Multi-threaded erasure coding in *Jerasure*

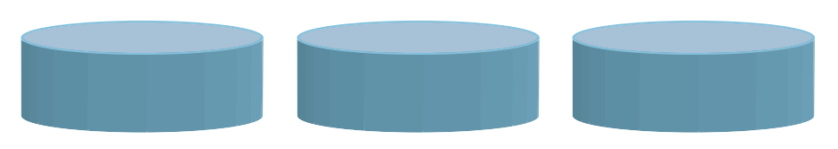
Michael Jugan
mjugan1@utk.edu

Background

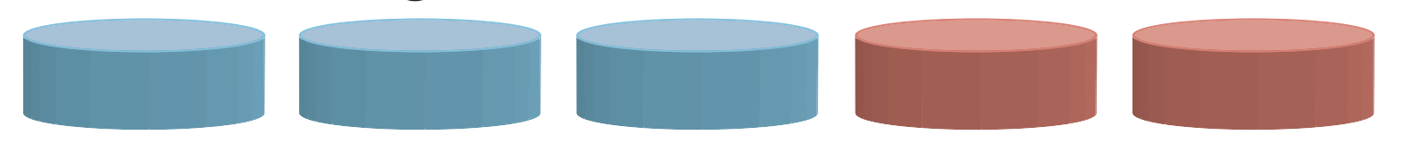
Erasure Coding

Defined:	**Applications:**
a technique used in computer systems to handle data loss and corruption	• network data transmission • file archiving • bar code reliability

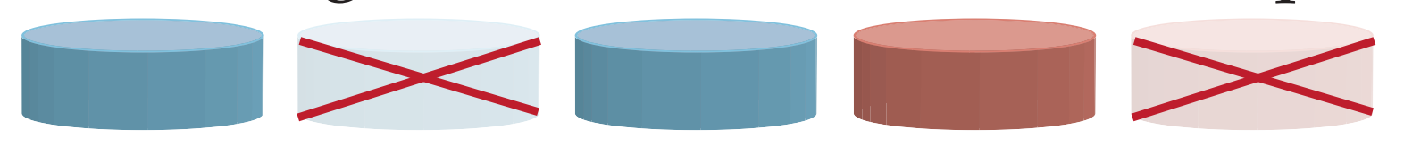
Start with *K* blocks of data.



Encode to get *M* additional blocks of coding data.



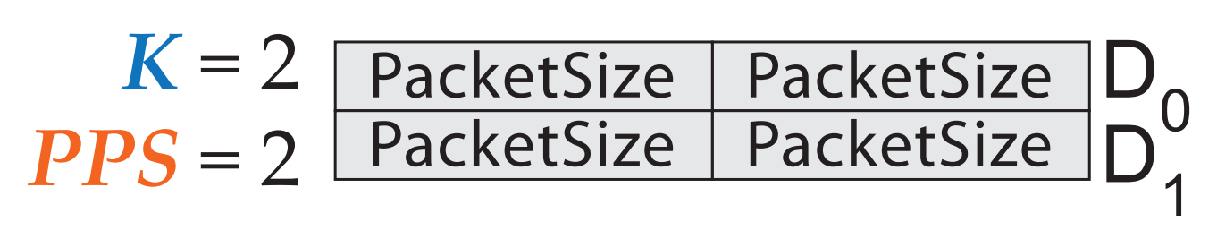
Decoding recovers the data when up to *M* blocks are lost.



Jerasure - an open-source erasure coding library

Reed-Solomon coding:

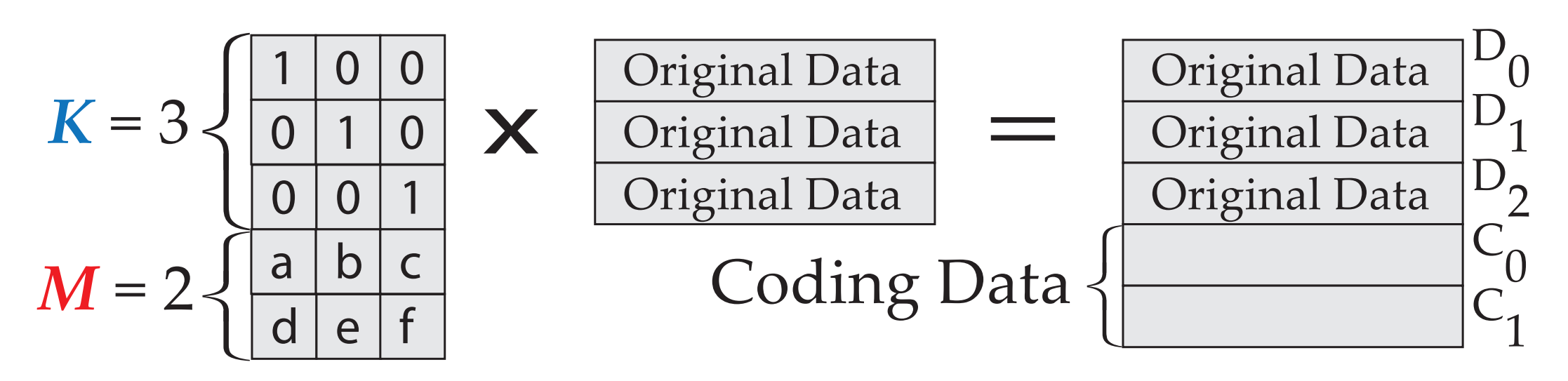
Data is organized in blocks.



PacketSize (*PS*)
x PacketsPerSlice (*PPS*)

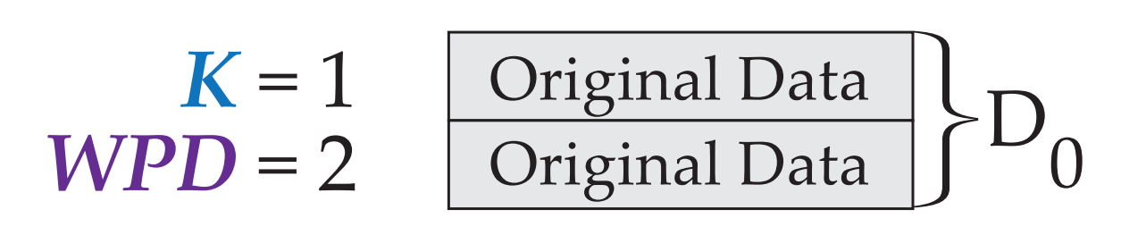
BlockSize

Performs matrix multiplication over Galois field GF(2^W)



Cauchy Reed-Solomon Coding:

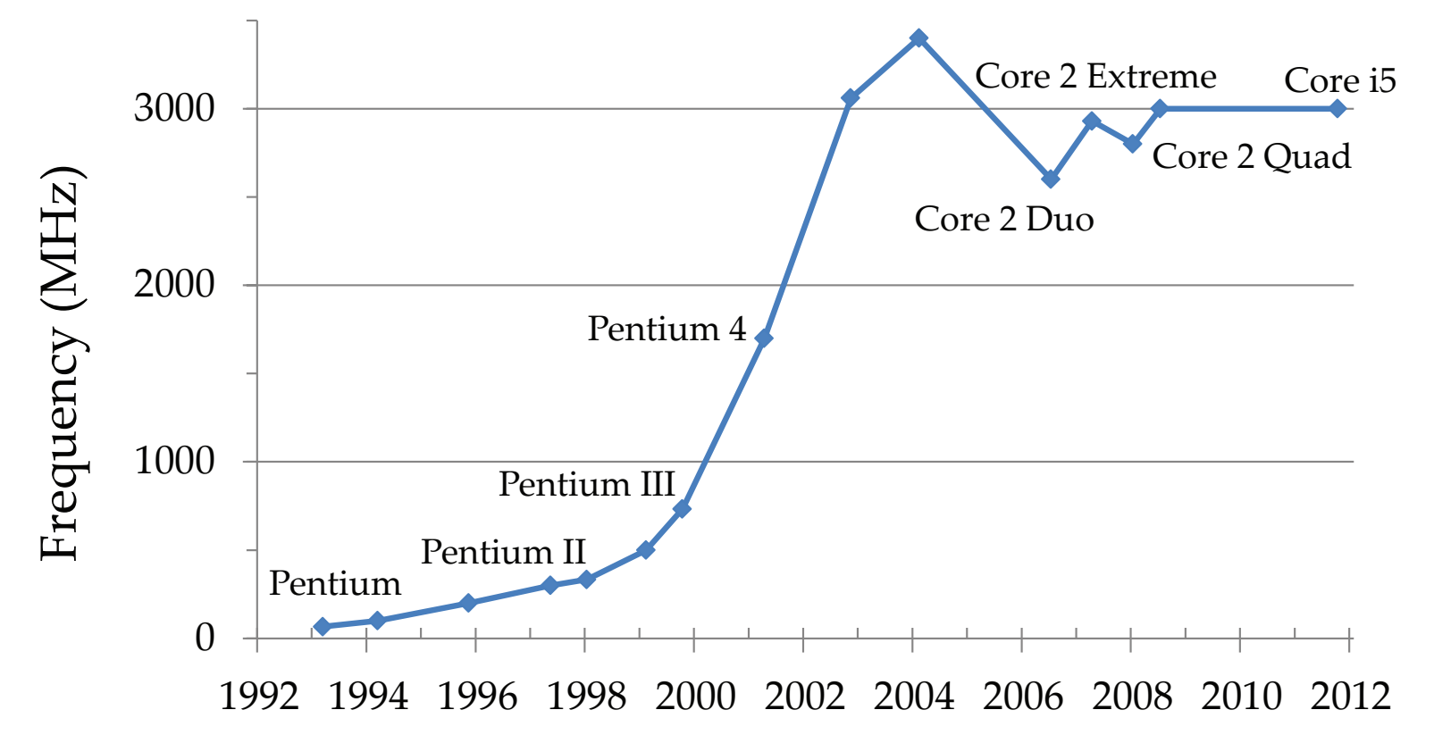
- Special case for $W = 1$
- Galois field arithmetic no longer needed
  only memcpy() and XOR
- New data-layout parameter: words-per-drive (*WPD*)



