Black-box Concurrent Data Structures for NUMA Architectures

Irina Calciu (VRG)

Siddhartha Sen (MSR)

Mahesh Balakrishnan (Yale)

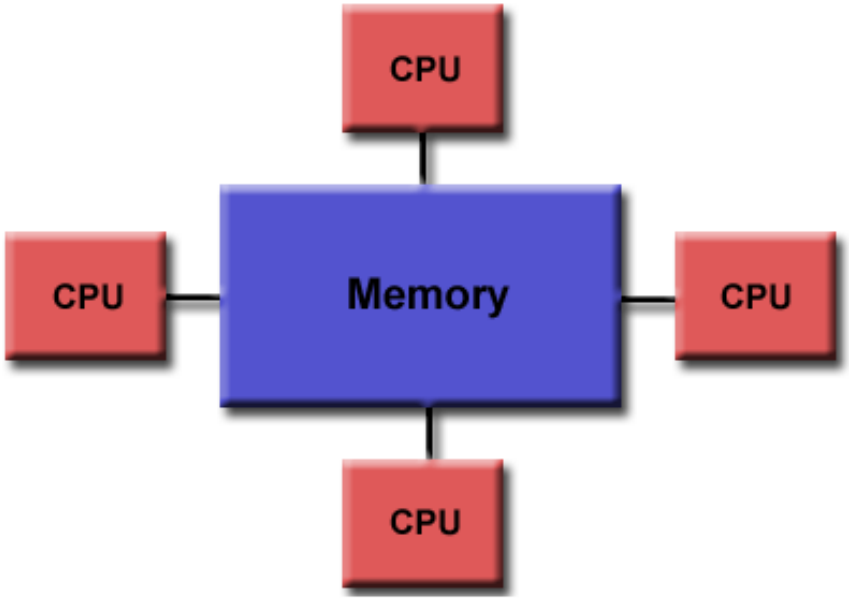
Marcos K. Aguilera (VRG)

Concurrent Data Structures (CDS)

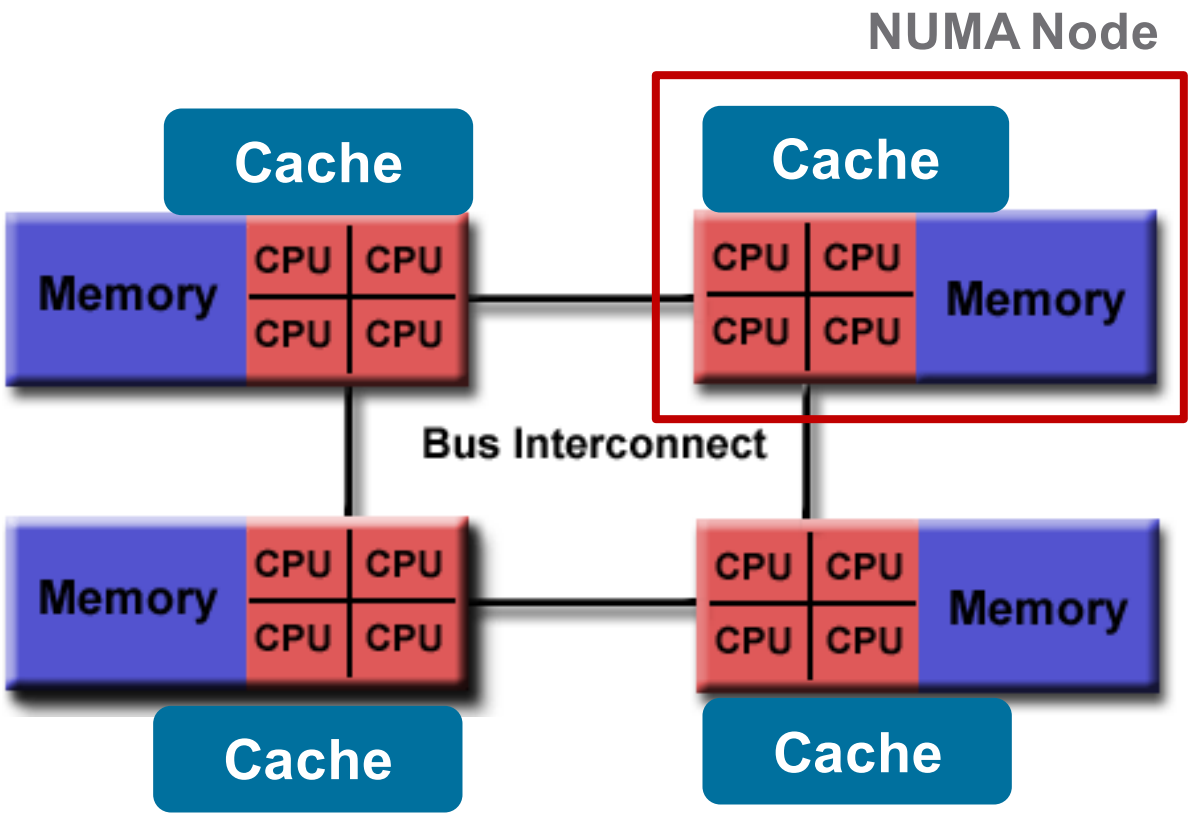
Used everywhere: kernel, libraries, applications

Issues:

