Survivable Distributed Storage with Progressive Decoding

Yunghsiang S. Han

Department of Electrical Engineering
National Taiwan University of Science and Technology

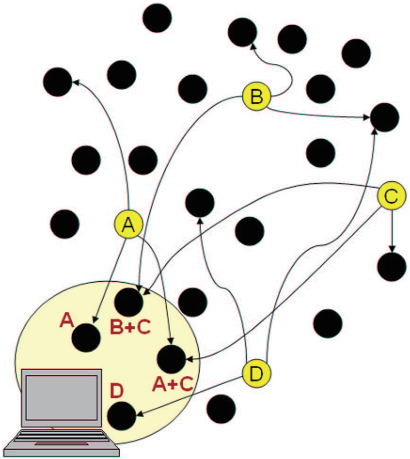
October 6, 2011

Presented at the *2010 IEEE INFOCOM Conference*
Joint work with Soji Omiwade and Rong Zheng at University of Houston

Outline

Distributed Networked Storage

- k data-generating nodes to generate data
- Distributed to n storage nodes
- Access any k storage nodes to recover data



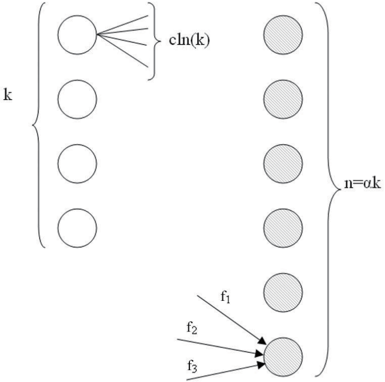
Common Requirements

- Resist crash-stop failures
- Decentralized scheme
- No communications between data generating (or storage) nodes
- Low communication cost
- $k < n$

DEC Approach

Decentralized Erasure Codes (DEC)

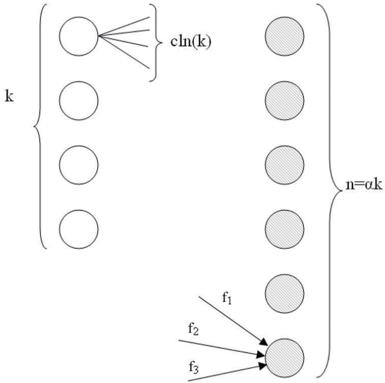
- Encoding is performed at storage nodes
- Encoding matrix is sparse
- Collect data from any k storage nodes can recover all data with high probability
- $c = 5n/k$
- Operate on very large finite field



A. G. Dimakis, V. Prabhakaran, and K. Ramchandran,"Decentralized erasure codes for distributed networked storage," *IEEE Trans. Inform. Theory*, June 2006. Extended version appeared in *IEEE/ACM Trans. Networking*, pp. 2809 - 2816, June 2006.

Encoding of DEC

- $$\begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1(n-1)} & f_{1n} \\ f_{21} & f_{22} & \cdots & f_{2(n-1)} & f_{2n} \\ & & \vdots & & \\ f_{k1} & f_{k2} & \cdots & f_{k(n-1)} & f_{kn} \end{bmatrix}$$
- Only $c \ln k$ nonzero elements in each column
- Total bits sent out in each data-generating node: $T_0 c \ln k$



DFC Approach

Decentralized Fountain Codes (DFC)

- Low encoding and decoding complexity
- When k is large, collect data from any k storage nodes can recover all data with high probability
- Large communication cost

Y. Lin, B. Liang, and B. Li, "Data persistence in large-scale sensor networks with decentralized fountain codes," in *Proceedings of the 26th IEEE INFOCOM*, 2007, pp. 612.

