EE447

COURSE PROJECT

DOCUMENTATION

Transmission Strategy on Time Variant Caching System

Author:
Huang WENDI

May 27, 2018

Contents

Abstract

Caching is a technique to reduce peak traffic rates by prefetching popular content into memories at the end users. The gain offered by this approach, which I term local caching gain, depends on the local cache size (i.e., the memory available at each individual user). In this paper, I base on a second, global, caching gain not utilized by conventional caching schemes. This gain depends on the aggregate global cache size (i.e., the cumulative memory available at all users), even though there is no cooperation among the users. Different from conventional cache, the model used in this paper is a time-variant cache of which size can increase over time. In this paper, I propose an algorithm to decide how to distribute files to users with such kind of cache.

1. Introduction

1.1. Background

The high temporal variability of network traffic results in communication systems that are congested during peak-traffic times and underutilized during off-peak times. One approach to reduce peak trafc is to take advantage of memories distributed across the network (at end users, servers, routers, etc) to duplicate content. This duplication of content, called content placement or caching, is performed during off-peak hours when network resources are abundant. During peak hours, when network resources are scarce, user requests can then be served from these caches, reducing network congestion. In this manner, caching effectively allows to shift traffic from peak to off-peak hours, thereby smoothing out traffic variability and reducing congestion.

From the above discussion, we see that the caching problem consists of two distinct phases. The first phase is the *placement phase*. In this phase, the network is not congested, and the main limitation is the size of the cache memories. The second phase is the *delivery phase*. In this phase, the network is congested, and the main limitation is the rate required to serve the requested content.

In[1], the author proposed a coded caching strategy to get a global gain which derives from jointly optimizing both the placement and delivery phases.

In practice, cache size doesn't remain fixed because as long as webpage isn't closed, the server can continuously transfer data to the user. From this perspective, the cache size is time-increasing. However, user may disconnect the link at any time so we still face the network traffic problem. Therefore, I follow the coded caching strategy and on this basis shatter the file into more pieces in order to handle the mutative coded strategy.

1.2. Model

To formally analyze the performance of the proposed coded caching approach, and in order to evaluate and isolate these two gains, we introduce a new, information-theoretic formulation of the caching problem focusing on its basic structure. In our setting, depicted in Fig. 1, K users are connected to a server through a shared, error-free link. The server has a database of N files of equal size. Each of the users has access to a cache memory big enough to store M of the files. During the placement phase, the caches are filled as a function of the database. During the delivery phase, each user may ask for any one of the N possible files. The objective is to design the placement and delivery phases such that the load of the shared link in the delivery phase is minimized. For simplicity, we restrict the discussion in the introduction section to the most relevant case, in which the number of files N is larger than or equal to the number of users K.

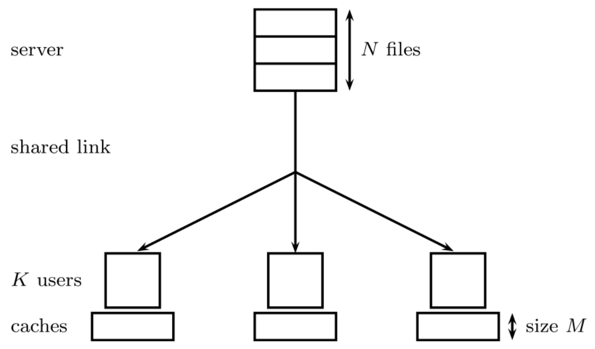


Figure 1: An example of caching system. A server containing N files of size F bits each is connected through a shared link to K users each with an isolated cache of size MF bits.In the figure, N = K = 3 and M = 1.

To better understand the effect of the gains, we can compare the rates of the conventional uncoded scheme (achieving only the local gain) versus the proposed coded scheme (achieving both the local and global gains) for a system with $N = 30$ files and $K = 30$ users as shown in Fig. 2.

As is in the example above, for each M, the rate of the uncoded caching scheme is less than that of uncoded caching. My work is to connect each point on the coded caching curve so that when M increases, the lower R can still be achieved in this process.

2. Existing Work

A.Problem Description

The server has access to a database of N files $W_1 \dots W_N$ each of size F bits. Each user k has an isolated cache memory Z_k of size MF bits for some real number $M \in [0,N]$.

The system operates in two phases: a placement phase and a delivery phase. In the placement phase, the users are given access to the entire database $W_1 \dots W_N$ of files. Each user k is then able to fill the content of its cache Z_k using the database. In the delivery phase, only the server has access to the database of files. Each user k requests one of the les W_{d_k} in the database. The server is informed of these requests and proceeds by transmitting a signal $X_{(d_1,\dots,d_k)}$ of size RF bits over the shared link for some fixed real number R. The quantities RF and R are referred to as the load and the rate of the shared link, respectively. Using the content Z_k of its cache and the signal $X_{(d_1,\dots,d_k)}$ received over the shared link, each user k aims to reconstruct its requested file W_{d_k}.

