Erasure Coding for Flash Devices and Systems

Flash Memory Summit 2015
PreConference Seminar A

Steven Hetzler
IBM Fellow
Manager, Cloud Data Architecture

Outline

- Discussion of uncorrectable bit errors (UBER)
- Basic erasure coding and non-recoverable read errors
- System failure targets
- PMDS codes
- Fun and games with erasure codes

UBER: Uncorrectable Bit Errors

The "other" component of reliability

- UBER is when there are more bit errors than the sector ECC can correct
  - For example, if the sector ECC can correct 50 bits, but there are more than 50 bits in error
- One component of non-recoverable read errors (NRRE)
  - 2 outcomes of an NRRE event:
    - The ECC detects the error count is too large, and declares the sector lost
    - The ECC blissfully applies the correction and produces an incorrect value (miscorrection)
      - This can be messy, as the number of errors will be > 2 x correction_bits + 1
      - It's common to add CRC to catch such events and convert to NRRE events
- I'll use the term NRRE going forward in this analysis

Non-Recoverable Read Errors

NRRE events

- NRRE events are contributors to data loss
  - Impact depends on the system architecture
  - Loss is at least a sector worth of bits

- NRRE is specified as an interval: e.g. < 1 in 10^{14} bits
- Or as a rate: e.g. <=10^{-14} per bit

- 10^{14} bits seems really large
  - But there are 0.08 x 10^{14} bits in a terabyte!

NRRE Specifications

Alternative expressions which are easier to use

- Express as rate per TB transferred
  - Nice for computing from data moved
  - NRRE/TB = error_interval/8 x 10^{12}
- Express as sector failure probability per operation (sector read)
  - More accurate, since we lose a sector on an NRRE event, not a bit
  - psfail = sector_bits/error_interval

Alternate NRRE Specifications

Some typical specifications (assume 1kB sectors)

- Examples of interval specifications and their equivalents
- Both the probability per TB and the probability of failure are easier to use for system reliability

	Consumer HDD	Enterprise SSD
Typical NRRE Spec (b)	**1e14**	**1e17**
NRRE/TB	**8%**	**8e-5**
psfail	**8.2e-11**	**8.2e-14**

NRRE Specs and Data Loss

Simple to estimate

- psfail = sector_bits/error_interval
  - Assuming error-interval is >> 1
- Sector_Ops/Y = 3,600*8,760*IOPS*sectors_per_IO*duty

  3,600: Seconds per Hour; 8,760: Hours per Year; IOPS: IO per Second; duty: Duty Cycle

- Mean Y/Sector Loss = 1/(Sector_Ops/Y * psfail)
- Duty cycle effects are small here
  - R/W typically 70/30, but depends on application
  - Active duty cycle: ~80% enterprise, ~20% consumer

Probability of NRRE per Year

We can work out annual reliability using prior equations

- Consider a consumer SSD and an enterprise SSD

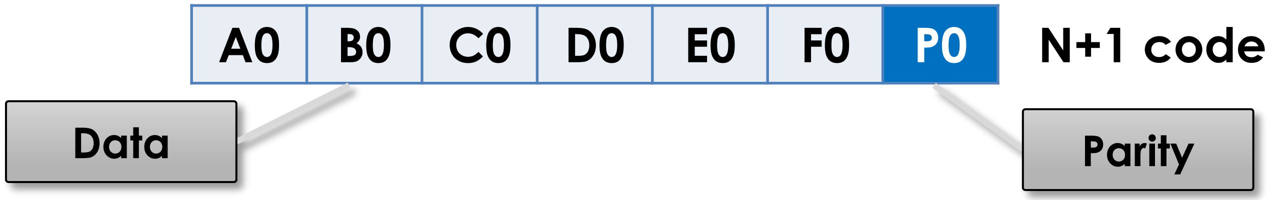
Device	Consumer SSD	Enterprise SSD
IOPS (4kB)	20,000	200,000
Sector Ops/Y (@duty)	1.3e11	5.1e12
NRRE interval (bits)	1e16	1e17
psfail	8.2e-13	8.2e-14
Mean Y/Sector Loss	2.4	0.6
Sector Loss/Y EV	0.4	1.7
MTTDL (MHours)	0.02	0.005
Drive MTBF (MHours)	2	2

- Rate of occurrence is high relative to 2MH drive MTBF
  - Good idea to use an erasure code to protect against NRRE
  - Oh, and device failure as well..

Erasure Correcting Codes

Protecting against device loss and sector loss

- Erasure correcting codes protect against unit loss
  - Can be hardware or software based
  - Unit can be a sector, a chip, a drive, …
- Terminology
  - Parity unit
    - A unit containing erasure code information
  - Erasure
    - An error whose location we know
    - Such as a unit that has failed



Data Loss Types

- Array Loss
  - Lose a 1st unit
  - Start rebuild onto spare
  - Lose 2nd unit during rebuild
  - Large data loss

- NRRE Loss
  - Lose a first unit
  - Hit an NRRE on rebuild
  - Small data loss

- Example here is for N+1 code

Should We Worry About NRRE?