Drawback of Existing Approaches

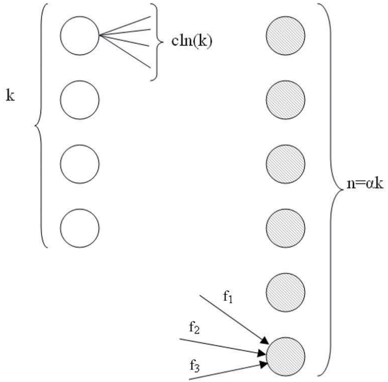
- Can only recover all data probabilistically
- Either operating on large finite fields or having high communication cost
- Cannot handle Byzantine failures

Byzantine Failures

- Return wrong data by a storage node
  - Software bugs or virus
  - Malicious attacks
  - Communication error

Why not Reed-Solomon Codes

- High communication cost: in average $(n-k+1)k/n$ nonzero elements in each column
- Centralized encoding
- Can Reed-Solomon (RS) codes be used in conventional distributed storage? Yes.
- Can it be used in distributed networked storage? Has been claimed NO!



New Approach Based on RS Codes

- Data generated by each data-generating node are usually long
- Perform encoding at each data-generating node
- The copies of data sent out by each data-generating node: n/k

Why Progressive Decoding

- The number of Byzantine nodes are usually small
- Accessing all n storage nodes to perform decoding consume too much bandwidth
- Access just enough storage nodes to perform decoding– error-erasure decoding
- Add error detection (CRC) for progressive decoding
- The undetected error for CRC with r redundancy bits is $1/2^r$

Encoding and dissemination of k data symbols

- Data symbols: $\boldsymbol{u} = \{u_0, u_1, \ldots, u_{k-1}\}$
- Storage symbols: $\boldsymbol{c} = \{c_0, c_1, \ldots, c_{n-1}\}$
- Data node generates $\boldsymbol{u}$; storage node i stores c_i
- Encoding data symbols: $\boldsymbol{c} = \boldsymbol{u}\boldsymbol{G}$, where $\boldsymbol{G}$ is

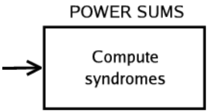
$$
\begin{bmatrix}
1 & 1 & 1 & \cdots & 1 \\
\alpha & \alpha^2 & \alpha^3 & \cdots & \alpha^n \\
\alpha^2 & (\alpha^2)^2 & (\alpha^3)^2 & \cdots & (\alpha^n)^2 \\
 & & & \vdots & \\
\alpha^{k-1} & (\alpha^2)^{k-1} & (\alpha^3)^{k-1} & \cdots & (\alpha^n)^{k-1}
\end{bmatrix}
$$

- Any k of n live non-byzantine nodes yield $\boldsymbol{u}$

RS Decoding to Reconstruct $\boldsymbol{u}$

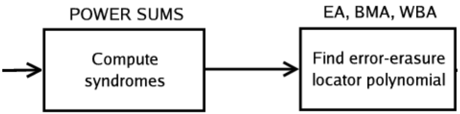
RS Decoding to Reconstruct $\boldsymbol{u}$

- Syndromes: needed for finding error-locator polynomial



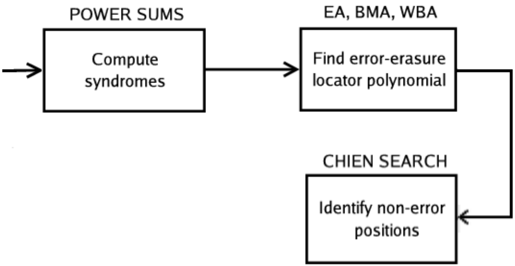
RS Decoding to Reconstruct *u*

- Syndromes: needed for finding error-locator polynomial



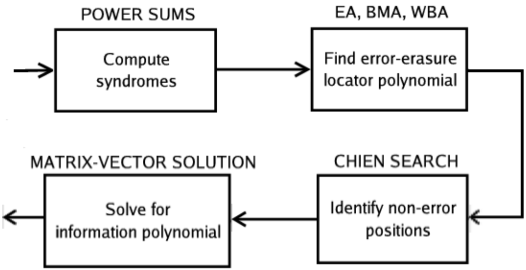
RS Decoding to Reconstruct $\boldsymbol{u}$

