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

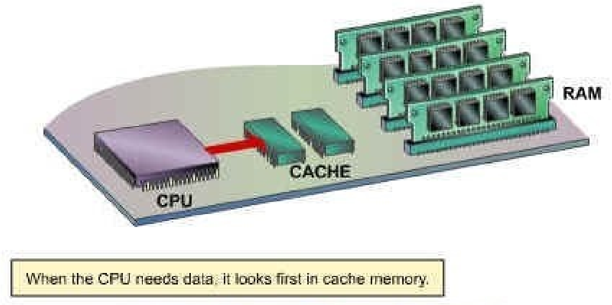
Kai Zhang and Chao Tian

Texas A&M University

ISIT 2018

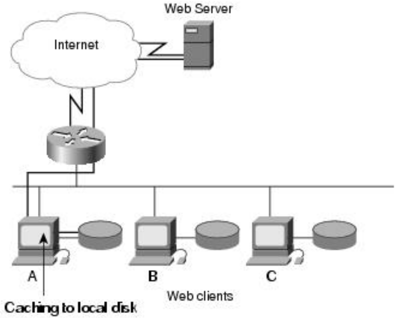
Coded Caching and Its Applications

- A data management strategy to reduce delay during peak-traffic time,
- Single user setting, e.g. on-CPU caches, RAM in computers,
- Multiple user setting, e.g. networked system.



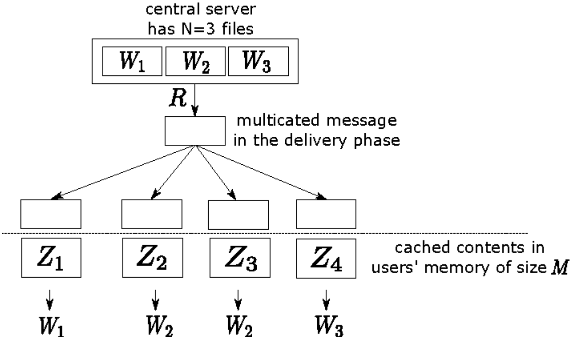
Coded Caching and Its Applications

- A data management strategy to reduce delay during peak-traffic time,
- Single user setting, e.g. on-CPU caches, RAM in computers,
- Multiple user setting, e.g. networked system.



An Information-theoretic Formulation

- N files of same size, K users, each with a local cache of size M,
- Prefetching phase: users fill their cache without knowing the requests,
- Delivery phase: each user requests a single file, server multicasts information to accommodate the requests.



Existing Schemes Using Uncoded Prefetching

Maddah-Ali & Niesen[1], improved by Yu et al[2], proved being optimal,

- Prefetching strategy:
  - Partition each file into $\binom{K}{t}$ segments, each associated with a cardinality-t subset,
  - Each user caches $\binom{K-1}{t-1}$ segments,
  - E.g.: $(N, K) = (3, 4)$, $t = 2$,

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}

- Delivery strategy
  - For any $(t + 1)$ users subset $\mathcal{B}$, send the XOR of requested segments.

---

[1] Maddah-Ali M A, Niesen U. Fundamental limits of caching[J]. IEEE Transactions on Information Theory, 2014, 60(5): 2856-2867.

[2] Yu Q, Maddah-Ali M A, Avestimehr A S. The exact rate-memory tradeoff for caching with uncoded prefetching[J]. IEEE Transactions on Information Theory, 2017.

Existing Schemes Using Uncoded Prefetching

- Server transmits:

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

Using a brief moment, let's look at the decoding steps for the 1st user, when demand is (A,A,B,C):

Existing Schemes Using Uncoded Prefetching

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

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

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}

Step 1: using A_{13} and B_{12} to decode A_{23};

Existing Schemes Using Uncoded Prefetching

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

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

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}

Step 2: using A_{14} and C_{12} to decode A_{24};

Existing Schemes Using Uncoded Prefetching

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

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

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}

Step 3: using B_{14} and C_{13} to decode A_{34};
User 1 now decodes the entire file A.

