SD Codes: Erasure Codes Designed for How Storage Systems Really Fail

James S. Plank
University of Tennessee

USENIX FAST
San Jose, CA
February 13, 2013.

Authors

Jim Plank
Tennessee

Liberation Codes
Rotated RS Codes
Jerasure

Mario Blaum
(IBM Almaden)

EVENODD Codes
Blaum-Roth Codes
Generalized EO & RDP
PMDS

Jim Hafner
IBM Almaden

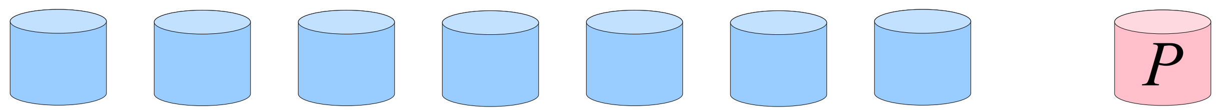
WEAVER Codes
HoVer Codes
REO Engine

Erasure Codes are Everywhere

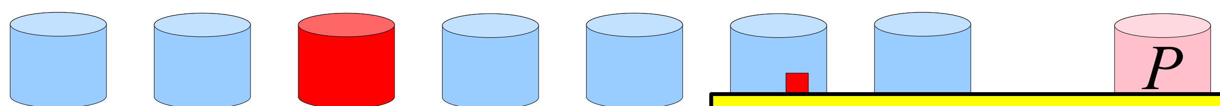
- Commercial systems:
  - From IBM, Microsoft, HP, Netapp, Panasas, EMC, Cleversafe, Amazon, etc...
- Non Commercial Systems:
  - HAIL, Tahoe-LAFS, Pergamum, POTSHARDS, Oceanstore, NC-Cloud, Hydra, etc...
- All employ erasure codes that tolerate more than one disk failure.

The RAID-6 Disconnect

Let's start with a RAID-5 system composed of *n* disks.

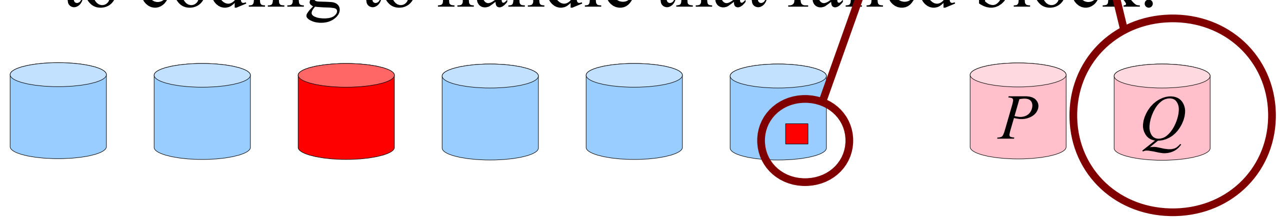


The catastrophic failure mode is a disk failure combined with a latent sector failure.



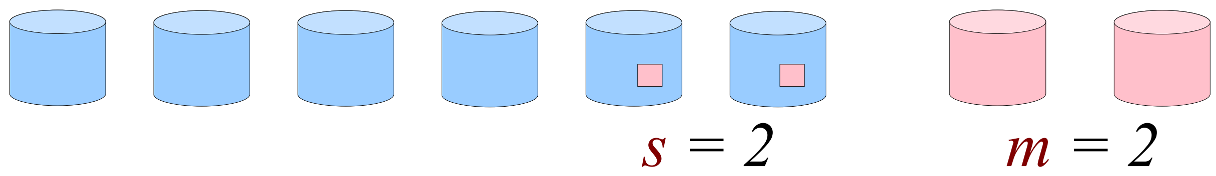
This seems wasteful.

The RAID-6 solution dedicates an entire extra disk to coding to handle that failed block.

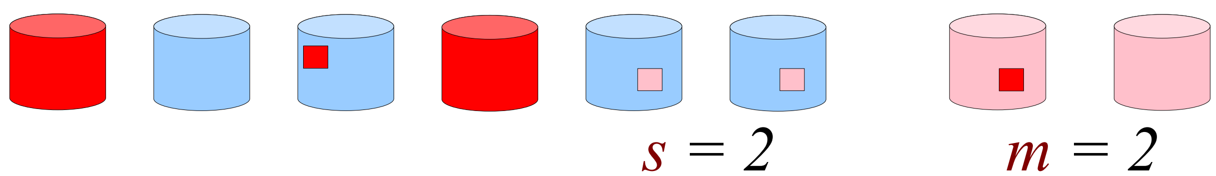


The SD Code Methodology

Dedicate *m* disks and *s* sectors per stripe to coding.

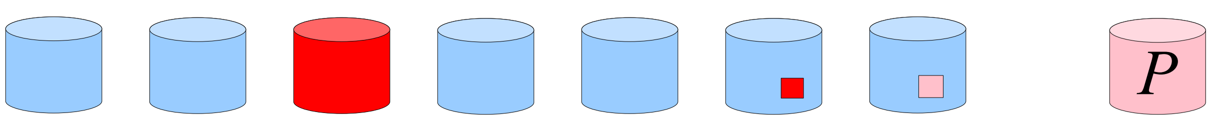


Tolerates the failure of any *m* disks and *s* sectors.



Thus, storage costs match failure modes.

Fixes the RAID-6 disconnect (*m=1, s=1*):

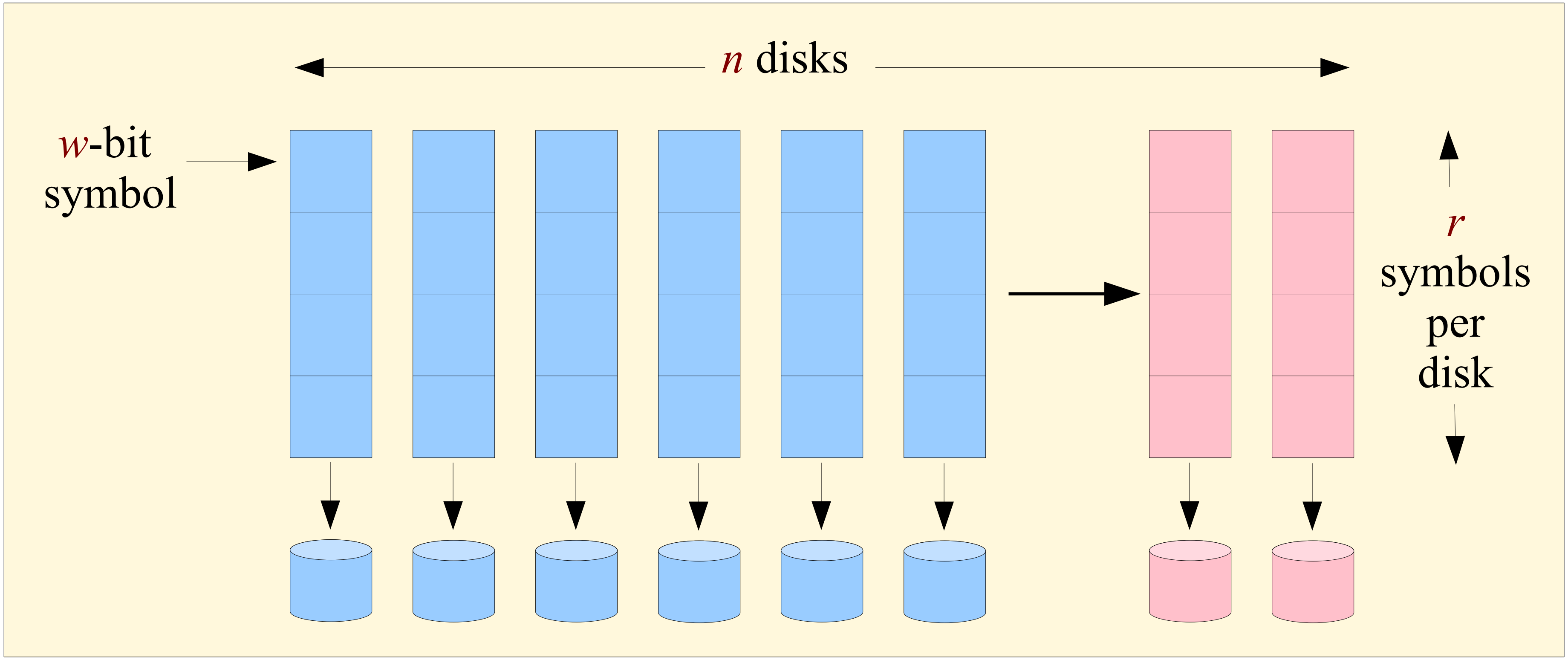


In This Talk

- Detail the SD methodology in FAST language.
  - How it works.
  - Constructions.
  - Performance (theoretical & actual).
  - Open source support.
  - Related work.

