Near-Optimal Mixed Partial Replications versus Uniform Replication

Ralf Vamosi[1][0009−0002−8712−6564]

Faculty of Computer Science, University of Vienna, Vienna, Austria
https://informatik.univie.ac.at
ralf.vamosi@cs.univie.ac.at

Abstract. Replication involves creating and managing multiple duplicates of objects on various storage devices, workers, or computers. Replication is widely used in RAID (Redundant Array of Independent Disks) systems, database systems, storage clusters, cloud storage networks, compute clusters, etc. The main goal of duplicating, known as replicating, is to enhance the availability of the data, thus improving fault tolerance and performance. The replication factor (degree of duplication, multiplicity) determines the number of copies or replicas of objects, indicating the level of redundancy and thus giving a guarantee. In many cases, such as multi-disk storage, distributed file systems, and database systems, policies fix the replication factor (3, for example) to balance storage expenses and performance. This research presents a global search with an evolutionary genetic algorithm for finding multiplicities of partial replications to fit the system and improve costs.

Keywords: replication · placement · optimization · evolution

1 Introduction

Replication involves generating duplicates or replicas of objects (e.g., files) and managing them in terms of access, updates, and lifetime. Multiple locations or nodes guarantee redundancy and accessibility. Replication and placement are part of data management that improve metrics such as latency, transfer rates, and local load and ensure fault tolerance. *Data tiering* is a placement technique that aims to classify objects into tiers. Hot objects (e.g., data) used or read more frequently are placed in the high tier. High-tier locations or sites provide high bandwidth or faster storage, increasing performance. The different tiers depend on the definition of performance or goodness and so on a metric or measure of some dimension.

Replication defines how often objects are replicated/duplicated/copied (the term depends on the field and is sometimes arbitrary). The *replication factor* / multiplicity / degree indicates the cardinality of objects as replicas / copies / duplicates among various locations such as nodes, servers, caches, nodes etc. A replication factor (Rf) of 1 implies only one primary copy of a particular object(s). This scenario is often described as a single-replica or non-replicated

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97557-8_27

configuration. A replication factor of 2 provides a single-point failsafe, implying that a single node can fail and that the full extension or operation can still be provided. Higher-fold replications are chosen to improve this guarantee or expectation on dimensions such as average latency, bandwidth, throughput, etc. The number of replications required depends on the desired guarantee. It is based on cost-benefit analysis (trade-off) by architects, administrators, or users. Replication always incurs certain costs for storage, communication, and management.

The *replication problem* covers three partial problems:

1. The replication factor or multiplicity - how many copies shall be produced
2. Which subset of objects are subject to replication
3. Where should the objects (subset) be placed or located

This work shows a replication method for 1) and 2) cases in which 'optimal' placement does not play a role due to location independence or lack of information. There are different use cases for that:

- One case may be an entire storage provided as one tier in one device. One tier means equal performance for the entire placement. This can be a logical device such as a file server or a network-attached storage (NAS).
- In another case, no information is available on the nodes (servers, storages, caches) or tiers. This would include, for example, a case with an external, managed storage or with alternating use patterns of objects that hinder the application in improving the placement for cost decrease.

Contribution: This work presents an algorithm that, in principle, generates a near-optimal replication with a superposition of partial replications. This covers tasks 1) and 2). The novel result indicates that optimal replication consists of several replication factors applied to subsets. The optimization model can update the allocation by running on a new set of objects and, hence, combine the previous and current allocation outputs.
Assumptions: Objects to be replicated are i.i.d. Observed values for the cost function (network) are accurate (sufficient). This means that the regime does not change.

2 State-of-the-Art

Replication has been a fundamental aspect since the advent of distributed databases and parallel computing. It enhances availability by distributing resources across multiple endpoints to mitigate the effects of individual endpoints becoming overloaded, unresponsive, or offline. The term 'running hot' refers to the condition where an endpoint is subjected to excessive requests, overwhelming its processing capacity. Replication also improves runtime performance, evidenced by reductions in latency. Many systems use process and data replication [1] [2]. A form of replication is data replication, which manages copies/replicas for distribution to machines or locations. Data management offers techniques like tiering, caching, or replication [1] [2] [3]. Publications either focus on practical applications and

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97557-8_27

best operational practices or on theoretical work showcasing optimization benefits. Applied methods include simplifications, local and greedy optimization, and various models such as linear, deterministic, and stochastic.

Numerous studies demonstrate the NP-hard nature of optimally resolving data placement and replication problems. It is evident from the combinatorial growth that replication falls within NP. [4] illustrates this by converting a decision/satisfaction problem in replication to an NP-complete partitioning issue. Additionally, they contend that certain instances of the replication problem are non-approximable, necessitating the use of heuristics or simplifications.