- Difficult to design and implement
- Complex and rigid
- Assume uniform memory



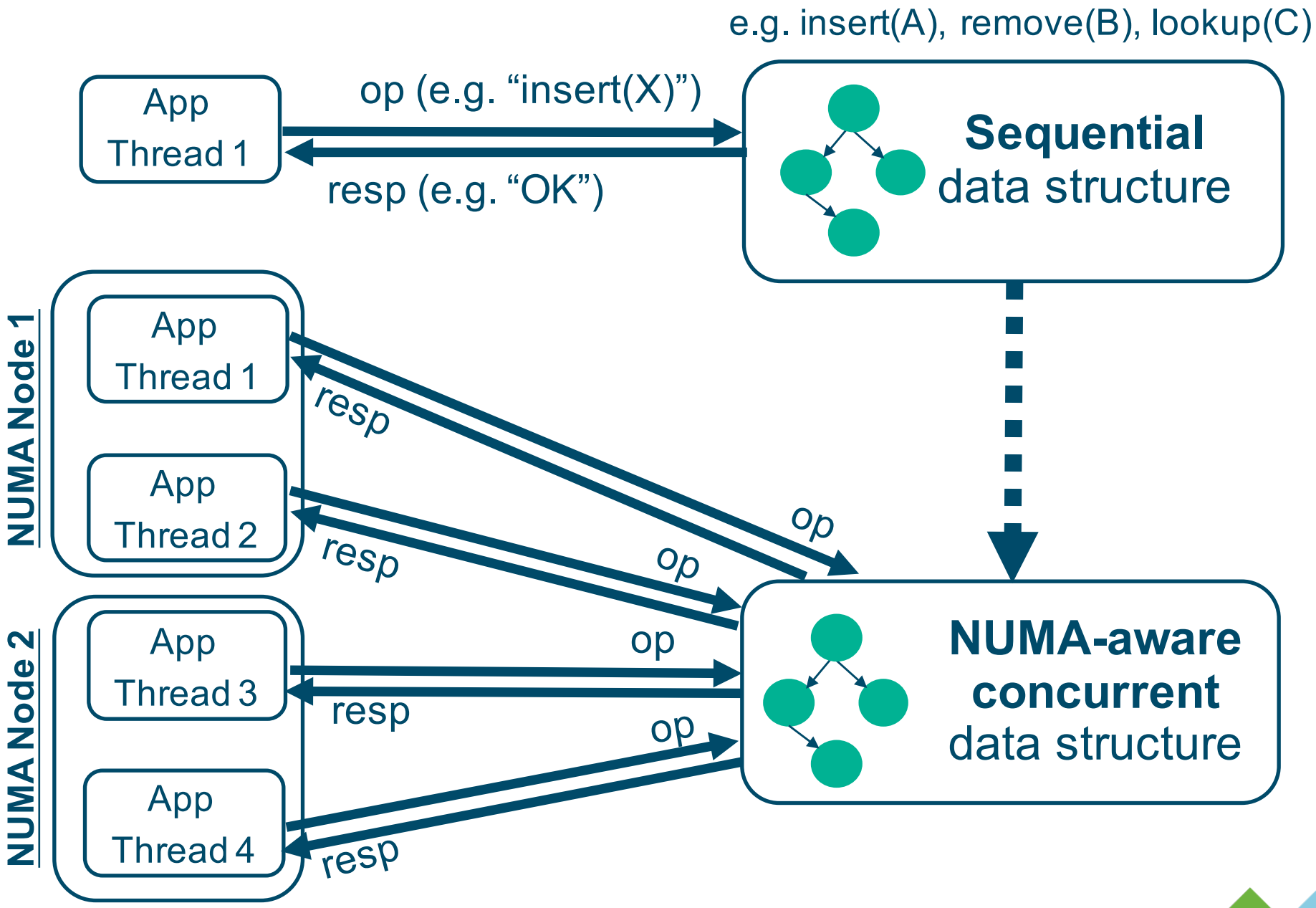
Non-Uniform Memory Access (NUMA)



Goals

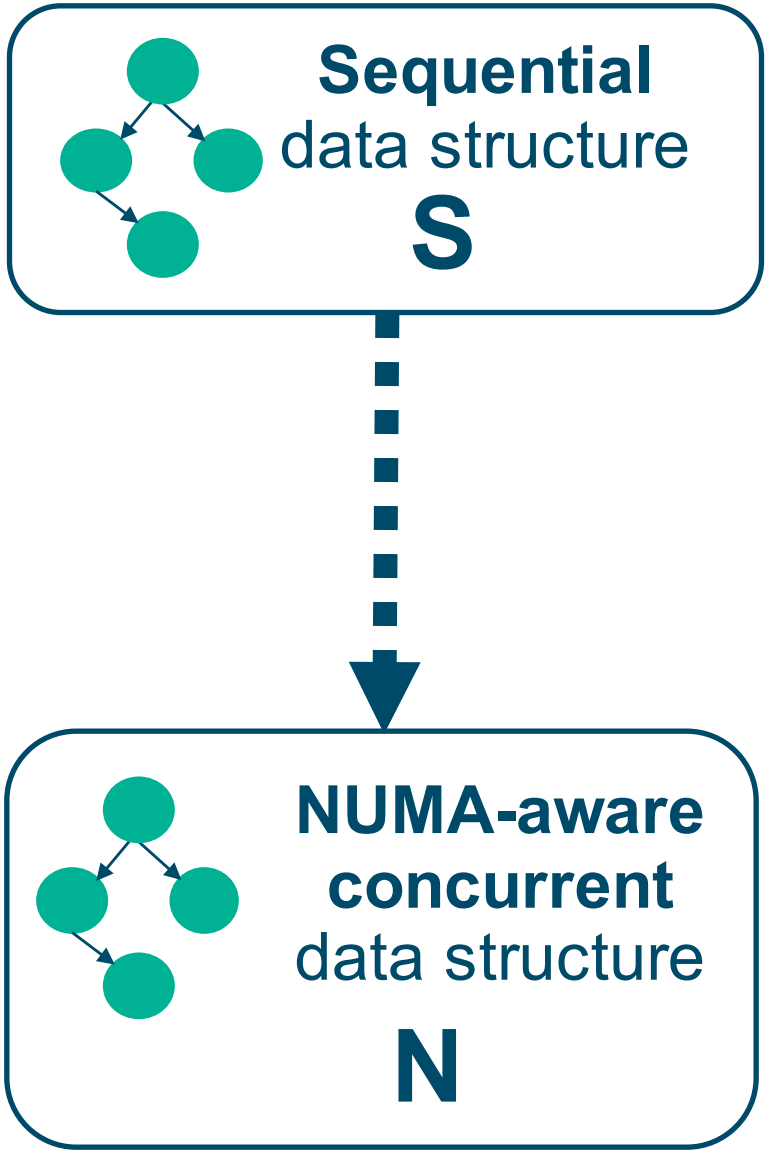
- Design efficient NUMA-aware CDS
- Black-box method: works for any data structure
- Application level

