DSN 2023

Āpta: Fault-tolerant object-granular CXL disaggregated memory for accelerating FaaS

Adarsh Patil

Vijay Nagarajan

Nikos Nikoleris

Nicolai Oswald

THE UNIVERSITY of EDINBURGH
informatics

arm

NVIDIA

Āpta
Reliable
CXL.mem

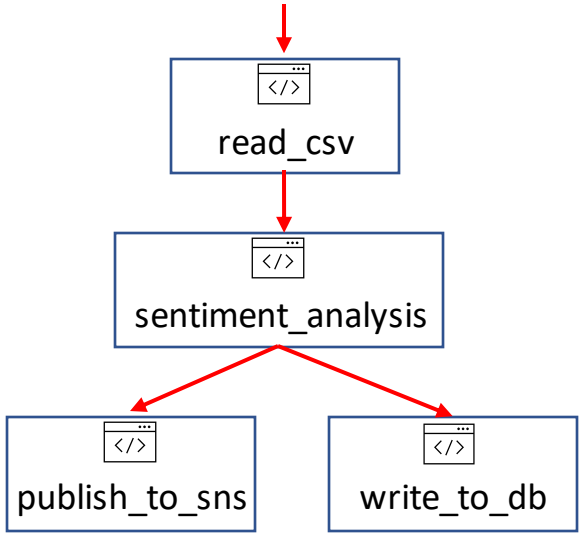
Function-as-a-Service or “Serverless”

Function-as-a-Service or “Serverless”

Function-as-a-Service or “Serverless”

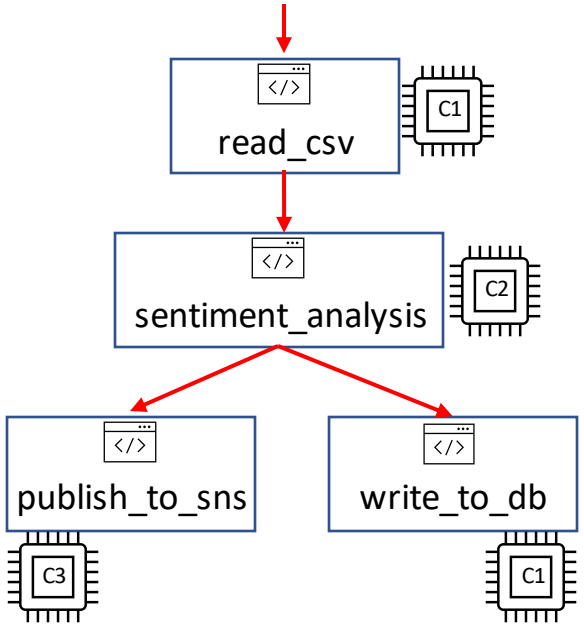
FaaS applications

- State machine workflow of *stateless* functions



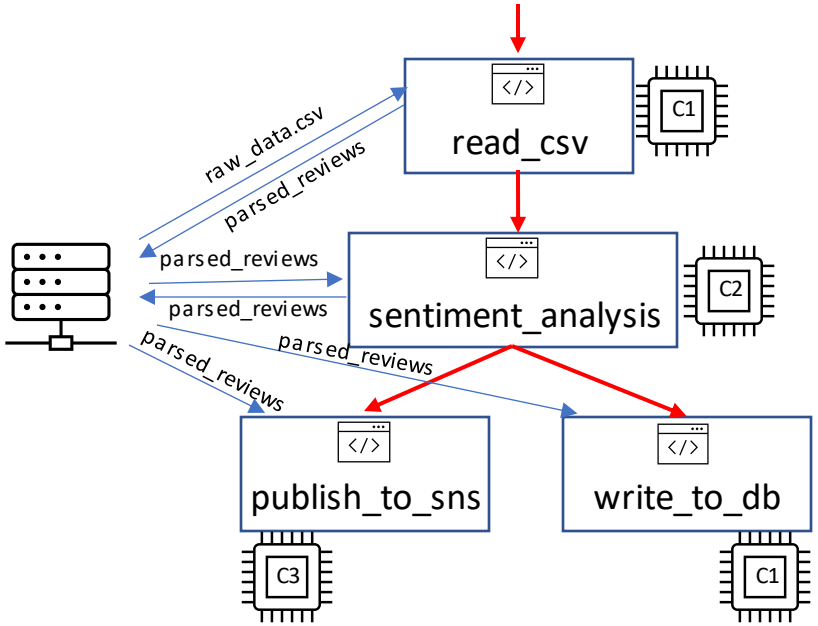
FaaS applications

- State machine workflow of *stateless* functions
- Cloud provider dynamically orchestrates and schedules functions on a fleet of compute servers



FaaS applications

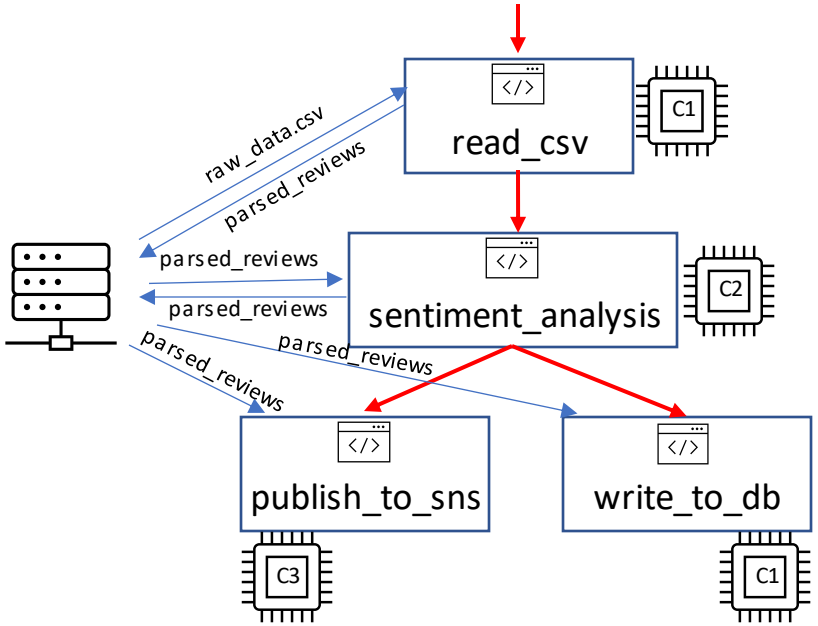
- State machine workflow of *stateless* functions
- Cloud provider dynamically orchestrates and schedules functions on a fleet of compute servers
- State maintained externally as objects in a remote data store



FaaS applications

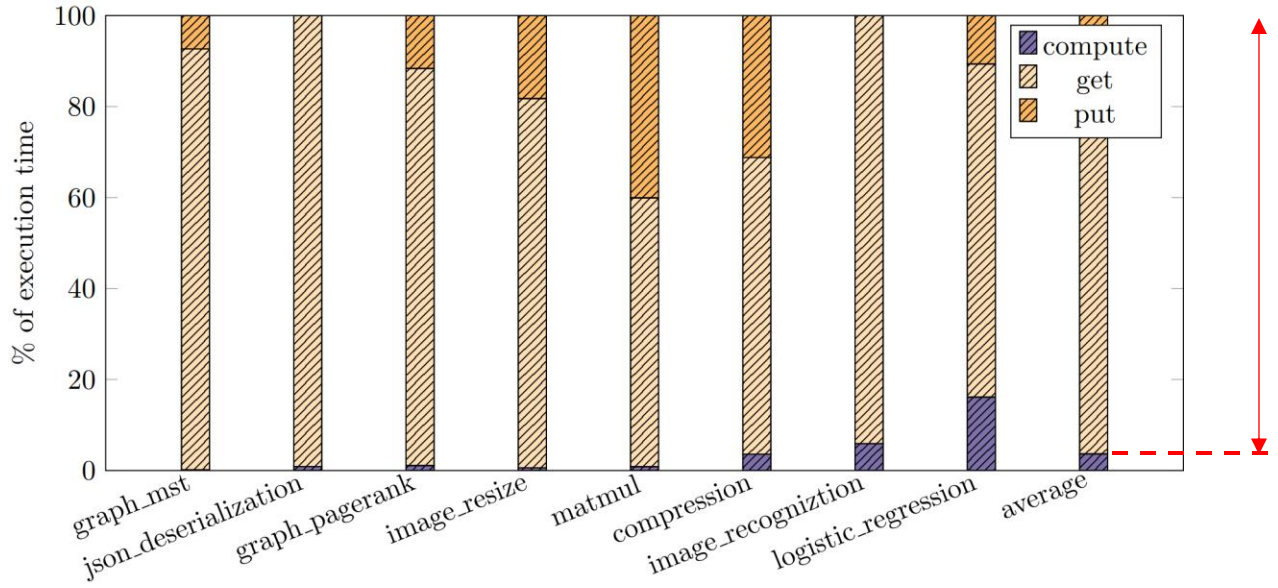
- State machine workflow of *stateless* functions
- Cloud provider dynamically orchestrates and schedules functions on a fleet of compute servers
- State maintained externally as objects in a remote data store
  - × Splitting state-compute adds communication overheads

How much?



Quantifying communication overheads

- Functions from FunctionBench and SeBS benchmark suites
- Compute - Optimized with Intel OneAPI, run on 16-core Skylake CPUs
- Communication - Amazon S3 object store (median of 100 executions)

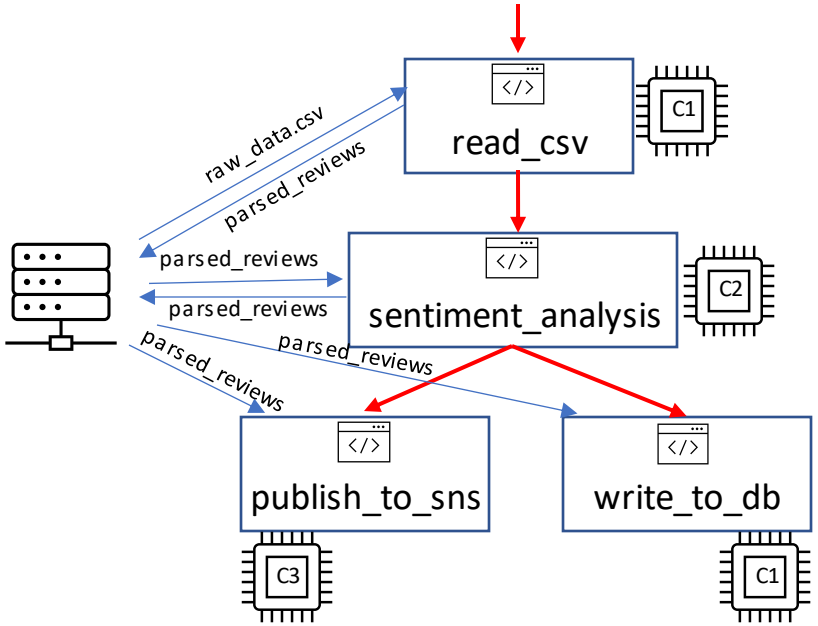


96% of execution time is spent in retrieving data from S3

Inefficiency of FaaS applications

- State machine workflow of *stateless* functions
- Cloud provider dynamically orchestrates and schedules functions on a fleet of compute servers
- State maintained externally as objects in a remote data store
  - × Splitting state-compute adds communication overheads
  - × Communication overheads severely limit performance

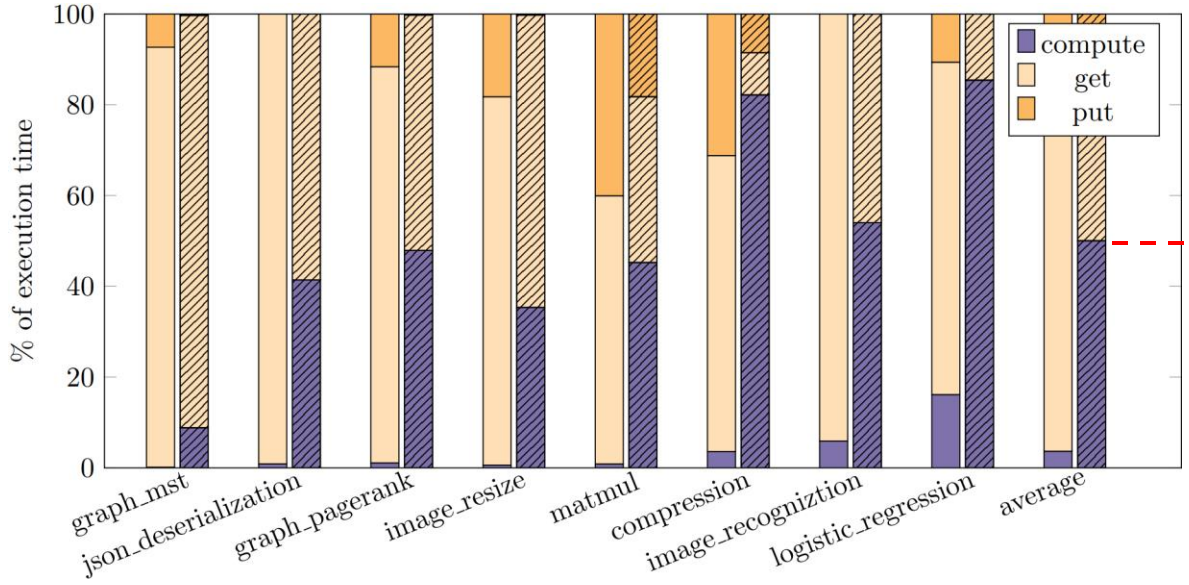
Can we do better?