B.Problem Statement

Let $(W_n)_{n=1}^{N}$ be N independent random variables each uniformly distributed over $[2^F]$ for some $F \in N$. Each W_n represents a file of size F bits.

The caching functions map the files $W_1 \dots W_N$ into the cache content

$$Z_k \triangleq \phi_k(W_1 \dots W_N)$$

for each user $k \in [K]$

3. Main Results

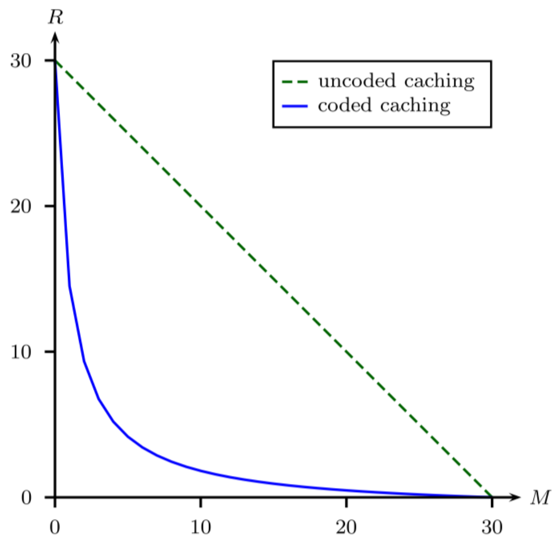


Figure 2: Rate R required in the delivery phase as a function of memory size M for N = 30 files and K = 30 users. The figure compares the performance of the proposed coded caching scheme to that of conventional uncoded caching.

Algorithm 1: Time-variant Coded Caching

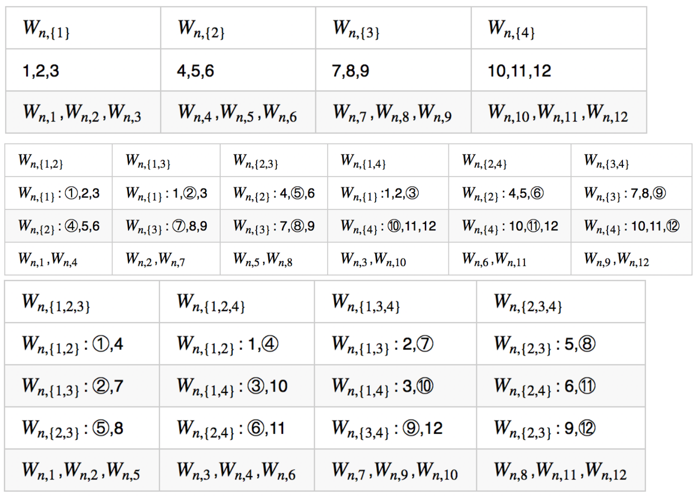
```
procedure PLACEMENT($W_1, \dots, W_N$)
for $n \in [N]$ do
    split $W_n$ into $(W_{n,i}, i \in [\xi])$ of equal size
end
$W_{n,\phi} \leftarrow \{W_{n,1}, W_{n,2}, \dots, W_{n,\xi}\}$;
for $i \in [N]$ do
    $s_i \leftarrow \xi / \binom{K}{i}$;
end
forall $t$ do
    $\Gamma_t \leftarrow \{T \subset [K] : |T| = t\}$;
end
Initialize t as $t = 1$;
for $T \in \Gamma_1$ do
    $W_{n,T} \leftarrow$randomly pick $s_1$ elements from $W_{n,\phi}$;
    $W_{n,\phi} \leftarrow W_{n,\phi} \backslash W_{n,T}$
end
for $k \in [K]$ do
    $Z_k \leftarrow (W_{n,T} : n \in [N], T \in \Gamma_t, k \in T)$
end
end procedure

procedure TRANSMISSION(i)
$\Gamma_i \leftarrow \{T \subset [K] : |T| = i\}$;
$\Gamma_{i+1} \leftarrow \{T \subset [K] : |T| = i+1\}$;
for $T \in \Gamma_{i+1}$ do
    $W_{n,T} = \Phi$;
    $\Sigma \leftarrow \{A \subset T : |A| = i\}$;
    for $\tau \in \Sigma$ do
        add a randomly picked $W_{n,\theta} \in \tau$ to $W_{n,T}$;
        $\tau \leftarrow \tau \backslash W_{n,\theta}$;
    end
end
for $k \in [K]$ do
    $Z_k \leftarrow (W_{n,T} : n \in [N], T \in \Gamma_{i+1}, k \in T)$;
    Transmit $\bigcup (W_{n,T} : n \in [N], T \in \Gamma_{i+1}, k \in T) \backslash$
    $\bigcup (W_{n,T} : n \in [N], T \in \Gamma_i, k \in T)$ to $Z_k$;
end
end procedure

procedure DELIVERY($W_1, \dots, W_N, d_1, \dots, d_K$)
$G \leftarrow \{S \subset [K] : |S| = t+1\}$;
$X_{(d_1,\dots,d_K)} \leftarrow (\oplus_{k \in S} W_{d_k, S \backslash k} : S \in G)$ end
  procedure
```

In the algorithm, ξ is lowest common multiple (LCM) of all $\binom{K}{i}$. For example, when $K = 4$, $\xi = 12$ and for $K = 5$, $\xi = 10$. I will give the expression of ξ later.

