2021 IEEE International Symposium on Information Theory (ISIT) | 978-1-5386-8209-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISIT45174.2021.9518121

Bandwidth Cost of Code Conversions in Distributed Storage: Fundamental Limits and Optimal Constructions

Francisco Maturana and K. V. Rashmi
Carnegie Mellon University, Pittsburgh, PA, USA
Email: fmaturan@cs.cmu.edu, rvinayak@cs.cmu.edu

***Abstract*—In distributed storage systems, an $[n, k]$ code encodes k message symbols into n codeword symbols which are then stored on n nodes in the system. Recent work has shown that significant savings in storage space can be obtained by tuning n and k to variations in device failure rates. Such tuning necessitates *code conversion*: the process of converting data encoded under an $[n^I, k^I]$ code to its equivalent under an $[n^F, k^F]$ code. The default approach for code conversion places significant burden on system resources. *Convertible codes* are a recently proposed class of codes for enabling resource-efficient conversions. Existing work on convertible codes has focused on minimizing access cost, i.e., the number of nodes accessed during conversion. Bandwidth, which corresponds to the amount of data read and transferred, is another important resource to optimize during conversions.**

In this paper, we initiate the study on the fundamental limits on conversion bandwidth and present constructions for conversion-bandwidth optimal convertible codes. First, we model the code conversion problem using information flow graphs with variable capacity edges. Second, focusing on MDS codes and an important subclass of convertible codes, we derive a lower bound on conversion bandwidth. The derived bound shows that conversion bandwidth can be significantly reduced even in regimes where access cost of conversion cannot be reduced. Third, we present an explicit construction for MDS convertible codes which match this lower bound and are thus conversion-bandwidth optimal.

I. Introduction

Erasure codes are a central tool in distributed storage systems used to add redundancy to data in order to avoid data loss when failures occur [1]–[4]. In particular, maximum distance separable (MDS) codes are widely used in practice because they require the minimum amount of storage overhead for a given level of failure tolerance. In this setting, an $[n, k]$ MDS code over finite field $\mathbb{F}_q$ is used to encode a message $\mathbf{m} \in \mathbb{F}_q^k$ into a codeword $\mathbf{c} \in \mathbb{F}_q^n$. Each of the n codeword symbols is then stored in a distinct node of the distributed storage system. Large-scale distributed storage systems usually comprise hundreds to thousands of nodes, while n is much smaller in comparison, meaning that these systems store many such codewords distributed across different subsets of nodes. The MDS property ensures that any subset of k out of the n symbols in the codeword is enough to decode $\mathbf{m}$. This provides tolerance for up to $(n - k)$ node failures.

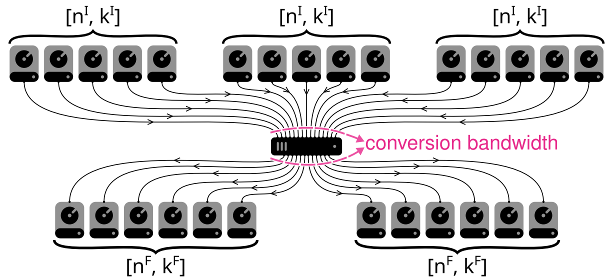

Fig. 1: Conversion process of codewords of an $[n^I, k^I]$ code into codewords of an $[n^F, k^F]$ code.

The parameters n and k are typically set based on the reliability of storage devices and additional requirements on system performance and storage overhead. Recent work by Kadekodi et al. [5] has shown that the failure rate of disks can vary drastically over time, and that significant savings in storage space (and hence operating costs) can be achieved by tuning the code rate to the observed failure rates. Such tuning typically needs to change both n and k of the code, due to other practical system constraints on these parameters [5]. Other reasons for tuning parameters include changing k in response to changes in data popularity, and adapting the code rate to limit the total amount of storage space used. Such tuning of parameters requires converting the already encoded data from one set of parameters to the newly chosen set of parameters. The *default approach* to achieving this is to read the encoded data, decode if necessary, re-encode it under the new code, and write it back. However, such an approach necessitates significantly high overhead in terms of network bandwidth, I/O, and CPU resources in the cluster.

This has led to the study of the *code conversion* problem [6]. Code conversion [6] (see Fig. 1) is the process of transforming a collection of codewords encoding data under an initial $[n^I, k^I]$ code $\mathcal{C}^I$ into a collection of codewords encoding the same data under a final $[n^F, k^F]$ code $\mathcal{C}^F$ (I and F stand for initial and final, respectively). Given certain parameters and decodability constraints (such as the MDS property) for $\mathcal{C}^I$ and $\mathcal{C}^F$, the goal is to design the codes $\mathcal{C}^I$ and $\mathcal{C}^F$ along with a conversion procedure from $\mathcal{C}^I$ to $\mathcal{C}^F$ that is efficient (according to some notion of conversion cost as will be discussed subsequently). Existing works [6], [7] on convertible codes have studied efficiency in terms of the *access cost* of conversion, which corresponds to the number of codeword

This work was funded in part by an NSF CAREER award (CAREER-1943409), an NSF CNS award (CNS-1956271), a Google faculty research award, and a Facebook distributed systems research award.

symbols accessed during conversion.