- Syndromes: needed for finding error-locator polynomial



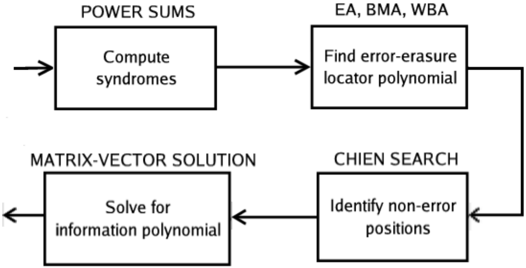
RS Decoding to Reconstruct *u*

- Syndromes: needed for finding error-locator polynomial



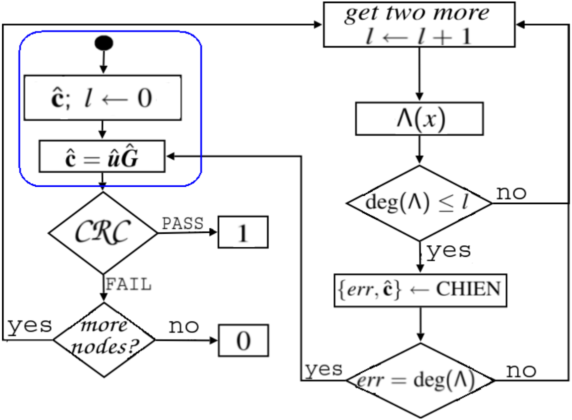
RS Decoding to Reconstruct *u*

- Syndromes: needed for finding error-locator polynomial
- When no errors, only last step performed



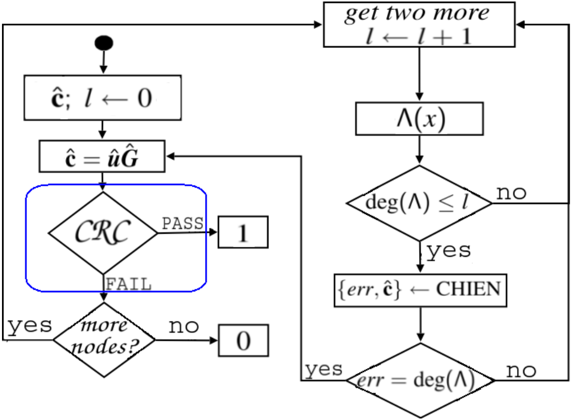
Summary of the Progressive Data Retrieval

Get k storage symbols
Solve for $\hat{\boldsymbol{u}}$



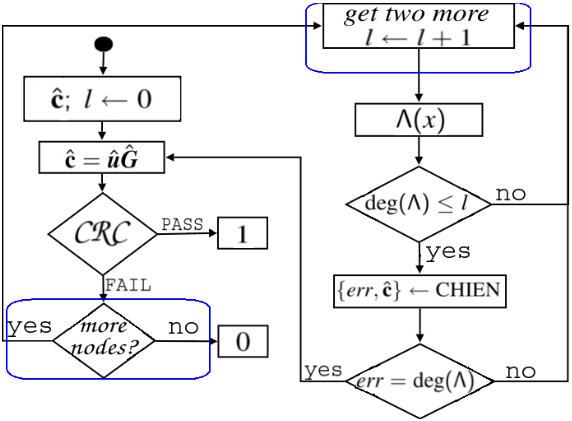
Summary of the Progressive Data Retrieval

Reconstuction successful, if CRC passes
Collect more symbols...if possible



Summary of the Progressive Data Retrieval

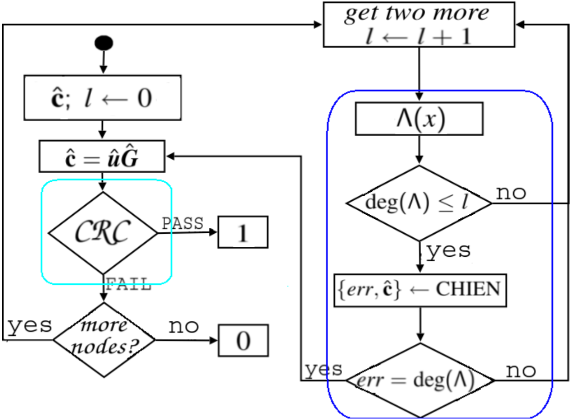
Reconstruction fails if no more symbols
Otherwise, CRC failure implies errors exist



