Mean Latency Optimization in Erasure-coded Distributed Storage Systems

Abubakr O. Al-Abbasi, Vaneet Aggarwal
Purdue University, West Lafayette, IN 47907, USA
email:{aalabbas,vaneet}@purdue.edu

***Abstract*—Modern distributed storage systems represent an elegant solution to accommodate the exponentially increasing need of storage space. They often use erasure coding because of its high resiliency with low storage overhead. Further, distributed data-center applications are known to be susceptible to long response time, and higher latency leads to reduction in customers' quality of experience. However, to the best of our knowledge, quantifying the impact of erasure coding on th mean latency is still an open problem for distributed erasure-coded storage. In this paper, we propose a framework for quantifying and optimizing the mean latency in erasure-coded storage systems. In particulate, we characterize mean latency in a tight upper bound. Then, we formulate an optimization problem to optimize the mean latency of all files over the choice of storage servers and auxiliary bound parameters. An alternating optimization algorithm is used to efficiently solve the nonconvex problem. Our evaluation results show the superiority of our approach as compared to the state of the art algorithms.**

***Index Terms*—Alternating Optimization, Distributed Storage Systems, Erasure Coding, Mean Latency.**

I. Introduction

A key advantage of distributed storage systems and cloud computing is the large-scale of resource sharing, flexibility and scalability. This motivates service providers to continuously keep increasing the number of applications that are hosted on the cloud. Some examples of these applications are storage (such as Google Drive, Dropbox, Apple's iCloud, One Drive), streaming (NetFlix, YouTube) and computing (Amazon EC2, Microsoft Azure). However, a counter effect of sharing the resources on the cloud lies in the latency experienced by the users due to queuing, scheduling among storage servers, virtualization, storage servers outage, etc. An elegant solution to reduce latency in such systems is the use of redundancy.

In cloud storage, replication a task on multiple servers and waiting for the earliest copy to complete can significantly reduce the latency [1]. However, task replication can result in increased storage cost and overuse of network resources and system bandwidth. To overcome these drawbacks of replication systems, erasure coding represents a promising technique to reduce the storage cost (up to 50% or even more for a given reliability) as compared to the replicated systems [2], [3]. Due to its reduced storage cost, erasure-coding has been widely used in modern storage systems by companies like Google [4], Facebook [5] and Microsoft [6]. However, erasure-coded storage is known to be susceptible to long latency which affects the quality of service that users experience and, hence, the file access time would increase. In this paper, we consider the file download time as a measure metric for the quality of experience (QoE). Users oftenly relate the latency to the quality of experiences for downloading the files and is thus one of the key focus in the studied latency reduction approaches, e.g., [7], [8].

Unlike replication systems, where a file can be recovered from any one of the n replica copies, for an (n, k) erasure-coded storage system [2], a file is encoded into n equal-size data chunks which allows retrieving the file from any subset of $k < n$ chunks. Thus, only k distinct chunks from different servers are needed to reconstruct the entire file. However, since the slowest server (hottest storage with highest congestion) becomes the bottleneck, this leads to a significant increase in the mean latency. Moreover, the overall response time in erasure coded data-storage systems is determined by the slowest server response of the required parallel operations [9], which experiences a high level of variability in delay performance [10]. Thus, an efficient fast and seamless service represents a challenge in cloud storage systems.

Recently, bounds on mean service latency [7], [8], [11]–[14] are provided. In [15], mean service latency is provided for replication based systems for homogeneous servers with independent exponential service times. However, erasure-coded based system is still an open problem. In [8], [10], [16]–[20], Fork-join approach is used to provide upper bounds for the mean service latency by forking each file request to all storage servers. Further, in [11], [14] a queuing-theoretic based approach is employed to propose a block-based-scheduling where only the first, say d, requests at the head of the queue are served. However, none of these approaches quantify tail latency because states of the corresponding queuing model have to consider not only the current status of the system (e.g., chunk placement and queued requests) but also prior history of how chunk requests are treated by individual nodes, which causes a state explosion problem. In [7], [12], mean latency bounds for arbitrary service time distribution and heterogeneous files are proposed using order statistic analysis and a probabilistic request scheduling policy. However, the proposed bounds in [7], [12] are not tight. In [21], using uniform probabilistic server access with exponential service times, authors show improved performance for erasure coding in terms of mean latency as compared to replication in the limit of large number of servers for replication-based systems. Further, some recent works on finding tight bounds for the average delay in storage systems are studied [22], [23]. However, our model and analysis are different than those works and further generalize some of the assumptions made by the authors.

