2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing

Taming latency in data center networking with erasure coded files

Yu Xiang, Vaneet Aggarwal, Yih-Farn R. Chen, and Tian Lan

***Abstract*—This paper proposes an approach to minimize service latency in a data center network where erasure-coded files are stored on distributed disks/racks and access requests are scattered across the network. Due to limited bandwidth available at both top-of-the-rack and aggregation switches, network bandwidth must be apportioned among different intra- and inter-rack data flows in line with their traffic statistics. We formulate this problem as weighted queuing and employ a class of probabilistic request scheduling policies to derive a closed-form outer-bound of service latency for erasure-coded storage with arbitrary file access patterns and service time distributions. The result enables us to propose a joint latency optimization over three entangled "control knobs": the bandwidth allocation at top-of-the-rack and aggregation switches, the probabilities for scheduling file requests, and the placement of encoded file chunks, which affects data locality. The joint optimization is shown to be a mixed-integer problem. We develop an iterative algorithm which decouples and solves the joint optimization as three sub-problems, which are either convex or solvable via bipartite matching in polynomial time. The proposed algorithm is prototyped in an open-source, distributed file system, *Tahoe*, and evaluated on a cloud testbed with 16 separate physical hosts in an OpenStack cluster. Experiments validate our theoretical latency analysis and show significant latency reduction for diverse file access patterns. The results provide valuable insight on designing low-latency data center networks with erasure-coded storage.**

I. Introduction

Data center storage is growing at an exponential speed and customers are increasingly demanding more flexibility in the tradeoffs among the reliability level, performance, and storage cost. These trends have caused the storage service providers to start shifting from simple data replication schemes to more powerful erasure codes, which can provide the same or higher level of reliability at a significantly lower storage cost. As a result, the effect of coding on content retrieval latency in data-center storage system is drawing more attention these days. Google and Amazon have published that every 500 ms extra delay means a 1.2% user loss [1]. Data centers often consist of multiple racks and all the data transfers between racks go through an aggregation switch, while data transfers within a rack go through a Top-of-Rack (TOR) switch. With knowledge of the access patterns of different files, the use of erasure coding enables a novel latency optimization of data center storage with respect to placement of erasure coded content on different racks and bandwidth reservations at different switches, which can be optimized jointly to reduce latency.

Y. Xiang and T. Lan are with Department of ECE, George Washington University, DC 20052 (email: xy336699@gwmail.gwu.edu, tlan@gwu.edu). V. Aggarwal is with the School of IE, Purdue University, West Lafayette IN 47907. He was with AT&T Labs-Research, Bedminster, NJ 07921 when this research was performed (email: vaneet@purdue.edu). Y. R. Chen is with AT&T Labs-Research, Bedminster, NJ 07921 (email: chen@research.att.com).

Exact latency for erasure-coded storage system is an open problem. Recently, there has been a number of attempts at finding latency bounds for an erasure-coded storage system [13], [16], [14], [28], [23]. In this work, we will use a latency analysis that is based on probabilistic scheduling developed in [28]. To the best of our knowledge, this is the first analysis that accounts for multiple files and arbitrary file access patterns in quantifying service latency. Further, the analysis applies to general service time distributions by modeling the delay at each storage node via the latency of M/G/1 queue. Knowing the exact delay from each storage node, a tight upper bound on the average service latency could be found by extending ordered statistic analysis in [36].

We consider a data center storage system with a hierarchical structure in this work. Each rack has a TOR switch that is responsible for routing data flows between different disks and associated storage servers in the rack, while data transfers between different racks are managed by an aggregation switch that connects all TOR switches. Multiple client files are stored distributively using an (n, k) erasure coding scheme, which allows each file to be reconstructed from any k-out-of-n encoded chunks. We assume that file access requests may be generated from anywhere inside the data center, e.g., a virtual machine spun up by a client on any of the racks. Due to limited bandwidth available at both the TOR and aggregation switches, a simple First Come First Serve (FCFS) policy to schedule all file requests falls short on minimizing service latency, not only because of its inability to differentiate heterogeneous flows or adapt to varying traffic patterns, but also due to the entanglement of different file requests. More precisely, the latency of each file request is determined by the maximum delay in retrieving k-out-of-n encoded chunks. Without proper coordination in processing each batch of chunk requests that jointly reconstructs a file, service latency is dominated by staggering chunk requests with the worst access delay performance, significantly increasing overall latency in the data center. To avoid this, bandwidth reservation can be made for routing traffic among racks [24], [25], [26]. Thus, we apportion bandwidth among different pairs of racks so that each pairwise allocated bandwidth has its own FCFS queue for the data transfer between the corresponding pair of racks. We jointly optimize bandwidth allocation and data locality (i.e., placement of encoded file chunks) to achieve service latency minimization.

For a given set of pair-wise bandwidth allocations among racks, and placement of different erasure-coded files in dif-

978-1-4799-8006-2/15 $31.00 © 2015 IEEE
DOI 10.1109/CCGrid.2015.142

ferent racks, we first find an upper bound on average latency of each file request when accessed from a certain rack. Since each file is erasure coded with an (n, k) erasure code, and the file request needs k file chunks rather than the whole file, the upper bound uses probabilistic scheduling proposed in [28] to choose different subsets with certain probabilities. These probabilities (which model load-balancing in scheduling file requests) can then be optimized to give a tighter upper bound.

Having studied the upper bound of latency for each request, we consider a joint optimization of average service latency (weighted by the rate of arrival of requests) over the placement of contents of each file, the bandwidth reservation between any pair of racks, and the scheduling probabilities. Knowing the access pattern of each file from different racks is the key in having asymmetric bandwidth allocations and placements in line with the access pattern. To the best of our knowledge, this is the first latency analysis for an erasure-coded single data-center storage considering the impact of TOR and aggregation switches. The joint optimization problem optimizes over the content placement, bandwidth allocations between pair of racks, and the access probabilities from different racks for each request. This optimization is shown to be a mixed-integer optimization, especially due to the integer constraints for the content placement. In this work, the latency minimization is decoupled into 3 sub-problems, two of which are proven to be convex. We propose an algorithm which iteratively minimizes service latency over the three engineering degrees of freedom with guaranteed convergence.

To validate our theoretical analysis and joint latency optimization for different tenants, we provide a prototype of the proposed algorithms in *Tahoe* [37], which is an open-source, distributed file system based on the *zfec* erasure coding library for fault tolerance. A Tahoe storage system consisting of 10 racks is deployed on hosts of virtual machines in an OpenStack-based data center environment, with each rack hosting 8 storage servers running Tahoe ports. Each rack has a client node deployed to issue storage requests. The experimental results show that the proposed algorithm converges within a reasonable number of iterations. We further find that the service time distribution is nearly proportional to the bandwidth of the server, which is an assumption used in the latency analysis, and implementation results also show that our proposed approach significantly improved service latency in storage systems compared to native storage settings of the testbed.

II. System Model

We consider a data center consisting of N racks, denoted by $\mathcal{N} = \{i = 1, 2, \ldots, N\}$, each equipped with m homogeneous servers that are available for data storage and hosting application. There is a Top-of-Rack (TOR) switch at each rack to route intra-rack traffic and an aggregate switch in the data center that connects all TOR switches for routing inter-rack traffic. A set of files $\mathcal{R} = \{1, 2, \ldots, R\}$ are stored among these $m * N$ servers and are accessed by client applications running in the data center.

