Storage and Computation: A Tradeoff in Secure Distributed Computing

Jiajun Chen[1], Chi Wan Sung[1] and Terence H. Chan[2]
[1]Department of Electrical Engineering, City University of Hong Kong
[2]Institute for Telecommunications Research, University of South Australia
Email: jiajuchen8-c@my.cityu.edu.hk, albert.sung@cityu.edu.hk, terence.chan@unisa.edu.au

***Abstract*—Cloud computing provides a flexible and cost-effective solution to big data applications. Data privacy, however, is a main concern. To avoid information leakage to a cloud service provider, a user may store portions of encoded data in multiple clouds and perform computing tasks in a distributed way. In this work, nested MDS codes and the criterion of perfect secrecy are adopted. The allocation of encoded data to be stored in heterogeneous clouds with different computing capability is formulated as an optimization problem, subject to data reliability and security constraints. The problem is shown to be feasible if and only if the storage budget is above a certain level. The tradeoff between minimum computation time and storage budget is analytically characterized, and the former is proved to be a piecewise-linear decreasing function of the latter. When the storage budget increases beyond a certain threshold, the optimal computation time levels off. Closed-form expressions of minimum computation time and optimal storage allocation are obtained. Numerical results show that the optimized allocation outperforms equal allocation significantly if the computing rates of different clouds have large variation.**

***Index Terms*—Coded distributed computing, distributed storage, perfect secrecy, data allocation**

I. Introduction

Distributed computing has been widely regarded as a promising direction for speeding up massive machine learning and data analysis. Recently, research efforts focus on how to deal with stragglers, which have much larger delay than other workers. A computation task then needs to wait for the completion of those stragglers. To mitigate the effect of stragglers, coding techniques are often used in either homogeneous clusters [1]–[4] or heterogeneous clusters [5], [6]. Moreover, they have been extended to address security issues: polynomial codes and staircase codes have been applied in [7], [8], respectively, to ensure any worker cannot steal any information about the private data or the computation results in the information theoretic sense. The security level, however, is low, as it cannot avoid collusion between workers.

Previous works in secure distributed computing usually consider a single cloud with a master and massive distributed workers. To perform computing tasks, a user needs to send data to the cloud, which incurs substantial delay for massive computing [9]. Storing data in the cloud in advance may solve the problem, but it brings security concerns [10]. If one single cloud is used, storing encrypted data involves complicated key management problems. One possible solution is to store data using nested MDS codes over multiple clouds [11].

In this work, we consider joint distributed storage and computation over multiple clouds. Those clouds are heterogeneous with different computing rate, and some clouds may collude to steal users' data. Perfect secrecy is guaranteed provided that the number of colluding clouds does not exceed a certain predefined limit. To the best of our knowledge, this is the first work considering joint storage and computation with security constraints. Our main contributions are listed as follows:

- A joint distributed storage and computation system based on nested MDS codes is proposed, which is shown to satisfy certain reliability and security requirements.
- Given a storage budget, the optimal storage allocation in multiple clouds which minimizes the overall computation time is determined.
- The optimal tradeoff between storage budget and computation time is explicitly characterized.

Notation: Define $\mathcal{V} \triangleq \{1, 2, ..., V\}$ for $V \in \mathbb{N}$. Let $\mathcal{J} \subset_J \mathcal{V}$ represent that $\mathcal{J}$ is a subset of $\mathcal{V}$ with cardinality J, and $\mathbf{1}_n$ be the all-one vector of length n.

II. System Model and Problem Formulation

Consider a setting in which a user wants to store an $m \times s$ confidential matrix $\mathbf{A}$ over V multiple clouds and perform some distributed computing task securely. Each of its rows can be regarded as a data record while the width s can be regarded as the number of features associated with each data record. The entries of $\mathbf{A}$ are independent and identical random variables, each of which is drawn uniformly from the finite field of size q, denoted by $\mathbb{F}_q$. The entropy of $\mathbf{A}$, $H(\mathbf{A})$, is equal to $ms \log_2 q$ bits.

A. Storage and Computation Model

To ensure security and avoid collusion, the confidential matrix $\mathbf{A}$ is encoded as $\tilde{\mathbf{A}} \in F_q^{n \times s}$, which can be partitioned into V submatrices as follows:

$$\tilde{\mathbf{A}} \triangleq \left[\tilde{\mathbf{A}}_1^T \ \tilde{\mathbf{A}}_2^T \ \cdots \ \tilde{\mathbf{A}}_V^T\right]^T,$$

where $\tilde{\mathbf{A}}_i \in F_q^{l_i \times s}$ and $n = \sum_{i=1}^{V} l_i > m$. There is a master node, which distributes $\tilde{\mathbf{A}}_i$ to Cloud i, for $i = 1, 2, \ldots, V$. We

This work was supported in part by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China, under Project CityU 11205318.

978-1-7281-5089-5/20/$31.00 ©2020 IEEE

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on September 08,2020 at 08:12:01 UTC from IEEE Xplore. Restrictions apply.

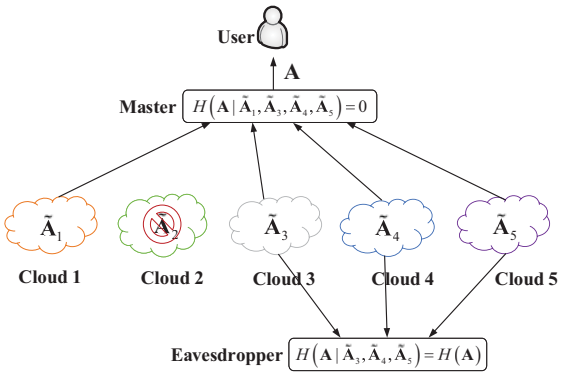


Fig. 1. In this $(5,4,3)$ system, Cloud 2 is unavailable, but the user can still retrieve the original data $\mathbf{A}$ from the other four clouds. Clouds 3, 4 and 5 collude to steal the confidential data, but no information is leaked to the eavesdropper.

call $\boldsymbol{l} \triangleq (l_1, l_2, ..., l_V)$ the storage allocation vector, where l_i represents the number of encoded rows allocated to Cloud i.