Sometimes, we need to read the entire drive

- We can compute the expectation value for hitting an NRRE during a drive read (DR) operation
- ev_NRRE_DR = psfail*sectors_per_drive
  - (If << 1, then same as probability)
  - Assuming NRRE are not correlated (which isn't true for flash…)

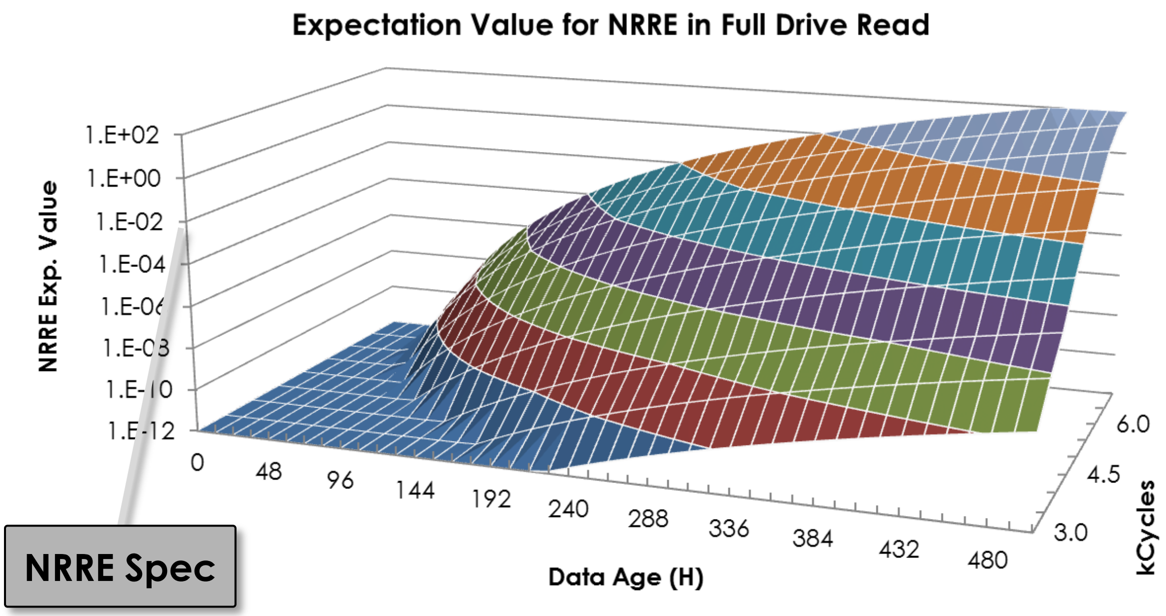
Device	Consumer SSD	Enterprise SSD
Sector kBytes	1	1
Drive TB	1	1
Sectors/drive	1e9	1e9
NRRE interval (bits)	1e16	1e17
psfail	8.2e-13	8.2e-14
ev NRRE/DR	8e-4	8e-5

NRRE on Rebuild

- An issue for rebuilds with no further protection
- On rebuild, we need to read all the remaining data
- Will have a distribution of cycle count/data ages
- Data shown for SSD with 1e15 NRRE interval spec (=4.3e-12)
- This 1TB SSD has 2e9 sectors, so @ spec, NRRE ev is 9e-3!

- Much of the surface is out of spec
  - Spec isn't very good to begin with
- An array has multiple devices
  - EV will be multiplied by # drives read
  - w/8 drives, spec prob NRRE is 7%!
- NRRE are correlated by common cycle count and data age in an erase block

Device data on expectation value when reading device capacity on a 1TB SSD



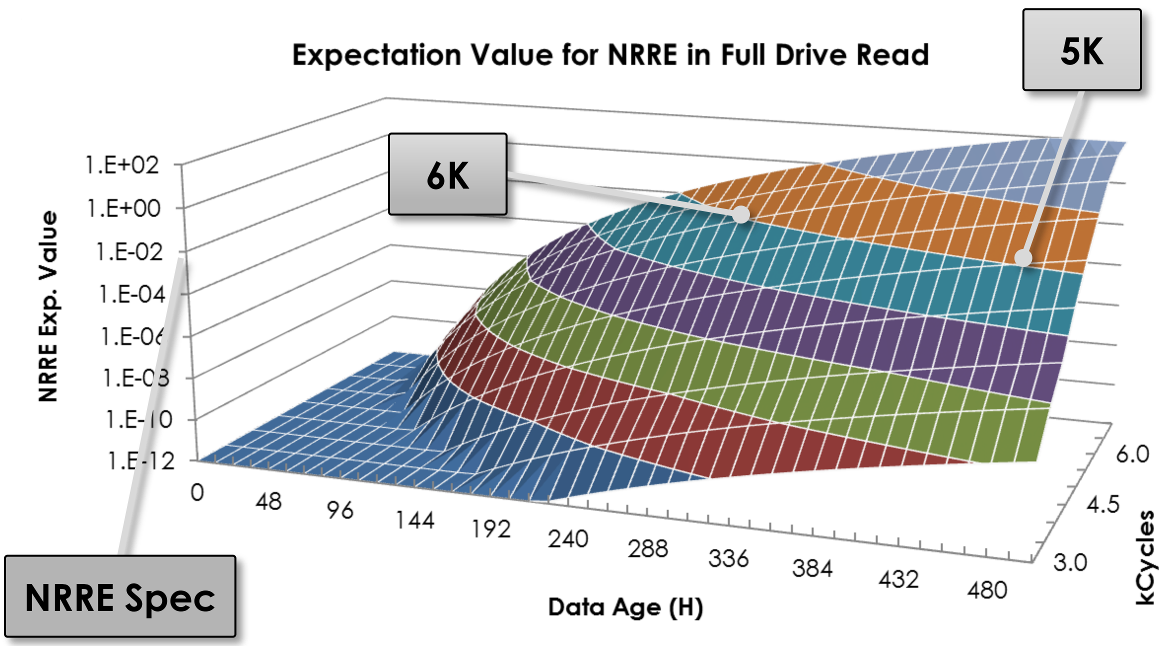
Cost of Protecting Against NRRE

- NRRE impacts reliability during rebuilds
- If units are lost, we need to rebuild the missing data
- The code will determine how many failures can be tolerated
- If there is no parity left, a sector loss becomes a data loss event