Āpta
Reliable
CXL.mem

Can we do better?

- High-performance in-memory object store
- One-sided RDMA verbs to read/write objects
- Infiniband network (Mellanox ConnectX-3 NIC on PCIe-gen3 x16)



51% of execution time is spent in retrieving data from object store

The problem: Communication overheads

"The two most expensive operations in terms of cost were the orchestration workflow and when data passed between distributed components."

Āpta architecture

Object-granular CXL disaggregated memory

Āpta architecture

Object-granular CXL disaggregated memory

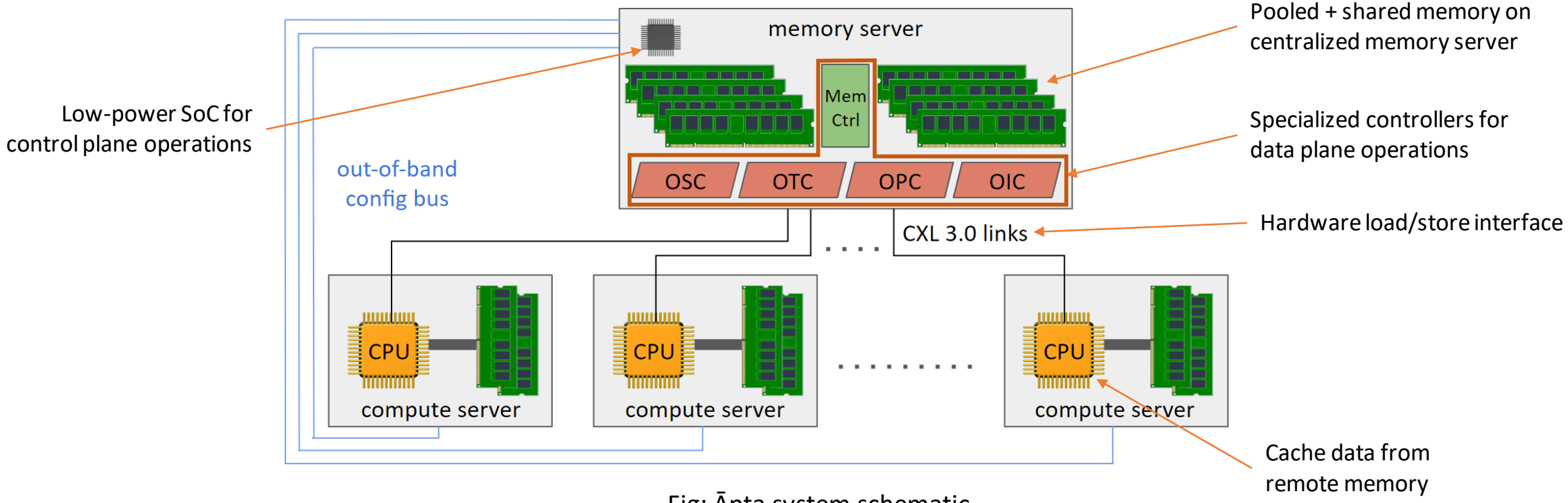
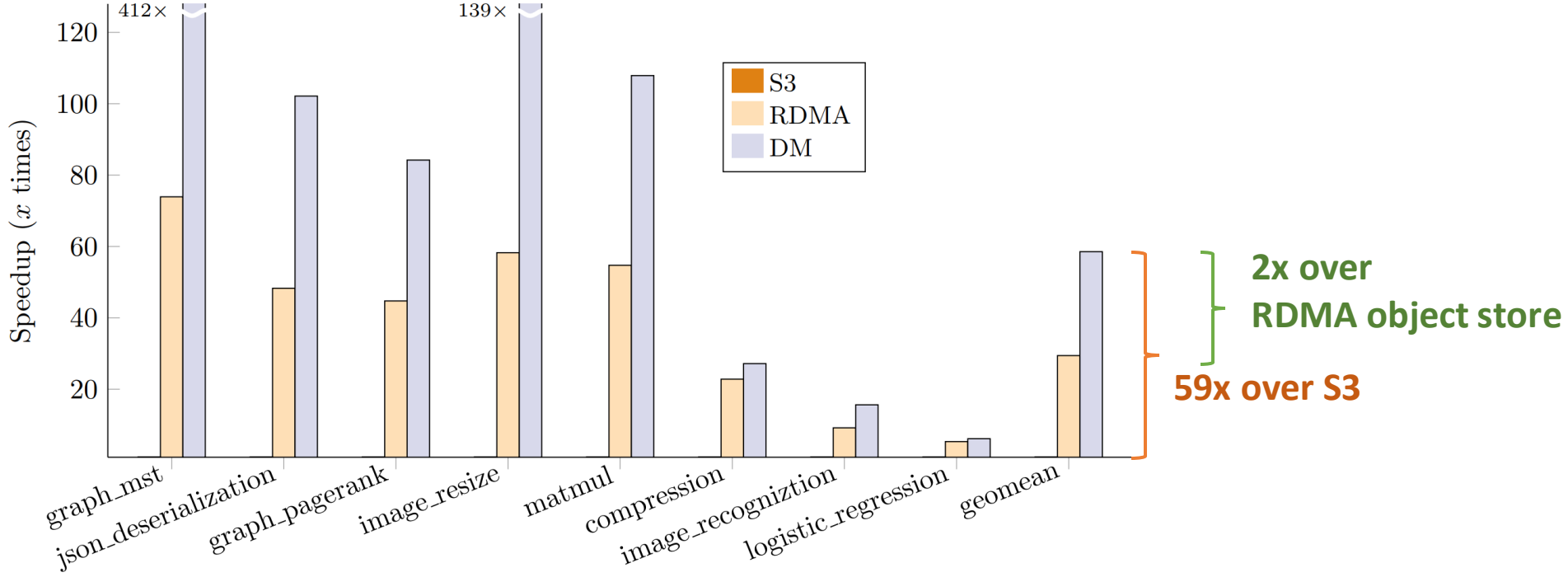


Fig: Āpta system schematic

Performance potential of Āpta

With disaggregated memory - OpenCAPI-like access latency / bandwidth†

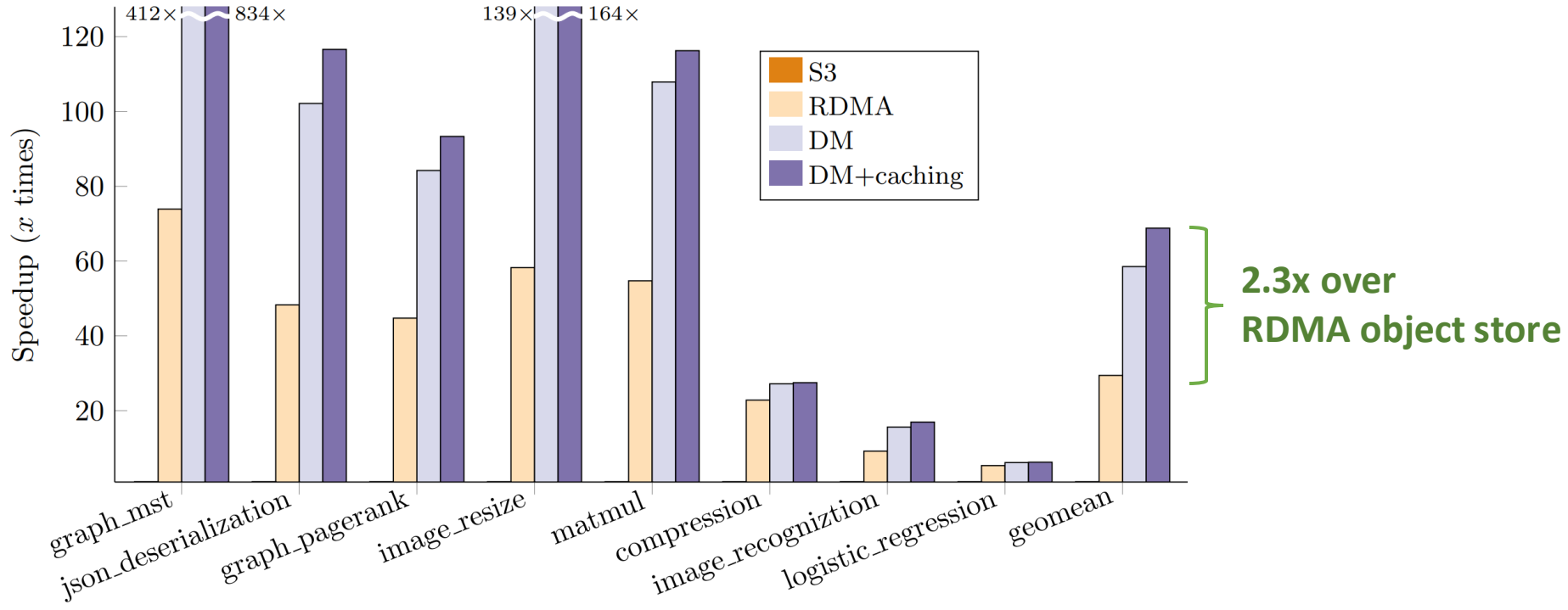


13% communication overheads (Recall 51% for RDMA-based object store)

† ThymesisFlow [MICRO 20]

