Received 13 March 2023, accepted 24 March 2023, date of publication 14 April 2023, date of current version 21 April 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3267106

RESEARCH ARTICLE

Optimized Proactive Recovery in Erasure-Coded Cloud Storage Systems

REKHA NACHIAPPAN, RODRIGO N. CALHEIROS, (Senior Member, IEEE), KENAN M. MATAWIE, AND BAHMAN JAVADI, (Senior Member, IEEE)
School of Computer, Data and Mathematical Sciences, Western Sydney University, Penrith, NSW 2751, Australia

Corresponding author: Rodrigo N. Calheiros (R.Calheiros@westernsydney.edu.au)

ABSTRACT Cloud data centers have started utilizing erasure coding in large-scale storage systems to ensure high reliability with limited overhead compared to replication. However, data recovery in erasure coding incurs high network bandwidth consumption compared to replication. Cloud storage systems also play an important role in the energy consumption of data centers. Heuristic proactive recovery algorithms select all data blocks from failure-predicted disk/machine and perform proactive replication that contributes to huge recovery bandwidth savings. However, they fail to optimize the selection. Optimization can further improve resource savings. To address this issue, we propose a recovery algorithm that applies minimization on data blocks selected for proactive replication by considering the necessary and appropriate constraints that are constructed based on the system's current network traffic and data duplication information. We evaluate the proposed algorithm using extensive simulations. Experiments show that the recovery algorithm reduces network traffic by 60% and storage overhead by 46% compared to the heuristic proactive recovery approach. Also, the proposed proactive recovery methods reduce the storage system's energy consumption by up to 52% compared to replication.

INDEX TERMS Cloud storage systems, data reliability, erasure codes, replication, energy consumption.

I. INTRODUCTION

Cloud storage system is an aggregation of individual hardware components that are subject to failure. The system has to deal with these hardware failures to guarantee the durability of data stored on these components. Cloud storage systems also have to handle software glitches, network outages, power outages, and machine reboots to ensure data availability. To enforce fault tolerance against failures, various data redundancy techniques are employed in cloud storage systems. The most common data redundancy techniques are replication and erasure coding [2]. Typical three-way replication replicates each data block in three different machines such that it can tolerate any 2 failures. Storage overhead is 3x for this three-way replication method. Replication is an inefficient method for today's cloud storage systems that store petabytes of data [16]. Erasure coding is a viable alternative to replication. The (n, k) erasure code divides data into k data blocks and stores them into n different machines along with $n-k$ parity blocks. The total of k original and $n-k$ parity blocks form a stripe such that it can tolerate any $n-k$ failures. Reed_Solomon code is one of the most popular erasure codes. It provides better reliability with less storage overhead compared to replication [2]. For example, the most popular (14,10) Reed_Solomon code can tolerate any 4 block failures and storage overhead is 1.4x. Many large-scale storage systems like Windows Azure Storage, Google's ColossusFS, Facebook's HDFS has adopted Reed-Solomon code [16], which provides significant cost savings with respect to storage.

In order to ensure data reliability, any failed data block in the cloud storage system has to be restored. The process of recovering failed data blocks is known as data recovery. In order to avoid unnecessary repairs of short-term transient node failures, data recovery is delayed for a certain amount of time. Google File System (GFS) delays recovery of unavailable nodes for 15 minutes [4]. An analysis in Facebook's 3000 machine production cluster revealed that in a day,

The associate editor coordinating the review of this manuscript and approving it for publication was Cristian Zambelli.

at least 20 machine failures activate data recovery, even after a delay is applied to eliminate unnecessary data recovery due to transient failures [3]. Due to network traffic issues, Facebook has applied the Reed-Solomon code for only 8% of their data. A considerable amount of research has focused on designing repair-efficient erasure codes. Several researchers [3], [9], [10], [11], [12], [17], [31] have proposed proactive recovery of erasure codes to reduce repair bandwidth and disk I/O. Few pieces of research concentrate on applying a delay in data recovery of erasure codes, to reduce repair bandwidth [12].

It has been estimated that if Facebook applied Reed-Solomon on 50% of their data, repair network traffic of the system would saturate their network links. Increased repair network traffic is one of the major performance issues of erasure coding which prevents it from being more pervasive in cloud storage systems.

Improving energy efficiency is another major challenge of cloud data centers. Storage systems consume up to 40% of a data center's total energy [26]. Read and write latency of the storage systems also reduces energy efficiency [23]. Since erasure coding reduces storage overhead significantly, it also reduces the energy consumption of storage systems. However, recovery network traffic in erasure code may jeopardize the energy savings of erasure code in terms of storage.

In a preliminary version of this paper [21], we proposed a cloud storage system that employs several proposed bandwidth-efficient proactive recovery methods. In the event of any disk/machine failure prediction, the client's durability, availability, and performance requirements are determined using a Service Level Agreement (SLA). Based on this information, the system selects an appropriate proactive recovery method. This heuristic proactive recovery technique selects a set of data blocks for proactive replication according to the selected recovery method. We observed that our recovery approach improves repair bandwidth efficiency and reduces network traffic in cloud storage systems with limited storage overhead compared to existing recovery approaches. The resource savings vary according to the selected proactive recovery technique. However, it does not take into account important system parameters like data duplication and the system's current network traffic during proactive recovery that may incur additional storage overhead and network bandwidth/traffic. This paper is the extended version of our previous paper [21] and the differences between this work and previous work are:

- We propose an optimization approach to further enhance the efficiency of proactive recovery methods proposed earlier in [21]. The optimization approach applies minimization on a set of data blocks that are selected for proactive replication by the aforementioned earlier heuristic approach. The Optimization approach takes into account the system's current network traffic and data duplication information.
- We propose an optimization algorithm that intends to limit proactive replications when the system's instantaneous network bandwidth reaches a certain limit and hence it minimizes bandwidth throttling. It also uses data duplication information to diminish the temporary storage and recovery bandwidth consumption due to unnecessary replication.
- We analyze the energy efficiency of proactive recovery methods. Activating proactive recovery in erasure coding reduces data transfers that can count toward energy savings. However, proactive recovery methods suggest additional temporary dedicated storage overhead that may increase system energy consumption. To analyze the energy consumption of storage systems, we estimate the energy consumption of storage and network devices, respectively. Using that we compare the system's consumption from replication, erasure coding, and various recovery approaches used in erasure coding.

II. RELATED WORK

Due to the recent advancements in the Internet of Things (IOT), Big Data applications demand petabytes of storage. Erasure code is becoming an important fault-tolerant method of industrial storage systems. Although it improves reliability with less storage overhead compared to replication, inefficient data reconstruction of erasure coding needs to be addressed.

Dimakis et al. [9] proposed regeneration codes that reduce network traffic by downloading small amounts of data from a higher number of nodes than the number of nodes involved in typical reconstruction. However, the exact repair of regeneration codes for several combinations of parameters remained unresolved. This was followed by several researchers [17], [18], [30] showing that the exact repair is possible for several parameters. The locally repairable code is another family of code proposed that reduces repair bandwidth. Other works [3], [10], [39] add local parity to reduce the number of data blocks accessed during reconstruction. This has the side effect of increasing storage overhead by 1.33x compared to Reed-Solomon [10].

TLRC [39] reduces the recovery overhead of distributed storage. However, storage overhead is almost the same as the methods proposed by Sathiamoorthy et al. [3] and Huang et al. [10], and the recovery performance is not better than proactive recovery. Hitchhiker code [11], built on top of Reed-Solomon code using a piggy-backing framework, reduces the network traffic by 35% while some encoding time overhead is incurred. Even though the above methods reduce repair network bandwidth/ traffic, none reduces recovery bandwidth as efficiently as replication.

Several works [12], [22], [27], [34], [35], [41] proposed system-level solutions like delaying data recovery, caching data read during recovery, parallel reconstructions, switching between two families of erasure codes, and proactive replication of data blocks. Silberstein et al. [12] suggest delaying the data repair to improve the repair network bandwidth. Though bandwidth savings can be achieved by delaying the data repair, it compromises data reliability. The technique LRTR [41] performs temporary redundancy of surviving chunks on risky stripes and repair of failed chunks is delayed to reduce network overhead. Though temporary redundancy improves reliability, the delay in the repair of failed chunks will increase the access latency of hot data. A combination of cooperative repair and caching techniques is proposed by Subedi at al. [22]. This reduces network traffic and execution time of MapReduce applications significantly.

Caching involves additional storage overhead. It only repairs the data blocks that are resided in the same stripe of the data that are read due to the MapReduce job and it will not eliminate data blocks that are degraded due to failure and its associated degraded read latency. Partial Parallel repair (PPR) [27] divides reconstruction operation into multiple nodes and combines partial results to reconstruct unavailable data blocks. This reduces repair time and degraded read time significantly while also reducing network pressure. SelectiveEC [40], a recovery task scheduling module, carefully selects nodes to read source data blocks for recovery and to store recovered ones. It increases the recovery throughput. However, the repair is not proactive in these works. HACFS [28] reduces data reconstruction time, degraded read latency, and network traffic. The complexity of this storage system increases as it uses two different erasure codes and dynamically adapts between them according to workload change. A framework called Zebra is presented by Li and Li [29] to reduce the reconstruction overhead of hot data. This encodes data into multiple tiers and employs different tiers with different values of parameters. It suggests maintaining less number of data blocks for hot data to reduce network traffic. Though it reduces network transfer for hot data, the storage overhead is linear with the amount of hot data.

