Parallel Reed/Solomon Coding on Multicore Processors

Peter Sobe

Institute of Computer Engineering
University of Luebeck, Germany

sobe@iti.uni-luebeck.de

currently at
Institute of Computer Science
University of Potsdam, Germany

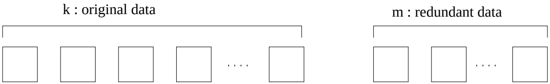
SNAPI Workshop, May 3rd, 2010

Contents

Erasure-tolerant Codes

k data storage resources, e.g. disks
m redundant resources

regular data striping across k resources
encoding: calculation of m independent redundant blocks



a code tolerates f failed storage resources: $f \leq m$

Criteria:

- Number tolerated faults: $f = m$ as the optimum
- Flexibility, when choosing k, m
- Computational cost for en- and decoding

Cauchy Reed/Solomon

Encoding:

- Multiplication of original data word o with a generator matrix G

$$a = \begin{bmatrix} o \\ c \end{bmatrix} = G \cdot o = \begin{bmatrix} I \\ G_{sub} \end{bmatrix} \cdot o$$

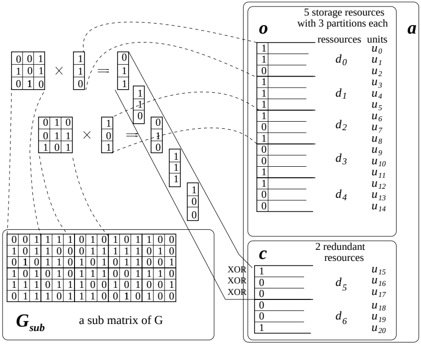
Example Reed/Solomon, 5+2:

- operations $+, \cdot$ within GF(2^3)

$$c = \begin{Bmatrix} 2 & 7 & 4 & 3 & 1 \\ 3 & 4 & 7 & 2 & 5 \end{Bmatrix} \cdot o$$

as a Cauchy-Reed/Solomon code:

- projection to GF(2^1), binary logic
- operations XOR, AND



Cauchy Reed/Solomon

Decoding:

- equations system used for data recalculation

$$o = G'^{-1} * a'$$

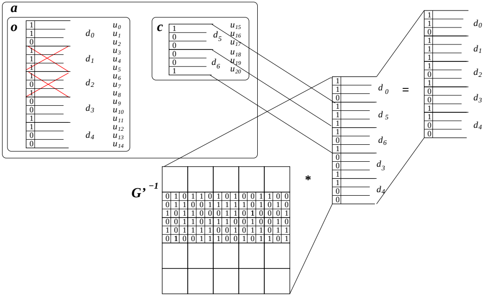
when 2nd and 3rd resource fail:

- operations $+, \cdot$ within $GF(2^3)$

$$o = \left\{ \begin{array}{ccccc} 1 & 0 & 0 & 0 & 0 \\ 4 & 5 & 3 & 2 & 1 \\ 2 & 3 & 5 & 4 & 4 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{array} \right\} \cdot \left\{ \begin{array}{c} d0 \\ d5 \\ d6 \\ d3 \\ d4 \end{array} \right\}$$

by Cauchy-Reed/Solomon:

- projection to $GF(2^1)$, binary logic
- operations XOR, AND



Cauchy Reed/Solomon: Equation-based definition

Equations refer different bits within storage resources

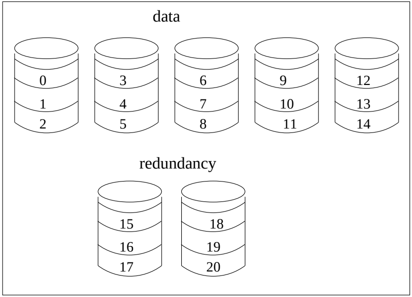
different bits $\Rightarrow$ units on resources $\Rightarrow$ partitions on disks

Unit assignment for k=5, m=2

	data					parities	
resource	$r0$	$r1$	$r2$	$r3$	$r4$	$r5$	$r6$
units	0	3	6	9	12	15	18
	1	4	7	10	13	16	19
	2	5	8	11	14	17	20

number of units per resource (ω)