Summary of the Progressive Data Retrieval

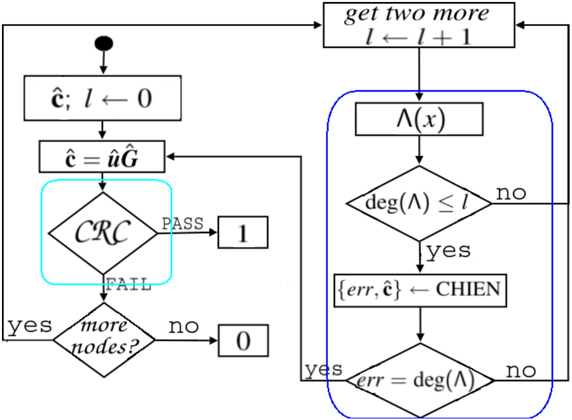
Solve for the error-erasure locator polynomial, $\Lambda(x)$

Relying mostly on Chien search, *incrementally* update $\Lambda(x)$



Summary of the Progressive Data Retrieval

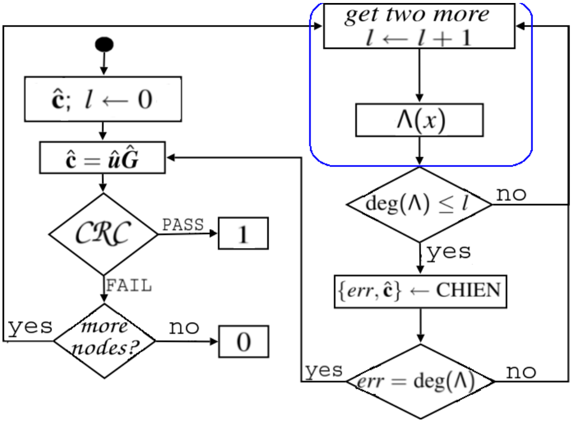
Obtain more symbols as needed



Summary of the Progressive Data Retrieval

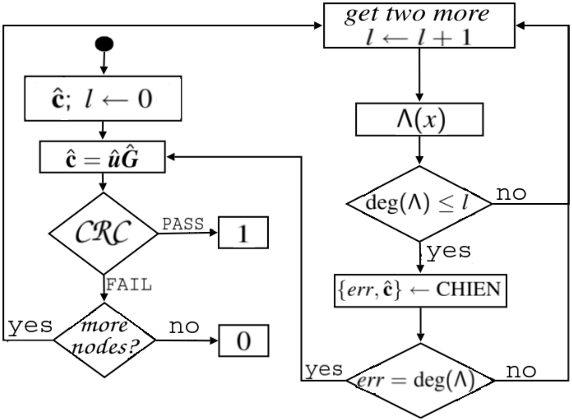
BMA design does not allow for incremental update

Proposed scheme only updates Λ



Summary of the Progressive Data Retrieval

Result: Minimal communication costs, with no wasted effort



Performance Comparison

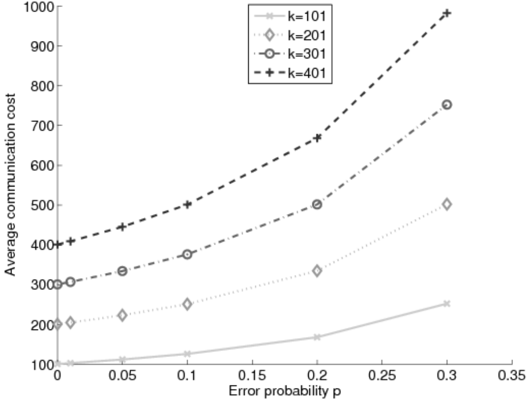
- Proposed RS coding matrix allows for fast decoding
- The error-erasure locator algorithm more efficient than RAID-6's
- Decentralized erasure and fountain infeasible in byzantine environments

	DEC	DFC	RAID-6	IncrRSDecode
Dissemination	$nT_0 \log k$	$nT_0 \log \frac{k}{\delta}$	–	nT
Erasure decoding	k^3	$k \log \frac{k}{\delta}$	k^3	k^2
Error detection	no	no	yes	yes
Error correction	no	no	yes	yes
Incremental update	–	–	no	yes

