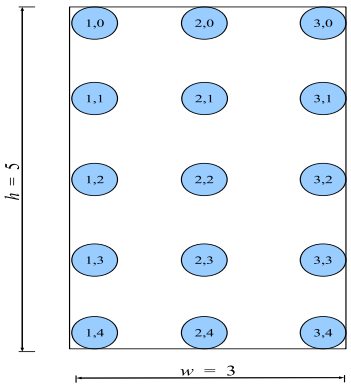
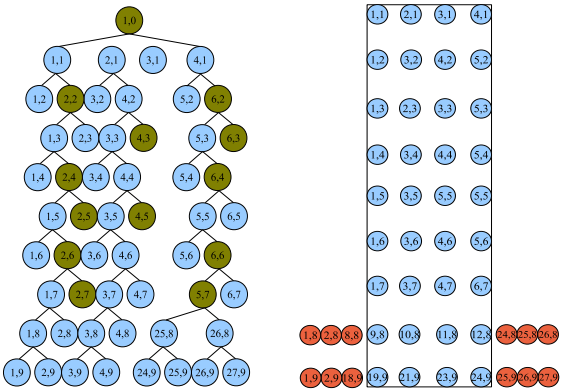


Figure 2: An example illustrated by the pair (replica number,level) of the rectangle structure for $n = 15$ replicas.



(a) Tree (b) Rectangle

Figure 3: Transformation of the tree structure into a rectangle structure.

operation of this structure using protocol of [3] has:

$$\begin{aligned}
&\text{A quorum size of } RD_{cost} &&= h\\
&\text{An availability of } RD_{av}\ (p) &&= \prod_{i=0}^{h-1}(1-(1-p)^w)\\
&\text{An optimal system load of } \mathcal{L}_{RD} &&= \frac{1}{w}\\
&\text{An energy cost of } EC_{Read} &&= h \times k_r + c
\end{aligned}$$

On the other hand, the write operation of such a structure using protocol of [3] has:

$$\begin{aligned}
&\text{A quorum size of } WR_{cost} &&= w\\
&\text{An availability of } WR_{av}\ (p) &&= 1-\prod_{i=0}^{h-1}(1-p^w)\\
&\text{An optimal system load of } \mathcal{L}_{WR} &&= \frac{1}{h}\\
&\text{An energy cost of } EC_{Write} &&= w \times k_w + c
\end{aligned}$$

It is important to note that the availability computations are carried out by assuming that every replica is independently available with a probability $p > 0.5$. Also, for the optimal load computation of the operations, interested readers can refer to the *Appendix* of [3].

Figure 2 illustrates an example of a rectangle structure composed of 15 replicas where each replica is denoted by the pair (replica number,level). As we will see later that such a structure has comparable characteristics in terms of the quorum size, availability and system load of its read and write operations with respect to those of *Algorithm 1* of [3], however with much reduced overall energy consumption.

5.3 Transformation

In this section, we demonstrate the way in which we transform the tree structure of Algorithm 1 of [3] into the rectangle one.

Since the original algorithm proposes to set 4 replicas at the first seven (physical) levels of the tree (see section 3 of [3] for more details), then we set the width of the rectangle to have a value of $w = 4$. Moreover, the height of the rectangle is set to have a value of $h = \sqrt{n}$, where n denotes the number of replicas of the initial tree structure. By setting the values of w and h as mentioned above to construct the rectangle structure, we propose to switch off any $(\frac{n-28}{\sqrt{n}-7} - 4)$ replicas at every physical level (other than the first seven ones) of the tree structure constructed using Algorithm 1 of [3].

Figure 3 illustrates an example of transforming a system composed of $n = 81$ replicas, from a tree structure of Algorithm 1 of [3] into the rectangle one. Note that not all replicas are demonstrated in both subfigures (for space considerations) and that only the blue circles represent replicas of the system which are denoted by the pair (replica number,level). In Figure 3(b), the red circles represent the replicas that have been switched off from the original tree structure of Figure 3(a).

We can notice from Tables 1 and 2 that we preserved comparable characteristics in terms of quorum size, availability and system load of read and write operations, while *reducing the energy cost* of write operations as well as *saving the energy consumption of* 45 machines.

Finally, it is important to note that, the load of the write operation increased when transforming from tree structure to rectangle, due to the fact that the protocol considers the switched off machines as crashed, and as a matter of fact no more write operations can take place on the levels where the switched off machines are found (in our example, these are levels 8 and 9 of Figure 3(b)). However, whenever additional replicas are needed, these turned off machines can be switched on anytime and then the protocol returns to its proper behaviour.

