Anwitaman Datta
Joint work with Frédérique Oggier (SPMS)

School of Computer Engineering
Nanyang Technological University, Singapore

Eurecom, Sophia-Antipolis/Telefonica Research, Barcelona/University Rovira i Virgili, Tarragona
Talk Outline

- Preliminaries
  - Networked distributed storage systems
  - Need of fault-tolerance
- Erasure codes tailor-made for distributed networked storage
  - Related works
    - Pyramid & Hierarchical codes
    - Regenerating codes
  - Self-repairing codes
- Wrap-up
Networked Distributed Storage Systems

- Diverse flavors
  - P2P storage/back-up systems
    - Do we need them under the shadow of the cloud?
  - Data centers

- Very different characteristics
  - Topology, availability model, …

- Common denominator
  - Individual nodes/components eventually fail
  - Distribution of data over multiple nodes
    - Need redundancy for fault tolerance
    - Need mechanisms for replenishing redundancy
Failure (of individual component) is inevitable

- But, failure of the system is not an option!
  - Failure is the pillar of rivals’ success …
- Solution: Redundancy & Distribution
Is the Danger Real? Yes

Online Backup Company Carbonite Loses Customers' Data, Blames And Sues Suppliers (Updated)
Robin Wauters
TechCrunch
Mar 23, 2009

Hotmail Data Loss Reveals Cloud Trust Issues
By Keir Thomas, PCWorld

I lot on the heels of what was possibly the first major cloud data leak a few weeks ago, as the new year got underway Microsoft followed up by appearing to wipe the e-mails of a significant number of Hotmail users.

Gmail outage passes 24 hours for some (updated)
By Seth Weintraub February 28, 2011: 1:01 PM ET

Amazon's Cloud Crash Disaster Permanently Destroyed Many Customers' Data
Henry Blodget | Apr. 28, 2011,

Cloud is NSFW
Data Center Fault-Tolerance

Preliminaries

- Faults are omnipresent
  - Hardware, network, software, human, misconfiguration, ...

- Cascade of failures in interdependent networks
  - Power failure => Network switches stop working
  - Network failure => Control system for power system ineffective

© 2011 A. Datta
NTU Singapore
**Redundancy Based Fault Tolerance**

**Preliminaries**
- **Replicate data**
  - e.g., 3 or more copies
    - The standard practice
  - In nodes on different racks
    - Can deal with switch failures
- **Power back-up**
  - Using battery between racks (e.g., Google)

© 2011 A. Datta
NTU Singapore
Redundancy Based Fault Tolerance

Preliminaries

- Using “independent” physical infrastructure
  - Over different availability zones (Amazon AZ)
    - How independent are components in a complex network?
  - Over multiple geographical regions
Amazon’s AWS: Availability Zones

Note: The recent (April 2011) AWS outage was the first region-wide failure.
Five Levels of Redundancy

- Physical
- Virtual resource
- Availability zone
- Region
- Cloud
But At What Cost?

- Failure is not an option, but …
  - … are the overheads acceptable?
Reducing the Overheads of Redundancy

- Erasure codes
  - Much lower storage overhead
  - High level of fault-tolerance
    - In contrast to replication or RAID based systems

- Has the potential to significantly improve the “bottomline”
  - Can it however match the performance needs?
    - An open question
Reducing the Overheads of Redundancy

- Many academic studies

Erasure Coding vs. Replication: A Quantitative Comparison

Hakim Weatherspoon and John D. Kubiatowicz
Computer Science Division
University of California, Berkeley

Does erasure coding have a role to play in my data center?