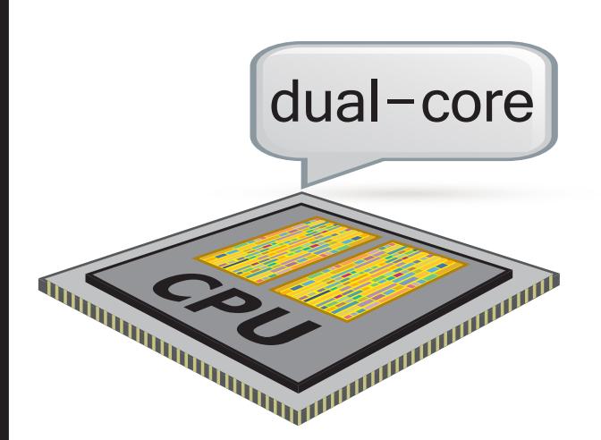
Requirement:
$$2^{WPD} + 1 \geq K + M$$

Parallel programming

In recent years , CPU speeds have leveled off due to device limitations such as power consumption.



Sources: [0],[1]



- Modern processor advancements focus heavily on multi-core CPUs.
- Software must be specially programmed to take advantage of the multiple cores.

Goals

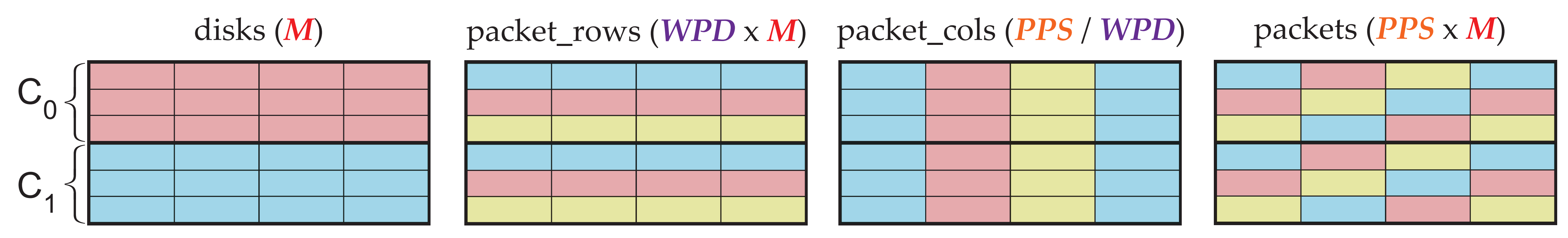
Update *Jearasure* to utilize multiple processor cores while coding

➤ increase performance
➤ provide an intuitive user interface

Implementation

Four methods for splitting the work among threads:

method name (max # of threads supported)



Each color represents the data coded by a single thread

Two new public variables were added to class JER_Slices.
- int NumberOfCores
- string MultiThreadMethod

Users add two additional lines of code before calling Encode().

```
slices->NumberOfCores = 4;
slices->MultiThreadMethod = “disks”;
slices->Encode();
```