Transform sequential DS to NUMA-aware CDS



API

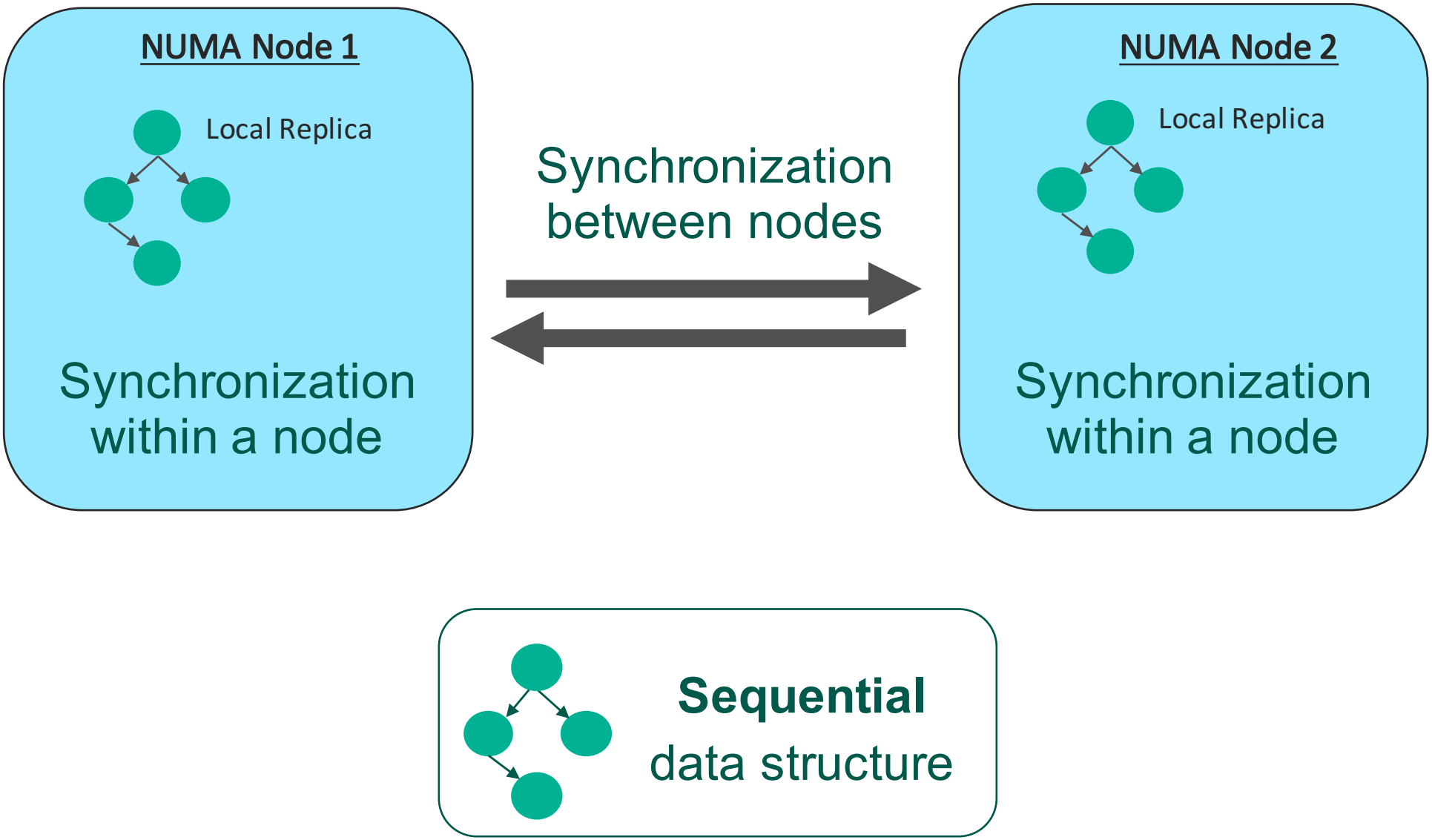
S.ExecuteSeq(op, args) -> Result



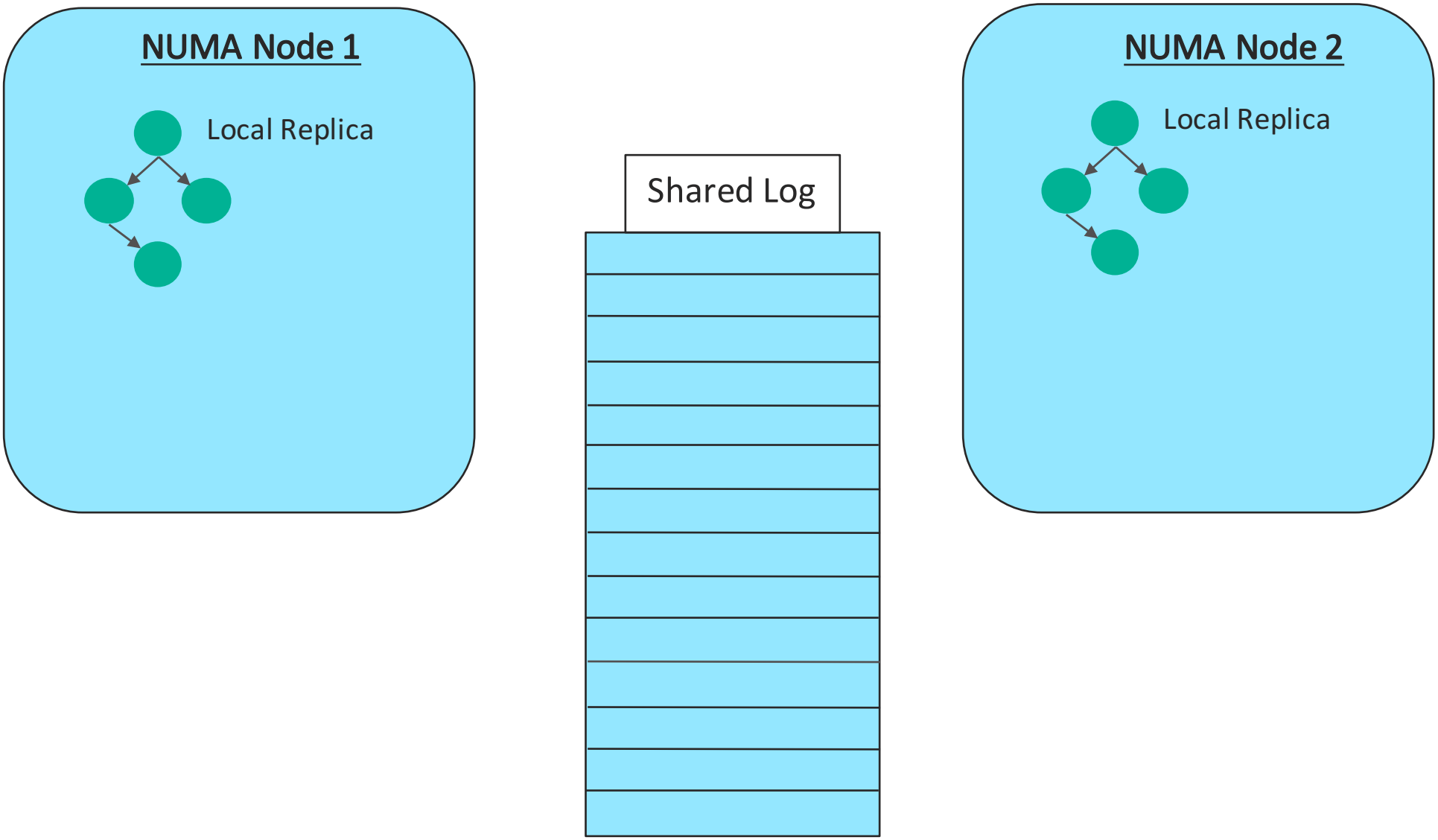