Zhe Zhang
Oak Ridge National Laboratory

Amey Deshpande, Xiaosong Ma
North Carolina State University

Eno Thereska, Dushyanth Narayanan
Microsoft Research Cambridge

- Some real deployments (for backup/archives)
Erasure Codes (ECs)

- Originally designed for communication
  - EC\((n,k)\)

Data = message

Encoding

Decoding

Lost blocks

Receive any \(k' \geq k\) blocks

\(k\) blocks

\(n\) encoded blocks

Original \(k\) blocks
Erasure Codes for Networked Storage

Data = Object

$k$ blocks

$n$ encoded blocks (stored in storage devices in a network)

Encoding

$k'$ (≥ $k$) blocks

Decoding

Lost blocks

Reconstruct Data

© 2011 A. Datta
NTU Singapore
Static Resilience

Preliminaries

Tailor-made erasure codes
- Traditional ECs
- Pyramid codes
- Regenerating Codes
- Self-repairing Codes

Wrap up

Replicated \( r \) times
- Faults that can be tolerated: \( r-1 \)
- Probability of failure: \( f^r \)
- Storage efficiency: \( 1/r \)
- Access: Find any one good replica

Erasure coded \((k \text{ of } n)\)
- Faults that can be tolerated: \( n-k \)
- Probability of failure:

\[
\sum_{j=1}^{k} \binom{n}{n-k+j} f^{n-k+j} (1-f)^{k-j}
\]
- Storage efficiency: \( k/n \)
- Access: Find \( k \) good blocks

Assumption: Peer failure is i.i.d. with failure probability \( f \)
Replenishing Lost Redundancy for ECs

- Repairs needed for long-term resilience

Retrieve any \( k' \geq k \) blocks

Decoding

Original \( k \) blocks

Encoding

Recreate lost blocks

Re-insert

Reinsert in (new) storage devices, so that there is (again) \( n \) encoded blocks

\[ n \] encoded blocks

- Repairs are expensive & slow!
What is the best one can do (w.r.to repairs)?

- Minimize bandwidth usage per repair
- Minimize number of live nodes used per repair

Erasure codes have some other drawbacks

- Coding/Decoding is “Expensive”
  - In contrast to replication or RAID/XOR based systems
  - Systematic codes can help (with decoding/access)!
    - Not adequate when load-balancing is also an issue!!
- More complex system design
- We do not attempt to address these explicitly
  - But, some solution we will arrive at will be amenable!
Can We Do Better?

**Preliminaries**

- Erasure codes tailor-made for distributed networked storage

- **Tailor-made erasure codes**
  - Traditional ECs
  - Pyramid codes
  - Regenerating Codes
  - Self-repairing Codes

**Wrap up**

- Pyramid codes
- Regenerating codes
- Self-repairing codes

© 2011 A. Datta
NTU Singapore
Pyramid & Hierarchical Codes

Preliminaries

Tailor-made erasure codes
- Traditional ECs
- Pyramid codes
- Regenerating Codes
- Self-repairing Codes

Wrap up
Pyramid & Hierarchical Codes

Pyramid Codes: Flexible Schemes to Trade Space for Access Efficiency in Reliable Data Storage Systems

Cheng Huang, Minghua Chen, and Jin Li
Microsoft Research, Redmond, WA 98052

Hierarchical Codes: How to Make Erasure Codes Attractive for Peer-to-Peer Storage Systems

Alessandro Duminuco and Ernst Biersack
EURECOM
Sophia Antipolis, France
Essentially, “nested” use of erasure codes

Pyramid Codes

Preliminaries
Tailor-made erasure codes
- Traditional ECs
- Pyramid codes
- Regenerating Codes
- Self-repairing Codes

Wrap up

Multi-hierarchical extension
Pyramid & Hierarchical Codes

**Pros**
- If `small` number of faults
  - Communication restricted within the `hierarchy` suffices
  - Progressively go higher-up for larger number of faults
  - Isolated faults can be repaired independently
- Naturally maps to hierarchical data-center design?