Performance tests and results

Testing Methods

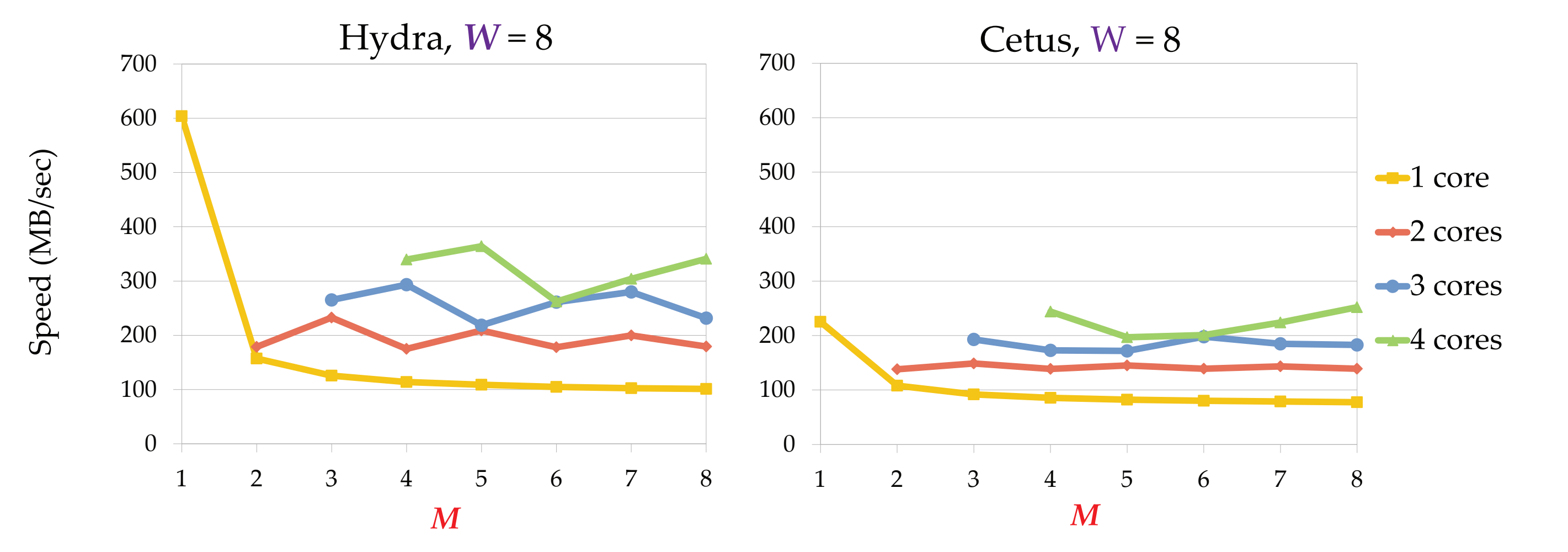
	Hydra	Cetus
CPU	Intel Xeon X5550 2.66 GHz Quad Core	Intel Core 2 Quad Q9300 2.55 GHz Quad Core
Bus speed	3200 MHz	1333 MHz
L1 Cache	256 KB	256 KB
L2 Cache	1 MB	6 MB
L3 Cache	8 MB	-
OS	Ubuntu 10.04 64-bit	Ubuntu 10.04 64-bit
Ram	12GB (1066 MHz DDR3)	4GB (800 MHz DDR2)
Memcpy() speed	6.33 GB/s	2.77 GB/s
XOR speed	4.53 GB/s	1.67 GB/s