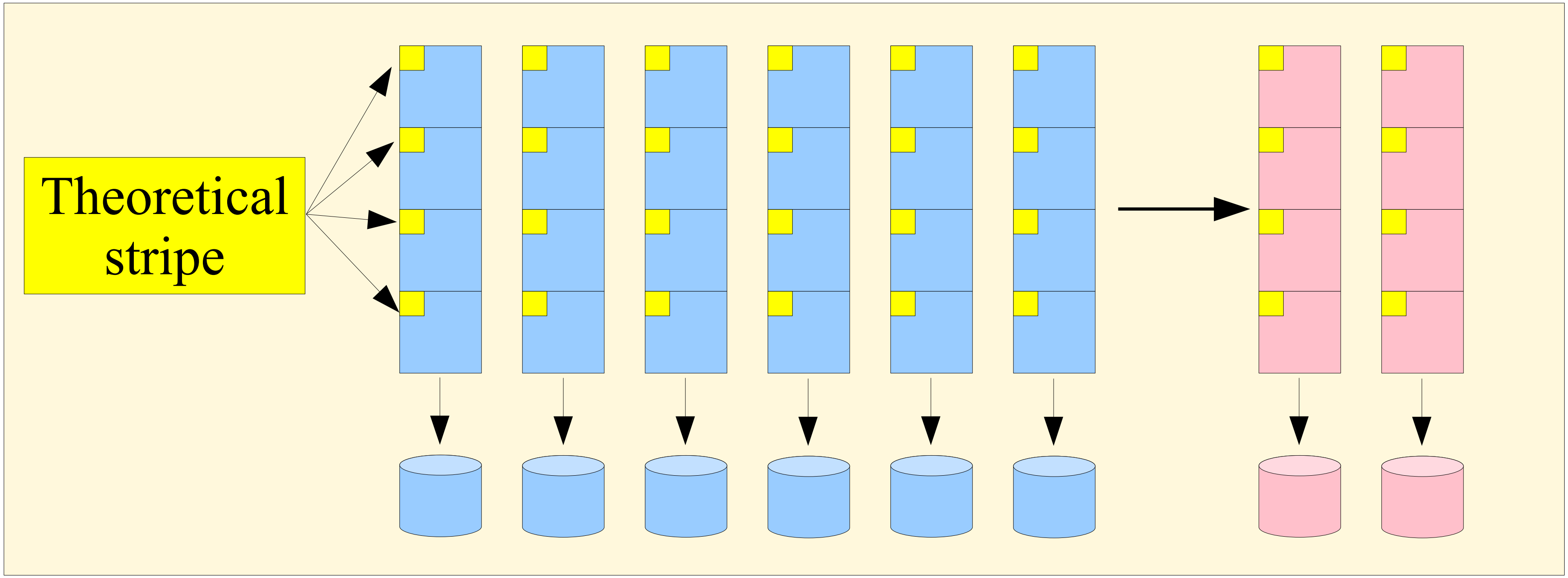
Two Views of a "Stripe"

- The Theoretical View:
  - Disks hold *w*-bit symbols rather than sectors.
  - Precisely: *r* symbols from each of *n* disks:



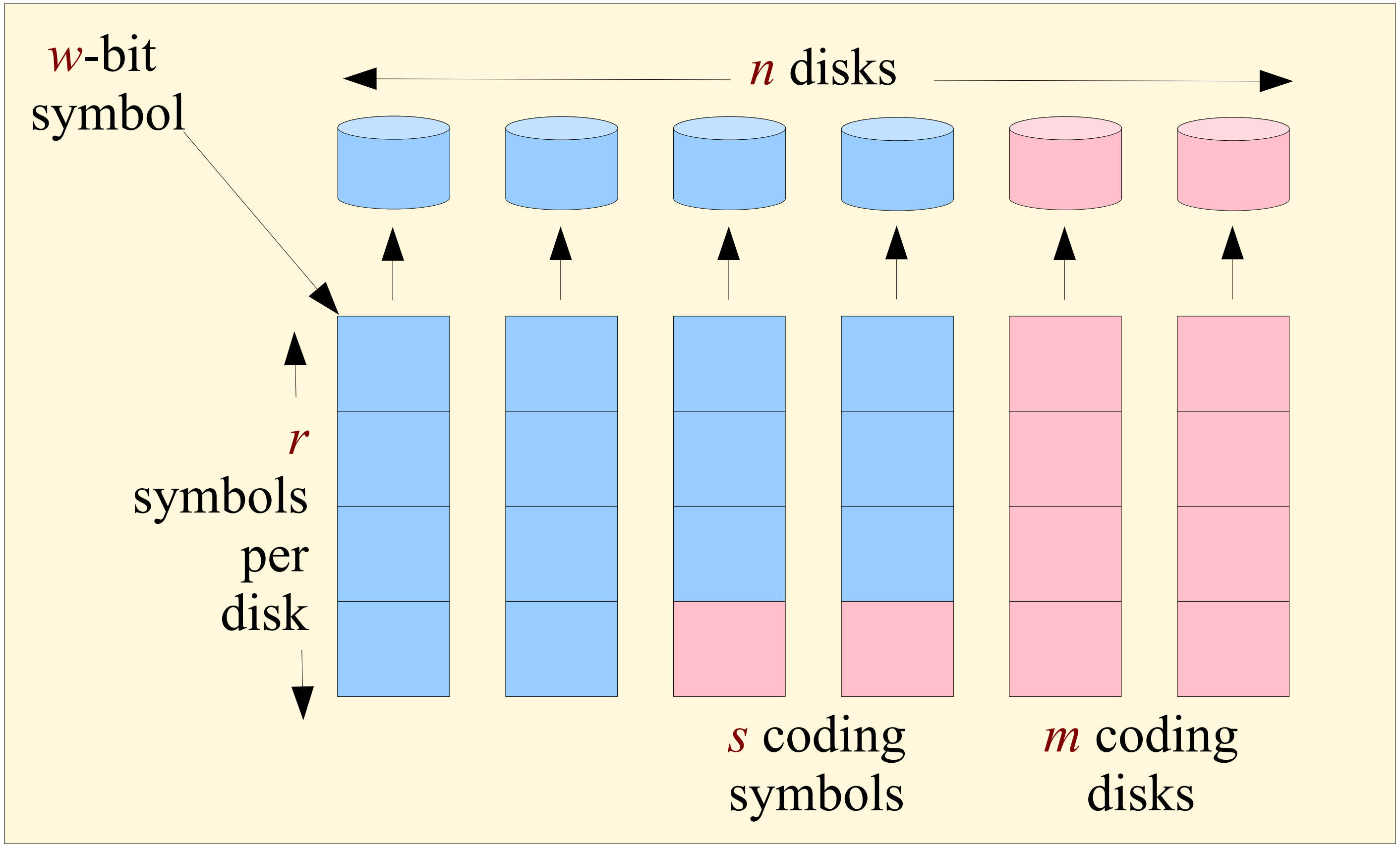
Two Views of a "Stripe"

- The Systems View:
  - Disks hold sectors/blocks rather than symbols.
  - Groups together theoretical stripes for performance.



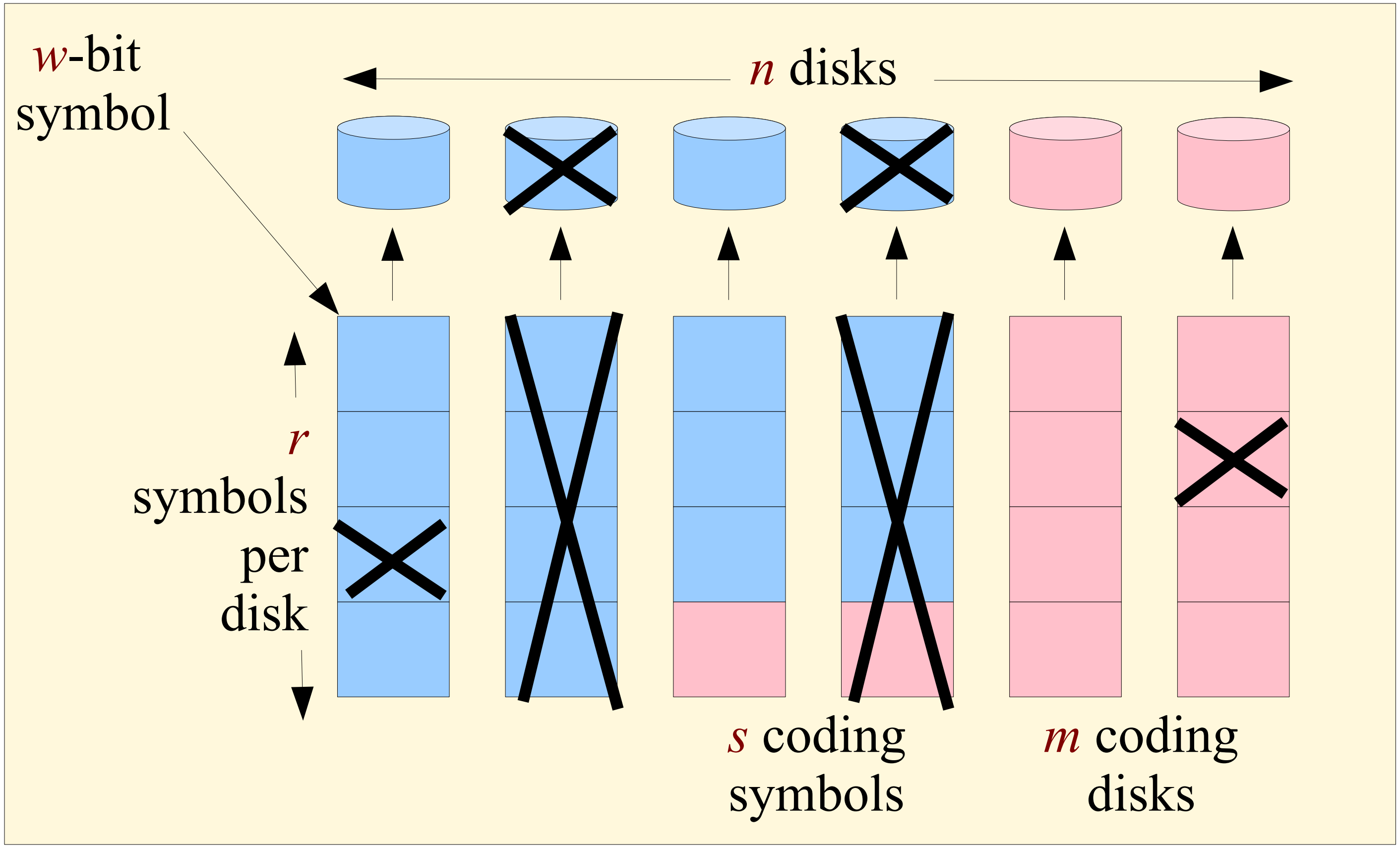
Presentation of SD Codes

- Uses the Theoretical view to define the code.
- With the understanding that you map it to the systems view when you implement it.



