2016 IEEE 36th International Conference on Distributed Computing Systems

Sprout: A functional caching approach to minimize service latency in erasure-coded storage

Vaneet Aggarwal[1], Yih-Farn R Chen[2], Tian Lan[3], and Yu Xiang[2]

[1] School of IE, Purdue University, West Lafayette, IN 47907, USA

[2] AT&T Labs-Research, Bedminster, NJ 07921, USA

[3] Department of ECE, George Washington University, DC 20052, USA

vaneet@purdue.edu, chen@research.att.com, tlan@gwu.edu, yxiang@research.att.com

The rapid growth of data traffic in storage systems has put a significant burden on the underlying networks of cloud storage systems. Historically, a key solution to relieve this traffic burden is caching [1]. Many companies have adopted erasure-coded storage systems. However, caching for data centers when the files are encoded with an erasure code has not been studied to the best of our knowledge. This paper proposes a new *functional caching* approach called *Sprout* that can efficiently capitalize on existing file coding in erasure-coded storage systems. In contrast to exact caching that stores d chunks identical to original copies, our functional caching approach forms d new data chunks, which together with the existing n chunks satisfy the property of being an $(n+d,k)$ MDS code. Thus, the file can now be recovered from any k out of $n+d$ chunks (rather than k out of n under exact caching), effectively extending coding redundancy, as well as system diversity for scheduling file access requests. The proposed functional caching approach saves latency due to more flexibility to obtain $k-d$ chunks from the storage system at a very minimal additional computational cost of creating the coded cached chunks. While quantifying service latency in erasure-coded storage systems is an open problem, we generalize previous results on probabilistic scheduling policy [2, 3] that distributes file requests to cache and storage nodes with optimized probabilities, and derive a closed-form upper bound on mean service latency for the proposed functional caching approach.

This analytical latency model for functional caching enables us to formulate a cache-content optimization problem. This problem is an integer optimization problem, which is very difficult to solve. Towards this end, for given data chunk placement and file request arrival rates, we propose a heuristic algorithm that iteratively identifies files whose service latency benefits most from caching and constructs new functional data chunks until the cache is filled up. The algorithm can be efficiently computed to allow online cache optimization and management with time-varying arrival rates.

System Model: We consider a distributed storage system consisting of m heterogeneous storage nodes, denoted by $\mathcal{M} = \{1, 2, \ldots, m\}$. To distributively store a set of r files, indexed by $i = 1, \ldots, r$, we partition each file i into k_i fixed-size chunks and then encode it using an (n_i, k_i) MDS erasure code to generate n_i distinct chunks of the same size for file i. The encoded chunks are stored on the disks of n_i distinct storage nodes. A set $\mathcal{S}_i$ of storage nodes, satisfying $\mathcal{S}_i \subseteq \mathcal{M}$ and $n_i = |\mathcal{S}_i|$ is used to store file i. Therefore, each chunk is placed on a different node to provide high reliability in the event of node or network failures. The use of (n_i, k_i) MDS erasure code allows the file to be reconstructed from any subset of k_i-out-of-n_i chunks, whereas it also introduces a redundancy factor of n_i/k_i.

The files are accessed by cloud servers located in the same datacenter. A networked cache of size C is available at each compute server to store a limited number of chunks of the r files in its cache memory. File access requests are modeled by a non-homogenous Poisson process. We make the assumption of time-scale separation, such that system service time is divided into multiple bins, each with different request arrival rates, while the arrival rates within each bin remain stationary. Let $\lambda_{i,j,t}$ be the arrival rate of file-i requests at compute server j in time bin t. Since each cache serves a single compute server, we consider a separate optimization for each cache and suppress server index j in the notations. Let $d_i \le k_i$ (chunks) be the size of cache memory allocated to storing file i chunks. These chunks in cache memory can be both prefetched in an offline fashion during a placement phase [1] (during hours of low workload) and updated on the fly when a file i request is processed by the system.

Under functional caching, d_i new coded data chunks of file i are constructed and cached, so that along with the existing n_i chunks satisfy the property of being an $(n_i + d_i, k_i)$ MDS code. Therefore, for given erasure coding and chunk placement on storage nodes and cache, a request to access file i can be processed using d_i cached chunks in conjunction with $k_i - d_i$ chunks on distinct storage nodes. After each file request arrives at the storage system, we model this by treating the file request as a *batch* of $k_i - d_i$ chunk requests that are forwarded to appropriate storage nodes, as well as d_i chunk requests that are processed by the cache. Each storage node buffers requests in a common queue of infinite capacity and process them in a FIFO manner. The file request is served when all k_i chunk requests are processed. Further, we consider chunk service time $\mathbf{X}_j$ of node j with *arbitrary distributions*, whose statistics can be inferred.