N.Execute(op, args) -> Result

N.IsReadOnly(op) -> Boolean

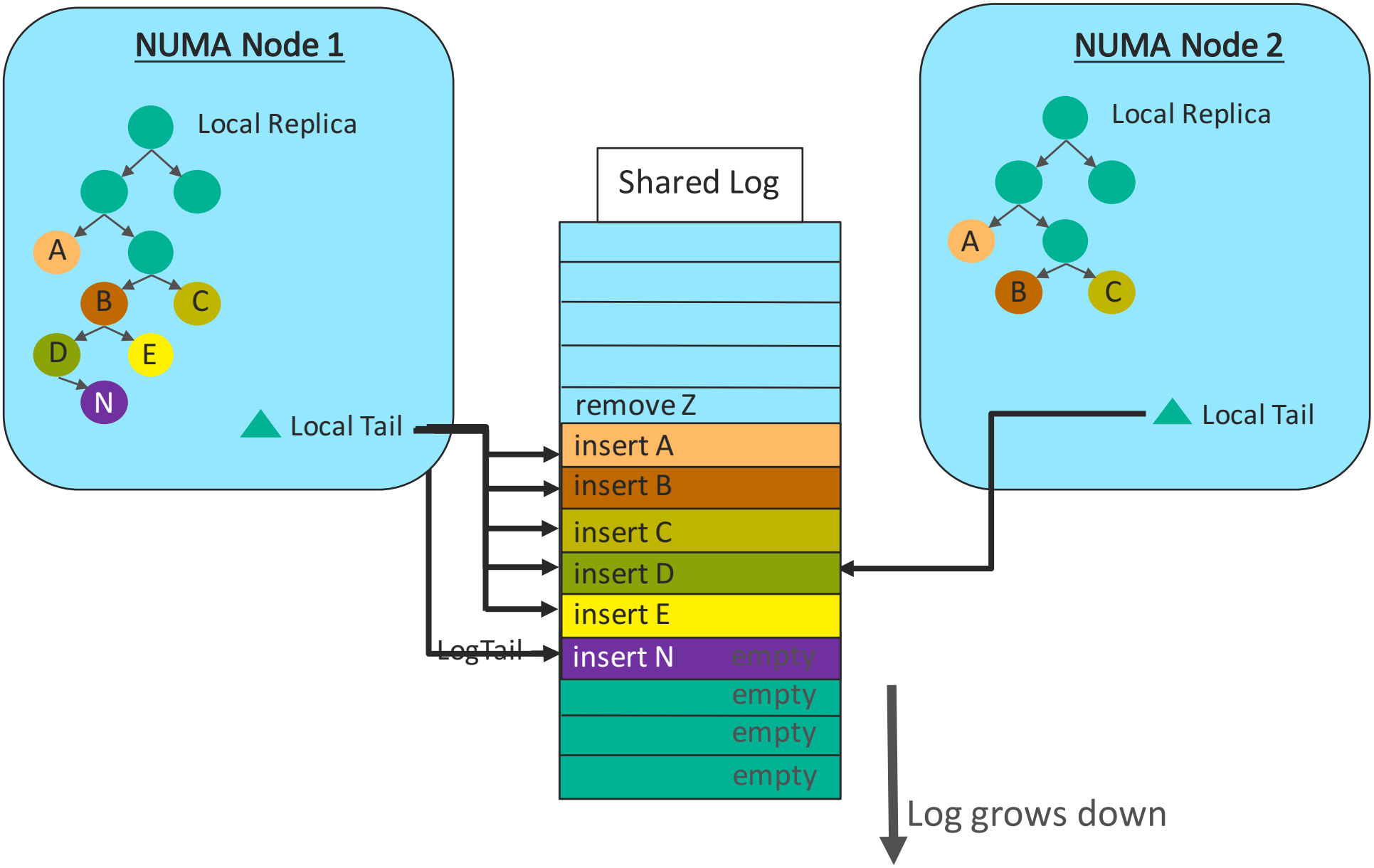
Our Method: Node Replication (NR)



