The Transfer Load-I/O Trade-off for Coded Caching

Tianqiong Luo and Borja Peleato
Electrical and Computer Engineering
Purdue University
West Lafayette IN 47907 Email: {luo133,bpeleato}@purdue.edu

***Abstract*—Caching has been investigated as a useful technique to reduce the network burden by prefetching some contents during off-peak hours. Coded caching [1] can reduce traffic load by broadcasting coded messages that can benefit multiple users but, in the case with redundant requests, it requires reading some data segments multiple times to compose different coded messages. Hence, coded caching requires more disk I/Os (Input/Output) than uncoded transmission. This paper proposes caching and delivery algorithms which combine coded and uncoded transmission to strike a trade-off between traffic load and disk I/Os. Our algorithms can improve the expected performance in terms of the user requests.**

***Index Terms*—coded caching, I/O, coding, prefetching**

I. Introduction

With the recent information explosion, users and applications demand accessing data at a higher speed and lower latency, which poses challenges to both networks and devices. This paper addresses two performance bottlenecks of storage systems: the number of read and write operations (disk I/Os) and the amount of data transferred (transfer load). Disk I/Os are a valuable resource which bounds many applications and a significant amount of research has gone into coding techniques to minimize them in storage systems [2].

Transfer load (or traffic) is the dominant factor in slow or congested networks. Caching has been investigated as a useful technique to relieve peak traffic by prefetching contents during off-peak hours. A caching scheme has two phases: placement and delivery. In the placement phase, users can access all files to fill their caches. In the delivery phase, only the server has database access and it delivers messages to the users to fulfill their requests. In [1], Maddah-Ali and Niesen proposed a caching and delivery scheme offering a worst case performance within a constant factor of the information-theoretic optimum, for a system with a single server broadcasting to multiple users and uniform file popularity. Inspired by their work, [3] studied the average performance of their scheme and the case with random demands. Further works improved the delivery scheme by exploiting commonality among users' demands [4]–[6] and introduced a decentralized version of the algorithm [7].

Maddah-Ali and Nisen's coded caching scheme in [1] (henceforth denoted "M-N scheme") has the lowest peak traffic load in the literature and its extension by Yu et. al. [5] (henceforth denoted "Yu's scheme") is proved to achieve the best average traffic load with uncoded prefetching. However, their I/O performance is suboptimal when there are redundant user demands. The same segment could be read multiple times if it is used to construct different messages, which dramatically increases I/O reads. In contrast, if all messages are transmitted uncoded, each data segment requested is read once and broadcast to all users. Inspired by this fact, we design algorithms which combine coded and uncoded transmissions, so as to strike a trade-off between traffic load and I/O. To the extent of our knowledge, this is the first paper considering the I/O performance of coded caching.

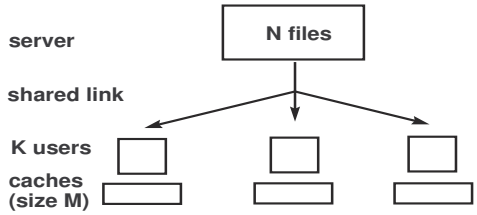

Figure 1: Caching system considered in this paper.

The rest of this paper is organized as follows. Section II introduces the system model and reviews the traditional caching schemes. Section III proposes two algorithms offering different trade-offs between traffic load and I/O access. Section IV provides simulations to support and illustrate our algorithms, and Section V concludes the paper.

II. Background

A. System Model

Our system model is identical to that in M-N scheme [1], shown in Fig. 1: a single server is connected to K users through a shared broadcast link, and N files of size F bits with uniform popularity. Each user has a cache of size MF bits. Users fill their caches during the placement phase and then independently request a file in the delivery phase. We denote these requests by $\mathbf{d} = \{d_1, \ldots, d_K\}$, where d_k is the index of the file requested by user k. $\mathbf{d}$ is uniformly distributed over $\mathcal{D} = \{1, \ldots, N\}^K$ and the number of distinct files requested is denoted by $N_e(\mathbf{d})$.