Gong et al. [35] proposed the SPSN algorithm that selects an optimal set of nodes to participate in data repair to improve repair performance. Bai et al. [33] proposed PPT and PPTC. They were designed to avoid network traffic during data recovery. They eliminate the bottleneck links during data recovery by analyzing the bandwidth gap among links. SMFRepair and MSRepair techniques [32] eliminate the low bandwidth links to reduce recovery time in the heterogeneous network. PivotRepair [42] utilizes storage nodes available up-link and down-link bandwidth to fast-track the repair by bypassing the congested links. The techniques proposed by Gong et al. [35], Bai et al. [33], Zhou.H et al. [32], and Yao et al. [42] are reactive. The proactive recovery techniques can reduce the data recovery time to zero as the recovery is proactive when the failures are successfully predicted.

Li et al. [8] defined a system that used failure prediction techniques to implement proactive replication in erasure codes for reducing degraded read latency and improving read performance. However, it did not address data unavailability due to machine failures. HP AutoRAID [5] automatically manages the migration of data between 2-way replication of active data and RAID 5 for inactive data with the help of access pattern change. However the 2-way replications and RAID5 offer limited reliability. Araujo et al. [7] proposed hybrid coding and double coding. Both strategies combine the use of replication and erasure coding. Li et al. [20] defined a cost-effective data reliability management mechanism to ensure the reliability of massive data with minimum replication based on a generalized data reliability model. None of the above works incorporate the client's expectation and nature of data to define bandwidth-efficient recovery of erasure codes. FastPR [31] and LFPR [43] combine two repair methods, migration, and reconstruction to reduce repair time. Though this method is proactive, the data access latency will be increased when the migration occurs.

Greenan et al. [24] estimate energy consumption of data recovery in erasure coding, taking into account only the power consumption of disks involved in the recovery operation. Several researches [24], [25] calculate data reconstruction energy consumption, on the basis of participating node's energy consumption and its active time during data recovery. Ahmadvand et al. [44] proposed a method and Dynamic voltage and frequency scaling (DVFS) technique to manage the energy consumption of big data processing. A summary of the related work on improving reliability, energy efficiency, storage efficiency, bandwidth efficiency and performance improvement is presented in Table 1.

Failure predictions in cloud storage systems enable cloud service providers to define efficient proactive failure management in cloud storage. Various statistical and machine learning methods are used to predict failures in cloud storage systems. A few methods [13], [14], [36], [37] are used to predict hard drive failures based on SMART attributes. Li et al. [13] achieved 95% predictions with a False Alarm rate of less than 0.1%. Hence the failure prediction of disk drives is high with fewer false positives.

Many researches focused on predicting failures in distributed systems based on system logs. Javadi et al. [15] presented the failure model as a predictive method of distributed systems availability and unavailability. Agarwal et al. [6] use log messages to predict failures in Hadoop clusters. Data access patterns in a distributed storage can be used to identify the popularity of data blocks in real-time over a certain period of time. Based on their popularity, data blocks can be classified as hot, warm, or cold. As the access pattern changes, the popularity of data blocks has to be updated. Various researches [1], [38] used popularity-based classification to improve the durability, availability, and read performance of cloud storage systems.

Existing research does not reduce recovery network bandwidth/ traffic of erasure coding to the same extent as replication can reduce. In this paper, we propose an optimization algorithm to enhance the efficiency of proposed proactive recovery methods. The optimization algorithm builds up on proactive recovery techniques that were proposed in the preliminary version of this paper [21] and that we briefly discuss in the following section.

TABLE 1. Summary of related work.

Author	Repair Bandwidth	Recovery Performance	Reliability	Energy Efficiency	Storage Overhead
Dimakis et al. [9]	Reduced network traffic	improved performance	improved Reliability	NA	No
Sathiamoorthy et al. [3]	reduces approximately 2x on network traffic	improved performance	High Mean Time To Data Loss	NA	14% Additional Storage Overhead
Hung et al. [3]	Reduced network traffic	improved performance	Tolerates any n-k-1 failures, and 86% of n-k failures	NA	1.6x of storage overhead
Wang et al. [39]	NA	improved performance	Improved Reliability	NA	Yes
Rashmi K.V et al. [11]	reduces the network traffic by 35%	32% reduction in the data read time	Improved Reliability	NA	No
Silberstein. M [12]	recovery reduces repair bandwidth by up to 76%	NA	Reduced Reliability	NA	No
Luo. L et al. [41]	reduces repair network traffic by 43.3%	NA	improves data reliability by 13.6 times	NA	yes
Subedi. P et al. [22]	yes	improves data reliability	job runtime in HAdoop	NA	yes
Mitra.S et al. [27]	Reduced network traffic	reduces repair time and degraded read time significantly	NA	NA	No
Xu.L et al. [40]	NA	increases the recovery throughput by up to 30.68%	NA	NA	No
Xia.M et al. [34]	NA	speeds up the failure recovery up to 2.49 times	NA	NA	No
Gong, Q et al. [35]	NA	Optimal regeneration time can be achieved	NA	NA	No
Yao, Q et al. [42]	NA	Reduced Repair time	NA	NA	No
Zhou.H et al. [32]	NA	Reduced Recovery time	NA	NA	No
Li et al. [8]	NA	78% drop in recovery time	improves reliability	NA	Yes
Wilkes.J et al. [5]	NA	improved performance	improves reliability	NA	Yes
Araujo et al. [7]	NA	improved performance	improves reliability	NA	Yes
Li et al. [20]	NA	NA	improves reliability	NA	Yes
Shen.Z et al. [31]	NA	improved performance	NA	NA	Yes
Wu.Y et al. [43]	NA	Reduces repair time	NA	NA	Yes
Greenan et al. [24]	NA	NA	NA	improved energy efficiency	No
Huang.J. et al. [25]	NA	NA	NA	can save energy up to 29%	No
Ahmadvand. H [44]	NA	NA	NA	NCan save energy up to 29%	No

III. PROACTIVE RECOVERY TECHNIQUES

In this section, we briefly discuss the proactive recovery techniques ProDisk, ProMachince, ProHot, and ProHot LazyCold proposed in the preliminary version of this paper [21].

- ProDisk: When disk failures are predicted, all the data blocks in the failure-predicted disks (all disks in the machines that are predicted to fail permanently) are proactively replicated as described by Li et al. [8]. Since it is crucial to handle disk failures pro-actively to ensure high durability of data, ProMachine, ProHot, and ProHot LazyCold also handles disk failures the same as ProDisk.
- ProMachine: Machine failures can be classified as permanent, long-term, or short-term depending on Mean Time To Repair (MTTR). ProMachine selects all data blocks from the failure-predicted machine for proactive replication. Typical reconstruction of erasure codes will be applied for unpredicted machine and disk failures.
- ProHot: The data which are more likely to be accessed soon is known as hot data. In the ProHot method, the system periodically identifies hot data blocks. It applies proactive recovery only for hot data blocks from failure-predicted machines. Typical reconstruction of erasure

codes is applied to recover all cold data blocks from failure predicted machine and unexpected failures.

- ProHot_LazyCold: In case of any temporary long-term machine failure predictions, it is acceptable to delay the recovery of cold data since it is not going to be accessed soon. ProHot_LazyCold selects hot data form failure predicted machine for proactive replication and it applies lazy recovery for cold data.

To improve the efficiency of proactive recovery methods, we proposed an optimization algorithm that is introduced in the next section.

IV. ADAPTIVE BANDWIDTH EFFICIENT CLOUD STORAGE SYSTEMS

An adaptive cloud storage system employs several proactive recovery methods. In the event of any failure prediction, it selects one of the proactive recovery methods that can meet the client's SLA. To improve the efficiency of proactive recovery, the proposed system adapts client SLA and chooses the most suitable method for recovery. Client data can be classified as hot, warm, or cold depending on the access frequency. Data recovery can be delayed for the data that is having less access frequency (cold data). The client may also accept a delay in accessing that data. At the same time, a delay in access latency is not acceptable for hot data. Activating proactive recovery of hot data can reduce access latency in the presence of failure. In this paper, we include an algorithm in the existing adaptive bandwidth-efficient cloud storage systems such that it minimizes the number of data blocks replicated during proactive recovery, regardless of the selection of any proactive recovery methods. This optimization algorithm was included in the existing proposed system; it helped to further increase the storage and network efficiency.

A. ARCHITECTURE AND DESIGN

The architecture of the adaptive bandwidth-efficient cloud storage system is proposed in the preliminary version of this paper [21]. We have introduced a new component called Enhanced Proactive Recovery (EPR) in the existing architecture, to further enhance the efficiency of the proposed system, regardless of the selection of any proactive recovery method to handle failure prediction. The overall architecture is presented in Figure 1. It is implemented as an extension of regular data storage. A dedicated proxy server extends the support of encoding and decoding erasure codes. It also handles failures in storage systems. The storage server stores and retrieves data. The storage server's availability status and disk health status are reported to the proxy server, which is responsible for increasing or decreasing the data replication factor. The system adjusts the replication factor of erasure-coded objects when failures are predicted.

