New Codes and Outer Bounds for Caching Systems: A Computer Aided Investigation

Chao Tian

The University of Tennessee Knoxville

August 2016

Based in part on joint work with Jun Chen.

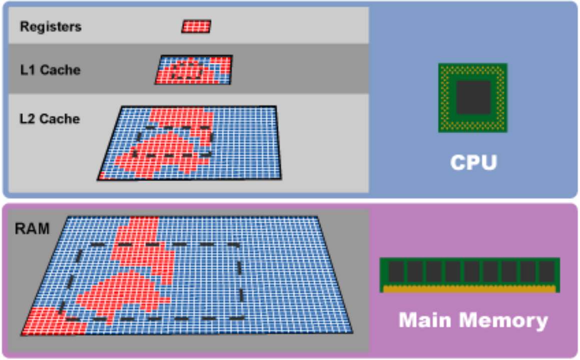
Outline

Outline

Caching and Its Applications

A natural management strategy when communication is bursty or costly

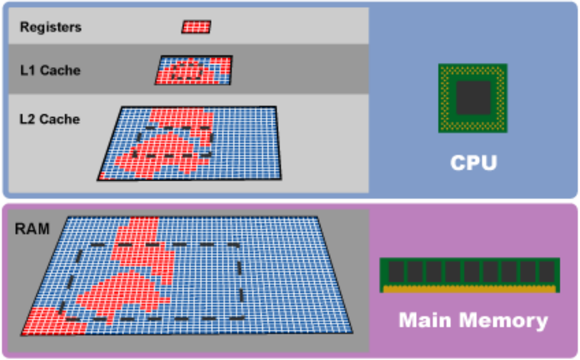
- Locally storing contents that are anticipated to be useful later;
- Prefetch data into local or faster memory;
- Useful on different time-space scales:



Caching and Its Applications

A natural management strategy when communication is bursty or costly

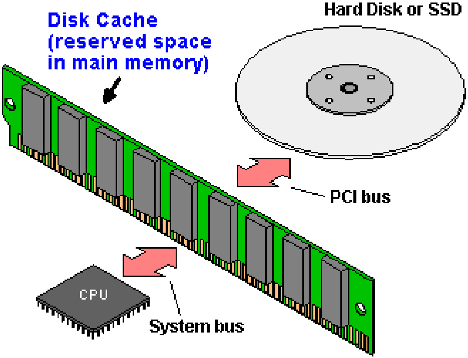
- Locally storing contents that are anticipated to be useful later;
- Prefetch data into local or faster memory;
- Useful on different time-space scales:



Caching and Its Applications

A natural management strategy when communication is bursty or costly

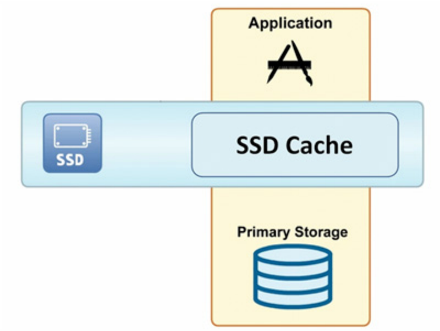
- Locally storing contents that are anticipated to be useful later;
- Prefetch data into local or faster memory;
- Useful on different time-space scales:



Caching and Its Applications

A natural management strategy when communication is bursty or costly

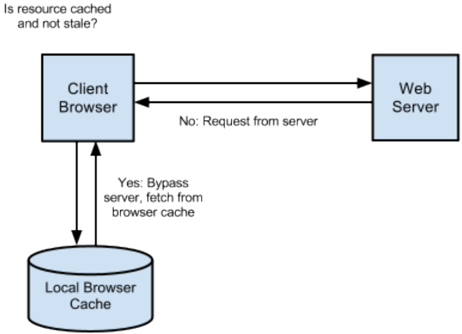
- Locally storing contents that are anticipated to be useful later;
- Prefetch data into local or faster memory;
- Useful on different time-space scales:



Caching and Its Applications

A natural management strategy when communication is bursty or costly

- Locally storing contents that are anticipated to be useful later;
- Prefetch data into local or faster memory;
- Useful on different time-space scales:



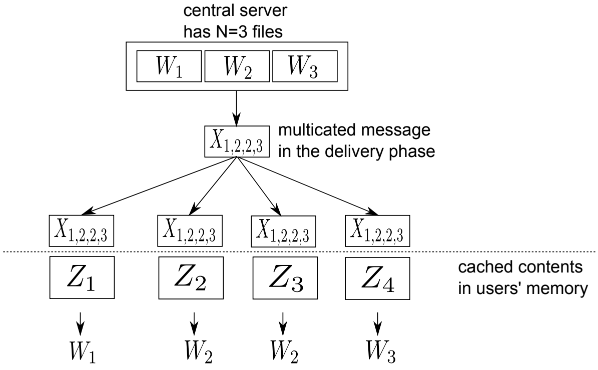
Caching for Content Delivery

- One central server and many users;
- Place contents in users' local caches during off-peak time;
- Peak time transmission can be reduced.

A Mathematical Model

Proposed by Maddah-Ali and Niesen (IT-14)

- N files, K users, each user has a cache of size M;
- Some data is cached during off-peak time: the placement phase;
- A common message to everyone in peak time: the delivery phase.

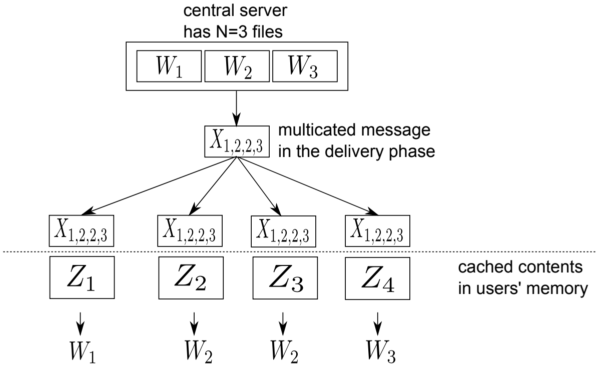


What is the fundamental limit of memory M vs. transmission rate R?

A Mathematical Model

Proposed by Maddah-Ali and Niesen (IT-14)

- N files, K users, each user has a cache of size M;
- Some data is cached during off-peak time: the placement phase;
- A common message to everyone in peak time: the delivery phase.

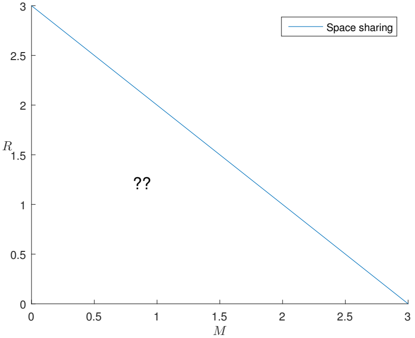


What is the fundamental limit of memory M vs. transmission rate R?

A Tradeoff between Memory and Transmission Rate

There is a tradeoff between M and R:

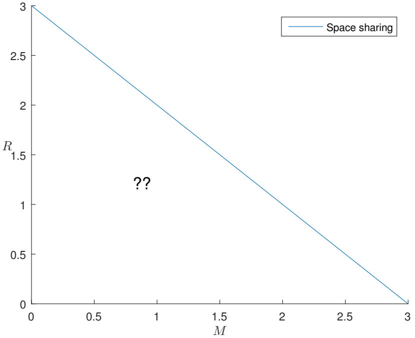
- Cache all content: $(M, R) = (N, 0)$;
- Cache nothing: $(M, R) = (0, K)$;
- Uncoded strategy: cache some parts, and transmit the missing
  - Can we do better? Yes, with coding.



A Tradeoff between Memory and Transmission Rate

There is a tradeoff between M and R:

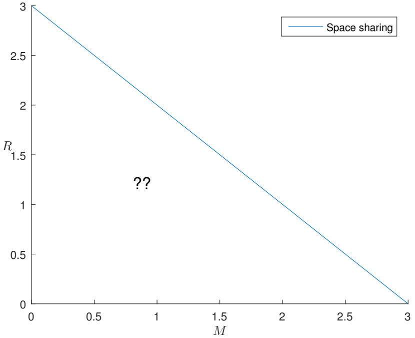
- Cache all content: $(M, R) = (N, 0)$;
- Cache nothing: $(M, R) = (0, K)$;
- Uncoded strategy: cache some parts, and transmit the missing
  - Can we do better? Yes, with coding.



A Tradeoff between Memory and Transmission Rate

There is a tradeoff between M and R:

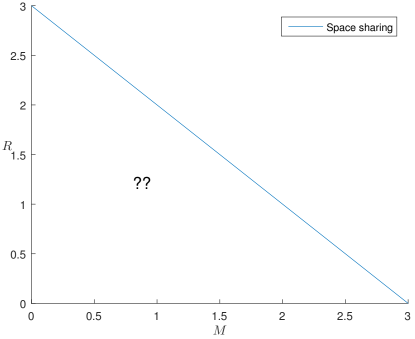
- Cache all content: $(M, R) = (N, 0)$;
- Cache nothing: $(M, R) = (0, K)$;
- Uncoded strategy: cache some parts, and transmit the missing
  - Can we do better? Yes, with coding.



A Tradeoff between Memory and Transmission Rate

There is a tradeoff between M and R:

- Cache all content: $(M, R) = (N, 0)$;
- Cache nothing: $(M, R) = (0, K)$;
- Uncoded strategy: cache some parts, and transmit the missing
  - Can we do better? Yes, with coding.



Inner Bounds, Outer Bounds and Approximation

Results by Maddah-Ali and Niesen, IT-14.

Theorem (A Rough Translation)

The following tradeoff pairs (and the lower convex hull) are achievable

$$(M, R) = \left(\frac{tN}{K}, (K-t)\min\left(\frac{1}{1+t}, \frac{N}{K}\right)\right), \quad t = 0, 1, \ldots, K. \tag{1}$$

The optimal transmission rate for a given memory M must satisfy $R \geq \max_{s \in \{1,2,\ldots,\min(n,k)\}}\left(s - \frac{s}{\lfloor N/s \rfloor} M\right)$. As a result, the tradeoff achieved in (1) is within a factor of 12 of the optimum.

An additional result by Chen et al., Arxiv-14

Theorem

When $N \leq K$, the tradeoff pair $\left(\frac{1}{K}, \frac{N(K-1)}{K}\right)$ is achievable.

Inner Bounds, Outer Bounds and Approximation

Results by Maddah-Ali and Niesen, IT-14.

Theorem (A Rough Translation)

The following tradeoff pairs (and the lower convex hull) are achievable

$$(M, R) = \left(\frac{tN}{K}, (K-t)\min\left(\frac{1}{1+t}, \frac{N}{K}\right)\right), \quad t = 0, 1, \ldots, K. \tag{1}$$

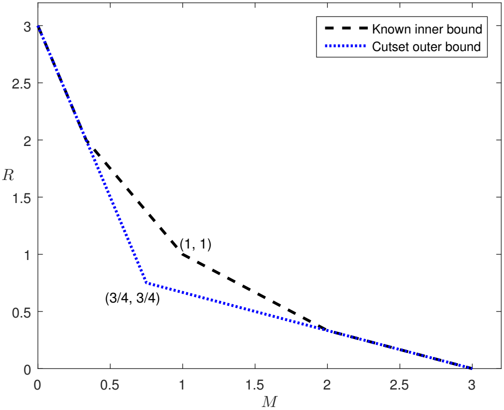
The optimal transmission rate for a given memory M must satisfy $R \geq \max_{s \in \{1,2,\ldots,\min(n,k)\}}(s - \frac{s}{\lfloor N/s \rfloor}M)$. As a result, the tradeoff achieved in (1) is within a factor of 12 of the optimum.

An additional result by Chen et al., Arxiv-14

Theorem

When $N \leq K$, the tradeoff pair $\left(\frac{1}{K}, \frac{N(K-1)}{K}\right)$ is achievable.

An Example $(N, K) = (3, 3)$



Main difficulty: in the placement phase, the requests are unknown

- Requests only revealed in the delivery phase.

The Maddah-Ali-Niesen Coding Scheme

Placement strategy:

- Partition each file into $\binom{K}{t}$ parts of equal size: each part associated with a subset of the users $\{1, 2, \ldots, K\}$ with t elements;
- Place each part in the users's cache of that subset (t copies in total);

Transmission strategy:

- A group of $t+1$ users: each needs a segment that all other users already have;
- An opportunity to use network coding: send XOR of these segments.

Example: $(N, K) = (3, 3), t = 2$, three files are (A, B, C), $\binom{3}{2} = 3$.

