On Theory and (Little) Practice of Coding Techniques for Distributed Networked Storage Systems

Frédérique Oggier
Joint work with Anwitaman Datta and Lluís Pàmies-Juárez

Nanyang Technological University, Singapore

Institute of Network Coding, Chinese University of Hong Kong,
January 4 2012

Outline

1. Coding for Distributed Networked Storage
2. Self-Repairing Codes: Constructions and Properties
3. A Little Bit of Practice

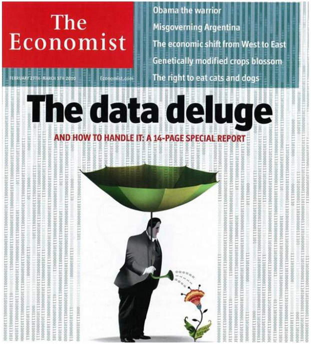
Distributed Networked Storage

- A data owner wants to *store* data over a network of nodes (e.g. data center, back-up or archival in peer-to-peer networks).
- Redundancy is essential for resilience (*Failure is the norm, not the exception*).
- Data from Los Alamos National Laboratory (Dependable Systems and Networks, 2006), gathered over 9 years, 4750 machines and 24101 CPUs. Distribution of failures:
  - Hardware 60%,
  - Software 20%,
  - Network/Environment/Humans 5%,

  Failures occurred between once a day to once a month.

What's New: More Numbers

- As of June 2011, a study sponsored by the information storage company EMC estimates that the world's data is more than doubling every 2 years, and reaching 1.8 zettabytes (1 zettabyte=10^{21} bytes) of data to be stored in 2011.[1]



- If you store this data on DVDs, the stack would reach from the earth to the moon and back.

[1] http://www.emc.com/about/news/press/2011/20110628-01.htm

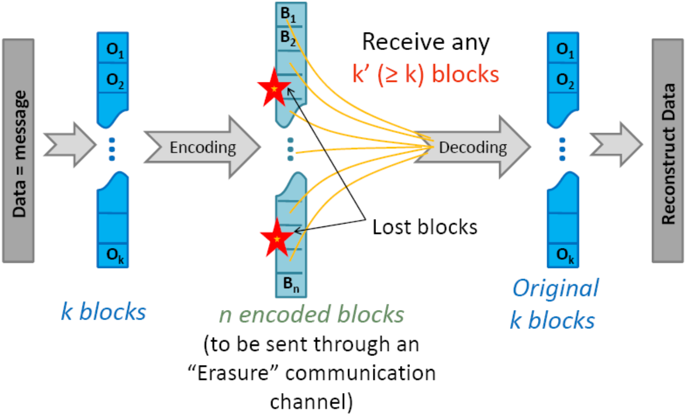
Redundancy through Coding

- *Replication*: good availability and durability, but very costly.
- *Erasure codes*: good trade-off of availability, durability and storage cost.

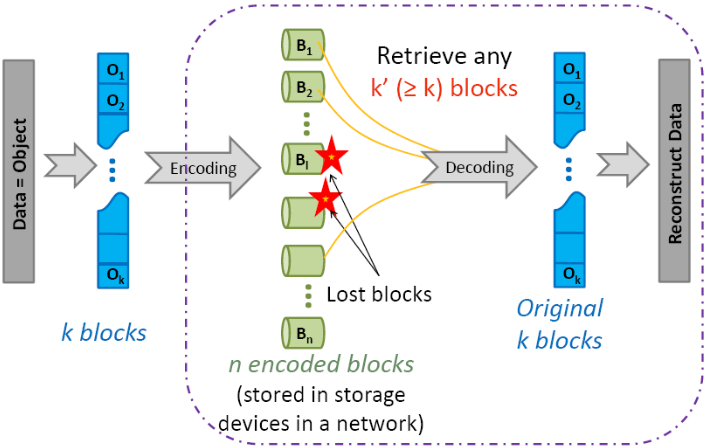
Erasure Codes

- A map that takes as input k blocks of data and outputs n blocks of data, $n - k$ of them thus giving redundancy.
- An (n, k) *erasure code* is characterized by (1) how many blocks are needed to decode (recover) the k blocks of original data - if any choice of k encoded blocks can do, the code is called *maximum distance separable (MDS)* and (2) its rate k/n (or storage overhead n/k).
- 3 way replication is a (3,1) erasure code.