In this work, we aim to provide a mathematical crystallization of the mean latency and thus illuminate key system design issues by optimizing the average QoE metric over different requests for the files with the help of bound parameters. An efficient algorithm is proposed to solve the proposed nonconvex problem. The proposed algorithm performs an alternating optimization over two subproblems -easy to handle- including the storage server access, and the auxiliary bound parameters. The optimization over probabilistic scheduling access parameters helps reduce the mean latency by differentiating files and hence providing more flexibility as compared to choosing the lowest-queue servers. The sub-problems have been shown to have convex constraints and thus can be efficiently solved using iNner cOnVex Approximation (NOVA) algorithm proposed in [24], [25]. The proposed algorithm is shown to converge to a local optimal. Our results demonstrate a significant improvement of the QoE metric as compared to the state of the art algorithms and some other considered baselines. To summarize, the key contributions of this paper are:

- We quantify a tight outer bound on the mean service latency for arbitrary erasure codes and for any number of files in distributed storage.
- We used the defined QoE metric to formulate a system optimization problem over the choice of the probabilistic scheduling access policy and the bound auxiliary parameters.
- We propose an alternating optimization algorithm with proven convergence for the joint QoE metrics optimization problem.
- Numerical results show that the proposed algorithm converges within a few iterations. Further, the QoE metric is shown to have a significant improvement as compared to the state of the art algorithms as well as considered baselines. For instance, the mean latency for the proposed algorithm, at the highest arrival rate, is one-fourth smaller than the mean latency obtained using the algorithm in [12], [13].

II. System Model

We consider a content delivery network as shown in Figure 1, consisting of m distributed storage servers (nodes) denoted by $\mathcal{M} = 1, \ldots, m$, and an edge router. A set of i contents denoted by $i = 1, \ldots, r$ are distributively stored in the storage servers. Multiple users are connected to the edge-router, where we assume that the connection between the users and the edge router is infinite and thus only consider the links from the servers to the edge router. Hence, we can consider edge router as an aggregation of multiple users. Further, we assume that each content is divided into k_i equal-size chunks and then encoded using (n_i, k_i) maximum distance separable (MDS) erasure code to create n_i distinct chunks of the same size for file i which in turn introduces a redundancy factor of n_i/k_i. We assume that the chunk size is the same for all files, although the coding across files can be different.

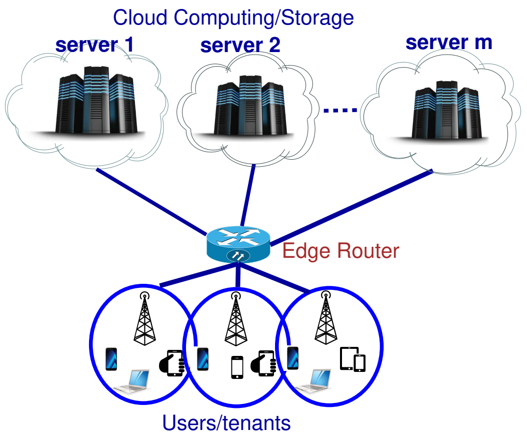

Fig. 1: An illustration of the system model for content delivery network, consisting of m storage servers and an edge router. Upon an arrival of file i to the edge router, a batch of k_i-out-of-n_i chunk requests are dispatched to appropriate a set of storage nodes that host file i.

Hence, the file can be retrieved from any subset of k_i-out-of-n_i servers. These encoded chunks are stored on n_i distinct storage nodes, denoted by $\mathcal{S}_i$ such that $\mathcal{S}_i \subseteq \mathcal{M}$ and $n_i = |\mathcal{S}_i|$. In order to serve the incoming requests at the edge router, k_i distinct servers are selected by the edge router to reconstruct the requested file. The selection process is performed probabilistically using the scheduling approach proposed in [12], so-called probabilistic scheduling and is explained as follows. For each file request, a batch of k_i chunk requests is forwarded to appropriate a set of nodes (denoted by set $\mathcal{G}_i$ of servers for file i) with known probability, i.e., $P(\mathcal{G}_i)$ for set $\mathcal{G}_i$ for file i. Then, each server manages the requests in a local queue of infinite capacity and treats them independently and in order. The file request is completed if all its chunk requests are processed (served) by individual nodes. It has been shown that a probabilistic scheduling policy with feasible probabilities $\{P(\mathcal{G}_i) : \forall i, \mathcal{G}_i\}$ exists if and only if there exists conditional probabilities $q_{i,j} \in [0,1]$ satisfying

$$\sum_{j=1}^{m} q_{i,j} = k_i \, \forall i, \text{ and } q_{i,j} = 0 \, \text{if } j \notin \mathcal{G}_i \tag{1}$$