Existing Schemes Using Coded Prefetching

Tian-Chen scheme[1],

- Prefetching strategy:
  - A file is partitioned into $\binom{K}{t}$ segments, each user allocated $\binom{K-1}{t-1}$ segments,
  - User encodes file segments using a rank metric code and stores the parities,
  - E.g.: $(N, K) = (3, 4), t = 2$, user 1 caches

$$\begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}_{(5\times 9)} \cdot$$

$$(A_{12}, A_{13}, A_{14}, B_{12}, B_{13}, B_{14}, C_{12}, C_{13}, C_{14})^T$$

In the example, a (14,9) systematic rank metric code is used, user 1 stores the 5 parities.

[1] Tian C, Chen J. Caching and delivery via interference elimination[C]. Information Theory (ISIT), 2016 IEEE International Symposium on. IEEE, 2016: 830-834.

Existing Schemes Using Coded Prefetching

- Delivery strategy: Linear combination of segments from one file.
  - E.g.: demand (A,A,B,C), transmits 9 (coded) symbols:

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

  - Each symbol serves two roles:
    1) content delivery for some users;
    2) help resolve coded symbols for some other users.

To be clear, let's check user 1,

Existing Schemes Using Coded Prefetching

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

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

$$\begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ & & & \cdot & & & & & \\ & & & & & \cdot & & & \\ & & & & & & \cdot & & \\ & & & & & & & \cdot & \end{pmatrix}_{(5\times9)} \qquad \cdot$$

$$(A_{12}, A_{13}, A_{14}, B_{12}, B_{13}, B_{14}, C_{12}, C_{13}, C_{14})^T$$

Step 1: user 1 collects 4 symbols $B_{12}, B_{14}, C_{12}, C_{13}$;
Together with 5 (coded) symbols in his cache, he can decode all 9 symbols.

Existing Schemes Using Coded Prefetching

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

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

Step 2: user 1 already resolved all symbols in the cache, as following:

User 1	A_{12}	A_{13}	A_{14}	B_{12}	B_{13}	B_{14}	C_{12}	C_{13}	C_{14}

He then collects $A_{13} + A_{23}, A_{14} + A_{24}$, he can decode A_{23}, A_{24};

Existing Schemes Using Coded Prefetching

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

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

Step 3: user 1 collects A_{34}, now he decodes the entire file A;

Two Quite Different Schemes

- Uncoded prefetching *vs.* coded prefetching,
- Binary code *vs.* non-binary code,
- Encoding/decoding using simple combinatorics *vs.* sophisticated coding techniques (rank metric codes),
- Uncoded prefetching performs better at high cache memory regime *vs.* coded prefetching performs better at low cache memory regime.

Largely unrelated...
Are there any connections?

Some Backgrounds

- Linearized Polynomial: A degree-q^{P-1} linearized polynomial in the finite field $\mathbb{F}_{q^m}$

$$f(x) = \sum_{i=1,2,\ldots,P} v_i x^{q^{i-1}}, v_i \in \mathbb{F}_{q^m} \quad (1)$$

can be uniquely identified from evaluations at any P points $x = \theta_i \in \mathbb{F}_{q^m}, i = 1, 2, \ldots, P$, that are linearly independent over $\mathbb{F}_q$.

Some Backgrounds

Lemma

Let $f(x)$ be a degree-q^{P-1} linearized polynomial in $\mathbb{F}_{q^m}$, and $\theta_i \in \mathbb{F}_{q^m}, i = 1, 2, \ldots, P_o$, be linearly independent over $\mathbb{F}_q$. Let G be a $P_o \times P$ full rank (rank P) matrix with entries in $\mathbb{F}_q$, then $f(x)$ can be uniquely identified from

$$[f(\theta_1), f(\theta_2), \ldots, f(\theta_{P_o})] \cdot G. \tag{2}$$

- $(v_1, \ldots, v_P)$ are information symbols to be encoded; $[f(\theta_1), \ldots, f(\theta_{P_o})]$ are coded symbols;
- A (P_o, P) MDS code in terms of rank metric.