Erasure codes for communication



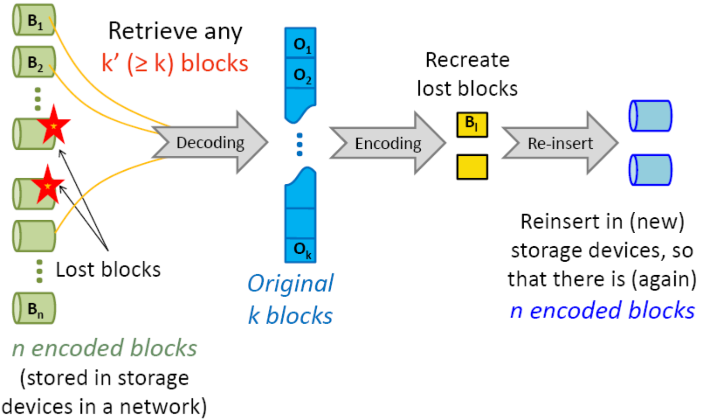
Erasure codes for storage systems



Codes for Storage: Repair

- Nodes may go offline, or may fail, so that the data they store becomes *unavailable*.
- Redundancy needs to be *replenished*, else data may be permanently lost over time (after multiple storage node failures)

Repair process using traditional Erasure Codes

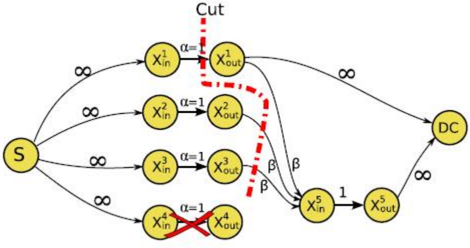


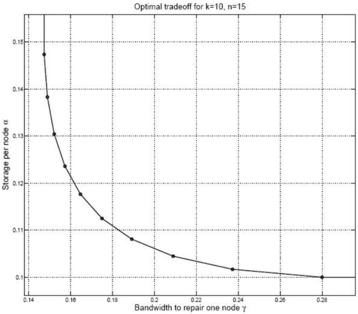
Related work

1. J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. OceanStore: An Architecture for Global-Scale Persistent Storage, ASPLOS 2000.
2. H. Weatherspoon, J. Kubiatowicz. Erasure Coding Vs. Replication: A Quantitative Comparison, Peer-to-Peer Systems, LNCS, 2002.
3. A. G. Dimakis, P. Brighten Godfrey, M. J. Wainwright, K. Ramchandran, The Benefits of Network Coding for Peer-to-Peer Storage Systems, Netcod 2007.
4. A. Duminuco, E. Biersack, Hierarchical Codes: How to Make Erasure Codes Attractive for Peer-to-Peer Storage Systems, Peer-to-Peer Computing (P2P), 2008.
5. K. V. Rashmi, N. B. Shah, P. V. Kumar and K. Ramchandran, Explicit Construction of Optimal Exact Regenerating Codes for Distributed Storage, Allerton Conf. on Control, Computing and Comm., 2009.
6. A.-M. Kermarrec, N. Le Scouarnec, G. Straub, Repairing Multiple Failures with Coordinated and Adaptive Regenerating Codes, NetCod 2011.

Regenerating Codes

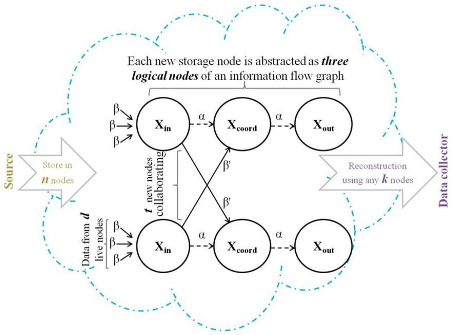
- Based on Network Coding (max flow-min cut argument) on top of an MDS (n, k) erasure code.
- Characterize storage overhead - repair bandwidth trade-off.
- Number of contacted live nodes to repair is at least k.