We assume that the arrival of requests at the edge router for each file i form an independent Poisson process with a known rate λ_i. Using probabilistic scheduling, the arrival of chunk requests at server j forms a Poisson Process with rate $\Lambda_j = \sum_i \lambda_i q_{i,j}$ which is the superposition of r Poisson processes each with rate $\lambda_i q_{i,j}$. Further, we assume chunk service time $\mathbb{X}_j$ of node j follows a shifted exponential distribution as demonstrated in realistic systems [12], [13], [18]. The service time distribution for the chunk at server j, $\mathbb{X}_j$, is given by the probability distribution function $g_j(x)$, which is

$$g_j(x) = \begin{cases} \alpha_j e^{-\alpha_j(x-\beta_j)} & x \geq \beta_j \\ 0 & x < \beta_j \end{cases} \tag{2}$$

The exponential distribution follows as a special case in the above expression if $\beta_j = 0$. Let $\mathbb{Z}_j(t) = \mathbb{E}\left[e^{t\mathbb{X}_j}\right]$ be the moment generating function of $\mathbb{X}_j$. Then, $\mathbb{Z}_j(t)$ is given by

$$\mathbb{Z}_j(t) = \frac{\alpha_j e^{\beta_j t}}{\alpha_j - t} \tag{3}$$

where α_j is the rate and β_j mainly represents the read time and related processing times.

III. Latency Analysis

In this section, for erasure-coded storage, we will provide a tight bound for the mean latency for file i. This bound is derived based on the probabilistic scheduling and since this policy is a feasible strategy, the obtained bound is an upper bound to the optimal strategy. Next, a quantification for the tail latency is provided.

A. Mean Latency Bound

Since local queues at different storage nodes are dependent of each other because each batch of chunk requests is jointly dispatched, the exact analysis of the queuing latency of probabilistic scheduling is impractical. Thus, we will use probabilistic scheduling to bound the mean latency and since probabilistic scheduling is a feasible strategy, the obtained bound is an upper bound to the optimal strategy. We define $\mathbb{W}_{i,j}$ as the random waiting time (sojourn time) in which a chunk request (for file i) spends in the queue of node j. Typically, the expected latency of file i, denoted as $\mathbb{Q}_i$, request is determined by the maximum latency that k_i chunk requests experience on distinct servers. These servers are probabilistically scheduled with a prior known probabilities, i.e., $q_{i,j}$. Thus, we have

$$\mathbb{Q}_i \triangleq \mathbb{E}_{\mathbb{W}_{i,j}}\left[\mathbb{E}_{\mathcal{G}_i}\left[\max_{j\in\mathcal{G}_i} \mathbb{W}_{i,j}\right]\right] \tag{4}$$

where the first expectation $\mathbb{E}_{\mathbb{W}_j}$ is taken over system queuing dynamics and the second expectation $\mathbb{E}_{\mathcal{G}_i}$ is taken over random dispatch decisions $\mathcal{G}_i$. Hence, we derive an upper-bound on the expected latency of a file i, i.e., $\mathbb{Q}_i$, as follows. Using Jensen's inequality [26], we have for $t_i > 0$

$$e^{t_i\mathbb{E}[\mathbb{Q}_i]} \leq \mathbb{E}\left[e^{t_i\mathbb{Q}_i}\right] \tag{5}$$

We notice from (5) that by bounding the moment generating function of $\mathbb{Q}_i$ we are bounding the mean latency of file i. Then,

$$\begin{aligned}
\mathbb{E}\left[e^{t_i\mathbb{Q}_i}\right] &\overset{(a)}{=} \mathbb{E}_{\mathcal{G}_i,\mathbb{W}_{i,j}}\left[\max_{j\in\mathcal{G}_i} e^{t_i\mathbb{W}_{i,j}}\right] && (6)\\
&= \mathbb{E}_{\mathcal{G}_i}\left[\mathbb{E}_{\mathbb{W}_{i,j}}\left[\max_{j\in\mathcal{G}_i} e^{t_i\mathbb{W}_{i,j}}|\mathcal{G}_i\right]\right] && (7)\\
&\overset{(b)}{\leq} \mathbb{E}_{\mathcal{G}_i}\left[\sum_{j\in\mathcal{G}_i}\mathbb{E}_{\mathbb{W}_{i,j}}\left[e^{t_i\mathbb{W}_{i,j}}\right]\right] && (8)\\
&= \sum_j \mathbb{E}_{\mathbb{W}_{i,j}}\left[e^{t_i\mathbb{W}_{i,j}}\right]\mathbb{E}_{\mathcal{G}_i}\left[\mathbf{1}_{(j\in\mathcal{G}_i)}\right] && (9)\\
&= \sum_j \mathbb{E}_{\mathbb{W}_{i,j}}\left[e^{t_i\mathbb{W}_{i,j}}\right]\mathbb{P}(j\in\mathcal{G}_i) && (10)\\
&\overset{(c)}{=} \sum_j q_{i,j}\mathbb{E}_{\mathbb{W}_{i,j}}\left[e^{t_i\mathbb{W}_{i,j}}\right] && (11)
\end{aligned}$$