The component *disk failure prediction* monitors the health status of individual disks, using classification and regression tree methods with information derived from SMART attributes [13]. *Node failure history and disk health information* component collects node failure history and calculates the node's Mean Time to Failure (MTTF) and MTTR using various statistics of availability and unavailability. It also collects disk failure alarms from the component Disk failure prediction. Data access pattern classifies data blocks as hot based on their popularity over a time period. Assuming that data blocks with high access frequency have more chance to be accessed in the future, we define those as hot. It is recorded as $H = \{h_{ij}\}$ where h_{ij} is the j^{th} block from disk i that is identified as hot. *Data block health monitor* collects information on failure-predicted nodes and disks from *node failure history and disk health information module*. It identifies and sets different flags of the data blocks that are predicted for failure due to disk, machine (Permanent, long-term, or short-term) failures. The Client's requirements in regard to durability, availability, and access latency are recorded in the client SLA.

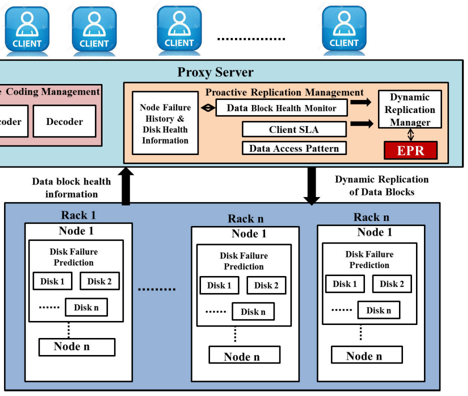


FIGURE 1. System architecture.

Dynamic replication manager chooses one of the best recovery techniques which can meet client SLA with limited resources from recovery methods ProDisk, ProMachine, ProHot, and ProHot_LazyCold. If the client requires high durability, ordinary availability, and access latency, the dynamic replication manager will select the recovery technique ProDisk. ProHot will be selected if they require high availability and low access latency of hot data. In case the client requests high durability, availability, and low latency, the technique ProMachine will be selected. The ProHot_LazyCold will be selected by the dynamic replication manager if the client requests high availability and low access latency for hot data and they ignore the availability and access latency of cold data. Based on the selection of the recovery technique, it chooses a set of data blocks for proactive recovery.

Enhanced Proactive Recovery (EPR) module attempts to reduce the data blocks that are selected for proactive replication. EPR selects an optimal subset of data, taking into account the system's current network traffic and data duplication. EPR deletes the data blocks on condition that the failure-predicted machine has come back to life or a failure-predicted disk does not fail as expected. It is also responsible for scaling up and down the number of dedicated temporary storage nodes, according to the predictions and amount of data to be stored in temporary storage during a period of time. EPR is also responsible for allocating a highly available node as a temporary storage such that any failure in this temporary storage node is minimal. Any prediction of failure in this temporary storage will also lead to proactive replication. The implementation of an optimization algorithm in this module is the main contribution of this paper and it is discussed in detail in the next section.

B. PROPOSED RECOVERY APPROACH

The overall functionality of the proposed enhanced adaptive bandwidth-efficient cloud storage system and the recovery approach performed by this system are discussed in this section.

The proposed system initially stores data with any (n, k) erasure code. With the help of disk/machine failure prediction methods employed in cloud storage systems, failure types and MTTR of node failures are predicted. Failures are also identified as disk, permanent machine, temporary long-term machine (MTTR$>$15 minutes), or temporary short-term machine (MTTR$<$15 minutes) failures. The set of data blocks $(b_1, b_2, \ldots, b_i)$ that is more likely to be accessed soon is defined as the hot data set H. Based on the failure types, and client SLAs, one of the appropriate recovery techniques is selected from the recovery techniques ProDisk, ProMachine, ProHot, ProHot_LazyCold.

When the disk/permanent machine failures are predicted, all the data blocks in the failure-predicted disk (all data blocks of each disk in a failure-predicted machine) are selected for proactive replication by the dynamic replication manager. Next, EPR applies minimization on the selected set of data blocks by considering the system's current network traffic and data duplication. It chooses a subset of data blocks for proactive replication. The selected subset of data is proactively replicated into the permanent storage as described in [8]. The counter variables of corresponding replicated data blocks are incremented. These counter variables are used to identify if the particular data blocks are replicated already or to delete data blocks against false prediction. A delay is applied while deleting data blocks that are replicated for the false prediction. Time In Advance (TIA), which is provided by the failure prediction algorithm, is used as a time delay to delete the data blocks that are replicated due to false positives of failure predictions. A time delay larger than TIA is the better choice, but this will result in extra storage. The choice of time delay varies and depends on the storage system where the system is utilized.

In the event of long-term temporary machine failure predictions, Dynamic Replication Manager either select all data blocks from the failure-predicted machine for the recovery method ProMachine or selects appropriate data blocks for the recovery methods ProHot and ProHot_LazyCold. Then EPR selects a subset of it. The subset of data, selected by EPR is replicated into the dedicated temporary storage. Data blocks that are not replicated are recovered by typical reconstruction of erasure codes. While data blocks are proactively replicated into temporary storage, the corresponding data block's counter variables are incremented. These variables are used to identify if the particular data blocks are already replicated or need to be deleted when the machine recovers from temporary machine failures. When the failure-predicted nodes recover from actual failure and if no further failures are predicted for the same nodes, proactively replicated data blocks corresponding to those nodes are deleted. In the occurrence of node/disk failure, the reference is made to the data blocks that are proactively replicated. This will reduce the number of data reconstructions in erasure-coded storage systems.

V. ENHANCED PROACTIVE RECOVERY

In this section, to further enhance the efficiency of proactive recovery, we discuss optimization on failure-predicted data blocks and select appropriate data blocks for proactive replication.

A. PROBLEM FORMULATION

Let B be a set of data blocks that are stored in the cloud storage system. Let an element b_{ij} in B denote a data block that is stored in the j-th location of disk i. Let DR $= \{\mathrm{dr}_{ij}\}$ be a set of counter variables which represents the number of times, the corresponding data block b_{ij} in set B is replicated. The counter variables in DR represent only the replication of data blocks on the occurrence of disk/permanent machine failures. Similarly, MR is a set of counter variables that are used to represent data blocks that are replicated due to temporary machine failures. An element mr_{ij} in MR represents a variable that is incremented if the corresponding data block b_{ij} is replicated on the node which is dedicated to handling machine failure. The cardinality of the set DR and MR is equal to the cardinality of set B. The value of mr_{ij} represents, the number of copies of block b_{ij} that exist in the dedicated temporary storage. The variables dr_{ij} are decremented at the actual failure of disk i. Variables mr_{ij} are decremented when the machine that holds disk i has come back from failure, once after the occurrence of actual failure of the same machine. The copy of the data block b_{ij} is also deleted from the node which is dedicated to handling temporary machine failure. In case of a false positive, the copy of a data block is deleted after applying the appropriate time delay.

In the event of any disk/permanent machine failures or temporary long-term machine failure prediction, Dynamic Replication Manager selects a set of data blocks for proactive replication according to the method selected for proactive recovery. Let FP be a set of data blocks selected for proactive replication. Let TIA be the time in advance of predicting failures, EPR has to select an appropriate subset of data for proactive replication from the set FP, taking into account data duplication and the system's current network traffic. Let X be a set of binary decision variables x_{ij}, such that each variable x_{ij} represents block b_{ij} from FP, $x_{ij} = 1$ if the data block b_{ij} from FP is selected for replication and $x_{ij} = 0$ otherwise. When the EPR sets the values of x_{ij} belonging to X, it selects the subset of b_{ij} from the set FP that are corresponding to x_{ij} equal to 1.