- Scrubbing can sometimes help
  – But adds to cycle pressure
- If we know the surface, we can pick a maximum data age for scrubbing

kCyc	Age(H)
3	1,500
4	850
5	460
6	280

Device data on expectation value when reading device capacity on a 1TB SSD



NRRE and Erasure Correcting Codes

Protecting against device loss and sector loss

- There are erasure codes which protect against both device loss and NRRE
  - While scrubbing can help, it's not enough by itself
  - Reducing the rebuild time window won't help much
    - NRRE expectation value largely independent of rebuild time
- Probability of data loss depends on the erasure code properties and the device properties
- Codes can be designed which can efficiently protect against these events
- First, determine how much protection is needed

Failure Targets

How to create data loss targets for a system

- Failure events should be expressed per unit time
  - This is how the customer experiences events
    - Not per byte, or per IO

- Program based targets
  - Look at the behavior of an entire field population
    - Helps for modeling warranty costs
    - Also helps with program financial targets

Program Failure Targets

Inputs to program based tarets

- Install base
  - Unit ships per year, field lifetime, program lifetime
- Usage characteristics
  - Total data operations, total data transferred
- Failure tolerance
  - Depends on the failure type
  - Is it a warranty event, loss of availability, loss of data or customer near-death experience?

Modeling Failures

(Precision is highly over rated here…)

- We need only compute first order terms!

- Why?
- Assumptions are errors are independent of each other and of time
  - These are rarely true
    - (Well, essentially not at all with Flash…)
- The biggest deviations will be from these assumptions
- So first order is good enough
  - Still a good idea to verify which terms are second order
- Thus, we can compute from binomials
  - Easy to do in a spreadsheet too!

System Data Loss Targets

Program Design	Value	Notes
Field lifetime (Y	5	Typical
Mean field units	250,000	Assume a successful program
Units/Array	8	Erasure code span
IO size (kB)	4	Assume transaction processing
Total field IOs	2e18	Assume 50,000 IOPS/unit
Arrays/field	31,250	

Program Loss Targets	Value	Notes
Data Loss Events/program	1	For the entire program
Target Prob data loss/array/Y	2e-6	Assume a successful program

System Data Loss Targets - Device

The system is built from these devices

- Here is an example device we might encounter
  - Recall ev_NRRE_DR = psfail*sectors_per_drive

Item	Value	Item	Value	Item	Value
IOPS	50,000	Capacity (TB)	1	Sector kB	1
AFR	0.5%	NRRE	1e16	Sectors	1e9
ev NRRE DR	8.8e-4	Rebuild (H)	2.8		

Computing Sector Failure Targets

psfail that meets the system target

- psfail1f is the sector failure rate with 1 unit failure
  – We have a 1TB SSD and1kB sectors here

- psfail1f needed to meet array data loss target with 1 unit failure
  – psfail1f = TgtDataLoss/Y / (sectorsread*P1fail/Y)
  – Sectorsread = (1TB/1kB)*(8 - 1)

    (1TB/1kB): Sectors per SSD; (8 - 1): SSDs read to rebuild

  – psfail1f = 4.55e-15
  – (NRRE1f = 4.87E-19 is the equivalent NRRE to psfail1f)

Cumulative Binomial

Useful for estimating failures

- cumbinomial(fails,trials,errorrate)
  - Fails is the number of failures
  - Trials is the total number of events (ops, bits, etc.)
  - errorrate is the failure rate per trial (e.g. ber, AFR)

- This is the cumulative binomial distribution
  - In Excel, use the Binom.Dist function as:

    1-Binom.Dist(fails,trials,errorrate,TRUE)

    - Be aware sometimes this runs out of precision when it shouldn't
    - Then it just reports 0 – happens around 1e-15, which isn't that small

Array Down 1 Unit

Probability an array has lost 1 unit in a year

- P1fail is the probability there is one failure in an array
- Probability an array is down 1 unit:
  - P1fail/Y = 1-binomial(0,arraysize,AFR) = 3.9% here
  - Not surprising:
    - 0.5% AFR * 8 units ~= 4%
    - 0.5% AFR = 1.75MH MTBF
  - Large MTBFs like 2MH don't mean things are super reliable
    - AFR = 8760/MTBF
    - So, at 2MH MTBF, annual failure rate is 0.44%
    - That is, for every 100 drives, there is a 44% chance of 1 of them failing in a year

Did you Notice?

