EC-Store

Bridging the Gap Between Storage and Latency in Distributed Erasure Coded Systems

Michael Abebe

Khuzaima Daudjee

Brad Glasbergen

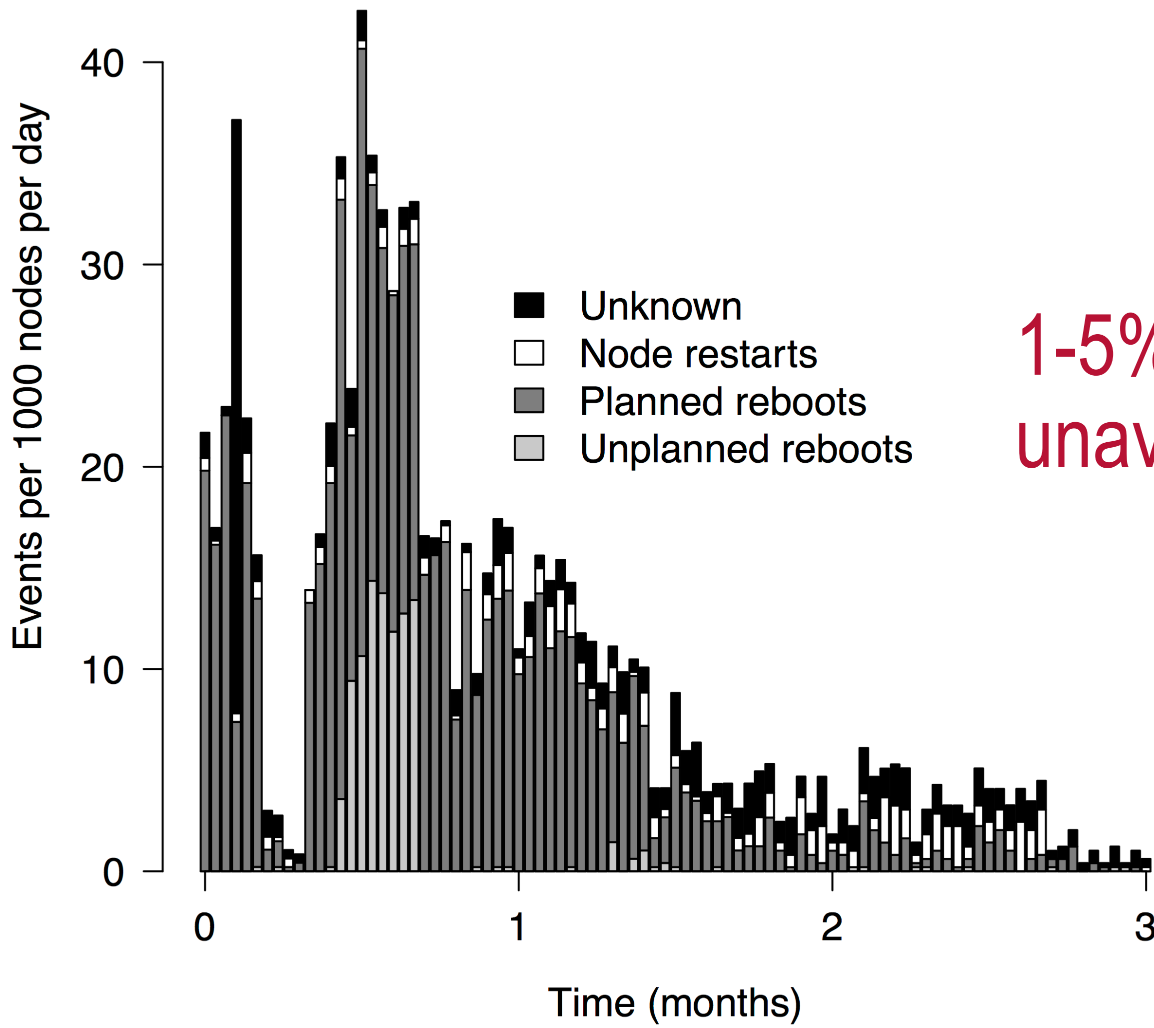
Yuanfeng Tian

UNIVERSITY OF WATERLOO

ICDCS

July 3, 2018

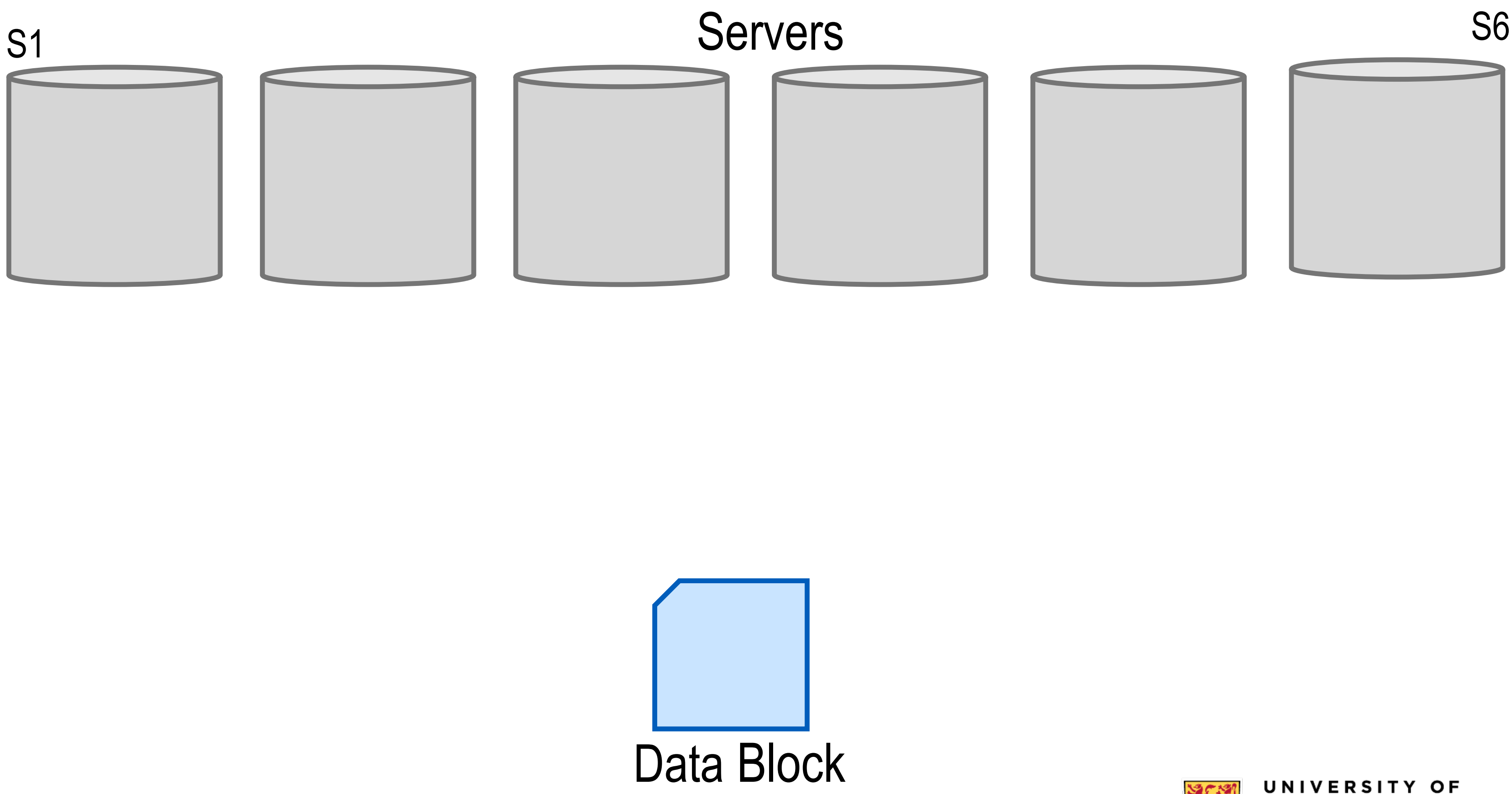
Failures are the Norm



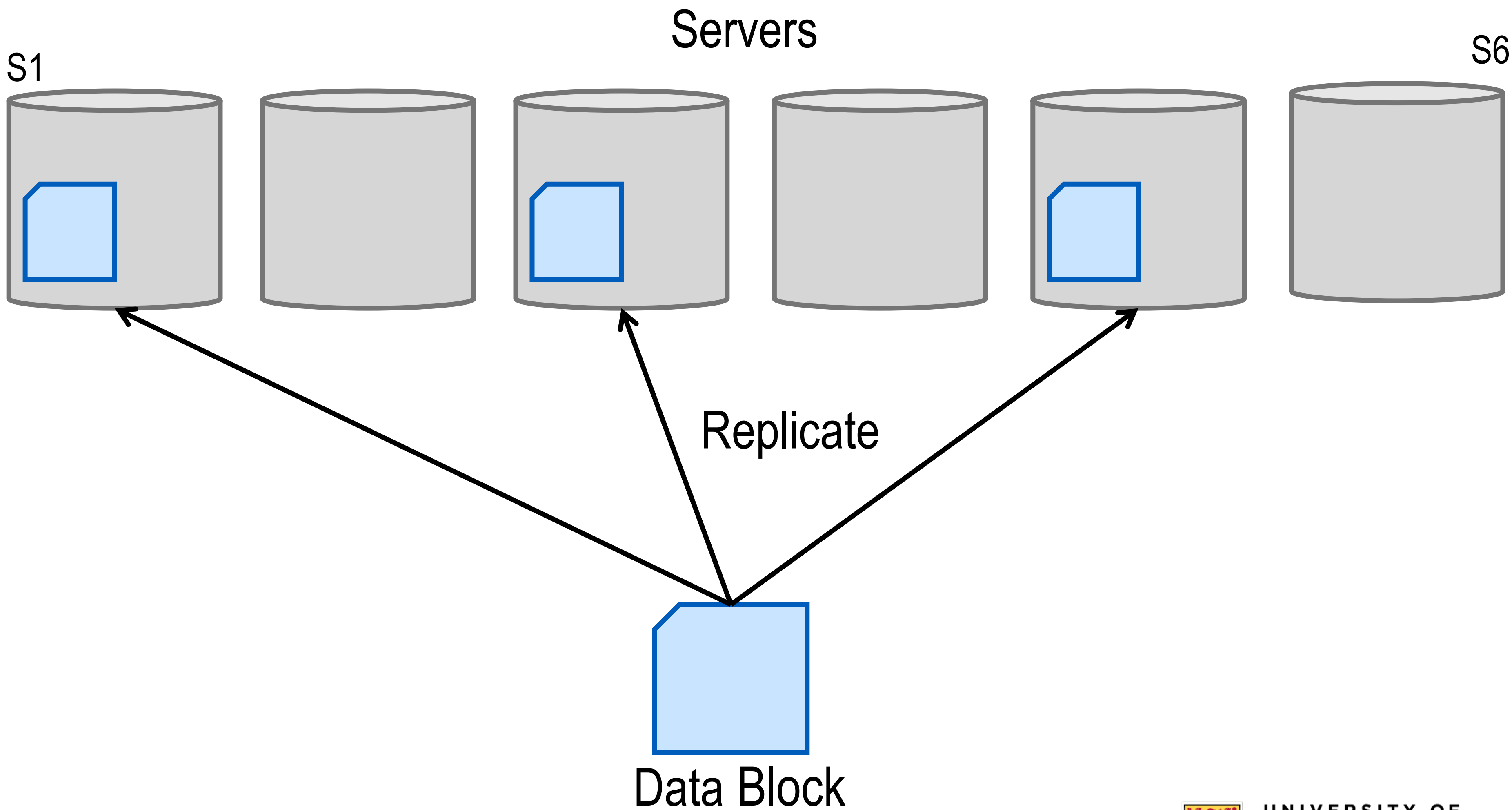
1-5% of cluster unavailable per day

UNIVERSITY OF WATERLOO

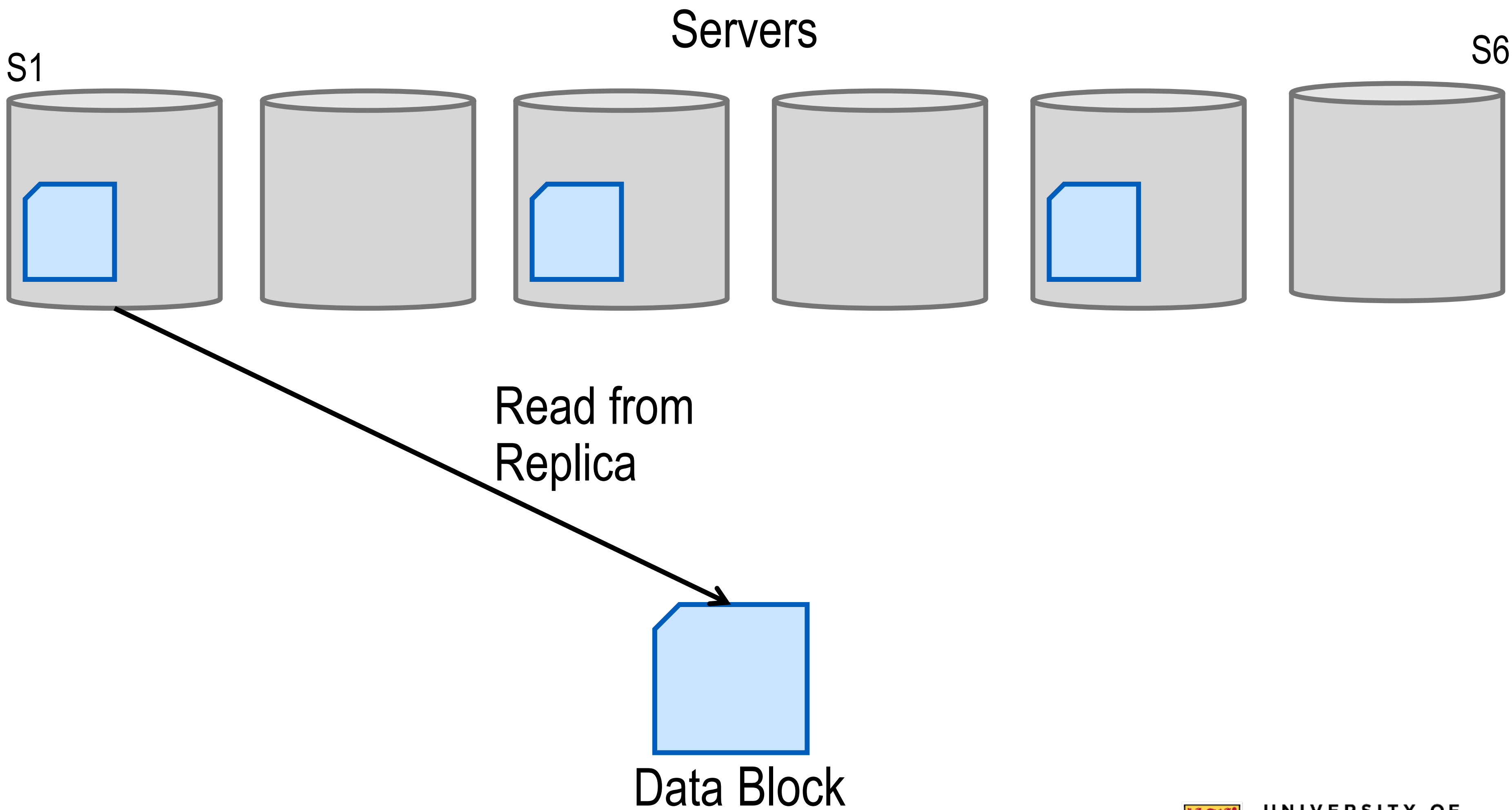
Replicated Storage Systems



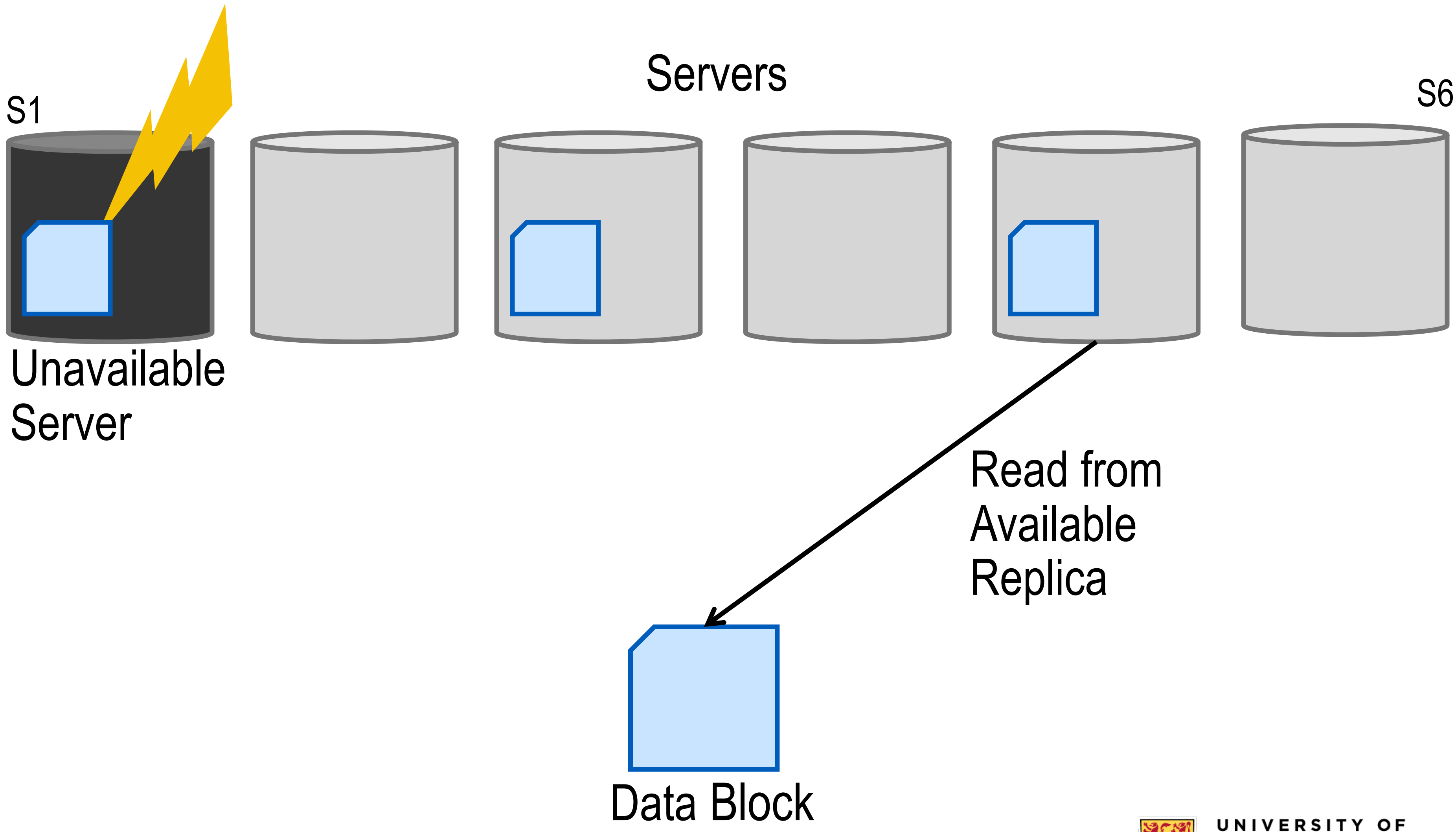
Replicated Storage Systems



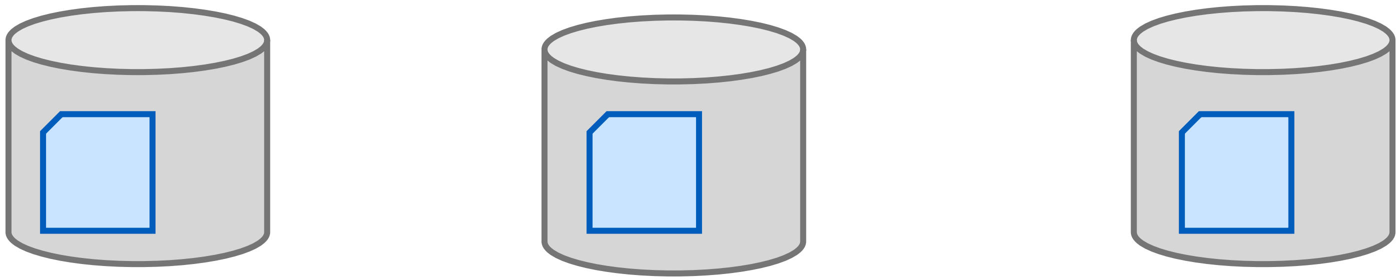
Replicated Storage Systems



Replicated Storage Systems

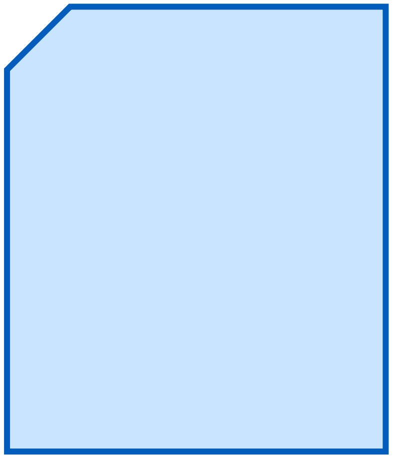


Overhead of Replicated Storage



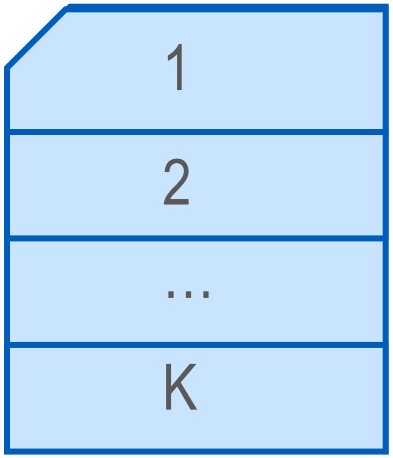
To tolerate R failures need R + 1 copies of data