where (a) follows from (4) and (5), (b) follows by replacing the $\max_{j\in\mathcal{G}_i}$ by $\sum_{j\in\mathcal{G}_i}$ and (c) follows by probabilistic scheduling. We note that the only inequality here is for replacing the maximum by the sum. However, since this term will be inside the logarithm for the mean latency, the gap between the term and its bound becomes additive rather than multiplicative. Since the request pattern is Poisson and the service time is general distributed, the Laplace-Stieltjes Transform of the waiting time $\mathbb{W}_{i,j}$ can be characterized using Pollaczek-Khinchine formula for M/G/1 queues [27] as follows

$$\mathbb{E}\left[e^{t_i\mathbb{W}_{i,j}}\right] = \frac{(1-\rho_j)\,t_i\mathbb{Z}_j(t_i)}{t_i - \Lambda_j\left(\mathbb{Z}_j(t_i)-1\right)} \tag{12}$$

where $\rho_j = \Lambda_j\mathbb{E}\left[\mathbb{X}_j\right] = \Lambda_j\left[\frac{d}{dt}\mathbb{Z}_j(t_i)\,|_{t_i=0}\right]$ and $\mathbb{Z}_j(t_i)$ is defined in (3). Plugging (12) in (11) and substituting in (5), we get the following Theorem.

Theorem 1. *The mean latency for file i is bounded by*

$$\mathbb{Q}_i \leq \frac{1}{t_i} log\left(\sum_{j=1}^{m} q_{i,j}\frac{(1-\rho_j)t_i\mathbb{Z}_j(t_i)}{t_i - \Lambda_j\left(\mathbb{Z}_j(t_i)-1\right)}\right) \tag{13}$$

for any $t_i > 0$, $\rho_j = \Lambda_j\left(\beta_j + \frac{1}{\alpha_j}\right)$ and $\Lambda_j\left(\mathbb{Z}_j(t_i)-1\right) < t_i$.

Note that the above Theorem holds only in the range of t_i when $t_i - \Lambda_j\left(\mathbb{Z}_j(t)-1\right) > 0$ which reduces to $t_i\left(t_i - \alpha_j + \Lambda_j\right) + \alpha_j\Lambda_j\left(e^{\beta_j t_i} - 1\right) < 0$. Further, the server utilization ρ_j must be less than 1 for stability of the system. To this end, we note that the authors of [12], [13] gave an upper bound for mean file download time using probabilistic scheduling. However, the bound in this paper is different and tighter since we use moment generating function based bound rather than bounding the maximum of m random variables by their sum as in [12], [13]. Additionally, the two bounds are compared in Section V and the bound in this paper is shown to outperform that in [12], [13]. Moreover, replication coding follows as a special case when $k_i = 1$ and thus the proposed upper bound for file download can be used to bound the latency of replication based systems by setting $k_i = 1$.

IV. Problem Formulation and Proposed Algorithm

In this section, we formulate the mean latency optimization assuming multiple and heterogeneous files.

A. Problem Formulation

We define $\boldsymbol{q} = (q_{i,j} \forall i = 1, \cdots, r$ and $j = 1, \cdots, m)$, and $\boldsymbol{t} = (t_1, t_2, \ldots, t_r)$. Our goal is to minimize the two proposed QoE metrics over the choice of access decisions and auxiliary bound parameters. The objective can be modeled as a convex combination of the two QoE metrics since this is a multi-objective optimization.