In this paper, we initiate the study of *conversion bandwidth*, which is the total amount of data transferred between nodes during conversion. We focus on MDS convertible codes that incur minimum conversion bandwidth (i.e. bandwidth-optimal convertible codes). To do this, we model code conversion using information flow graphs with variable capacity edges. We specifically focus on a parameter regime known as the *merge regime*, which has been shown to play the most critical role in the analysis and construction of convertible codes [6]. The merge regime corresponds to conversions where multiple initial codewords are merged into one final codeword (i.e. $k^F = \varsigma k^I$ for integer $\varsigma \geq 2$) and arbitrary values for n^I and n^F.

We show that significant savings in conversion bandwidth compared to the default approach are possible when $k^I > (n^F - k^F)$. This is perhaps surprising, since this *includes large parameter regimes where it is known that access cost cannot be reduced* [7]. In order to achieve this, we utilize *vector codes* (also called *array codes*), which treat each code symbol as a vector, in contrast to *scalar codes* of previous works [6], [7]. This allows for the download of only a fraction of each symbol, which is essential for minimizing conversion bandwidth.

First, we model code conversion as an information flow graph (§III). Then, by using this model, we derive a lower bound on the conversion bandwidth of MDS convertible codes in the merge regime (§IV). Next, we propose an explicit construction for bandwidth-optimal MDS convertible codes in the merge regime (§V). This construction considers fixed values for the initial and final parameters. However, in practice the exact value of the final parameters n^F and k^F might not be known at the time of code construction, as it might depend on future failure rates. Finally, we present a transformation to ensure bandwidth-optimal conversion even in the case where the final parameters are unknown ahead of time, but are drawn from a finite set of possibilities (§V-B).

Proofs are omitted due to lack of space, and are available in the extended version of this paper [8].

II. Background and related work

We start by reviewing concepts from the literature and doing an overview of related work.

A. Vector codes and puncturing

An $[n, k, \alpha]$ vector code $\mathcal{C}$ over a finite field $\mathbb{F}_q$ is an $\mathbb{F}_q$-linear subspace $\mathcal{C} \subseteq \mathbb{F}_q^{\alpha n}$ of dimension αk. For a given codeword $\mathbf{c} \in \mathcal{C}$ and $i \in [n]$, define $\mathbf{c}_i = (c_{\alpha(i-1)+1}, \ldots, c_{\alpha i}) \in \mathbb{F}_q^{\alpha}$ as the i-th *symbol* of $\mathbf{c}$. We refer to elements from the base field $\mathbb{F}_q$ as *subsymbols*. As an abuse of notation, we also use $\mathcal{C}$ as an *encoding function* $\mathcal{C}(\mathbf{m})$ mapping messages to codewords. A code $\mathcal{C}$ is said to be *systematic* if $\mathbf{m}$ is a prefix of $\mathcal{C}(\mathbf{m})$ for all $\mathbf{m} \in \mathbb{F}_q^{k\alpha}$. For linear codes, $\mathcal{C}(\mathbf{m}) = \mathbf{mG}$ where $\mathbf{G} \in \mathbb{F}_q^{k\alpha \times n\alpha}$ is called the *generator matrix* of $\mathcal{C}$, and the columns of $\mathbf{G}$ are called *encoding vectors*. Vector code $\mathcal{C}$ is maximum distance separable (MDS) iff any k out of the n symbols suffice to recover $\mathbf{m}$. A *scalar code* is a vector code with $\alpha = 1$ (we omit α in this case). A *puncturing* of a vector code $\mathcal{C}$ is the vector code that results from removing a fixed subset of symbols from every codeword $\mathbf{c} \in \mathcal{C}$.

B. Convertible codes [6], [7]

Let $\mathcal{C}^I$ be an $[n^I, k^I]$ code over $\mathbb{F}_q$, $\mathcal{C}^F$ an $[n^F, k^F]$ code over $\mathbb{F}_q$, $r^I = (n^I - k^I)$, and $r^F = (n^F - k^F)$. In order to allow for a change in code dimension from k^I to k^F, a message $\mathbf{m}$ of length $M = \operatorname{lcm}(k^I, k^F)$ is considered. In the initial configuration $\mathbf{m}$ is encoded as $\lambda^I = (M/k^I)$ codewords of $\mathcal{C}^I$ and in the final configuration it is encoded as $\lambda^F = (M/k^F)$ codewords of $\mathcal{C}^F$. Symbols of $\mathbf{m}$ are mapped to different codewords according to two partitions of $[M]$: $\mathcal{P}_I$ and $\mathcal{P}_F$. For a subset $\mathcal{I} \subseteq [M]$, we denote the restriction of $\mathbf{m}$ to the coordinates in $\mathcal{I}$ as $\mathbf{m}|_{\mathcal{I}}$. Convertible codes are defined as:

Definition 1 (Convertible code [6]): An $(n^I, k^I; n^F, k^F)$ convertible code over $\mathbb{F}_q$ is defined by: (1) initial and final codes $(\mathcal{C}^I, \mathcal{C}^F)$ over $\mathbb{F}_q$, where $\mathcal{C}^I$ is an $[n^I, k^I]$ code and $\mathcal{C}^F$ is an $[n^F, k^F]$ code, (2) initial and final partitions $(\mathcal{P}_I, \mathcal{P}_F)$ of $[M]$ such that $|P_i^I| = k^I$ for $P_i^I \in \mathcal{P}_I$ and $|P_j^F| = k^F$ for $P_j^F \in \mathcal{P}_F$, (3) a conversion procedure that takes initial codewords $\{\mathcal{C}^I(\mathbf{m}|_{P_i^I}) : i \in [\lambda^I]\}$ as input and outputs final codewords $\{\mathcal{C}^F(\mathbf{m}|_{P_i^F}) : i \in [\lambda^F]\}$ for all $\mathbf{m} \in \mathbb{F}_q^M$. ▸

1) Access cost: Access cost is the sum of the number of symbols read and number of symbols written during conversion. An *access-optimal convertible code* has the minimum access cost over all convertible codes with parameters $(n^I, k^I; n^F, k^F)$. Similarly, an $[n^I, k^I]$ code is (n^F, k^F)-access-optimally convertible if it is the initial code of an access-optimal $(n^I, k^I; n^F, k^F)$ convertible code.