The server must then fulfill those requests. We wish to study the trade-off between the resulting load on the shared link and the disk I/O. Disks are read one page at a time, all of the same size [8]. Therefore, the disk I/O is approximately proportional to the total data read. Moreover, if the same file segment is used to construct k messages, we assume that it needs to be read k times. Denoting the load on the shared link by R_t and

the total data read by R_{IO} (both normalized by the file size), the objective is to design an algorithm to minimize the cost

$$R_{\text{cost}}(\alpha, \mathbf{d}) = \alpha R_t(\alpha, \mathbf{d}) + (1-\alpha) R_{IO}(\alpha, \mathbf{d}), \quad (1)$$

where $\alpha \in [0, 1]$ is the trade-off coefficient. Although this paper will focus on $R_{\text{cost}}(\alpha, \mathbf{d})$, the proposed algorithms could be easily applied to other cost functions. Also, we use an overline to denote expected values, *e.g.*

$$\overline{R_{\text{cost}}(\alpha)} = \alpha E_{\mathbf{d}}[R_t(\alpha, \mathbf{d})] + (1-\alpha) E_{\mathbf{d}}[R_{IO}(\alpha, \mathbf{d})]. \quad (2)$$

B. Uncoded scheme

In the uncoded scheme, all users cache the same M/N fraction from every file and the server sends plain missing segments to all users. Since each missing data segment is read once and broadcasted, the total data read (R^u_{IO}) is identical to the total data sent (R^u_t):

$$R^u_{IO}(\mathbf{d}) = R^u_t(\mathbf{d}) = N_e(\mathbf{d}) g, \quad (3)$$

where $g = 1 - \frac{M}{N}$ is the local caching gain. Each file is requested with probability $p_r = 1 - (1 - \frac{1}{N})^K$, so

$$\overline{R^u_{cost}} = \overline{R^u_t} = \overline{R^u_{IO}} = N p_r g. \quad (4)$$

Lemma 2.1: The uncoded scheme is optimal in terms of expected data read among all schemes with uncoded prefetching, *i.e.*, the $\overline{R_{IO}}$ in Eq. (4) is minimal among all schemes with uncoded prefetching.

Proof: Denote m_{ij} the fraction of file j being cached by user i, for $j = 1, \ldots, N$ and $i = 1, \ldots, K$. Given a list of demands $\mathbf{d}$, let $\mathcal{U} = \{u_1, \ldots, u_{N_e(\mathbf{d})}\}$ be an arbitrary subset of users requesting distinct files $\{f_1, \ldots, f_{N_e(\mathbf{d})}\}$. Then

$$R_{IO}(\mathbf{d}) \geq \sum_{i=1}^{N_e(\mathbf{d})} (1 - m_{u_i f_i}), \quad (5)$$

since the total data read cannot be lower than that delivered.

Each user $u_i \in \mathcal{U}$ has probability $\frac{1}{N}$ of requesting file j, so the average I/O for user u_i is at least $\frac{1}{N}\sum_{j=1}^{N}(1 - m_{u_i j})$. Since $\sum_{j=1}^{N} m_{u_i j} = M$, the average I/O for u_i is at least $\frac{1}{N}(N - M) = g$. Combined with Eq. (5), $\overline{R_{IO}}$ is bounded by

$$\overline{R_{IO}} \geq E_{\mathbf{d}}[N_e(\mathbf{d}) g] = N p_r g.$$

■

Lemma 2.1 proves that the uncoded scheme yields the best $\overline{R_{IO}}$ with uncoded prefetching, but in some cases the I/O could be lowered by caching coded segments [9]. For example, consider a system with two files (A, B), two users, and cache size $M = \frac{1}{2}$. The uncoded scheme yields $\overline{R^u_{IO}} = 1.125$, but dividing each file into two identical segments (A_1, A_2, B_1, B_2) and caching $A_i \oplus B_i$ at user i would only require $R_{IO} = 1$, regardless of the requests (*e.g.*, if user 1 requests A and user 2 requests B, then the server only needs to transmit A_2, B_1). However, the optimal I/O for coded prefetching is a complex problem beyond the scope of this paper. Instead, we focus our discussion on uncoded prefetching.