To manage the complexity of replica placement, algorithms often simplify elements of the problem or its solution. Similar to other complex issues, local optimization is examined and applied to replication challenges. Some approaches integrate properties like caches, bandwidth, or input data. In the research by [3], a greedy cooperative cache management strategy is introduced, dividing the input set into N caches, each of which is optimized locally. A different greedy approach focused on cost metrics per object and network costs is introduced in [5]. This method uses a greedy replication algorithm to handle caching. Objects are ranked on the basis of their cost contributions, determining their placement in specific locations.

In [6], a segment of the cache servers' storage is dedicated to holding the targeted content data. The authors suggest enhancing these cache portions to strike a balance between improved network performance and computational expense.

A number of studies incorporate the concept of data popularity as a weighting factor. This popularity is often referred to as (local) probability, weight, or hotness. As noted in [3], popularity can be influenced by the user community. Additionally, a temporal aspect can affect popularity. According to [7], the probability of data utilization is examined over time and location, accounting for varying resources and user requests. This research also investigates an uncertain job dispatching mechanism that impacts the cost function.

For high-cardinality problems, the machine learning method of clustering may be used. Instead of dealing with single objects, similar objects are grouped into clusters that can be handled as input objects. [6] [7]

Multiple requests to websites, especially to those with high demand, can cause the site to become overwhelmed. Strategically placing duplicates in advance helps to spread the potential load during peak times [1]. The endpoints in consideration can be geographically dispersed, even on a global scale, such as in cloud or grid computing, where replication techniques are frequently used. Duplicating processes allows for parallel processing and enhances fault tolerance, while data replication is essential for ensuring fault resilience and boosting performance [2].

Database systems implement protection and replication strategies for failover and load balancing, ensuring high availability (HA) and fast data access. Data parallelization requires intricate coding within the database management system (DBMS) because the DBMS must write, read, and update data using multiple table replicas concurrently. [8] Managing and updating replicas necessitates a flexible approach.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97557-8_27

Data replication plays a vital role in RAID systems (Redundant Array of Independent Disks), which are utilized in personal computers, servers, and network-attached storage (NAS). The hardware involved may consist of various technologies like disk or solid-state storage. RAID configurations employ data stripping and replication to distribute data segments across multiple persistent storage units. The specific numbers vary based on the RAID level and system, guaranteeing data accessibility and integrity even during storage failures. Upon disk replacement, duplicate data is restored to the new disk to maintain secure and regular operation. [9] RAID continues to see active development, such as integration into Unix kernel modules [10].

Storage clusters must balance network load across nodes to ensure reliable, fast access through redundancy [11]. The Hadoop Distributed File System (HDFS) is intended for large volumes of files and stores files on commodity hardware, offering a large capacity for computations or video content. It is highly fault-tolerant and provides high data throughput due to replication. Files are partitioned into blocks, each replicated multiple times by default with a factor of 3, adjustable at the application level [12].

The vast landscape of grid computing, where data- and compute-intensive applications typically comprise multiple data centers, can be considered from a storage cost and energy cost perspective. [13] [14].

Some approaches select a fixed number of replicas (replication factor) based on constraints or best practices. These approaches miss the opportunity to vary the number of replications to expand the search space and eventually find a better solution.

3 Network and Cost

A network has several costs depending on the resources and user, such as CPU wall time, energy per request, communication cost, transfer time, etc. A specific communication cost is chosen to showcase the optimization algorithm, which is the latency of files that affects the delays. Such delays are central to many applications and perceived by users, especially for data-intensive computing or content networks.

Several network architectures exist depending on their use and extension, such as local, wide, and global networks. The network of content providers or grids may be hierarchical. Content providers have autonomous systems (AS) connected to internet service providers (ISPs) and add several caching servers to provide the (replicated) content. In grids, there may be some high-tier sites with connected low-tier nodes (nucleus) each.

An example network is depicted in Fig. 1. The network may be different in shape, which must be considered in the cost function.

This distributed network can be modeled as a graph in Fig.2. $client_x$ and $node_x$ represent nodes nearby at location x. This locality implies the shortest path between them modeled as one hop. Clients trigger actions on objects. For all M=8 nodes, storage is defined by capacity. The capacity values for those logical

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97557-8_27

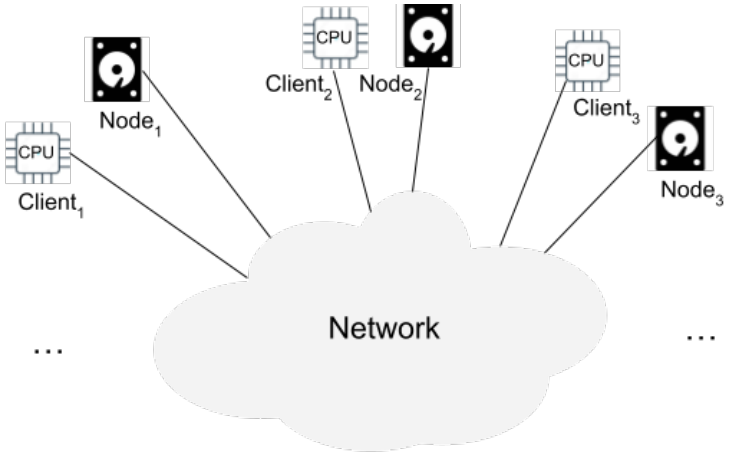

Fig. 1. Storage network connected to a network. Clients request objects from the storage nodes.

bins $bin_1, ..., bin_M$, $S_1, ..., S_M$, are randomly chosen in the interval $[0.2, 1]$ each. The capacity values are then normalized to provide the necessary capacity for all replicated objects. Only relative values and not absolute values are interesting.