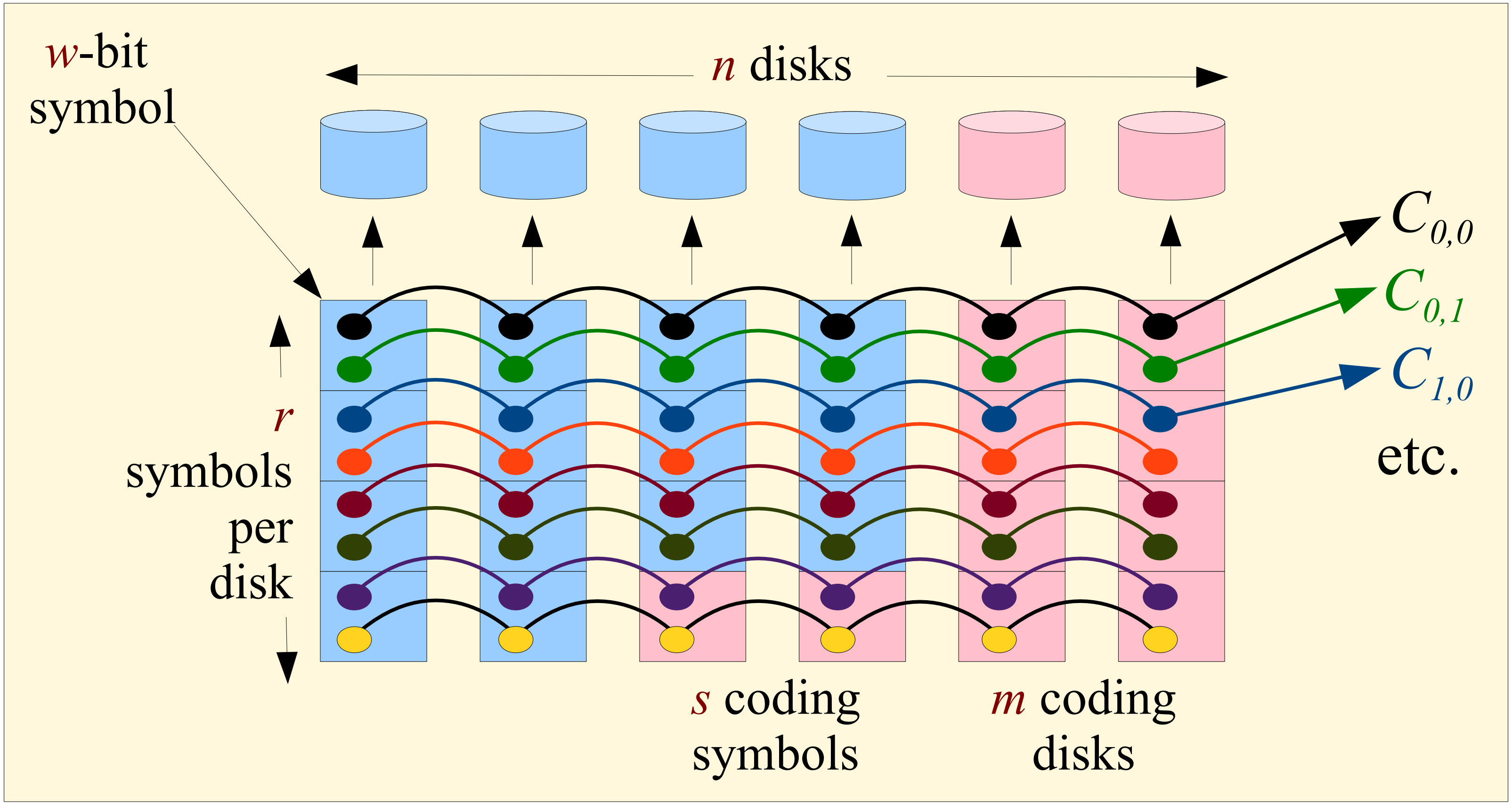
Presentation of SD Codes

- The goal is to tolerate any *m* disk failures, coupled with any additional *s* block failures.



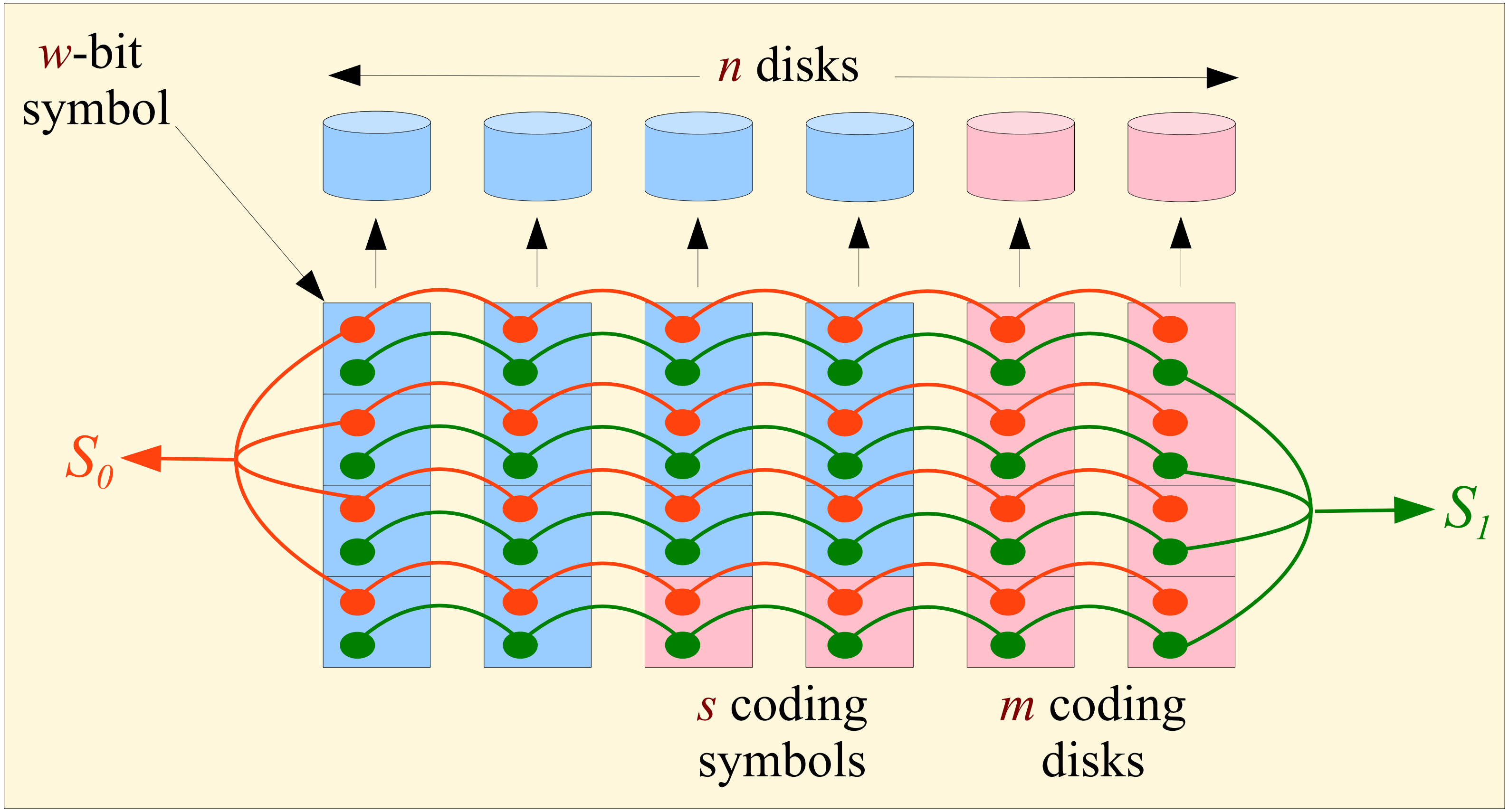
Code Definition

- There are *mr* separate coding equations that involve only rows of the stripes: $C_{i,j}$



Code Definition

- Plus *s* more equations that involve all of the symbols in the stripe: S_x



Code Def nition

- All arithmetic is in a Galois Field *GF(2^w)*
  - Just like Reed-Solomon coding
  - Open source libraries (See Friday's Talk)
  - Larger *w* are slower.
  - But larger *w* yield more codes with the SD property.
- Each equation governed by a different coefficient a_i.

