Mojette Erasure Code for Distributed Storage

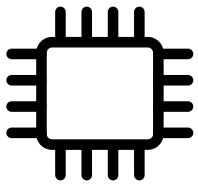
Code à effacement Mojette pour le stockage distribué

Dimitri Pertin

PhD Defense - 2016/04/25 - Polytech Nantes

UNIVERSITÉ DE NANTES

Hot vs Cold Data



Hot data

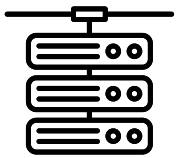
- Performance-oriented storage
- Expensive hardware
- High Performance Computing (instant access)

Cold Data

- Durability-oriented storage
- Commodity hardware, strong availability
- Archiving[1] (write once, read in few hours)

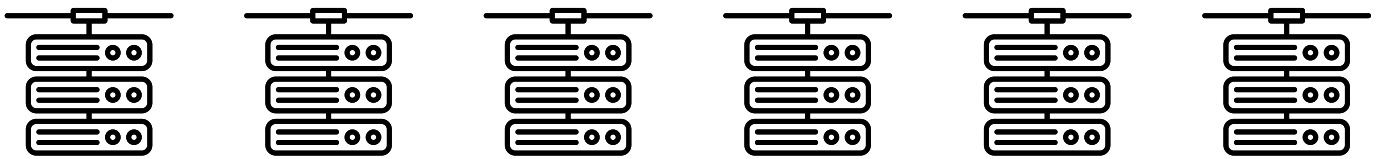
[1] Amazon Glacier guarantees < 1 sec unavailability every 3000 years

Erasure coding reduces the huge impact of data protection



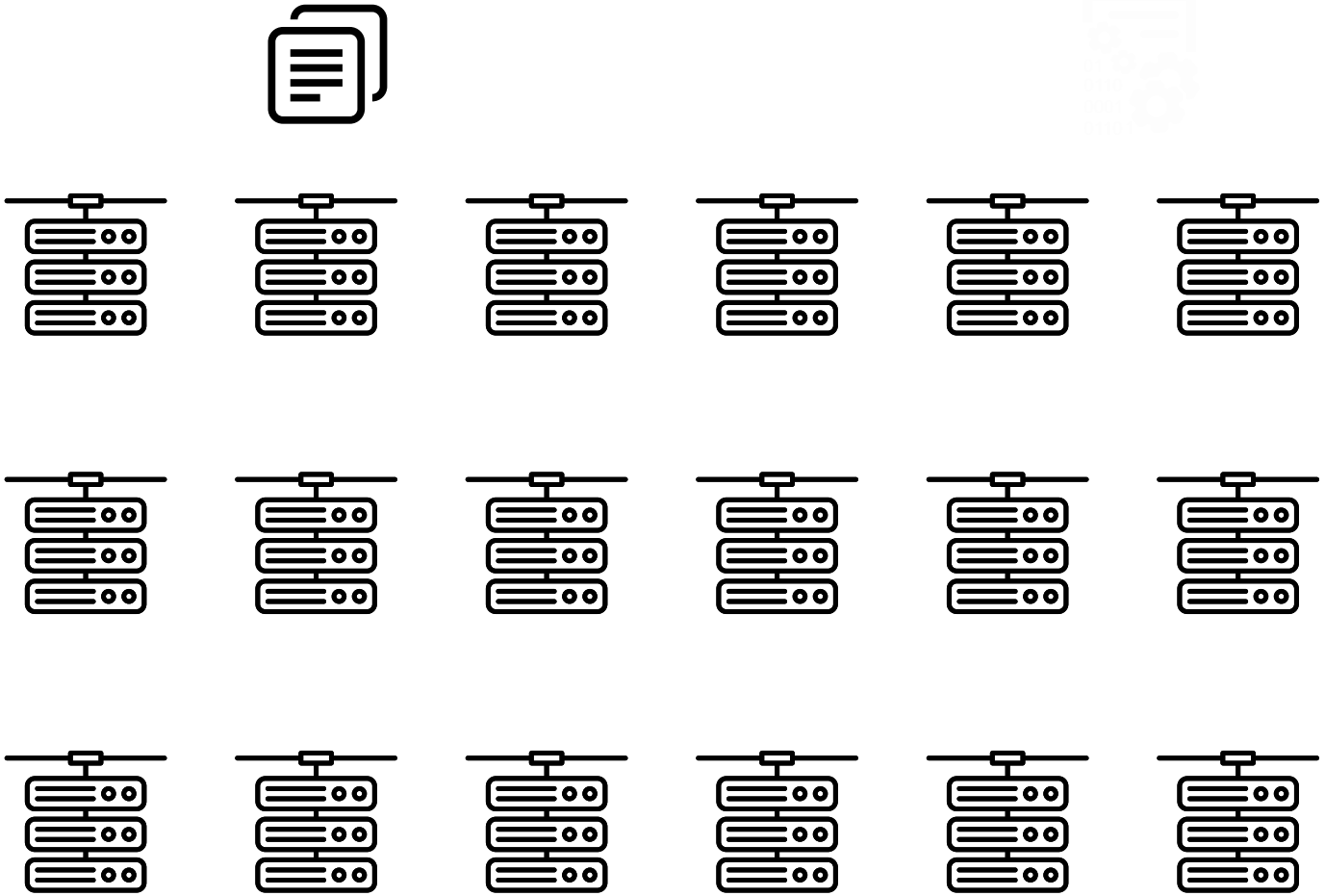
[Weatherspoon et al., 2002]

Erasure coding reduces the huge impact of data protection



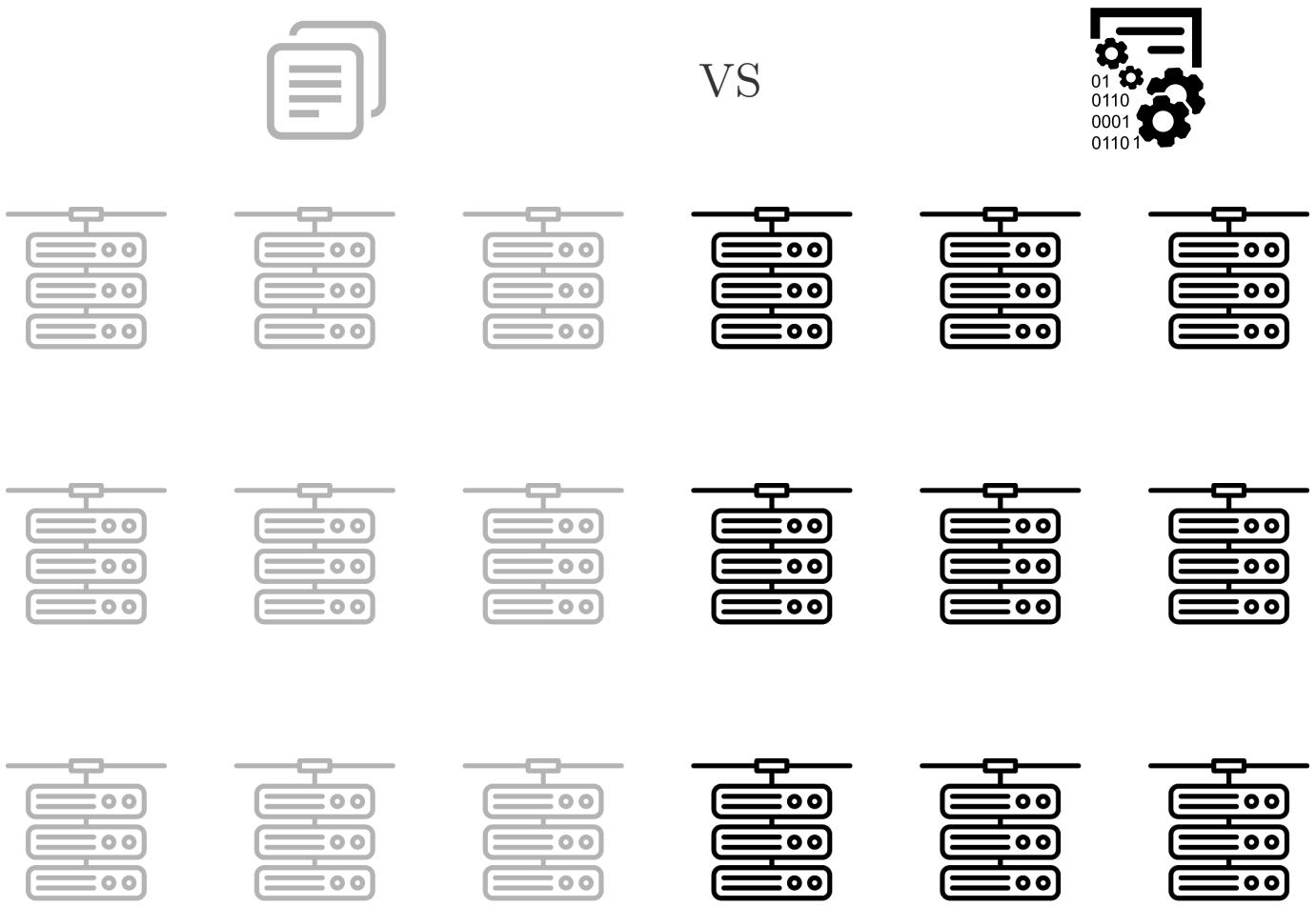
[Weatherspoon et al., 2002]

Erasure coding reduces the huge impact of data protection



[Weatherspoon et al., 2002]

Erasure coding reduces the huge impact of data protection

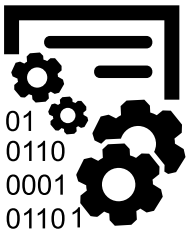


[Weatherspoon et al., 2002]

Problem: erasure codes limited to Cold Data

Replication for Hot Data

- no computation



Erasure coding limited to Cold Data [André et al., 2014]

- extra computations (encoding, decoding)
- delays during reads & writes

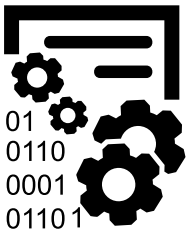
Problem: Today, no storage system can use erasure coding with hot data

Question: How can we design an efficient erasure code to build a storage system that can manage both hot and cold data?

Problem: erasure codes limited to Cold Data

Replication for Hot Data

- no computation



Erasure coding limited to Cold Data [André et al., 2014]

- extra computations (encoding, decoding)
- delays during reads & writes

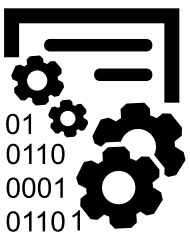
Problem: Today, no storage system can use erasure coding with hot data

Question: How can we design an efficient erasure code to build a storage system that can manage both hot and cold data?

Problem: erasure codes limited to Cold Data

Replication for Hot Data

- no computation



Erasure coding limited to Cold Data [André et al., 2014]

- extra computations (encoding, decoding)
- delays during reads & writes

Problem: Today, no storage system can use erasure coding with hot data

Question: How can we design an efficient erasure code to build a storage system that can manage both hot and cold data?

Projet ANR FEC4Cloud

- ANR project (Emergence)
- Favors erasure codes (EC) over plain replication for distributed storage
- Promotes EC designed in French labs
- Explores approaches based on discrete geometry

Outline

1. State of the Art

2. Systematic Mojette Erasure Code

3. Reprojection without reconstruction

4. Distributed Storage System: RozoFS

5. Conclusion

Outline

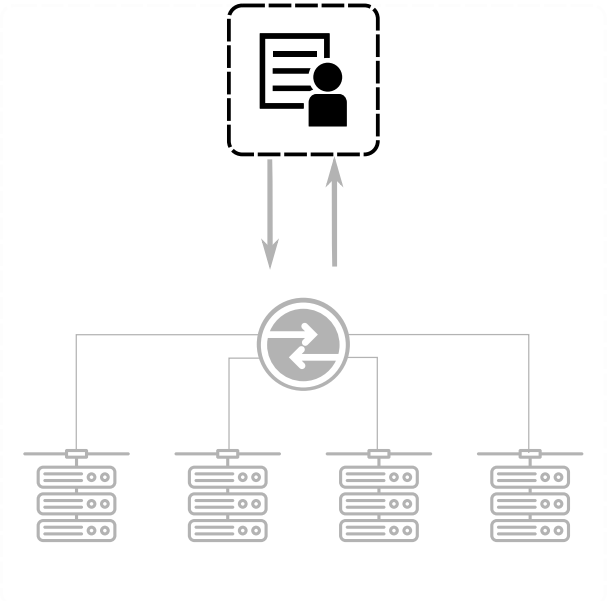
1. State of the Art

2. Systematic Mojette Erasure Code

3. Reprojection without reconstruction

4. Distributed Storage System: RozoFS

5. Conclusion



Outline

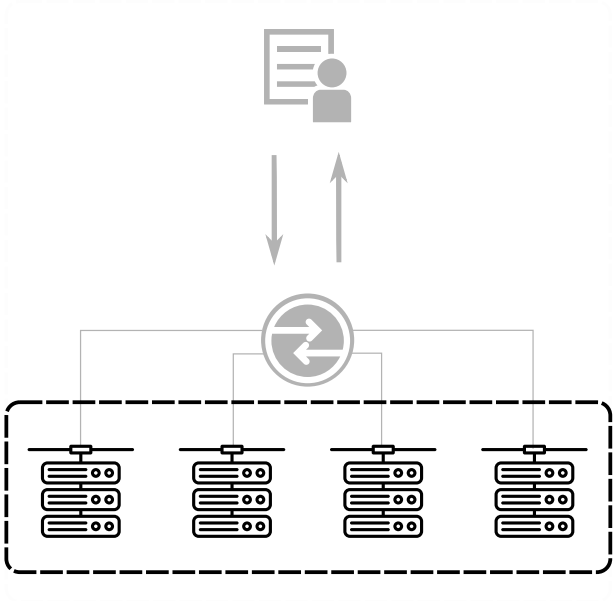
1. State of the Art

2. Systematic Mojette Erasure Code

3. Reprojection without reconstruction

4. Distributed Storage System: RozoFS

5. Conclusion



Outline

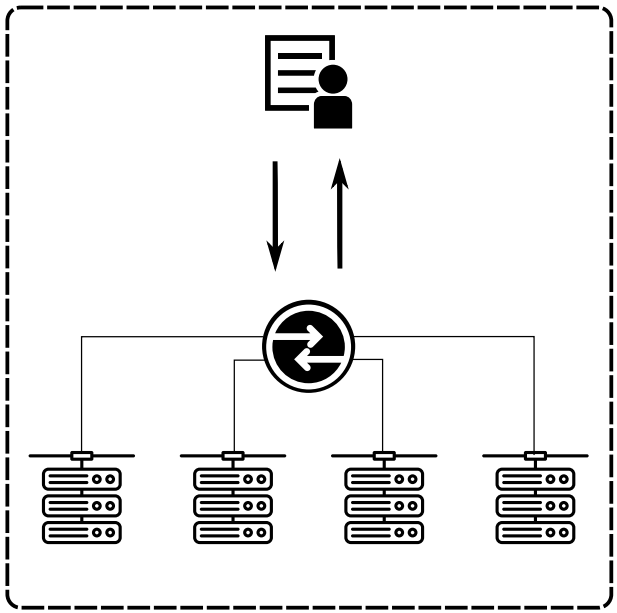
1. State of the Art

2. Systematic Mojette Erasure Code

3. Reprojection without reconstruction

4. Distributed Storage System: RozoFS

5. Conclusion



Outline

1. State of the Art

2. Systematic Mojette Erasure Code

3. Reprojection without reconstruction

4. Distributed Storage System: RozoFS

5. Conclusion

Outline

1. State of the Art

1.1 Distributed storage

1.2 Erasure Coding

2. Systematic Mojette Erasure Code

3. Reprojection without reconstruction

4. Distributed Storage System: RozoFS

5. Conclusion

Outline

1. State of the Art

1.1 Distributed storage

1.2 Erasure Coding

2. Systematic Mojette Erasure Code

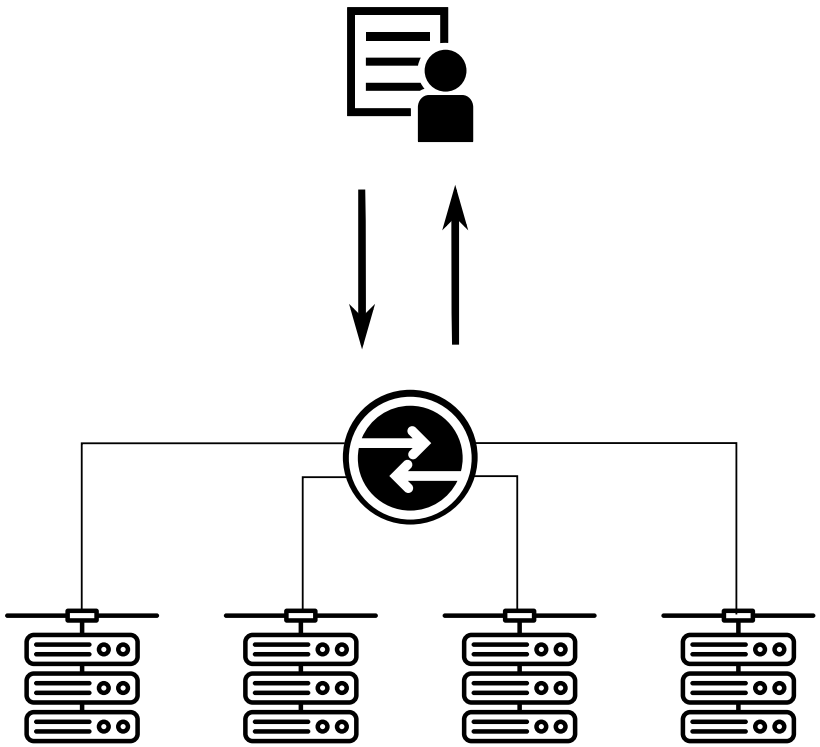
3. Reprojection without reconstruction

4. Distributed Storage System: RozoFS

5. Conclusion

NDSS[2] goals:

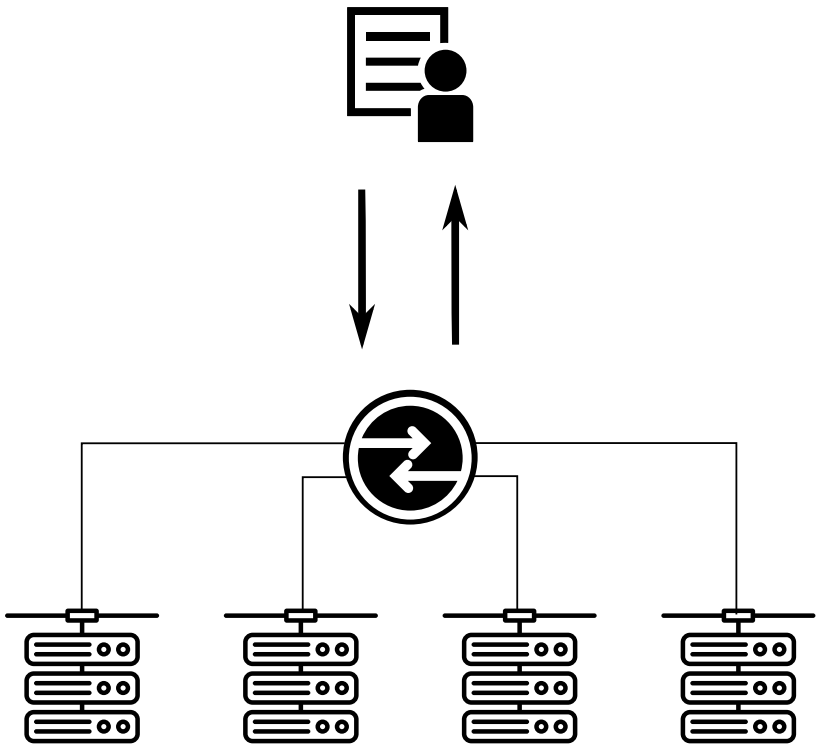
- Scalability
- Fault-tolerance:
  - creation
  - over the time
- Performance
- Financial cost



[2] Network Distributed Storage Systems [Oggier et al., 2012]

NDSS[2] goals:

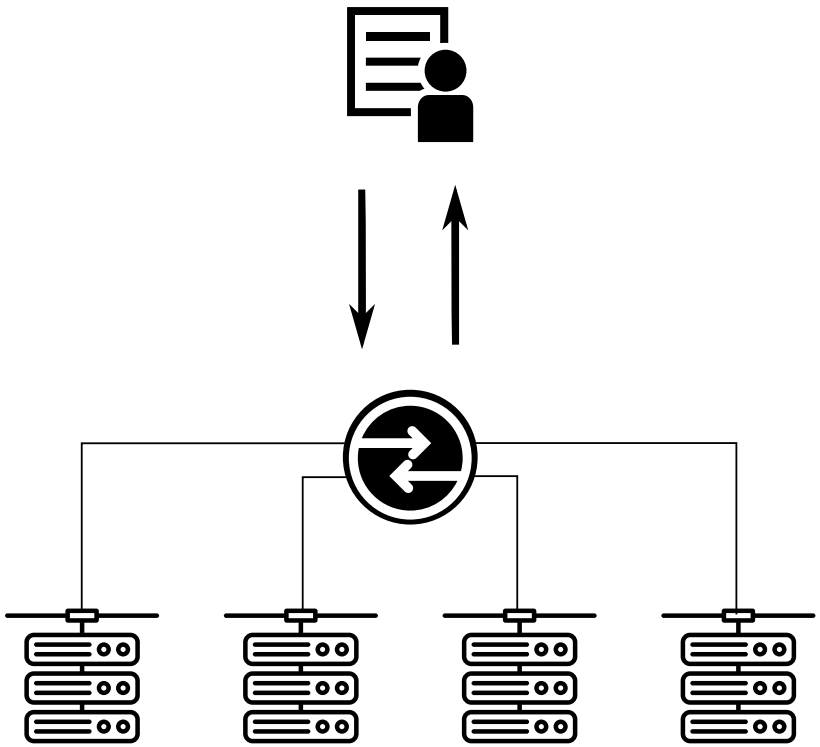
- Scalability
- Fault-tolerance:
  - creation
  - over the time
- Performance
- Financial cost



[2] Network Distributed Storage Systems [Oggier et al., 2012]

NDSS[2] goals:

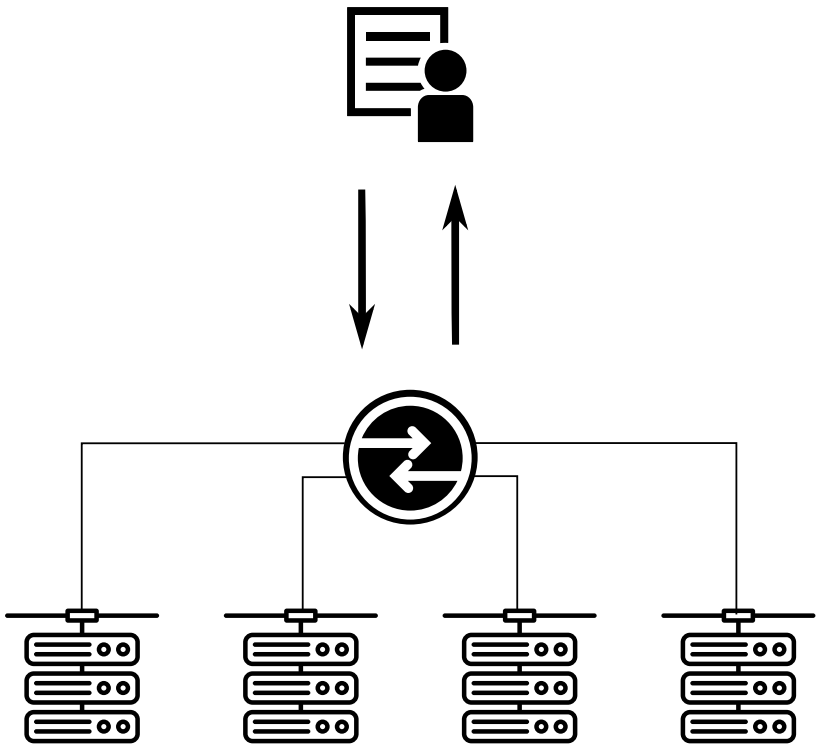
- Scalability
- Fault-tolerance:
  - creation
  - over the time
- Performance
- Financial cost



[2] Network Distributed Storage Systems [Oggier et al., 2012]

NDSS[2] goals:

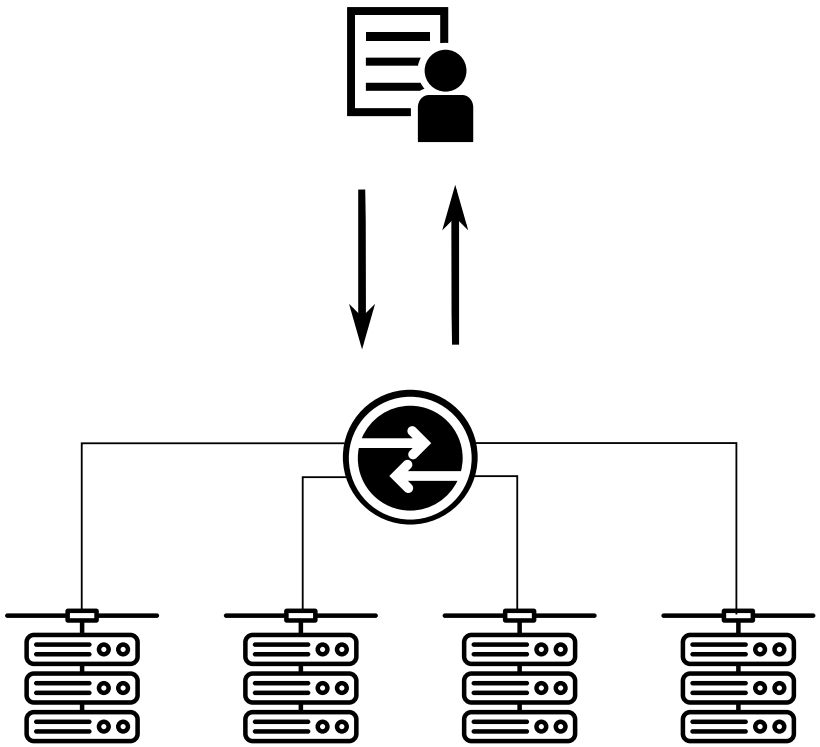
- Scalability
- Fault-tolerance:
  - creation
  - over the time
- Performance
- Financial cost



[2] Network Distributed Storage Systems [Oggier et al., 2012]

NDSS[2] goals:

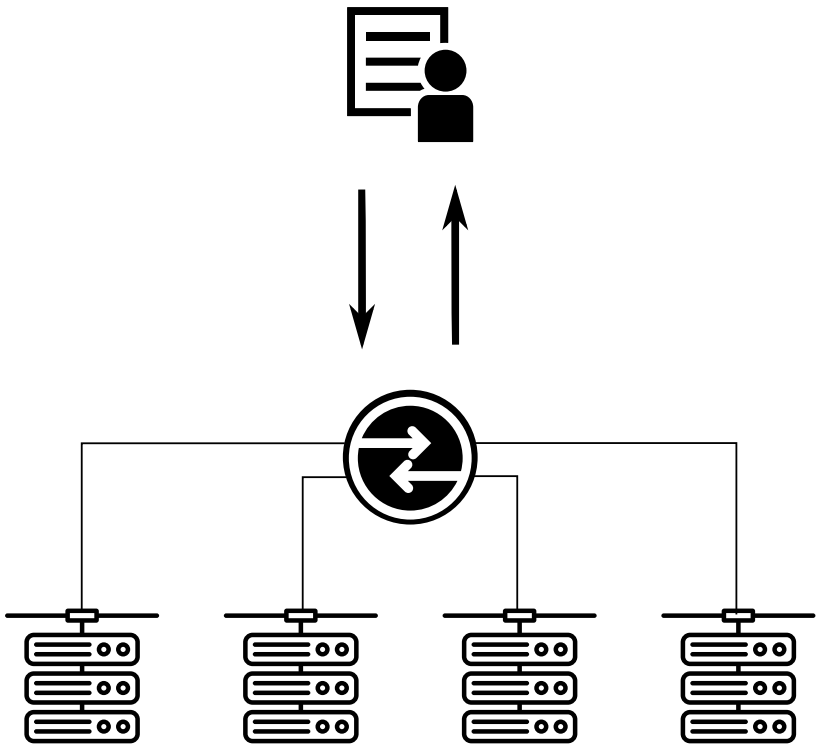
- Scalability
- Fault-tolerance:
  - creation
  - over the time
- Performance
- Financial cost



[2] Network Distributed Storage Systems [Oggier et al., 2012]

NDSS[2] goals:

- Scalability
- Fault-tolerance:
  - creation
  - over the time
- Performance
- Financial cost



[2] Network Distributed Storage Systems [Oggier et al., 2012]

Outline

1. State of the Art

1.1 Distributed storage

1.2 Erasure Coding

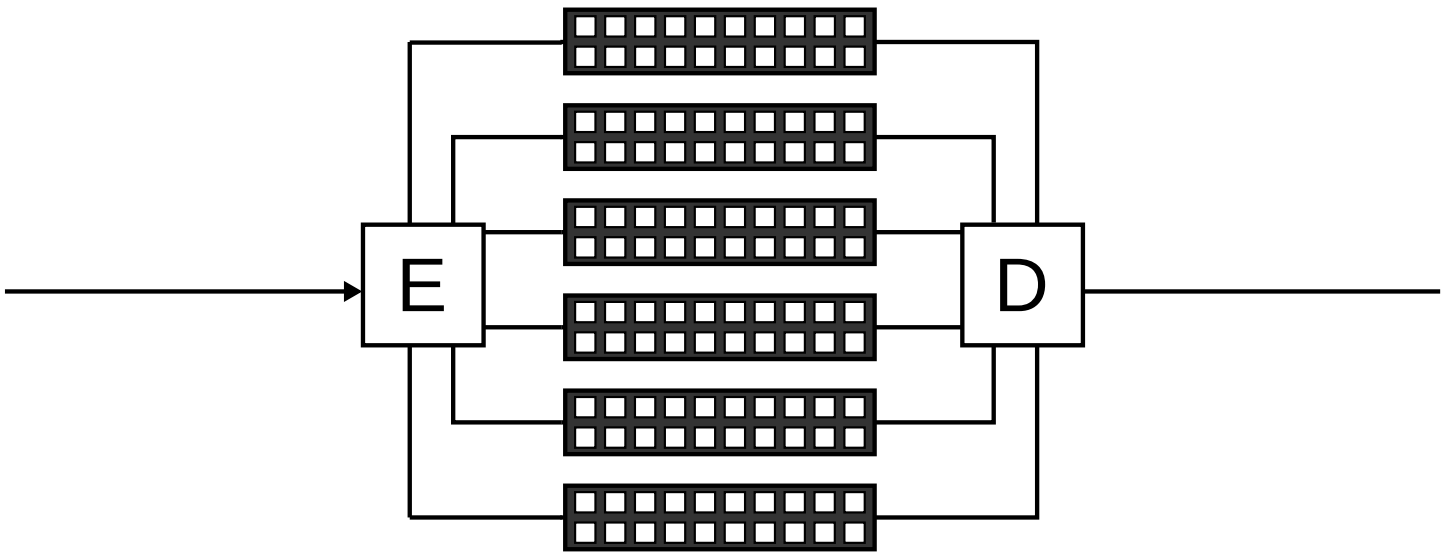
2. Systematic Mojette Erasure Code

3. Reprojection without reconstruction

4. Distributed Storage System: RozoFS

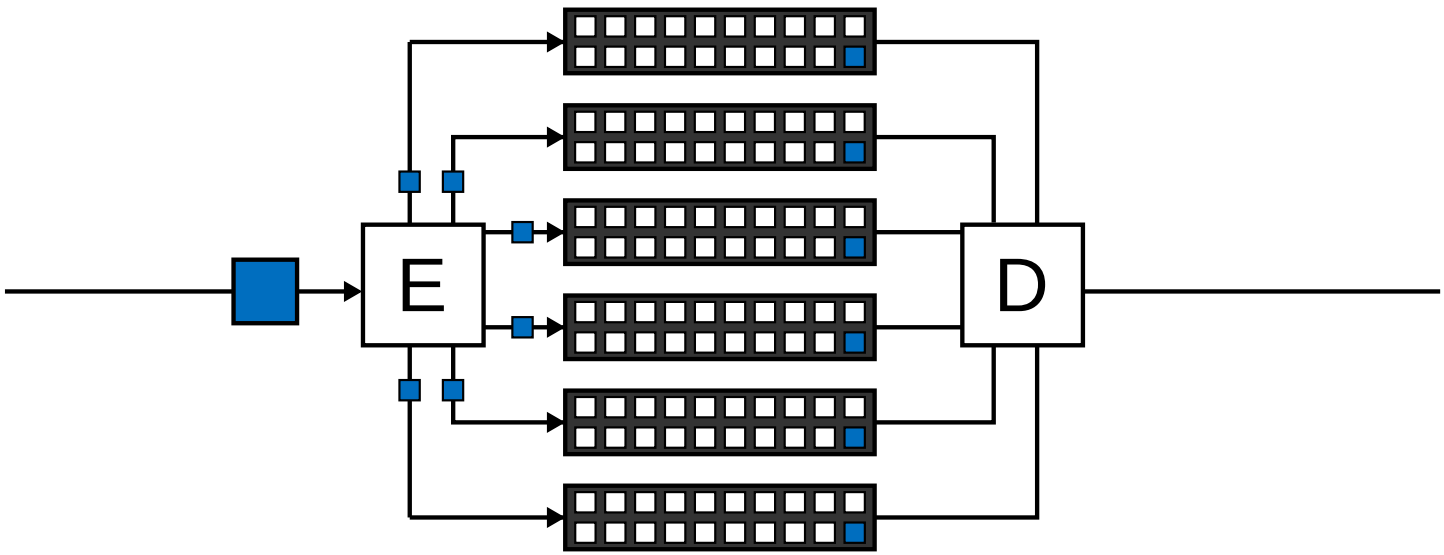
5. Conclusion

Erasure coding: compute redundancy



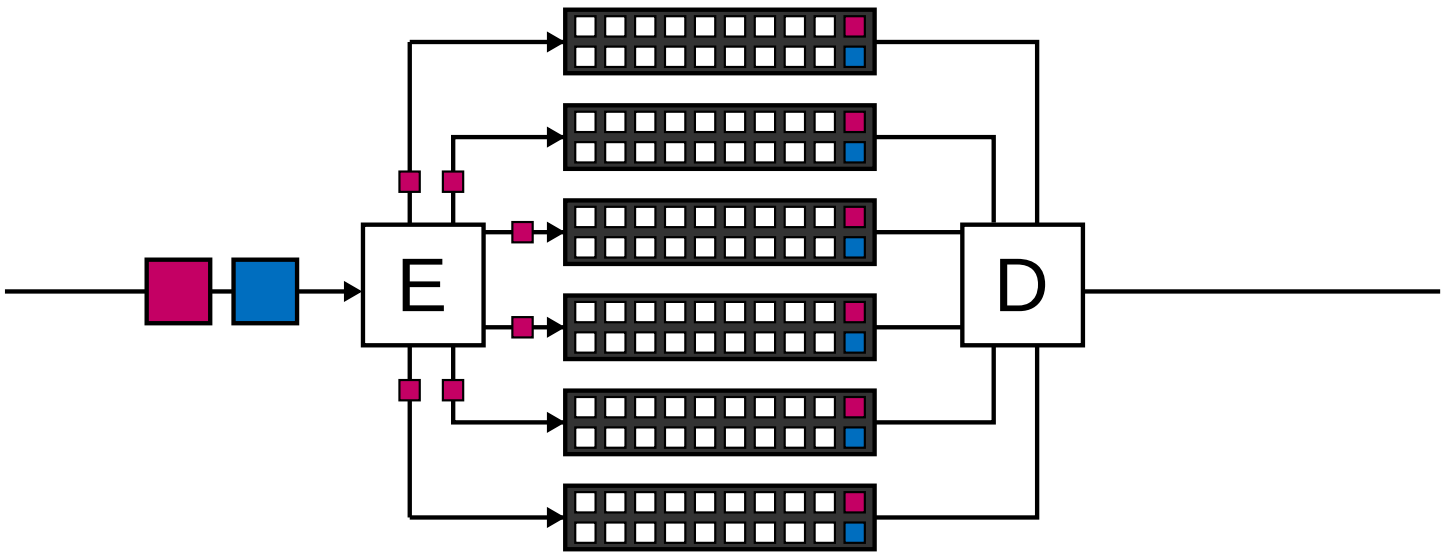
NDSS model with input/output and storage supports