To incorporate for weighted fairness and differentiated services, we assign a positive weight w_i for each QoE for file i. Without loss of generality, each file i is weighted by the arrival rate λ_i in the objective (so larger arrival rates are weighted higher). However, any other weights can be incorporated to accommodate for weighted fairness or differentiated services. Let $\overline{\lambda} = \sum_i \lambda_i$ be the total arrival rate. Hence, $w_i = \lambda_i / \overline{\lambda}$ is the ratio of file i requests. Our objective is to minimize the mean latency, averaged over all the file requests, and is given as $\sum_i \frac{\lambda_i}{\overline{\lambda}} \mathbb{Q}_i$. By using the expression for the mean latency in Section (III-A), optimization of the weighted mean latency metric can be formulated as follows.

$$\textbf{min} \quad \sum_{i=1}^{r} \frac{\lambda_i}{\overline{\lambda}} \frac{1}{t_i} \log \left(\sum_{j=1}^{m} q_{i,j} \frac{(1-\rho_j) t_i \mathbb{Z}_j(t_i)}{t_i - \Lambda_j \left(\mathbb{Z}_j(t_i) - 1 \right)} \right) \tag{14}$$

$$\text{s.t.} \quad \mathbb{Z}_j(t_i) = \frac{\alpha_j}{\alpha_j - t_i} e^{\beta_j t_i} \ , \forall j \tag{15}$$

$$\rho_j = \frac{\Lambda_j}{\alpha_j} + \Lambda_j \beta_j < 1 \ , \forall j \tag{16}$$

$$\Lambda_j = \sum_i \lambda_i q_{i,j} \ , \forall j \tag{17}$$

$$\sum_j q_{i,j} = k_i \ , \forall i \tag{18}$$

$$q_{i,j} = 0, \ j \notin \mathcal{G}_i \ , \forall i, j \tag{19}$$

$$q_{i,j} \in [0, 1] \ , \forall i, j \tag{20}$$

$$t_i > 0 \ , \forall i \tag{21}$$

$$t_i (t_i - \alpha_j + \Lambda_j) + \Lambda_j \alpha_j (e^{\beta_j t_i} - 1) < 0 \tag{22}$$

$$\text{var} \quad \boldsymbol{q}, \boldsymbol{t},$$

where $\theta \in [0, 1]$ is a trade-off factor that determines the relative significance of mean latency and latency tail probability in the objective function. By changing θ from $\theta = 1$ to $\theta = 0$, the solution for (14) spans the solutions that minimize the mean latency to ones that minimize the tail latency probability. Note that constraint (16) gives the load intensity of server j. Constraint (17) gives the aggregate arrival rate Λ_j for each node for the given probabilistic scheduling probabilities $q_{i,j}$ and arrival rates λ_i. Constraints (18)-(20) guarantee that the scheduling probabilities are feasible. Also, Constraint (22) ensures that the moment generating function given in (12) exists. Note that the optimization over $\boldsymbol{q}$ helps decrease the overall latency which gives significant flexibility over choosing the lowest-queue servers for accessing the files. We further note that the optimization problem in (14) is non-convex as, for instance, Constraint (22) is non-convex in $(\boldsymbol{q}, \boldsymbol{t})$ jointly. In the next subsection, we will explain our proposed algorithm to solve this problem efficiently.

B. Proposed Algorithm

The optimization problem given in (14)-(22) is optimized over two set of variables: scheduling probabilities $\boldsymbol{q}$ and auxiliary parameters $\boldsymbol{t}$. Since the problem is non-convex, we propose an iterative algorithm to solve the problem. The proposed algorithm divides the problem into two subproblems (easy to handle) that optimize one variable while fixing the another. The two sub-problems are labeled as *(i) Access Optimization* optimizes $\boldsymbol{q}$ for given $\boldsymbol{t}$, *(ii) Auxiliary Variables Optimization* optimizes $\boldsymbol{t}$ for given $\boldsymbol{q}$. This algorithm is summarized as follows.

1) **Initialization:** Initialize $\boldsymbol{t}$ and $\boldsymbol{q}$ in the feasible set.
2) **While Objective Converge**
   a) Run Access Optimization using current values of $\boldsymbol{t}$ to get new values of $\boldsymbol{q}$.
   b) Run Auxiliary Variables Optimization using current values of $\boldsymbol{q}$ to get new values of $\boldsymbol{t}$.

We next describe the two sub-problems along with the proposed solutions for the sub-problems.

We first initialize $q_{i,j}$ and $t_i \ \forall \ i, j$ such that the choice is feasible for the problem. Then, we do alternating minimization over the two sub-problems defined above. Since each sub-problem converges (decreasing) and the overall problem is bounded from below, we have the following result.

Theorem 2. *The proposed algorithm converges to a local optimal solution.*

V. Evaluation and Implementation

A. Parameter Setup

To validate our proposed algorithm for joint mean-tail latency and evaluate its performance, we simulate our algorithm in a distributed storage system of $m = 12$ distributed nodes, $r = 1000$ files, all of size 200 MB and using $(7, 4)$. However, our model can be used for any given number of storage servers, any number of files, and for any erasure coding setting. We consider a shifted-exponential distribution for the

TABLE I: Storage Node Parameters Used in our Simulation (Shift $\beta_j = 10\, msec$, $\forall j$ and rate α in 1/s).