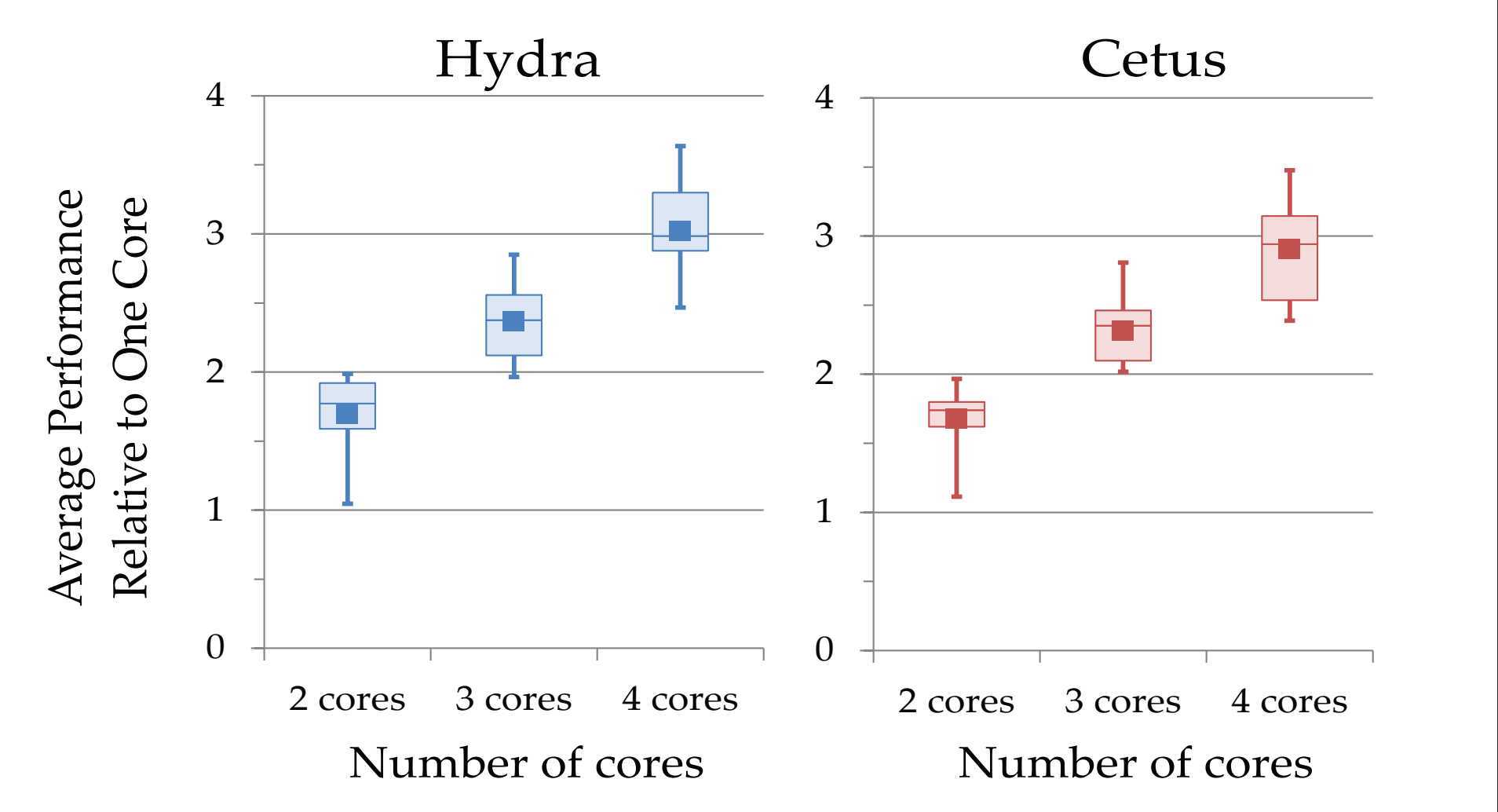
- Encoded 100 MB with Reed-Solomon and Cauchy Reed-Solomon
- -O3 compiler optimizations
- Averaged 5 runs where each run averaged 10 encodes

Coding Parameters:

Reed-Solomon	• W = 8, 16, 32
Cauchy Reed-Solomon	• *WPD* = 5, (the smallest valid value for all tested values of *K* and *M*
Both	• $K = 10$ • *M* varied from 1 to 8 • NumberOfCores increased from 1 to 4

