Dynamic Single Node Failure Recovery in Distributed Storage Systems

M. Itani[a], S. Sharafeddine[b], I. ElKabbani[a]

[a] *Beirut Arab University, Beirut, Lebanon*
[b] *Lebanese American University, Beirut, Lebanon*

Abstract

With the emergence of many erasure coding techniques that help provide reliability in practical distributed storage systems, we use fractional repetition coding on the given data and optimize the allocation of data blocks on system nodes in a way that minimizes the system repair cost. We selected fractional repetition coding due to its simple repair mechanism that minimizes the repair and disk access bandwidths together with the property of uncoded repair process. To minimize the system repair cost we formulate our problem using incidence matrices and solve it heuristically using genetic algorithms for all possible cases of single node failures. We then address three practical extensions that respectively account for newly arriving blocks, newly arriving nodes and variable priority files. A re-optimization mechanism for the storage allocation matrix is proposed for the first two extensions that can be easily implemented in real time without the need to redistribute original on-node blocks. The third extension is addressed by implementing variable fractional repetition codes which is shown to achieve significant cost reduction. We present a wide range of results for the various proposed algorithms and considered scenarios to quantify the achievable performance gains.

Keywords: Distributed storage systems, fractional repetition codes, failure recovery, genetic algorithms, variable fractional repetition codes

1. Introduction

Data centers are becoming a top priority for businesses and have become critical for the very functioning of big enterprises. Any interruptions in the data center operations might cause huge losses if corrective measures for interruptions or failures were not considered[1, 2]. Data centers use inexpensive hardware components that are prone to failure. In a study that examines a 3000-node production cluster of Facebook, node failures spike to more than 100 in a single day with an average failure rate of 22 nodes a day [3].

Thus there is a great need for data protection from device failures, and for mechanisms to quickly recover or at least mask the effects of node failures from users and connected devices with minimum performance cost. The easiest way for a storage system to tolerate failures and prevent data loss is to store replicas. This, however, results in decreased storage efficiency. Another alternative is to store encoded data using erasure coding [4] . Classical erasure codes transform a given message into a longer one (codeword) such that the original message can be recovered from the codeword. Although traditional erasure codes can reduce the storage overhead as compared to replication, extensive network resources are needed to repair a lost node. This is due to the fact that a surviving node should read and process all its data, then send a linear combination of them to the replacement node causing huge bandwidth consumption. To minimize the bandwidth consumed during the repair process, regenerating codes were introduced in the literature where a failed node can be recovered by connecting to a subset of surviving nodes and downloading one block of data from each [5].

In this work, we consider a family of regenerating erasure codes that provide exact and uncoded repair where a surviving node reads the exact amount of data it needs to send to a replacement node without any processing. This allows for a low complexity repair process which is achieved by fractional repetition (FR) codes that were first introduced in [6]. We use FR codes in this work to minimize the total system failure recovery cost under various dynamic scenarios. The contributions of this work are the following:

- Generating an optimized block distribution scheme among the nodes of a given data center for fixed and variable size blocks. In the sequel, we refer to this as the Main Failure Recovery (MFR) problem.
- Dynamic optimization of storage allocation subject

Email addresses: may.itani@lau.edu.lb (M. Itani), sanaa.sharafeddine@lau.edu.lb (S. Sharafeddine), islam.kabani@bau.edu.lb (I. ElKabbani)

to the arrival of new data blocks. In the sequel, we refer to this as the Main Failure Recovery with Newcomer Blocks (MFR-NB) problem.

- Dynamic optimization of storage allocation subject to the addition of new storage nodes. In the sequel, we refer to this as the Main Failure Recovery with New-comer Nodes (MFR-NN) problem.
- Generating an optimized block distribution scheme among the nodes by accounting to varying priorities among data blocks. In the sequel,we refer to this as the Main Failure Recovery with variable Priority Files (MFR-VPF) problem.

A full system recovery plan that minimizes the total system repair cost is generated for the above four problems. We demonstrate the performance of the suggested algorithms through simulation results for various scenarios.

The rest of the paper is organized as follows. In Section 2, we provide a summary of the literature. In Section 3, we define our system model followed by the main problem formulation in Section 4. We describe the three dynamic recovery schemes in Section 5. Furthermore, we provide our genetic algorithms description and implementation for the four problems in Section 6. In Section 7, we provide simulation results with analysis. Finally, we conclude our work in Section 8.

2. Related Work

Several erasure codes [8, 9, 10] are being designed for the purpose of optimizing performance in distributed storage systems. Performance metrics include storage space, reliability and recovery cost. Of these codes, regenerating codes were first proposed by Dimakis as a new paradigm in coding that achieves minimum bandwidth requirements [5]. However, in case of node failure, regenerating codes require a helper node to read all its data and generate a linear combination of them to send it to the new-comer node. In this way the recovery approach does not satisfy the uncoded repair property [6].

Later in 2010 constructions of FR codes were introduced by Rouayheb et al [6]. Of these, deterministic constructions based on regular graphs and Steiner systems were presented as a first step in the study of fractional repetition codes, where a helper node reads only one of its stored packets and sends it to the new-comer with no processing. Pawar et al proposed a randomized construction of FR codes based on the balls and bins probabilistic model [11]. These randomized constructions provided more flexibility in terms of possible system parameters compared to those constructions of El-Rouayheb et al. Both constructions take different design considerations into account and are based on mathematical models that do not provide an optimal block storage scheme to achieve the goal of minimal recovery cost. Zhu et al in [18] adopted group divisible designs to generate other versions of fractional repetition codes; variable fractional repetition (VFR) codes, with variable replication degrees that replicate native packets more than parity check packets.

