MEF UNIVERSITY
Your Freedom in Learning

MEF University Big Data Program (invited lecture)

Clustered Data Protection, Erasure Coding and Jerasure 2.0.

Outline

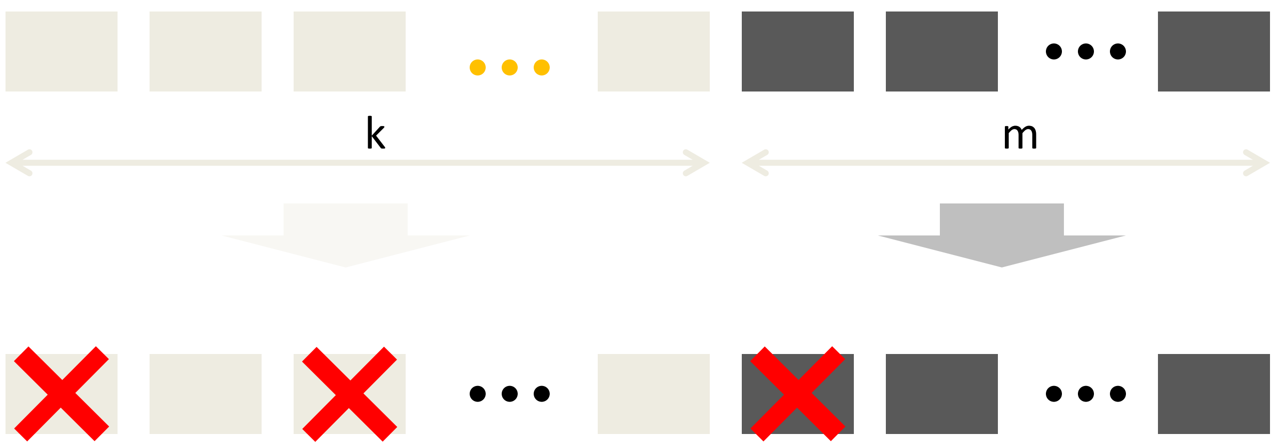
- What is Erasure coding?
- What is Jerasure?
- Current status
- Some contributions to Jerasure 2.0
- How to use it
- Performance Results
- Real life applications: Clustered Storage, OpenStack SWIFT, Ceph, etc…

MEF UNIVERSITY
Your Freedom in Learning

What's erasure coding?

- It is about creating redundancy to combat against losses.

Encoding: recipe for creating redundancy

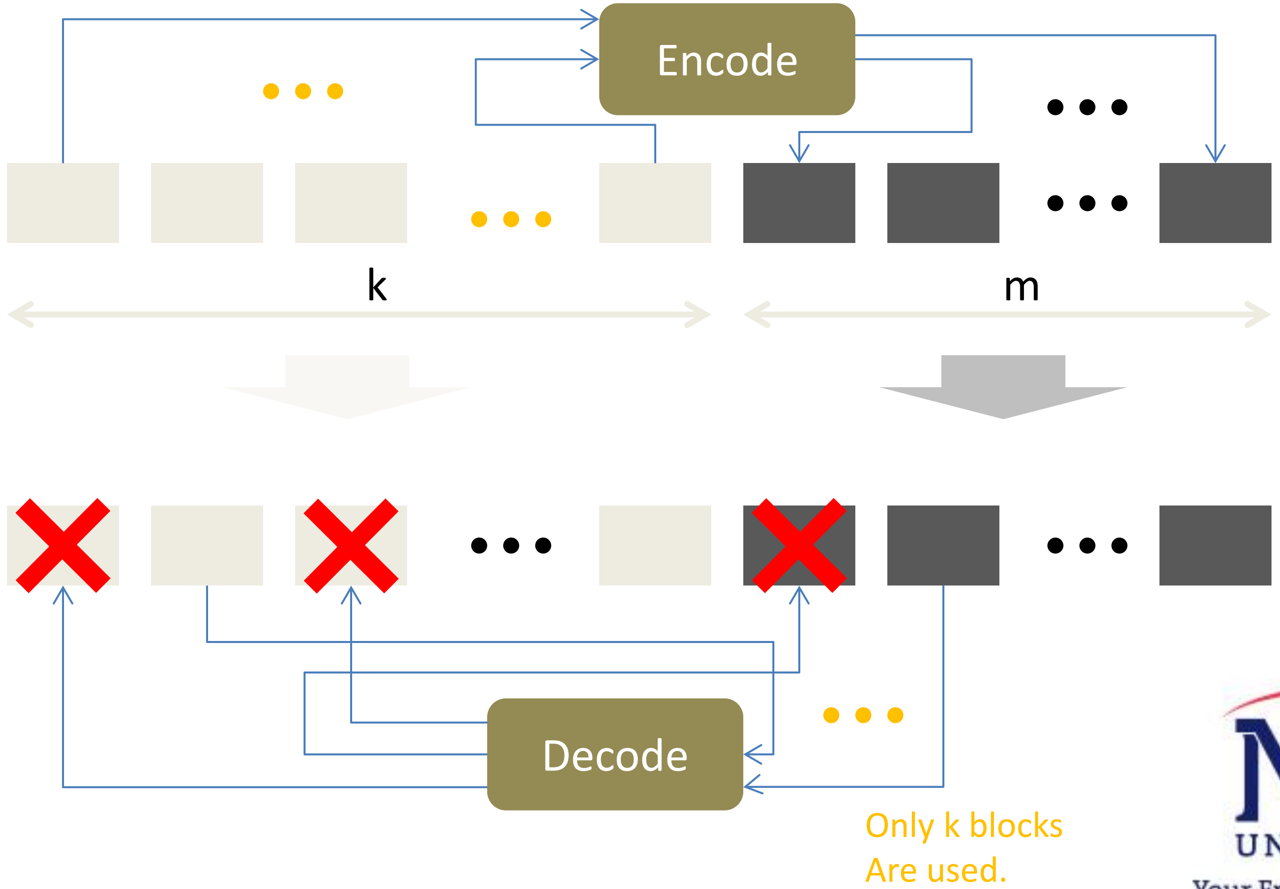


Decoding: recipe for recovering data

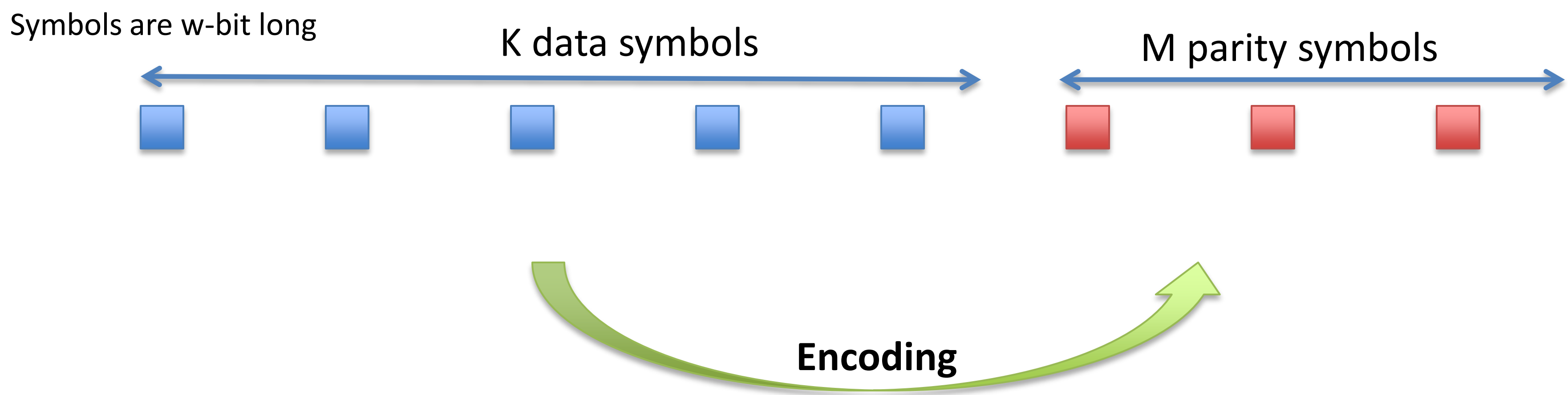
MEF UNIVERSITY
Your Freedom in Learning

What's erasure coding?

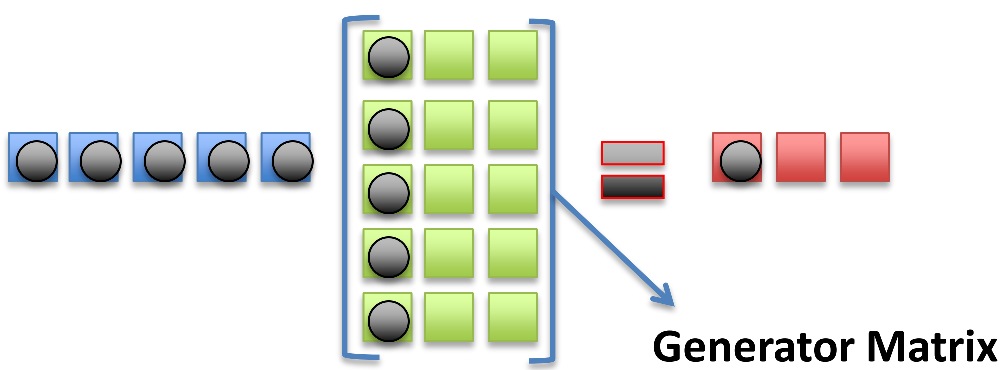
- It is about creating redundancy to combat against losses.



Erasure Coding



- Data is divided into K symbols.
- Encoder generates extra M parity symbols.
- Data is clear at the output: Systematic Coding.
- In storage systems, we encode data blocks!

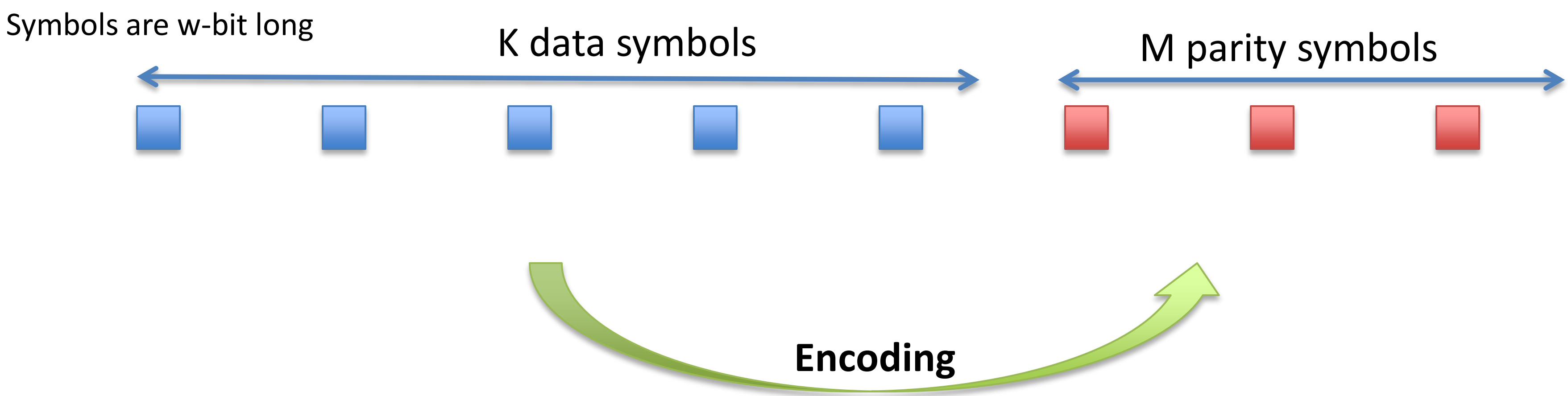


Use Galois Field arithmetic to calculate parities.

MULTIPLICATION & ADDITION

MEF UNIVERSITY
Your Freedom in Learning

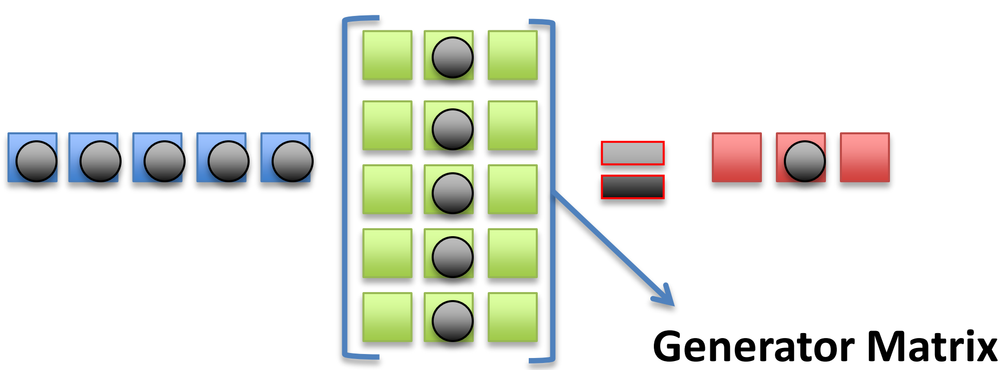
Erasure Coding



- Data is divided into K symbols.
- Encoder generates extra M parity symbols.
- Data is clear at the output: Systematic Coding.
- In storage systems, we encode data blocks!

Use Galois Field arithmetic to calculate parities.

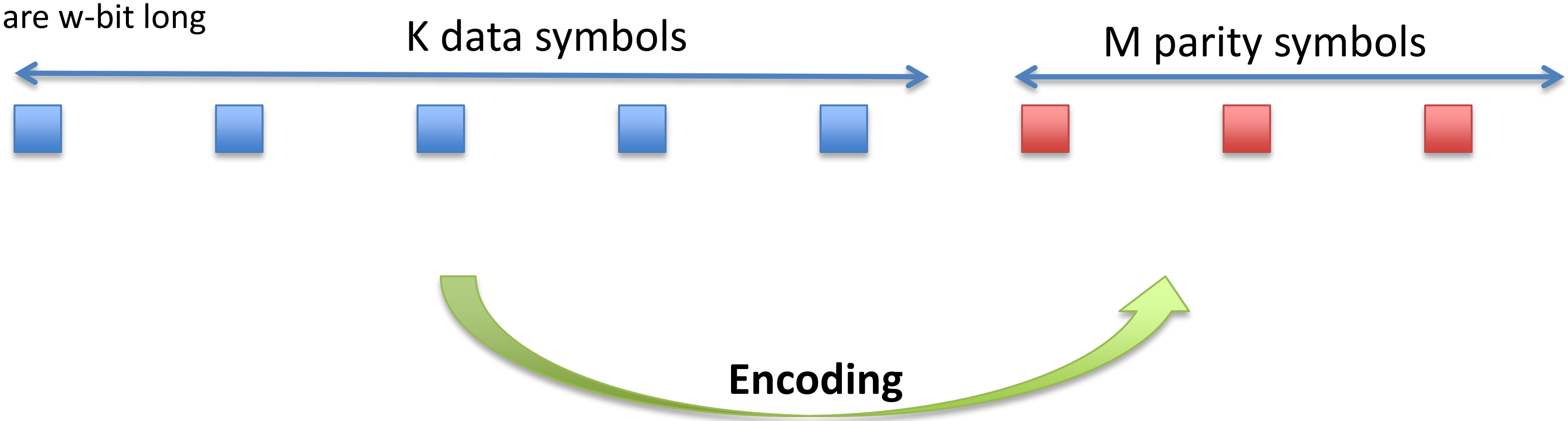
MULTIPLICATION & ADDITION



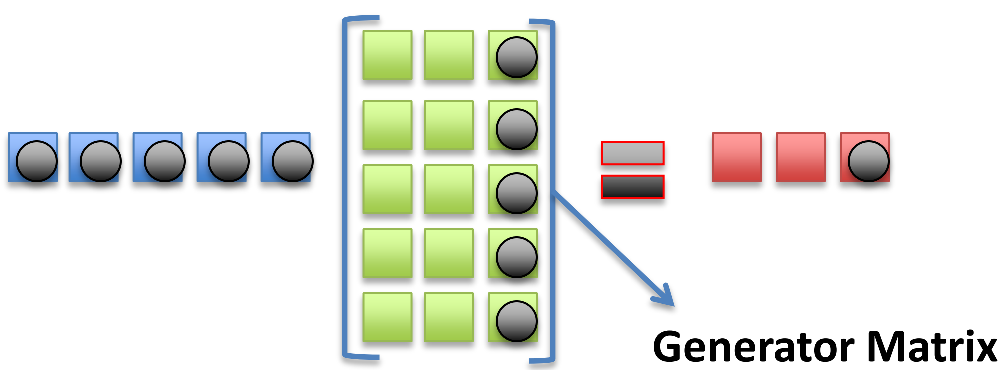
MEF UNIVERSITY
Your Freedom in Learning

Erasure Coding

Symbols are w-bit long



- Data is divided into K symbols.
- Encoder generates extra M parity symbols.
- Data is clear at the output: Systematic Coding.
- In storage systems, we encode data blocks!