5.4 Discussion

Isn't it possible to simply switch off every replica, found at the physical levels other than the first seven ones, of the tree structure of Algorithm 1 of [3] without having to transform it into the rectangle structure? To answer to this question, let us assume the following scenario from the example given in section 5.3: suppose that before taking the decision of shutting down all replicas of levels 8 and 9 of Figure 3(a), a write operation took place which updated the data of every replica found at level 9 of the given tree. Afterwards, we decided to switch off all the replicas of levels 8 and 9 in order to save energy. Hence two problems arise: (1) no more read operations can take place, because according to the protocol of [3], there exists at least one whole level of the tree whose replicas are all crashed, (2) if we suppose that we can bypass

Table 1: Read Operation of 81 replicas

	Tree	Rectangle
RD_{cost}	9	9
RD_{av} (0.7)	0.94	0.92
$\mathcal{L}_{RD}$	0.25	0.25
$EC_{Read}(min)$	$9k_r + c$	$9k_r + c$
$EC_{Read}(avg)$	$9k_r + c$	$9k_r + c$
$EC_{Read}(max)$	$9k_r + c$	$9k_r + c$

Table 2: Write Operation of 81 replicas

	Tree	Rectangle
WR_{cost}	9	4
WR_{av} (0.7)	0.85	0.915
$\mathcal{L}_{WR}$	0.111	0.142
$EC_{Write}(min)$	$4k_w + c$	$4k_w + c$
$EC_{Write}(avg)$	$9k_w + c$	$4k_w + c$
$EC_{Write}(max)$	$27k_w + c$	$4k_w + c$

the above mentioned problem and we succeeded to read a data, then that read operation will return a stale copy of the data since the replicas that hold the most recent value have been shut down. For these two reasons, it is necessary to perform the proposed logical transformation in order to *ensure data consistency* among the replicas and hence to achieve one-copy equivalence. It is worthwhile to note that the transformation into the rectangle structure is *logical* and it does not need to be propagated to the whole system i.e. no reconfiguration is needed.

6. COMPARISON

In this section, we compare the *tree structure* constructed using Algorithm 1 of [3] with the proposed *rectangle structure* by taking into account each one's quorum size, availability, system load as well as the energy cost of read and write operations. For this purpose, we set up several configurations composed of $n = 75$, 100, 150 and 200 replicas respectively. For the availability computations, we suppose that every replica is independently available with a probability $p = 0.7$. Figures 4 and 6 are presented in logarithmic

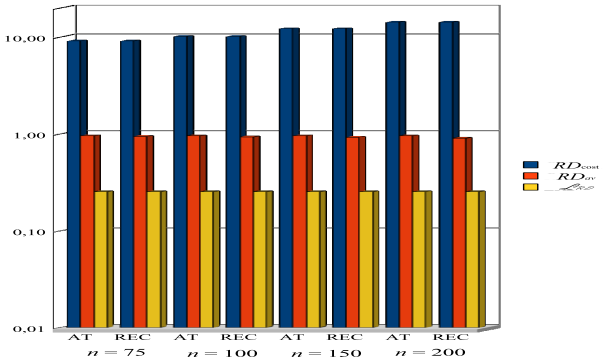


Figure 4: Read operation's quorum size (RD_{cost}), availability (RD_{av}) and system load ($\mathcal{L}_{RD}$) of Tree (AT) and Rectangle (REC) structures.

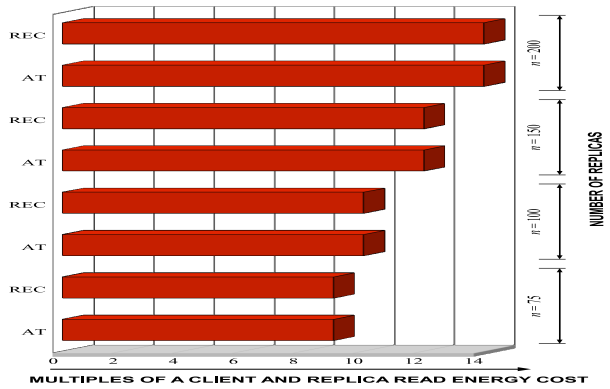


Figure 5: Read operation's energy cost of Tree (AT) and Rectangle (REC) structures.