Average Number of Access

$$
\begin{aligned}
&\bar{N}(n,k)\\
=\ & \sum_{v=0}^{n-k}\binom{n}{v}p^{v}(1-p)^{n-v}\sum_{i=0}^{\min(v,\lfloor\frac{n-k}{2}\rfloor,n-v-k)}(k+2i)\frac{\binom{n-v}{i+k-1}\binom{v}{i}}{\binom{n}{2i+k-1}}\times\frac{k}{i+k}\times\frac{n-v-(i+k-1)}{n-(2i+k-1)}\\
+\ & \sum_{v=0}^{n-k}n\binom{n}{v}p^{v}(1-p)^{n-v}\left(1-\sum_{i=0}^{\min(v,\lfloor\frac{n-k}{2}\rfloor,n-v-k)}\frac{\binom{n-v}{i+k-1}\binom{v}{i}}{\binom{n}{2i+k-1}}\times\frac{k}{i+k}\times\frac{n-v-(i+k-1)}{n-(2i+k-1)}\right)\\
+\ & \sum_{v=n-k+1}^{n}n\binom{n}{v}p^{v}(1-p)^{n-v}.
\end{aligned}
$$

Numerical Results of $\bar{N}(1023, k)$

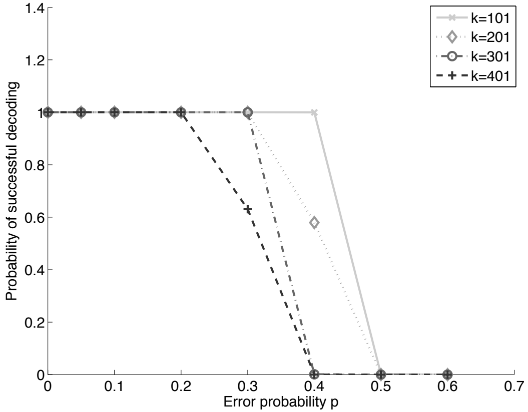


Successful Decoding Rate

$$\Pr_{suc}(n,k)$$

$$= \sum_{v=0}^{n-k} \binom{n}{v} p^{v}(1-p)^{n-v} \sum_{i=0}^{\min(v,\lfloor \frac{n-k}{2} \rfloor, n-v-k)} \frac{\binom{n-v}{i+k-1}\binom{e}{i}}{\binom{n}{2i+k-1}} \times \frac{k}{i+k} \times \frac{n-v-(i+k-1)}{n-(2i+k-1)}$$

Numerical Results of $\Pr_{suc}(1023, k)$



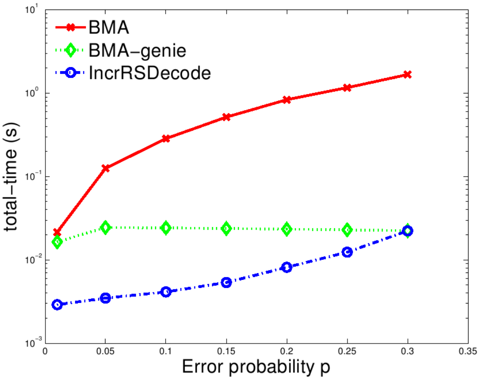
Security

- Compromised nodes can collude to forge data
- The security-strength– the maximum number of compromised nodes that cannot forge the information data even they collude
- The security strength is $\min\{k, \lceil \frac{n-k+2}{2} \rceil\} - 1$
- Using cryptographic hash function to increase the security-strength
- The 32-bit CRC code can be replaced with a 128-bit MD5 code