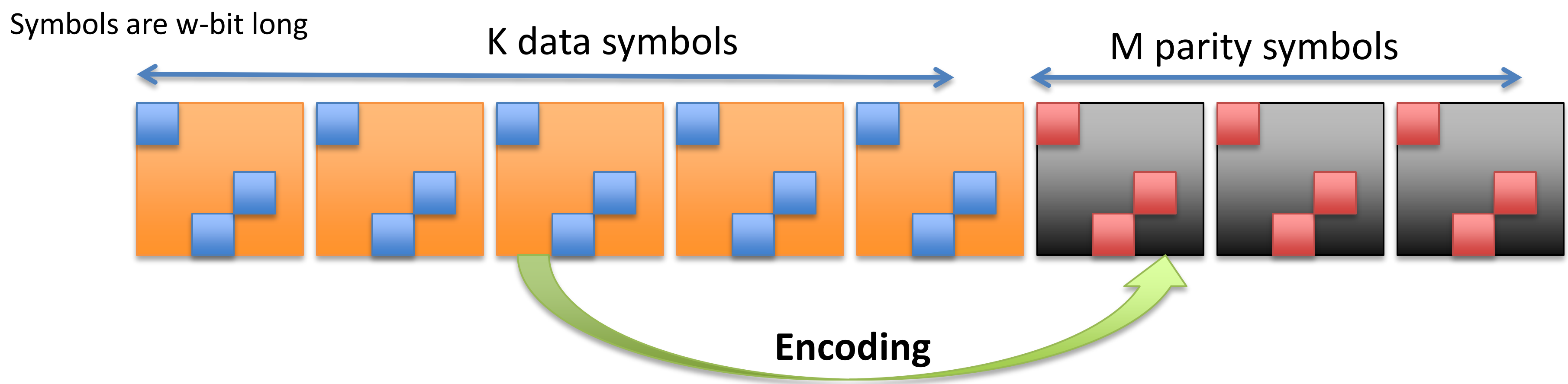


Use Galois Field arithmetic to calculate parities.

MULTIPLICATION & ADDITION

MEF UNIVERSITY
Your Freedom in Learning

Erasure Coding

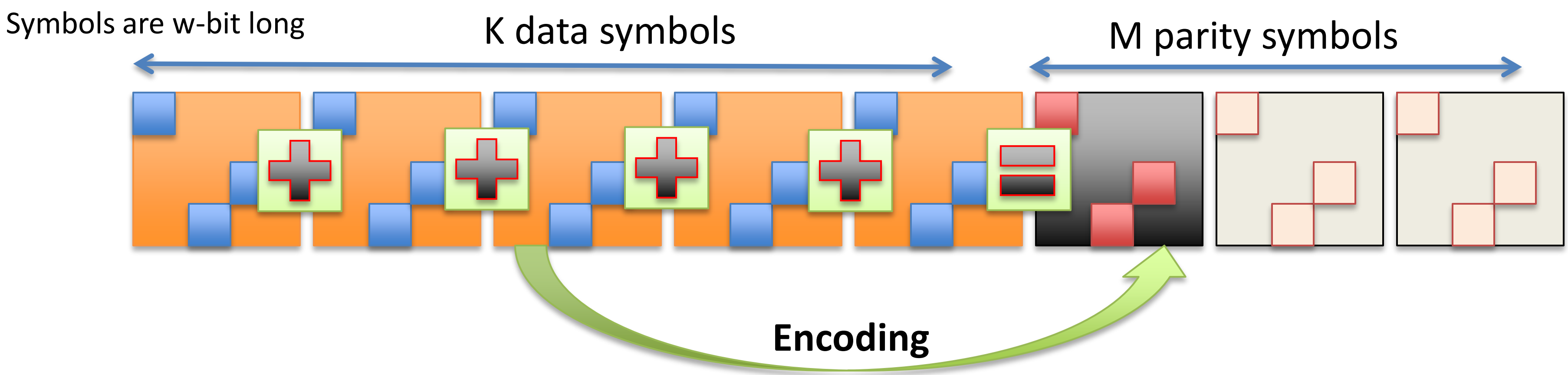


- Data is divided into K symbols.
- Encoder generates extra M parity symbols.
- Data is clear at the output: Systematic Coding.
- In storage systems, we encode data blocks!

<u>We XOR and multiply regions!</u>

MEF UNIVERSITY
Your Freedom in Learning

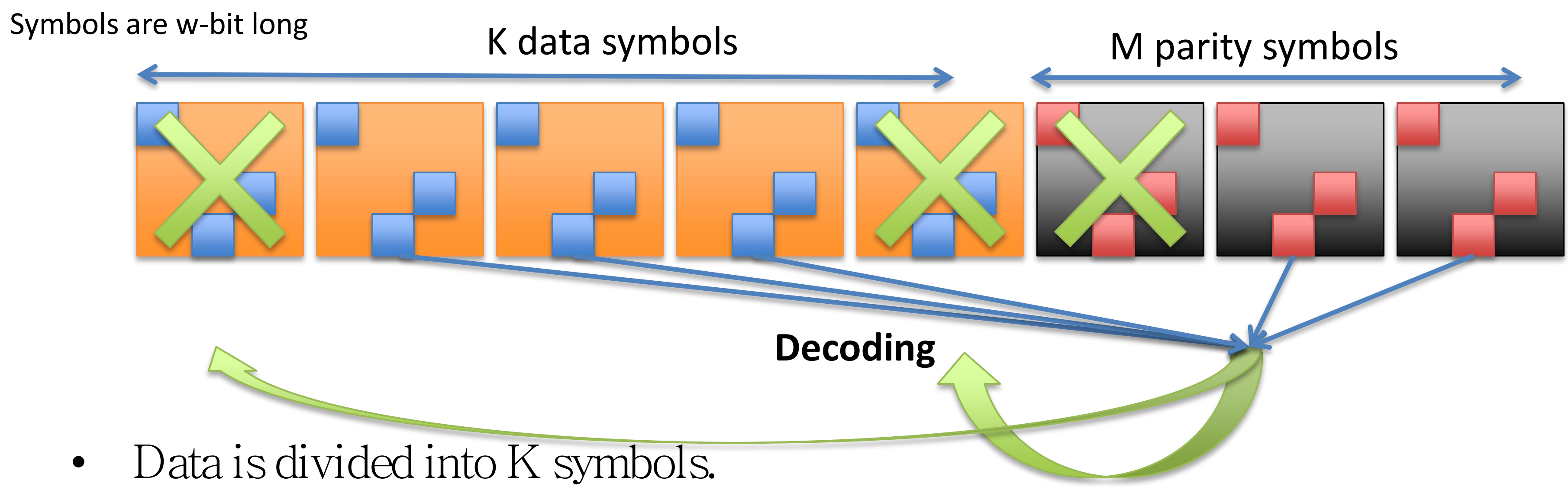
Erasure Coding



- Data is divided into K symbols.
- Encoder generates extra M parity symbols.
- Data is clear at the output: Systematic Coding.
- In storage systems, we encode data blocks!
- Use Galois Field arithmetic to calculate parities.
  - Use XOR to compute simple parities.
  - Use Galois multiplication and XOR the rest.
  - We XOR and multiply regions!
  - Generator Matrices tell us which data symbols take part in parity computations.

MEF UNIVERSITY
Your Freedom in Learning

Erasure De-Coding

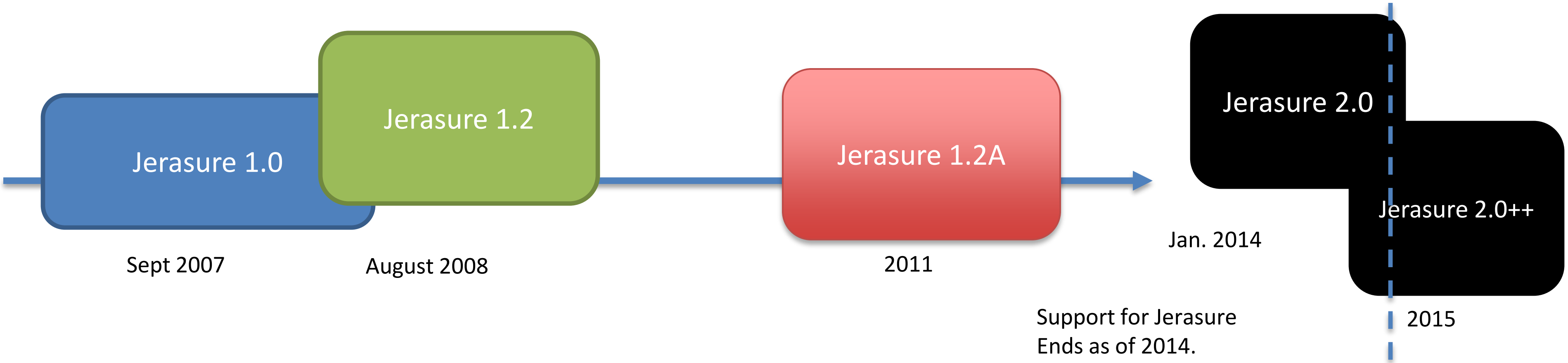


- Data is divided into K symbols.
- Encoder generates extra M parity symbols.
- Data is clear at the output: Systematic Coding.
- In storage systems, we encode data blocks!
- Use Galois Field arithmetic to calculate parities.
  - Use XOR to compute simple parities.
  - Use Galois multiplication and XOR the rest.
  - <u>We XOR and multiply regions!</u>
- Decoding: Any M failures can be tolerated!

MEF UNIVERSITY
Your Freedom in Learning

What is Jerasure?

- In 2007, Dr. Plank from Univ. of Tennesse has started a study of implementing high-performance erasure coding based on various published techniques.

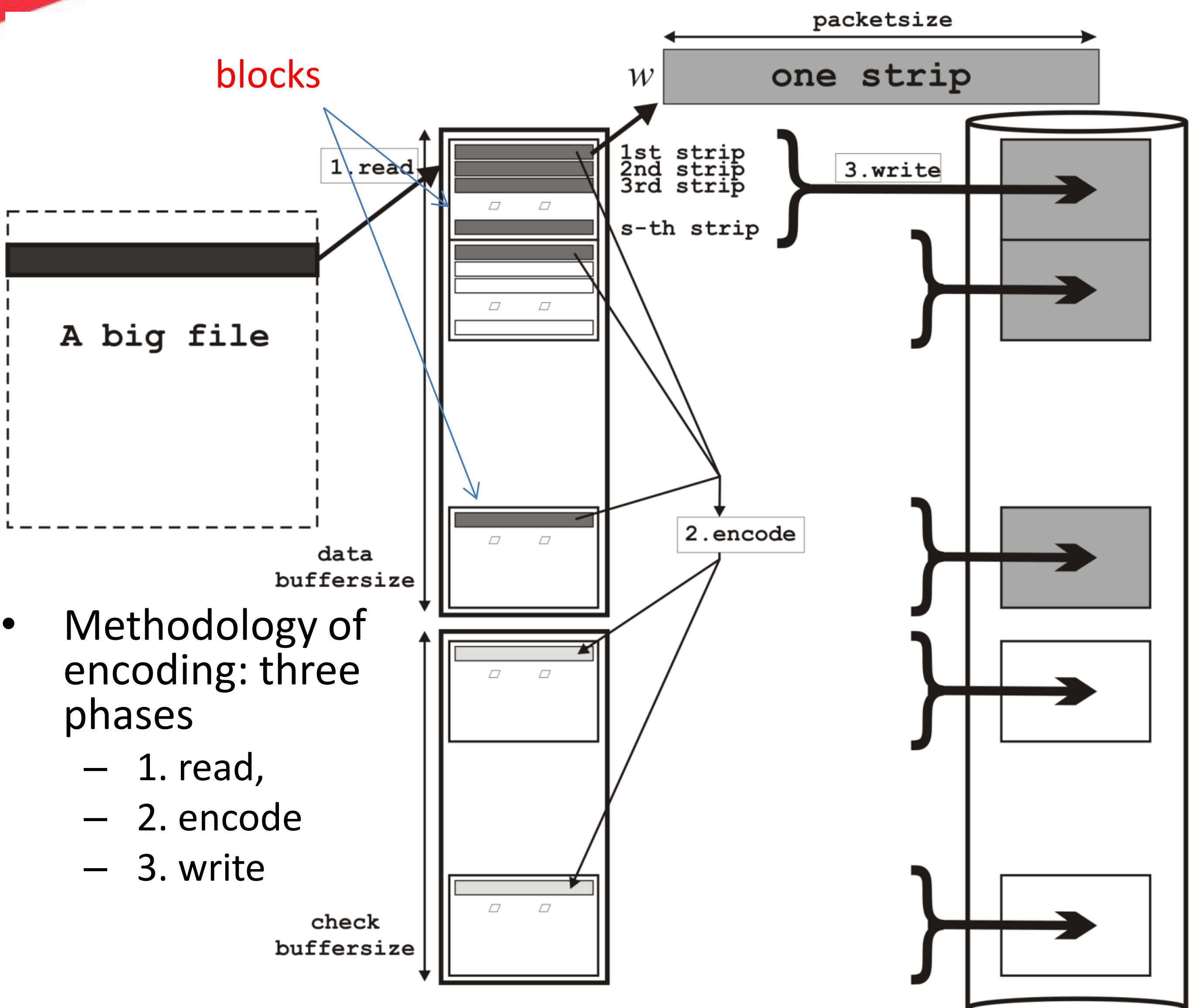


- Open Source Erasure Coding Library that found place in many open source projects.
- GF-Complete/Jerasure libraries are no longer supported by Dr. Plank and his team.
- Jerasure 2.0++ is a C++ implementation by the same group of people, probably at the beginning of 2015. The source code is not available but the document is:
  - http://jerasure2.googlecode.com/svn/trunk/jerasure3/documentation/paper.pdf

- Ceph and Open Stack Swift integration status:
  - Both Ceph and Swift has plug-ins for Jerasure 2.0.
- Kaminario (a company selling all-flash arrays) uses jerasure 1.2 in their software offerings (see references).
- Jerasure is based on a library that is capable of running fast Galois operations that goes with the name **GF-complete**.

MEF UNIVERSITY
Your Freedom in Learning

Review of Jerasure Encode Structure



- Methodology of encoding: three phases
  - 1. read,
  - 2. encode
  - 3. write

- Performance parameters subject to optimization:
  - Packetsize
  - Buffersize
  - K (# data blocks)
  - M (# parity blocks)
  - W (word size)

  ->Galois field parameter

  ***MSFT* uses w = 4**

  ***CLEVERSAFE* uses w = 8**

  ***EMC* uses w = 8**

  Packetsize is not random (usually dependent on the cache sizes)

  Small/Big buffersizes can change the number of I/O we make with the disk.

MEF UNIVERSITY

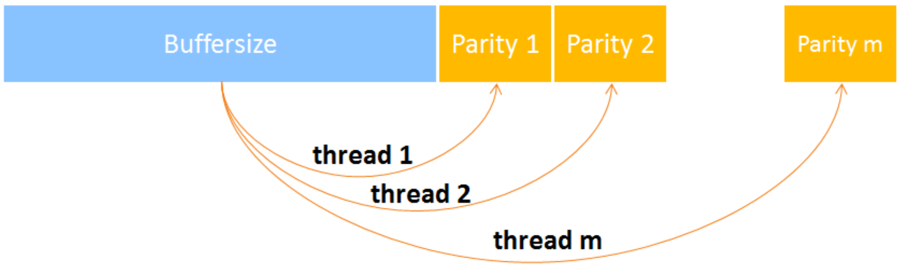
Your Freedom in Learning

Caveats by Jerasure's originators

- "Reducing cache misses is more important than reducing XOR operations."
  - So maintaining the existing memory hierarchy is a plus.
- "In any performance study, effects due to the memory hierarchy must be observed, and a final experiment demonstrates clearly that the encoding parameters should take account of the machine's cache size to achieve best performance."
  - So the optimized performance is quite dependent on the hardware.
- We will strictly follow these guidelines.