Our device is out of spec for the system

- Recall our SSD had NRRE 1e16
- Which has psfail = 8.8e-13
- But we need psfail = 4.55e-15!
- So, this device doesn't work here as specified
  - And a drive at 1e-17 won't quite work either
  - Just a little hint of what's to come

Reliability for N+1 Array

Single parity per array: 87.5% efficient on 8 units

- There are 2 terms – array loss and NRRE loss
  - Both start at prob 1 unit is lost
  - Array loss
    - And 2^{nd} unit lost during rebuild interval (RH is rebuild time in hours)
    - P2Fail/R = 1-binomial(0,arraysize-1,AFR*RH/8760)
      = 1.1e-5
  - NRRE loss
    - And NRRE occurs during rebuild
    - PNRRE/R = ev_NRRE_DR * (arraysize-1) (if ec_NRRE_DR << 1)
      = 5.6e-3
- $PFail/Y = P1Fail/Y\sqrt{(PNRRE/R)^2 + (P2Fail/R)^2}$
  = 2.2e-4
  - Oops – 100x out of spec (which is 2e-6)
    - Prob array loss only is 2e-7, so this would be OK

Reliability for N+2 Array

Two parities per array: 75% efficient on 8 units

- Again, there are 2 terms – array loss and NRRE loss
  - Both start at prob 1 unit is lost * prob 2nd fail in rebuild
    - P2Fail/Y = P1Fail/Y* (1-binomial(0,arraysize-2,AFR*RH/8760))/2 = 2.2e-7
  - Array loss
    - And 3rd unit lost during rebuild interval (RH is rebuild time in hours)
    - P3Fail/R = 1-binomial(0,arraysize-2,AFR*RH/8760) = 9.7e-6
  - NRRE loss
    - And NRRE occurs during rebuild (if ec_NRRE_DR << 1)
    - PNRRE/R = ev_NRRE_DR * (arraysize-2) = 4.8e-3
- $PFail/Y = P2Fail/Y\sqrt{(PNRRE/R)^2 + (P3Fail/R)^2}$ = 1.1e-9
  - Much better than spec, which is 2e-6

Results

So, we are done then?

- N+1 didn't meet the data loss target, but N+2 did
- Note that N+2 operates at 75% data efficiency
  - Uses an entire unit's worth of data to protect against an NRRE
  - This is wasteful
  - You can see it in the terms
    - P3Fail/R = 9.7e-6
    - ev_NRRE_DR = 4.8e-3
  - NRRE term dominates during rebuild
- What we need is an erasure code that separates NRRE protection from unit protection
  - Don't use a hammer to kill a fly (fun though it may be)
- New term: fpof – first point of failure
  - The minimum number of losses that cause a failure

PMDS Codes

Optimized for both device and sector protection

- New erasure codes designed for this very problem
  - (I know, I was there when it happened)
  - Parity group is now multiple sectors from each device (columns)

A0	B0	C0	D0	E0	P0
A1	B1	C1	D1	E1	P1
A2	B2	C2	D2	E2	P2
A3	B3	C3	D3	E3	P3
A4	B4	C4	q_a	q_b	P4

P0 is row 0 parity (Example with 6 units)

P1 is row 1 parity

P2 is row 2 parity

P3 is row 3 parity

P4 is row 4 parity, q_a, q_b group parities

- Unit loss protection via row parities P_n
- Floating sector loss protection via group parities q_n
  - The q_n can be placed anywhere in the parity group
  - Invoked only *after* more than 1 sector in a row is lost
  - This code is called PMDS 1+2 (1 unit + 2 group)

What Are PMDS Codes

Partial Maximum Distance Separable Codes

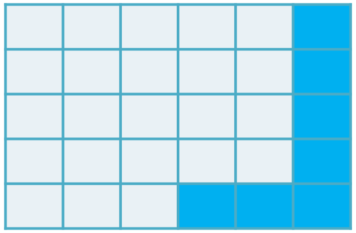
- An extension of MDS codes
- MDS codes can correct as many erasures as they have parities
  - N+1 and N+2 are MDS codes
- PMDS codes are partially MDS in efficiency
  - They have more parities per correction than MDS codes
  - But they can approach the MDS in efficiency
- We classify them as n+m
  - Where n is the number of full parity columns
  - m is the number of group parities
    - The group parities can correct errors anywhere in the goup
  - If the number of rows is large, the efficiency can ~MDS

Blaum, Hafner, Hetzler: "Partial-MDS Codes and Their Application to RAID Type of Architectures", IEEE Transactions on Information Theory 59(7): 4510-4519 (2013).

PMDS Code Correction

Why PMDS codes are so handy

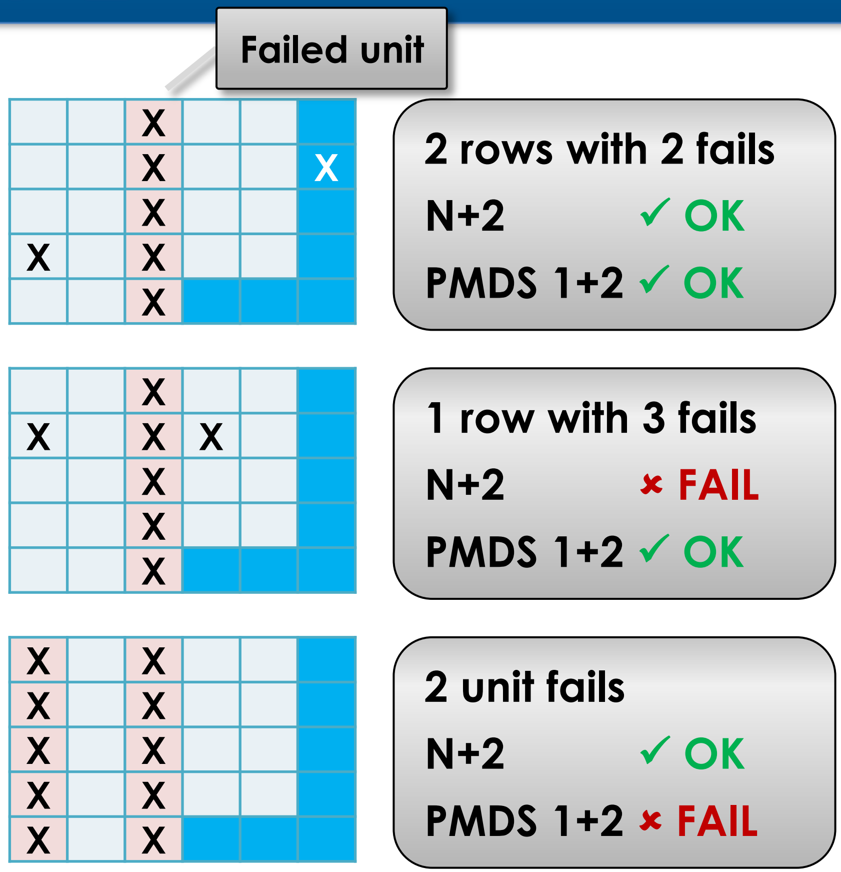
- Consider the PMDS 1+2 code below (6 units)