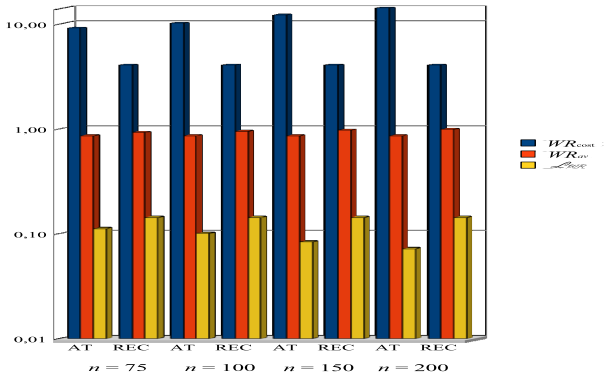


Figure 6: Write operation's quorum size (WR_{cost}), availability (WR_{av}) and system load ($\mathcal{L}_{WR}$) of Tree (AT) and Rectangle (REC) structures.

scales in order to be able to map the *quorum size* (greater than 1), *availability* (between 0 and 1) and *system load* (between 0 and 1) into a single figure. Furthermore, the abscissa of these figures denotes the number of replicas of each structure, whereas the ordinate refers to the values of above mentioned three metrics. Figures 5 and 7 represent respectively the multiples of read and write energy costs of a system composed of a single client and replica.

6.1 Read Operation

Figure 4 illustrates the quorum size (RD_{cost}), availability (RD_{av}) and system load ($\mathcal{L}_{RD}$) of read operations of both *tree* (AT) and *rectangle* (REC) structures. Figure 5 gives the energy cost of the read operations for both structures in terms of multiples of a single client and replica read energy cost. We can notice in these two figures that both structures have quite identical characteristics in terms of the quorum size, availability, system load and energy cost. As we can see in Table 3 that, with the rectangle structure, we achieve the same characteristics of the tree structure while *using fewer number of replicas* (reduced overall energy consumption).

6.2 Write Operation

Figure 6 illustrates the quorum size (WR_{cost}), availability (WR_{av}) and system load ($\mathcal{L}_{WR}$) of write operations of both

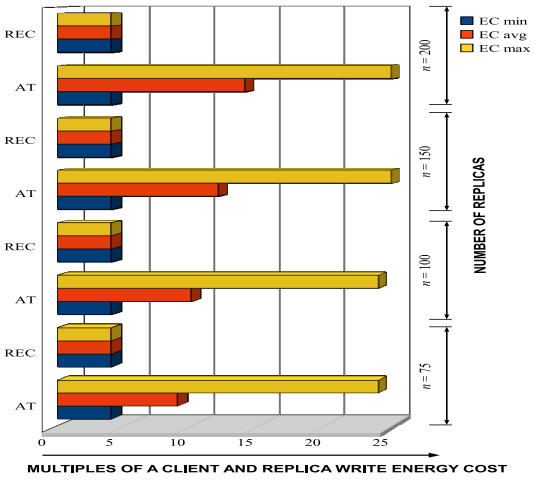


Figure 7: Write operation's energy cost of Tree (AT) and Rectangle (REC) structures.

Table 3: Number of machines saved from energy consumption.

Number of replicas	Tree	Rectangle
$n = 75$	0	39
$n = 100$	0	60
$n = 150$	0	102
$n = 200$	0	144

tree (AT) and *rectangle* (REC) structures. We can notice in this figure that the rectangle (REC) structure has smaller quorum sizes than the tree (AT) one due to the fact that at every level there are always 4 replicas. On the other hand, both structures have quite comparable availability for their write operations. Finally, we can notice that, the tree structure has a smaller system load ($\frac{1}{\sqrt{n}}$) than rectangle (always $\frac{1}{7}$) and that such a load diminishes as the number of replicas of the system becomes larger. It is worthwhile to note that, we can sacrifice load which is at the worst case around 14% higher than the tree structure, however as we will see in Figure 7, with much *less energy cost* and *reduced overall energy consumption*.

Figure 7 gives the energy cost of the write operations for both structures in terms of multiples of a single client and replica write energy cost. We can notice in this figure that the write operation of the rectangle structure has a much fewer *energy cost* than that of the tree structure. Furthermore, we can observe that, in the worst case, the write operation of the tree structure has an energy cost more than 5 times greater than that of the rectangle structure. Finally, it is important to note that, unlike the rectangle structure, the average energy cost of write operations of the tree structure increases as the number of replicas of the system increments.

6.3 Overview

Table 3 indicates the number of turned off machines for the corresponding number of replicas. As we can notice, by a simple logical structural transformation from tree to rectangle, we are capable of achieving the same characteristics (although with slight sacrifices in load of write operations) of the tree structure of Algorithm 1 of [3], yet with much reduced *energy cost* for write operations as well as reduced overall *energy consumption*.