Two parts to multi-threaded implementation

- First approach (dropped it)
  - There are multiple reads of independent segments of the file
  - Each thread is responsible for encoding an individual segment.
  - This requires a different memory hierarchchy and architecture.
  - Memory is exclusively used for read/write.
    - Destroying the exisiting memory hierarchy led us to have reduced performance!
- Second approach (we adapted)
  - Each segment gets encoded in a loop to compute parity blocks.
  - Each parity block is computed by a different thread.
  - This requires no change to the memory hierarchy and architecture.
  - Memory is shared for read-only.
    - Multiple threads accessing the the same memory locations.

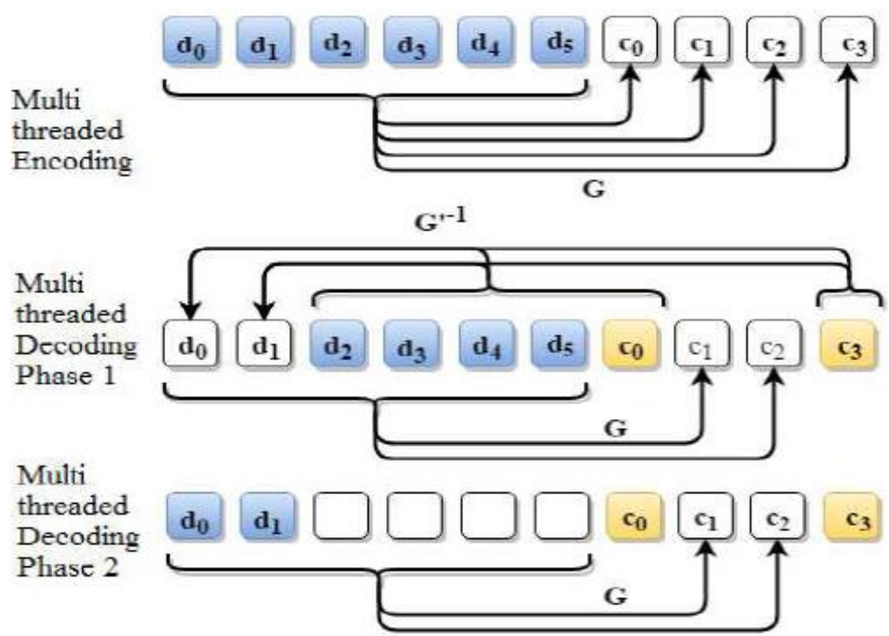


MEF UNIVERSITY
Your Freedom in Learning

Multi-threaded Implementations

- All three stages can be parallelized.
  - We focus on the pure encoding/decoding and report performance in terms of throughput.
- In order not to change memory architecture,
  - We let each thread to compute each parity block.
    - Data buffer is shared among the threads and read-only! No mutexes are used.

Parallel Regions



G: Generator Matrix

Two approaches

- POSIX THREADS (pthreads)
  - More flexible and higher level control over threads, manual management of threads: create, join, sync, etc.
  - Need to define extra data types and functions, manually create, join threads, overhead of thread creation.
- OpenMP
  - Shared memory standard, higher level control, task-based, easier to deploy, automatic management of threads.
  - Easy to create and join threads using pragmas. Avoid false sharing through copying shared variables.

OpenMP Examples:

```
/* Encoder main for loop */
#pragma omp parallel for shared(data, coding, blocksize, matrix, k) private(init, j,  sptr, dptr)
    num_threads(m) schedule(dynamic, 1)
/* Decoder main for loop */
#pragma omp parallel for shared(erased, data, coding, blocksize, decoding_matrix, k,  dm_ids) private(init,
     j, matrix_row, sptr, dptr) num_threads(k) schedule(dynamic, 1)
```

Unknown a priori

the number of parities to be generated (for the encoder)

the indexes /the number of lost blocks to be recovered (for the decoder)

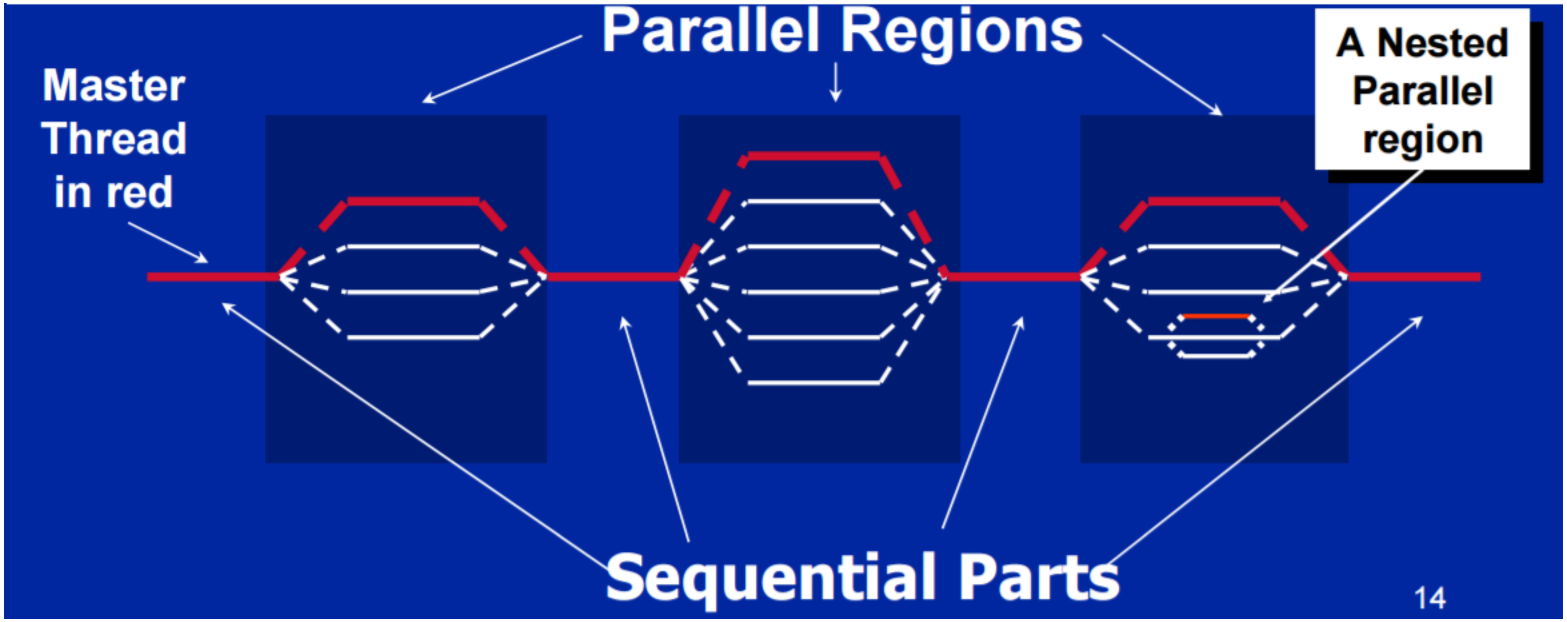
MEF UNIVERSITY
Your Freedom in Learning

Short on openMP standard

- OpenMP is a multi-threading, shared address model.
  - Threads communicate by sharing variables
- Constructs in OpenMP are compiler directives.
  - #pragma omp construct [clause [clause]…]
  - Example: #pragma omp parallel num_threads(4)
- Race condition: when the program's outcome changes as the threads are scheduled differently.
- Use synchronization to protect data conflicts (Expensive).
- Have to include: #include <omp.h>

MEF UNIVERSITY
Your Freedom in Learning

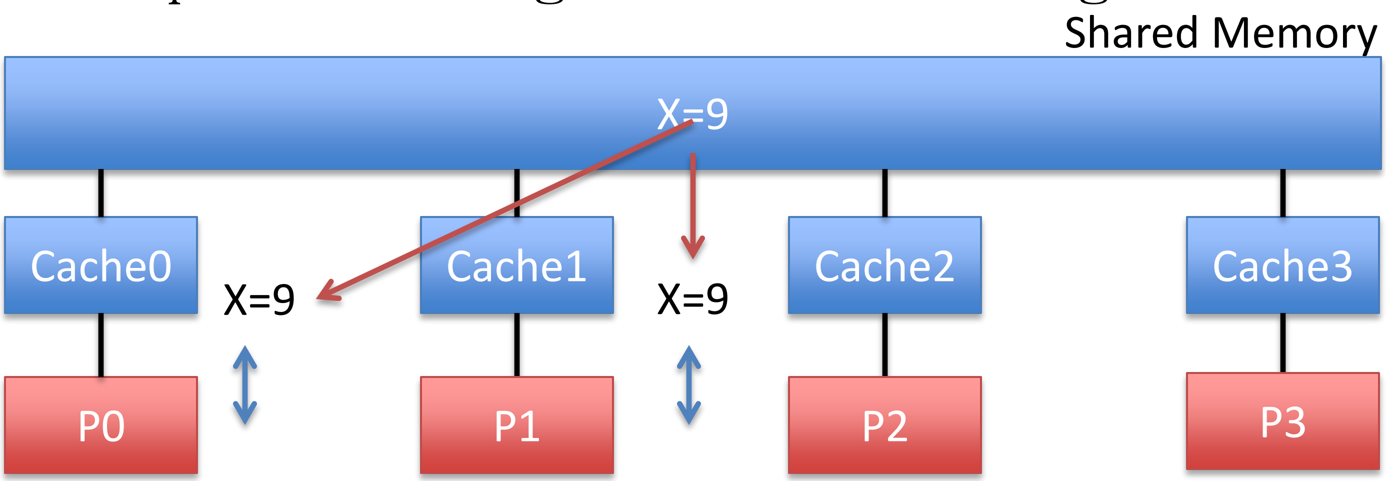
A Picture to consider…



- Master thread spawns a team of threads as needed
- Parallelism added incrementally until performance goals are met: i.e. the sequential program evolves into a parallel program.
- Mutual Exclusion: Only one thread at a time can enter a critical region

Some more details…

- The schedule clause affects how loop iterations are mapped onto threads.
- One of the performance degradation: False Sharing.



- False sharing degrades performance when all of the following conditions occur.
  - Shared data is modified by multiple processors.
  - Multiple processors update data within the same cache line.
  - This updating occurs very frequently (for example, in a tight loop).
- Note that shared data that is **read-only** in a loop does not lead to false sharing.
- Solution: Use private copies at the caches.

MEF UNIVERSITY
Your Freedom in Learning

Contributions

- Encoder and Decoder are re-written.
  - *There are new functions and struct definitions that could be placed in other jerasure files, but for the sake of keeping the rest intact, we included all of them in the encoder and decoder files.*
- Encoder's parity generation loop is fully multi-threaded.
  - *With all memory allocations are managed and no mutexes are used.*
  - *Core functions of GF-complete (which heavily use SIMD instructions) are called by all the threads explicitly.*
- Decoding structure of Jerasure is strictly serial:
  - Assume k'<k data blocks, m'<m parity blocks are lost.
  - First k'-1 data blocks are recovered, then using XOR (low complexity) to recover the remaining data block. Finally, lost parities are recovered by re-encoding.
- Decoding of multi-threaded version has minimum serial parts.
  - *Recovers k' data blocks all at the same time, each recovery is accomplished by a different thread. Also, assuming the lost data blocks are Zero, m' parity blocks are partially recovered by multiple threads. (Total # of threads = k' + m')*
  - *Available data is shared amoung threads (potentially across caches)*
  - *Using only the restored data, m' parity blocks are all re-encoded at the same time. The result of the first step is combined with the result of this step (using simple XOR) to gerenate the full recovery of parities (again using multiple threads).*
- And ….other optimizations in looping, elimination of some of the conditional statements etc.

MEF UNIVERSITY
Your Freedom in Learning

Performance Results

- We run jerasure and our own version on different systems.
  - To generate results that can be compared and verified, we picked two laptops that are close in configuration to that of Dr. Plank's to generate similar results.
  - To generate real-use-case results, we have also used high-end CPUs.
- We compared pure encode/decode speeds to be able to avoid the Disk/RAM I/O time.
- There are four testing combinations:
  - Single process single thread
  - Single process multiple thread
  - Multiple process single thread (each process is single threaded)
  - Multiple process multiple thread (each process is multi-threaded)
- Python is a great tool to simulate multiprocessing environments!

MEF UNIVERSITY
Your Freedom in Learning

Test System description

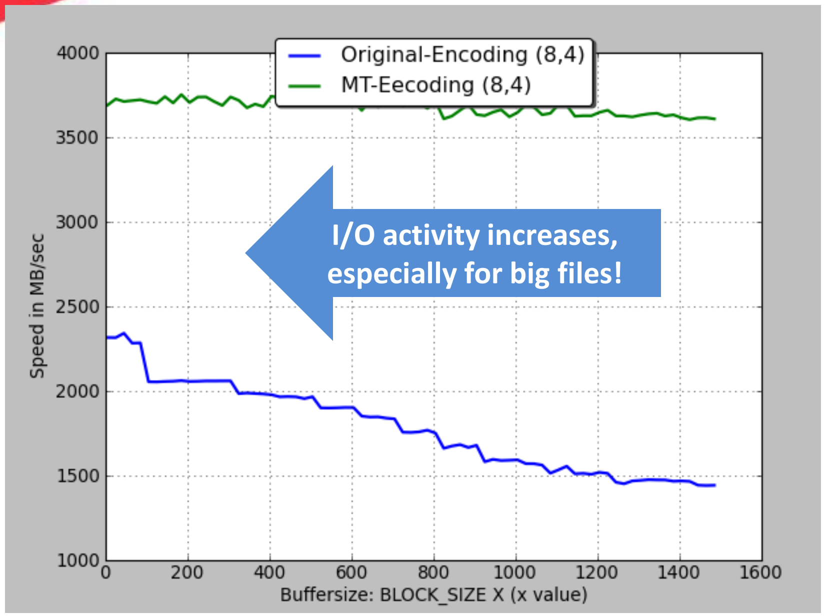
- Jerasure "reed_sol_van" using
  - File: 256MiB binary file.
  - K=8, M=4, W=8, packetsize = 2000/6000bytes. Buffersize is variable.
  - Since M=4, we use four threads one for each parity computation.
- High-end System info:

Intel Xeon CPU E5620 2.40GHz

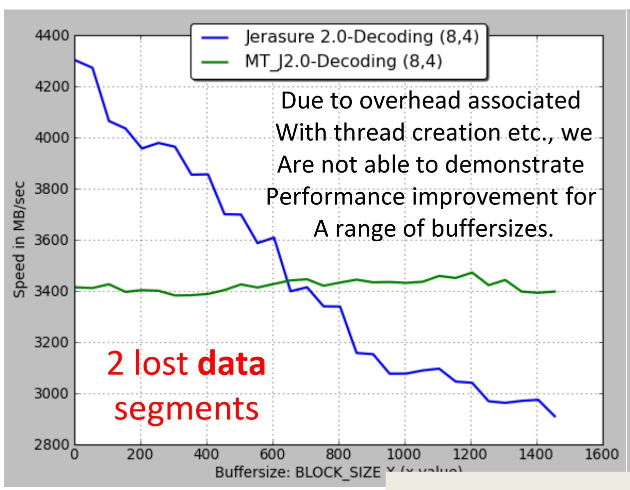
OS	Centos 6
Sockets	2
CPU Cores	8 (4 for each socket)
CPU Threads	16
Arc	64-bit
CPU speed	1.6GHz
L1i/d Cache	32KB
L2 Cache	256K
L3 Cache	12288K
Memory	~49GB
SIMD enabled	yes