- It has a minimum Hamming distance of 4 to sector loss
  - That is, it can correct any 3 such failures (4 in a row fail)
- It can correct many patterns of 4 failures
  - Such as 2 rows with 2
  - Or 2 columns with 2 (important after a column failure)
  - These are important for flash as sector failures can correlate in both rows and columns

PMDS 1+2

- PMDS 1+2 in most cases is stronger than N+2
  - Consider rebuild (1 unit fail)
  - N+2:
    - Correct all 1 sector fail/row
    - Correct 0 2 sector fail/row
    - Fpof = **2** sectors + 1 unit
  - PMDS 1+2:
    - Correct 2 1 sector fail/row
    - Correct 1 2 sector fail/row
    - Fpof = **3** sectors + 1 unit
  - For independent failures, first order is # of failures



- **PMDS 1+2 is stronger to sector failure on rebuild**
- **PMDS 1+2 is weaker to unit fails**
  - Mitigated by short rebuild time

PMDS 1+2 Reliability

PMDS 1+2 with 32 rows: 87% efficiency on 8 units

- There are 2 terms – array loss and NRRE loss
  - Both start at prob 1 unit is lost
  - Array loss – same as N+1: P2Fail/R = 1.1e-5
  - NRRE loss
    - Prob of 3 NRREs in the remaining sectors in the group
    - PGFail/R = cumbinomial(3,32x7,psfail) = 1.2e-30
    - Prob rebuild fails = cumbinomial(1,ngroups,PGFail/R) = 3.4e-23
- $PFail/Y = P1Fail/Y\sqrt{(PNRRE/R)^2 + (P2Fail/R)^2}$ = 2.2e-7
  - Which exceeds the spec of 2e-6
- PMDS is almost as efficient as N+1, but 1,000x more reliable

General PMDS Codes

PMDS codes can be highly customized

- PMDS n+m codes can be created for various configurations of n and m
- Examples:
  - PMDS 1+1 – reduced sector loss coverage compared to 1+2
  - PMDS 2+1 – N+2 with single sector loss coverage
  - PMDS 2+2 – N+2 with double sector loss coverage
- I like to have at least n+2 to give some coverage for correlated NRRE events
  - Flash has a high degree of correlation
    - All drives in array have almost identical cycle counts and data ages
- While the example here was a cross-drive array, the analysis holds for intra-drive arrays (across dies)

Some Games We Can Play

We extract more value from the sector protection of PMDS 1+2

- We can allow the unit NRRE to be much greater than the specification, and let the PMDS code reconstruct the data
- We can do this by reducing the power of the in unit sector ECC, improving the overall data efficiency
- I call this DNR ECC ("Do Not Resuscitate")
  - The unit should not try so hard to recover from sector errors

DNR ECC

We let sectors fail at a higher rate with DNR ECC

- Failure (at the flash layer) is acceptable with a proper erasure code at the array level
  - With larger limits than solo devices permit
- System can be optimized by adjusting the correction at each level
- No need to try so hard at the flash layer
  - DNR – we deliberately set a higher failure rate target at the component level
  - Improves flash efficiency, simplifies encode/decode
    - Need to correct fewer errors
    - Makes the components more testable
      - Lowered expectations being more common these days…

Aside on Computing NRRE Targets

How to get the raw bit error rate from the NRRE and the sector ECC

- We can compute the raw ber from the psfail spec and ECC if we know the sector ECC
  - psfail target was 4.6e-15
  1. Assume BCH 66 code on 1kB
     - Corrects 66 bit errors out of 1,024 data bytes
     - Requires 924 check bits
     - sectorbits = databits + checkbits + metadata ~ 9,212
  2. psfail = (1 – cumbinomial(66,sectorbits,ber))/sectorbits
  3. Invert by iteration to solve for ber
  4. Here: ber = 2.65e-3
  5. To meet system target need @ ber 2.65e-3 need 75 bits
     - 9,338 sectorbits

     - Hint: you can use Goal Seek in Excel to quickly iterate to find the ber

Our SSD

Need some further information

- Let's assume our SSD has an internal ECC
- Corrects up to 66 bits in error
- The sector has a total overhead of 924 bits
- So the sector size is 9,212b (8,192b are user data)
- Data efficiency is thus 89%
- Now, we can compute the required psfail to reach our array target
  - And thus the ECC correction bits required

DNR Results for N+1 and N+2

Code Type	N+1	N+2
sparity/pgroup	0	1
sectors/pgroup	10	10
pgroup/array	1e9	1e9
Code data efficiency	0.90	0.80

Failure computations		
parrayfail	2.0e-6	2.0e-6
psfail	4.6e-15	3.4e-8
ECC corr bits needed	75	55
Sector efficiency	0.88	0.90

Net data efficiency	0.79	0.72

Not the answer the judges were looking for!