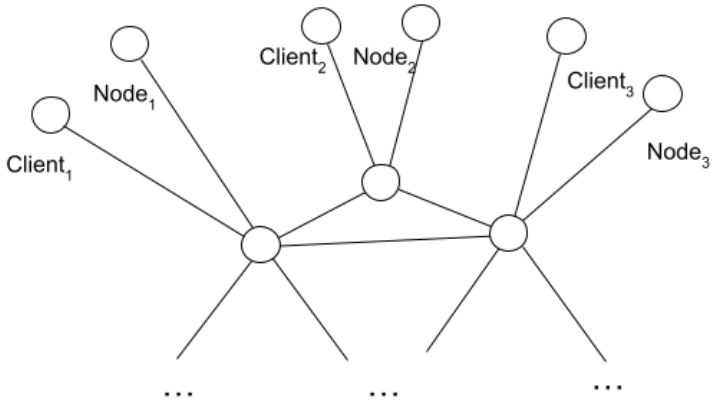

Fig. 2. Storage network connected to a dense network. Clients request objects from the storage nodes.

The core network is densely connected, so some inner nodes are interconnected, providing the communication network as would be for a *wide area network (WAN)*. Clouds, grids, communication networks, and content networks are

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97557-8_27

WANs that are part of the Internet. The experiment shall impose a variation of architectures on the optimization model. A link between two inner nodes is established with probability 0.5 (50 %), which decides whether a link exists. The inner nodes are routers/ core switches. The transfer rate depends on the transfer speeds of the involved nodes, so the hops the communication path extends.

The **use patterns** of objects affect the cost. Since read operations (retrieving a page, playing a video file, reading measurement data, etc.) are much more likely than write operations, the average ratio is set to 3:1 between read and writes, reflected as a factor of 3 in the probabilities. Read operations request object copies (pull), which have been improved due to replication. Let $\boldsymbol{P}^r = [P_1^r, P_2^r, \ldots, P_N^r]^T$ be the read probabilities of the object. T stands for transposed. Write operations update all the replicas of the related object, which results in a higher effort (cost). $\boldsymbol{P}^w = [P_1^w, P_2^w, \ldots, P_N^w]^T$ the write probabilities for objects with the same index. The probabilities express the demands for objects to be fetched or updated. The values are randomly sampled from a Zipf distribution. Zipf means that the probability of observed frequency is inversely proportional to a top-to-bottom order. The distribution is observed in many applications. Some items are frequently in demand, while the majority are rarely used. Since there is no location-dependent information on the clients, the cost function incorporates only a random client selection (uninformed).

A *replicated placement* of objects is defined by the parameter $\boldsymbol{A} = [a_{mn}]$. $a_{mn} = 1$ if and only if a replica of the object n is stored at m, 0 otherwise. The index n always points to the object (prototype) from which replicas are placed. The index m points to the location (endpoint, node, storage, etc.) that is virtually considered bin as depicted in Fig. 3 and Fig. 4. A cost is generally defined based on the system and the intended dimension on which to act and depends on $\boldsymbol{A}$. The total cost related to the selected replication, $c = c(\boldsymbol{A})$, factorizes into N terms, c_n, per object (prototype) n:

$$
\begin{aligned}
c(\boldsymbol{A}) =& \frac{1}{N} \sum_{n=1,\ldots,N} c_n(\boldsymbol{A}) = \\
=& \frac{1}{N} \sum_{n=1,\ldots,N} (P_n^r t_{mn}^r(\boldsymbol{A}) + P_n^w t_{mn}^w(\boldsymbol{A}))|_{m=client()} \\
\boldsymbol{A} =& \begin{bmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \vdots \\ \boldsymbol{a}_M \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M1} & a_{M2} & \cdots & a_{MN} \end{bmatrix} \quad a_{ij} \in \{0, 1\}
\end{aligned}
\tag{1}
$$

, in which $t_{mn}^r(\boldsymbol{A})$ is a composed cost function for read operations from client m to object n, and $t_{mn}^w(\boldsymbol{A})$ is a composed cost function for write operations analog as before, and $client : \Omega \to \{1, 2, \ldots, M\}$, with $\mathbb{P}(client = m) = \frac{1}{M} \quad \forall m \in \{1, 2, \ldots, M\}$ selects a random client. Before evaluation, M^2 minimum routes between M clients and M endpoints for t_{mn}^r and t_{mn}^w must be stored (number of hops as a predictor distance and thereby delay time).