The criterion for security is secret sharing, which is commonly referred to *perfect secrecy*. No information of the secret can be revealed unless the number of gathered participants reaches a threshold J. In other words, the *perfect secrecy requirement* is that no information would be disclosed to any J colluding clouds $(J < V)$ in the information-theoretic sense. Define $\tilde{\mathbf{A}}(\mathcal{S})$ as the matrix obtained by collecting $\tilde{\mathbf{A}}_i$ for all $i \in \mathcal{S}$, then the perfect secrecy requirement can be written as

$$H(\mathbf{A}|\tilde{\mathbf{A}}(\mathcal{J})) = H(\mathbf{A}),\ \forall \mathcal{J} \subset_J \mathcal{V}. \tag{1}$$

To ensure data reliability, it is also required that the master can retrieve the original matrix $\mathbf{A}$ from any K available clouds $(J < K \leq V)$. This reconstruction requirement for distributed storage can be expressed as

$$H(\mathbf{A}|\tilde{\mathbf{A}}(\mathcal{K})) = 0,\ \forall \mathcal{K} \subset_K \mathcal{V}. \tag{2}$$

We refer it as a (V, K, J) system, where K and J represent the desired reliability and security levels, respectively. Fig. 1 shows an example of a $(5, 4, 3)$ system.

Now consider the common computation task of matrix-vector multiplication $\mathbf{y} = \mathbf{A}\mathbf{x}$, where the input vector $\mathbf{x} \in F_q^{s\times 1}$ has no security requirement. Different from distributed storage, distributed computing task does not have to be completed only in K clouds. Instead, we can utilize any available cloud of the remaining $V - K$ clouds to compute, speeding up the computation. In each cloud, there is a sub-master to manage a group of distributed workers for computing. While Cloud i stores $\tilde{\mathbf{A}}_i$ with l_i rows, in general, the sub-master may only use r_i rows $(0 \leq r_i \leq l_i)$ for computing. Denote the load vector $\boldsymbol{r} \triangleq (r_1, r_2, \ldots, r_V)$. Let $\tilde{\mathbf{B}}_i$ be a submatrix of $\tilde{\mathbf{A}}_i$ with r_i rows. After computing, the sub-master sends $\tilde{\mathbf{B}}_i\mathbf{x}$ to the master.

After collecting the computing results $\tilde{\mathbf{B}}_1\mathbf{x}$, $\tilde{\mathbf{B}}_2\mathbf{x}$, $\ldots$, $\tilde{\mathbf{B}}_V\mathbf{x}$, the master stacks them up to form $\tilde{\mathbf{y}}$, which is a subvector of $\tilde{\mathbf{A}}\mathbf{x}$. The length of $\tilde{\mathbf{y}}$ is $\sum_{i=1}^{V} r_i$. Then the reconstruction requirement for distributed computation can be expressed as

$$H(\mathbf{y}|\tilde{\mathbf{y}}) = 0. \tag{3}$$

B. Code Construction Achieving Perfect Secrecy

The proposed code construction is based on the *nested MDS code* to achieve perfect secrecy. Consider a wiretap channel: n symbols are transmitted. A user and an eavesdropper receive φ and θ symbols, respectively. It is proved in [12] and [13] that an (n, φ, θ) nested MDS code can achieve the secrecy capacity $\varphi - \theta$ while ensuring the perfect secrecy. First we give the definition of the nested MDS code proposed in [12]:

Definition 1 (Nested MDS Codes)**.** *Let $\mathbf{G}$, of size $\varphi \times n$, be a generator matrix of an (n, φ) MDS code. If it can be partitioned into the form*

$$\mathbf{G} \triangleq \begin{bmatrix} \mathbf{G}_1 \\ \mathbf{G}_2 \end{bmatrix},$$

where $\mathbf{G}_1$, of size $\theta \times n$ $(\theta < \varphi)$, is itself a generator matrix of an (n, θ) MDS code, then the code constructed by $\mathbf{G}$ is referred to an (n, φ, θ) nested MDS code.

In our (V, K, J) system, the user receives at least K submatrices from V clouds and the stealer receives at most J submatrices. Given storage allocation vector $\boldsymbol{l}$, we could use an (n, φ, θ) nested MDS code to achieve perfect secrecy (1), where

$$n = \sum_{i\in\mathcal{V}} l_i,\ \varphi = \min_{\mathcal{K}\subset_K\mathcal{V}} \sum_{i\in\mathcal{K}} l_i \text{ and } \theta = \max_{\mathcal{J}\subset_J\mathcal{V}} \sum_{i\in\mathcal{J}} l_i.$$

If the secrecy capacity is no less than m, i.e.,

$$\varphi - \theta \geq m, \tag{4}$$

the reconstruction requirement for distributed storage (2) are fulfilled.