Towards Lower Overhead Storage



Towards Lower Overhead Storage

Fragment into
K chunks

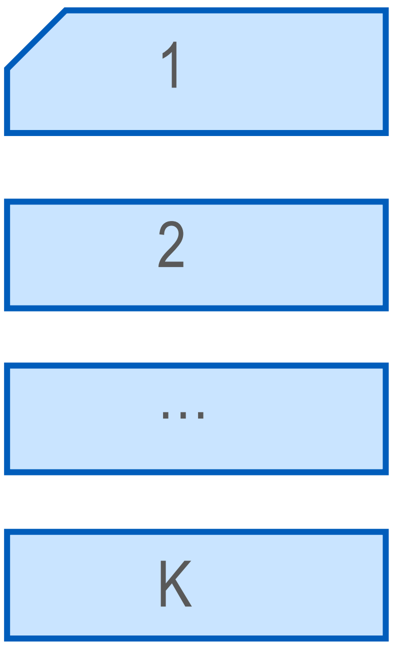


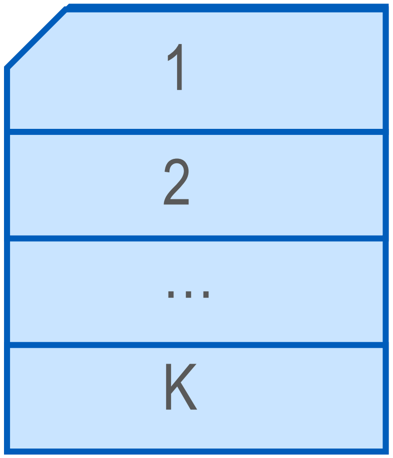
Data Block

Towards Lower Overhead Storage

Fragment into
K chunks

K = # chunks
needed for
reconstruction



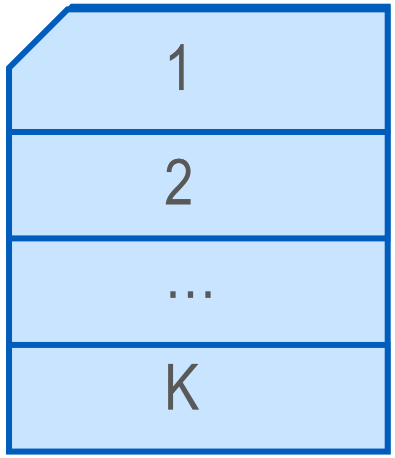


Data Block

Towards Lower Overhead Storage

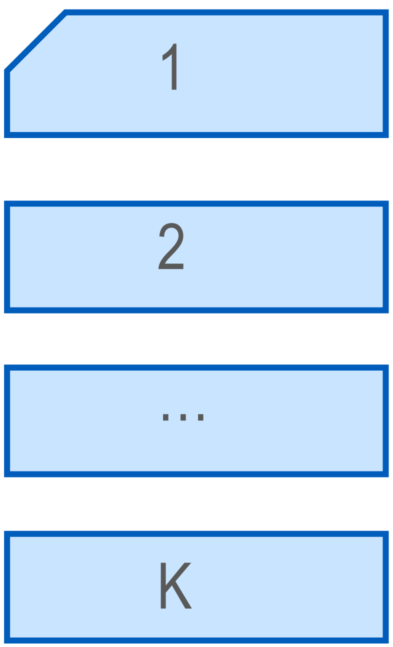
Fragment into K chunks

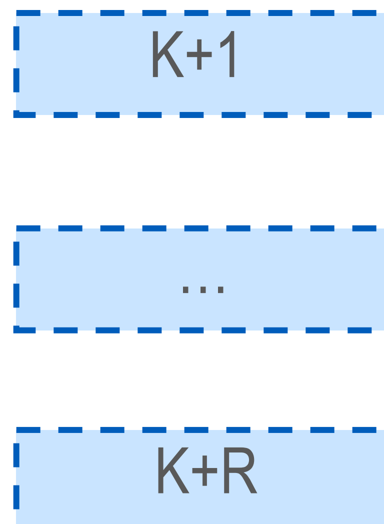
K = # chunks needed for reconstruction



Data Block

Generate R parity chunks





UNIVERSITY OF WATERLOO

Towards Lower Overhead Storage

Fragment into K chunks

K = # chunks needed for reconstruction

1

2

...

K

Data Block

1

2

...

K

Generate R parity chunks

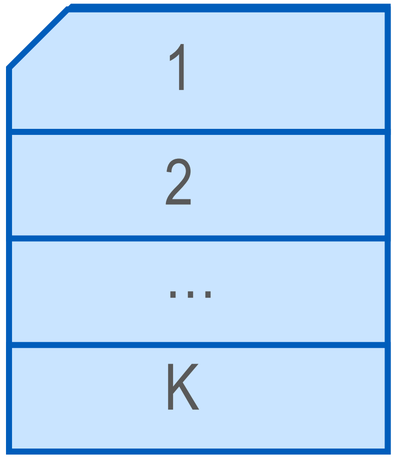
R = # faults tolerated

K+1

...

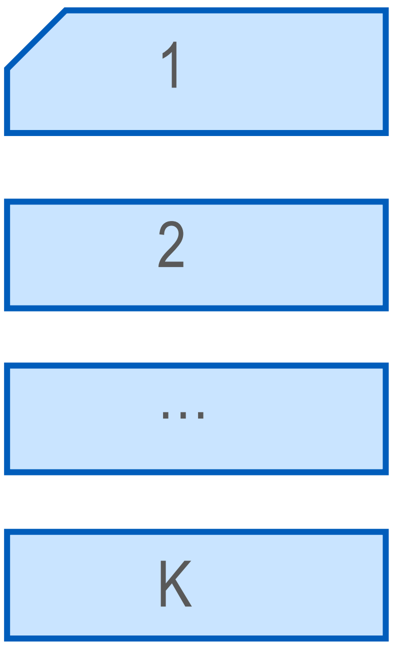
K+R

Towards Lower Overhead Storage



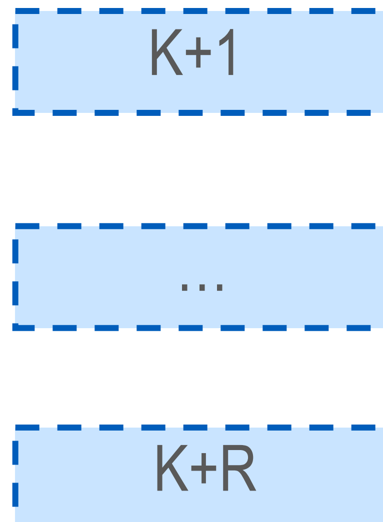
Data Block

K = # chunks needed for reconstruction

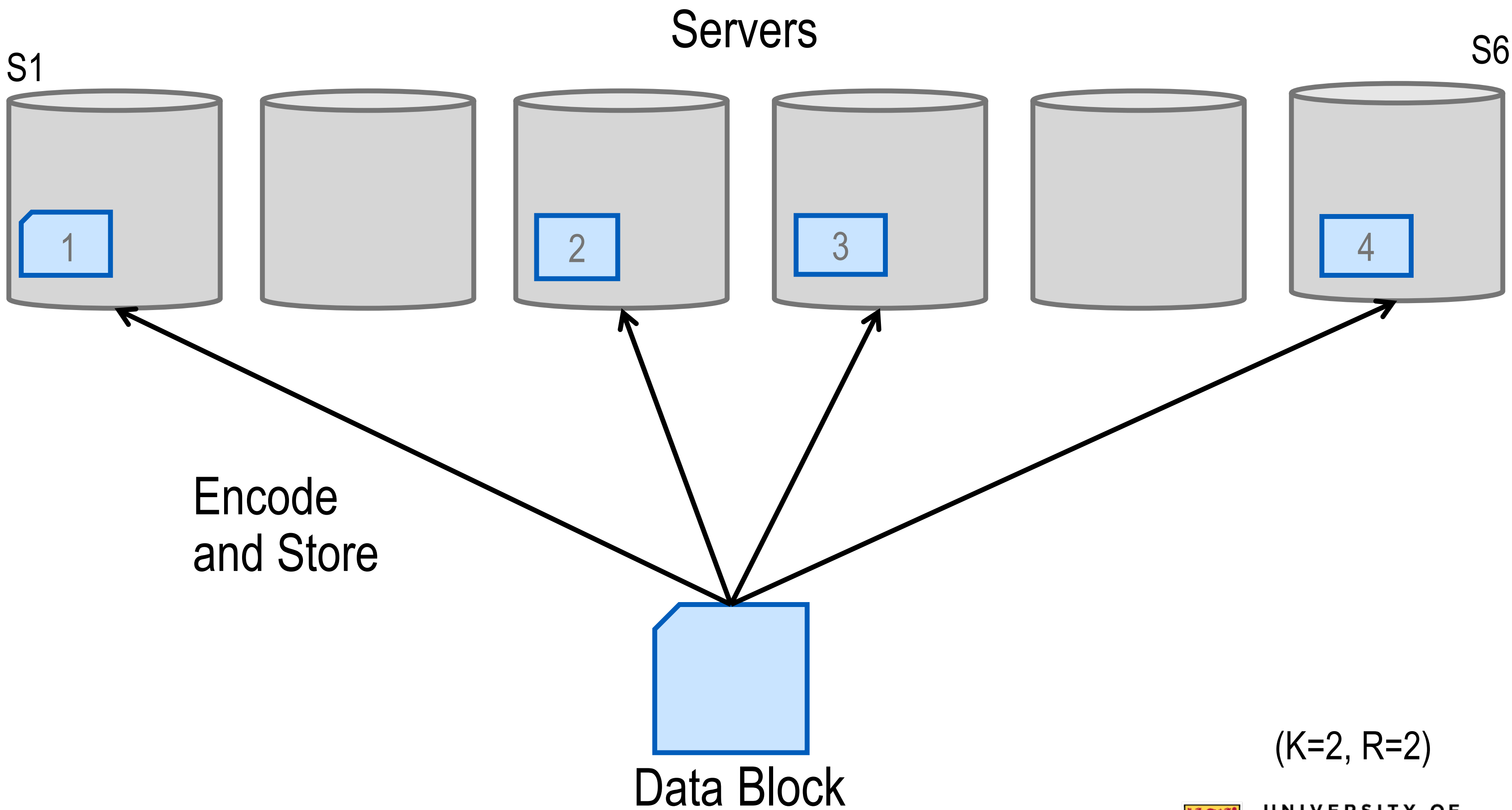


To tolerate R faults, a storage overhead of (K+R)/K

R = # faults tolerated

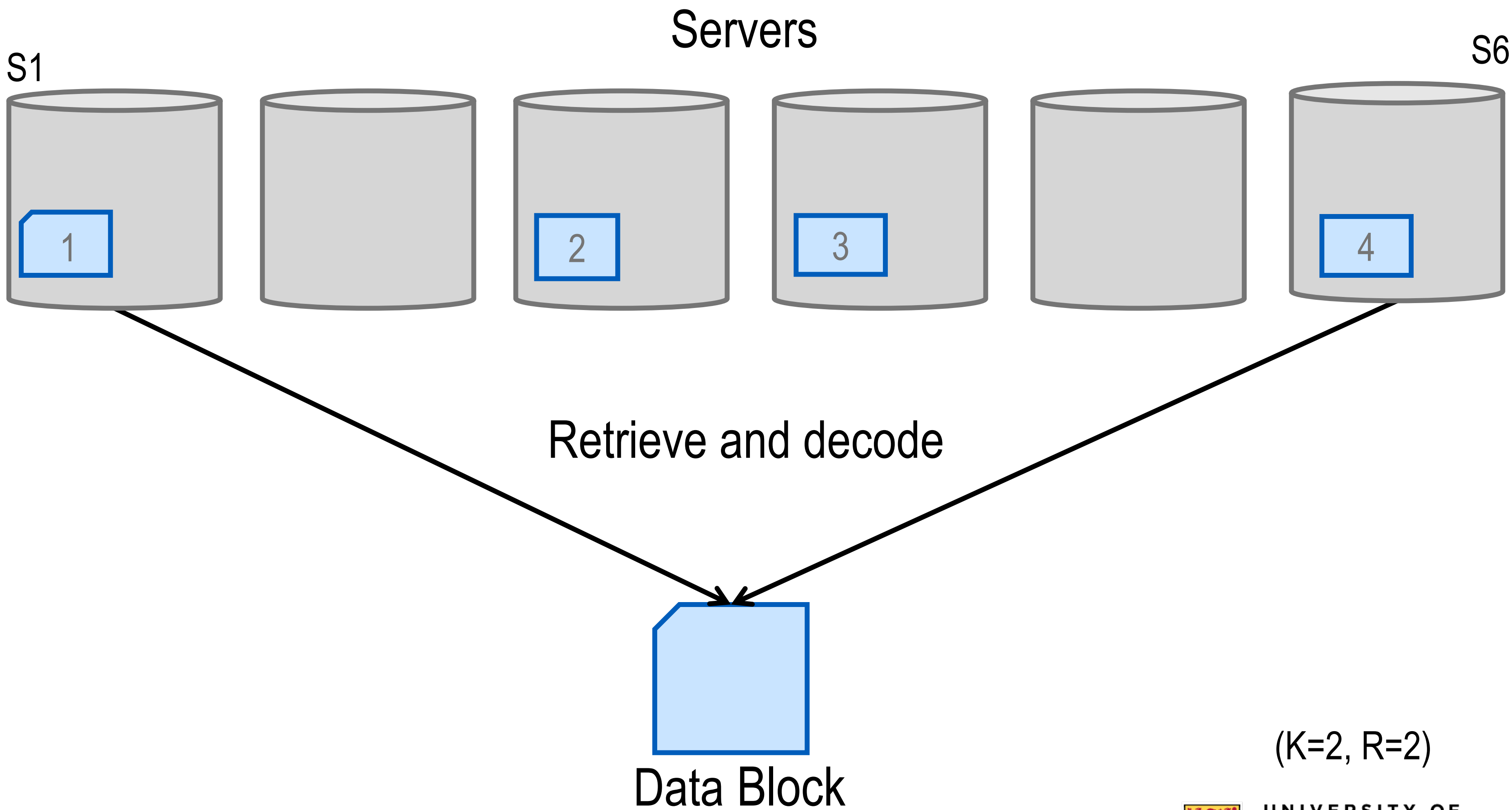


Erasure Coded Storage Systems

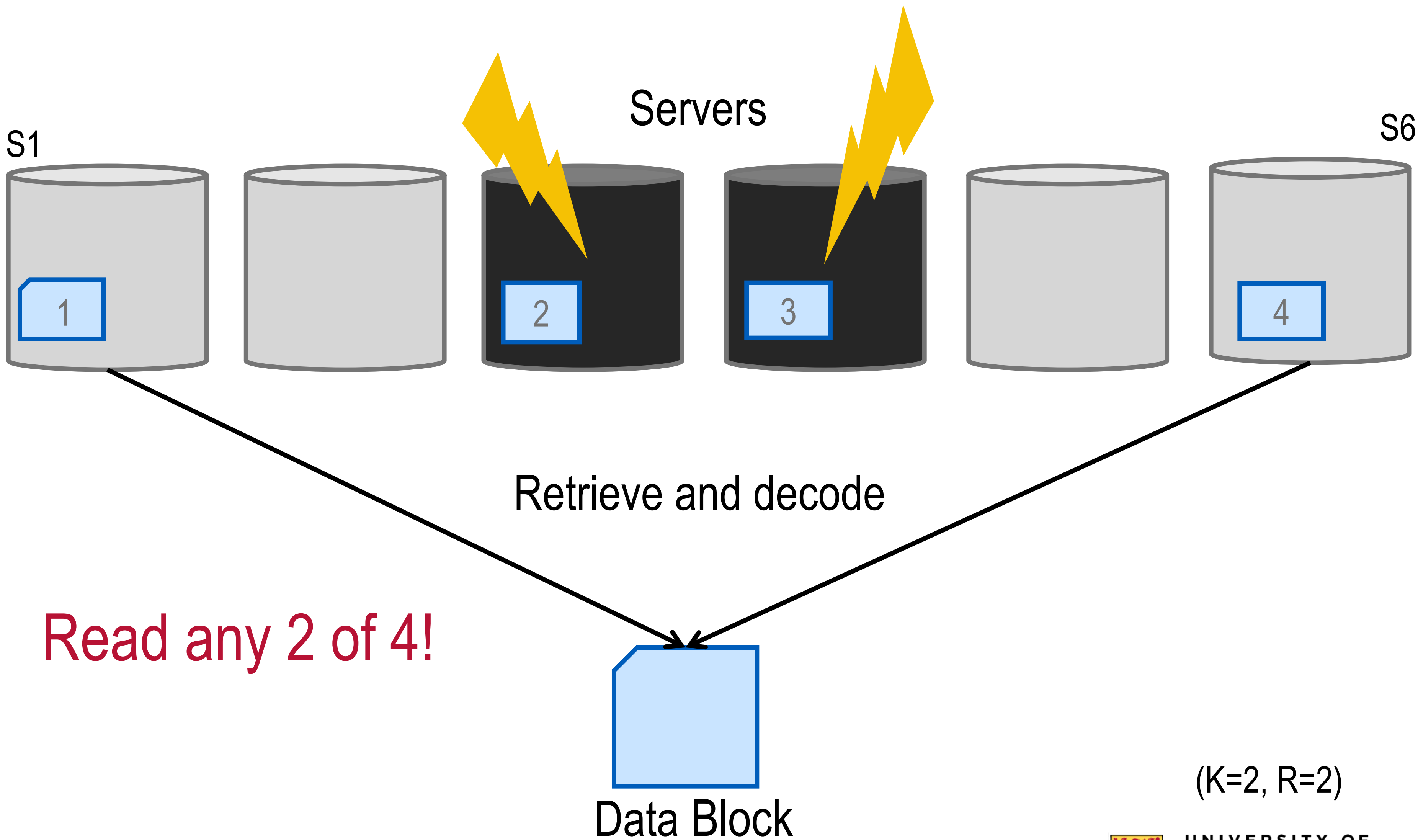


(K=2, R=2)