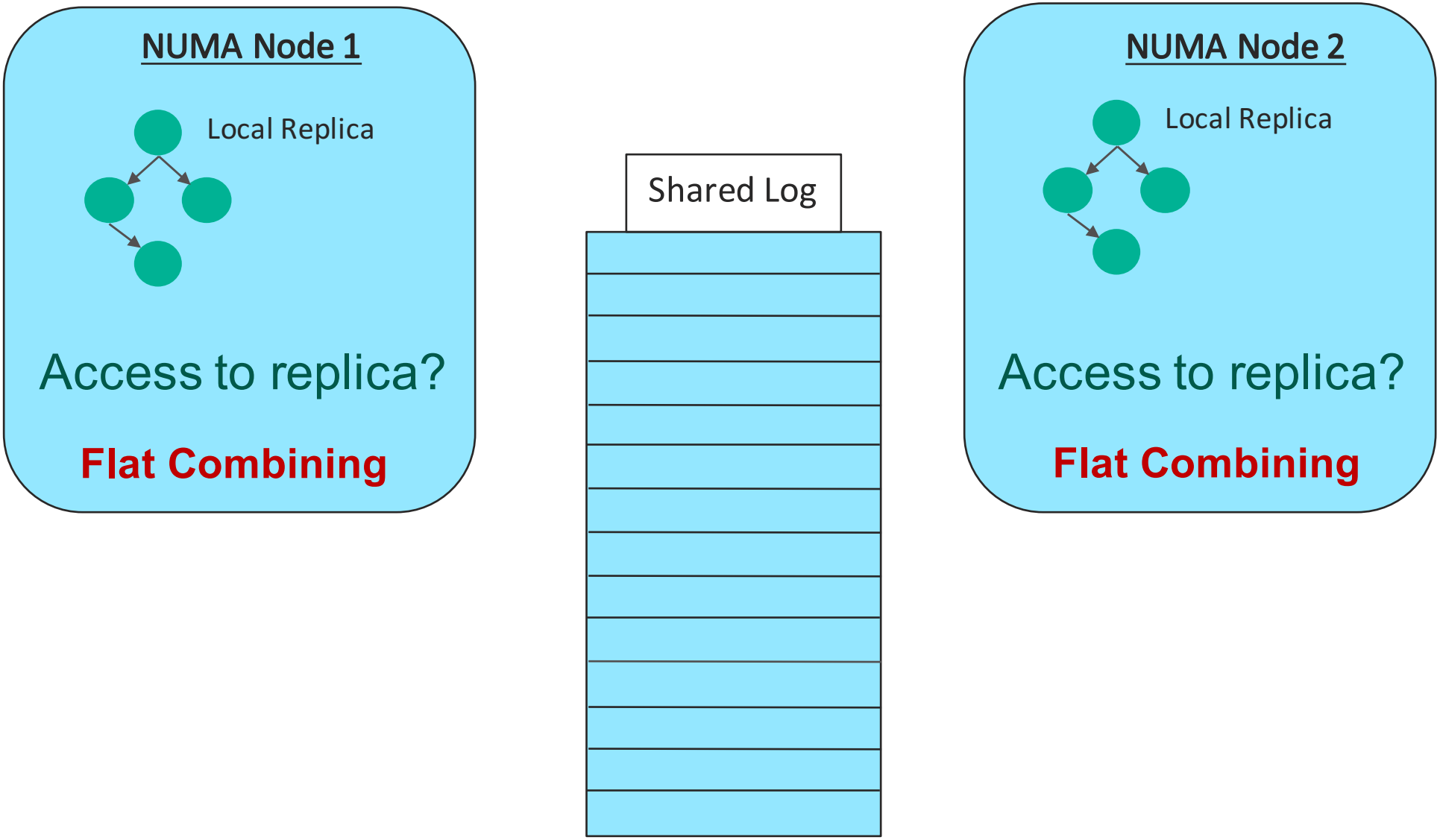
Synchronizing Replicas (Cross-node)



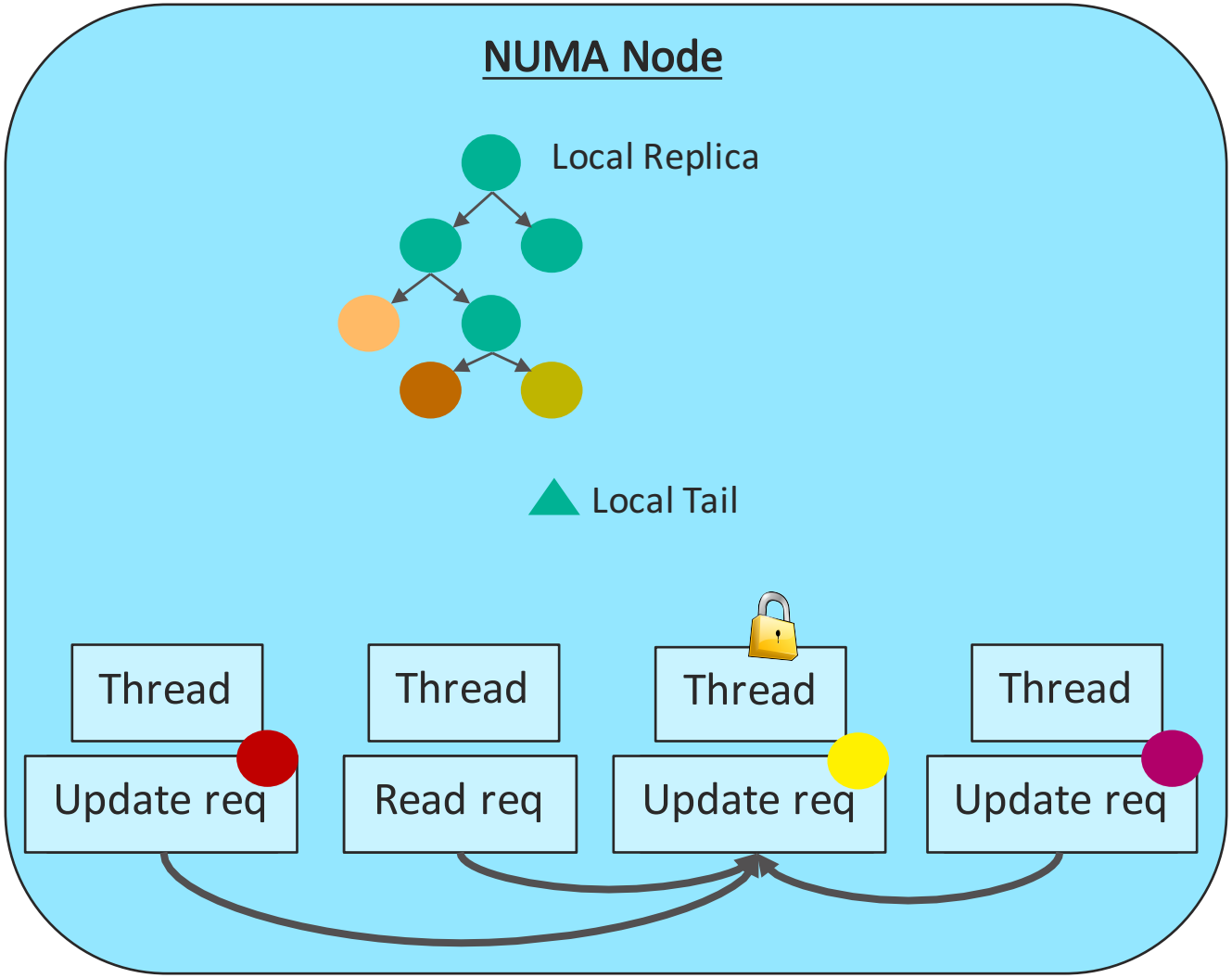
Synchronizing Replicas (Cross-node)



Synchronizing Each Replica (Intra-node)

