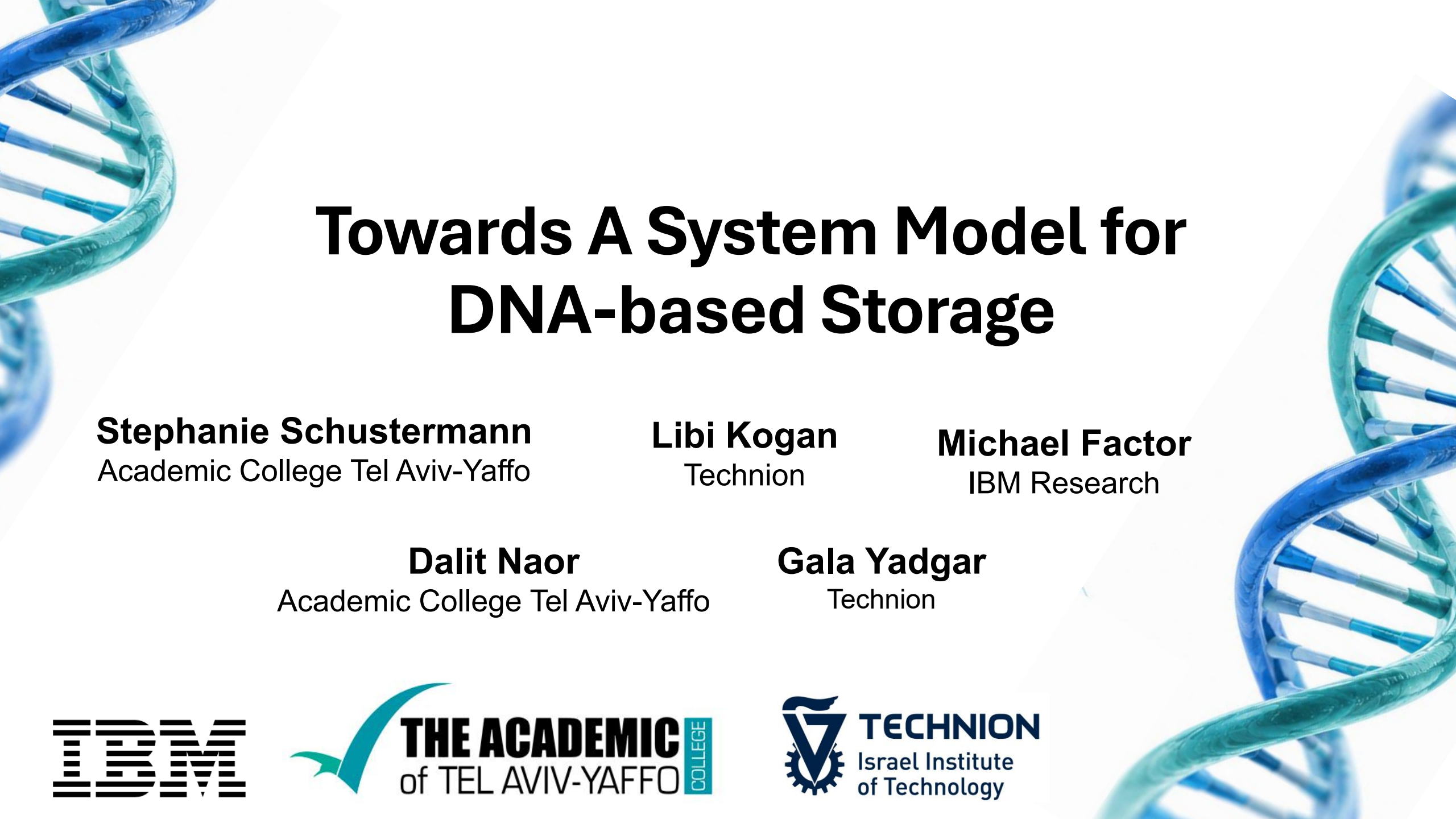


A blue and green DNA double helix structure is visible on the left and right sides of the slide, framing the central text.

Towards A System Model for DNA-based Storage

Stephanie Schustermann
Academic College Tel Aviv-Yaffo

Libi Kogan
Technion

Michael Factor
IBM Research

Dalit Naor
Academic College Tel Aviv-Yaffo

Gala Yadgar
Technion



DNA STORAGE



Preserved for
thousands of years

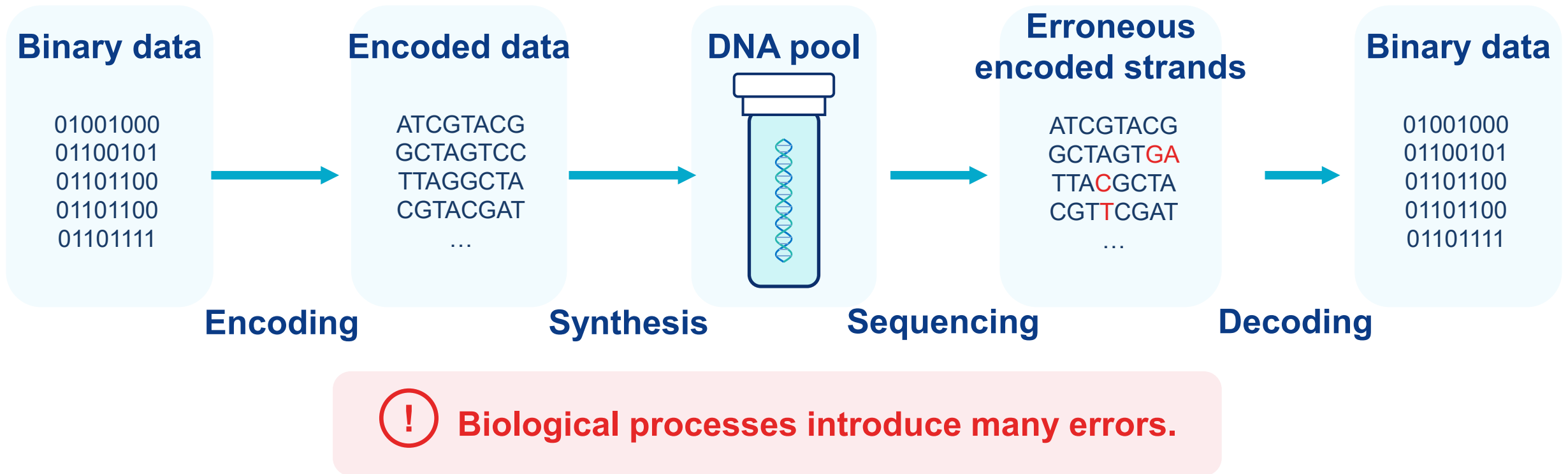


Theoretical storage density:

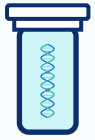
455 EB/g

What is DNA storage?