Erasure Coded Storage Systems



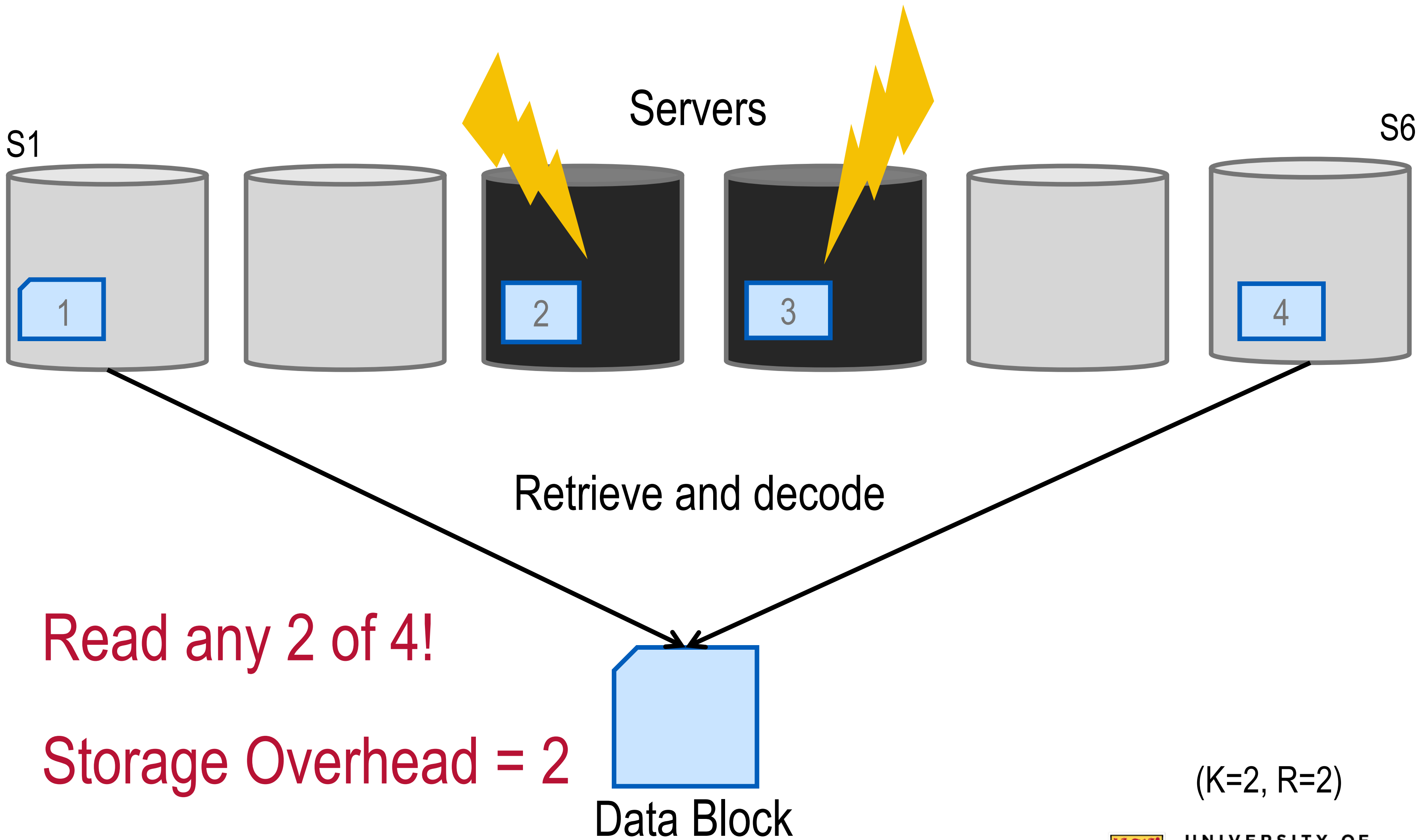
Erasure Coded Storage Systems



Read any 2 of 4!

(K=2, R=2)

Erasure Coded Storage Systems

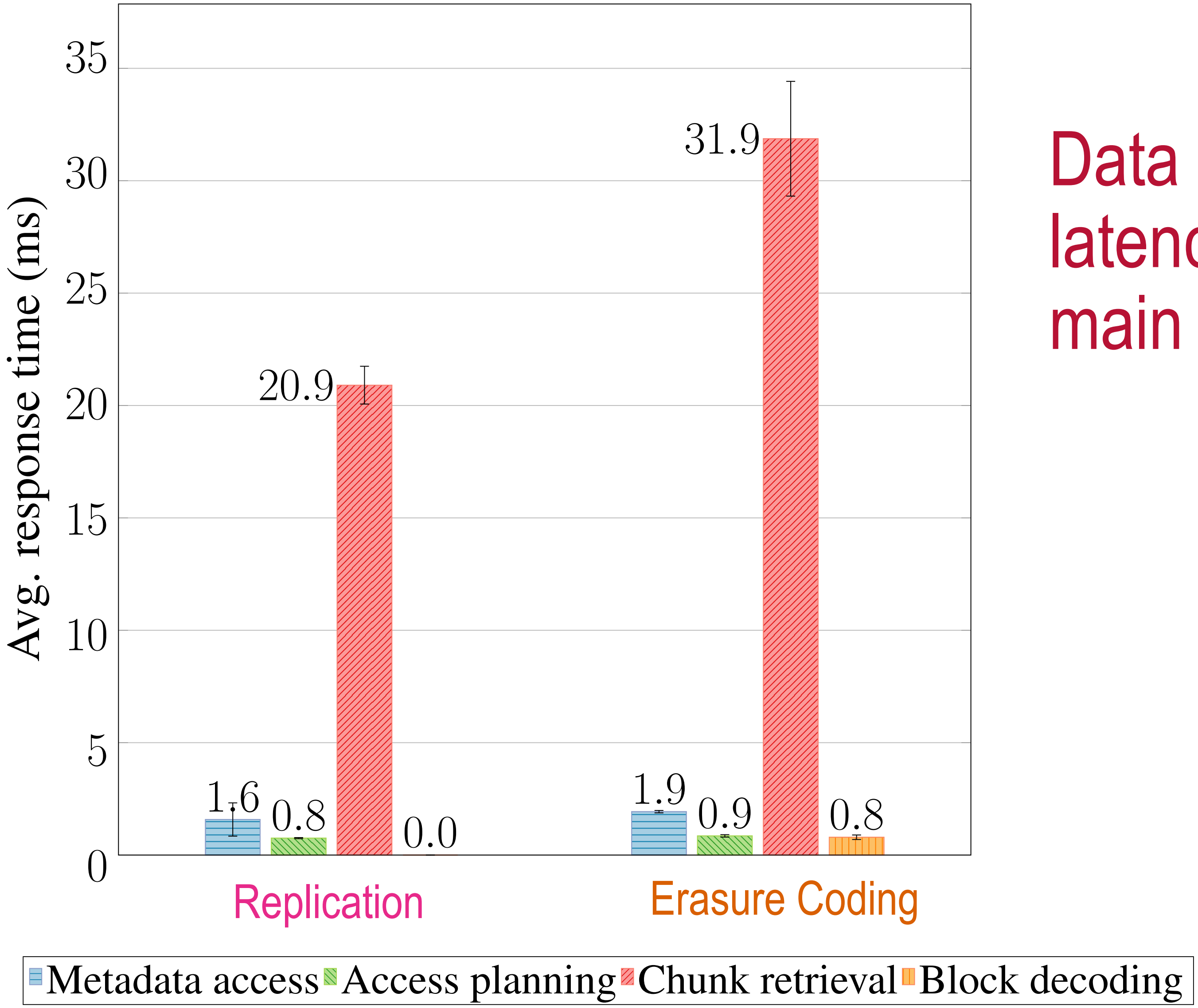


Read any 2 of 4!

Storage Overhead = 2

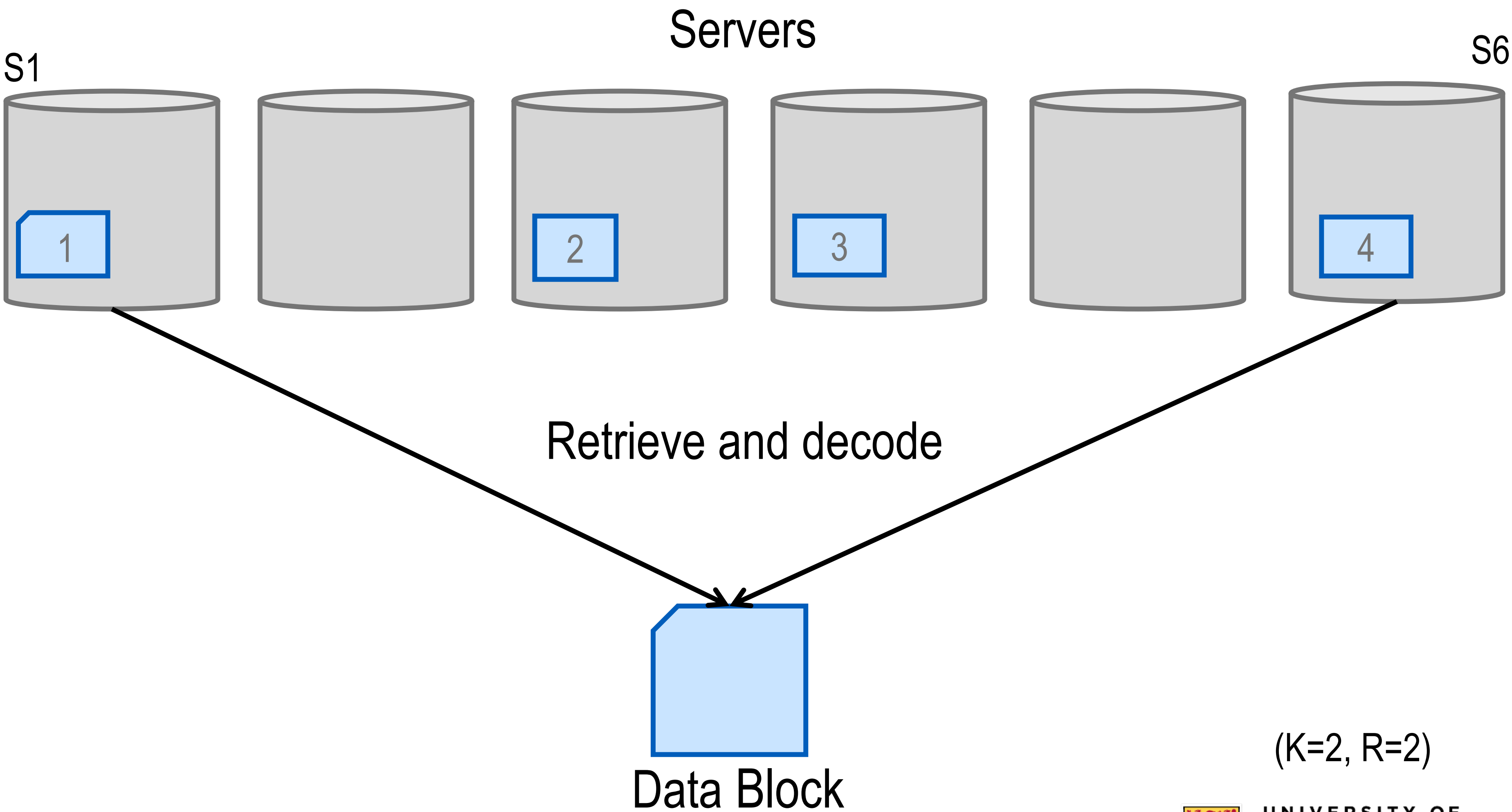
(K=2, R=2)

Breakdown of Response Times (YCSB)



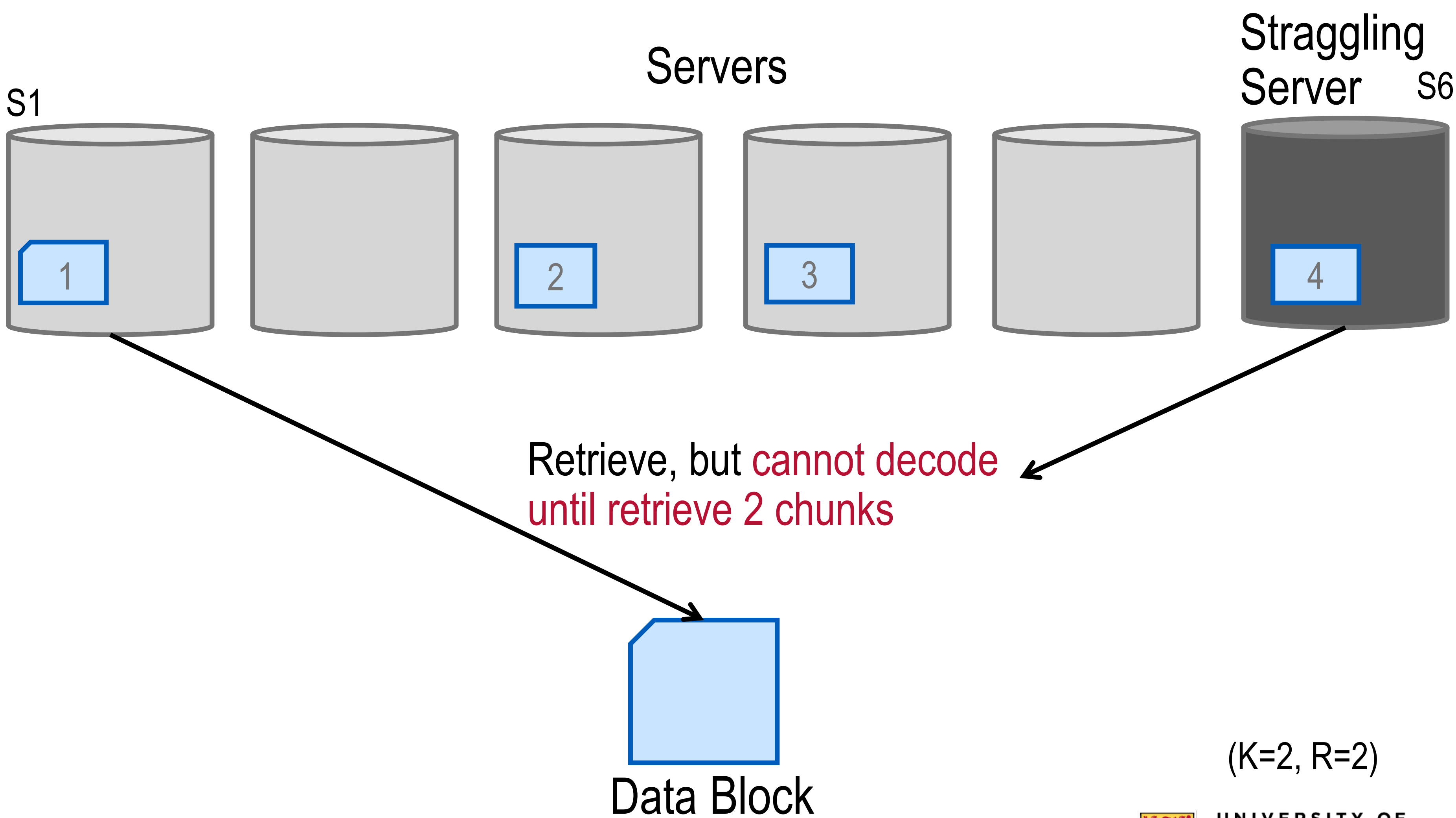
Data access latencies are main difference

Straggling Data Access



(K=2, R=2)

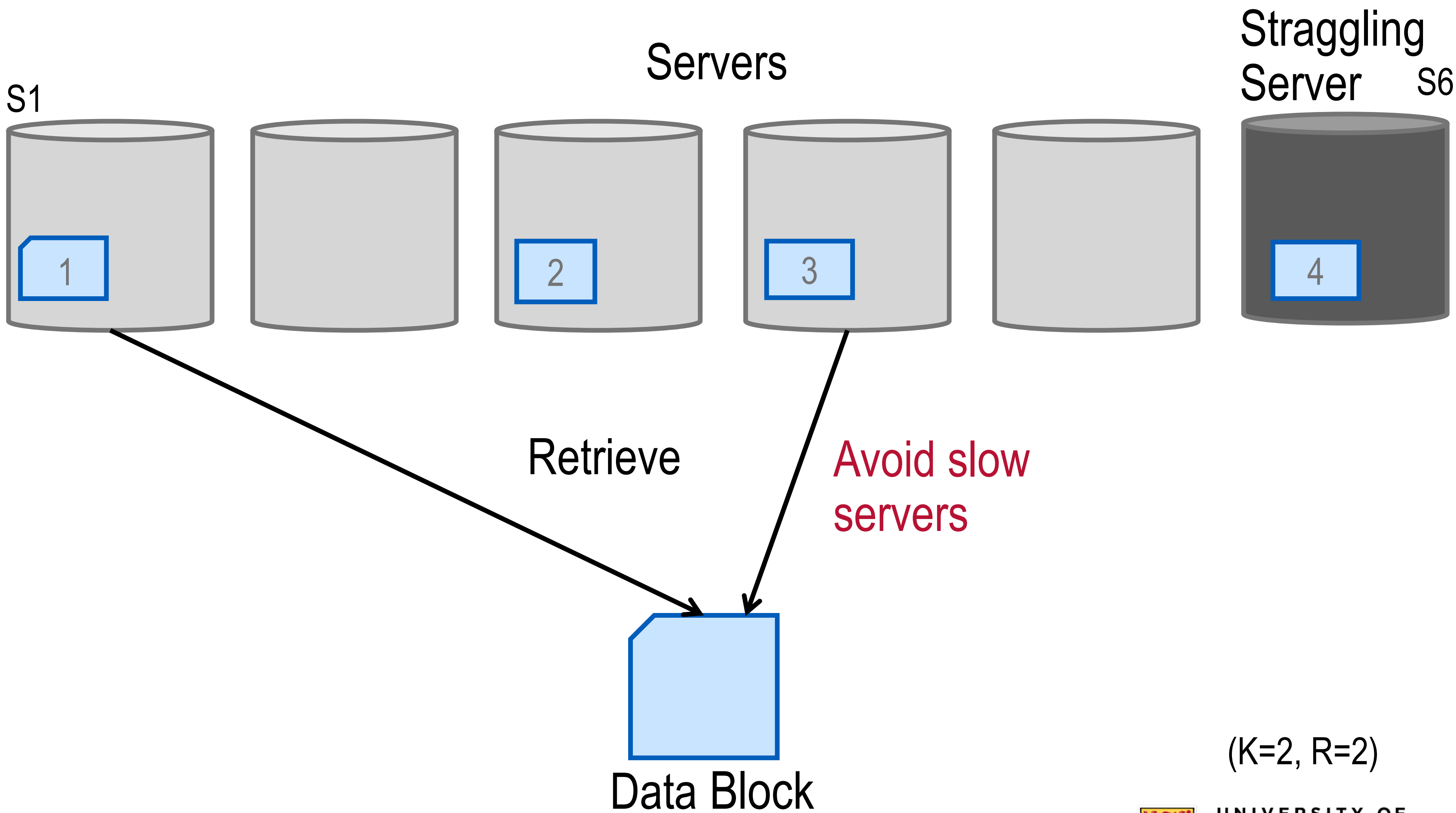
Straggling Data Access



(K=2, R=2)