The generator matrix of an (n, φ, θ) nested MDS code is $\mathbf{G} \in F_q^{\varphi\times n}$, which can be partitioned into V submatrices $\mathbf{G} = [\mathbf{G}_1\ \mathbf{G}_2\ \cdots\ \mathbf{G}_V]$, where $\mathbf{G}_i \in F_q^{\varphi\times l_i}$. Specifically, the proposed coded storage scheme is described as follows:

1) Generate a random matrix $\mathbf{R} \in F_q^{\theta\times s}$ uniformly;
2) Encode $\mathbf{A}$ into $\tilde{\mathbf{A}}$ by the (n, φ, θ) nested MDS code:

$$\tilde{\mathbf{A}} \triangleq \begin{bmatrix} \tilde{\mathbf{A}}_1 \\ \vdots \\ \tilde{\mathbf{A}}_V \end{bmatrix} = \mathbf{G}^T \begin{bmatrix} \mathbf{R} \\ \mathbf{A} \\ \mathbf{Z} \end{bmatrix}, \tag{5}$$

   where $\mathbf{Z} \in F_q^{(\varphi-\theta-m)\times s}$ is the zero matrix;
3) For $i \in \mathcal{V}$, coded submatrix $\tilde{\mathbf{A}}_i \in F_q^{l_i\times s}$ is sent to Cloud i for storage.

To retrieve the original data $\mathbf{A}$, the master should fetch encoded data $\tilde{\mathbf{A}}_i$ from K clouds $(i \in \mathcal{K}, \mathcal{K} \subset_K \mathcal{V})$, and stacks them up as a matrix $\tilde{\mathbf{A}}(\mathcal{K})$ with size $\sum_{i\in\mathcal{K}} l_i \times s$. Accordingly, $\mathbf{G}^T(\mathcal{K})$, of size $\sum_{i\in\mathcal{K}} l_i \times \varphi$, is the collection of $\mathbf{G}_i^T$ $(i \in \mathcal{K})$. Since $\sum_{i\in\mathcal{K}} l_i \geq \varphi$, some rows from $\mathbf{G}^T(\mathcal{K})$ can be removed to obtain a square matrix, which is invertible due to the property of nested MDS codes. $\tilde{\mathbf{A}}(\mathcal{K})$, with the corresponding rows removed, is then left multiplied by the inverse of the square matrix to undo the operation in (5), which decodes the value of $\mathbf{A}$.

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on September 08,2020 at 08:12:01 UTC from IEEE Xplore. Restrictions apply.

For distributed computing, the master sends input $\mathbf{x}$ to each cloud. The sub-master of Cloud i just needs to calculate the product of $\tilde{\mathbf{B}}_i$ (a submatrix of $\tilde{\mathbf{A}}_i$ with r_i rows) and $\mathbf{x}$, and transfers the outcome to the master. The master collects the computing results and stacks them up to form $\tilde{\mathbf{y}}$, which is a subvector of $\tilde{\mathbf{A}}\mathbf{x}$. From the property of nested MDS code, we have

$$H(\mathbf{A}|\tilde{\mathbf{A}}(\mathcal{I})) = 0,\ \forall \mathcal{I} \subset_{|\mathcal{I}|} \{1, 2, ..., n\}, \tag{6}$$

where $|\mathcal{I}| = \varphi$. Because (6) can be satisfied by the proposed coding scheme, we can meet the reconstruction requirement in (3) to restore $\mathbf{y} = \mathbf{A}\mathbf{x}$ by the following decoding procedure: Let the length of $\tilde{\mathbf{y}}$ satisfy

$$\sum_{i\in\mathcal{V}} r_i = \varphi. \tag{7}$$

The master then truncates $\mathbf{G}^{\mathrm{T}}$ by preserving only the corresponding rows to obtain the square submatrix $\mathbf{G}_0^{\mathrm{T}}$, which is invertible due to the property of nested MDS codes. The desired product, $\mathbf{y} = \mathbf{A}\mathbf{x}$, can be restored by the following decoding procedure:

$$\begin{bmatrix}\mathbf{R}\\ \mathbf{A}\\ \mathbf{Z}\end{bmatrix}\mathbf{x} = (\mathbf{G}_0^T)^{-1}\tilde{\mathbf{y}}.$$

C. Problem Formulation

Given a (V, K, J) system, we encode the confidential matrix $\mathbf{A} \in F_q^{m\times s}$ into $\tilde{\mathbf{A}}$, and then allocate it to multiple clouds with storage allocation vector $\boldsymbol{l}$ and load vector $\boldsymbol{r}$ for distributed storage and computing, respectively. In this paper, we focus on the scenario $J > V - K$, which corresponds to a high security level. In that scenario, as long as K clouds are secure and available, the user can retrieve the stored matrix and perform the computing task securely without leaking any information to a hacker who may have completely controlled the other $V - K$ clouds.

Let S be the storage budget on multiple clouds, such that

$$\sum_{i\in\mathcal{V}} l_i \le S. \tag{8}$$

For $i \in \mathcal{V}$, Cloud i provides computing service with rate μ_i, so its computation time is r_i/μ_i. Without loss of generality, we assume $\mu_1 \ge \mu_2 \ge \cdots \ge \mu_V$. The overall computation time on multiple clouds is denoted by T, which is bounded below by the computation time of each individual cloud.

Our objective is to determine $\boldsymbol{l}$ and $\boldsymbol{r}$ which minimize T under storage cost budget (8), and the reconstruction and security requirement (4), (7). Since $\boldsymbol{l}$, $\boldsymbol{r}$, and S naturally grow with the number of data records, m, we normalize all of them by m. With slight abuse of notation, we use the same symbols but interpret them as the storage amount, computing amount, and total storage budget, *per data record*.