Digital information is encoded into DNA, stored, and later read back.



Current research efforts



Biotechnology:

sequencing, synthesis,
preservation, random access



Information theory:

data density, error-tolerance

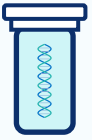


Systems:

updates, primer libraries,
encryption, lookup, prototypes

Single-tube context

Current research efforts



Biotechnology:

sequencing, synthesis,
preservation, random access



Information theory:

data density, error-tolerance



Systems:

updates, primer libraries,
encryption, lookup, prototypes

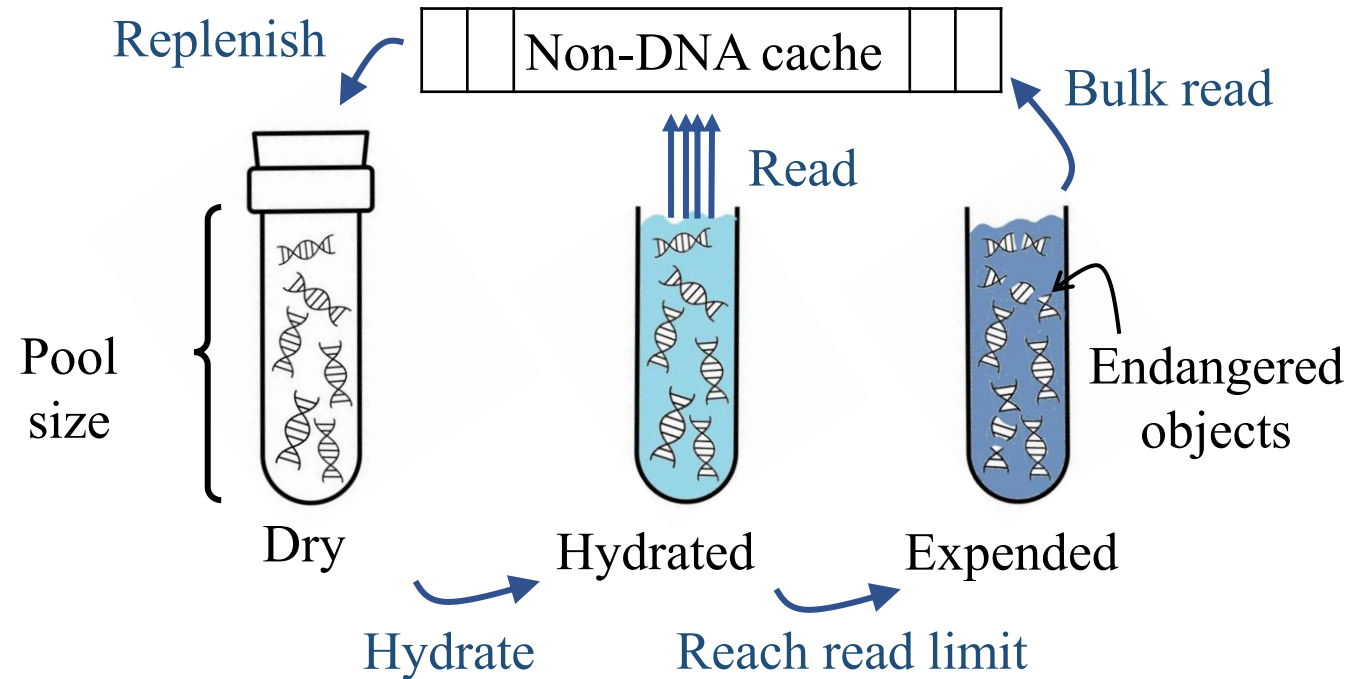


?