Reed-Solomon



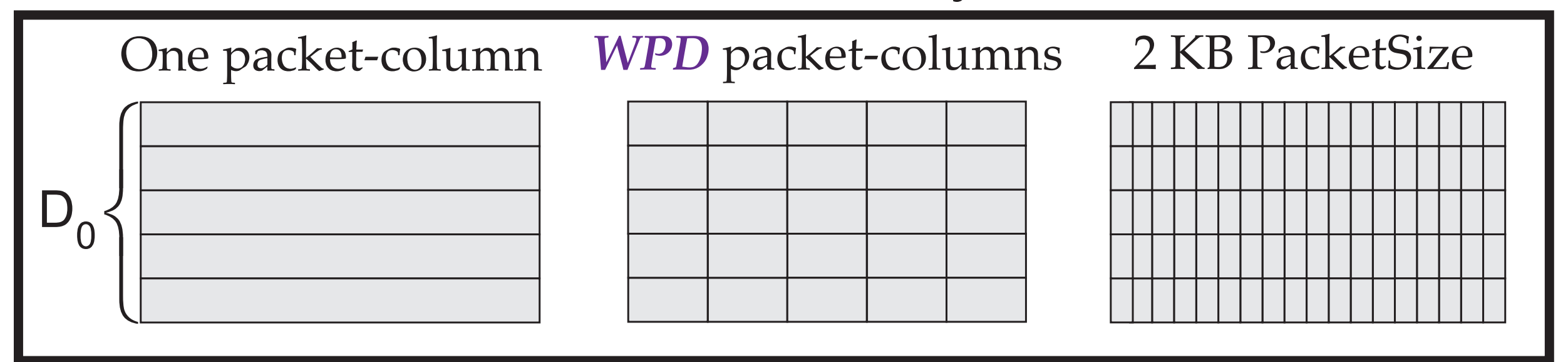


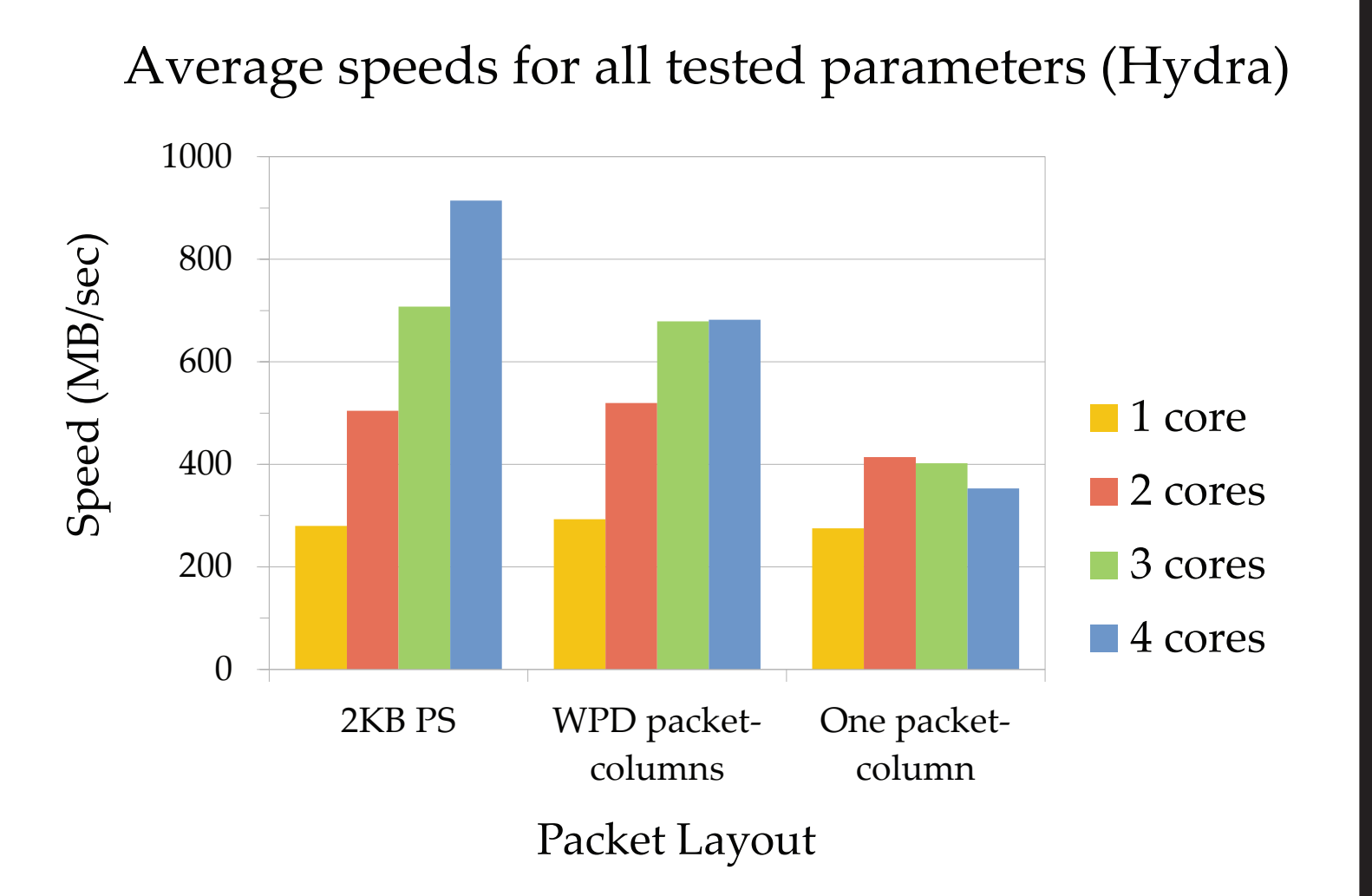
- Single-threaded encoding is extremely fast when $M = 1$.
  - The first row of the Reed-Solomon coding matrix contains all ones.
  - Therefore, Galois multiplication is not used to encode the first coding drive; the data is simply copied and XOR’d.