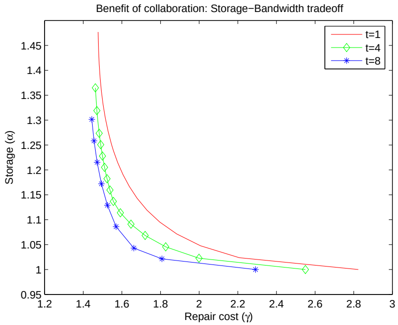




Collaborative Regenerating Codes

- Allow collaboration among new comers.
- Improve the storage overhead - repair bandwidth trade-off.
- Tolerates multiple faults.





Codes for Storage: Wish List

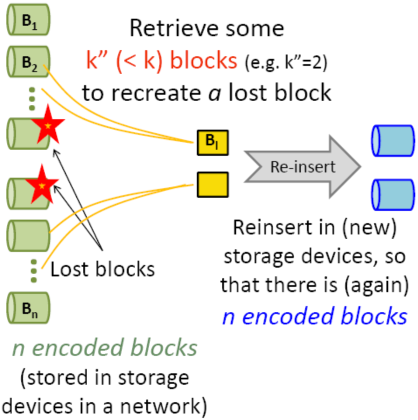
- Low storage overhead,
- Good fault tolerance,
- Low repair bandwith cost,
- Low repair time,
- Low complexity,
- I/O
- ...

Outline

Self-Repairing Codes (SRC)

- Motivation: *minimize* the number of nodes necessary to repair a missing block.
  - The minimum is 2, cannot be achieved without sacrificing the MDS property.
- Self-repairing codes are (n, k) codes such that
  - encoded fragments can be repaired directly from other subsets of encoded fragments,
  - a fragment can be repaired from a fixed number of encoded fragments (typically 2 or 3), independently of which specific blocks are missing (analogous to erasure codes supporting reconstruction using any $n - k$ losses, independently of which).

Self-Repairing Codes (a black-box view)

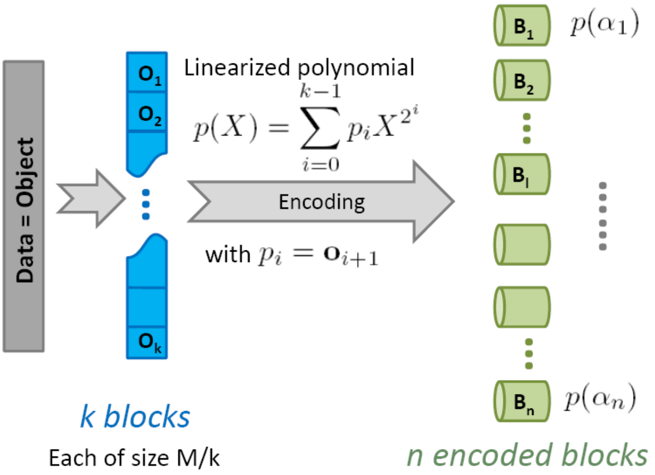


Homomorphic SRC (HSRC)

- A first instance of self-repairing code.
- Based on polynomial evaluation.
- An object is cut into k pieces, which represent coefficients of a polynomial p. The k pieces are mapped to n encoded fragments, by performing n polynomial evaluations $(p(\alpha_1), \ldots, p(\alpha_n))$.

Self-repairing Homomorphic Codes for Distributed Storage Systems
F. Oggier, A. Datta, *INFOCOM 2011*

HSRC: Encoding Illustration



HSRC: Decoding and Repair

1. *Decoding* is ensured by Lagrange interpolation.
2. *Repair*: $p(a+b) = p(a) + p(b)$.
3. Computational cost of a repair: XORs.