UNIVERSITY OF WATERLOO

Straggling Data Access



(K=2, R=2)

Latency and Storage Gap

Replication

+ Fast data retrieval

- High storage overhead

Latency and Storage Gap

Replication

+ Fast data retrieval
- High storage overhead

Erasure Coding

+ Low storage overhead
- Slow data retrieval

Latency and Storage Gap

Replication

+ Fast data retrieval

- High storage overhead

Erasure Coding

+ Low storage overhead

- Slow data retrieval

How to Improve Storage System Performance

- Intelligently access data to minimize access costs
- Reduce number of sites accessed through data co-access
- Dynamically move data to improve future accesses

Data Access Strategies

- Data Access Strategies
- Data Movement Strategies
- Complementary Strategies
- Experimental Evaluation

ICDCS

July 3, 2018

UNIVERSITY OF WATERLOO

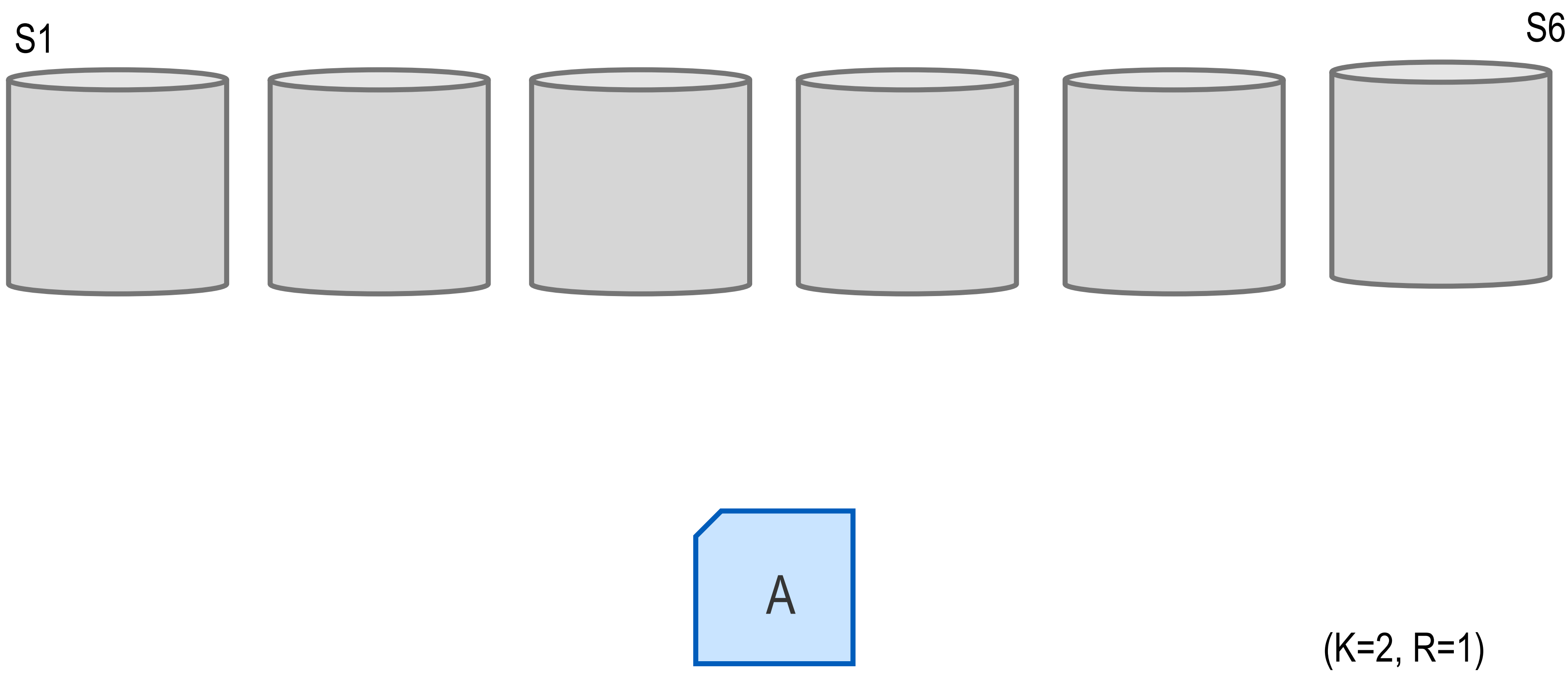
Multi-Item Access

Netherlands

The Netherlands, also known informally as Holland, is a country in Western Europe with a population of seventeen million. It is the main constituent country of the Kingdom of the Netherlands, which is further comprised of three island territories in the Caribbean: Bonaire, Sint Eustatius and Saba.

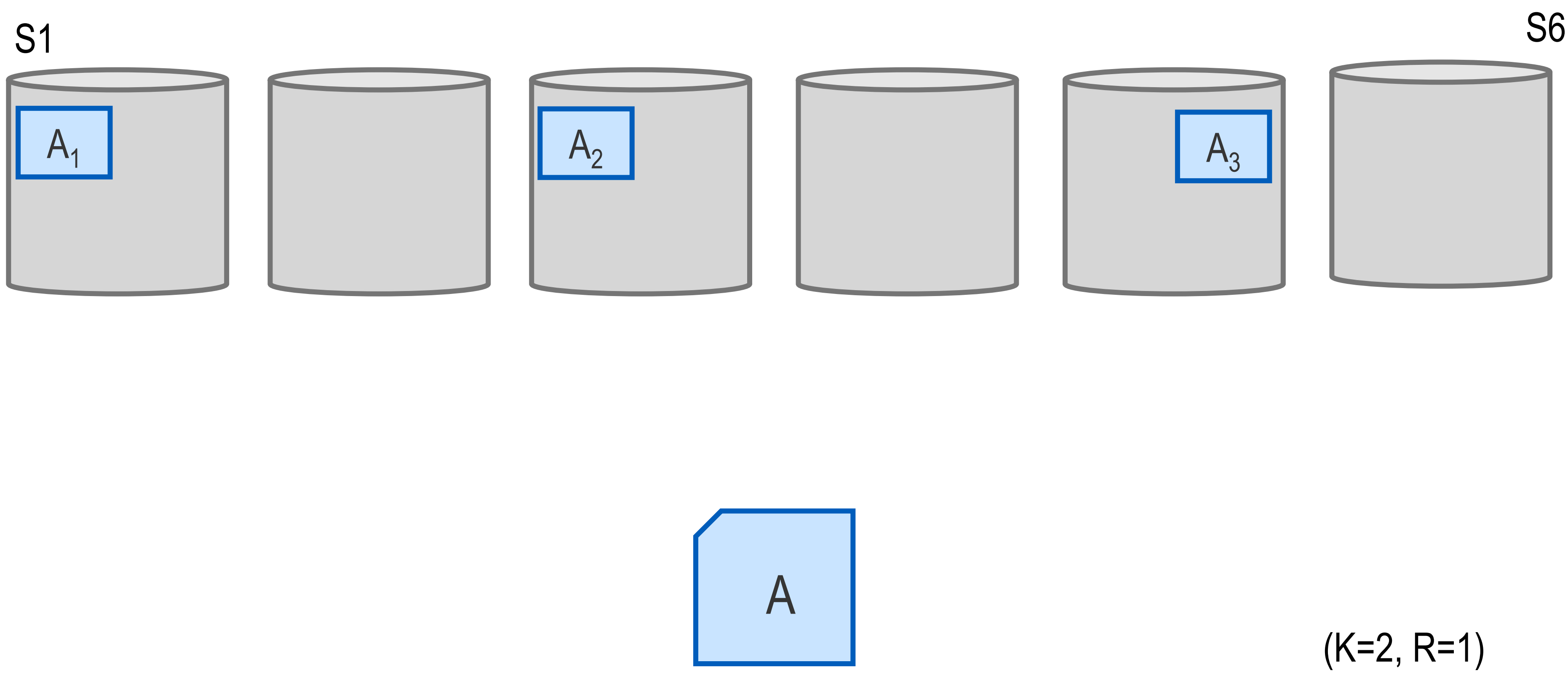
Gallery of Dutch Art

Initial Data Placement



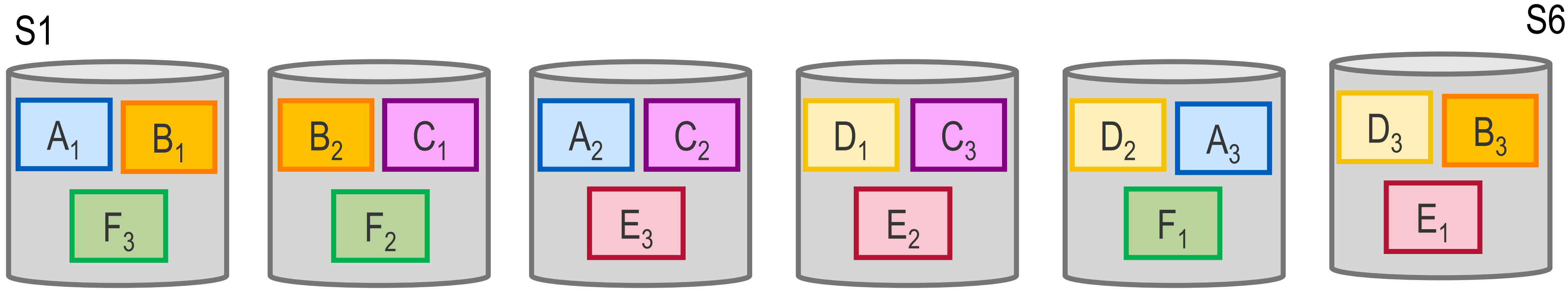
(K=2, R=1)

Initial Data Placement



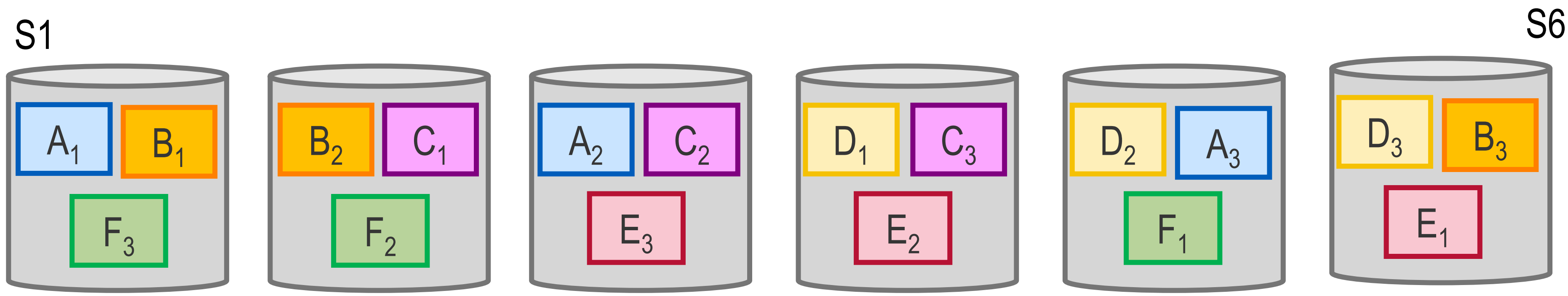
(K=2, R=1)

Multi-Item Data Access Strategy



(K=2, R=1)

Multi-Item Data Access Strategy

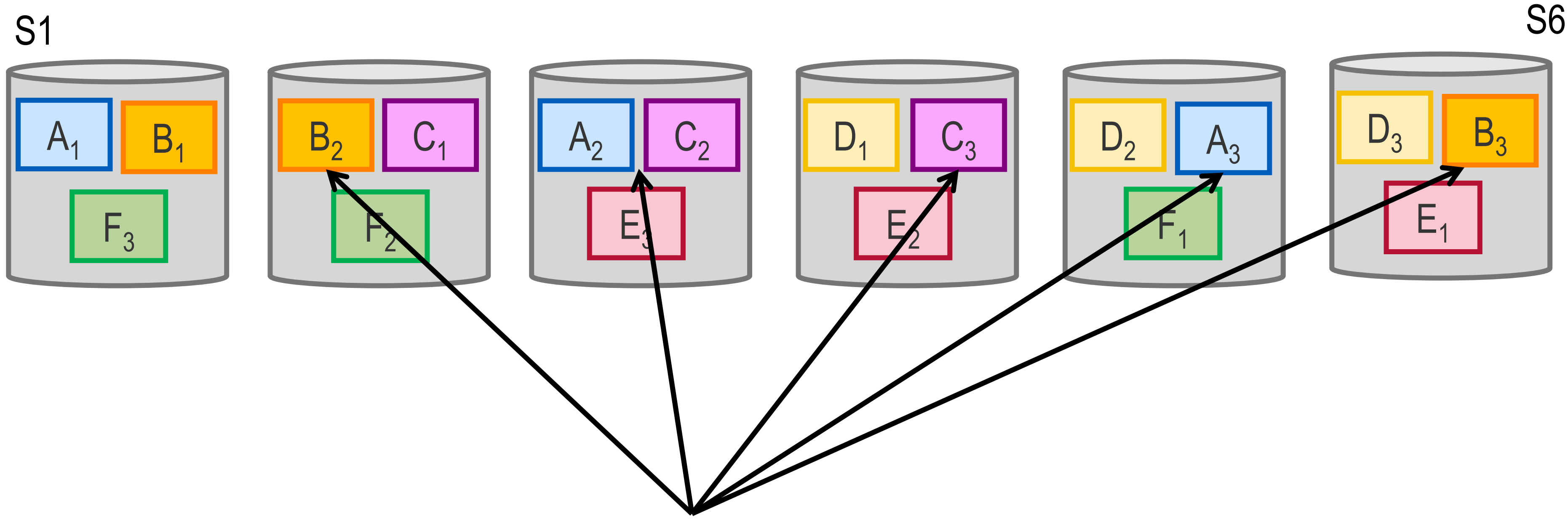


Retrieve: A, B, C

(K=2, R=1)

UNIVERSITY OF WATERLOO

Multi-Item Data Access Strategy

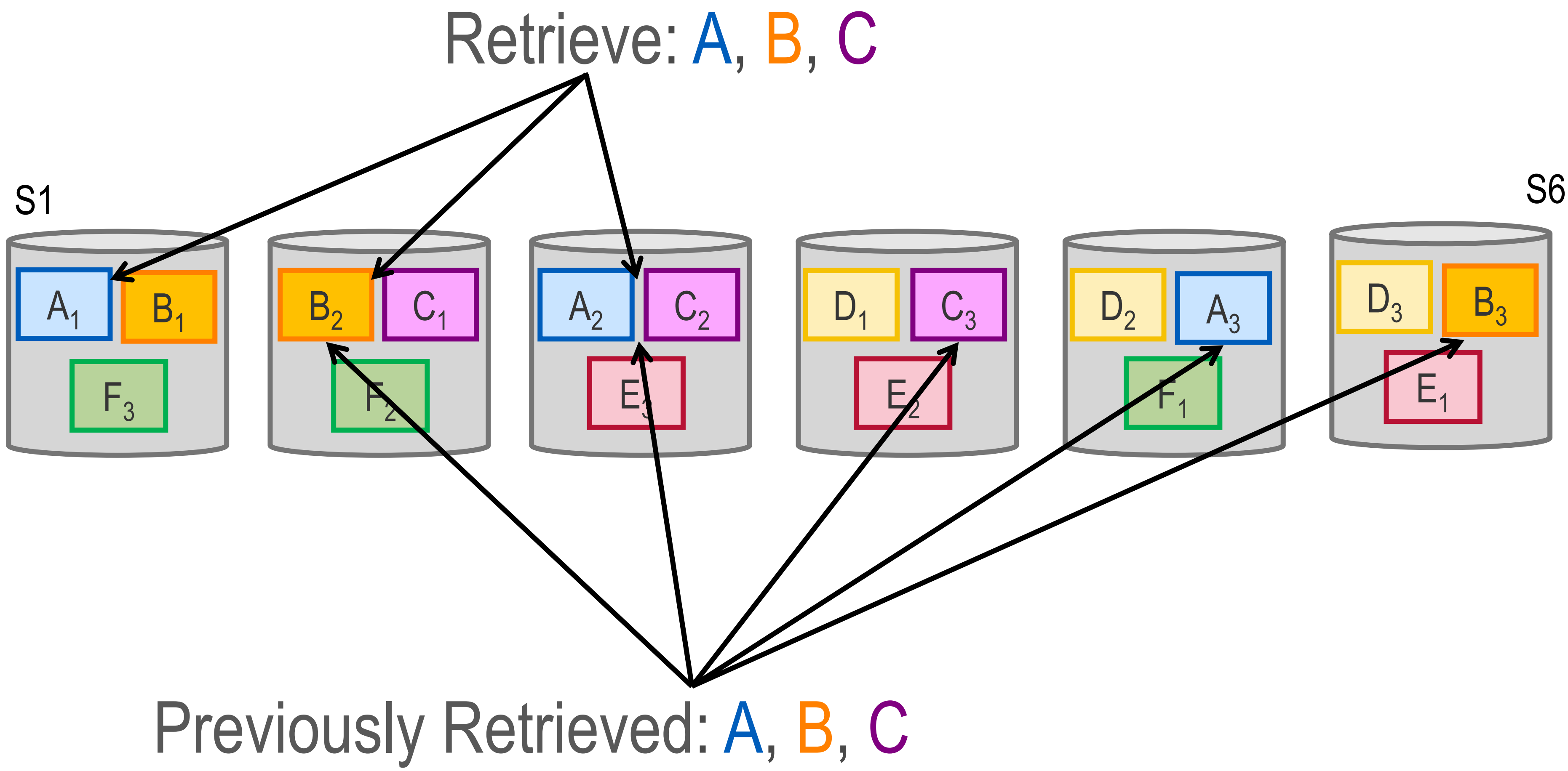


Retrieve: A, B, C

(K=2, R=1)