User 1	A_1	B_1	C_1	A_2	B_2	C_2
User 2	A_1	B_1	C_1	A_3	B_3	C_3
User 3	A_2	B_2	C_2	A_3	B_3	C_3

Users want (A, B, C): sending $A_3 + B_2 + C_1$.

The Maddah-Ali-Niesen Coding Scheme

Placement strategy:

- Partition each file into $\binom{K}{t}$ parts of equal size: each part associated with a subset of the users $\{1, 2, \ldots, K\}$ with t elements;
- Place each part in the users's cache of that subset (t copies in total);

Transmission strategy:

- A group of $t + 1$ users: each needs a segment that all other users already have;
- An opportunity to use network coding: send XOR of these segments.

Example: $(N, K) = (3, 3), t = 2$, three files are (A, B, C), $\binom{3}{2} = 3$.

User 1	A_1	B_1	C_1	A_2	B_2	C_2
User 2	A_1	B_1	C_1	A_3	B_3	C_3
User 3	A_2	B_2	C_2	A_3	B_3	C_3

Users want (A, B, C): sending $A_3 + B_2 + C_1$.

The Maddah-Ali-Niesen Coding Scheme

Placement strategy:

- Partition each file into $\binom{K}{t}$ parts of equal size: each part associated with a subset of the users $\{1, 2, \ldots, K\}$ with t elements;
- Place each part in the users's cache of that subset (t copies in total);

Transmission strategy:

- A group of $t + 1$ users: each needs a segment that all other users already have;
- An opportunity to use network coding: send XOR of these segments.

Example: $(N, K) = (3, 3), t = 2$, three files are (A, B, C), $\binom{3}{2} = 3$.

User 1	A_1	B_1	C_1	A_2	B_2	C_2
User 2	A_1	B_1	C_1	A_3	B_3	C_3
User 3	A_2	B_2	C_2	A_3	B_3	C_3

Users want (A, B, C): sending $A_3 + B_2 + C_1$.

Summary of Existing Results

- Maddah-Ali-Niesen scheme: uncoded placement, coded transmission;
- Cut-set outer bound: not tight in general;
- Approximation: with a constant factor the optimum;
- Question 1: Inner bound: coded placement and coded transmission?
  - Maddah-Ali and Niesen gave one for $(N, K) = (2, 2)$;
  - Extended by Chen et al. to $N \leq K$: only a single tradeoff point;
  - Code constructions of this type very limited
- Question 2: Outer bounds: tight (or tighter) bounds?
  - There are a few works on this (three independent papers in ISIT-15);
  - Even for small (N, K) values, no conclusive solutions except $(2, 2)$.

In this talk: results presented at ISIT 2016

- Part 1: A novel scheme with coded placement and transmission.
- Part 2: A set of outer bound results.

Summary of Existing Results

- Maddah-Ali-Niesen scheme: uncoded placement, coded transmission;
- Cut-set outer bound: not tight in general;
- Approximation: with a constant factor the optimum;
- Question 1: Inner bound: coded placement and coded transmission?
  - Maddah-Ali and Niesen gave one for $(N, K) = (2, 2)$;
  - Extended by Chen et al. to $N \leq K$: only a single tradeoff point;
  - Code constructions of this type very limited
- Question 2: Outer bounds: tight (or tighter) bounds?
  - There are a few works on this (three independent papers in ISIT-15);
  - Even for small (N, K) values, no conclusive solutions except $(2, 2)$.

In this talk: results presented at ISIT 2016

- Part 1: A novel scheme with coded placement and transmission.
- Part 2: A set of outer bound results.

Outline

A New Code: An Example for $(N, K) = (2, 4)$

- Two files (A, B);
- Each partitioned into $\binom{4}{2} = 6$ segments (symbols);
- **Linear combinations** are cached;
- Delivery phase: send 6 symbols.

User 1	$A_1 + B_1$	$A_2 + B_2$	$A_3 + B_3$	$A_1 + A_2 + A_3 + 2(B_1 + B_2 + B_3)$
User 2	$A_1 + B_1$	$A_4 + B_4$	$A_5 + B_5$	$A_1 + A_4 + A_5 + 2(B_1 + B_4 + B_5)$
User 3	$A_2 + B_2$	$A_4 + B_4$	$A_6 + B_6$	$A_2 + A_4 + A_6 + 2(B_2 + B_4 + B_6)$
User 4	$A_3 + B_3$	$A_5 + B_5$	$A_6 + B_6$	$A_3 + A_5 + A_6 + 2(B_3 + B_5 + B_6)$

A New Code: An Example for $(N, K) = (2, 4)$

User 1	$A_1 + B_1$	$A_2 + B_2$	$A_3 + B_3$	$A_1 + A_2 + A_3 + 2(B_1 + B_2 + B_3)$
User 2	$A_1 + B_1$	$A_4 + B_4$	$A_5 + B_5$	$A_1 + A_4 + A_5 + 2(B_1 + B_4 + B_5)$
User 3	$A_2 + B_2$	$A_4 + B_4$	$A_6 + B_6$	$A_2 + A_4 + A_6 + 2(B_2 + B_4 + B_6)$
User 4	$A_3 + B_3$	$A_5 + B_5$	$A_6 + B_6$	$A_3 + A_5 + A_6 + 2(B_3 + B_5 + B_6)$

Requests are (A, A, A, B), send

Step 1: B_1, B_2, B_4;

Step 2: $A_3 + 2A_5 + 3A_6, A_3 + 3A_5 + 4A_6$;

Step 3: $A_1 + A_2 + A_4$.

- User 1:

A New Code: An Example for $(N, K) = (2, 4)$

User 1	$A_1 + B_1$	$A_2 + B_2$	$A_3 + B_3$	$A_1 + A_2 + A_3 + 2(B_1 + B_2 + B_3)$
User 2	$A_1 + B_1$	$A_4 + B_4$	$A_5 + B_5$	$A_1 + A_4 + A_5 + 2(B_1 + B_4 + B_5)$
User 3	$A_2 + B_2$	$A_4 + B_4$	$A_6 + B_6$	$A_2 + A_4 + A_6 + 2(B_2 + B_4 + B_6)$
User 4	$A_3 + B_3$	$A_5 + B_5$	$A_6 + B_6$	$A_3 + A_5 + A_6 + 2(B_3 + B_5 + B_6)$

Requests are (A, A, A, B), send

Step 1: B_1, B_2, B_4;

Step 2: $A_3 + 2A_5 + 3A_6, A_3 + 3A_5 + 4A_6$;

Step 3: $A_1 + A_2 + A_4$.

- User 1: after step 1: has (A_1, A_2), and $(A_3 + B_3, A_3 + 2B_3)$

A New Code: An Example for $(N, K) = (2, 4)$

User 1	$A_1 + B_1$	$A_2 + B_2$	$A_3 + B_3$	$A_1 + A_2 + A_3 + 2(B_1 + B_2 + B_3)$
User 2	$A_1 + B_1$	$A_4 + B_4$	$A_5 + B_5$	$A_1 + A_4 + A_5 + 2(B_1 + B_4 + B_5)$
User 3	$A_2 + B_2$	$A_4 + B_4$	$A_6 + B_6$	$A_2 + A_4 + A_6 + 2(B_2 + B_4 + B_6)$
User 4	$A_3 + B_3$	$A_5 + B_5$	$A_6 + B_6$	$A_3 + A_5 + A_6 + 2(B_3 + B_5 + B_6)$

Requests are (A, A, A, B), send

Step 1: B_1, B_2, B_4;

Step 2: $A_3 + 2A_5 + 3A_6, A_3 + 3A_5 + 4A_6$;

Step 3: $A_1 + A_2 + A_4$.

- User 1: after step 1: has (A_1, A_2, A_3)

A New Code: An Example for $(N, K) = (2, 4)$

User 1	$A_1 + B_1$	$A_2 + B_2$	$A_3 + B_3$	$A_1 + A_2 + A_3 + 2(B_1 + B_2 + B_3)$
User 2	$A_1 + B_1$	$A_4 + B_4$	$A_5 + B_5$	$A_1 + A_4 + A_5 + 2(B_1 + B_4 + B_5)$
User 3	$A_2 + B_2$	$A_4 + B_4$	$A_6 + B_6$	$A_2 + A_4 + A_6 + 2(B_2 + B_4 + B_6)$
User 4	$A_3 + B_3$	$A_5 + B_5$	$A_6 + B_6$	$A_3 + A_5 + A_6 + 2(B_3 + B_5 + B_6)$

Requests are (A, A, A, B), send

Step 1: B_1, B_2, B_4;

Step 2: $A_3 + 2A_5 + 3A_6, A_3 + 3A_5 + 4A_6$;

Step 3: $A_1 + A_2 + A_4$.

- User 1: after step 2: has (A_1, A_2, A_3) and $(2A_5 + 3A_6, 3A_5 + 4A_6)$

A New Code: An Example for $(N, K) = (2, 4)$

User 1	$A_1 + B_1$	$A_2 + B_2$	$A_3 + B_3$	$A_1 + A_2 + A_3 + 2(B_1 + B_2 + B_3)$
User 2	$A_1 + B_1$	$A_4 + B_4$	$A_5 + B_5$	$A_1 + A_4 + A_5 + 2(B_1 + B_4 + B_5)$
User 3	$A_2 + B_2$	$A_4 + B_4$	$A_6 + B_6$	$A_2 + A_4 + A_6 + 2(B_2 + B_4 + B_6)$
User 4	$A_3 + B_3$	$A_5 + B_5$	$A_6 + B_6$	$A_3 + A_5 + A_6 + 2(B_3 + B_5 + B_6)$

Requests are (A, A, A, B), send

Step 1: B_1, B_2, B_4;

Step 2: $A_3 + 2A_5 + 3A_6, A_3 + 3A_5 + 4A_6$;

Step 3: $A_1 + A_2 + A_4$.

- User 1: after step 2: has $(A_1, A_2, A_3, A_5, A_6)$

A New Code: An Example for $(N, K) = (2, 4)$

User 1	$A_1 + B_1$	$A_2 + B_2$	$A_3 + B_3$	$A_1 + A_2 + A_3 + 2(B_1 + B_2 + B_3)$
User 2	$A_1 + B_1$	$A_4 + B_4$	$A_5 + B_5$	$A_1 + A_4 + A_5 + 2(B_1 + B_4 + B_5)$
User 3	$A_2 + B_2$	$A_4 + B_4$	$A_6 + B_6$	$A_2 + A_4 + A_6 + 2(B_2 + B_4 + B_6)$
User 4	$A_3 + B_3$	$A_5 + B_5$	$A_6 + B_6$	$A_3 + A_5 + A_6 + 2(B_3 + B_5 + B_6)$

Requests are (A, A, A, B), send

Step 1: B_1, B_2, B_4;

Step 2: $A_3 + 2A_5 + 3A_6, A_3 + 3A_5 + 4A_6$;

Step 3: $A_1 + A_2 + A_4$.

- User 1: after step 3: has $(A_1, A_2, A_3, A_4, A_5, A_6)$

A New Code: An Example for $(N, K) = (2, 4)$

User 1	$A_1 + B_1$	$A_2 + B_2$	$A_3 + B_3$	$A_1 + A_2 + A_3 + 2(B_1 + B_2 + B_3)$
User 2	$A_1 + B_1$	$A_4 + B_4$	$A_5 + B_5$	$A_1 + A_4 + A_5 + 2(B_1 + B_4 + B_5)$
User 3	$A_2 + B_2$	$A_4 + B_4$	$A_6 + B_6$	$A_2 + A_4 + A_6 + 2(B_2 + B_4 + B_6)$
User 4	$A_3 + B_3$	$A_5 + B_5$	$A_6 + B_6$	$A_3 + A_5 + A_6 + 2(B_3 + B_5 + B_6)$

Requests are (A, A, A, B), send

Step 1: B_1, B_2, B_4;

Step 2: $A_3 + 2A_5 + 3A_6, A_3 + 3A_5 + 4A_6$;

Step 3: $A_1 + A_2 + A_4$.

