Majority Quorum Protocol Dedicated to General Threshold Schemes

Theodore J. R. RELAZA
TOULOUSE UNIVERSITY - IRIT
Email: relaza@irit.fr

Jacques Jorda
TOULOUSE UNIVERSITY - IRIT
Email: jorda@irit.fr

Abdelaziz M'zoughi
TOULOUSE UNIVERSITY - IRIT
Email: mzoughi@irit.fr

***Abstract*—In this paper, we introduce a majority quorum system dedicated to p-m-n general threshold schemes where p, n and m are respectively the minimal number of chunks that provide some information (but not necessarily all) on the original data, the total number of nodes in which the chunks of an object are stored and the minimal number of nodes needed to retrieve the original data using this protocol. In other words, less than p chunks reveal absolutely no information about the original data and less than m chunks can't reconstruct the original data. The p-m-n general threshold schemes optimize the usage of storage resources by reducing the total size of data to write and ensure fault-tolerance up to $(n - m)$ nodes failure. With such a data distribution, a specific value of m can be set to have a good tradeoff between resources utilization and fault-tolerance. The only drawback of such schemes is the lack of any consistency protocol.**

If fact, consistency protocols like classical majority quorum are based on full replication. To successfully read or write a data using the majority quorum protocol, an absolute majority of replicas must be read / written correctly. This condition ensures that any read and write operations will contain at least one common replica, which guarantees their consistency. However, when a threshold scheme is used, an adaptation is needed. In fact, classical majority quorum protocol can no longer ensure that m chunks will have the latest version when $\lfloor \frac{n}{2} \rfloor + 1 < m \leq n$. In this paper, we introduce a new majority quorum protocol dedicated to general threshold schemes. As for the classical majority quorum protocol, the complexity of the quorum size of our protocol is $\mathbf{O}(n)$ but the utilization of storage resources is greatly optimized.

***Keywords*-Data availability; error codes; coherency; quorums;**

I. INTRODUCTION

Data replication is heavily used to ensure data availability in distributed storage systems. Both grid storage systems (like GFarm or XtreemFS) and cloud storage systems (like Ceph or GlusterFS) use this mechanism to handle network and / or nodes failures. However the replication introduces an additional overhead due to replicas management and the consistency of replicated data must be ensured. The simplest solution to maintain the consistency of the replicas is the ROWA (*Read-One Write-All*) protocol [13] a single node is enough to read the data while all nodes are required for a successful write operation. The strong point of the ROWA protocol is the reliability of read operations. However, the main drawbacks of this protocol are the large overhead and the lack of fault-tolerance for write operations: all replicas must be written for each write operation, and if only one node fails to write successfully, the whole operation fails. Moreover, full replication involves an under-utilization of storage resources making this solution of high cost when large data should be stored. For data consistency protocols, the goal is to reduce the minimum number of nodes (called size of the write quorum) needed to write data while maintaining a high level of availability, to make the protocol fault-tolerant and to guarantee the consistency for each replica of the same data. For data replication protocol, the goal is to reduce the total space used while maintaining a high level of availability.

The existing data consistency protocols are almost all dedicated to full replication. Although the quorum protocol allows to reduce the number of nodes required to write data, the problem due to full replication remains. To minimize the usage of storage resources, we decided to use $p-m-n$ general threshold schemes because of the adaptability of the algorithm to users' needs. Both availability and high storage resources utilization may be ensured with adequate values of p, m and n. We will propose an adaption of classical majority quorum protocol in the general threshold schemes context.

The remaining of the paper is organized as follows. Section 2 reviews the context and related work. We describe the classical data distribution and data consistency protocols in section 3. Numerical evaluations and analysis are given in section 4 and we conclude in section 5.

II. CONTEXT AND RELATED WORK

We are first going to present classical way to distribute data to ensure availability, presenting existing solutions used either at nodes or disks levels. We will then introduce the most commonly used data consistency protocols, and we will conclude by the general context of our study.

A. Data distribution

The mostly used solution to increase data availability and reliability is the full replication. This data distribution implies the higher availability possible, but incurs an excessive usage of system resources [12]. It also increases a system's costs by 40% to 60% [8]. Moreover, the write operations in full replication cause a scalability problem in the bandwidth usage [12].