- $\omega = 3$,
- generally $2^{\omega} > k + m$



Cauchy Reed/Solomon: Equation-based definition

Coding algorithm is an execution of several equations:

- either: instant application on data
- or: store and transform equations, apply them on a sequence of code words

Example of a 5+2 Reed/Solomon code:

direct encoding (45 XOR op.)

```
15 = XOR(2, 3, 4, 5, 7, 9, 11, 12)
16 = XOR(0, 2, 3, 7, 8, 9, 10, 11, 13)
17 = XOR(1, 3, 4, 6, 8, 10, 11, 14)
18 = XOR(0, 2, 4, 6, 7, 8, 11, 12, 13)
19 = XOR(0, 1, 2, 4, 5, 6, 9, 11, 14)
20 = XOR(1, 2, 3, 5, 6, 7, 10, 12)
```

direct decoding (42 XOR op.)

```
6 = XOR(0, 5, 17, 18, 20, 13, 14)
7 = XOR(1, 3, 5, 15, 17, 18, 19, 20, 12, 13)
8 = XOR(2, 4, 16, 19, 20, 12, 13, 14)
9 = XOR(2, 3, 4, 15, 16, 17, 19, 12, 13)
10 = XOR(0, 2, 5, 15, 19, 20, 14)
11 = XOR(1, 3, 15, 16, 18, 20, 12)
```

iterative encoding (33 XOR op.)

```
15 = XOR(B, C, D)
16 = XOR(D, E, F)
17 = XOR(3, 4, 8, E, H)
18 = XOR(2, 4, 6, 7, C, F)
19 = XOR(0, 2, 9, 11, B, H)
20 = XOR(5, 7, 10, 12, A, G)
A = XOR(2, 3)       E = XOR(10, 11)
B = XOR(4, 5)       F = XOR(0, 8, 13)
C = XOR(11, 12)     G = XOR(1, 6)
D = XOR(7, 9, A)    H = XOR(14, G)
```

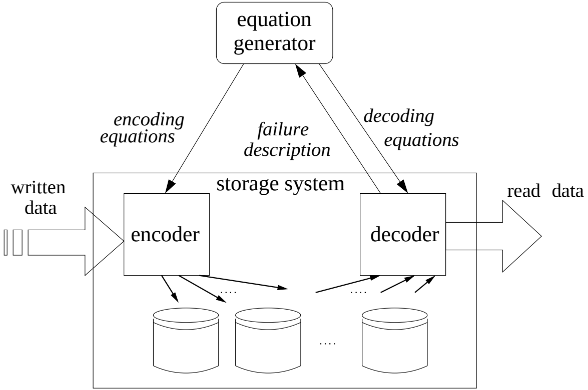
iterative decoding (29 XOR op.)

```
6 = XOR(B, C)
7 = XOR(5, C, D, F)
8 = XOR(19, 14, A, G)
9 = XOR(3, 17, 13, D, G)
10 = XOR(2, 20, B, D)
11 = XOR(15, 16, 18, 20, F)
A = XOR(20, 13)       D = XOR(15, 19)
B = XOR(0, 5, 14)     F = XOR(1, 3, 12)
C = XOR(17, 18, A)    G = XOR(2, 4, 12, 16)
```

Equations for coding

Separation:

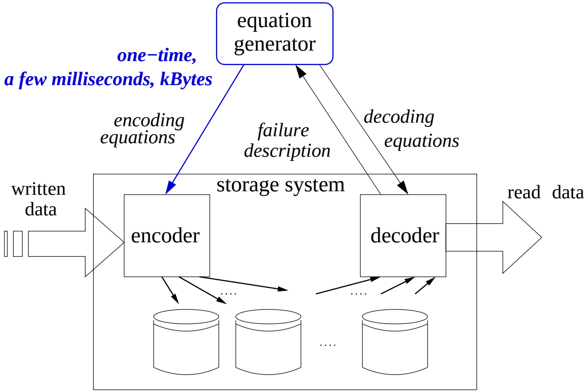
- equation preparation
- equation interpretation for coding



Equations for coding

Separation:

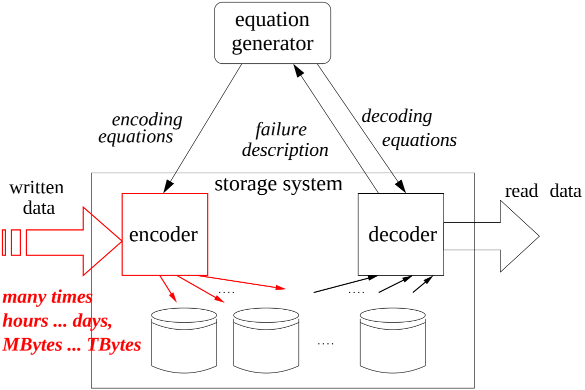
- equation preparation
- equation interpretation for coding



Equations for coding

Separation:

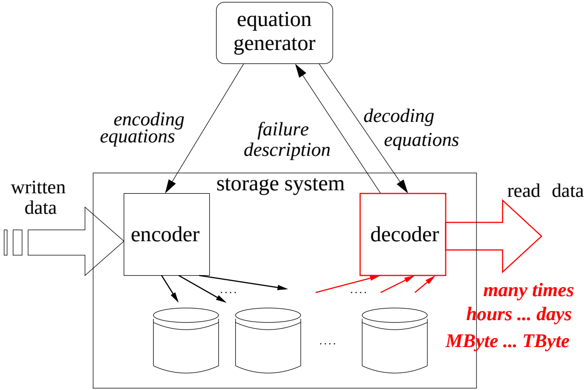
- equation preparation
- equation interpretation for coding



Equations for coding

Separation:

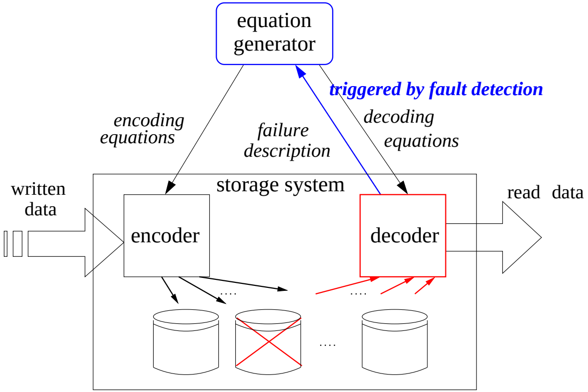
- equation preparation
- equation interpretation for coding



Equations for coding

Separation:

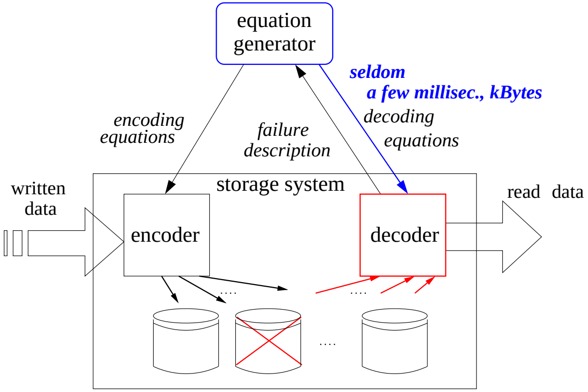
- equation preparation
- equation interpretation for coding



Equations for coding

Separation:

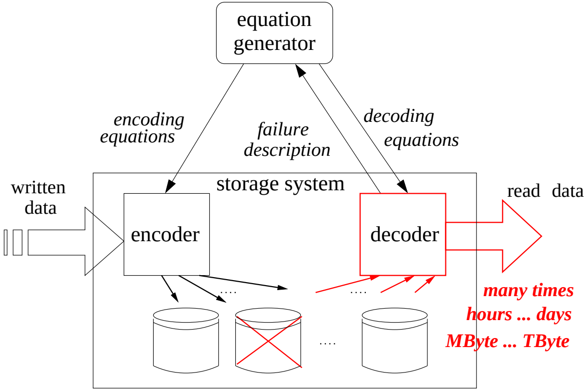
- equation preparation
- equation interpretation for coding



Equations for coding

Separation:

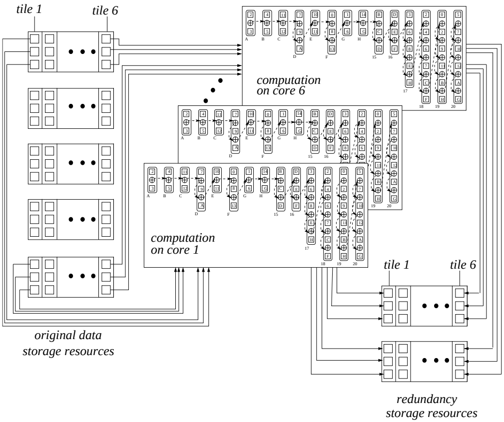
- equation preparation
- equation interpretation for coding



Parallel Coding

Obvious parallelism: block parallel coding

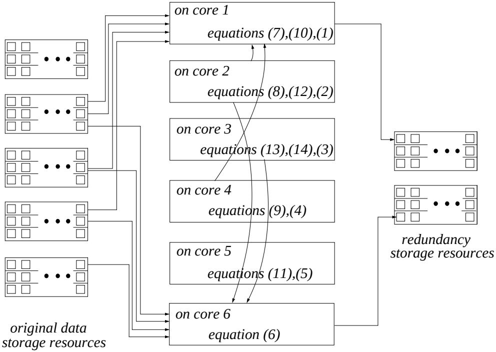
- same coding function on different data blocks
- a core interprets all equations
- a core streams only a part of the input data



Parallel Coding - Equation oriented

Equation-oriented coding

- a core interprets dedicated equations
- a core streams data which is refered by the dedicated equations



Parallel Coding Schedules

encoding and decoding equations extended to schedules

schedule:

- equations assigned to cores
- XOR ops assigned to time steps
- data dependencies resolved

cores	steps 1	2	3	4	5	6
1				B ⊕ C	⊕ D	
		7 ⊕ 9	⊕ A			
	2 ⊕ 3					
2				F ⊕ E	⊕ D	
		0 ⊕ 8	⊕ 13			
	4 ⊕ 5					
3			3 ⊕ 4	⊕ 8	⊕ E	⊕ H
		14 ⊕ G				
	1 ⊕ 6					
4		2 ⊕ 4	⊕ 6	⊕ 7	⊕ C	⊕ F
	10 ⊕ 11					
5		0 ⊕ 2	⊕ 9	⊕ 11	⊕ B	⊕ H
	11 ⊕ 12					
6	7 ⊕ 5	⊕ 10	⊕ 12	⊕ A	⊕ G	

Schedule preparation: stacking of equations

Parallel Coding Schedules

Stacking of equations

Encoding equations,
33 XOR operations

- Terminal equations

```
15 = XOR(B,C,D)
16 = XOR(D,E,F)
17 = XOR(3,4,8,E,H)
18 = XOR(2,4,6,7,C,F)
19 = XOR(0,2,9,11,B,H)
20 = XOR(5,7,10,12,A,G)
```

- Temporary equations

```
A = XOR(2,3)
B = XOR(4,5)
C = XOR(11,12)
D = XOR(7,9,A)
E = XOR(10,11)
F = XOR(0,8,13)
G = XOR(1,6)
H = XOR(14,G)
```

cores	steps												
		1		2		3		4		5		6	
1							B	⊕	C	⊕	D		
			7	⊕	9	⊕	A						
	2	⊕	3										
2							F	⊕	E	⊕	D		
			0	⊕	8	⊕	13						
	4	⊕	5										
3					3	⊕	4	⊕	8	⊕	E	⊕	H
			14	⊕	G								
	1	⊕	6										
4			2	⊕	4	⊕	6	⊕	7	⊕	C	⊕	F
	10	⊕	11										
5			0	⊕	2	⊕	9	⊕	11	⊕	B	⊕	H
	11	⊕	12										
6	7	⊕	5	⊕	10	⊕	12	⊕	A	⊕	G		
temporary units required													
				G		A		B,~~A~~, F,C		D,~~G~~, E		H	
temporary units available													
			A,B, C,G,E		H		D,F						