	Node 1	Node 2	Node 3	Node 4	Node 5	Node 6
α_j	18.23	24.06	11.88	17.06	20.19	23.91
	Node 7	**Node 8**	**Node 9**	**Node 10**	**Node 11**	**Node 12**
α_j	27.01	21.39	9.92	24.96	26.53	21.80

chunk service times as it has been shown in real system measurements on Tahoe and Amazon S3 servers [16], [18], [28]. The service time parameters α_j and β_j are shown in Table I. Unless otherwise explicitly stated, the arrival rate for the first 500 files is $0.002s^{-1}$ while for the next 500 files is set to be $0.003s^{-1}$. In order to initialize our algorithm, we use a random placement of each file on 7 out of the 12 servers. Further, we set $q_{i,j} = k/n$ on the placed servers with $t_i = 0.01$ $\forall i$ and $j \in \mathcal{G}_i$. However, these choices of $q_{i,j}$ and t_i may not be feasible. Thus, we modify the initialization to be closest norm feasible solution.

B. Comparisons

We compare our proposed approach with the following strategies:

- We compare our approach with that in [12], [13] where a bound for latency in erasure-coded storage is provided with the help of auxiliary variables at which each file is assigned an auxiliary variable to further tighten the bound. In [12], [13], authors simply replaced the maximum of m-random variables by their sum which made the bound very loose especially for large m (where m is the number of storage servers). As will be shown below in this section, the bound here performs much better than that in [12], [13]. Further, the proposed bound performs better than that in [29], as it has been shown that the bound in [12], [13] outperforms the bound in [29]. Thus, the bound here beats both of these bounds.
- PSP (*Projected Service-Rate Proportional* Access) Policy: The access probabilities $\boldsymbol{q}$ are assigned proportional to the service rates of the storage nodes, i.e., $q_{i,j} = k_i \frac{\mu_j}{\sum_j \mu_j}$, where $\mu_j = 1 \Big/ \left(\frac{1}{\alpha_j} + \beta_j\right)$. This policy assigns servers proportional to their service rates. These access probabilities are projected toward feasible region in (14) to ensure stability of the storage system. With these fixed access probabilities, the QoE metrics are optimized over the auxiliary variables $\mathbf{t}$.
- PEA (Projected Equal Access) Policy: In this strategy, we set $q_{i,j} = k/n$ on the placed servers with $t_i = 0.01$ $\forall i$ and $j \in \mathcal{G}_i$. We then modify the initialization of $\boldsymbol{q}$ to be closest norm feasible solution given above values of $\boldsymbol{t}$. Finally, an optimization over $\boldsymbol{t}$ is performed to the objective.

C. Mean Latency Optimization

Convergence of the Proposed Algorithm: Figure 2 shows the convergence of our proposed algorithm, which alternatively optimizes the mean latency over all files over the choice of the scheduling probabilities $\boldsymbol{q}$ and auxiliary variables $\boldsymbol{t}$. With 1000 files and $m = 12$, the mean latency converges to the optimal value within 300 iterations.

Effect of Arrival Rate: Figures 3 plots the effect of different arrival rates on the mean latency where we compare our proposed algorithm with three different policies. Here, the arrival rate of each file λ_i is varied from $0.2 \times \lambda_i$ to $1.2 \times \lambda_i$, where λ_i is the base arrival rate. We note that our proposed algorithm outperforms all these strategies for the QoE metric of mean latency. Thus, both access and file-based auxiliary variables of files are both important for the reduction of mean latency. We also note that uniformly accessing servers (PEA) and simple service-rate-based scheduling (PSP) are unable to optimize the request based on different factors like arrival rates, different latency weights, thus leading to much higher latency. As expected, the mean latency increases with arrival rates. However, at high arrival rates, we see significant reduction in mean latency for our proposed approach. For example, we see, at the highest arrival rate, approximately 25% reduction in weighted mean latency as compared to the proposed approach in [12], [13].