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97557-8_27

$t^r_{mn}(\boldsymbol{A})$ seeks the replicas of the object n and outputs an approximate read latency. Since the bandwidth can be shared, the replication factor/multiplicity changes the weighing with the inverse of the replication number $1/|n|$. For each replica, the shortest route is selected, equal to a route with the minimal number of hops, and the transfer t is calculated by $t = s_n \cdot c^r_n \cdot |hops|/|n|$ where s_n is the size of the object n transferred, $|n|$ is the replication factor/multiplicity of n, c^r_n is a category-dependent cost multiplier, and $|hops|$ is number of hops between the source and the destination. Each category is given a random cost multiplier in $[.25, 1]$ at the beginning of each evaluation. Such categories may be based on the use profiles of user groups (personas) or resource dependency of the object. The calculated t's for all replicas of n are then averaged, obtaining the cost $t^r_{mn}(\boldsymbol{A})$.

$t^w_{mn}(\boldsymbol{A})$ seeks the longest of the shortest paths from the client m to all replicas of n and outputs an estimated update time. This is an approximated worst-case for updating replicas of n in a protocol with multi-peer communication. With the one selected path, the final update time is calculated as $t^w_{mn} = s_n \cdot c^w_n \cdot |hops|$ with the variables as before, but with c^w_n as a category-dependent cost multiplier for updates of n. This is also randomly selected for each category in $[.25, 1]$.

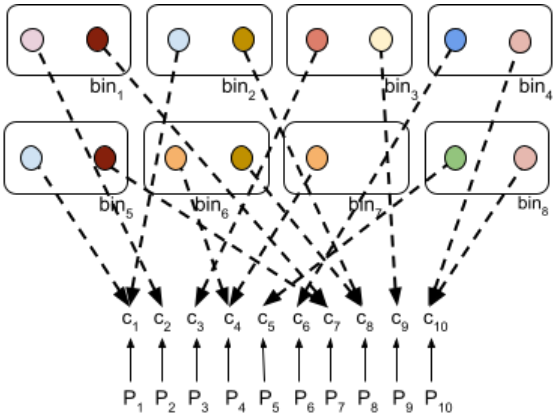

Fig. 3. Single cost values c_n dependent on topology, replication, and object properties.

4 Optimization

Replication is a selection problem with particular boundaries. Depending on the use case, an algorithm must generate sets of replicas and place them at locations such as sites or machines. If M is the number of bins (subsets), there are maximal M-fold replicated objects. For the replication factors $\boldsymbol{Rf} = [Rf_1, Rf_2{+}2, \ldots, M]$, the same number of replication subsets correspond. If all possible replication factors are computationally feasible, then $\boldsymbol{Rf} = [1, 2, \ldots, M]$ for M bins. A case

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97557-8_27

with eight bins and ten different objects is shown in Fig. 4. With these variables, the optimization task is the following:

$$
\begin{aligned}
&\underset{\hat{\boldsymbol{A}}}{\text{argmin}}\ c(\boldsymbol{A}_{t=1}) \\
&\quad \boldsymbol{Rf} = [Rf_1, Rf_1+1, \ldots, Rf_{max}] \quad \mathbb{N} \ni Rf_{max} \leq M > 1 \\
&\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathbb{N} \ni Rf_1 \geq 1 \\
&\quad \boldsymbol{A}_{t=1} = place(\hat{\boldsymbol{A}}) \quad \hat{\boldsymbol{A}} = \begin{bmatrix} Rf_1 \cdot \boldsymbol{a}_1 \\ (Rf_1+1) \cdot \boldsymbol{a}_2 \\ \vdots \\ Rf_{max} \cdot \boldsymbol{a}_{Rf_{max}} \end{bmatrix} \\
&s.t. \\
&\quad \langle \boldsymbol{A}_{t=1}, \boldsymbol{s} \rangle \leq \boldsymbol{S}
\end{aligned}
\tag{2}
$$

where $\hat{\boldsymbol{A}}$ are the partial replication sets $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_M$ of the objects denoted as a matrix, $\boldsymbol{A}_{t=1}$ the replicated objects placed to the bins, and the cost $c(\cdot)$ has been defined before. $\hat{\boldsymbol{A}}$ is not an allocation matrix as was introduced with Boolean entries since it also indicates the multiplicity/replication number. $\boldsymbol{s}$ is the sizes of the objects in vector form, $\boldsymbol{S}$ is the size vector of the bins.

In the dynamic case, objects are added (altered) to the system (others are removed). The number of nodes is constant, and the optimization considers a steady state in which the values of the system (properties of the system) do not change. Otherwise, the mapping of the bins and the cost function must be changed.