- User 1: after step 3: has $(A_1, A_2, A_3, A_4, A_5, A_6)$
- User 4:

A New Code: An Example for $(N, K) = (2, 4)$

User 1	$A_1 + B_1$	$A_2 + B_2$	$A_3 + B_3$	$A_1 + A_2 + A_3 + 2(B_1 + B_2 + B_3)$
User 2	$A_1 + B_1$	$A_4 + B_4$	$A_5 + B_5$	$A_1 + A_4 + A_5 + 2(B_1 + B_4 + B_5)$
User 3	$A_2 + B_2$	$A_4 + B_4$	$A_6 + B_6$	$A_2 + A_4 + A_6 + 2(B_2 + B_4 + B_6)$
User 4	$A_3 + B_3$	$A_5 + B_5$	$A_6 + B_6$	$A_3 + A_5 + A_6 + 2(B_3 + B_5 + B_6)$

Requests are (A, A, A, B), send

Step 1: B_1, B_2, B_4;

Step 2: $A_3 + 2A_5 + 3A_6, A_3 + 3A_5 + 4A_6$;

Step 3: $A_1 + A_2 + A_4$.

- User 1: after step 3: has $(A_1, A_2, A_3, A_4, A_5, A_6)$
- User 4: after step 1: has (B_1, B_2, B_4), and has $A_3 + A_5 + A_6$

A New Code: An Example for $(N, K) = (2, 4)$

User 1	$A_1 + B_1$	$A_2 + B_2$	$A_3 + B_3$	$A_1 + A_2 + A_3 + 2(B_1 + B_2 + B_3)$
User 2	$A_1 + B_1$	$A_4 + B_4$	$A_5 + B_5$	$A_1 + A_4 + A_5 + 2(B_1 + B_4 + B_5)$
User 3	$A_2 + B_2$	$A_4 + B_4$	$A_6 + B_6$	$A_2 + A_4 + A_6 + 2(B_2 + B_4 + B_6)$
User 4	$A_3 + B_3$	$A_5 + B_5$	$A_6 + B_6$	$A_3 + A_5 + A_6 + 2(B_3 + B_5 + B_6)$

Requests are (A, A, A, B), send

Step 1: B_1, B_2, B_4;

Step 2: $A_3 + 2A_5 + 3A_6, A_3 + 3A_5 + 4A_6$;

Step 3: $A_1 + A_2 + A_4$.

- User 1: after step 3: has $(A_1, A_2, A_3, A_4, A_5, A_6)$
- User 4: after step 2: has (B_1, B_2, B_4), and (A_3, A_5, A_6)

A New Code: An Example for $(N, K) = (2, 4)$

User 1	$A_1 + B_1$	$A_2 + B_2$	$A_3 + B_3$	$A_1 + A_2 + A_3 + 2(B_1 + B_2 + B_3)$
User 2	$A_1 + B_1$	$A_4 + B_4$	$A_5 + B_5$	$A_1 + A_4 + A_5 + 2(B_1 + B_4 + B_5)$
User 3	$A_2 + B_2$	$A_4 + B_4$	$A_6 + B_6$	$A_2 + A_4 + A_6 + 2(B_2 + B_4 + B_6)$
User 4	$A_3 + B_3$	$A_5 + B_5$	$A_6 + B_6$	$A_3 + A_5 + A_6 + 2(B_3 + B_5 + B_6)$

Requests are (A, A, A, B), send

Step 1: B_1, B_2, B_4;

Step 2: $A_3 + 2A_5 + 3A_6, A_3 + 3A_5 + 4A_6$;

Step 3: $A_1 + A_2 + A_4$.

- User 1: after step 3: has $(A_1, A_2, A_3, A_4, A_5, A_6)$
- User 4: after step 2: has $(B_1, B_2, B_3, B_4, B_5, B_6)$

A New Code: An Example for $(N, K) = (2, 4)$

User 1	$A_1 + B_1$	$A_2 + B_2$	$A_3 + B_3$	$A_1 + A_2 + A_3 + 2(B_1 + B_2 + B_3)$
User 2	$A_1 + B_1$	$A_4 + B_4$	$A_5 + B_5$	$A_1 + A_4 + A_5 + 2(B_1 + B_4 + B_5)$
User 3	$A_2 + B_2$	$A_4 + B_4$	$A_6 + B_6$	$A_2 + A_4 + A_6 + 2(B_2 + B_4 + B_6)$
User 4	$A_3 + B_3$	$A_5 + B_5$	$A_6 + B_6$	$A_3 + A_5 + A_6 + 2(B_3 + B_5 + B_6)$

Requests are (A, A, B, B), send

Step 1: B_1, A_6;

Step 2: $A_2 + 2A_4, A_3 + 2A_5, B_2 + 2B_3, B_4 + 2B_5$.

Step 3:

- User 1:

A New Code: An Example for $(N, K) = (2, 4)$

User 1	$A_1 + B_1$	$A_2 + B_2$	$A_3 + B_3$	$A_1 + A_2 + A_3 + 2(B_1 + B_2 + B_3)$
User 2	$A_1 + B_1$	$A_4 + B_4$	$A_5 + B_5$	$A_1 + A_4 + A_5 + 2(B_1 + B_4 + B_5)$
User 3	$A_2 + B_2$	$A_4 + B_4$	$A_6 + B_6$	$A_2 + A_4 + A_6 + 2(B_2 + B_4 + B_6)$
User 4	$A_3 + B_3$	$A_5 + B_5$	$A_6 + B_6$	$A_3 + A_5 + A_6 + 2(B_3 + B_5 + B_6)$

Requests are (A, A, B, B), send

Step 1: B_1, A_6;

Step 2: $A_2 + 2A_4, A_3 + 2A_5, B_2 + 2B_3, B_4 + 2B_5$.

Step 3:

- User 1: after step 1: has (A_1, A_6), and (B_1)

A New Code: An Example for $(N, K) = (2, 4)$

User 1	$A_1 + B_1$	$A_2 + B_2$	$A_3 + B_3$	$A_1 + A_2 + A_3 + 2(B_1 + B_2 + B_3)$
User 2	$A_1 + B_1$	$A_4 + B_4$	$A_5 + B_5$	$A_1 + A_4 + A_5 + 2(B_1 + B_4 + B_5)$
User 3	$A_2 + B_2$	$A_4 + B_4$	$A_6 + B_6$	$A_2 + A_4 + A_6 + 2(B_2 + B_4 + B_6)$
User 4	$A_3 + B_3$	$A_5 + B_5$	$A_6 + B_6$	$A_3 + A_5 + A_6 + 2(B_3 + B_5 + B_6)$

Requests are (A, A, B, B), send

Step 1: B_1, A_6;

Step 2: $A_2 + 2A_4, A_3 + 2A_5, B_2 + 2B_3, B_4 + 2B_5$.

Step 3:

- User 1: after step 1: has (A_1, A_6), and $(B_1, B_2 + B_3)$

A New Code: An Example for $(N, K) = (2, 4)$

User 1	$A_1 + B_1$	$A_2 + B_2$	$A_3 + B_3$	$A_1 + A_2 + A_3 + 2(B_1 + B_2 + B_3)$
User 2	$A_1 + B_1$	$A_4 + B_4$	$A_5 + B_5$	$A_1 + A_4 + A_5 + 2(B_1 + B_4 + B_5)$
User 3	$A_2 + B_2$	$A_4 + B_4$	$A_6 + B_6$	$A_2 + A_4 + A_6 + 2(B_2 + B_4 + B_6)$
User 4	$A_3 + B_3$	$A_5 + B_5$	$A_6 + B_6$	$A_3 + A_5 + A_6 + 2(B_3 + B_5 + B_6)$

Requests are (A, A, B, B), send

Step 1: B_1, A_6;

Step 2: $A_2 + 2A_4, A_3 + 2A_5, B_2 + 2B_3, B_4 + 2B_5$.

Step 3:

- User 1: after step 2: has (A_1, A_6), and $(B_1, B_2 + B_3, B_2 + 2B_3)$

A New Code: An Example for $(N, K) = (2, 4)$

User 1	$A_1 + B_1$	$A_2 + B_2$	$A_3 + B_3$	$A_1 + A_2 + A_3 + 2(B_1 + B_2 + B_3)$
User 2	$A_1 + B_1$	$A_4 + B_4$	$A_5 + B_5$	$A_1 + A_4 + A_5 + 2(B_1 + B_4 + B_5)$
User 3	$A_2 + B_2$	$A_4 + B_4$	$A_6 + B_6$	$A_2 + A_4 + A_6 + 2(B_2 + B_4 + B_6)$
User 4	$A_3 + B_3$	$A_5 + B_5$	$A_6 + B_6$	$A_3 + A_5 + A_6 + 2(B_3 + B_5 + B_6)$

Requests are (A, A, B, B), send

Step 1: B_1, A_6;

Step 2: $A_2 + 2A_4, A_3 + 2A_5, B_2 + 2B_3, B_4 + 2B_5$.

Step 3:

- User 1: after step 2: has (A_1, A_6), and (B_1, B_2, B_3)

A New Code: An Example for $(N, K) = (2, 4)$

User 1	$A_1 + B_1$	$A_2 + B_2$	$A_3 + B_3$	$A_1 + A_2 + A_3 + 2(B_1 + B_2 + B_3)$
User 2	$A_1 + B_1$	$A_4 + B_4$	$A_5 + B_5$	$A_1 + A_4 + A_5 + 2(B_1 + B_4 + B_5)$
User 3	$A_2 + B_2$	$A_4 + B_4$	$A_6 + B_6$	$A_2 + A_4 + A_6 + 2(B_2 + B_4 + B_6)$
User 4	$A_3 + B_3$	$A_5 + B_5$	$A_6 + B_6$	$A_3 + A_5 + A_6 + 2(B_3 + B_5 + B_6)$

Requests are (A, A, B, B), send

Step 1: B_1, A_6;

Step 2: $A_2 + 2A_4, A_3 + 2A_5, B_2 + 2B_3, B_4 + 2B_5$.

Step 3:

- User 1: after step 2: has (A_1, A_2, A_3, A_6) and needs (A_4, A_5)

A New Code: An Example for $(N, K) = (2, 4)$

User 1	$A_1 + B_1$	$A_2 + B_2$	$A_3 + B_3$	$A_1 + A_2 + A_3 + 2(B_1 + B_2 + B_3)$
User 2	$A_1 + B_1$	$A_4 + B_4$	$A_5 + B_5$	$A_1 + A_4 + A_5 + 2(B_1 + B_4 + B_5)$
User 3	$A_2 + B_2$	$A_4 + B_4$	$A_6 + B_6$	$A_2 + A_4 + A_6 + 2(B_2 + B_4 + B_6)$
User 4	$A_3 + B_3$	$A_5 + B_5$	$A_6 + B_6$	$A_3 + A_5 + A_6 + 2(B_3 + B_5 + B_6)$

Requests are (A, A, B, B), send

Step 1: B_1, A_6;

Step 2: $A_2 + 2A_4, A_3 + 2A_5, B_2 + 2B_3, B_4 + 2B_5$.

Step 3:

- User 1: after step 2: has (A_1, A_2, A_3, A_6) and $(A_2 + 2A_4, A_3 + 2A_5)$

A New Code: An Example for $(N, K) = (2, 4)$

User 1	$A_1 + B_1$	$A_2 + B_2$	$A_3 + B_3$	$A_1 + A_2 + A_3 + 2(B_1 + B_2 + B_3)$
User 2	$A_1 + B_1$	$A_4 + B_4$	$A_5 + B_5$	$A_1 + A_4 + A_5 + 2(B_1 + B_4 + B_5)$
User 3	$A_2 + B_2$	$A_4 + B_4$	$A_6 + B_6$	$A_2 + A_4 + A_6 + 2(B_2 + B_4 + B_6)$
User 4	$A_3 + B_3$	$A_5 + B_5$	$A_6 + B_6$	$A_3 + A_5 + A_6 + 2(B_3 + B_5 + B_6)$

Requests are (A, A, B, B), send

Step 1: B_1, A_6;

Step 2: $A_2 + 2A_4, A_3 + 2A_5, B_2 + 2B_3, B_4 + 2B_5$.

Step 3:

- User 1: after step 2: has $(A_1, A_2, A_3, A_4, A_5, A_6)$.

Some Simple Rules

User 1	$A_1 + B_1$	$A_2 + B_2$	$A_3 + B_3$	$A_1 + A_2 + A_3 + 2(B_1 + B_2 + B_3)$
User 2	$A_1 + B_1$	$A_4 + B_4$	$A_5 + B_5$	$A_1 + A_4 + A_5 + 2(B_1 + B_4 + B_5)$
User 3	$A_2 + B_2$	$A_4 + B_4$	$A_6 + B_6$	$A_2 + A_4 + A_6 + 2(B_2 + B_4 + B_6)$
User 4	$A_3 + B_3$	$A_5 + B_5$	$A_6 + B_6$	$A_3 + A_5 + A_6 + 2(B_3 + B_5 + B_6)$

- Each file is partitioned into $\binom{K}{t}$ segments;
- A segment is cached at a subset of users, but as a component of **linear combinations**;
- When a user request a file, other components in his cached linear combinations are **interferences**;
- Need to eliminate the interferences and recover the wanted segments;
- What are the rules for the transmission steps?

