Local Processing in Distributed Storage

Arya Mazumdar

Based on Joint works with
Viveck Cadambe, Venkat Chander, Greg Wornell (MIT)
Ankit Rawat, Sriram Viswanath (UTAustin)

Arya Mazumdar - Univ. of Minnesota
An information storage graph

- Can store 1 bit in each vertex
- Content of any vertex can be determined by looking at the contents of its neighbors

\[ N(1) = \{2, 3, 4, 5\} \]
\[ N(2) = \{1, 3\} \]
\[ N(3) = \{1, 2, 4\} \]
\[ N(4) = \{1, 3, 5\} \]
\[ N(5) = \{1, 4\}. \]

How many bits of information can be stored in this graph?

- Information theoretic formulation
- Answer: wait till the end of the talk

Why ask this question?

Arya Mazumdar - Univ. of Minnesota
Large-scale distributed storage: Network of servers

- Link Failure
- Long queue
- Power-down
- Hardware problems
- System crash
Local repair

Minimize repair bandwidth (amount of data downloaded)

Arya Mazumdar - Univ. of Minnesota
Update-efficiency

Slight change in message
Not much change in the codeword

Good update-efficient codes that can handle node (disk) failures?
Challenges?

Unorthodox constraints – Local Recovery/Smooth Encoding

Optimal rate – Best codes (Information theory)

How to build codes? (Coding theory)

Explicit (fast algorithmic) constructions of Codes that support LOCAL PROCESSING
Outline

- Distributed Storage
  - Local Repair
  - Update Efficiency
  - Topology of Distributed Network Storage
Local repair
Repair of single failed node

Called regenerative codes

Minimize repair bandwidth (amount of data downloaded)
How coding helps?

Repetition (Protection against 1 erasure)

\[ \begin{array}{cccc}
A & A & B & B \\
\end{array} \]
Rate = 1/2

Coding

\[ \begin{array}{ccc}
A & B & A+B \\
\end{array} \]
Rate = 2/3

Arya Mazumdar - Univ. of Minnesota
Benefits of repetition

• Many sophisticated coding schemes
• STILL repetition is used .. WHY?
  
  – Local Recovery
Example:

Repetition (Protection against 1 erasure)

A, A, B, B

LOCALITY = 1
Rate = 1/2

A, B, A+B Coding

A, B, A+B

LOCALITY = 2
Rate = 2/3
Locally repairable codes (Gopalan, Huang, Simitci, Yekhanin, 2011)

1. Each Symbol: Stored at a different node of network
2. Each Symbol: Represents a packet or block of bits of arbitrary size
3. n: Number of servers  k: amount of information

Arya Mazumdar - Univ. of Minnesota
Local repair

- Gopalan, Yekhanin et al. (r: locality)

\[ d_{\text{min}} \leq n - k - \left\lceil \frac{k}{r} \right\rceil + 2. \]

- Constructions
  - Gopalan et al.; Dimakis et al. (existence)
  - Barg and Tamo

- Server size: any role?
New fundamental limits: with server size

\[ k \leq \min_{t \in \mathbb{Z}^+} \left[ tr + k_{opt}^{(q)}(n - t(r + 1), d) \right] \]

- q: size of the node
- Uses the optimal (unknown) dimension of error-correcting codes
- The last bound (Gopalan et al., Dimakis et al.) is a special case
- Tight constructions are possible at various points

CadambeMazumdar2013

Arya Mazumdar - Univ. of Minnesota
Local Recovery

Cooperative Local Recovery?
Cooperative local repair

• Systems may encounter multiple node failures in quick succession [Ford et al. 2010].
  – Deliberately accumulate failures and repair them simultaneously.
  – [Shum and Hu 2011] show that cooperative repair may lead to smaller repair bandwidth.

• We propose the cooperative repair with small locality as performance metric.
Cooperative repair

- $(n,k)$ code
- $(r, \ell)$ locality
- Any $\ell$ symbols of a codeword $\leftrightarrow r$ other symbols of the codeword.

Usual local repair $\ell = 1$. 
Hadamard (Simplex) codes

Simplex code: \([n = 2^k - 1, k, 2^{k-1}]_2\)

Each symbol can be recovered from 2 other symbol (Mazumdar, Chandar, Wornell, 2012, and Cadambe, Mazumdar, 2013)
Hadamard codes

\[(r = \ell + 1, \ell)\)-locality for any \(\ell \leq \frac{n-1}{2}\]

Mazumdar Rawat Vishwanath 2014

Why? Dual of Hamming code.