Erasure coding: compute redundancy



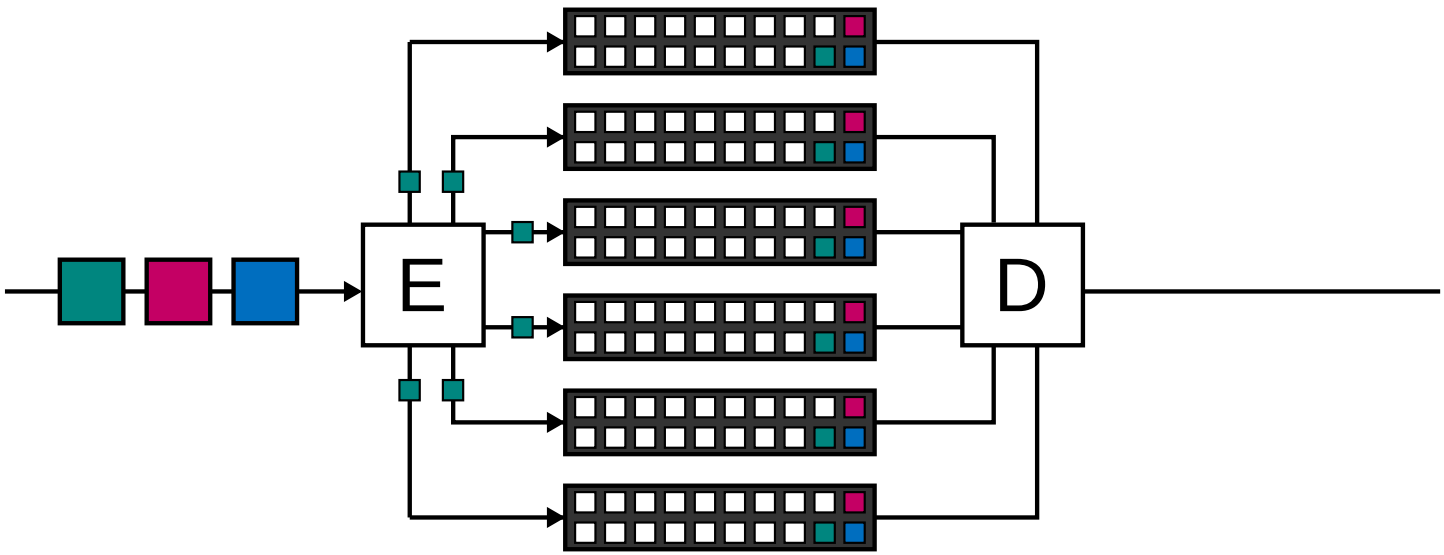
Packets of $\mathscr{M}$ bytes is fragmented and redundant blocks are distributed

Erasure coding: compute redundancy



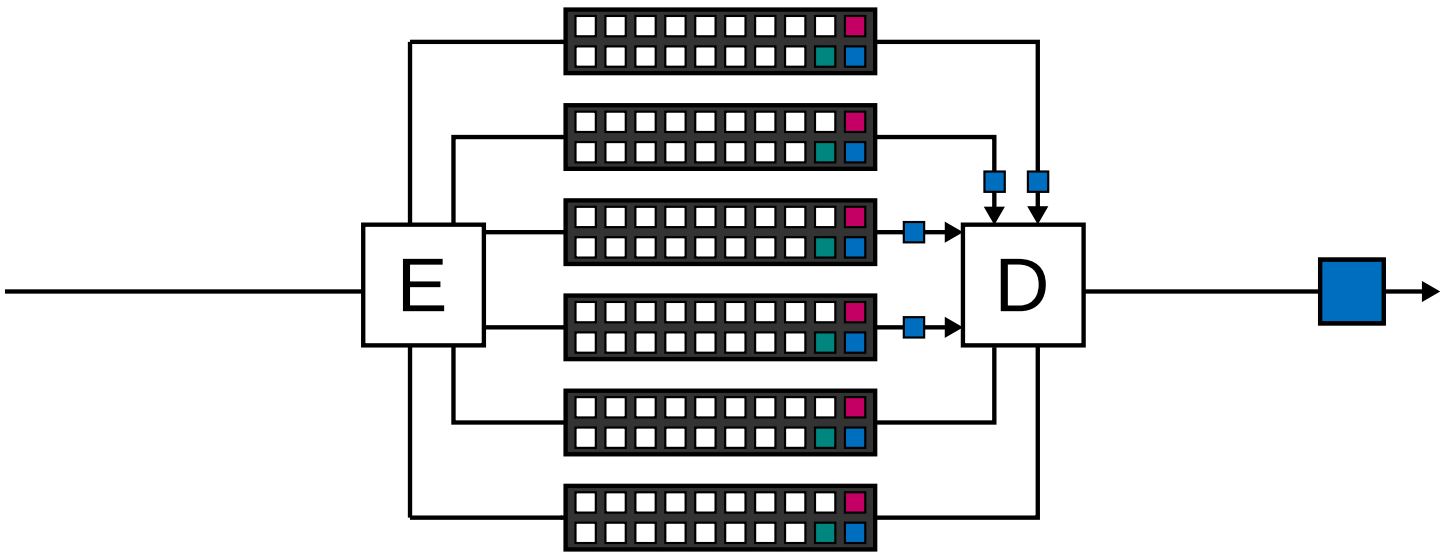
Packets of $\mathscr{M}$ bytes is fragmented and redundant blocks are distributed

Erasure coding: compute redundancy



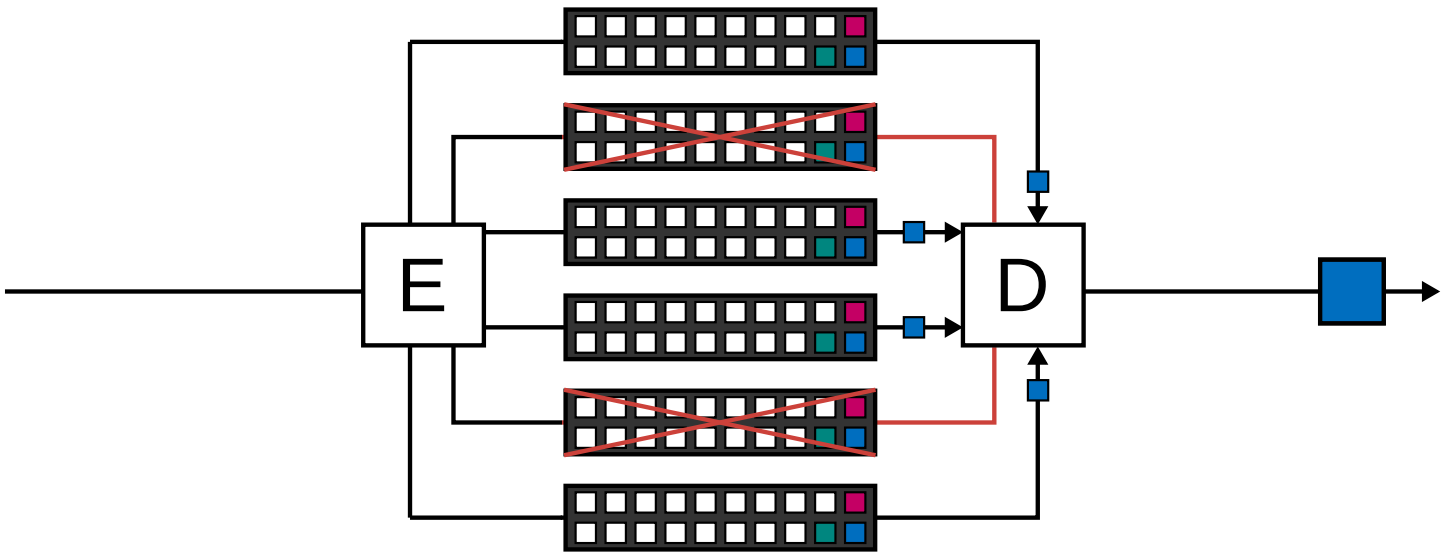
Packets of $\mathscr{M}$ bytes is fragmented and redundant blocks are distributed

Erasure coding: compute redundancy



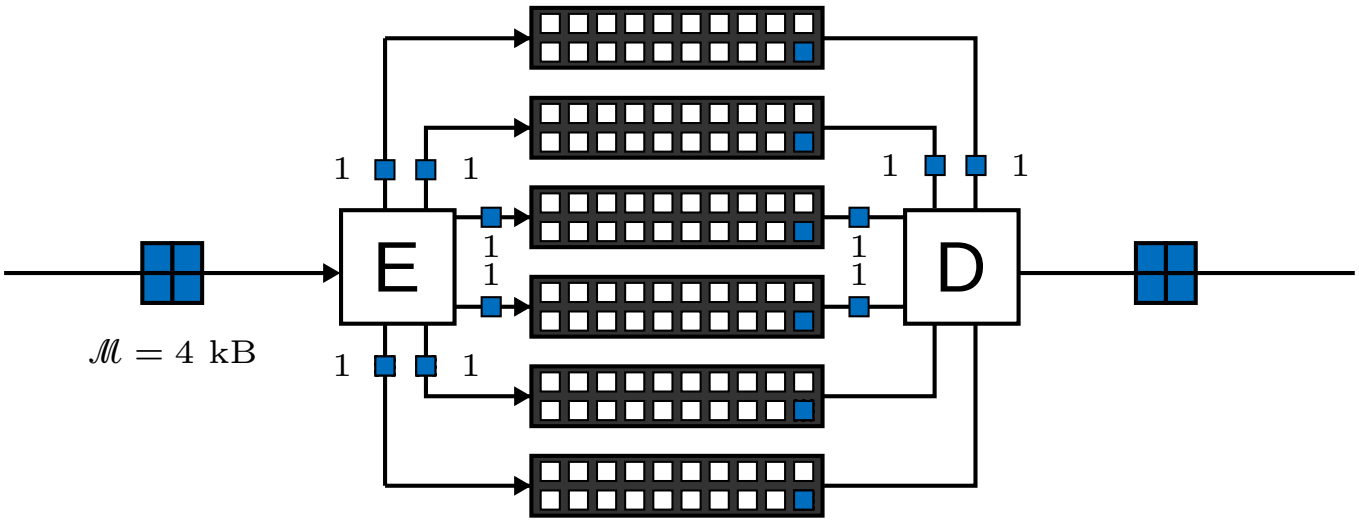
A subset of blocks can rebuild the original packet

Erasure coding: compute redundancy



A subset of blocks can rebuild the original packet

(n, k) MDS codes are optimal

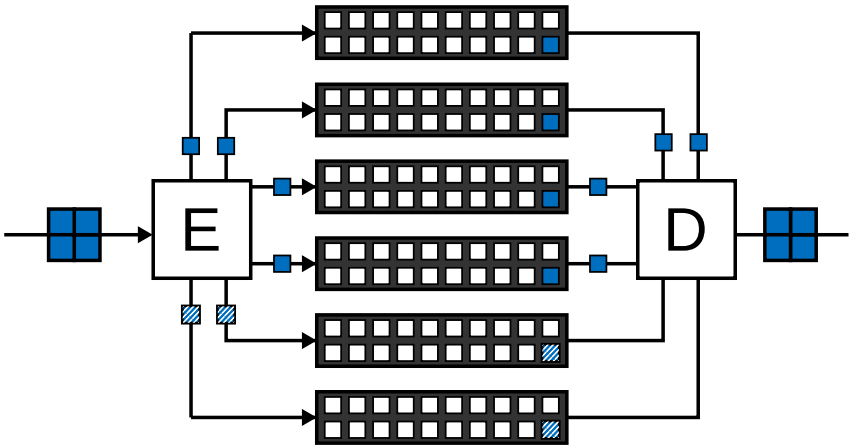


MDS codes properties:

- compute $n = 6$ encoded blocks from $k = 4$ data blocks
  - generate 6×1 kB from $\mathscr{M} = 4$ kB
- optimal: decode from any set of $k = 4$ encoded blocks among $n = 6$ (each block is $\frac{\mathscr{M}}{k}$)

Maximum Distance Separable (MDS) codes [Singleton, 1964]

Systematic vs Non-systematic

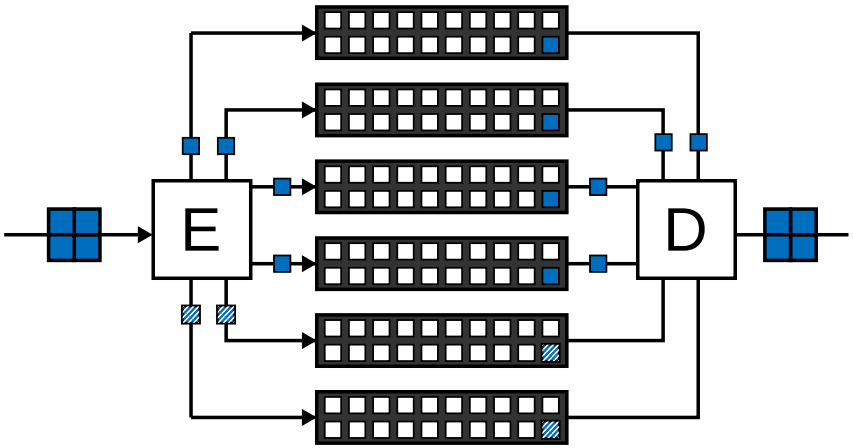


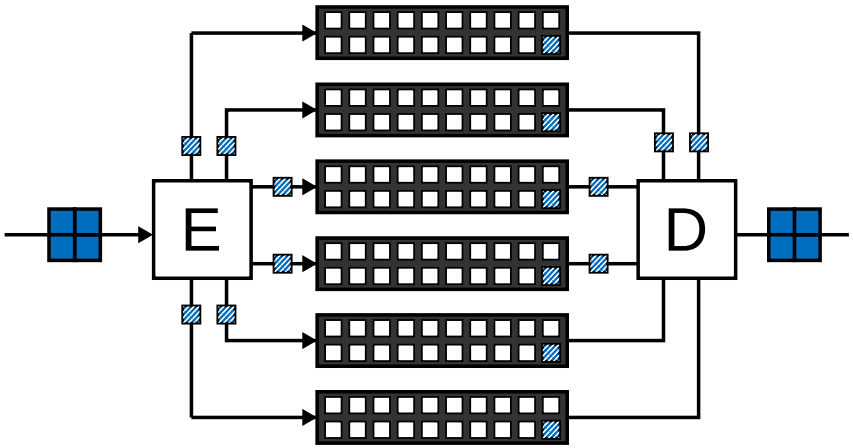
Systematic

- compute $(n-k)$ parity blocks

+ better encoding
+ no decoding if no erasure

Systematic vs Non-systematic





Systematic

- compute $(n-k)$ parity blocks

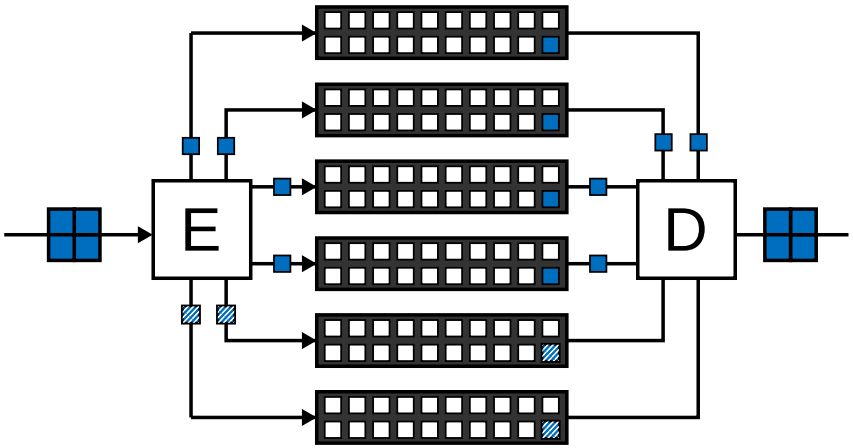
+ better encoding
+ no decoding if no erasure

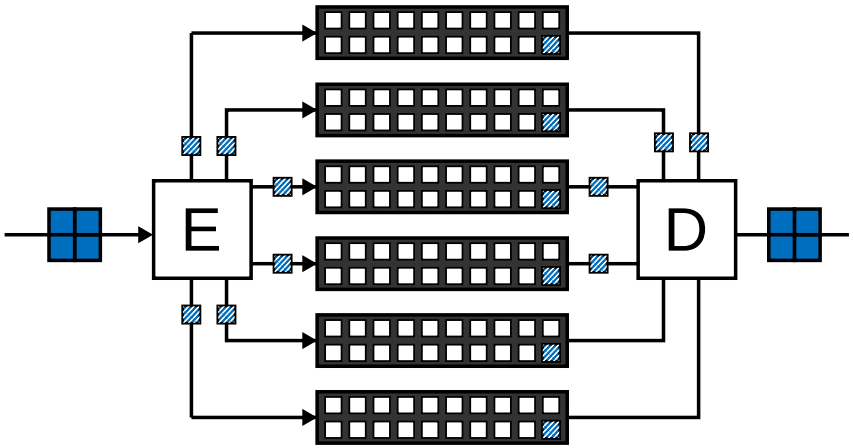
Non-systematic

- computes n encoded blocks

+ encoded blocks not readable
+ same weight for each block

Systematic vs Non-systematic





Systematic

- compute $(n-k)$ parity blocks

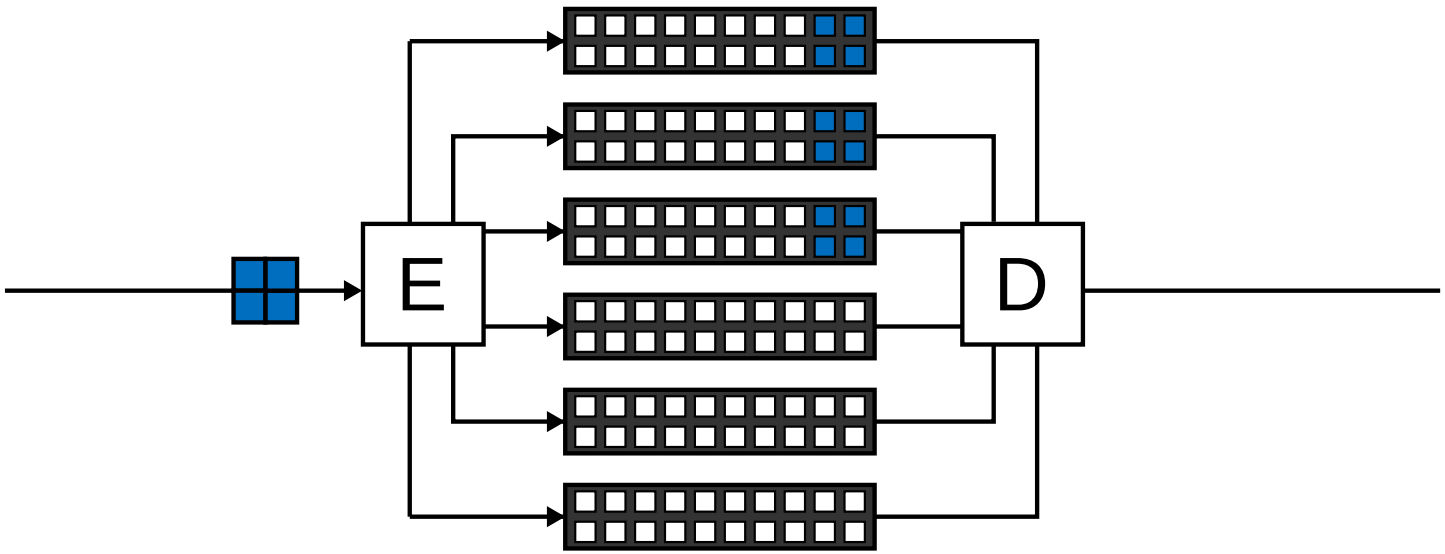
+ better encoding
+ no decoding if no erasure

Non-systematic

- computes n encoded blocks

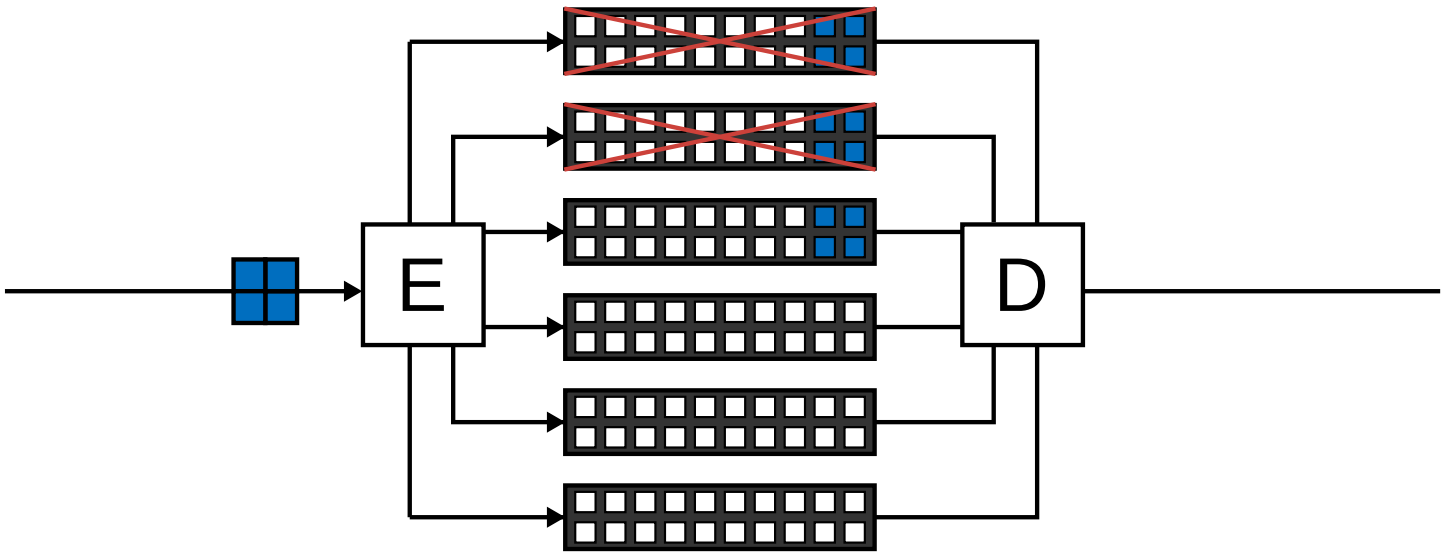
+ encoded blocks not readable
+ same weight for each block

1. $(n, k = 1)$ Repetition code: replication



$(n = 3, k = 1)$ MDS systematic erasure code (repetition code)

1. $(n, k = 1)$ Repetition code: replication

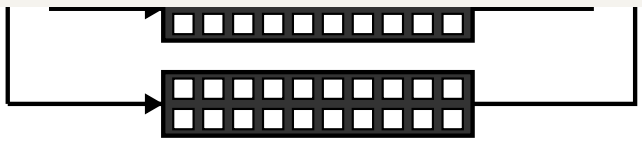


$(n = 3, k = 1)$ MDS systematic erasure code (repetition code)

1. $(n, k = 1)$ Repetition code: replication

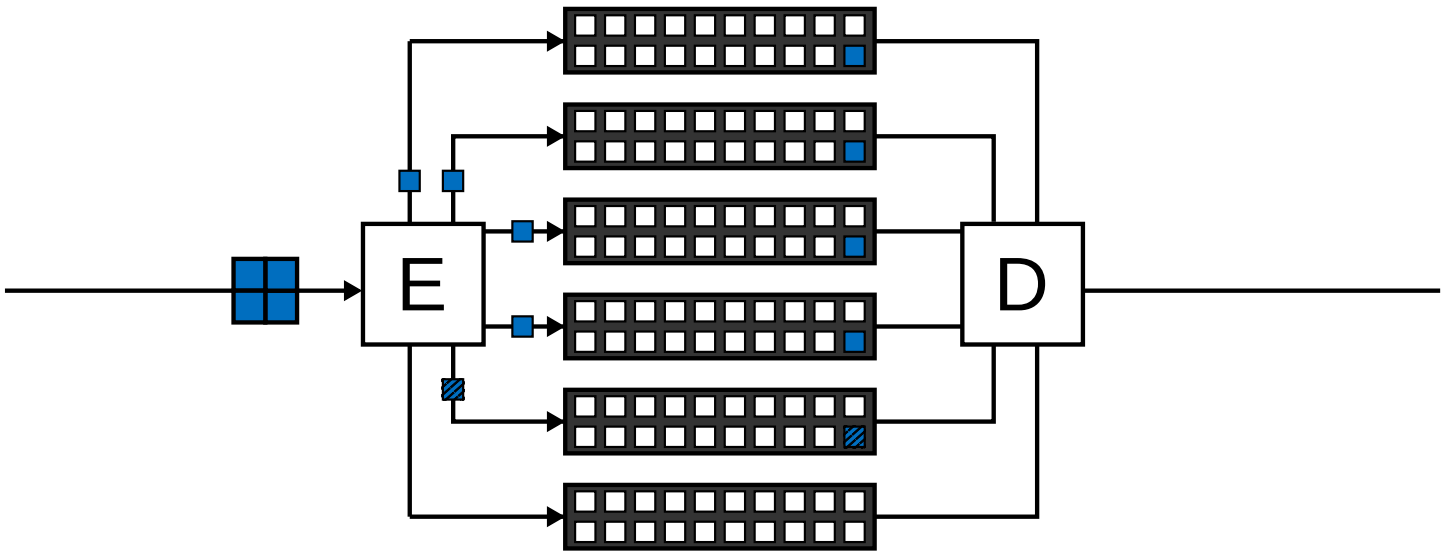
Repetition code:

- copy is fast
- but stores a lot of redundancy (200%)



$(n = 3, k = 1)$ MDS systematic erasure code (repetition code)

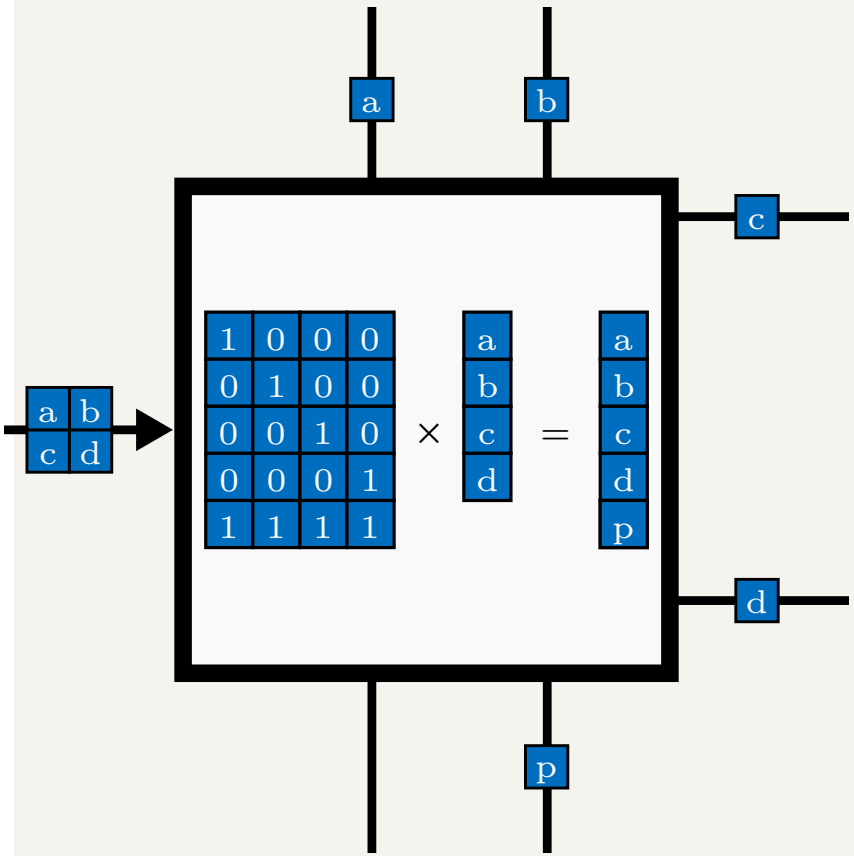
2. $(n, k = n - 1)$ Parity code: single parity



$(n = 5, k = 4)$ MDS systematic erasure code (parity code)

2. $(n, k = n - 1)$ Parity code: single parity

Parity encoding:



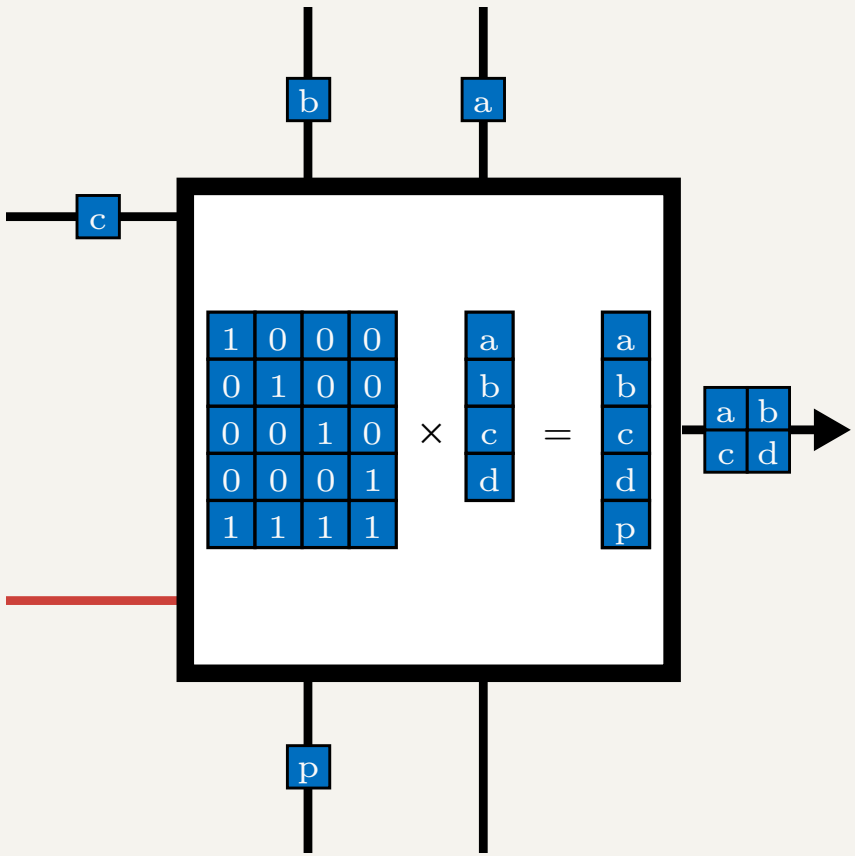
XOR-based computations:

- $p = a \oplus b \oplus c \oplus d$

Parity decoding:

XOR-based computations:

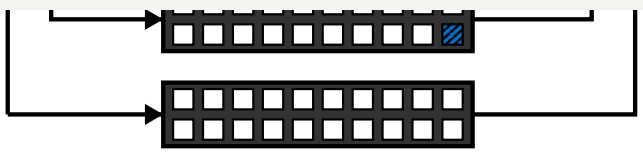
- $d = a \oplus b \oplus c \oplus p$



2. $(n, k = n - 1)$ Parity code: single parity

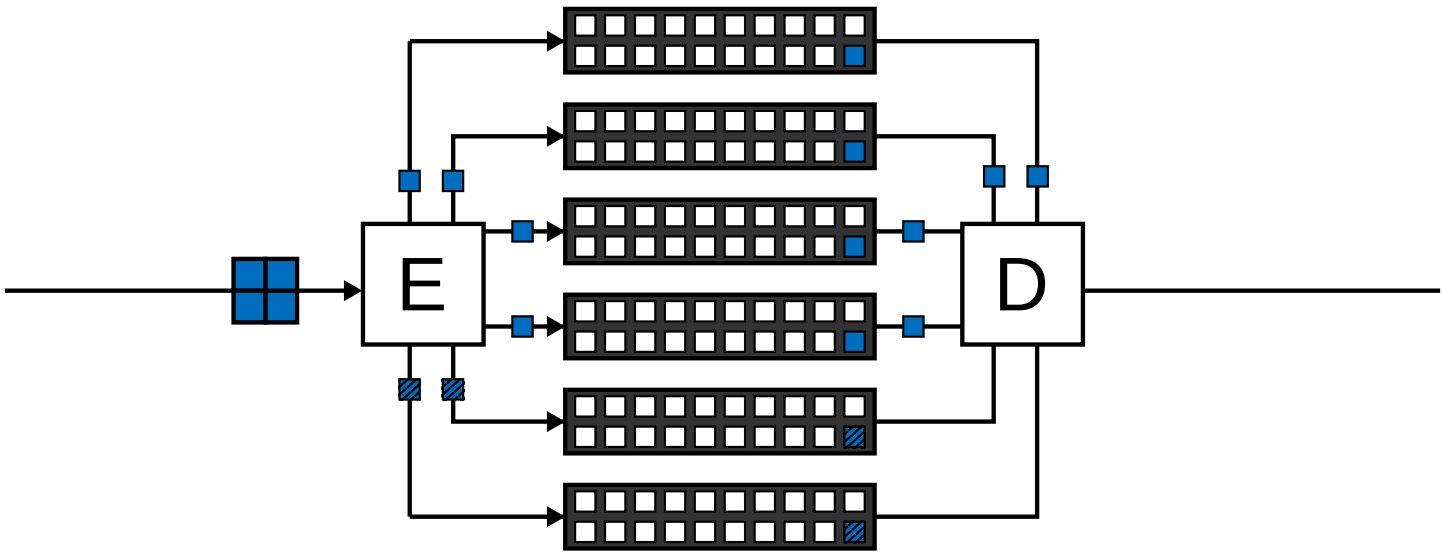
Parity code:

- minimum redundancy and performs well
- but tolerance limited to one failure



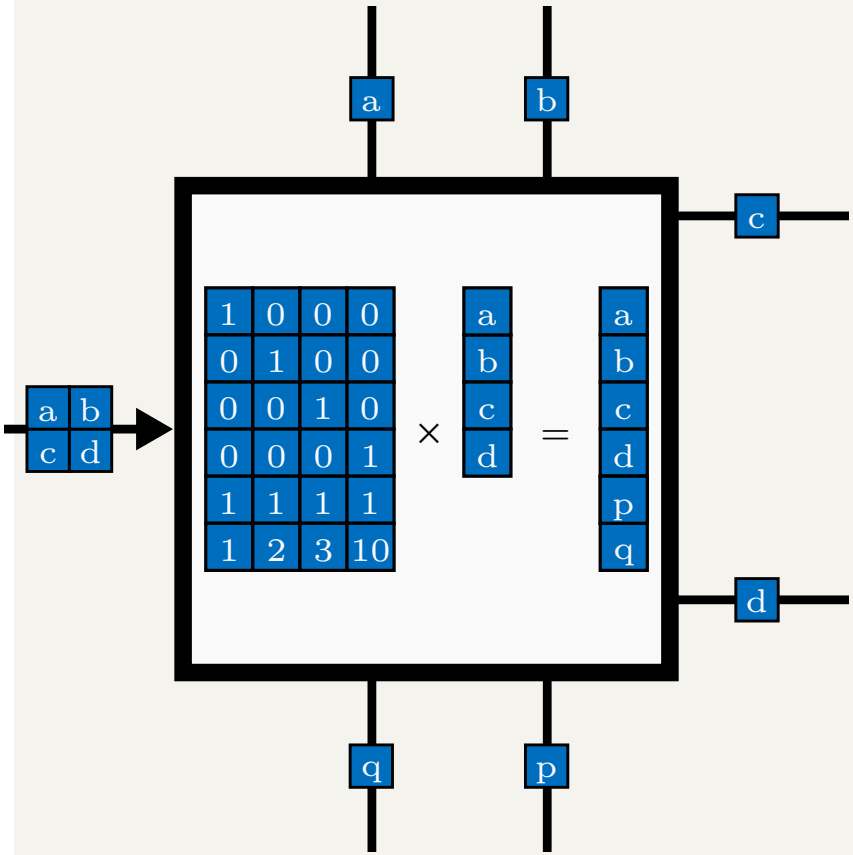
$(n = 5, k = 4)$ MDS systematic erasure code (parity code)

3. (n, k) Reed-Solomon codes: multiple parity



$(n = 6, k = 4)$ MDS systematic erasure code (Reed-Solomon code)

Reed-Solomon encoding:



- $p = a + b + c + d$
- $q = 1 \times a + 2 \times b + 3 \times c + 10 \times d$

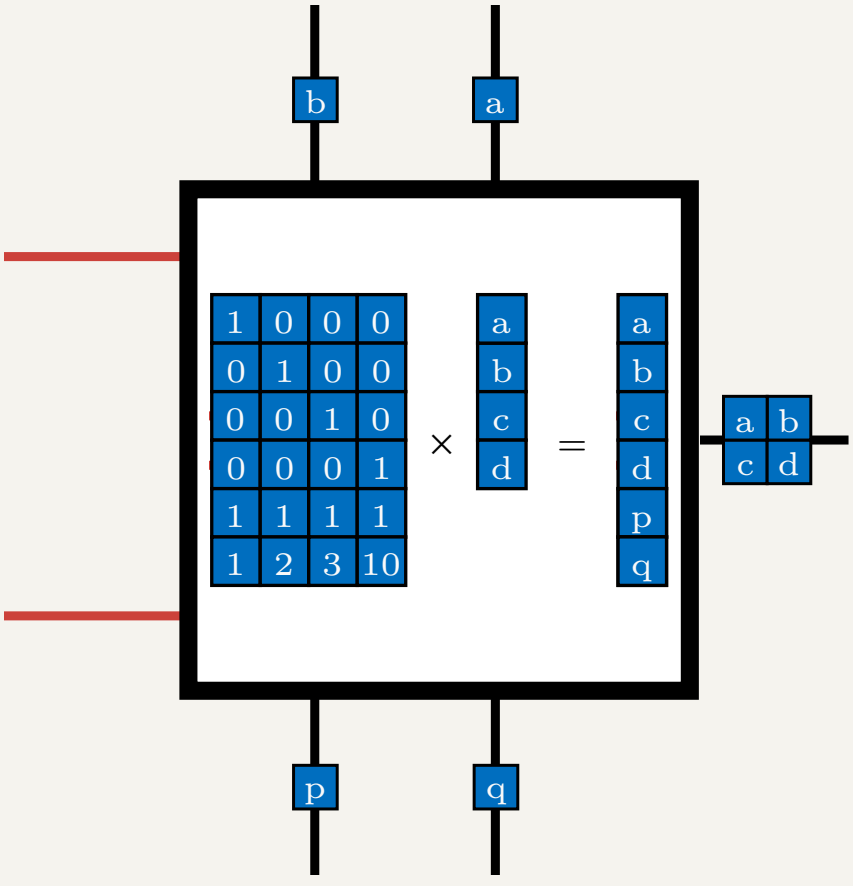
Coefficients extracted from invertible matrix:

- Vandermonde [Rizzo, 1997]
- Cauchy [Blömer et al., 1995]

[Reed et al., 1960]

Reed-Solomon decoding:

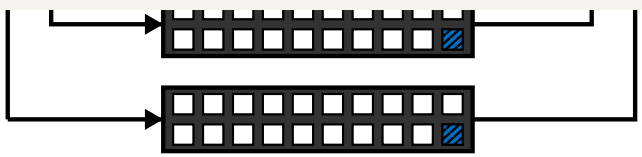
- $p = a + b + c + d$
- $q = 1 \times a + 2 \times b + 3 \times c + 10 \times d$



[Reed et al., 1960]

Reed-Solomon codes:

- generic parameters and minimum redundancy (50%)
- but more complicated encoding and decoding processes



$(n = 6, k = 4)$ MDS systematic erasure code (Reed-Solomon code)

Problems to solve

Design an erasure code, providing redundancy, that:

1. is MDS (or near-MDS), minimizing the redundancy
2. is systematic to enhance encoding and decoding operations

Design a mechanism to compute extra encoded blocks

Embed its erasure code in a practical distributed storage system

Problems to solve

Design an erasure code, providing redundancy, that:

1. is MDS (or near-MDS), minimizing the redundancy
2. is systematic to enhance encoding and decoding operations

Design a mechanism to compute extra encoded blocks

Embed its erasure code in a practical distributed storage system

Problems to solve

Design an erasure code, providing redundancy, that:

1. is MDS (or near-MDS), minimizing the redundancy
2. is systematic to enhance encoding and decoding operations

Design a mechanism to compute extra encoded blocks

Embed its erasure code in a practical distributed storage system

Problems to solve

Design an erasure code, providing redundancy, that:

1. is MDS (or near-MDS), minimizing the redundancy
2. is systematic to enhance encoding and decoding operations

Design a mechanism to compute extra encoded blocks

Embed its erasure code in a practical distributed storage system

Outline

1. State of the Art

2. Systematic Mojette Erasure Code

2.1 Mojette erasure code

2.2 Systematic version

2.3 Evaluations

3. Reprojection without reconstruction

4. Distributed Storage System: RozoFS

5. Conclusion

Outline

1. State of the Art

2. Systematic Mojette Erasure Code

2.1 Mojette erasure code

2.2 Systematic version

2.3 Evaluations

3. Reprojection without reconstruction

4. Distributed Storage System: RozoFS

5. Conclusion

Forward Mojette transform

P

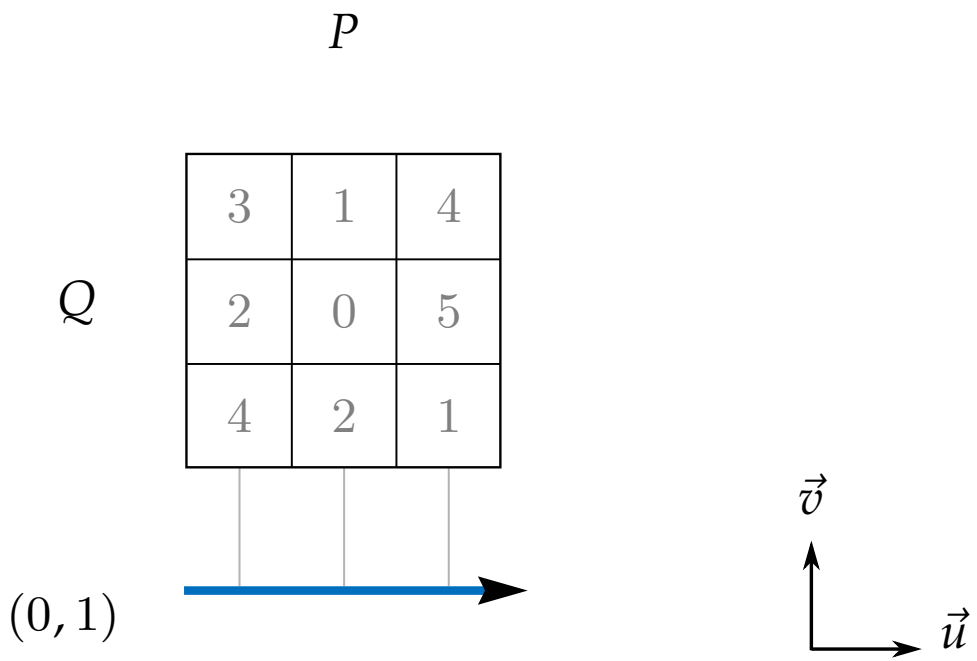
Q

3	1	4
2	0	5
4	2	1

$f(k,l)$

[Guédon et al., 1995]

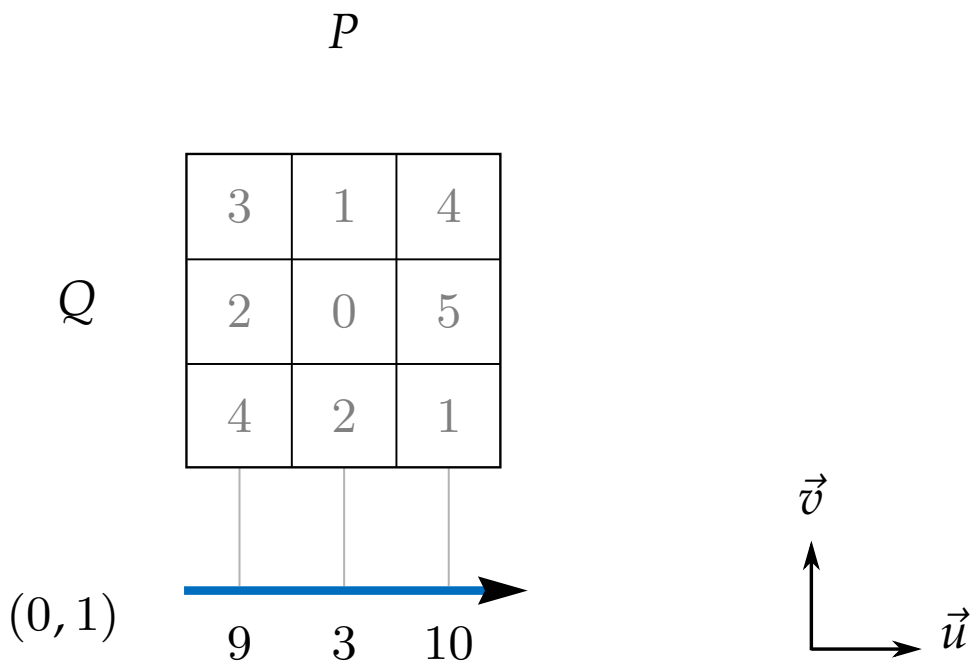
Forward Mojette transform



$$(p = 0, q = 1)$$

[Guédon et al., 1995]

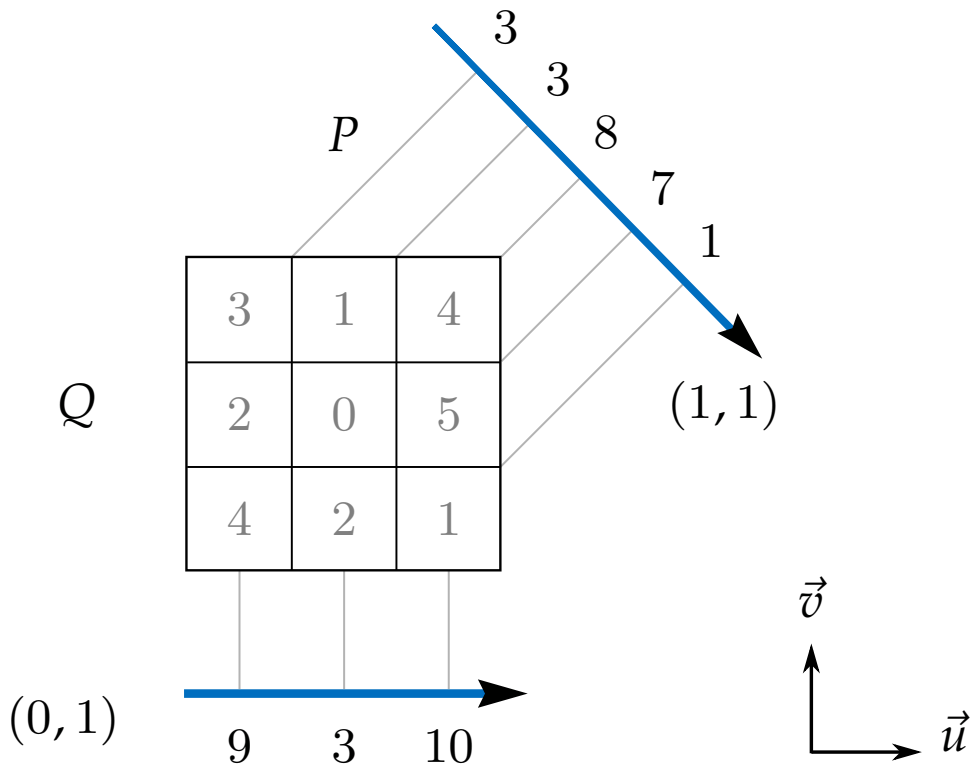
Forward Mojette transform



$$[\mathcal{M}f](b, p_i, q_i) = \sum_{k=0}^{Q-1} \sum_{l=0}^{P-1} f(k,l)\, \Delta(b + kq_i - lp_i)$$

[Guédon et al., 1995]

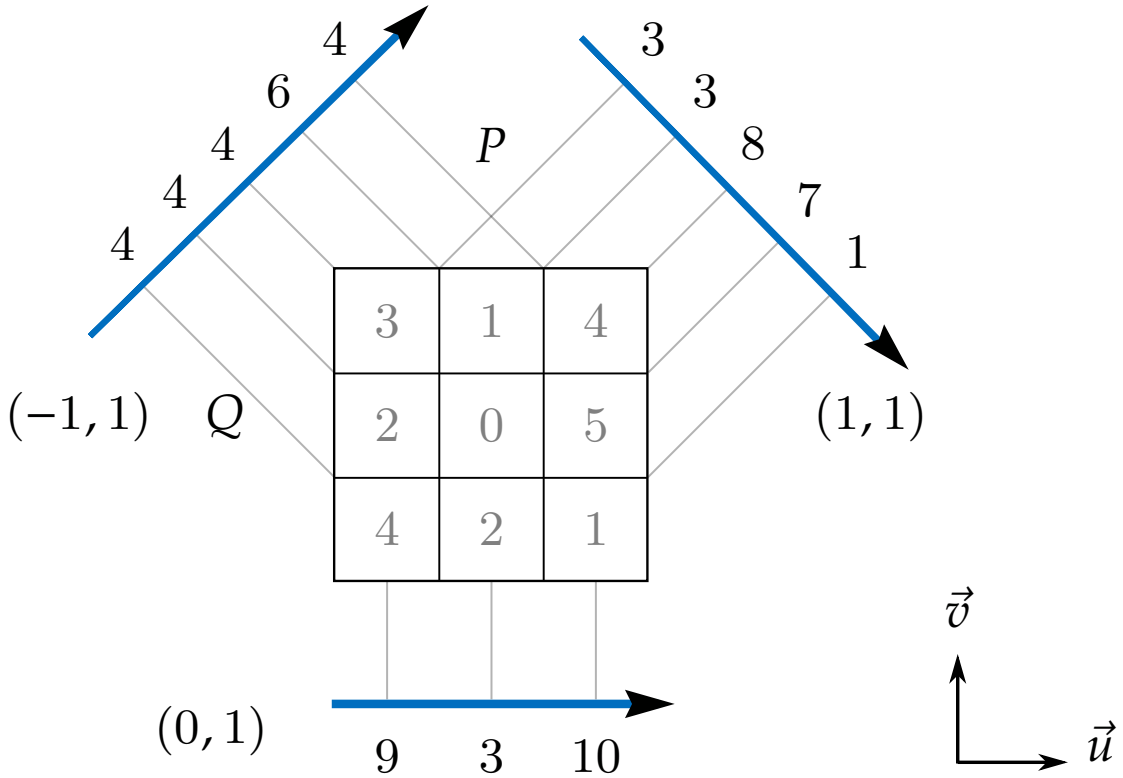
Forward Mojette transform



$$[\mathcal{M}f](b, p_i, q_i) = \sum_{k=0}^{Q-1} \sum_{l=0}^{P-1} f(k, l)\, \Delta(b + kq_i - lp_i)$$

[Guédon et al., 1995]

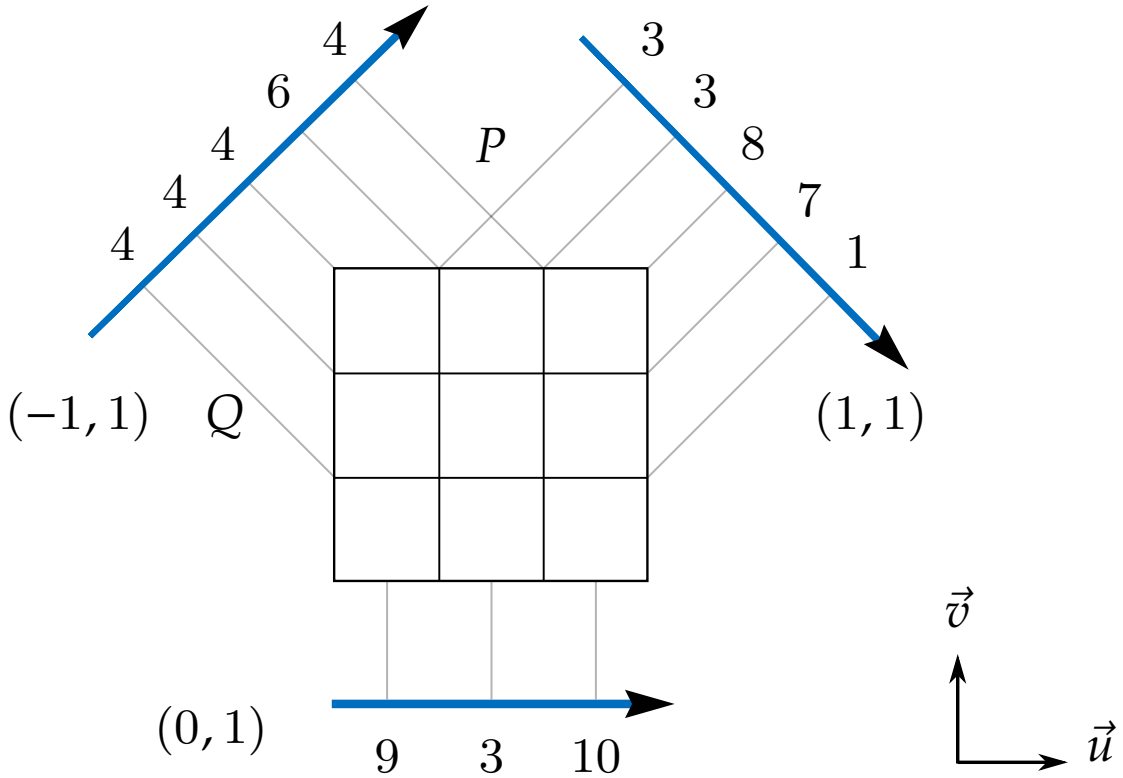
Forward Mojette transform



$$[\mathcal{M}f](b, p_i, q_i) = \sum_{k=0}^{Q-1} \sum_{l=0}^{P-1} f(k, l)\, \Delta(b + kq_i - lp_i)$$

[Guédon et al., 1995]

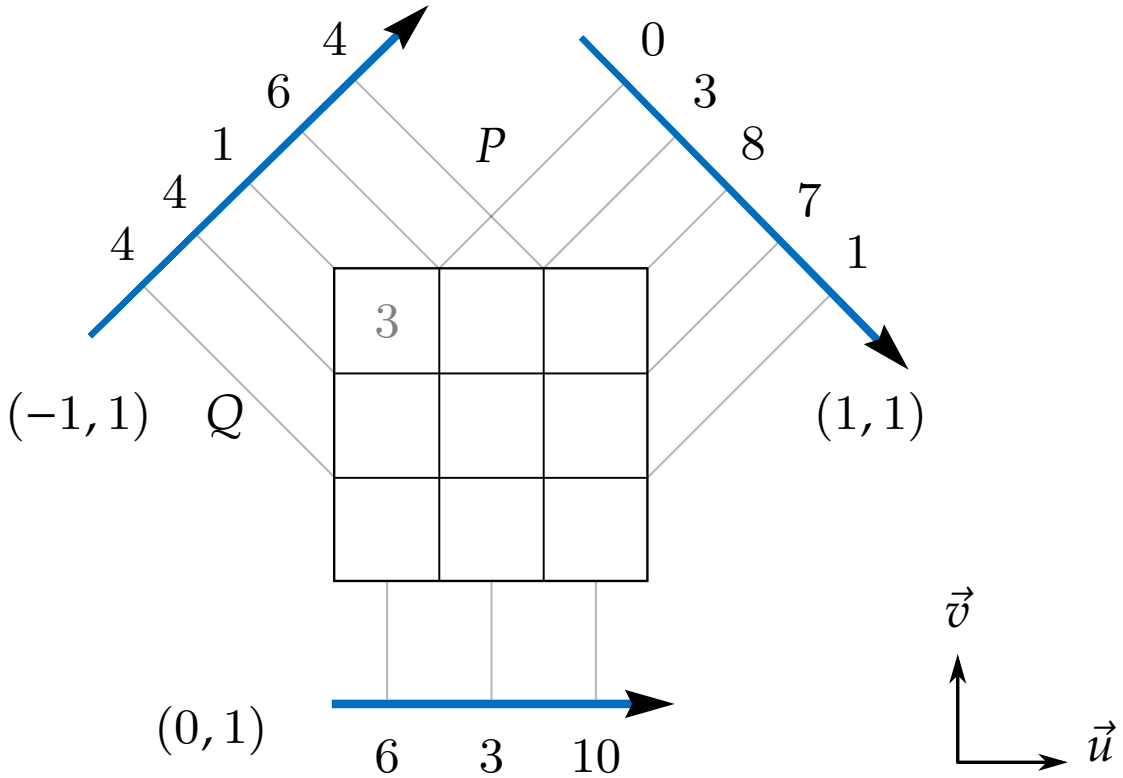
Inverse Mojette transform



$$[\mathcal{M}f](b, p_i, q_i) = \sum_{k=0}^{Q-1} \sum_{l=0}^{P-1} f(k, l)\, \Delta(b + kq_i - lp_i)$$

[Normand et al., 2006]

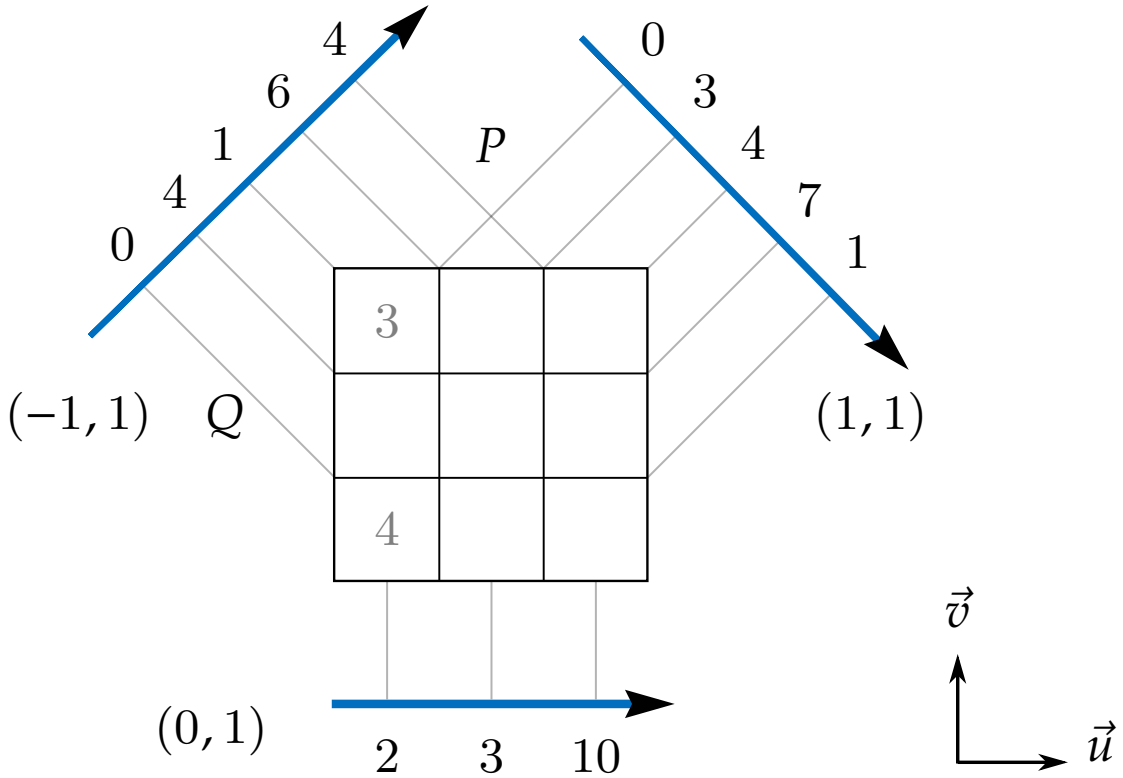
Inverse Mojette transform



$$[\mathcal{M}f](b, p_i, q_i) = \sum_{k=0}^{Q-1} \sum_{l=0}^{P-1} f(k, l)\, \Delta(b + kq_i - lp_i)$$

[Normand et al., 2006]

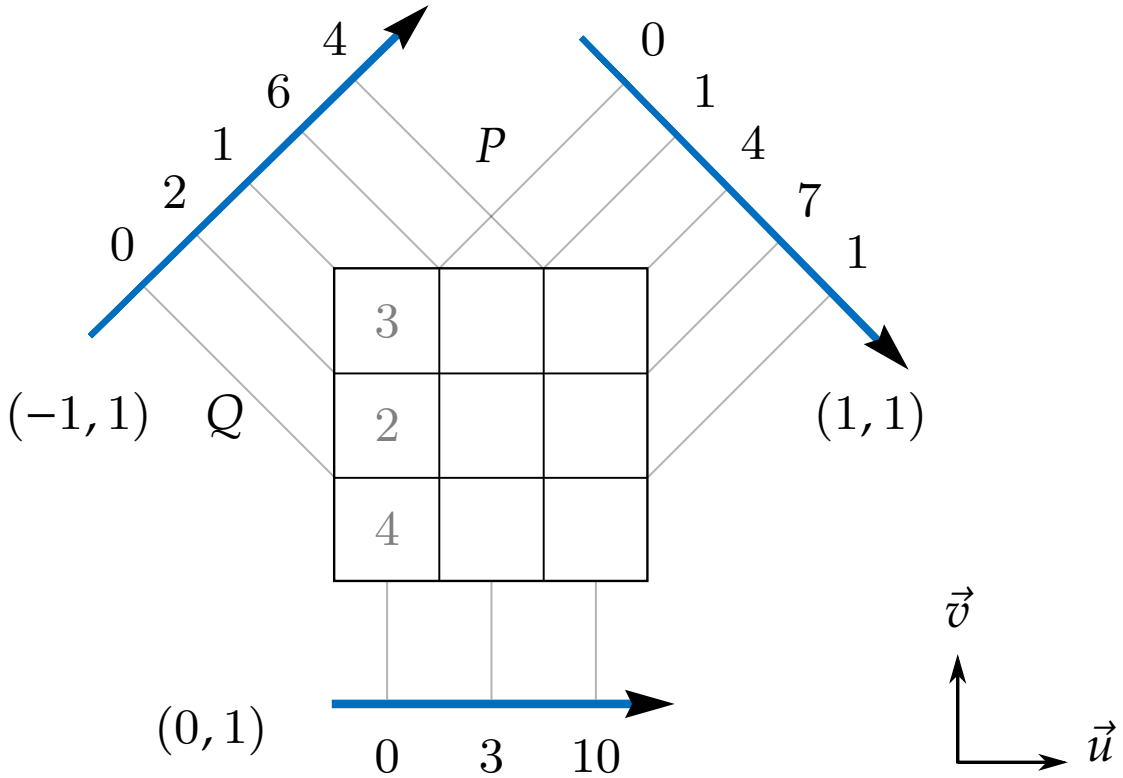
Inverse Mojette transform



$$[\mathcal{M}f](b, p_i, q_i) = \sum_{k=0}^{Q-1} \sum_{l=0}^{P-1} f(k, l)\, \Delta(b + kq_i - lp_i)$$

[Normand et al., 2006]

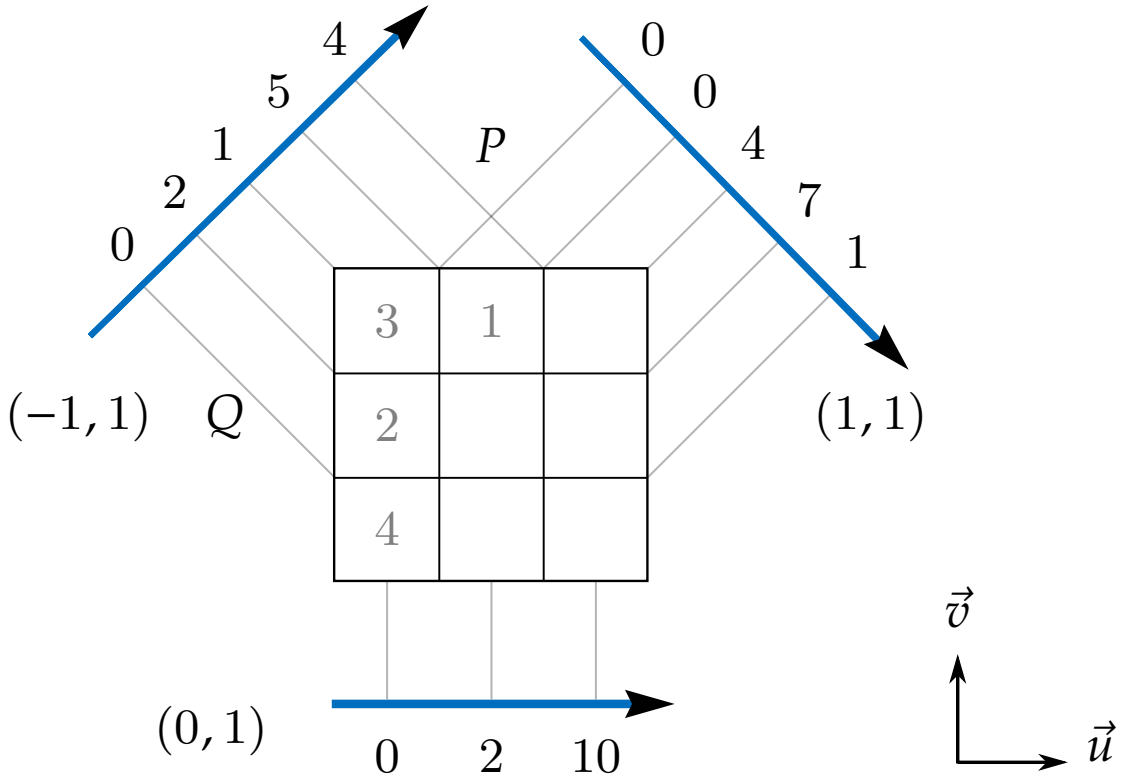
Inverse Mojette transform



$$[\mathcal{M}f](b, p_i, q_i) = \sum_{k=0}^{Q-1} \sum_{l=0}^{P-1} f(k, l)\, \Delta(b + kq_i - lp_i)$$