Some Simple Rules

User 1	$A_1 + B_1$	$A_2 + B_2$	$A_3 + B_3$	$A_1 + A_2 + A_3 + 2(B_1 + B_2 + B_3)$
User 2	$A_1 + B_1$	$A_4 + B_4$	$A_5 + B_5$	$A_1 + A_4 + A_5 + 2(B_1 + B_4 + B_5)$
User 3	$A_2 + B_2$	$A_4 + B_4$	$A_6 + B_6$	$A_2 + A_4 + A_6 + 2(B_2 + B_4 + B_6)$
User 4	$A_3 + B_3$	$A_5 + B_5$	$A_6 + B_6$	$A_3 + A_5 + A_6 + 2(B_3 + B_5 + B_6)$

- Each file is partitioned into $\binom{K}{t}$ segments;
- A segment is cached at a subset of users, but as a component of **linear combinations**;
- When a user request a file, other components in his cached linear combinations are **interferences**;
- Need to eliminate the interferences and recover the wanted segments;
- What are the rules for the transmission steps?

Some Simple Rules

User 1	$A_1 + B_1$	$A_2 + B_2$	$A_3 + B_3$	$A_1 + A_2 + A_3 + 2(B_1 + B_2 + B_3)$
User 2	$A_1 + B_1$	$A_4 + B_4$	$A_5 + B_5$	$A_1 + A_4 + A_5 + 2(B_1 + B_4 + B_5)$
User 3	$A_2 + B_2$	$A_4 + B_4$	$A_6 + B_6$	$A_2 + A_4 + A_6 + 2(B_2 + B_4 + B_6)$
User 4	$A_3 + B_3$	$A_5 + B_5$	$A_6 + B_6$	$A_3 + A_5 + A_6 + 2(B_3 + B_5 + B_6)$

Requests are (A, A, A, B), send

$$\begin{aligned} &\text{Step 1: } B_1, B_2, B_4; \\ &\text{Step 2: } A_3 + 2A_5 + 3A_6, A_3 + 3A_5 + 4A_6; \\ &\text{Step 3: } A_1 + A_2 + A_4. \end{aligned}$$

- Step 1 is uncoded;
- Only transmit when this segment is not present at any users requesting this file.

Some Simple Rules

User 1	$A_1 + B_1$	$A_2 + B_2$	$A_3 + B_3$	$A_1 + A_2 + A_3 + 2(B_1 + B_2 + B_3)$
User 2	$A_1 + B_1$	$A_4 + B_4$	$A_5 + B_5$	$A_1 + A_4 + A_5 + 2(B_1 + B_4 + B_5)$
User 3	$A_2 + B_2$	$A_4 + B_4$	$A_6 + B_6$	$A_2 + A_4 + A_6 + 2(B_2 + B_4 + B_6)$
User 4	$A_3 + B_3$	$A_5 + B_5$	$A_6 + B_6$	$A_3 + A_5 + A_6 + 2(B_3 + B_5 + B_6)$

Requests are (A, A, A, B), send

Step 1: B_1, B_2, B_4;

Step 2: $A_3 + 2A_5 + 3A_6, A_3 + 3A_5 + 4A_6$;

Step 3: $A_1 + A_2 + A_4$.

- Step 2 is coded;
- Linear combinations of segments of a single file: maintain linear independence, then each transmission can provide **rank reduction**.

Some Simple Rules

User 1	$A_1 + B_1$	$A_2 + B_2$	$A_3 + B_3$	$A_1 + A_2 + A_3 + 2(B_1 + B_2 + B_3)$
User 2	$A_1 + B_1$	$A_4 + B_4$	$A_5 + B_5$	$A_1 + A_4 + A_5 + 2(B_1 + B_4 + B_5)$
User 3	$A_2 + B_2$	$A_4 + B_4$	$A_6 + B_6$	$A_2 + A_4 + A_6 + 2(B_2 + B_4 + B_6)$
User 4	$A_3 + B_3$	$A_5 + B_5$	$A_6 + B_6$	$A_3 + A_5 + A_6 + 2(B_3 + B_5 + B_6)$

Requests are (A, A, A, B), send

Step 1: B_1, B_2, B_4;

Step 2: $A_3 + 2A_5 + 3A_6, A_3 + 3A_5 + 4A_6$;

Step 3: $A_1 + A_2 + A_4$.

- Step 1 is uncoded, Step 2 is coded;
- The first two steps together need to guarantee: with enough linear combinations, all the symbols at a user can be resolved.

Some Simple Rules

User 1	$A_1 + B_1$	$A_2 + B_2$	$A_3 + B_3$	$A_1 + A_2 + A_3 + 2(B_1 + B_2 + B_3)$
User 2	$A_1 + B_1$	$A_4 + B_4$	$A_5 + B_5$	$A_1 + A_4 + A_5 + 2(B_1 + B_4 + B_5)$
User 3	$A_2 + B_2$	$A_4 + B_4$	$A_6 + B_6$	$A_2 + A_4 + A_6 + 2(B_2 + B_4 + B_6)$
User 4	$A_3 + B_3$	$A_5 + B_5$	$A_6 + B_6$	$A_3 + A_5 + A_6 + 2(B_3 + B_5 + B_6)$

Requests are (A, A, A, B), send

Step 1: B_1, B_2, B_4;

Step 2: $A_3 + 2A_5 + 3A_6, A_3 + 3A_5 + 4A_6$;

Step 3: $A_1 + A_2 + A_4$.

- Step 1 is uncoded, Step 2 is coded;
- The first two steps together need to guarantee: with enough linear combinations, all interferences at a user can be eliminated completely.

Some Simple Rules

User 1	$A_1 + B_1$	$A_2 + B_2$	$A_3 + B_3$	$A_1 + A_2 + A_3 + 2(B_1 + B_2 + B_3)$
User 2	$A_1 + B_1$	$A_4 + B_4$	$A_5 + B_5$	$A_1 + A_4 + A_5 + 2(B_1 + B_4 + B_5)$
User 3	$A_2 + B_2$	$A_4 + B_4$	$A_6 + B_6$	$A_2 + A_4 + A_6 + 2(B_2 + B_4 + B_6)$
User 4	$A_3 + B_3$	$A_5 + B_5$	$A_6 + B_6$	$A_3 + A_5 + A_6 + 2(B_3 + B_5 + B_6)$

Requests are (A, A, A, B), send

$$\begin{aligned}&\text{Step 1: } B_1, B_2, B_4;\\&\text{Step 2: } A_3 + 2A_5 + 3A_6, A_3 + 3A_5 + 4A_6;\\&\text{Step 3: } A_1 + A_2 + A_4.\end{aligned}$$

- Step 1 is uncoded, Step 2 is coded: eliminate interferences.
- Step 3 transmission then completes the missing pieces among users requesting the same file.

Efficient Interference Elimination

The first two step transmissions guarantee elimination of interferences

- For small (N, K): reasonably straightforward, as in the example;
- When (N, K) are large: a complication.

An Example for $(N, K) = (3, 6)$

Example $(N, K) = (3, 6)$, $t = 3$

- Three files (A, B, C), each partitioned into $\binom{6}{3} = 20$ segments;
- Label a segment of a file by the corresponding subset: e.g., $A_{1,2,4}$
- Each user caches 18 linear combinations of the appropriate segments;
- Consider the requests (A, A, A, B, B, C);
- After step 1, the following interferences are present at users $(4, 5, 6)$

User 4	$A_{1,4,5}$	$A_{2,4,5}$	$A_{3,4,5}$	$A_{1,4,6}$	$A_{2,4,6}$	$A_{3,4,6}$
User 5	$A_{1,4,5}$	$A_{2,4,5}$	$A_{3,4,5}$	$A_{1,5,6}$	$A_{2,5,6}$	$A_{3,5,6}$
User 6	$A_{1,4,6}$	$A_{2,4,6}$	$A_{3,4,6}$	$A_{1,5,6}$	$A_{2,5,6}$	$A_{3,5,6}$

An Example for $(N, K) = (3, 6)$

Example $(N, K) = (3, 6)$, $t = 3$

- Consider the requests (A, A, A, B, B, C);
- After step 1, the following interferences are present at users $(4, 5, 6)$

User 4	$A_{1,4,5}$	$A_{2,4,5}$	$A_{3,4,5}$	$A_{1,4,6}$	$A_{2,4,6}$	$A_{3,4,6}$
User 5	$A_{1,4,5}$	$A_{2,4,5}$	$A_{3,4,5}$	$A_{1,5,6}$	$A_{2,5,6}$	$A_{3,5,6}$
User 6	$A_{1,4,6}$	$A_{2,4,6}$	$A_{3,4,6}$	$A_{1,5,6}$	$A_{2,5,6}$	$A_{3,5,6}$

An Example for $(N, K) = (3, 6)$

Example $(N, K) = (3, 6)$, $t = 3$

- Consider the requests (A, A, A, B, B, C);
- After step 1, the following interferences are present at users $(4, 5, 6)$

User 4	$A_{1,4,5}$	$A_{2,4,5}$	$A_{3,4,5}$	$A_{1,4,6}$	$A_{2,4,6}$	$A_{3,4,6}$
User 5	$A_{1,4,5}$	$A_{2,4,5}$	$A_{3,4,5}$	$A_{1,5,6}$	$A_{2,5,6}$	$A_{3,5,6}$
User 6	$A_{1,4,6}$	$A_{2,4,6}$	$A_{3,4,6}$	$A_{1,5,6}$	$A_{2,5,6}$	$A_{3,5,6}$

To eliminate these interferences, we send linear combinations of them

An Example for $(N, K) = (3, 6)$

Example $(N, K) = (3, 6)$, $t = 3$

- Consider the requests (A, A, A, B, B, C);
- After step 1, the following interferences are present at users $(4, 5, 6)$

User 4	$A_{1,4,5}$	$A_{2,4,5}$	$A_{3,4,5}$	$A_{1,4,6}$	$A_{2,4,6}$	$A_{3,4,6}$
User 5	$A_{1,4,5}$	$A_{2,4,5}$	$A_{3,4,5}$	$A_{1,5,6}$	$A_{2,5,6}$	$A_{3,5,6}$
User 6	$A_{1,4,6}$	$A_{2,4,6}$	$A_{3,4,6}$	$A_{1,5,6}$	$A_{2,5,6}$	$A_{3,5,6}$

To eliminate these interferences, we send linear combinations of them

- Strategy 1: transmit linear combinations of interferences of each user
  - 1 transmission=1 dimension reduction at one user.

An Example for $(N, K) = (3, 6)$

Example $(N, K) = (3, 6)$, $t = 3$

- Consider the requests (A, A, A, B, B, C);
- After step 1, the following interferences are present at users $(4, 5, 6)$

User 4	$A_{1,4,5}$	$A_{2,4,5}$	$A_{3,4,5}$	$A_{1,4,6}$	$A_{2,4,6}$	$A_{3,4,6}$
User 5	$A_{1,4,5}$	$A_{2,4,5}$	$A_{3,4,5}$	$A_{1,5,6}$	$A_{2,5,6}$	$A_{3,5,6}$
User 6	$A_{1,4,6}$	$A_{2,4,6}$	$A_{3,4,6}$	$A_{1,5,6}$	$A_{2,5,6}$	$A_{3,5,6}$

To eliminate these interferences, we send linear combinations of them

- Strategy 1: transmit linear combinations of interferences of each user
  - 1 transmission=1 dimension reduction at one user.
- Strategy 2: transmit the common subspace, e.g., linear combinations of $A_{1,4,5}, A_{2,4,5}, A_{3,4,5}$
  - 1 transmission=1 dimension reduction at two users.

The General Scheme

Placement strategy:

1. Partition each file into $\binom{K}{t}$ segments;
2. A fixed number of linear combinations of these segments at each user.

Delivery strategy:

1. For the users requesting the same file, transmit uncoded segments that none of them have;
2. For all users not requesting a given file, collect segments of each common subspaces, and transmit their linear combinations separately;
3. Clean up any remaining missing segments.

The General Scheme

Placement strategy:

1. Partition each file into $\binom{K}{t}$ segments;
2. A fixed number of linear combinations of these segments at each user.

Delivery strategy:

1. For the users requesting the same file, transmit uncoded segments that none of them have;
2. For all users not requesting a given file, collect segments of each common subspaces, and transmit their linear combinations separately;
3. Clean up any remaining missing segments.