In this paper, we focus on file storage systems that employ erasure coding to achieve optimal space efficiency, while

Symbol	Meaning
N	Number of racks, indexed by $i = 1, \ldots, N$
m	Number of storage servers in each rack
R	Number of files in the system, indexed by $r = 1, \ldots, R$
(n, k)	Erasure code for storing files
B	Total available bandwidth at the aggregate switch
b	Total available bandwidth at each top-of-the-rack switch
$B^{\text{eff}}_{i,j}$	Effective bandwidth for servicing requests from rack i to rack j
$w_{i,j}$	Weight for apportioning aggregate switch bandwidth
λ^i_r	Arrival rate of request for file r from rack i
$\pi^r_{i,j}$	Probability of routing rack-i file-r request to rack j
S_r	Set of racks for placing encoded chunks of file r
$N_{i,j}$	Connection delay for service from rack i to rack j
$Q_{i,j}$	Queuing delay for service from rack i to rack j
$\bar{T}^i_r$	Expected latency of a request of file r from rack i

TABLE I: Main notation.

enabling sufficient redundancy for reliability. More precisely, each file r is partitioned into k fixed-size chunks and then encoded using an (n, k) erasure code to generate n chunks of equal size. The encoded chunks are assigned to and stored on n out of $m * N$ distinct servers in the data center. While it is possible to place multiple chunks in the same rack[1], we choose servers in n distinct racks to maximize the distribution of chunks across all racks, which achieves the highest reliability against rack failures. This is a common practice adopted by QFS[12], an erasure-coded storage file system that is desgined to replace HDFS for Map/Reduce processing. For each file r, we use $\mathcal{S}_r$ to denote the set of racks selected for placing its encoded chunks, satisfying $\mathcal{S}_r \subseteq \mathcal{N}$ and $|\mathcal{S}_r| = n$. To process file access requests, an (n, k) MDS erasure code allows the file to be reconstructed from any subset of k-out-of-n chunks. Therefore, a file request generated by a client application must be routed to a subset of k racks in $\mathcal{S}_r$. The selection of these k racks needs to be determined for each file request and it must take load balancing into account, so that the access latency is minimized. We refer to this routing and selection problem as the *request scheduling problem*.

There are series of requests generated by applications to access R files. We model the arrival of requests for each file r as an independent Poisson process. Let λ^i_r be the rate of file r requests that are generated by a client application running in rack i. We note that a file r request can be generated from a client application in any of the N racks with certain probabilities (= $\frac{\lambda^i_r}{\sum_i \lambda^i_r}$). The overall request arrival for file r is a composition of Poisson process with rate $\lambda_r = \sum_i \lambda^i_r$.

[1]Our results in this paper can be easily extended to the case where multiple chunks are placed on each rack using the techniques in [28].

Since solving the optimal request scheduling is still an open problem for erasure-coded storage [13], [16], [14], [28], we employ the *probabilistic scheduling policy* proposed in [28], which provides a practical solution to the request scheduling problem as well as an outer bound of service latency. Upon the arrival of each request, a probabilistic scheduler selects k-out-of-n racks in $\mathcal{S}_r$ hosting the file chunks according to some known probability and route the resulting traffic to the client application. It is shown that determining the probability distribution of each k-out-of-n combination is equivalent to solving the marginal probabilities for scheduling requests λ_r^i,

$$\pi_{i,j}^r = \mathbb{P}[j \in \mathcal{S}_i^r \text{ is selected} \mid k \text{ racks are selected}], \quad (1)$$

under constraints $\pi_{i,j}^r \in [0,1]$ and $\sum_j \pi_{i,j}^r = k$ [28]. Here $\pi_{i,j}^r \in [0,1]$ is the probability of routing a file r request from rack i to rack j. It is easy to see that $\pi_{i,j}^r = 0$ if no chunk of file r exists on rack j, i.e., $j \notin \mathcal{S}_i^r$. Further, constraint $\sum_j \pi_{i,j}^r = k$ is due to the fact that k distinct racks containing desired chunks are needed for file reconstruction.

To accommodate various cloud applications with different bandwidth requirements, we propose a weighted queuing model to apportion bandwidth available at the TOR and aggregate switches among different data flows. At each rack j, we buffer all incoming requests generated by applications in rack i in a local queue named $q(i,j)$. Therefore, each rack j manages N independent queues, which include 1 queue (i.e., $q(j,j)$) that manages intra-rack traffic traveling through the TOR switch and $N-1$ queues (i.e., $q(i,j)$) that manages inter-rack traffic to other racks $i \neq j$.

Assume the total bi-direction bandwidth at the aggregate switch is B, which is apportioned among the $N(N-1)$ queues for inter-rack traffic. Let $\{w_{i,j},\ \forall i \neq j\}$ be a set of $N(N-1)$ non-negative weights satisfying $\sum_{i,j:i\neq j} w_{i,j} = 1$. We assign to each queue $q(i,j)$ a share of B that is proportional to $\{w_{i,j}$, i.e., queue $q(i,j)$ receives a dedicated service bandwidth $B_{i,j}^{\text{eff}}$ on the aggregate switch, i.e.,

$$B_{i,j}^{\text{eff}} = B \cdot w_{i,j},\ \forall i \neq j. \quad (2)$$

According to our routing model, the same amount of bandwidth has to be reserved on the TOR switches of both racks i and j. Then, any remaining bandwidth on the TOR switches will be available for intra-rack traffic routing. On rack j, it is computed by the total TOR bandwidth b minus aggregate incoming and outgoing inter-rack traffic [2], i.e.,

$$B_{i,j}^{\text{eff}} = b - \sum_{k:k\neq i} w_{i,k}B - \sum_{k:k\neq i} w_{k,i}B,\ \forall i = j. \quad (3)$$

By optimizing $w_{i,j}$, the weighted queuing provides a fair allocation of data center bandwidth among different data flows both within and across racks. Bandwidth under-utilization and congestion can be mitigated because queues with heavy workload will receive more bandwidth and those with light workload will get less.

Under the probabilistic scheduling policy, it is easy to see that requests for file r chunk from rack i to rack j form a (decomposed) Poisson process with rate $\lambda_r^i \pi_{i,j}$. Thus, the aggregate arrival of requests from rack i to rack j becomes a (superpositioned) Poisson process with rate

$$\Lambda_{i,j} = \sum_r \lambda_r^i \pi_{i,j}^r. \quad (4)$$

It implies that the system can be modeled as N^2 M/G/1 queues, where service time per chunk is determined by the allocation of bandwidth $B \cdot w_{i,j}$ to each queue $q(i,j)$ handling inter-rack traffic or available bandwidth b_j to $q(j,j)$ handling intra-rack traffic. The service latency for each data flow can be computed using ordered statistics analysis in [28]. Our goal in this work is to quantify service latency under this model and to minimize average service latency for all traffic in the data center by solving an optimization problem over three dimensions: placement of encoded file chunks $\mathcal{S}_r$, scheduling probabilities $\pi_{i,j}$ for load-balancing, and allocation of system bandwidth through weights $w_{i,j}$.

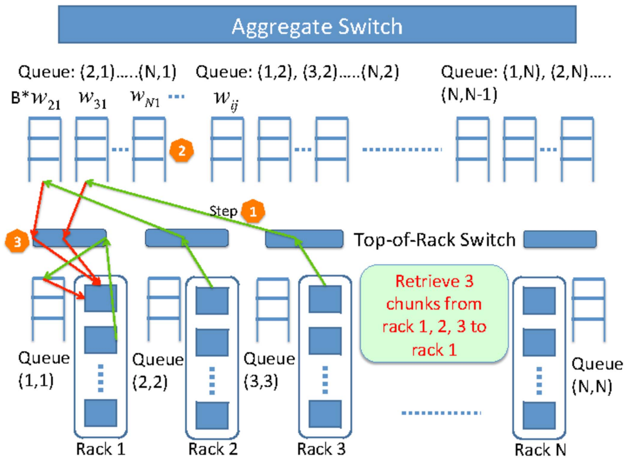

Fig. 1: System model

Fig. 1 shows our system model with an example file request generated by rack 1. Due to $(5,3)$ erasure coding of files, 3 chunks from rack 1, rack 2 and rack 3 are retrieved respectively. Inter-rack data flows from rack 2 and rack 3 travel through the TOR switches and then are routed via the aggregate switch, which assigns the requests to 2 weighted queues with bandwidth $B_{2,1}^{\text{eff}} = B \cdot w_{2,1}$, $B_{3,1}^{\text{eff}} = B \cdot w_{3,1}$. The data flows are received by the TOR switch at rack 1 and then go to the destination server. On the other hand, intra-rack data flow is routed via the TOR switch of rack 1 only with available bandwidth $B_{1,1}^{\text{eff}} = b - \sum_{j:j\neq 1} w_{1,j}B - \sum_{j:j\neq 1} w_{j,1}B$.