2) Conversion procedure: During conversion, each codeword symbol is either an: (1) *unchanged symbol*, which is present both in the initial and final codewords without modifications; (2) *retired symbol*, which is only present in the initial codewords; or (3) *new symbol*, which is only present in the final codewords. Both unchanged and retired symbols can be read during conversion and used to write the new symbols. Convertible codes which have the maximum number of unchanged symbols (M when $k^I \neq k^F$) are called *stable*.

3) Access-optimal code for merge regime: When $r^I \geq r^F$ and $r^F < k^I$, access-optimal codes in the merge regime read r^F code symbols from each initial codeword. These symbols are then used to compute r^F new code symbols. In [6], a construction of access-optimal convertible codes for all parameters in the merge regime is proposed. Codes built using this construction are (1) systematic, (2) linear, (3) during conversion only access the first r^F parities from each initial stripe (assuming $r^F \leq r^I$), and (4) when constructed with a given value of $\lambda^I = \varsigma$ and $r^F = r$, the initial $[n^I, k^I]$ code is (n^F, k^F)-access-optimally convertible for all $k^F = \varsigma' k^I$ and $n^F = k^F + r'$ such that $1 \leq \varsigma' \leq \varsigma$ and $1 \leq r' \leq r$. In §V we use this code as part of our construction of conversion-bandwidth optimal codes for the merge regime.

C. Network information flow [9]

For the purposes of this paper, an *information flow graph* is a dag $G = (V, E)$, where $E \subseteq V \times V \times \mathbb{R}_{\geq 0}$ is a set of edges with non-negative capacities, and $(i, j, c) \in E$ represents that information can be sent noiselessly from node i to node j at rate c. Let $\{X_1, X_2, \ldots, X_m\}$ be mutually independent information sources with rates $\{x_1, x_2, \ldots, x_m\}$ respectively. Each information source X_i is associated with a source $s_i \in V$, where it is generated, and a sink $t_i \in V$, where it is required. In this paper we mainly make use of the *information max-flow bound* [10] which indicates that $x_i \leq \text{max-flow}(s_i, t_i)$ for all $i \in [m]$ is necessary. In our analysis, we consider s_i-t_i-cuts of the information flow graph, which give an upper bound on $\text{max-flow}(s_i, t_i)$ and thus an upper bound on x_i as well.

D. Piggybacking framework for constructing vector codes

The *Piggybacking framework* [11], [12] constructs $[n, k, \alpha]$ vector codes by using α instances of an existing $[n, k]$ code as a *base code*, and then adding carefully designed functions of the data (called *piggybacks*) from one instance to the others. Piggybacks in instance $2 \leq i \leq \alpha$ are a function of the data in instances $\{1, \ldots, i-1\}$. The piggyback functions are chosen to confer additional properties to the resulting code. To decode a piggybacked code, instance 1 is decoded first using the base code decoding procedure. The data of instance 1 is then used to subtract the piggybacks from instance 2 and the base code decoding procedure is then used to decode instance 2. Decoding proceeds in this fashion until all instances have been decoded. If the $[n, k]$ base code is MDS, then the resulting $[n, k, \alpha]$ vector code is MDS. In this paper, we use the Piggybacking framework to design a code where piggybacks store data which helps in making the conversion process efficient.

E. Other related work

Some special cases of code conversion have been studied in the literature: [13], [14] study the problem of minimizing bandwidth usage when adding extra parities (i.e. $k^I = k^F$ and $n^I < n^F$); [15] proposes two pairs of non-MDS codes that support conversion; and [16] studies a type of conversion in the context of distributed matrix multiplication.

Several works study the *scaling problem* [17]–[28]. This problem considers upgrading an erasure-coded storage system with s new empty data nodes. The general goal is to efficiently and evenly redistribute each codeword's data across all nodes, while updating parities to reflect the new placement of the data. This is a different problem from the code conversion problem we study in this paper, due to the scaling problem's need to *redistribute data from each codeword across nodes* which makes it an inefficient approach to achieve conversion.

III. Modeling conversion for optimizing bandwidth

In this section, we model conversion as an information flow problem. We utilize this model to derive a lower bound on conversion bandwidth. While we consider a single value for the final parameters n^F and k^F, the resulting bound still applies when multiple values are considered.

Previous work (§II-B) considered only scalar codes, which is sufficient when optimizing for access cost. However, when optimizing for network bandwidth, vector codes can perform better because they allow partial download from nodes. For this reason, we consider the initial code $\mathcal{C}^I$ as an $[n^I, k^I, \alpha]$ MDS code and the final code $\mathcal{C}^F$ as an $[n^F, k^F, \alpha]$ MDS code, where $\alpha \geq 1$ is considered as a free parameter chosen to minimize network bandwidth cost. For MDS convertible codes, message size is $B = M\alpha = \text{lcm}(k^I, k^F)\alpha$, interpreted as M symbols composed of α subsymbols. The number of subsymbols downloaded from node s during conversion is $\beta(s)$ and we define $\beta(\mathcal{S}) := \sum_{s \in \mathcal{S}} \beta(s)$.