Performance potential of Āpta

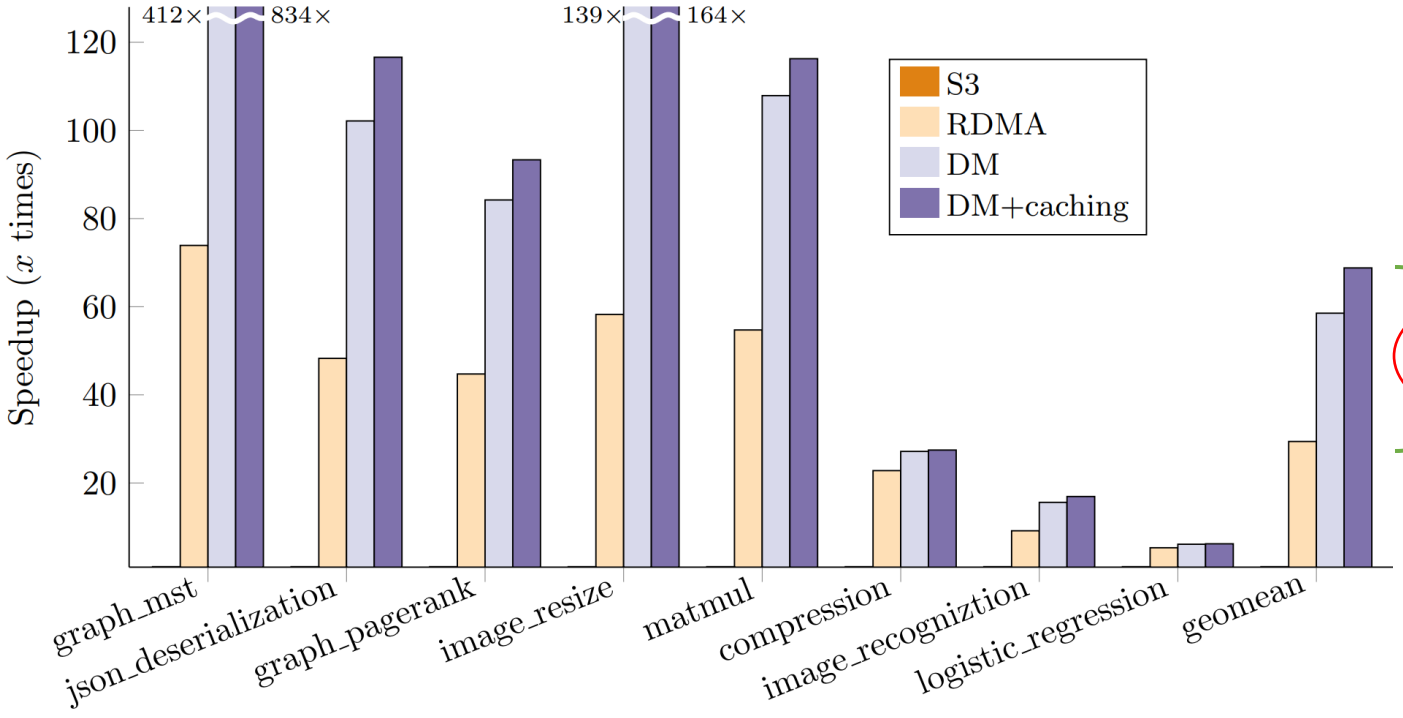
With object caching at compute server



2.3x over RDMA object store

Performance potential of Āpta

With object caching at compute server



2.3x over RDMA object store

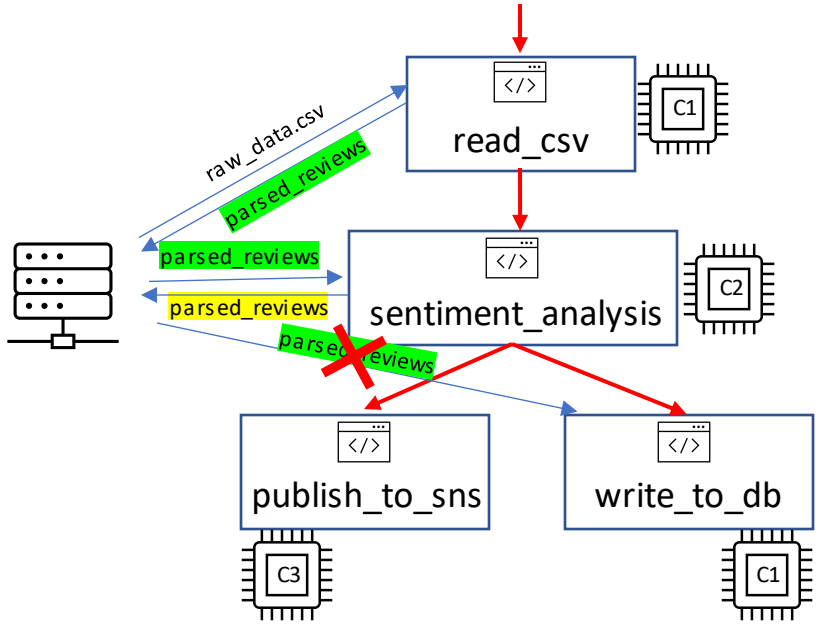
Our target!

The CXL.mem coherence

- Enforcing strong consistency in presence of caching

The CXL.mem coherence

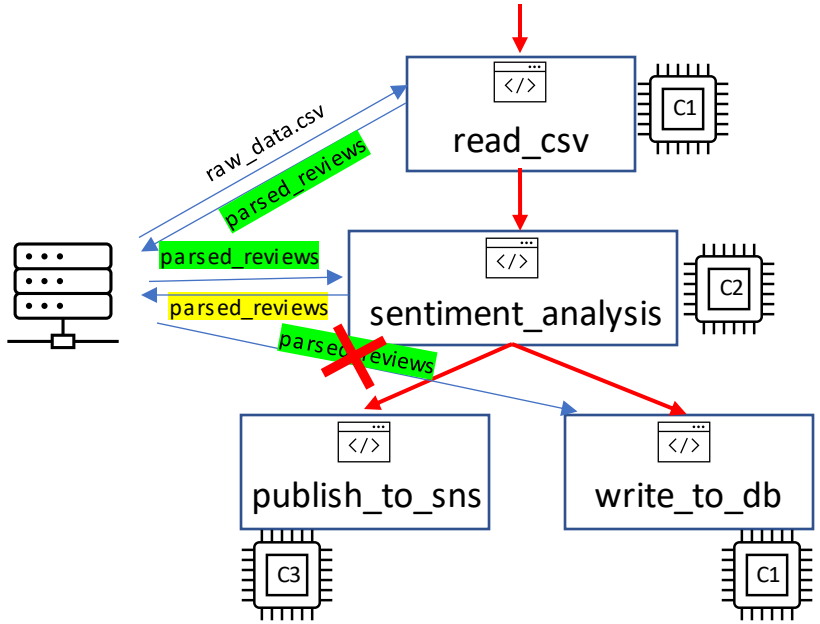
- Enforcing strong consistency in presence of caching



Āpta
Reliable CXL.mem

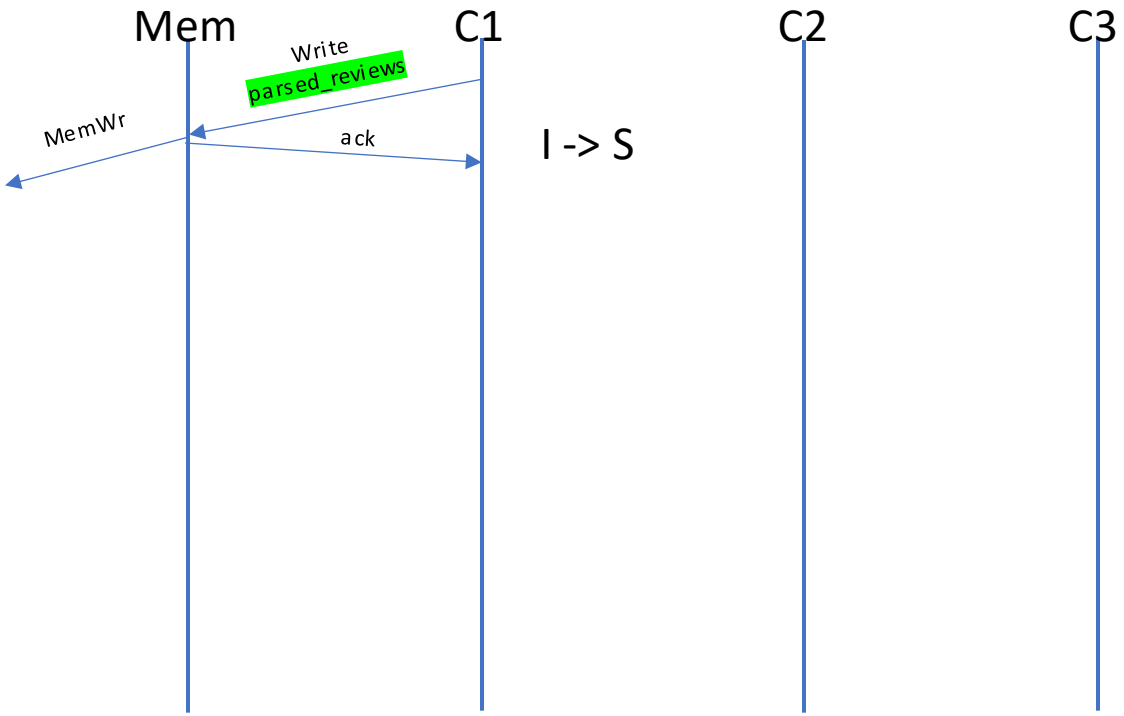
The CXL.mem coherence

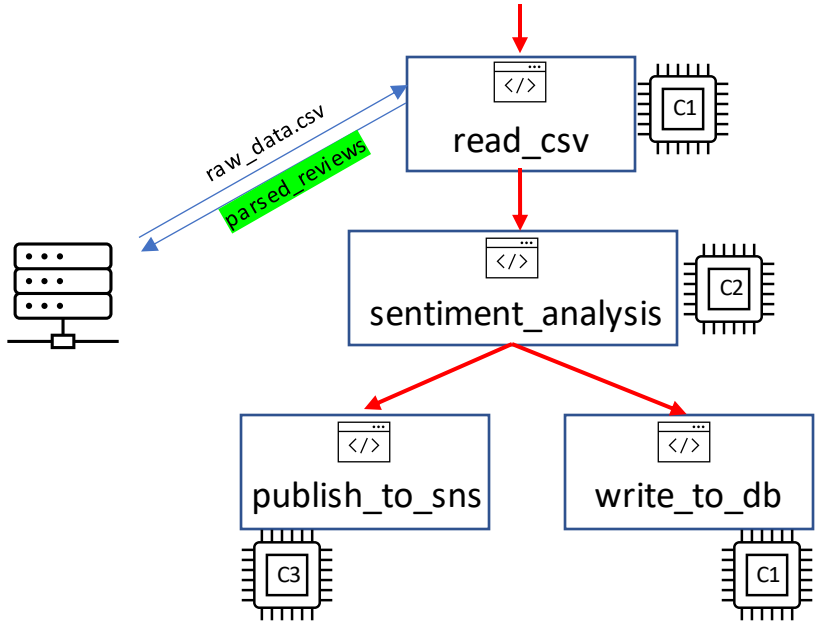
- Enforcing strong consistency in presence of caching
  - CXL 3.0 inter-node coherence protocol
  - Enforces SWMR invariant



The CXL.mem coherence

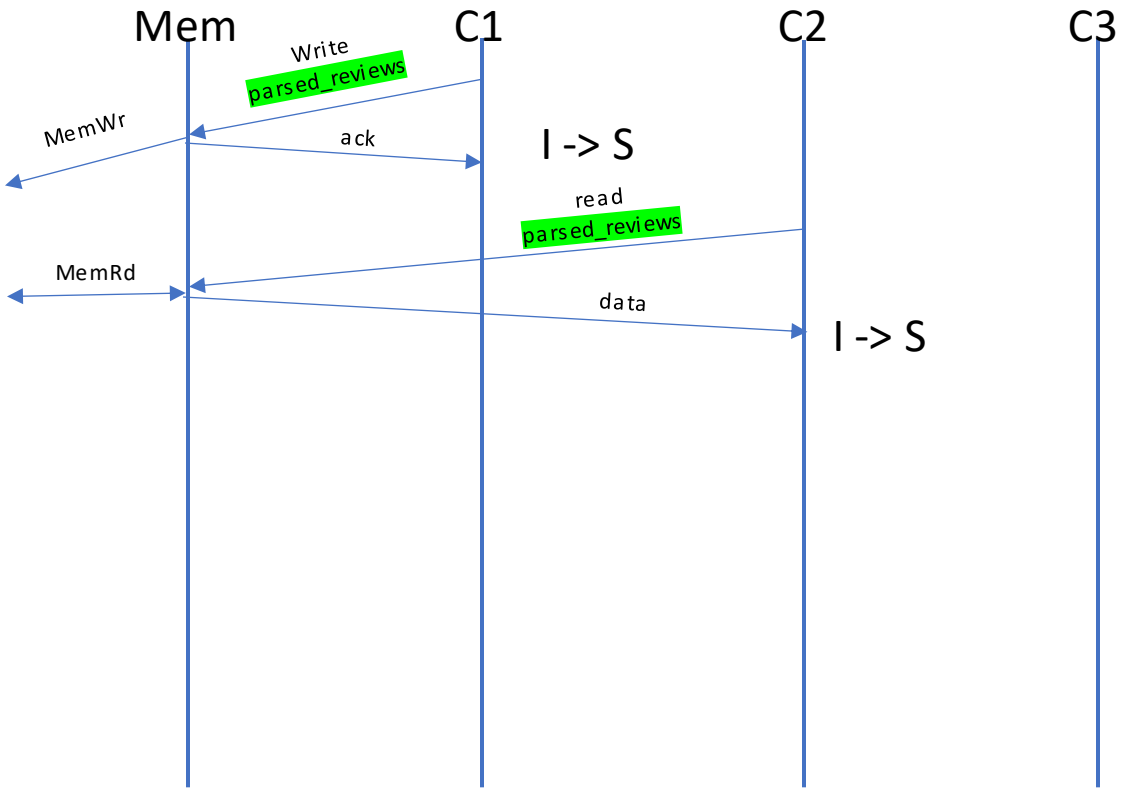
- Enforcing strong consistency in presence of caching
  - CXL 3.0 Inter-node coherence protocol
  - Enforces SWMR invariant