III. Analyzing Service Latency for Data Requests

In this section, we will derive an outer bound on the service latency of a file for each client application running in the data center. For each chunk request, we consider two latency components: *connection delay* $\mathbf{N}_{i,j}$ that includes the overhead to set up the network connection between client application in rack i and storage server in rack j and *queuing delay* $\mathbf{Q}_{i,j}$ that measures the time a chunk request experiences at the

[2]all the analysis and algorithm can be trivially modified depending on whether the TOR switch is non-blocking and/or duplex

bandwidth service queue $q(i,j)$, i.e., the time to transfer a chunk with the allocated network bandwidth. Let $\bar{T}_i^r$ be the average service latency for accessing file r by an application in rack i. Due to the use of (n,k) erasure coding for distributed storage, each file request is mapped to k parallel chunk requests. The chunk requests are scheduled to k different racks $\mathcal{A}_i^r \subset \mathcal{S}_r$ storing the desired file chunks. A file is reconstructed if all $k = |\mathcal{A}_i^r|$ chunks are successfully retrieved, which takes on average

$$\bar{T}_r^i = \mathbb{E}_{\mathcal{A}_i^r}[\max_{j\in\mathcal{A}_i^r}(\mathbf{N}_{i,j} + \mathbf{Q}_{i,j})], \tag{5}$$

where the expectation is taken over set $\mathcal{A}_i^r$ that are randomly selected with respect to known probabilities $\pi_{i,j}^r$ for all j under the probabilistic scheduling policy.

According to (5), average latency $\bar{T}_r^i$ is given by the highest order statistic of random delay $\mathbf{N}_{i,j} + \mathbf{Q}_{i,j}$ measured at k randomly selected racks $\mathcal{A}_i^r$. Without relying on any service time assumptions, we can use the method developed in [28] to obtain a closed-form upper bound of average latency using first and second moments of $\mathbf{N}_{i,j} + \mathbf{Q}_{i,j}$. The result is summarized in Lemma 1.

Lemma 1: (*Bound for the random order statistic* [28].) For arbitrary $z \in \mathbb{R}$, the average service time is bounded by

$$\bar{T}_r^i \le \Bigg\{ z + \sum_{j\in\mathcal{S}_r} \frac{\pi_{i,j}^r}{2}\left(\mathbb{E}[\mathbf{D}_{i,j}] - z\right) + \sum_{j\in\mathcal{S}_r} \frac{\pi_{i,j}^r}{2}\left[\sqrt{(\mathbb{E}[\mathbf{D}_{i,j}] - z)^2 + \mathrm{Var}[\mathbf{D}_{i,j}]}\right]\Bigg\}, \tag{6}$$

where $\mathbf{D}_{i,j} = \mathbf{N}_{i,j} + \mathbf{Q}_{i,j}$ is the combined delay. The tightest bound that is obtained by optimal $z \in \mathbb{Z}$ is tight in the sense that there exists a distribution of $\mathbf{D}_{i,j}$ to achieve the bound with exact equality.

We assume that connection delay $\mathbf{N}_{i,j}$ is independent of queuing delay $\mathbf{Q}_{i,j}$. With the proposed system model, the chunk requests arriving at each queue $q_{i,j}$ form a Poison process with rate $\Lambda_{i,j} = \sum_r \lambda_r^i \pi_{i,j}^r$. Therefore, each queue $q(i,j)$ can be modeled as a M/G/1 queue that processes chunk requests in an FCFS manner. Due to the fixed chunk size in our system, we denote $\mathbf{X}$ as the standard (random) service time per chunk when bandwidth B is available. We assume that the service time is inversely proportional to the bandwidth allocated to $q(i,j)$, i.e., $B_{i,j}^{\text{eff}}$. We obtain the distribution of actual service time $\mathbf{X}_{i,j}$:

$$\mathbf{X}_{i,j} \sim \mathbf{X} \cdot B / B_{ij}^{\text{eff}}, \ \forall i,j \tag{7}$$

With the service time distributions above, we can derive the mean and variance of queueing delay $\mathbf{Q}_{i,j}$ using Pollaczek-Khinchine formula. Let $\mu = \mathbb{E}[\mathbf{X}]$, $\sigma^2 = \mathrm{Var}[\mathbf{X}]$, and $\Gamma_t = \mathbb{E}[\mathbf{X}^t]$, be the mean, variance, t^{th} order moment of $\mathbf{X}$, $\eta_{i,j}$ and $\xi_{i,j}^2$ are mean and variance for connection delay $\mathbf{N}_{i,j}$. These statistics can be readily available from existing work on network delay [32], [10] and file-size distribution [34], [33].

Lemma 2: The mean and variance of combined delay $\mathbf{D}_{i,j}$ for any i,j is given by

$$\mathbb{E}[\mathbf{D}_{i,j}] = \eta_{i,j} + \frac{\Lambda_{i,j}\Gamma_2 B^2}{2B_{i,j}^{\text{eff}}(B_{i,j}^{\text{eff}} - \Lambda_{i,j}\mu B)} \tag{8}$$

$$\mathrm{Var}[\mathbf{D}_{i,j}] = \xi_{i,j}^2 + \frac{\Lambda_{i,j}\Gamma_3 B^3}{3(B_{i,j}^{\text{eff}})^2(B_{i,j}^{\text{eff}} - \Lambda_{i,j}\mu B)} + \frac{\Lambda_{i,j}\Gamma_2^2 B^4}{4(B_{i,j}^{\text{eff}})^2(B_{i,j}^{\text{eff}} - \Lambda_{i,j}\mu B)^2} \tag{9}$$

where $B_{i,j}^{\text{eff}}$ is the effective bandwidth assigned to the queue.

Proof: Consider an M/G/1 queue $q(i,j)$ with effective bandwidth $B_{i,j}^{\text{eff}}$. Under our system model, its service time per chunk has distribution $\mathbf{X}_{i,j} \sim \mathbf{X} \cdot B/B_{i,j}^{\text{eff}}$. Applying Pollaczek-Khinchine formula [36], we can obtain that:

$$\mathbb{E}[\mathbf{Q}_{i,j}] = \frac{\Lambda_{i,j}\mathbb{E}[\mathbf{X}_{i,j}^2]}{2(1 - \Lambda_{i,j}\mathbb{E}[\mathbf{X}_{i,j}])}, \tag{10}$$

and similarly,

$$\mathrm{Var}[\mathbf{Q}_{i,j}] = \frac{\Lambda_{i,j}\mathbb{E}[\mathbf{X}_{i,j}^3]}{3(1 - \Lambda_{i,j}\mathbb{E}[\mathbf{X}_{i,j}])} + \frac{\Lambda_{i,j}(\mathbb{E}[\mathbf{X}_{i,j}^2])^2}{4(1 - \Lambda_{i,j}\mathbb{E}[\mathbf{X}_{i,j}])^2}, \tag{11}$$

where $\Lambda_{i,j} = \sum_r \lambda_r^i \pi_{i,j}^r$ is the total arrival rate of chunk requests from rack i to rack j.

Finally, using our service time model and bandwidth allocation, we recognize that $\mathbb{E}[(\mathbf{X}_{i,j})^t] = \mathbb{E}[\mathbf{X}^t] \cdot (B/B_{i,j}^{\text{eff}})^t$. Plugging this into (10) and (11), we get the mean and variance for both cases $i \neq j$ and $i = j$. Let $\eta_{i,j} = \mathbb{E}[\mathbf{N}_{i,j}]$ and $\xi_{i,j}^2 = \mathrm{Var}[\mathbf{N}_{i,j}]$ be the mean and variance of connection delay. Since $N_{i,j}$ and $Q_{i,j}$ are independent, we obtain the mean and variance of delay of queue $q(i,j)$ as desired. $\square$

Combining these results, we derive an upper bound for average service latency $\bar{T}_i^r$ as a function of chunk placement $\mathcal{S}_r$, scheduling probability $\pi_{i,j}^r$, and bandwidth allocation $B_{i,j}^{\text{eff}}$ (which is a function of the bandwidth weights $w_{i,j}$ in (2) and (3)). The main result of our latency analysis is summarized in the following theorem.