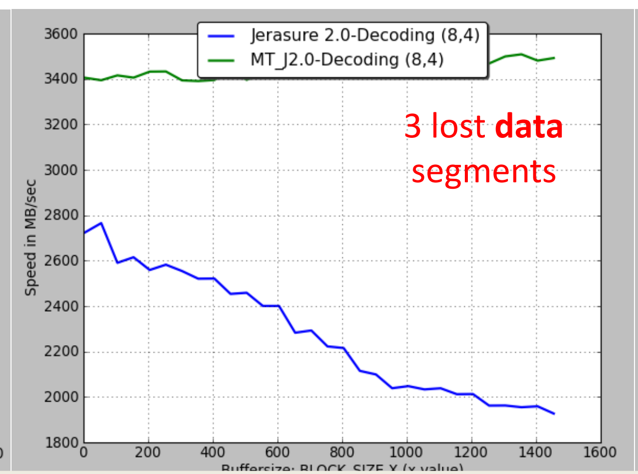
MEF UNIVERSITY
Your Freedom in Learning

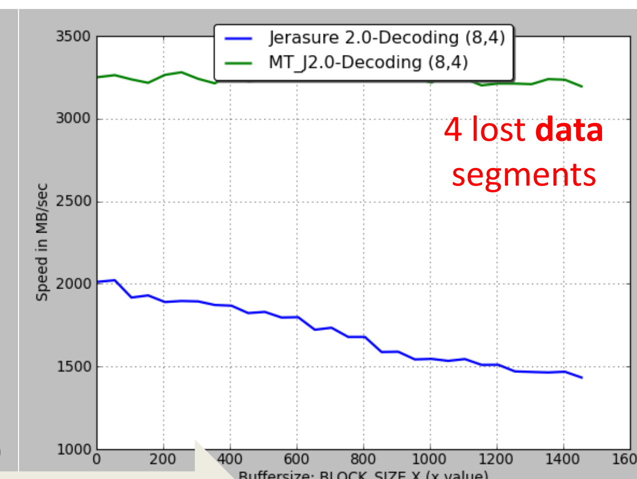
Policy: (K=8,M=4)



- Pure encoding/decoding performance as a function of buffersize.
- Optimized packetsize (2K-20K)
  - Based on the cache structure.
- Encoding/Decoding performance is pretty stable for all buffersizes.
- Multi-threaded implementation is less effected by the cache sizes.
- **BLOCK_SIZE = 10KB.** (sim. Parameter)



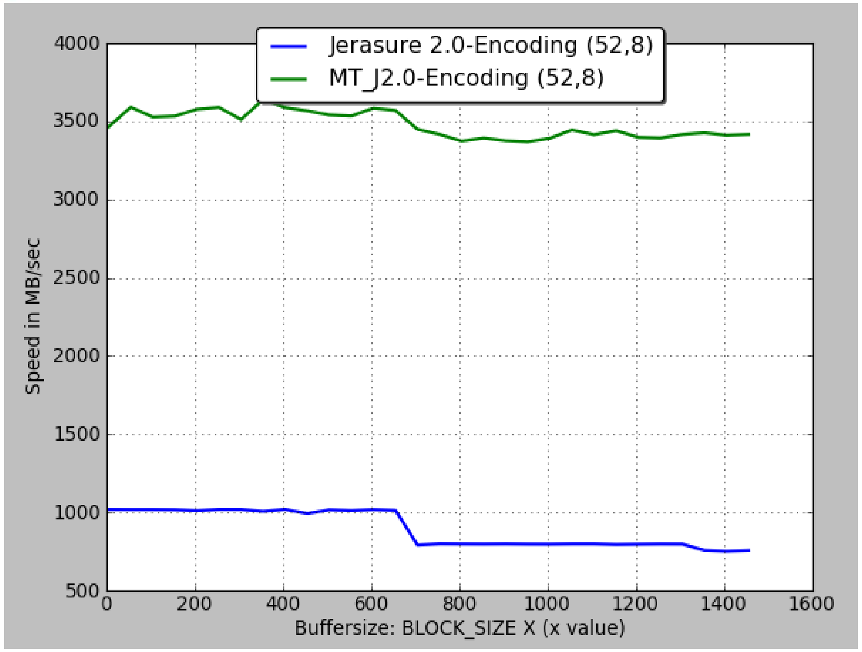


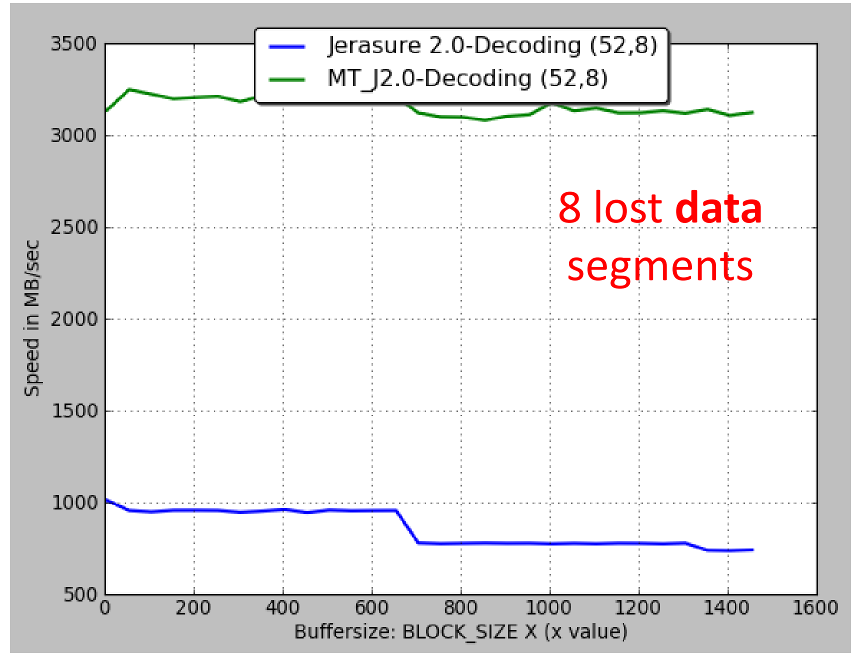


More potential for multithreading

Your Freedom in Learning

Policy: (K=52,M=8)

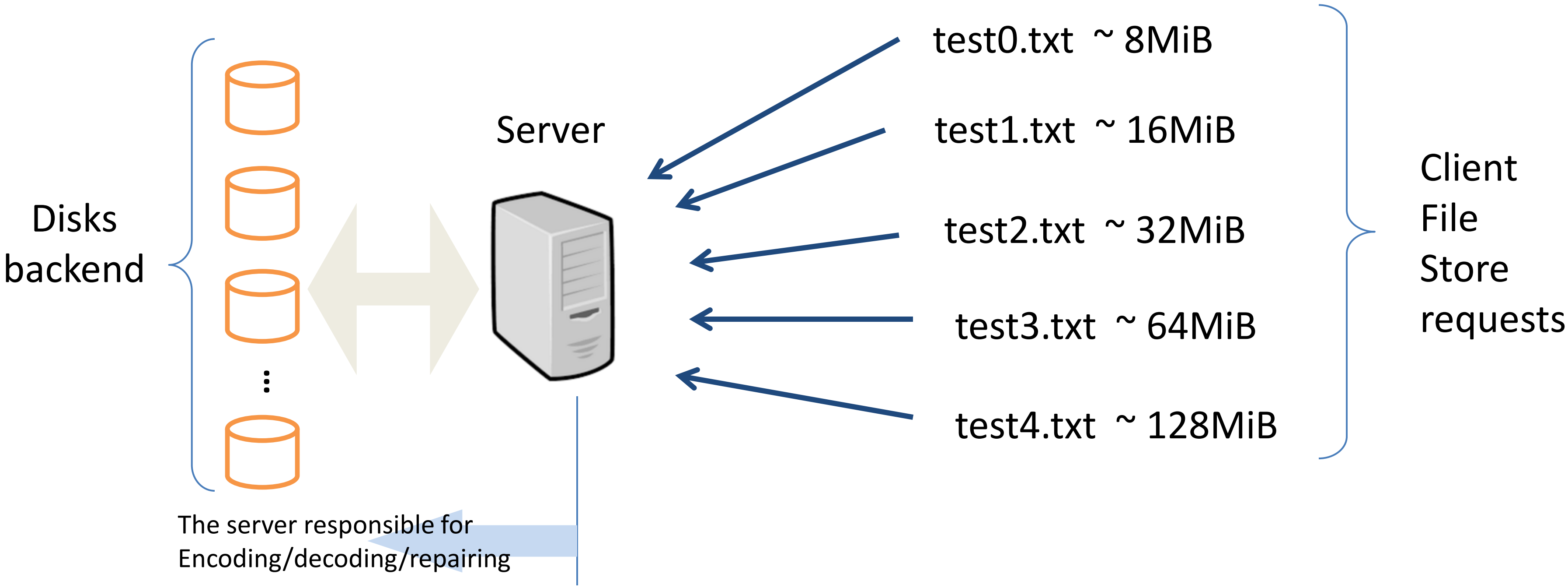




- Jerasure's performance looks more stable.
- Same trends can be observed with respect to multi-threaded version.
- More than 3x improvement for both the encoder and decoder.

MEF UNIVERSITY
Your Freedom in Learning

A multiprocess scenario (test case)



- A compute node serves 5 file-store requests.
- For now, we only focused on the encoder.

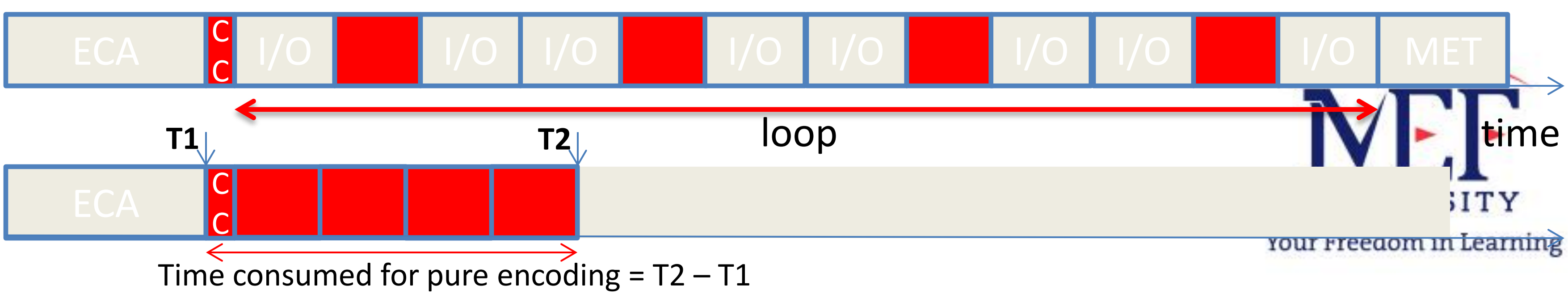
MEF UNIVERSITY
Your Freedom in Learning

Recap: Jerasure and pure computation time

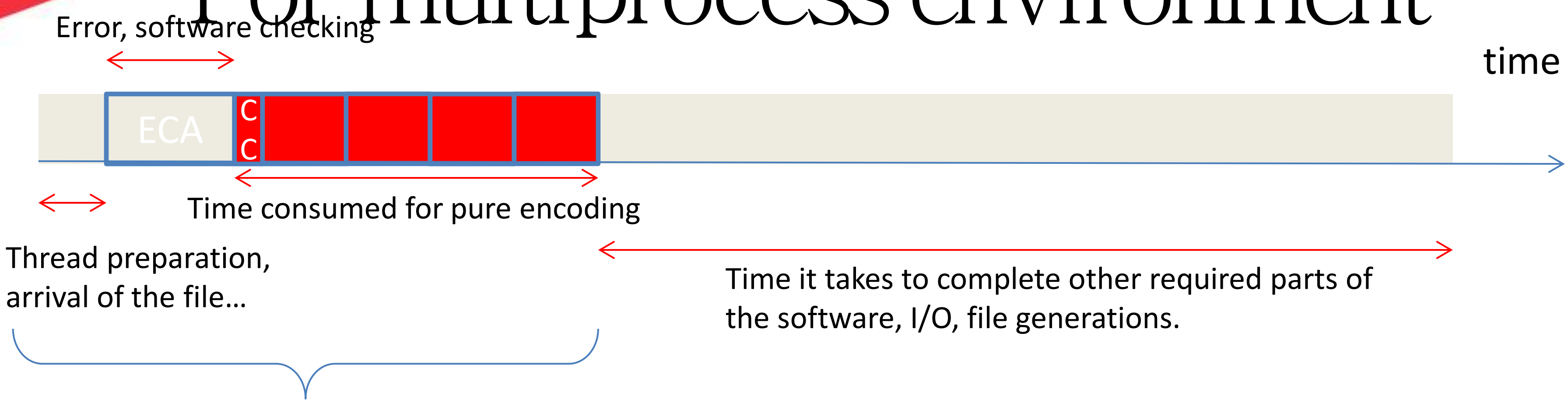
- Main engine for encoding is encoder.c
  - Variable definitions
  - Error check arguments (ECA)
    - Packetsize, buffersize adjustments.
    - Arguments/Parameters validy check.
    - File size adjustments
  - Create coding/bit/schedule matrices (CC)
  - Loop starts (L)
    - File read, zero padding, pointer set (I/O)
    - Encoding
    - Write the data/parities to files (I/O)
  - Create/write metadata (MET)

Encoding Speed (MB/sec)

Total Speed (MB/sec)

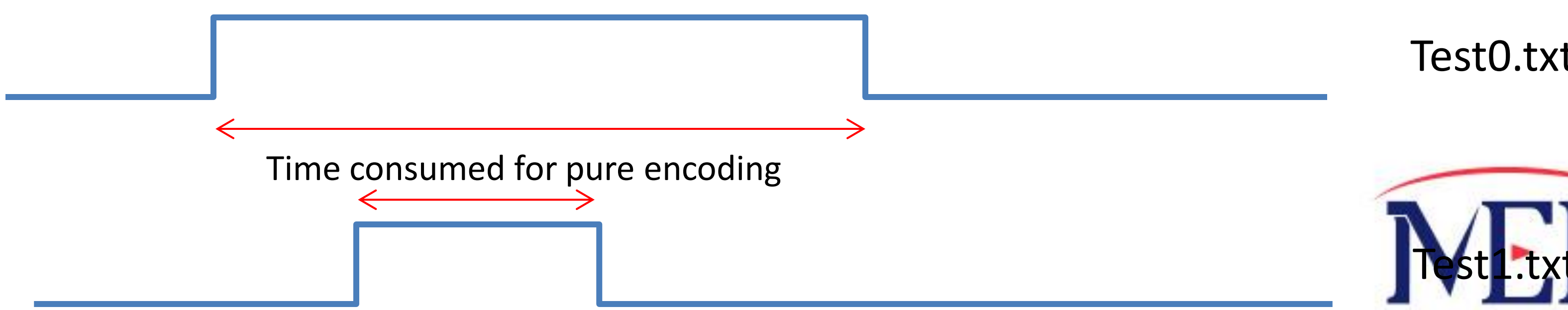


For multiprocess environment

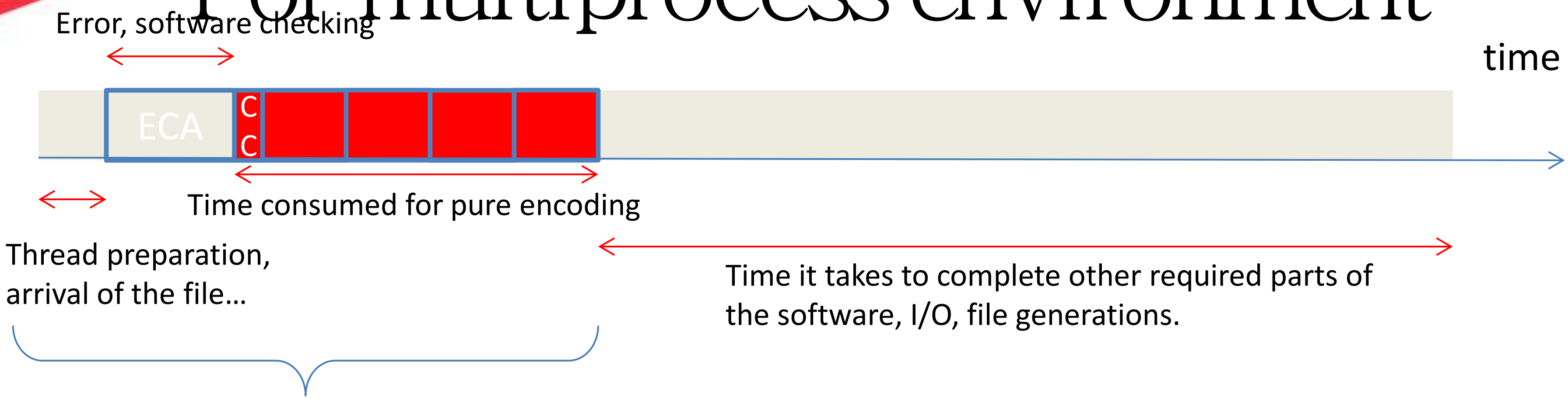


All random variables that should be parameterized by the file contents, size etc.
HARD to MODEL!