As to recovery approaches, several studies were conducted on the failure recovery problem in distributed storage systems. Zhu proposed a recovery solution for distributed systems that use row diagonal parity (RDP) and EvenOdd RAID6 erasure codes. Single node failures can be recovered by reconstructing the data of a failed node using decoding techniques [12]. Another work was conducted on Facebook warehouse clusters, where a framework was designed for a recovery approach based on codes that are efficient in the amount of data read and downloaded during node-repair. The basic idea behind this framework was to take multiple stripes of existing data on nodes and carefully add functions of the data of one stripe to other stripes. This technique required extra parities and computation of specific functions, and thus used coded repair [13]. As we observe, the above two approaches minimize the amount of data read during repair but require extra computations and decoding. The most recent and relevant work is that of Yu [14]. The author proposed a new version of FR codes, irregular FR codes, where different storage capacities of nodes, different communication costs, and different number of packets per coded block are assumed. The design considerations require selection of helper nodes from a limited set of nodes determined using hyper graphs.

In this work, we address single node failures in distributed storage systems with focus on dynamic operation to account for practical scenarios. The MFR-NB and MFR-NN problems provide ease of managing the network under the condition of achieving optimized recovery cost. They constitute tolerating new blocks on the current system nodes without varying the optimal allocation of current blocks, and allowing for network extensions via adding nodes and selecting their optimal load configuration again without interrupting the current optimal allocation. The MFR-VPF problem is a new implementation of VFR codes where different repetition degrees are assigned for more popular blocks allowing better real-time availability. The problems are modeled and solved using the concept of incidence matrices such that all nodes in the network are candidates for being helper nodes to any failed node.

3. System Model

Data center infrastructure design is based on a layered approach where issues regarding improving scalability, performance, flexibility, resiliency, and maintenance are considered. The three functional layers of the data center are the core layer, aggregation layer and access layer [15] and they are described below:

1. The core layer is central to the data center network and provides interconnection between the aggregation layers. It uses high performance low latency

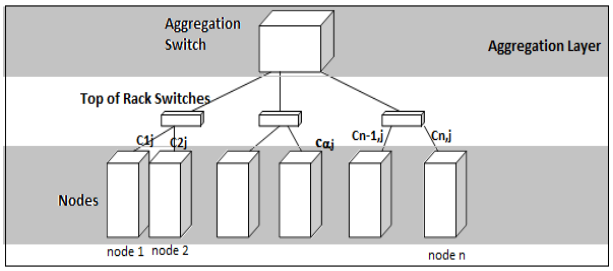

Figure 1: Distributed Storage System Diagram

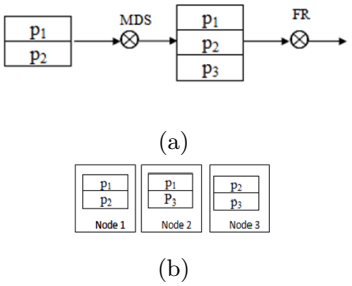

Figure 2: (a) An FR code with repetition degree ρ =2. A file consisting of k=2 packets is encoded using (3,2) MDS code. (b) θ =3 distinct encoded packets are replicated and distributed on n=3 nodes.

switches providing high densities that operate only on layer 3 devices.

2. The aggregation layer acts as a services layer for the data center. Services such as load balancing, SSL (Secure Socket Layer Optimization), firewalling, etc are typically found at this layer. It is used also as an interconnection point for multiple access layer switches.
3. The access layer is where the servers are physically attached to the network. This layer contains modular switches that afford layers 2 and 3 topologies and allow future proofing the network [15].

We consider an (n,k,d) distributed storage system where n is the total number of storage nodes, k is the total number of nodes contacted to retrieve a given file (k < n), and d is the number of nodes contacted to replace a failed node during node repair (d $\geq$ k) [6]. Our system model is depicted in Fig. 1 where different nodes are connected via top of rack switches and the retrieval cost of a block j from a helper node α is represented by $c_{\alpha j}$. The underlying networked environment connecting the nodes may have different transmission bandwidths and topologies. Thus each storage node will have different communication costs.

In our system, we use an FR code with repetition factor rho as the redundancy scheme. FR codes consist of a concatenation of an outer maximum distance separable (MDS) code and an inner fractional repetition code. First, a file of size k packets is encoded using a (θ;k) MDS code such that any k out of θ packets can recover the file (MDS property). Then θ distinct packets are replicated ρ times and distributed on n storage nodes where each coded packet is replicated on distinct nodes. A user contacting k nodes can always decode the stored file and this is achieved by the MDS property of the outer code[6]. Fig. 2 shows an example of a two-packet file encoded using a (3,2) MDS code. Then the three generated packets are distributed and replicated on three different nodes. If node 1 fails then it can be recovered by downloading packet 1 from node 3 and packet 2 from node 2 in parallel. In general, a node failure can be repaired by constructing a new node that contacts a specific set of d nodes for repair depending on which node has failed. d represents also the minimum node storage capacity expressed in packets.

A fractional repetition code can be represented by a {0, 1}- incidence matrix $\underline{B}$ of dimensions n $\times$ θ defined by:

$$B_{i,j} = \begin{cases} 1, & \text{if block j} \in \text{node i} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

where rows represent storage nodes (n) and columns represent blocks (θ). These codes will be used in modeling the MFR, MFR-NB, and MFR-NN problems. As for the MFR-VPF problem, a new modification of FR codes partially adopted from [18] will be used; where higher priority blocks will be associated with higher repetition degrees. The variable fractional repetition code (VFR) is defined using two different replication factors $\{\rho_1, \rho_2\}$. Note that if $\rho_1 = \rho_2$ then VFR codes are identical to regular FR codes.