Revisiting the Example

User 1	$A_1 + B_1$	$A_2 + B_2$	$A_3 + B_3$	$A_1 + A_2 + A_3 + 2(B_1 + B_2 + B_3)$
User 2	$A_1 + B_1$	$A_4 + B_4$	$A_5 + B_5$	$A_1 + A_4 + A_5 + 2(B_1 + B_4 + B_5)$
User 3	$A_2 + B_2$	$A_4 + B_4$	$A_6 + B_6$	$A_2 + A_4 + A_6 + 2(B_2 + B_4 + B_6)$
User 4	$A_3 + B_3$	$A_5 + B_5$	$A_6 + B_6$	$A_3 + A_5 + A_6 + 2(B_3 + B_5 + B_6)$

Requests are (A, A, A, B), send

Step 1: B_1, B_2, B_4;

Step 2: $A_3 + 2A_5 + 3A_6, A_3 + 3A_5 + 4A_6$;

Step 3: $A_1 + A_2 + A_4$.

The Main Theorem

Key difficulty:

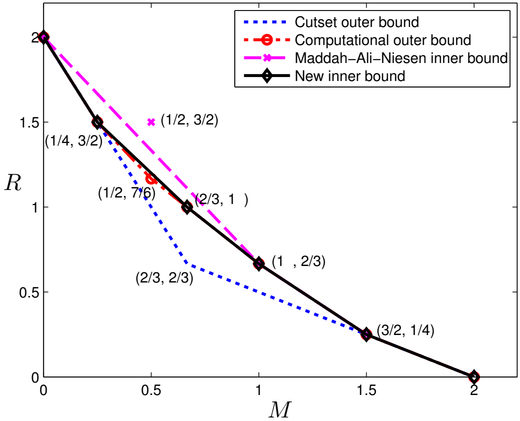
- Choose the numbers of combinations nicely (placement, 1st and 2nd step transmissions): guarantee interference elimination;
  - Linear combination coefficients not critical: full rank.
- Correctness and performance are tied to these numbers.

Theorem

For $N \in \mathbb{N}$ files and $K \in \mathbb{N}$ users each with a cache of size M, where $\mathbb{N}$ is the set of natural numbers and $N \leq K$, the following (M, R) pair is achievable

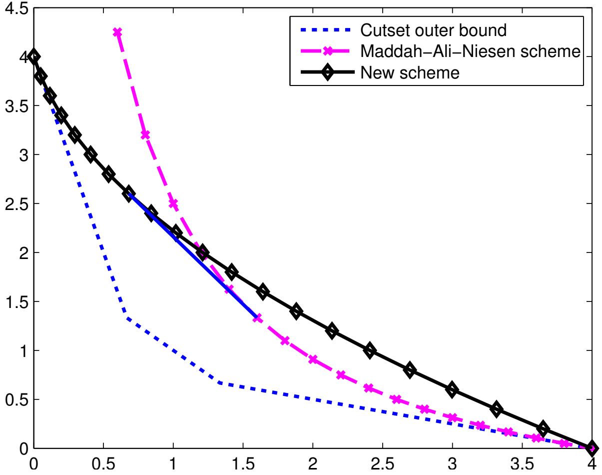
$$\left(\frac{t[(N-1)t + K - N]}{K(K-1)}, \frac{N(K-t)}{K} \right), \qquad t = 0, 1, \ldots, K.$$

Performance Example $(N, K) = (2, 4)$



- One new corner point on the inner bound for this case;
- Optimal tradeoff now known for $M \in [0, 1/4] \cup [2/3, 2]$.

Performance Example $(N, K) = (4, 20)$



Recap: What Just Happened?

We present a new code construction

- The caching strategy and transmission strategies (mysteriously) work;
- Its performance can be analyzed with nice closed form formulas;
- Some simple rules are provided as guiding principles;
- Where did this come from?

In fact some key insights came from the investigation of outer bounds.

Recap: What Just Happened?

We present a new code construction

- The caching strategy and transmission strategies (mysteriously) work;
- Its performance can be analyzed with nice closed form formulas;
- Some simple rules are provided as guiding principles;
- Where did this come from?

In fact some key insights came from the investigation of outer bounds.

Recap: What Just Happened?

We present a new code construction

- The caching strategy and transmission strategies (mysteriously) work;
- Its performance can be analyzed with nice closed form formulas;
- Some simple rules are provided as guiding principles;
- Where did this come from?

In fact some key insights came from the investigation of outer bounds.

Outline

Fundamental Limits: The Conventional Approach

An art more than a science:

1. Develop a good understanding of the engineering problem;
2. Chain of inequalities: trial-and-error with information inequalities.

Often heard comments:

- Need a smarter student!
- He really needs to spend more time on it!

⇓

Heavy reliance on humans: human ingenuity and effort

Fundamental Limits: The Conventional Approach

An art more than a science:

1. Develop a good understanding of the engineering problem;
2. Chain of inequalities: trial-and-error with information inequalities.

Often heard comments:

- Need a smarter student!
- He really needs to spend more time on it!

⇓

Heavy reliance on humans: human ingenuity and effort

Fundamental Limits: New Approaches?

Question: how can we reduce the human factors?

$\Downarrow$

Derivation of the chains of inequalities as an optimization procedure:

- Many possible information inequalities: choose the right combination.

$\Downarrow$

Idea: computers to do some or all the work?

$\Uparrow$

A key driver: recent development in optimization software and hardware.

Fundamental Limits: New Approaches?

Question: how can we reduce the human factors?

$\Downarrow$

Derivation of the chains of inequalities as an optimization procedure:

- Many possible information inequalities: choose the right combination.

$\Downarrow$

Idea: computers to do some or all the work?

$\Uparrow$

A key driver: recent development in optimization software and hardware.

Fundamental Limits: New Approaches?

Question: how can we reduce the human factors?

$\Downarrow$

Derivation of the chains of inequalities as an optimization procedure:

- Many possible information inequalities: choose the right combination.

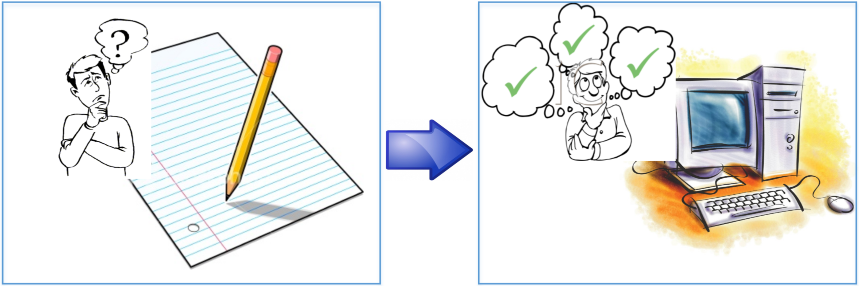
$\Downarrow$

Idea: computers to do some or all the work?

$\Uparrow$

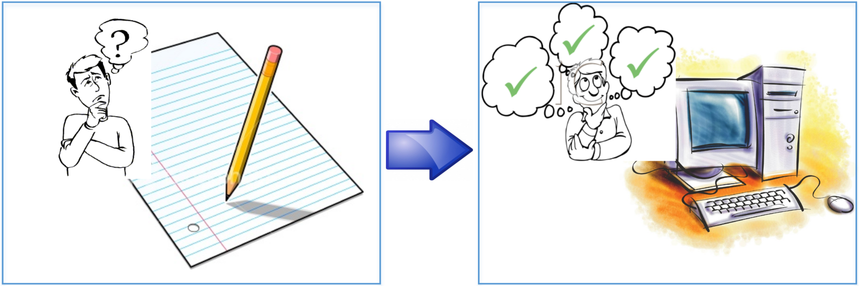
A key driver: recent development in optimization software and hardware.

From Analytical to Computational



Has anyone thought of this already?

From Analytical to Computational



Has anyone thought of this already?

Yeung's Linear Program to Prove Information Inequalities

Is a certain information inequality true? "Yes or can't-determine"

- Use all inequalities from the basic properties (Shannon-type);
- Linear inequalities: one joint entropy represented by one LP variable;
  - Example LP variables: $v_8 = H(S_1 X_1 X_7), v_{28} = H(S_2 \hat{S}_1 X_3 X_5)...$;
  - Example LP constraints:
    $H(S_1 X_1 X_6) + H(X_1 X_3 X_6) \geq H(X_1 X_6) + H(S_1 X_1 S_3 X_6)$
- Note: there are indeed non-Shannon-type inequalities.
- Shannon-type inequalities sufficient to prove most results in the literature for "practical" coding problems!

ITIP: A software package with a matlab interface ('97).

- Investigation of entropic region;
- As an auxiliary tool for confirming simple conjectured inequalities.

Yeung's Linear Program to Prove Information Inequalities

Is a certain information inequality true? "Yes or can't-determine"

- Use all inequalities from the basic properties (Shannon-type);
- Linear inequalities: one joint entropy represented by one LP variable;
  - Example LP variables: $v_8 = H(S_1 X_1 X_7), v_{28} = H(S_2 \hat{S}_1 X_3 X_5)$...;
  - Example LP constraints:
    $H(S_1 X_1 X_6) + H(X_1 X_3 X_6) \geq H(X_1 X_6) + H(S_1 X_1 S_3 X_6)$
- Note: there are indeed non-Shannon-type inequalities.
- Shannon-type inequalities sufficient to prove most results in the literature for "practical" coding problems!

ITIP: A software package with a matlab interface ('97).

- Investigation of entropic region;
- As an auxiliary tool for confirming simple conjectured inequalities.

Yeung's Linear Program to Prove Information Inequalities

Is a certain information inequality true? "Yes or can't-determine"

- Use all inequalities from the basic properties (Shannon-type);
- Linear inequalities: one joint entropy represented by one LP variable;
  - Example LP variables: $v_8 = H(S_1X_1X_7), v_{28} = H(S_2\hat{S}_1X_3X_5)...$;
  - Example LP constraints:
    $H(S_1X_1X_6) + H(X_1X_3X_6) \geq H(X_1X_6) + H(S_1X_1S_3X_6)$
- Note: there are indeed non-Shannon-type inequalities.
- Shannon-type inequalities sufficient to prove most results in the literature for "practical" coding problems!

ITIP: A software package with a matlab interface ('97).

- Investigation of entropic region;
- As an auxiliary tool for confirming simple conjectured inequalities.

Why Hasn't ITIP Been Used More Widely?

1. LP exponential in the number of random variables.
   - n-variable problem: $2^n - 1$ LP variables and $\binom{n}{2}2^{n-2}$ LP constraints.
   - Quickly runs beyond manageable range: roughly $n < 14$.
2. It's an inequality prover: what inequality to prove?
   - In engineering problems: hopefully the fundamental limit;
   - How do we find it in the first place?

Why Hasn't ITIP Been Used More Widely?

1. LP exponential in the number of random variables.
   - n-variable problem: $2^n - 1$ LP variables and $\binom{n}{2}2^{n-2}$ LP constraints.
   - Quickly runs beyond manageable range: roughly $n < 14$.

2. It's an inequality prover: what inequality to prove?
   - In engineering problems: hopefully the fundamental limit;
   - How do we find it in the first place?

Our New Approach

A more domain-specific LP approach:

1. Symmetry and other-factors to reduce LP;
2. Finding boundary (instead of decision on a conjectured inequality);
3. LP dual to generate human-readable proofs.

First used on the regenerating code problem (Tian, ISIT-13, JSAC-14)

- First time the entropy LP approach used on an engineering problem;
- Showed functional-repair and exact-repair are fundamentally different;
- Applied on MLD coding with regeneration (Tian-Liu, Allerton-14);
- Inspired several follow-up works
  - Ho et al. (ISIT-2014): ITIP now produces human-readable proofs;
  - Li et al. (arxiv:1407.5659): applied the method on MLD coding;
  - Ye et al. (ISIT-2016): secure regenerating codes with 5 nodes.

Our New Approach

A more domain-specific LP approach:

1. Symmetry and other-factors to reduce LP;
2. Finding boundary (instead of decision on a conjectured inequality);
3. LP dual to generate human-readable proofs.

First used on the regenerating code problem (Tian, ISIT-13, JSAC-14)

- First time the entropy LP approach used on an engineering problem;
- Showed functional-repair and exact-repair are fundamentally different;
- Applied on MLD coding with regeneration (Tian-Liu, Allerton-14);
- Inspired several follow-up works
  - Ho et al. (ISIT-2014): ITIP now produces human-readable proofs;
  - Li et al. (arxiv:1407.5659): applied the method on MLD coding;
  - Ye et al. (ISIT-2016): secure regenerating codes with 5 nodes.