Optimization Formulation: At time t, we consider the cache optimization problem, which decides the optimal number $d_{i,t}$ of file-i chunks to store in the cache memory, satisfying cache capacity constraint $\sum_i d_{i,t} \le C$, in order to minimize mean service latency of all files. Under functional caching, each file-i request is served by accessing $d_{i,t}$ chunks in the cache, along with $k_i - d_{i,t}$ distinct chunks that are selected from n_i storage nodes. Thus, the latency to access file i under functional caching is determined by the maximum processing (queuing) delay of the selected $k_i - d_{i,t}$ storage nodes. Quantifying service latency in such erasure-coded system is an open problem. In this paper, we use probabilistic scheduling proposed in [2] to derive an upper bound on the average file latency.

We denote $\mathbf{X}_j$ as the chunk service time at node j, which has an arbitrary distribution satisfying finite mean $\mathbb{E}[\mathbf{X}_j] = 1/\mu_j$, variance $\mathbb{E}[\mathbf{X}_j^2] - \mathbb{E}[\mathbf{X}_j]^2 = \sigma_j^2$, second moment $\mathbb{E}[\mathbf{X}_j^2] = \Gamma_j^2$, and third moment $\mathbb{E}[\mathbf{X}_j^3] = \hat{\Gamma}_j^3$. Following [2], the expected latency $\bar{T}_{i,t}$ of file i in time-bin t under probabilistic scheduling is upper bounded by $\bar{U}_{i,t}$, given by

$$\begin{aligned}\bar{U}_{i,t} &= \min_{z_{i,t}\in\mathbb{R}} \Bigg\{ z_{i,t} + \sum_{j\in\mathcal{S}_{i,t}} \frac{\pi_{i,j,t}}{2}\left(\mathbb{E}[\mathbf{Q}_{j,t}] - z_{i,t}\right) \\ &\quad + \sum_{j\in\mathcal{S}_{i,t}} \frac{\pi_{i,j,t}}{2}\left[\sqrt{(\mathbb{E}[\mathbf{Q}_{j,t}] - z_{i,t})^2 + Var[\mathbf{Q}_{j,t}]}\right]\Bigg\},\end{aligned}$$

where
$$\mathbb{E}[\mathbf{Q}_{j,t}] = \frac{1}{\mu_j} + \frac{\Lambda_{j,t}\Gamma_j^2}{2(1-\rho_{j,t})},$$
$$Var[\mathbf{Q}_{j,t}] = \sigma_j^2 + \frac{\Lambda_{j,t}\hat{\Gamma}_j^3}{3(1-\rho_{j,t})} + \frac{\Lambda_{j,t}^2\Gamma_j^4}{4(1-\rho_{j,t})^2},$$

where $\rho_{j,t} = \Lambda_{j,t}/\mu_j$ is the request intensity at node j, and $\Lambda_{j,t} = \sum_i \lambda_{i,t}\pi_{i,j,t}$ is the mean arrival rate at node j. The bound is tight in the sense that there exists a distribution of $\mathbf{Q}_{j,t}$ such that (1) is satisfied with exact equality.

We now formulate the cache optimization in a single time-bin. The optimization is over cache content placement $d_{i,t}$, scheduling probabilities $\pi_{i,j,t}$, and auxiliary variable $z_{i,t}$ in the upper bound. Let $\hat{\lambda}_t = \sum_i \lambda_{i,t}$ be the total arrival rate, so $\lambda_{i,t}/\hat{\lambda}$ is the fraction of file i requests, and average latency of all files is given by $\sum_i(\lambda_{i,t}/\hat{\lambda}_t)\bar{T}_{i,t}$. Our objective is to minimize an average *latency* objective, i.e.,

$$\begin{aligned} min &\quad \sum_{i=1}^{r} \frac{\lambda_{i,t}}{\hat{\lambda}_t}\bar{U}_{i_t} \\ s.t. &\quad (1),\ (1),\ (1),\ \sum_{j=1}^{m}\pi_{i,j,t} = k_i - d_{i,t},\ \pi_{i,j,t}, d_{i,t} \geq 0, \\ &\quad \sum_i d_{i,t} \leq C,\ \pi_{i,j,t} = 0 \text{ for } j \notin \mathcal{S}_i,\ \pi_{i,j,t} \leq 1, \\ &\quad z_{i,t} \geq 0, d_{i,t} \in \mathbb{Z}. \\ var. &\quad \{\pi_{i,j,t},\ d_{i,t},\ z_{i,t}\},\ \forall i,j,t. \end{aligned}$$