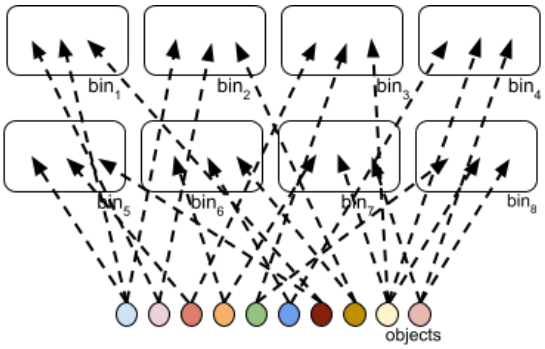

Fig. 4. Replication of objects into bins.

The optimization method presented separates the replication and placement steps for computational feasibility. The optimization model needs to perform the following steps:

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97557-8_27

1. The model selects feasible replication factors, $\boldsymbol{Rf} = [Rf_1, Rf_1{+}1, \ldots, Rf_{max}]$. The first replication factor, Rf_1, depends on the guarantee and may be one for persistence without fail safety or 2 with single-point fail safety. The maximum in $\boldsymbol{Rf}$ depends on the total capacity relative to the object sizes and the number of bins (subsets). An object in a bin has no cardinality other than 0 or 1. The search for subsets must occur within a sufficient but feasible range $\boldsymbol{Rf}$ to approach the global optimum.
2. The model selects partial subsets $[\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_M]$, corresponding to the numbers in $\boldsymbol{Rf}$. Each $\boldsymbol{a}_m$ has length N (objects) and indicates a selected object n with 1 at position n, otherwise 0. These vectors are Boolean and are disjoint from a set perspective. If an object has been selected for a replication factor, it cannot be selected again. A subset can be varied in size as long as the capacity allows, as was denoted in Equ. 2. The subsets $\boldsymbol{a}_1 + \boldsymbol{a}_2 + \cdots + \boldsymbol{a}_M$ are further related by the size condition imposed on them. The capacity occupied by a $\boldsymbol{Rf}$ is the weighted sum $Rf_1 \langle \boldsymbol{s}, \boldsymbol{a}_1 \rangle + \cdots + (Rf_1 + M) \langle \boldsymbol{s}, \boldsymbol{a}_M \rangle$ where $\langle \cdot, \cdot \rangle$ is the inner product.
3. After sets of replicas/copies are generated, placement is necessary in distributed systems. $place(\cdot)$ is defined as hash binning to be applied. Objects are assigned to bins (storage locations) to obtain the $\boldsymbol{A}_{t=1}$. Hash binning is a weighted placement: Objects are hashed into an extensive range of virtual bins mapped into the M bins in proportional numbers. If the capacities of all bins were equal, the same number of virtual bins would be assigned per bin. If there is no prediction on the client side reading and updating the objects, then this process cannot be improved. In a uniform system, where transactions are uniformly distributed (load balancing); just the replication factor determines the performance/cost. With a feasible placement/final allocation $\boldsymbol{A}_{t=1}$ (capacities $\boldsymbol{S}$), a cost value can be calculated to measure the goodness (fitness in an evolutionary context).

5 Method

$\boldsymbol{Rf}$ determines the number of subsets in the optimization and, therefore, the complexity of the search. Simplifications may be necessary depending on the length of $\boldsymbol{Rf}$. The complexity of the search increases exponentially with an increase in length of $\boldsymbol{Rf}$. This work leaves the optimal range open to discussion. There must be a balance of computational cost and accuracy. For the search of subsets to replication factors, an Allocation Optimization Model has been developed. Within this model, solution candidates are in-memory objects that use clustering for subset generation. There are $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_M$ subsets for which cluster methods are used. $\boldsymbol{a}_m$ can be further decomposed into subclusters $\boldsymbol{a}'_m, \boldsymbol{a}''_m, \ldots, \boldsymbol{a}_m^{max_m}$ whereas it always starts with one basic cluster $\boldsymbol{a}'_m$. A dynamic max_m increases the computational effort to vary the output subset $\boldsymbol{a}_m$. So, the allocation model should increase max_k of a random k and trial and error to improve the outcome. In the same way, it can decrease this number to make the search simpler. This is a subordinate random walk in the parameter

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97557-8_27

max_k in the evolutionary process. The decomposed subsets are generated by *clustering methods* illustrated in Fig. 5. These clustering methods are part of the allocation model of the solution candidate. The allocation model has controlled clustering based on a nearest-neighbor search. The control in the clustering means that the clustering parameters can be modified in different trajectories in the evolutionary process:

- The center of the cluster
- The size of the cluster can be varied within the size constraint under Eq.2 in which replication factors affect the output size of replicated subsets.