We are particularly interested in the asymptotic case where m is large, which is relevant to big data applications. While those variables are rational numbers by nature, it is justifiable to make a relaxation and treat them as real numbers. Then we can formulate the following optimization problem over the real vector space as follows:

$$\min_{\boldsymbol{l},\boldsymbol{r}\in\mathbb{R}^V} T \tag{9}$$

subject to

$$\sum_{i\in\mathcal{V}} l_i \le S \tag{10}$$

$$\min_{\mathcal{K}\subset_K\mathcal{V}} \sum_{i\in\mathcal{K}} l_i - \max_{\mathcal{J}\subset_J\mathcal{V}} \sum_{i\in\mathcal{J}} l_i \ge 1 \tag{11}$$

$$0 \le r_i \le l_i,\ \forall i \in \mathcal{V} \tag{12}$$

$$T \ge \frac{r_i}{\mu_i},\ \forall i \in \mathcal{V} \tag{13}$$

$$\sum_{i\in\mathcal{V}} r_i = \min_{\mathcal{K}\subset_K\mathcal{V}} \sum_{i\in\mathcal{K}} l_i \tag{14}$$

III. Main Results

In this section, we solve the optimization problem stated in Section II and find the optimal storage allocation vector $\boldsymbol{l}^* \triangleq (l_1^*, \ldots, l_V^*)$ and optimal load vector $\boldsymbol{r}^* \triangleq (r_1^*, \ldots, r_V^*)$.

Lemma 1. *The optimization problem (9) is feasible if and only if $S \ge \frac{V}{K-J}$. When S reaches its lower bound, the storage allocation must be equal, i.e., $l_1 = \cdots = l_V = \frac{1}{K-J}$.*

Proof. If the problem is feasible, let the allocation vector $\boldsymbol{l} \triangleq (l_1, l_2, \ldots, l_V)$ and load vector $\boldsymbol{r} \triangleq (r_1, r_2, \ldots, r_V)$ be one of the feasible solutions. We sort $l_1, l_2, \ldots, l_V$, in decreasing order and denote them by $l_{(1)}, l_{(2)}, \ldots, l_{(V)}$ such that

$$l_{(1)} \ge l_{(2)} \ge \cdots \ge l_{(V)} \ge 0. \tag{15}$$

Rewrite (11) as

$$\begin{aligned} 1 &\le \sum_{i=V-K+1}^{V} l_{(i)} - \sum_{i=1}^{J} l_{(i)} \overset{(a)}{=} \sum_{i=J+1}^{V} l_{(i)} - \sum_{i=1}^{V-K} l_{(i)} \\ &= \sum_{i=V-K+J+1}^{V} l_{(i)} - \sum_{i=1}^{V-K} [l_{(i)} - l_{(J+i)}] \le \sum_{i=V-K+J+1}^{V} l_{(i)}, \end{aligned} \tag{16}$$

where (a) holds due to $V \ge K > J > V - K$. Since the sequence is in decreasing order, (16) implies

$$l_{(V-K+J+1)} \ge \frac{1}{K-J}, \tag{17}$$

and the equality holds if and only if $l_{(V-K+J+1)} = l_{(V-K+J+2)} = \cdots = l_{(V)} = \frac{1}{K-J}$. From (10),

$$\begin{aligned} S \ge \sum_{i\in\mathcal{V}} l_{(i)} &= \sum_{i=1}^{V-K+J} l_{(i)} + \sum_{i=V-K+J+1}^{V} l_{(i)} \\ &\overset{(b)}{\ge} \sum_{i=1}^{V-K+J} l_{(i)} + 1 \\ &\overset{(c)}{\ge} \frac{V-K+J}{K-J} + 1 = \frac{V}{K-J}, \end{aligned} \tag{18}$$

where (b) follows from (16), and (c) from (17).

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on September 08,2020 at 08:12:01 UTC from IEEE Xplore. Restrictions apply.

If $S \geq \frac{V}{K-J}$, it is easy to check that $\boldsymbol{l} = \frac{1}{K-J}\mathbf{1}_V$ and $\boldsymbol{r} = \frac{K}{V(K-J)}\mathbf{1}_V$ satisfy all the constraints and form a feasible solution. Therefore, it is a also a sufficient condition for the problem to be feasible.

Consider the special case where $S = \frac{V}{K-J}$. We have proved that the problem is feasible, so both the lower and upper bounds in (18) hold. Since the two bounds coincide, equality holds in both (b) and (c). The former occurs if and only if $l_{(V-K+J+1)} + \ldots + l_{(V)} = 1$, and the latter occurs if and only if $l_{(1)} = \cdots = l_{(V-K+J)} = \frac{1}{K-J}$. Combining the two conditions with (15), we obtain that $l_{(i)} = \frac{1}{K-J}$ for all $i \in \mathcal{V}$, which completes the proof. □

Lemma 2. *There exists an optimal solution at which inequality* (11) *holds with equality.*

Proof. Let $\boldsymbol{r}^*$ and $\boldsymbol{l}^*$ be optimal allocations, which achieve the minimum completion time, T^*. We sort the sequence of $l_1^*, l_2^*, \ldots, l_V^*$ in decreasing order and denote them by $l_{(1)}^*, l_{(2)}^*, \ldots, l_{(V)}^*$, such that

$$l_{(1)}^* \geq l_{(2)}^* \geq ... \geq l_{(V)}^* \geq 0. \tag{19}$$

For all $i \in \mathcal{V}$ such that $l_{(i)}^* \triangleq l_j^*$, define $r_{(i)}^* \triangleq r_j^*$ accordingly.