Theorem 1: For arbitrary $z \in \mathbb{R}$, the expected latency $\bar{T}_r^i$ of a request of file r, requested from rack i is upper bounded by

$$\bar{T}_r^i \le z + \sum_{j\in\mathcal{S}_r}\left[\frac{\pi_{i,j}^r}{2} \cdot f(z, \Lambda_{i,j}, B_{i,j}^{\text{eff}})\right], \tag{12}$$

where function $f(z, \Lambda_{i,j}, B_{i,j}^{\text{eff}})$ is an auxiliary function depending on aggregate rate $\Lambda_{i,j}$ and effective bandwidth $B_{i,j}^{\text{eff}}$ of queue $q(i,j)$, i.e.,

$$f(z, \Lambda_{i,j}, B_{i,j}^{\text{eff}}) = H_{i,j} + \sqrt{H_{i,j}^2 + G_{i,j}} \tag{13}$$

$$H_{i,j} = \eta_{i,j} + \frac{\Lambda_{i,j}\Gamma_2 B^2}{2B_{i,j}^{\text{eff}}(B_{i,j}^{\text{eff}} - \Lambda_{i,j}\mu B)} - z \tag{14}$$

$$G_{i,j} = \xi_{i,j}^2 + \frac{\Lambda_{i,j}\Gamma_3 B^3}{3(B_{i,j}^{\text{eff}})^2(B_{i,j}^{\text{eff}} - \Lambda_{i,j}\mu B)} + \frac{\Lambda_{i,j}\Gamma_2^2 B^4}{4(B_{i,j}^{\text{eff}})^2(B_{i,j}^{\text{eff}} - \Lambda_{i,j}\mu B)^2} \tag{15}$$

IV. Joint Latency Minimization in Cloud

We consider a joint latency minimization problem over 3 design degrees of freedom in managing datacenter traffic: (i) placement of encoded file chunks $\{\mathcal{S}_r\}$ that determines datacenter traffic locality, (ii) allocation of bandwidth at aggregate/TOR switches through weights $\{w_{i,j}\}$ that affect chunk service time for different data flows, and (iii) scheduling probabilities $\{\pi^r_{i,j}\}$ for file retrievals that optimize load-balancing under probabilistic scheduling policy. Let $\lambda_{\text{all}} = \sum_i \sum_r \lambda^i_r$ be the total file request rate in the datacenter. The optimization objective is to minimize the upper bound on the average service latency, which is defined by

$$\begin{aligned}
&\sum_{r=1}^{R}\sum_{i=1}^{N}\frac{\lambda^i_r}{\lambda_{\text{all}}}\left[z+\sum_{j\in S_r}\frac{\pi^r_{i,j}}{2}\cdot f(z,\Lambda_{i,j},B^{\text{eff}}_{i,j})\right]\\
&= z+\sum_{r=1}^{R}\sum_{i=1}^{N}\sum_{j\in S_r}\frac{\lambda^i_r\pi^r_{i,j}}{2\lambda_{\text{all}}}f(z,\Lambda_{i,j},B^{\text{eff}}_{i,j})\\
&= z+\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{r=1}^{R}\frac{\lambda^i_r\pi^r_{i,j}}{2\lambda_{\text{all}}}f(z,\Lambda_{i,j},B^{\text{eff}}_{i,j})\\
&= z+\sum_{i=1}^{N}\sum_{j=1}^{N}\frac{\Lambda_{i,j}}{2\lambda_{\text{all}}}f(z,\Lambda_{i,j},B^{\text{eff}}_{i,j}). \qquad (16)
\end{aligned}$$

The second step follows from $\sum_r\sum_i \lambda^i_r = \lambda_{\text{all}}$. In the third step, we use the condition that $\pi^r_{i,j} = 0$ for all $j \notin S_r$ to extend the limit of summation (because a rack not hosting a desired file chunk should never be scheduled) and exchange the order. The last step is due to $\sum_r \lambda^i_r\pi^r_{i,j} = \Lambda_{i,j}$.

We now define the Joint Latency and Weights Optimization (JLWO) problem as follows:

$$\begin{aligned}
\min\quad & z+\sum_{i=1}^{N}\sum_{j=1}^{N}\frac{\Lambda_{i,j}}{2\lambda_{\text{all}}}f(z,\Lambda_{i,j},B^{\text{eff}}_{i,j}) && (17)\\
\text{s.t.}\quad & \Lambda_{i,j}=\sum_{r=1}^{R}\lambda^i_r\pi^r_{i,j}\le\mu_{i,j}\frac{B^{\text{eff}}_{i,j}}{B},\forall i,j && (18)\\
& \sum_{j=1}^{N}\pi^r_{i,j}=k \text{ and } \pi^r_{i,j}\in[0,1],\ \forall i,j,r && (19)\\
& |\mathcal{S}_r|=n \text{ and } \pi^r_{i,j}=0\ \forall j\notin\mathcal{S}_i,\ \forall i,r && (20)\\
& \sum_{i=1}^{N}\sum_{j\neq i}w_{i,j}=1. && (21)\\
& B^{\text{eff}}_{i,j}=w_{i,j}B,\ \forall i\neq j && (22)\\
& B^{\text{eff}}_{i,j}=b-\sum_{i:i\neq j}w_{i,j}B-\sum_{i:i\neq j}w_{j,i}B,\ \forall i=j && (23)\\
\text{var.}\quad & z,\ \{\mathcal{S}^r_i\},\ \{\pi^r_{i,j}\},\ \{w_{i,j}\}.
\end{aligned}$$

Here we minimize service latency bound in Section III over $z \in \mathbb{R}$ to get a tighter upper bound. Feasibility of Problem JLWO is ensured by (18), which requires arrival rate to be no greater than chunk service rate received by each queue. Encoded chunks of each file are placed on a set $\mathcal{S}_i$ of servers in (20), and each file request is mapped to k chunk requests and processed by k servers in $\mathcal{S}_i$ with probabilities $\pi^r_{i,j}$ in (19). Finally, weights $w_{i,j}$ should add up to 1 so there is no bandwidth left unutilized. Bandwidth assigned to each queue $B^{\text{eff}}_{i,j}$ is determined by our allocaiton policy in (2) and (3).

Problem JLWO is a mixed-integer optimization and hard to compute in general. In this work, we develop an iterative optimization algorithm that alternates among the 3 optimization dimension of problem JLWO and solves each sub-problem repeatedly to generate a sequence of monotonically decreasing objective values. To introduce the proposed algorithm, we first recognize that Problem JLWO is convex in either $\{\pi^r_{i,j}\}$ or $\{w_{i,j}\}$ when all other variables are fixed, respectively.

Lemma 3: (*Convexity of the scheduling sub-problem* [28].) When $\{z, w_{i,j}, S_r\}$ are fixed, Problem JLWO is a convex optimization over probabilities $\{\pi^r_{i,j}\}$.

Proof: The proof is straightforward due to the convexity of $\Lambda_{i,j}f(z,\Lambda_{i,j},B^{\text{eff}}_{i,j})$ over $\Lambda_{i,j}$ (which is a linear combination of $\{\pi^r_{i,j}\}$) as shown in [28], and the fact that all constraints are linear with respect to $\pi^r_{i,j}$. □

Lemma 4: (*Convexity of the bandwidth allocation sub-problem.*) When $\{z, \pi^r_{i,j}, S_r\}$ are fixed, Problem JLWO is a convex optimization over weights $\{w_{i,j}\}$.

Proof: Since all constraints in Problem JLWO are linear with respect to weights $\{w_{i,j}\}$, we only need to show that the optimization objective $f(z,\Lambda_{i,j},B^{\text{eff}}_{i,j})$ is convex in $\{w_{i,j}\}$ with other variables fixed. Notice that effective bandwidth $B^{\text{eff}}_{i,j})$ is a linear function of the bandwidth allocation weights $\{w_{i,j}\}$ for both inter-rack traffic queues (22) and intra-rack traffic queues (23). Therefore, $f(z,\Lambda_{i,j},B^{\text{eff}}_{i,j})$ is convex in $\{w_{i,j}\}$ if it is convex in $B^{\text{eff}}_{i,j})$.

