Regenerating Codes: Theory and Practice

Preetum Nakkiran

Department of EECS, UC Berkeley

Overview

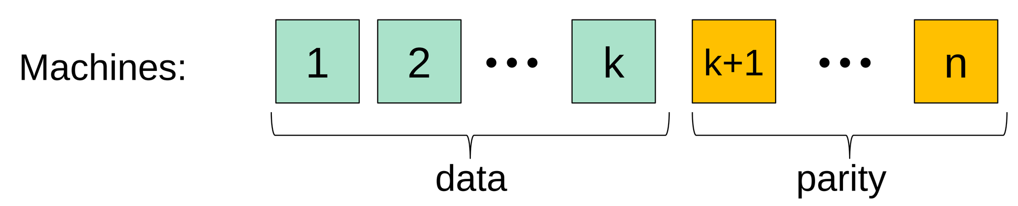
Distributed storage systems are increasingly using erasure codes, instead of replication, for fault-tolerance. While traditional codes provide significant savings in storage, they require large network bandwidth to reconstruct a small amount of missing data (eg, when a machine fails). A recently-proposed class of "regenerating codes" address this bandwidth problem.

Here we investigate various theoretical and practical aspects of regenerating codes.

Background

Erasure Codes in Distributed Storage:

Split file into '*k*' blocks, and compute '*r*' additional parities. Store blocks on *n=k+r* machines.



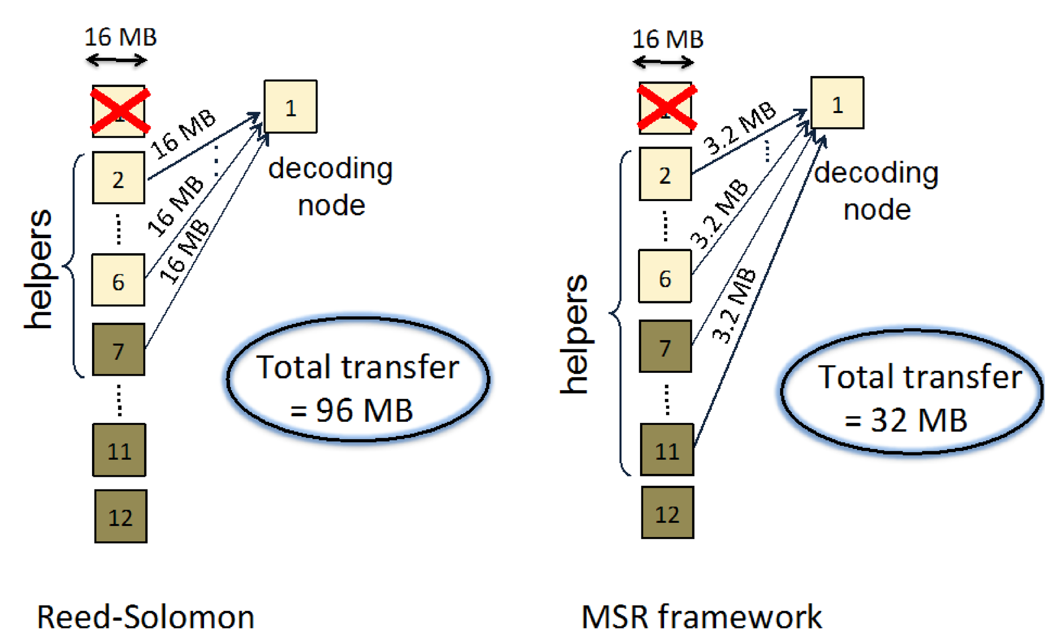
Failure model: Individual machines fail, but we want our data to survive. Want to recover the data from ***any k*** (of n) surviving machines.

(n,k) Reed-Solomon code:

- Problem: When one node fails ("node repair"), must download entire data & re-encode to repair.