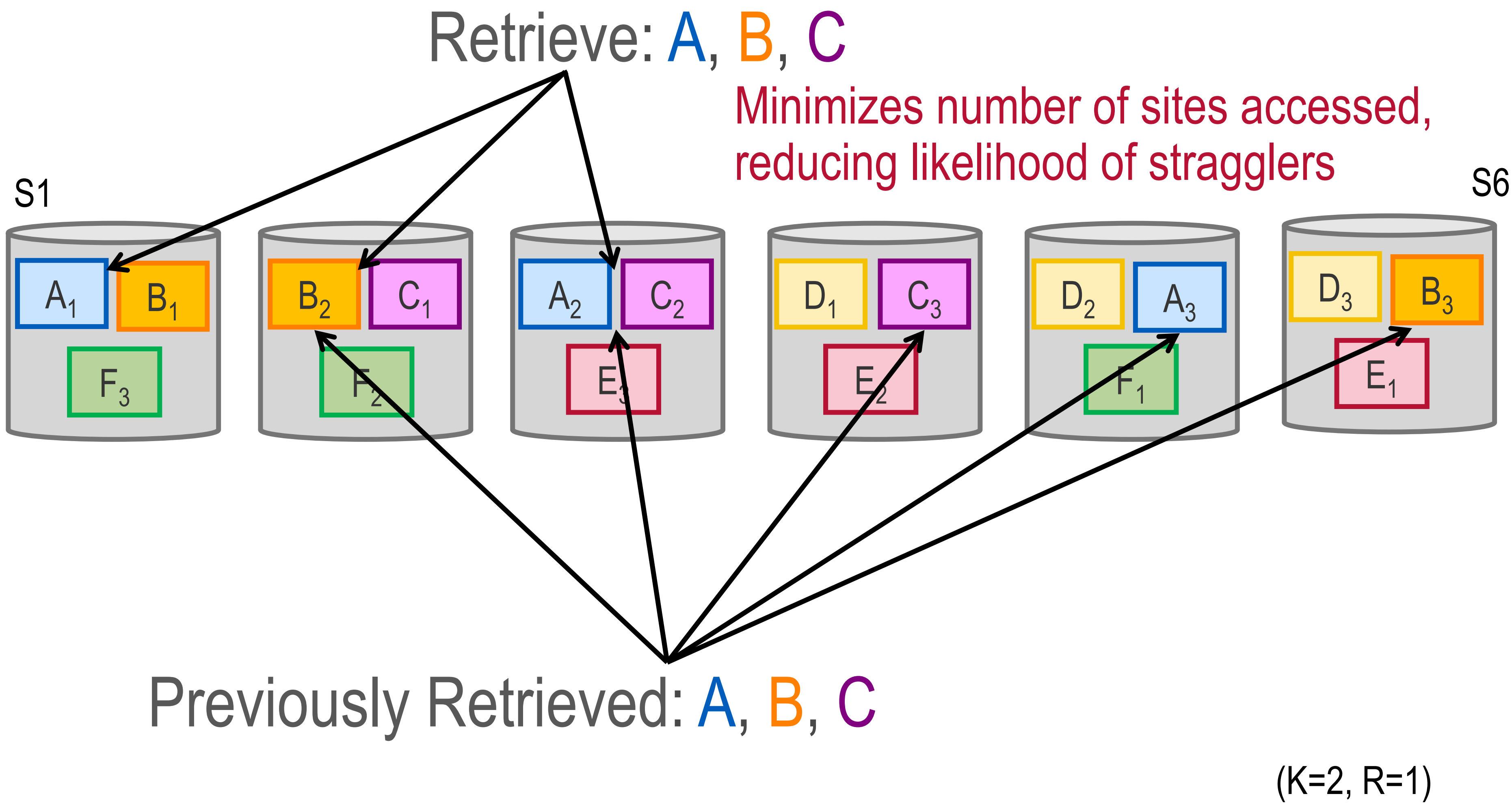
UNIVERSITY OF WATERLOO

Intelligent Multi-Item Data Access Strategy



(K=2, R=1)

Intelligent Multi-Item Data Access Strategy



Intelligent Data Access Strategy

- Retrieve enough (K) chunks to reconstruct block

Intelligent Data Access Strategy

- Retrieve enough (K) chunks to reconstruct block
- Quantify cost of access
  - Cost of accessing site (load)
  - Cost of accessing chunk at site (I/O)

Intelligent Data Access Strategy

- Retrieve enough (K) chunks to reconstruct block
- Quantify cost of access
  - Cost of accessing site (load)
  - Cost of accessing chunk at site (I/O)
- Find minimal cost access strategy with Integer Linear Programming

Details in Paper
tiny.cc/ecstore

UNIVERSITY OF WATERLOO

Data Movement Strategies

- Data Access Strategies
- Data Movement Strategies
- Complementary Strategies
- Experimental Evaluation

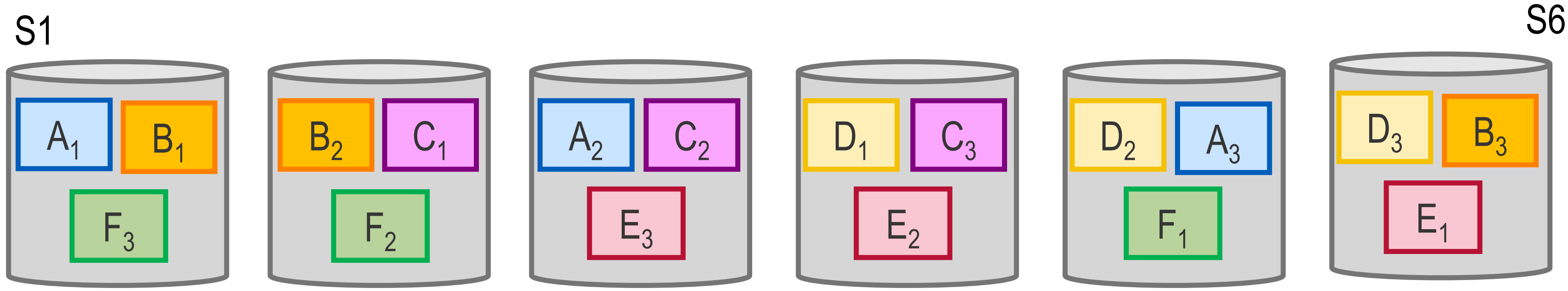
ICDCS

July 3, 2018

UNIVERSITY OF WATERLOO

Workload Change

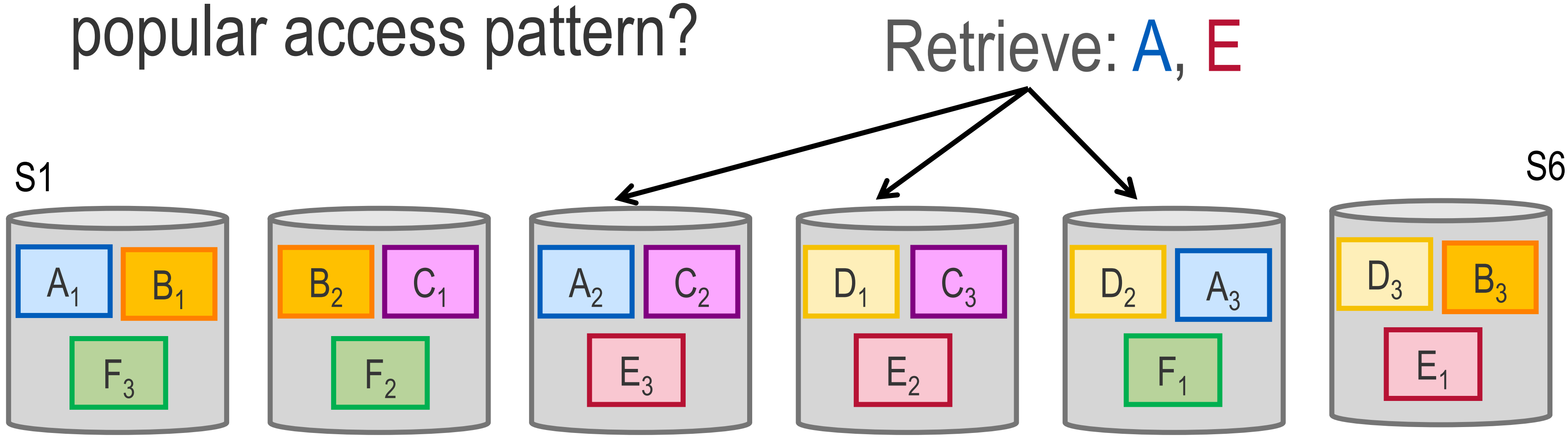
What if: A, E becomes a popular access pattern?



(K=2, R=1)

Workload Change

What if: A, E becomes a popular access pattern?

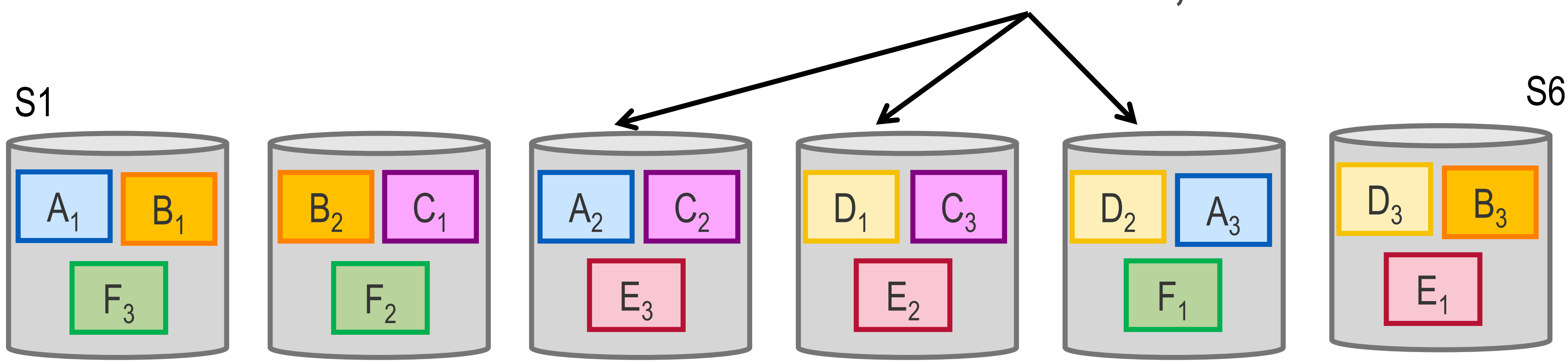


(K=2, R=1)

Can We Improve Access Costs?

What if: A, E becomes a popular access pattern?

Retrieve: A, E



We could move: A_1 E_2 A_3 E_1

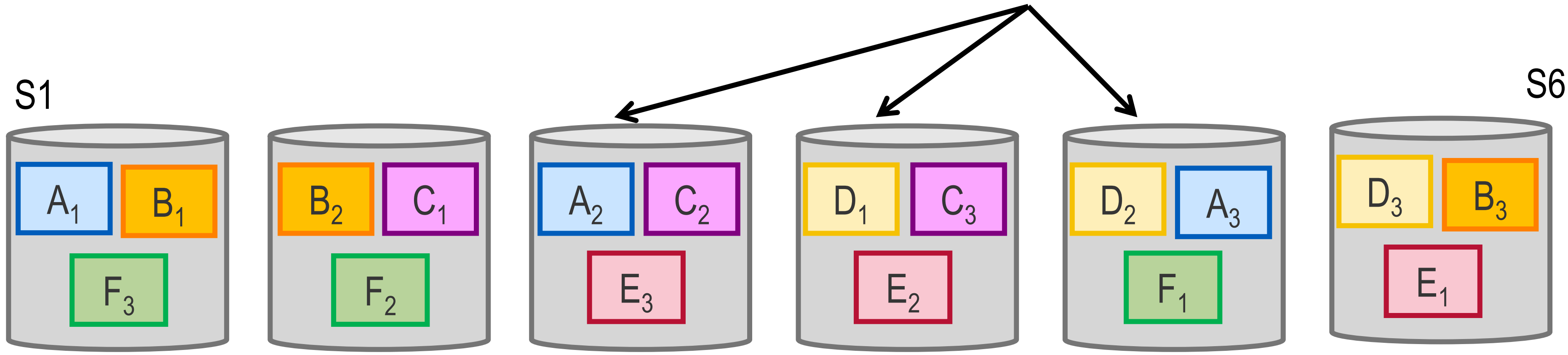
(K=2, R=1)

UNIVERSITY OF WATERLOO

Can We Improve Access Costs?

What if: A, E becomes a popular access pattern?

Retrieve: A, E



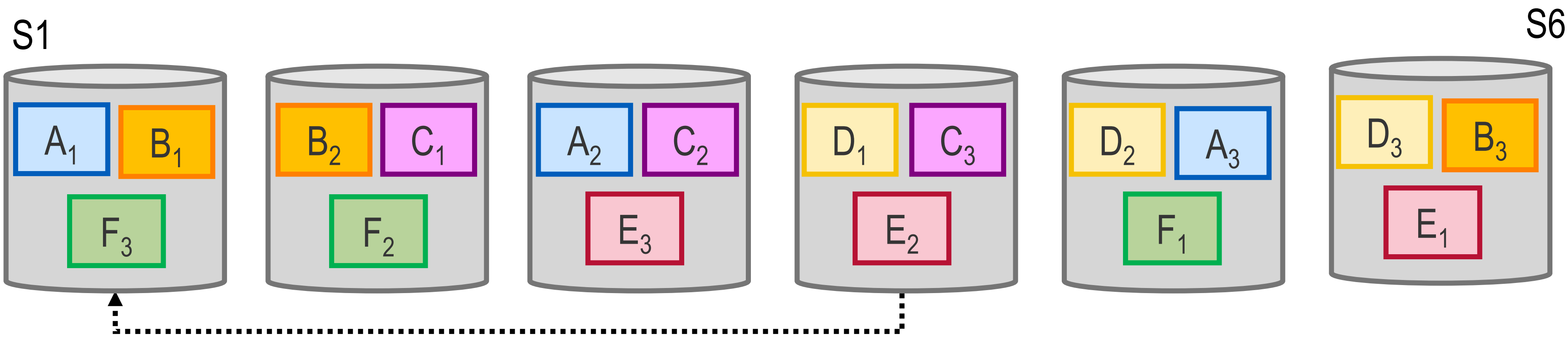
We could move: E_2

Where to move?

(K=2, R=1)

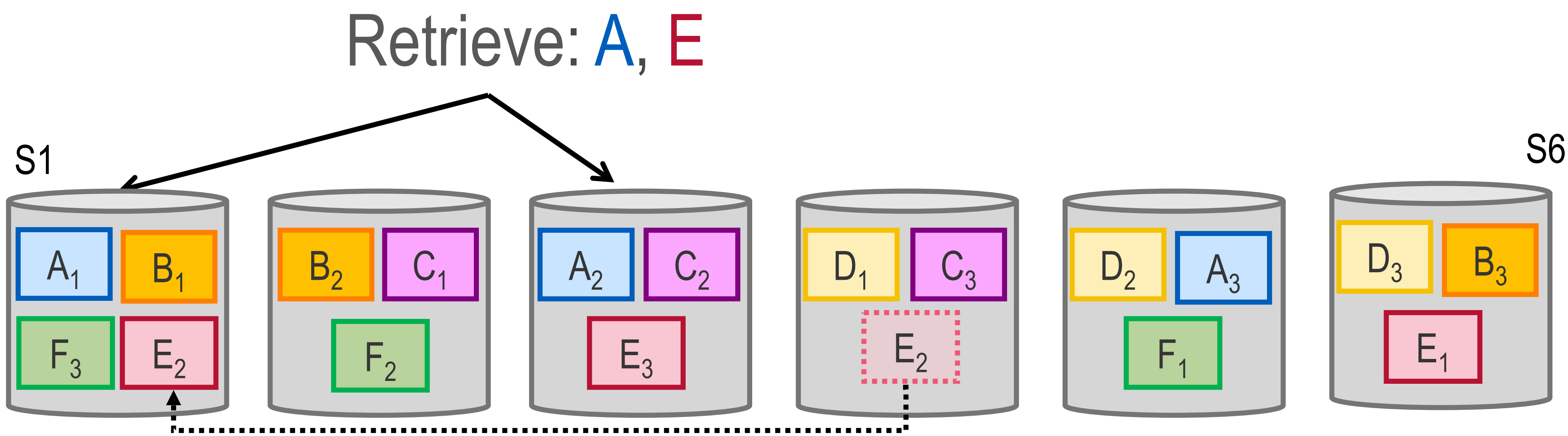
UNIVERSITY OF WATERLOO

Data Movement Decisions



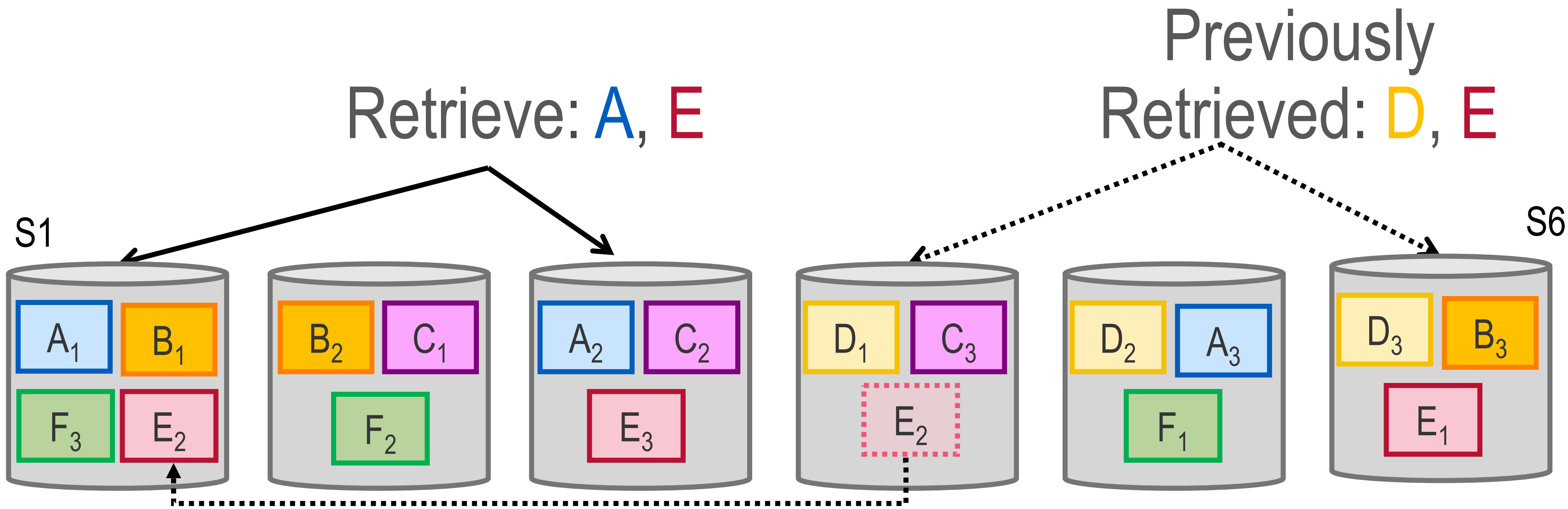
(K=2, R=1)

Data Movement Decisions



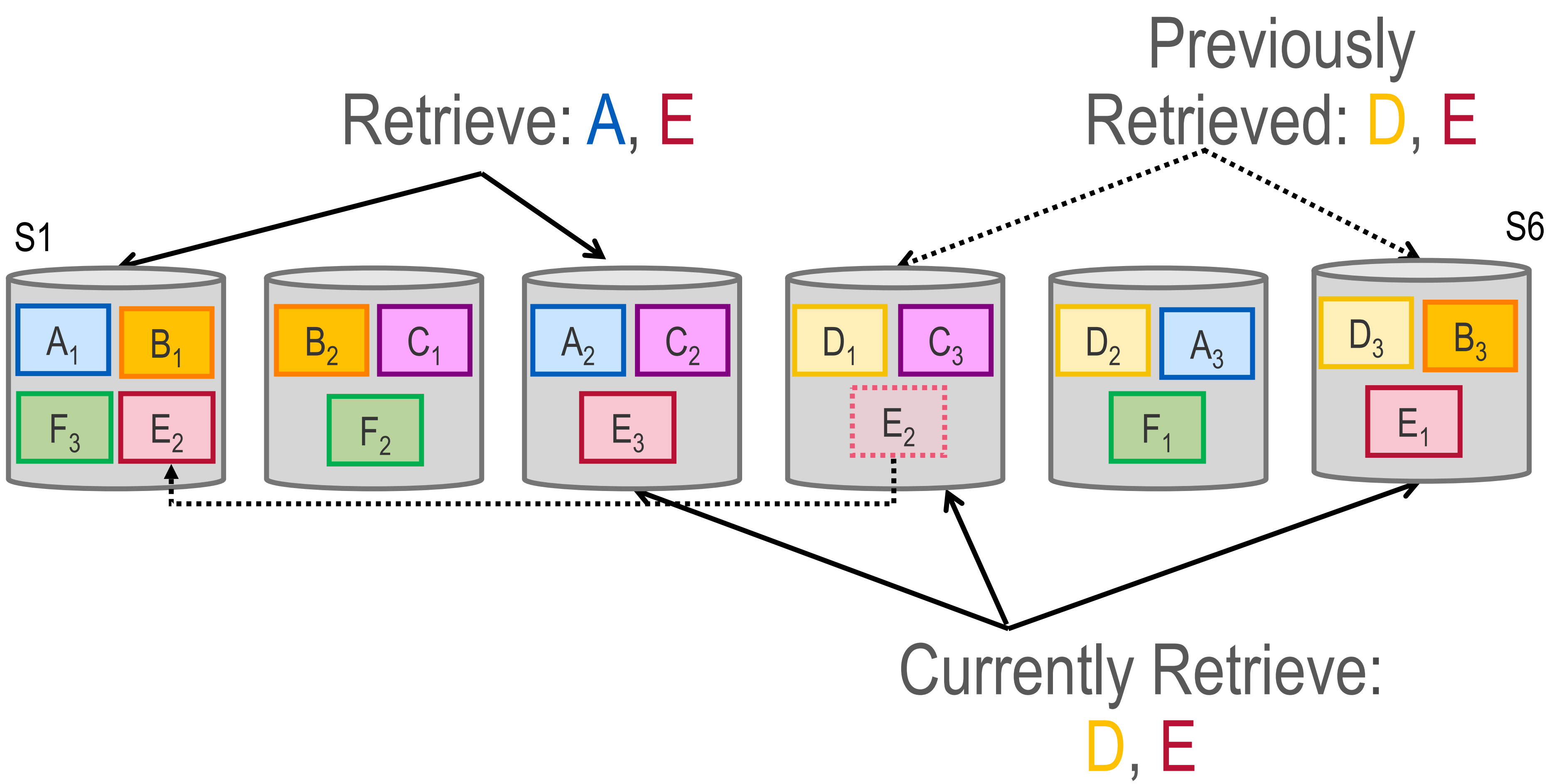
(K=2, R=1)