[Normand et al., 2006]

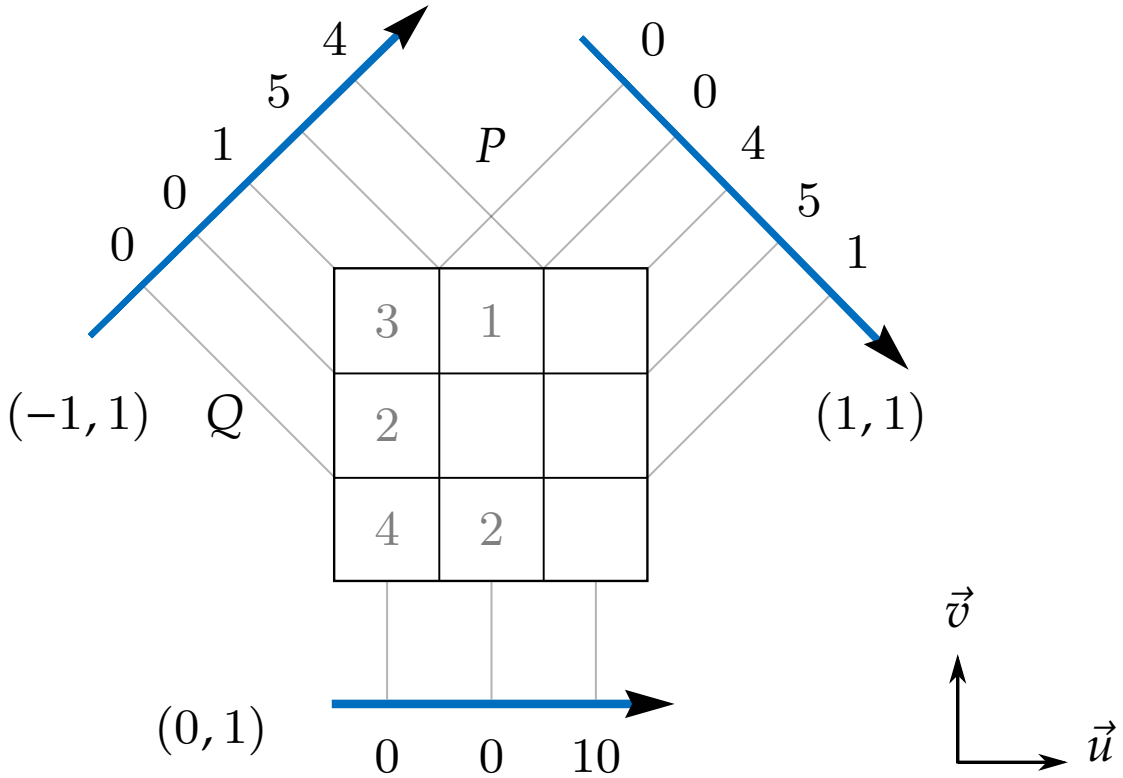
Inverse Mojette transform



$$[\mathcal{M}f](b, p_i, q_i) = \sum_{k=0}^{Q-1} \sum_{l=0}^{P-1} f(k, l)\, \Delta(b + kq_i - lp_i)$$

[Normand et al., 2006]

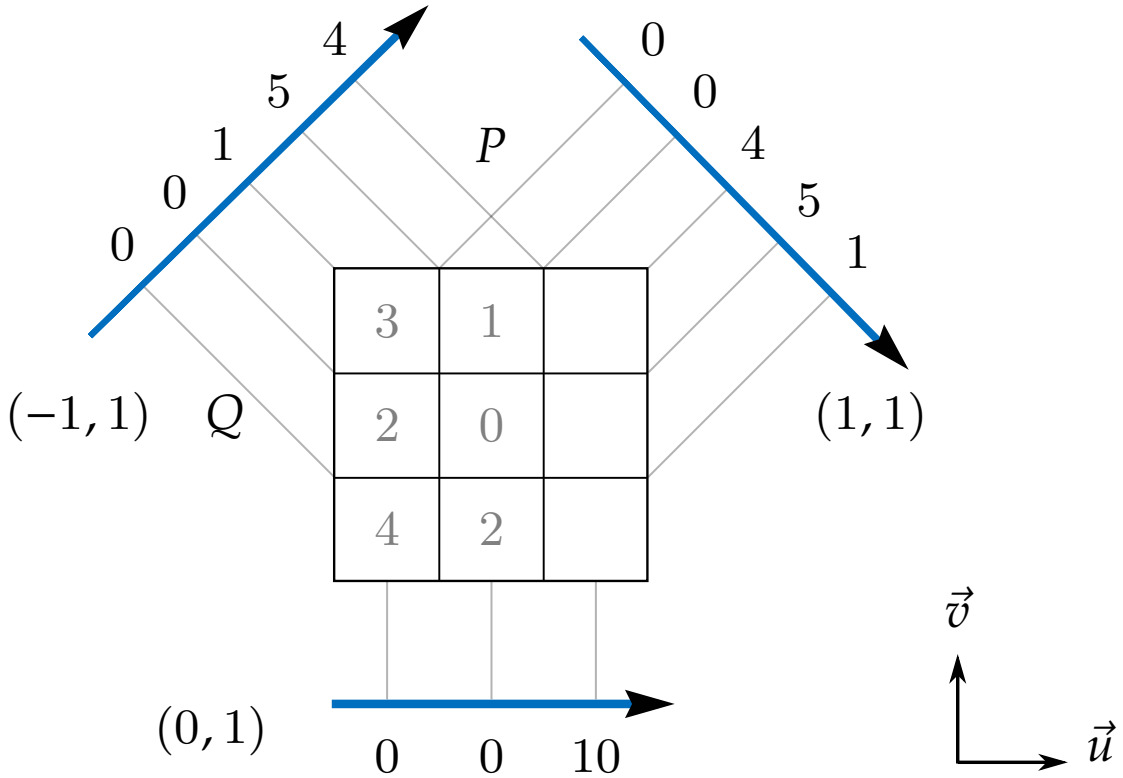
Inverse Mojette transform



$$[\mathcal{M}f](b, p_i, q_i) = \sum_{k=0}^{Q-1} \sum_{l=0}^{P-1} f(k, l)\, \Delta(b + kq_i - lp_i)$$

[Normand et al., 2006]

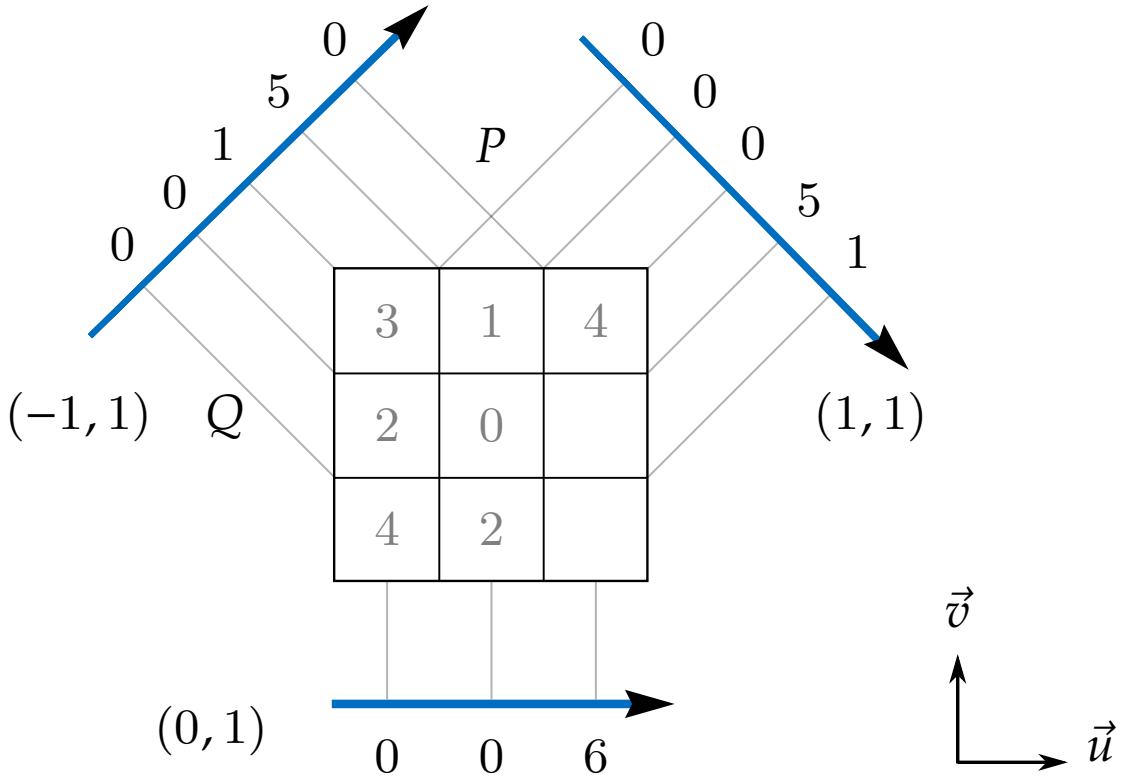
Inverse Mojette transform



$$[\mathcal{M}f](b, p_i, q_i) = \sum_{k=0}^{Q-1} \sum_{l=0}^{P-1} f(k, l)\, \Delta(b + kq_i - lp_i)$$

[Normand et al., 2006]

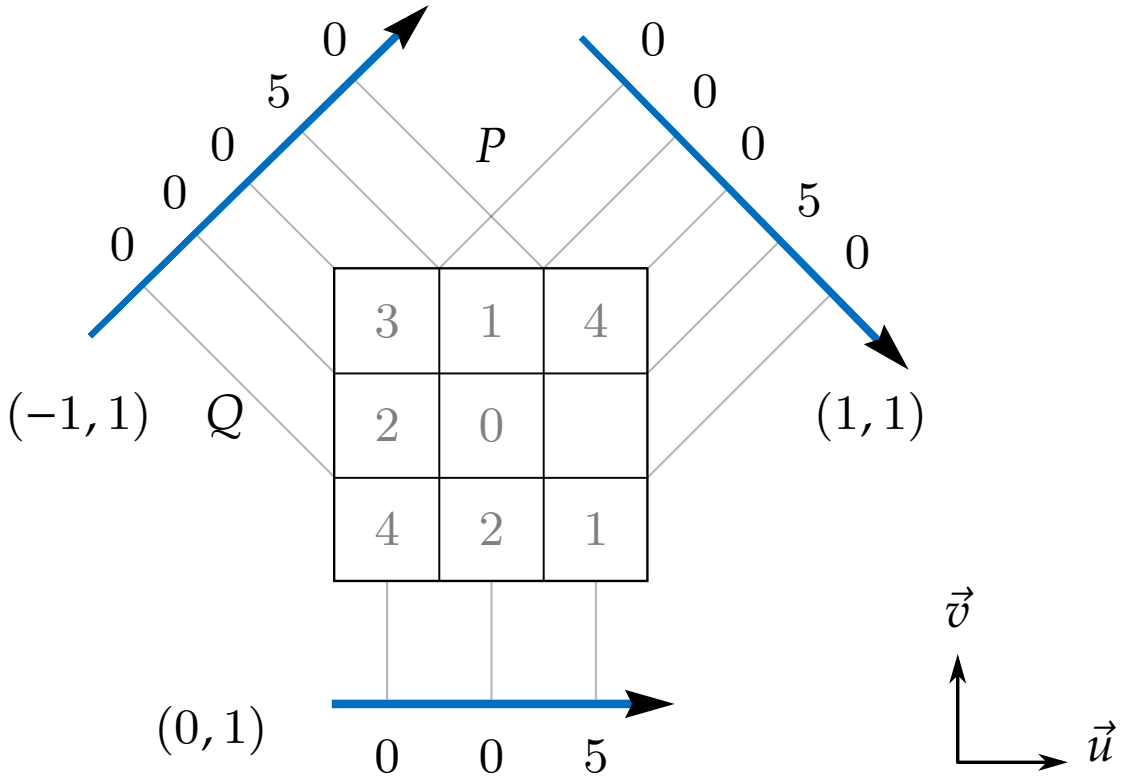
Inverse Mojette transform



$$[\mathcal{M}f](b, p_i, q_i) = \sum_{k=0}^{Q-1} \sum_{l=0}^{P-1} f(k, l)\, \Delta(b + kq_i - lp_i)$$

[Normand et al., 2006]

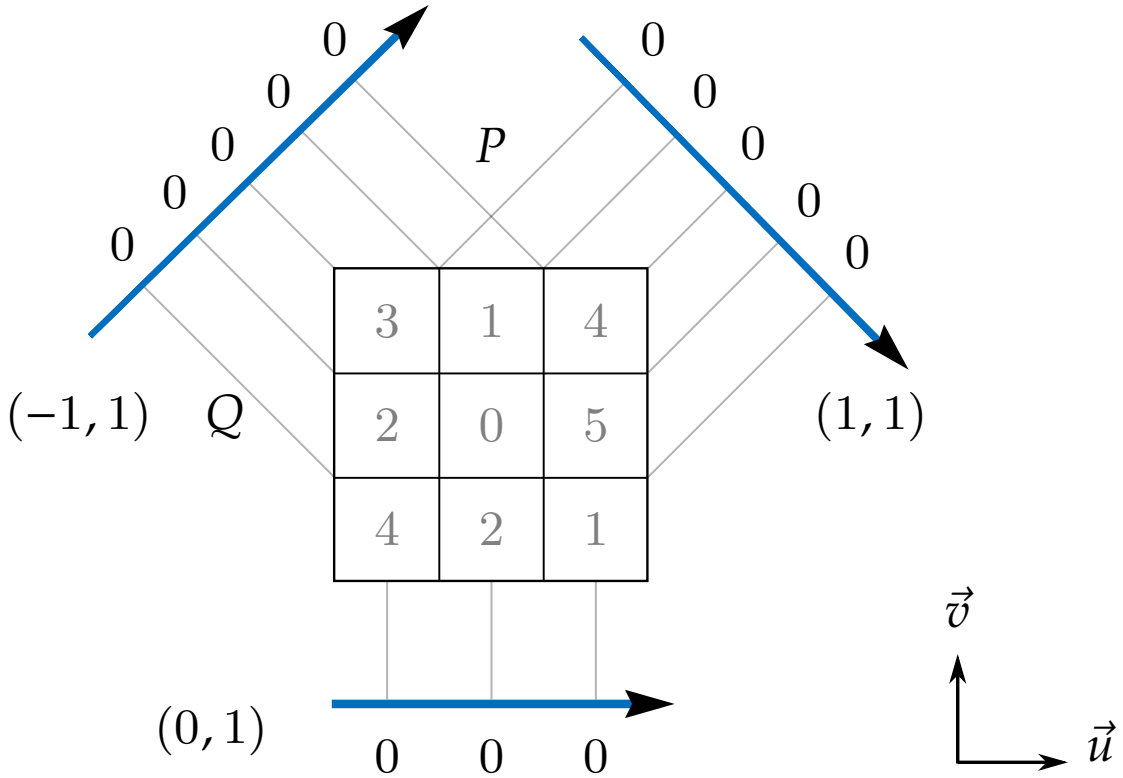
Inverse Mojette transform



$$[\mathcal{M}f](b, p_i, q_i) = \sum_{k=0}^{Q-1} \sum_{l=0}^{P-1} f(k, l)\, \Delta(b + kq_i - lp_i)$$

[Normand et al., 2006]

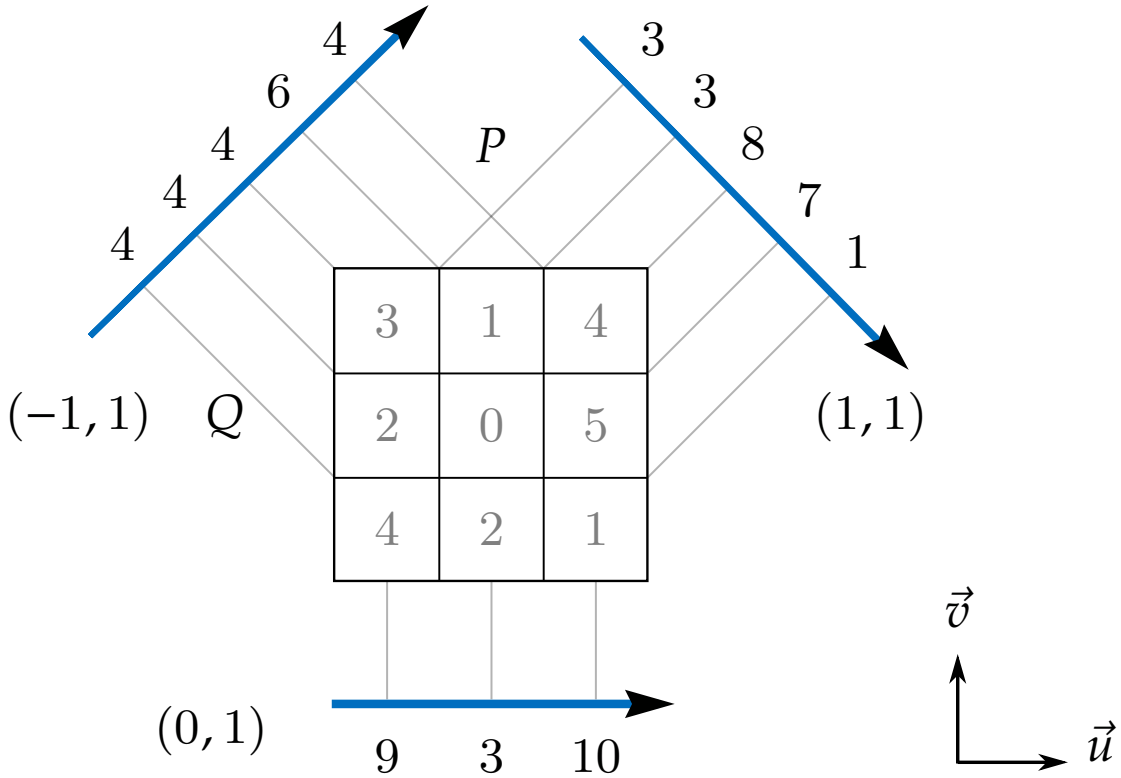
Inverse Mojette transform



$$[\mathcal{M}f](b, p_i, q_i) = \sum_{k=0}^{Q-1} \sum_{l=0}^{P-1} f(k, l)\, \Delta(b + kq_i - lp_i)$$

[Normand et al., 2006]

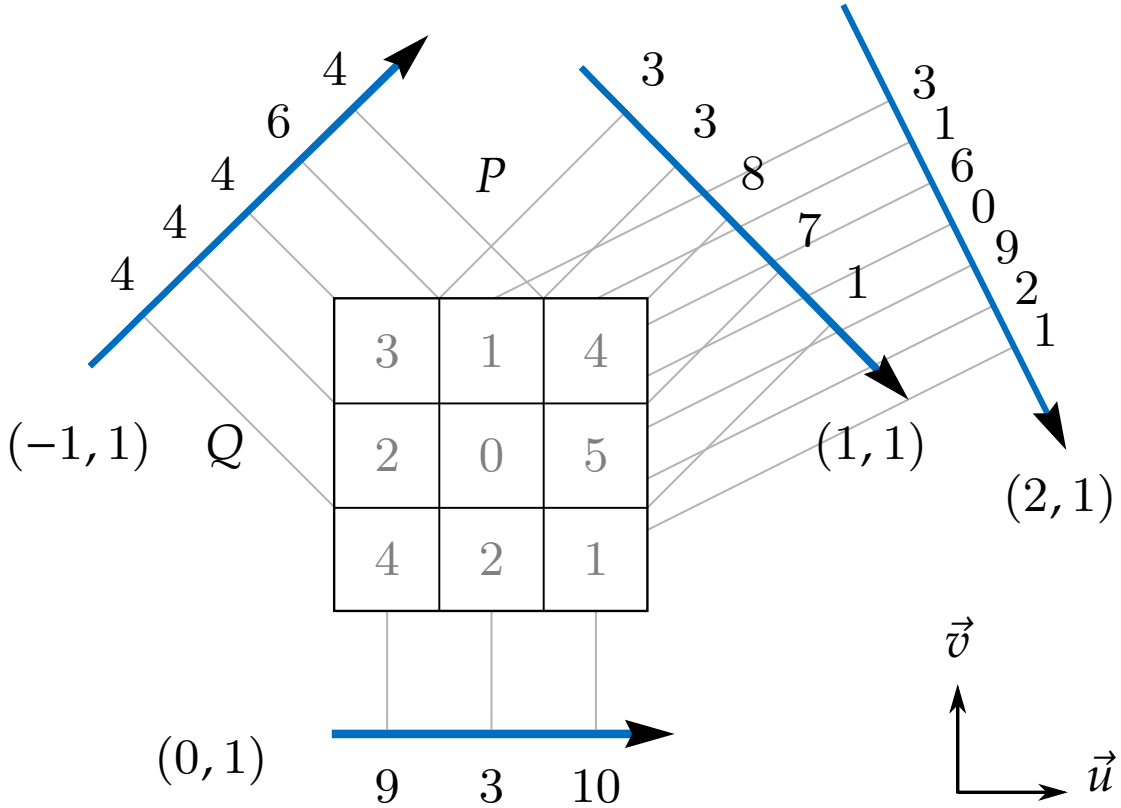
Redundant representation



$$\sum_{i=1}^{I} |q_i| \geq Q \text{ (Katz' criterion)}$$

[Normand et al., 2006]

Redundant representation



$$\sum_{i=1}^{I} |q_i| \geq Q \text{ (Katz' criterion)}$$

[Normand et al., 2006]

(n, k) Mojette erasure code

Non-systematic (n, k) Mojette encoding:

- consider $k = Q$ lines in the discrete grid
- given the following constraint: $(p_i, q_i = 1)_{i \in \mathbb{Z}_n}$ [Parrein, 2001]
- compute n projections

Non-systematic (n, k) Mojette decoding:

- reconstruction is unique given any k projections among the n (Katz' criterion)
- apply reconstruction algorithm [Normand et al., 2006]

(n, k) Mojette erasure code

Non-systematic (n, k) Mojette encoding:

- consider $k = Q$ lines in the discrete grid
- given the following constraint: $(p_i, q_i = 1)_{i \in \mathbb{Z}_n}$ [Parrein, 2001]
- compute n projections

Non-systematic (n, k) Mojette decoding:

- reconstruction is unique given any k projections among the n (Katz' criterion)
- apply reconstruction algorithm [Normand et al., 2006]

Outline

1. State of the Art

2. Systematic Mojette Erasure Code

2.1 Mojette erasure code

2.2 Systematic version

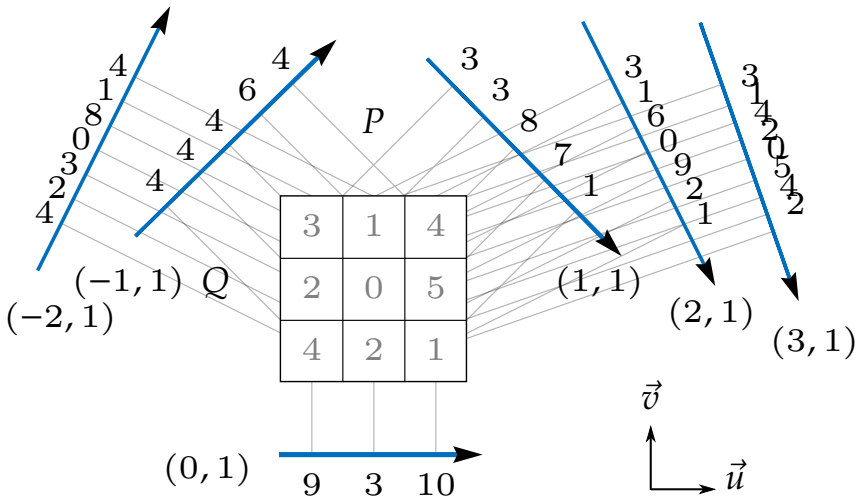
2.3 Evaluations

3. Reprojection without reconstruction

4. Distributed Storage System: RozoFS

5. Conclusion

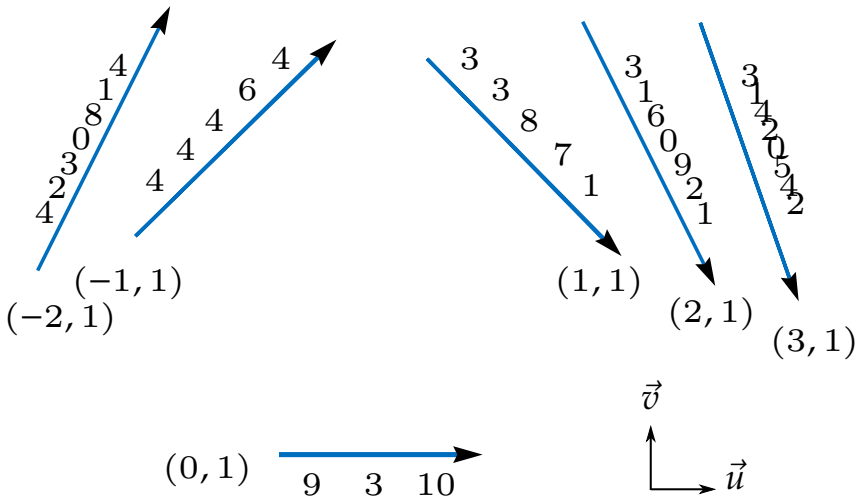
Non-systematic vs systematic Mojette erasure code



$(6, 3)$ Non-systematic

- n encoded blocks: 6 projections
- size grows with value $|p_i|$

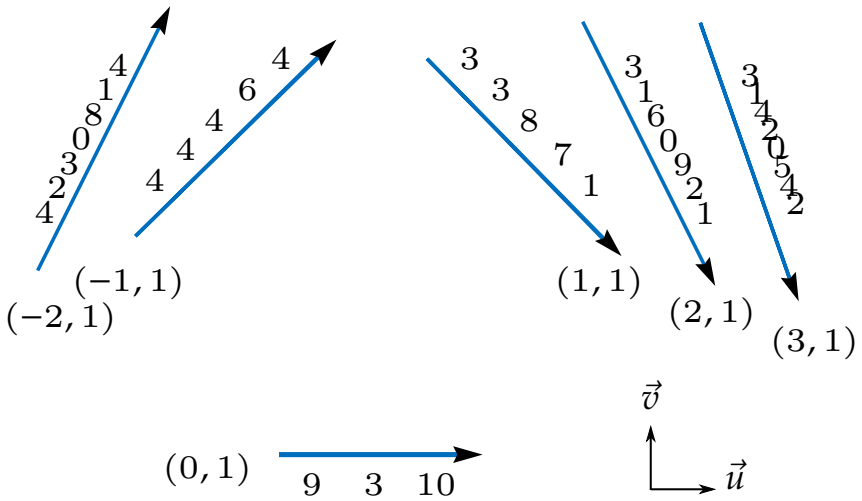
Non-systematic vs systematic Mojette erasure code

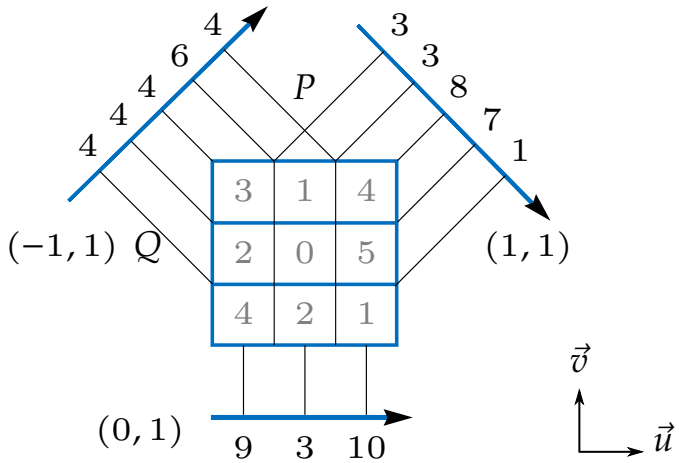


$(6, 3)$ Non-systematic

- n encoded blocks: 6 projections
- size grows with value $|p_i|$

Non-systematic vs systematic Mojette erasure code





$(6, 3)$ Non-systematic

- n encoded blocks: 6 projections
- size grows with value $|p_i|$

$(6, 3)$ Systematic

- k data blocks: 3 lines from grid
- $n - k$ parity blocks: 3 projections

Contributions: Inverserse agorithm + implementation

- Sylvain David et al. “Procédé et appareil permettant de reconstruire un bloc de données”. Centre National de la Recherche Scientifique (CNRS) Université de Nantes. WO Patent App. PCT/EP2014/071,310. Oct. 18, 2013
- Dimitri Pertin et al. “Comparison of RAID-6 Erasure Codes”. In: The third Sino-French Workshop on Information and Communication Technologies. SIFWICT. Nantes, France, June 2015
  - we designed an algorithm based on [Normand et al., 2006]
  - we implemented and evaluated our solution

Outline

1. State of the Art

2. Systematic Mojette Erasure Code

2.1 Mojette erasure code

2.2 Systematic version

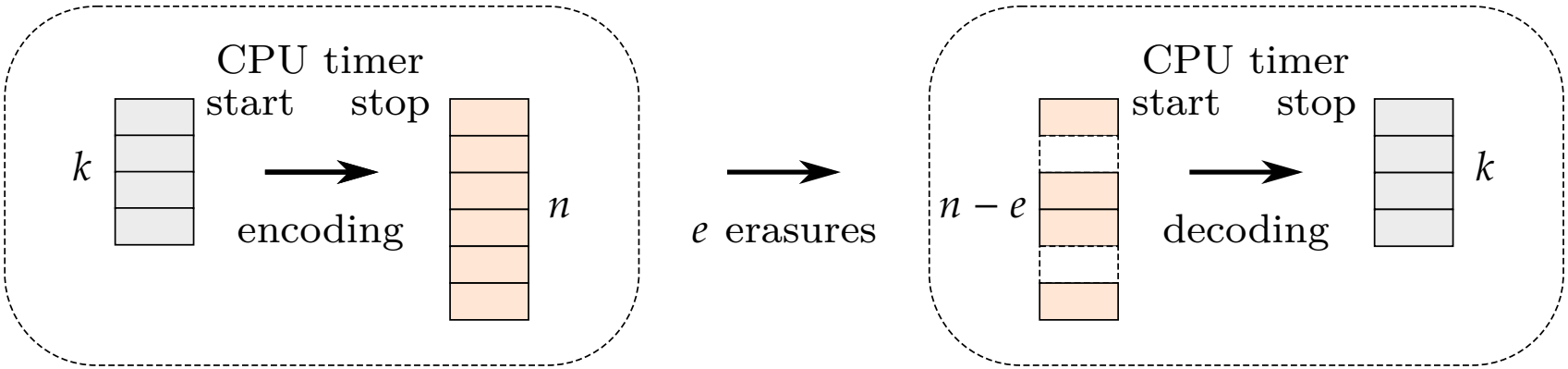
2.3 Evaluations

3. Reprojection without reconstruction

4. Distributed Storage System: RozoFS

5. Conclusion

Experimentation



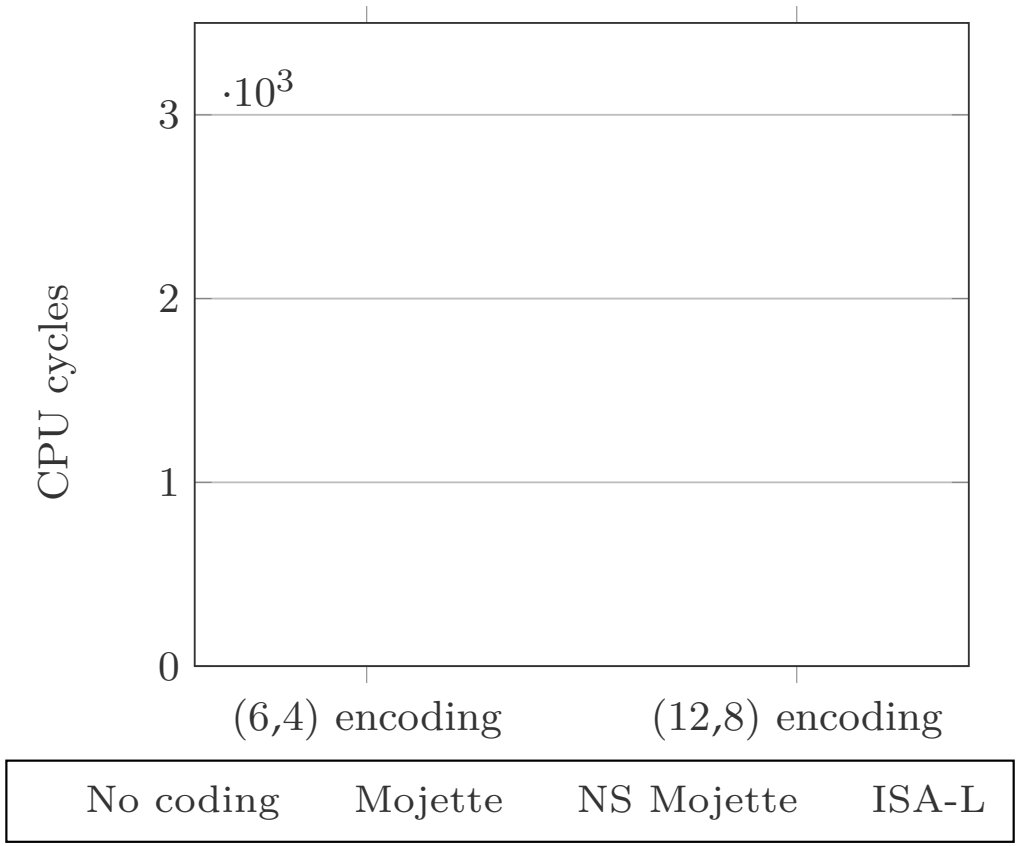
Encode $\mathscr{M}$ bytes (in memory)