**What about a
collection of pools?**

Our system model

A simple abstraction of a tube's states and operations

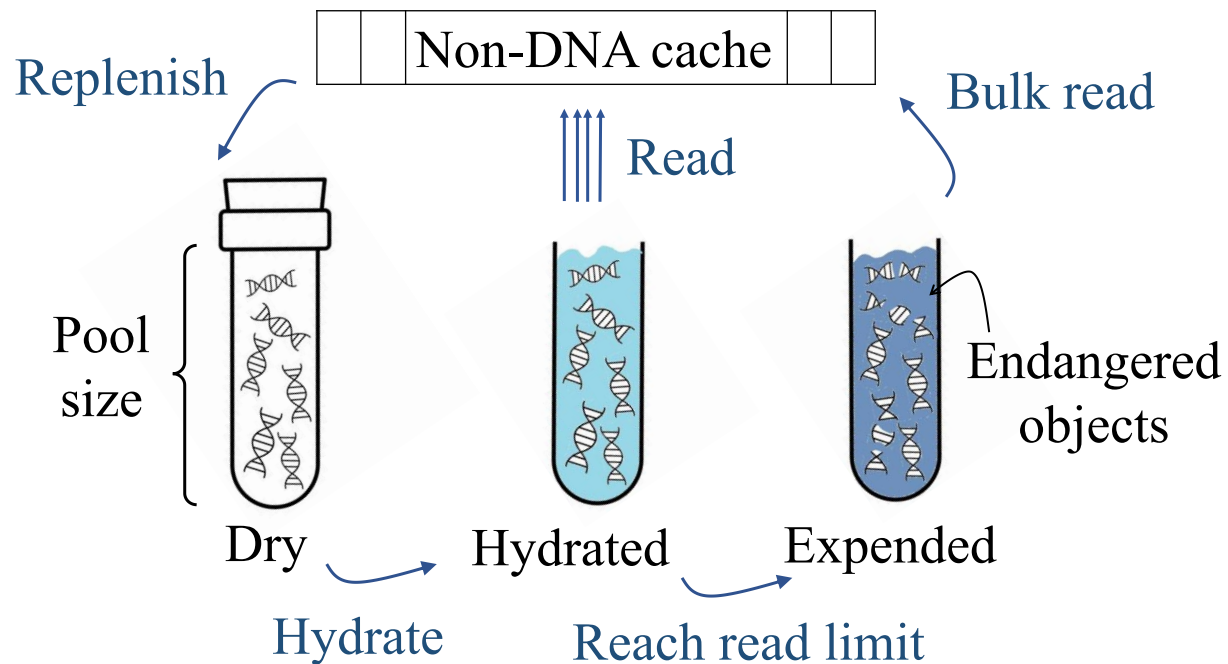


key system-level effects



lower-level biological details

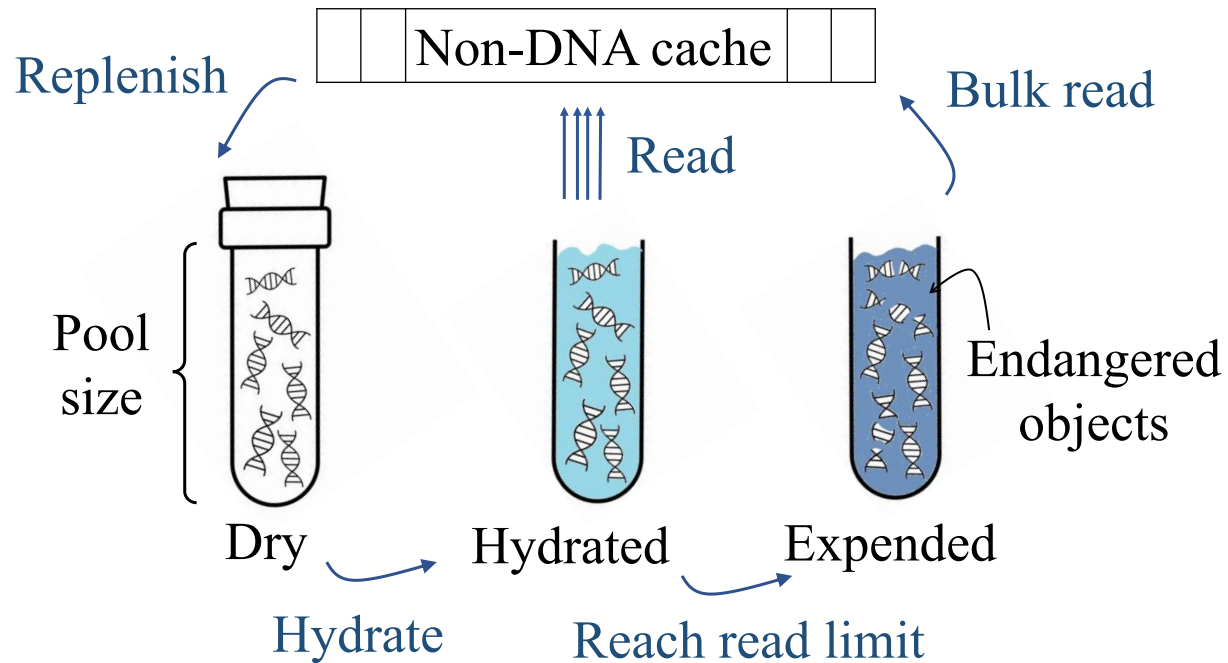
Costs and bottlenecks



Costs and bottlenecks

- System size: num of pools
- Pool size: num objects × object size
- Read limit: num of read operations
- Bulk read: multi-object reads vs. limit
- Digital cache size

Policies and objectives



Policies

- Inter-tube replication
- Mapping objects to pools
- Routing reads to pools
- Cache eviction

Objectives

- Minimum endangered objects
- Minimum hydrated pools

Simulation setup

Parameter	Values
Number of objects	100M
Number of requests	15M
Pool size	100, 1K, 10K
Read limit	1%, 10%, 100% of pool size
Replication factor	1, 2, 3, 4
Placement policy	r-hash, r-pool, copysets
Access distribution	Uniform, Zipf, Zipf-pool
Cache size	0, 0.1%, 1%
Cache policy	LRU
Read preference	Prefer hydrated pools

Access distributions

Uniform

all objects
equally likely



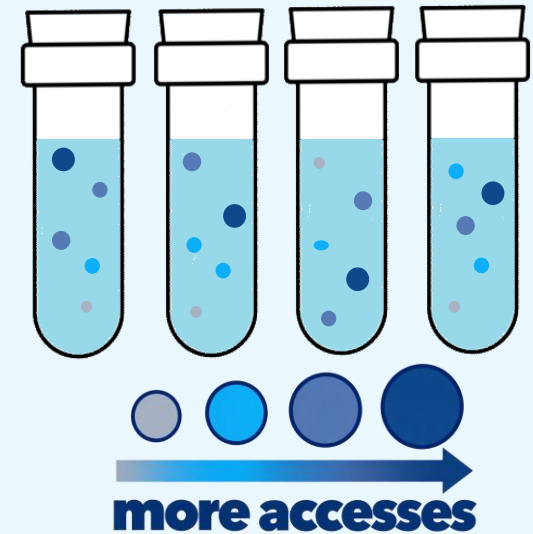
Simulation setup

Parameter	Values
Number of objects	100M
Number of requests	15M
Pool size	100, 1K, 10K
Read limit	1%, 10%, 100% of pool size
Replication factor	1, 2, 3, 4
Placement policy	r-hash, r-pool, copysets
Access distribution	Uniform, Zipf, Zipf-pool
Cache size	0, 0.1%, 1%
Cache policy	LRU
Read preference	Prefer hydrated pools

Access distributions

Zipf

popularity skew across objects

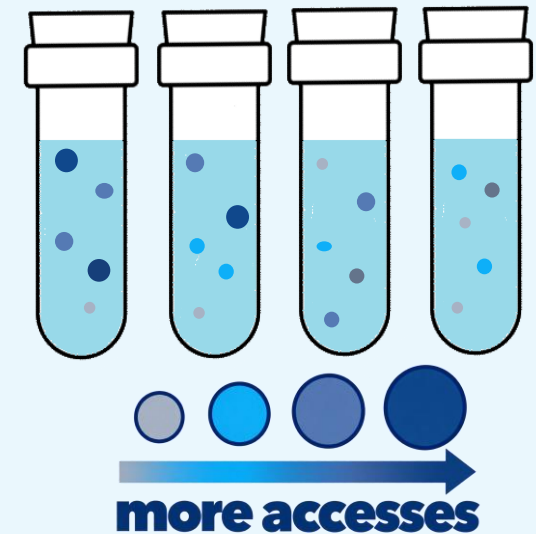


Simulation setup

Parameter	Values
Number of objects	100M
Number of requests	15M
Pool size	100, 1K, 10K
Read limit	1%, 10%, 100% of pool size
Replication factor	1, 2, 3, 4
Placement policy	r-hash, r-pool, copysets
Access distribution	Uniform, Zipf, Zipf-pool
Cache size	0, 0.1%, 1%
Cache policy	LRU
Read preference	Prefer hydrated pools

Access distributions

Zipf-pool popularity is skewed across pools and within each pool

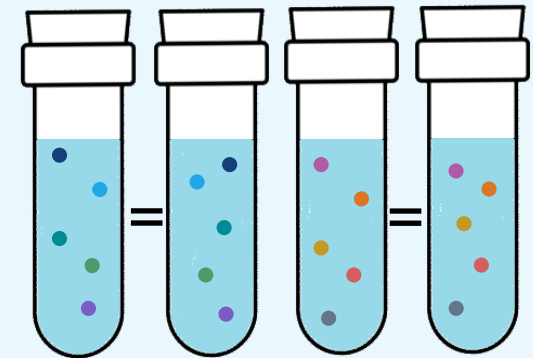


Simulation setup

Parameter	Values
Number of objects	100M
Number of requests	15M
Pool size	100, 1K, 10K
Read limit	1%, 10%, 100% of pool size
Replication factor	1, 2, 3, 4
Placement policy	r-hash, r-pool, copysets
Access distribution	Uniform, Zipf, Zipf-pool
Cache size	0, 0.1%, 1%
Cache policy	LRU
Read preference	Prefer hydrated pools