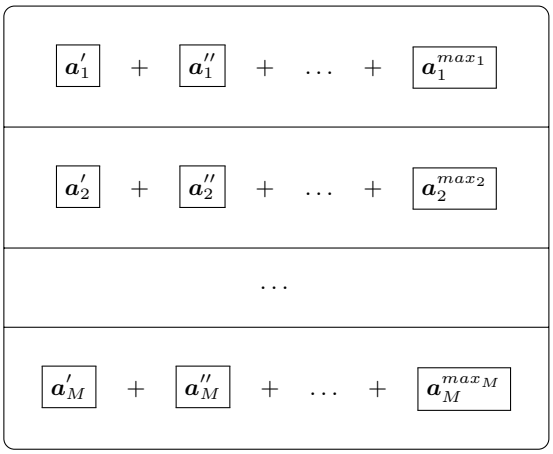

Fig. 5. The allocation model outputs subsets $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_M$ that are composed of subsets from clusterings.

For improving the solution to the nonconvex problem, the Allocation Optimization Model is based on an evolutionary algorithm. In an evolutionary context, multiple solution candidates live in a pool of candidates that communicate and compete with each other. The evolution contains candidate solutions. Evolutionary genetic techniques change how variation evolves. Evolution accelerates the convergence to an optimal solution with different operators, such as selection and propagation. The parameters are modified iteratively and shared among solution candidates. Candidates for a better solution are more likely to persist (chance). The solution candidates run in parallel (processes), and any solution may take the lead. The solution candidates in the evolution pool/population are simplified in Fig.6. The colored boxes represent the collected subsets.

Genetic/evolutionary operators must be adapted to the solution parameters to provide the full potential of the meta-optimization. The set of operators is at least the following:

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97557-8_27

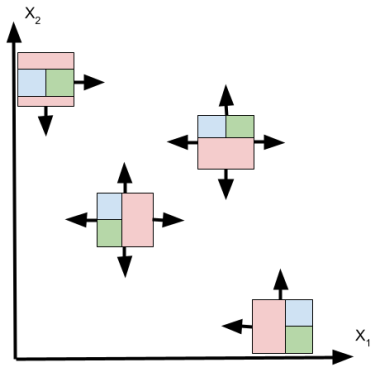

Fig. 6. Solution candidates 'walk' in their parameter space and probe the replication subsets (in colors).

- The selection operator selects candidate solutions from the population based on their fitness to pick winners and losers. Fitness is the inverse cost.
- Crossover or recombination combines parameters from two solutions to produce an offspring. A child z by single-point crossings x and y is represented as $z = (x_0, x_1, \ldots, x_{p-1}, y_p, y_{p+1}, \ldots, y_{n-1})$
  A child z from a 2-point crossover between x and y is represented as: $z = (x_0, x_1, \ldots, x_{p-1},\ y_p, y_{p+1}, \ldots, y_q,\ x_{q+1}, x_{q+2}, \ldots, x_{n-1})$ The exchanged parameters are the subsets of the allocation $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_M$ with their individual clusterings as illustrated in Fig. 5.
- Mutation or variation randomly changes the parameters to match the changing parameters to introduce diversity into the pool (random walk).
- Discard or annihilation selects candidate solutions to discard.

6 Justification

A validation has been carried out with the global optimization as discussed. All properties of the network model and objects are randomly sampled for each experiment run, as discussed in Section 3. The number of bins are fixed to 8. There are $N = 100'000$ objects that represent files in the considered showcase and, for which the following data are given, $\boldsymbol{X} = [\boldsymbol{s}, [\boldsymbol{k}]_{0/1}, \boldsymbol{P}^r, \boldsymbol{P}^w]$ with the columns size $\boldsymbol{s}$, columns category in Boolean encoding $[\boldsymbol{k}]_{0/1}$, read and write probabilities columns $\boldsymbol{P}^r, \boldsymbol{P}^w$ per object. Nominal features of the category are represented in 1-hot encoding.

The observed relative gains of the repetitions of the experiments are averaged. Each experiment consists of a run for each total capacity value as multipliers such as 1x, 2x, etc. ticked on the x-axis in Fig.7.

- The *complexity of the calibration* of the model is very high due to the number of iterations needed to probe the points in the search space. For each opti-

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97557-8_27

mization run, one million iterations and more are executed, but this number by far does not exhaust the search.
- With the calibration of the model, the *static case* is done.
- *Dynamic means* that objects are passed to the model that have been unseen before. Usually, new objects are inbound from a stream of data from users, computations, or measurements. The Allocation Optimization Model is aware of a previous placement, $\boldsymbol{A}_{t=0}$. An update, $\boldsymbol{A}_{t=1}$, must be complementary to the previous subsets defined with a prior $\boldsymbol{A}_{t=0}$. With this in mind, the final placement is $\boldsymbol{A}_{t=0} + \boldsymbol{A}_{t=1}$. This term is plugged into Equ. 2. This leads to argmin of the cost for the prior and update placements

$$\underset{\hat{\boldsymbol{A}}}{\operatorname{argmin}}\ c(\boldsymbol{A}_{t=0} + \boldsymbol{A}_{t=1})$$

. such that $\langle \boldsymbol{A}_{t=0} + \boldsymbol{A}_{t=1}, \boldsymbol{s} \rangle \leq \boldsymbol{S}$ whereas $\boldsymbol{S}$ must be updated for the additional allocation. The control variable $\boldsymbol{Rf}$ affects only $\boldsymbol{A}_{t=1} = \boldsymbol{A}_{t=1}(\boldsymbol{Rf})$.