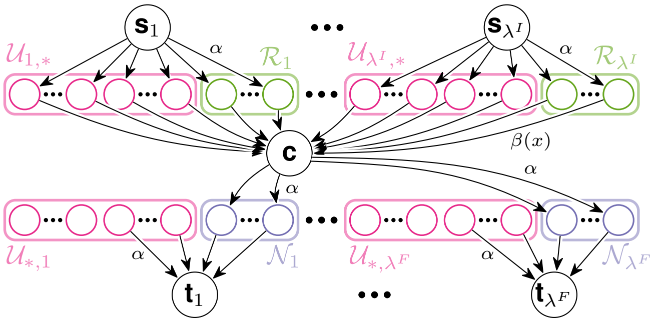

Fig. 2: Information flow graph of conversion in the general case. Unchanged, retired, and new nodes are shown in different colors. Each unchanged node is drawn twice (in the initial and final codewords) for clarity. Some representative edges are labeled with their capacities.

The information flow graph for a convertible code comprises the following nodes (see Fig. 2):

- unchanged nodes $\mathcal{U}_{i,j} = \{u_{i,j,1}, \ldots, u_{i,j,|\mathcal{U}_{i,j}|}\}$ for all $i \in [\lambda^I]$, $j \in [\lambda^F]$, which are present both in the initial and final codewords (drawn twice in Fig. 2);
- retired nodes $\mathcal{R}_i = \{v_{i,1}, \ldots, v_{i,|\mathcal{R}_i|}\}$ for $i \in [\lambda^I]$, which are only present in the initial codewords;
- new nodes $\mathcal{N}_j = \{w_{j,1}, \ldots, w_{j,|\mathcal{N}_j|}\}$ for $j \in [\lambda^F]$, which are only present in the final codewords;
- source nodes s_i for $i \in [\lambda^I]$, representing the message;
- sink nodes t_j for $j \in [\lambda^F]$, representing the data decoded;
- a coordinator node c, which models the central location where new symbols are computed.

Symbols of $\mathbf{m}$ are modeled as information sources $\{X_1, \ldots, X_M\}$ of rate α (over $\mathbb{F}_q$) each. Information source X_ℓ is generated at node s_i iff $\ell \in P_i^I$, and recovered at node t_j iff $\ell \in P_j^F$. When $*$ is used as an index, it denotes union of the indexed set over the range of that index, e.g. $\mathcal{U}_{*,j} = \bigcup_{i=1}^{\lambda^I} \mathcal{U}_{i,j}$. The graph has the following set of edges E:

- $\{(s_i, x, \alpha) : x \in \mathcal{U}_{i,*} \cup \mathcal{R}_i\} \subseteq E$ for each $i \in [\lambda^I]$, representing data stored on unchanged nodes;
- $\{(x, c, \beta(x)) : x \in \mathcal{U}_{i,*} \cup \mathcal{R}_i\} \subseteq E$ for each $i \in [\lambda^I]$, representing data downloaded from x;
- $\{(c, y, \alpha) : y \in \mathcal{N}_j\} \subseteq E$ for each $j \in [\lambda^F]$, representing data stored on new nodes; and
- $\{(y, t_j, \alpha) : y \in V_j\} \subseteq E$ for $V_j \subseteq \mathcal{U}_{*,j} \cup \mathcal{N}_j$ such that $|V_j| = k^F$, for all $j \in [\lambda^F]$, representing data downloaded to recover the message in a final codeword.

The set V_j represents a choice of k symbols for decoding a final codeword. A necessary condition for conversion is to satisfy all sinks t_j for all possible $V_1, \dots, V_{\lambda^F}$. The sets $\mathcal{U}_{i,j}, \mathcal{R}_i, \mathcal{N}_j$ and the capacities $\beta(x)$ are determined by the conversion procedure of the convertible code.

Definition 2 (Conversion bandwidth): The *conversion bandwidth* is the total network bandwidth used during conversion $\gamma := \beta(\mathcal{U}_{*,*} \cup \mathcal{R}_*) + |\mathcal{N}_*|\alpha$.

Information flow provides us with lower bounds on the capacities $\beta(x)$. Therefore, part of designing convertible codes is to set $\mathcal{U}_{i,j}, \mathcal{R}_i, \mathcal{N}_j$ so as to minimize the lower bound on γ.

IV. Optimizing network bandwidth of conversion in the merge regime

In this section, we use the information flow model presented in §III to derive a lower bound on conversion bandwidth for MDS codes in the merge regime. We focus on this regime because it is of practical importance and has been shown to play a central role in the analysis of convertible codes in general [7]. Recall from §II-B, that convertible codes in the merge regime are those where $k^F = \varsigma k^I$ for some integer $\varsigma \geq 2$, i.e., this regime corresponds to conversions were multiple initial codewords are merged into a single final codeword. We omit partitions from our analysis because in the merge regime all choices of partitions are equivalent up to relabeling [6].

First, we derive a general lower bound on conversion bandwidth in the merge regime by analyzing the information flow graph of conversion. Intuitively, this lower bound emerges from the fact that new symbols need to have a certain amount of information from each initial codeword in order to fulfill the MDS property of the final code.