Effect of the Number of Files and their Weights: We next show the effect of varying the weights w_i's as well as varying the number of files on the weighted mean latency for our proposed approach. In Figure 4, we vary the number of files in each set from 200 files in the base case to 250 files, 300 files, and 350 files, as graphically captured in the figure. We observe that the mean latency increases with the number of files, as this leads consequently to higher arrival rates and, hence, increase in the overall system workload. Our optimization algorithm optimizes the access of the files as well as the auxiliary variables to keep weighted mean latency at lower levels. Note that the higher arrival rate (and hence higher weights in the objective function) group, i.e., File Set 3, has lower weighted mean latency which results in reducing the overall system latency. Finally, we note that our approach efficiently chooses the access probabilities $\boldsymbol{q}$ which helps in

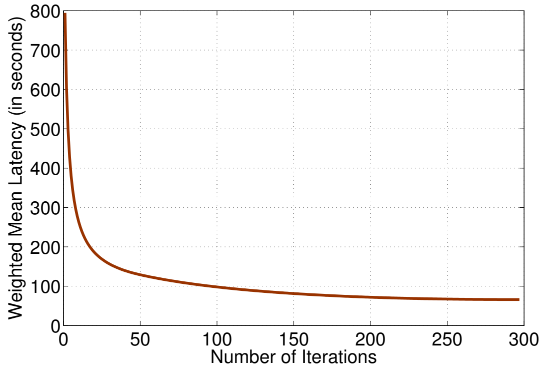

Fig. 2: Convergence of weighted mean latency.

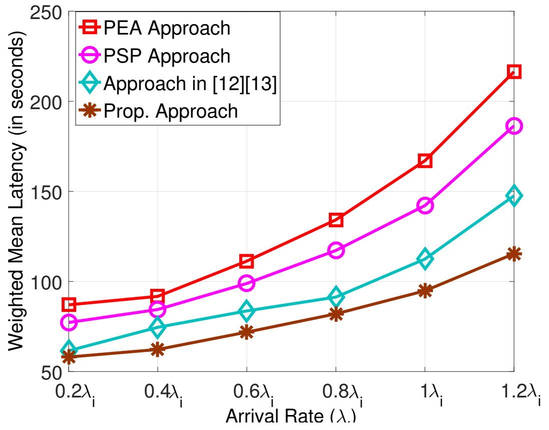

Fig. 3: Weighted mean latency for different file arrival rates. We vary the arrival rate of file i from $0.2 \times \lambda_i$ to $1.2 \times \lambda_i$, where λ_i is the base arrival rate.

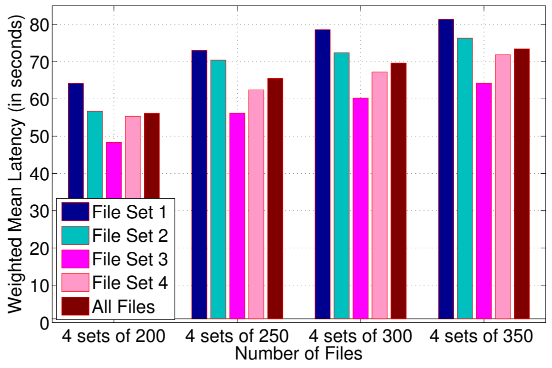

Fig. 4: Weighted mean latency for different number of files. We vary the number of files per set from 200 files to 350 files with an increment step of 50 files. For each set, the base arrival rate λ_i is scaled by 2, 3, 6, 4, respectively.

differentiating file latencies as compared to the lowest-queue based access strategy where minimum queue-length servers are selected to access the content obtaining lower weighted mean latency.

VI. Conclusions

In this paper, we have provided an analytical upper bound for the mean latency of erasure-coded storage systems. Then, we have formulated an optimization problem to optimize this QoE metric over the choice of the storage server access and bound auxiliary variables. Further, we have used an efficient alternating optimization algorithm to solve the problem efficiently with proven convergence. Our results have demonstrated a significant reduction in the weighted mean latency as compared to the state of the art algorithms.

References

[1] G. Ananthanarayanan, A. Ghodsi, S. Shenker, and I. Stoica, "Effective straggler mitigation: Attack of the clones." in *NSDI*, vol. 13, 2013, pp. 185–198.

[2] H. Weatherspoon and J. Kubiatowicz, "Erasure coding vs. replication: A quantitative comparison," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, ser. IPTPS '01. Springer-Verlag, 2002.

[3] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A survey on network codes for distributed storage," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 476–489, 2011.

[4] A. Fikes, "Storage architecture and challenges (talk at the google faculty summit)," http://bit.ly/nUylRW, Tech. Rep., 2010.

[5] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, "Xoring elephants: Novel erasure codes for big data," in *Proceedings of the 39th international conference on Very Large Data Bases.*, 2013.

[6] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin, "Erasure coding in windows azure storage," in *Proceedings of the 2012 USENIX Conference on Annual Technical Conference*, ser. USENIX ATC'12. USENIX Association, 2012.

[7] Y. Xiang, T. Lan, V. Aggarwal, and Y.-F. R. Chen, "Joint latency and cost optimization for erasure-coded data center storage," in *Proceedings of IFIP WG 7.3 Performance 2014*, 2014.