Example: $n=6,\ m=2,\ s=2,\ r=4,\ a_i = 2^i$

Each $C_{i,j}$ equation is the sum of exactly n terms, partitioned by rows.

$$1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \;\circleddash\!\!\rightarrow 0 \qquad C_{0,0}$$

$$1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \;\circleddash\!\!\rightarrow 0 \qquad C_{1,0}$$

$$1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \;\circleddash\!\!\rightarrow 0 \qquad C_{2,0}$$

$$1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \;\circleddash\!\!\rightarrow 0 \qquad C_{3,0}$$

$$1 \oplus 2 \oplus 2^2 \oplus 2^3 \oplus 2^4 \oplus 2^5 \;\circleddash\!\!\rightarrow 0 \qquad C_{0,1}$$

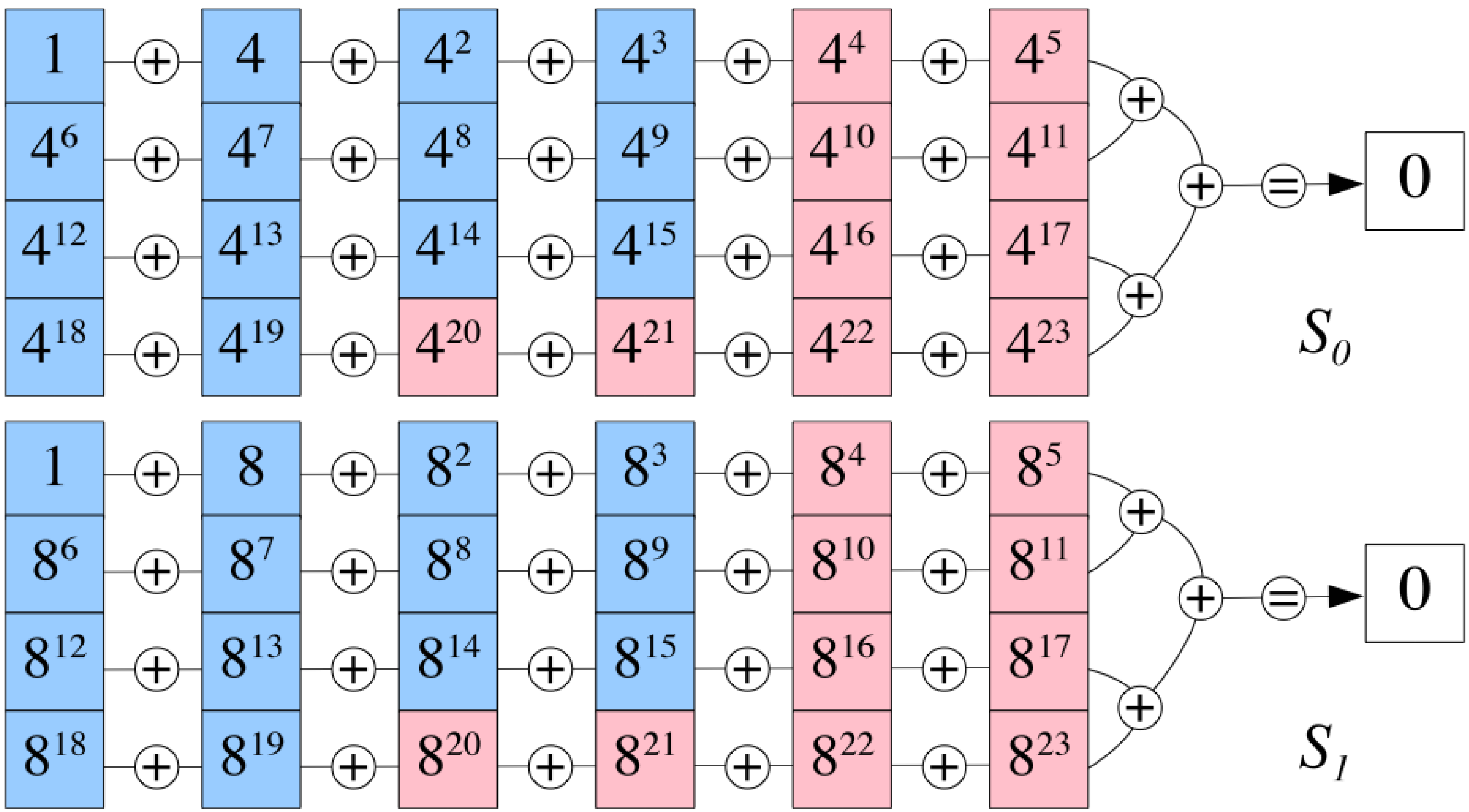
$$2^6 \oplus 2^7 \oplus 2^8 \oplus 2^9 \oplus 2^{10} \oplus 2^{11} \;\circleddash\!\!\rightarrow 0 \qquad C_{1,1}$$

$$2^{12} \oplus 2^{13} \oplus 2^{14} \oplus 2^{15} \oplus 2^{16} \oplus 2^{17} \;\circleddash\!\!\rightarrow 0 \qquad C_{2,1}$$

$$2^{18} \oplus 2^{19} \oplus 2^{20} \oplus 2^{21} \oplus 2^{22} \oplus 2^{23} \;\circleddash\!\!\rightarrow 0 \qquad C_{3,1}$$

Example: $n=6,\ m=2,\ s=2,\ r=4,\ a_i = 2^i$

Each S_x equation is the sum of all *nr* terms.



Decoding

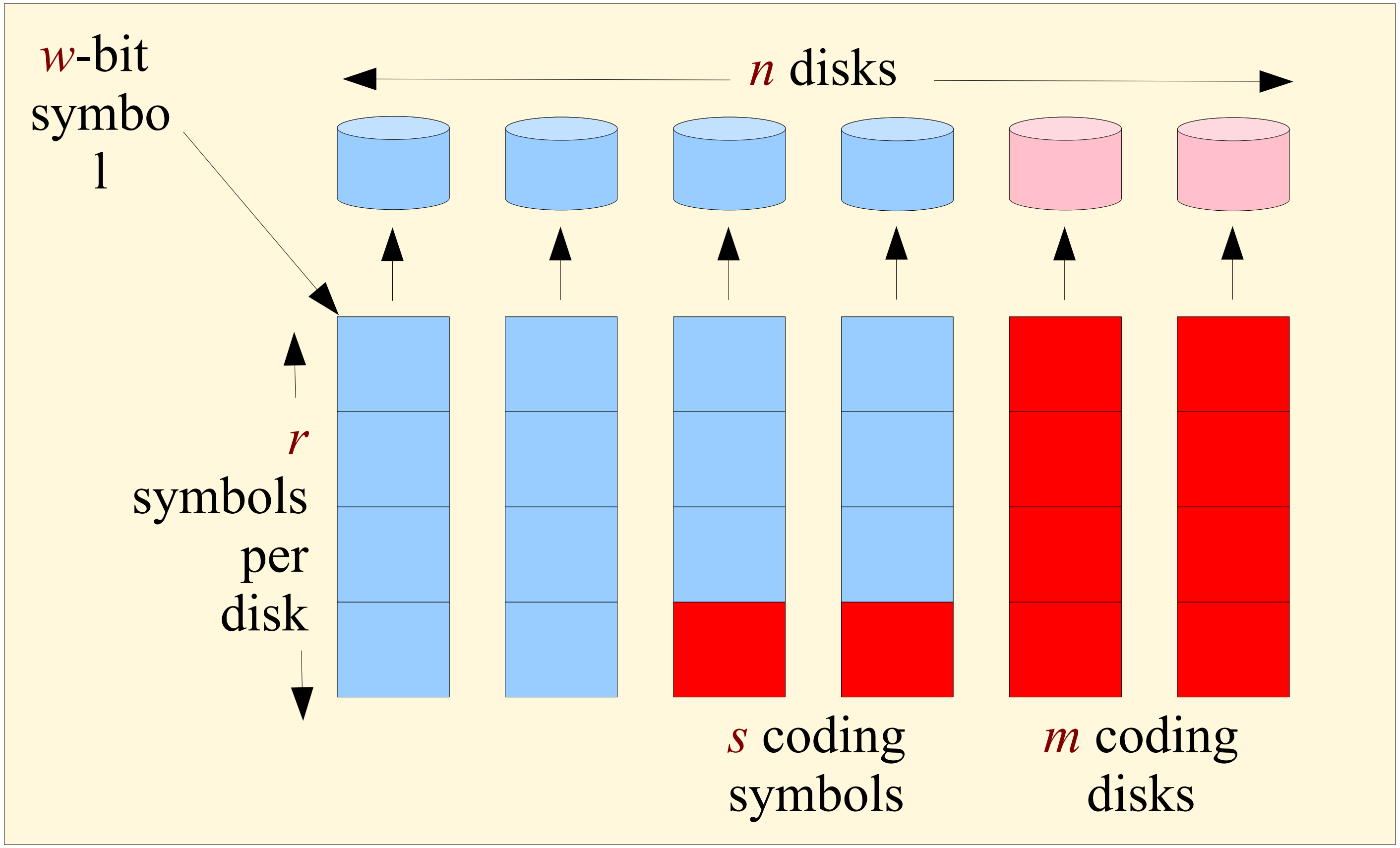
- Recall that there are *mr+s* equations.
- When *m* disks and *s* sectors fail, you lose *mr+s* symbols in the stripe.
- That gives you:

mr+s equations with *mr+s* unknowns.