Lemma 3: Consider an MDS $(n^I, k^I; n^F, \varsigma k^I)$ convertible code. Then $\gamma \geq (\varsigma\alpha \min\{r^F, k^I\} + r^F\alpha)$, where equality is only possible for stable codes. ■

Note that access-optimal codes meet this bound when $r^I \geq r^F$ [6], making them also bandwidth-optimal in this case.

When $r^I < r^F$, no savings in access cost are possible with respect to the default approach [6]. However, we will next derive a lower bound on conversion bandwidth which does leave room for savings. In §IV, we show that this bound is indeed tight, which implies that access-optimal codes are not bandwidth-optimal in general.

The following bound is also derived from the information flow graph. Intuitively, data downloaded from retired nodes is "more useful" than data downloaded from unchanged nodes, since unchanged nodes already form part of the final codeword. At the same time, it is better to have the maximum amount of unchanged nodes per initial codeword (k^I) because this minimizes the number of new nodes that need to be constructed. However, this leads to fewer retired nodes per initial codeword (r^I). If the number of retired nodes per initial codeword is less than the number of new nodes ($r^I < r^F$), then conversion procedures are forced to download data from unchanged nodes. This is because one needs to download at least $r^F\alpha$ from each initial codeword (by Lemma 3) Since data from unchanged nodes is "less useful", more data needs to be downloaded in order to construct the new nodes.

Lemma 4: Consider an MDS $(n^I, k^I; n^F, \varsigma k^I)$ convertible code, with $r^I < r^F \leq k^I$. Then:

$$\gamma \geq \varsigma\alpha\left(r^I + k^I\left(1 - \frac{r^I}{r^F}\right)\right) + r^F\alpha,$$

where equality is only possible for stable codes. ■

By combining Lemmas 3 and 4 we obtain the following general lower bound on conversion bandwidth of MDS convertible codes in the merge regime.

Theorem 5: For all MDS $(n^I, k^I; n^F, \varsigma k^I)$ convertible code:

$$\gamma \geq \begin{cases} \varsigma\alpha\min\{k^I, r^F\} + r^F\alpha, & \text{if } r^I \geq r^F \text{ or } k^I \leq r^F \\ \varsigma\alpha\left(r^I + k^I\left(1 - \frac{r^I}{r^F}\right)\right) + r^F\alpha, & \text{otherwise,} \end{cases}$$

where equality is only possible for stable convertible codes. ■

In §V, we show that this lower bound is achievable. We refer to codes that meet this bound as *conversion-bandwidth optimal* (abbreviated cBW-opt).

Definition 6 ((n^F, k^F)-cBW-opt): An $[n^I, k^I]$ code is said to be (n^F, k^F)-*cBW-opt* if it is the initial code of a cBW-opt $(n^I, k^I; n^F, k^F)$ convertible code.

The fraction of savings $\rho \in [0, 1]$ in bandwidth relative to the default approach obtained by cBW-opt codes can be computed as a function of its parameters. Note that the bandwidth used for writing is $r^F\alpha$ in both approaches, so we ignore it.

Corollary 7: Consider a cBW-opt MDS $(n^I, k^I; n^F, \varsigma k^I)$ convertible code, and let $\tilde{r}^I = (r^I/k^I)$, $\tilde{r}^F = (r^F/k^F)$. Then:

$$\rho = \begin{cases} 1 - \tilde{r}^F, & \text{if } \tilde{r}^F \leq \tilde{r}^I \text{ and } \tilde{r}^F < 1, \\ \tilde{r}^I(1/\tilde{r}^F - 1), & \text{if } \tilde{r}^I < \tilde{r}^F < 1, \\ 0, & \text{if } \tilde{r}^F \geq 1. \end{cases}$$

V. Explicit construction of cBW-opt MDS convertible codes in the merge regime

In this section, we present an explicit construction of cBW-opt codes in the merge regime. The key idea of our construction is to employ the Piggybacking framework (§II-D) with an access-optimal convertible code as base code. First, in §V-A, we describe our construction for fixed final parameters n^F and k^F along with an example. Then, in §V-B, we show that initial codes built with this construction are not only (n^F, k^F)-cBW-opt, but also cBW-opt for other pairs of (n^F, k^F). Finally, we present a construction which given any finite set of final parameters, constructs an $[n^I, k^I]$ code which is simultaneously (n^F, k^F)-cBW-opt for every (n^F, k^F) in that set.

A. Bandwidth-optimal MDS codes with fixed final parameters

When $r^F \geq k^I$, the default approach is bandwidth-optimal, and when $r^I \geq r^F$, access-optimal codes are also conversion-bandwidth optimal. Thus, we assume $r^F < k^I$ and $r^I < r^F$. The main challenge in this case is that r^I is too small for access-optimal codes to be efficient. To solve this, we use piggybacks to store information from an access-optimal code with larger r^I which can enable more efficient conversion. We start by describing the base code used in our construction, followed by the design of the piggybacks and the conversion procedure, and end with a detailed example of the construction.

initial codeword i $(\mathcal{C}^I)$

a_{4i-3}	b_{4i-3}
$\vdots$	$\vdots$
a_{4i}	b_{4i}
$\mathbf{a}^{(i)}\mathbf{p}_1^I$	$\mathbf{b}^{(i)}\mathbf{p}_1^I + \mathbf{a}^{(i)}\mathbf{p}_2^I$

final codeword $(\mathcal{C}^F)$

a_1	b_1
$\vdots$	$\vdots$
a_8	b_8
$\mathbf{ap}_1^F$	$\mathbf{bp}_1^F$
$\mathbf{ap}_2^F$	$\mathbf{bp}_2^F$