Decode the $\mathscr{M}$ bytes (in memory)

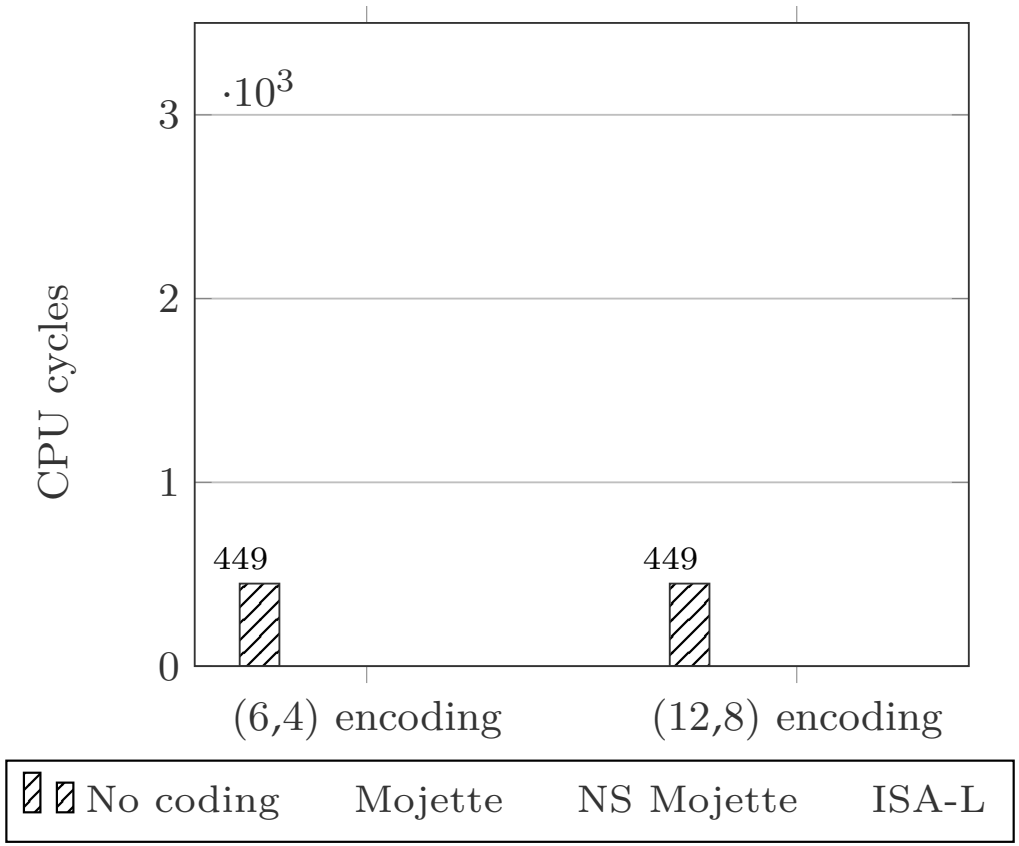
Experimentation setup

- Mojette vs Reed-Solomon (INTEL ISA-L)
- Small block sizes (i.e. $\mathscr{M} = 4$ KB)
- FEC4Cloud platform: Intel Xeon 1.80 GHz, RAM 16 GB

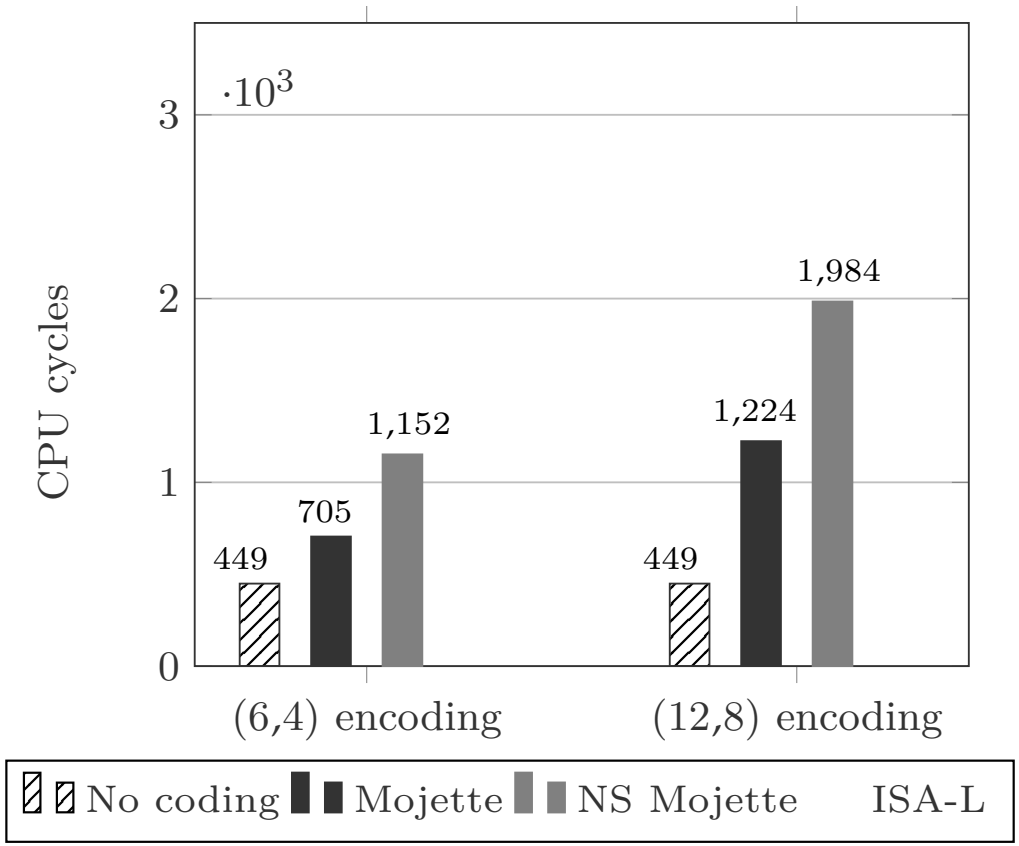
Encoding performance



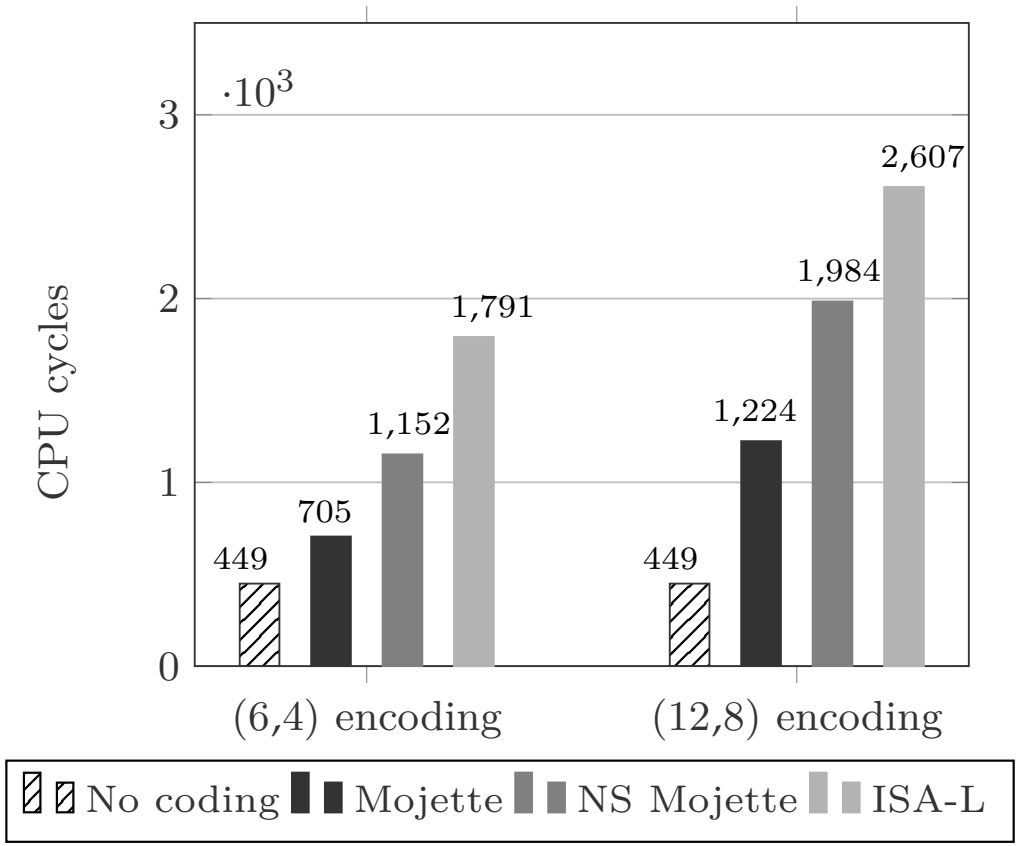
Encoding performance



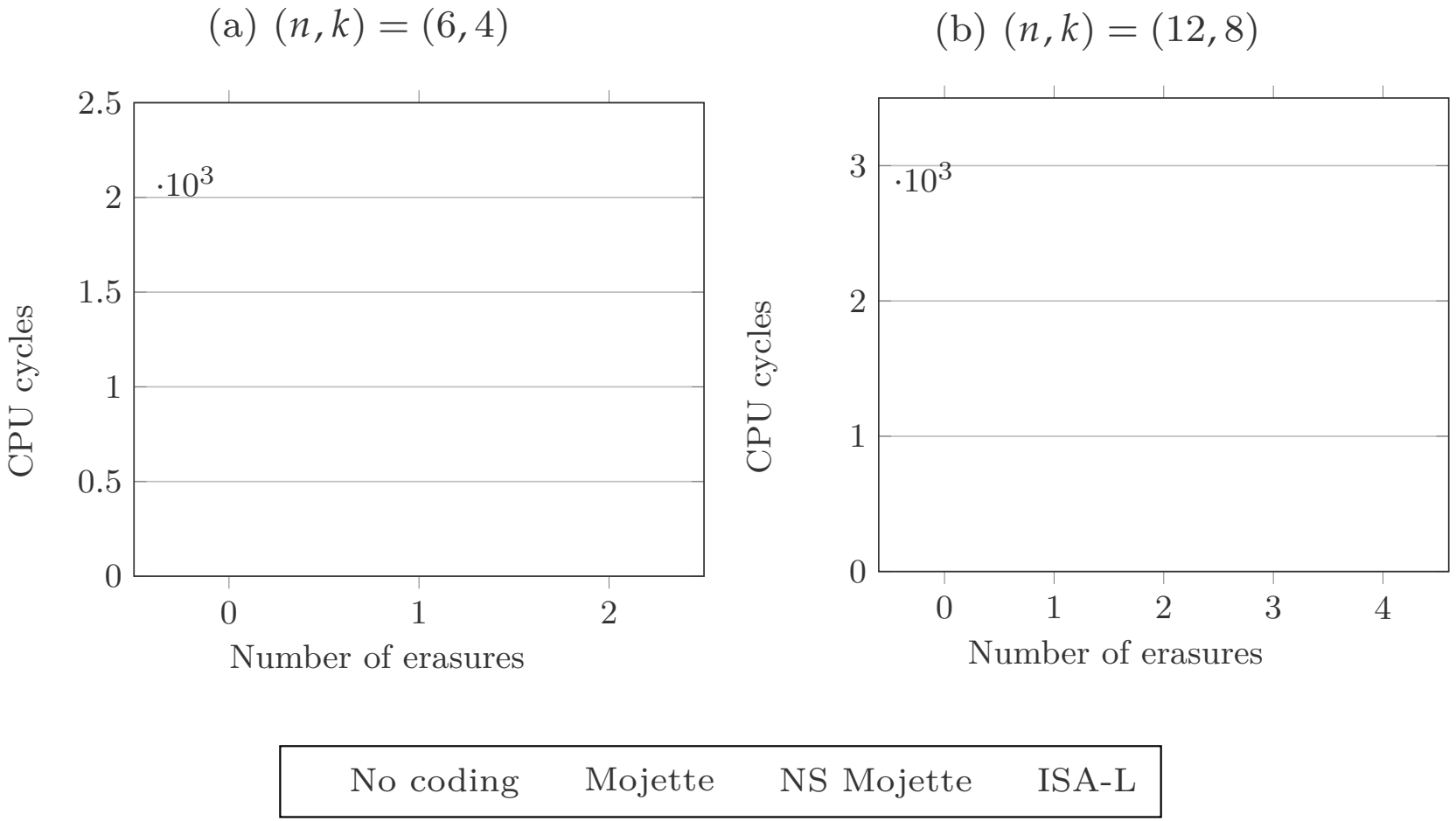
Encoding performance



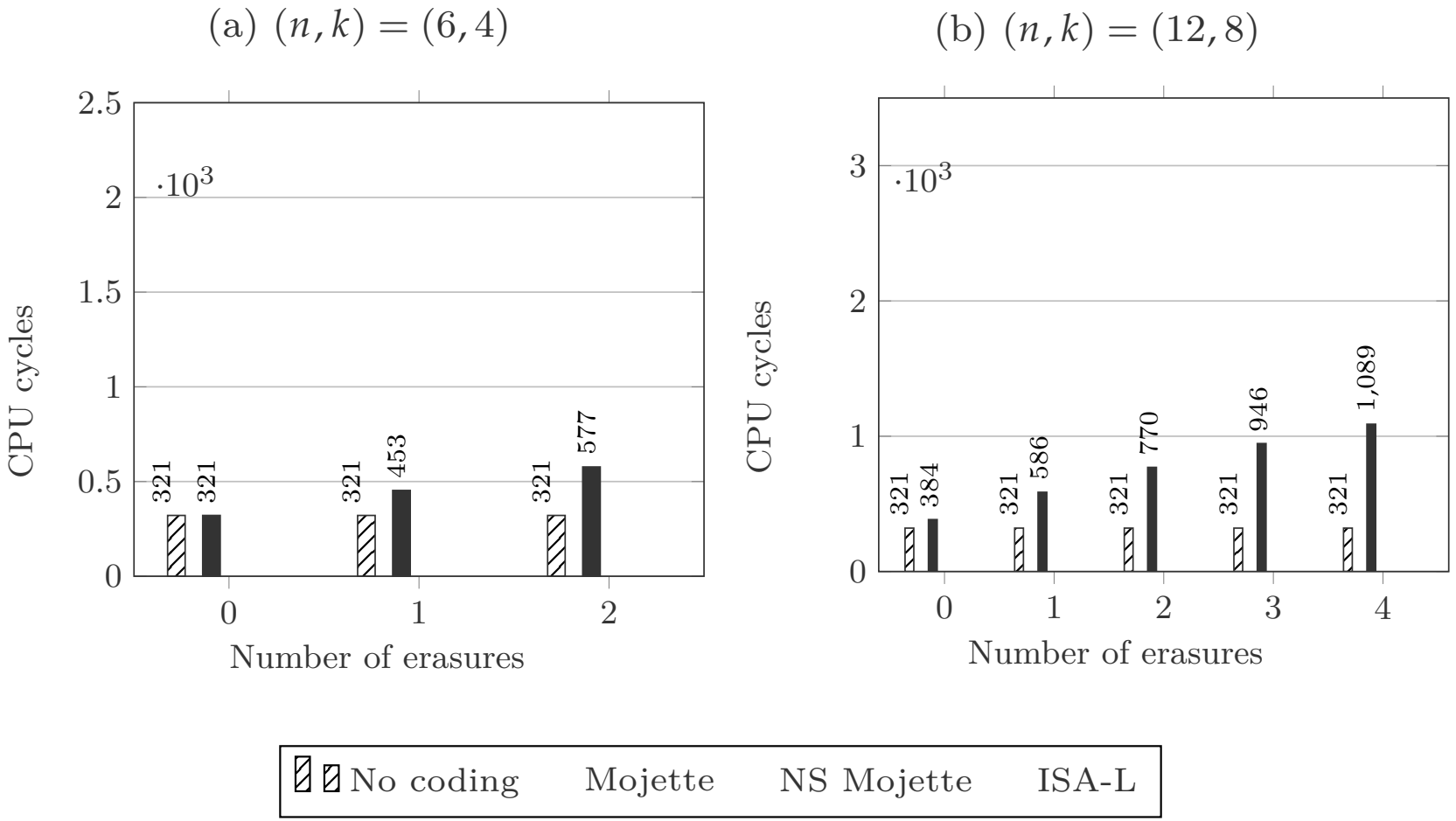
Encoding performance



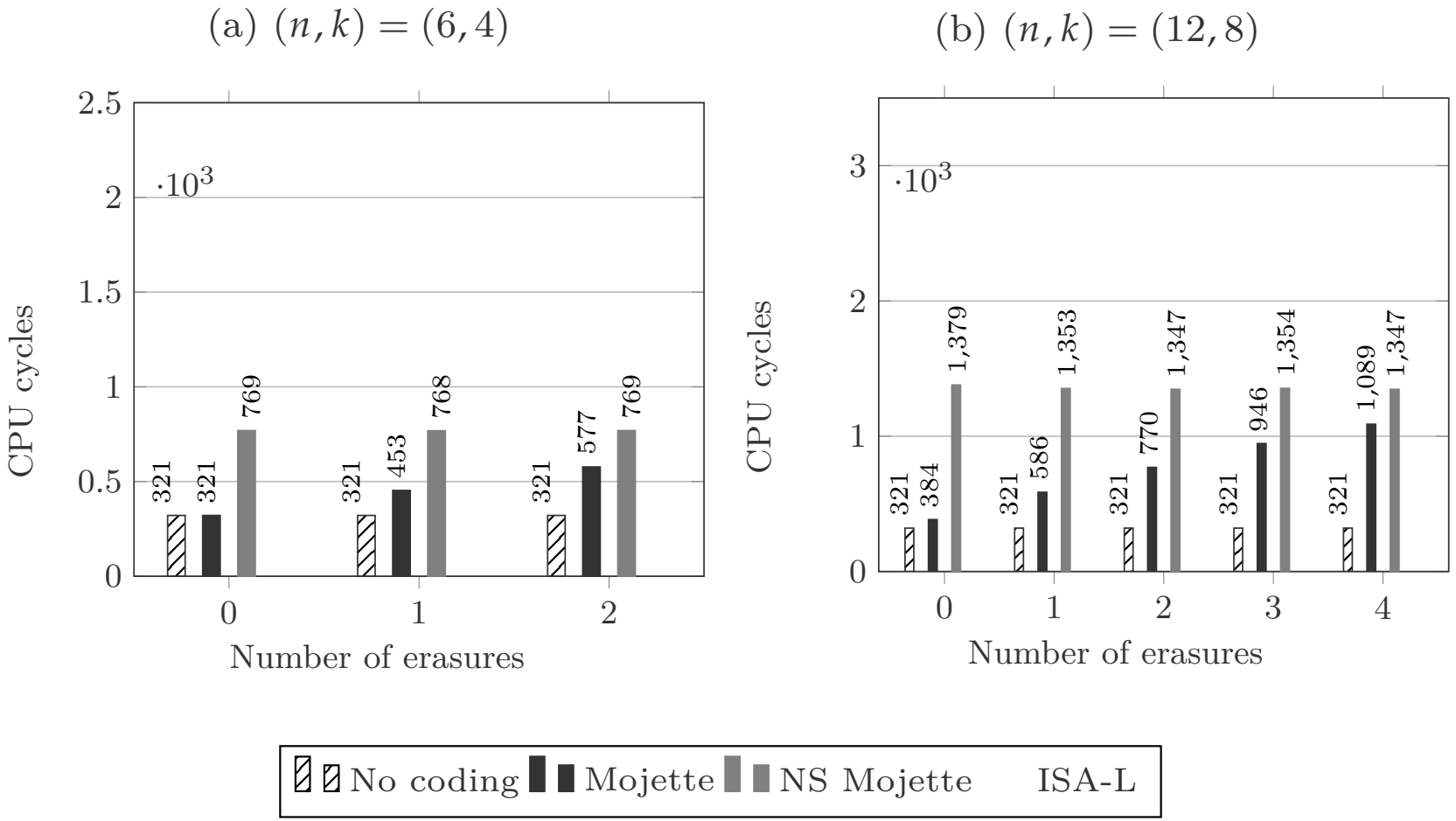
Decoding performance



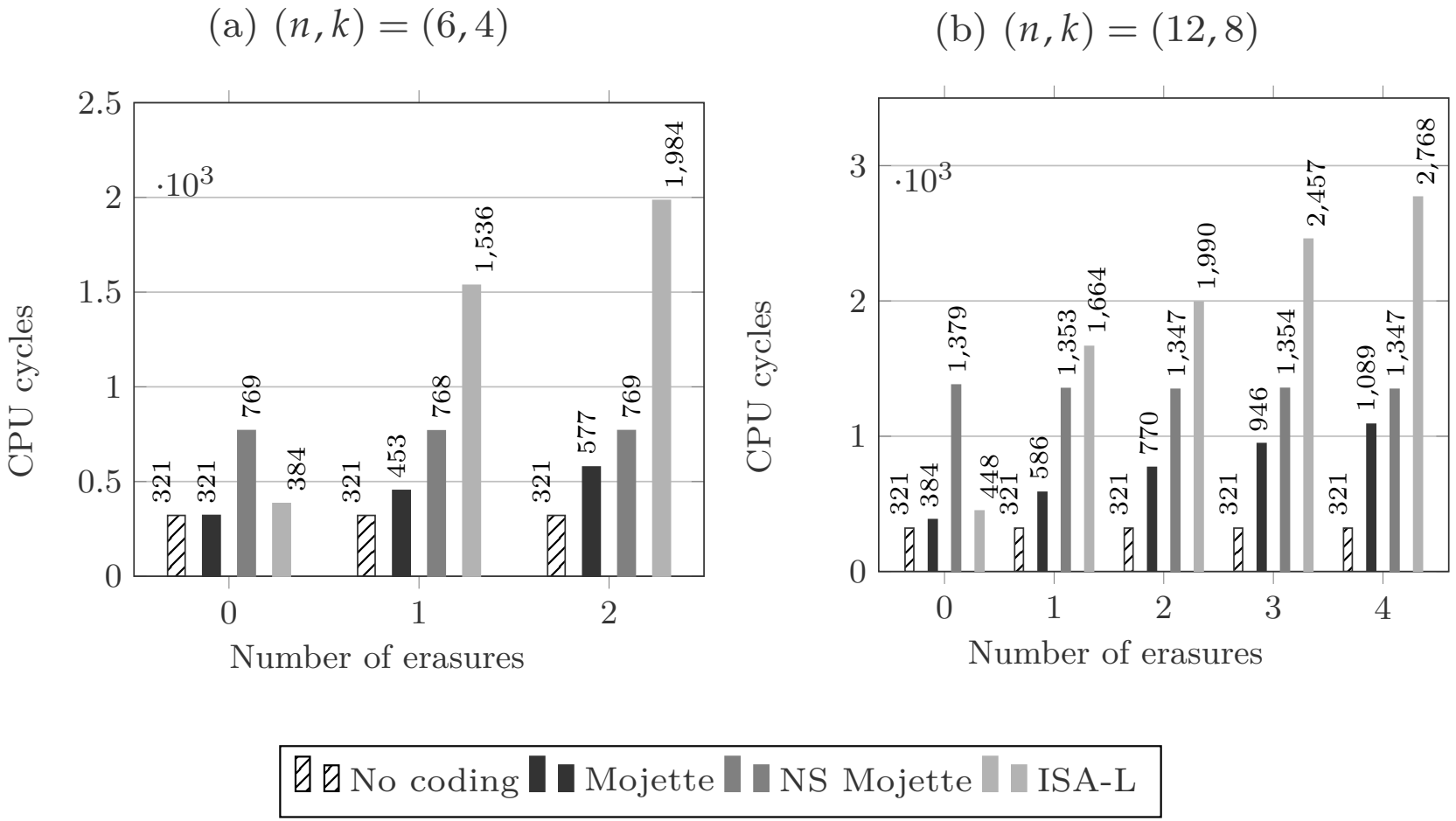
Decoding performance



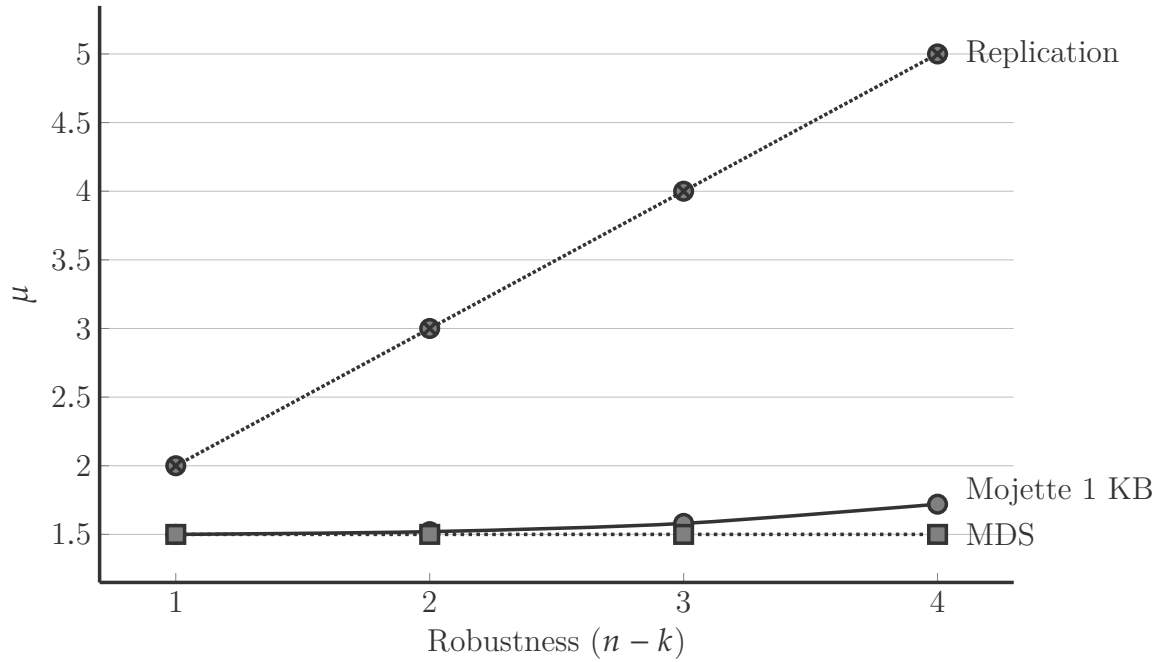
Decoding performance



Decoding performance

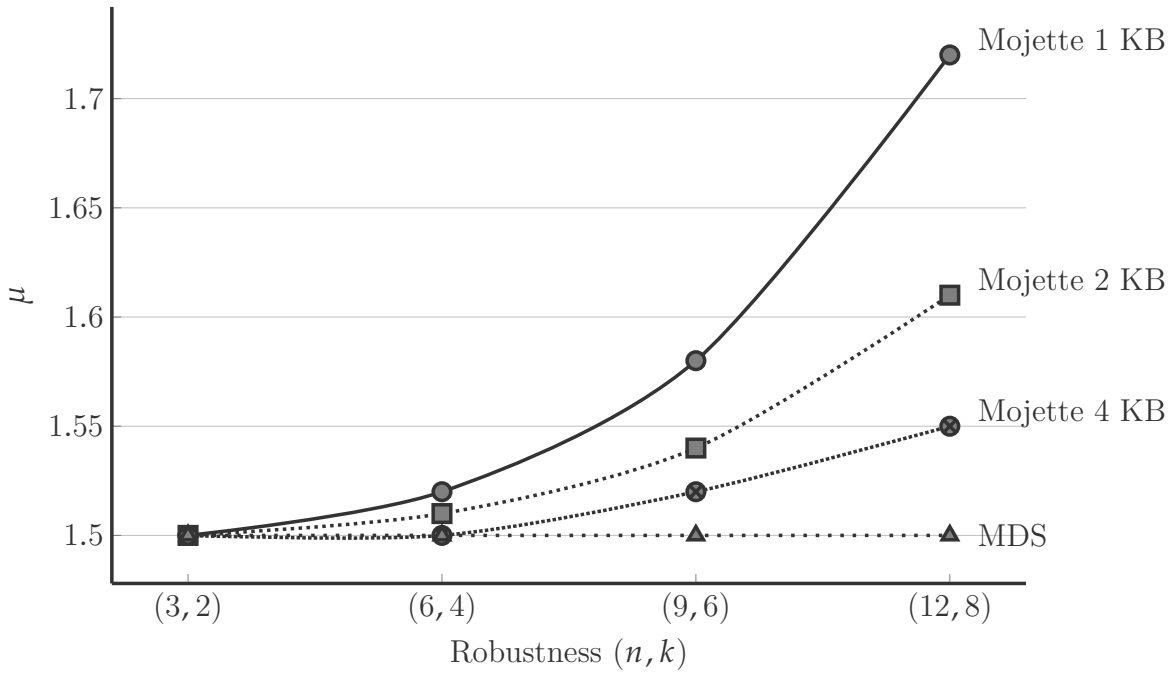


Storage overhead evaluation



- $\mu = \frac{\text{size of } n \text{ encoded blocks}}{\text{size of } k \text{ data blocks}}$
  - $\mu_{\text{MDS}} = \frac{n}{k}$
  - $\mu_{\text{Mojette}} = \frac{\#_{\text{pixel and bins}}}{\#_{\text{pixels}}}$

Storage overhead evaluation



- $\mu = \frac{\text{size of } n \text{ encoded blocks}}{\text{size of } k \text{ data blocks}}$
  - $\mu_{\text{MDS}} = \frac{n}{k}$
  - $\mu_{\text{Mojette}} = \frac{\#_{\text{pixel and bins}}}{\#_{\text{pixels}}}$

Conclusion on Mojette erasure code

1. systematic version outperforms the non-systematic version
2. ...and Reed-Solomon codes from ISA-L (factor ×3)
3. cost only 3% extra data compared to optimal codes (near-MDS)

- Dimitri Pertin et al. The Mojette erasure code for distributed file systems. Session poster. Amsterdam, The Netherlands: EuroSys'14, Apr. 2014
- Dimitri Pertin et al. "Performance evaluation of the Mojette erasure code for fault-tolerant distributed hot data storage". In: CoRR abs/1504.07038 [Apr. 2015]