Toward this end, we consider $f(z,\Lambda_{i,j},B^{\text{eff}}_{i,j}) = H_{i,j} + \sqrt{H^2_{i,j}+G_{i,j}}$ given in (13), (14) and (15). We find the second order derivatives of $H_{i,j}$ with respect to $B^{\text{eff}}_{i,j})$:

$$\frac{\partial^2 H_{i,j}}{dw^2_{i,j}} = \frac{\Lambda_{i,j}\Gamma_2(3w^2_{i,j}-3\Lambda_{i,j}\mu w_{i,j}-1)}{w^3_{i,j}(w_{i,j}-\Lambda_{i,j}\mu)^3} \qquad (24)$$

which is positive as long as $1-\Lambda_{i,j}\mu/w_{i,j} > 0$. This is indeed true because $\rho = \Lambda_{i,j}\mu/w_{i,j} < 1$ in M/G/1 queues. Thus, $H_{i,j}$ is convex in $B^{\text{eff}}_{i,j})$. Next, considering $G_{i,j}$ we have

$$\frac{\partial^2 G_{i,j}}{dw^2_{i,j}} = \frac{pw^3_{i,j}+qw^2_{i,j}+sw_{i,j}+t}{6w^4_{i,j}(w_{i,j}-\Lambda_{i,j}\mu)^4} \qquad (25)$$

where the auxiliary variables are given by where we have:

$$p = 24\Lambda_{i,j}\Gamma_3$$

$$q = 2\Lambda_{i,j}(15\Gamma^2_2 - 28\Lambda_{i,j}\mu\Gamma_3)$$

$$s = 2\Lambda^2_{i,j}\mu(22\Lambda_{i,j}\mu\Gamma_3 - 15\Gamma^2_2)$$

$$t = 3\Lambda^3_{i,j}\mu^2(3\Gamma^2_2 - 4\Lambda_{i,j}\mu\Gamma_3)$$

which give out the solution for $pw^3_{i,j}+qw^2_{i,j}+sw_{i,j}+t$ as $w_{i,j} > \Lambda_{i,j}\mu$, which is equivalent to $1-\Lambda_{i,j}\mu/w_{i,j} > 0$, which has been approved earlier. Thus $G_{i,j}$ is also convex in $B^{\text{eff}}_{i,j})$.

Finally, to prove that $f = H_{i,j} + \sqrt{H_{i,j}^2 + G_{i,j}}$ is convex in $B_{i,j}^{\text{eff}}$), we notice that f is convex in $H_{i,j}$ and $G_{i,j}$ because its Hessian matrix is positive semi-definite, i.e.,

$$\nabla^2 f = \frac{1}{2\left(H_{i,j}^2 + G_{i,j}\right)^{\frac{3}{2}}} \cdot \begin{bmatrix} H_{i,j}^2 & H_{i,j} \\ H_{i,j} & 1 \end{bmatrix} \succeq \mathbf{0}.$$

Since f is increasing and convex in $H_{i,j}$ and $G_{i,j}$, and $H_{i,j}$ and $G_{i,j}$ are both convex in $B_{i,j}^{\text{eff}}$), we conclude that their composition $f(z, \Lambda_{i,j}, B_{i,j}^{\text{eff}})$ is also convex in $B_{i,j}^{\text{eff}}$). This completes the proof. $\square$

Next, we consider the placement sub-problem that minimizes average latency over $\{S_r\}$ for fixed $\{z, \pi_{i,j}^r, w_{i,j}\}$. In this problem, for each file r we permute the set of racks that contain each file r to have a new placement $S'_r = \{\beta(j).\ \forall j \in S_r\}$ where $\beta(j) \in \mathcal{N}$ is a permutation. The new probability of accessing file r from rack $\beta(j)$ when client is at rack i becomes $\pi_{i,\beta j}^r$. Our objective is to find such a permutation that minimizes the average service latency, which can be solved via a matching problem between the set of scheduling probabilities $\{\pi_{i,j}^r, \forall i\}$ and racks, with respect to their load excluding the contribution of file r. Let $\Lambda_{i,j}^{-r} = \Lambda_{i,j} - \lambda_r^i \pi_{i,j}^r$ be the total request rate between racks i and j excluding the contribution of file r. We define a complete bipartite graph $\mathcal{G}_r = (\mathcal{U}, \mathcal{V}, \mathcal{E})$ with disjoint vertex sets $\mathcal{U}, \mathcal{V}$ of equal size N and edge weights given by

$$D_{jk} = \sum_{i=1}^{N} \frac{\Lambda_{i,j}^{-r} + \lambda_r^i \pi_{ik}^r}{\lambda_{\text{all}}} f(z, \Lambda_{i,j}^{-r} + \lambda_r^i \pi_{ik}^r, w_{i,j}),\ \forall j,k. \quad (26)$$

It is easy to see that a minimum-weight matching on $\mathcal{G}_r$ finds $\beta(j)\ \forall j$ to minimize

$$\sum_{j=1}^{N} D_{j\beta(j)} =$$
$$\sum_{j=1}^{N}\sum_{i=1}^{N} \frac{\Lambda_{i,j}^{-r} + \lambda_r^i \pi_{i,\beta(j)}^r}{\lambda_{\text{all}}} f(z, \Lambda_{i,j}^{-r} + \lambda_r^i \pi_{i,\beta(j)}^r, w_{i,j}),$$

which is exactly the optimization objective of Problem JLWO if a chunk request is scheduled with probability $\pi_{i,\beta(j)}^r$ to a rack with existing load $\Lambda_{i,j}^{-r}$.

Lemma 5: (*Bipartite matching equivalence of the placement sub-problem.*) When $\{z, \pi_{i,j}^r, w_{i,j}\}$ are fixed, the optimization of Problem JLWO over placement variables S_r is equivalent to a balanced Bipartite matching problem of size N.

Our proposed algorithm that solves the 3 sub-problems interactively is summarized in Algorithm JLMO. It generates a sequence of monotonically decreasing objective values and therefore is guaranteed to converge. Notice that scheduling and bandwidth allocation sub-problems as well as the minimization over z are convex and can be efficiently computed by any off-the-shelf convex solvers, e.g., MOSEK [29]. The placement sub-problem is a balanced bipartite matching that can be solved by Hungarian algorithm [30] in polynomial time.

Theorem 2: The proposed algorithm generates a sequence of monotonically decreasing objective values and is guaranteed to converge to a fixed point of Problem JLMO.

```
Algorithm JLWO :

Initialize $t = 0$, $\epsilon > 0$.
Initialize feasible $\{z(0), \pi_{i,j}^r(0), S_r(0)\}$.
while $O(t) - O(t-1) > \epsilon$
    // Solve bandwidth allocation for given $\{z(t), \pi_{i,j}^r(t), S_r(t)\}$
    $w_{i,j}(t+1) = \arg\min_{w_{i,j}}$ (17) s.t. (18), (21), (22), (23).
    // Solve scheduling for given $\{z(t), S_r(t), w_{i,j}(t+1)\}$
    $\pi_{i,j}(t+1) = \arg\min_{\pi_{i,j}^r}$ (17) s.t. (18), (19).
    // Solve placement for given $\{z(t), w_{i,j}(t+1), \pi_{i,j}^r(t+1)\}$
    for $r = 1, \ldots, R$
        Calculate $\Lambda_{i,j}^{-r}$ using $\{\pi_{i,j}^r(t+1)\}$.
        Calculate $D_{jk}$ from (26).
        $(\beta(j)\forall j \in \mathcal{N})$=$Hungarian\_Algorithm(\{D_{jk}\})$.
        Update $\pi_{i,\beta(j)}^r(t+1) = \pi_{i,j}^r(t)\ \forall i,j$.
        Initialize $S_r(t+1) = \{\}$.
        for $j = 1, \ldots, N$
            if $\exists i$ s.t. $\pi_{i,j}^r(t+1) > 0$
                Update $S_t(t+1) = S_r(t+1) \cup \{j\}$.
            end if
        end for
    end for
    // Update bound for given $\{w_{i,j}(t+1), \pi_{i,j}^r(t+1), S_r(t+1)\}$
    $z(t+1) = \arg\min_{z\in\mathbb{R}}.$ (17).
    Update objective value $B^{(t+1)}$=(17).
    Update $t = t + 1$.
end while
Output: $\{\mathcal{S}_r(t), \pi_{i,j}r^{(t)}, w_{i,j}(t)\}$
```

V. Implementation and Evaluation

A. Tahoe Testbed

To validate the queuing model in our single data-center system model and evaluate the performance, we implemented the algorithms in *Tahoe* [37], which is an open-source, distributed file-system based on the *zfec* erasure coding library. It provides three special instances of a generic *node*: (a) *Tahoe Introducer*: it keeps track of a collection of storage servers and clients and introduces them to each other. (b) *Tahoe Storage Server*: it exposes attached storage to external clients and stores erasure-coded shares. (c) *Tahoe Client*: it processes upload/download requests and connects to storage servers through a Web-based REST API and the Tahoe-LAFS (Least-Authority File System) storage protocol over SSL.