The module EPR should not select the data block b_{ij} from the set FP for proactive replication if the system has more than one copy of the data block b_{ij} at the failure prediction time t. This will avoid unnecessary data duplication of the block b_{ij}. For example, consider a disk "disk i" is predicted for failure. Dynamic Replication Manager selects all blocks in "disk i" for replication. Let b_{ij} be a data block belonging to "disk i" which is selected for replication by the Dynamic replication manager. The system may already hold a copy of b_{ij} due to the failure prediction of the machine which contains the "disk i". In this case, EPR will not select the data block b_{ij} for proactive replication. Doing this will not only save the bandwidth of creating extra copies but also save storage. On the actual occurrence of failure of "disk i", appropriate reference could be made to the copy of data block b_{ij} such that it can handle the failure of "disk i". The scenario is similar when the block b_{ij} is marked for temporary machine failure. Hence we have,

$$x_{ij} = 0 \quad \forall\, b_{ij} \in D \quad if\ \mathrm{mr}_{ij} = 1 \tag{1}$$
$$x_{ij} = 0 \quad \forall\, b_{ij} \in M \quad if\ \mathrm{dr}_{ij} = 1 \tag{2}$$

'Systems network traffic can be effectively managed by eliminating proactive recovery for some appropriate failure predictions using the system's Current Recovery Bandwidth (CRB). When the system's current recovery bandwidth reaches the system's recovery bandwidth capacity, EPR should avoid proactive recovery of the events, which are expected to increase the system's recovery bandwidth above Recovery Bandwidth Capacity (RBC) at the given time.

$$(S * \sum x_{ij})/\mathrm{TIA} + \mathrm{CRB} <= \mathrm{RBC} \quad \forall x_{ij} \in X \tag{3}$$

We formulate the problem of selecting a subset of data blocks for proactive replication, from the set of data blocks with a predicted failure FP as an integer programming as follows:

$$Minimize \sum x_{ij} \quad \forall x_{ij} \in X \tag{4}$$

Subject to:

$$x_{ij} = 0 \quad \forall\, b_{ij} \in D \quad if\ \mathrm{mr}_{ij} = 1 \tag{5}$$
$$x_{ij} = 0 \quad \forall\, b_{ij} \in M \quad if\ \mathrm{dr}_{ij} = 1 \tag{6}$$
$$x_{ij} = 1 \quad \forall\, b_{ij} \in D \quad if\ \mathrm{dr}_{ij} = 0 \tag{7}$$
$$x_{ij} = 1 \quad \forall\, b_{ij} \in M \quad if\ \mathrm{mr}_{ij} = 0 \tag{8}$$
$$(S * \sum x_{ij})/\mathrm{TIA} + \mathrm{CRB} <= \mathrm{RBC} \quad \forall x_{ij} \in X \tag{9}$$
$$x_{ij} = \{0, 1\} \quad \forall x_{ij} \in X \tag{10}$$

In order to maintain the system's network traffic to the level of the system's RBC, we eliminate proactive recovery when the system's CRB reaches RBC. As a result, typical reconstruction will be conducted to recover data blocks that were not proactively handled and this may increase the system's network traffic substantially.

Algorithm 1 Enhanced Proactive Recovery Algorithm

```
INPUT: $FP$, $FT$
 1: if BANDWIDTH_CONSTRAINT (FP)=true then
 2:     for each $b_{ij}$ in FP do
 3:         if $m_{ij} \leq 1$ and $d_{ij} \leq 1$ then
 4:             $x_{ij} = 1$
 5:         else if $Ft$ =machine and $d_{ij} \geq 1$ then
 6:             $x_{ij} = 0$
 7:         else if $Ft$ =disk and $m_{ij} \geq 1$ then
 8:             $m_{ij} = m_{ij} - 1$
 9:             $d_{ij} = d_{ij} + 1$
10:             define disk holding copy of $d_{ij}$ as permanent
11:             $x_{ij} = 0$
12:         end if
13:     end for
14: else
15:     for each $b_{ij}$ in FP do
16:         $x_{ij} = 0$
17:     end for
18: end if
OUTPUT: $X$
```

B. ENHANCED PROACTIVE RECOVERY ALGORITHM

The objective of the algorithm is to minimize the number of data blocks to be replicated during proactive recovery. To solve the problem of minimizing the number of proactive replicated data blocks, we designed an algorithm called Enhance Proactive Recovery Algorithm (EPRA). Upon any failure predictions, the proposed algorithm determines the set of data blocks that are needed to be handled proactively by taking into account the system's current network traffic and data duplication information. The EPRA is presented in Algorithm 1.

On receipt of any failure prediction event, the algorithm examines how the system's network traffic will be affected while activating proactive recovery. The required calculations are represented in Algorithm 2. This algorithm calculates the total Transfers Required (TR) to proactively handle the predicted event. Using TR, the total Projected Bandwidth Need (PBN) to proactively handle the predicted event is calculated as follows,

$$\mathrm{PBN} = \mathrm{TR}/\mathrm{TIA} \tag{11}$$

The algorithm also calculates the Projected Network Traffic (PNT) of the system using the system's CRB as follows,

$$\mathrm{PNT} = \mathrm{CRB} + \mathrm{PBN} \tag{12}$$

Based on the calculated PNT, the system determines whether to proactively handle the predicted failure or not at any given time. Following that, data de-duplication is performed in lines 2 to 9. For any failure prediction of 'disk i',

Algorithm 2 Bandwidth_Constraint(FP)

```
 1: procedure Bandwidth_Constraint(FP)
 2:     initialize TR=0
 3:     initialize PBN=0
 4:     for each $b_{ij}$ in FP do
 5:         if $m_{ij} \leq 1$ and $d_{ij} \leq 1$ then
 6:             TR=TR+S
 7:         end if
 8:     end for
 9:     PBN=TR/TIA
10:     PNT=CRB+PBN
11:     if PNT $\leq$ RBC then
12:         return true
13:     else
14:         return false
15:     end if
16: end procedure
```

Let us consider a data block b_{ij} in 'disk i' is already replicated due to the proactive recovery of the machine, that contains 'disk i'. The system should avoid replicating the block b_{ij} due to the prediction of 'disk i'. However, an appropriate reference has to be made to the copy of b_{ij} such that it cannot be deleted during the eviction process, which is activated when the machine containing 'disk i' recovers from failure. On the other hand, consider a scenario where a data block b_{ij} has to be proactively handled on receipt of a machine failure prediction to which it belongs. If the system already has a copy of b_{ij}, due to failure prediction of 'disk i' in advance. The system will simply avoid replicating block b_{ij}.

The algorithm gets a set of data blocks in failure predicted machine/disk and the failure type from Dynamic Replication Manager and sets the decision variable $x_{ij} = 1$ if the corresponding data block b_{ij} from the failure predicted set has to be replicated. EPRA sends X, a set of decision variables to the Dynamic Replication manager. Dynamic Replication manager replicates the data block b_{ij} from set FP only if the corresponding decision variable x_{ij} is 1.

VI. ENERGY CONSUMPTION ANALYSIS

Several metrics are used to measure the energy consumption of storage systems. Some metrics use the energy consumption of hardware/software components to calculate the energy consumption of the storage systems. Some others measure the energy consumption of storage systems by measuring the application's usage of physical resources like storage, network, and memory.

In order to compare energy efficiency from replication and erasure coding with various proactive recovery techniques, we estimate the energy consumption of storage and network devices. The energy consumption of the storage systems is estimated by calculating the usage of the disks. We calculate the energy consumption of the network devices in terms of the amount of data transferred via the top of the rack switch during recovery.

Since we use these energy models to compare the energy consumption of various recovery methods, we do not consider the energy consumption of the intervening application running on top of the storage system.

A. ENERGY CONSUMPTION OF STORAGE DEVICES

As mentioned above, we estimate the storage energy by calculating the energy consumption of the disk drives in the storage system. Even though several other devices like machines, racks, and cooling systems are involved in the energy consumption of storage systems, the disk remains the most important component of the storage systems, and the energy consumption of the storage system is directly proportional to the number of disks employed in a storage system. Since we use this model for the system comparison for various recovery methods, we will ignore the energy consumption of other devices. As with any models, it makes some abstractions for empirical investigations. Hence, energy consumption of the storage devices is expressed as:

$$E_t = D_t * T_t * U \tag{13}$$

where E_t is the energy consumption of the storage system during the time period t as the product of the number of disks active D_t, the amount of time active T_t and the energy consumption of the disk per unit time U.

Replication uses more disks compared to erasure coding as it stores more data blocks to ensure reliability. Proactive replication in erasure coding uses additional disks for a certain period of time. Disk failure/permanent machine failure predictions use the additional disk from the time of prediction till the actual occurrence of predicted disk failure. Temporary machine failure uses the additional storage from the time of prediction until the recovery of the corresponding machine. The number of disks used in case of disk failure/permanent machine failure predictions is equal to the number of disks predicted for failure, whereas it varies with respect to the selection of recovery methods in the event of temporary machine failure predictions.

B. ENERGY CONSUMPTION OF NETWORK DEVICES

We estimate data recovery energy consumption by calculating the amount of data transferred through the router during the recovery of each failure event. We have calculated the energy consumption of data recovery using the model proposed by Viswanath et al. [19],

$$E_t = E_p * R_t/S + E_{st} * R_t \tag{14}$$

We calculated the energy consumption of a router as a sum of the energy consumption of processing and storing the data blocks that were involved in recovery during time t. Energy consumption caused by data processing in routers is expressed as the product of per-packet processing energy E_p and the incoming data rate during recovery time t. The input data rate can be calculated using data transfer rate (amount of data transferred per second) R_t and data block size S as R_t/S. Data storage energy in the router is calculated as a product of per byte storage energy E_{st} and the input data rateR_t. The total energy consumption of the router during time T is calculated as the sum of all individual recovery energy consumption E_t during time T. Since we use Equation 14 to measure the energy consumption of routers during recovery, we do not consider the router's idle power. Also, we do not consider the energy consumption of data transfer with respect to the intervention of applications running on top of it.

Finally, we calculated the total energy consumption of the system during time T as the sum of the storage and network device's energy consumption during that time.