C. Coded scheme

The centralized coded caching scheme proposed by Maddah-Ali and Niesen [1] splits each file into $\binom{K}{t}$ nonoverlapping segments of equal size, with $t = \frac{KM}{N}$, and caches each segment in a distinct group of t users. In the delivery phase, the server sends one message to each subset of $t+1$ users, for a total of $\binom{K}{t+1}$ messages. Each message is composed as the XOR of the $t+1$ segments requested by one user and cached by the others. Each user can then cancel out the segments that it is caching to recover the desired segment. This algorithm has the lowest known traffic load in the worst case, *i.e.*, when all users request distinct files.When $N \geq K$, it is given by

$$R^m_t = \binom{K}{t+1} \Big/ \binom{K}{t} = Kg/(1 + KM/N), \quad (6)$$

where $g = 1 - \frac{M}{N}$. Each message is the XOR of $t+1$ segments, thus the normalized I/O is $R^m_{IO} = Kg$.

Yu's scheme and our own research [5], [6] extended this work to the case with redundant requests and more general values of N and K. Both use the same placement as M-N. For the delivery, the server picks $N_e(\mathbf{d})$ "leader" users requesting distinct files and only sends messages to subsets of $t+1$ users containing at least one leader. The corresponding transfer load for a given set of requests $\mathbf{d}$ is:

$$R^c_t(\mathbf{d}, t) = \frac{\binom{K}{t+1} - \binom{K - N_e(\mathbf{d})}{t+1}}{\binom{K}{t}}. \quad (7)$$

This extension is shown to yield the lowest expected traffic load with uncoded prefetching[1]. Since each message is the XOR of $t+1$ segments, the expected cost is:

$$\overline{R^c_{\text{cost}}(\alpha, t)} = E_{\mathbf{d}}[\alpha R^c_t(\mathbf{d}, t) + (1-\alpha)(1+t) R^c_{IO}(\mathbf{d}, t)], \quad (8)$$

where R^c_{IO} is the disk I/O for the coded scheme.

Further extensions [3], [5] proposed a decentralized version of the algorithm without coordination in the content placement: users randomly cache a subset of $\frac{MF}{N}$ bits from every file. The delivery phase takes a K-step greedy approach: for bits which are stored in exactly i users $(i = 0, \ldots, K-1)$, it constructs messages by XORing $i+1$ segments, similarly to the centralized scheme.

Both M-N and Yu's schemes have optimal I/O in the worst case (*i.e.*, no repeated requests), given by Kg, as well as the best peak traffic load in the literature. Hence, they also yield the lowest worst-case cost $R_{\text{cost}}(\alpha)$. They are the basis for the algorithms proposed in this paper. It is therefore recommended that readers have a clear understanding of both M-N and Yu's schemes before proceeding.

III. General Algorithms

This section proposes algorithms aiming at minimizing the cost functions in Eq. (1) and (2). Subsection III-A proposes an adaptive delivery algorithm that provides the same worst-case

[1]It is worth noting that M-N and Yu's schemes are identical in the worst case, *i.e.*, $R^c_t(\mathbf{d}, t) = R^m_t$ when $N_e(\mathbf{d}) = K$.

user\ segment	1	2	3	4	5	6	request
1	X	X	X				C
2	X			X	X		B
3		X		X		X	A
4			X		X	X	A

Table I: Mapping of file segments to user caches. Each cache stores the same three segments for every file, marked with X.

performance as M-N and Yu's scheme, but reduces $R_{\text{cost}}(\mathbf{d})$ in the case with redundant requests. Subsection III-B proposes a new placement and delivery algorithm which sacrificew worst case performance to further reduce the average cost.

A. Adaptive delivery

As mentioned in Section II-C, the coded caching schemes provide the best R_{cost} in the worst case. However, the cost can be further reduced when there are redundant requests by sending some segments uncoded. Take the following case as an example.