Given a distributed system with n nodes. θ distinct blocks are to be stored on n nodes with a given replication factor ρ. Every node can store a minimum of d blocks. Given different communication costs and link bandwidths between nodes, the problem of failure recovery with different failure scenarios is modeled and solved using incidence matrices.

4. The Main Failure Recovery (MFR) Problem

We will start with the main failure recovery problem where we consider the system model described above. We present the general problem formulation followed by a detailed example.

4.1. Problem Formulation

Let the communication cost matrix represent the cost of new-comer node β to retrieve block j from a helper node α. The MFR problem is modeled as follows:

$$\mathbf{RC_{opt}} = \min \sum_{i=1}^{n} \sum_{j=1}^{\theta} \min_{\substack{\alpha=1,2,\ldots,n \\ \alpha \neq i}} c_{\alpha j} \cdot x_{ij} \cdot x_{\alpha j} \quad (2)$$

subject to

$$\sum_{i=1}^{n} x_{ij} = \rho \qquad \forall \text{ block j} \quad (3)$$

$$\sum_{j=1}^{\theta} x_{ij} = d \qquad \forall \text{ node i} \qquad (4)$$

$$\sum_{j=1}^{\theta} s_j \leq SC \qquad \forall \text{ node i} \qquad (5)$$

We need to solve for a block assignment matrix B that minimizes the value of the retrieval cost (RC) in Equation (2). The retrieval cost constitutes the sum of single block retrieval costs per node and is generalized to handle all possible single node failure scenarios, where the helper node α holds a retrieval block j that satisfies $c_{\alpha j}$ minimal. Note that the helper node α is one of the nodes that have block j in common with the failed node i ($x_{\alpha j} \& x_{ij} \neq 0$).

$x_{ij} \in \{0, 1\}$ is a Boolean variable indicating that node i holds block j. The value RC is to be minimized under the condition that the block assignment matrix satisfies the constraints in Equations (3),(4), and (5) adopted from the FR code requirements, and other additional constraints explained next.

Equation (3) is a replication factor constraint where the total number of replicas of a specific block j in all nodes is ρ.

Equation (4) specifies the total number of blocks per node. Assuming different size blocks and a specific storage capacity per node, we should add the storage constraint (5), where s_j is the size associated with every block j and SC is the total storage capacity of each node in the system.

4.2. Detailed Example

To clarify our formulated problem, consider a distributed system with 6 nodes such that an encoded data object with 4 distinct blocks is to be stored. Assume that the system can tolerate 2 node failures for a replication factor of 3. Each block will be replicated on three different nodes and each node will store two distinct blocks. The communications costs of different nodes need not be the same. Block sizes may or may not be the same. An initial random block assignment matrix B where rows correspond to nodes and columns correspond to blocks would be as follows together with a given communication cost matrix C, is given below. The cost matrix associates with every block in a node a generic retrieval cost.

$$\mathrm{B} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad \mathrm{C} = \begin{bmatrix} 2 & 2 & 2 & 8 \\ 7 & 2 & 10 & 2 \\ 5 & 7 & 6 & 6 \\ 3 & 9 & 7 & 4 \\ 1 & 6 & 1 & 6 \\ 9 & 8 & 9 & 4 \end{bmatrix}$$

For the case of one node failure, assuming node 1 fails the optimal recovery scheme is to recover block 3 from node 3 and block 4 from node 4. If node 2 fails, it is optimal to recover block 1 from node 4 and block 2 from node 5, and so on as specified in retrieval plan matrix R.

$$\mathrm{R} = \begin{bmatrix} 0 & 0 & 3 & 4 \\ 4 & 5 & 0 & 0 \\ 4 & 0 & 1 & 0 \\ 3 & 0 & 0 & 5 \\ 0 & 2 & 0 & 4 \\ 0 & 2 & 1 & 0 \end{bmatrix}$$

The total recovery cost would be the cost of recovering all nodes individually taking into consideration all single node failure scenarios. The total optimized recovery cost for this distribution schema in this case will be $\mathrm{RC} = c_{33} + c_{44} + c_{41} + c_{52} + c_{41} + c_{13} + c_{31} + c_{54} + c_{22} + c_{44} + c_{22} + c_{33} + c_{44} = 45$. However, if the block distribution matrix B was given as:

$$\mathrm{B_{opt}} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

then the total recovery cost for all nodes would be $\mathrm{RC_{opt}} = 23$. Hence, there should be an optimal distribution of blocks and replicated blocks among nodes so that the system recovery cost would be minimal. The optimal recovery schema for the second block assignment matrix $\mathrm{B_{opt}}$ is given below in modified matrix R, where the new-comer node will replace the failed node by downloading respective blocks from the nodes specified in $\mathrm{R_{opt}}$.

$$\mathrm{R_{opt}} = \begin{bmatrix} 0 & 2 & 5 & 0 \\ 0 & 1 & 0 & 6 \\ 0 & 0 & 5 & 2 \\ 5 & 2 & 0 & 0 \\ 4 & 0 & 1 & 0 \\ 5 & 0 & 0 & 2 \end{bmatrix}$$

5. Dynamic Failure Recovery Schemes

The three dynamic failure recovery schemes that account for new-comer blocks, new-comer nodes, and variable priority files distribution are explained below.

5.1. Arrival of New Blocks (MFR-NB) Problem

For the case of new-comer blocks, once a block arrives, it is replicated ρ times and split among the nodes in an optimal way such that the original optimized block distribution matrix and the recovery plan are not changed. The new block locations are added on top of the optimized MFR problem plan as described in the next example.