Rewrite (11) and (14), respectively, as

$$\sum_{i=V-K+1}^{V} l_{(i)}^* - \sum_{i=1}^{J} l_{(i)}^* = \sum_{i=J+1}^{V} l_{(i)}^* - \sum_{i=1}^{V-K} l_{(i)}^* \geq 1, \tag{20}$$

$$\sum_{i\in\mathcal{V}} r_{(i)}^* = \sum_{i=V-K+1}^{V} l_{(i)}^*. \tag{21}$$

Suppose equality in (20) does not hold. Now we show that we can adjust $\boldsymbol{l}^*$ and $\boldsymbol{r}^*$ to produce another optimal solution at which equality in (20) holds. First, notice that there must exist $\delta_{(i)} \geq 0$ for $i \in \mathcal{I} \triangleq \{J+1, J+2, ..., V\}$, such that reducing $l_{(i)}^*$ by $\delta_{(i)}$ for all $i \in \mathcal{I}$ could meet the lower bound in (20). Next, to ensure that (12) still holds, we can reduce $r_{(i)}^*$ to $(r_{(i)}^* - \delta_{(i)})^+$ for all $i \in \mathcal{I}$, where

$$(x)^+ = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}.$$

If $r_{(i)}^* < \delta_{(i)}$ for some $i \in \mathcal{I}$, the left hand side of (21) will be strictly greater than the right hand side. If that is the case, we reduce the values of those $r_{(i)}^*$'s that are positive such that equality holds. This can always be done, since $r_{(i)}$'s are free to decrease to any non-negative values. After this change, the resultant completion time remains the same as T^*, since T^* is already at the minimum. □

Lemma 3. *Given a feasible problem instance, there always exits an optimal allocation vector $\boldsymbol{l}^*$ which satisfies $l_1^* \geq l_2^* \geq ... \geq l_V^*$.*

The proof is omitted due to space limitation.

From Lemmas 2 and 3, we can rewrite the original optimization problem as

$$\min_{\boldsymbol{l},\boldsymbol{r}} T \tag{22}$$

subject to

$$\sum_{i\in\mathcal{V}} l_i \leq S \tag{23}$$

$$l_1 \geq l_2 \geq ... \geq l_V \tag{24}$$

$$\sum_{i=J+1}^{V} l_i - \sum_{i=1}^{V-K} l_i = 1 \tag{25}$$

$$0 \leq r_i \leq l_i, \ \forall i \in \mathcal{V} \tag{26}$$

$$T \geq \frac{r_i}{\mu_i}, \ \forall i \in \mathcal{V} \tag{27}$$

$$\sum_{i\in\mathcal{V}} r_i = \sum_{i=V-K+1}^{V} l_i \tag{28}$$

For any feasible problem instance, we only find the optimal allocation vector $\boldsymbol{l}^*$ which satisfies $l_1^* \geq ... \geq l_V^*$. The optimal load vector $\boldsymbol{r}^*$ that achieves T^* must satisfy $r_i^* \leq l_i^*$ and $r_i^* \leq \mu_i T^*$, $i \in \mathcal{V}$. Denote the storage budget by $S \triangleq S_0 + \Delta s \triangleq V l_0 + \Delta s$, where $l_0 \triangleq \frac{1}{K-J}$. Given any $\Delta s \geq 0$, we define $\mathcal{B}_{\Delta s} \triangleq \{i \in \mathcal{V} : r_i^* = \mu_i T^*, S = S_0 + \Delta s\}$, which contains the indices of clouds halting at T^*. We call them the *bottleneck* clouds. Accordingly, the index set of the *non-bottleneck* clouds for Δs should be the complement of $\mathcal{B}_{\Delta s}$, i.e., $\bar{\mathcal{B}}_{\Delta s} \triangleq \mathcal{V} \setminus \mathcal{B}_{\Delta s} = \{i \in \mathcal{V} : r_i^* < \mu_i T^*, S = S_0 + \Delta s\}$. Let $x \triangleq |\bar{\mathcal{B}}_{\Delta s}|$ be the number of non-bottleneck clouds. For notation simplicity, the dependence of x on Δs is not explicitly shown.

First, we consider the scenario where the storage budget is minimal, i.e., $\Delta s = 0$. Denote the corresponding minimum completion time by T_0, and the optimal storage allocation vector and load vector achieving T_0 by $\boldsymbol{l}_0 \triangleq (l_1^0, l_2^0, \ldots, l_V^0)$ and $\boldsymbol{r}_0 \triangleq (r_1^0, r_2^0, \ldots, r_V^0)$, respectively. According to Lemma 1, when $S = S_0$ the allocation vector must be $\boldsymbol{l}_0 = l_0\mathbf{1}_V$, and then the total computation load in (28) becomes

$$\sum_{i\in\mathcal{V}} r_i^0 = K l_0. \tag{29}$$

Recall $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_V$. For notation simplicity, let μ_0 be a sufficiently large number such that $\frac{l_0}{\mu_0} < T_0$. Then there exists $Q \in \{0, 1, \ldots, V\}$ such that

$$\frac{l_0}{\mu_0} \leq \frac{l_0}{\mu_1} \leq \cdots \leq \frac{l_0}{\mu_Q} < T_0 \leq \frac{l_0}{\mu_{Q+1}} \leq \cdots \leq \frac{l_0}{\mu_V}. \tag{30}$$

Note that Q can be uniquely determined as T_0 is well defined.

Lemma 4. *If $\Delta s = 0$, then $x = Q$, and r_i^0 equals l_0 for $i \in \bar{\mathcal{B}}_0$ and equals $\mu_i T_0$ for $i \in \mathcal{B}_0$, where $T_0 = \frac{(K-Q)l_0}{\sum_{i\in\mathcal{B}_0}\mu_i}$, $\bar{\mathcal{B}}_0 = \{1, 2, \ldots, Q\}$ and $\mathcal{B}_0 = \{Q+1, Q+2, \ldots, V\}$.*

Proof. Define $\mathcal{I} \triangleq \{1, 2, \ldots, Q\}$. By the definition of Q, $r_i^0 \leq l_i < \mu_i T_0, \forall i \in \mathcal{I}$. Then $\mathcal{B}_0 \cap \mathcal{I} = \varnothing$, i.e., $\mathcal{B}_0 \subseteq \mathcal{V} \setminus \mathcal{I}$.