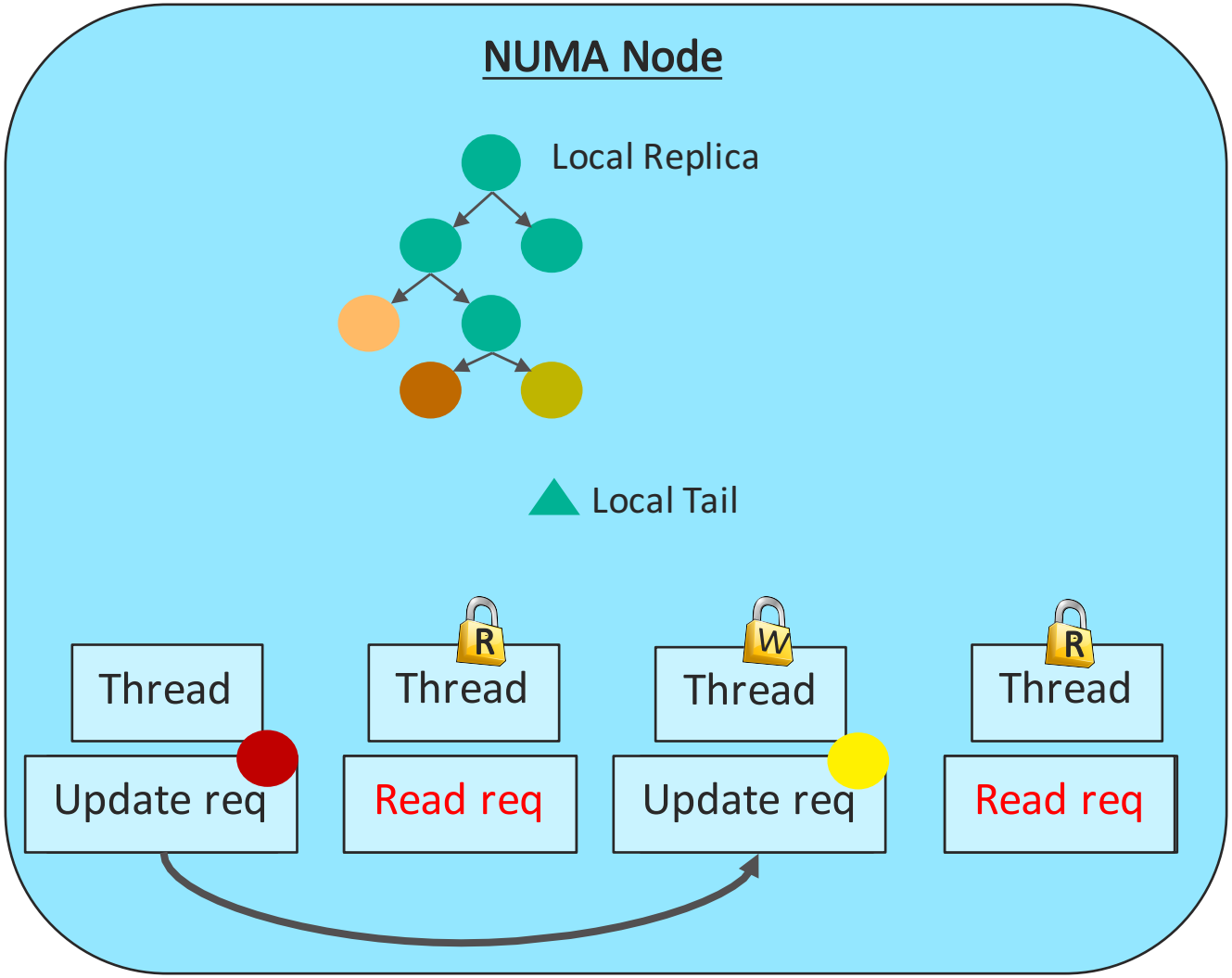


Access to Replica



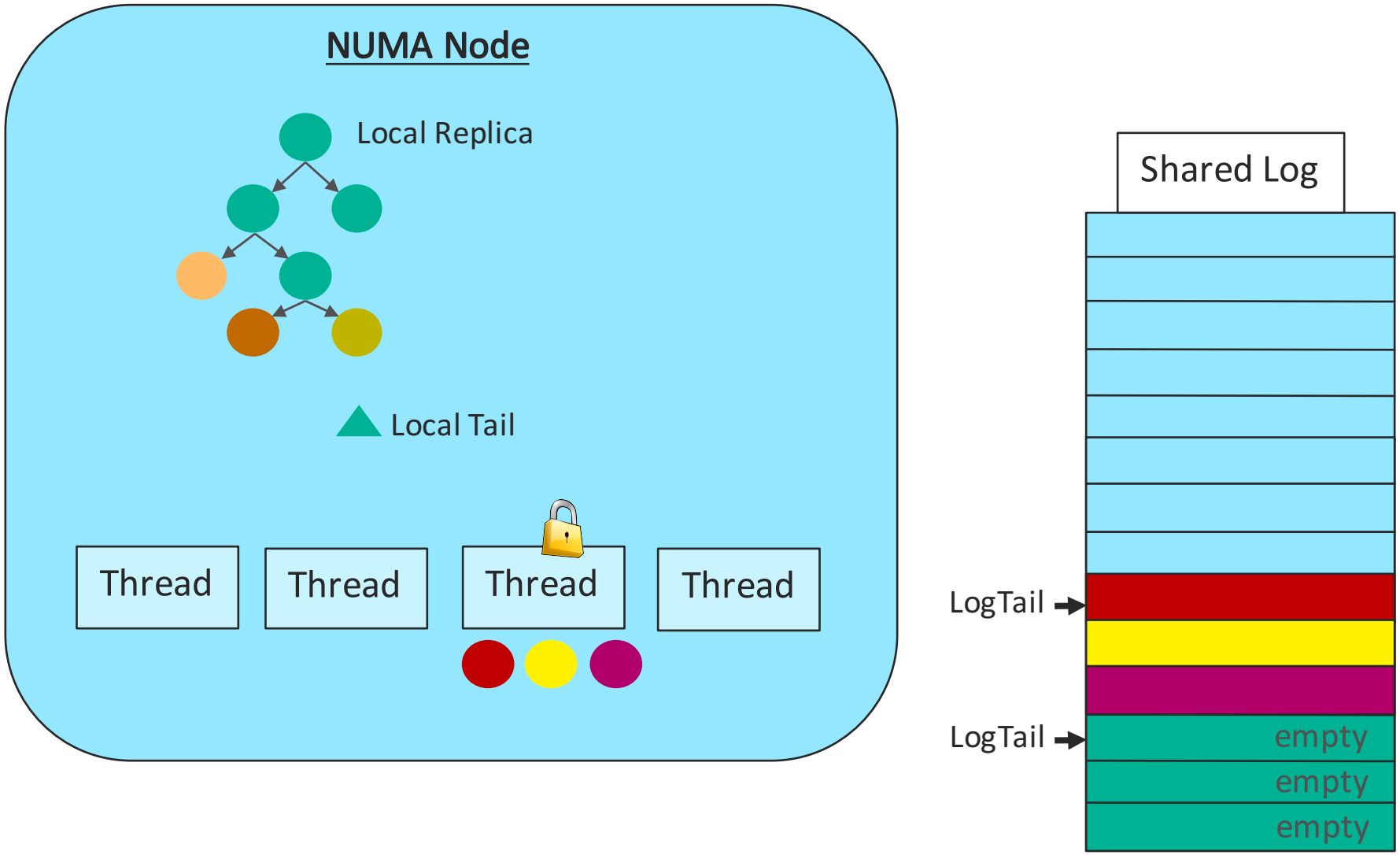
Flat Combining: [Hendler et al., 2010]