Our New Approach

A more domain-specific LP approach:

1. Symmetry and other-factors to reduce LP;
2. Finding boundary (instead of decision on a conjectured inequality);
3. LP dual to generate human-readable proofs.

First used on the regenerating code problem (Tian, ISIT-13, JSAC-14)

- First time the entropy LP approach used on an engineering problem;
- Showed functional-repair and exact-repair are fundamentally different;
- Applied on MLD coding with regeneration (Tian-Liu, Allerton-14);
- Inspired several follow-up works
  - Ho et al. (ISIT-2014): ITIP now produces human-readable proofs;
  - Li et al. (arxiv:1407.5659): applied the method on MLD coding;
  - Ye et al. (ISIT-2016): secure regenerating codes with 5 nodes.

Our New Approach

A more domain-specific LP approach:

1. Symmetry and other-factors to reduce LP;
2. Finding boundary (instead of decision on a conjectured inequality);
3. LP dual to generate human-readable proofs.

First used on the regenerating code problem (Tian, ISIT-13, JSAC-14)

- First time the entropy LP approach used on an engineering problem;
- Showed functional-repair and exact-repair are fundamentally different;
- Applied on MLD coding with regeneration (Tian-Liu, Allerton-14);
- Inspired several follow-up works
  - Ho et al. (ISIT-2014): ITIP now produces human-readable proofs;
  - Li et al. (arxiv:1407.5659): applied the method on MLD coding;
  - Ye et al. (ISIT-2016): secure regenerating codes with 5 nodes.

Our New Approach

A more domain-specific LP approach:

1. Symmetry and other-factors to reduce LP;
2. Finding boundary (instead of decision on a conjectured inequality);
3. LP dual to generate human-readable proofs.

First used on the regenerating code problem (Tian, ISIT-13, JSAC-14)

- First time the entropy LP approach used on an engineering problem;
- Showed functional-repair and exact-repair are fundamentally different;
- Applied on MLD coding with regeneration (Tian-Liu, Allerton-14);
- Inspired several follow-up works
  - Ho et al. (ISIT-2014): ITIP now produces human-readable proofs;
  - Li et al. (arxiv:1407.5659): applied the method on MLD coding;
  - Ye et al. (ISIT-2016): secure regenerating codes with 5 nodes.

Our New Approach

A more domain-specific LP approach:

1. Symmetry and other-factors to reduce LP;
2. Finding boundary (instead of decision on a conjectured inequality);
3. LP dual to generate human-readable proofs.

First used on the regenerating code problem (Tian, ISIT-13, JSAC-14)

- First time the entropy LP approach used on an engineering problem;
- Showed functional-repair and exact-repair are fundamentally different;
- Applied on MLD coding with regeneration (Tian-Liu, Allerton-14);
- Inspired several follow-up works
  - Ho et al. (ISIT-2014): ITIP now produces human-readable proofs;
  - Li et al. (arxiv:1407.5659): applied the method on MLD coding;
  - Ye et al. (ISIT-2016): secure regenerating codes with 5 nodes.

Our New Approach

A more domain-specific LP approach:

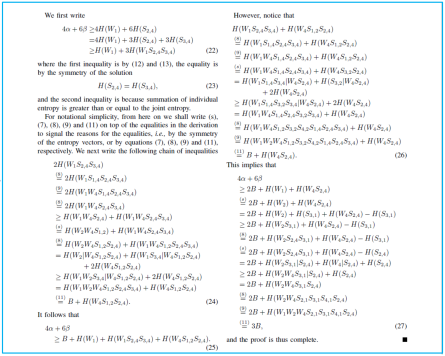
1. Symmetry and other-factors to reduce LP;
2. Finding boundary (instead of decision on a conjectured inequality);
3. LP dual to generate human-readable proofs.

First used on the regenerating code problem (Tian, ISIT-13, JSAC-14)

- First time the entropy LP approach used on an engineering problem;
- Showed functional-repair and exact-repair are fundamentally different;
- Applied on MLD coding with regeneration (Tian-Liu, Allerton-14);
- Inspired several follow-up works
  - Ho et al. (ISIT-2014): ITIP now produces human-readable proofs;
  - Li et al. (arxiv:1407.5659): applied the method on MLD coding;
  - Ye et al. (ISIT-2016): secure regenerating codes with 5 nodes.

From Table to Chain (to a Research Paper)

B	α	β	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8
	7	7		−7						
	−3			6			−3			
	1	−1	1						−1	
−1	−1			1					1	
−1						−1	1			1
−1					−1		1	1		
			−1		1	1		−1		
							1			−1
−3	4	6								



We first write

$$\begin{aligned} 4\alpha + 6\beta &\geq 4H(W_1) + 6H(S_{2,4}) \\ &= 4H(W_1) + 3H(S_{2,4}) + 3H(S_{3,4}) \\ &\geq H(W_1) + 3H(W_1 S_{2,4} S_{3,4}) \end{aligned} \tag{22}$$

where the first inequality is by (12) and (13), the equality is by the symmetry of the solution

$$H(S_{2,4}) = H(S_{3,4}), \tag{23}$$

and the second inequality is because summation of individual entropy is greater than or equal to the joint entropy.

For notational simplicity, from here on we shall write (s), (7), (8), (9) and (11) on top of the equalities in the derivation to signal the reasons for the equalities, *i.e.*, by the symmetry of the entropy vectors, or by equations (7), (8), (9) and (11), respectively. We next write the following chain of inequalities

$$\begin{aligned} &2H(W_1 S_{2,4} S_{3,4}) \\ &\overset{(8)}{=} 2H(W_1 S_{1,4} S_{2,4} S_{3,4}) \\ &\overset{(9)}{=} 2H(W_1 W_4 S_{1,4} S_{2,4} S_{3,4}) \\ &\overset{(8)}{=} 2H(W_1 W_4 S_{2,4} S_{3,4}) \\ &\geq H(W_1 W_4 S_{2,4}) + H(W_1 W_4 S_{2,4} S_{3,4}) \\ &\overset{(s)}{=} H(W_2 W_4 S_{1,2}) + H(W_1 W_4 S_{2,4} S_{3,4}) \\ &\overset{(8)}{=} H(W_2 W_4 S_{1,2} S_{2,4}) + H(W_1 W_4 S_{1,2} S_{2,4} S_{3,4}) \\ &= H(W_2 | W_4 S_{1,2} S_{2,4}) + H(W_1 S_{3,4} | W_4 S_{1,2} S_{2,4}) \\ &\qquad + 2H(W_4 S_{1,2} S_{2,4}) \\ &\geq H(W_1 W_2 S_{3,4} | W_4 S_{1,2} S_{2,4}) + 2H(W_4 S_{1,2} S_{2,4}) \\ &= H(W_1 W_2 W_4 S_{1,2} S_{2,4} S_{3,4}) + H(W_4 S_{1,2} S_{2,4}) \\ &\overset{(11)}{=} B + H(W_4 S_{1,2} S_{2,4}). \end{aligned} \tag{24}$$

It follows that

$$\begin{aligned} &4\alpha + 6\beta \\ &\geq B + H(W_1) + H(W_1 S_{2,4} S_{3,4}) + H(W_4 S_{1,2} S_{2,4}). \end{aligned} \tag{25}$$

However, notice that

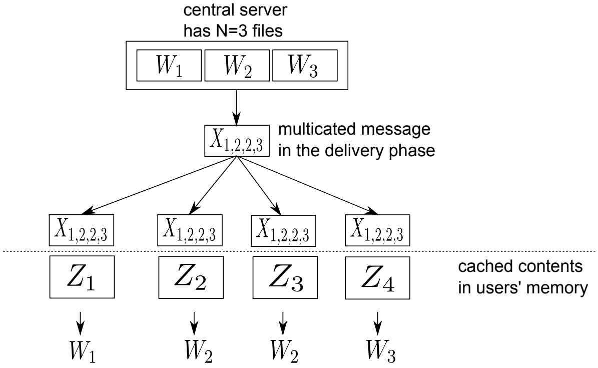
$$\begin{aligned} &H(W_1 S_{2,4} S_{3,4}) + H(W_4 S_{1,2} S_{2,4}) \\ &\overset{(8)}{=} H(W_1 S_{1,4} S_{2,4} S_{3,4}) + H(W_4 S_{1,2} S_{2,4}) \\ &\overset{(9)}{=} H(W_1 W_4 S_{1,4} S_{2,4} S_{3,4}) + H(W_4 S_{1,2} S_{2,4}) \\ &\overset{(s)}{=} H(W_1 W_4 S_{1,4} S_{2,4} S_{3,4}) + H(W_4 S_{3,2} S_{2,4}) \\ &= H(W_1 S_{1,4} S_{3,4} | W_4 S_{2,4}) + H(S_{3,2} | W_4 S_{2,4}) \\ &\qquad + 2H(W_4 S_{2,4}) \\ &\geq H(W_1 S_{1,4} S_{3,2} S_{3,4} | W_4 S_{2,4}) + 2H(W_4 S_{2,4}) \\ &= H(W_1 W_4 S_{1,4} S_{2,4} S_{3,2} S_{3,4}) + H(W_4 S_{2,4}) \\ &\overset{(8)}{=} H(W_1 W_4 S_{1,2} S_{3,2} S_{4,2} S_{1,4} S_{2,4} S_{3,4}) + H(W_4 S_{2,4}) \\ &\overset{(9)}{=} H(W_1 W_2 W_4 S_{1,2} S_{3,2} S_{4,2} S_{1,4} S_{2,4} S_{3,4}) + H(W_4 S_{2,4}) \\ &\overset{(11)}{=} B + H(W_4 S_{2,4}). \end{aligned} \tag{26}$$

This implies that

$$\begin{aligned} &4\alpha + 6\beta \\ &\geq 2B + H(W_1) + H(W_4 S_{2,4}) \\ &\overset{(s)}{=} 2B + H(W_2) + H(W_4 S_{2,4}) \\ &= 2B + H(W_2) + H(S_{3,1}) + H(W_4 S_{2,4}) - H(S_{3,1}) \\ &\geq 2B + H(W_2 S_{3,1}) + H(W_4 S_{2,4}) - H(S_{3,1}) \\ &\overset{(8)}{=} 2B + H(W_2 S_{2,4} S_{3,1}) + H(W_4 S_{2,4}) - H(S_{3,1}) \\ &\overset{(s)}{=} 2B + H(W_2 S_{2,4} S_{3,1}) + H(W_4 S_{2,4}) - H(S_{2,4}) \\ &= 2B + H(W_2 S_{3,1} | S_{2,4}) + H(W_4 | S_{2,4}) + H(S_{2,4}) \\ &\geq 2B + H(W_2 W_4 S_{3,1} | S_{2,4}) + H(S_{2,4}) \\ &= 2B + H(W_2 W_4 S_{3,1} S_{2,4}) \\ &\overset{(8)}{=} 2B + H(W_2 W_4 S_{2,1} S_{3,1} S_{4,1} S_{2,4}) \\ &\overset{(9)}{=} 2B + H(W_1 W_2 W_4 S_{2,1} S_{3,1} S_{4,1} S_{2,4}) \\ &\overset{(11)}{=} 3B, \end{aligned} \tag{27}$$

and the proof is thus complete. ■

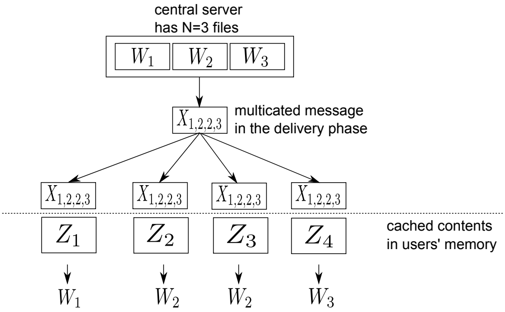
Symmetry in the Caching Problem



Quantities in the problem: $n = N + K + N^K$

- N files: $\mathcal{W} = \{W_1, W_2, ..., W_N\}$;
- Cached contents at K users: $\mathcal{Z} = \{Z_1, Z_2, ..., Z_K\}$;
- Transmission for demands $(d_1, d_2, \ldots, d_K)$: $\mathcal{X} = \{X_{d_1,d_2,...,d_K}\}$.