**Cons**
- Asymmetry
  - Different encoded blocks have different importance
    - Repair traffic is arbitrary: depending on which blocks are lost
  - Difficult to analyze
  - Complex algorithm (for decoding/repair) and system design
Regenerating Codes

Preliminaries

Tailor-made erasure codes
- Traditional ECs
- Pyramid codes
- Regenerating Codes
- Self-repairing Codes

Wrap up

Network Coding for Distributed Storage Systems
Alexandros G. Dimakis, P. Brighten Godfrey, Yunnan Wu,
Martin Wainwright and Kannan Ramchandran

Explicit Construction of Optimal Exact Regenerating Codes for Distributed Storage

K. V. Rashmi†, Nihar B. Shah†, P. Vijay Kumar‡, Kannan Ramchandran#

† Dept. of ECE, Indian Institute Of Science, Bangalore, India.
Email: {rashmikv, nihar, vijay}@eee.iisc.ernet.in
‡ Dept. of EECS, University of California, Berkeley, USA.
Network information flow based arguments to determine “optimal” trade-off of storage/repair-bandwidth
Regenerating Codes

- Example code (w/ Functional Repair)
  - Based on random linear network coding
    - Exact repairing codes have also been proposed

Preliminaries

- Tailor-made erasure codes
  - Traditional ECs
  - Pyramid codes
  - Regenerating Codes
  - Self-repairing Codes

Wrap up
Regenerating Codes

**Pros**

- Network information flow analysis determines optimal (w.r.to repair bandwidth)
  - Catch: Subject to MDS property of code (more on this, later)
- Some proposed codes apply network coding on top of ECs
  - Inherit the properties for EC for de/coding

**Cons**

- Codes for only specific points on trade-off curve
  - Information flow analysis itself does not suggest any code
- Restrictive
  - Some proposed codes can carry out repair only for one fault
  - Needs to contact all live nodes for repair to be optimal
- Not simple: Algorithmic as well as system design
Self-repairing Codes

Preliminaries

- Tailor-made erasure codes
- Traditional ECs
- Pyramid codes
- Regenerating Codes
- Self-repairing Codes

Wrap up

Two families of SRCs have been proposed

Self-repairing Homomorphic Codes for Distributed Storage Systems

Frédérique Oggier
Division of Mathematical Sciences
School of Physical and Mathematical Sciences
Nanyang Technological University

Anwitaman Datta
Division of Computer Science
School of Computer Engineering
Nanyang Technological University

Self-Repairing Codes for Distributed Storage — A Projective Geometric Construction

Frédérique Oggier
Division of Mathematical Sciences
School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore

Anwitaman Datta
Division of Computer Science
School of Computer Engineering
Nanyang Technological University, Singapore

© 2011 A. Datta
NTU Singapore

Infocom 2011

On arXiv
Self-repairing Codes (SRC)

- Self-repairing codes are \((n, k)\) codes s.t.
  - encoded fragments can be repaired directly from other subsets of encoded fragments.
  - a fragment can be repaired from a fixed number of encoded fragments, independently of which specific blocks are missing
    - Analogous to erasure codes supporting reconstruction using any \(n - k\) losses, independently of which

- Note
  - \(k\) encoded blocks are enough to recreate the object
    - Caveat: not any arbitrary \(k\) (i.e., SRCs are not MDS)
    - However, there are many such \(k\) combinations
Self-repairing Codes: Blackbox View

Retrieve some \( d(< k) \) blocks (e.g. \( d=2 \)) to recreate a lost block

Lost blocks

Reinsert in (new) storage devices, so that there is (again) \( n \) encoded blocks

\( n \) encoded blocks

(Stored in storage devices in a network)
Self-repairing Codes

- There is at least one pair to repair each node, for up to \((n - 1)/2\) simultaneous failures
  - Parallel & fast repair under multiple failures
  - Much less interference/dependencies among repairs
A weakly linearized polynomial $p(X)$ over $\mathbb{F}_q$, $q = 2^m$, has the form

$$p(X) = \sum_{i=0}^{k-1} p_i X^{2^i}, \ p_i \in \mathbb{F}_q.$$  

Let $a, b \in \mathbb{F}_{2^m}$ and let $p(X)$ be a weakly linearized polynomial given by $p(X) = \sum_{i=0}^{k-1} p_i X^{2^i}$. We have

$$p(a + b) = p(a) + p(b).$$
Take an object \( o \) of length \( M \):

\[
o = (o_1, \ldots, o_k), \quad o_i \in \mathbb{F}_{2^M/k}.
\]

