Fundamental Limits of Coded Caching: From Uncoded Prefetching to Coded Prefetching

Kai Zhang and Chao Tian

Texas A&M University

ISIT 2018

Caching and Its Applications

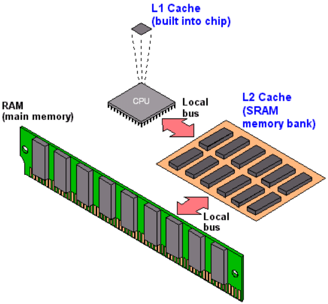
A natural data management strategy

- Prefetching data to local/fast memory space;
- Single user setting, e.g. RAM in computers, on-CPU caches;
- Multiple user setting, e.g. networked system.

Caching and Its Applications

A natural data management strategy

- Prefetching data to local/fast memory space;
- Single user setting, e.g. RAM in computers, on-CPU caches;
- Multiple user setting, e.g. networked system.



Caching and Its Applications

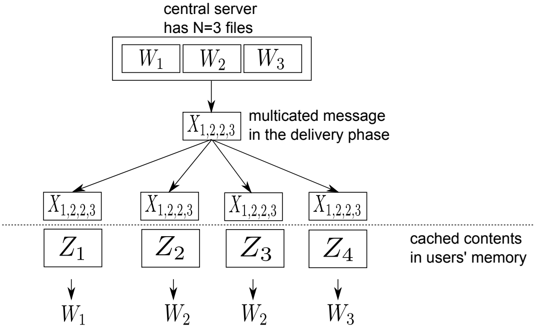
A natural data management strategy

- Prefetching data to local/fast memory space;
- Single user setting, e.g. RAM in computers, on-CPU caches;
- Multiple user setting, e.g. networked system.

An Information-theoretic Formulation

Proposed by Maddah-Ali & Niesen[1]

- N files of same size, K users, each with local cache of size M;
- Prefetching phase: users fill cache before knowing requests;
- Delivery phase: server multicasts information.

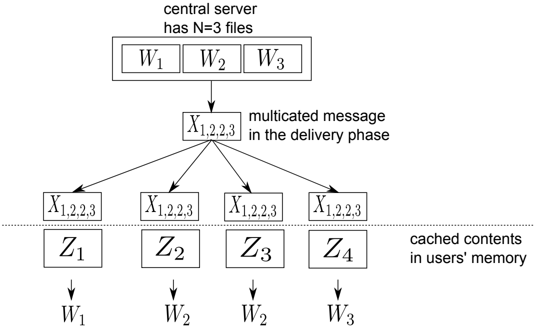


[1] Maddah-Ali & Niesen, "Fundamental limits of caching," TIT-14.

An Information-theoretic Formulation

Proposed by Maddah-Ali & Niesen[1]

- N files of same size, K users, each with local cache of size M;
- Prefetching phase: users fill cache before knowing requests;
- Delivery phase: server multicasts information.



[1] Maddah-Ali & Niesen, "Fundamental limits of caching," TIT-14.

Existing Schemes Using Uncoded Prefetching

Maddah-Ali-Niesen scheme: uncoded prefetching

- Partition each file into $\binom{K}{t}$ segments;
- Each user allocated $\binom{K-1}{t-1}$ uncoded segments.

Example $(N, K) = (3, 4), t = 2$:

User 1	A_{12}	A_{13}	A_{14}	B_{12}	B_{13}	B_{14}	C_{12}	C_{13}	C_{14}
User 2	A_{12}	A_{23}	A_{24}	B_{12}	B_{23}	B_{24}	C_{12}	C_{23}	C_{24}
User 3	A_{13}	A_{23}	A_{34}	B_{13}	B_{23}	B_{34}	C_{13}	C_{23}	C_{34}
User 4	A_{14}	A_{24}	A_{34}	B_{14}	B_{24}	B_{34}	C_{14}	C_{24}	C_{34}

Existing Schemes Using Uncoded Prefetching

Maddah-Ali-Niesen scheme: uncoded prefetching

- Partition each file into $\binom{K}{t}$ segments;
- Each user allocated $\binom{K-1}{t-1}$ uncoded segments.