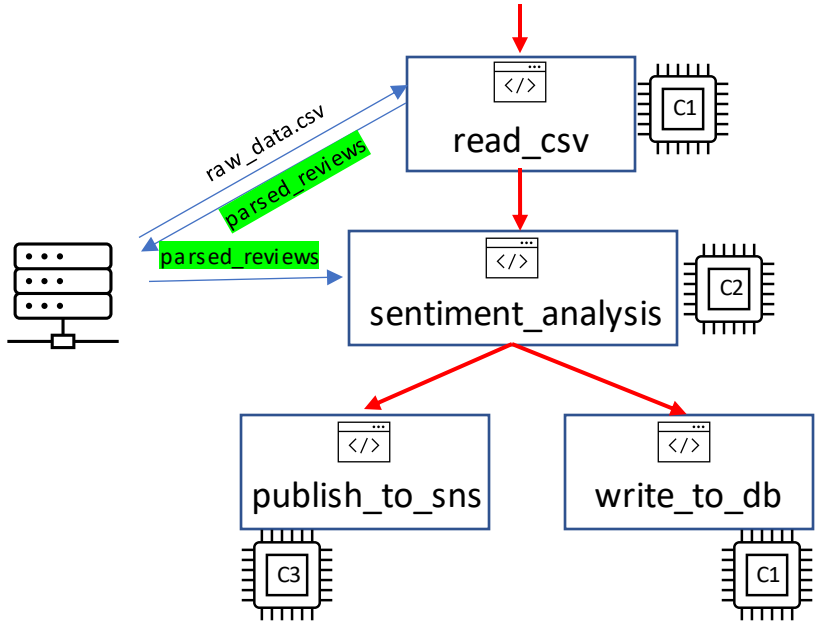




The CXL.mem coherence

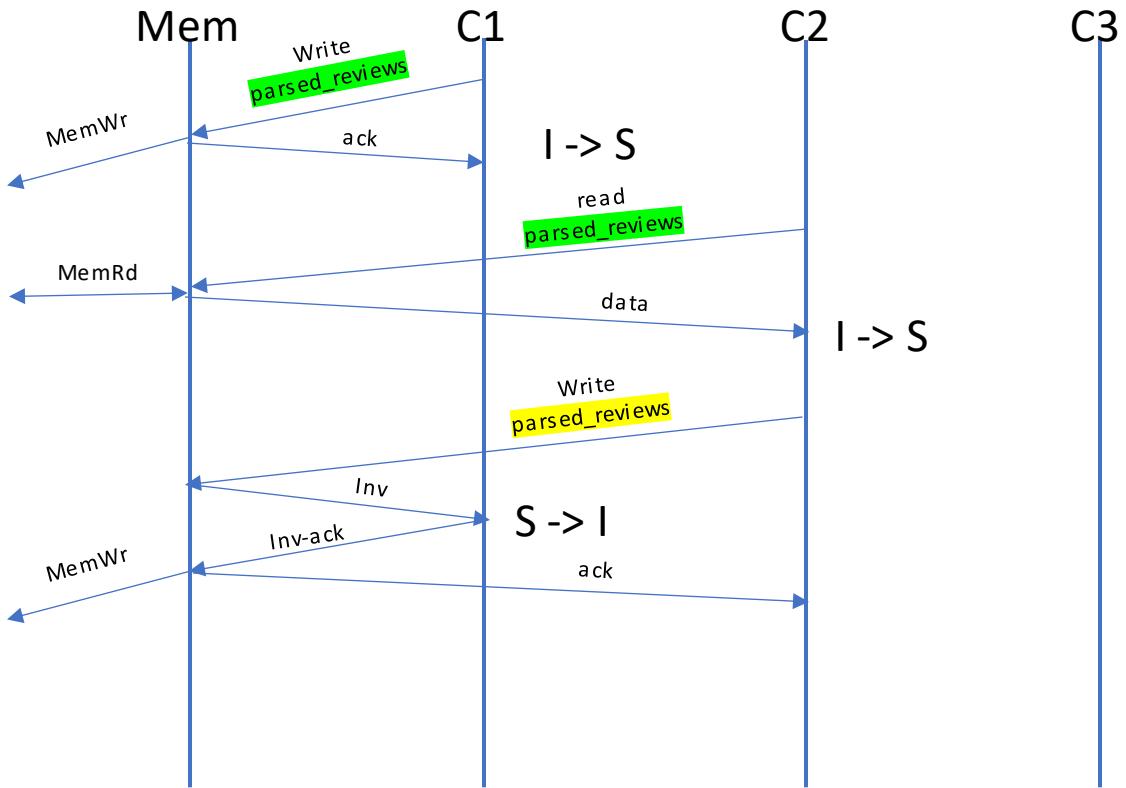
- Enforcing strong consistency in presence of caching
  - CXL 3.0 Inter-node coherence protocol
  - Enforces SWMR invariant

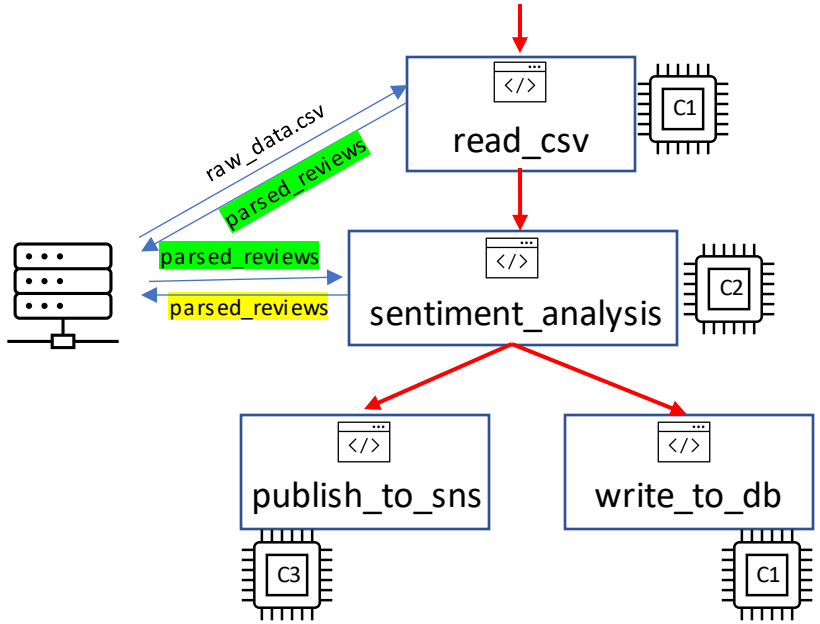




The CXL.mem coherence

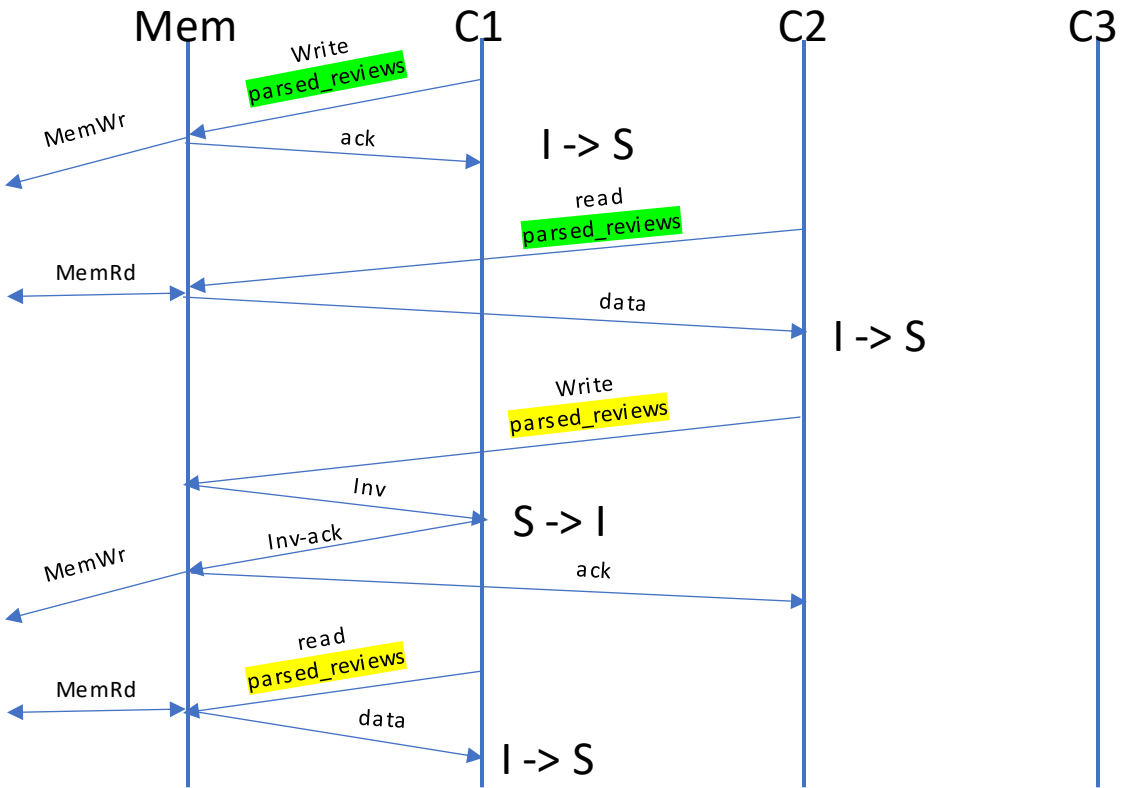
- Enforcing strong consistency in presence of caching
  - CXL 3.0 Inter-node coherence protocol
  - Enforces SWMR invariant

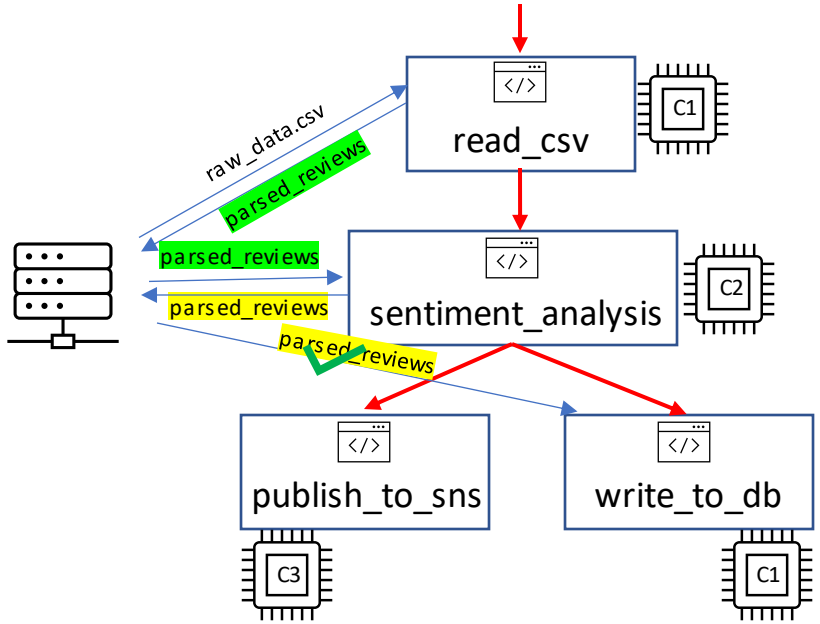




The CXL.mem coherence

- Enforcing strong consistency in presence of caching
  - CXL 3.0 Inter-node coherence protocol
  - Enforces SWMR invariant