Given the optimal block distribution scheme $\mathrm{B_{opt}}$, suppose a set of packets grouped in a single new block is to be stored in the system. Using the same concept of minimizing total system repair cost on top of the given optimized arrangement, the re-optimized block assignment matrix would be

$$B = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

given new retrieval costs $c_{ff6}{}^{T} = \begin{bmatrix} 3 & 1 & 4 & 8 & 9 & 8 \end{bmatrix}$.

The corresponding re-optimized recovery plan is evaluated as:

$$R = \begin{bmatrix} 0 & 2 & 5 & 0 & 2 \\ 0 & 1 & 0 & 6 & 1 \\ 0 & 0 & 5 & 2 & 0 \\ 5 & 2 & 0 & 0 & 0 \\ 4 & 0 & 1 & 0 & 2 \\ 5 & 0 & 0 & 2 & 0 \end{bmatrix} \quad C = \begin{bmatrix} 2 & 2 & 2 & 8 & 3 \\ 7 & 2 & 10 & 2 & 1 \\ 5 & 7 & 6 & 6 & 4 \\ 3 & 9 & 7 & 4 & 8 \\ 1 & 6 & 1 & 6 & 9 \\ 9 & 8 & 9 & 4 & 8 \end{bmatrix}$$

The interpretation of the above R would be the same as that of R_{opt} except for the new-comer block which should be retrieved from node 2 if nodes 1 or 6 fail and from node 1 if node 2 fails. This will be the best practical solution that will take minimal computations which allows it to be implemented on the spot.

We will use the same system parameters as the MFR problem for the practical case of new block arrivals. The MFR problem is extended so that the cost to be minimized will include the new block retrieval costs for new comer blocks.

The re-optimization cost metric is expressed as follows:

$$RC_{MFR-NB} = RC_{opt} + \min \sum_{i=1}^{n} \sum_{j=\theta+1}^{\theta'} \min_{\substack{\alpha=1,2,\ldots,n \\ \alpha \neq i}} c_{\alpha j} \cdot x_{ij} \cdot x_{\alpha j} \tag{6}$$

where θ' is an integer such that $\theta' - \theta$ is the number of new-comer blocks. If $\theta' = \theta + 1$ then there is only one new-comer block and so on. Equation (6) is used in Algorithm 3.

5.2. Arrival of New Nodes (MFR-NN)

For the case of new-comer nodes, once a node arrives, it will be loaded with d blocks extracted in an effective way from the current nodes in the system.

Again, given the optimal block distribution scheme B_{opt}, we implement an algorithm that generates a row assignment vector for the new-comer node. Our algorithm selects the best entry swapping criteria that minimizes the total system repair cost. For the previous example given new retrieval costs $c_{7j} = [3\ 1\ 4\ 8]$, the best new comer node load generated by our algorithm and saved on top of previous optimized arrangement will be as follows:

$$B = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

and the optimal recovery plan will be modified as follows:

$$R = \begin{bmatrix} 0 & 7 & 7 & 0 \\ 0 & 7 & 0 & 6 \\ 0 & 0 & 1 & 2 \\ 5 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 \\ 5 & 0 & 0 & 2 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

For this practical setting of new nodes entering the system, new rows will be added to the optimized block assignment matrix and specific entries of optimal matrix columns will be modified. Modifications depend on the location of blocks in the newly added nodes (represented by rows) together with the corresponding column recovery cost (CC) of specific blocks (represented by columns) generated as follows:

$$CC_{MFR-NN} = \sum_{i=1}^{n+n'} \min_{\substack{\alpha=1,2,\ldots,n,n+n' \\ \alpha \neq i}} c_{\alpha j} \cdot x_{ij} \cdot x_{\alpha j} \tag{7}$$

where n' is an integer that represents the number of new-comer nodes. Equation (7) is used in our greedy Algorithm 4.

5.3. Variable Priority Files (MFR-VPF)

As for the case of variable priority files, the problem constitutes assigning blocks on nodes with variable repetition degrees based on priority. The objective function will be that of Equation (2). However, constraint (3) is substituted by constraints (8) and (9).

$$\sum_{\substack{i=1 \\ j \in \theta_1}}^{n} x_{ij} = \rho_1 \tag{8}$$

$$\sum_{\substack{i=1 \\ j \in \theta_2}}^{n} x_{ij} = \rho_2 \tag{9}$$

where θ_1 represents the set of blocks with higher priority that should be replicated ρ_1 times, and θ_2 represents the set of blocks with less priority that should be replicated ρ_2 times.

$$\rho_1 > \rho_2, \theta = \theta_1 + \theta_2 \tag{10}$$

The four failure recovery problems are integer linear programming problems and cannot be solved optimally in polynomial time for large network sizes for large network size. If the storage network consists of no more than 10 nodes, it roughly needs one week to get an optimal solution of our ILP (averaged result over 100 runs). If there are 10 20 storage nodes, it roughly needs more than one month to solve our ILP optimally (brute force) $O(2^n)$. But our heuristic can get an estimation of the minimum system repair cost within few minutes when there are more than 50 storage nodes. The next section will present the heuristic approach used to tackle the problems.

6. Genetic Algorithm Based Solution and Implementation

The implemented code for the solution methodology was designed as a self-cross-over genetic algorithm that starts with a random distribution of blocks on nodes and then searches within the feasible space by redistributing the blocks and generating a close-to-optimal solution [16, 17]. A description of the algorithm phases is stated next.