Take a linearized polynomial with coefficients in \( \mathbb{F}_{2^M/k} \)

\[
p(X) = \sum_{i=0}^{k-1} p_i X^{2^i},
\]

and encode the \( k \) fragments \( p_i = o_{i+1}, \ i = 0, \ldots, k - 1. \)
**HSRC Encoding**

**Preliminaries**
- Tailor-made erasure codes
  - Traditional ECs
  - Pyramid codes
  - Regenerating Codes
  - Self-repairing Codes

**Wrap up**

---

**Linearized polynomial**

\[ p(X) = \sum_{i=0}^{k-1} p_i X^{2^i} \]

with \( p_i = o_{i+1} \)

**k blocks**
Each of size \( M/k \)

**n encoded blocks**

Evaluate \( p(X) \) in \( n \) non-zero values \( \alpha_1, \ldots, \alpha_n \) of \( \mathbb{F}_{2^M/k} \) to get a \( n \)-dimensional codeword

\[ (p(\alpha_1), \ldots, p(\alpha_n)) \]

and each \( p(\alpha_i) \) is given to node \( i \) for storage.
Repair: Express $\alpha_i$ in a $\mathbb{F}_2$-basis $B = \{b_1, \ldots, b_{M/k}\}$ of $\mathbb{F}_{2^{M/k}}$, then

$$\alpha_i = \sum_{j=1}^{M/k} \alpha_{ij} b_j, \quad \alpha_{ij} \in \mathbb{F}_2 \Rightarrow p(\alpha_i) = \sum_{j=1}^{M/k} \alpha_{ij} p(b_j).$$
A data file $\mathbf{o} = (o_1, \ldots, o_{12})$ of $M = 12$ bits is cut into $k = 3$ fragments ($M/k = 4$)

$$
\mathbf{o}_1 = (o_1, \ldots, o_4), \quad \mathbf{o}_2 = (o_5, \ldots, o_8), \quad \mathbf{o}_3 = (o_9, \ldots, o_{12}) \in \mathbb{F}_2^4.
$$

$\mathbb{F}_2^* = \langle w \rangle$, with $w^4 = w + 1$. Encode with the polynomial

$$
p(X) = \sum_{i=1}^{4} o_i w^i X + \sum_{i=1}^{4} o_{i+4} w^i X^2 + \sum_{i=1}^{4} o_{i+8} w^i X^4.
$$

For $n = 7$, evaluate $p(X)$ at say $1, w, w^2, w^4, w^5, w^8, w^{10}$. We get:

$$(p(1), p(w), p(w^2), p(w^4), p(w^5), p(w^8), p(w^{10}))$$
### Tailor-made erasure codes

- Traditional ECs
- Pyramid codes
- Regenerating Codes
- Self-repairing Codes

### Wrap up

#### Preliminaries

<table>
<thead>
<tr>
<th>missing fragment(s)</th>
<th>pairs to reconstruct missing fragment(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p(1) )</td>
<td>( (p(w), p(w^4)); (p(w^2), p(w^8)); (p(w^5), p(w^{10})) )</td>
</tr>
<tr>
<td>( p(w) )</td>
<td>( (p(1), p(w^4)); (p(w^2), p(w^5)); (p(w^8), p(w^{10})) )</td>
</tr>
<tr>
<td>( p(w^2) )</td>
<td>( (p(1), p(w^8)); (p(w), p(w^5)); (p(w^4), p(w^{10})) )</td>
</tr>
<tr>
<td>( p(1) ) and ( p(w) )</td>
<td>( (p(w^2), p(w^8)) ) or ( (p(w^5), p(w^{10})) ) for ( p(1) )</td>
</tr>
<tr>
<td></td>
<td>( (p(w^8), p(w^{10})) ) or ( (p(w^2), p(w^5)) ) for ( p(w) )</td>
</tr>
<tr>
<td>( p(1) ) and ( p(w) ) and ( p(w^2) )</td>
<td>( (p(w^5), p(w^{10})) ) for ( p(1) )</td>
</tr>
<tr>
<td></td>
<td>( (p(w^8), p(w^{10})) ) for ( p(w) )</td>
</tr>
<tr>
<td></td>
<td>( (p(w^4), p(w^{10})) ) for ( p(w^2) )</td>
</tr>
</tbody>
</table>
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 block