The need for fault-tolerant coherence

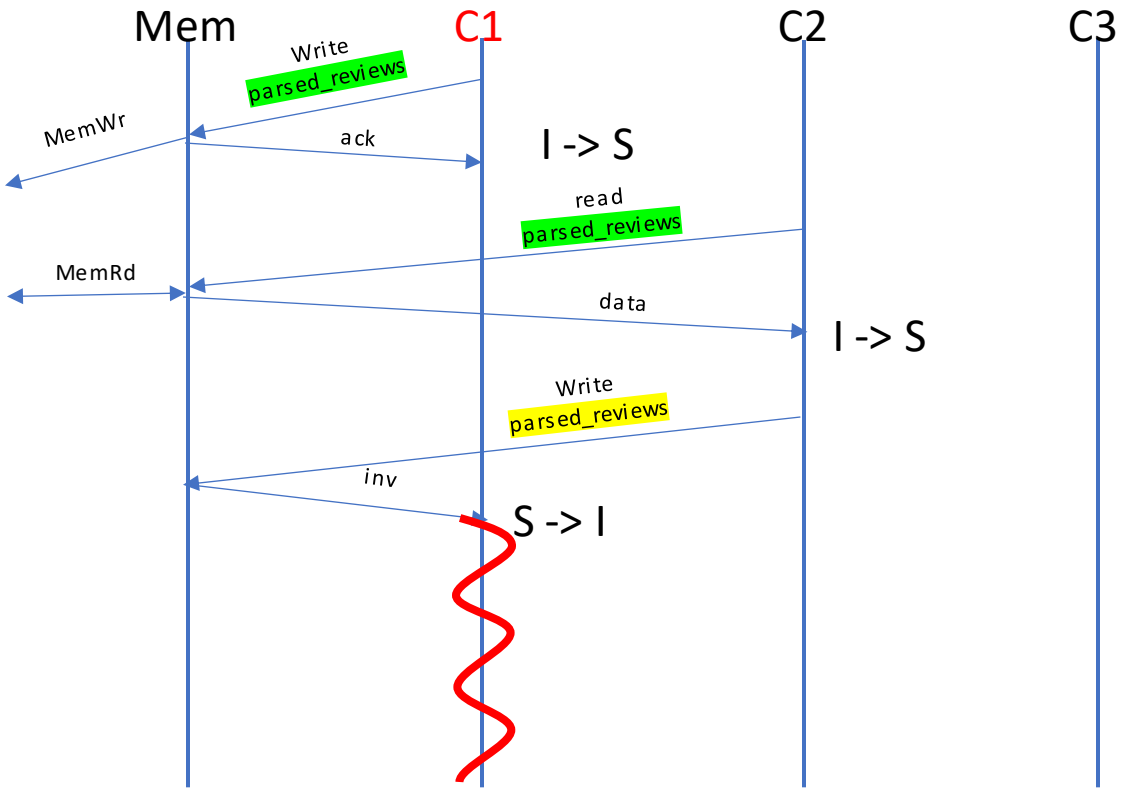
- The fault tolerance problem

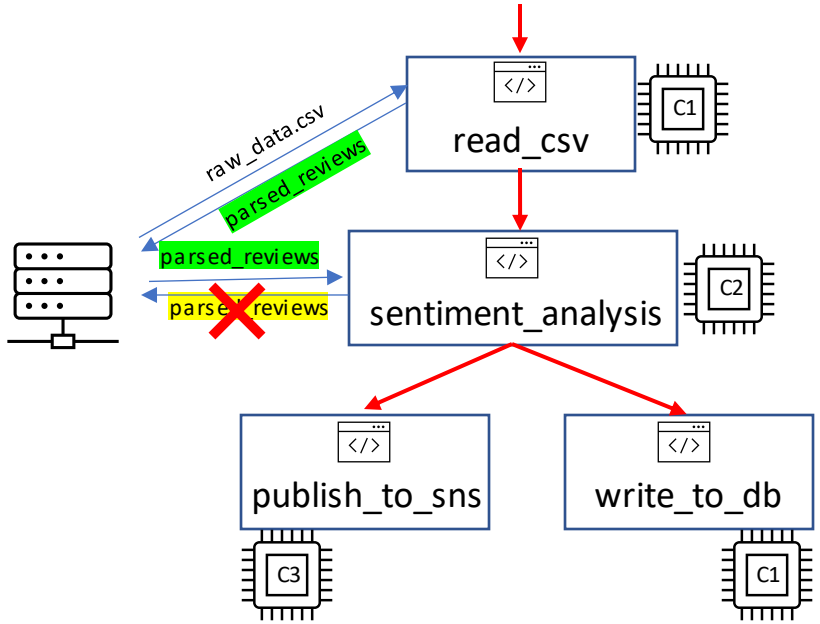
  Compute server failures

The need for fault-tolerant coherence

- The fault tolerance problem

  Compute server failures

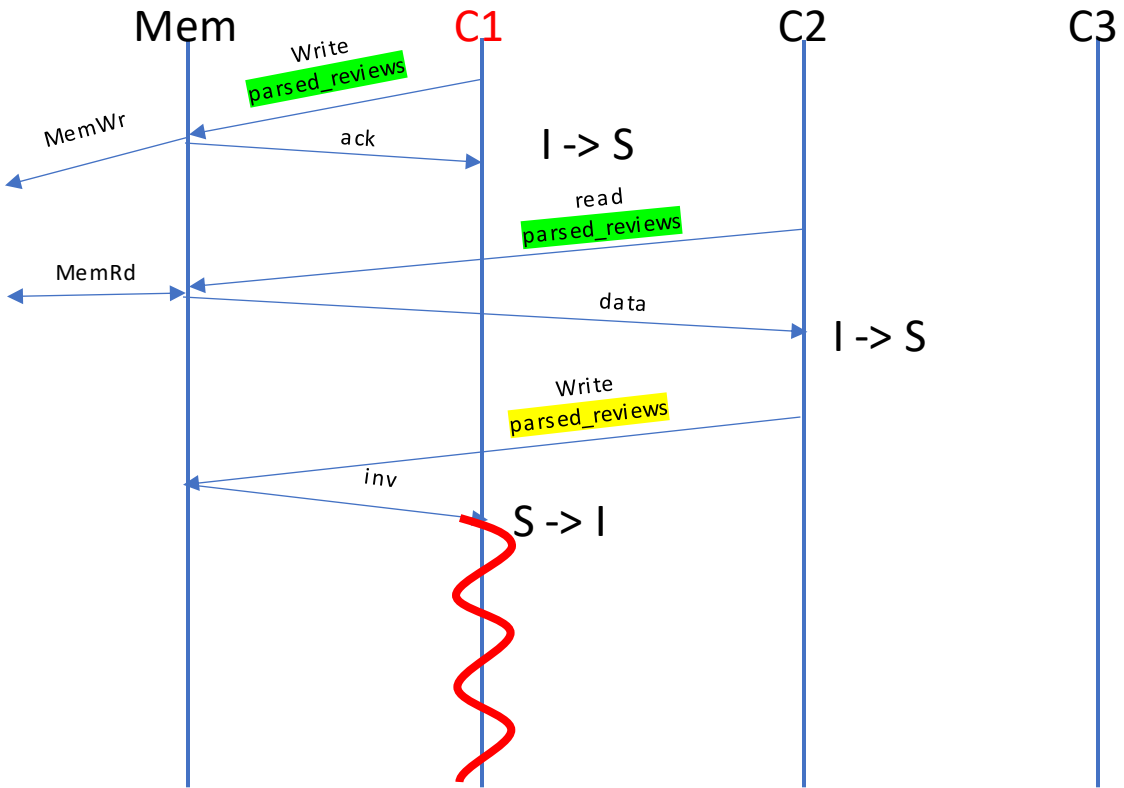


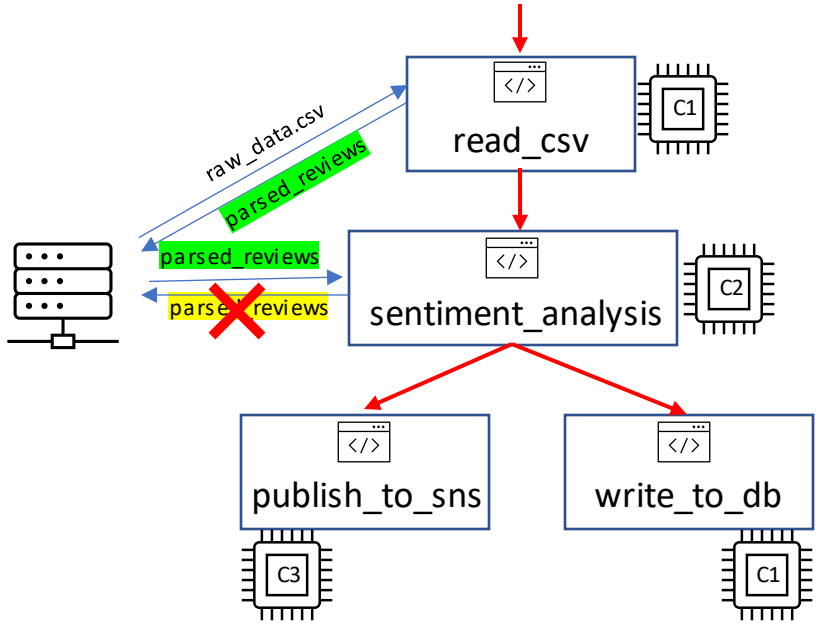


The need for fault-tolerant coherence

- The fault tolerance problem

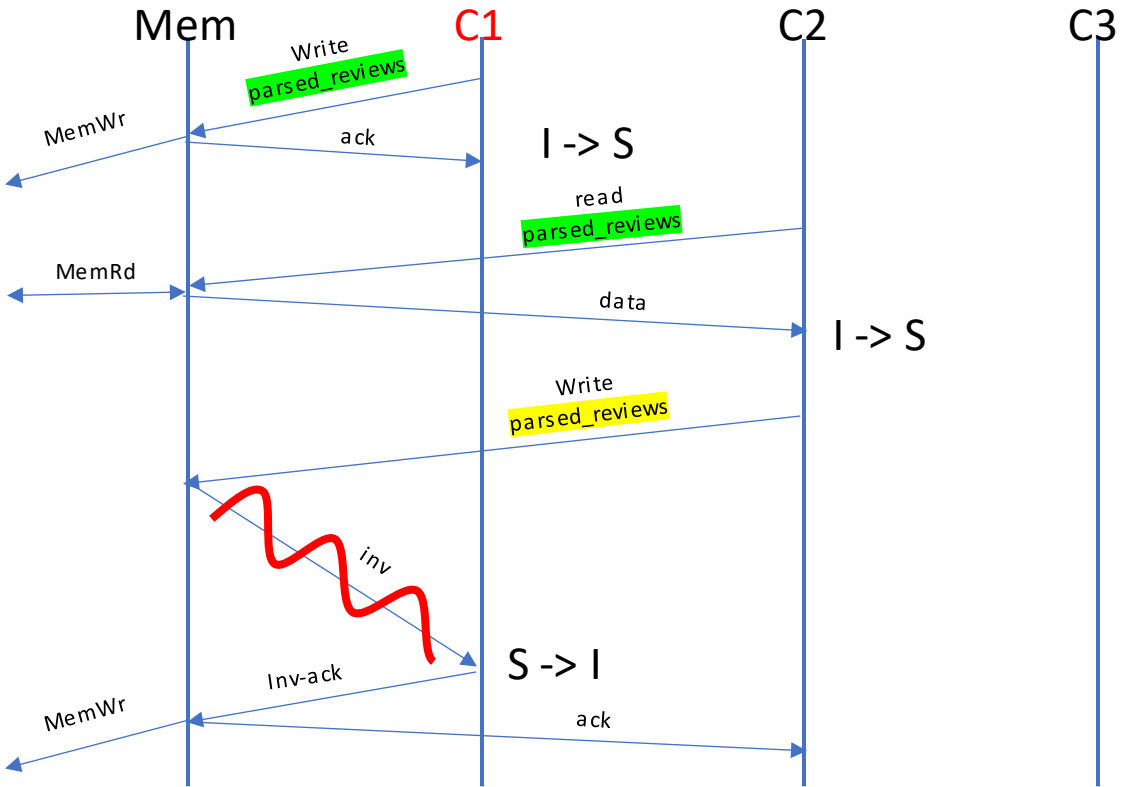
  Compute server failures – blocking