PMDS DNR Results

Code Type	N+1	N+2	PMDS 1+1	PMDS 1+2
sparity/pgroup	0	1	1	2
sectors/pgroup	10	10	160	1,280
pgroup/Array	1e9	1e9	6.3e7	7.8e6
Code data efficiency	0.90	0.80	0.89	0.90

Failure computations				
parrayfail	2.0e-6	2.0e-6	2.0e-6	2.0e-6
psfail	4.6e-15	3.4e-8	7.2e-9	2.5e-7
ECC corr bits	75	55	56	52
Sector efficiency	0.88	0.90	0.90	0.91

We have a winner

Net data efficiency	0.79	0.72	0.81	0.82

DNR Results with PMDS Codes

PMDS 1+2 makes DNR ECC cost effective

- Efficiency is increased by letting the NRRE (psfail) increase
  - Up to 3% more efficient in this example
- May not sound like much, but worthwhile
  - Goes straight to margin
    - What else would you do for 3 margin points?
  - Can also be used to increase yields
  - May save cost in ECC decoders
  - Can allow use of consumer parts in enterprise applications
- This was just a simple example, we may be able to do better with other configurations
- If you need dual unit failure protection, there are PMDS codes for those as well
  - If 2^{nd} parity is protecting against a second unit failure, it's not available for sector loss protection
    - I have shown you how to do the math

PMDS 1+2 DNR ECC

If we can't change the ECC, we can push the device

- Target data loss per year is 2e-6
- Our example has 3.9% AFR, P2Fail/R = 1.1e-5
- Recall PMDS 1+2 had PNRRE/R = 3.4e-23
- So we can tolerate much higher NRRE

- $PFail/Y = P1Fail/Y\sqrt{(PNRRE/R)^2 + (P2Fail/R)^2}$
- So our target is PNRRE/R = 2e-6 (17 orders higher!)
  - PNRRE/R = cumbinomial(1,ngroups,PGFail/R) = 5e-5
  - PGFail/R = cumbinomial(3,32x7,psfail) = 1.6e-12 (invert)
  - psfail = 9.5e-7
  - This is 1e6 x the psfail = 8.8e-13 for 1e16 devices

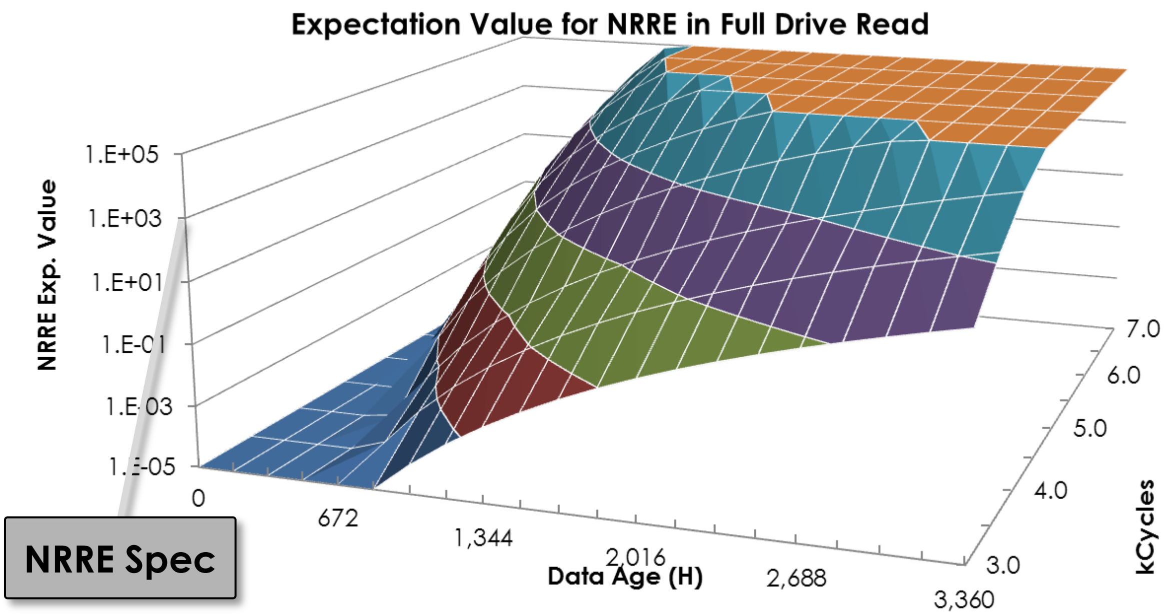
PMDS 1+2 DNR Results

- We have psfail = 9.5e-7
- On the 1TB drive, this gives ev NRRE/DR = 9.5e2 (Yes Virginia, it loses data!)
- This will give us relief in cycle count or data age (our choice)
- Nice gains for adding NRRE protection

- What might be expected

kCyc	N Age(H)	P Age (H)
3	1,500	3,370
4	850	2,500
5	460	1,200
6	280	680
7		520
8		440
9		340
10		260