1. \((c_1, \ldots, c_n)\) is a codeword \(\implies c_i + c_{2j} = c_{i+2j}\) (local repairability)

2. The set of prefixes (of length \(2^{k'} - 1, k' < k\)) of codewords is also a Simplex code

Use induction then.

What about general codes?
First: What is possible

\[ \ell = 3, r = 4 \]

- How many codewords can have the same prefix?
- Codewords with same prefix: subcode with same distance
- Use bounds on codes

\[ r + \ell \text{ symbol contains } r \text{ amount of information.} \]

\[ t(r + \ell) \text{ symbol contains } tr \text{ amount of information. Say, this is the prefix.} \]
Limit of cooperative repair

\[ d_{\text{min}} \leq n - k + 1 - \ell \left( \left\lfloor \frac{k}{r} \right\rfloor - 1 \right). \]

Suppose we do not care about minimum distance. \((n, k)\) code with \((r, \ell)\)-locality must have:

\[ \frac{k}{n} \leq \frac{r}{r + \ell}. \]

Also have: bounds with server size as an argument

MazumdarRawatVishwanath2014

Arya Mazumdar - Univ. of Minnesota
Constructions

- Partition MDS codes

\[
\frac{r}{r + \ell^2} \leq R_{n,k}(r, \ell) \leq \frac{r}{r + \ell}.
\]
Tightness of the bounds

Main open question: how tight is the bound $\frac{r}{r+\ell}$ on rate?

Random distribution of $\ell$ errors: Rate can be arbitrarily close to $\frac{r}{r+\ell} \implies$ the bound is tight for probabilistic model of errors.
Random failures: Capacity of local repair

\[ \epsilon \text{-away-from-capacity} \quad \Rightarrow \quad \text{local recoverability} \quad \Omega(\log \frac{1}{\epsilon}) \]

For any code \( C \) of length \( n \) and rate \( 1 - p - \epsilon \) that achieves probability of error less than \( \delta \) for any \( \delta > 0 \) when used on a BEC\((p)\), its local recoverability is at least \( c \log \frac{1}{\epsilon} \), for some constant \( c > 0 \).
Concentration of output entropy

- Entropy of the unerased variables: 1-Lipschitz (of random erasures by the channel)

- Average (output entropy = #unerased variables − #useless variables)

- Useless variables = Non-erased but the recovery sets erased (estimate this)

- Use the estimated output entropy in (say) Fano’s inequality
Update-efficiency
A Code: Mapping from Messages to Codewords (encoding)

0 → 00
1 → 11

00 → 000
01 → 011
10 → 101
11 → 110

Repetition

A, B, A+B Code

0000 → 0000000
0001 → 0001111
0010 → 0010110
0011 → 0011001
0100 → 0100101
0101 → 0101010
0110 → 0110011
0111 → 0111100
1000 → 1000011
1001 → 1001100
1010 → 1010101
1011 → 1011010
1100 → 1100110
1101 → 1101001
1110 → 1110000
1111 → 1111111

[7,4,3] Hamming Code (Corrects 2 erasures)
Update-efficient codes for distributed storage

Message Codewords

Slight change in message Not much change in the codeword

Good update-efficient codes that can handle node (disk) failures?
Update-efficiency: Why?

- Frequently changing DATABASE
- Social NETWORKS
- Changes are SMALL, but updating consumes
  - Bandwidth
  - Energy

Measure: Maximum Number of Symbols changed per one symbol change in message (we want sublinear growth).
Example:

00 → 000
01 → 011
10 → 101
11 → 110

A, B, A+B Code

0000 → 0000000
0001 → 0001111
0010 → 0010110
0011 → 0011001
0100 → 0100101
0101 → 0101010
0110 → 0110011
0111 → 0111100
1000 → 1000011
1001 → 1001101
1010 → 1010101
1011 → 1011010
1100 → 1100110
1101 → 1101001
1110 → 1110000
1111 → 1111111

Update-efficiency = 2


Arya Mazumdar - Univ. of Minnesota
Minimum distance of codes

- Pairwise minimum distance between any two codewords.
  \[ d = t + 1, \quad t = \text{number of correctable erasures} \]

- Update-efficiency \( u \geq d \)

- We want:
  - \( u \) small
  - \( t \) large

- NOT going to happen

- When \( u = d \): The code is OPTIMAL
  - Hamming code is OPTIMAL

Arya Mazumdar - Univ. of Minnesota
Linear codes: generator matrix

c = Gx

c = codeword; x = message
c: n dimensional; x: k dimensional; G: n by k matrix

- Update-efficiency = Maximum row weight of a generator matrix
- Update-efficient codes must have an LDGM representation
Construction of codes