Example $(N, K) = (3, 4), t = 2$:

User 1	A_{12}	A_{13}	A_{14}	B_{12}	B_{13}	B_{14}	C_{12}	C_{13}	C_{14}
User 2	A_{12}	A_{23}	A_{24}	B_{12}	B_{23}	B_{24}	C_{12}	C_{23}	C_{24}
User 3	A_{13}	A_{23}	A_{34}	B_{13}	B_{23}	B_{34}	C_{13}	C_{23}	C_{34}
User 4	A_{14}	A_{24}	A_{34}	B_{14}	B_{24}	B_{34}	C_{14}	C_{24}	C_{34}

Existing Schemes Using Uncoded Prefetching

Delivery phase: multicast information to fulfill all requests

- Enumerate all $(t+1)$-user set $\mathcal{B}$;
- Each set $\mathcal{B}$ determines a transmission;
- Each transmission contains only one missing symbol for a user.

Example $(N, K) = (3, 4), t = 2$, demand (A, A, B, C)

$$\mathcal{B} = \{1, 2, 3\} : A_{23} + A_{13} + B_{12};$$
$$\mathcal{B} = \{1, 2, 4\} : A_{24} + A_{14} + C_{12};$$
$$\mathcal{B} = \{1, 3, 4\} : A_{34} + B_{14} + C_{13};$$
$$\mathcal{B} = \{2, 3, 4\} : A_{34} + B_{24} + C_{23}.$$

Yu's scheme[1] improved it by removing linearly redundant terms for some (N, K) values.

[1] Yu et al. "The exact rate-memory tradeoff for caching with uncoded prefetching," TIT-17.

Existing Schemes Using Uncoded Prefetching

Delivery phase: multicast information to fulfill all requests

- Enumerate all $(t+1)$-user set $\mathcal{B}$;
- Each set $\mathcal{B}$ determines a transmission;
- Each transmission contains only one missing symbol for a user.

Example $(N, K) = (3, 4), t = 2$, demand (A, A, B, C)

$$\mathcal{B} = \{1, 2, 3\} : A_{23} + A_{13} + B_{12};$$
$$\mathcal{B} = \{1, 2, 4\} : A_{24} + A_{14} + C_{12};$$
$$\mathcal{B} = \{1, 3, 4\} : A_{34} + B_{14} + C_{13};$$
$$\mathcal{B} = \{2, 3, 4\} : A_{34} + B_{24} + C_{23}.$$

Yu's scheme[1] improved it by removing linearly redundant terms for some (N, K) values.

[1] Yu et al. "The exact rate-memory tradeoff for caching with uncoded prefetching," TIT-17.

Existing Schemes Using Uncoded Prefetching

Delivery phase: multicast information to fulfill all requests

- Enumerate all $(t+1)$-user set $\mathcal{B}$;
- Each set $\mathcal{B}$ determines a transmission;
- Each transmission contains only one missing symbol for a user.

Example $(N,K)=(3,4)$, $t=2$, demand (A,A,B,C)

$$\mathcal{B}=\{1,2,3\}: A_{23}+A_{13}+B_{12};$$
$$\mathcal{B}=\{1,2,4\}: A_{24}+A_{14}+C_{12};$$
$$\mathcal{B}=\{1,3,4\}: A_{34}+B_{14}+C_{13};$$
$$\mathcal{B}=\{2,3,4\}: A_{34}+B_{24}+C_{23}.$$

Yu's scheme[1] improved it by removing linearly redundant terms for some (N,K) values.

[1] Yu et al. "The exact rate-memory tradeoff for caching with uncoded prefetching," TIT-17.

Existing Schemes Using Uncoded Prefetching

Delivery phase: multicast information to fulfill all requests

- Enumerate all $(t+1)$-user set $\mathcal{B}$;
- Each set $\mathcal{B}$ determines a transmission;
- Each transmission contains only one missing symbol for a user.

Example $(N, K) = (3, 4), t = 2$, demand (A, A, B, C)

$$\mathcal{B} = \{1, 2, 3\} : A_{23} + A_{13} + B_{12};$$
$$\mathcal{B} = \{1, 2, 4\} : A_{24} + A_{14} + C_{12};$$
$$\mathcal{B} = \{1, 3, 4\} : A_{34} + B_{14} + C_{13};$$
$$\mathcal{B} = \{2, 3, 4\} : A_{34} + B_{24} + C_{23}.$$

Yu's scheme[1] improved it by removing linearly redundant terms for some (N, K) values.