Access to Replica

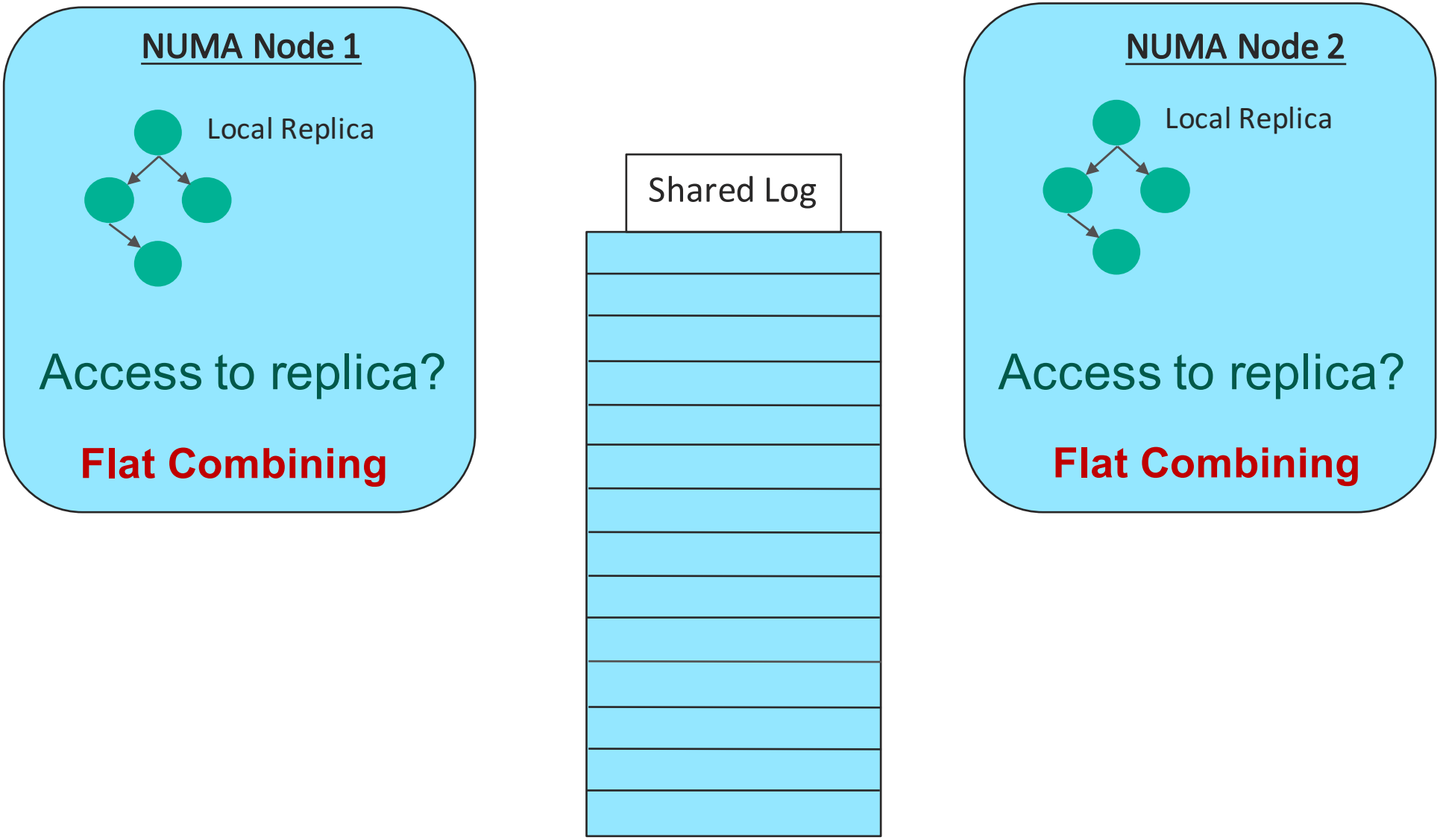


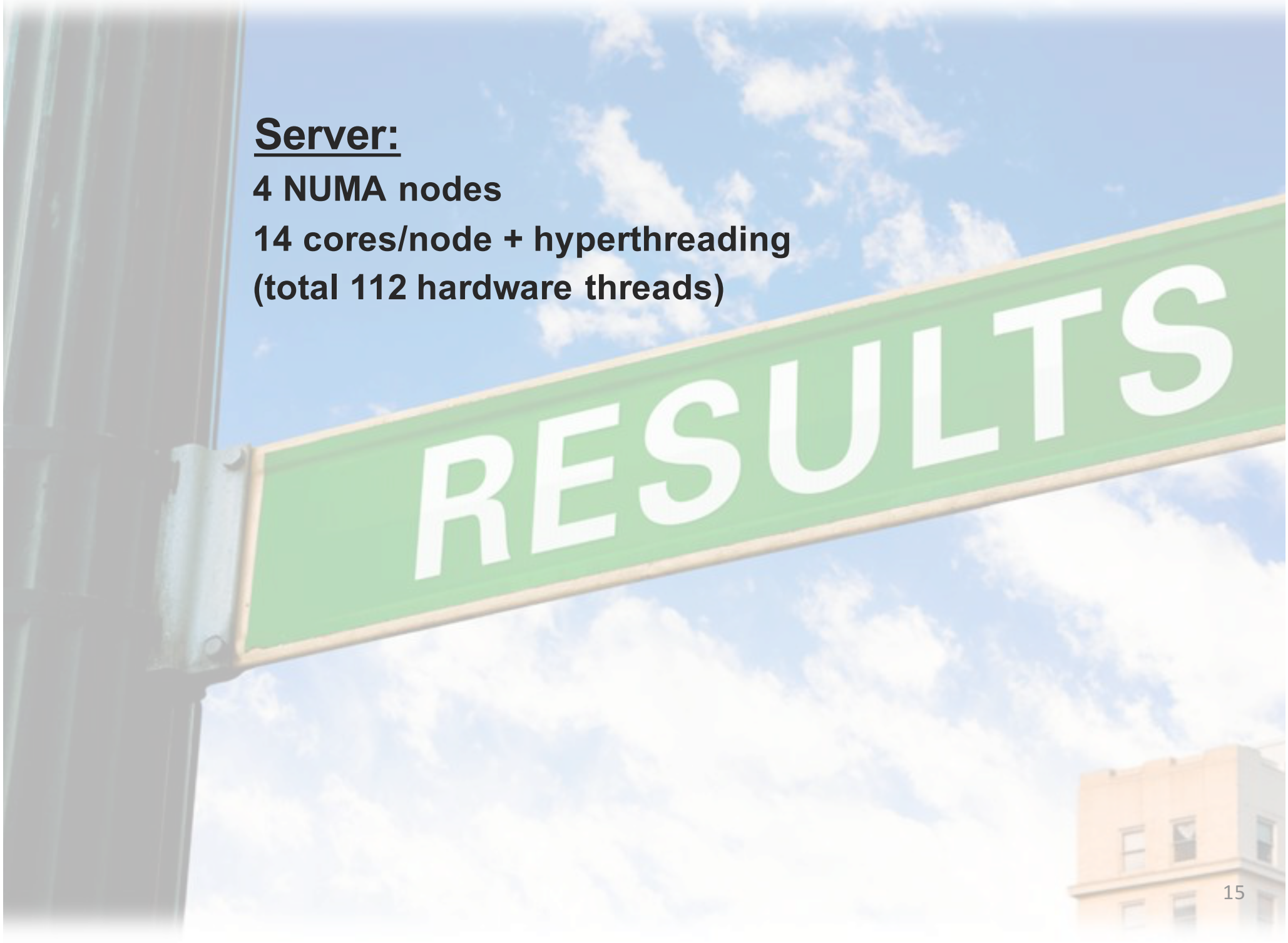
Flat Combining: [Hendler et al., 2010]