Placement policy

r-pool each pool replicated
r times



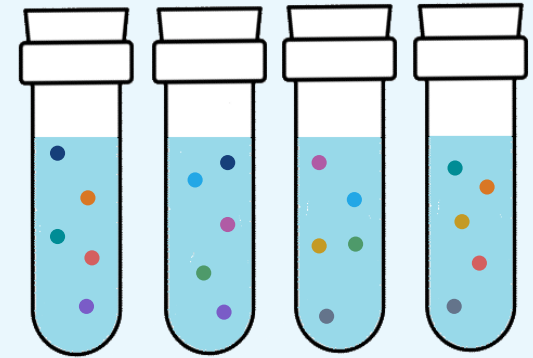
Same color = same object

Simulation setup

Parameter	Values
Number of objects	100M
Number of requests	15M
Pool size	100, 1K, 10K
Read limit	1%, 10%, 100% of pool size
Replication factor	1, 2, 3, 4
Placement policy	r-hash, r-pool, copysets
Access distribution	Uniform, Zipf, Zipf-pool
Cache size	0, 0.1%, 1%
Cache policy	LRU
Read preference	Prefer hydrated pools

Placement policy

r-hash replicas placed by r hash functions



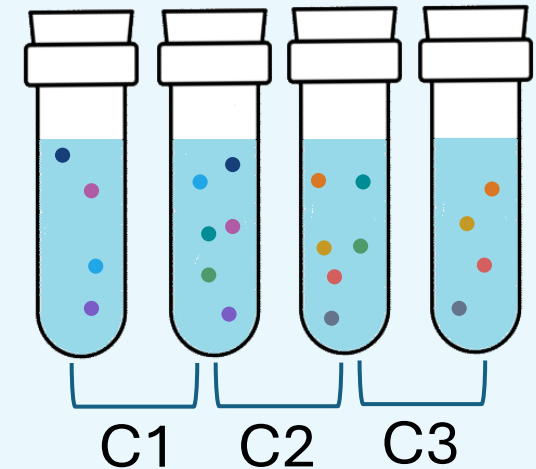
Same color = same object

Simulation setup

Parameter	Values
Number of objects	100M
Number of requests	15M
Pool size	100, 1K, 10K
Read limit	1%, 10%, 100% of pool size
Replication factor	1, 2, 3, 4
Placement policy	r-hash, r-pool, copysets
Access distribution	Uniform, Zipf, Zipf-pool
Cache size	0, 0.1%, 1%
Cache policy	LRU
Read preference	Prefer hydrated pools