- Use standard algebra to solve.
- (We went over this in yesterday's tutorial.)

Encoding

- It's just a special case of decoding.
- For that reason, the location of the coding symbols is really arbitrary.



SD Code Constructions

- Given *n, m, s* and *r*.
- Our goal is to find *m+s* coefficients a_i such that every combination of *m* disk and *s* sector failures may be tolerated in $GF(2^w)$, where:
  - *w* = 8 is preferred (because it's fastest),
  - Then *w* = 16,
  - Then *w* = 32.

SD Code Constructions

- When $a_i = 2^i$, we have some theory, which allows us to test a stricter, PMDS condition.
  - We call this the "Main Construction."
- Otherwise, we simply test all failure scenarios.

$$\binom{n}{m}\binom{r(n-m)}{s} \text{ of these.}$$

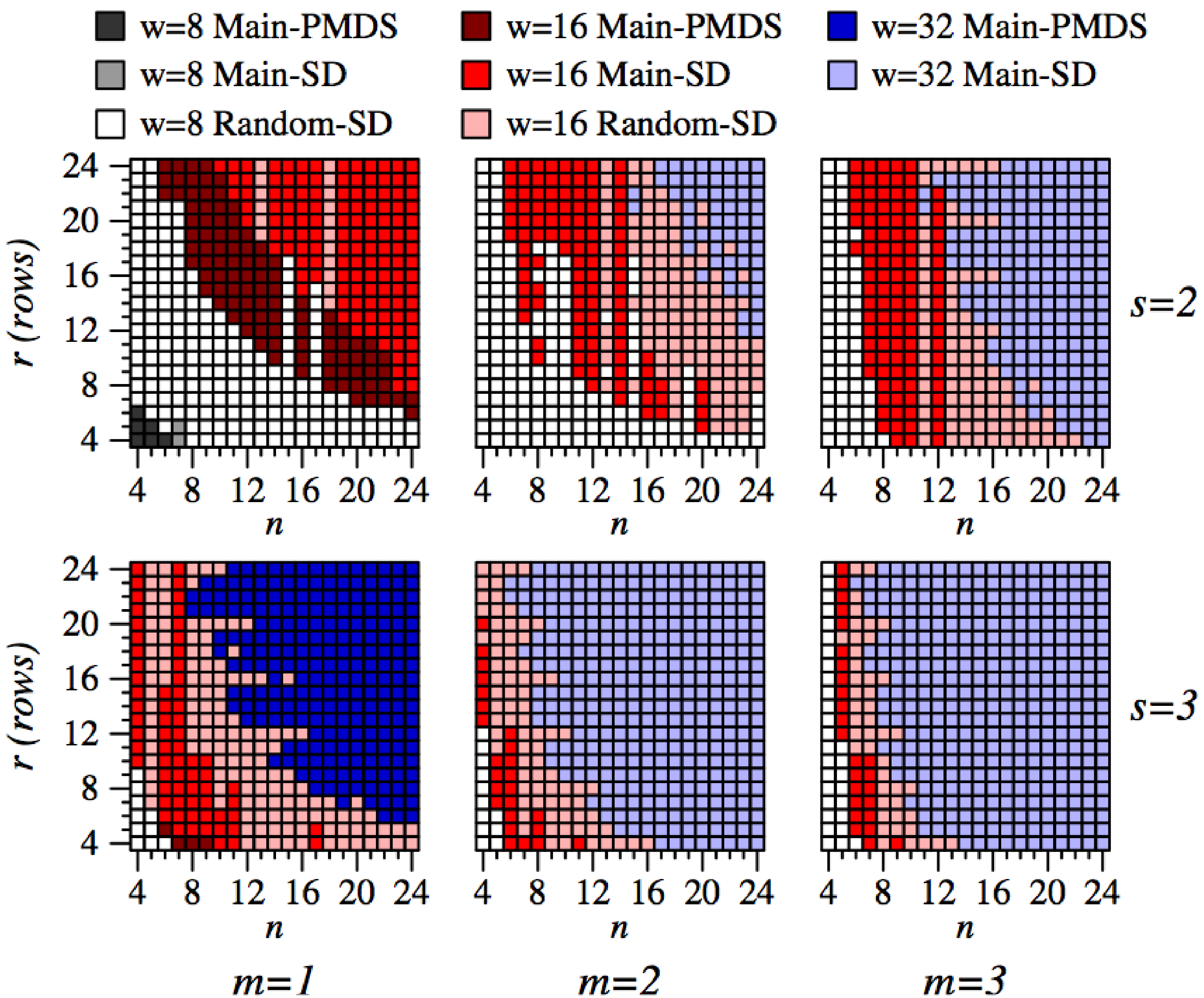
- Do it with brute-force enumeration.
  - (I made it a lab in my CS302 Algorithms course)

SD Code Constructions

- When $m = 1$ and $s = 1$, the main construction is PMDS (therefore SD) when $n \le 2^w$.
  - This is the RAID-6 replacement.
  - $w = 8$ handles 256-disk systems.
- When $m > 1$ and $s = 1$, the main construction is PMDS (therefore SD) when $nr \le 2^w$.

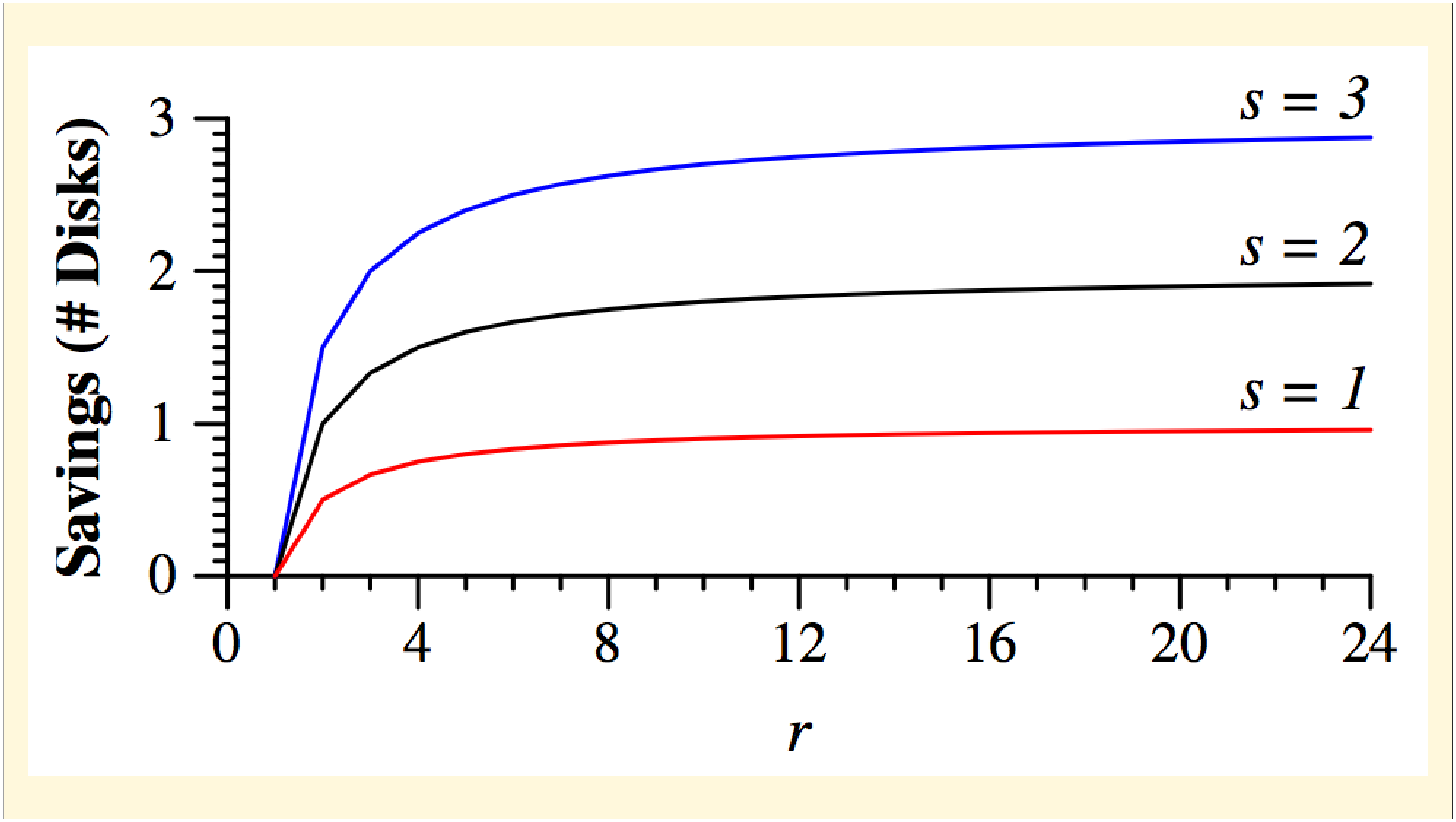
Otherwise...

They exist, but not in any general form.