Our experiment is done on a Tahoe testbed that consists of 16 separate physical hosts in an Openstack cluster, 10 out of which have been used by our experiment. We simulated each host as a rack in the cluster. Each host has 4 VM instances, and each instance runs 2 Tahoe service ports, i.e., simulated as Tahoe storage servers in the racks. We effectively simulated an Openstack cluster with 10 racks and 8 Tahoe storage servers on each rack. The cluster uses a Cisco Catalyst 4948 switch, which has 48 ports. Each port supports non-blocking, 1Gbps bandwidth in the full duplex mode. The weighted-queuing model is supposed to have $N(N-1) = 90$ queues for inter-rack traffic at the aggregate switch; however, bandwidth reservation through ports of the switch is not possible since the Cisco switch does not support the OpenFlow protocol, so we made pairwise bandwidth reservations (using a bandwidth control tool from our Cloud QoS platform) between different

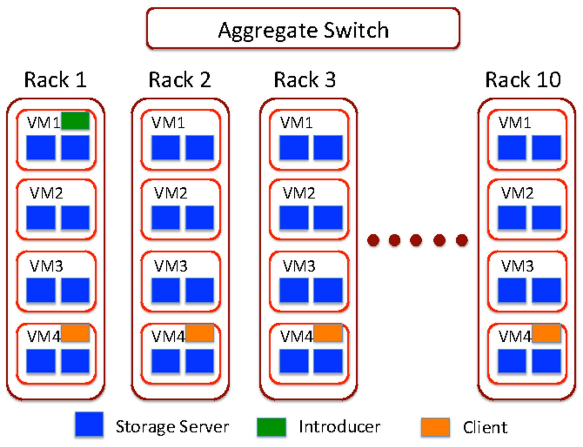

Fig. 2: Our Tahoe testbed with ten racks and each has 8 Tahoe storage servers

Tahoe Clients and Tahoe storage servers. The Tahoe introducer node resides on rack 1 and each rack has a client node with multiple tahoe ports to simulate multiple clients to initiate file requests coming from rack i. Our Tahoe testbed is shown in Fig 2.

Tahoe is an erasure-coded distributed storage system with some unique properties that make it suitable for storage system experiments. In Tahoe, each file is encrypted, and is then broken into a set of segments, where each segment consists of k blocks. Each segment is then erasure-coded to produce n blocks (using an (n, k) encoding scheme) and then distributed to (ideally) n storage servers regardless of their server placement, whether in the same rack or not. The set of blocks on each storage server constitute a chunk. Thus, the file equivalently consists of k chunks which are encoded into n chunks and each chunk consists of multiple blocks[3]. For chunk placement, the Tahoe client randomly selects a set of available storage servers with enough storage space to store n chunks. For server selection during file retrievals, the client first asks all known servers for the storage chunks they might have, again regardless of which racks the servers reside in. Once it knows where to find the needed k chunks (from among the fastest servers with a pseudo-random algorithm), it downloads at least the first segment from those servers. This means that it tends to download chunks from the "fastest" servers purely based on round-trip times (RTT). However, we consider RTT plus expected queuing delay and transfer delay as a measure of latency.

We had to make several changes in Tahoe in order to conduct our experiments. First, we need to have the number of racks $N \geq n$ in order to meet the system requirement that each rack can have at most one chunk of the original file. In addition, since Tahoe has its own rules for chunk placement and request scheduling, while our experiment requires client-defined chunk placement in different racks, and also with our server selection algorithms in order to minimize joint latency, we modified the upload and download modules in the Tahoe storage server and client to allow for customized and explicit server selection for both chunk placement and retrieval, which is specified in the configuration file that is read by the client when it starts. Finally, Tahoe performance suffers from its single-threaded design on the client side; we had to use multiple clients (multiple threads on one client node) in each rack with separate ports to improve parallelism and bandwidth usage during our experiments.

[3]If there are not enough servers, Tahoe will store multiple chunks on one sever. Also, the term "chunk" we used in this paper is equivalent to the term "share" in Tahoe terminology. The number of blocks in each chunk is equivalent to the number of segments in each file.

B. Basic Experiment Setup

We use (7,4) erasure code in the Tahoe testbed we introduced above throughout the experiments described in the implementation section. The algorithm first calculates the optimal chunk placement through different racks, which will be set up in the client configuration file for each write request. File retrieval request scheduling and weight assignment decisions for inter-rack traffic also comes from Algorithm JLWO. The system calls a bandwidth reservation tool to reserve the assigned bandwidth $Bw_{i,j}$ based on optimal weights of each inter-rack pair, where total bandwidth capacity at the aggregate switch is 96 Gbps, 48 ports with 1Gbps in each direction since we are simulating host machines as racks and VM's as storage servers. Intra-rack bandwidth as measured from *iPerf* measurement is 706Mbps, disk read bandwidth for sequential workload is 386 Mbps, and write bandwidth is 118 Mbps. Requests are generated based on arrival rates at each rack and submited from client nodes at all racks.

C. Experiments and Evaluation

Convergence of Algorithm. We implemented Algorithm JLWO using MOSEK, a commercial optimization solver. With 10 racks and 8 simulated distributed storage servers on each rack, there are a total of 80 Tahoe servers in our testbed. Figure 3 demonstrates the convergence of our algorithms, which optimizes the latency of all requests coming from different racks for the weighted queuing model at the aggregate switch: chunk placement $\mathcal{S}_i$, load balancing $\pi_{i,j}$ and bandwidth weights distribution $w_{i,j}$ (for weighted queuing model). The JLCM algorithm, which has been applied as part of our JLWO algorithm, was proven to converge in Theorem 2 of [28]. In this paper, we see the convergence of the proposed optimized queuing algorithms in Fig. 3. By performing similar speedup techniques as in [28], our algorithms for the two models efficiently solve the optimization problem with $r = 500$ files at each of the 10 racks. It is observed that the normalized objective converges within 172 iterations for a tolerance $\epsilon = 0.01$, where each iteration has an average run time of 1.38 sec, when running on an 8-core, 64-X86 machine, therefore the algorithm converges within 3.89 min on average from observation. To achieve dynamic file management, our optimization algorithm can be executed repeatedly upon file arrivals and departures.

Validate Experiment Setup. While our service delay bound applies to arbitrary distribution and works for systems hosting

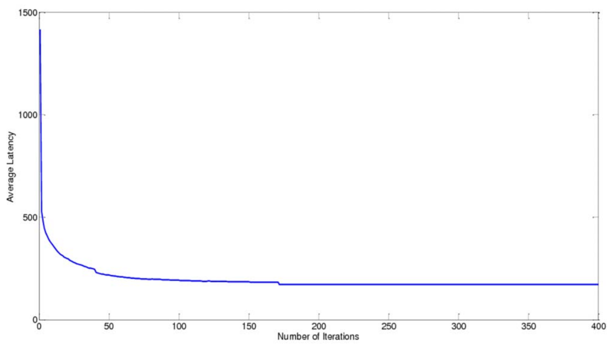

Fig. 3: Convergence of Algorithm JLWO with r=1000 requests for heterogeneous files from each rack on our 80-node testbed. Algorithm JLWO efficiently compute the solution in 172 iterations.