A Hidden Connection

Continuing example $(N, K) = (3, 4)$, $t = 2$, demand = (A,A,B,C):

- Yu et al scheme transmits (4 symbols):

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

Separate different files & remove repeated transmissions,

$$\Downarrow$$

- Tian-Chen scheme (9 symbols):

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

Instead of fully decomposing a transmission, what if we partially decompose them?

An Alternative Scheme

- A partial decomposition: deliver 5 coded symbols:
  - Demand (A,A,B,C), transmits

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

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

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

An Alternative Scheme

- A partial decomposition: deliver 5 coded symbols:
  - Demand (A,A,A,C), transmits

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

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

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

An Alternative Scheme

- Each user only needs to cache 8 linear combinations,
- Every transmission could have different ways to be partially decomposed, i.e. *decomposition pattern*s,
- Above is a good choice of decomposing, in fact produces a new corner point at $(M, R) = (4/3, 5/6)$.

We will give a generalized scheme and code examples.

Partial Decomposition and A New Code Example

To facilitate understanding, we introduce some notions:

- Demand vector $\boldsymbol{d} = \{d_1, \ldots, d_K\}$ denotes the demands by K users, $d_k \in [1:N]$,
- Each transmission in Yu scheme can be represented as

$$\oplus_{k \in \mathcal{B}} W_{d_k, \mathcal{B} \setminus k}, \quad |\mathcal{B}| = t + 1,$$

a transmission type $\boldsymbol{t} = (t_1, \ldots, t_N)$ is associated with a transmission where t_n denotes the number of users in subset $|\mathcal{B}|$ requesting file W_n. Set $\mathcal{T}_{\boldsymbol{d}}^{(t)}$ denotes all valid transmission types for a demand $\boldsymbol{d}$.

Partial Decomposition and A New Code Example

- A partial decomposition $\mathcal{P}_{\boldsymbol{t},\boldsymbol{d}}$ of transmission type $\boldsymbol{t}$ is a partition of $\mathbb{supp}(\boldsymbol{t})$. The full set of decomposition patterns is $\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)} = \{\mathcal{P}_{\boldsymbol{t},\boldsymbol{d}} | \boldsymbol{t} \in \mathcal{T}_{\boldsymbol{d}}^{(t)}\}$.
- The collection of all possible $\mathcal{P}_{\boldsymbol{t},\boldsymbol{d}}$ is $\mathfrak{P}_{\boldsymbol{t},\boldsymbol{d}}$.
- E.g.: $(N, K) = (3, 4)$, $t = 2$, $\boldsymbol{d} = (1, 1, 2, 3)$, $\mathcal{T}_{\boldsymbol{d}}^{(t)} = \{(2, 1, 0), (2, 0, 1), (1, 1, 1)\}$, for $\boldsymbol{t} = (1, 1, 1)$:

  $$\begin{array}{lcl} \mathcal{B} = \{1,3,4\} : A_{34} + B_{14} + C_{13} & \xrightarrow{\text{decompose}} & \{A_{34}, B_{14} + C_{13}\} \\ \mathcal{B} = \{2,3,4\} : A_{34} + B_{24} + C_{23} & & \{A_{34}, B_{24} + C_{23}\} \end{array}$$

  Decomposition pattern $\mathcal{P}_{\boldsymbol{t},\boldsymbol{d}} = \mathcal{P}_{(1,1,1)(1,1,2,3)} = \{\{1\}, \{2, 3\}\}$.

Partial Decomposition and A New Code Example

Partial decomposition could produce new memory-rate points:

- For each demand $\boldsymbol{d}$, using multiple coding instances,
- Fix a decomposition pattern $\mathcal{P}_{\boldsymbol{t},\boldsymbol{d}}$ for each $\boldsymbol{t}$,
- Set an auxiliary variable $\alpha_{\boldsymbol{d},\mathcal{P}^{(t)}_{\boldsymbol{d}}}$ for each decomposition pattern, s.t.