VII. PERFORMANCE EVALUATION

We used the ds-sim [12] simulator to evaluate the efficiency of proactive recovery methods while applying EPRA. We have added a significant amount of codes in ds-sim to implement failure prediction, proactive recovery, optimization of proactive recovery, and energy consumption.

The ds-sim simulator simulates 3-tier storage components including disks, machines, and racks. ds-sim stores data in blocks and multiple data blocks form a stripe. A stripe is composed of a set of original and parity data blocks such that any data block in a stripe can be reconstructed using a subset of data blocks in a stripe. In n way replication a stripe consists of all replicas of a block, whereas it is comprised of k original and $n-k$ parity blocks for Reed-Solomon (n, k) erasure code. ds-sim randomly chooses racks to store blocks of data. ds-sim randomly chooses n racks to store n blocks of a stripe such that no two blocks in the stripe are placed on nodes in the same rack.

ds-sim generates failure and recovery events for all hardware components using either synthetic probability distributions or failure traces. Each storage component disk, machine, and rack are incorporated with separate failure and recovery distribution. Disk failures include latent sector failures and permanent disk failures. Latent sector failures are detected and recovered using a technique called scrubbing. Machine failures include both transient and permanent failures. The ds-sim starts recovery of a permanent machine failure immediately and it applies 15 minutes delay for initiating recovery of transient failures. Rack failures are assumed to be transient. The ds-sim performs a run time simulation and records all instantaneous properties of the system including repair bandwidth, repair energy, repair storage overhead for proactive recovery, and the number of degraded stripes. Table 2 lists the values of simulation parameters. The simulation parameters such as total data, number of disks/racks or machines, and disk capacity are based on the work by Silberstein et al. [12]. The choices of energy consumption of disk and router's storage energy consumption are made according to Harnik et al. [26] and Li et al. [20] respectively.

In order to compare and evaluate the efficiency of the optimization method EPRA compared to its heuristics, we have

TABLE 2. Simulation parameters.

Parameter	Value
Total data	3 PB
Disk Capacity	750 GB
Recovery bandwidth capacity	650 TB/day
Disks/machine	20
Machines/rack	11
Disk Energy Consumption while operating	43.2 kJ/h
Router's per byte storage energy	14 nJ
Packet processing energy	1375 nJ
Prediction percentage	90
Number of iterations	50

used the same set of failure events for heuristics and optimization. The heuristics methods are proposed in our early work [21]. This eliminates any difference in metrics due to the variation of failure events generated by the simulator.

A. PERFORMANCE ANALYSIS

In this section, we analyze and compare the energy consumption of replication, the most popular erasure code Reed-Solomon (14, 10), and various proactive recovery techniques. We also show the trade-off between dedicated temporary storage overhead and bandwidth savings of various proactive recovery methods.

1) SYSTEM ENERGY CONSUMPTION

We run simulations with the configuration parameters and failure prediction rate as listed in Table 2. We use time in advance, 24 hours, which is found reasonable according to Li et al. [13] and Agrawal et al. [6]. As the failures are generated by the simulator, the recovery energy consumption and energy consumption of dedicated temporary storage are calculated for each failure event, except for machine failures for less than 15 minutes. Since we do not consider the intervention of any application running on top of the storage, we have calculated the storage energy of disks by assuming that disks are active during the entire period of simulation. The shutting down of the inactive disks is not considered here which is usually used to enhance energy savings of storage systems. In case of any disk failure prediction, additional disks are activated to support proactive recovery that would always incur some energy consumption. It is calculated by assuming that the storage system activates an extra disk at the time of failure prediction and it is active from the time of failure prediction till the actual occurrence of failure on the same failure-predicted disk. In the event of temporary machine failure prediction, the system will activate the number of disks proportional to the number of data blocks that are selected for proactive recovery. Those disks will be active from the time of failure prediction till recovery of the failure-predicted machine and energy consumption is calculated accordingly.

Figure 2 shows the comparison of average energy consumption in KJ/day for replication, (14, 10) Reed-Solomon, and several proactive recovery methods. The figure also shows the average energy consumption of the system, with

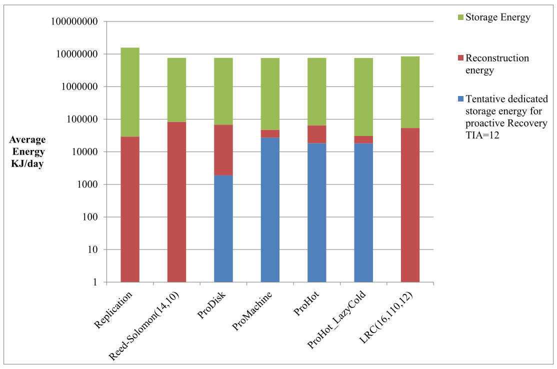


FIGURE 2. Storage system's average energy consumption in KJ per day.

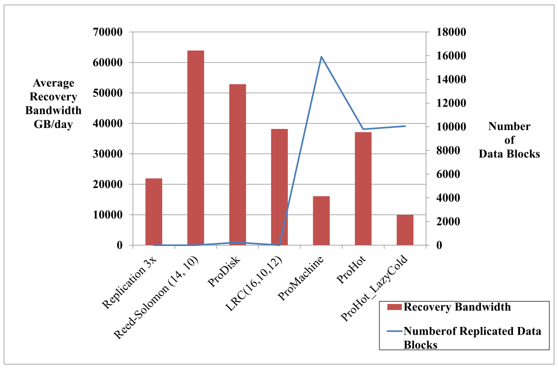


FIGURE 4. Storage overhead and average recovery bandwidth.

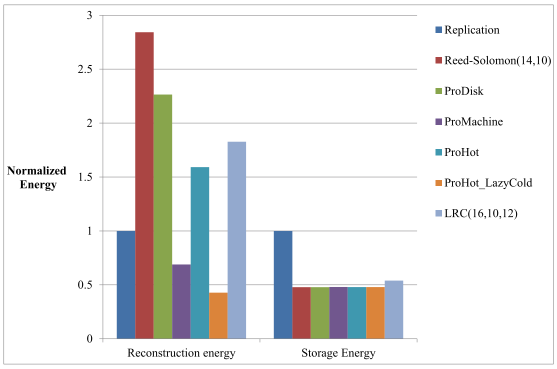


FIGURE 3. Average storage and recovery energy consumption in KJ per day.

individual split-ups for energy consumption of storage, recovery bandwidth, and temporary dedicated storage for proactive recovery, respectively. Reed-Solomon (14, 10) saves the system's overall energy consumption up to 51.7% compared to replication. ProDisk reduces energy consumption by up to 51.8% compared to Replication. ProHot reduces energy consumption up to 51.9%, ProMachine reduces energy consumption by 51.8% and ProHot_LazyCold reduces energy consumption by 52% compared to the same approach. The energy savings of proactive recovery methods are very limited. This is due to the fact that the energy consumption of dedicated storage compensates for the energy savings of recovery bandwidth. Figure 3 shows data reconstruction and storage energy for various coding schemes and recovery methods that are normalized against replication. Energy consumption of temporary storage overhead of proactive recovery methods is calculated for TIA 12 hours. Figure 3 shows that the storage energy consumption overhead of proactive recovery methods is the least. The power consumption can also be reduced by carefully scheduling the proactive replication in appropriate TIA which is a future research direction.

2) TEMPORARY DEDICATED STORAGE OVERHEAD

To evaluate resource savings from proactive replication, we calculated the average number of data blocks replicated per day. Figure 4 shows the trade-off between recovery bandwidth and temporary dedicated storage for various proactive recovery methods. ProMachine reduces the recovery bandwidth of Reed-Solomon (14, 10) up to 75% with approximately 1.3% storage overhead compared to Reed-Solomon. ProHot reduces recovery bandwidth up to 41% where as ProHot_LazyCold reduces recovery bandwidth by 85% with 0.75% additional storage savings compared to Reed-Solomon. The storage overhead of replication, ProDisk, ProMachine, ProHot and ProHot_LazyCold are 90%, 0.01%, 1.3%, 0.75%, and 0.75%, respectively compared to Reed-Solomon. Figure 4 shows that proactive recovery methods offer excellent bandwidth savings as compensation for dedicated temporary storage overhead due to proactive replication.

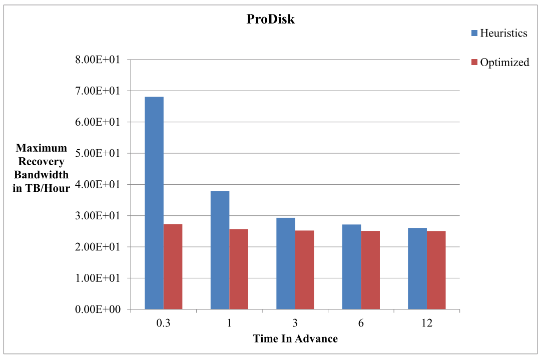
B. ENHANCED PROACTIVE RECOVERY

In this section, we investigate how the optimization of proactive recovery (EPRA) further improves the efficiency of storage systems. To eliminate any difference in measurement due to variations of events generated by the simulator, we applied optimization on the same set of events that we used for heuristic proactive recovery. We compare repair network traffic/bandwidth, energy, and storage overhead of heuristic proactive recovery against optimized for several proactive recovery techniques.