Symmetry in the Caching Problem



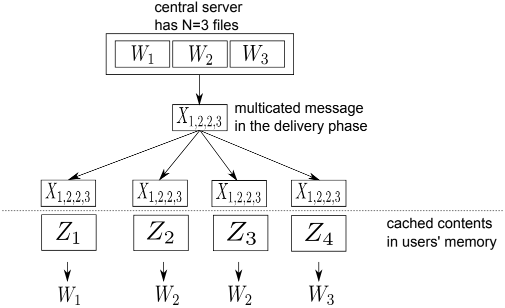
User index symmetry $\bar{\pi}$: permute the cached contents Z_i at users

- Delivery: need to transmit the corresponding $X_{d_1,\ldots,d_K}$.

File index symmetry $\hat{\pi}$: permute the files before encoding

- Delivery: use the same encoding function on the permuted files;

Symmetry in the Caching Problem



User index symmetry $\bar{\pi}$: permute the cached contents Z_i at users

- Delivery: need to transmit the corresponding $X_{d_1,\ldots,d_K}$.

File index symmetry $\hat{\pi}$: permute the files before encoding

- Delivery: use the same encoding function on the permuted files;

The Existence of Optimal Symmetric Codes

What are symmetric codes?

- For all permutation-induced mappings, joint entropies the same.

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

- User-index: $\bar{\pi} = \begin{pmatrix} 1234 \\ 2314 \end{pmatrix}$, $H(W_2, Z_2, X_{1,2,3,2}) = H(W_2, Z_3, X_{3,1,2,2})$
- File-index: $\hat{\pi} = \begin{pmatrix} 123 \\ 231 \end{pmatrix}$, $H(W_3, Z_3, X_{1,2,3,2}) = H(W_1, Z_3, X_{2,3,1,3})$

Proposition

For any caching code, there is a code with the same or smaller caching memory and transmission rate, which is both user-index-symmetric and file-index-symmetric.

The Existence of Optimal Symmetric Codes

What are symmetric codes?

- For all permutation-induced mappings, joint entropies the same.

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

- User-index: $\bar{\pi} = \begin{pmatrix} 1234 \\ 2314 \end{pmatrix}$, $H(W_2, Z_2, X_{1,2,3,2}) = H(W_2, Z_3, X_{3,1,2,2})$
- File-index: $\hat{\pi} = \begin{pmatrix} 123 \\ 231 \end{pmatrix}$, $H(W_3, Z_3, X_{1,2,3,2}) = H(W_1, Z_3, X_{2,3,1,3})$

Proposition

For any caching code, there is a code with the same or smaller caching memory and transmission rate, which is both user-index-symmetric and file-index-symmetric.

The Existence of Optimal Symmetric Codes

What are symmetric codes?

- For all permutation-induced mappings, joint entropies the same.

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

- User-index: $\bar{\pi} = \begin{pmatrix} 1234 \\ 2314 \end{pmatrix}$, $H(W_2, Z_2, X_{1,2,3,2}) = H(W_2, Z_3, X_{3,1,2,2})$
- File-index: $\hat{\pi} = \begin{pmatrix} 123 \\ 231 \end{pmatrix}$, $H(W_3, Z_3, X_{1,2,3,2}) = H(W_1, Z_3, X_{2,3,1,3})$

Proposition

For any caching code, there is a code with the same or smaller caching memory and transmission rate, which is both user-index-symmetric and file-index-symmetric.

More about the Symmetry

A "simple" question: after the symmetry reduction, how many unique joint entropy values do we have?

- Estimate: $2^{N+K+N^K}/N!K!$;
- More accurate: Polya's theory for counting (generating function and cycle index).

Symmetry induced by permutation groups

- The base symmetric groups: S_N and S_K;
- First induced permutation group: $\mathcal{W} \cup \mathcal{Z} \cup \mathcal{X} \to \mathcal{W} \cup \mathcal{Z} \cup \mathcal{X}$;
  - Compositions of any induced permutations by $\bar{\pi} \in S_K$ and $\hat{\pi} \in S_N$;
- Second induced permutation group: $2^{\mathcal{W} \cup \mathcal{S} \cup \mathcal{X}} \to 2^{\mathcal{W} \cup \mathcal{S} \cup \mathcal{X}}$.

More about the Symmetry

A "simple" question: after the symmetry reduction, how many unique joint entropy values do we have?

- Estimate: $2^{N+K+N^K}/N!K!$;
- More accurate: Polya's theory for counting (generating function and cycle index).

Symmetry induced by permutation groups

- The base symmetric groups: S_N and S_K;
- First induced permutation group: $\mathcal{W} \cup \mathcal{Z} \cup \mathcal{X} \to \mathcal{W} \cup \mathcal{Z} \cup \mathcal{X}$;
  - Compositions of any induced permutations by $\bar{\pi} \in S_K$ and $\hat{\pi} \in S_N$;
- Second induced permutation group: $2^{\mathcal{W} \cup \mathcal{S} \cup \mathcal{X}} \to 2^{\mathcal{W} \cup \mathcal{S} \cup \mathcal{X}}$.

Demand Types

Some demands are equivalent, but not all

- E.g., $N = 3, K = 5$: $(2, 2, 1, 1, 3)$ is equivalent to $(1, 3, 3, 2, 2)$, but not $(1, 1, 1, 2, 3)$;
- Symmetric optimal solutions exist, but only up to such symmetry;
- Demand type: represented as an N-dimensional non-negative integer vector, in decreasing order, that sums to K.

(N, K)	Demand types
$(2, 3)$	$(3, 0), (2, 1)$
$(2, 4)$	$(4, 0), (3, 1), (2, 2)$
$(3, 2)$	$(2, 0, 0), (1, 1, 0)$
$(3, 3)$	$(3, 0, 0), (2, 1, 0), (1, 1, 1)$
$(3, 4)$	$(4, 0, 0), (3, 1, 0), (2, 2, 0), (2, 1, 1)$
$(4, 2)$	$(2, 0, 0, 0), (1, 1, 0, 0)$
$(4, 3)$	$(3, 0, 0, 0), (2, 1, 0, 0), (1, 1, 1, 0)$

A Complete Characterization for $K = 2$

Theorem

For any integer $N \geq 3$, any memory-rate tradeoff pair for the $(N, K) = (N, 2)$ caching problem must satisfy

$$3M + NR \geq 2N, \quad M + NR \geq N. \tag{2}$$

Conversely, for any integer $N \geq 3$, there exist codes for any nonnegative (M, R) pair satisfying (2).

- The first slice of cases to have a complete solution;
- First investigate $N = 3, 4$ using the computational approach, then use the proofs to deduce a general pattern;
- This generalization is not computer-produced ☹.

A Complete Characterization for $K = 2$

Theorem

For any integer $N \geq 3$, any memory-rate tradeoff pair for the $(N, K) = (N, 2)$ caching problem must satisfy

$$3M + NR \geq 2N, \quad M + NR \geq N. \tag{2}$$

Conversely, for any integer $N \geq 3$, there exist codes for any nonnegative (M, R) pair satisfying (2).

- The first slice of cases to have a complete solution;
- First investigate $N = 3, 4$ using the computational approach, then use the proofs to deduce a general pattern;
- This generalization is not computer-produced ☹.

A Complete Characterization for $K = 2$

Theorem

For any integer $N \geq 3$, any memory-rate tradeoff pair for the $(N, K) = (N, 2)$ caching problem must satisfy

$$3M + NR \geq 2N, \quad M + NR \geq N. \tag{2}$$

Conversely, for any integer $N \geq 3$, there exist codes for any nonnegative (M, R) pair satisfying (2).

- The first slice of cases to have a complete solution;
- First investigate $N = 3, 4$ using the computational approach, then use the proofs to deduce a general pattern;
- This generalization is not computer-produced ☹.

A Complete Characterization for $K = 2$

Theorem

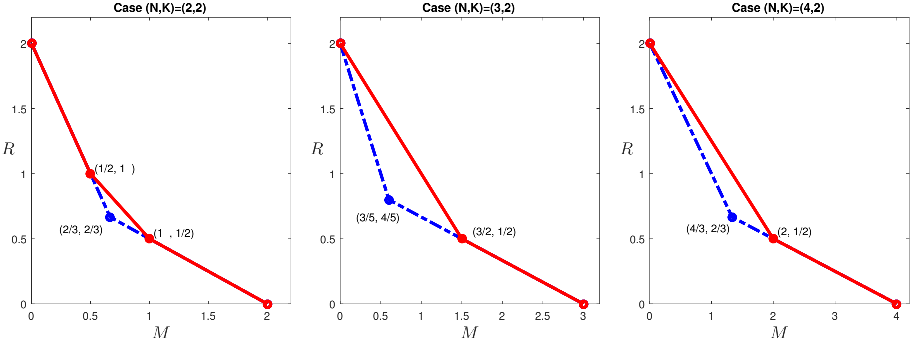
For any integer $N \geq 3$, any memory-rate tradeoff pair for the $(N, K) = (N, 2)$ caching problem must satisfy

$$3M + NR \geq 2N, \quad M + NR \geq N. \tag{2}$$

Conversely, for any integer $N \geq 3$, there exist codes for any nonnegative (M, R) pair satisfying (2).

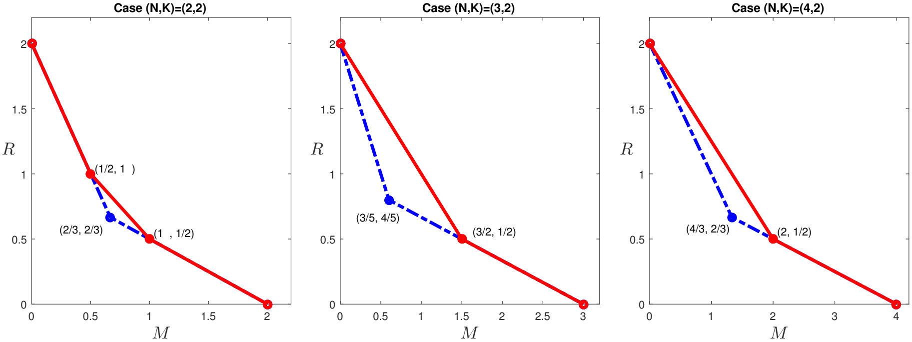
- The first slice of cases to have a complete solution;
- First investigate $N = 3, 4$ using the computational approach, then use the proofs to deduce a general pattern;
- This generalization is not computer-produced ☹.

How Did We Form This Hypothesis?



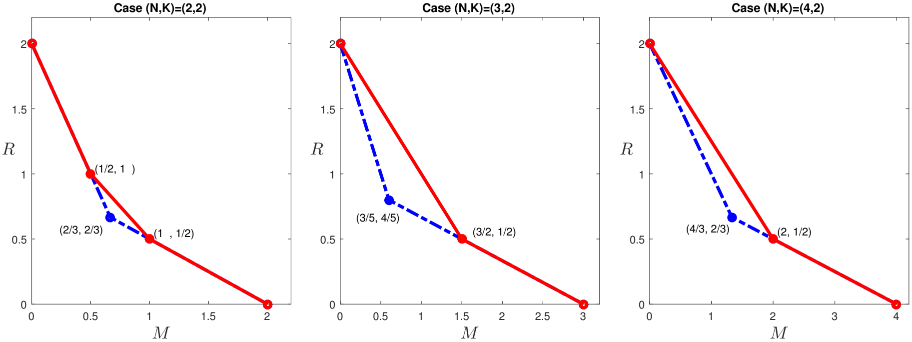
- $(N, K) = (2, 2)$ previously known: tradeoff has two corner points;
- Use the computational approach to first find solutions for $N = 3, 4$;
- For $N \geq 3$, has only one corner point (surprise!);
- Analyze the proofs and extend it to $N > 4$.

How Did We Form This Hypothesis?



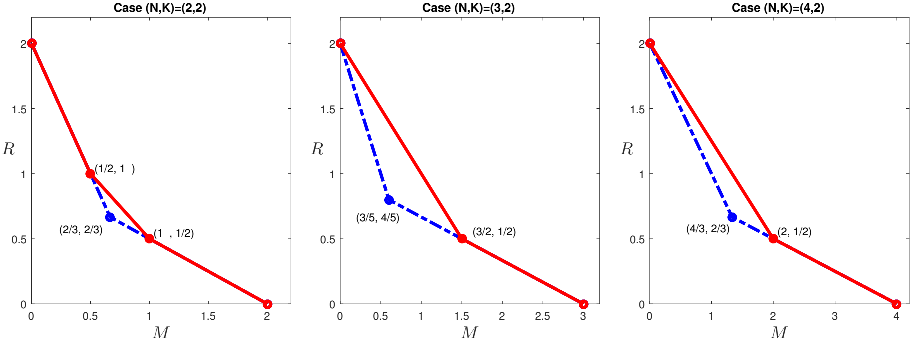
- $(N, K) = (2, 2)$ previously known: tradeoff has two corner points;
- Use the computational approach to first find solutions for $N = 3, 4$;
- For $N \geq 3$, has only one corner point (surprise!);
- Analyze the proofs and extend it to $N > 4$.