- $[n,k,d]$ linear code $\Rightarrow$ An update efficient $[n,k,d]$ code with $u = d$ (Simonis 1992)
- How: with an exponential time algorithm
- Task: Can we do this in polynomial time?
Long codes

- n is large
- d = δn

- u is at least d
- Unacceptable

- Assume RANDOM error (like a Binary Erasure Channel)
- Next results hold for larger alphabets as well
Capacity achieving codes

- \( n \) is large
- Capacity of BEC = 1 - \( \delta \)

Results:

- Can have a family of codes with
  - Rate = 1 - \( \delta \)
  - \( u = O(\log n) \)
  - This construction is explicit

- Cannot have a code with
  - Rate > 0
  - \( u = c \log n \); \( c \) is a constant.
  - Follows from concentration output entropy

Arya Mazumdar - Univ. of Minnesota
Topology of distributed storage network
A recoverable distributed storage network

- All servers are not connected to each other
- Some links are easy to establish
- Consideration for
  - Physical Proximity
  - Architecture
  - Platform
  - Connections

Arya Mazumdar - Univ. of Minnesota
How much storage is possible?

Storage graph: \( G(V, E) \)
\[ V = \{1, 2, \ldots, n\}. \]

\[ N(1) = \{2, 3, 4, 5\} \]
\[ N(2) = \{1, 3\} \]
\[ N(3) = \{1, 2, 4\} \]
\[ N(4) = \{1, 3, 5\} \]
\[ N(5) = \{1, 4\}. \]
Formally ...

- \( V = \{1, 2, \ldots, n\} \)
- \( X_1, X_2, \ldots, X_n \) : content of the vertices, \( X_i \in \mathbb{F}_q, i = 1, \ldots, n. \)
- RDSS code \( \mathcal{C} \subseteq \mathbb{F}_q^n \)
- A set of deterministic recovery functions, \( f_i : \mathbb{F}_q^{\left| N(i) \right|} \rightarrow \mathbb{F}_q \) for \( i = 1, \ldots, n \)
- For any codeword \( (X_1, X_2, \ldots, X_n) \in \mathbb{F}_q^n \),

\[
X_i = f_i(\{X_j : j \in N(i)\}), \quad i = 1, \ldots, n.
\]
Results!

How much storage is possible? Answer: Don’t even know the hardness

But say the answer is $= \text{RDSS}$

Explicit code: with storage at least
\[ \text{RDSS}/ \log (\#\text{Servers}) \log \log (\#\text{Servers}) \]
Sketch

RDSS $\leq$ Feedback Vertex Set

Construct a vertex disjoint cycle packing
However, more interestingly ...

The Recoverable Distributed Storage Problem

An approximate Reduction

Index Coding Problem

Use existing algorithms from the Index coding problem!

\[ n - \text{RDSS}(G) \leq \text{INDEX}(G) \leq n - \text{RDSS}(G) + O(\log n) \]

Mazumdar2014 Also see ShanmugamDimakis2014
Reduction to a dual problem! **Index coding**

- A set of \( n \) users \( \{1, 2, \ldots, n\} \)
- Each want some information \( \{x_1, x_2, \ldots, x_n\} \)
- Each has some information \( \{S_1, S_2, \ldots, S_n\} \) (side information graph \( G \))
  \[ S_i \subseteq \{x_1, x_2, \ldots, x_n\} \]

How many bits should be **BROADCAST** so that everyone gets what they need? **INDEX(G)**

Bar-Yossef, Birk, Jayram, Kol, ‘06

Alon, Hasidim, Lubetzky, Stav, Weinstein, ‘08
RDSS \neq n - INDEX

Codewords
\{00000, 01100, 00011, 11011, 11101\}

\[ RDSS_2(G) = \log_2 5 \text{ and } \text{INDEX}_2^2(G) = 3 \text{ (achieved by linear functions!)} \]
An index code of length $\ell \Rightarrow$ an RDSS code of dimension $n - \ell$
Easy to see.
An RDSS code of dimension $k \Rightarrow$ An index code of length $n - k + \log_q(\ln 2 + k \ln q)$ for the side information graph $G$.
Proof in arXiv.

\[
n - \text{RDSS}_q(G) \leq \text{INDEX}_q(G) \leq n - \text{RDSS}_q(G) + \log_q \left( \ln 2 + \text{RDSS}_q(G) \ln q \right).
\]
Network information flow

- Index coding is the **hardest instance** of network coding
  [RouayhebEtAl2010, LangbergEtAl2012, BlasiakKleinbergLubetzky2011]

- For general network coding, **only linear function capacity** is known to be NP Hard
  [LehmanRasala2004]

- RDSS is coding theoretic dual of INDEX : A Basic problem

- Example of one instance which is not that hard to solve
Thank You