[1] Yu et al. "The exact rate-memory tradeoff for caching with uncoded prefetching," TIT-17.

Existing Schemes Using Uncoded Prefetching

Delivery phase: multicast information to fulfill all requests

- Enumerate all $(t+1)$-user set $\mathcal{B}$;
- Each set $\mathcal{B}$ determines a transmission;
- Each transmission contains only one missing symbol for a user.

Example $(N, K) = (3, 4), t = 2$, demand (A, A, B, C)

$$\begin{aligned}
\mathcal{B} = \{1,2,3\} &: A_{23} + A_{13} + B_{12}; \\
\mathcal{B} = \{1,2,4\} &: A_{24} + A_{14} + C_{12}; \\
\mathcal{B} = \{1,3,4\} &: A_{34} + B_{14} + C_{13}; \\
\mathcal{B} = \{2,3,4\} &: A_{34} + B_{24} + C_{23}.
\end{aligned}$$

Yu's scheme[1] improved it by removing linearly redundant terms for some (N, K) values.

[1] Yu et al. "The exact rate-memory tradeoff for caching with uncoded prefetching," TIT-17.

Existing Schemes Using Coded Prefetching

T. and Chen[1] proposed a scheme using coded prefetching

- Partition each file into $\binom{K}{t}$ segments;
- User encodes allocated file segments using rank metric code;
- The parity symbols are stored.

Delivery phase uses MDS code

- Cached contents $\perp$ delivered transmissions;
- Each transmission contains segments from only one file.

Example: $(N, K) = (3, 4)$, demand = (A,A,B,C), transmits:

$$A_{34},\ B_{12},\ B_{14},\ B_{24},\ C_{12},\ C_{13},\ C_{23},\ A_{13} + A_{23},\ A_{14} + A_{24}$$

$A_{14} + A_{24}$: needed by users 1&2; help resolve symbols at user 4.

[1] T.&Chen, "Caching and delivery via interference elimination," TIT-18.

Existing Schemes Using Coded Prefetching

T. and Chen[1] proposed a scheme using coded prefetching

- Partition each file into $\binom{K}{t}$ segments;
- User encodes allocated file segments using rank metric code;
- The parity symbols are stored.

Delivery phase uses MDS code

- Cached contents $\perp$ delivered transmissions;
- Each transmission contains segments from only one file.

Example: $(N, K) = (3, 4)$, demand = (A,A,B,C), transmits:

$$A_{34},\ B_{12},\ B_{14},\ B_{24},\ C_{12},\ C_{13},\ C_{23},\ A_{13} + A_{23},\ A_{14} + A_{24}$$

$A_{14} + A_{24}$: needed by users 1&2; help resolve symbols at user 4.

[1] T.&Chen, "Caching and delivery via interference elimination," TIT-18.

Existing Schemes Using Coded Prefetching

T. and Chen[1] proposed a scheme using coded prefetching

- Partition each file into $\binom{K}{t}$ segments;
- User encodes allocated file segments using rank metric code;
- The parity symbols are stored.

Delivery phase uses MDS code

- Cached contents $\perp$ delivered transmissions;
- Each transmission contains segments from only one file.

Example: $(N, K) = (3, 4)$, demand $=$ (A,A,B,C), transmits:

$$A_{34},\ B_{12},\ B_{14},\ B_{24},\ C_{12},\ C_{13},\ C_{23},\ A_{13} + A_{23},\ A_{14} + A_{24}$$

$A_{14} + A_{24}$: needed by users 1&2; help resolve symbols at user 4.

[1] T.&Chen, "Caching and delivery via interference elimination," TIT-18.

Existing Schemes Using Coded Prefetching

T. and Chen[1] proposed a scheme using coded prefetching

- Partition each file into $\binom{K}{t}$ segments;
- User encodes allocated file segments using rank metric code;
- The parity symbols are stored.

Delivery phase uses MDS code

- Cached contents $\perp$ delivered transmissions;
- Each transmission contains segments from only one file.