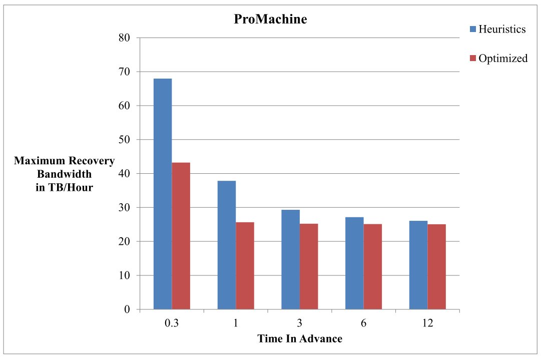
1) REPAIR NETWORK TRAFFIC

We examine how the EPRA reduces repair network traffic compared to its respective heuristics. The results of examining network traffic of proactive recovery method with varying TIA of failure predictions are presented in Figure 5. Since recovery bandwidth is inversely proportional to recovery time, the reduction of TIA increases network traffic of storage systems. In this experiment, the maximum recovery bandwidth capacity is set to 650 TB/day and when TIA is set to 12 hours optimization does not show much savings. When TIA was 30 minutes optimization shows significant savings in network traffic.

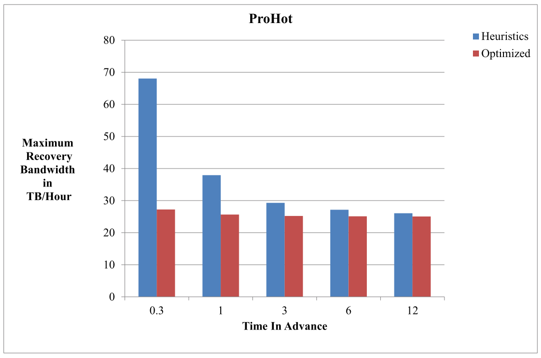
The optimization of recovery methods ProDisk, ProHot, and ProHot_LazyCold reduces network traffic up to 60%, 37%, 60%, and 49%, respectively compared to its



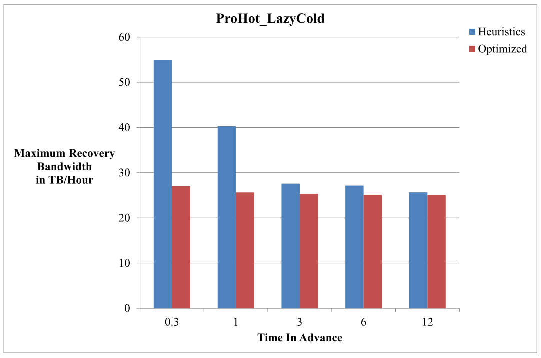
(a)



(b)



(c)



(d)

FIGURE 5. Maximum instantaneous recovery bandwidth, in TB/h. (a) ProDisk (b) ProMachine (c) ProHot and (d) ProHot_LazyCold.

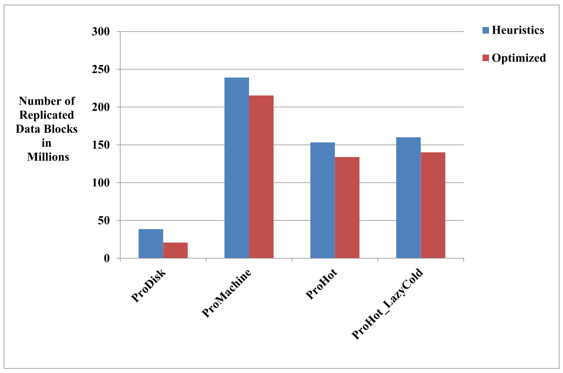


FIGURE 6. Total number of proactively replicated slices due to proactive recovery.

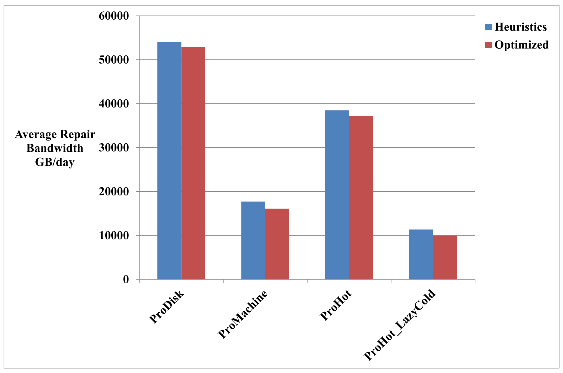


FIGURE 7. Average recovery bandwidth in GB per day.

corresponding heuristic methods. Hence the computation time due to optimization can be ignored considering the reduction in network traffic compared to its heuristics. Similarly, optimized ProDisk, ProMachine, ProHot, and ProHot_LazyCold recovery methods reduce network traffic around 4%, 4%, 4%, and 3.6%, respectively compared to when TIA is 12 hours. Clearly the network savings due to proactive recovery increase as TIA decreases. Applying optimization to proactive recovery methods reduces recovery bandwidth around the system's recovery bandwidth capacity. However, ProMachine could not reduce network traffic further while TIA is 30 minutes. This is due to the fact that this method handles a large amount of data compared to other methods, during proactive recovery.

2) TEMPORARY DEDICATED STORAGE OVERHEAD

To evaluate temporary storage savings due to the optimization of proactive replication, we have calculated the total amount of data transferred over the simulation period. Figure 6 shows the comparison of the total amount of data transferred while using heuristics and optimization, respectively for various proactive recovery methods. Applying optimization on ProDisk provides up to 46% storage savings compared to the heuristic of the same method. Similarly, ProMachine, ProHot, and ProHot_LazyCold offer up to 10%, 12%, and 12% of storage savings respectively compared to their corresponding heuristics. Storage savings of optimized proactive recovery methods are increased with respect to the number of data blocks handled proactively by those methods.

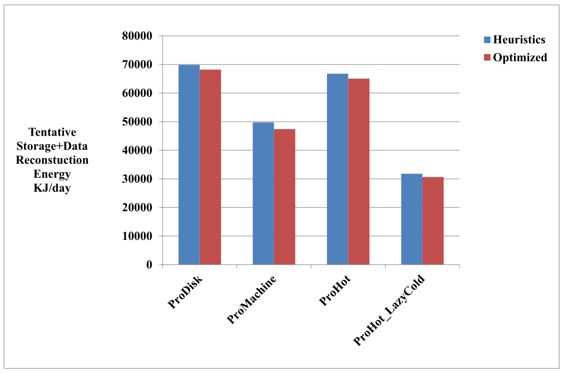


FIGURE 8. Average energy consumption in KJ per day.

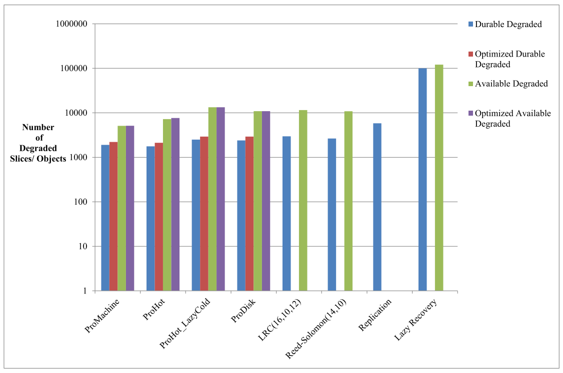


FIGURE 9. Average number of durable degraded and available degraded slices per day.

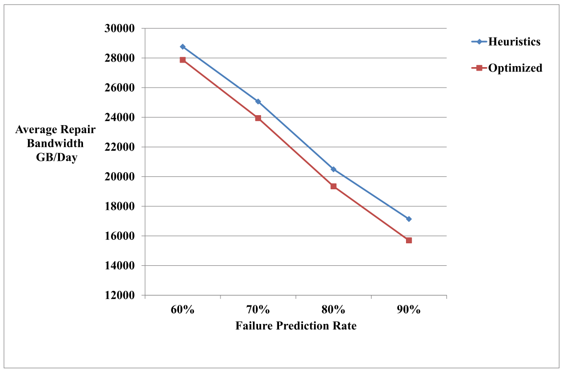


FIGURE 10. Average repair bandwidth, in GB/day with varying prediction rate.

3) REPAIR BANDWIDTH

To examine how the optimization of proactive recovery saves recovery bandwidth, we calculated the average recovery bandwidth per day. Figure 7 shows the bandwidth savings of optimization compared to its heuristics in recovery methods ProDisk, ProMachine, ProHot, and ProHot_LazyCold. Optimizing proactive recovery of ProDisk, ProMachine, ProHot, and ProHot_LazyCold methods offers up to 3%, 9.5%, 4%, and 12% of savings respectively compared to its heuristics. Bandwidth savings of optimized proactive recovery methods are increased since it saves bandwidth by avoiding data duplication.