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

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

- $\alpha_{\boldsymbol{d},\mathcal{P}^{(t)}_{\boldsymbol{d}}}$: how much fraction of instances using a decomposition pattern;
- Intuitively same as "time sharing", but not equivalent.

Partial Decomposition and A New Code Example

- Prefetching:
  - Partition each file into $r\binom{K}{t}$ segments (r instances each of $\binom{K}{t}$ segments), each to be in $\mathbb{F}_{2^m}$ for sufficiently large m,
  - Each user is allocated $P = rN\binom{K-1}{t-1}$ symbols; caches $P_o - P$ linear combinations:

    $$P_o - P = rM'_r\binom{K}{t}$$

    (M'_r to be defined later)
  - Using (P_o, P) systematic rank metric code to encode P symbols, caches the parities;
  - $m \geq P_o$ suffices for existence of rank metric code.

Partial Decomposition and A New Code Example

- Delivery:
  - For demand $\boldsymbol{d}$ = (A,A,B,C)=(1,1,2,3), let $\alpha_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}} = 1$, $\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)} = \boldsymbol{\mathcal{P}}_{(1,1,2,3)}^{(2)}$ are:

$$
\begin{aligned}
\mathcal{P}_{(2,1,0),(1,1,2,3)} &= \{\{1,2\}\} = \{\{A,B\}\} : A_{23} + A_{13} + B_{12}, \\
\mathcal{P}_{(2,0,1),(1,1,2,3)} &= \{\{1,3\}\} = \{\{A,C\}\} : A_{24} + A_{14} + C_{12}, \\
\mathcal{P}_{(1,1,1),(1,1,2,3)} &= \{\{1\},\{2,3\}\} = \{\{A\},\{B,C\}\} : \\
&\qquad A_{34}, \quad B_{14} + C_{13}, \; B_{24} + C_{23}
\end{aligned}
$$

Thus $R_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}} = 5, M_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)},k} = 8$, for $k \in [1:4]$.

Partial Decomposition and A New Code Example

- Delivery:
  - For demand $\boldsymbol{d} = $ (A,A,B,B)=(1,1,2,2), two decomposition patterns:
    1st pattern: $\mathcal{P}_{\boldsymbol{d}}^{(t)}$ without any decomposition, let $\alpha_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} = 1/2$,

$$\begin{aligned} &A_{23}^{(1)} + A_{13}^{(1)} + B_{12}^{(1)}, \quad A_{24}^{(1)} + A_{14}^{(1)} + B_{12}^{(1)} \\ &A_{34}^{(1)} + B_{14}^{(1)} + B_{13}^{(1)}, \quad A_{34}^{(1)} + B_{24}^{(1)} + B_{23}^{(1)} \end{aligned}$$

    For this pattern, $R_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} = 4$, $M_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)},k} = 9$, $k \in [1:4]$.

Partial Decomposition and A New Code Example

- Delivery:
  - For demand $\boldsymbol{d}$ = (A,A,B,B)=(1,1,2,2), two decomposition patterns:
    2nd pattern: the following decompositions $\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}$:

$$\mathcal{P}_{(2,1,0),(1,1,2,2)} = \{\{1\},\{2\}\} = \{\{A\},\{B\}\} : A_{23}^{(2)} + A_{13}^{(2)}, B_{12}^{(2)}, \\ A_{24}^{(2)} + A_{14}^{(2)}, B_{12}^{(2)};$$

$$\mathcal{P}_{(1,2,0),(1,1,2,2)} = \{\{1\},\{2\}\} = \{\{A\},\{B\}\} : A_{34}^{(2)}, B_{14}^{(2)} + B_{13}^{(2)}, \\ A_{34}^{(2)}, B_{24}^{(2)} + B_{23}^{(2)};$$

For this pattern, $R_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}} = 6, M_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)},k} = 7$, for $k \in [1:4]$.