Data Movement Decisions



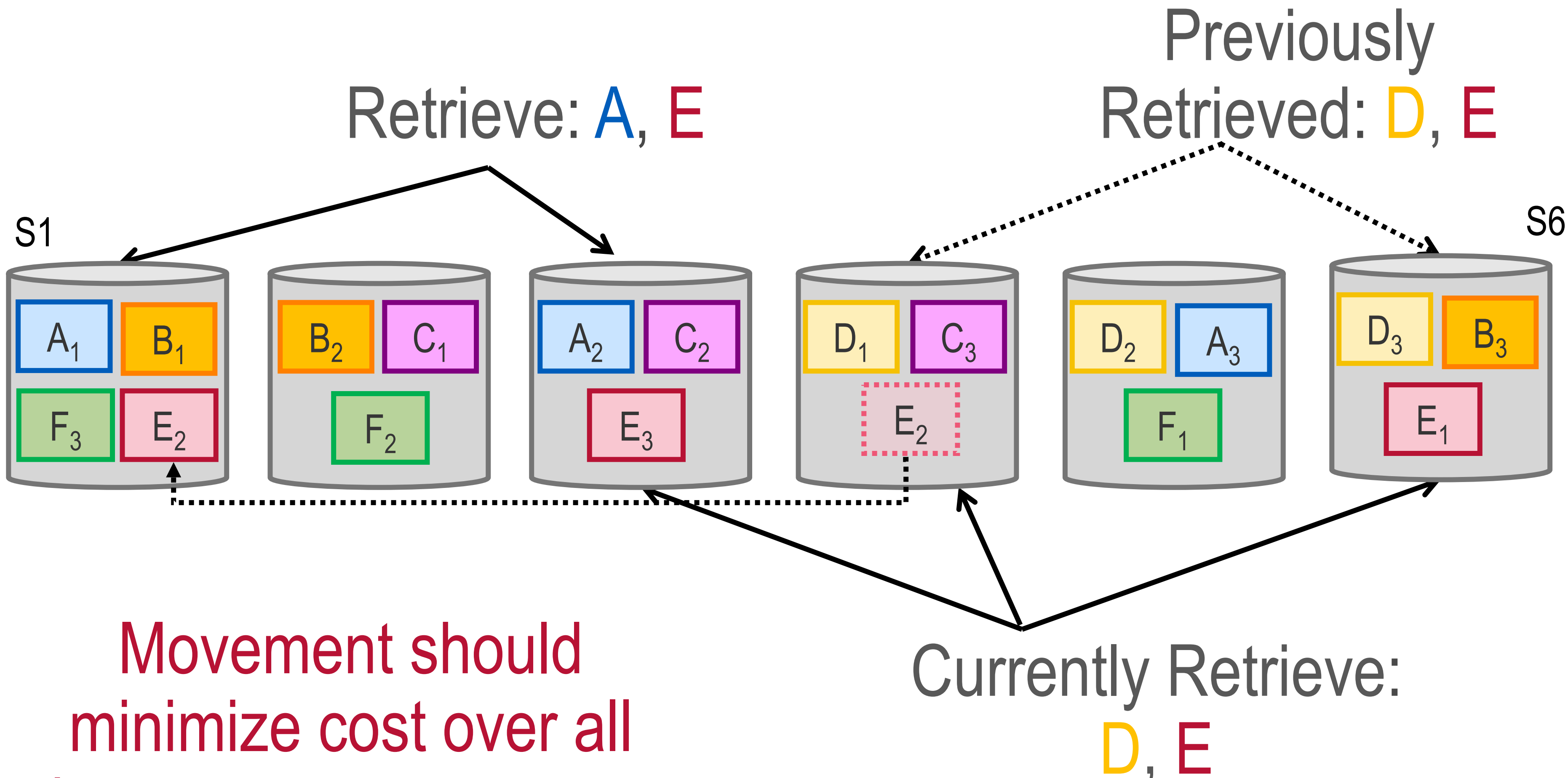
(K=2, R=1)

Data Movement Decisions



(K=2, R=1)

Data Movement Decisions

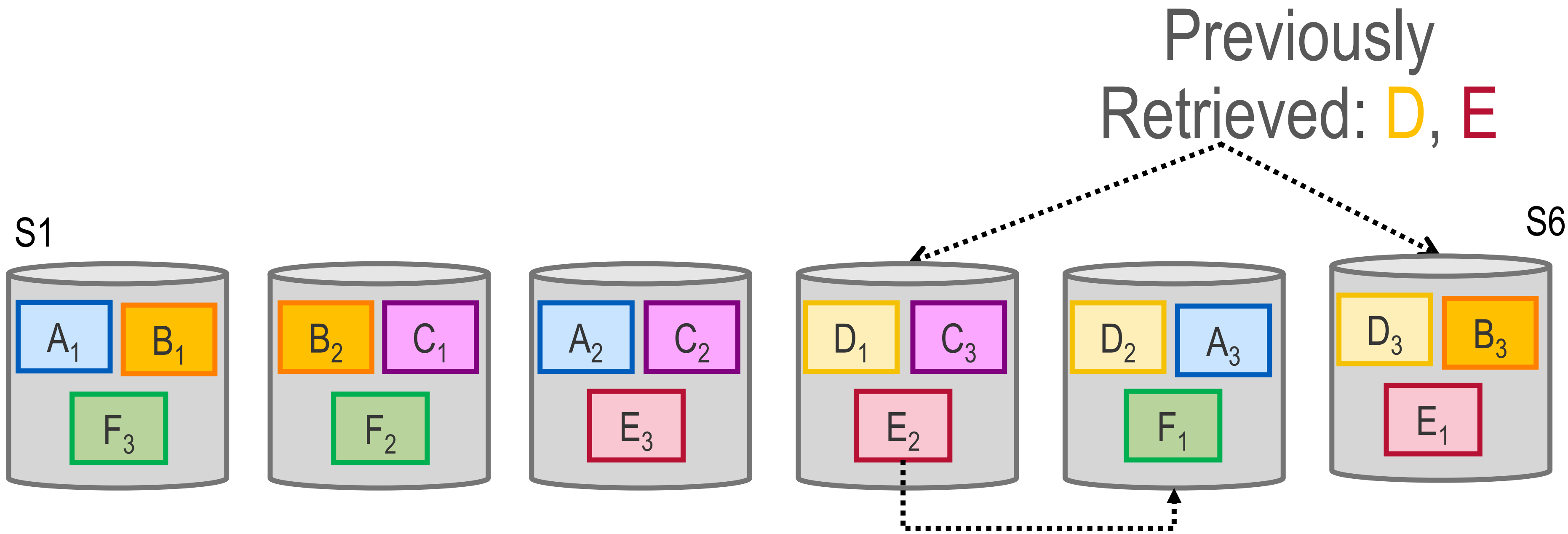


Movement should minimize cost over all known access patterns

(K=2, R=1)

UNIVERSITY OF WATERLOO

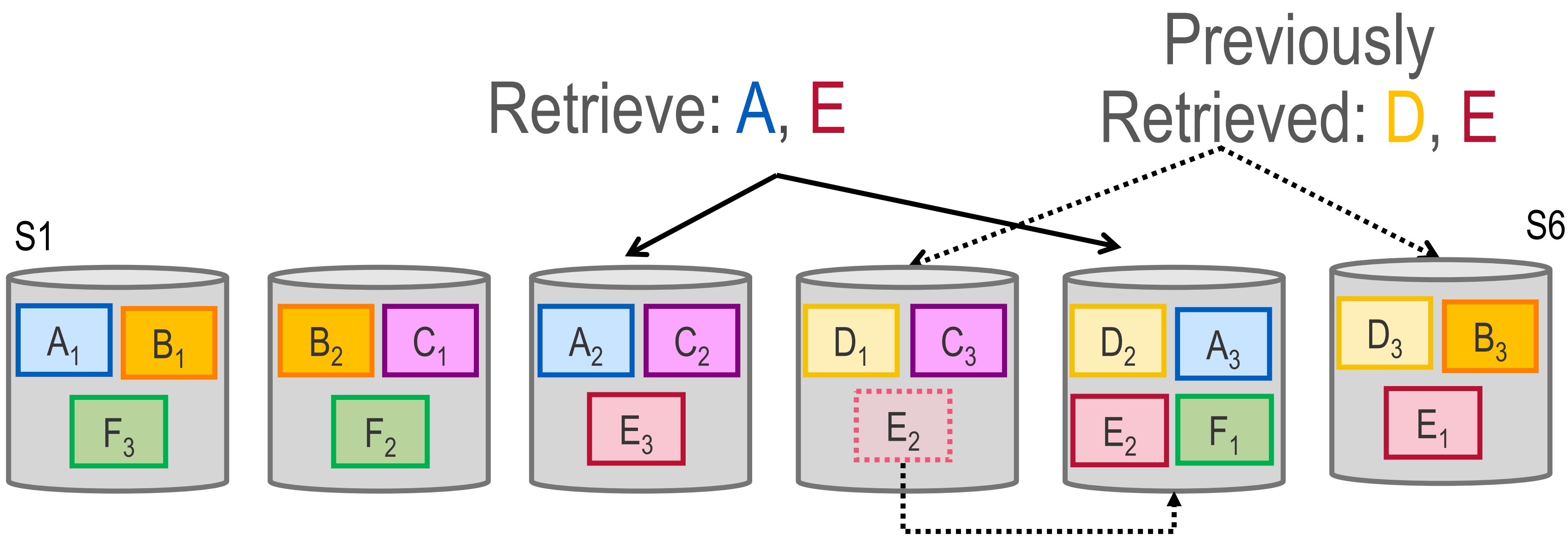
Improved Data Movement Decisions



Movement should minimize cost over all known access patterns

(K=2, R=1)

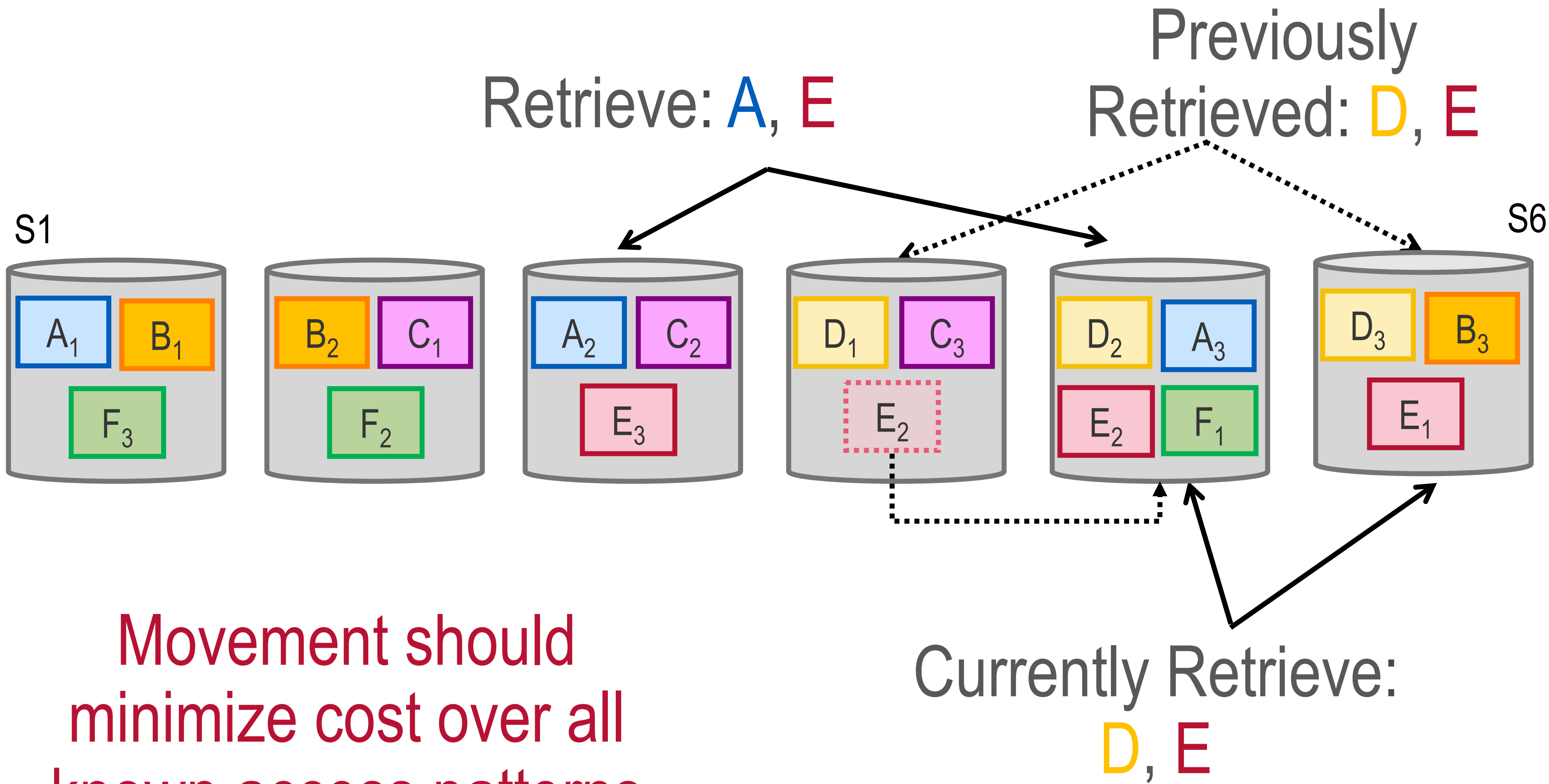
Improved Data Movement Decisions



Movement should minimize cost over all known access patterns

(K=2, R=1)

Improved Data Movement Decisions



Movement should minimize cost over all known access patterns

(K=2, R=1)

UNIVERSITY OF WATERLOO

Dynamic Data Movement

- Move data in response to changes in access patterns and load

Dynamic Data Movement

- Move data in response to changes in access patterns and load
- Consider future access costs and load balance

Details in Paper
tiny.cc/ecstore

Dynamic Data Movement

- Move data in response to changes in access patterns and load
- Consider future access costs and load balance

Details in Paper
tiny.cc/ecstore

- Move recently or frequently accessed chunks to sites with co-accessed chunks or lighter load

Complementary Strategies

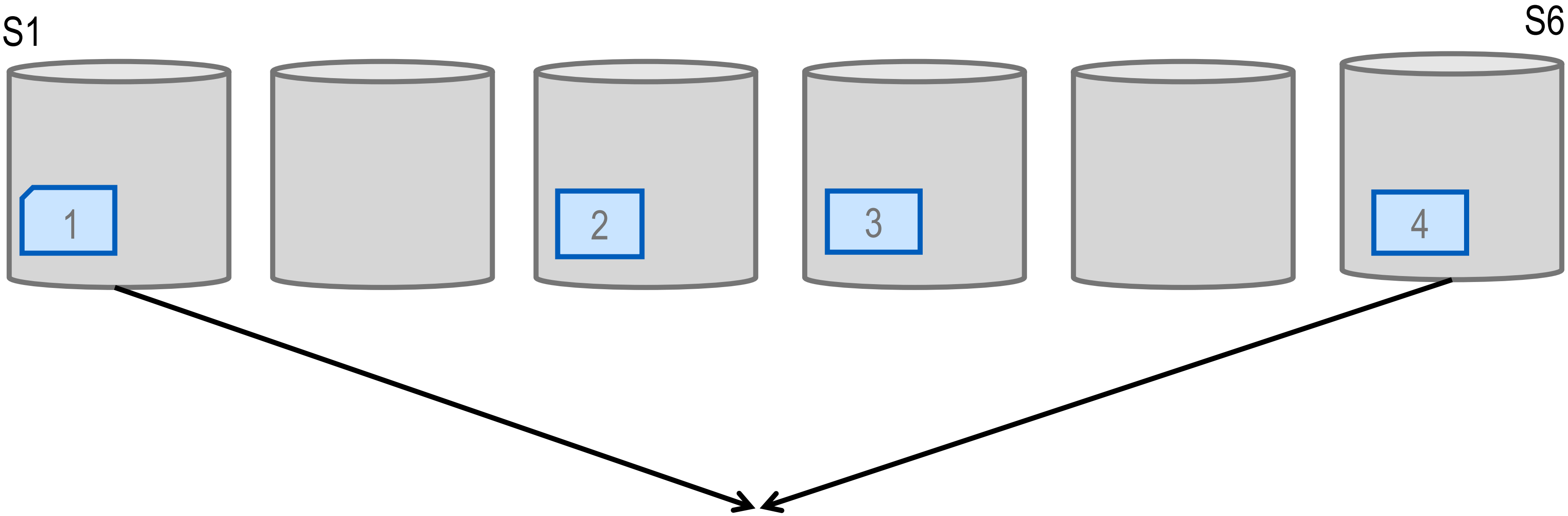
- Data Access Strategies
- Data Movement Strategies
- Complementary Strategies
- Experimental Evaluation