A way to illustrate what's going on is to use staircase plots

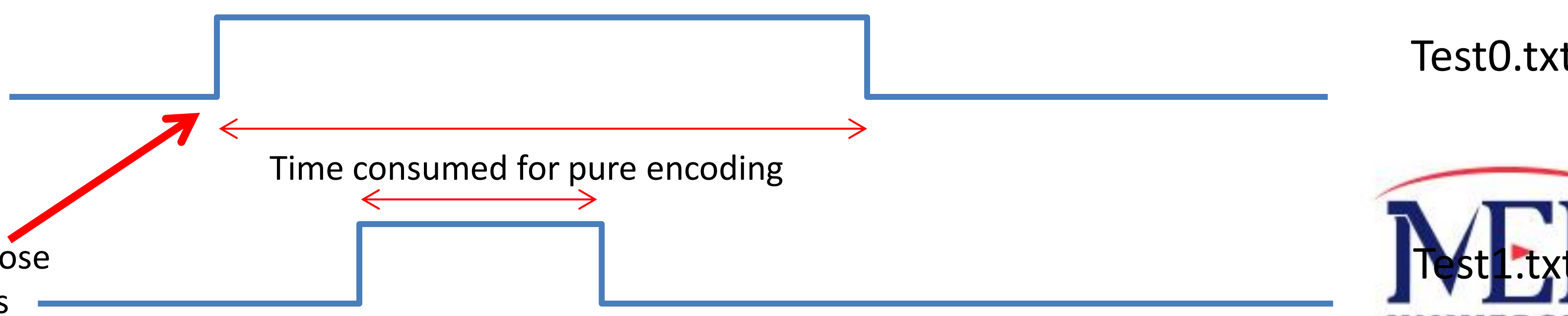


For multiprocess environment



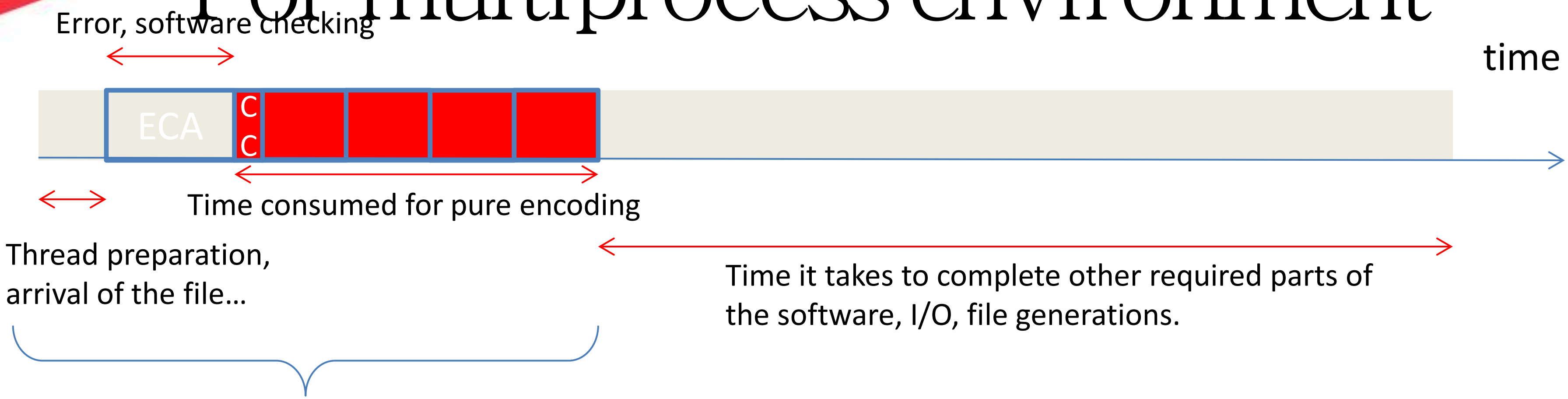
All random variables that should be
Parameterized by the file contents, size etc.
HARD to MODEL!

A way to illustrate what's going on is to use staircase plots



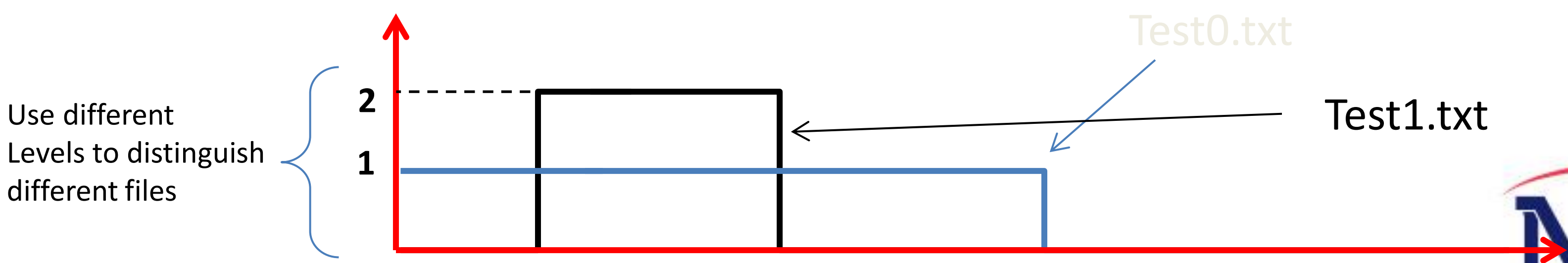
Suppose
this is
the origin
where the computation starts.

For multiprocess environment



All random variables that should be
Parameterized by the file contents, size etc.
HARD to MODEL!

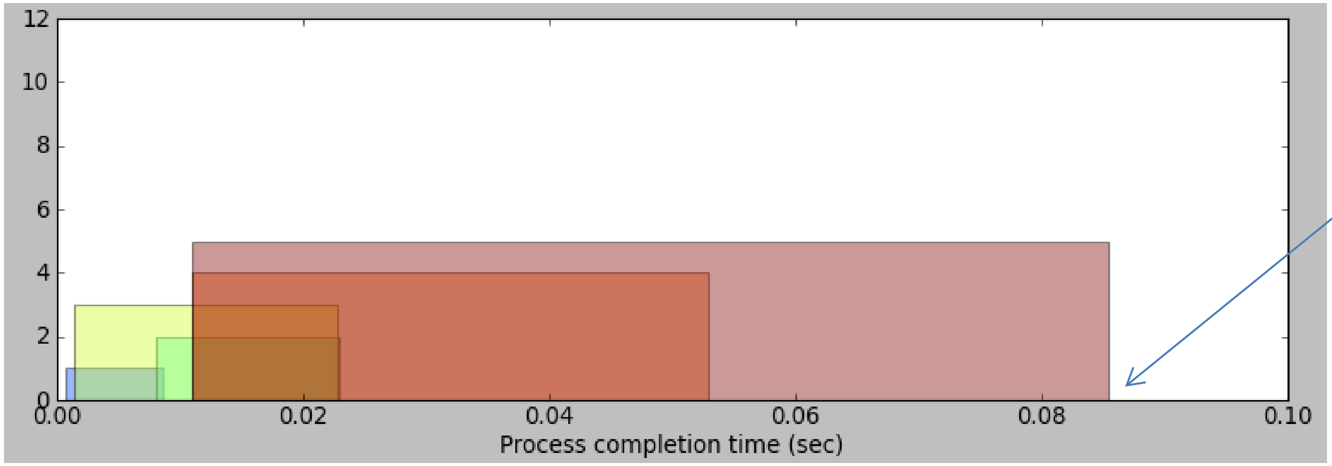
A way to illustrate what's going on is to use staircase plots



Following results are average of 50 independent runs.

(8,4) psize = 2000, bsize=5M

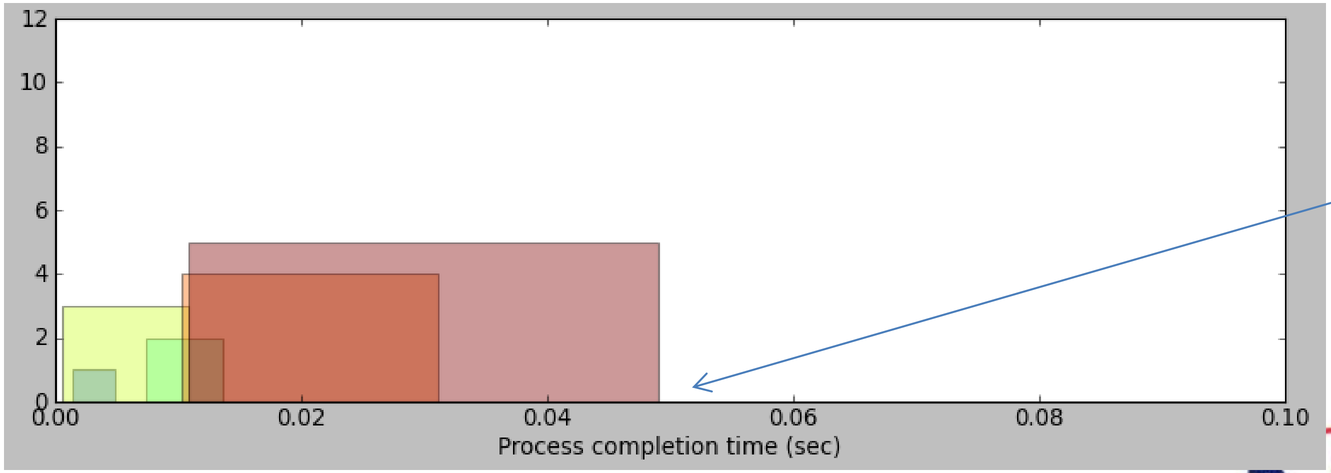
encoder.c



0.08665 sec

aggragate Throughput of pure encoding = **2862MB/sec**

encoderMT2.c



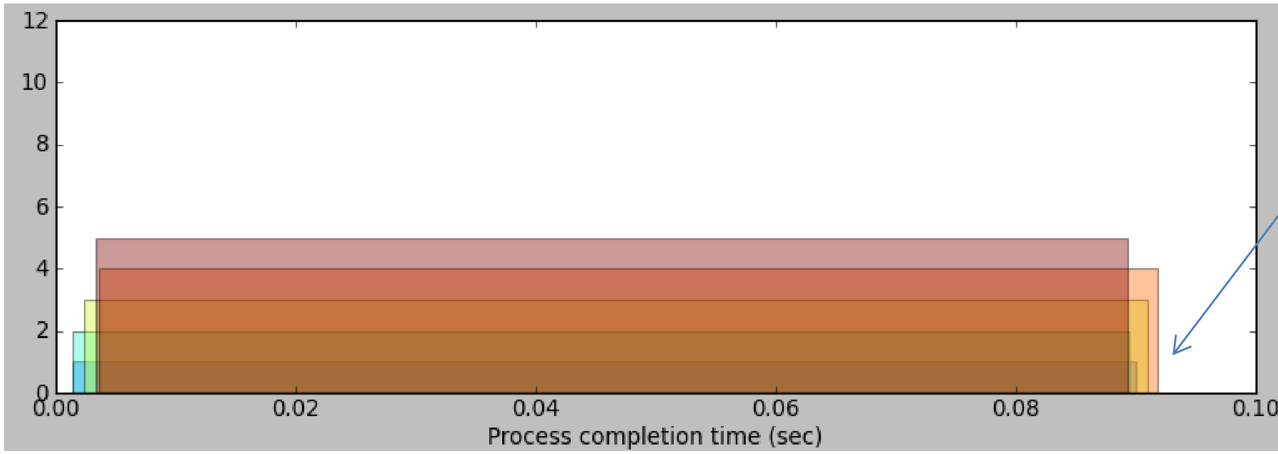
0.04907 sec

aggragate Throughput of pure encoding = **5054MB/sec**

MEF UNIVERSITY
Your Freedom in Learning

(8,4) →If files are the same size ~ 128MiB

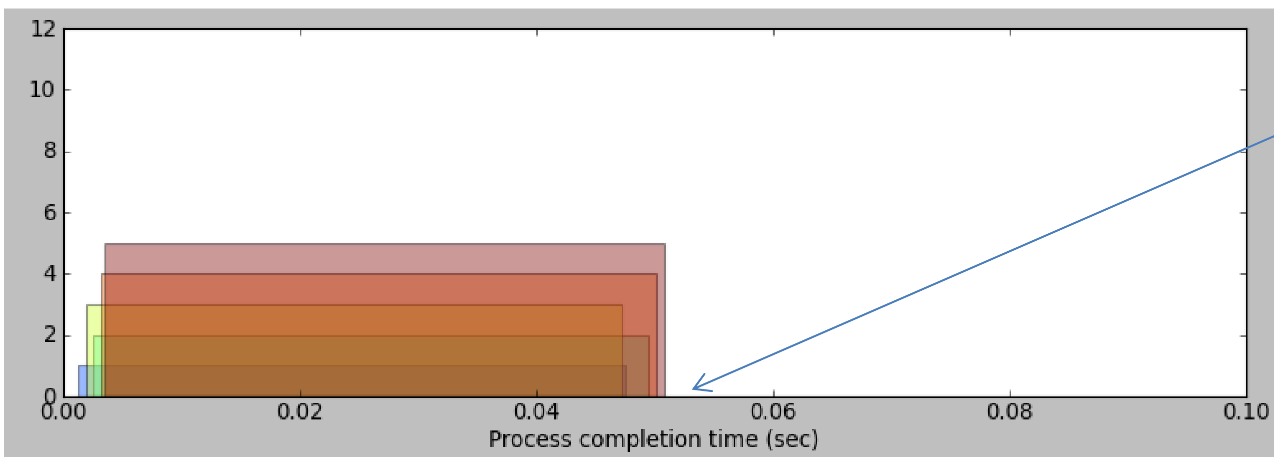
encoder.c



0.091766

aggregate Throughput of pure encoding = **6974MB/sec**

encoderMT2.c



0.05080

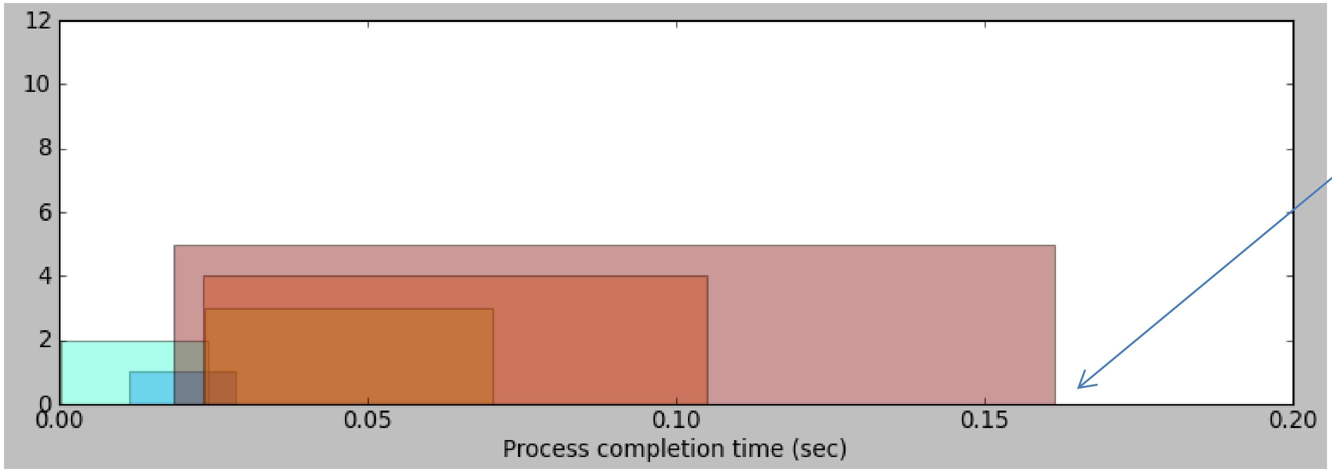
aggregate Throughput of pure encoding = **12598MB/sec**

Run with 52/8 the same simulation??