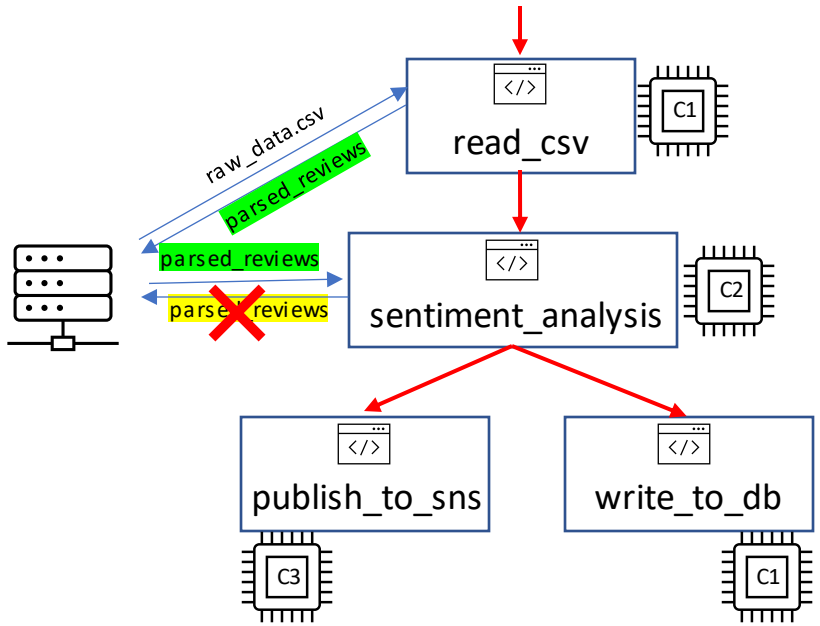




The need for fault-tolerant coherence

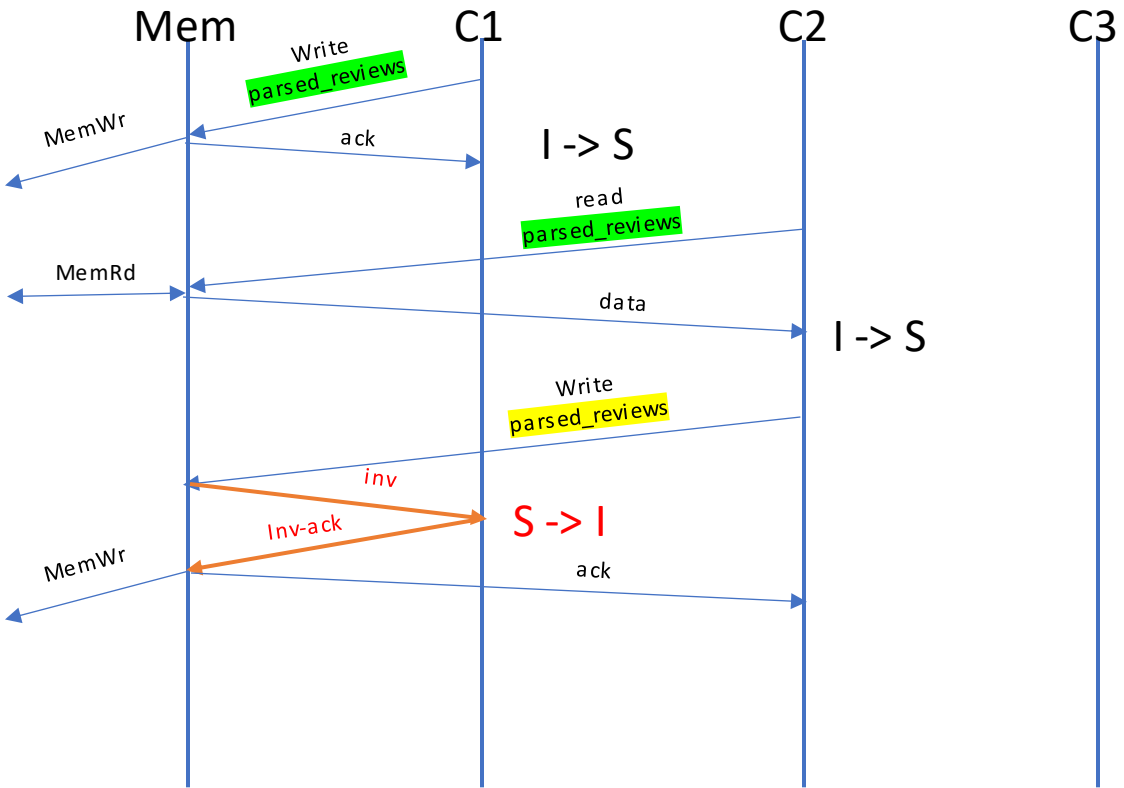
- The fault tolerance problem
  - Compute server failures – blocking
  - Network congestions – high tail latency

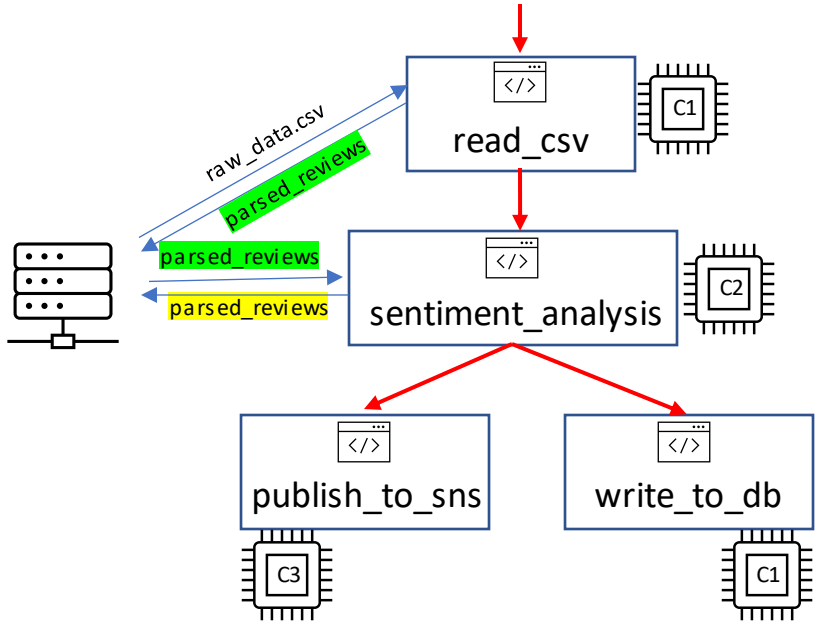




Key Problem – CXL not fault-tolerant!

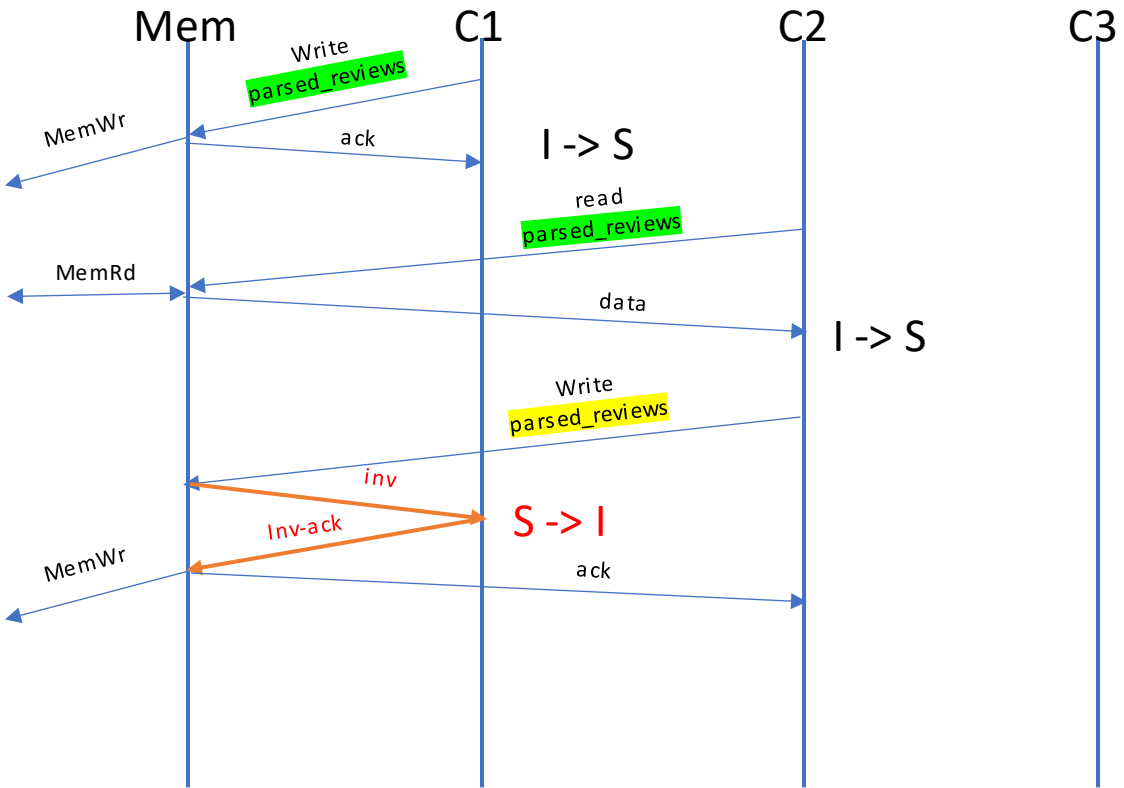
- Invalidation in critical path of write => Writes block when compute servers fail

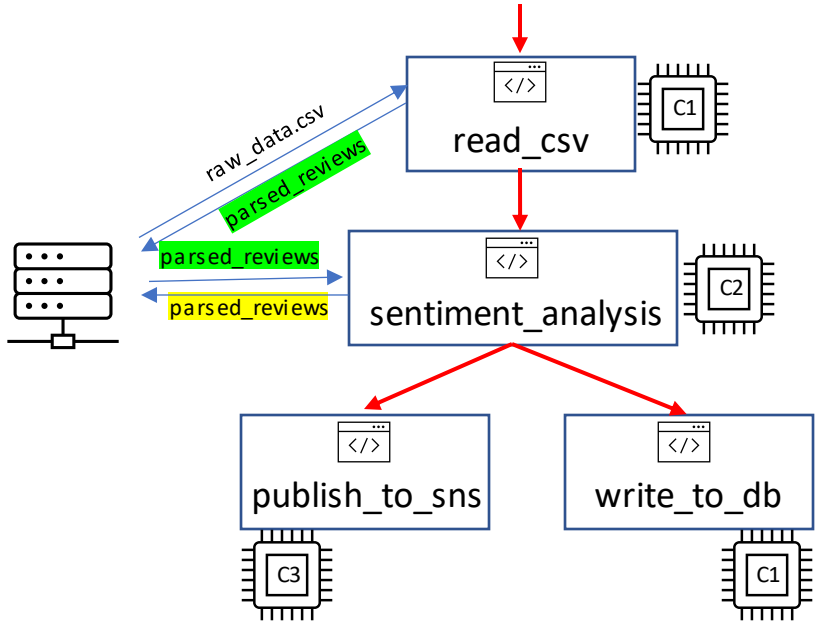




Key Problem – CXL not fault-tolerant!