ICDCS

July 3, 2018

UNIVERSITY OF WATERLOO

Standard Request Model

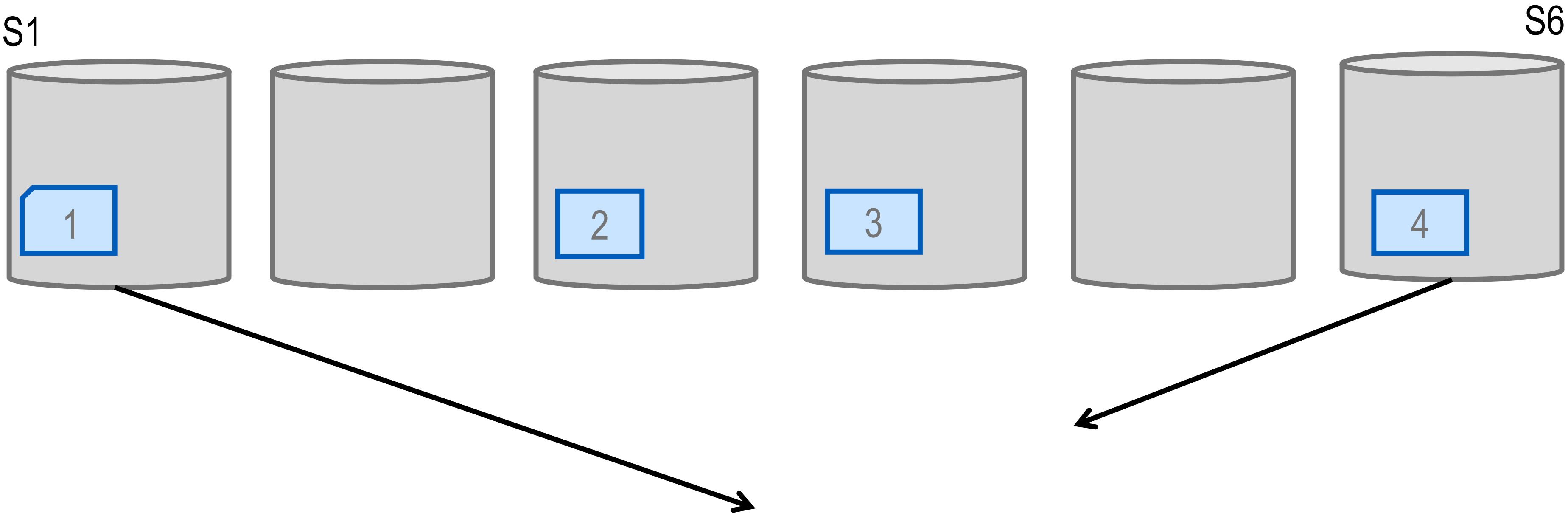


Request chunks from: K sites

Wait for K responses

(K=2, R=2)

Standard Request Model



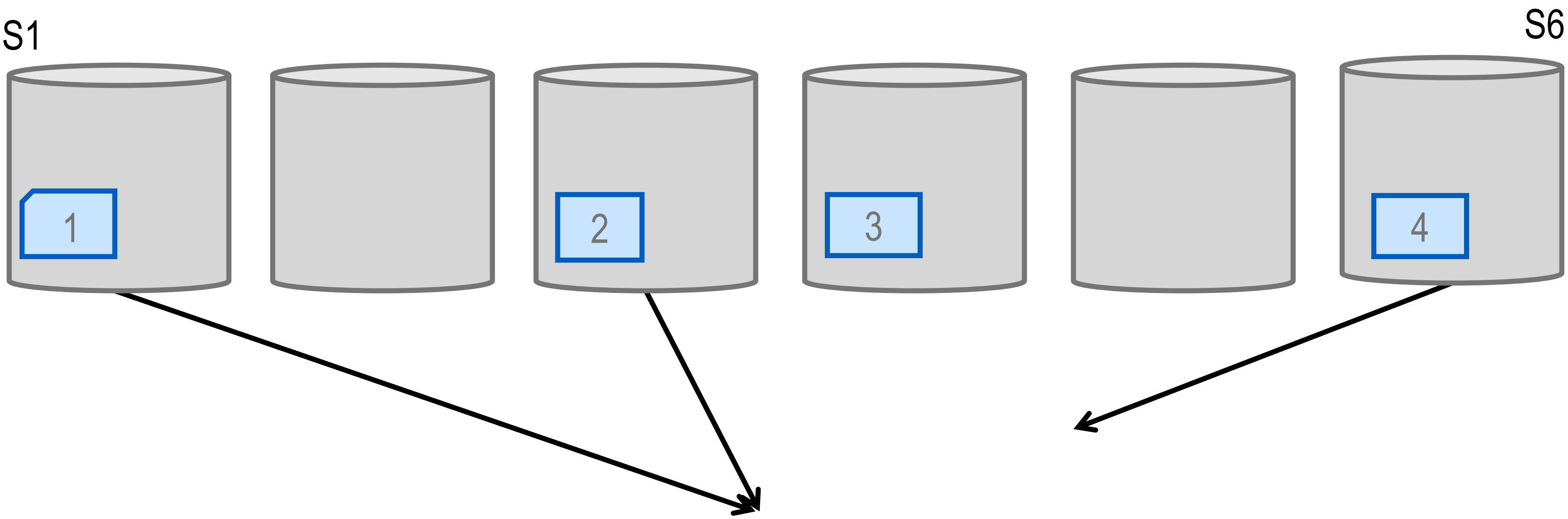
Request chunks from: K sites

Wait for K responses

(K=2, R=2)

UNIVERSITY OF WATERLOO

Late Binding Model



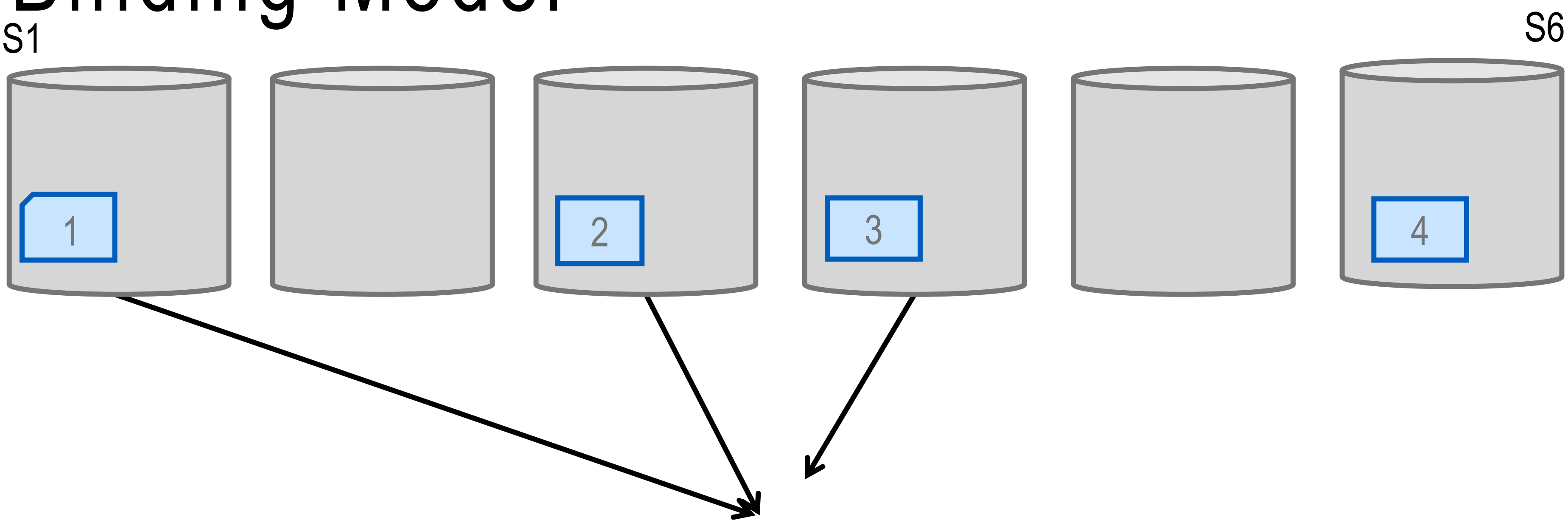
Request chunks from: K + Δ sites

Wait for first K responses

(K=2, R=2)

UNIVERSITY OF WATERLOO

Intelligent Access Strategy + Late Binding Model



Request chunks from: K + Δ sites

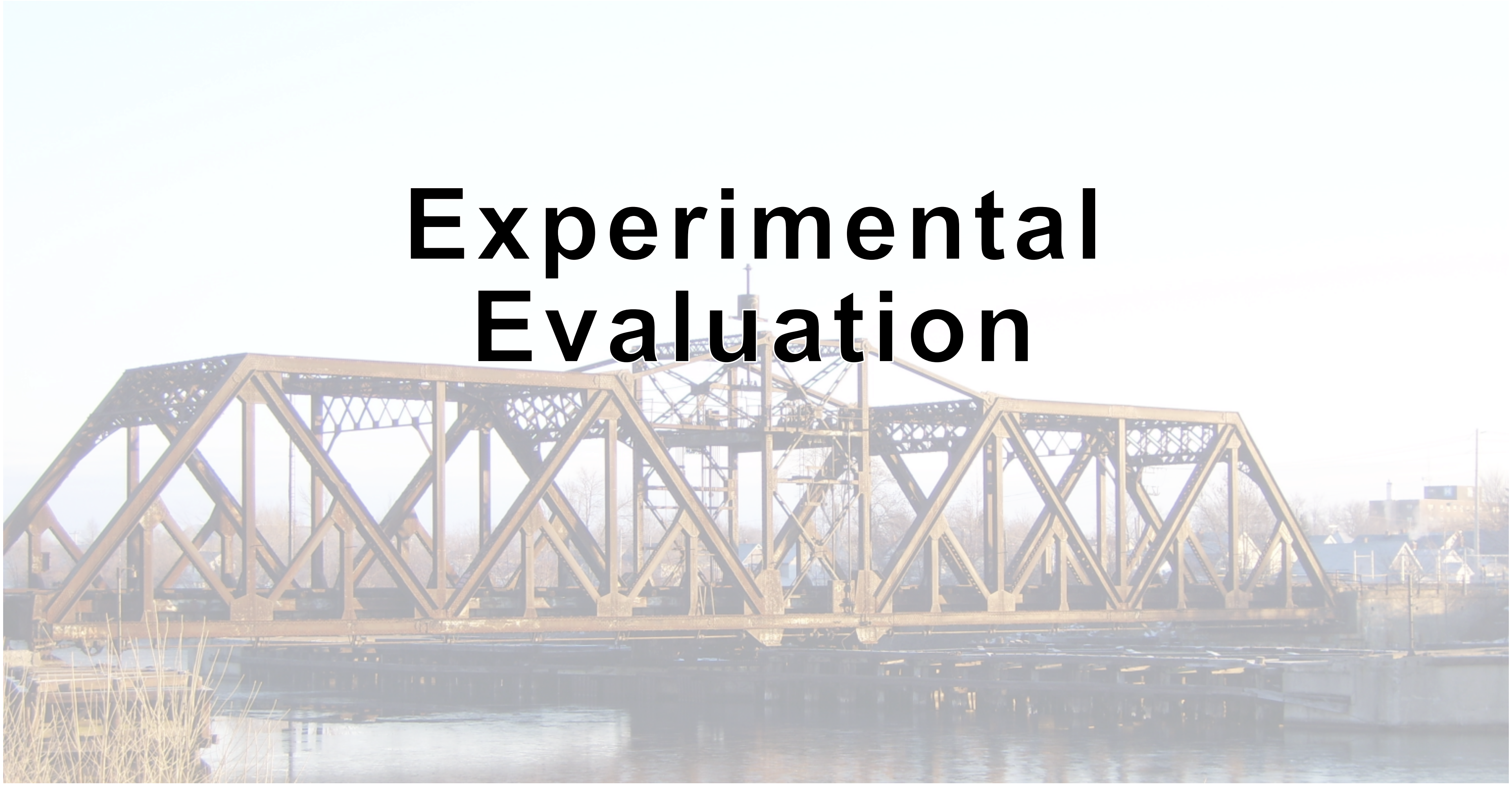
Intelligently select Δ extra requests

Wait for first K responses

Details in Paper
tiny.cc/ecstore

(K=2, R=2)

Experimental Evaluation



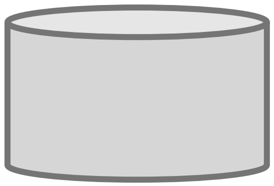
- Data Access Strategies
- Data Movement Strategies
- Complementary Strategies
- Experimental Evaluation

ICDCS

July 3, 2018

UNIVERSITY OF WATERLOO

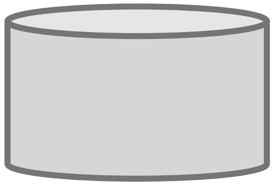
Experimental Setup



x 32 Storage Nodes

System should tolerate 2 faults

Experimental Setup

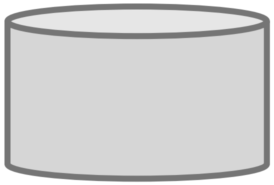


x 32 Storage Nodes

System should tolerate 2 faults

- Replication
- Erasure Coding
  - + Late Binding

Experimental Setup

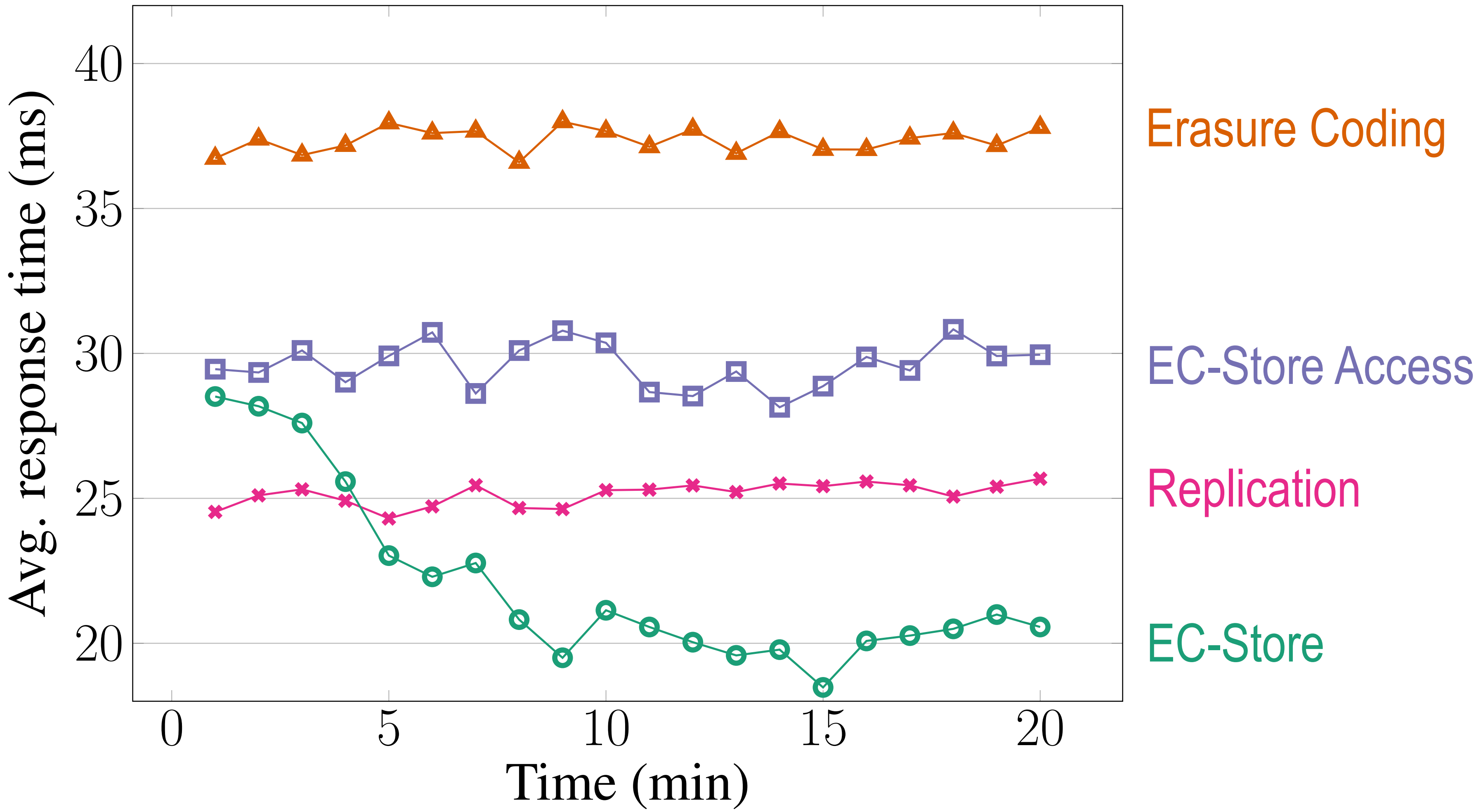


x 32 Storage Nodes

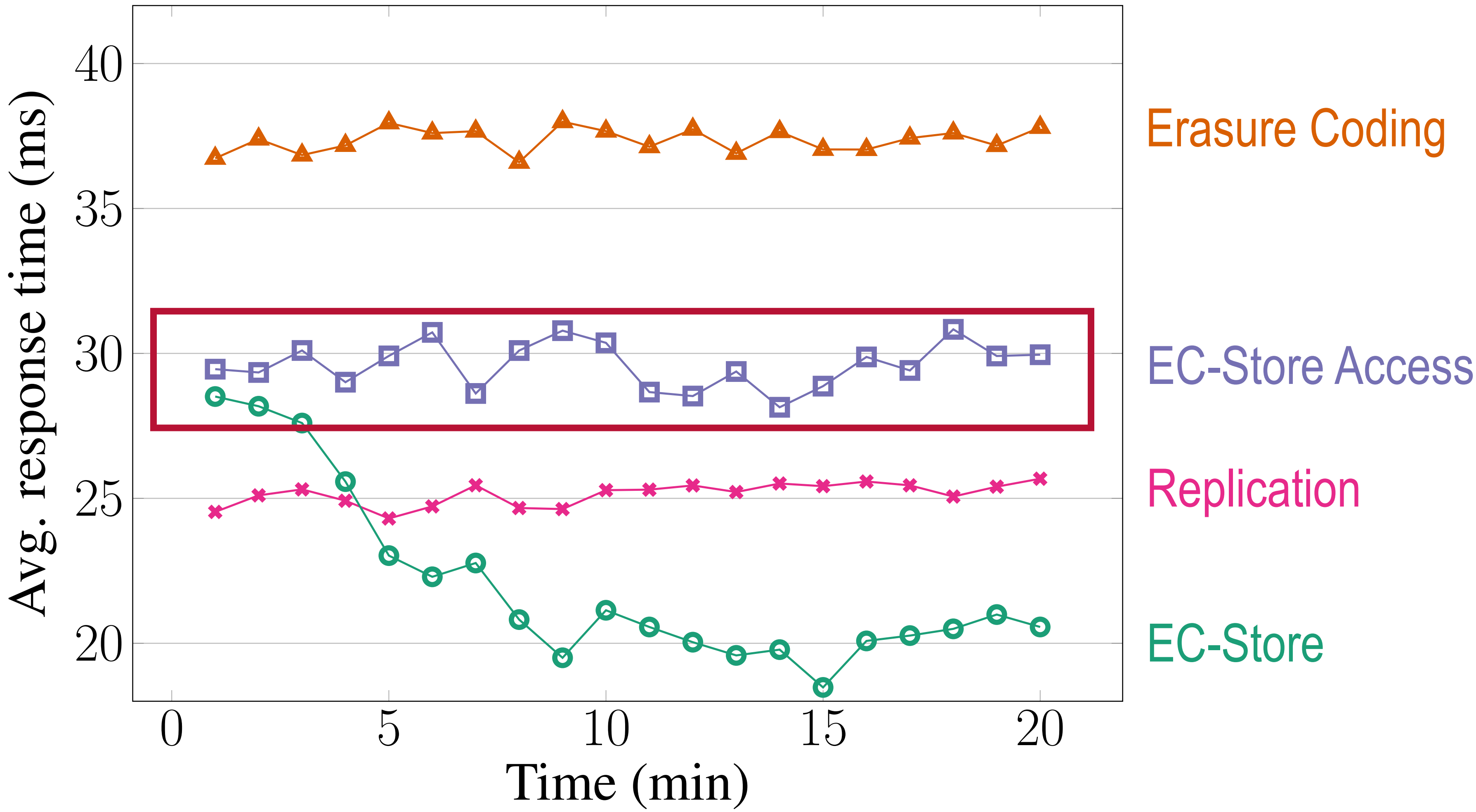
System should tolerate 2 faults

- Replication
- Erasure Coding
  - + Late Binding
- Erasure Coding
  - + Access Strategies (EC-Store Access)
  - + Movement Strategies (EC-Store)
  - + Late Binding (EC-Store + LB)