MEF UNIVERSITY
Your Freedom in Learning

(k=52,m=8) packetsize = 2000, buffersize=5M

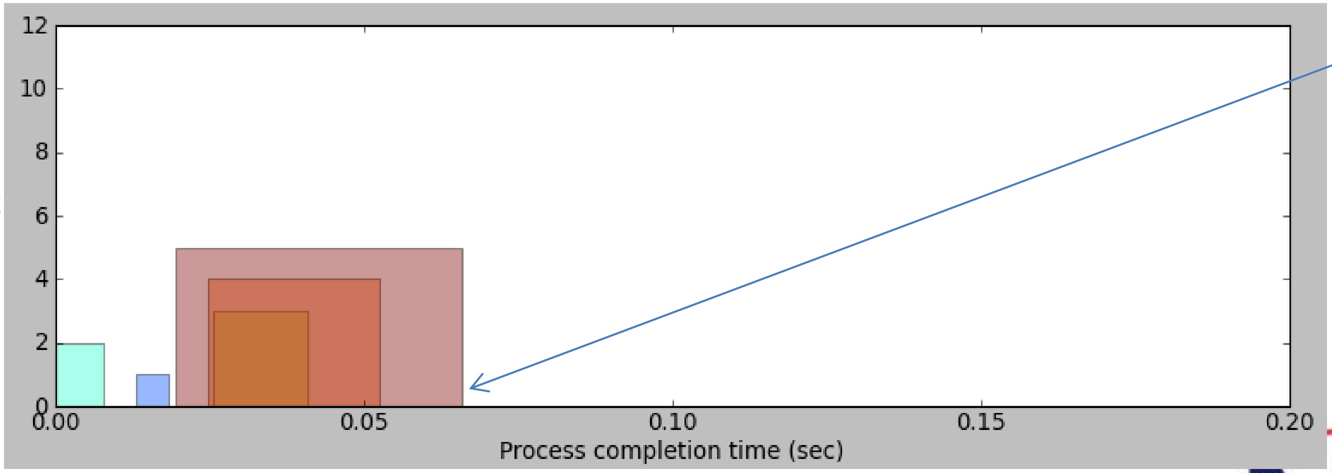
encoder.c



0.1613 sec

aggregate Throughput of pure encoding = **1537MB/sec**

encoderMT2.c



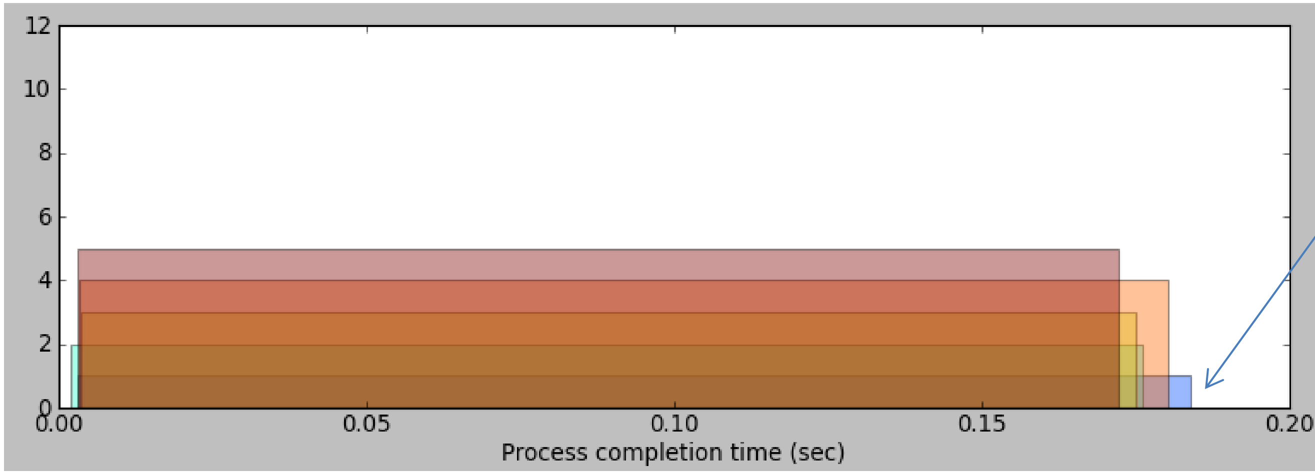
0.0657 sec

aggregate Throughput of pure encoding = **3775MB/sec**

MEF UNIVERSITY
Your Freedom in Learning

(k=52,m=8) →If files are the same size ~ 128MiB

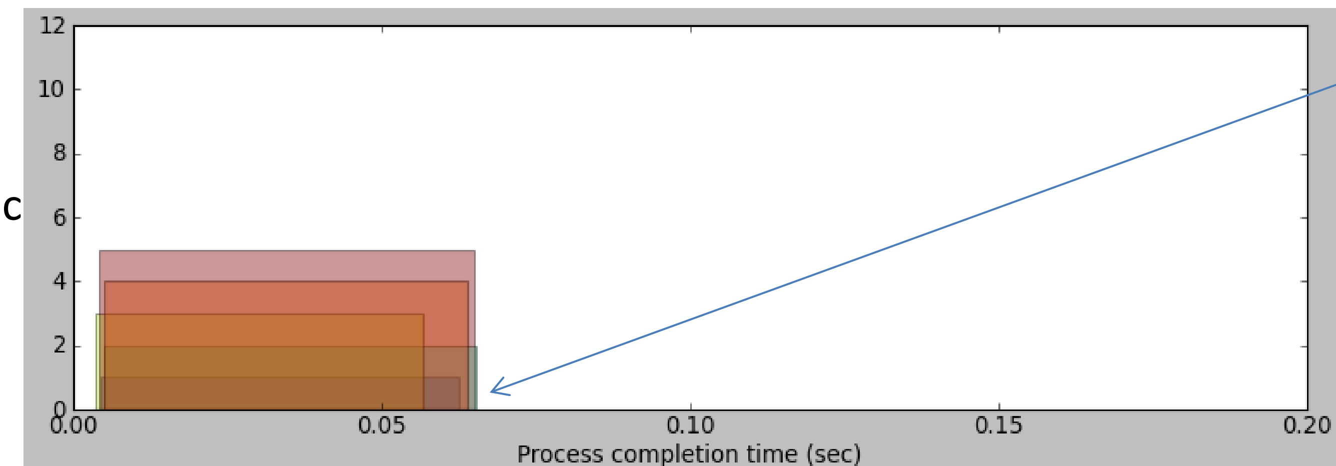
encoder.c



0.184 sec

aggregate Throughput of pure encoding = **3478MB/sec**

encoderMT2.c



0.0626 sec

aggregate Throughput of pure encoding = **10224MB/sec**

MEF UNIVERSITY

Your Freedom in Learning

Summary

- Multithreaded version maintains the same performance across different buffersizes.
  - Good for big files.
  - Buffersize optimization seem not to be relevant.
- On average, more than 3x pure encode/decode speed gain.
- In a multi-process environment, while the size and the number of files are changing…
  - The performance of the multi-threaded version shows the least change.
- Compared to original encoder/decoder, multi-threaded version has more potential for improvement and room for optimizations.

MEF UNIVERSITY
Your Freedom in Learning

References

- James Plank, "**Fast Galois Field Arithmetic Library in C/C++"** Technical Report UT-CS-07-593, U of Tenessee, 2014.
- Kaminario open source project: http://kaminario.com/resources/files/Kaminario_Open_Source_DOC1200021_00.pdf
- Ceph's jerasure testing: http://dachary.org/?p=3665
- J. S. Plank, J. Luo, C. D. Schuman, L. Xu, and Z. W. O'Hearn. A performance evaluation and examination of open-source erasure coding libraries for storage. In 7th USENIX FAST, pages 253–265, 2009.
- Erasure codes and swift: http://www.snia.org/sites/default/files/Luse_Kevin_SNIATutorialSwift_Object_Storage2014_final.pdf

MEF UNIVERSITY
Your Freedom in Learning

Jerasure 2.0

COMPLETE SET-UP GUIDE

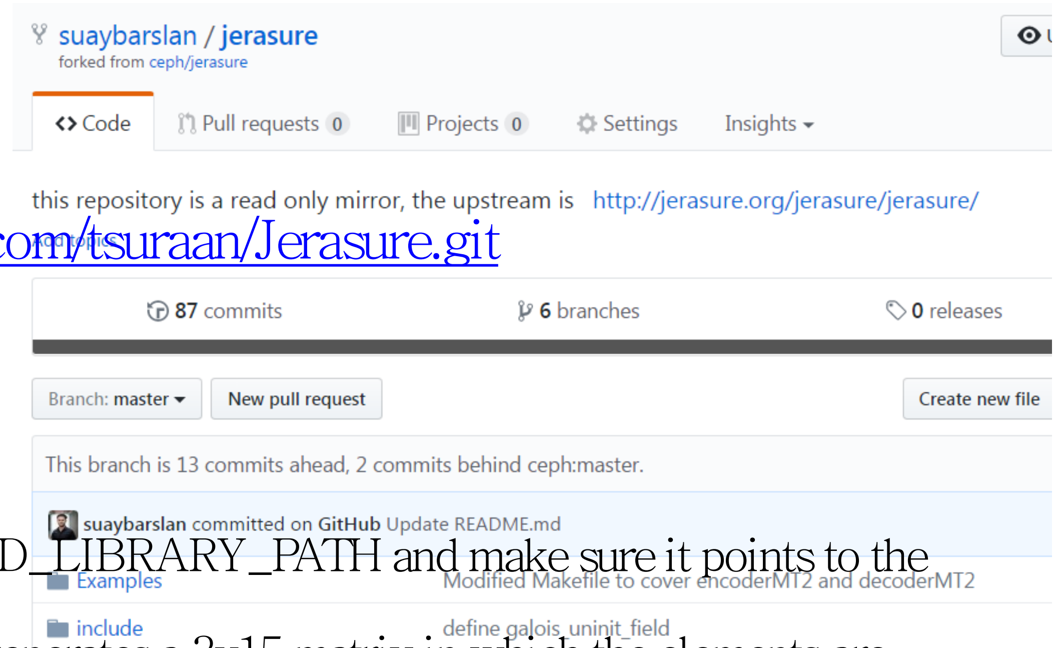
MEF UNIVERSITY
Your Freedom in Learning

Download/Install GF-Complete

- To be able to use Jerasure library, we need to install GF-Complete library (current version 1.02).
  - For easy cloning use git (yum install git):
    - git clone https://github.com/ceph/gf-complete.git
  - To be able to run ./configure, we need
    - yum install libtool
    - autoreconf –install
  - Then, configure using autoconf/automake
    - ./configure
    - sudo make
    - sudo make install
  - In case, the executable paths could not be found:
    - Check the enviroment variable LD_LIBRARY_PATH and make sure it points to the right directory.
- Try/test if it is succesfully installed:
  - gf_mult 5 4 4 **(multiply 5 by 4 in GF(2^4))**
  - gf_div 7 5 4 **(divide 7 by 5 in GF(2^4))**

MEF UNIVERSITY
Your Freedom in Learning

Download/Install Jerasure 2.0



- For easy cloning use git
  - git clone https://github.com/tsuraan/Jerasure.git
  - autoreconf –install
  - ./configure
  - sudo make
  - sudo make install
- Check the enviroment variable LD_LIBRARY_PATH and make sure it points to the right directory (ex: /usr/lib/bin/)
- Try/test jerasure_01 3 15 8 that generates a 3x15 matrix in which the elements are chosen from GF(2^8).
- In case, your autoconf and/or automake is out of date, please upgrade them and include the install directory in your $PATH variable.
  - Run autreconf –ivf
  - Then, make and make install.
- If autoreconf –ivf asks for (if not installed already) libtoolize, install it:
  - Sudo yum install libtool

MEF UNIVERSITY
Your Freedom in Learning

Jerasure 2.0

MATH BEHIND IT

MEF UNIVERSITY
Your Freedom in Learning

How does GF operations work?

- GF(q) is a finite set of elements on which two major operations, **addition** and **multiplication** are defined.
- In order to satisfy the axioms of a mathematical field, q must be either a prime number or a power of prime.
- Example: GF(5) = {0,1,2,3,4}.
  - 2 + 1 = 3,
  - 2 + 3 = 0,
  - 3 + 4 = 2
  - 2 x 3 = 1,
  - 3 x 4 = 2 …

Extended GF

- **Definition:** A primitive element α is an element such that every field element except zero can be expressed as a power of α.
- **Example:** 2 and 3 are primitive elements of GF(5).
- Extended field $GF(q^m)$ requires polynomial algebra.
- Polynomials represent the elements in the extended field.
- Polynomial arithmetic is similar to real number system except coefficients of the polynomials obeys the axioms of GF(q).
- **Definition:** The root of the primitive polynomial is known as the primitive element of $GF(q^m)$.

MEF UNIVERSITY
Your Freedom in Learning

Binary Base Field / Hardware

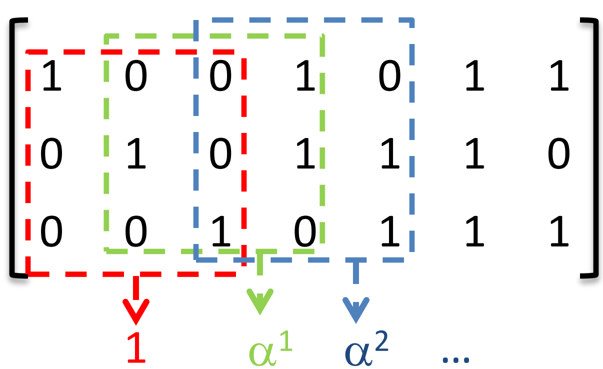
- Use $GF(2^m)$ to generate hardware friendly representation of field elements i.e., binary vectors.
- The coefficients of the polynomials are from GF(2).
- **Example:** Let us construct $GF(2^3)$ using the primitive element of the form $x^3 + x + 1$.
  - Let α be a primitive element and thus $\alpha^3 = \alpha + 1$.
  - Similarly, other elements can be found as given by the table to the right.
  - Each element can be expressed as a three-bit tuple in hardware.

Power	Polynomial	Binary
α^0	1	(0,0,1)
α^1	α	(0,1,0)
α^2	α^2	(1,0,0)
α^3	$\alpha+1$	(0,1,1)
α^4	$\alpha^2+\alpha$	(1,1,0)
α^5	$\alpha^2+\alpha+1$	(1,1,1)
α^6	α^2+1	(1,0,1)

MEF UNIVERSITY
Your Freedom in Learning

Major operations: Addition/Multiplication

- Addition in GF is simple.
  - Element/s addition is equivalent to binary representation addition – XOR operation.
- For multiplication, we need to find an alternative representation of elements using matrices instead of vectors.
- Here is how:



With this new representation…

- We can do matrix multiplications in binary:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & \alpha & \alpha^2 \\ 1 & \alpha^2 & \alpha^4 \end{bmatrix} \begin{bmatrix} \alpha \\ 1 \\ \alpha^5 \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{matrix} \\ \alpha^2 \\ \\ \\ 1 \\ \\ \\ \alpha \\ \\ \end{matrix}$$

9 XOR operations

12 XOR operations

15 XOR operations

Encoding/Decoding

- Matrix algebra. **G** is the generator matrix. **d** is data vector.

$$[\mathbf{d}_{1\times k} \quad \mathbf{c}_{1\times m}]^T = \mathbf{G}_{n\times k} \times \mathbf{d}^T_{k\times 1}$$

- **G** can be decomposed to compute parity vector **c**.

$$\mathbf{G} = [\mathbf{I}_{k\times k} \quad \mathbf{P}_{k\times m}]^T \qquad \mathbf{c}^T_{m\times 1} = \mathbf{P}_{m\times k} \times \mathbf{d}^T_{k\times 1}$$

- From n symbols, suppose that we lost some of them.
- From the unlost symbol set, choose any k symbols.
  - This corresponds to a particular k rows of G. ➔ $\mathbf{G}'_{k\times k}$
- Multiply the unlost/selected k symbols with the inverse of $\mathbf{G}'_{k\times k}$

$$\mathbf{d}^T_{k\times 1} = {\mathbf{G}'_{k\times k}}^{-1} \times [\mathbf{d}' \quad \mathbf{c}']^T_{k\times 1}$$

Reconstruction

Jerasure 2.0

WHAT EXTRAS J 2.0 BRINGS?