MSR regenerating code:

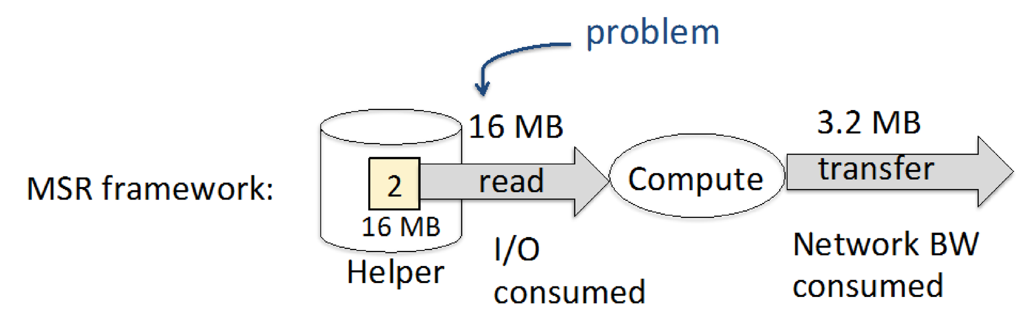
- Minimal "repair-bandwidth" among MDS codes.
- Retains fault-tolerance of RS code.

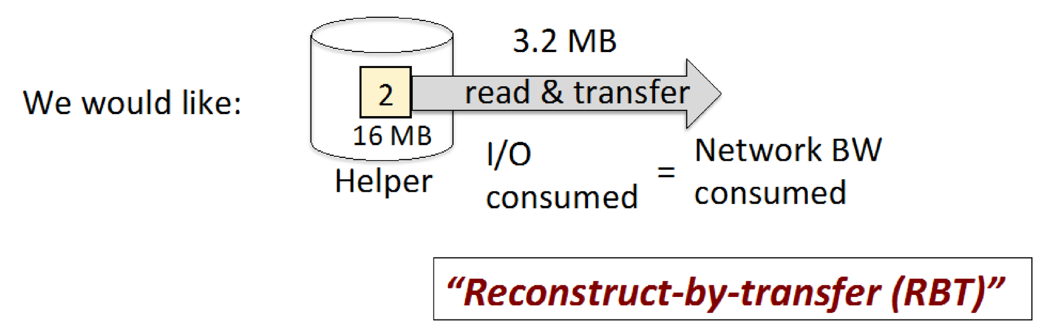


1. Optimizing Codes for I/O, Storage & Bandwidth

Joint work with: KV Rashmi, Jingyan Wang, Nihar Shah, and Kannan Ramchandran
In USENIX FAST 2015.

Problem: MSR codes are optimal w.r.t. storage & repair-bandwidth. But they have high disk I/O in repair:



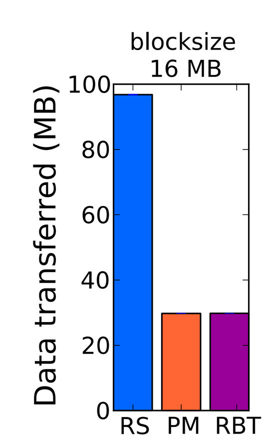


Can we construct codes optimal for disk I/O as well?

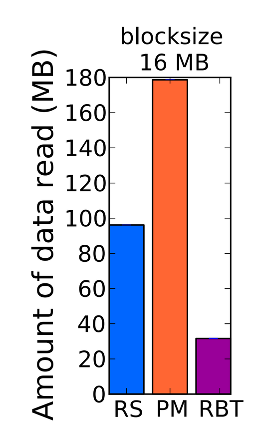
Our Results:

- Explicit transformation to locally-minimize disk I/O.
- Algorithm to globally-minimize expected disk I/O (under probabilistic failure model).

Our algorithms provide significant reduction in IOPS consumed, ~5x for typical parameters.



RBT: Same bandwidth as MSR ("PM")



RBT: Minimizes Disk IO

2. Understanding and Constructing Sparse Regenerating Codes

Joint work with: KV Rashmi
In preparation.

Problem: The additional structure of MSR codes often comes at the cost of code-complexity.

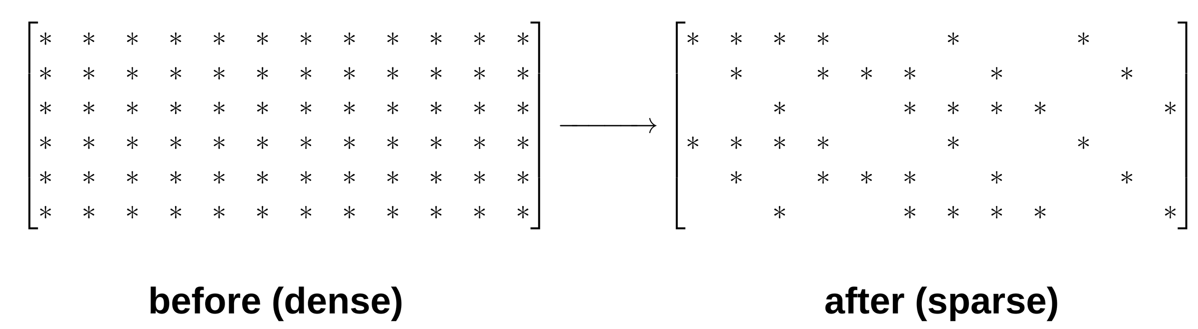
- (n, k) Reed-Solomon code:
  blocksize = **k** symbols.
- Same redundancy MSR code:
  blocksize = **k²** symbols.
  (slower encoding)