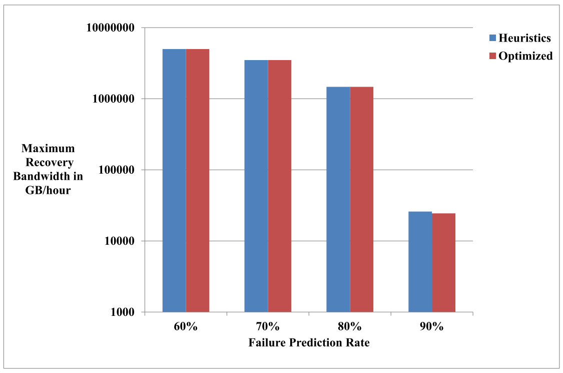


FIGURE 11. Maximum instantaneous recovery bandwidth, in GB/h with varying prediction percentage.

4) ENERGY CONSUMPTION

To examine how the optimization of proactive recovery saves energy consumption of the storage system, we calculated the average recovery bandwidth per day. Figure 8, shows the energy savings of optimization compared to respective heuristics for several proactive recovery methods. Since optimization eliminates data duplication, it saves energy in terms of temporary storage and recovery bandwidth energy consumption. Applying optimization on proactive recovery of ProDisk, ProMachine, ProHot, and ProHot_LazyCold methods offers up to 3%, 5%, 3.5%, and 4% of energy savings compared to its own heuristics.

5) RELIABILITY

To examine how the optimization of proactive recovery affects the reliability of the storage system, we have calculated the average durable degraded and available degraded objects per day. Figure 9, shows the number of degraded slices for various reliability techniques and the impact of optimization on reliability compared to its respective heuristics for several proactive recovery methods. Since optimization minimizes recovery network traffic, it slightly increases the number of degraded slices compared to heuristics. However, the number of degraded slices in optimized proactive recovery is still much better than the native methods.

C. SENSITIVITY ANALYSIS

To determine how EPRA is influenced by failure prediction rate, we measured parameters like network traffic/bandwidth, storage overhead, and energy consumption with varying failure prediction rates.

For analyzing how the system is affected by the failure prediction rate, we measured network traffic with varying disk failure prediction rates. Li et al. [13], showed that more than 90% accuracy of disk failure prediction is possible. We run the simulation with failure prediction accuracy varying from 60% to 90% and calculated recovery network bandwidth and traffic of ProMachine method with TIA equal to 12 hours, as shown in Figure. 10 and 11, respectively.

From Figure 10 bandwidth savings at failure predictions rate of 90%, 80%, 70%, and 60% of ProMachine method with TIA 12 hours are 75%, 68%, 61%, and 56% compared to (14,10) Reed-Solomon. Optimizing proactive recovery in the storage systems provides bandwidth savings 8%, 6%, 4%, and 3% compared to its heuristics with failure prediction percentages of 90%, 80%, 70%, and 60%, respectively. Bandwidth savings due to optimization reduces as the failure prediction rate decreases. The reduction in network traffic due to optimization is depicted in Figure 11. The optimization reduces network traffic up to 60% when the prediction percentage is 90%. Even though optimization offers excellent bandwidth savings with a prediction percentage of 90%, savings in network traffic is becoming very limited as the failure prediction rate decreases. This is due to the fact that when the prediction rate is low, the system's network traffic increases as it activates the typical reconstruction of unpredicted failures. Optimizing proactive recovery in the storage systems reduces network traffic (max instantaneous recovery bandwidth in MB/h) by eliminating proactive recovery for some failure predictions. However, with less failure prediction, this has a minimum effect.

Hence optimization of proactive recovery provides significant improvements in repair network bandwidth, energy consumption, and temporary dedicated storage with a smaller failure prediction percentage. However, network traffic savings due to optimizations are limited with a smaller failure prediction percentage.

VIII. CONCLUSION AND FUTURE WORKS

The two primary reliability mechanisms employed by cloud storage systems—replication and erasure coding—have their own drawbacks. Even though erasure code offers tremendous storage savings compared to replication, reconstructing lost or corrupted data blocks involve large communication overhead. In our recent work, to achieve maximum recovery bandwidth savings in erasure codes, we have proposed several prediction-based proactive recovery techniques that are defined using combinations of replications, erasure coding, and lazy recovery. As an extension of this, we proposed an optimization approach and algorithm in this paper that minimizes the number of data blocks to be replicated during proactive recovery. This optimization contributes to increasing resource savings of the storage systems. We also analyzed the energy consumption of replication, erasure coding, and erasure coding with several recovery approaches in cloud storage systems. Experiments showed that the proposed optimization algorithm increased resource savings compared to its respective heuristics. It also showed that the proactive recovery methods in erasure offer limited energy savings.

In future work, we plan to investigate the scheduling of proactive replicas in distributed storage such that it reduces degraded read latency in cloud storage. Another promising area of future research is the energy-efficient scheduling of proactive replicas in cloud storage.

REFERENCES

[1] J. Liu and H. Shen, "A popularity-aware cost-effective replication scheme for high data durability in cloud storage," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Washington, DC, USA, Dec. 2016, pp. 384–389.

[2] J. Plank, "T1: Erasure codes for storage applications," in *Proc. 4th USENIX Conf. File Storage Technol.*, San Francisco, CA, USA, 2015, pp. 1–74.

[3] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, "XORing elephants: Novel erasure codes for big data," *Proc. VLDB Endowment*, vol. 6, no. 5, pp. 325–336, Mar. 2013.

[4] D. Ford, F. Labelle, F. I. Popovici, M. Stokely, V. A. Truong, L. Barroso, C. Grimes, and S. Quinlan, "Availability in globally distributed storage systems," in *Proc. 9th USENIX Conf. Operating Syst. Design Implement.*, Vancouver, BC, Canada, 2010, pp. 1–14.

[5] J. Wilkes, R. Golding, C. Staelin, and T. Sullivan, "The HP AutoRAID hierarchical storage system," *ACM Trans. Comput. Syst.*, vol. 14, no. 1, pp. 108–136, Feb. 1996.

[6] B. Agrawal, T. Wiktorski, and C. Rong, "Analyzing and predicting failure in Hadoop clusters using distributed hidden Markov model," in *Proc. Int. Conf. Cloud Comput. Big Data Asia*, Huangshan, China, 2016, pp. 232–246.

[7] J. Araujo, F. Giroire, and J. Monteiro, "Hybrid approaches for distributed storage systems," in *Proc. Int. Conf. Manag. Grid Syst.*, Toulouse, France, 2011, pp. 1–12.

[8] P. Li, J. Li, R. J. Stones, G. Wang, Z. Li, and X. Liu, "ProCode: A proactive erasure coding scheme for cloud storage systems," in *Proc. IEEE 35th Symp. Reliable Distrib. Syst. (SRDS)*, Budapest, Hungary, Sep. 2016, pp. 219–228.

[9] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.

[10] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin, "Erasure coding in windows azure storage," in *Proc. USENIX Annu. Tech. Conf. (ATC)*, Boston, MA, USA, 2012, pp. 15–26.

[11] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A 'hitchhiker's' guide to fast and efficient data reconstruction in erasure-coded data centers," in *Proc. ACM Conf. SIGCOMM*, Chicago, IL, USA, Aug. 2014, pp. 331–342.

[12] M. Silberstein, L. Ganesh, Y. Wang, L. Alvisi, and M. Dahlin, "Lazy means smart: Reducing repair bandwidth costs in erasure-coded distributed storage," in *Proc. Int. Conf. Syst. Storage*, Haifa, Israel, Jun. 2014, pp. 1–7.

[13] J. Li, X. Ji, Y. Jia, B. Zhu, G. Wang, Z. Li, and X. Liu, "Hard drive failure prediction using classification and regression trees," in *Proc. 44th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Atlanta, GA, USA, Jun. 2014, pp. 383–394.

[14] J. Li, R. J. Stones, G. Wang, X. Liu, Z. Li, and M. Xu, "Hard drive failure prediction using decision trees," *Rel. Eng. Syst. Saf.*, vol. 164, pp. 55–65, Aug. 2017.

[15] B. Javadi, D. Kondo, D. Epema, and A. Iosup, "The failure trace archive: Enabling the comparison of failure measurements and models of distributed systems," *J. Parallel Distrib. Comput.*, vol. 73, no. 8, pp. 1208–1223, Aug. 2013.

[16] R. Nachiappan, B. Javadi, R. N. Calheiros, and K. M. Matawie, "Cloud storage reliability for big data applications: A state of the art survey," *J. Netw. Comput. Appl.*, vol. 97, pp. 35–47, Nov. 2017.

[17] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction," *IEEE Trans. Inf. Theory*, vol. 57, no. 8, pp. 5227–5239, Aug. 2011.

[18] I. Tamo, Z. Wang, and J. Bruck, "MDS array codes with optimal rebuilding," in *Proc. IEEE Int. Symp. Inf. Theory*, St. Petersburg, Russia, Jul. 2011, pp. 1240–1244.

[19] A. Vishwanath, J. Zhu, K. Hinton, R. Ayre, and R. S. Tucker, "Estimating the energy consumption for packet processing, storage and switching in optical-IP routers," in *Proc. Opt. Fiber Commun. Conf./Nat. Fiber Optic Eng. Conf.*, Anaheim, CA, USA, 2013, pp. 1–3.

