Coding for Information of Different Versions

Zhiying Wang, Viveck Cadambe†, Tsachy Weissman*, Nancy Lynch†*

*Stanford University, †Massachusetts Institute of Technology

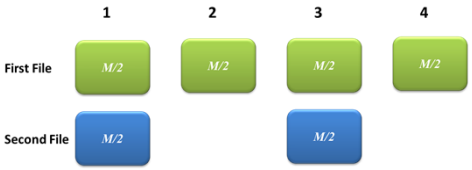
Introduction

Motivations

- Big file system with multiple servers as a network
- Frequent updates of information in an asynchronies manner
- Clients connected to a subset of servers
- Request for newest possible pieces of information
- Goal: minimize worst-case storage capacity

Problem settings

- n – Total number of servers
- k – Number of servers connected to the client
- M – File size of each version
- d – Total number of versions
- C – Capacity of each server to store all needed versions
- Files arrive at each server in order
- From *any* k servers, need to recover the newest common version



In this example, n=4 and suppose k=2. Encode every file version of size M using a (4,2) maximum distance separable (MDS) code. Each server requires a capacity of C=M. If a client connects to servers 1 and 3, decode the second file; otherwise decode the first file.

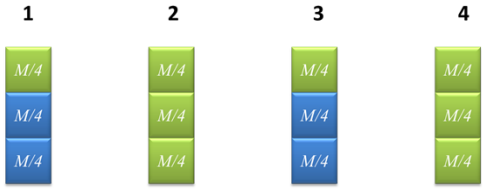
Code Construction

Construction

- Store a piece of size C for the first file
- For each new incoming file, discard certain amount of previous files
- The size of a total of any k pieces of newest common version sums up to M
- No coding among different versions

Theorem: Capacity of the above construction satisfies

$$C = \frac{dk - d + 1}{k^2} M.$$



In this example, k=2, every server stores information of size C=3M/4, saving a quarter compared to the naïve method. The first file (green) is encoded using a (12, 4) MDS code, and the second file (blue) is encoded using a (8, 4) MDS code. Check correctness:
From servers 2 and 4 can recover 1st file;
From servers 1 and 2 can recover 1st file;
From servers 1 and 3 can recover 2nd file.

Lower Bounds

Recovery Requirement

- k servers with d-th file
- One server with i-th file, for all i=1,2,…,d-1
- They together should recover all the d files

Theorem: A lower bound on the server capacity is

$$C \geq \frac{d}{k+d-1}$$

Improved Lower Bound

- Use the fact that every k servers recovers the newest common file
- Can show that non-coding method achieves optimal capacity for d=2.

Theorem: The capacity is $C \geq \frac{sM}{1+s}$

with $s = \left(\frac{k}{k-1}\right)^d - 1$

Sketch of Proof: Prove for special case of k=2, d=2. Let the 1st and 2nd files be random variables A, B. Consider a stage where 2 servers have the second file. Denote the information stored in them as random variables X, Y. Say $H(X|A) \geq H(Y|A)$, then one can show that

$$I(X;A) \geq M - C,$$
$$I(X;B|A) \geq M/2.$$
$$C \geq I(X;A,B)$$

Thus

$$C \geq I(X;A,B) = I(X;A) + I(X;B|A)$$
$$\geq 3M/2 - C$$

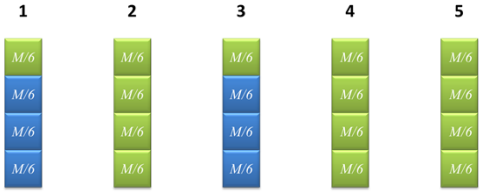
and

$$C \geq 3M/4.$$

Generalization

Hot and Cold Information

- Different $k(i)$ for the i-th file
- Typically newer files are hot and older files are cold
- $k(1) \geq k(2) \geq \ldots \geq k(d)$



In this example k(1)=3, k(2)=2. The first file (green) is encoded using a (20,6) MDS code and the second file (blue) uses a (15,6) MDS code. From servers 1 and 3, we can recover the second file. From any three servers, we can recover the first file. Notice that from server 1, 2 and 3 we can recover both files. The capacity is C = 2M/3, compared to $5M/6$ in the naïve method.

References

[1] U. Manber. "Finding similar files in a large file system." In *Proc. of the USENIX winter technical conference*, pages 1–10, Berkeley, CA, USA, 1994. USENIX Association.

[2] D. T. Meyer, and W. J. Bolosky, "A study of practical deduplication." *ACM Transactions on Storage (TOS)*, vol 7, no 4, pages 14, 2012.

[3] Y. Cassuto, and E. Yaakobi, "Short Q-ary WOM Codes with Hot/Cold Write Differentiation." *Proc. of the 2012 IEEE International Symposium on Information Theory, ISIT 2012, Boston* MA, USA, July 2012.