The first step of the optimization model is to read a batch of 5, 6, 7, 8, 9, and 10 % of input objects (calibration set). Small batch inputs are desirable to achieve the number of optimization iterations or evolution steps. The entire data population is used for validation in the static replication case. The best optimization run among these batches is selected. Fig.7 shows the yield versus the benchmark case with a uniform global replication factor (1) measured by the defined cost function. There are three cases in total:

1. Uniform global replication is the benchmark with one replication factor at a time, $\boldsymbol{Rf} = [Rf']$ for $Rf' = 1, 2, \ldots, 8$ as depicted on the x-axis in Fig.7. The replication factor Rf' defines the total capacity in terms of the order of magnitude of the size of the input objects for all cases (1.), (2.), and (3.). The cost values are normalized to 1 (0 %).
2. The search includes all possible replication factors for the M = 8 bins, $\boldsymbol{Rf} = [1, 2, 3, 4, 5, 6, 7, 8]$. The **static case** covers all objects at once. The entire set is the validation set. The **dynamic case** means that the capacity is extended by the size of the added objects (50% of objects in the first static run and 50% of objects in the dynamic run). The proportion of bin capacities is conserved and could be changed due to the adaptability of $place(\cdot)$. The subsets are searched using the capacity available. Other parameters could be changed in the Allocation Optimization Model for update runs. It can be seen that the yield of the dynamic case drops a bit. This is likely because global optimization in a static case becomes two iterative local optimizations that combine the outputs. On the other hand, 50 % implies that the convergence is faster, which can be investigated in more depth with feasibility and the definition of $\boldsymbol{Rf}$ in another study.

The minimum and maximum cases, 1x and 8x, cannot be improved in contrast to a uniform replication: 1x is the edge case where one and only one replica/copy exists per object (primary copy), and no additional capacity is left. In the introduced notation, the variation would be searched for in $\boldsymbol{Rf} = [1]$ and

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97557-8_27

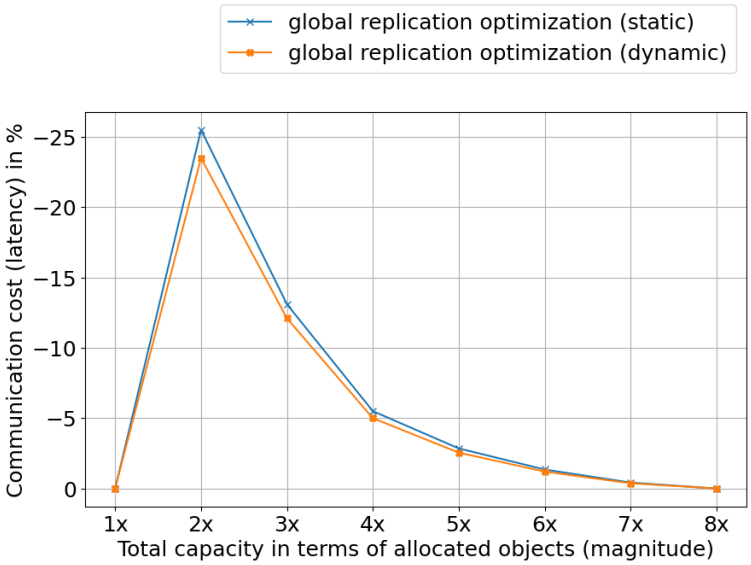

Fig. 7. Replication methods versus uniform, global replication.

nothing can be done. With 8x, all objects are maximally replicated; in this case, the search would occur in $\boldsymbol{Rf} = [1, 2, 3, 4, 5, 6, 7, 8]$ but no further improvement can be achieved. The global optimization method outperforms uniform replication, especially with replication factors RF=2,3. The relative improvement at low replication factors is higher. At high replication factors (with higher total capacity), the effect of an optimization is weakened asymptotically toward 0%.

7 Conclusion and Future Work

The findings imply that uniform global replication, for example, with replication factor 3, can be improved without including placement in optimization. This is valuable when placement dependency does not play a role or cannot be predicted. Spatial load may be balanced or change continuously.

Not only can a minimal replication factor Rf_1 be defined with the intended range of replication factors, but a search for replication factors in relation to storage or transfer cost can also be analyzed (not covered in this paper). The minimal replication factor of objects in the system should follow a policy according to a service-level agreement (SLA). The maximal replication factor can be varied to observe the gain over a cost function. This cost function can consider the storage cost / management cost, or may be balanced versus such cost (vector optimization).