Example: $(N, K) = (3, 4)$, demand = (A,A,B,C), transmits:

$$A_{34},\ B_{12},\ B_{14},\ B_{24},\ C_{12},\ C_{13},\ C_{23},\ A_{13} + A_{23},\ A_{14} + A_{24}$$

$A_{14} + A_{24}$: needed by users 1&2; help resolve symbols at user 4.

[1] T.&Chen, "Caching and delivery via interference elimination," TIT-18.

Two Quite Different Schemes

Uncode prefetching:

- Binary code
- Simple combinatorics
- Better at high memory regime

Coded prefetching:

- Non-binary code
- MDS and rank metric codes
- Better at low memory regime

Are there any connections?

Two Quite Different Schemes

Uncode prefetching:

- Binary code
- Simple combinatorics
- Better at high memory regime

Coded prefetching:

- Non-binary code
- MDS and rank metric codes
- Better at low memory regime

Are there any connections?

A Hidden Connection

Putting them side-by-side

- Maddah-Ali Niesen scheme transmits 4 symbols:

$$A_{23} + A_{13} + B_{12}, \quad A_{24} + A_{14} + C_{12}$$
$$A_{34} + B_{14} + C_{13}, \quad A_{34} + B_{24} + C_{23}$$

$$\Downarrow$$

- Tian-Chen scheme transmits 9 symbols:

$$A_{23} + A_{13}, \quad B_{12}, \qquad A_{24} + A_{14}, \quad C_{12}$$
$$A_{34}, \quad B_{14}, \quad C_{13}, \qquad B_{24}, \quad C_{23}$$

Question: Will other decomposition give better performance?

A Hidden Connection

Putting them side-by-side

- Maddah-Ali Niesen scheme transmits 4 symbols:

$$
\begin{array}{ll}
A_{23} + A_{13} + B_{12}, & A_{24} + A_{14} + C_{12} \\
A_{34} + B_{14} + C_{13}, & A_{34} + B_{24} + C_{23}
\end{array}
$$

$$\Downarrow$$

- Tian-Chen scheme transmits 9 symbols:

$$
\begin{array}{llllll}
A_{23} + A_{13}, & B_{12}, & & A_{24} + A_{14}, & C_{12} \\
A_{34}, \quad B_{14}, & C_{13}, & & B_{24}, & C_{23}
\end{array}
$$

Question: Will other decomposition give better performance?

Yes! A New Scheme for $(N, K) = (3, 4)$

A partial decomposition scheme $(N, K) = (3, 4), t = 2$,

- Demand (A,A,B,C), transmits

$$\begin{array}{ll} A_{23} + A_{13} + B_{12}, & A_{24} + A_{14} + C_{12} \\ A_{34}, \quad B_{14} + C_{13}, & B_{24} + C_{23} \end{array}$$

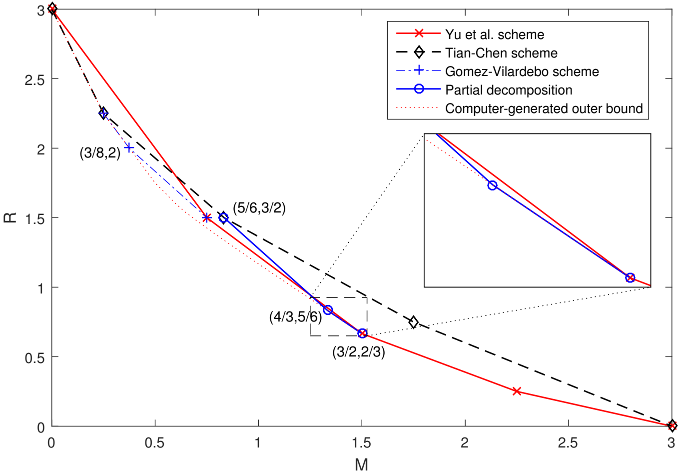
  i.e. only the last two transmissions, file A is decomposed out.

- Demand (A,A,B,B), transmits

$$\begin{array}{ll} A_{23} + A_{13} + B_{12}, & A_{24} + A_{14} + B_{12} \\ A_{34}, \quad B_{14} + B_{13}, & B_{24} + B_{23} \end{array}$$

- ....