Can we construct and understand the structure of sparse regenerating codes?

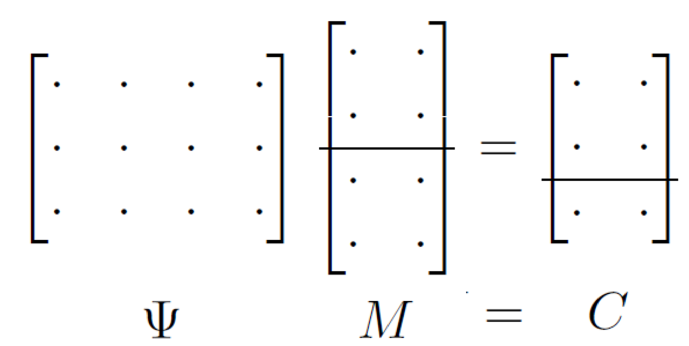
Our Results:

- MSR codes with sparsity **O(k)** per-symbol. (Based on Product-Matrix codes).
- General connection between "repair-by-transfer" (RBT) and sparsity.
- General framework for understanding systematic-remapping in MSR codes.

Generator matrix for parity nodes:



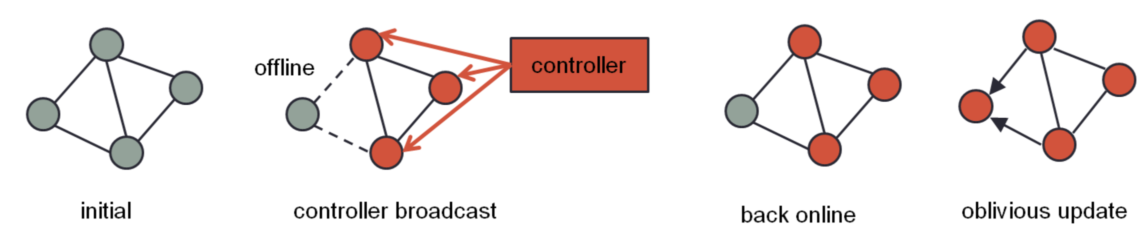
Product-Matrix encoding:



3. Communication Complexity of Oblivious Updates

Joint work with: Nihar Shah and KV Rashmi
In IEEE GLOBECOM 2014.

Problem: When data gets updated, can stale nodes get updated in a decentralized fashion? (Stale nodes update from updated nodes, without central controller).



All nodes store encoded data.

One node offline during update .

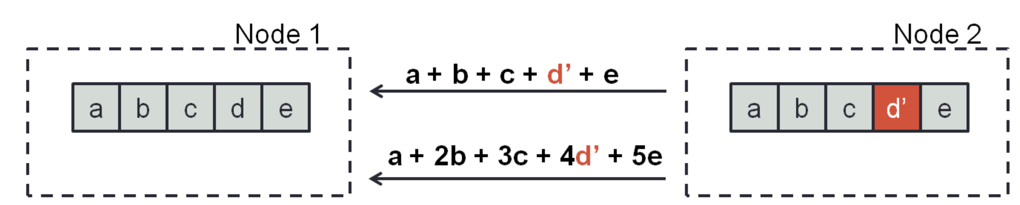
Stale node "obliviously" updates from other nodes.

Differences from Node Repair:

- Node repair: Assumes total node failure – no useful stored data
- Oblivious update: Stale node has stale data – potentially useful

Can we do (much) better than node repair?

Toy Example:



Our Results:

- Lower-bounds for linear codes:
  - ➢ Total download ≥ **2** × (change size)
- Lower-bounds for linear (n,k) MDS codes:
  - ➢ Total download ≥ **2k** × (change size)
- Matching upper-bounds (code constructions) for both cases.