Beyond reCAP: Local Reads and Linearizable Asynchronous Replication

A. Katsarakis[*†], E. Gioratmis[*♠], V. Gavrielatos[†], P. Bhatotia[♠], A. Dragojevic[•], B. Grot[♣], V. Nagarajan[♣], P . Fatourou[♥]

[†] Huawei Research, [♠] TU Munich, [•] Citadel Securities, [♣] University of Edinburgh, [♥] University of Crete and FORTH, [*] Equal contribution

Motivation

Online Services & Cloud Applications

Characterized by

- Many **concurrent requests**
- **Read intensive** workloads
- Need for **data reliability** → run on fault-prone h/w

Fault-tolerant Replicated Datastores

- **Crash-tolerance**: data are replicated
- **High performance**: especially for reads
- **Strong consistency** under **asynchrony** → correct — even if timeouts do not hold

Crash-tolerant Replication Protocols

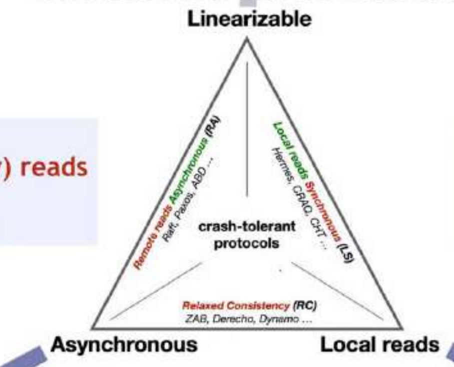
determine actions for *reads* and *writes*

Ideal features

1. **Linearizable**
2. **Asynchronous**
3. **Local reads**: for max perf.

Theory

Crash-tolerant protocols: 2 out of 3



RA protocols:

- Remote (costly) reads
- + Linearizable
- + Asynchronous

LS protocols:

- Synchronous
- + Linearizable
- + Local reads

RC protocols:

- Relaxed Consistency
- + Asynchronous + Local reads

The L^2AW theorem

Any *Linearizable* *Asynchronous* read/write register implementation that tolerates a crash (**W**ithout blocking reads or writes), has no *Local reads*.

So can we not improve read performance without compromizes?

L^2AW vs. CAP

Both Linearizability & Asynchrony

L^2AW read performance in its tradeoff
Key for read-dominant workloads

Fault-tolerance

CAP: network partitions
+ msg loss + partitioned nodes exec ops to violate safety

L^2AW: server crashes
+ no msg loss + crashed nodes do not exec ops to violate safety

When must compromise?

CAP: during network partitions
(not during partition-free)
sacrifice safety or progress of ops

L^2AW: always sacrifice local reads
(even if crashes have not occurred)

Practice

Almost Local Reads (ALRs)

Inevitably ALR latency > local reads

But **little or no extra** network and processing **costs to remote replicas**

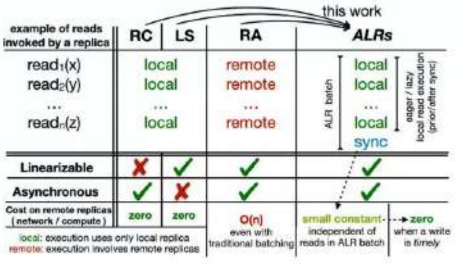
ALRs batch reads with a twist

Exec all reads in batch w/ local replica
+ one sync per batch on remote nodes

Syncs are cheap!

- writes act as implicit zero-cost syncs
- explicit sync has small constant cost
- 1 sync per batch regardless its size

Add missing piece to protocols of all 3 (RC, LS, RA) categories

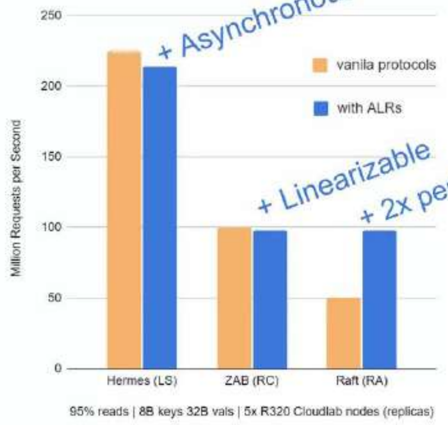

this work
ALR batch
eager / lazy local read execution (prior/after sync)

example of reads invoked by a replica	RC	LS	RA	ALRs	
$read_1(x)$	local		remote	local	
$read_2(y)$	local		remote	local	
...	...		...	...	
$read_n(z)$	local		remote	local	
				sync	
Linearizable	✗	✓	✓	✓	
Asynchronous	✓	✗	✓	✓	
Cost on remote replicas (network / compute)	zero	zero	O(n) even with traditional batching	small constant - independent of reads in ALR batch	→ zero when a write is timely

local: execution uses only local replica
remote: execution involves remote replicas

✓ RC with ALRs → **Linearizable**
✓ LS with ALRs → **Asynchronous**
✓ RA with ALRs → **Performant**

ALR-enhanced throughput of state-of-the-art protocols



95% reads | 8B keys 32B vals | 5x R320 Cloudlab nodes (replicas)

HUAWEI · TUM · THE UNIVERSITY of EDINBURGH informatics · persist · H.F.R.I. Hellenic Foundation for Research & Innovation

[†] This work occurred while the authors were at the University of Edinburgh