Outline

1. State of the Art

2. Systematic Mojette Erasure Code

3. Reprojection without reconstruction

3.1 The reprojection problem

3.2 Distributed reprojection

3.3 Evaluation

4. Distributed Storage System: RozoFS

5. Conclusion

Outline

1. State of the Art

2. Systematic Mojette Erasure Code

3. Reprojection without reconstruction

3.1 The reprojection problem

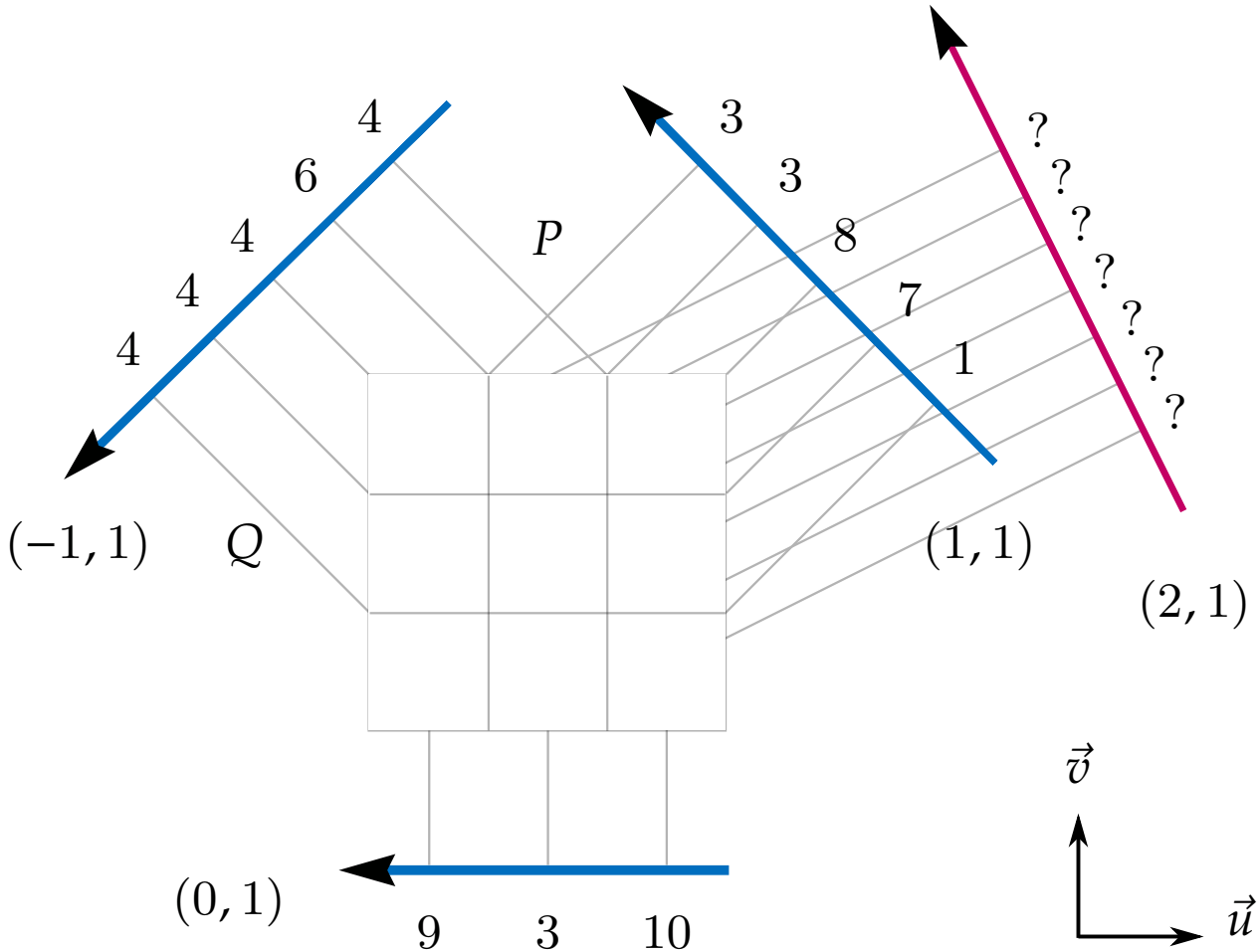
3.2 Distributed reprojection

3.3 Evaluation

4. Distributed Storage System: RozoFS

5. Conclusion

Reprojection problem



Outline

1. State of the Art

2. Systematic Mojette Erasure Code

3. Reprojection without reconstruction

3.1 The reprojection problem

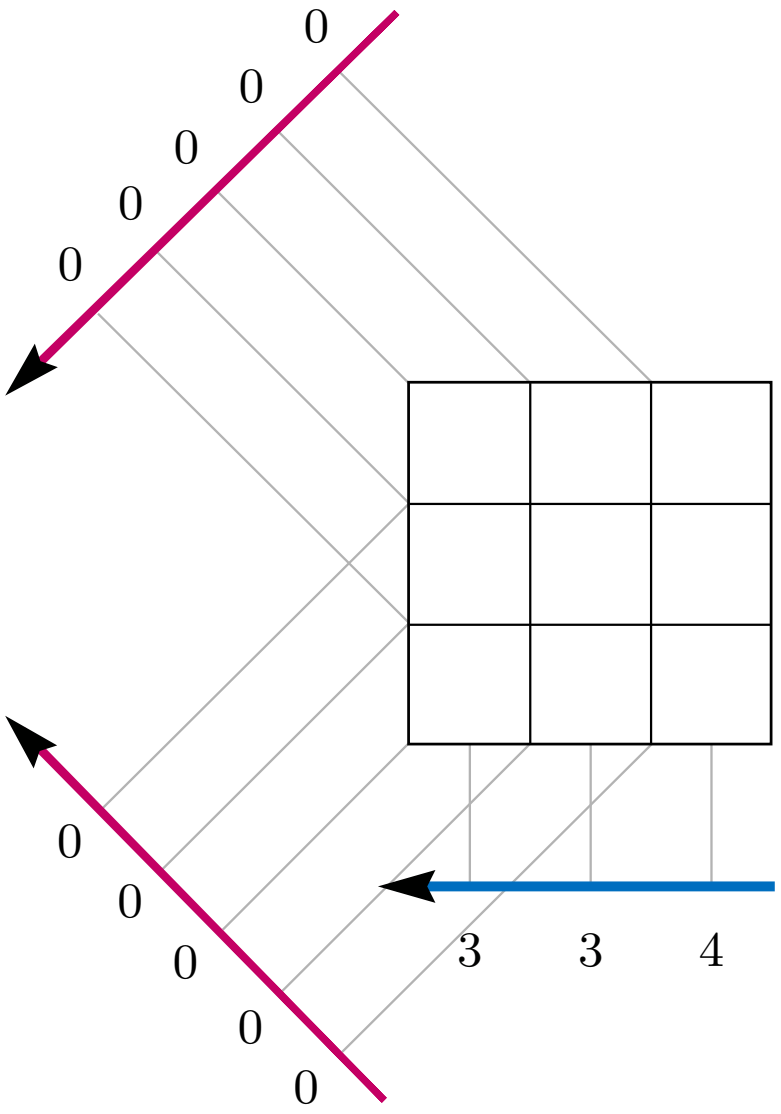
3.2 Distributed reprojection

3.3 Evaluation

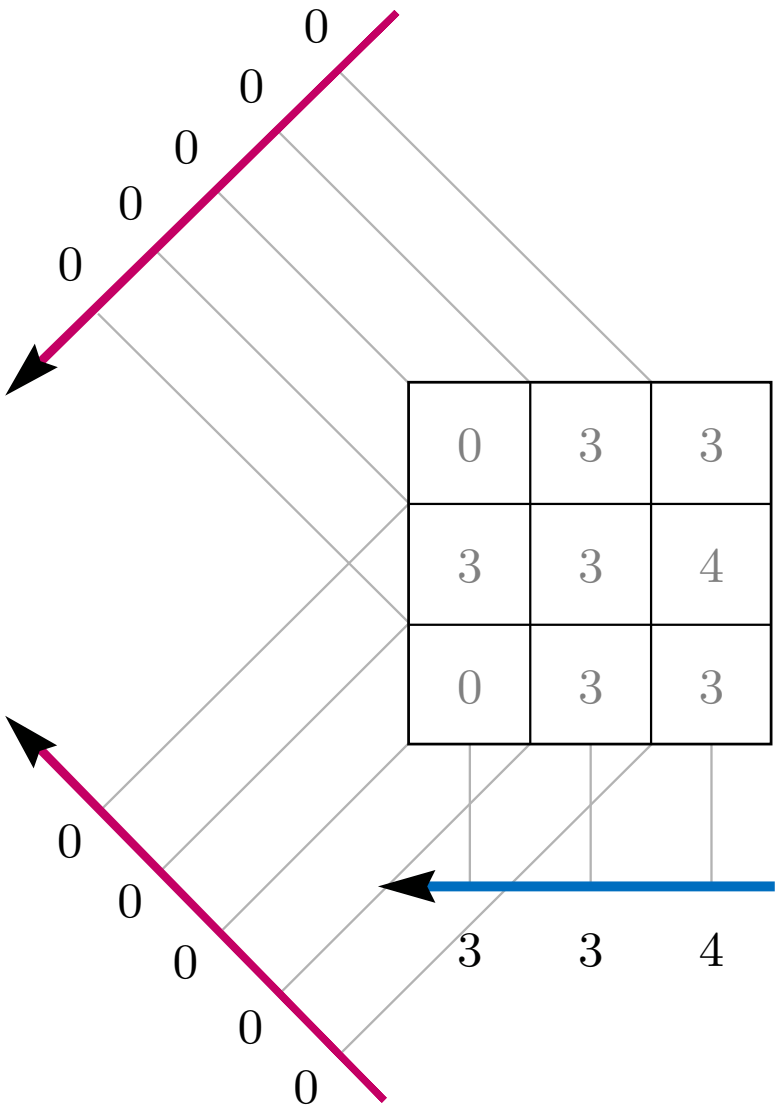
4. Distributed Storage System: RozoFS

5. Conclusion

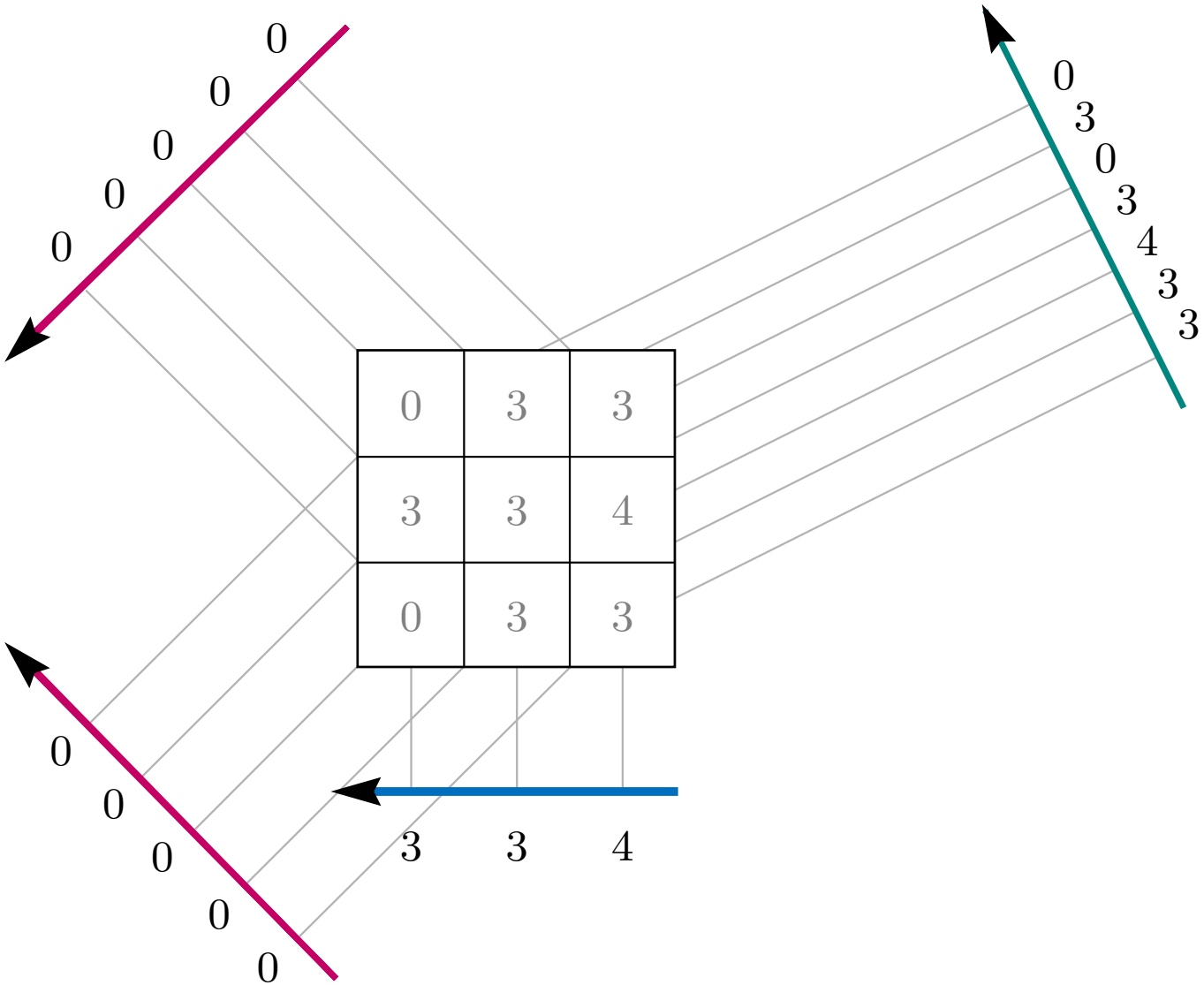
Partial reconstruction



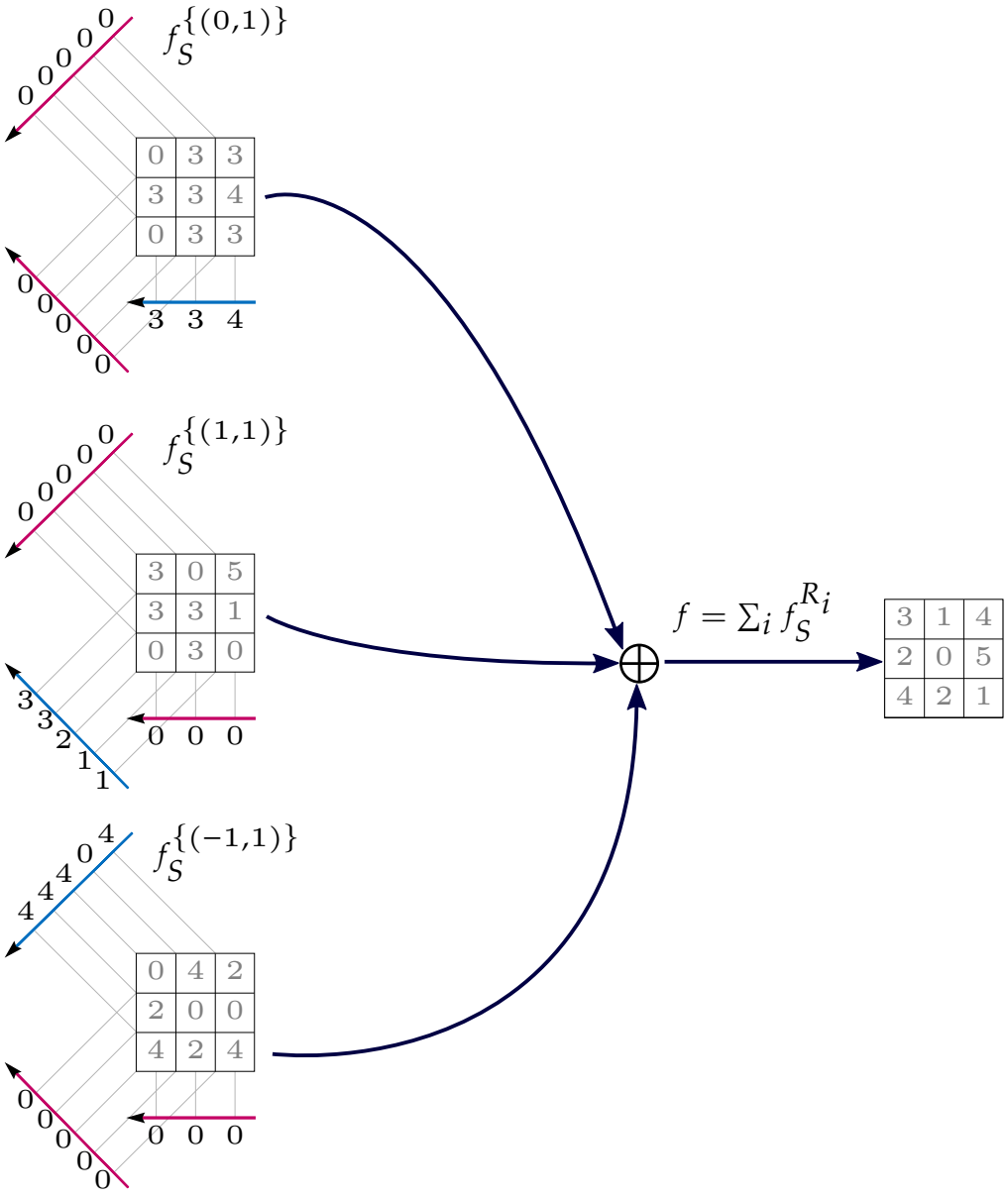
Partial reconstruction



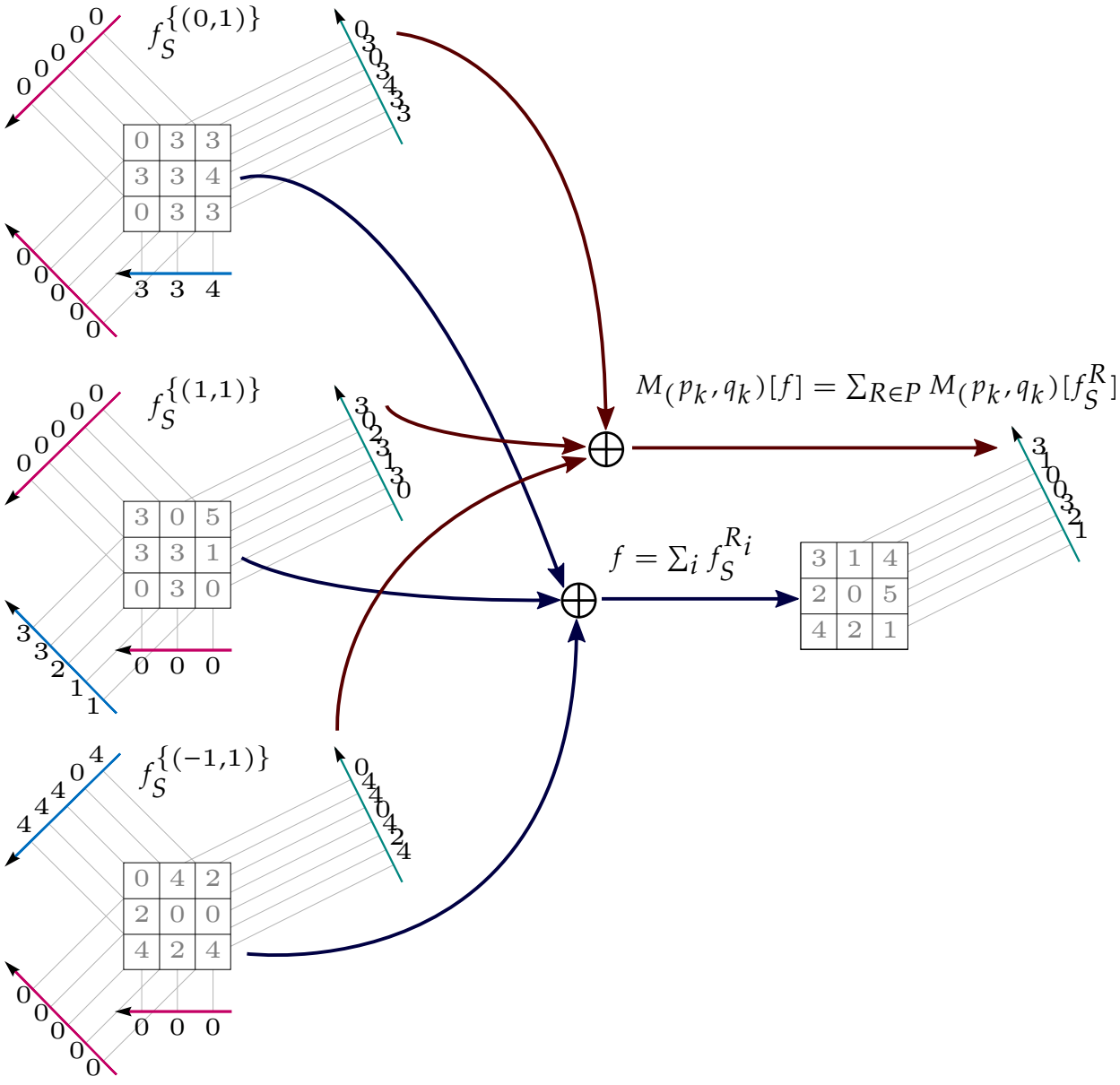
Partial reconstruction



Distributed Reprojection



Distributed Reprojection



Outline

1. State of the Art

2. Systematic Mojette Erasure Code

3. Reprojection without reconstruction

3.1 The reprojection problem

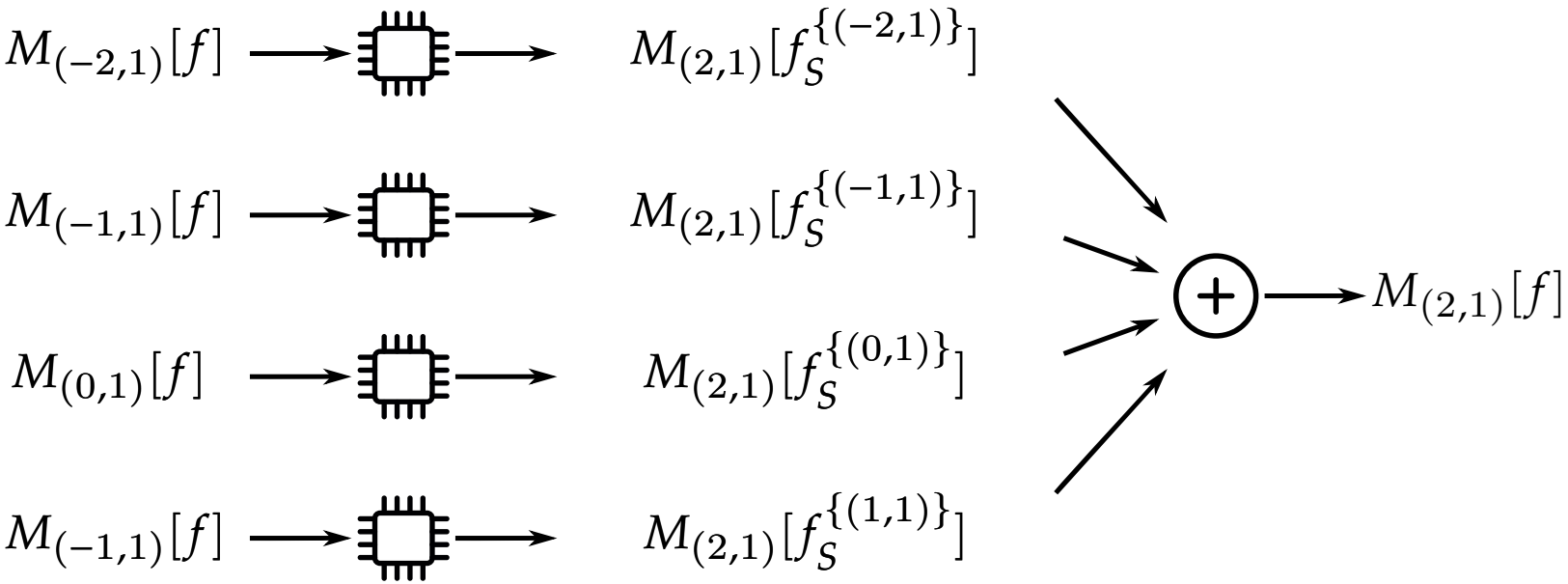
3.2 Distributed reprojection

3.3 Evaluation

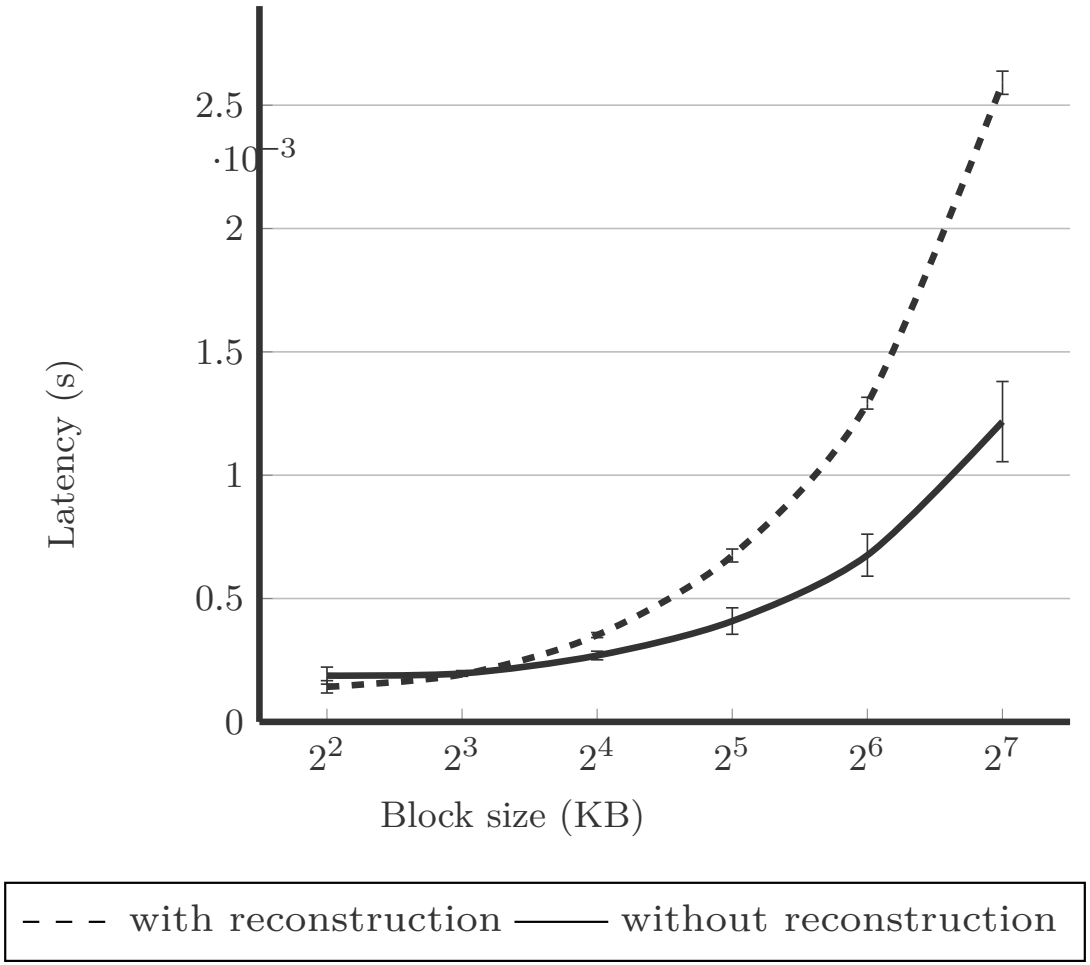
4. Distributed Storage System: RozoFS

5. Conclusion

Experimentation setup



Evaluation


2.5
·10−3
2
1.5
1
0.5
0
Latency (s)
2^2
2^3
2^4
2^5
2^6
2^7
Block size (KB)
with reconstruction
without reconstruction

Conclusion on distributed reprojection

1. distribute the reprojection process
2. ...without reconstructing original data (grid)
3. ...is faster by a factor 2

- Dimitri Pertin et al. “Re-projection without Reconstruction”. In: 9ème Journées du Groupe de travail de Géométrie Discrète, Reims Image 2014. Reims, France, Nov. 2014, p. 43

Outline

1. State of the Art

2. Systematic Mojette Erasure Code

3. Reprojection without reconstruction

4. Distributed Storage System: RozoFS

4.1 Distributed File System

4.2 RozoFS Architecture

4.3 Evaluations

5. Conclusion

Outline

1. State of the Art

2. Systematic Mojette Erasure Code

3. Reprojection without reconstruction

4. Distributed Storage System: RozoFS

4.1 Distributed File System

4.2 RozoFS Architecture

4.3 Evaluations

5. Conclusion

inodes

UNIX FS[3]

- unique number
- POSIX metadata
  - size
  - permissions
  - timestamps
  - ...
- pointer to data blocks

RozoFS

- unique file identifier
- POSIX metadata
  - size
  - permissions
  - timestamps
  - ...
- extended attributes
  - storage node ids
  - Mojette code parameters
  - ...

[3] FS: File System

inodes

UNIX FS[3]

- unique number
- POSIX metadata
  - size
  - permissions
  - timestamps
  - ...
- pointer to data blocks

RozoFS

- unique file identifier
- POSIX metadata
  - size
  - permissions
  - timestamps
  - ...
- extended attributes
  - storage node ids
  - Mojette code parameters
  - ...

[3] FS: File System

Outline

1. State of the Art

2. Systematic Mojette Erasure Code

3. Reprojection without reconstruction

4. Distributed Storage System: RozoFS

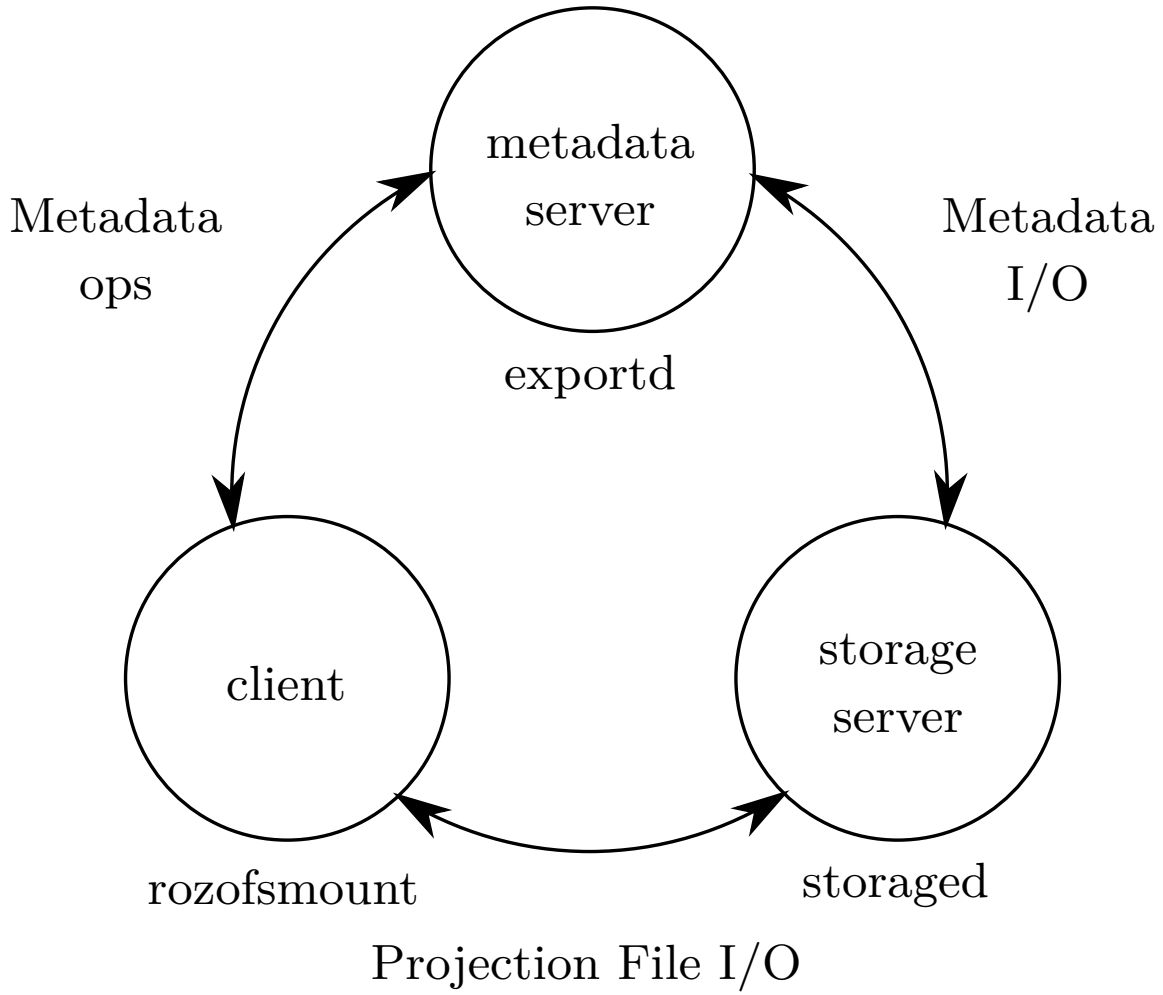
4.1 Distributed File System

4.2 RozoFS Architecture

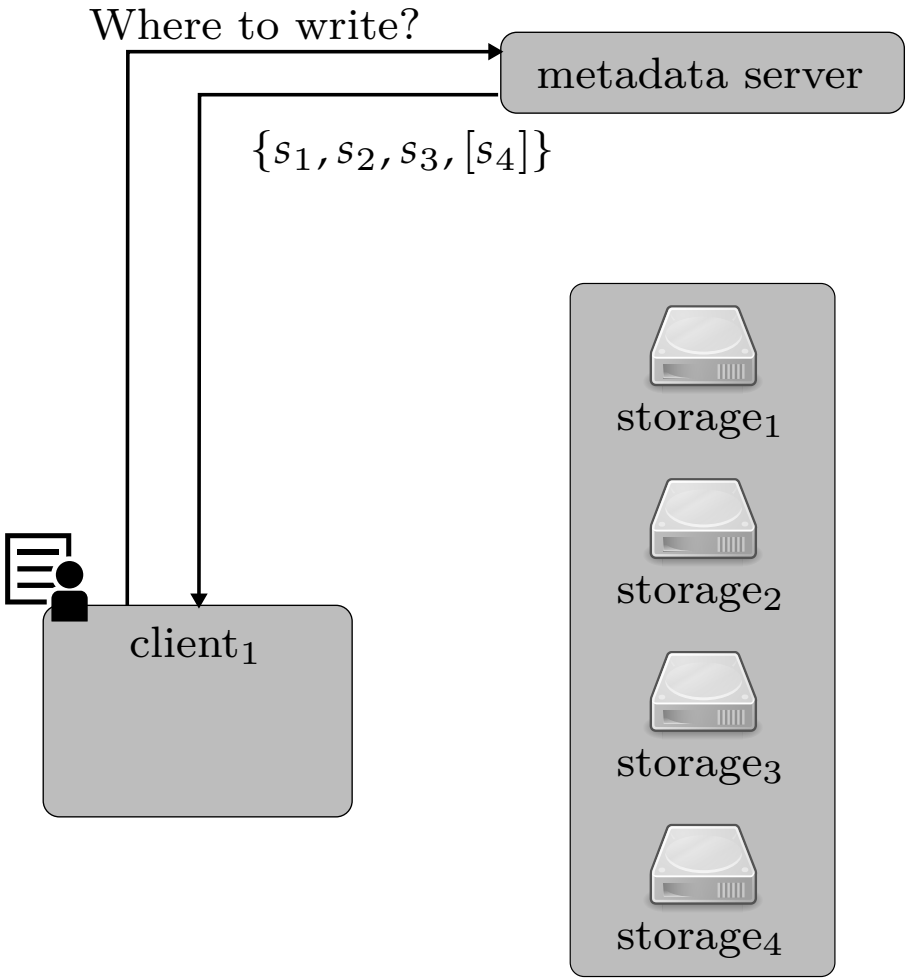
4.3 Evaluations

5. Conclusion

RozoFS Architecture

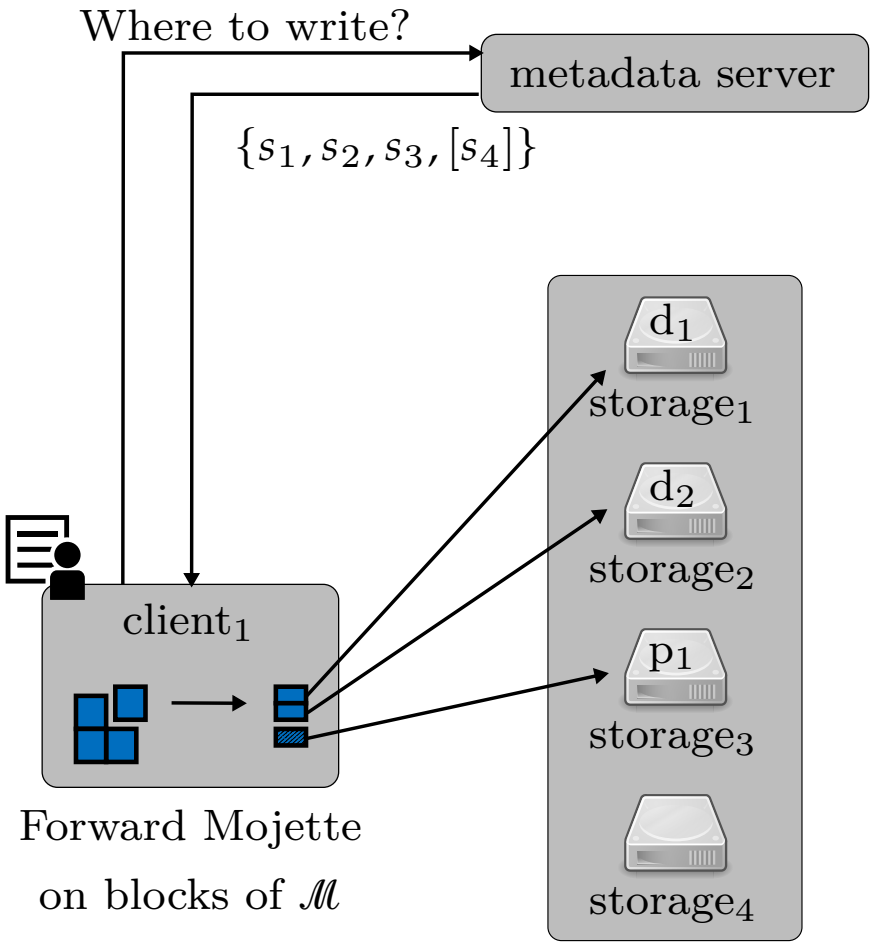


Data Input (writes) and Output (reads)