6.1. Chromosome Representation

The modeling of a chromosome was that each one constitutes a binary block assignment matrix generated using different permutations and satisfying the constraints (2), (3) and (4). Every chromosome represents a feasible solution. An example of a chromosome representation for the values (n=3, d=2, θ=3, ρ=2) is [1 0 1 1 1 0 0 1 1] where the n$\times\theta$ block assignment matrix is transformed to a single (n$\times\theta$,1) row matrix.

6.2. Generation of Initial Population

The phase that follows is generating an initial population for reproduction after carrying out the chromosome encoding phase. Given a pre-specified population size p, the initial population will include p chromosomes that are generated at random using a self-designed constructive method implemented as a function that generates feasible allocation scheme.

6.3. Self Cross-over and Mutation Operations

The conventional cross-over technique cannot be applied in our model since it will result in an in-feasible new chromosome. This paper adopts the self-cross-over method inspired by the fact that the feasible allocation matrices can be generated from one another by exchanging rows within a single matrix. The implemented cross-over procedure was done as specified in Algorithm 1.

Algorithm 1. Self Cross-over Implementation

1. Select single parent chromosome from population based on fitness function
2. Generate one, or two cross-over points (either random or specified by user)
3. Genes between cross-over points move to the off-spring swapped.
4. Repeat steps 1 to 3 till we generate a certain number of off-springs.

Note that elitism was used to help keep up the optimal chromosome in every generation. Mutation is also used to maintain diversity in the population and to make sure that the achieved solution is not a local optima. In mutation, the whole chromosome bits are swapped as if the chromosome is read from right to left as follows:

original chromosome: [1 0 1 1 1 0 0 1 1]
mutated chromosome: [1 1 0 0 1 1 1 0 1]

6.4. Fitness Function

Every chromosome or individual is associated with a fitness function equivalent to the optimization problem objective function defined in Equation (2) and calculated using Algorithm 2.

Algorithm 2. Fitness Function Calculation

Given a chromosome that represents a binary block assignment matrix, perform the following:

1. for $i = 1$, while $i < n$, i++
2. Let i =1 be the failed node
3. $\forall$ block j belongs to node i
4. Repeat
5. Check for all nodes α such that node α has a replica of block j
6. Select the node α with minimum retrieval cost $c_{\alpha j}$
7. Assign α as one of the helper nodes for failed node i and save it in recovery plan matrix
8. Update node i retrieval cost value to be $\sum_{j=1}^{\theta} \min_{\substack{\alpha=1,2,\ldots,n \\ \alpha\neq i}} c_{\alpha j} \cdot x_{ij} \cdot x_{\alpha j}$
9. Update recovery plan to include helper nodes for recovering all blocks j of node i
10. End of inner loop
11. Update total retrieval cost value for all nodes to be $\sum_{\substack{i=1 \\ \alpha\neq i}}^{n} \sum_{j=1}^{\theta} \min_{\substack{\alpha=1,2,\ldots,n \\ \alpha\neq i}} c_{\alpha j} \cdot x_{ij} \cdot x_{\alpha j}$
12. Update system recovery plan for next failure scenario
13. i=i+1
14. End of outer loop

Algorithm 3 accounts for new-comer blocks and generates a post-optimal recovery plan with the best possible distribution of new-comer blocks on the system nodes.

Algorithm 3. New-comer Blocks Allocation

Given a chromosome that represents an optimal binary block assignment matrix, perform the following:
for $j = 1$, while j <number of new-comer blocks, j++

1. Replicate block j , ρ times
2. Distribute replicas of j by augmenting the optimal block assignment matrix that was generated based on best fitness using Algorithm 2
3. Calculate the fitness of the newly augmented column
4. Update the value of the optimal fitness; i.e. cost generated by Algorithm 2
5. Select chromosomes with best fitness
6. Apply cross-over and mutation operations only on augmented columns of chromosomes with best fitness
7. After 20 or more generations, select the best chromosome that has minimal retrieval cost and update the original optimized block allocation schema together with the optimized recovery plan to account for the new-comers
8. Repeat steps 1-7 for all new-comer blocks

Algorithm 4 is a greedy algorithm that uses the optimal distribution generated from Algorithm 2. It modifies and extends it to account for new comer nodes and solve the MFR-NN problem as provided next.

Algorithm 4. New-comer Nodes Configuration

Given a chromosome that represents an optimal binary block assignment matrix, perform the following:
for $k = 1$, while k <number of new-comer nodes, k++

1. Distribute d blocks on node k randomly and augment them to the optimal matrix
2. for all blocks j of the d blocks present on new-comer node k
3. Select the corresponding column j for all current nodes
4. Check which set of ρ nodes best suits for allocating block j based on minimum cost value CC of Equation (7)
5. Repeat steps 2-4 for all d blocks present on node k
6. Calculate the new fitness value based on the sum of all minimum column costs
7. Select chromosomes with best fitness.
8. Apply cross-over and mutation operations only on augmented rows of chromosomes with best fitness
9. After a specific number of generations, select the optimal chromosome and update the original optimal block allocation scheme together with the optimal recovery plan to account for the new-comers
10. Repeat steps 1-9 for all new-comer nodes

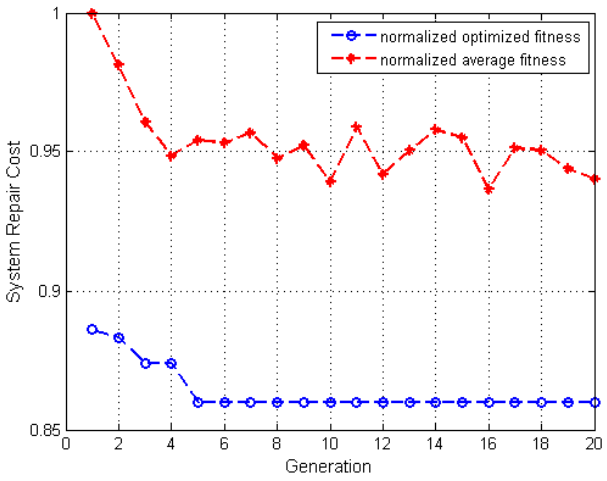

Figure 3: Average and minimum system repair cost for n=10, θ=50, ρ=4

As to the MFR-VPF problem, it is solved heuristically with the same fitness function as that stated in Algorithm 2. However, the difference is in generating feasible binary block assignment matrices that satisfy constraints (8) and (9) to have feasible candidate solutions, and then use these candidates in the algorithm to select the optimal one.