Performance of the New Scheme



- Produces $(M, R) = (\frac{4}{3}, \frac{5}{6})$: a new corner point and optimal.

The General Idea

We know the extremes:

- No decomposition: Maddah-Ali-Niesen's (Yu's) transmissions;
- Full decomposition: Tian-Chen's transmissions.

Partial decomposition of Maddah-Ali-Niesen transmissions:

- No. of transmissions will increase;
- No. of cached linear combinations can decrease (how much?);
- Code design: need to guarantee decodability for all demands.

Question: how to best decompose the transmissions?

The General Idea

We know the extremes:

- No decomposition: Maddah-Ali-Niesen's (Yu's) transmissions;
- Full decomposition: Tian-Chen's transmissions.

Partial decomposition of Maddah-Ali-Niesen transmissions:

- No. of transmissions will increase;
- No. of cached linear combinations can decrease (how much?);
- Code design: need to guarantee decodability for all demands.

Question: how to best decompose the transmissions?

The General Idea

We know the extremes:

- No decomposition: Maddah-Ali-Niesen's (Yu's) transmissions;
- Full decomposition: Tian-Chen's transmissions.

Partial decomposition of Maddah-Ali-Niesen transmissions:

- No. of transmissions will increase;
- No. of cached linear combinations can decrease (how much?);
- Code design: need to guarantee decodability for all demands.

Question: how to best decompose the transmissions?

Transmission Types

Transmission type: a new notion

- Defined as $\boldsymbol{t} = (t_1, \ldots, t_N)$;
- t_n: number of users requesting file W_n;
- Each $\boldsymbol{t}$ is associated with one or more transmissions.

Example: $(N, K) = (3, 4), t = 2, \boldsymbol{d} = (A, A, B, C)$

$$
\begin{aligned}
&A_{23} + A_{13} + B_{12}, \quad \boldsymbol{t} = (2,1,0) \\
&A_{24} + A_{14} + C_{12}, \quad \boldsymbol{t} = (2,0,1) \\
&\left.\begin{aligned} A_{34} + B_{14} + C_{13} \\ A_{34} + B_{24} + C_{23} \end{aligned}\right\}, \boldsymbol{t} = (1,1,1)
\end{aligned}
$$

Decomposing a Transmission Type

Transmissions of the same type are decomposed the same way:

Example: Two transmissions of the same type

$$A_{34} + B_{14} + C_{13}, A_{34} + B_{24} + C_{23}.$$

- Full decomposition pattern $\{\{A\}, \{B\}, \{C\}\}$:

$$A_{34},\ B_{14},\ C_{13};\quad A_{34},\ B_{24},\ C_{23}.$$

- A partial decomposition pattern $\{\{A, C\}, \{B\}\}$:

$$A_{34} + C_{13},\ B_{14};\quad A_{34} + C_{23},\ B_{24}.$$

Joint Coding over Multiple Instances

We code across multiple $(= r)$ code instances:

- Each instance is $N\binom{K}{t}$ segments from all files;
- Encoding are done jointly over multiple instances;
- Apply a decomposition pattern on some instances;
- Decoding are done jointly;
- Similar as "time-sharing", but not equivalent.

Benefit: reduce unevenness in users' useful transmission collection.

Joint Coding over Multiple Instances

We code across multiple $(= r)$ code instances:

- Each instance is $N\binom{K}{t}$ segments from all files;
- Encoding are done jointly over multiple instances;
- Apply a decomposition pattern on some instances;
- Decoding are done jointly;
- Similar as "time-sharing", but not equivalent.

Benefit: reduce unevenness in users' useful transmission collection.

Joint Coding over Multiple Instances

We code across multiple $(= r)$ code instances:

- Each instance is $N\binom{K}{t}$ segments from all files;
- Encoding are done jointly over multiple instances;
- Apply a decomposition pattern on some instances;
- Decoding are done jointly;
- Similar as "time-sharing", but not equivalent.

Benefit: reduce unevenness in users' useful transmission collection.