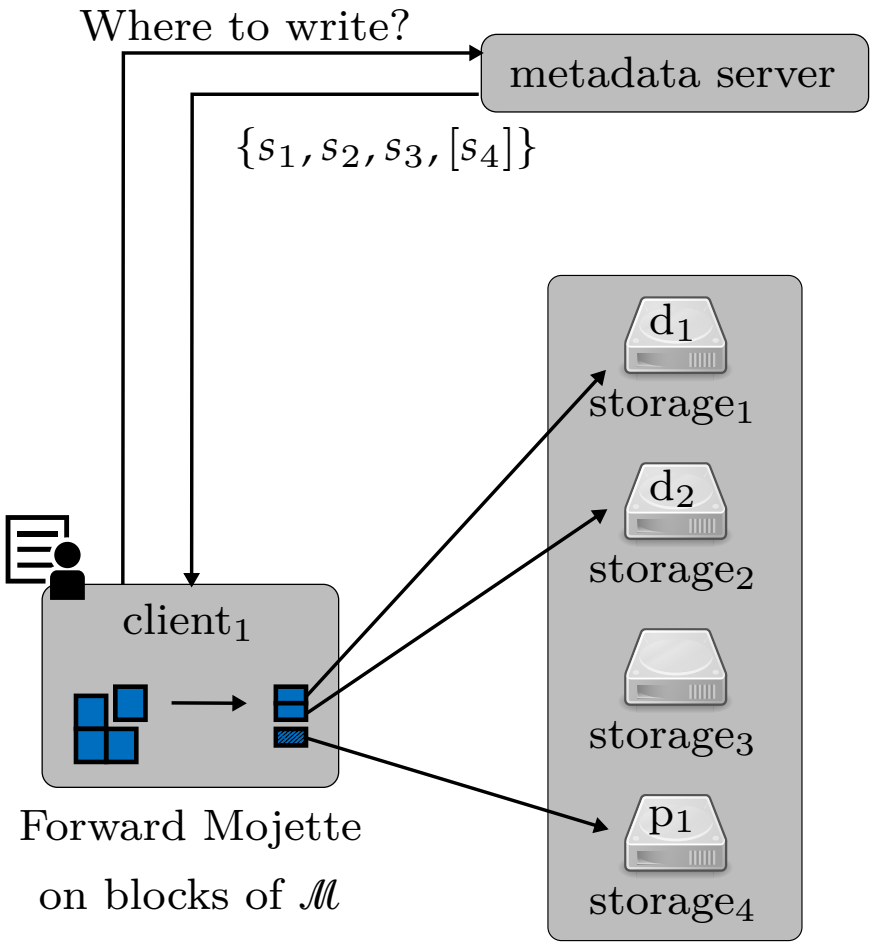
RozoFS architecture using $(3, 2)$ systematic Mojette code

Data Input (writes) and Output (reads)



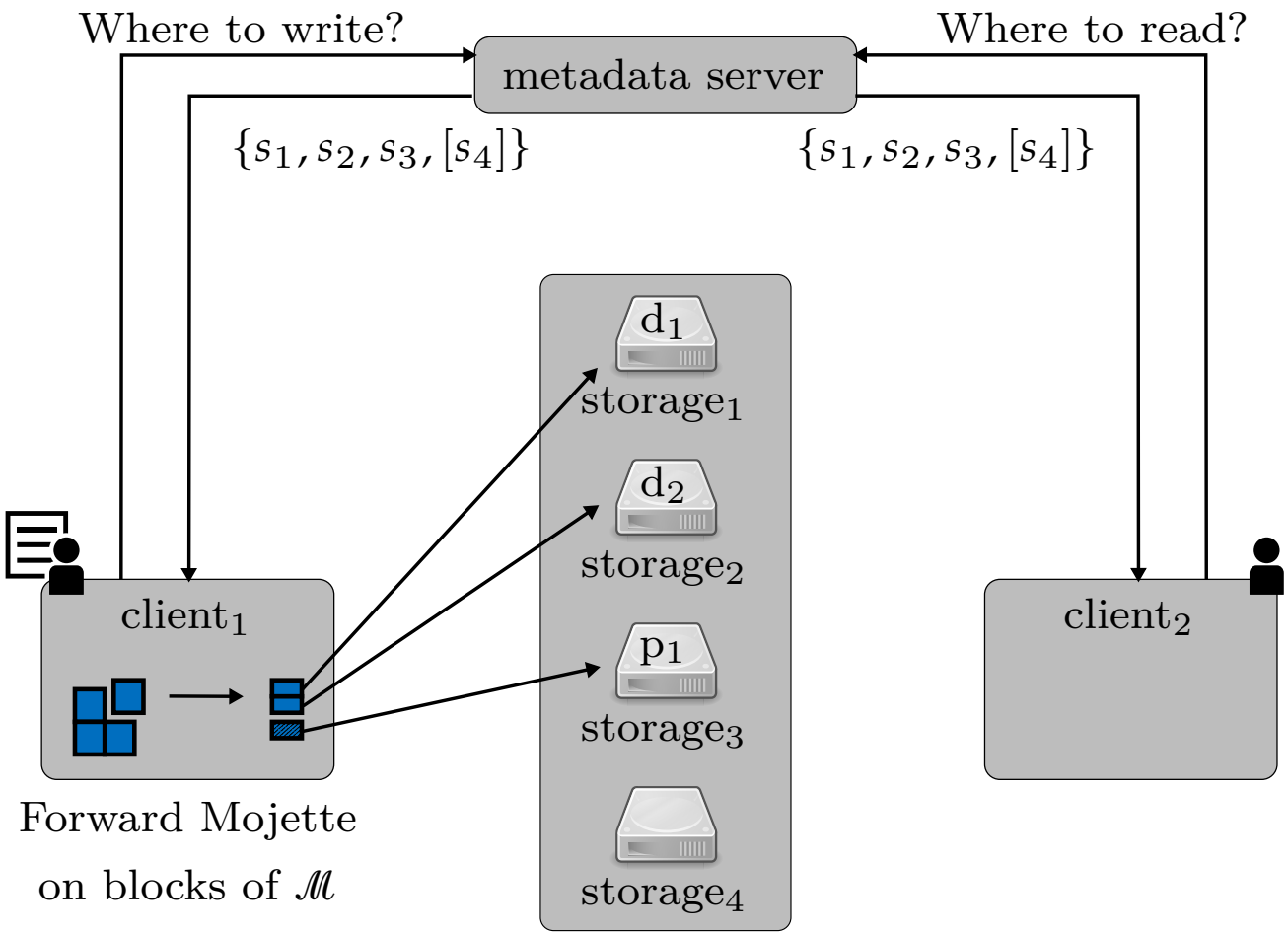
RozoFS architecture using $(3, 2)$ systematic Mojette code

Data Input (writes) and Output (reads)



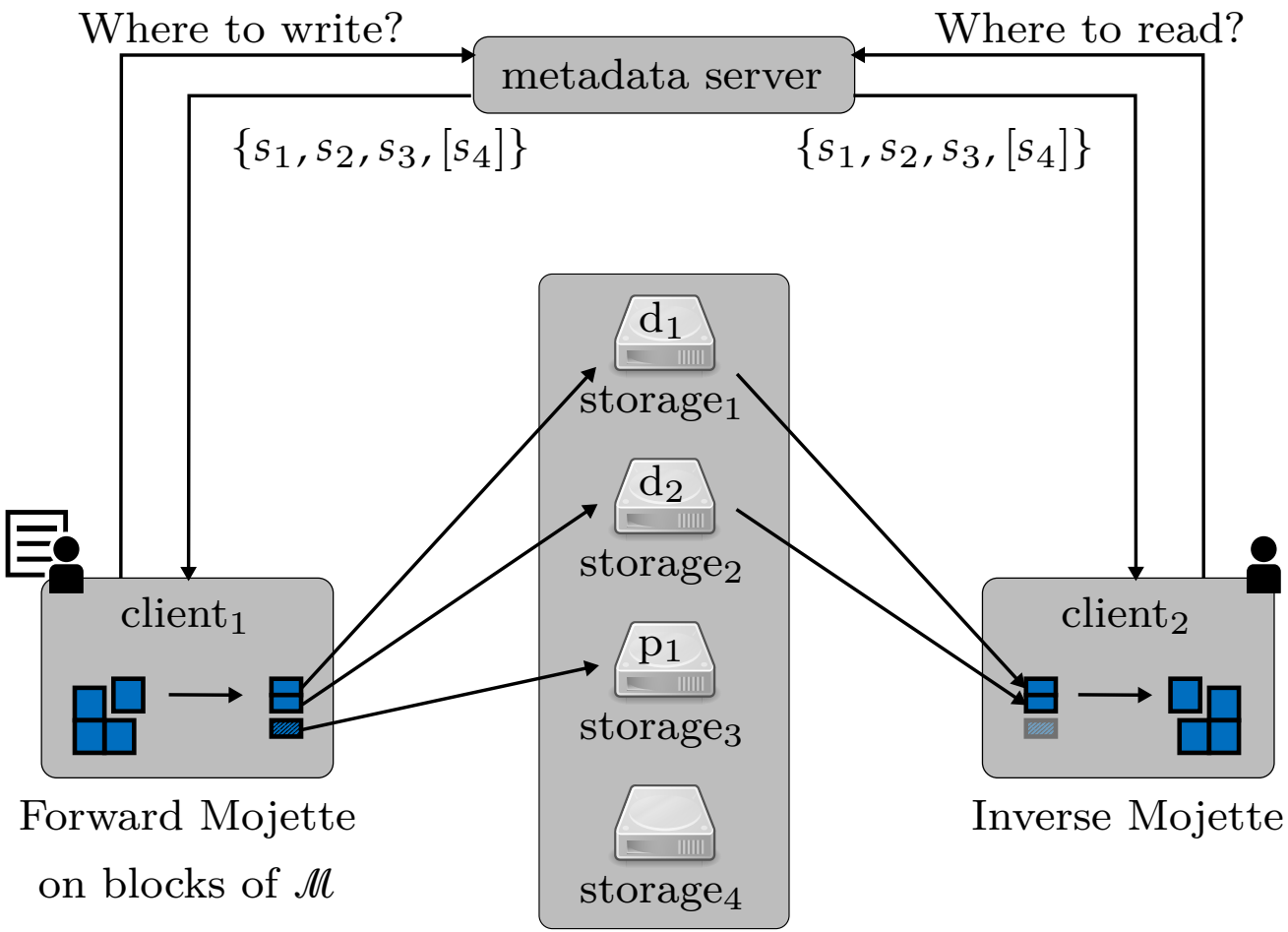
RozoFS architecture using $(3, 2)$ systematic Mojette code

Data Input (writes) and Output (reads)



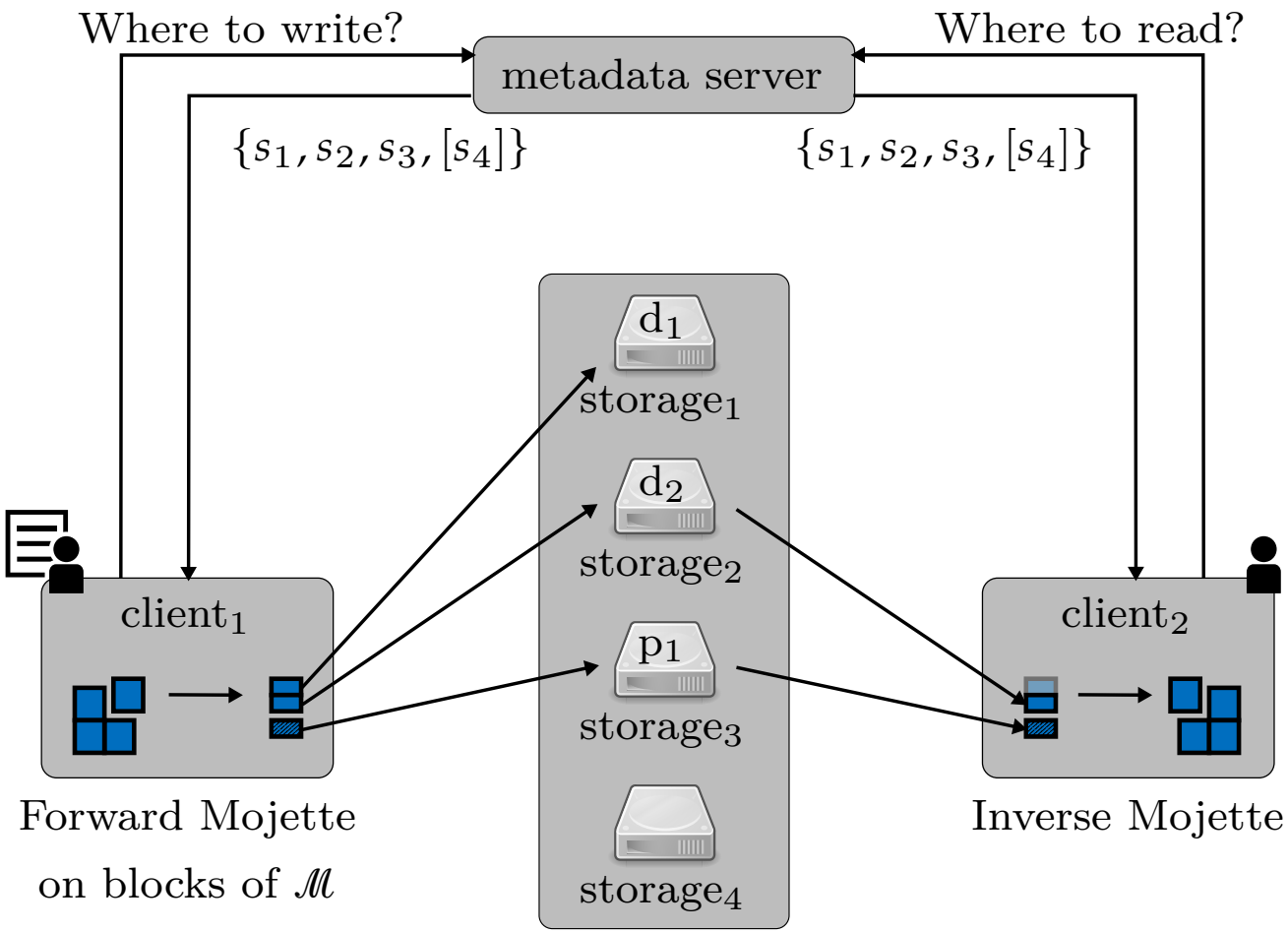
RozoFS architecture using $(3, 2)$ systematic Mojette code

Data Input (writes) and Output (reads)



RozoFS architecture using $(3, 2)$ systematic Mojette code

Data Input (writes) and Output (reads)



RozoFS architecture using $(3, 2)$ systematic Mojette code

Outline

1. State of the Art

2. Systematic Mojette Erasure Code

3. Reprojection without reconstruction

4. Distributed Storage System: RozoFS

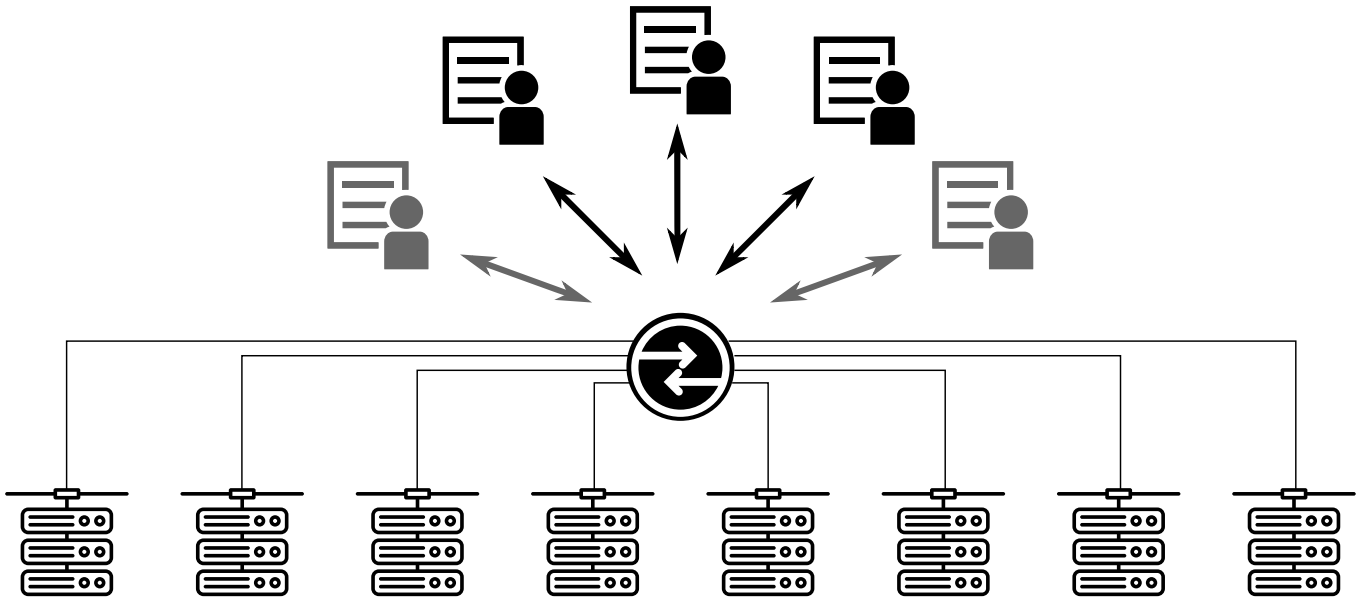
4.1 Distributed File System

4.2 RozoFS Architecture

4.3 Evaluations

5. Conclusion

Experimentation setup



- RozoFS (Mojette$(6, 4)$) vs CephFS (3-rep): 8 storage nodes
- IOzone (read/write, random/sequential)
- GRID'5k econome platform:
  - Intel Xeon 2.20 GHz, RAM 64 GB, 10 GbE, 7200 RPM disks

Competitor: CephFS

Pros:

- very popular
- General purpose file system (HDFS is not)
- fault-tolerance by replication (Lustre is not)

Cons:

- based on objects
- file system interface is still in development

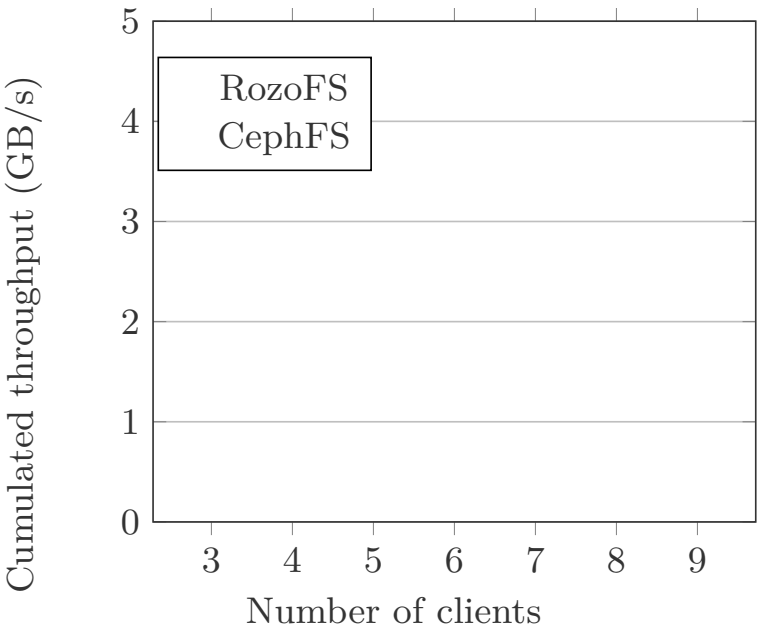
Possible competitors:

- GlusterFS
- Tahoe-LAFS

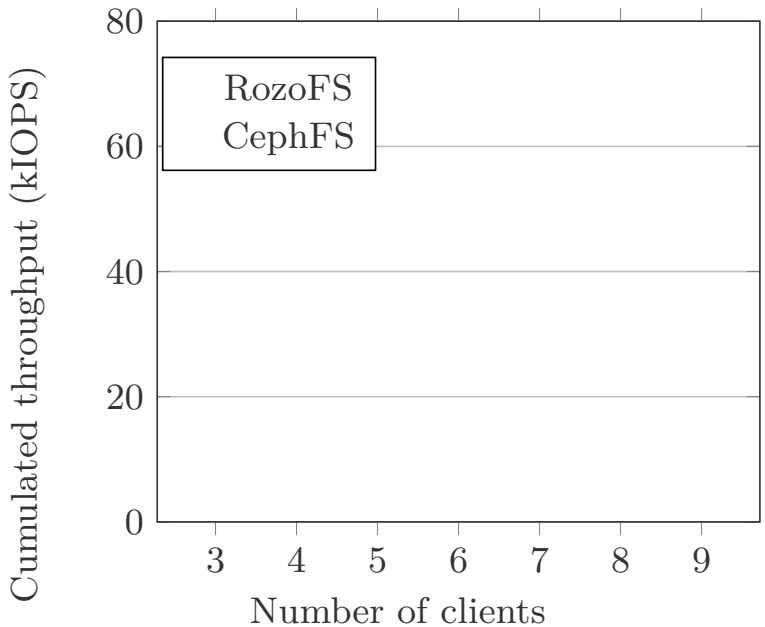
HDFS: Hadoop Distributed File System

Write evaluation

(a) Sequential write

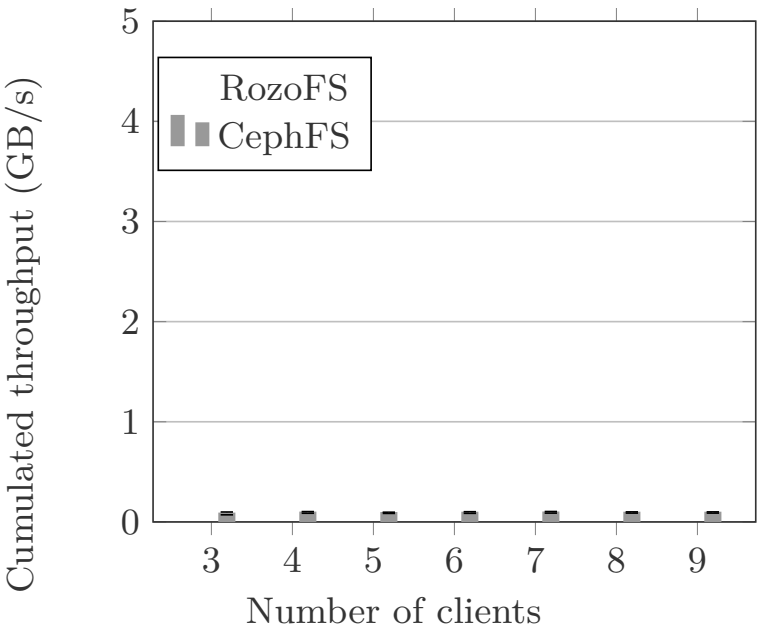


(b) Random write

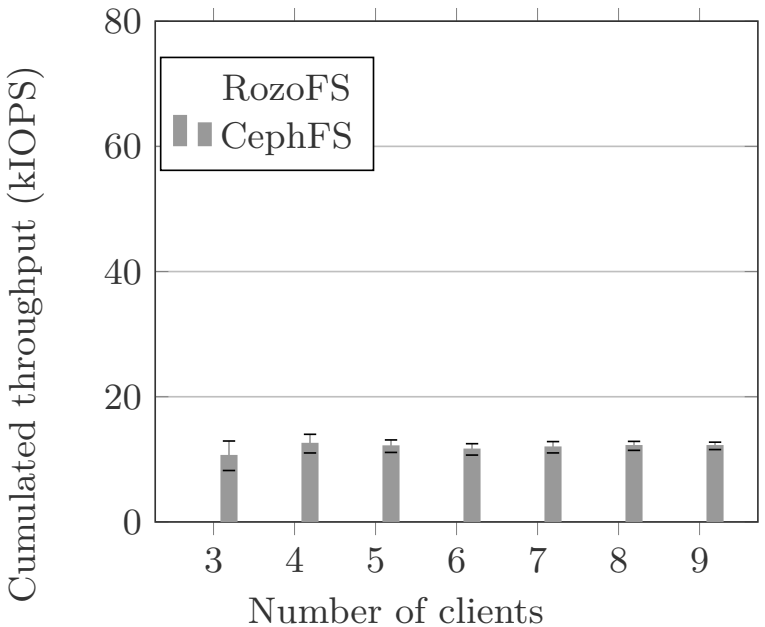


Write evaluation

(a) Sequential write

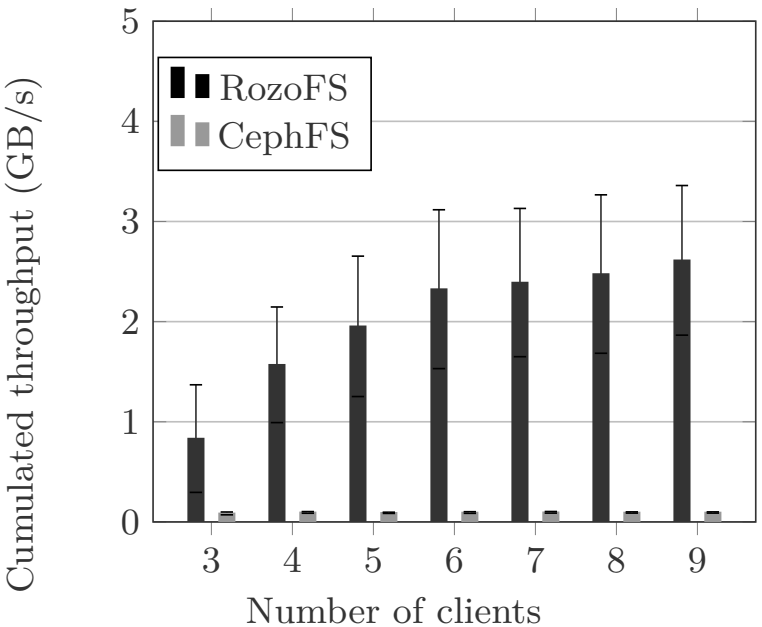


(b) Random write

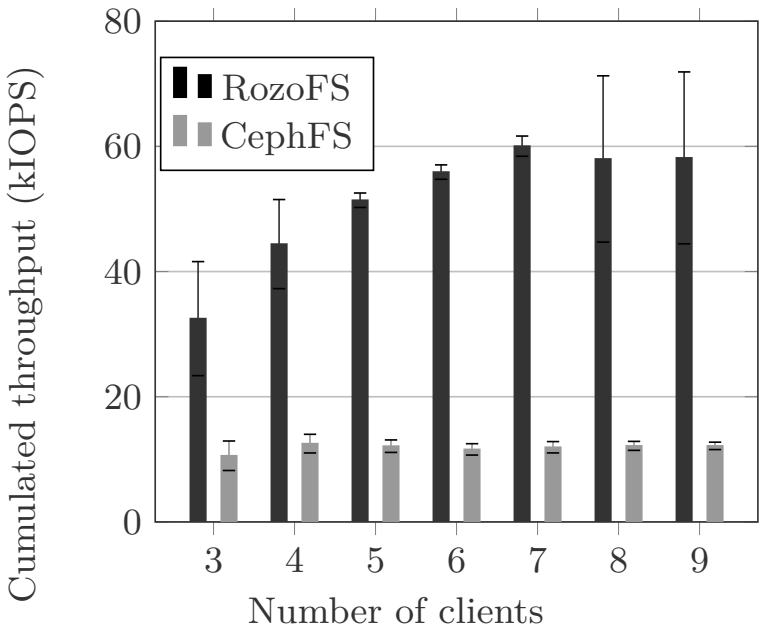


Write evaluation

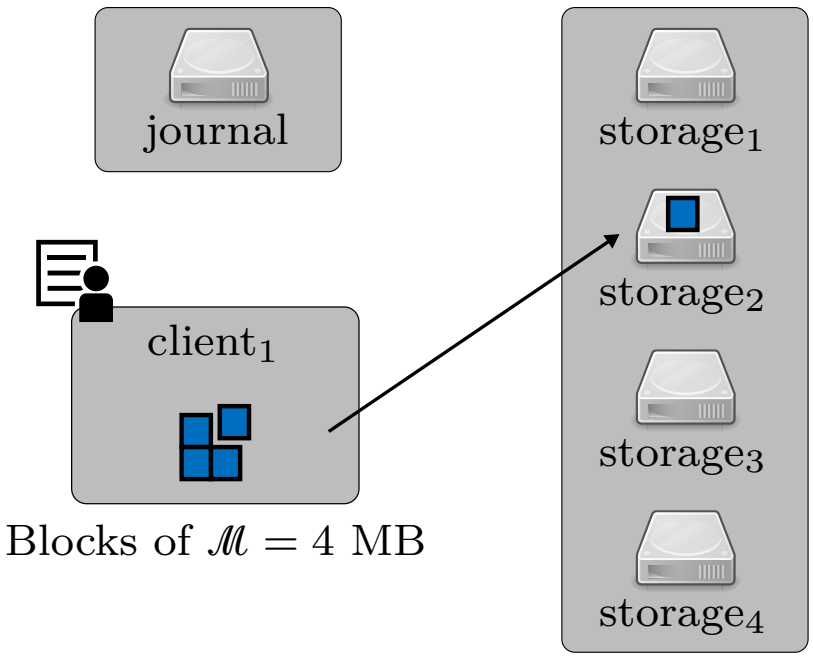
(a) Sequential write



(b) Random write

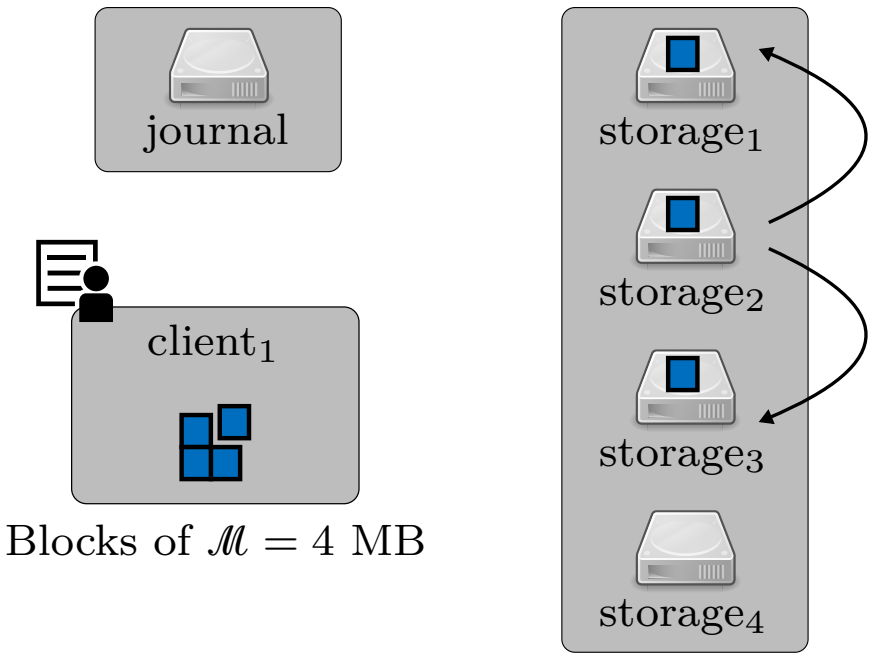


Ceph architecture



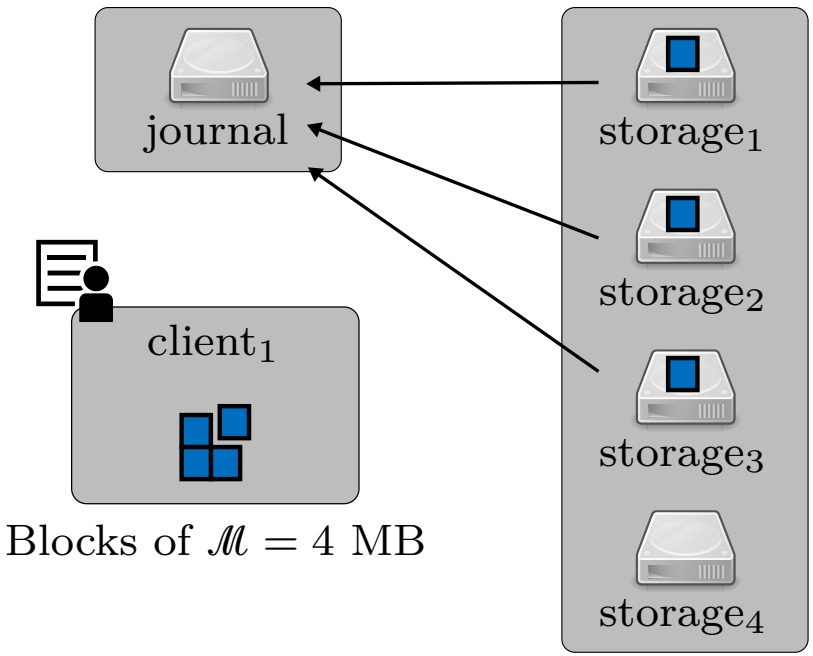
CephFS architecture using triplication and journal