<table>
<thead>
<tr>
<th>fragment</th>
<th>suitable pairs to reconstruct</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p(1)$</td>
<td>$(p(w^7), p(w^9)); (p(w^{11}), p(w^{12}))$</td>
</tr>
<tr>
<td>$p(w)$</td>
<td>$(p(w^7), p(w^{14}); (p(w^8), p(w^{10}))$</td>
</tr>
<tr>
<td>$p(w^2)$</td>
<td>$(p(w^7), p(w^{12}); (p(w^9), p(w^{11}); (p(w^{12}), p(w^{10}))$</td>
</tr>
<tr>
<td>$p(w^3)$</td>
<td>$(p(w^8), p(w^{13}); (p(w^{10}), p(w^{12}))$</td>
</tr>
<tr>
<td>$p(w^4)$</td>
<td>$(p(w^9), p(w^{14}); (p(w^{11}), p(w^{13}))$</td>
</tr>
<tr>
<td>$p(w^5)$</td>
<td>$(p(w^7), p(w^{13}); (p(w^{12}), p(w^{14}))$</td>
</tr>
<tr>
<td>$p(w^6)$</td>
<td>$(p(w^7), p(w^{10}); (p(w^8), p(w^{14}))$</td>
</tr>
</tbody>
</table>
Fast & Parallel Repair: w/ HSRC(15,4)

- Tailor-made erasure codes
  - Traditional ECs
  - Pyramid codes
  - Regenerating Codes
  - Self-repairing Codes

One possible parallelized repair schedule

<table>
<thead>
<tr>
<th>node</th>
<th>(p(w^0))</th>
<th>(p(w^1))</th>
<th>(p(w^2))</th>
<th>(p(w^3))</th>
<th>(p(w^4))</th>
<th>(p(w^5))</th>
<th>(p(w^6))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time 1</td>
<td>(p(w^7))</td>
<td>(p(w^8))</td>
<td>(p(w^9))</td>
<td>(p(w^{13}))</td>
<td>(p(w^{11}))</td>
<td>(p(w^{12}))</td>
<td>(p(w^{10}))</td>
</tr>
<tr>
<td>Time 2</td>
<td>(p(w^9))</td>
<td>(p(w^{10}))</td>
<td>(p(w^{11}))</td>
<td>(p(w^8))</td>
<td>(p(w^{13}))</td>
<td>(p(w^{14}))</td>
<td>(p(w^7))</td>
</tr>
</tbody>
</table>

© 2011 A. Datta
NTU Singapore
**Example**

- **PSRC**(5,3)

<table>
<thead>
<tr>
<th>node</th>
<th>basis vectors</th>
<th>data stored</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_1$</td>
<td>$v_1 = (1000)$, $v_2 = (0110)$</td>
<td>${o_1, o_2 + o_3}$</td>
</tr>
<tr>
<td>$N_2$</td>
<td>$v_3 = (0100)$, $v_4 = (0011)$</td>
<td>${o_2, o_3 + o_4}$</td>
</tr>
<tr>
<td>$N_3$</td>
<td>$v_5 = (0010)$, $v_6 = (1101)$</td>
<td>${o_3, o_1 + o_2 + o_4}$</td>
</tr>
<tr>
<td>$N_4$</td>
<td>$v_7 = (0001)$, $v_8 = (1010)$</td>
<td>${o_4, o_1 + o_3}$</td>
</tr>
<tr>
<td>$N_5$</td>
<td>$v_9 = (1100)$, $v_{10} = (0101)$</td>
<td>${o_1 + o_2, o_2 + o_4}$</td>
</tr>
</tbody>
</table>
# Self-repair Toy Example: w/ PSRC(5,3)