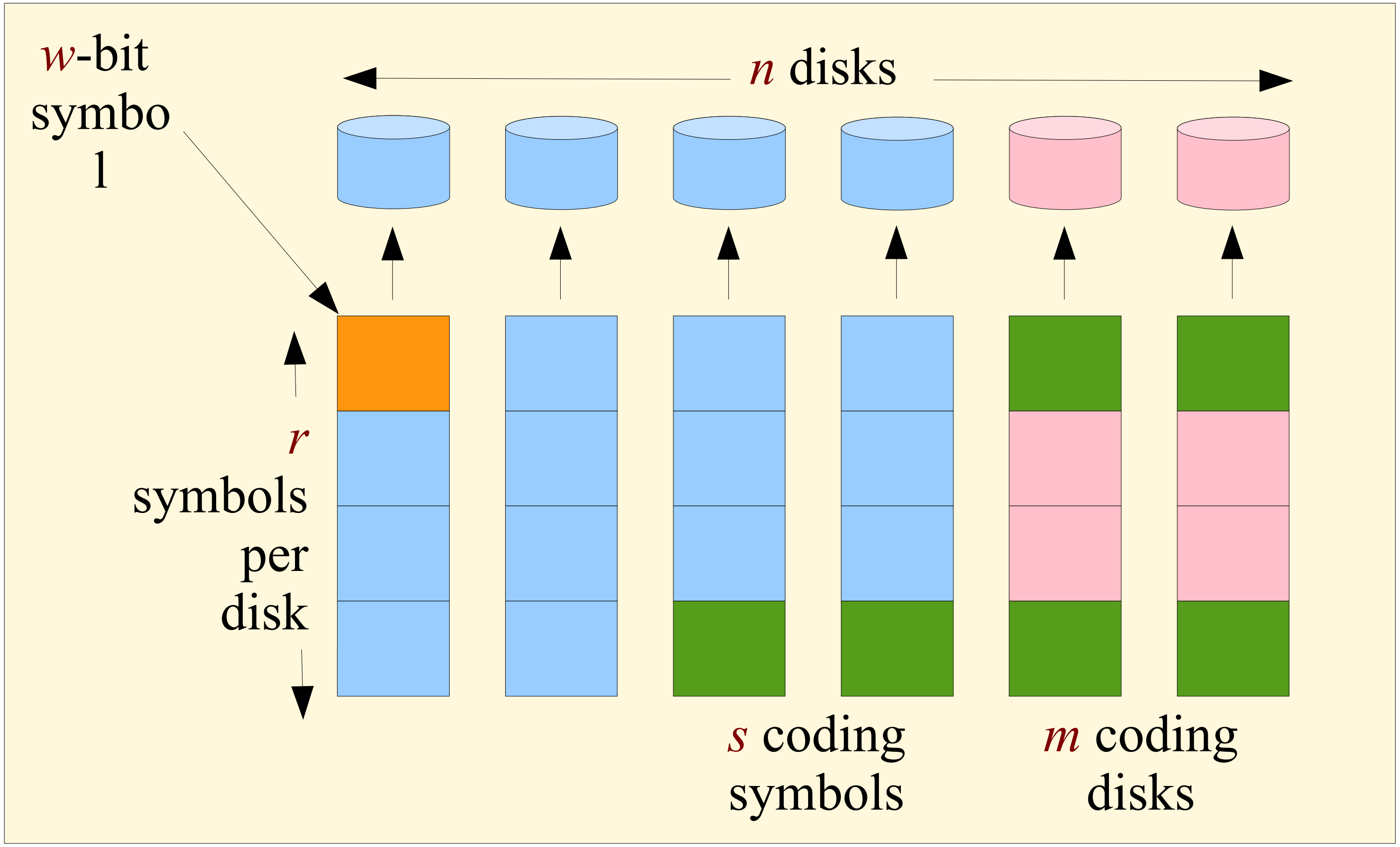
Properties: Storage Overhead

- Pretty obvious, but also pretty drastic.



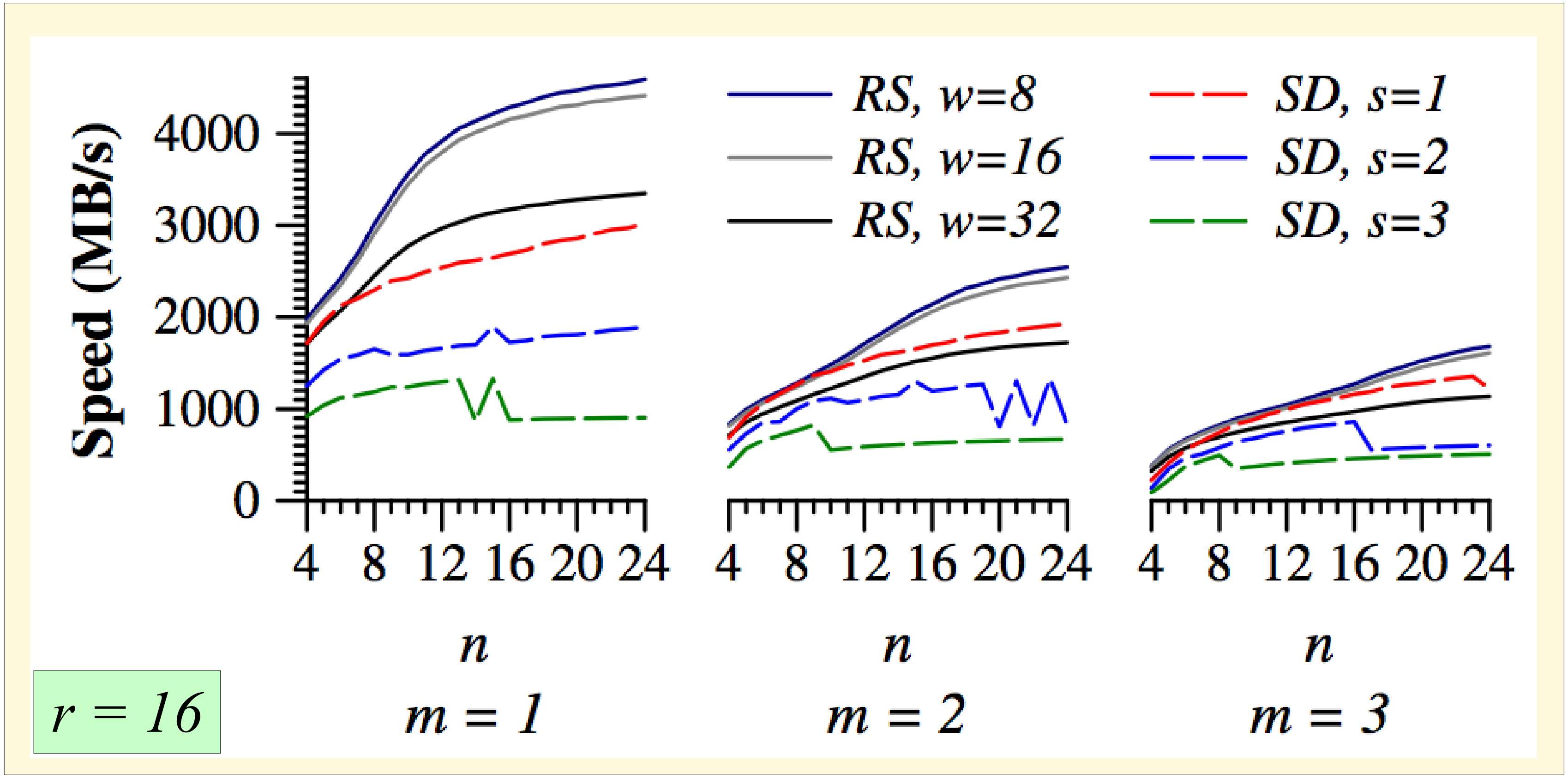
Properties: Update Penalty

- Roughly *2m+s* – not too good.
- Applicable to cloud/log-based systems.