Partial Decomposition and A New Code Example

- Delivery:
  - For demand $\boldsymbol{d} = $ (A,A,A,A)=(1,1,1,1), two decomposition patterns:
    1st pattern: $\mathcal{P}_{\boldsymbol{d}}^{(t)}$ without any decomposition, let $\alpha_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} = 5/6$,

$$\begin{aligned} &A_{23}^{(i)} + A_{13}^{(i)} + A_{12}^{(i)}, \quad A_{24}^{(i)} + A_{14}^{(i)} + A_{12}^{(i)} \\ &A_{34}^{(i)} + A_{14}^{(i)} + A_{13}^{(i)}, \quad \text{for } i = 1,2,3,4,5. \end{aligned}$$

    For this pattern, $R_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} = 3$, $M_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)},k} = 9$, $k \in [1:4]$.

Partial Decomposition and A New Code Example

- Delivery:
  - For demand $\boldsymbol{d}$ = (A,A,A,A)=(1,1,1,1), two decomposition patterns:

    2nd pattern: special uncoded decomposition pattern $\breve{\mathcal{P}}^{(t)}_{\boldsymbol{d}}$, let $\alpha_{\boldsymbol{d},\mathcal{P}^{(t)}_{\boldsymbol{d}}} = 1/6$,

$$A^{(6)}_{23}, A^{(6)}_{13}, A^{(6)}_{24}, A^{(6)}_{14}, A^{(6)}_{12}, A^{(6)}_{34}$$
$$B^{(6)}_{23}, B^{(6)}_{13}, B^{(6)}_{24}, B^{(6)}_{14}, B^{(6)}_{12}, B^{(6)}_{34}$$

For this pattern, $R_{\boldsymbol{d},\mathcal{P}^{(t)}_{\boldsymbol{d}}} = 12, M_{\boldsymbol{d},\mathcal{P}^{(t)}_{\boldsymbol{d}},k} = 3, k \in [1:4]$.

Partial Decomposition and A New Code Example

- The above example uses $r = 6$ instances of code,
- Of all 6 instances, $r_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}}$ of them adopt decomposition $\mathcal{P}_{\boldsymbol{d}}^{(t)}$,
- Memory-rate pair

$$(M, R) = \frac{1}{r\binom{K}{t}} \left(\sum_{\mathcal{P}_{\boldsymbol{d}}^{(t)}} r_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} M_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)},k}, \sum_{\mathcal{P}_{\boldsymbol{d}}^{(t)}} r_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} R_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} \right)$$

would be

$$(M, R) = (4/3, 5/6).$$

A New Information-Theoretic Inner Bound

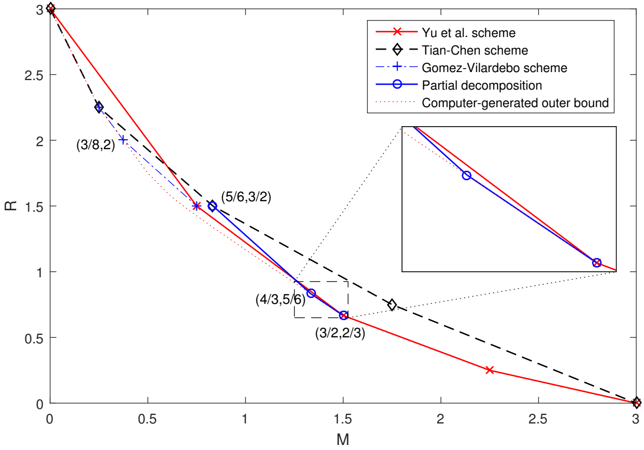


Figure: A new information-theoretic inner-bound for the example caching system $(N, K) = (3, 4)$, $t = 2$.

A New Information-Theoretic Inner Bound

- For some demands, special uncoded transmission needed to keep the decomposition rule;
- As the parameters increase, number of decomposition patterns for each $\boldsymbol{t}$ grows;
- The best decompositions for all $\boldsymbol{t}$? Exhaustive search, practically impossible to calculate by hand;
- We can use a computer-aided approach.