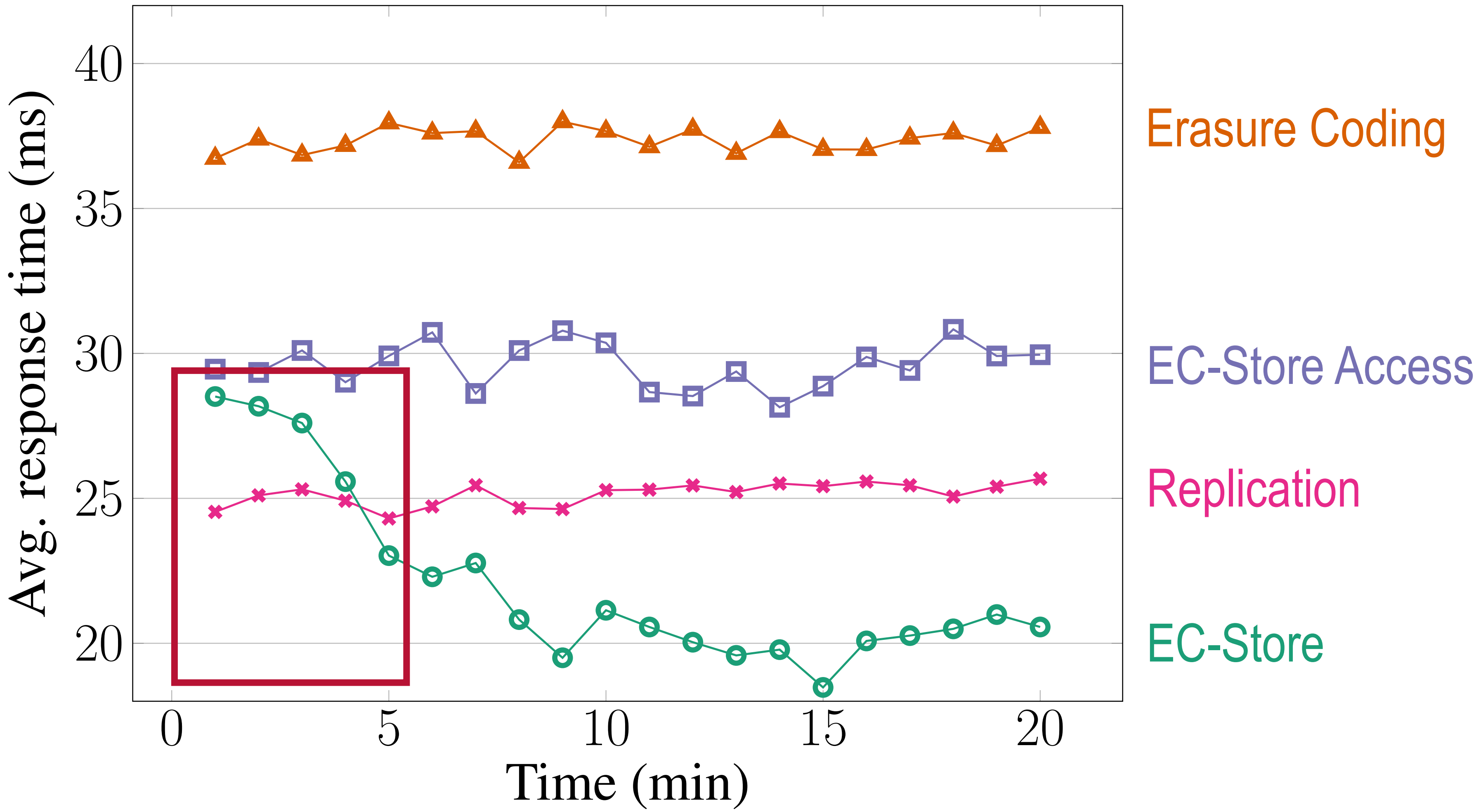
Response time Over time (YCSB)



Response time Over time (YCSB)

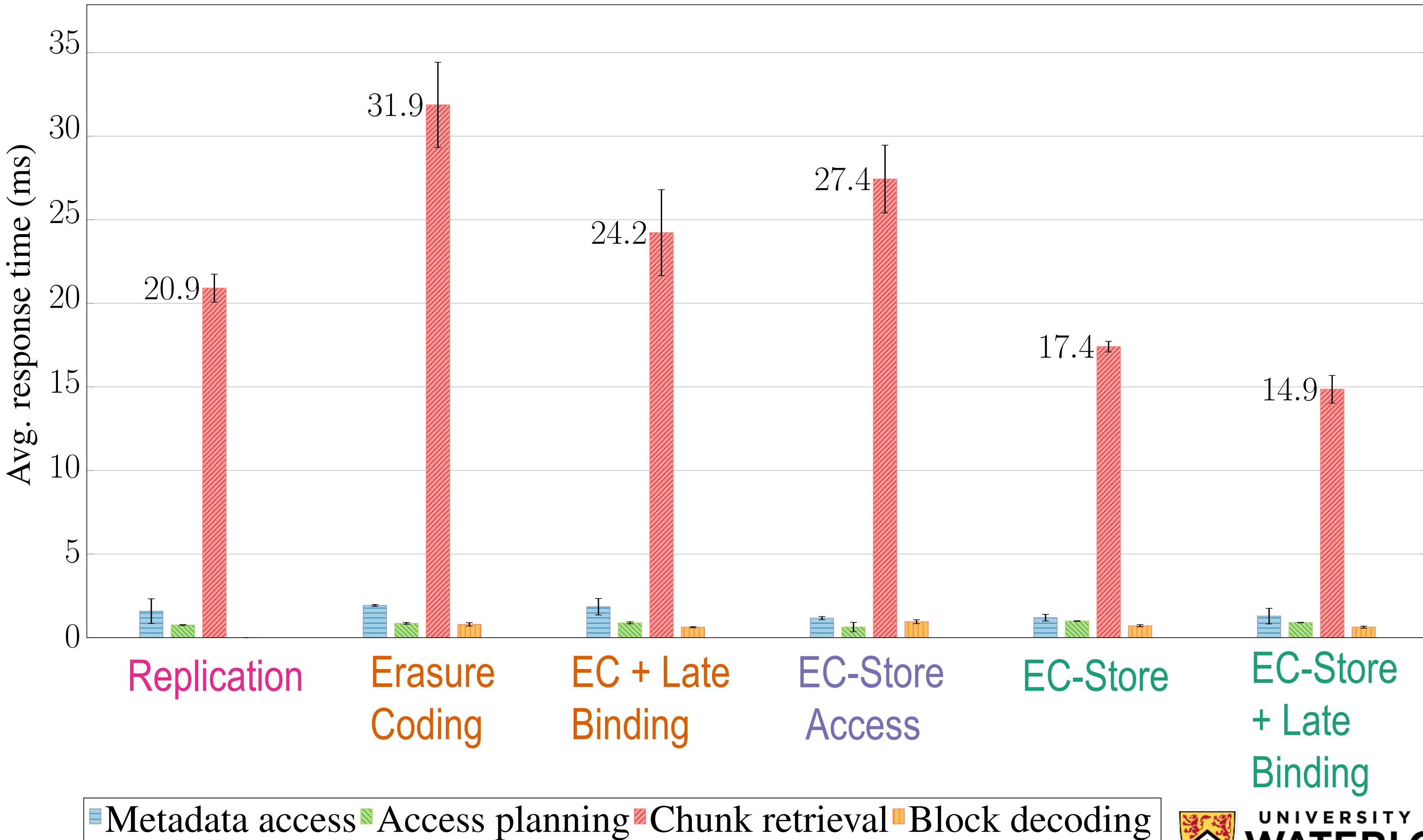


Response time Over time (YCSB)



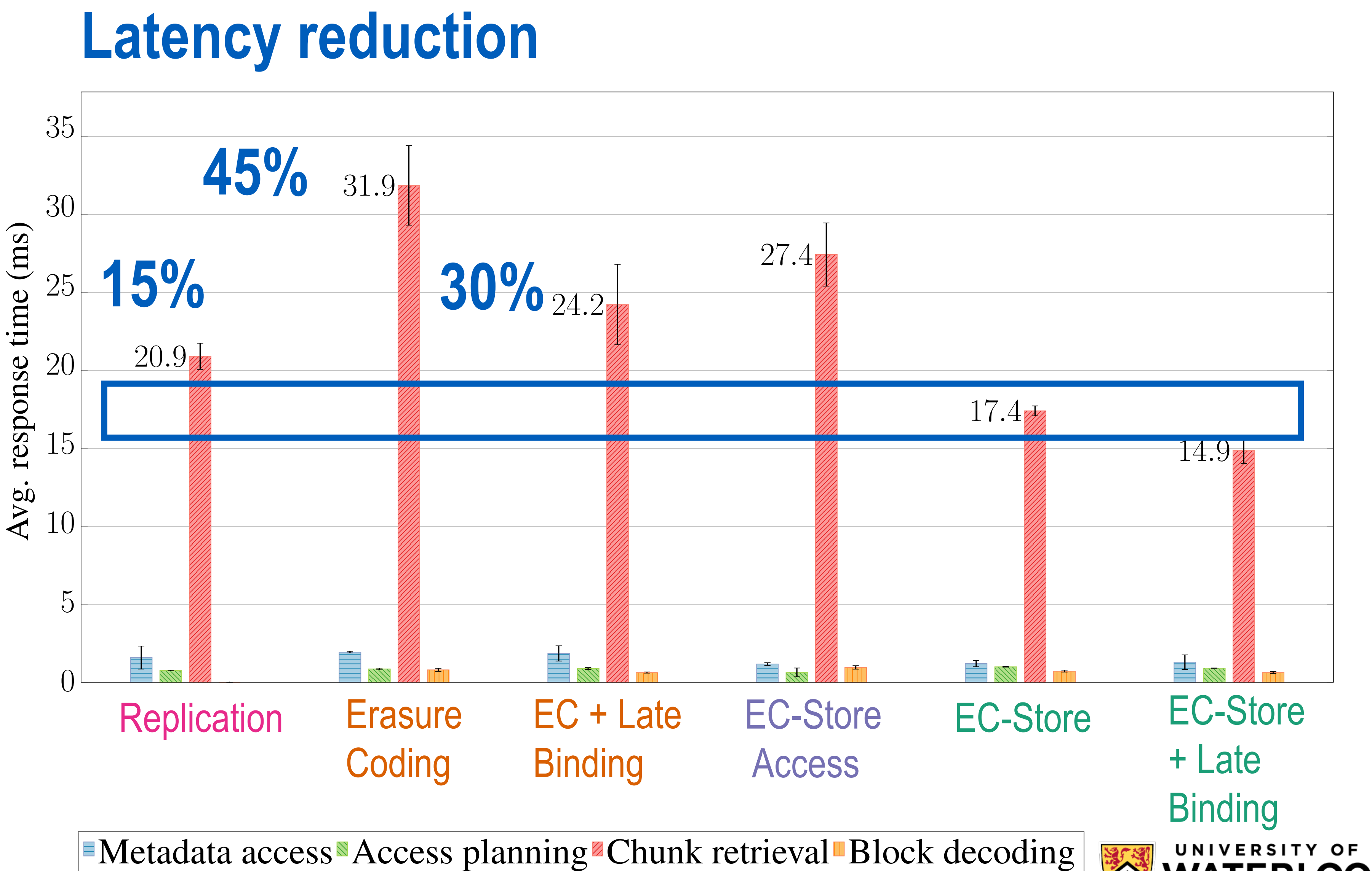
Improves over time!

Breakdown of Response Times (YCSB)

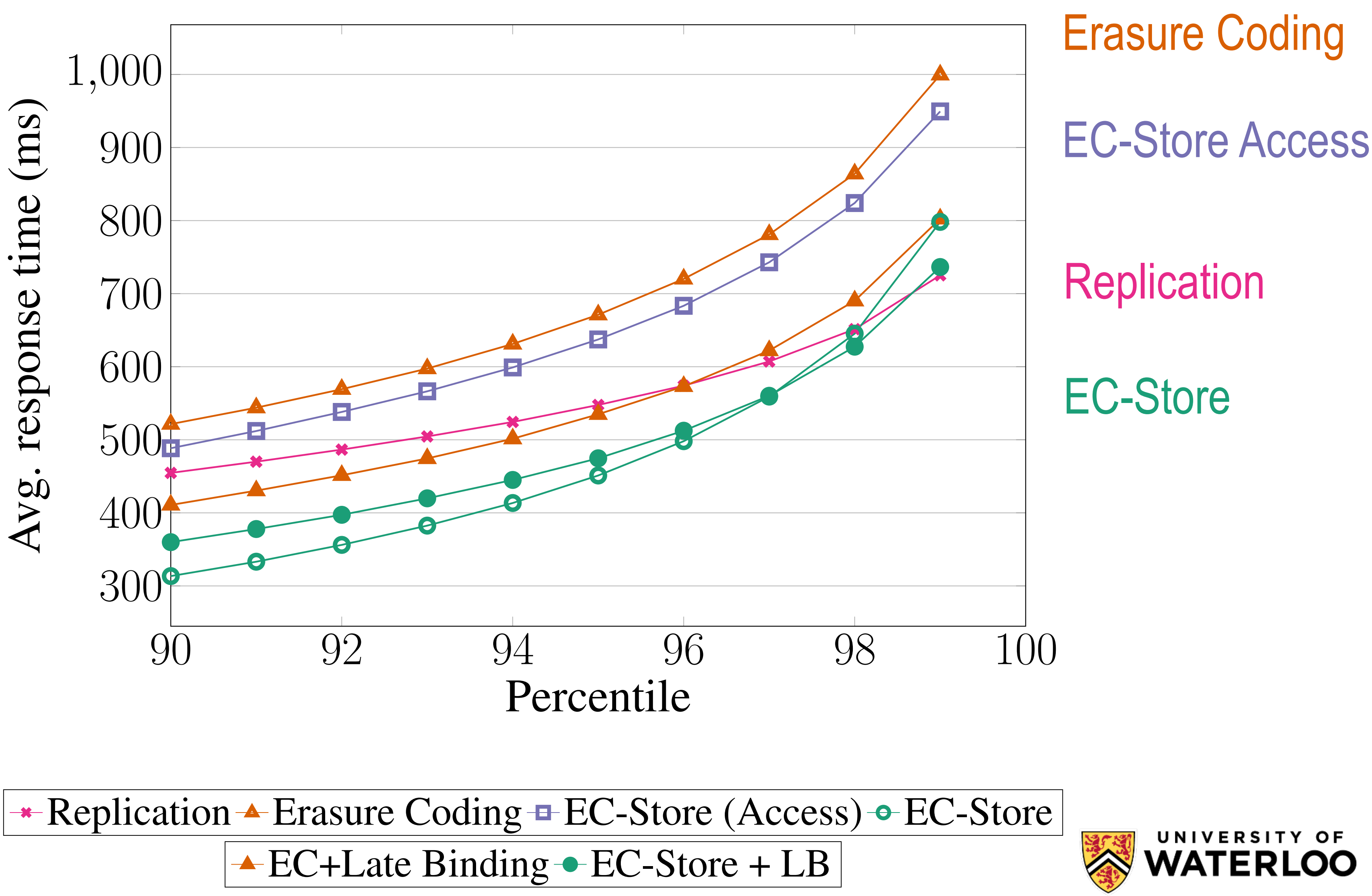


UNIVERSITY OF WATERLOO

Breakdown of Response Times (YCSB)



Tail Latencies (Wikipedia)



UNIVERSITY OF WATERLOO

EC-Store Takeaways

- Achieves low storage overhead and low latency data access
- EC-Store uses dynamic data access and data movement strategies for erasure coded storage systems

EC-Store Takeaways

- Achieves low storage overhead and low latency data access

- EC-Store uses dynamic data access and data movement strategies for erasure coded storage systems

tiny.cc/ecstore

EC-Store: Bridging the Gap Between Storage and Latency in Distributed Erasure Coded Systems

Michael Abebe, Khuzaima Daudjee, Brad Glasbergen, Yuanfeng Tian
Cheriton School of Computer Science, University of Waterloo
{mtabebe, kdaudjee, bjglasbe, y48tian}@uwaterloo.ca

***Abstract*—Cloud storage systems typically choose between replicating or erasure encoding data to provide fault tolerance. Replication ensures that data can be accessed from a single site but incurs a much higher storage overhead, which is a costly downside for large-scale storage systems. Erasure coding has a lower storage requirement but relies on encoding/decoding and distributed data retrieval, which can result in straggling requests that increase response times. We propose strategies for data access and data movement within erasure-coded storage systems that significantly reduce data retrieval times. We present EC-Store, a system that incorporates these dynamic strategies for data access and movement based on workload access patterns. Through detailed evaluation using two benchmark workloads, we show that EC-Store incurs significantly less storage overhead than replication while achieving better performance than both replicated and erasure-coded storage systems.**

***Keywords*-distributed storage, erasure coding, replication, data movement, data placement**

I. Introduction

The need to store and retrieve big data has fueled the development and adoption of cloud storage systems. In cloud deployments, however, machines frequently experience downtime. For example, Google observed that at any point in time, up to 5% of the nodes within their storage system were unavailable [12]. To ensure data remains available in the presence of these failures, systems must be fault tolerant. Large-scale distributed storage systems typically provide fault tolerance either by replicating [4,14] or erasure encoding data [11,15,19,23,30,52]. Replication creates complete copies of data, incurring a significant storage overhead over erasure coding that partitions data and stores the partitions and their parity fragments on multiple nodes to provide the same level of fault tolerance as replication. Consequently, while erasure encoding stores less data, accessing it requires multi-node retrieval resulting in an increase in data access cost compared to replication [51].

To demonstrate that performance in erasure-coded distributed storage systems is largely determined by the cost of data retrieval, we show a breakdown of average response times in Figure 1 for a workload that retrieves multiple 100 KB blocks.[1] The response time is divided into four categories: the cost of locating data (metadata access), determining which data chunks to retrieve (access planning), retrieving data, and decoding data. As Figure 1 shows, the performance difference between replication and erasure coding is primarily due to

[1]Details are in Section VI.

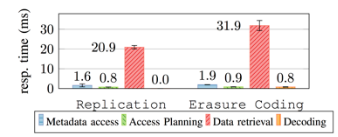


Fig. 1: Response time breakdown for replication and erasure coding under skewed access. Data retrieval times dominate the overall response time.

the time it takes to retrieve data, which dominates overall response times. However, while both systems can tolerate the same number of faults (two, in the example of Figure 1), the replicated system stores 50% more data than the erasure-coded system. These differences motivate a fault tolerant storage system that can achieve the best of both worlds: low storage overhead *and* low latency data retrieval.

When compared to replicated data stores, retrieval costs are higher for erasure-coded storage systems because of the effects of stragglers: the time taken to retrieve the slowest, or *straggling*, data chunk dominates retrieval time [19,29]. Even when parallelism is leveraged, straggler effects are more pronounced in systems that must wait for multiple requests to complete (e.g. in erasure-coded storage) than in systems that wait for only a single request to complete (e.g. in replicated storage) [9,26,46,49,53]. Given that large-scale storage systems are typically deployed in distributed environments, concurrent clients issuing requests in parallel over the distributed storage system inevitably result in the occurrence of stragglers [9].

In our erasure-coded storage system, **EC-Store**, we propose a novel approach to the stragglers problem: intelligently select chunks to retrieve so as to avoid stragglers. This *dynamic data access strategy* uses chunk location information to generate a cost-effective strategy on-the-fly for data retrieval. By incorporating this strategy in EC-Store, we reduce data access latencies and satisfy our best of both worlds goal.

To mitigate the effects of stragglers, some systems use a late binding strategy [19,38,49] in which additional requests are made and the slowest responses are ignored. Late binding can reduce response time but places additional load on the system: responses that will be ignored must still be generated. In contrast, EC-Store's dynamic data access strategy offers excellent performance and places little additional load on the