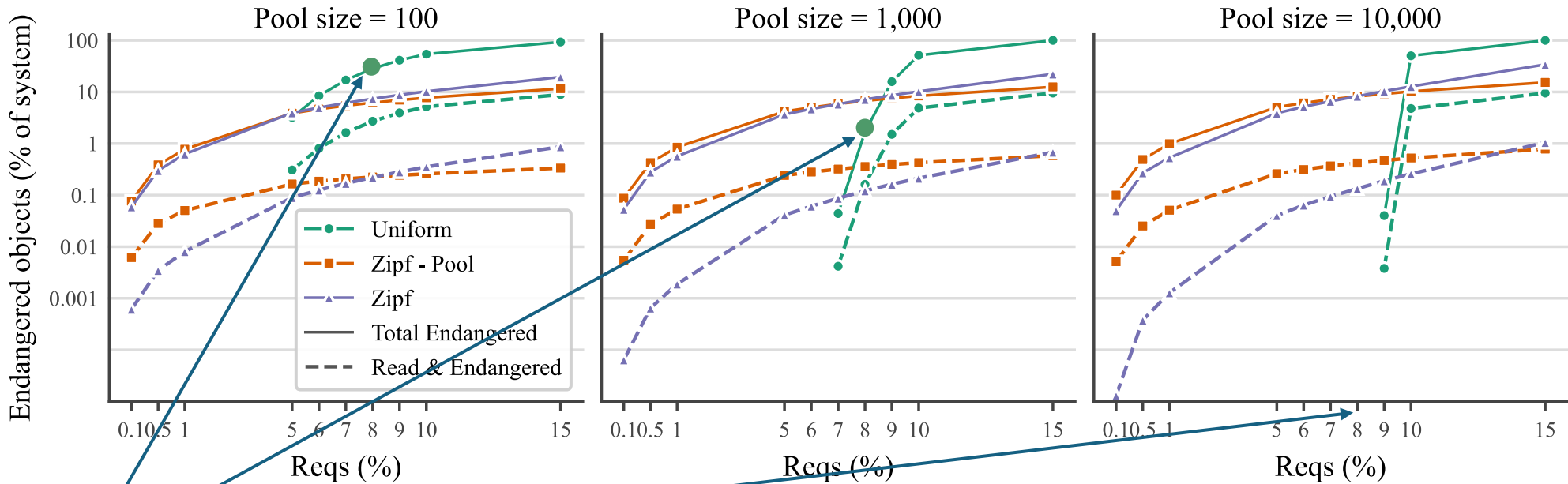
Placement policy

copysets replicas stay within one copyset



Pool size: endangered objects

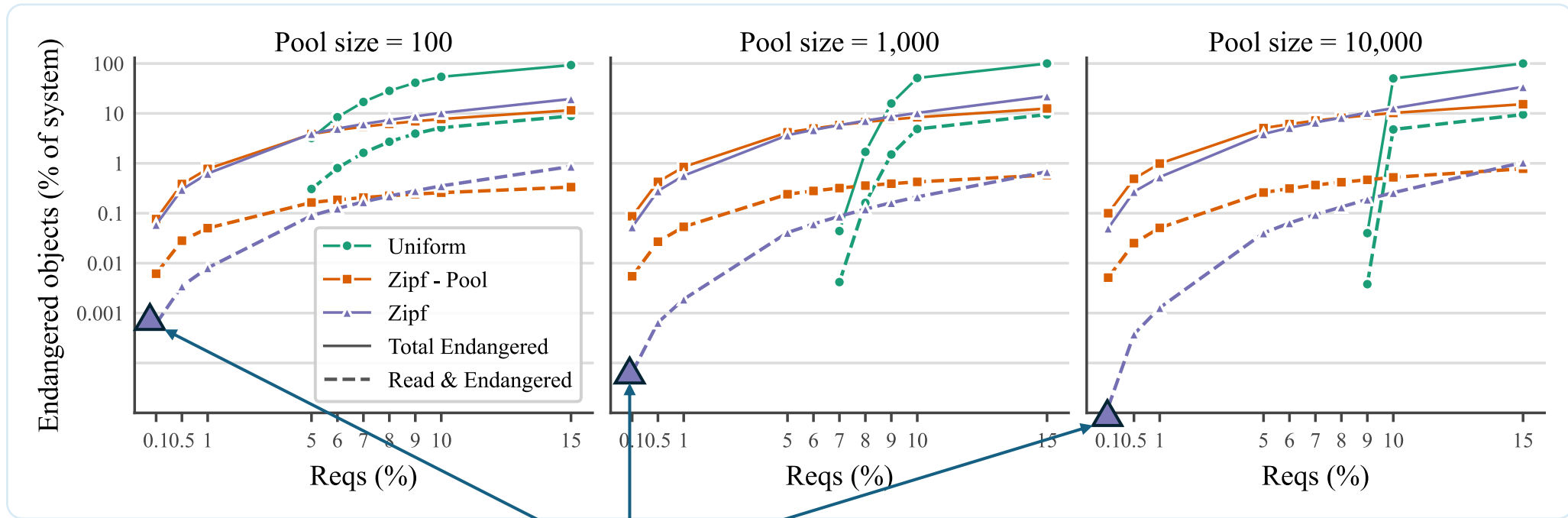
Read limit = 10% Replicas = 1



Larger pools → less endangered objects

Pool size: endangered objects

Read limit = 10% Replicas = 1

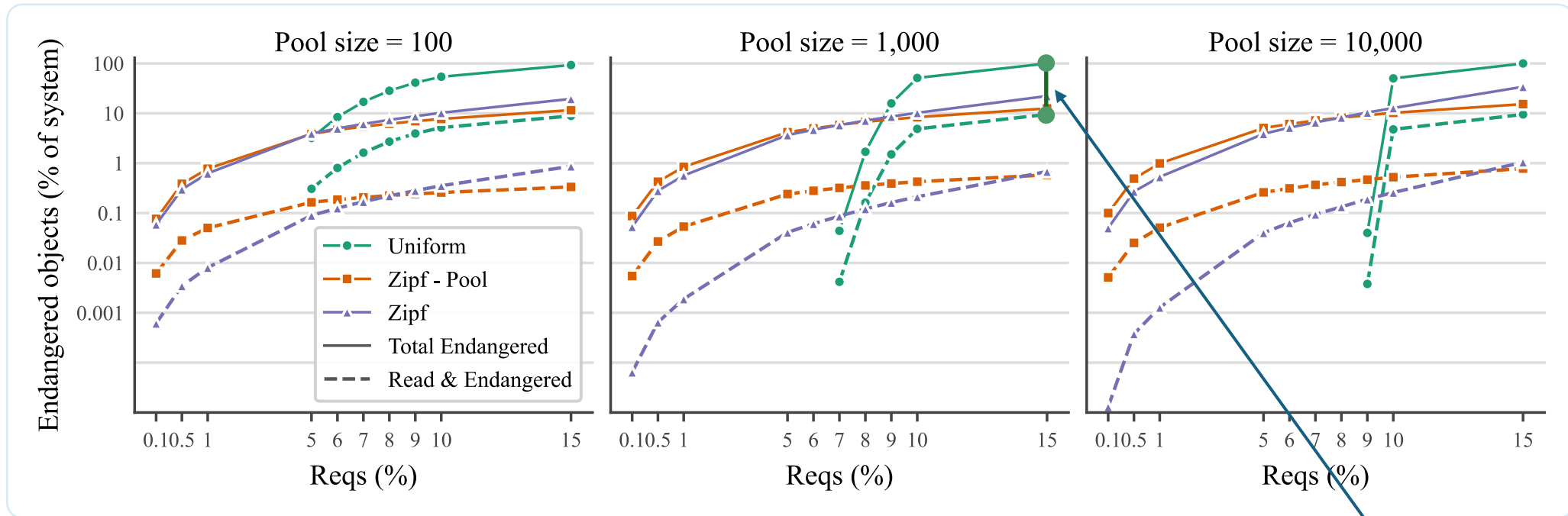


Larger pools → less endangered objects

Skewed access creates endangered objects very early

Pool size: endangered objects

Read limit = 10% Replicas = 1



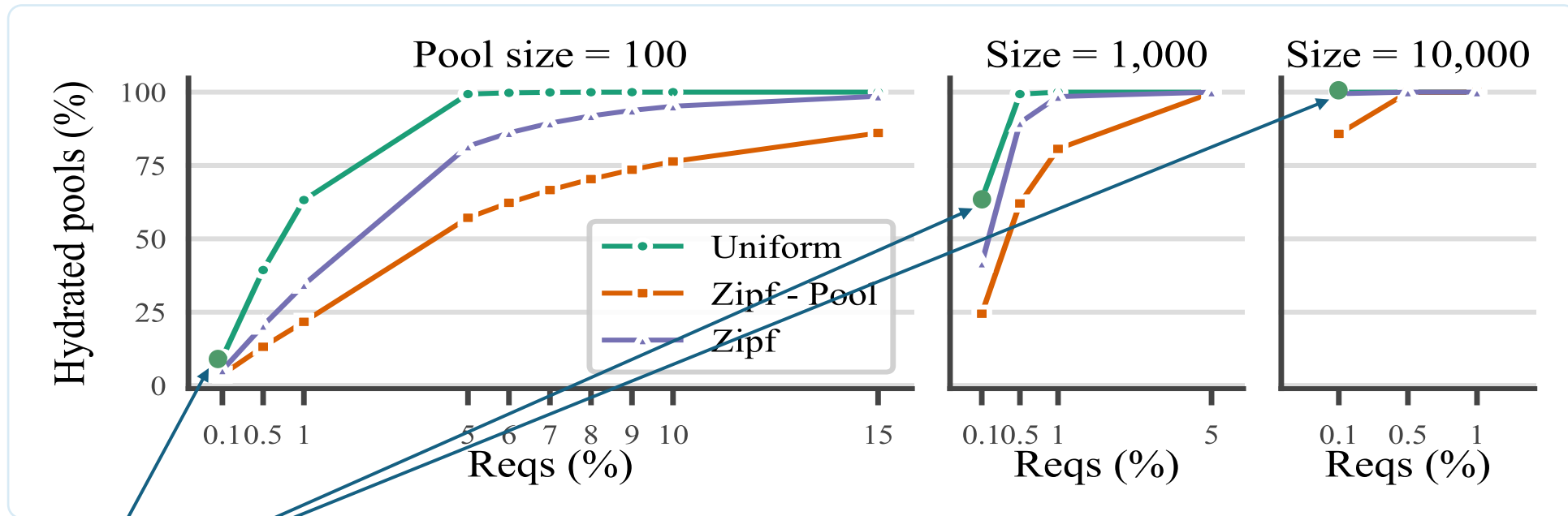
Larger pools → less endangered objects

Skewed access creates endangered objects very early

~90–95% of endangered objects were never read

Pool size: hydrated pools

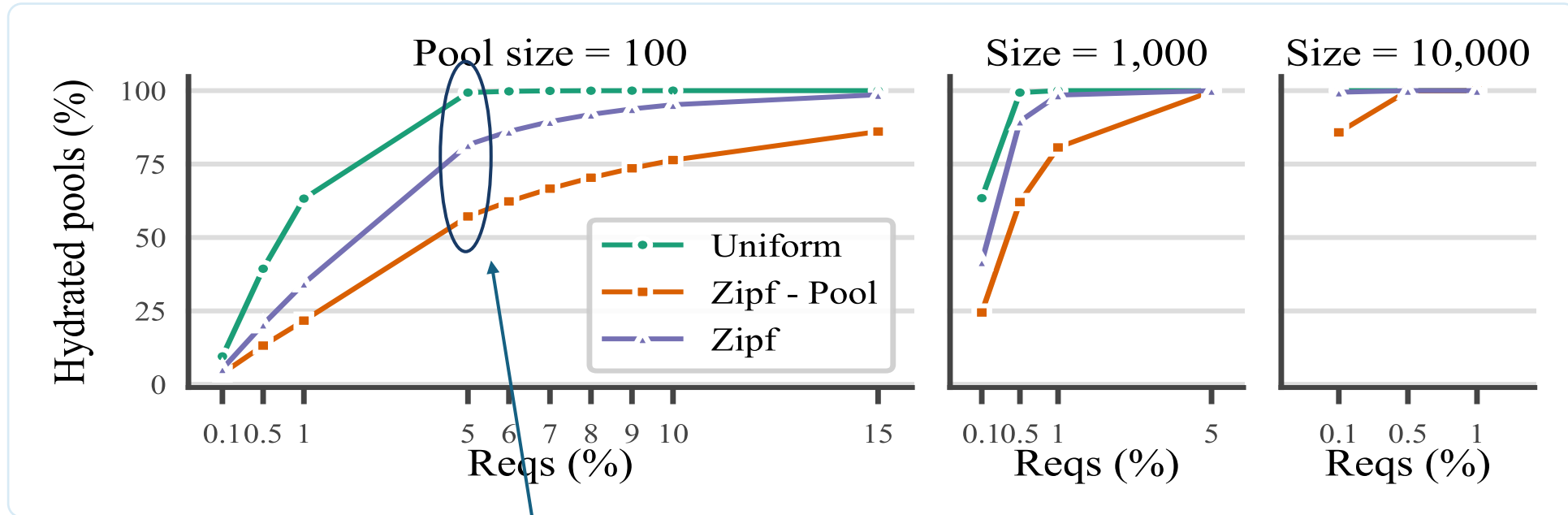
Read limit = 10% Replicas = 1



Larger pools → more hydrated pools

Pool size: hydrated pools

Read limit = 10% Replicas = 1