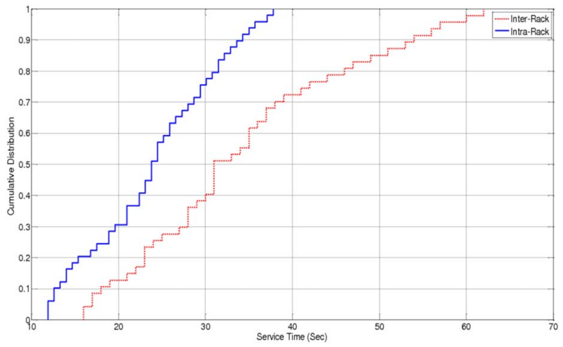

Fig. 4: Actual service time distribution of chunk retrieval through intra-rack and inter-rack traffic for weighted queuing; each of them has 1000 files of size $100MB$ using erasure code (7,4) with the aggregate request arrival rate set to $\lambda_i = 0.25$ /sec in each model

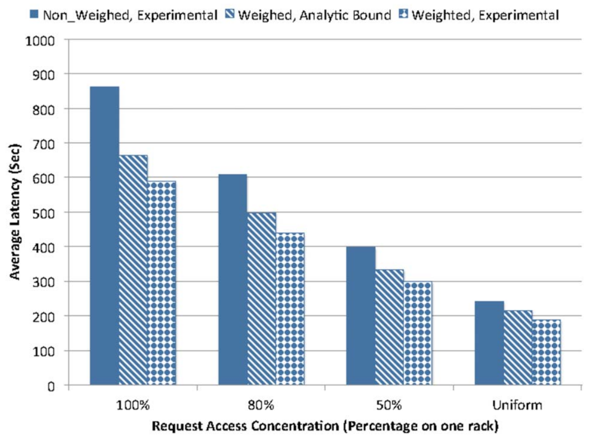

Fig. 5: Comparison of average latency with different access patterns. Experiment is set up for 100 heterogeneous files, each with 10 requests. The figure shows the percentage that these 1000 requests are concentrated on the same rack. Aggregate arrival rate 0.25/sec, file size 200M. Latency improved significantly with weighted queuing. Analytic bound for both cases tightly follows actual latency as well.

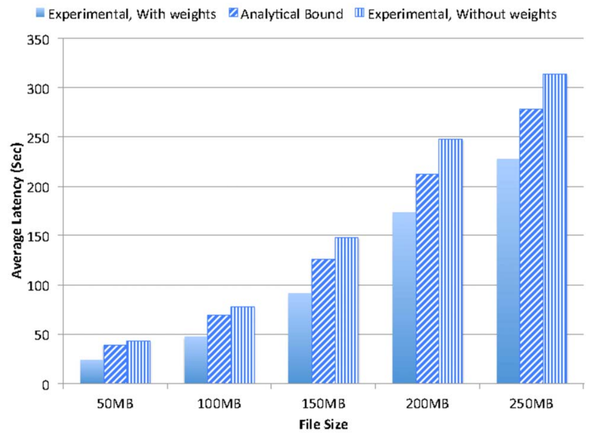

Fig. 6: Evaluation of different file sizes in the weighted queuing model. Aggregate rate 0.25/sec. Compared with *Tahoe's* built-in upload/download algorithm, our algorithm provides relatively lower latency with heterogeneous file sizes. Latency increases as file size increases. Our analytic latency bound taking both network and queuing delay into account tightly follows actual service latency.

any number of files, we first run an experiment to understand actual service time distribution for both intra-rack and inter-rack retrieval in weighted queuing models on our testbed, (intra-rack traffic has a weight of 1, i.e., receives full intra-rack bandwidth). We uploaded $r = 1000$ files of size $100MB$ file using a $(7, 4)$ erasure code from the client at each rack based on the algorithm output of S_i. Inter-rack bandwidth was reserved based on the weights output from the algorithm. We then initiated 1000 file retrieval requests (each request for a unique file) from the clients distributed in the data center, using the algorithm output $\pi_{i,j}$ for retrieval request scheduling with the same erasure code. The experiment has 1000 file requests in total (for 10 racks), with an aggregate request arrival rate of 0.25/sec for clients at all racks and requests are evenly distributed across the racks. Based on the optimal sets for retrieving chunks of each file request provided by our algorithm, we get measurements of service time for the for both inter-rack and intra-rack processes. The average inter-rack bandwidth over all racks is 514 Mbps and the intra/inter-rack bandwidth ratio is $840Mbps/514Mbps = 1.635$. Figure 4 depicts the Cumulative Distribution Function (CDF) of the chunk service time for both intra-rack and inter-rack traffic. We note that intra-rack requests have a mean chunk service time of 26 sec and inter-rack chunk requests have a mean chunk service time of 40 sec, which is a ratio of 1.538 which is very close to the bandwidth ratio of 1.635. This means the chunk service time is nearly proportional to the bandwidth reservation on inter/intra-rack traffic.

Validate algorithms and joint optimization. In order to validate that Algorithm JLWO works for our system model (with weighted queuing model for inter-rack traffic at the switch), we compare our weighted queuing model with Tahoe's native upload/download method without weighted queuing in the cases of different access patterns. In this experiment, we have 100 files of the same file size 200MB, and aggregate arrival rate is 0.25/sec. Each file has 50 requests coming from different racks, and we are measuring latency

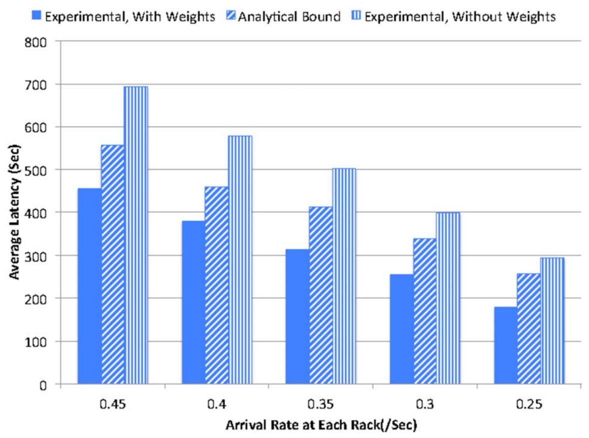

Fig. 7: Evaluation of different request arrival rate in weighted queuing. File size 200M. Compared with *Tahoe's* built-in upload/download algorithm, our algorithm provides relatively lower latency with heterogeneous request arrival rates. Latency increases as requests arrive more frequently. Our analytic latency bound taking both network and queuing delay into account tightly follows actual service latency for both classes.

for the following access patterns: 100% concentration means 100% of the 50$\times$100 requests concentrate on one of the racks. Similarly, 80% or 50% concentration means 80% or 50% of the total requests of each file come from one rack, the rest of the requests spread uniformly among other racks. Uniform access means that for each file, the 50$\times$100 requests are uniformly distributed across the racks. We compare average latency for these requests for each case of the access patterns when we applied the queuing model with and without weighted queuing.

As shown in Fig 5, experimental results indicate that our weighted queuing model can effectively mitigate the long latency due to congestion at one rack as compared with Tahoe's native method. For example, when the request concentration level is 100%, weighted queuing improves average latency by 32%, and when concentration level is 80%, the improvement is 27%, as compared to the 24% improvement provided by 50% concentration and 21% by the uniform distribution. We can see that this improvement in average latency increases as the requests become more concentrated, while in this case with our weighted queuing and the optimal chunk placement and retrieval scheduling, we see more weights allocated to the queues that have much heavier traffic than others. Optimal weight allocation allows us to utilize bandwidth more efficiently and reduce overall latency. We also calculated our analytic bound of average latency when the above access patterns are applied, from the figure we can also see that our analytic bound is tight enough to follow the actual average latency.

Evaluate the performance of our solution To demonstrate the effectiveness of our algorithms, we vary file size in the experiments from 50MB to 250MB with an aggregate request arrival rate for all files at 0.25/sec. We assume uniform random access, i.e., each file will be uniformly accessed from ten racks in the data center with a certain request arrival rate. Upload/download server selection is based on the algorithm output $S_i/\pi_{i,j}$, and bandwidth reserved according to output $w_{i,j}$ from the optimization. Then we submit $r = 1000$ requests from the clients distributed among the racks. We also run experiments with the same settings without weighted queuing, i.e., using *Tahoe's* upload/download policy we introduced in the beginning of this section. Results in Fig 6 show that although Tahoe is using load balancing for dispatching file requests, our algorithm still improves the average latency of requests over all racks significantly. For instance, weighted queuing has a 22% improvement on average for the 5 sample file sizes in this experiment. Latency increases as requested file size increases when arrival rates are set to be the same. Since larger file size means longer service time, it increases queuing delay and thus average latency. We also observe that our analytic latency bound follows actual average service latency in this experiment. We note that the actual service latency involves other aspects of delay beyond queuing delay, and the results show that optimizing the metric of the proposed latency upper bound improves the actual latency with the queuing models.