Fig. 3: Example of a bandwidth-optimal $(5, 4; 10, 8)$ convertible code. Each block represents a codeword, and each row is a symbol made up of $\alpha = 2$ subsymbols. Shaded rows correspond to retired nodes in the initial codewords, and new nodes in the final codeword. Text color is used to emphasize piggybacks and the subsymbols they produce.

a) Base code for piggybacking: As the base code, we use a punctured initial code of an access-optimal $(k^I + r^F, k^I; n^F, k^F)$ convertible code $(\mathcal{C}^{I'}, \mathcal{C}^{F'})$ (as described in §II-B3). Let $\mathcal{C}^{I''}$ be the puncturing of $\mathcal{C}^{I'}$ where the last $(r^F - r^I)$ parity symbols are removed.

b) Piggyback design: Now, we describe how to construct the $[n^I, k^I, \alpha]$ initial vector code $\mathcal{C}^I$ and the $[n^F, k^F, \alpha]$ final vector code $\mathcal{C}^F$ that make up the conversion-bandwidth optimal $(n^I, k^I; n^F, \varsigma k^I)$ convertible code. This piggybacking design is inspired by one of the designs proposed in [12].

The first step is to choose the value of α. By examining Theorem 5, we can see that the bound is met when $\alpha = r^F$ and $\beta_1 = r^F$ subsymbols are downloaded from each retired node and $\beta_2 = (r^F - r^I)$ subsymbols are downloaded from each unchanged node.

The next step is to design the piggybacks. Intuitively, we can use the β_2 subsymbols from unchanged nodes to store β_2 piggyback on each retired node. Since β_2 is also the number of new nodes, we can recover r^I subsymbols for each new node from the piggybacks.

Let $\mathbf{m}_j^{(s)} \in \mathbb{F}_q^{k^I}$ be the message of instance $j \in [\alpha]$ of the base code in initial codeword $s \in [\lambda^I]$. Since $\mathcal{C}^{I''}$ is systematic, $\mathbf{m}_j^{(s)}$ corresponds to the j-th coordinate of the k^I systematic symbols in initial codeword s. Let $c_{i,j}^I(s)$ denote the j-th coordinate of parity symbol i in initial codeword s of $\mathcal{C}^I$, and $c_{i,j}^F$ let denote the same for the final codeword of $\mathcal{C}^F$. Then, for $s \in [\lambda^I]$:

$$c_{i,j}^I(s) = \begin{cases} \mathbf{m}_j^{(s)}\mathbf{p}_i^I, & \text{for } i \in [r^I],\ 1 \leq j \leq r^I, \\ \mathbf{m}_j^{(s)}\mathbf{p}_i^I + \mathbf{m}_i^{(s)}\mathbf{p}_j^I, & \text{for } i \in [r^I],\ r^I < j \leq r^F, \end{cases}$$

$$c_{i,j}^F = [\mathbf{m}_j^{(1)} \cdots \mathbf{m}_j^{(\lambda^I)}]\mathbf{p}_i^F, \qquad \text{for } i \in [r^F],\ j \in [r^F],$$

where $\mathbf{p}_i^I$ (resp. $\mathbf{p}_i^F$) is the encoding vector of the i-th parity of $\mathcal{C}^{I'}$ (resp. $\mathcal{C}^{F'}$). By using the access-optimal conversion of the base code, we can compute $c_{i,j}^F = [\mathbf{m}_j^{(1)} \cdots \mathbf{m}_j^{(\lambda^I)}]\mathbf{p}_i^F$ from $\{\mathbf{m}_j^{(s)}\mathbf{p}_i^I : s \in [\lambda^I]\}$ for all $i \in [r^F]$ and $j \in [r^F]$.

c) Conversion procedure: Execute the following steps.

1) Download $D = \{\mathbf{m}_j^{(s)} : s \in [\lambda^I],\ r^I < j \leq r^F\}$, $C_1 = \{c_{i,j}^I(s) : s \in [\lambda^I],\ i \in [r^I],\ 1 \leq j \leq r^I\}$, and $C_2 = \{c_{i,j}^I(s) : s \in [\lambda^I],\ i \in [r^I],\ r^I < j \leq r^F\}$.
2) Recover the piggybacks $C_3 = \{\mathbf{m}_j^{(s)}\mathbf{p}_i^I : s \in [\lambda^I],\ r^I < i \leq r^F,\ 1 \leq j \leq r^I\}$ by computing $\mathbf{m}_i^{(s)}\mathbf{p}_j^I$ from D and obtaining $\mathbf{m}_j^{(s)}\mathbf{p}_i^I = c_{j,i}^I(s) - \mathbf{m}_i^{(s)}\mathbf{p}_j^I$ using C_2.
3) Compute the remaining base code symbols from the punctured symbols $C_4 = \{\mathbf{m}_i^{(s)}\mathbf{p}_j^I : s \in [\lambda^I],\ r^I < i \leq r^F,\ r^I < j \leq r^F\}$ using D.
4) Compute the parity symbols of the final codeword $C_5 = \{c_{i,j}^F : i \in [r^F],\ j \in [r^F]\}$ using the conversion from the base code on C_1, C_2, C_3, C_4.

Now we show a concrete example of our construction.