Example 3.1: Consider a server with 4 files (denoted A, B, C and D), 4 users with a normalized cache size $M = 2$, and a trade-off parameter $\alpha = 0.2$. In the placement phase, file A is split into 6 segments (denoted $A_1, A_2, \ldots, A_6$) and each segment is cached by two users. The same goes for files B, C, D. Table I indicates the indices of the three segments that each user stores, assumed to be the same for all files.

Let the requests be C, B, A, A. According to Eq. (8), $R^c_{cost} = 1.73$ in the delivery phase. In this example, we notice that A_1 is needed by users 3 and 4, so Yu's scheme would include it in two messages: $A_1 \oplus B_2 \oplus C_4$ and $A_1 \oplus B_3 \oplus C_5$. If the server broadcasts A_1 uncoded along with $B_2 \oplus C_4$ and $B_3 \oplus C_5$, and transmits all other messages as in Yu's scheme, the users are still able to recover the requested files. The traffic load is higher but the I/O is reduced. The resulting $R_{\text{cost}} = 1.63$ is better than both M-N and Yu's schemes.

The proposed delivery algorithm is as follows. Segments requested by more than $\frac{1}{1-\alpha}$ users are transmitted uncoded. Other messages are composed according to Yu's scheme, but omitting the segments already sent uncoded. If this omission causes duplicates among the messages in Yu's scheme, only one of them should be transmitted.

This adaptive delivery ensures that the users get the same information as in Yu's and M-N scheme to retrieve the requested file, but with lower $R_{cost}(\alpha, \mathbf{d})$ when there are redundant user requests. If a segment is requested by j users ($j = 1, \ldots, K$), transmitting it uncoded increases R_t to $R'_t = R_t + 1/\binom{K}{t}$ and decreases R_{IO} to $R'_{IO} = R_{IO} - (j-1)/\binom{K}{t}$. Therefore, $R'_{\text{cost}} = \alpha R'_t + (1-\alpha)R'_{IO}$ as defined in Eq. (1) is lower than R_{cost} when $j > \frac{1}{1-\alpha}$.

When each segment is requested by only one user, our adaptive delivery scheme is identical to M-N and Yu's scheme, ensuring an optimal R_{cost} in the worst case. This adaptive delivery algorithm can also be easily extended to the decentralized scheme in [5], [7], following the same principle: the bits which are requested by more than $\frac{1}{1-\alpha}$ users are sent uncoded and all the others coded.

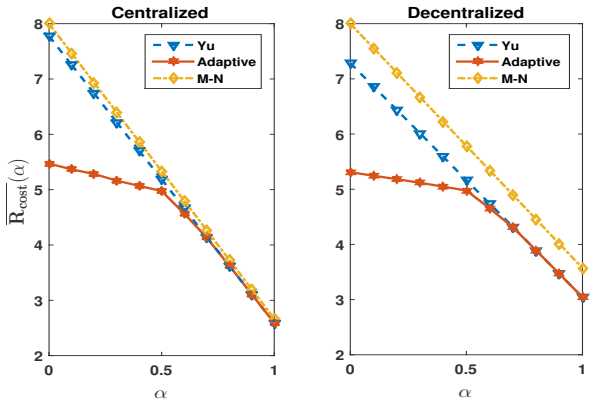

Figure 2: Comparison of adaptive and coded schemes for varying trade-off coefficient α ($K = 10, N = 10, M = 2$).

Fig. 2 compares the expected cost $\overline{R_{\text{cost}}}$ of M-N, Yu's and our adaptive scheme in both the centralized and decentralized settings. It shows that, when α is small, the proposed algorithm has a much better performance than M-N scheme and Yu's scheme in both settings.