Similarly, we vary the aggregate arrival rate at each rack from 0.25/sec to 0.45/sec. This time we fix all file requests for file size 200MB. In this experiment we also compare our weighted queuing model with *Tahoe's* built-in up/download scheme without weights-reserved bandwidth allocation. We assume uniform access as before. For weighted queuing model, we use optimized server selection for upload and download for each file request, and optimal bandwidth reservation from Algorithm JLWO. Clients across the data center submit $r = 1000$ requests with an aggregate arrival rate varying from 0.25/sec to 0.45/sec. From Fig 7 we can see that our algorithm outperforms *Tahoe* in terms of average latency in this case as well. The proposed algorithm has an average improvement of 24% in latency. Further, as the arrival rate increases and there is more contention at the queues, this improvement becomes more significant. Thus our algorithm can mitigate traffic contention and reduce latency very efficiently compared to Tahoe's native access policy. In both the cases, the average latency increases as the request arrival at each rack increases since waiting time increases with the workload.

VI. Conclusions

This paper proposes an optimized erasure-coded single data center storage solution. The mean latency of all file requests is jointly optimized over the placement of erasure-coded file chunks and the scheduling of file access requests, as well as the bandwidth reservation at different switches. With the knowledge of the file-access patterns, the proposed solution significantly reduces average latency. In conclusion, this paper demonstrates that knowing the access probabilities of different files from different racks can lead to an efficient optimization of erasure-coded storage, that works significantly better as compared to a storage solution which ignores the file access patterns.

VII. Acknowledgement

The authors would like to thank Chao Tian from University of Tennessee Knoxville (was at AT&T Labs-Research, Bed-

minster when the research was performed) for many helpful discussions.

References

[1] E. Schurman and J. Brutlag, " The user and business impact of server delays, additional bytes and http chunking in web search," *OReilly Velocity Web performance and operations conference*, June 2009.

[2] Y. Lu, Q. Xie, G. Kliot, A. Geller, J. Larus, and A. Greenberg, "Joinidle-queue: A novel load balancing algorithm for dynamically scalable web services," in Proc. *IFIP Perforamnce*, 2010.

[3] Dell data center design, "Data Center Design Considerations with 40GbE and 100GbE,", Aug 2013.

[4] A. Fallahi and E. Hossain, "Distributed and energy-Aware MAC for differentiated services wireless packet networks: a general queuing analytical framework," *IEEE CS, CASS, ComSoc, IES, SPS*, 2007.

[5] A.S. Alfa, "Matrix-geometric solution of discrete time MAP/PH/1 priority queue," *Naval research logistics*, vol. 45, 00. 23-50, 1998.

[6] N.E. Taylor and Z.G. Ives, "Reliable storage and querying for collaborative data sharing systems," *IEEE ICED Conference*, 2010.

[7] J.H. Kim and J.K. Lee, "Performance of carrier sense multiple access with collision avoidance in wireless LANs," *Proc. IEEE IPDS.*, 1998.

[8] E. Ziouva and T. Antoankopoulos, "CSMA/CA Performance under high traffic conditions: throughput and delay analysis," *Computer Comm*, vol. 25, pp. 313-321, 2002.

[9] S. Mochan and L. Xu, "Quantifying Benefit and Cost of Erasure Code based File Systems." *Technical report available at* $http://nisl.wayne.edu/Papers/Tech/cbefs.pdf$, 2010.

[10] H. Weatherspoon and J. D. Kubiatowicz, "Erasure Coding vs. Replication: A Quantitative Comparison." *In Proceedings of the First IPTPS*,2002

[11] C. Angllano, R. Gaeta and M. Grangetto, "Exploiting Rateless Codes in Cloud Storage Systems," *IEEE Transactions on Parallel and Distributed Systems*, Pre-print 2014.

[12] M. Ovsiannikov, S. Rus, D. Reeves, P. Sutter,S. Rao and J. Kelly, "The quantcast file system," *Proceedings of the VLDB Endowment*, vol. 6, pp. 1092-1101, 2013.

[13] L. Huang, S. Pawar, H. Zhang and K. Ramchandran, "Codes Can Reduce Queueing Delay in Data Centers," *Journals CORR*, vol. 1202.1359, 2012.

[14] N. Shah, K. Lee, and K. Ramachandran, "The MDS queue: analyzing latency performance of erasure codes," *Information Theory (ISIT), 2014 IEEE International Symposium on*, July. 2014.

[15] 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. 629660, 1989.

[16] G. Joshi, Y. Liu, and E. Soljanin, "On the Delay-Storage Trade-off in Content Download from Coded Distributed Storage Systems," *arXiv:1305.3945v1*, May 2013.

[17] B. Dekeris, T. Adomkus, A. Budnikas, "Analysis of qos assurance using weighted fair queueing (WQF) scheduling discipline with low latency queue (LLQ)," *Information Technology Interfaces, 28th International Conference on*, 2006.

[18] M. Ashour, N. Le-Ngoc, "Performance Analysis of Weighted Fair Queues with Variable Service Rates," *Digital Telecommunications, International Conference on*, 2006. ICDT '06.

[19] M.L. Ma, J.Y.B. Lee, "," *Peer-to-Peer Computing (P2P), IEEE Tenth International Conference on*, 2010.

[20] P. Xie, J.H. Cui, "An FEC-based Reliable Data Transport Protocol for Underwater Sensor Networks," *Computer Communications and Networks, Proceedings of 16th International Conference on*, 2007. ICCCN 2007.

[21] Y. Yang, K.M.M. Aung, E.K.K. Tong, C.H. Foh, "Dynamic Load Balancing Multipathing in Data Center Ethernet," *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), IEEE International Symposium on*,2010.

[22] Bu. Lee, R. Kanagavelu, K.M.M. Aung, "An efficient flow cache algorithm with improved fairness in Software-Defined Data Center Networks," *Cloud Networking (CloudNet), IEEE 2nd International Conference on*, 2013.

[23] S. Chen, Y. Sun, U.C. Kozat, L. Huang, P. Sinha, G. Liang, X. Liu and N.B. Shroff, " When Queuing Meets Coding: Optimal-Latency Data Retrieving Scheme in Storage Clouds," *IEEE Infocom*, April 2014.

[24] O. N. C. Yilmaz, C. Wijting, P. Lunden, J. Hamalainen, "Optimized Mobile Connectivity for Bandwidth- Hungry, Delay-Tolerant Cloud Services toward 5G," *Wireless Communications Systems (ISWCS), 11th International Symposium on*, 2014.

[25] D. Niu, C. Feng and B. Li, "Pricing cloud bandwidth reservations under demand uncertainty," *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems*, pp. 151-162, June 2012.

[26] S.Suganya and Dr.S.Palaniammal, "A Well-organized Dynamic Bandwidth Allocation Algorithm for MANET," International Journal of Computer Applications, vol. 30(9), pp. 11-15, September 2011.

[27] A. Kumar, R. Tandon and T.C. Clancy, "On the Latency of Erasure-Coded Cloud Storage Systems," arXiv:1405.2833, May 2014.

[28] Y. Xiang, T. Lan, V. Aggarwal, and Y. R. Chen, "Joint Latency and Cost Optimization for Erasure-coded Data Center Storage," *Proc. IFIP Performance*, Oct. 2014 (available at arXiv:1404.4975).

[29] MOSEK, "MOSEK: High performance software for large-scale LP, QP, SOCP, SDP and MIP," available online at *http://www.mosek.com/*.

[30] Hungarian Algorithm, available online at *http://www.hungarianalgorithm.com*

[31] D. Bertsimas and K. Natarajan, "Tight bounds on Expected Order Statistics," *Probability in the Engineering and Informational Sciences*, 2006.

[32] A. Abdelkefi and J. Yuming, "A Structural Analysis of Network Delay," *Ninth Annual CNSR*, 2011.

[33] F. Paganini, A. Tang, A. Ferragut and L.L.H. Andrew, "Network Stability Under Alpha Fair Bandwidth Allocation With General File Size Distribution," *IEEE Transactions on Automatic Control*, 2012.

[34] A.B. Downey, "The structural cause of file size distributions," *Proceedings of Ninth International Symposium on MASCOTS*, 2011.

[35] M. Bramson, Y. Lu, and B. Prabhakar, "Randomized load balancing with general service time distributions," *Proceedings of ACM Sigmetrics*, 2010.

[36] D. Bertsimas and K. Natarajan, "Tight bounds on Expected Order Statistics," *Probability in the Engineering and Informational Sciences*, 2006.

[37] B. Warner, Z. Wilcox-O'Hearn, and R. Kinninmont, "Tahoe-LAFS docs," *available online at https://tahoe-lafs.org/trac/tahoe-lafs*.