Suppose $r_i^0 < l_0$ for some $i \in \bar{\mathcal{B}}_0$. If we set r_i^0 as $r_i^0 + \epsilon$ and r_j^0 as $r_j^0 - \frac{\epsilon}{|\mathcal{B}_0|}$ $\forall j \in \mathcal{B}_0$, where $\epsilon > 0$ is arbitrarily small, then the resultant computation time $T'' < T_0$, which contradicts with T_0 being optimal. So $r_i^0 = l_0$ $\forall i \in \bar{\mathcal{B}}_0$, and

$$T_0 = \frac{\sum_{i\in\mathcal{V}} r_i^0 - \sum_{i\in\bar{\mathcal{B}}_0} r_i^0}{\sum_{i\in\mathcal{B}_0}\mu_i} = \frac{K l_0 - Q l_0}{\sum_{i\in\mathcal{B}_0}\mu_i}. \tag{31}$$

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on September 08,2020 at 08:12:01 UTC from IEEE Xplore. Restrictions apply.

Suppose $\mathcal{B}_0 \subsetneq \mathcal{V} \setminus \mathcal{I}$. Pick any $j \in (\mathcal{V} \setminus \mathcal{I}) \setminus \mathcal{B}_0$, and we set r_i^0 as $r_i^0 - \eta$ for all $i \in \mathcal{B}_0$ and r_j^0 as $r_j^0 + |\mathcal{B}_0|\eta$, where $\eta > 0$ is arbitrarily small. Then the resultant computation time $T' < T_0$, which contradicts with T_0 being optimal. So $\mathcal{B}_0 = \mathcal{V} \setminus \mathcal{I}$ and thus $\bar{\mathcal{B}}_0 = \mathcal{I}$, i.e., $x = Q$. □

Lemma 4 implies $0 \le Q < K$. From (30) and (31), we can determine $Q \in \{0, 1, \ldots, K-1\}$ as the minimal integer

$$Q + \frac{1}{\mu_{Q+1}} \sum_{i \in \mathcal{B}_0} \mu_i \ge K. \tag{32}$$

For $i \in \mathcal{V}$, define $\Delta l_i \triangleq l_i - l_0$, which is not necessarily non-negative. To ensure feasibility, Δl_i's have to satisfy (23), (24) and (25) respectively:

$$0 \le \sum_{i \in \mathcal{V}} \Delta l_i \le \Delta s, \tag{33}$$

$$\Delta l_1 \ge \Delta l_2 \ge \cdots \ge \Delta l_V, \tag{34}$$

$$\sum_{i=J+1}^{V} \Delta l_i = \sum_{i=1}^{V-K} \Delta l_i. \tag{35}$$

Lemma 5. *$0 \le \Delta l_J \le \lambda \Delta s$. The upper bound is achieved if and only if $\Delta l_1 = \cdots = \Delta l_J = \lambda \Delta s$, where $\lambda \triangleq \frac{1}{J+V-K}$.*

Proof. Assume $\Delta l_J < 0$. According to (34) and (35) we have $\sum_{i=1}^{V-K} \Delta l_i < 0$. The smallest term, Δl_{V-K}, must be negative, which by (35) again, implies all subsequent terms are negative. Therefore, $\sum_{i \in \mathcal{V}} \Delta l_i < 0$, which contradicts with (33), so we must have $\Delta l_J \ge 0$.

From (33) and (35) we have

$$\sum_{i \in \mathcal{V}} \Delta l_i = \sum_{i=1}^{V-K} \Delta l_i + \sum_{i=1}^{J} \Delta l_i \le \Delta s.$$

By (34) and that $J > V - K$, Δl_J is the smallest term among the $1/\lambda$ terms in the above two summations, so

$$\Delta l_J \le \lambda \Delta s, \tag{36}$$

with equality if and only if $\Delta l_1 = \cdots = \Delta l_J = \lambda \Delta s$. □

Let T be the overall computing time, which is achieved by $\boldsymbol{l} \triangleq (l_0 + \Delta l_1, l_0 + \Delta l_2, \ldots, l_0 + \Delta l_V)$ and $\boldsymbol{r} \triangleq (r_1, r_2, \ldots, r_V)$. Since $T \ge \frac{r_i}{\mu_i}$ for all $i \in \mathcal{V}$,

$$\begin{aligned} T &\ge \frac{\sum_{i \in \mathcal{B}_0} r_i}{\sum_{i \in \mathcal{B}_0} \mu_i} = \frac{\sum_{i \in \mathcal{V}} r_i - \sum_{i \in \bar{\mathcal{B}}_0} r_i}{\sum_{i \in \mathcal{B}_0} \mu_i} \\ &\ge \frac{\sum_{i=V-K+1}^{V} l_i - \sum_{i \in \bar{\mathcal{B}}_0} l_i}{\sum_{i \in \mathcal{B}_0} \mu_i} \overset{(a)}{=} \frac{(K-Q) l_0 + \sum_{i=1}^{J} \Delta l_i - \sum_{i=1}^{Q} \Delta l_i}{\sum_{i \in \mathcal{B}_0} \mu_i}, \end{aligned} \tag{37}$$

where (a) comes from (35) and $J > V - K$. If the lower bound is achieved, the storage allocation and computation load vectors are optimal, which occurs if and only if $\mathcal{B}_{\Delta s} \supseteq \mathcal{B}_0$ and $r_i^* = l_i^*$ $\forall i \in \bar{\mathcal{B}}_0$.