HSRC: A toy example

- Cut a file into $k = 3$ fragments, which serve as coefficients for a polynomial p.
- For $n = 7$, evaluate $p(X)$ at say $1, w, w^2, w^4, w^5, w^8, w^{10}$. We get:

$$(p(1), p(w), p(w^2), p(w^4), p(w^5), p(w^8), p(w^{10}))$$

missing fragment(s)	pairs to reconstruct missing fragment(s)
$p(1)$	$(p(w), p(w^4))$;$(p(w^2), p(w^8))$;$(p(w^5), p(w^{10}))$
$p(w)$	$(p(1), p(w^4))$;$(p(w^2), p(w^5))$;$(p(w^8), p(w^{10}))$
$p(w^2)$	$(p(1), p(w^8))$;$(p(w), p(w^5))$;$(p(w^4), p(w^{10}))$
$p(1)$ and $p(w)$	$(p(w^2), p(w^8))$ or $(p(w^5), p(w^{10}))$ for $p(1)$ $(p(w^8), p(w^{10}))$ or $(p(w^2), p(w^5))$ for $p(w)$
$p(1)$ and $p(w)$ and $p(w^2)$	$(p(w^5), p(w^{10}))$ for $p(1)$ $(p(w^8), p(w^{10}))$ for $p(w)$ $(p(w^4), p(w^{10}))$ for $p(w^2)$

Self-Repairing Codes from Projective Geometry (PSRC)

- A second instance of self-repairing code, based on spreads.
- Spread=partition of the space into subspaces, nodes store inner product of the data with basis vectors of subspaces.

Self-Repairing Codes for Distributed Storage - A Projective Geometric Construction, F. Oggier, A. Datta, *ITW 2011*

PSRC: A toy example

node	basis vectors	data stored
N_1	$v_1 = (1000)$, $v_2 = (0110)$	$\{o_1, o_2 + o_3\}$
N_2	$v_3 = (0100)$, $v_4 = (0011)$	$\{o_2, o_3 + o_4\}$
N_3	$v_5 = (0010)$, $v_6 = (1101)$	$\{o_3, o_1 + o_2 + o_4\}$
N_4	$v_7 = (0001)$, $v_8 = (1010)$	$\{o_4, o_1 + o_3\}$
N_5	$v_9 = (1100)$, $v_{10} = (0101)$	$\{o_1 + o_2, o_2 + o_4\}$

Static resilience

- There is at least one pair to repair a node, for up to $(n-1)/2$ simultaneous failures
- Static resilience of a distributed storage system is the probability that an object stored in the system stays available without any further maintenance, even when a fraction of nodes become unavailable.

Static resilience: HSRC versus EC

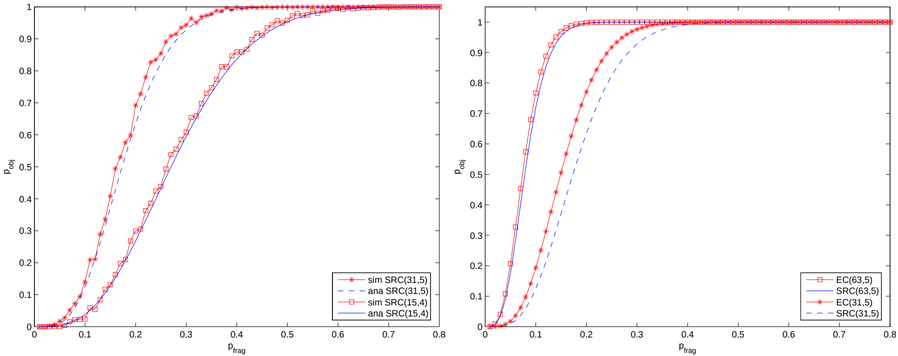
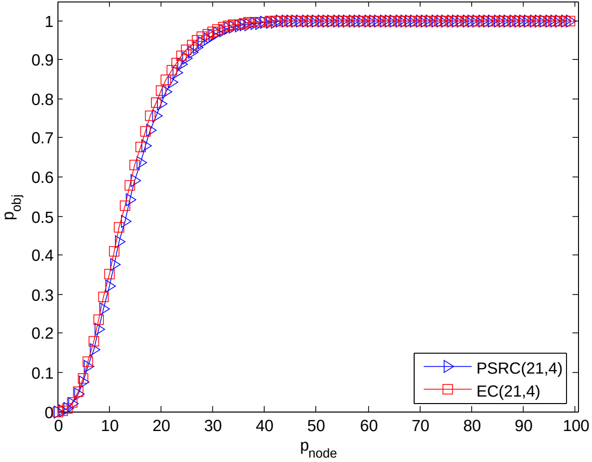
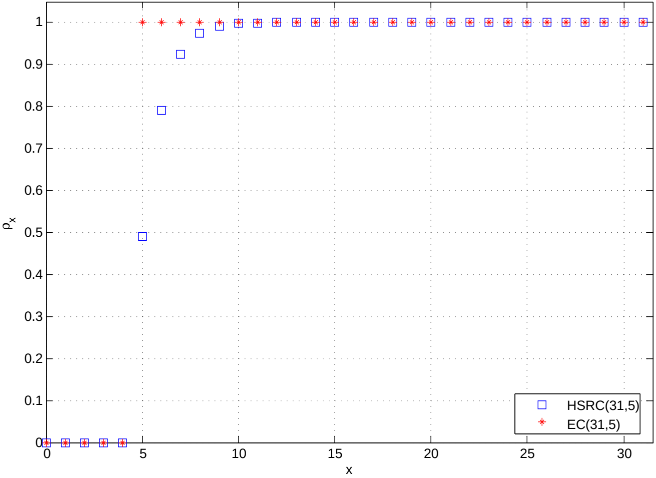