The erasure resilient codes (ERC-codes) are beginning to move from disks to distributed storage systems. However, almost all studies focus on in-place updates and I/Os / network usage optimization on nodes failure. To achieve these goals, data are often considered as (nearly) immutable: updates are done by writing new data - the old blocks being cleaned using a background garbage collector process. Moreover, the data consistency is either not discussed [10] or the protocols used are not strict: for example, a "reasonably strong consistency" may be used [3]where a read concurrent to write operations returns a non deterministic value. To our best knowledge, the only implementation of a strong consistency protocol seals the writes on failures to guarantee the consistency [5]. However, these solutions are not adequate for some kind of storage. For example, data disks of users' virtual machines in a virtualization context need a strong consistency of the underlying files to guarantee a proper execution of applications.

In order to generalize our discussion, we consider General Threshold Schemes [7] as they can represent a broad class of data

distribution. This type of protocol may be particularly suited to the context of cloud storage where many levels of quality of services are often proposed. The flexibility is one of the strengths of the general threshold schemes: the p, m and n parameters may be set according to the storage system requirements (cost, performance, etc.).

B. Data consistency protocol

The most basic replica control is the ROWA protocol. The non fault-tolerance and the high cost for write operations are the major drawbacks of the ROWA protocol [13]. Some protocols have been developed to ensure not only data consistency but also concurrency control over replicated data (e.g., quorum consensus protocol [9]). But with these protocols, the communication cost is high (O(N)) due to the large quorum size.

Facing with this dilemma, many protocols impose a logical structure on the infrastructure [1], [6], [11]. These structures are logical and do not tie in the actual physical structures of the infrastructure containing the nodes storing the replicas [1]. The Tree Quorum protocol [1], [2] requires a quorum of size O($\log N$), in which the nodes are structured to form a tree. The Hierarchical Quorum Consensus [11] is based on logically organizing the data replicas into a multilevel tree. With this protocol, the quorum size required is O($N^{0.63}$). The Triangular Mesh protocol [6] uses a quorum of size O($\sqrt{N}$), in which the nodes are structured to form a triangle. In the General Hybrid Data Replication protocol [4], organizes replicas in a logical trapeze.

C. Context of our work

To reduce the storage space needed to ensure data availability, we focus on general threshold schemes to distribute data among nodes . All data consistency protocols described above being based on full replication, we need to modify them to make them fit into our distribution model. In this paper, we study the data consistency in the presence of general threshold schemes using Majority Quorum Protocol. Actually, majority quorum protocol can only ensure that $\lfloor \frac{n}{2} \rfloor + 1$ chunks (absolute majority) have the latest version while m chunks are required by $p-m-n$ general threshold schemes to reconstruct the original data (with $1 \leq m \leq n$). Therefore, $\lfloor \frac{n}{2} \rfloor + 1$ is not enough to reconstruct the original data using a threshold scheme when $\lfloor \frac{n}{2} \rfloor + 1 < m \leq n$. Thus we are going to show how to modify the existing algorithm and what performance can be expected from this new storage system.

III. MODEL

The storage system we are going to study is made of two parts. The first part is the data distribution method which ensures the data slicing, the redundancy schemes and the data recovery mechanisms.The second part is the replica control protocol which goal is to maintain the consistency of the replicas. Our replica control protocol will be based on the Majority Quorum Protocol, adapted to the chosen data distribution solution.

A. General Threshold Schemes

In our storage system, we use general threshold schemes to store the data. It is a flexible data distribution mechanism which can be adapted according to the users' objectives and constraints(e.g., storage cost, data availability or data confidentiality).A $p-m-n$ general threshold scheme divides data into n chunks such that any m of the chunks can reconstruct the original data and less than p chunks reveal absolutely no information about the original data. The p, m and n parameters must satisfy the following conditions: $1 \leq p \leq m \leq n$. Table I shows a list of specifics threshold scheme depending on the values of the parameters p and m.

Threshold scheme	Description
$1-1-n$	Full replication
$1-n-n$	Decimation (Striping)
$n-n-n$	Splitting (XORing)
$1-m-n$	Information Dispersal
$m-m-n$	Secret Sharing
$p-m-n$	Ramp Scheme

Table I
LIST OF SPECIFICS THRESHOLD SCHEME [7]

B. Data Consistency Protocol

In this section, we will define the read and write quorum of majority quorum protocol dedicated to general threshold schemes. A ***general threshold scheme compliant write quorum*** (WQ) is any set of $|WQ|$ nodes required to write data, where

$$|WQ| = \max(m, \lfloor \frac{n}{2} \rfloor + 1) \quad (1)$$

WRITE procedure in algorithm 1 describes the way write operations are implemented for a majority quorum protocol dedicated to threshold schemes. For any two quorum $WQ1$ and $WQ2$, the property $WQ1 \cap WQ2 \neq \emptyset$ is always verified.