Example 8 (Bandwidth-optimal conversion in the merge regime): We construct a conversion-bandwidth optimal $(5, 4; 10, 8)$ convertible code over $\mathbb{F}_q$ (assume q is sufficiently large). As a base code, we use a punctured access-optimal $(6, 4; 10, 8)$ convertible code. Thus, $\mathcal{C}^{I'}$ is a $[6, 4]$ code, $\mathcal{C}^{F'}$ is a $[10, 8]$ code, and $\mathcal{C}^{I''}$ is a $[5, 4]$ code. Let $\mathbf{p}_1^I, \mathbf{p}_2^I \in \mathbb{F}_q^{4\times 1}$ be the encoding vectors for the parities of $\mathcal{C}^{I'}$, and $\mathbf{p}_1^F, \mathbf{p}_2^F \in \mathbb{F}_q^{8\times 1}$ be the encoding vector for the parities of $\mathcal{C}^{F'}$. Since $\alpha = 2$, we construct a $[5, 4, 2]$ initial vector code $\mathcal{C}^I$ and a $[10, 8, 2]$ final vector code $\mathcal{C}^F$. Let $\mathbf{a}^{(i)} = (a_{4i-3}, \ldots, a_{4i})$, $\mathbf{a} = (\mathbf{a}^{(1)}, \mathbf{a}^{(2)})$ and likewise for $\mathbf{b}$. Figure 3 shows the resulting code.

During conversion, only 12 subsymbols need to be downloaded: $\mathbf{b}$ and the parity symbols from both codewords. From these subsymbols, we recover the piggybacks $\mathbf{a}^{(1)}\mathbf{p}_2^I$ and $\mathbf{a}^{(2)}\mathbf{p}_2^I$, and compute $\mathbf{b}^{(1)}\mathbf{p}_2^I$ and $\mathbf{b}^{(2)}\mathbf{p}_2^I$ from $\mathbf{b}$. Finally, we use all these subsymbols with the access-optimal conversion to compute $\mathbf{ap}_1^F, \mathbf{ap}_2^F, \mathbf{bp}_1^F$ and $\mathbf{bp}_2^F$. The default approach would require one to download 16 subsymbols from the initial nodes. Both approaches require downloading 4 subsymbols from the coordinator node. Thus, the proposed construction leads to 20% reduction in conversion bandwidth. ▸

B. Convertible codes with bandwidth-optimal conversion for multiple final parameters

In practice, (n^F, k^F), and thus (λ^I, r^F), may be unknown at code construction time. To solve this problem, we present convertible codes which support conversion-bandwidth optimal conversion *simultaneously* for multiple final parameters.

1) Supporting multiple values of λ^I: Due to property (4) (see §II-B), the construction from §V given $\lambda^I = \varsigma$ inherently supports bandwidth-optimal conversion for any $\lambda^I = \varsigma' < \varsigma$.

2) Supporting multiple values of r^F: Consider two cases:

Case 1 $(r^F \leq r^I)$: due to property (4) (§II-B), the base code supports access-optimal conversion for any $r^F = r$ such that $r \leq r^I$. Thus, one can achieve bandwidth-optimal conversion for any $r \leq r^I$ by using the access-optimal conversion on the α instances of the base code, ignoring the piggybacks.

Case 2 $(r^F > r^I)$: to support $r^F \in \{r_1, \ldots, r_s\}$ such that $r_i > r^I$ $(i \in [s])$, we start with an access-optimal code having $r^F = \max_i r_i$. Then repeat the piggybacking step (§V-A) for each r_i, using the output from step i (with the punctured symbols from $\mathcal{C}^{I'}$ added back) as a base code for step $(i{+}1)$. Thus, the code has $\alpha = \prod_{i=1}^s r_i$. Since piggybacking preserves the MDS property of a base code, the resulting code will be MDS. Conversion for one of the supported $r^F = r_i$ is performed as described in §V-A on each of the additional instances created by steps $(i+1), \ldots, s$ (i.e. $\prod_{i'=(i+1)}^s r_{i'}$ in total).

References

[1] S. Ghemawat, H. Gobioff, and S. Leung, "The Google file system," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles 2003, SOSP 2003, Bolton Landing, NY, USA, October 19-22, 2003*, M. L. Scott and L. L. Peterson, Eds. ACM, 2003, pp. 29–43.

[2] D. Borthakur, R. Schmidt, R. Vadali, S. Chen, and P. Kling, "HDFS RAID - Facebook," available on: http://www.slideshare.net/ydn/hdfs-raid-facebook. Accessed: 2019-07-23.

[3] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin, "Erasure coding in Windows Azure storage," in *2012 USENIX Annual Technical Conference, Boston, MA, USA, June 13-15, 2012*, G. Heiser and W. C. Hsieh, Eds. USENIX Association, 2012, pp. 15–26.

[4] Apache Software Foundation, "Apache hadoop: HDFS erasure coding," available on: https://hadoop.apache.org/docs/r3.0.0/hadoop-project-dist/hadoop-hdfs/HDFSErasureCoding.html. Accessed: 2019-07-23.

[5] S. Kadekodi, K. V. Rashmi, and G. R. Ganger, "Cluster storage systems gotta have HeART: improving storage efficiency by exploiting disk-reliability heterogeneity," in *17th USENIX Conference on File and Storage Technologies, FAST 2019, Boston, MA, February 25-28, 2019*, A. Merchant and H. Weatherspoon, Eds. USENIX Association, 2019, pp. 345–358.