Figure: Static resilience of self-repairing codes (SRC): Validation of analysis, and comparison with erasure codes (EC)

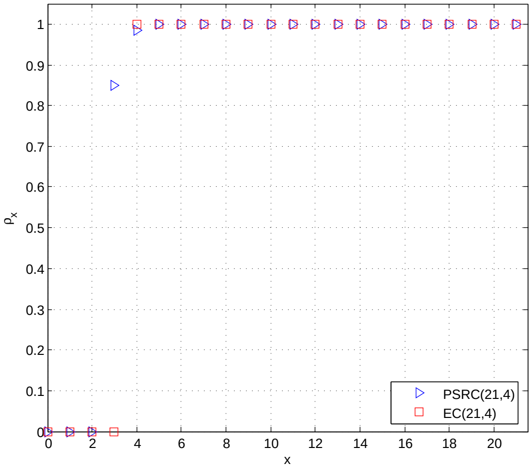
Static resilience: PSRC versus EC



More on Resilience: HSRC versus EC



More on Resilience: PSRC versus EC



Fast & parallel repairs using HSRC: A toy example

- Consider:
  - (15,4) code, nodes storing $p(w^i)$ for $i = 0, 1, 2, 3, 4, 5, 6$ are missing
  - Nodes have upload/download bandwidth limit: one block per time unit
- Possible pairs to repair each missing block:

fragment	suitable pairs to reconstruct
$p(1)$	$(p(w^7), p(w^9));(p(w^{11}), p(w^{12}))$
$p(w)$	$(p(w^7), p(w^{14}));(p(w^8), p(w^{10}))$
$p(w^2)$	$(p(w^7), p(w^{12}));(p(w^9), p(w^{11}));(p(w^{12}), p(w^{10}))$
$p(w^3)$	$(p(w^8), p(w^{13}));(p(w^{10}), p(w^{12}))$
$p(w^4)$	$(p(w^9), p(w^{14}));(p(w^{11}), p(w^{13}))$
$p(w^5)$	$(p(w^7), p(w^{13}));(p(w^{12}), p(w^{14}))$
$p(w^6)$	$(p(w^7), p(w^{10}));(p(w^8), p(w^{14}))$

- A parallelized schedule:

node	$p(w^0)$	$p(w^1)$	$p(w^2)$	$p(w^3)$	$p(w^4)$	$p(w^5)$	$p(w^6)$
Time 1	$p(w^7)$	$p(w^8)$	$p(w^9)$	$p(w^{13})$	$p(w^{11})$	$p(w^{12})$	$p(w^{10})$
Time 2	$p(w^9)$	$p(w^{10})$	$p(w^{11})$	$p(w^8)$	$p(w^{13})$	$p(w^{14})$	$p(w^7)$