[20] W. Li, Y. Yang, and D. Yuan, "Ensuring cloud data reliability with minimum replication by proactive replica checking," *IEEE Trans. Comput.*, vol. 65, no. 5, pp. 1494–1506, May 2016.

[21] R. Nachiappan, B. Javadi, R. N. Calheiros, and K. M. Matawie, "Adaptive bandwidth-efficient recovery techniques in erasure-coded cloud storage," in *Proc. 24th Int. Conf. Parallel Distrib. Comput.*, Turin, Italy, 2018, pp. 325–338.

[22] P. Subedi, P. Huang, T. Liu, J. Moore, S. Skelton, and X. He, "CoARC: Co-operative, aggressive recovery and caching for failures in erasure coded Hadoop," in *Proc. 45th Int. Conf. Parallel Process. (ICPP)*, Philadelphia, PA, USA, Aug. 2016, pp. 288–293.

[23] A. Kumar, R. Tandon, and T. C. Clancy, "On the latency and energy efficiency of distributed storage systems," *IEEE Trans. Cloud Comput.*, vol. 5, no. 2, pp. 221–233, Apr. 2017.

[24] W. Xu and Y. Lu, "Energy analysis of Hadoop cluster failure recovery," in *Proc. Int. Conf. Parallel Distrib. Comput., Appl. Technol.*, Taipei, Taiwan, Dec. 2013, pp. 141–146.

[25] J. Huang, F. Zhang, X. Qin, and C. Xie, "Exploiting redundancies and deferred writes to conserve energy in erasure-coded storage clusters," *ACM Trans. Storage*, vol. 9, no. 2, pp. 1–29, Jul. 2013.

[26] D. Harnik, D. Naor, and I. Segall, "Low power mode in cloud storage systems," in *Proc. IEEE Int. Symp. Parallel Distrib. Process.*, Rome, Italy, May 2009, pp. 1–8.

[27] S. Mitra, R. Panta, M.-R. Ra, and S. Bagchi, "Partial-parallel-repair (PPR): A distributed technique for repairing erasure coded storage," in *Proc. 11th Eur. Conf. Comput. Syst.*, Rome, Italy, Apr. 2016, pp. 1–16.

[28] M. Xia, M. Saxena, M. Blaum, and D. A. Pease, "A tale of two erasure codes in HDFS," in *Proc. 13th USENIX Conf. File Storage Technol.*, Santa Clara, CA, USA, 2015, pp. 213–226.

[29] J. Li and B. Li, "Demand-aware erasure coding for distributed storage systems," *IEEE Trans. Cloud Comput.*, vol. 9, no. 2, pp. 532–545, Apr. 2021.

[30] M. Vajha, V. Ramkumar, B. Puranik, G. Kini, E. Lobo, B. Sasidharan, P. V. Kumar, A. Barg, M. Ye, S. Narayanamurthy, and S. Hussain, "Clay codes: Moulding MDS codes to yield an MSR code," in *Proc. 16th USENIX Conf. File Storage Technol.*, Oakland, CA, USA, 2018, pp. 139–154.

[31] Z. Shen, X. Li, and P. P. C. Lee, "Fast predictive repair in erasure-coded storage," in *Proc. 49th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Portland, OR, USA, Jun. 2019, pp. 556–567.

[32] H. Zhou, D. Feng, and Y. Hu, "Bandwidth-aware scheduling repair techniques in erasure-coded clusters: Design and analysis," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 3333–3348, Dec. 2022.

[33] Y. Bai, Z. Xu, H. Wang, and D. Wang, "Fast recovery techniques for erasure-coded clusters in non-uniform traffic network," in *Proc. 48th Int. Conf. Parallel Process.*, Kyoto Japan, 2019, pp. 1–10.

[34] L. Xu, M. Lyu, Z. Li, Y. Li, and Y. Xu, "Deterministic data distribution for efficient recovery in erasure-coded storage systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 10, pp. 2248–2262, Oct. 2020.

[35] Q. Gong, J. Wang, D. Wei, J. Wang, and X. Wang, "Optimal node selection for data regeneration in heterogeneous distributed storage systems," in *Proc. 44th Int. Conf. Parallel Process.*, Beijing, China, 2015, pp. 390–399.

[36] L. Hu, L. Han, Z. Xu, T. Jiang, and H. Qi, "A disk failure prediction method based on LSTM network due to its individual specificity," *Proc. Comput. Sci.*, vol. 176, pp. 791–799, Jan. 2020.

[37] D. Liu, B. Wang, P. Li, R. J. Stones, T. G. Marbach, G. Wang, X. Liu, and Z. Li, "Predicting hard drive failures for cloud storage systems," in *Proc. 19th Int. Conf. Algorithms Architectures Parallel Process.*, Melbourne, VIC, Australia, 2019, pp. 373–388.

[38] L. Pang, A. Alazzawe, K. Kant, and J. Swift, "Data heat prediction in storage systems using behavior specific prediction models," in *Proc. IEEE 38th Int. Perform. Comput. Commun. Conf. (IPCCC)*, London, U.K., Oct. 2019, pp. 1–8.

[39] Z. Wang, Z. Xie, and D. Tang, "An erasure code with low recovery-overhead based on a particular three-hierarchical redundancy structure," *Int. J. Netw. Secur.*, vol. 24, no. 5, pp. 965–974, Sep. 2022.

[40] L. Xu, M. Lyu, Q. Li, L. Xie, C. Li, and Y. Xu, "SelectiveEC: Towards balanced recovery load on erasure-coded storage systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 10, pp. 2386–2400, Oct. 2022.

[41] L. Luo, Y. Tan, D. Liu, M. Duan, W. Wang, Y. Wu, and X. Chen, "Lazy repair with temporary redundancy(LRTR): Reducing repair network traffic in erasure-coded storage," in *Proc. 19th ACM Int. Conf. Comput. Frontiers*, Turin, Italy, May 2022, pp. 85–93.

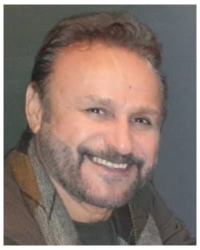
[42] Q. Yao, Y. Hu, X. Tu, P. P. C. Lee, D. Feng, X. Zhu, X. Zhang, Z. Yao, and W. Wei, "PivotRepair: Fast pipelined repair for erasure-coded hot storage," in *Proc. IEEE 42nd Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Bologna, Italy, Jul. 2022, pp. 614–624.

[43] Y. Wu, D. Liu, Y. Tan, M. Duan, L. Luo, W. Wang, and X. Chen, "LFPR: A lazy fast predictive repair strategy for mobile distributed erasure coded cluster," *IEEE Internet Things J.*, vol. 10, no. 1, pp. 704–719, Jan. 2023.

[44] H. Ahmadvand, F. Foroutan, and M. Fathy, "DV-DVFS: Merging data variety and DVFS technique to manage the energy consumption of big data processing," *J. Big Data*, vol. 8, no. 1, pp. 1–16, Dec. 2021.

REKHA NACHIAPPAN received the B.Sc. degree in mathematics from Alagappa University, India, in 2006, the Master of Computer Applications degree from Madurai Kamaraj University, India, in 2009, the Master of Philosophy degree from Prist University, India, in 2012, and the Ph.D. degree from Western Sydney University, Australia. Her research interests include cloud storage systems, decentralized storage systems, reliability, security, and fault tolerance.

RODRIGO N. CALHEIROS (Senior Member, IEEE) is currently an Associate Professor with the School of Computer, Data, and Mathematical Sciences, Western Sydney University, Australia. He conducts applied research in diverse aspects of distributed computing systems, including cloud computing, edge computing, and the Internet of Things. He has coauthored more than 70 papers, which attracted over 19,000 Google Scholar citations. He is a fellow of the Advance HE (formerly Higher Education Academy) and a Senior Member of the ACM.



KENAN M. MATAWIE received the Ph.D. degree from The University of New South Wales, Australia. He is currently a Senior Lecturer in statistics with the School of Computer, Data, and Mathematical Sciences, Western Sydney University, Australia. His research interests include application of statistics, particularly statistical modeling, advanced data analysis, and developing statistical methods motivated by real-world problems.

BAHMAN JAVADI (Senior Member, IEEE) received the M.S. and Ph.D. degrees in computer engineering from the Amirkabir University of Technology, in 2001 and 2007, respectively. He was a Research Fellow with The University of Melbourne, Australia. From 2008 to 2010, he was also a Postdoctoral Fellow with INRIA Rhone-Alpes, France. He has been a Research Scholar with the School of Engineering and Information Technology, Deakin University, Australia, during his Ph.D. course. He is currently an Associate Professor in networking and cloud computing with Western Sydney University, Australia. He is also a Co-Founder of Failure Trace Archive, which serves as a public repository of failure traces and algorithms for distributed systems. His research interests include cloud computing, performance evaluation of large-scale distributed computing systems, and reliability and fault tolerance. He is a member of ACM. He has received numerous best paper awards at IEEE/ACM conferences for his research papers. He served on a program committee of many international conferences and workshops.