7. Simulation Results and Analysis

In this section, we present the results of our implementation for different cases of single node failures. The machine employed for simulation is a Lenovo laptop with an Intel (R) Core i7 CPU running at 2 GHz with 8 GB RAM. The operating system is Windows 7, and the computer is a 64-bit machine. The simulation programs were written in MATLAB.

The simulation results are averaged over 20 runs for each value of n ; i.e. a total of 80 runs and the post-optimal cost after distribution of new-comer blocks (Fig.6) and nodes (Fig. 7) is normalized by the maximum average cost for each generation for fair comparison.

To begin with, we present examples of our simulation results for the MFR problems in Fig. 3 and Fig. 4 respectively. Fig. 3 shows the convergence curve of the optimal repair cost together with the average cost for each generation given n=10 nodes, $\theta = 50$ blocks and replication factor $\rho = 4$. Fig. 4 shows the curve for values of n= 50 nodes, $\theta = 125$ blocks and replication factor $\rho = 2$.

As observed from Fig. 3 and Fig. 4, the average fitness of individuals decreases in each generation. This illustrates the fact that better solutions are being generated in newer populations. It is also clear that the optimal fitness, i.e., optimized system repair cost decreases till it converges to a minimum effective value. We can try to avoid being stuck in a local optima by increasing the mutation rate and re-running the algorithm.

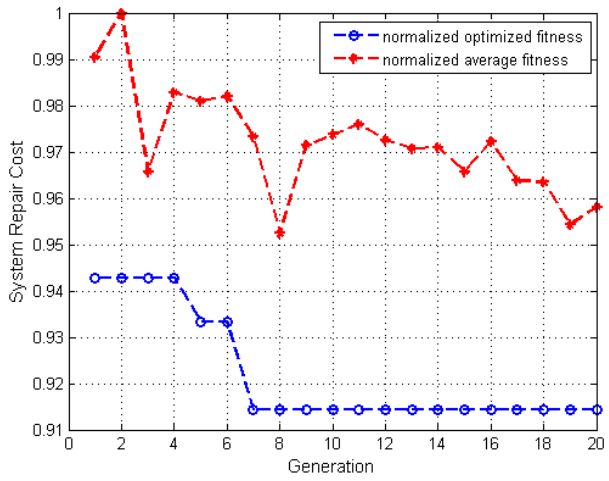

Figure 4: Average and minimum system repair cost for n=50, θ=125, ρ=2

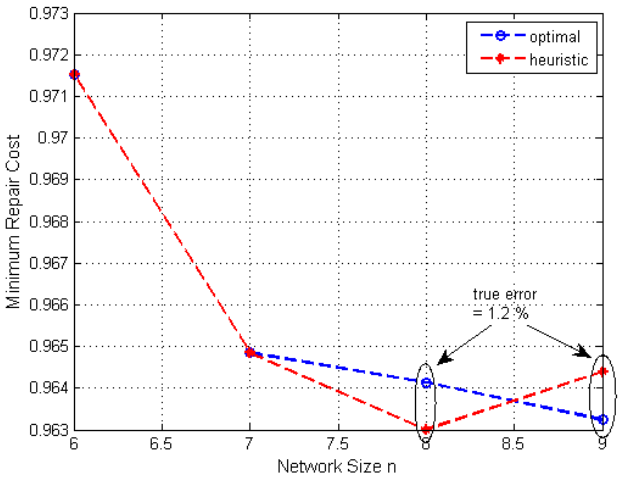

Figure 6: Minimum post-optimal system repair cost for different network sizes for the case of new-comer blocks

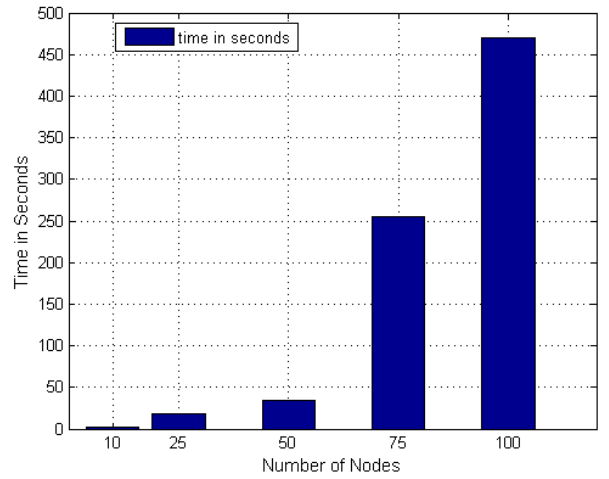

Figure 5: Average time to generate optimized minimum system cost for different number of storage nodes

Moreover, it is worth to state that the optimized solution is achieved at the fifth generation for n = 10 nodes (Fig. 3) and at the sixth generation when the number of nodes is increased to 50 nodes (Fig. 4). That shows the quick convergence of the algorithm even when we have large number of nodes.