A ***general threshold scheme compliant read quorum*** (RQ) is any set of $|RQ|$ nodes among which at least m nodes contain the latest version of the data, where

$$|RQ| = \max(m, \lceil \frac{n}{2} \rceil) \quad (2)$$

Given any read quorum RQ and write quorum WQ, they satisfy $RQ \cap WQ \neq \emptyset$. READ procedure in algorithm 1 describes the way write operations are implemented for a majority quorum protocol dedicated to threshold schemes. The write quorum and read quorum are described in algorithms later in this section. The symbols used in the following algorithms and their explanations are listed in Table II on page 2.

Symbol	Description
p, m, n	Threshold scheme parameters
x	Object x
id_x	ID of object x
r	Replica ID of object x
L_{node}	List of node in the system

Table II
SYMBOLS USED IN THE FOLLOWING ALGORITHMS

IV. ANALYSIS

In this part, we will evaluate the impact of each parameter on the availability, the storage space used and the confidentiality. We will model the availability using a probabilistic approach. Then we will show some numerical evaluations of our majority quorum protocol dedicated to general threshold schemes compared to the classical version dedicated to full replication.

Notations

- a: denotes node availability
- MQP-FR and MQP-GTS: refer to Majority Quorum Protocol in the Full Replication respectively the General Threshold Schemes context.
- w and r: denote size of the write respectively read quorum for the MQP-GTS context.
- $\Phi(i, j)$refer the probability that at least *i* nodes out of *j* would be available.

Algorithm 1 Write and read an object x

```
1: procedure WRITE(id_x, x, p, m, n, L_node)                ▷ Write n chunks
2:     T_chunks ←GENERATE-BLOCKS(p, m, n, x)  ▷ Generate n blocks from x
3:     v ←GET-HIGHEST-VERSION(id_x, n, m, L_node)
4:     S ← ∅
5:     size ← GET-WRITE-QUORUM-SIZE(n, m)
6:     for all chunk ∈ T_chunks do
7:         r ← ID of replica chunk
8:         node ←SELECT-NODE(id_x, r, L_node)
9:         write chunk in node
10:        if "write is OK" then
11:            S ← S ∪ {node}
12:            SETVERSION(id_x, node, v + 1)
13:        end if
14:    end for
15:    if |S| ≥ size then
16:        return OK                                  ▷ WRITE operation is done
17:    else
18:        for all node ∈ S do
19:            INVALIDATE-WRITE(id_x, node)
20:            SETVERSION(id_x, node, v)
21:        end for
22:        return NO OK                               ▷ WRITE operation failed
23:    end if
24: end procedure

25: procedure READ(id_x, p, m, n, L_node)                   ▷ Read the object x
26:    x ← ∅
27:    v ←GET-HIGHEST-VERSION(id_x, n, m, L_node)
28:    counter ← 0
29:    X_chunks ← ∅
30:    for all r ∈ replicas of object with ID id_x do
31:        node ←SELECT-NODE(id_x, r, L_node)
32:        get the version v_r of the replica r of object x stored in node
33:        if v = v_r then
34:            chunk ←read replica r of object ID id_x in node
35:            X_chunks(counter) ← chunk
36:            counter ← counter + 1
37:        end if
38:        if counter = m then
39:            x ← decode(id_x, p, m, n, X_chunks)
40:            break
41:        end if
42:    end for
43:    return x
44: end procedure
```

$$\Phi(i,j) \equiv \sum_{k=i}^{k=j} \binom{j}{k} a^k (1-a)^{j-k} \tag{3}$$

With no loss of generality, we assume that node availability is the same and equal to a for all nodes in the system, nodes fail independently of each other, each node stops on failure (fail-stop) and there is no failure on communication links.