7. CONCLUSION AND PERSPECTIVES

In a large *replication-based* system, multiple copies of data must be kept synchronized by means of replica control protocols. On the other hand, until lately, no attempts have been made in order to compute the energy cost of read and write operations of the replication protocols. In this paper, we suggested to compute the energy cost of read and write operations of these protocols by proposing a new *quorum-based energy cost* model. Also, we introduced an approach to reduce the energy cost as well as consumption of the tree-structured protocol of [3]. We showed that by shutting down a significant amount of replicas and by performing a logical structural transformation, our approach has fewer energy cost for its write operations than that of [3] while conserving its major characteristics. Also, the read operations of our approach has the same energy cost as that of [3], however with much reduced overall energy consumption. It is important to note that our proposal does not require the reconfiguration of the protocol whenever energy efficiency requirements change in the system.

As a future work, it could be interesting to add the *energy cost* factor as a metric in order to compare the state-of-the-art replica control protocols by taking into account the quorum size, availability, system load as well as energy cost of their read and write operations.

8. ACKNOWLEDGEMENT

The research leading to these results has received funding from the European Community's Seventh Framework Programme in the context of the FIT4Green project (grant agreement no. 249020) and the EuroNF Network of Excellence (grant agreement no. 216366).

9. REFERENCES

[1] D. Agrawal and A. E. Abbadi. The tree quorum protocol: An efficient approach for managing replicated data. In *Proceedings of the sixteenth international conference on Very Large Databases*, pages 243–254, 1990.

[2] D. Agrawal and A. E. Abbadi. An efficient and fault-tolerant solution for distributed mutual exclusion. *ACM Transactions on Computer Systems*, 9(1):1–20, 1991.

[3] J. P. Bahsoun, R. Basmadjian, and R. Guerraoui. An arbitrary tree structured replica control protocol. *The 28th International Conference on Distributed Computing Systems*, pages 502–511, 2008.

[4] P. Bernstein and N. Goodman. An algorithm for concurrency control and recovery in replicated distributed databases. *ACM Transactions on Database Systems*, 9(4):596–615, 1984.

[5] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle. Managing energy and server resources in hosting centers. *ACM SIGOPS Operating Systems Review*, 35(5):103–116, 2001.

[6] S. Y. Cheung, M. H. Ammar, and M. Ahamad. The grid protocol: A high performance scheme for maintaining replicated data. *IEEE Transactions on Knowledge and Data Engineering*, 4(6):438–445, 1990.

[7] S. C. Choi, H. Y. Youn, and J. S. Choi. Symmetric tree replication protocol for efficient distributed storage system. *International Conference on Computational Science*, pages 474–484, 2003.

[8] T. Heath, A. P. Centeno, P. George, L. E. S. Ramos, Y. Jaluria, and R. G. Bianchini. Mercury and freon: temperature emulation and management for server systems. In *Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*, pages 106–116, 2006.

[9] K. H. Kim, R. Buyya, and J. Kim. Power aware scheduling of bag-of-tasks applications with deadline constraints on dvs-enabled clusters. In *Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid*, pages 541–548, 2007.

[10] H. Koch. An efficient replication protocol exploiting logical tree structures. *The 23rd Annual International Symposium on Fault-Tolerant Computing*, pages 382–391, 1993.

[11] A. Kumar. Hierarchical quorum consensus: A new algorithm for managing replicated data. *IEEE Transactions on Computers*, 40(9):996–1004, 1991.

[12] M. Maekawa. A $\sqrt{n}$ algorithm for mutual exclusion in decentralized systems. *ACM Transactions on Computer Systems*, 3(2):145–159, May 1985.

[13] M. Naor and A. Wool. The load, capacity, and availability of quorum systems. *SIAM Journal on Computing*, 27:214–225, 1998.

[14] D. Peleg and A. Wool. The availability of quorum systems. *Information and Computation*, 123(2):210–223, 1995.

[15] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath. Dynamic cluster reconfiguration for power and performance. In *Compilers and operating systems for low power*, pages 75–93, 2003.

[16] C. Seo, G. Edwards, D. Popescu, S. Malek, and N. Medvidovic. A framework for estimating the energy consumption induced by a distributed system's architectural style. In *Proceedings of the 8th international workshop on Specification and verification of component-based systems*, 2009.

[17] A. Silberschartz, H. F. Korth, and S. Sudarshan. *Database System Concepts*. McGraw-Hill, fourth edition, 2002.

[18] R. H. Thomas. A majority consensus approach to concurrency control for multiple copy databases. *ACM Transactions on Database Systems*, 4(2):180–209, June 1979.

[19] N. Vasic, M. Barisits, V. Salzgeber, and D. M. Kostic. Making cluster applications energy-aware. In *Proceedings of the 1st workshop on Automated control for data centers and clouds*, pages 37–42, 2009.