The novel algorithm adapts to the provided capacity (multiple of the input size of the objects) and exhibits potential for broad application in distributed

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97557-8_27

systems using static (one-time) and dynamic replication (objects are added or changed). Clustering and evolution work in a broad field of data cases. The one shown is just an example. Non-continuous cost functions are feasible for evolutionary algorithms as well.

The demonstrated optimization model can be plugged into a replication system with with a fixed global replication, which lacks near-optimal replication to improve the system. The range of replication factors can be set. Prominent storage systems like Hadoop and Ceph have a default global replication factor (number of copies, e.g., 3), and some of them provide a configuration-based replication configuration per keyspace or bucket, or object-wise [15–19].

In the future, the heuristic may be investigated in more detail to introduce an improvement to guide the coordination of the partial replication sets. Furthermore, the problem has affine properties, and some parameters of the replication subsets may be approximated with an affine model (in a continuous space instead of a discrete one). From such a model, there may be insight for the overall optimization.

References

1. M. Baentsch, L. Baum, G. Molter, S. Rothkugel, and P. Sturm. Enhancing the Web's infrastructure: from caching to replication. *IEEE Internet Computing*, 1(2):18–27, April 1997.
2. Christopher Dabrowski. Reliability in grid computing systems. *Concurrency and Computation: Practice and Experience*, 21(8):927–959, June 2009.
3. Sem Borst, Varun Gupta, and Anwar Walid. Distributed Caching Algorithms for Content Distribution Networks. In *2010 Proceedings IEEE INFOCOM*, pages 1–9, San Diego, CA, USA, March 2010. IEEE.
4. Uroš Čibej, Boštjan Slivnik, and Borut Robič. The complexity of static data replication in data grids. *Parallel Computing*, 31(8-9):900–912, August 2005.
5. Jussi Kangasharju, James Roberts, and Keith W. Ross. Object replication strategies in content distribution networks. *Computer Communications*, 25(4):376–383, March 2002.
6. Yanhua Li, Haiyong Xie, Yonggang Wen, and Zhi-Li Zhang. Coordinating In-Network Caching in Content-Centric Networks: Model and Analysis. In *2013 IEEE 33rd International Conference on Distributed Computing Systems*, pages 62–72, Philadelphia, PA, USA, July 2013. IEEE.
7. Ralf Vamosi and Erich Schikuta. Dynamic Data Replication for Short Time-to-Completion in a Data Grid. In Jiří Mikyška, Clélia De Mulatier, Maciej Paszynski, Valeria V. Krzhizhanovskaya, Jack J. Dongarra, and Peter M.A. Sloot, editors, *Computational Science – ICCS 2023*, volume 10475, pages 668–675. Springer Nature Switzerland, Cham, 2023. Series Title: Lecture Notes in Computer Science.
8. Tarandeep Singh, Parvinder S. Sandhu, and Harbax Singh Bhatti. Replication of Data in Database Systems for Backup and Failover – An Overview. *International Journal of Computer and Communication Engineering*, pages 535–538, 2013.
9. B. Bhargava and J. Riedl. Implementation of RAID. In *Proceedings [1988] Seventh Symposium on Reliable Distributed Systems*, pages 157–166, Columbus, OH, USA, 1988. IEEE Comput. Soc. Press.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97557-8_27

10. Jun Li, Balazs Gerofi, Francois Trahay, Zhigang Cai, and Jianwei Liao. Rep-raid: An Integrated Approach to Optimizing Data Replication and Garbage Collection in RAID-Enabled SSDs. In *Proceedings of the 24th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems*, pages 99–110, Orlando FL USA, June 2023. ACM.
11. Matthew Leslie, Jim Davies, and Todd Huffman. A Comparison Of Replication Strategies for Reliable Decentralised Storage. *JOURNAL OF NETWORKS*, 1(6), 2006.
12. Dhruba Borthakur. The Hadoop Distributed File System: Architecture and Design. 2007.
13. Dejene Boru, Dzmitry Kliazovich, Fabrizio Granelli, Pascal Bouvry, and Albert Zomaya. Energy-Efficient Data Replication in Cloud Computing Datacenters. 2013.
14. Jinoh Kim and Doron Rotem. Using Replication for Energy Conservation in RAID Systems.
15. Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The hadoop distributed file system. In *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*, pages 1–10. Ieee, 2010.
16. Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 29–43, 2003.
17. Sage Weil, Scott A Brandt, Ethan L Miller, Darrell DE Long, and Carlos Maltzahn. Ceph: A scalable, high-performance distributed file system. In *Proceedings of the 7th Conference on Operating Systems Design and Implementation (OSDI'06)*, pages 307–320, 2006.
18. Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: Amazon's highly available key-value store. *ACM SIGOPS operating systems review*, 41(6):205–220, 2007.
19. Avinash Lakshman and Prashant Malik. Cassandra: a decentralized structured storage system. *ACM SIGOPS operating systems review*, 44(2):35–40, 2010.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97557-8_27