Theorem 1. *If $Q \le J$, for any $S \ge S_0$, the optimal computation time is $T^* = T_0$, which is achieved by $\boldsymbol{l}^* = \boldsymbol{l}_0$ and $\boldsymbol{r}^* = \boldsymbol{r}_0$.*

Proof. According to (37) and Lemma 5, we have

$$T - T_0 \ge \frac{\sum_{i=1}^{J} \Delta l_i - \sum_{i=1}^{Q} \Delta l_i}{\sum_{i \in \mathcal{B}_0} \mu_i} \ge 0, \tag{38}$$

which means the computation time cannot be less than T_0 for any $S \ge S_0$. Therefore, the optimal computation time is $T^* = T_0$, which can be achieved by $\boldsymbol{l}^* = \boldsymbol{l}_0$ and $\boldsymbol{r}^* = \boldsymbol{r}_0$. □

Now we consider the scenario where $Q > J$. Define $\mathcal{D}_1 \triangleq \{i \in \bar{\mathcal{B}}_0 : \mu_i = \mu_Q\} = \{Q - q_1 + 1, \ldots, Q\}$, where $q_1 = |\mathcal{D}_1|$. If $J \notin \mathcal{D}_1$, define $\mathcal{D}_2 \triangleq \{i \in \bar{\mathcal{B}}_0 : \mu_i = \mu_{Q-q_1}\} = \{Q - q_2 + 1, \ldots, Q - q_1\}$, where $q_2 = |\mathcal{D}_1| + |\mathcal{D}_2|$. If $J \notin \mathcal{D}_2$, continue defining $\mathcal{D}_3$, $\mathcal{D}_4$, ... until $J \in \mathcal{D}_{n+1}$, where $\mathcal{D}_{n+1} \triangleq \{i \in \bar{\mathcal{B}}_0 : \mu_i = \mu_{Q-q_n}\}$. For example, if $\mu_1 > \mu_2 > \cdots > \mu_V$, then $n = Q - J$, and $D_i = \{Q - i + 1\}$ and $q_i = i$ for $i = \{1, 2, \ldots, n+1\}$. The result is shown below. Due to space limitation, the proof of is omitted.

Theorem 2. *Denote $q_0 \triangleq 0$. If $Q > J$, for any $\Delta s \ge 0$,*

$$x = \begin{cases} Q - q_0 & 0 \le \Delta s < \Delta s_1 \\ Q - q_1 & \Delta s_1 \le \Delta s < \Delta s_2 \\ \vdots & \\ Q - q_{n+1} & \Delta s_{n+1} \le \Delta s \end{cases},$$

where for $i = 1, 2, \ldots, n+1$,

$$\Delta s_i = \frac{\mu_y l_0 (K - y) - l_0 \sum_{i=y+1}^{V} \mu_i}{\lambda \sum_{i=y+1}^{V} \mu_i + \lambda \mu_y (y - J)}, \; y \triangleq Q - q_{i-1}. \tag{39}$$

(i) If $0 \le \Delta s < \Delta s_{n+1}$, the optimal computing time is

$$T^* = \frac{(K - x) l_0 - (x - J) \lambda \Delta s}{\sum_{i=x+1}^{V} \mu_i},$$

which is achieved by the optimal allocation

$$r_i^* = \begin{cases} l_0 + \lambda \Delta s = l_i^* & i \in \{1, 2, \ldots, x\} \\ \mu_i T^* & i \in \{x+1, x+2, \ldots, V\} \end{cases},$$

and

$$l_i^* = r_i^* + \delta_i, \;\; i \in \{x+1, x+2, \ldots, V\},$$

where δ_i can be any non-negative number satisfying

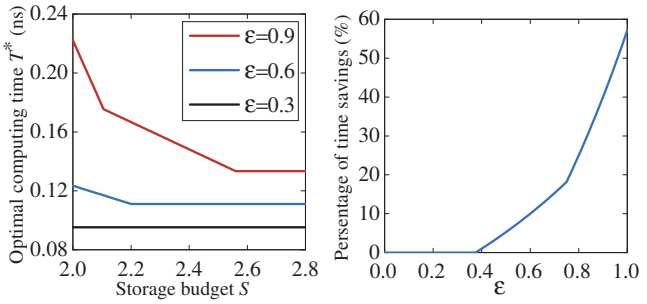
$$\sum_{i=x+1}^{V} \delta_i = (V - K) l_0 + (1 - \lambda J) \Delta s, \tag{40}$$

and $l_0 + \lambda \Delta s \ge l_{x+1}^* \ge l_{x+2}^* \ge \cdots \ge l_V^*$.

(ii) If $\Delta s \ge \Delta s_{n+1}$, T^*, r_i^*'s and l_i^*'s are equal to the case where $\Delta s = \Delta s_{n+1} - \epsilon$, where $\epsilon \to 0^+$.

Theorem 2 implies that T^* is a piecewise linear function of

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on September 08,2020 at 08:12:01 UTC from IEEE Xplore. Restrictions apply.