- The performance for a set number of cores varied depending upon the value of *M*.
- Altering *W* did not significantly change the relative speedup.

Cauchy Reed-Solomon

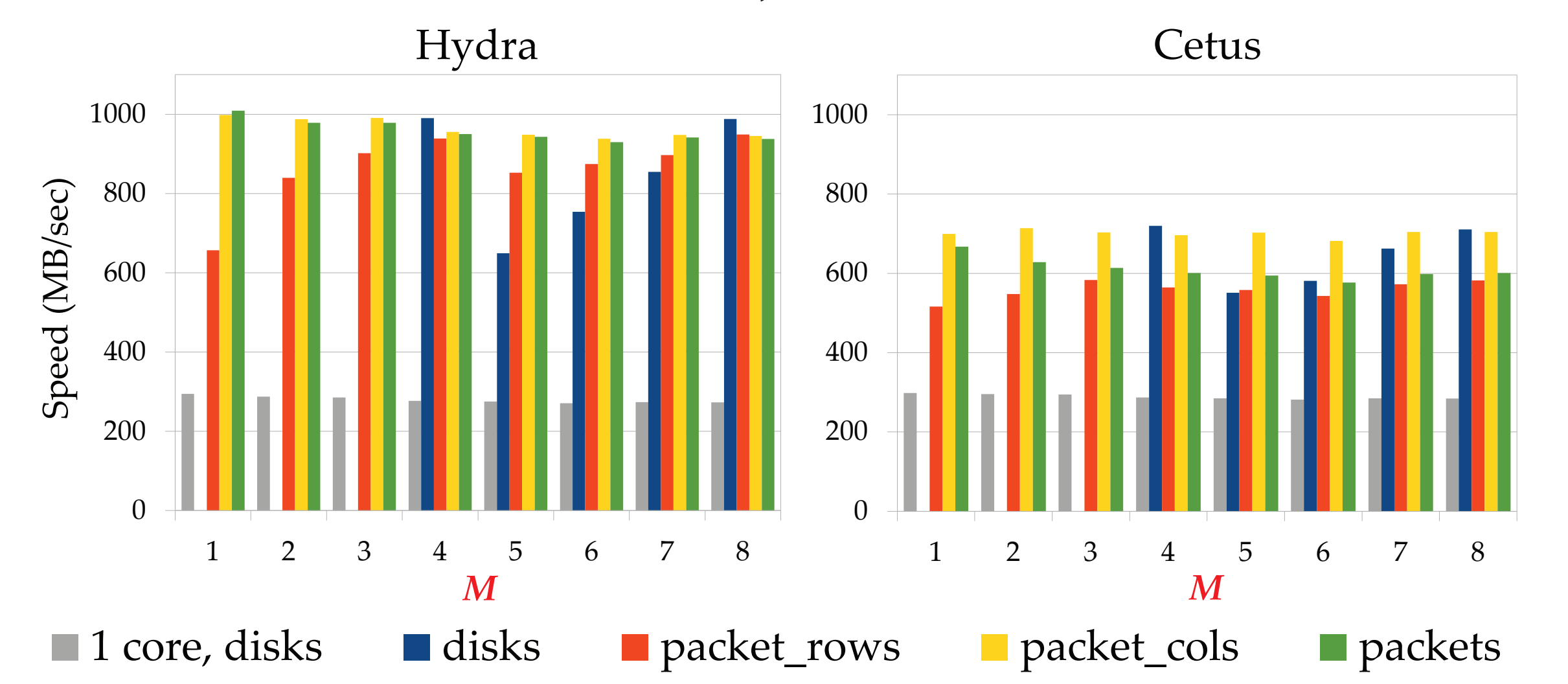
Tested with three different data layouts