A. Write availability

The ***write availability*** represents the probability that the data can be written into the system. The MQP-FR requires at least an absolute majority of nodes to validate the write operation. Then, at least $\lfloor \frac{n}{2} \rfloor + 1$ nodes out of n should be available. Whereas in the case of MQP-GTS the validation of chunks' write operation on at least w nodes out of n is required. Therefore,

$$P_{write} = \begin{cases} \Phi(\lfloor \frac{n}{2} \rfloor + 1, n) & \text{for MQP-FR} \\ \Phi(w, n) & \text{for MQP-GTS} \end{cases} \tag{4}$$

B. Read availability

The ***read availability*** represents the probability that the data can be read from the system. The MQP-FR requires at least $\lceil \frac{n}{2} \rceil$ nodes to validate the read operation. Using $\beta(i,j)$ to refer the probability that exactly i nodes out of j would be available and $\psi_i(k)$ to refer the probability that at least i nodes out of k nodes which are available during the current read operation was also available during the last valid write operation, we have:

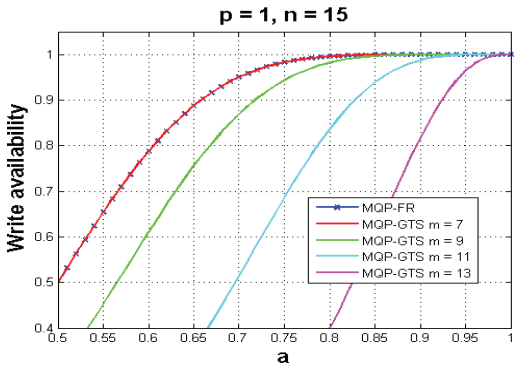

Figure 1. Write availability of MQP-FR and MQP-GTS as a function of the node availability a

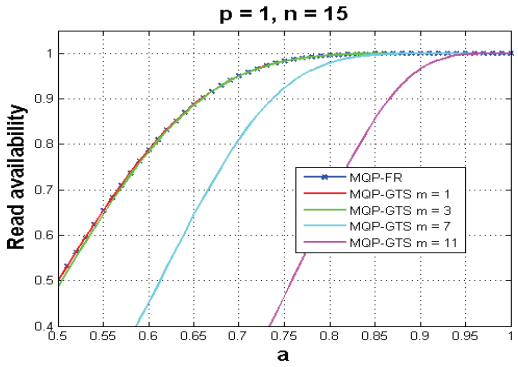

Figure 2. Read availability for MQP-FR and MQP-GTS as a function of the node availability a

$$P_{read} = \begin{cases} \Phi(\lceil \frac{n}{2} \rceil, n) & \text{for MQP-FR} \\ \sum_{k=r}^{k=n} \psi_m(k).\beta(k,n) & \text{for MQP-GTS} \end{cases} \tag{5}$$

C. Simulations

Simulation results (Figure 1-4) show the highlights of MQP-GTS compared to MQP-FR. For any m lower or equal to 8, these two protocols have the same write and read availability (igure 1 and Figure 2) but MQP-GTS allows to save more than 80% of storage space compared to MQP-FR (Figure 3). In Figure 3, storage space used by MQP-FR is taken as a reference to compute the percentage of storage space used by each protocol to store a data. The storage space used by MQP-GTS is inversely proportional to p (Figure 4). By contrast, increasing p increases the degree of confidentiality. For $n = 15$ and $a = 0.9$, using MQP-GTS instead of MQP-FR allows to save storage space up to about 85% while keeping the same performance.

Table III shows an example of numerical evaluations of write availability (P_{write}), read availability (P_{read}) and storage space used (SP_{used}) for a node availability greater than 0.9.

V. CONCLUSION

General threshold schemes give the theoretical framework to study data distribution in a parallel storage system. They allow to define precisely the way data is splited and distributed among nodes. Given three parameters, any choice can be made to balance

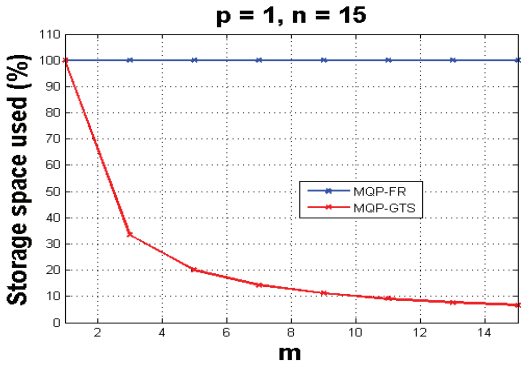

Figure 3. Percentage of storage space used in MQP-GTS compared to MQP-FR as a function of m