B. Partial Caching

The adaptive delivery scheme used the same placement as [1] to ensure optimal R_{cost} in the worst case. This subsection proposes an algorithm achieving a lower expected cost $\overline{R_{\text{cost}}(\alpha)}$, as defined in Eq. (2), at the cost of suboptimal worst case performance. Intuitively, for small α, $\overline{R_{\text{cost}}(\alpha)}$ is dominated by the I/O, so all segments should be sent uncoded; vice versa, when α is close to 1, they should all be sent coded. Hence, if the users only cache a fraction $p \in [\frac{M}{N}, 1]$ of each file and the rest is always transmitted uncoded, we obtain a better average performance for intermediate values of α. Choosing $p = \frac{M}{N}$ corresponds to the uncoded scheme, where all users cache the same segments, while $p = 1$ corresponds to M-N and Yu's scheme, where the whole file is cached among the users.

The proposed algorithm is as follows. In the placement phase, we choose a fraction p of each file to be cached at the users' end. This portion is divided into $t' = \frac{KM}{Np}$ segments and the rest (of size $(1-p)F$) is not cached. In the delivery phase, the cached part is transmitted coded using Yu's scheme and the rest is transmitted uncoded. The fraction p is optimized to minimize $\overline{R_{\text{cost}}}$:

$$p = \arg\min_{p} \left((1-p) R^u_{\text{cost}}(\alpha) + p R^c_{\text{cost}}(\alpha, t') \right),$$

where $t' = \frac{KM}{Np}$, R^u_{cost} is defined in Eq. (4) and R^c_{cost} is defined in Eq. (8). This algorithm can be easily extended to the decentralized case by caching a random portion p of each file at each user, employing the decentralized transmission strategy mentioned in section II-C for these portions, and transmitting the uncached portions uncoded.

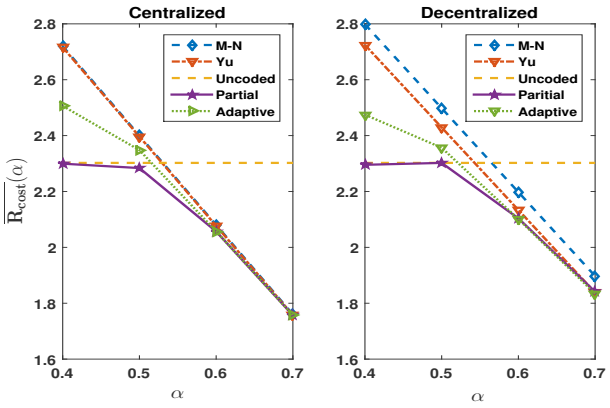

Figure 3: Comparison of partial caching, adaptive, coded and uncoded schemes for varying α ($K = 8, N = 6, M = 3$).

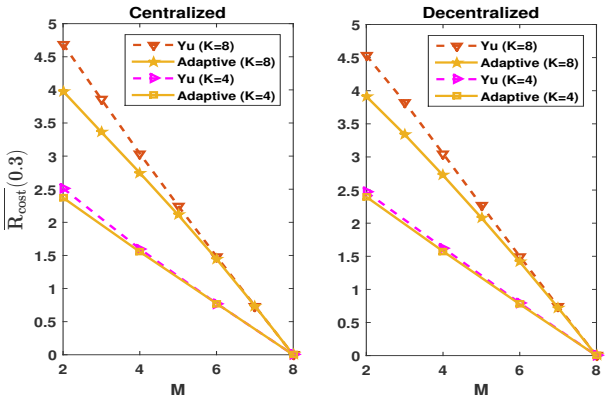

Figure 4: Comparison of adaptive and coded schemes for varying cache capacity and users ($N = 8, \alpha = 0.3$).

The coded, uncoded and partial caching schemes are compared in Fig. 3 for both the centralized and decentralized cases. When α is small, the partial caching scheme takes $p = \frac{M}{N}$ and transmits all the segments uncoded; vice versa, when α is close to 1, the algorithm takes $p = 1$ and it is equivalent to Yu's scheme. Partial caching offers lower $\overline{R_{\text{cost}}}$ than all the other schemes, including the proposed adaptive delivery, for all values of α.

IV. Simulations

This section compares the proposed algorithms with traditional schemes through simulations. It fixes the number of files as $N = 8$ and studies the trade-off between traffic load and I/O by varying the cache size M and the number of users K.