Before the details, we show a new corner point for caching system $(N, K) = (4, 8), t = 2$.

This time it is calculated using a computer program.

A New Information-Theoretic Inner Bound

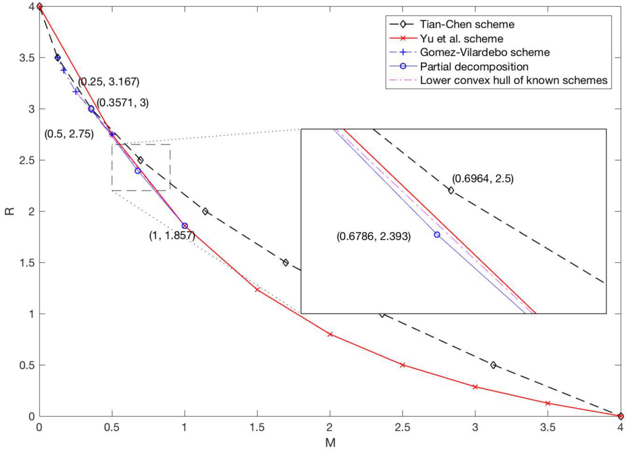


Figure: A new information-theoretic inner-bound for the example caching system $(N, K) = (4, 8)$, $t = 2$.

A Linear Programming Framework

The inner-bound of the new coding scheme can be calculated using linear programming:

- Define rate region $\mathcal{R}^{(t)}$: collection of memory-rate pairs (M, R) such that exists set of $\{\alpha_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}}\}$ satisfying

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

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

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

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

A Linear Programming Framework

The inner-bound of the new coding scheme can be calculated using linear programming:

- Further define

$$(M'_{\boldsymbol{d}}, R'_{\boldsymbol{d}}) = \frac{1}{r\binom{K}{t}} \left(\max_{k \in [1:K]} \sum_{\mathcal{P}_{\boldsymbol{d}}^{(t)}} r_{\boldsymbol{d}, \mathcal{P}_{\boldsymbol{d}}^{(t)}} M_{\boldsymbol{d}, \mathcal{P}_{\boldsymbol{d}}^{(t)}, k}, \sum_{\mathcal{P}_{\boldsymbol{d}}^{(t)}} r_{\boldsymbol{d}, \mathcal{P}_{\boldsymbol{d}}^{(t)}} R_{\boldsymbol{d}, \mathcal{P}_{\boldsymbol{d}}^{(t)}} \right)$$

  and

$$M'_r \triangleq \max_{\boldsymbol{d} \in \mathcal{D}} M'_{\boldsymbol{d}}, \qquad R'_r \triangleq \max_{\boldsymbol{d} \in \mathcal{D}} R'_{\boldsymbol{d}}.$$

A Linear Programming Approach

The inner-bound of the new coding scheme can be calculated using linear programming:

- The number of instances r can be chosen s.t. exists $\{r_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}}\}$

$$\left| \frac{r_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}}}{r} - \alpha_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} \right| \leq \epsilon.$$

By making r large and choosing appropriate $\{r_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}}\}$ s.t. $\epsilon \geq 0$ being arbitrarily small. We have

$$\lim_{r\to\infty} (M'_r, R'_r) = (M, R),$$

it will be the effective memory-rate pair of the new code.

Conclusion

- A single scheme unifying two general classes of schemes (uncoded and coded),
- The Yu et al scheme and Tian-Chen scheme are two extreme points of the new scheme,
- The notion of Transmission type plays an important role and is overlooked before this work,
- The performance improvement is not quite large, and mostly reside in the middle M regime, which are not surprise.

Conclusion

- The new coding scheme can not incorporate the coding scheme of Gómez-Vilardebó[1], a more generalized code may exist?
- The complexity: certain decomposing patterns are obviously bad choice; Simplifying the proposed scheme appears worthwhile.

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