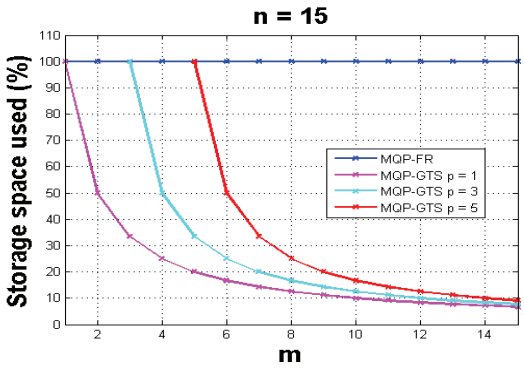

Figure 4. Percentage of storage space saved in MQP-GTS compared to MQP-FR as a function of m for different values of p

availability, cost and confidentiality. However, no data consistency protocol exists on such schemes.

Traditional consistency protocols only refer to full data distribution. In such contexts, each replica is identical to others and a single one is enough to retrieve the original data. Thus such protocols cannot be used with general threshold schemes.

In this paper, we have introduced an adaptation of the majority quorum protocol dedicated to general threshold schemes. We have detailed the read and write operations, and show the balance between availability and storage space saved. This gain is at no cost since the complexity of the algorithm is unchanged.

We are now going to explore more data consistency protocols in the context of general threshold schemes since the majority quorum protocol is one of the simpler and least efficient one.

a	type	P_{write}	P_{read}	SP_{used}
0.93	Full replication	0.999997	0.999997	100%
	Threshold scheme	0.999967	0.999602	11.11%
0.96	Full replication	0.999999	0.999999	100%
	Threshold scheme	0.999999	0.999988	11.11%

Table III
EXAMPLE OF NUMERICAL EVALUATIONS OF WRITE AVAILABILITY, READ AVAILABILITY AND STORAGE SPACE USED AS A FUNCTION OF NODE AVAILABILITY a AND THE DATA DISTRIBUTION PROTOCOL

REFERENCES

[1] D. Agrawal and A. El Abbadi. The tree quorum protocol : An efficient approach for managing replicated data. *Proceedings of the 16th VLDB Conference Brisbane, Australia*, 1990.

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

[3] Marcos K. Aguilera, Ramaprabhu Janakiraman, and Lixao Xu. Using erasure codes efficiently for storage in a distributed system. In *Proceedings of the 2005 International Conference onDependable Systems and Networks*, DSN'05, pages 336–345. IEEE, 2005.

[4] Masayuki Arai, Tabito Suzuki, and Mamoru Ohara. Analysis of read and write availability for generalized hybrid data replication protocol. *Proceedings of the 10th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC'04)*, 2004.

[5] Brad Calder et al. Windows azure storage: A highly available cloud storage service with strong consistency. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 143–157, New York, NY, USA, 2011. ACM.

[6] Ye-In Chang and Yao-Jen Chang. A fault-tolerant trangular mesh protocol for dirstibuted mutual exclusion. *Proceedings. Seventh IEEE Symposium on Parallel and Distributed Processing*, pages 694–701, 1995.

[7] Gregory R. Ganger, Pradeep K. Khosla, and Mehmet Bakkaloglu. Survivable storage systems. *DARPA Information Survivability Conference & Exposition II*, 2:184–195, 2001.

[8] G.A. Gibson. Redundant disks arrays: Reliable, parallel secondary storage. *PhD dissertation, Dept. Computer Science, UC Berkeley*, Apr 1991.

[9] David K. Gifford. Weighted voting for replicated data. *in Proc. of the 7th Symposium on Operating Systems Principles*, pages 150–159, 1979.

[10] Cheng Huang, Minghua Chen, and Jin Li. Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems. *Trans. Storage*, 9(1):3:1–3:28, March 2013.

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

[12] Gunnar Mathiason et al. Virtual full replication by adaptative segmentation. *13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 327–336, 2007.

[13] Michael Rabinovich and Edward D. Lazowska. An efficient and highly available read-one write-all protocol for replicated data management. *Proceedings of the Second International Conference on Parallel and Distributed Information Systems*, pages 56–65, 1993.