(a) Tradeoff between storage budget and optimal computing time

(b) Time reduction by our proposed allocation

Fig. 2. Numerical results for a $(6, 5, 2)$ system

Δs. When $x > J$, it decreases linearly at rate k, where

$$k = \frac{-(x-J)\lambda}{\sum_{i=x+1}^{V} \mu_i}. \tag{41}$$

If x drops to J or below, T^* becomes a constant function of Δs. The next result, which characterizes the geometric shape of the tradeoff curve between optimal completion time and storage budget, follows immediately from Theorems 1 and 2:

Corollary 1. *T^* is a piecewise linear function of S. If $\mu_i \neq \mu_j$ $\forall i, j \in \mathcal{V}$, then there are $Q - J$ turning points.*

IV. Numerical Results

In this section, we demonstrate the performance of our proposed allocation scheme numerically. For comparison purpose, we use the *equal* allocation scheme as a benchmark. For any storage budget greater than or equal to S_0, the equal allocation scheme stores only S_0, which achieves the completion time of T_0 by $\boldsymbol{l} = \boldsymbol{l}_0$ and $\boldsymbol{r} = \boldsymbol{r}_0$. The storage budget is not fully utilized because if we increase the storage amount of each cloud equally, the computation time will increase due to the security constraint.

We consider a $(V, K, J) = (6, 5, 2)$ system, which requires a minimum storage budget $S_0 = 2$. The computing rates of the six clouds, measured in terms of giga rows per second, are assumed to be $3 + 3\epsilon$, $3 + 2\epsilon$, $3 + \epsilon$, $3 - \epsilon$, $3 - 2\epsilon$, and $3 - 3\epsilon$, where $\epsilon \in [0, 1]$ is a parameter that controls the variation of the computing rates among the clouds and also determines the value of Q.

In the first test, we consider these three systems: $(\epsilon, Q) = (0.3, 2), (0.6, 3)$, and $(0.9, 4)$. Fig. 2(a) shows that the optimal computing time T^* achieved by our proposed allocation is no more than T_0 achieved by equal allocation in all three cases as expected. Besides, less variation of computing rates (i.e., smaller value of ϵ) results in shorter computing time. When $Q > J$, i.e., $\epsilon = 0.9$ or 0.6, T^* drops in a piecewise-linear fashion with the growth of S and there are $Q - J$ turning points in each of the two tradeoff curves. When $Q \leq J$, i.e., $\epsilon = 0.3$, T^* is always equal to T_0, so both allocations have the same performance.

In the second test, we calculate the percentage reduction of computing time achieved by our proposed allocation. Fig. 2(b) is obtained by increasing the value of ϵ with a step size of 0.001. It can be seen that the percentage reduction increases rapidly for large value of ϵ, showing that a significant gain can be obtained when the computing rates of the clouds are vastly different.

V. Conclusion

Security is a major concern for users performing big-data computing in clouds. To ensure data confidentiality, a user may store data and perform computing over multiple clouds. This work establishes a framework for the use of nested MDS code to ensure perfect secrecy. The tradeoff between storage amount and minimum computing time is explicitly characterized. The optimal computing time is shown to be a piece-wise linear decreasing function of the storage budget. When the storage budget exceeds a certain threshold, the tradeoff curve becomes flat and further increase in storage budget cannot reduce the computing time. In general, the clouds may not have the same computing rate. To better utilize the cloud services, an optimal allocation scheme is derived. Numerical results demonstrated that it outperforms equal allocation, especially when the computing capability of different clouds varies largely.

References

[1] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514-1529, Mar. 2018.

[2] S. Li, S. M. Mousavi Kalan, A. S. Avestimehr and M. Soltanolkotabi, "Near-Optimal Straggler Mitigation for Distributed Gradient Methods," *IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 857-866, May 2018.

[3] A. Severinson, A. G. i Amat, and E, Rosnes, "Block-diagonal and LT codes for distributed computing with straggling servers," *IEEE Transactions on Communications*, vol. 67, no. 3, pp. 1739-1753, Mar. 2019.

[4] H. Park, K. Lee, J. Sohn, C. Suh and J. Moon, "Hierarchical coding for distributed computing," *IEEE International Symposium on Information Theory (ISIT),* pp. 1630-1634, Jun. 2018.

[5] A. Reisizadeh, S. Prakash, R. Pedarsani and A. S. Avestimehr, "Coded computation over heterogeneous clusters," *IEEE Transactions on Information Theory*, vol. 65, no. 7, pp. 4227-4242, July 2019.

[6] D. J. Kim, H. Park, and J. Choi, "Optimal load allocation for coded distributed computation in heterogeneous clusters," *arXiv preprint arXiv:1904.09496,* 2019.

[7] H. Yang and J. Lee, "Secure distributed computing with straggling servers using polynomial codes," *IEEE Transactions on Information Forensics and Security,* vol. 14, no. 1, pp. 141-150, Jan. 2019.

[8] R. Bitar, P. Parag, and S.El Rouayheb, "Minimizing latency for secure distributed computing," *IEEE International Symposium on Information Theory (ISIT),* pp. 2900-2904, Jun. 2017.

[9] S. Li, M. A. Maddah-Ali, Q. Yu and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Transactions on Information Theory,* vol. 64, no. 1, pp. 109-128, Jan. 2018.

[10] M. Ali, S. U. Khan, and A. V. Vasilakos, "Security in cloud computing: Opportunities and challenges," *Information sciences*, vol. 305, pp. 357-383, Jun. 2015.

[11] P. Hu, C. W. Sung, S. Ho and T. H. Chan, "Optimal coding and allocation for perfect secrecy in multiple clouds," *IEEE Transactions on Information Forensics and Security,* vol. 11, no. 2, pp. 388-399, Feb. 2016.

[12] S. Arunkumar and S. W. Mclaughlin, "MDS codes on erasure-erasure wire-tap channel," *Available: arXiv:0902.3286v1*, 2009.

[13] S. Pawar, S. El Rouayheb, and K. Ramchandran, "Securing dynamic distributed storage systems against eavesdropping and adversarial attacks," *IEEE Transactions on Information Theory,* vol. 57, no. 10, pp. 6734-6753, Oct. 2011.

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on September 08,2020 at 08:12:01 UTC from IEEE Xplore. Restrictions apply.