How Did We Form This Hypothesis?



- $(N, K) = (2, 2)$ previously known: tradeoff has two corner points;
- Use the computational approach to first find solutions for $N = 3, 4$;
- For $N \geq 3$, has only one corner point (surprise!);
- Analyze the proofs and extend it to $N > 4$.

How Did We Form This Hypothesis?



- $(N, K) = (2, 2)$ previously known: tradeoff has two corner points;
- Use the computational approach to first find solutions for $N = 3, 4$;
- For $N \geq 3$, has only one corner point (surprise!);
- Analyze the proofs and extend it to $N > 4$.

A Partial Characterization for $N = 2$

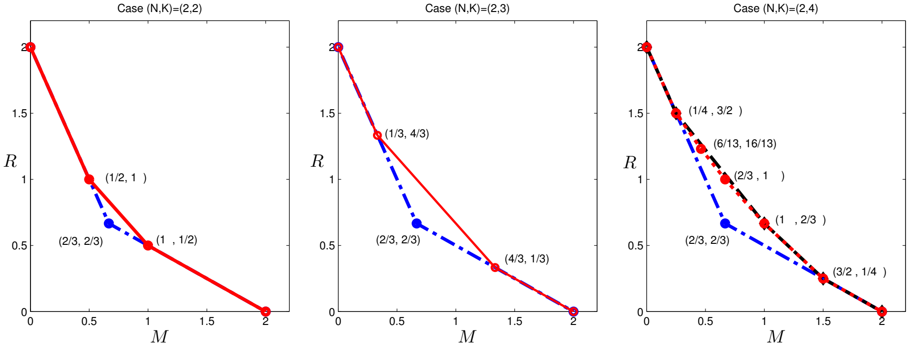
Theorem

When $K \geq 3$ and $N = 2$, any (M, R) pair must satisfy

$$K(K+1)M + 2(K-1)KR \geq 2(K-1)(K+2). \tag{3}$$

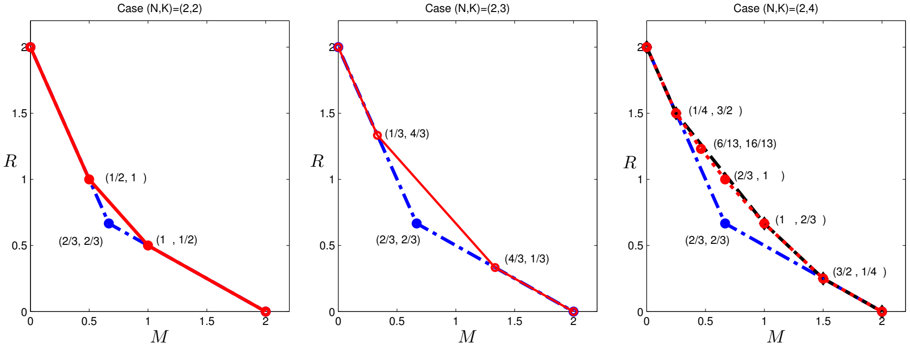
As a consequence, the Maddah-Ali-Niesen scheme is optimal when $M \geq \frac{2(K-2)}{K}$, for the cases with $K > 3$ and $N = 2$.

How Did We Form This Hypothesis?



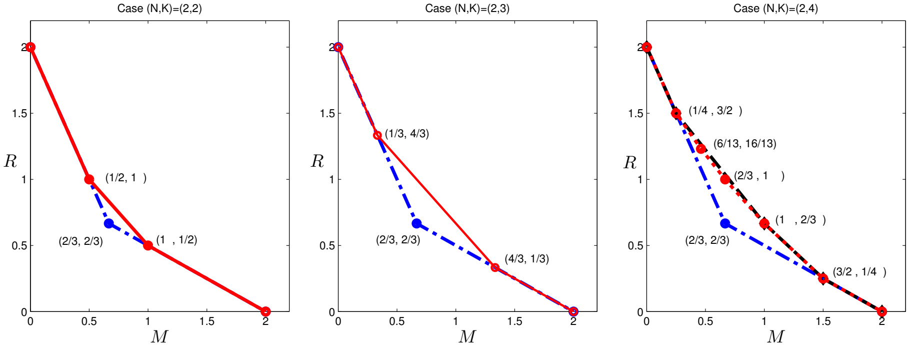
- $(N, K) = (2, 2)$ previously known;
- Use the computational approach to first find solutions for $K = 3, 4$;
- High memory regime the Maddah-Ali-Niesen scheme optimal;
- Analyze the computed generated proofs and extend it to $K > 4$.

How Did We Form This Hypothesis?



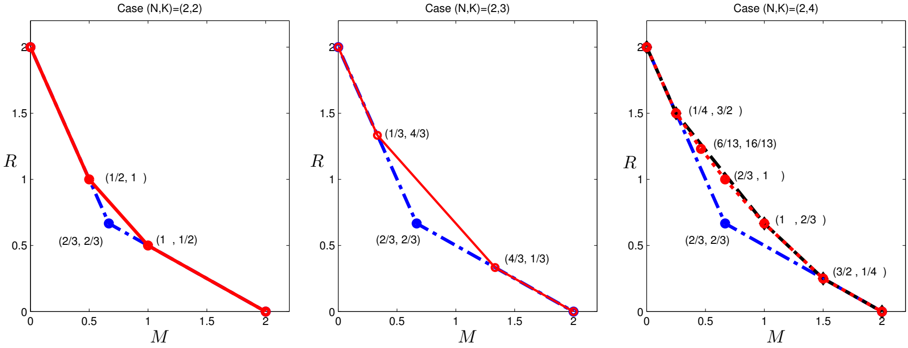
- $(N, K) = (2, 2)$ previously known;
- Use the computational approach to first find solutions for $K = 3, 4$;
- High memory regime the Maddah-Ali-Niesen scheme optimal;
- Analyze the computed generated proofs and extend it to $K > 4$.

How Did We Form This Hypothesis?



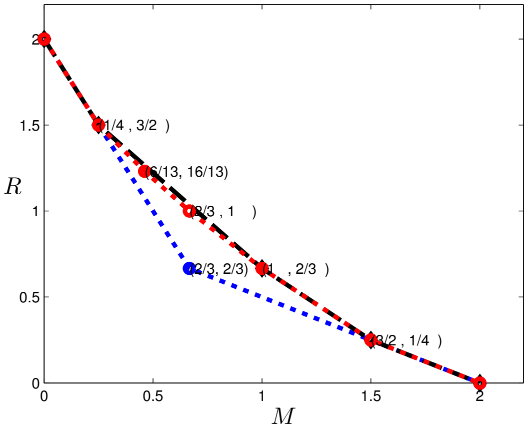
- $(N, K) = (2, 2)$ previously known;
- Use the computational approach to first find solutions for $K = 3, 4$;
- High memory regime the Maddah-Ali-Niesen scheme optimal;
- Analyze the computed generated proofs and extend it to $K > 4$.

How Did We Form This Hypothesis?



- $(N, K) = (2, 2)$ previously known;
- Use the computational approach to first find solutions for $K = 3, 4$;
- High memory regime the Maddah-Ali-Niesen scheme optimal;
- Analyze the computed generated proofs and extend it to $K > 4$.

Reverse-Engineering the Code for $(N, K) = (2, 4)$



- Bounds tight for $M \in [0, 1/4] \cup [1, 2]$;
- Investigate the bounds, identify a corner point not achievable;
- ASSUMING it achievable: attempt to design code (success ☺).

Be Nice If We Know More?

Assuming each file has 6 symbols in some finite field:

Joint entropy	Value∗6	$H(\cdot\|A)$
$H(A, Z_1)$	9	3
$H(A, Z_1, Z_2)$	11	5
$H(A, Z_1, Z_2, Z_3)$	12	6
$H(A, X_{1,2,2,2})$	9	3

Target: find a linear code with the given joint entropy structure

- Each user cache 3, combination of any two gives 5, any three gives 6;
- Delivery and cached are linear independent.

Be Nice If We Know More?

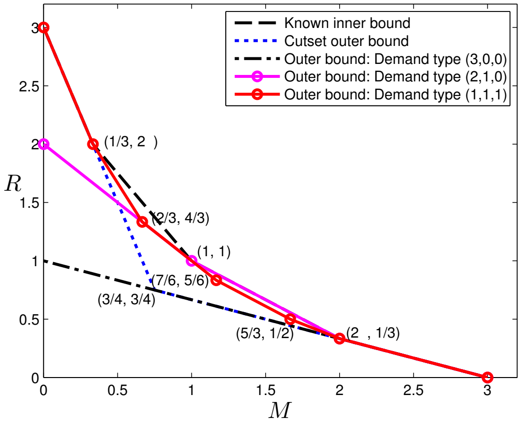
Assuming each file has 6 symbols in some finite field:

Joint entropy	Value$*$6	$H(\cdot\|A)$
$H(A, Z_1)$	9	3
$H(A, Z_1, Z_2)$	11	5
$H(A, Z_1, Z_2, Z_3)$	12	6
$H(A, X_{1,2,2,2})$	9	3

Target: find a linear code with the given joint entropy structure

- Each user cache 3, combination of any two gives 5, any three gives 6;
- Delivery and cached are linear independent.

Results for $N = K = 3$



- Bounds tight for $M \in [0, 1/3] \cup [1, 3]$.
- Investigate the bounds, identify a corner point;
- Assuming it achievable: attempt to design code (no luck ☹).

Outline

Conclusion

A new code construction for the caching problem

- Coded placement and coded transmission;
- Based on interference elimination;
- Roughly a dual of the Maddhuh-Ali-Niesen scheme.

New outer bound results

- Computer aided approach can provide important clues;
- The notion of demand types;
- Complete characterizations for $K = 2$;
- Partial characterizations for $N = 2$;
- A bunch of other bounds: many hypotheses in our target list.

Conclusion

A new code construction for the caching problem

- Coded placement and coded transmission;
- Based on interference elimination;
- Roughly a dual of the Maddhuh-Ali-Niesen scheme.

New outer bound results

- Computer aided approach can provide important clues;
- The notion of demand types;
- Complete characterizations for $K = 2$;
- Partial characterizations for $N = 2$;
- A bunch of other bounds: many hypotheses in our target list.

Conclusion

The conventional outer bound approach has too many human factors

- Reduce the human factors by introducing more machine intelligence;
- A more domain specific LP approach;
- Application on several research problems proves its effectiveness.
- More than proofs for simple inequalities: new insights, for both fundamental limits and code constructions.

The main challenge:

- High complexity: how much power can we squeeze out?
- Incorporating more domain knowledge into the approach?
- Computerized proof checking?
- Data-driven automatic hypothesis forming and proof?

Solutions of Computed Information-Theoretic Limits
http://web.eecs.utk.edu/~ctian1/SCITL.html

Conclusion

The conventional outer bound approach has too many human factors

- Reduce the human factors by introducing more machine intelligence;
- A more domain specific LP approach;
- Application on several research problems proves its effectiveness.
- More than proofs for simple inequalities: new insights, for both fundamental limits and code constructions.

The main challenge:

- High complexity: how much power can we squeeze out?
- Incorporating more domain knowledge into the approach?
- Computerized proof checking?
- Data-driven automatic hypothesis forming and proof?

Solutions of Computed Information-Theoretic Limits
http://web.eecs.utk.edu/~ctian1/SCITL.html

Conclusion

The conventional outer bound approach has too many human factors

- Reduce the human factors by introducing more machine intelligence;
- A more domain specific LP approach;
- Application on several research problems proves its effectiveness.
- More than proofs for simple inequalities: new insights, for both fundamental limits and code constructions.

The main challenge:

- High complexity: how much power can we squeeze out?
- Incorporating more domain knowledge into the approach?
- Computerized proof checking?
- Data-driven automatic hypothesis forming and proof?

Solutions of Computed Information-Theoretic Limits
http://web.eecs.utk.edu/~ctian1/SCITL.html

How about AlphaIT?

"The system could process much larger volumes of data and surface the structural insight to the human expert in a way that is much more efficient–or maybe not possible for the human expert..."–Demis Hassabis, Google Deepmind Leader.