To gain more understanding about the performance, we vary the parameters n, θ and ρ. We then increase the network size and measure its running time. The elapsed time for generating an optimized solution for a system with 10 storage nodes is 1.845 seconds, around 10 seconds for 25 nodes and 34 seconds for a system of 50 nodes. Our heuristic can get an estimation of the minimum system repair cost within few minutes when we have large network sizes (hundreds of nodes). This is demonstrated in Fig. 5.

We next compare the minimum system repair cost of our heuristic implementation for the case of new-comer blocks (MFR-NB problem) to that of the optimal brute force implementation for different network sizes and this is shown in Fig. 6. Our results are shown to be near optimal since the difference between the heuristic solution and optimal solution for the specific tested scenarios is calculated at most 1.2%.

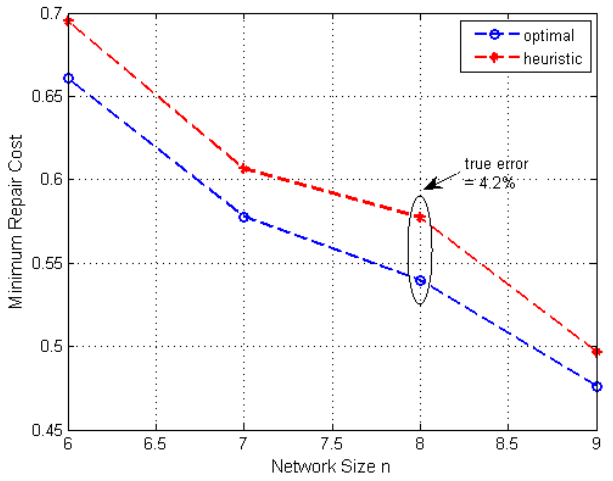

Figure 7: Minimum post-optimal system repair cost for different network sizes for the case of new-comer nodes

In Fig. 7 we compare the minimum system repair cost of our heuristic implementation for the case of new-comer nodes (MFR-NN problem) to that of the optimal brute force implementation for different network sizes. Our results are shown to be near optimal for the considered scenario since the difference between the heuristic solution and optimal solution is calculated at most 4.2%.

Fig. 8 and Fig. 9 show the system cost for different number of storage nodes in the cases of new-comer blocks and nodes respectively. We can interpret the decreasing

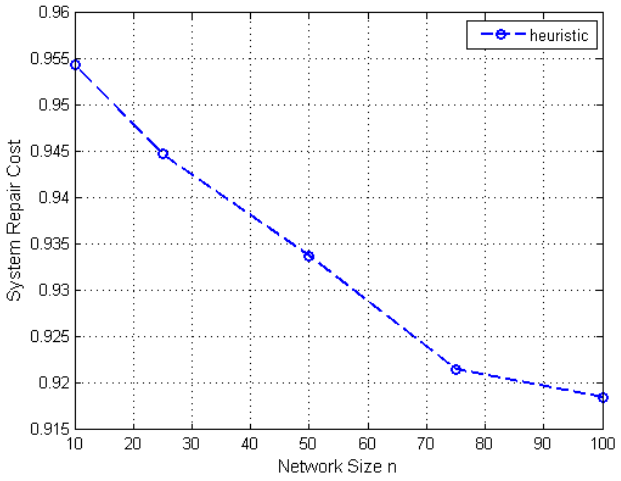

Figure 8: Normalized minimum system repair cost for different number of storage nodes in the case of new-comer blocks

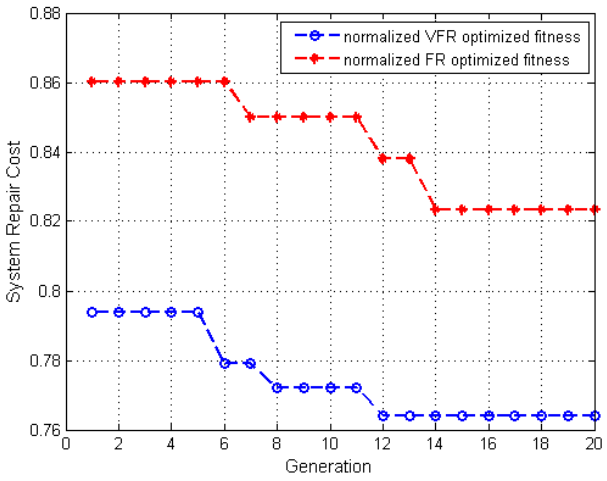

Figure 10: Normalized optimal recovery cost for n=20, $\theta = 50$, $\rho = 4$, $\rho_1 = 4$, $\rho_2 = 3$

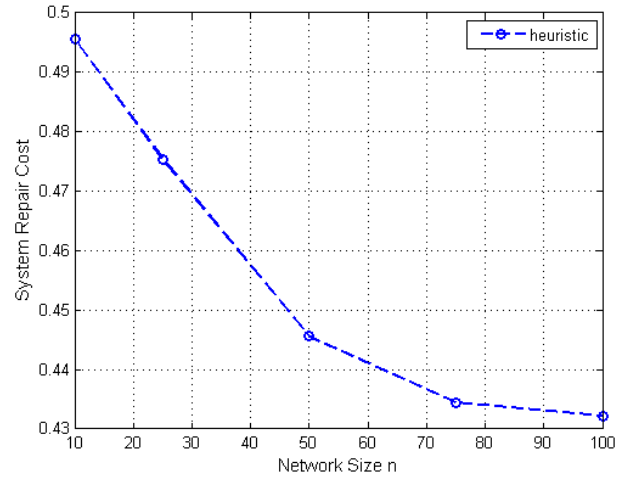

Figure 9: Normalized minimum system repair cost for different number of storage nodes in the case of new-comer nodes

result due to the fact that as the network increases, the repair cost becomes higher, thus the normalized cost decreases.