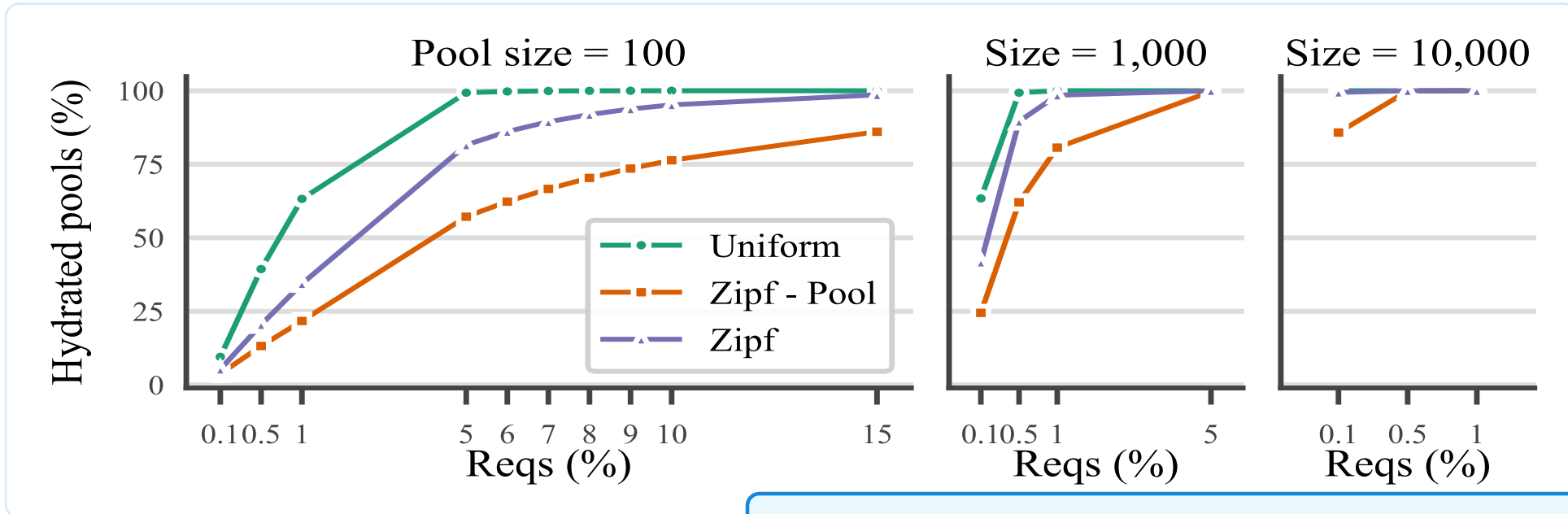


Larger pools → more hydrated pools

Zipf-pool → delays hydration

Pool size: hydrated pools

Read limit = 10% Replicas = 1



Larger pools → more hydrated pools

Zipf-pool → delays hydration

Main takeaways:

1

Large pools
→ expiration due to hydration

2

Large pools
→ might incur excessive costs

Additional takeaways (in the paper)

1

Mapping experiment:

load balancing delays pool expenditure but hydrates more tubes

2

Read-limit experiment:

higher read limits reduce replenishment costs

3

Caching experiment:

caching reduces costs under some workloads, but expended-pool reads remain

4

Caching experiment:

cache should be DNA-state aware

Summary



What we did

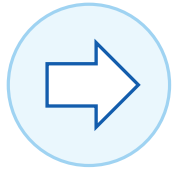
An initial system model for DNA storage

Capturing the system-level effects of DNA's physical constraints.