Ceph architecture



CephFS architecture using triplication and journal

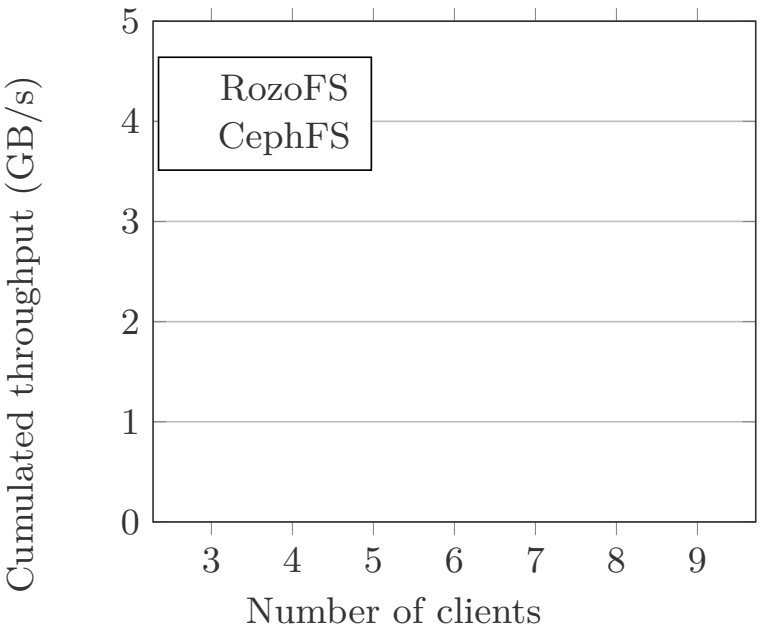
Ceph architecture



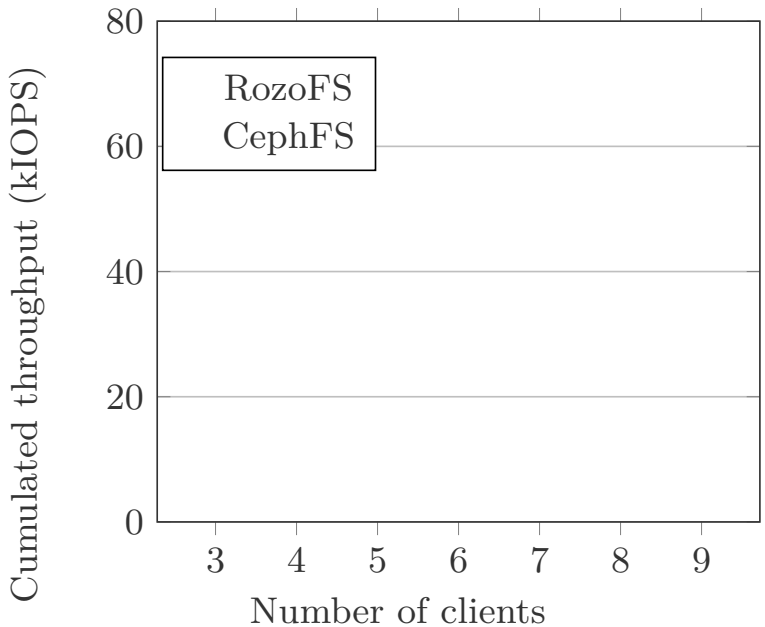
CephFS architecture using triplication and journal

Read evaluation

(a) Sequential read.

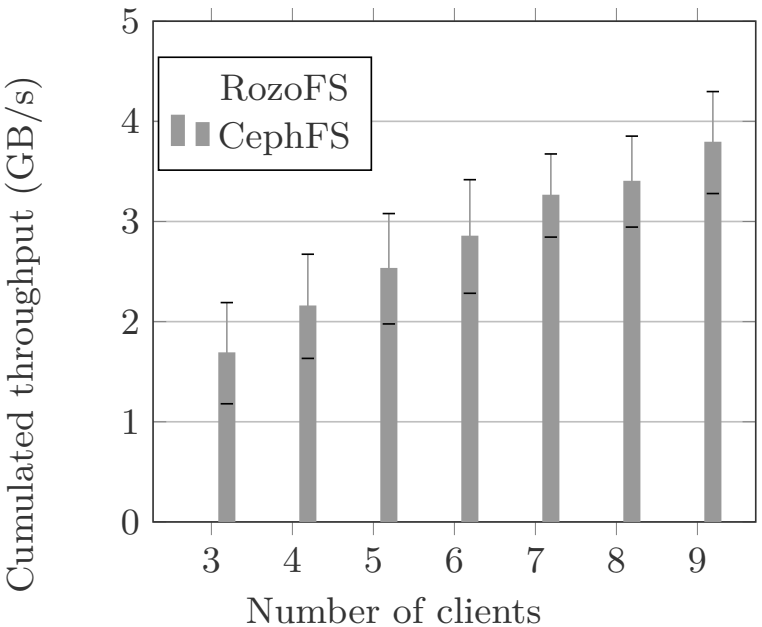


(b) Random read.

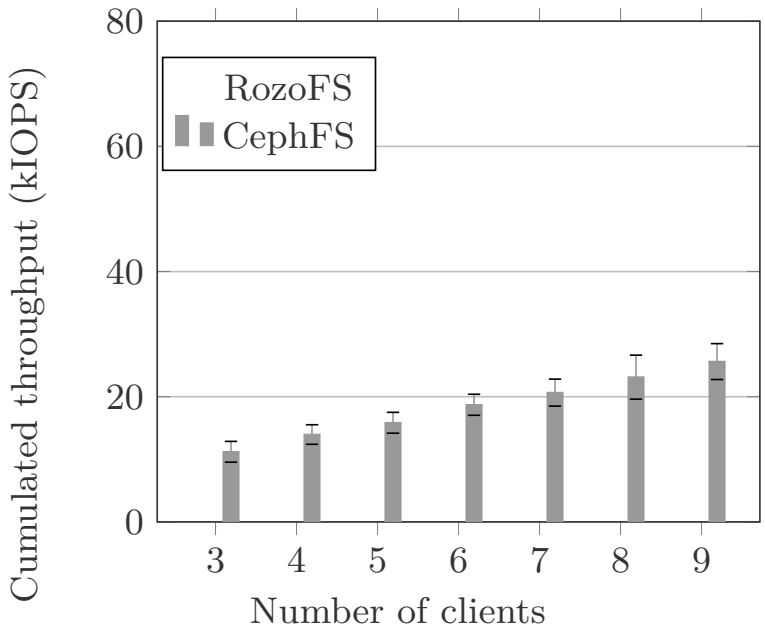


Read evaluation

(a) Sequential read.

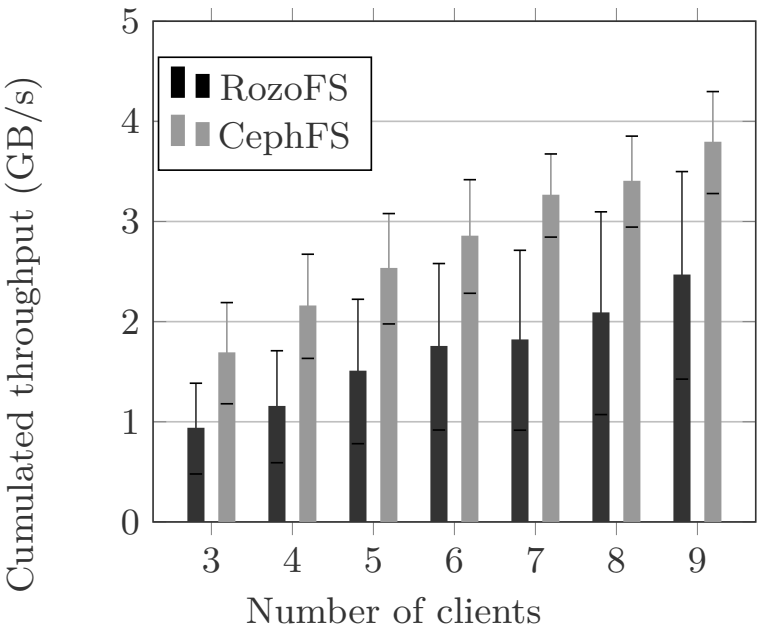


(b) Random read.

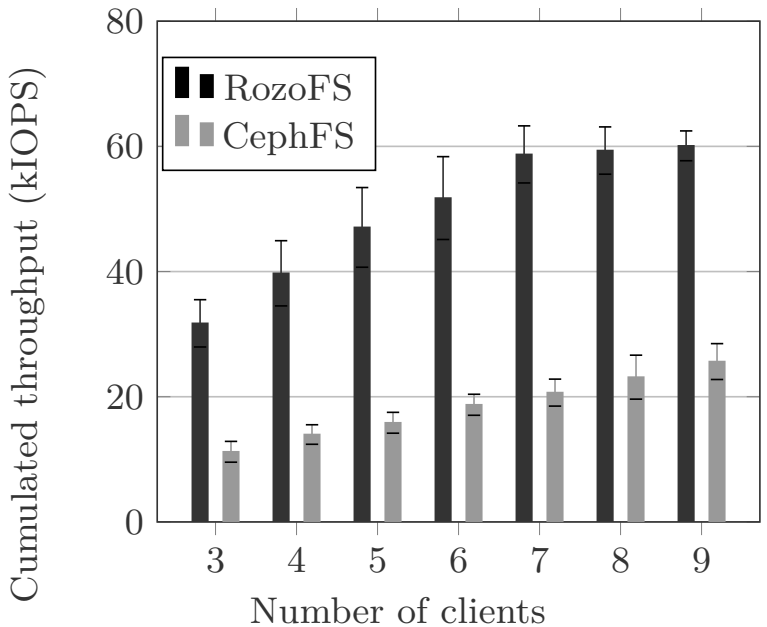


Read evaluation

(a) Sequential read.



(b) Random read.



Evaluation of the storage consumption

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	Total
RozoFS	5.2	5.2	5.1	5.1	5.2	5.2	5.2	5.2	41.4
CephFS	14	9.6	8.5	12	12	9.8	12	12	89.9

Table: Evaluation of the storage consumption (in GB) for RozoFS and CephFS. Each of the 8 nodes in the cluster is depicted as s_i.

Conclusion on RozoFS

1. RozoFS can manage both hot and cold data (up to 60 kIOPS)
2. ...while providing fault-tolerance by the Mojette erasure code
3. ...and cuts by half the storage volume compared to CephFS.

- Dimitri Pertin et al. "Distributed File System Based on Erasure Coding for I/O-Intensive Applications". In: Proceedings of the 4th International Conference on Cloud Computing and Services Science. CLOSER. Barcelona, Spain, Apr. 2014, pp. 451–456
- Dimitri Pertin et al. "RozoFS: an erasure-coded distributed file system for I/O intensive workloads". Submitted to ACM Transactions on Storage (Dec. 2015)

Outline

1. State of the Art

2. Systematic Mojette Erasure Code

3. Reprojection without reconstruction

4. Distributed Storage System: RozoFS

5. Conclusion

Conclusion (1/3)

1. The systematic Mojette erasure code:
   - we designed an algorithm based on [Normand et al., 2006]
   - we showed our implementation in C outperforms:
     - the non-systematic version of Mojette-NS
     - up to 3× (encoding and decoding) RS codes from ISA-L
   - and only 3% overhead compared to optimal codes (near-MDS)

Conclusion (2/3)

2. Reprojection without reconstruction:
   - we designed a distributed method to compute new projections
   - ...without rebuilding the original data
   - we implemented this method in C
   - we showed it outperforms by a factor of 2 the classic method

Conclusion (3/3)

3. Application in a distributed file system RozoFS:
   - we embedded the systematic Mojette erasure code in RozoFS (client-side) to provide fault-tolerance
   - we showed RozoFS can manage both cold and hot data (60 kIOPS)
   - and outperforms CephFS by a factor 3 in random accesses
   - ...while dividing the volume of stored data by 2

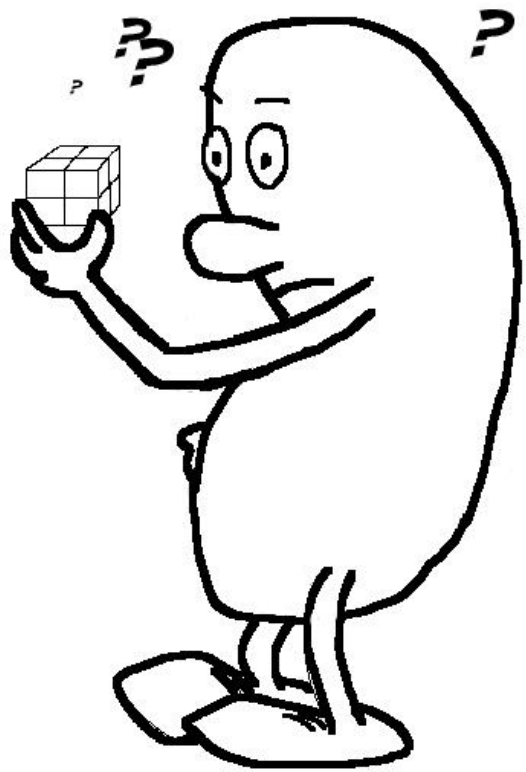
Future works: code (1/2)

1. Explore the links with other codes:
   - relations between Mojette and LDPC codes [Gallager, 1962]
   - iterative reconstruction algorithm
2. Analysis of parity relations between bins of different projections:
   - we discovered that groups of bins can sum to zero
   - reconstruct erroneous bins without transferring whole projections
   - ongoing work with Şuayb Arslan

Future works: storage systems (2/2)

1. Going further on RozoFS experimentations:
   - active benchmarking [Gregg, 2013]
   - impact of an increasing number of storage servers
   - impact of errors on performances
2. Distributed management of metadata:
   - metadata server is a single point of failure
   - distribute metadata as Mojette projections
   - ongoing work in the thesis of Bastien Confais (Oct 2015)

Thank you.



Communications

1. International journal (w/ committee)

   Dimitri Pertin et al. “RozoFS: an erasure-coded distributed file system for I/O intensive workloads”. Submitted to ACM Transactions on Storage (Dec. 2015).

2. International conferences (w/ committee)

   Dimitri Pertin and Nicolas Normand. “Re-projection without Reconstruction”. In: 9ème Journées du Groupe de travail de Géométrie Discrète, Reims Image 2014. Reims, France, Nov. 2014, p. 43.

   Dimitri Pertin et al. “Comparison of RAID-6 Erasure Codes”. In: The third Sino-French Workshop on Information and Communication Technologies. SIFWICT. Nantes, France, June 2015.

   Dimitri Pertin et al. “Distributed File System Based on Erasure Coding for I/O-Intensive Applications”. In: Proceedings of the 4th International Conference on Cloud Computing and Services Science. CLOSER. Barcelona, Spain, Apr. 2014, pp. 451–456.

   Dimitri Pertin et al. “Spatial Implementation for Erasure Coding by Finite Radon Transform”. In: International Symposium on signal, Image, Video and Communication. Valenciennes, France, July 2012.

   Dimitri Pertin et al. The Mojette erasure code for distributed file systems. Session poster. Amsterdam, The Netherlands: EuroSys’14, Apr. 2014.

3. International communications (w/o acts)

   Benoît Parrein et al. FEC4Cloud: a research project for promoting erasure codes in Cloud storage architectures. Bordeaux, France: Algebra, Codes and Networks (ACN), June 2014.

   Dimitri Pertin. RozoFS: A Distributed File System based on Erasure Coding for I/O Intensive Workloads. Workshop on Storage and Processing of Big Data, WOS 4. Technicolor, Cesson-Sévigné, France, Dec. 4, 2014.

   Dimitri Pertin. RozoFS: A High-Performance Encoded-based Distributed Filesystem. Future Cloud Symposium. Session poster. Inria Conference Centre, Rennes, France: EIT Digital, 19 10, 2015.

   Benoît Parrein et al. “FEC4Cloud: a research project promoting erasure coding for Cloud storage architectures”. In: Rendez-vous de la Recherche et de l’Enseignement de la Sécurité des Systèmes d’Information (RESSI). Troyes, France, May 2015.

   Benoît Parrein et al. RozoFS: a fault tolerant I/O intensive distributed file system based on Mojette erasure code. Toulouse, France: Workshop Autonomic, Oct. 2014.

   Dimitri Pertin and Nicolas Normand. Re-projection without reconstruction. Mojette Day 2015. Polytech Nantes, Nantes, France, Feb. 2015.

4. Demonstrations

   Benoît Parrein et al. Video streaming over RozoFS with fault tolerance. NEM SUMMIT. Cité de Congrès, Nantes, France, Oct. 28, 2013.

   Dimitri Pertin et al. RozoFS: a fault tolerant distributed file system based on the Mojette transform. Third Workshop On Storage and Cloud Computing, WOS 3. Technicolor, Cesson-Sevigné, France, Nov. 2013.

5. Vulgarisations

   Dimitri Pertin. Reduce the storage consumption of your storage clusters with RozoFS. Free and Open Source Software Developers’ European Meeting FOSDEM’14. Université Libre de Bruxelles, Belgium, Feb. 2014.

   Dimitri Pertin. Stockage distribuée à la Mojette : Gastronomie, Tomographie, Transmission et Stockage. Pas Sage En Seine. Paris, France, Apr. 2014.

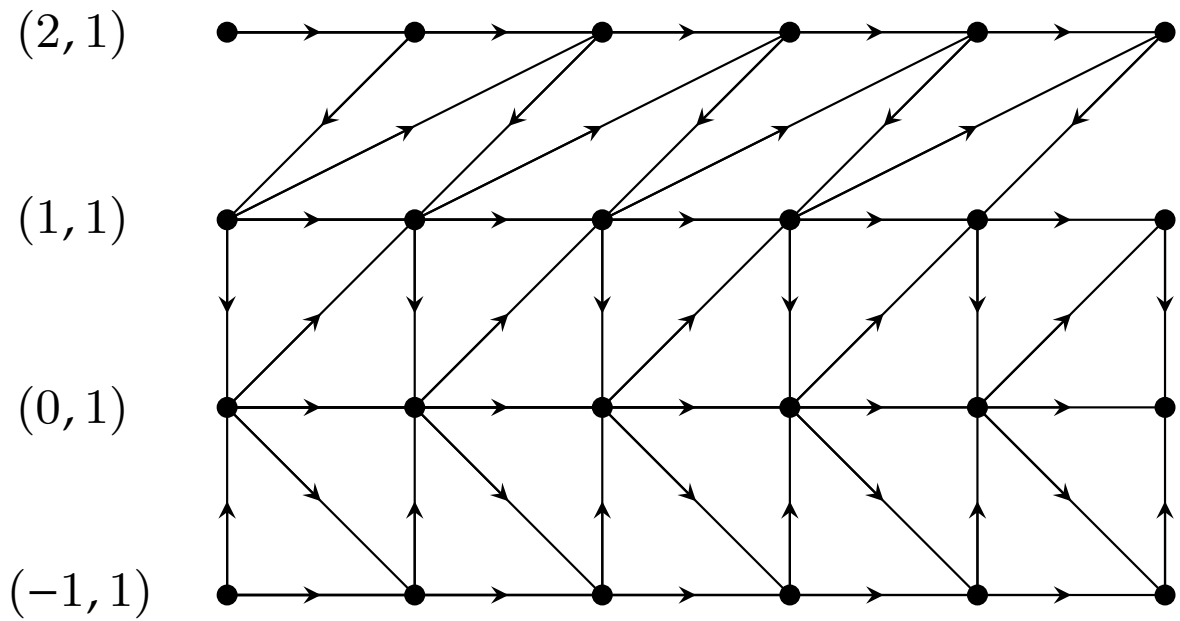
   Dimitri Pertin and Nicolas Normand. RozoFS: Le Système de Fichiers Distribués basé sur un code à effacement. Forum de Médias. Montaigu, France: Lycée Léonard de Vinci, Feb. 2014.

Projection size

$$B(P, Q, p_i, q_i) = (Q - 1)|p_i| + P \tag{1}$$

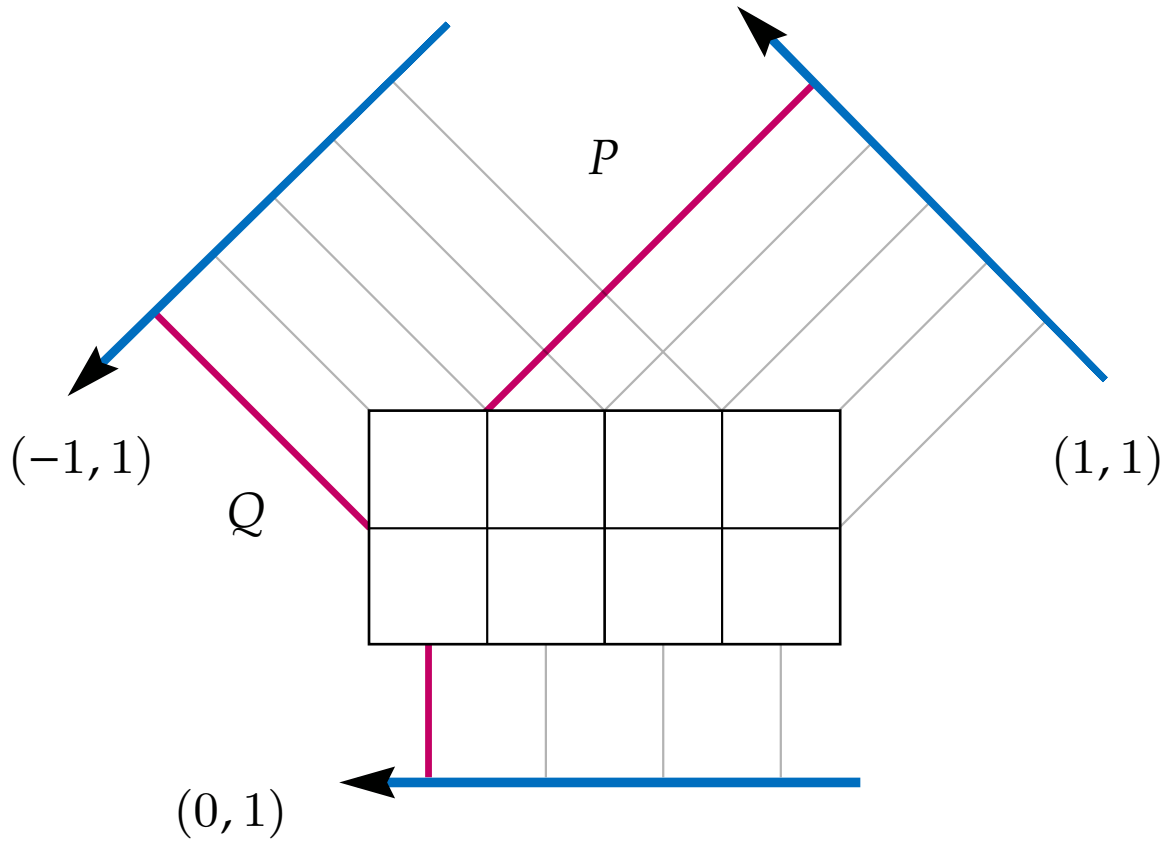
$$\mu = \frac{\sum_{i=0}^{n-1} B(P, Q, p_i, q_i)}{P \times Q}. \tag{2}$$

Dependence graph

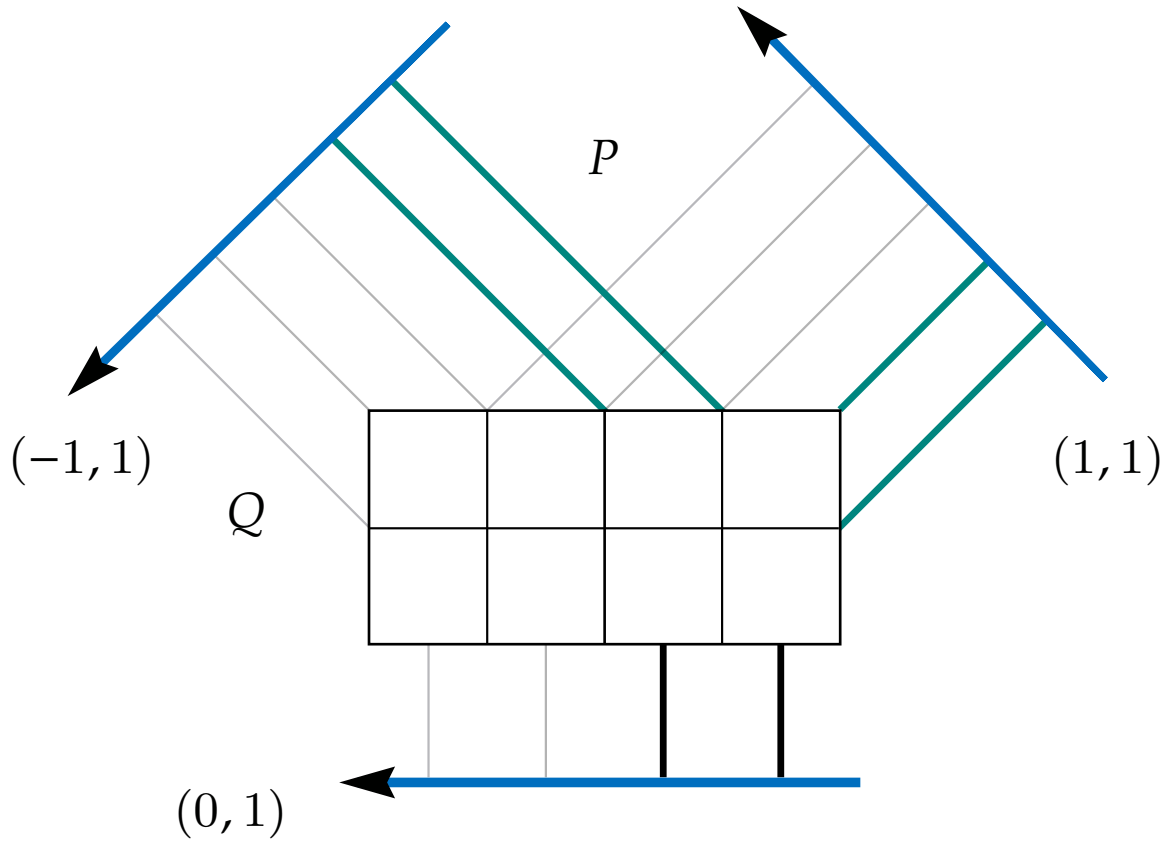


Dependence graph for a (6×4) Mojette code

Parity relations



Parity relations



RozoFS layouts

Layout	k	n	storage nodes	robustness
0	2	3	4	1
1	4	6	8	2
2	8	12	16	4

Ghost decomposition

If Katz is invalid, it is not possible to reconstruct uniquely the grid

$$f = f_{SC} + \sum_i a_i g_i \tag{3}$$

- start rebuilding grid
- f_{SC} is the rebuilt image when zone erodée is null
- f is decomposed into f_{SC} and convolution of ghosts and coefficients

[Philippe, 1998]

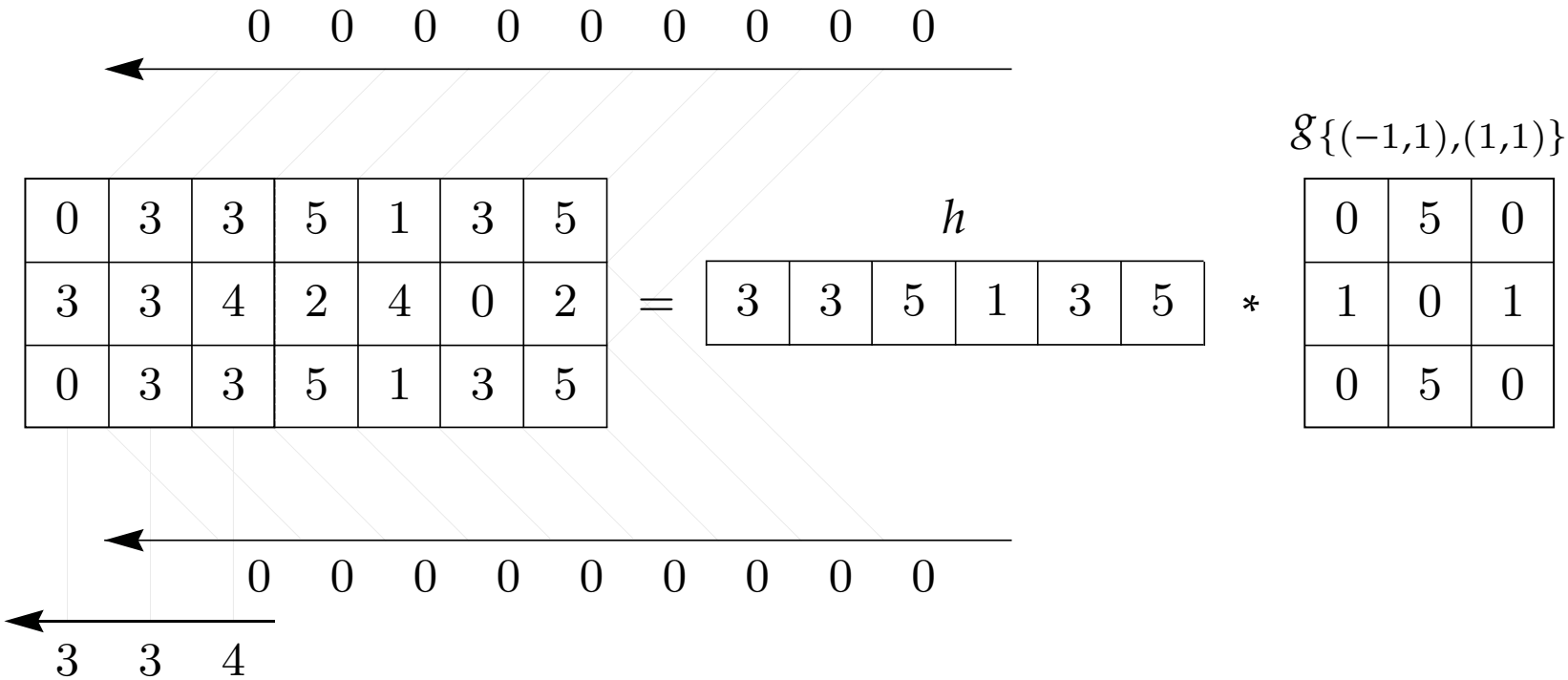
Ghost decomposition

- fix projection values to zero
- $f_{SC} = 0$ (so the reconstruction algorithm ends)
- since we consider only one projection: height of zone erodée is 1

$$f_S^{\{(p_i,q_i=1)\}} = h * g_{S\setminus\{(p_i,q_i)\}} \tag{4}$$

$$M_{\{(p_k,q_k)\}}\left[f_S^{\{(p_i,q_i)\}}\right] = \underbrace{M_{\{(p_k,q_k)\}}[h]}_{h} * M_{\{(p_k,q_k)\}}\left[g_{S\setminus\{(p_i,q_i)\}}\right] \tag{5}$$

Convolution by ghost



Convolution of f with the ghost $g_{\{(-1,1),(1,1)\}}$. The result gives an image whose related projections are null. Operations are done modulo 6.

$$M_{\{(p_i,q_i)\}}\left[f_S^{\{(p_i,q_i)\}}\right] = h * M_{\{(p_i,q_i)\}}\left[g_{S\setminus\{(p_i,q_i)\}}\right] \tag{6}$$

$$M_{\{(p_k,q_k)\}}\left[f_S^{\{(p_i,q_i)\}}\right] = h * M_{\{(p_k,q_k)\}}\left[g_{S\setminus\{(p_i,q_i)\}}\right] \tag{7}$$

$$\begin{aligned} M_{\{(p_k,q_k)\}}\left[f_S^{\{(p_i,q_i)\}}\right] = {} & M_{\{(p_i,q_i)\}}[f] \\ & *^{-1} M_{\{(p_i,q_i)\}}\left[g_{\{S\setminus\{p_i,q_i\}\}}\right] \\ & * M_{\{(p_k,q_k)\}}\left[g_{\{S\setminus\{p_i,q_i\}\}}\right] . \end{aligned} \tag{8}$$