[6] F. Maturana and K. V. Rashmi, "Convertible codes: new class of codes for efficient conversion of coded data in distributed storage," in *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, ser. LIPIcs, T. Vidick, Ed., vol. 151. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, pp. 66:1–66:26.

[7] F. Maturana, V. S. C. Mukka, and K. V. Rashmi, "Access-optimal linear MDS convertible codes for all parameters," in *IEEE International Symposium on Information Theory, ISIT 2020, Los Angeles, California, USA, June 21-26, 2020*, 2020.

[8] F. Maturana and K. V. Rashmi, "Bandwidth cost of code conversions in distributed storage: Fundamental limits and optimal constructions," 2020, arXiv:2008.12707.

[9] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[10] R. W. Yeung, *A First Course in Information Theory*. Boston, MA: Springer US, 2002.

[11] K. V. Rashmi, N. B. Shah, and K. Ramchandran, "A piggybacking design framework for read-and download-efficient distributed storage codes," in *2013 IEEE International Symposium on Information Theory, ISIT 2013, Istanbul, Turkey, July 7-12, 2013*. IEEE, 2013, pp. 331–335.

[12] ——, "A piggybacking design framework for read-and download-efficient distributed storage codes," *IEEE Transactions on Information Theory*, vol. 63, no. 9, pp. 5802–5820, 2017.

[13] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Enabling node repair in any erasure code for distributed storage," in *2011 IEEE International Symposium on Information Theory Proceedings, ISIT 2011, St. Petersburg, Russia, July 31 - August 5, 2011*, A. Kuleshov, V. M. Blinovsky, and A. Ephremides, Eds. IEEE, 2011, pp. 1235–1239.

[14] S. Mousavi, T. Zhou, and C. Tian, "Delayed parity generation in MDS storage codes," in *2018 IEEE International Symposium on Information Theory, ISIT 2018, Vail, CO, USA, June 17-22, 2018*. IEEE, 2018, pp. 1889–1893.

[15] M. Xia, M. Saxena, M. Blaum, and D. Pease, "A tale of two erasure codes in HDFS," in *Proceedings of the 13th USENIX Conference on File and Storage Technologies, FAST 2015, Santa Clara, CA, USA, February 16-19, 2015*, J. Schindler and E. Zadok, Eds. USENIX Association, 2015, pp. 213–226.

[16] X. Su, X. Zhong, X. Fan, and J. Li, "Local re-encoding for coded matrix multiplication," in *IEEE International Symposium on Information Theory, ISIT 2020, Los Angeles, California, USA, June 21-26, 2020*, 2020.

[17] G. Zhang, W. Zheng, and J. Shu, "ALV: A new data redistribution approach to RAID-5 scaling," *IEEE Transactions on Computers*, vol. 59, no. 3, pp. 345–357, 2010.

[18] W. Zheng and G. Zhang, "Fastscale: accelerate RAID scaling by minimizing data migration," in *9th USENIX Conference on File and Storage Technologies, San Jose, CA, USA, February 15-17, 2011*, G. R. Ganger and J. Wilkes, Eds. USENIX, 2011, pp. 149–161.

[19] C. Wu and X. He, "GSR: A global stripe-based redistribution approach to accelerate RAID-5 scaling," in *41st International Conference on Parallel Processing, ICPP 2012, Pittsburgh, PA, USA, September 10-13, 2012*. IEEE Computer Society, 2012, pp. 460–469.

[20] G. Zhang, W. Zheng, and K. Li, "Rethinking RAID-5 data layout for better scalability," *IEEE Transactions on Computers*, vol. 63, no. 11, pp. 2816–2828, 2014.

[21] J. Huang, X. Liang, X. Qin, P. Xie, and C. Xie, "Scale-RS: an efficient scaling scheme for RS-coded storage clusters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 6, pp. 1704–1717, 2015.

[22] S. Wu, Y. Xu, Y. Li, and Z. Yang, "I/O-efficient scaling schemes for distributed storage systems with CRS codes," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 9, pp. 2639–2652, 2016.

[23] X. Zhang, Y. Hu, P. P. C. Lee, and P. Zhou, "Toward optimal storage scaling via network coding: from theory to practice," in *2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, April 16-19, 2018*. IEEE, 2018, pp. 1808–1816.

[24] Y. Hu, X. Zhang, P. P. C. Lee, and P. Zhou, "Generalized optimal storage scaling via network coding," in *2018 IEEE International Symposium on Information Theory, ISIT 2018, Vail, CO, USA, June 17-22, 2018*. IEEE, 2018, pp. 956–960.

[25] X. Zhang and Y. Hu, "Efficient storage scaling for MBR and MSR codes," *IEEE Access*, vol. 8, pp. 78 992–79 002, 2020.

[26] B. K. Rai, V. Dhoorjati, L. Saini, and A. K. Jha, "On adaptive distributed storage systems," in *IEEE International Symposium on Information Theory, ISIT 2015, Hong Kong, China, June 14-19, 2015*. IEEE, 2015, pp. 1482–1486.

[27] B. K. Rai, "On adaptive (functional MSR code based) distributed storage systems," in *2015 International Symposium on Network Coding, NetCod 2015, Sydney, Australia, June 22-24, 2015*. IEEE, 2015, pp. 46–50.

[28] S. Wu, Z. Shen, and P. P. C. Lee, "On the optimal repair-scaling trade-off in locally repairable codes," in *2020 IEEE Conference on Computer Communications, INFOCOM 2020, Virtual Conference, July 6-9, 2020*. IEEE, 2020.