Fig. 10 shows simulation results for the recovery problems implementing FR and VFR codes respectively for the same cost matrix. The chosen parameters for simulation were 20 storage nodes and 50 different blocks. For the case of FR code, all blocks were replicated 4 times. Whereas, for the case of VFR code, 20 of the blocks were replicated 4 times and 30 were replicated 3 times. This shows a significant difference in cost where VFR costs can achieve same availability for specific blocks obviously with less costs.

8. Conclusion

We have presented the design of a main failure recovery approach that minimizes the system repair cost in case of single node failures in a distributed storage system together with three practical dynamic settings. We have proposed and evaluated an optimized solution to the four failure recovery problems using FR codes. FR codes provide a simple repair mechanism that minimizes the repair and disk access bandwidth together with the property of un-coded repair process. We formulated the main problem using incidence matrices and solved it heuristically using a genetic algorithm for all possible scenarios of single node failures. We then presented and solved the three practical variations of the main problem that respectively account for new-comer blocks, new-comer nodes and variable repetition factors. A post-optimal storage allocation matrix for the first two practical scenarios is computed in a way that can be easily implemented in reality without the need to redistribute original on-node blocks. The third new problem that constitutes implementing VFR codes showed significant cost reduction over FR codes. We have carried out extensive simulations and showed that our proposed approach exhibits a promising performance.

References

[1] A. Carroll, "Why Data Centers are Necessary for Enterprise Businesses," 2013, doi: http://www.lifelinedatacenters.com/data-center/why-data-centers-are-necessary-for-enterprise-businesses/

[2] B. Lavallee, "Proliferation of Remote Data Centers Creates New Networking Challenges," 2015, doi: http://www.datacenterknowledge.com/archives/2015/05/06/proliferation-remote-data-centers-creates-new-networking-challenges/

[3] R. Harris, "Facebook's Advanced Erasure Codes," 2013, doi: http://storagemojo.com/2013/06/21/facebooks-advanced-erasure-codes/

[4] J.S. Plank and M. Blaum, "Sector-Disk (SD) Erasure Codes for Mixed Failure Modes in RAID Systems," *ACM Trans. on Storage (TOS)*, vol. 10, no. 1 , pp. 4, Jan. 2014.

[5] A. Dimakis, P. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network Coding for Distributed Storage Systems,"

IEEE Trans. on Information Theory, vol. 56, no.9, pp. 4539-4551, 2010.

[6] S. ElRouayheb and K. Ramchandran, "Fractional Repetition Codes for Repair in Distributed Storage Systems," *48th IEEE Annual Allerton Conference on Communication, Control and Computing,* 2010.

[7] O. Olmez and A. Ramamoorthy, "Constructions of fractional repetition codes from combinatorial designs," *IEEE Asilomar Conference on Signals, Systems and Computers,* pp. 647-651, Nov 2013.

[8] O. Khan, R.C. Burns, J.S. Plank, W. Pierce and C. Huang, "Rethinking Erasure Codes for Cloud File Systems: Minimizing I/O for Recovery and Degraded Reads," *FAST Proc.,* pp. 20, Feb. 2012.

[9] M. Li and P.P. Lee, "STAIR Codes: A General Family of Erasure Codes for Tolerating Device and Sector Failures in Practical Storage Systems," *ACM Trans. on Storage (TOS),* vol. 10, no. 4, pp.14, Oct. 2014.

[10] D.S. Papailiopoulos, J. Lou, A.G. Dimakis, C. Huang and J. Li, "Simple Regenerating Codes: Network Coding for Cloud Storage," *IEEE INFOCOM Proc.,* pp. 2801-2805, March 2012.

[11] S. Pawar, N. Noorshams, N. ElRouyaheb and K. Ramchandran,"DRESS Codes for the Storage Cloud: Simple Randomized Constructions," *Proc. of IEEE on Information Theory (ISIT),* pp. 2338-2342, July 2011.

[12] Y. Zhu, P.P. Lee, L. Xiang, Y. Xu, and L. Gao,"A Cost Based Heterogeneous Recovery Scheme for Distributed Storage Systems with RAID-6 Codes," *42nd IEEE/IFIP Intl. Conf. on Dependable Systems and Networks,* pp. 1-12, June 2012.

[13] K.V. Rashmi, N.B. Shah, D. Gu, H. Kuang, D. Borthakur and K. Ramchandran, "A Solution to the Network Challenges of Data Recovery in Erasure Coded Storage Systems: A Study on the Facebook Warehouse Cluster," *UNISEX Hotstorage,* 2013.

[14] Q. Yu,C.W. Sung, and T.H. Chan, "Irregular Fractional Repetition Code Optimization for Heterogeneous Cloud Storage," *IEEE Journal on Selected Areas in Communications,* vol. 32, no. 5, pp. 1048-1060, 2014.

[15] CISCO, "Cisco Data Center Infrastructure," 2011, doi: http://www.cisco.com.

[16] S. Hou, Y. Liu, H. Wen and Y. Chen, "A Self-crossover Genetic Algorithm for Job Shop Scheduling Problem," *IEEE Intl. Conf. on Industrial Engineering and Engineering Management,* pp. 549-554, Dec. 2011.

[17] M. Negnevitsky, "Artificial Intelligence: A Guide to Intelligent Systems," Pearson Education, 2005.

[18] B. Zhu, H. Li, H. Hou, and K.W. Shum,"Replication-based distributed storage systems with variable repetition degrees," *IEEE Twentieth National Conference on Communications (NCC),* pp. 1-5, Feb 2014.