Adapting STAR Code for Non-Volatile Memory Systems

Rui Chen, Lihao Xu
Wayne State University
Outline

• Motivation
• What is STAR Code?
• Why STAR Code?
• How to adapt STAR Code?
• Conclusion
Motivation

- With NVM emergence, I/O throughput can reach 10s of GB/s instead of GB/s
- Requires higher performance in erasure codes, rather than traditional RS-Code
What is STAR Code?

- One kind of Erasure Codes
Erasure Code

- Structure for a typical \((k, m)\) erasure code
- One storage unit here can be one NVM chip/disk/node
Erasure Code

- Matrix-based erasure codes
  - Reed-Solomon Code

- Array-based erasure codes
  - STAR Code
Reed-Solomon Code

- An example for (4, 2) RS Code
- Multiplications done by discrete logarithm tables --- computationally expensive
### STAR Code

- Uses only XOR operations
- Can tolerate up to 3 erasures
- A example of (5, 3) STAR Code

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>E</th>
<th>D</th>
<th>C</th>
<th>B</th>
<th>X</th>
<th>X</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data columns</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>A</td>
<td>E</td>
<td>D</td>
<td>C</td>
<td>X</td>
<td>X</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>B</td>
<td>A</td>
<td>E</td>
<td>D</td>
<td>X</td>
<td>X</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td>B</td>
<td>A</td>
<td>E</td>
<td>X</td>
<td>X</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>D</td>
<td>C</td>
<td>B</td>
<td>A</td>
<td>X</td>
<td>X</td>
<td>E</td>
<td></td>
</tr>
</tbody>
</table>

**Parity column III**

**imaginary row**
Why STAR Code?

• Performance evaluation through experiments
Experiment Setup

• Lenovo ThinkCentre M900
  - i5-6500 (L1: 256KB, L2: 1MB, L3: 6MB)
  - Ubuntu 16.04.3
  - gcc 5.4.0 (-O3)
Baseline

- `memcpy` & XOR
- `blocksize=1KB/2KB`
- 1000 stripes (codewords)
- `k: 6 – 17`
Testing Libraries

• **RS-Code**
  - Jerasure 2.0 (C)
  - Intel’s ISA-L 2.17 (C & asm)
  - both includes Intel’s SSE

• **STAR Code**
  - own implementation (C)
  - using Intel’s SSE
How to adapt STAR Code?

• Intel’s SSE
• Parameter Configuration
  - p=17
  - adapting blocksize & k
Intel’s SSE

- Jerasure & ISA-L already include SSE
- 8 128-bit general-purpose registers
  - XMM0 to XMM7
  - each consists of 4 32-bit floating-point numbers
Results

- Adapting SSE
- Encoding
- Decoding
Adapting SSE

- blocksize=1KB/2KB
- 1000 stripes
- k: 6-17
- 3 erasures for decoding
Encoding Performance

- 1000 stripes (codewords)
- m=3
- w=8 for RS-Code
- p=17 for STAR
Adapting blocksize

- $k=10$
Adapting $k$
Decoding Performance

- 1000 stripes (codewords)
- $m=1/2/3$ respectively
- $w=8$ for RS-Code
- $p=17$ for STAR
Adapting blocksize with k=10

m=1

m=2

m=3
Adapting k with m=1

blocksize=1KB

blocksize=2KB

blocksize=4KB

ISA-L
STAR
Jerasure
Adapting $k$ with $m=2$
Adapting k with m=3

blocksize=1KB  blocksize=2KB  blocksize=4KB

Speed (Gbps)

k

ISA-L  STAR  Jerasure
Conclusion

• Intel’s SSE
• Jerasure vs. ISA-L
• STAR vs. RS-Code
• Parameter configuration guideline
Questions?
Thank you!