MEF UNIVERSITY
Your Freedom in Learning

What is SIMD?

- There are four different types of comupters:
  - SISD: single instruction single data
  - SIMD: single instruction multiple data
  - MISD: multiple instruction single data
  - MIMD: multiple instruction multiple data
- Interesting one is SIMD:
  - all parallel units share the same instruction, but they carry it out on different data elements. The idea is that you can, say, add the arrays [0,1,7,3] and [2,3,5,4] element-wise to obtain the array [2,4,12,7] in one step: for this, there have to be four arithmetic units at work, but they can all share the same instruction (here, "add"), and work by all performing the same actions in tight, lock-step synchronicity.
  - This usually means putting multiple data-manipulation techniques inside the same processing core as one instruction decoder, for the sake of the tight timekeeping.
- An example to MIMD is multi-threaded processing.

GF addition/multiplications using SIMD

- Basic data type in SIMD instructions is 128-bit words (machine dependent).
  - One interesting question is how much of a change does Jerasure implementation require when we have 256-bit and 512-bit numeric processing capabilities.
  - Reference: https://software.intel.com/en-us/blogs/2013/avx-512-instructions
- Instructions used in Jerasure:
  - mm_set1_epi8, mm_set1_epi16, mm_set1_epi32, mm_set1_epi64.
    - Generates a 128-bit var iable by replicating 1-byte, 2-byte, 4-byte and 8-byte copies.
  - mm_and_si128(a, b), mm_xor_si128(a, b)
    - Performs addition and XOR of 128-bit words a and b
  - mm_srli_epi64(a, b), mm_slli_epi64(a, b)
    - Treat each input as two 64-bit word and right/left shift each by by bits.

MEF UNIVERSITY
Your Freedom in Learning

Example: Multiple 128-bit region *A* by 7 in GF(2^4)

byte	f	e	d	c	b	a	9	8	7	6	5	4	3	2	1	0
table1:	0b	0c	05	02	04	03	0a	0d	06	01	08	0f	09	0e	07	00
table2 = mm_slli_epi64(table1, 4):	b0	c0	50	20	40	30	a0	d0	60	10	80	f0	90	e0	70	00
mask1 = mm_set1_epi8(0xf):	0f	0f	0f	0f	0f	0f	0f	0f	0f	0f	0f	0f	0f	0f	0f	0f
mask2 = mm_set1_epi8(0xf0):	f0	f0	f0	f0	f0	f0	f0	f0	f0	f0	f0	f0	f0	f0	f0	f0
A:	39	1d	9f	5a	aa	ab	15	c3	63	e0	7c	43	fb	83	16	23
1 *l = mm_and_si128(A, mask1):*	09	0d	0f	0a	0a	0b	05	03	03	00	0c	03	0b	03	06	03
2 *l = mm_shuffle_epi8(l, table1):*	0a	05	0b	03	03	04	08	09	09	00	02	09	04	09	01	09
3 *h = mm_and_si128(A, mask2):*	30	10	90	50	a0	a0	10	c0	60	e0	70	40	f0	80	10	20
4 *h = mm_srli_epi64(h, 4):*	03	01	09	05	0a	0a	01	0c	06	0e	07	04	0f	08	01	02
5 *h = mm_shuffle_epi8(h, table2):*	90	70	a0	80	30	30	70	20	10	c0	60	f0	b0	d0	70	e0
6 *yA = mm_xor_si128(h, l):*	9a	75	ab	83	33	34	78	29	19	c0	62	f9	b4	d9	71	e9

All the elements in GF(2^4)

Table for 7*element in GF(2^4)

Mask for isolation of the LS 4-bits

Mask for isolation of the MS 4-bits

Isolate the LS 4-bits

Look up the corresponding value from the table

Isolate MS 4-bits

Shift to LS position

Look up the corresponding value from the table

Finally XOR LS and MS 4-bits to obtain the result.

Instruction Sequences

LS: Least Significant

MS: Most Significant

- SIMD example: Performing operations 128-bit at a time.
- For 32 independent multiplications, we perform only 6 instructions.
- More to come later.
- For more info please see: http://web.eecs.utk.edu/~plank/plank/papers/FAST-2013-GF.pdf

Jerasure 2.0

ENCODING/DECODING ARCHITECTURE

MEF UNIVERSITY
Your Freedom in Learning

Operations/timing: encoder.c

- Main engine for encoding is encoder.c
  - Variable definitions
  - Error check arguments
    - Packetsize, buffersize adjustments.
    - Arguments/Parameters validy check.
    - File size adjustments
  - Create coding/bit/schedule matrices
  - Loop starts
    - File read, zero padding, pointer set
    - Encoding
    - Write the data/parities to files
  - Create/write metadata

Encoding Speed (MB/sec)

Total Speed (MB/sec)

MEF UNIVERSITY
Your Freedom in Learning

Operations/timing: decoder.c

- Main engine for decoding is decoder.c
  - Variable definitions
  - Read metadata and error check
  - Create coding/bit/schedule matrices
  - Loop starts
    - Determine erased files, readin available data
    - Decoding
    - Write the decoded data into a single output file.

Decoding Speed (MB/sec)

Total Speed (MB/sec)

MEF UNIVERSITY
Your Freedom in Learning

Inputs

- *Inputfile*: File that is input to encoder
- *k*: number of data files.
- *m*: number of coding files.
- *coding technique*: one of the following:
  - reed_sol_van, reed_sol_r6_op, cauchy_orig, cauchy_good, liberation, blaum_roth, liber8tion.
- *w* : word size.
- *packetsize*: architectural parameter(default=0 & see next page)
- *buffersize*: architectural parameter (default=0 & see next page)
  - Must be a multiple of *sizeof(long)*w*k*packetsize (packetsize is not 0)*, otherwise choose the least bigger number that is a multiple of *sizeof(long)*w*k*packetsize.*

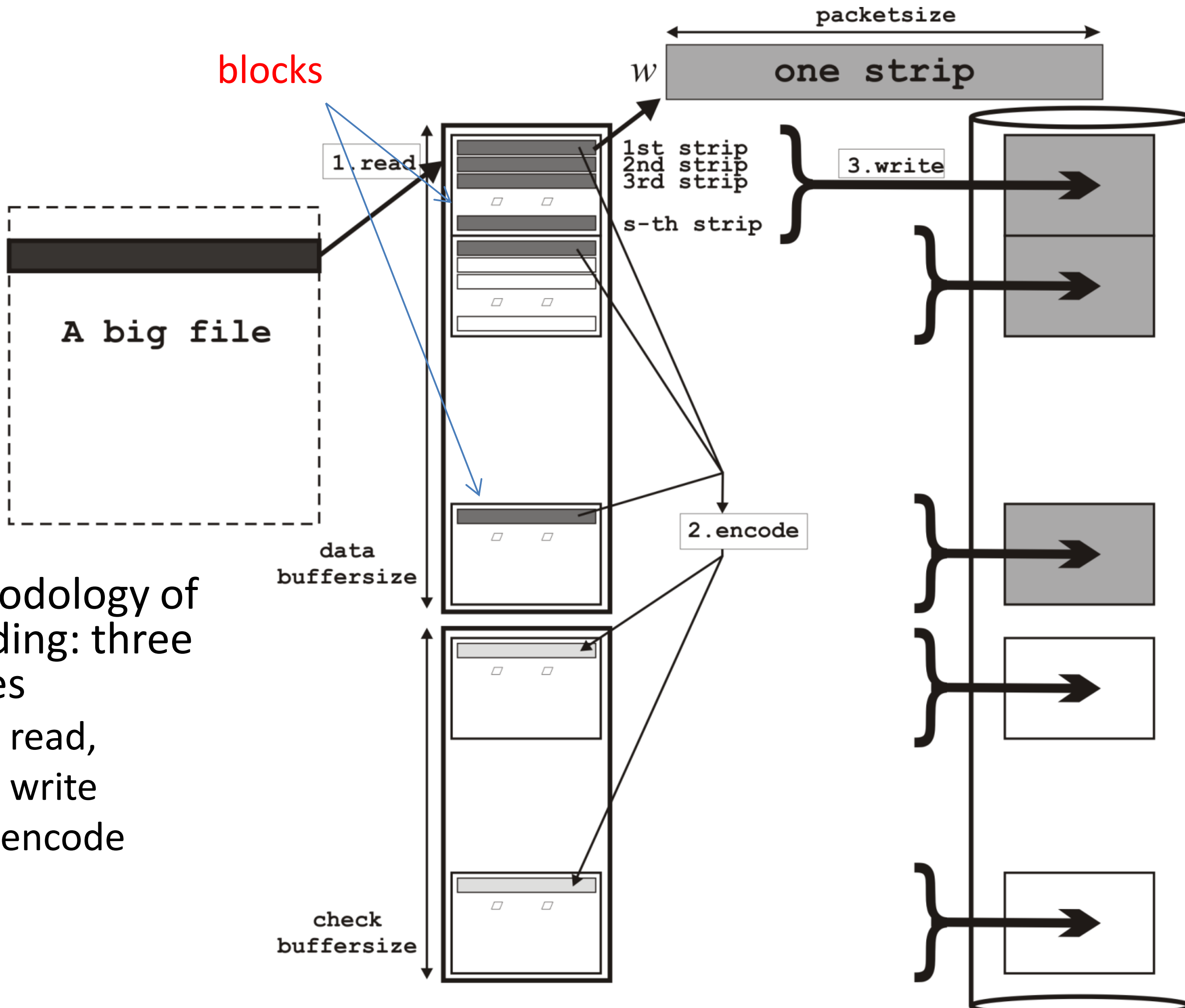
MEF UNIVERSITY
Your Freedom in Learning

File size adjustment and blocksize

- Perform the following operations sequentially (packetsize is NOT 0):
- while *filesize* is NOT a multiple of *sizeof(long)*w*k*packetsize*
  - Increment *filesize*
- while *filesize* is NOT a multiple of *buffersize*
  - Increment *filesize*
- blocksize = filesize/k

MEF UNIVERSITY
Your Freedom in Learning

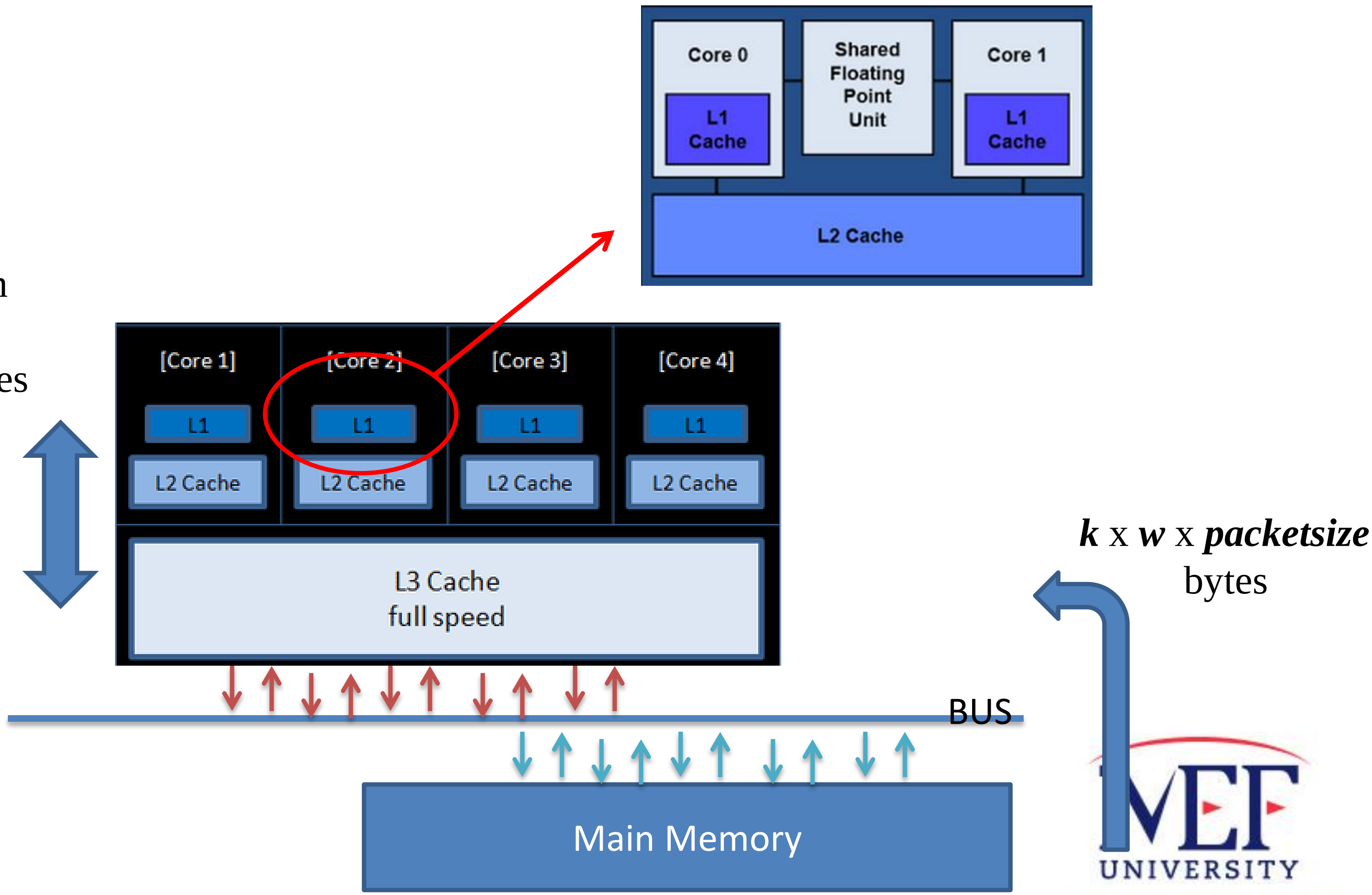
Encoding a Big file…



- Methodology of encoding: three phases
  - 1. read,
  - 2. write
  - 3.encode

Hardware structure

Interaction between CPU caches Cannot be controlled Through J2.0 software.