Here the constraints $\sum_{j=1}^m \pi_{i,j,t} = k_i - d_{i,t}$ and $\pi_{i,j,t} \leq 1$ ensure that $k_i - d_{i,t}$ distinct storage nodes (along with $d_{i,t}$ chunks in the cache) are selected to process each file request, following probabilistic scheduling in [2]. Clearly, storage nodes without desired chunks cannot be selected, i.e., $\pi_{i,j,t} = 0$ for $j \notin \mathcal{S}_i$. Finally, the cache has a capacity constraint $\sum_i d_{i,t} \leq C$.

Algorithm: Solving the cache optimization gives us the optimal cache content placement and scheduling policy to minimize file access latency. We first note that the variable $d_{i,t}$ can be absorbed into scheduling decision $\pi_{i,j,t}$ because of the equality constraint $d_{i,t} = k_i - \sum_{j=1}^m \pi_{i,j,t}$. In order to solve the problem, we use an alternating minimization over two dimensions - the first decides on $z_{i,t}$ given $\pi_{i,j,t}$, and the second decides on $\pi_{i,j,t}$ given $z_{i,t}$. The first problem is convex, and can be easily solved by gradient descent. However, the second problem has integer constraint. In order to deal with this, we first remove integer constraint to solve the problem. Then, a certain percentage of files whose fractional part of content accessed from the disk is highest are added a part in the disk to make the part in disk as integers. The optimization over $\pi_{i,j,t}$ keeps running until $\sum_{j=1}^m \pi_{i,j,t}$ for all files is an integer.

Numerical Results: We simulated our algorithm in a cluster of $m = 12$ storage servers holding $r = 1000$ files of size 100 MB each using a (7,4) erasure code. Unless stated otherwise, cache size remains as 500 times of the chunk size (i.e., 500 times of 25 MB). The arrival rate for each file is set at a default value of $\lambda_i = 0.000156$/sec, 0.000156/sec, 0.000125/sec, 0.000167/sec, 0.000104/sec for every five out of the 1000 files of each size. It gives an aggregate arrival rate of all files to be 0.1416/sec. The inverse of mean service times for the 12 servers are set based on measurements of real service time in the distributed storage system, and they are {0.1, 0.1, 0.1, 0.0909, 0.0909, 0.0667, 0.0667, 0.0769, 0.0769, 0.0588, 0.0588} for the 12 storage servers respectively. The placement of files on the servers is chosen at random.

Figure 1 demonstrates the convergence of our algorithm in one time-bin for cache size $C\times25$ MB. A random initialization is chosen for $C = 100$, while the converged solution for $C = 100$ is taken as initialization for $C = 200$ and so on. We note that the algorithm converges within a few iterations, and converges in less than 20 iterations with a threshold of 0.01 on latency for all cache size values in Figure 1. Fig 2 shows that the average latency decreases as cache size increases, where average latency is 23 sec when no file has any content in cache, and is 0 sec when cache size is 4000 chunk-size since 4 chunks of each file can be in the cache. We note that the latency is convex decreasing function of the cache size, depicting that our algorithm is effectively updating content in cache and showing diminishing returns in decrease of latency after reaching certain cache size.

References

[1] R. Pedarsani, M. A. Maddah-Ali, and U. Niesen, "Online coded caching," in *IEEE International Conference on Communications, ICC 2014, Sydney, Australia, June 10-14, 2014*, 2014, pp. 1878–1883.

[2] Y. Xiang, T. Lan, V. Aggarwal, and Y.-F. Chen, "Joint latency and cost optimization for erasure-coded data center storage," *Networking, IEEE/ACM Transactions on*, vol. PP, no. 99, pp. 1–1, 2015.

[3] Y. Xiang, T. Lan, V. Aggarwal, and Y. F. R. Chen, "Joint latency and cost optimization for erasurecoded data center storage," *SIGMETRICS Perform. Eval. Rev.*, vol. 42, no. 2, pp. 3–14, Sep. 2014.