What we learned

**DNA storage is a new systems problem,
not just a new storage medium.**



Research opportunity

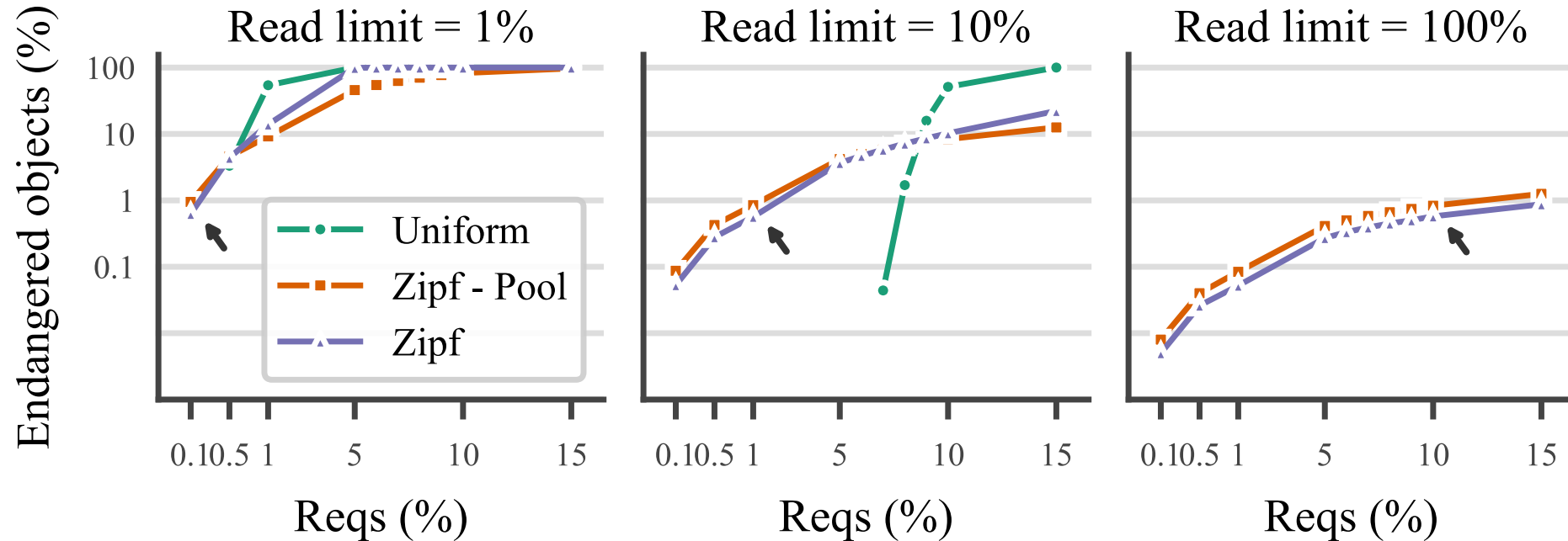
A new design space for the storage community.

DNA's physical constraints call for new system approaches and policies.

Thank you — Questions?