Fig. 4 compares the performance of the adaptive scheme and Yu's scheme with trade-off parameter $\alpha = 0.3$ when the cache size M changes. It shows that the adaptive scheme has an advantage over Yu's scheme in terms of $\overline{R_{\text{cost}}}$ when M is small, but the gap closes as M increases. Moreover, the figure presents results for 4 and 8 users, showing that the gains are more prominent as the number of users increases. This is because both small cache size and more users increase the chance that a segment is requested by multiple users.

Fig. 5 compares the performance of the coded, uncoded and partial caching schemes for different number of users. As mentioned in section III-B, as α increases, the portion p of each file that is transmitted coded also increases. The threshold α for which p is no longer 0 is bigger for the system with 8 users than for the system with 4 users. This is because when there are more users, the probability that some users request the same file increases, which benefits the uncoded transmission.

V. Conclusion

This paper proposes algorithms to study the trade-off between traffic load and I/O for coded caching in both the centralized and decentralized settings. The proposed algorithms strike a balance between coded and uncoded transmissions, showing better expected performance than traditional schemes.

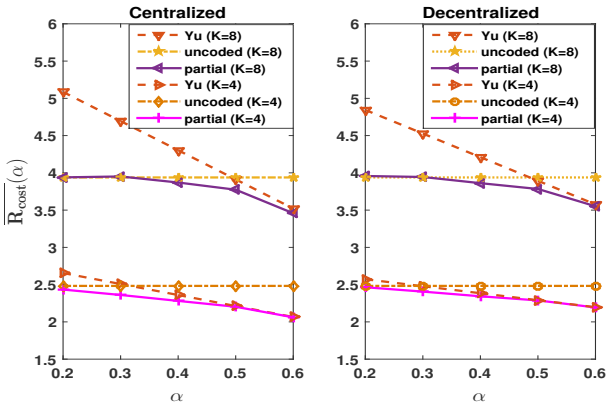

Figure 5: Comparison of partial caching, coded and uncoded schemes for different number of users ($N = 8, M = 2$).

The proposed algorithms could also be extended to address nonuniform file popularities, for example by grouping the files by popularity and dealing with each group independently, as in [3]. Future research could study the transfer load-I/O trade-off for schemes with coded prefetching.

References

[1] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," IEEE Trans. Inf. Theory, vol. 60, no. 5, pp. 2856–2867, 2014.

[2] K. Rashmi, P. Nakkiran, J. Wang, N. B. Shah, and K. Ramchandran, "Having your cake and eating it too: Jointly optimal erasure codes for I/O, storage, and network-bandwidth." in FAST, 2015, pp. 81–94.

[3] U. Niesen and M. A. Maddah-Ali, "Coded caching with nonuniform demands," IEEE Trans. Inf. Theory, vol. 63, no. 2, pp. 1146–1158, 2017.

[4] S. A. Saberali, H. E. Saffar, L. Lampe, and I. Blake, "Adaptive delivery in caching networks," IEEE Communications Letters, vol. 20, no. 7, pp. 1405–1408, 2016.

[5] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "The exact rate-memory tradeoff for caching with uncoded prefetching," in Int. Symp. on Information Theory (ISIT). IEEE, 2017, pp. 1613–1617.

[6] T. Luo, V. Aggarwal, and B. Peleato, "Coded caching with distributed storage," submitted to IEEE Trans. Inf. Theory, 2016.

[7] M. A. Maddah-Ali and U. Niesen, "Decentralized coded caching attains order-optimal memory-rate tradeoff," IEEE/ACM Trans. On Networking, vol. 23, no. 4, pp. 1029–1040, 2015.

[8] E. J. O'neil, P. E. O'neil, and G. Weikum, "The LRU-K page replacement algorithm for database disk buffering," ACM SIGMOD Record, vol. 22, no. 2, pp. 297–306, 1993.
[9] Z. Chen, P. Fan, and K. B. Letaief, "Fundamental limits of caching: Improved bounds for small buffer users," IET Commun., vol. 10, no. 17, pp. 2315–2318, 2016.