Implementation Details

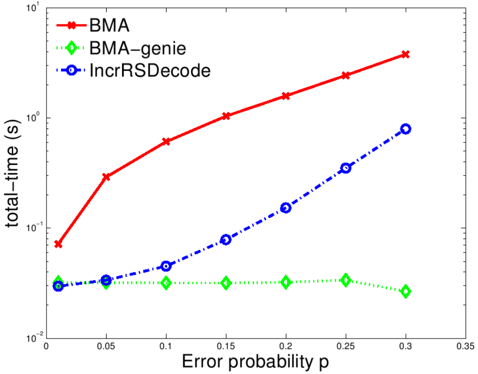
- C on 2.66GHz Intel Xeon CPU; 4MB cache; 2GB RAM
- Memory buffers simulate storage devices
- Three algorithms are considered:
  - BMA: Progressively retrieves data symbols until CRC passes
  - BMA-Genie: Knows *a priori* how many symbols needed
  - IncrRSDecode: Proposed incremental decoding algorithm
- Both BMA and IncrRSDecode minimize communication
- We analyze the minimization effect on the decoding computation
  - As error probability increases
  - Breakdown of the computation time

Average Computation Time for Decoding One Group



$$k = 101, \ n = 1023$$

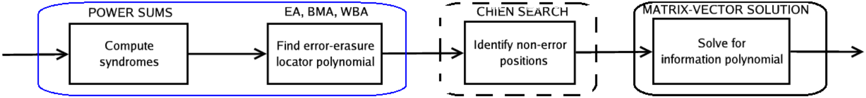
Average Computation Time for Decoding One Group



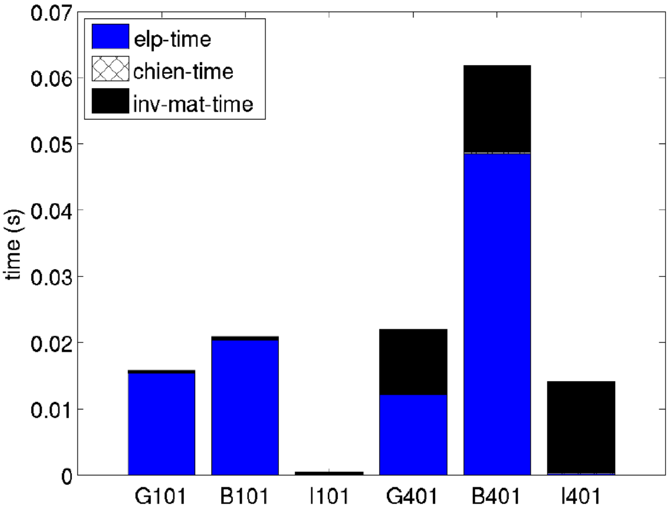
$$k = 401, \ n = 1023$$

Average Computational Time Breakdown for Decoding One Codeword

Time divided into elp time, chien-time, inv-mat-time

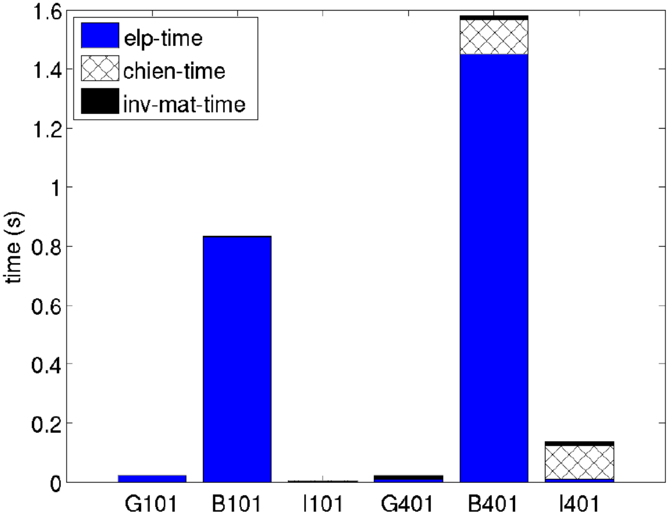


Average Computational Time Breakdown for Decoding One Codeword



$p = 1\%$

Average Computational Time Breakdown for Decoding One Codeword



$p = 20\%$

Summary

- Proposed a storage coding scheme for survivable distributed networked storage
    - Storage-optimal
    - Handles malicious/malfunctioning storage nodes
    - Minimum reconstruction communication costs
    - Efficient in decoding computation
- Scheme desirable for energy critical systems
    - Minimum communication
    - Minimal computation
- Can be extended to regenerating codes for distributed storage

Thanks

Q&A