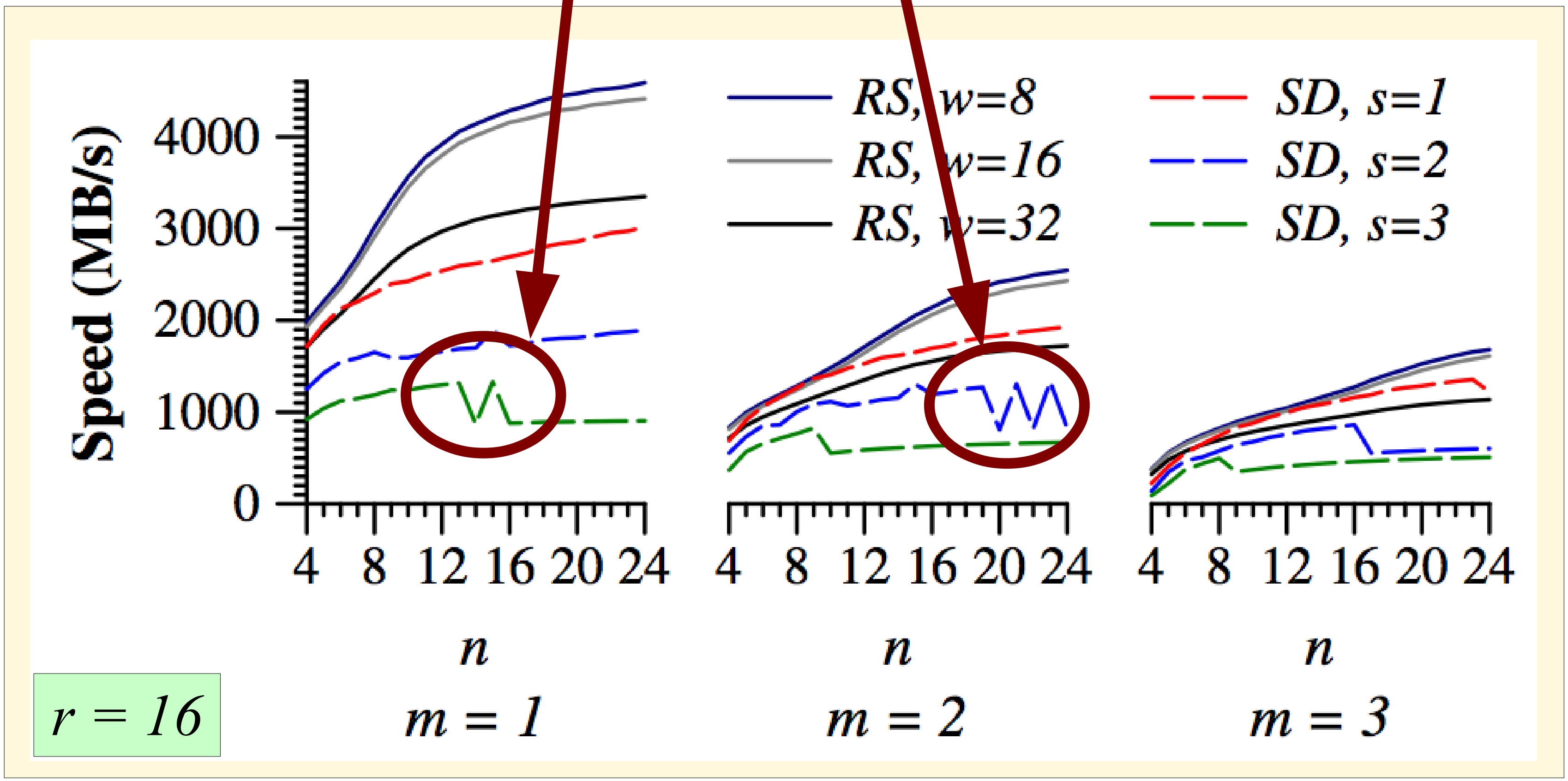
Properties: Encoding Speed

- 32M stripes. Intel Core i7, 3.02 GHz.
- Using SSE for GF Arithmetic (Friday)



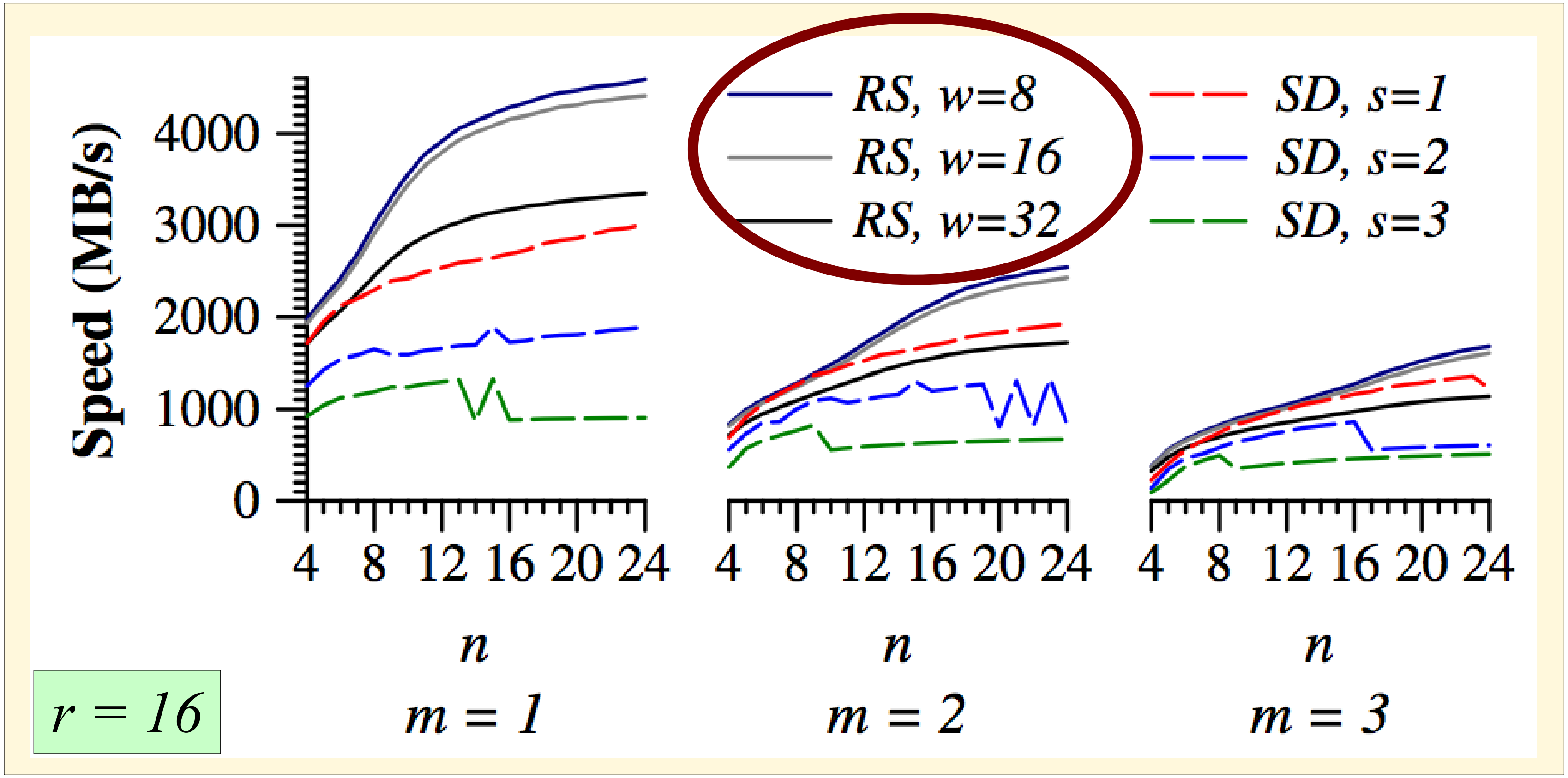
Properties: Encoding Speed

Jagged lines are when you switch between values of *w*.



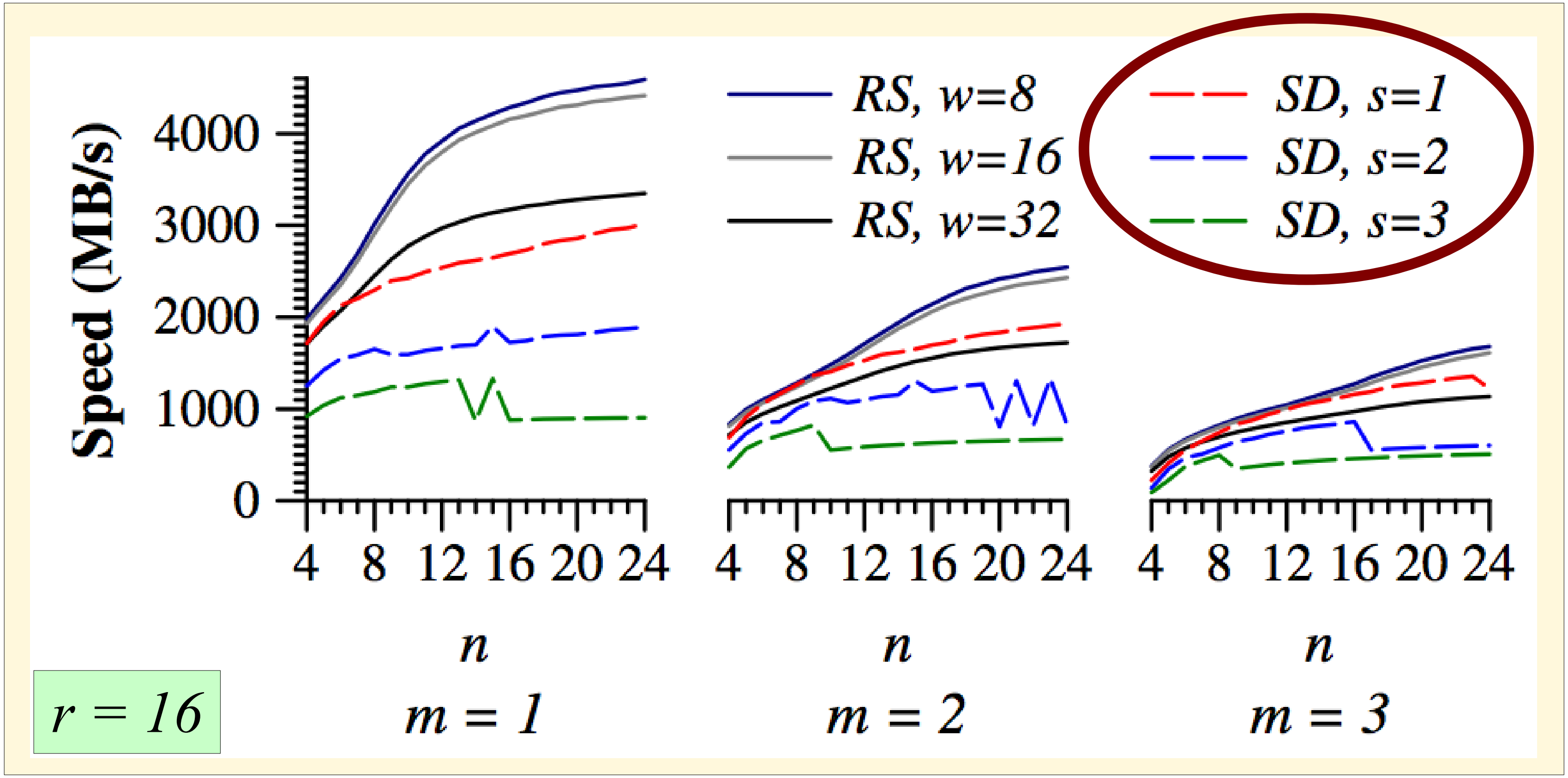
Properties: Decoding Speed

Up to *m* failures per row equals Reed-Solomon Speed



Properties: Decoding Speed

For maximum failures, decoding speed equals encoding speed.