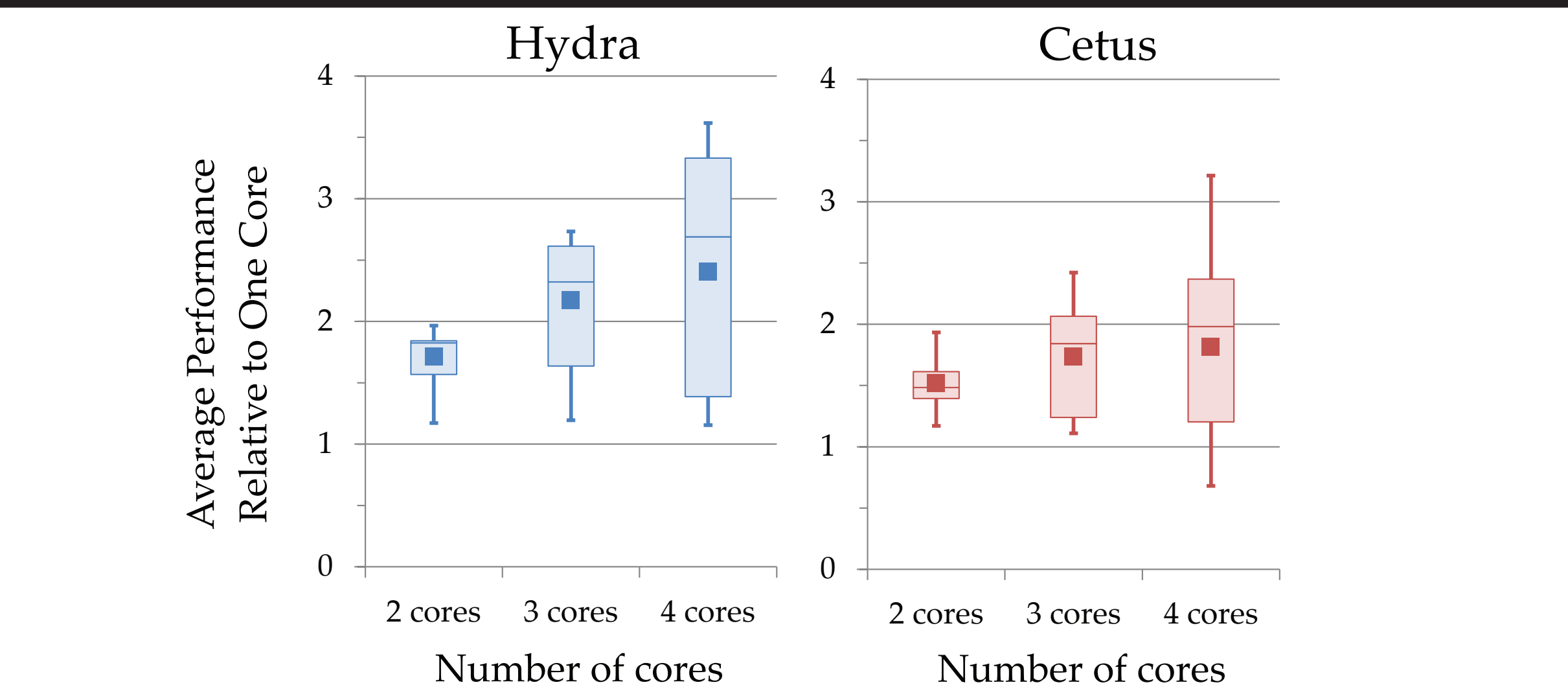


As the number of cores is increased, it becomes vital to use small packets.

4 cores, 2KB *PS*



🟦 Disks - fast when *M* is a multiple of NumberOfCores. Otherwise, performance suffers because some threads independently encode a second disk.
🟨 Packet_cols - consistently performs well when PS = 2KB



- Max speedup of 3.62x using 4 cores on Hydra
- Slower than the single-threaded case when large packet-sizes are used

Conclusion

- Increased performance
  - 2 cores ➔ almost 2x speedup
  - depends heavily on the chosen coding parameters
- Easy to use
  - requires two additional lines of code.
- Future work
  - automate the process of finding optimal settings

References

[0] "Intel Microprocessor Quick Reference Guide - Product Family," http://www.intel.com/pressroom/kits/quickreffam.htm.
[1] "Ark | Your source for information on Intel products," http://ark.intel.com.