Device data on expectation value when reading device capacity on a 1TB SSD



Operational Gains From PDMS 1+2

There are advantages to allowing failures

- Endurance gains at constant retention

kCycles	3	4	4	6	7
Retention Gains	2.3	2.9	2.7	2.4	2.5

- Retention gains at constant endurance

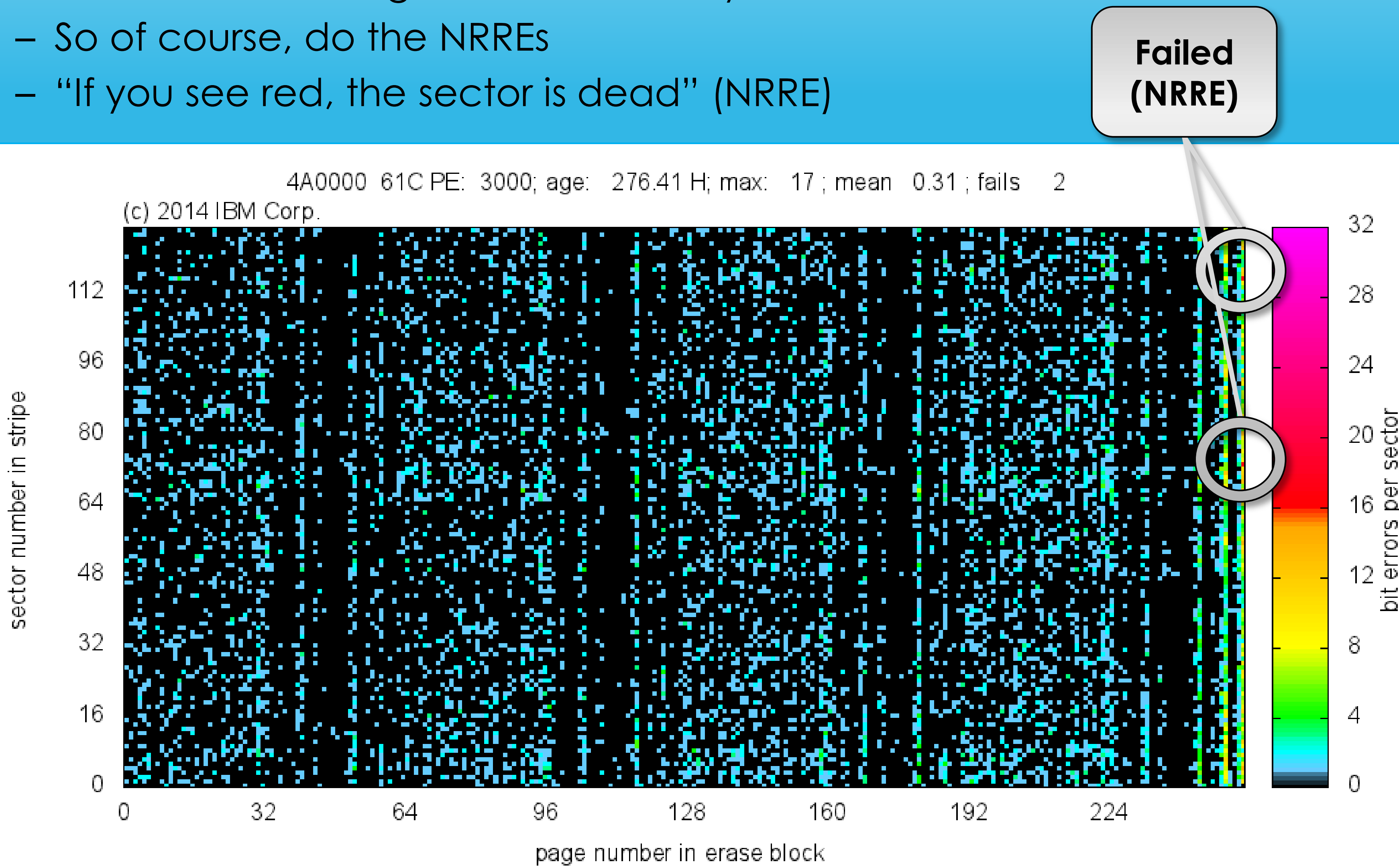
Retention (H)	1,500	850	450	300	200
Cycle Gains	1.6	1.4	1.6	1.6	1.8

- The gains here are substantial

- However, reality can intrude

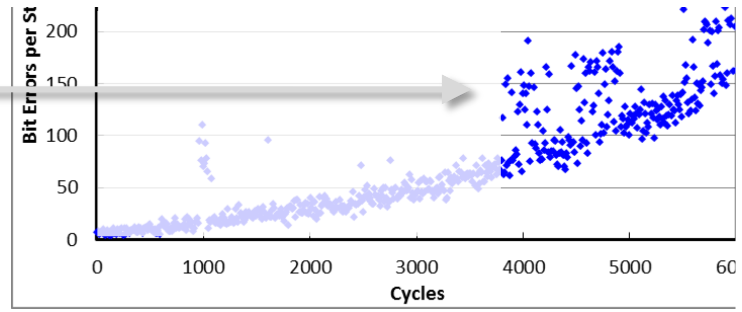
Actual SSD Data on Error Behavior

- Sector error count bitmap from an SSD
  - Bit errors have a significant tendency to cluster
  - So of course, do the NRREs
  - "If you see red, the sector is dead" (NRRE)