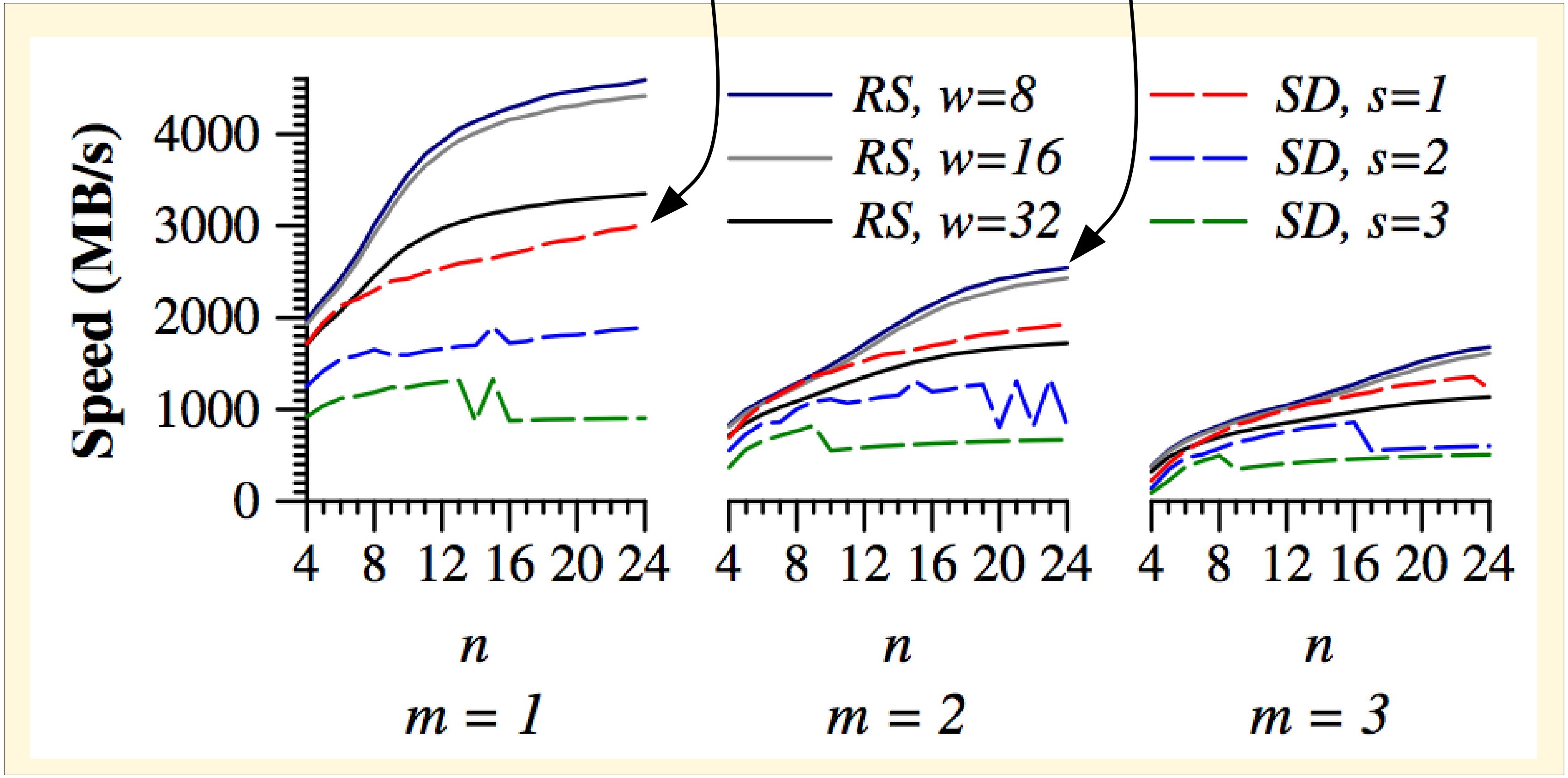


Bottom line

Sure, it's slower than RS coding, but its faster than using extra coding disks.

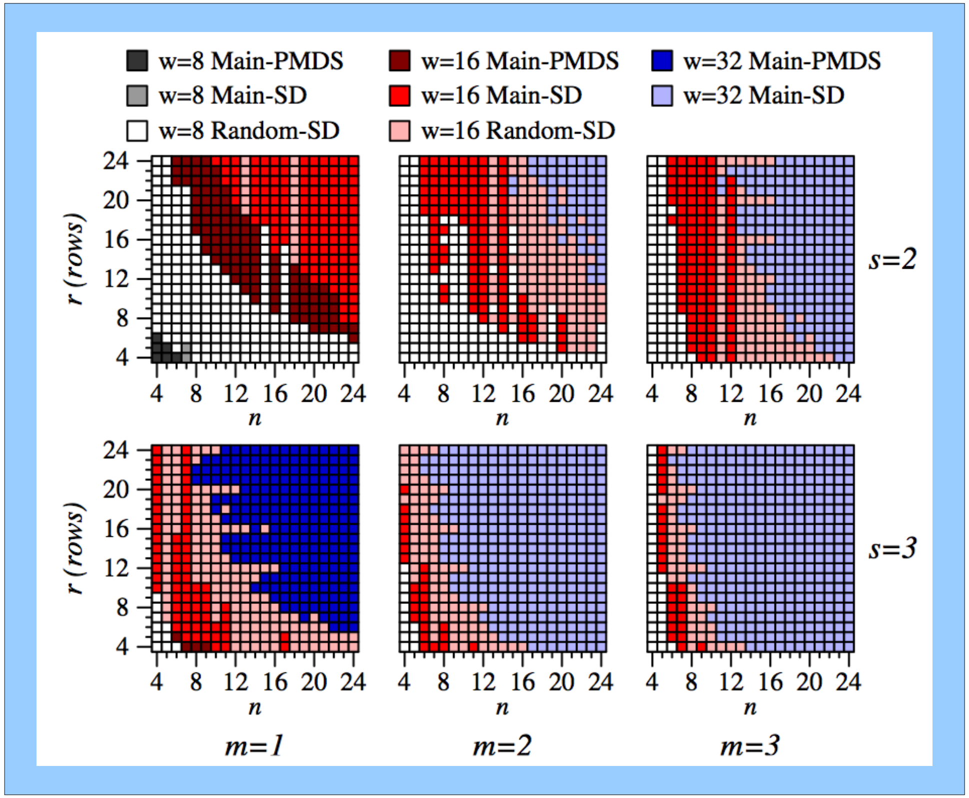
The RAID 6 Replacement

RAID-6



Open Source Code

- SD Programs available from my web site (C).
- Includes encoder/decoder, plus all the constructions from the big yucky picture.
- Fast SSE.
- Doesn't implement RAID – intent is to be a first building block.
- Releasing on Friday.



Related Work

- "Couldn't I just use a *(n,n-m-s)* Reed-Solomon code?"
- Yes, but:
  - Then all coding symbols are functions of all of the data words.
  - Update penalty is all coding blocks.
  - Decoding the common case is expensive.
- "Intradisk Redundancy" [Dholakia, 2009]
  - Reduced failure coverage.

Related Work – Two recent codes

- PMDS Codes
  - IBM
  - *IEEE Transactions on Information Theory, 2013.*
  - Same methodology as SD codes, but with enhanced theory for verifying constructions.

- LRC Codes
  - Microsoft Azure
  - *USENIX ATC 2012.*
  - Intended model is for systems where each block is on a different disk.
  - Current constructions limited to $m = 1$.

Both codes are "maximally recoverable," meaning they tolerate more failure scenarios, with the SD scenario being a subset (Overkill for RAID).

Conclusion

- New erasure-coding methodology to address the failure mode of current storage systems.

- In particular, covers the RAID-6 failure mode without the wasted storage.

- Yes, you've gotta eat some math, but I've got open-source C code that does it all for you.
  - The Galois Field arithmetic.
  - The decoding equations.
  - Constructions for $n, r \leq 24, m, s \leq 3$.

- Performance better than Reed-Solomon substitutes.

SD Codes: Erasure Codes Designed for How Storage Systems Really Fail

James S. Plank

University of Tennessee

USENIX FAST

San Jose, CA

February 13, 2013.