## Preliminaries

### Tailor-made erasure codes
- Traditional ECs
- Pyramid codes
- Regenerating Codes
- Self-repairing Codes

## Data Storage

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

### Repair using two nodes

- Say $N_1$ and $N_3$
  - $(o_1+o_2+o_4) + (o_1) => o_2+o_4$
  - $(o_3) + (o_2+o_3) => o_2$
  - $(o_1) + (o_2) => o_1+o_2$

Four pieces needed to regenerate two pieces

### Repair using three nodes

- Say $N_2$, $N_3$ and $N_4$
  - $(o_2) + (o_4) => o_2+o_4$
  - $(o_1+o_2+o_4) + (o_4) => o_1+o_2$

Three pieces needed to regenerate two pieces
Object Reconstruction: w/ PSRC(5,3)

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

Reconstruction, say using $N_3$, $N_4$ and $N_5$:

- $o_3 + (o_1 + o_3) \Rightarrow o_1$
- $(o_1) + (o_4) + (o_1 + o_2 + o_4) \Rightarrow o_2$
Symmetry in SRCs

- All encoded blocks have symmetric role
  - Equivalent importance of all blocks
    - for both data reconstruction & repair
- Symmetry is good
  - Easy to analyze, understand and implement
  - Simpler algorithm and system design

© 2011 A. Datta
NTU Singapore
Maximum Distance Separable (MDS)?

- SRC is not MDS (and can not be!)
  - Does it matter?
    - Not much
    - In practice, access will be “planned”...
  - PSRC needs less bandwidth than ‘optimal’ RGC!

- Preliminaries
  - Tailor-made erasure codes
    - Traditional ECs
    - Pyramid codes
    - Regenerating Codes
    - Self-repairing Codes

- Wrap up

© 2011 A. Datta
NTU Singapore
Practical Properties

- (Current) SRCs are not systematic
  - But, PSRC is like systematic
  - Caveat: Need to contact more nodes (than k)
    - To obtain systematic `pieces`
    - But: Same total bandwidth usage
      - Parallel download for access can even be an `advantage`
    - `mixed` strategies for access, i.e. get some systematic pieces, and some others are possible …
      - Power saving (by switching off nodes) strategies possible

<table>
<thead>
<tr>
<th>node</th>
<th>basis vectors</th>
<th>data stored</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_1$</td>
<td>$v_1 = (1000)$, $v_2 = (0110)$</td>
<td>${o_1, o_2 + o_3}$</td>
</tr>
<tr>
<td>$N_2$</td>
<td>$v_3 = (0100)$, $v_4 = (0011)$</td>
<td>${o_2, o_3 + o_4}$</td>
</tr>
<tr>
<td>$N_3$</td>
<td>$v_5 = (0010)$, $v_6 = (1101)$</td>
<td>${o_3, o_1 + o_2 + o_4}$</td>
</tr>
<tr>
<td>$N_4$</td>
<td>$v_7 = (0001)$, $v_8 = (1010)$</td>
<td>${o_4, o_1 + o_3}$</td>
</tr>
<tr>
<td>$N_5$</td>
<td>$v_9 = (1100)$, $v_{10} = (0101)$</td>
<td>${o_1 + o_2, o_2 + o_4}$</td>
</tr>
</tbody>
</table>
Practical Properties

- Self-repair implies
  - somewhat “locally decodable”
    - If access to only part of the whole object is desired
- Coding/decoding in PSRC are both using
  - XOR operations only
Outlook

○ 2020: Self-repairing codes in a data-center near you?

○ Ongoing:
  ▪ Concepts/Implementation
  ▪ Prototype miniature data-center testbed
  ▪ Template for preassembled component of a modular 4G+ data center?

○ More details
  http://sands.sce.ntu.edu.sg/CodingForNetworkedStorage/

○ Contact: {anwitaman, frederique}@ntu.edu.sg