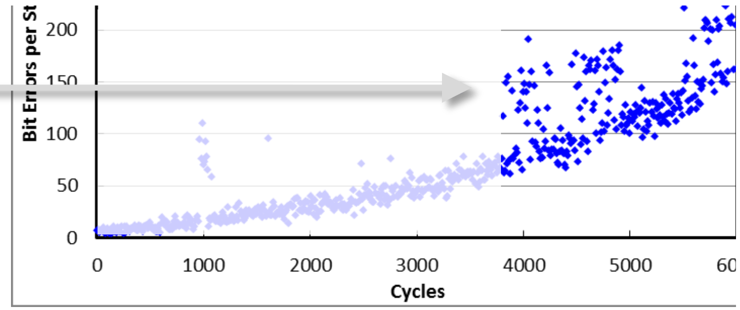
Error Clustering in Space and Time

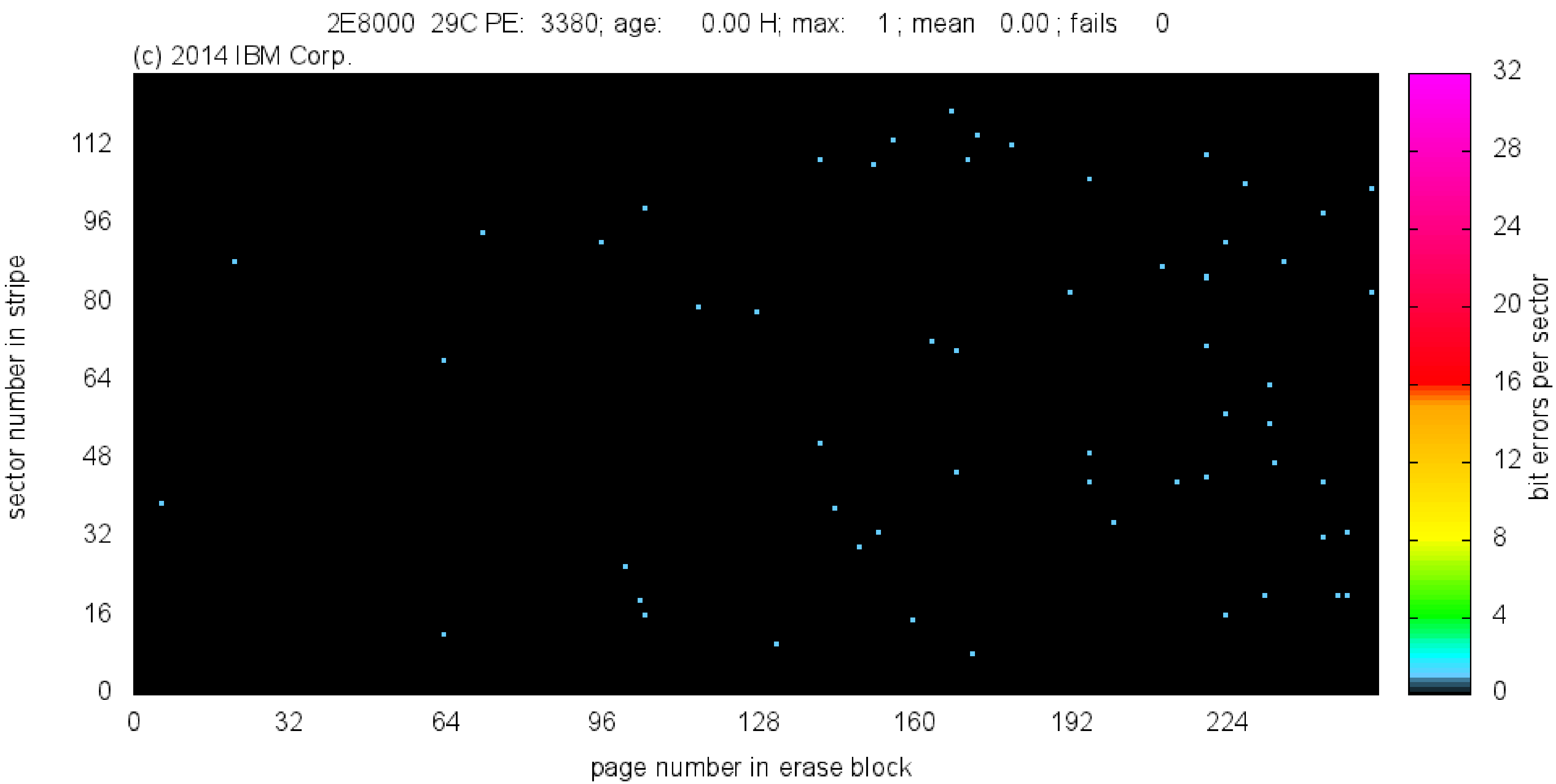
- Let's look at PE 4,000 – 6,000
  - Watch the evolution



Error Clustering in Space and Time

- Let's look at PE 4,000 – 6,000
  - Watch the evolution
  - Did you see the double?





Effects of Errors Being Non-random

- The net effect is that the reliability calculations we have performed will be too optimistic
- This means we should leave some headroom in the targets
- Increases the need for stronger erasure codes
- Increases the value of codes like PMDS
  - Don't want to pay a large penalty for the average sector, when outliers are the problem
  - Adding NRRE protection like PMDS efficiently targets the issue
  - The relative gains might be greater than shown here
  - However, need data on error behavior to confirm

Summary

- I have shown the importance of handling NRRE events
- I have shown how to set reliability targets for flash systems
- I have shown how to compute reliability for systems using various erasure codes
- PMDS codes are more efficient than classic N+M parity
  - PMDS codes are designed to protect against both unit loss and NRRE events
- DNR ECC can be combined with PMDS to codes for even greater efficiency