To better understand the algorithm, I will give an example with $N = K = 4$.



$W_{n,\{1\}}$	$W_{n,\{2\}}$	$W_{n,\{3\}}$	$W_{n,\{4\}}$
1,2,3	4,5,6	7,8,9	10,11,12
$W_{n,1}, W_{n,2}, W_{n,3}$	$W_{n,4}, W_{n,5}, W_{n,6}$	$W_{n,7}, W_{n,8}, W_{n,9}$	$W_{n,10}, W_{n,11}, W_{n,12}$

$W_{n,\{1,2\}}$	$W_{n,\{1,3\}}$	$W_{n,\{2,3\}}$	$W_{n,\{1,4\}}$	$W_{n,\{2,4\}}$	$W_{n,\{3,4\}}$
$W_{n,\{1\}}$: ①,2,3	$W_{n,\{1\}}$: 1,②,3	$W_{n,\{2\}}$: 4,⑤,6	$W_{n,\{1\}}$:1,2,③	$W_{n,\{2\}}$: 4,5,⑥	$W_{n,\{3\}}$: 7,8,⑨
$W_{n,\{2\}}$: ④,5,6	$W_{n,\{3\}}$: ⑦,8,9	$W_{n,\{3\}}$: 7,⑧,9	$W_{n,\{4\}}$: ⑩,11,12	$W_{n,\{4\}}$: 10,⑪,12	$W_{n,\{4\}}$: 10,11,⑫
$W_{n,1}, W_{n,4}$	$W_{n,2}, W_{n,7}$	$W_{n,5}, W_{n,8}$	$W_{n,3}, W_{n,10}$	$W_{n,6}, W_{n,11}$	$W_{n,9}, W_{n,12}$

$W_{n,\{1,2,3\}}$	$W_{n,\{1,2,4\}}$	$W_{n,\{1,3,4\}}$	$W_{n,\{2,3,4\}}$
$W_{n,\{1,2\}}$: ①,4	$W_{n,\{1,2\}}$: 1,④	$W_{n,\{1,3\}}$: 2,⑦	$W_{n,\{2,3\}}$: 5,⑧
$W_{n,\{1,3\}}$: ②,7	$W_{n,\{1,4\}}$: ③,10	$W_{n,\{1,4\}}$: 3,⑩	$W_{n,\{2,4\}}$: 6,⑪
$W_{n,\{2,3\}}$: ⑤,8	$W_{n,\{2,4\}}$: ⑥,11	$W_{n,\{3,4\}}$: ⑨,12	$W_{n,\{2,3\}}$: 9,⑫
$W_{n,1}, W_{n,2}, W_{n,5}$	$W_{n,3}, W_{n,4}, W_{n,6}$	$W_{n,7}, W_{n,9}, W_{n,10}$	$W_{n,8}, W_{n,11}, W_{n,12}$

Figure 3: An example of caching process. N = K = 4 and M increases from 1 to 3.

The expression of ξ is

$$\prod_{p \leq K; p \in \mathbf{N}} \frac{p^{[\log_p K]}}{(p^{[\log_p K]}, K+1)}$$

where (a, b) means the greatest common divisor of a and b.

4. Future Work

We suppose the situation that t is an integer, however, in reality, t may not always be an integer. A future work can design a continuous transmission strategy.

Sometimes, the files don't share same priority. Knowing the popularity distribution of the files, we can populate the caches, such as to minimize the expected load of the shared link. In this situation, how to transmit files to time-variant cache still remains a problem.

5. Conclusion

In this project, we design a tranmission algorithm to support the coded caching method so that it can be used in a time-increasing cache, which satisifies reality much more. To fulfill this algorithm, we also need to calculate the lowest common multiple and split each file into such many pieces. During the placement phase, we split files and place parts of them in users. After that, cache increases with time. We calculate the parts to transmit and transmit them to every user so that the coded caching strategy stays available. At last, when link is cut, cache process is over. During the delivery phase, the server delivery certain files to each user according to their demands.

References

[1] [M. A. Maddah-Ali and U. Niesen](2014). *Fundamental limits of caching. IEEE Trans. Inf. Theory, vol. 60, no. 5, pp. 28562867, May 2014.*