Revisiting the New Code Example

Example $(N, K) = (3, 4), t = 2, r = 6$, demand $\boldsymbol{d} = (A, A, B, B)$

- Pattern-1: on 3 instances, 6 transmissions each

$$\{\{A\}, \{B\}\} : A_{23} + A_{13},\ B_{12}; A_{24} + A_{14},\ B_{12};$$
$$\{\{A\}, \{B\}\} : A_{34},\ B_{14} + B_{13}; A_{34},\ B_{24} + B_{23}.$$

- Pattern-2: on 3 instances, 4 transmissions each

$$\{\{A, B\}\} : A_{23} + A_{13} + B_{12}; A_{24} + A_{14} + B_{12};$$
$$\{\{A, B\}\} : A_{24} + A_{14} + B_{12}; A_{34} + B_{24} + B_{23}.$$

Revisiting the New Code Example

Demand	(A,A,B,C)	(A,A,B,B)		(A, A, A, B)		(A, A, A, A)	
Pattern	1	1	2	1	2	1	2
No. of transmissions	5	4	6	4	6	3	12
No. of Instances	6	3	3	3	3	5	1
Fraction	1	1/2	1/2	1/2	1/2	5/6	1/6

- Coding over $r = 6$ instances;
- The (M, R) pair $(\frac{4}{3}, \frac{5}{6})$ can be achieved by all demands.

A Linear Program

Compute optimal decomposition patterns by linear programming

$$\sum_{\text{all } \mathcal{P}_{\boldsymbol{d}}^{(t)}} \alpha_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} = 1, \quad \boldsymbol{d} \in \mathcal{D},$$

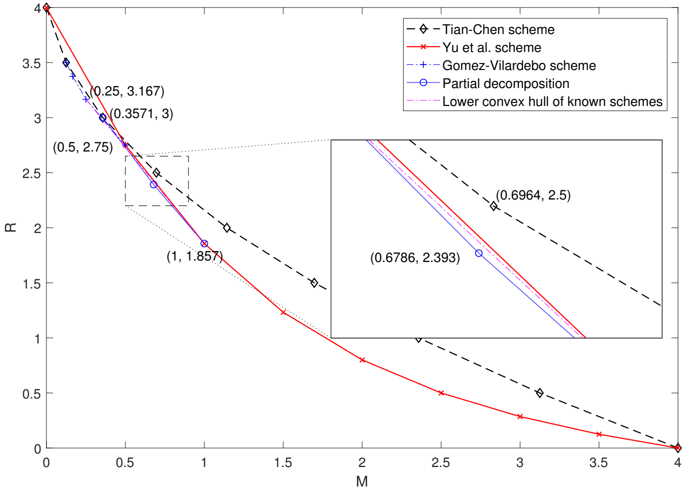
$$1 \geq \alpha_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} \geq 0, \quad \boldsymbol{d} \in \mathcal{D}, \quad \text{for all } \mathcal{P}_{\boldsymbol{d}}^{(t)},$$

$$\sum_{\text{all } \mathcal{P}_{\boldsymbol{d}}^{(t)}} \alpha_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} R_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} \leq R\binom{K}{t}, \quad \boldsymbol{d} \in \mathcal{D},$$

$$\sum_{\text{all } \mathcal{P}_{\boldsymbol{d}}^{(t)}} \alpha_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} M_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)},k} \leq M\binom{K}{t}, \quad \boldsymbol{d} \in \mathcal{D}, k \in [1:K].$$

- Representing combinations of decomposition patterns $\mathcal{P}_{\boldsymbol{d}}^{(t)}$;
- Optimize over the proportions of the decomposition patterns;
- Find the best (M, R) achieved by all demands.

A New Information-Theoretic Inner Bound



A new information-theoretic inner-bound for caching system $(N, K) = (4, 8)$

Conclusion

Summary of this work

- A single scheme unifying two general classes of schemes;
- Yu scheme and Tian-Chen scheme are two extremes;
- Transmission type plays an important role;
- Performance improvement not large: intermediate M regime.

Future work

- The proposed scheme does not include Gómez-Vilardebó[1]'s scheme: a more generalized code may exist?
- Simplify linear programming: identify and remove those "bad" decomposing patterns?

[1] J. Gómez-Vilardebó, "Fundamental limits of caching: Improved bounds with coded prefetching," *arXiv:1612.09071*, Dec. 2016.