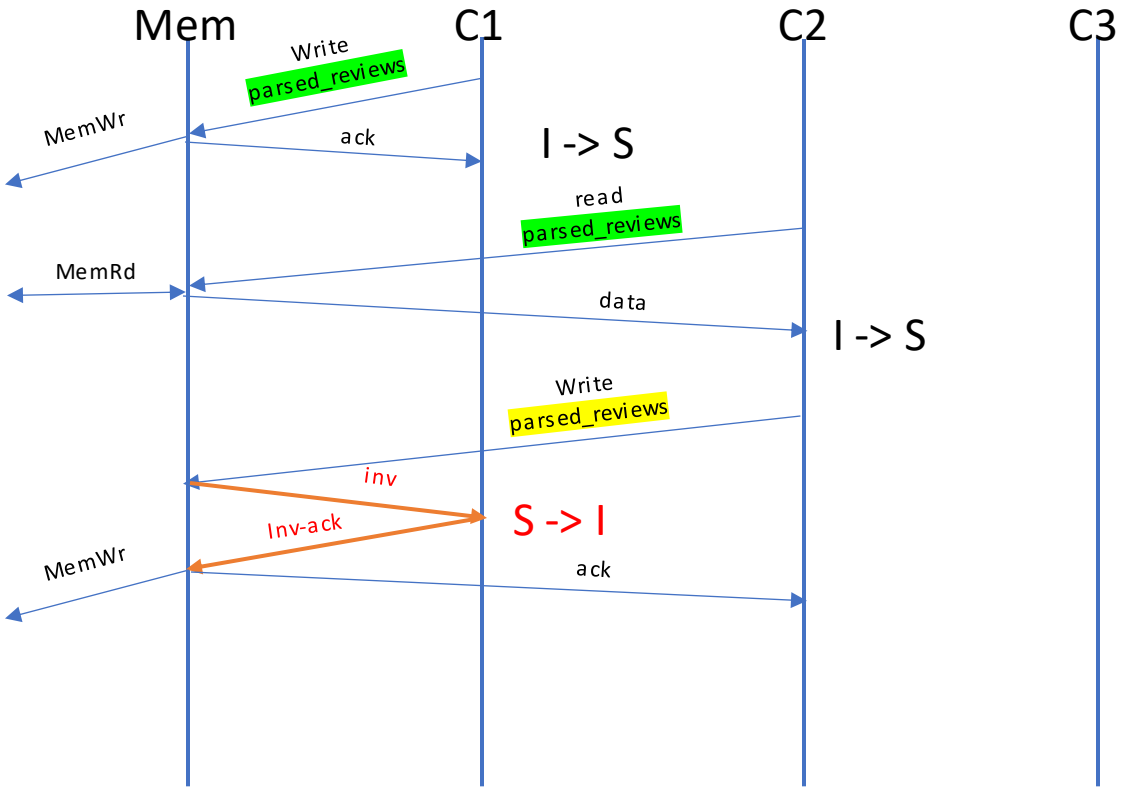
- Invalidation in critical path of write => Writes block when compute servers fail
- Insufficient RAS capabilities in CXL specification

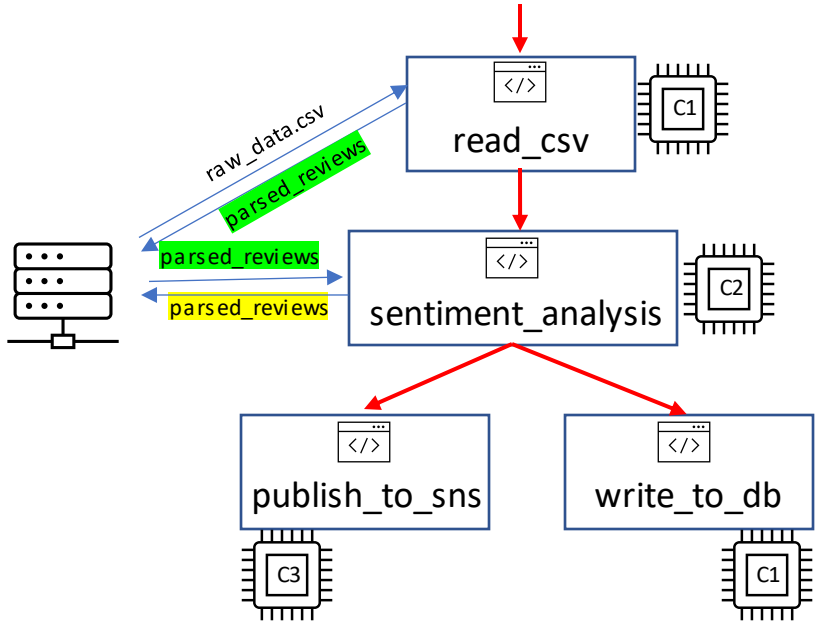




Key Problem – CXL not fault-tolerant!

- Invalidation in critical path of write => Writes block when compute servers fail
- Insufficient RAS capabilities in CXL specification
- FaaS embraces fault-tolerance => CXL must likewise