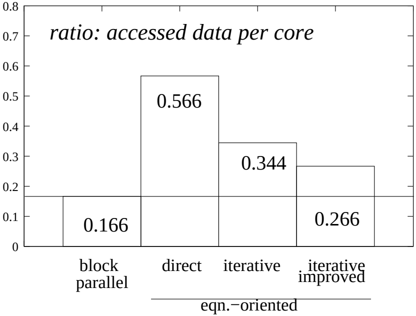
Evaluation

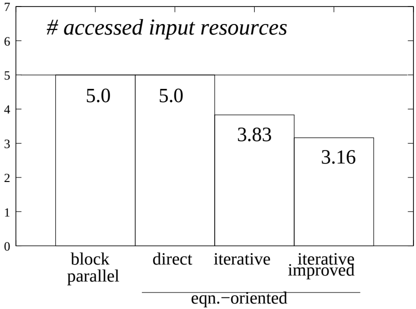
Question: Is equation-oriented parallel coding beneficial?

Criteria:

- accessed data per core
- number of referenced storage resources (input, output)
- number of referenced units per resource and per core
- multiplicity of references
- number of temporary results taken from other cores
- number of time steps of a schedule
  (under absence of access delays, and synchrony of XOR operations)

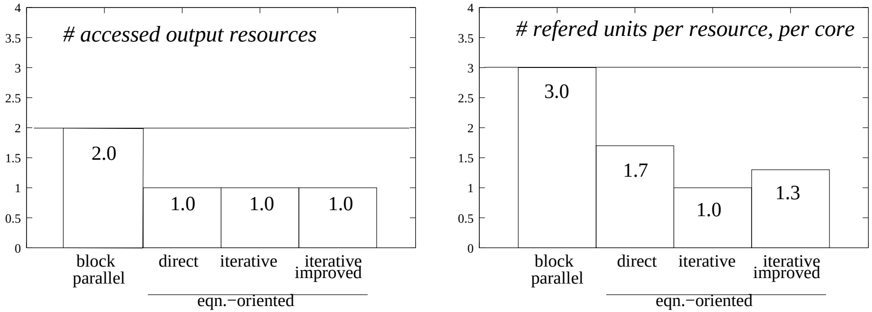
Evaluation





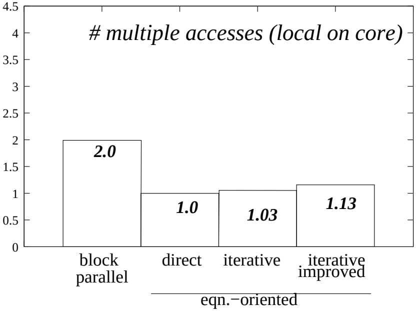
Lower values are better

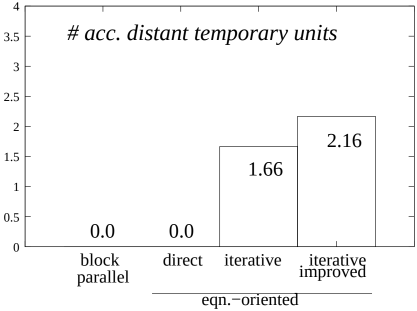
Evaluation



Lower values are better

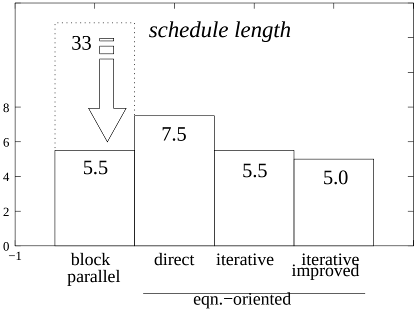
Evaluation





Lower values are better

Evaluation



Equation-oriented parallel coding:

- iterative equations only!
- improve: selecting good Cauchy matrices

Performance benefits:

- minimal schedule length
- multiple accesses reduced
- locality of accesses (resources, units)

Performance obstacles:

- access to temporary results from other cores

Summary

- Cauchy-Reed/Solomon code: XOR based
- Decomposition of coding into several parts,
  described by equations
- Equations: parameterize the encoding and decoding function
- Schedules: pre-calculated placement of equations on cores
- Iterative schedules:
  concentration of data accesses of a core on local regions
- Advantage for software-based coding performance on multicore processors