Systematic Object Retrieval using PSRC: A toy example

node	basis vectors	data stored
N_1	$v_1 = (1000),\ v_2 = (0110)$	$\{o_1, o_2 + o_3\}$
N_2	$v_3 = (0100),\ v_4 = (0011)$	$\{o_2, o_3 + o_4\}$
N_3	$v_5 = (0010),\ v_6 = (1101)$	$\{o_3, o_1 + o_2 + o_4\}$
N_4	$v_7 = (0001),\ v_8 = (1010)$	$\{o_4, o_1 + o_3\}$
N_5	$v_9 = (1100),\ v_{10} = (0101)$	$\{o_1 + o_2, o_2 + o_4\}$

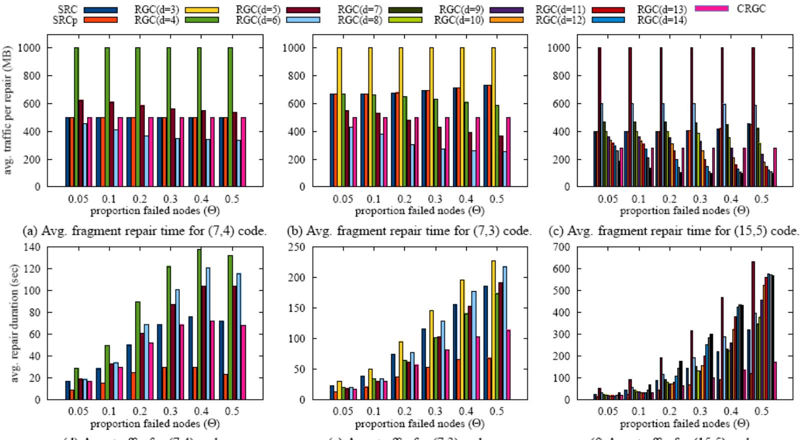
Outline

A More Realistic Scenario

- A network with 1000 (full duplex) nodes,
- 10 000 objects of size 1GB are stored,
- Multiple failures.

Pipelined codes are also considered.

Simulation Results: Storage of Multiple Objects



(a) Avg. fragment repair time for (7,4) code. (b) Avg. fragment repair time for (7,3) code. (c) Avg. fragment repair time for (15,5) code.

(d) Avg. traffic for (7,4) code. (e) Avg. traffic for (7,3) code. (f) Avg. traffic for (15,5) code.

Figure 4: Analysis of the system performance using different codes when a fraction Θ of nodes fails simultaneously. The size of the objects is $B = 1$GB, $L = 10,000$ objects are randomly stored in $N = 1,000$ nodes.

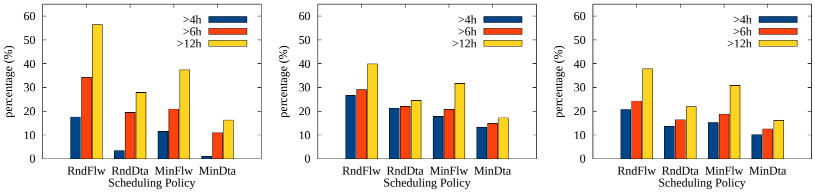
Data Insertion

- Replication: To store a new object, a source node uploads one replica to a first node, which can concurrently forward it to another storage node, etc
- Erasure Codes: The source node computes and uploads the encoded fragments to the corresponding storage nodes.
- *Issue*: insertion time, possibly worsened by mismatched temporal constraints (e.g. F2F).

In-Network Redundancy Generation for Opportunistic Speedup of Backup, L. Pamies-Juarez, A. Datta, F. Oggier *preprint*

Simulation Results: In-Network Coding

- IM traces for F2F scenario
- Figure (1): storage throughput increases with node availability. Figure (2): total traffic increases, scales with storage throughput. Figure (3): reduction of data upload at the source, up to 40%.



Future/ongoing work

- Efficient decoding, other instances of SRC
- Implementation & integration in a distributed storage system
- Various systems/algorithmic issues: Topology optimized placement, repair scheduling

Q&A

- More information: http://sands.sce.ntu.edu.sg/CodingForNetworkedStorage/
- Contact: {frederique,anwitaman}@ntu.edu.sg