[8] G. Joshi, Y. Liu, and E. Soljanin, "On the delay-storage trade-off in content download from coded distributed storage systems," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 5, pp. 989–997, 2014.

[9] L. A. Barroso, "Warehouse-scale computing: Entering the teenage decade," 2011.

[10] G. Liang and U. Kozat, "Fast cloud:pushing the envelope on delay performance of cloud storage with coding," *IEEE/ACM Transactions on Networking*, pp. 124–137, 2013.

[11] T. M. queue: analyzing latency performance of codes and redundant requests, "Tahoe-lafs docs," arXiv:1211.5405, Tech. Rep., 2012.

[12] Y. Xiang, T. Lan, V. Aggarwal, and Y. F. R. Chen, "Joint latency and cost optimization for erasure-coded data center storage," *IEEE/ACM Transactions on Networking*, vol. 24, no. 4, pp. 2443–2457, Aug 2016.

[13] ——, "Joint latency and cost optimization for erasure-coded data center storage," *SIGMETRICS Perform. Eval. Rev.*, vol. 42, no. 2, pp. 3–14, Sep. 2014. [Online]. Available: http://doi.acm.org/10.1145/2667522.2667524

[14] L. Huang, S. Pawar, H. Zhang, and K. Ramchandran, "Codes can reduce queueing delay in data centers," in *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*. IEEE, 2012, pp. 2766–2770.

[15] K. Gardner, S. Zbarsky, M. Velednitsky, M. Harchol-Balter, and A. Scheller-Wolf, "Understanding response time in the redundancy-d system," *ACM SIGMETRICS Performance Evaluation Review*, vol. 44, no. 2, pp. 33–35, 2016.

[16] F. Baccelli, A. Makowski, and A. Shwartz, "The fork-join queue and related systems with synchronization constraints: stochastic ordering and computable bounds," *Advances in Applied Probability*, pp. 629–660, 1989.

[17] G. Liang and U. C. Kozat, "Tofec: Achieving optimal throughput-delay trade-off of cloud storage using erasure codes," in *INFOCOM, 2014 Proceedings IEEE*. IEEE, 2014, pp. 826–834.

[18] S. Chen, Y. Sun, U. Kozat, L. Huang, P. Sinha, G. Liang, X. Liu, and N. Shroff, "When queuing meets coding: Optimal-latency data retrieving scheme in storage clouds," in *Proceedings of IEEE Infocom*, 2014.

[19] A. Kumar, R. Tandon, and T. Clancy, "On the latency and energy efficiency of distributed storage systems," *IEEE Transactions on Cloud Computing*, 2015.

[20] W. Huajin, L. Jianhui, S. Zhihong, and Z. Yuanchun, "Approximations and bounds for (n, k) fork-join queues: A linear transformation approach," *CoRR*, vol. abs/1703.08337, 2017. [Online]. Available: https://arxiv.org/abs/1707.08860v2

[21] B. Li, A. Ramamoorthy, and R. Srikant, "Mean-field-analysis of coding versus replication in cloud storage systems," in *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on*. IEEE, 2016, pp. 1–9.

[22] Y. Sun, Z. Zheng, C. E. Koksal, K. H. Kim, and N. B. Shroff, "Provably delay efficient data retrieving in storage clouds," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, April 2015, pp. 585–593.

[23] Y. Sun, C. E. Koksal, and N. B. Shroff, "On delay-optimal scheduling in queueing systems with replications," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, Feb 2017, p. 78.

[24] G. Scutari, F. Facchinei, and L. Lampariello, "Parallel and distributed methods for constrained nonconvex optimization-part i: Theory." *IEEE Trans. Signal Processing*, vol. 65, no. 8, pp. 1929–1944, 2017.

[25] G. Scutari, F. Facchinei, L. Lampariello, and P. Song, "Parallel and distributed methods for nonconvex optimization- part i: Theory," *IEEE Trans. Signal Process*, 2014.

[26] M. Kuczma, *An introduction to the theory of functional equations and inequalities: Cauchy's equation and Jensen's inequality*. Springer Science & Business Media, 2009.

[27] A. Zwart and O. J. Boxma, "Sojourn time asymptotics in the m/g/1 processor sharing queue," *Queueing systems*, vol. 35, no. 1-4, pp. 141–166, 2000.

[28] A. S3, "Amazon simple storage service," http://aws.amazon.com/s3/, Tech. Rep., 2014.

[29] M. Bramson, Y. Lu, and B. Prabhakar, "Randomized load balancing with general service time distributions," in *ACM SIGMETRICS performance evaluation review*, vol. 38, no. 1. ACM, 2010, pp. 275–286.