Āpta: Fault-tolerant Coherence Protocol

i. Lazy invalidation policy

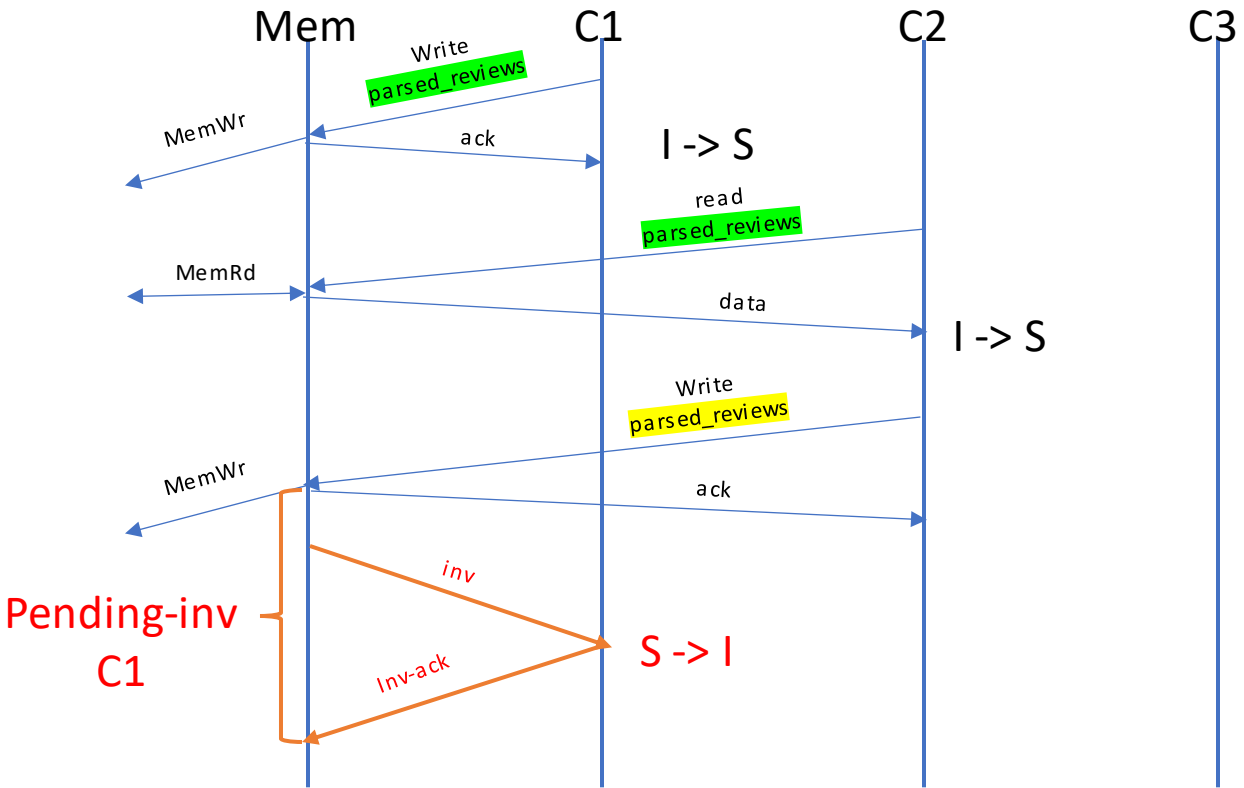
ii. Coherence-aware function scheduling

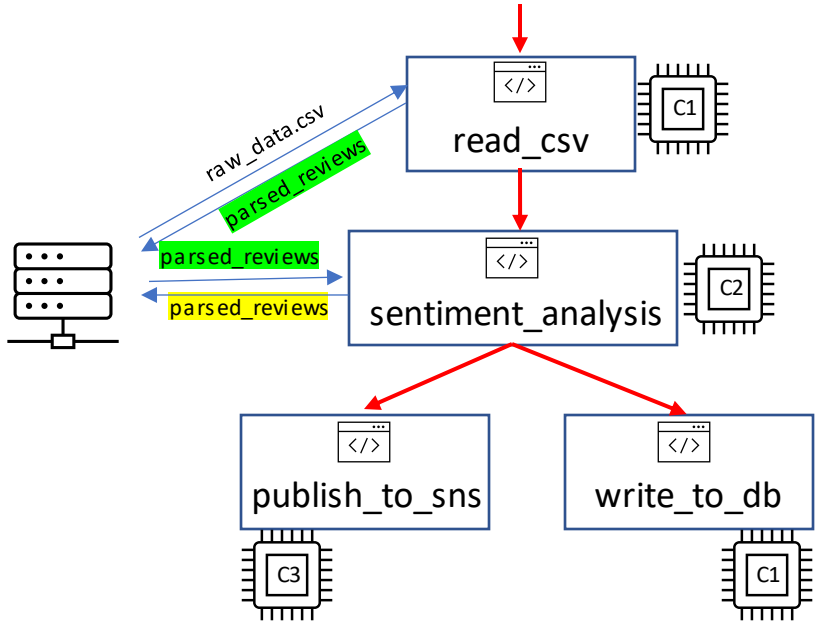
Āpta: Fault-tolerant Coherence Protocol

i. Lazy invalidation policy

Write is acknowledged immediately

Invalidation messages are sent asynchronously and tracked

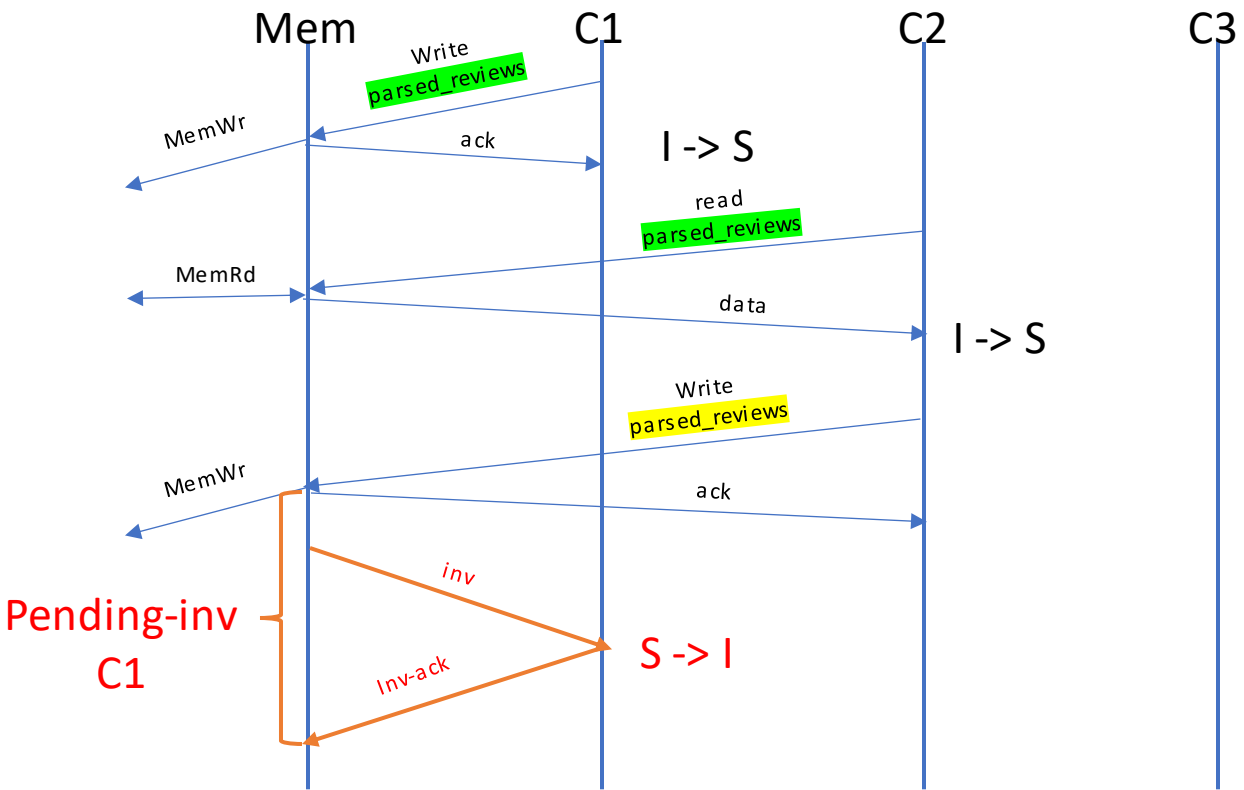


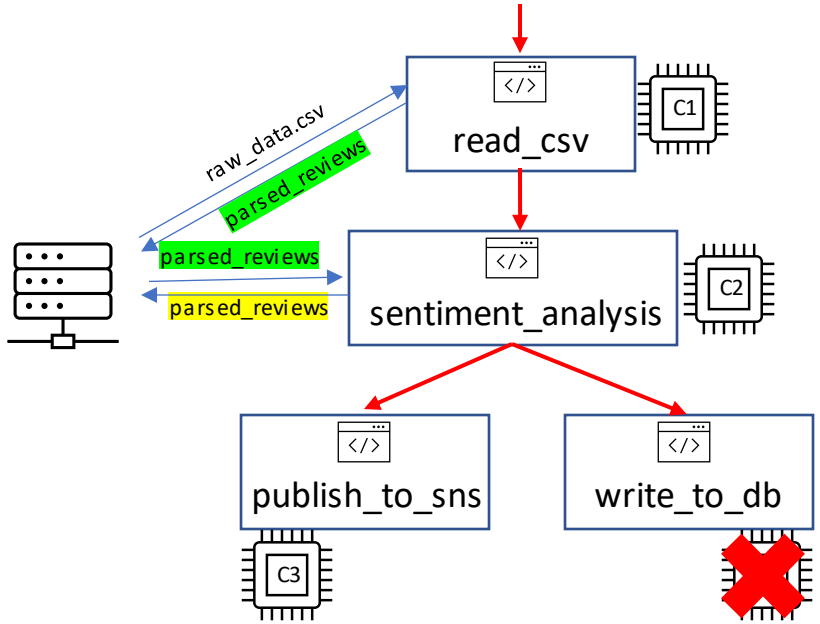


Āpta: Fault-tolerant Coherence Protocol

ii. Coherence-aware function scheduling

Never schedules function invocations on servers with pending invalidation-acknowlegements

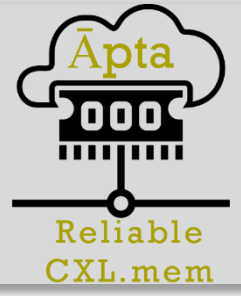




Āpta: Fault-tolerant Coherence Protocol

Strong consistency + Availability

- ✓ Lazy linearizability
  - Lazy invalidation protocol + Coherence-aware scheduling
- ✓ Fault-tolerant operation
  - Resilient to failure of compute server
- ✓ Provides line-rate coherence
  - Enables deployment on DPUs / SmartNIC / ToR switches



Āpta's design

a) CXL disaggregated memory-based object store
   - Extended shmem IPC
   - Defines a caching policy
   - Locality-aware scheduling

b) Fault-tolerant coherence protocol
   - Tailored coherence protocol
   - Lazy Invalidation of sharers and coherence-aware scheduling

This talk

c) Object-granular disaggregated memory
   - Bulk cache-line loads
   - Transactional atomic durability

Āpta
Reliable
CXL.mem

Realizing Āpta's design

Object-granular CXL disaggregated memory

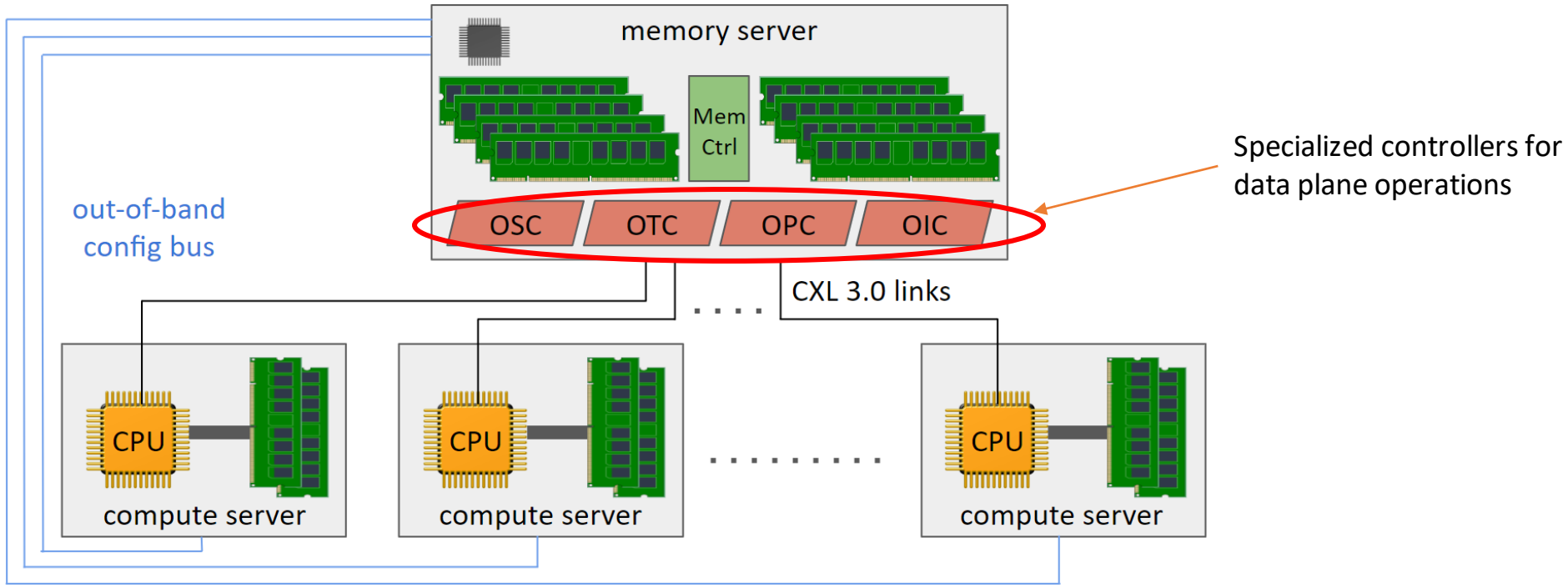
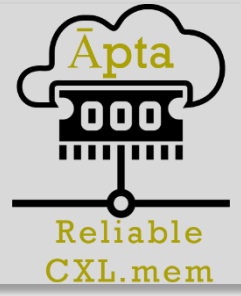


Fig: Āpta system schematic



Realizing Āpta's design

(b) Fault-tolerant coherence

OTC

Object Tracking Controller

Directory for the coherence protocol

OIC

Object Invalidation Controller

Reverse address translation + tracks invalidation-acks

(c) Object-granular disaggregated mem

OSC

Object Serving Controller

Address translation + bulk cache line response

OPC

Object Persistence Controller

Persists entire object atomically using one-phase commit

Performance Evaluation

Custom trace-driven gem5 simulation

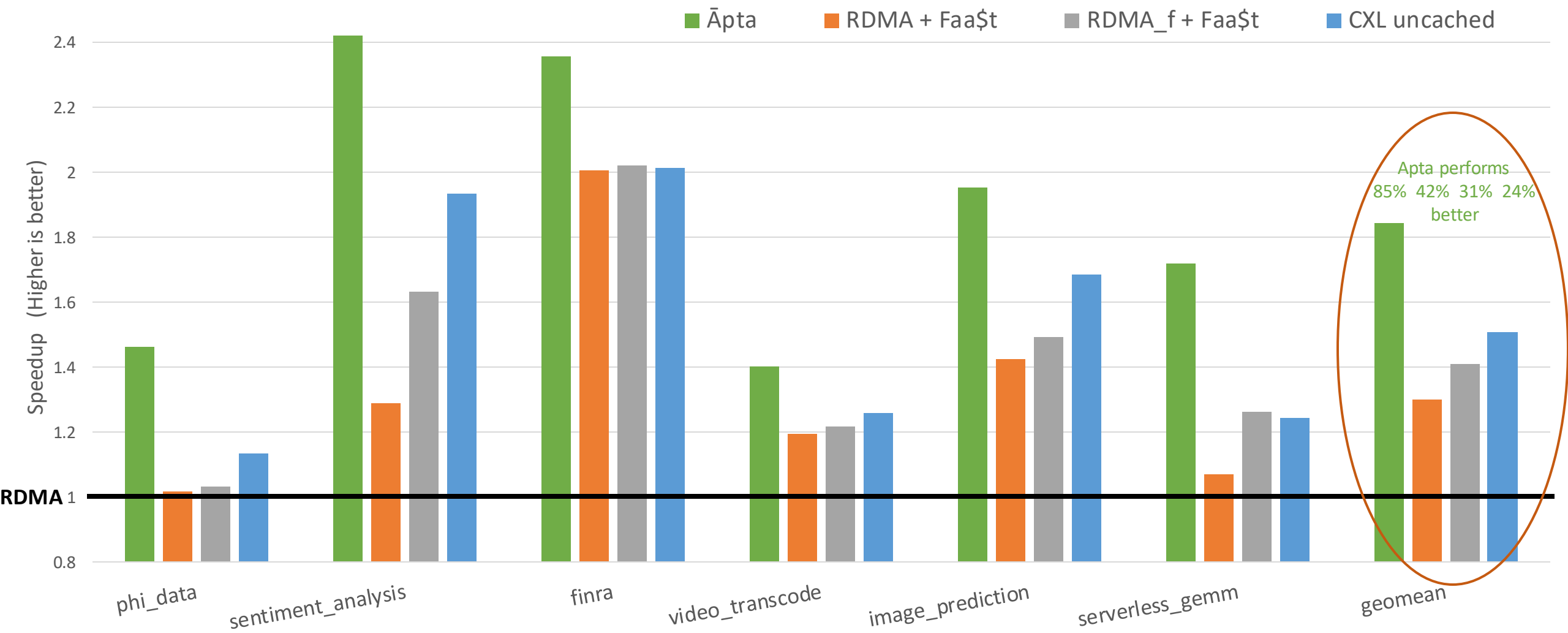
- **Prism traces** [†] annotated with phase of execution
- 3 compute servers, 1 disaggregated memory server
- Compute server: single socket 3GHz, per-core L1, shared L2, 2 x 8GB DDR4
- Memory server: 2 x 8GB DDR4, modelled controllers (OTC, OIC, OPC, OSC)
- Coherence Protocol: MOESI (intra-server), Āpta (inter-server)
- Interconnect: point-to-point (500ns, 80bps), full-duplex

Benchmarks

- **Full FaaS applications** - 6 workflows, 27 functions
- Different domains, communication patterns, realistic scheduling decisions
- Applications from AWS use cases and serverless frameworks (numpywren, THIS)
  PHI data, Sentiment analysis, FINRA, Video transcode, Image prediction, Serverless GEMM

[†] SynchroTrace, ISPASS ‘15

Performance Evaluation



Āpta Summary

Accelerating function-as-a-service

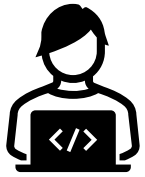
40% – 142% over RDMA
21% – 90% over RDMA + caching
15% – 42% over un-cached CXL

Fault-tolerant coherence protocol

Protocol verified in Murφ model checker
32% lower standard deviation of exec time

Object-granular Disaggregated Memory

Shared memory IPC
Bulk cache-line loads
Transaction atomic durability



Artifacts available

https://github.com/adarshpatil/apta
https://adar.sh/apta

#OpenToWork @adarshpatil