Read limit: endangered objects

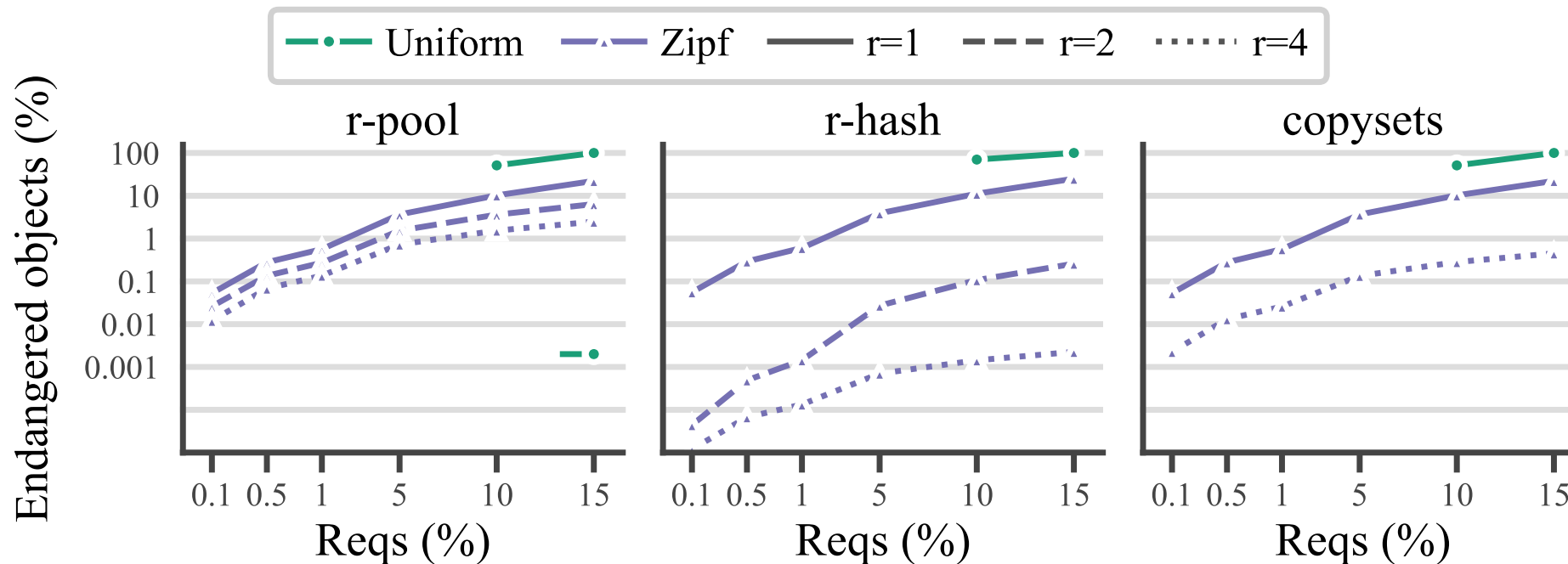
Pool size = 1000
Replicas = 1



higher read limits reduce
replenishment costs

Replication factor: endangered objects

Pool size = 1000
Read limit = 10%



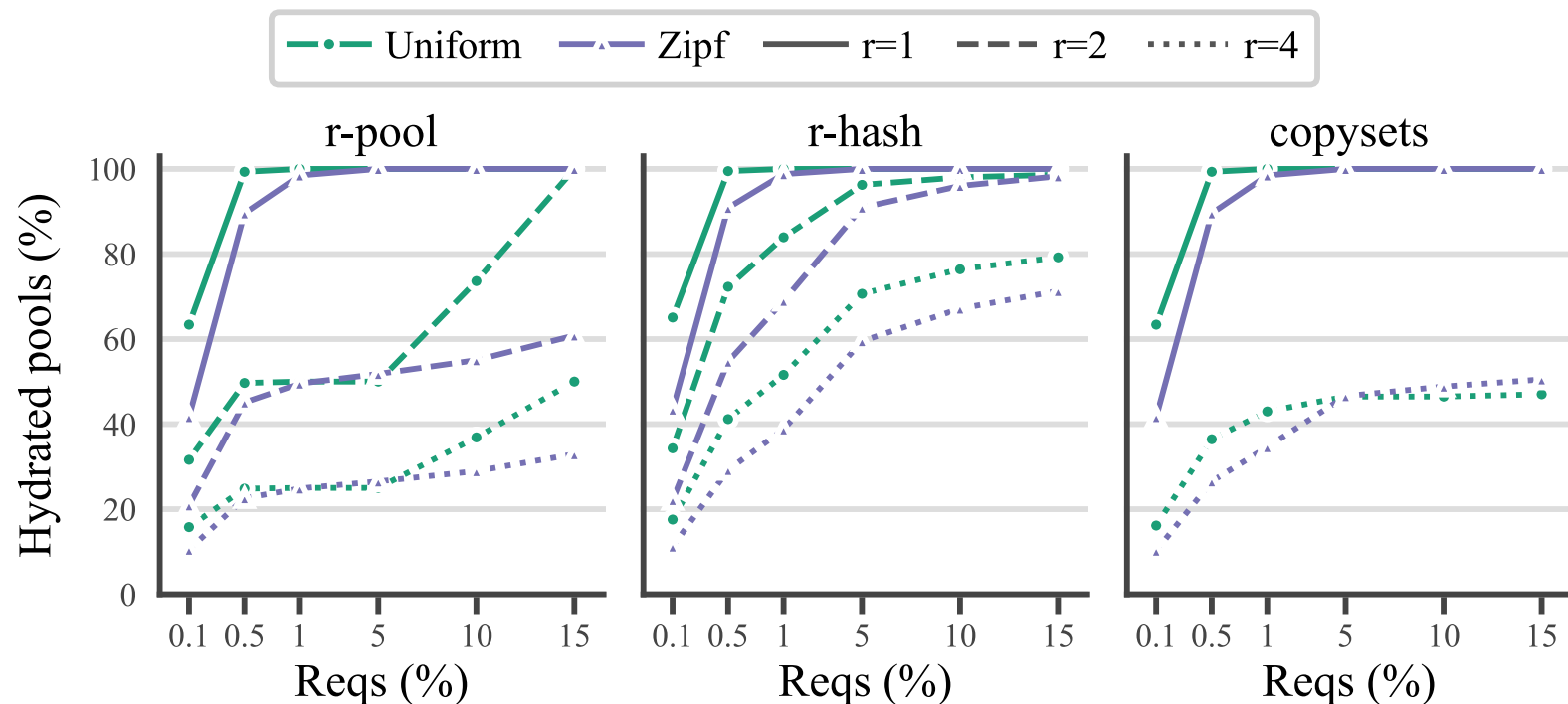
Uniform: adding even one replica almost eliminates object

Zipf & r-pool: modest effect, reads are concentrated

Zipf & r-hash: spreads replicas, balances read load

Replication factor: hydrated pools

Pool size = 1000
Read limit = 10%



r-pool: replicated pools → limited hydration spread

r-hash: Scattered replicas → large hydration blast radius

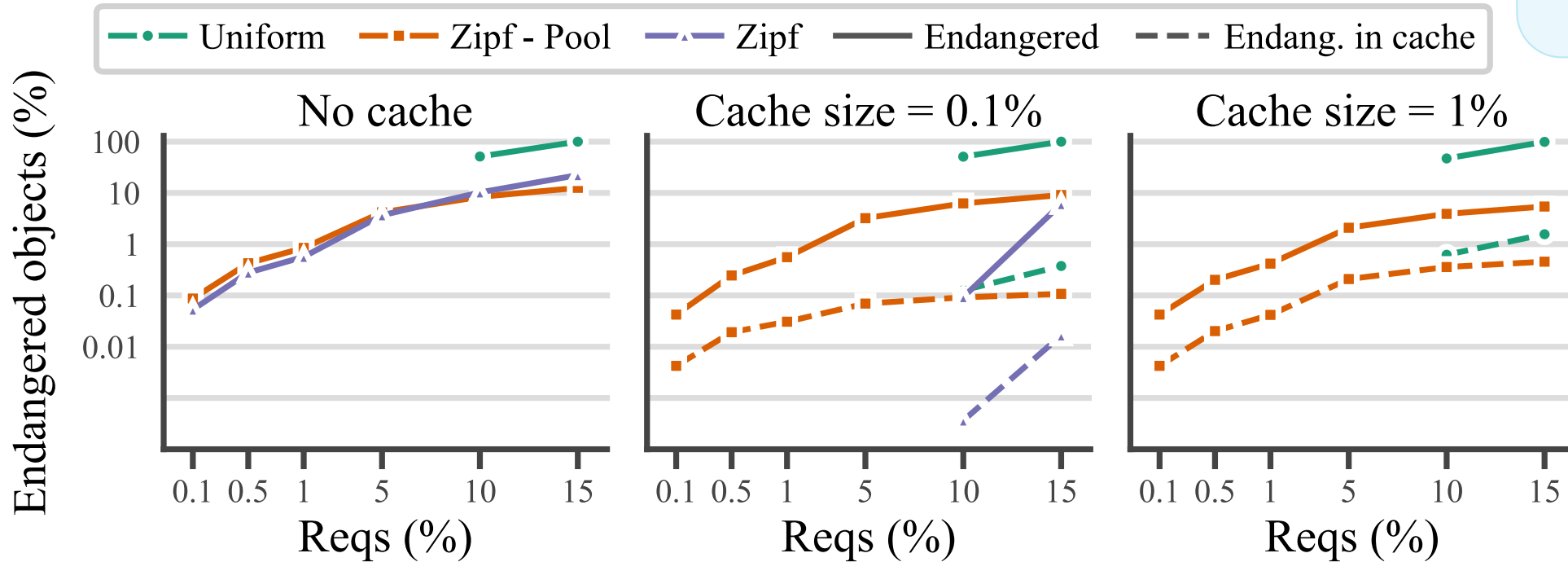
copysets: Constrained replicas → contains blast radius

Main takeaway:

load balancing delays pool expenditure
but hydrates more tubes

Caching: endangered objects

Pool size = 1000
Read limit = 10%
Replicas = 1
LRU



Uniform:
no effect

Zipf-pool:
modest effect

Zipf: can
eliminate
endangerment

Main takeaways:

1

Cache should be
DNA-state aware

2

Caching helps, but
expended-pool
reads remain