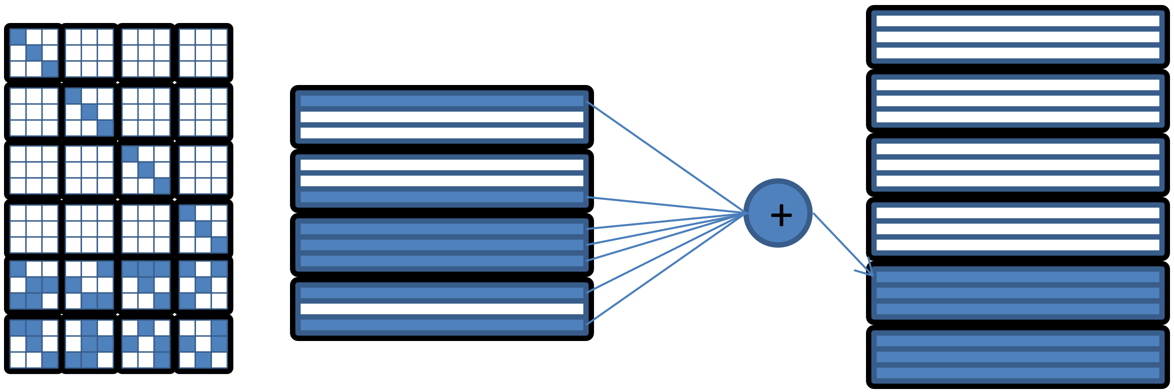


Example run and the output: Original

- Suppose we have an input file *input.txt*
- *Run:*
  - *encoder input.txt 8 4 reed_sol_van 8 2048 5000000*
- *Output:*
  - *Encode speed:*
    - *Encoding (MB/sec): 1382.26*
    - *En_Total (MB/sec): 296.35*
  - */Coding directory including 8 data files/4 parity files and a metadata file:*
    - *Input_k1.txt, input_k2.txt, …*
    - *Input_m1.txt, input_m2.txt…*
    - *Input_meta.txt*

MEF UNIVERSITY
Your Freedom in Learning

Strips encoding… (based on Plank's simulation study-not ours)



- Why different encoding methods quite different?
  - Generator matrix design given k, m, w (particularly number of XORs)
  - Buffersize, packetsize and s are tied together. If we set two of those, the remaining parameter is known.
- Observations:
  - Lower packet sizes have less tight XOR loops, but better cache behavior. Higher packetsizes perform XORs over larger regions, but cause more cache misses.
  - Optimal packetsize is where the code makes best use of L1 cache.
- Optimal packetsize decreases as any one of k,m,w increases.

MEF UNIVERSITY
Your Freedom in Learning

RAM disk for improved I/O

- Check the available space in your RAM:
  - `free -g`
- Create a folder to use as a mount point for your RAM disk:
  - `mkdir /mnt/ramdisk`
- Use mount command to create a RAM disk
  - `mount -t [TYPE] -o size=[SIZE] [FSTYPE] [MOUNTPOINT]`
  - Ff
  - [TYPE] : tmpfs/ramfs
  - [SIZE] : SIZE of the RAMdisk
  - [FSTYPE]: tmpfs/ramfs/ext4
- Example: **`mount -t tmpfs -o size=512m tmpfs /mnt/ramdisk`**
- Add the mount enrty into /etc/fstab to make the RAM disk persist over reboots.
- Make sure you run “Jerasure” under this folder.

MEF UNIVERSITY
Your Freedom in Learning

Changes in configuration/make files

- I took a design approach so as to make minimum changes to the configuration and compilation process.
- Here is the list:
  - Add a new flag in configure.ac: *line 16*: ${CFLAGS='-g -O3 -Wall -lpthread'}
  - Let us assume we have a new script called encoderMT.c inside the /Examples folder.
    - Add encoderMT to **bin_PROGRAMS** in Examples/Makefile.am
    - Add encoderMT_LDADD = $(LDADD) ../src/libtiming.a at the end of the same file
    - Also need to add the source file: encoderMT_SOURCES = encoderMT.c
- Need to go through the same procedure for every file you add to *Jerasure*.

MEF UNIVERSITY
Your Freedom in Learning

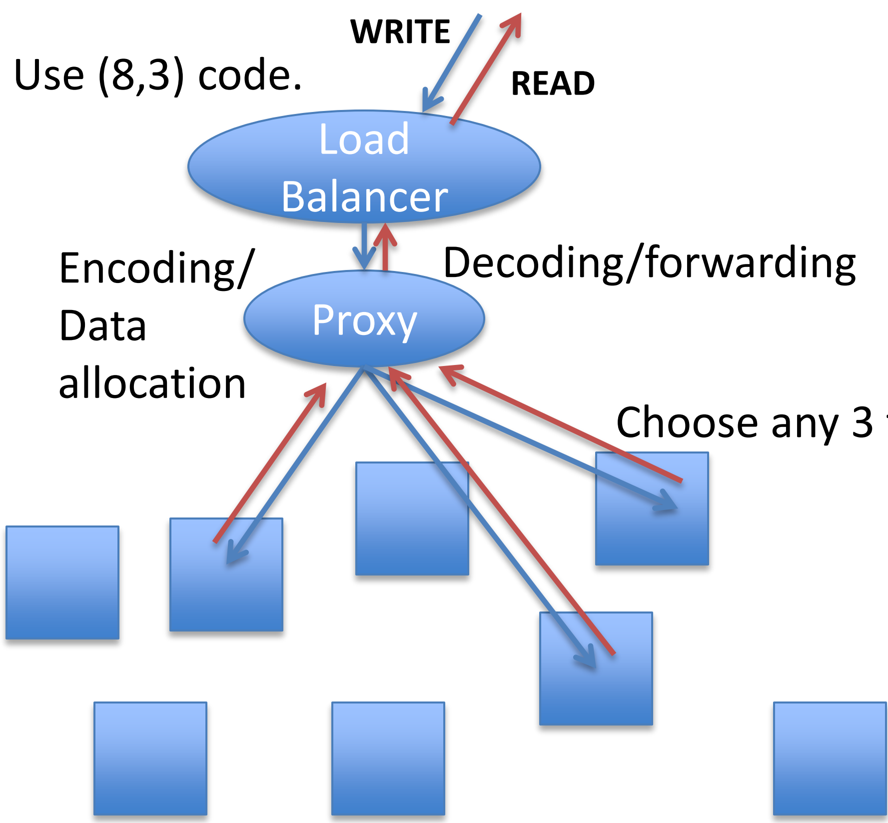
Example run and the output: MT

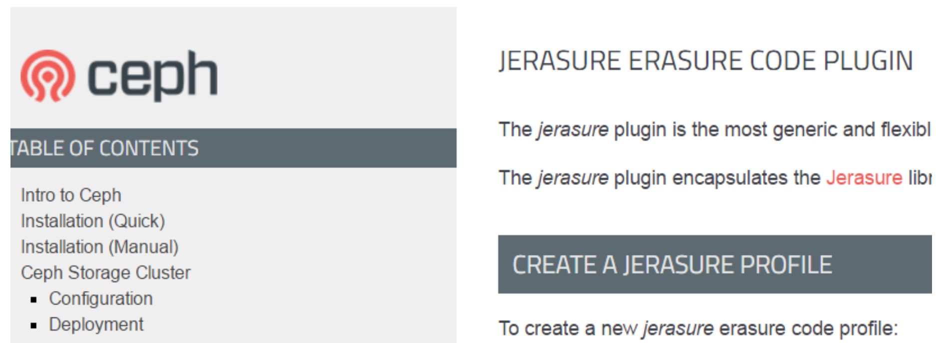
- Suppose we have an input file *input.txt*
- *Run:*
  - *encoderMT2 input.txt 8 4 8 2048 5000000*
  - *encoder_omp input.txt 8 4 8 2048 5000000*
- *Output:*
  - *Encode speed:*
    - *Encoding (MB/sec): 4252.88*
    - *En_Total (MB/sec): 697.61*
  - */Coding directory including 8 data files/4 parity files and a metadata file:*
    - *Input_k1.txt, input_k2.txt, …*
    - *Input_m1.txt, input_m2.txt…*
    - *Input_meta.txt*
- Decoder is pretty simple:
  - decoder(decoderMT2, decoder_omp) [file_name]

MEF UNIVERSITY
Your Freedom in Learning

Cluster Requirements for Data Protection

- Example: Distributed Storage Systems… OpenStack Swift, Ceph…





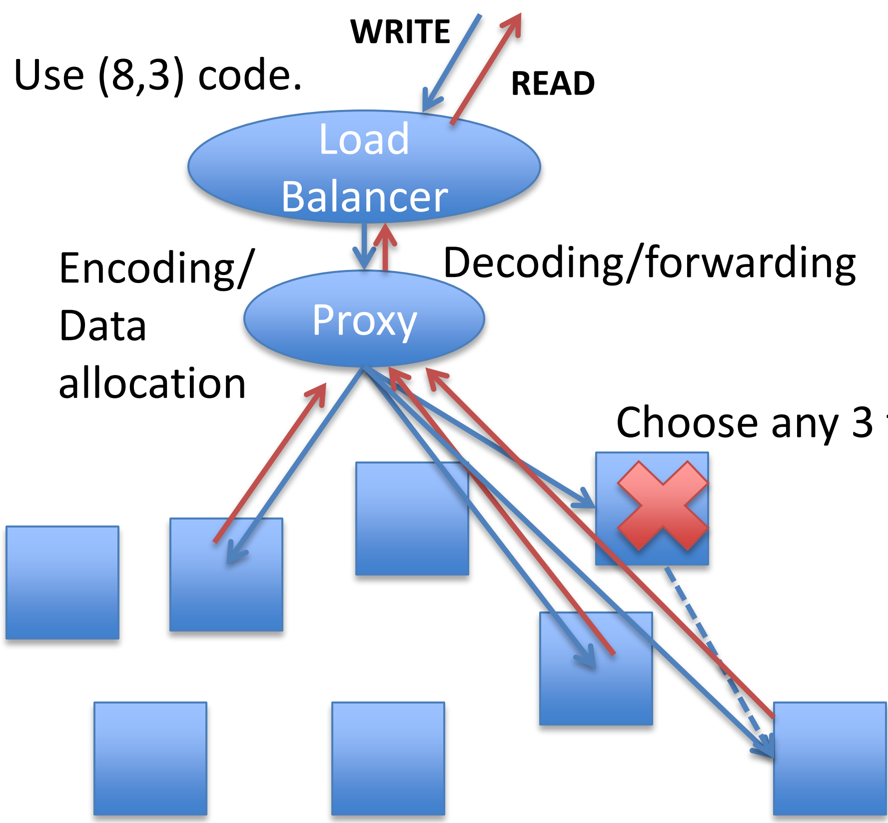
Choose any 3 that are most responsive.

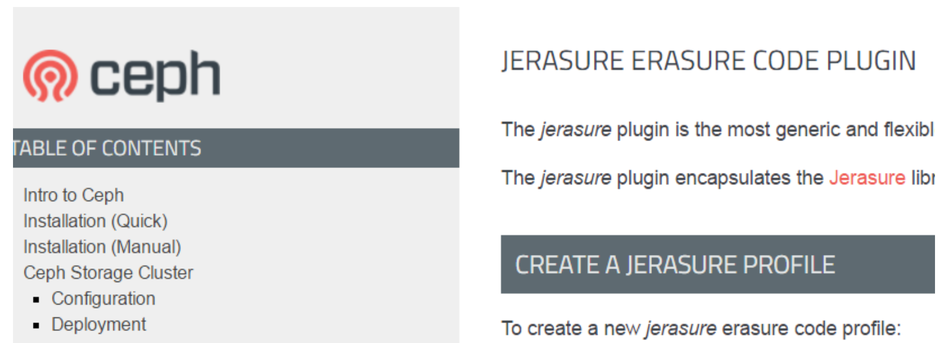
WRITE/READ requests on
A distributed system protected
By the erasure code such as
Jeasure.

MEF UNIVERSITY
Your Freedom in Learning

Cluster Requirements for Data Protection: Fault Tolerance

- Example: Distributed Storage Systems…OpenStack Swift, Ceph…





ceph

TABLE OF CONTENTS

JERASURE ERASURE CODE PLUGIN

The *jerasure* plugin is the most generic and flexibl

The *jerasure* plugin encapsulates the Jerasure libı

CREATE A JERASURE PROFILE

To create a new *jerasure* erasure code profile:

Choose any 3 that are most responsive.

In case of failure of a node:
Degraded reads occur.

MEF UNIVERSITY
Your Freedom in Learning

Cluster Requirements for Data Protection: Cluster Data Repair

- Example: Distributed Storage Systems…OpenStack Swift, Ceph…

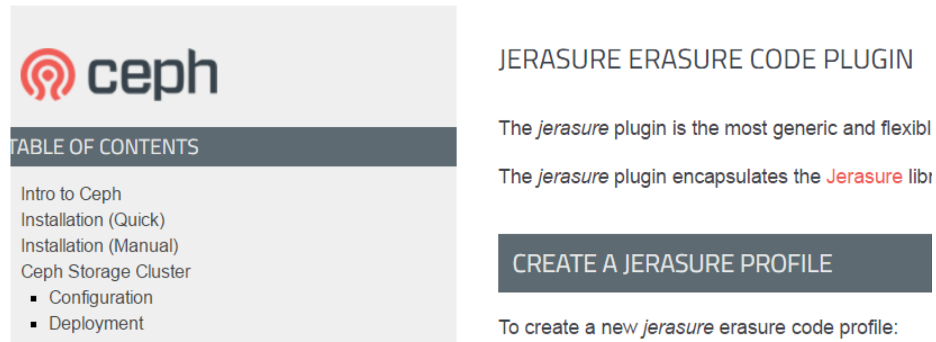

ceph

TABLE OF CONTENTS

Intro to Ceph
Installation (Quick)
Installation (Manual)
Ceph Storage Cluster
- Configuration
- Deployment

JERASURE ERASURE CODE PLUGIN

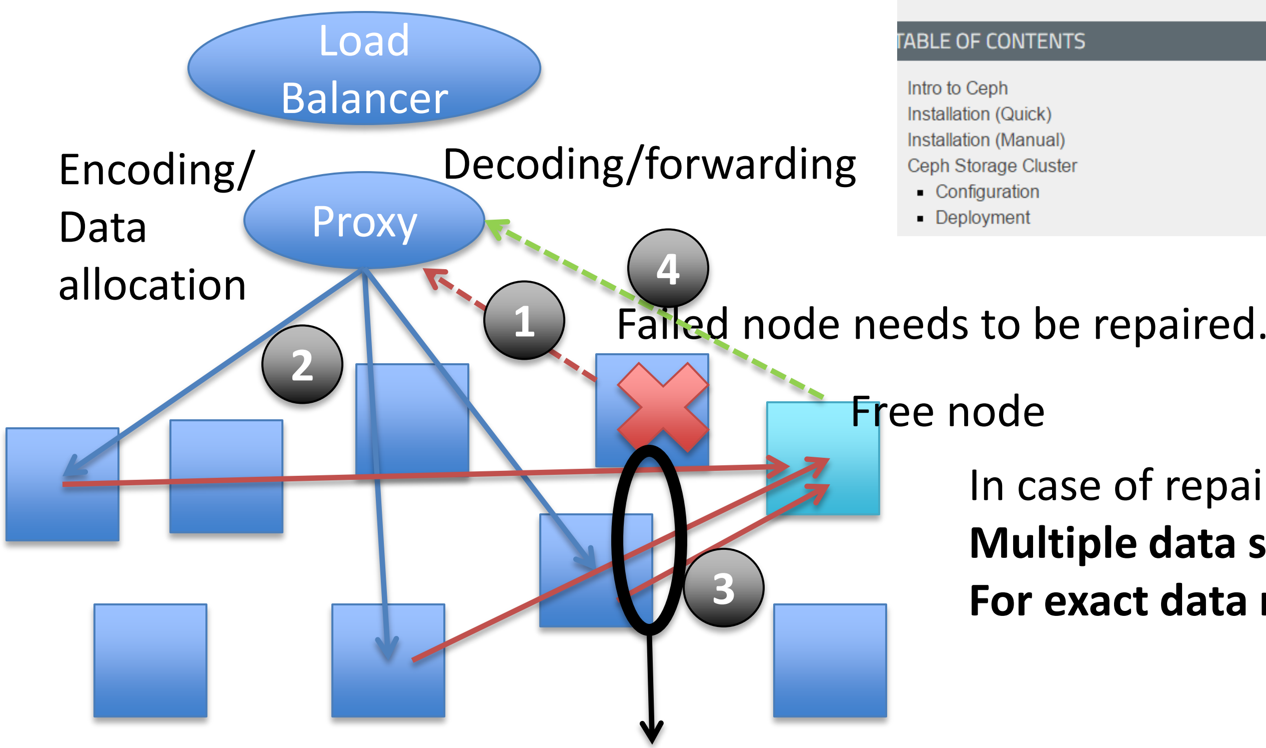
The *jerasure* plugin is the most generic and flexibl

The *jerasure* plugin encapsulates the Jerasure libr

CREATE A JERASURE PROFILE

To create a new *jerasure* erasure code profile:

Use (8,3) code.



In case of repair of a node:
Multiple data streams into free node
For exact data reconstruction.

One active research area:
How to minimize this communication.

MEF UNIVERSITY
Your Freedom in Learning