Putting It All Together



Algorithm Summary: Replication, Log, Combining





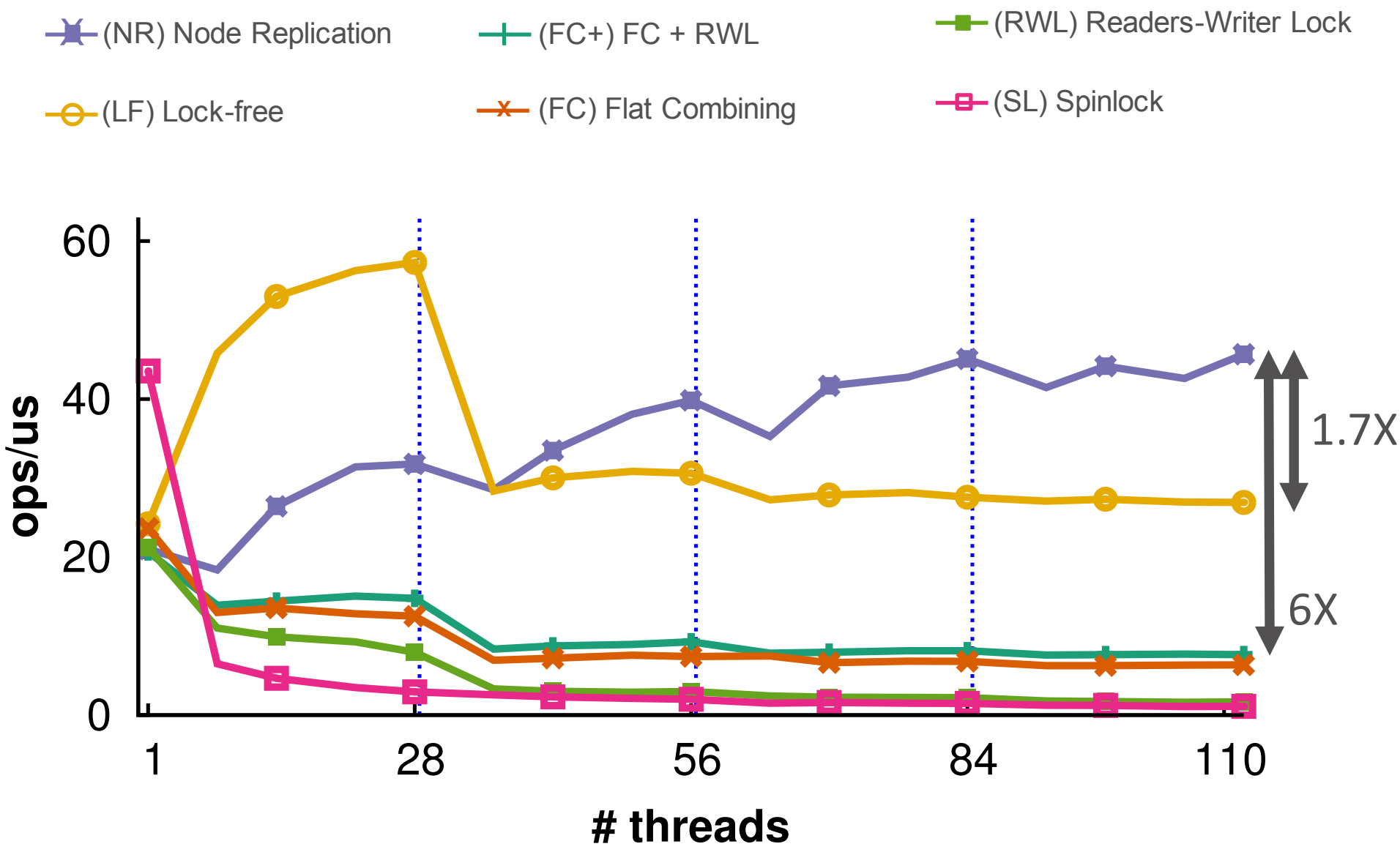
Server:

4 NUMA nodes

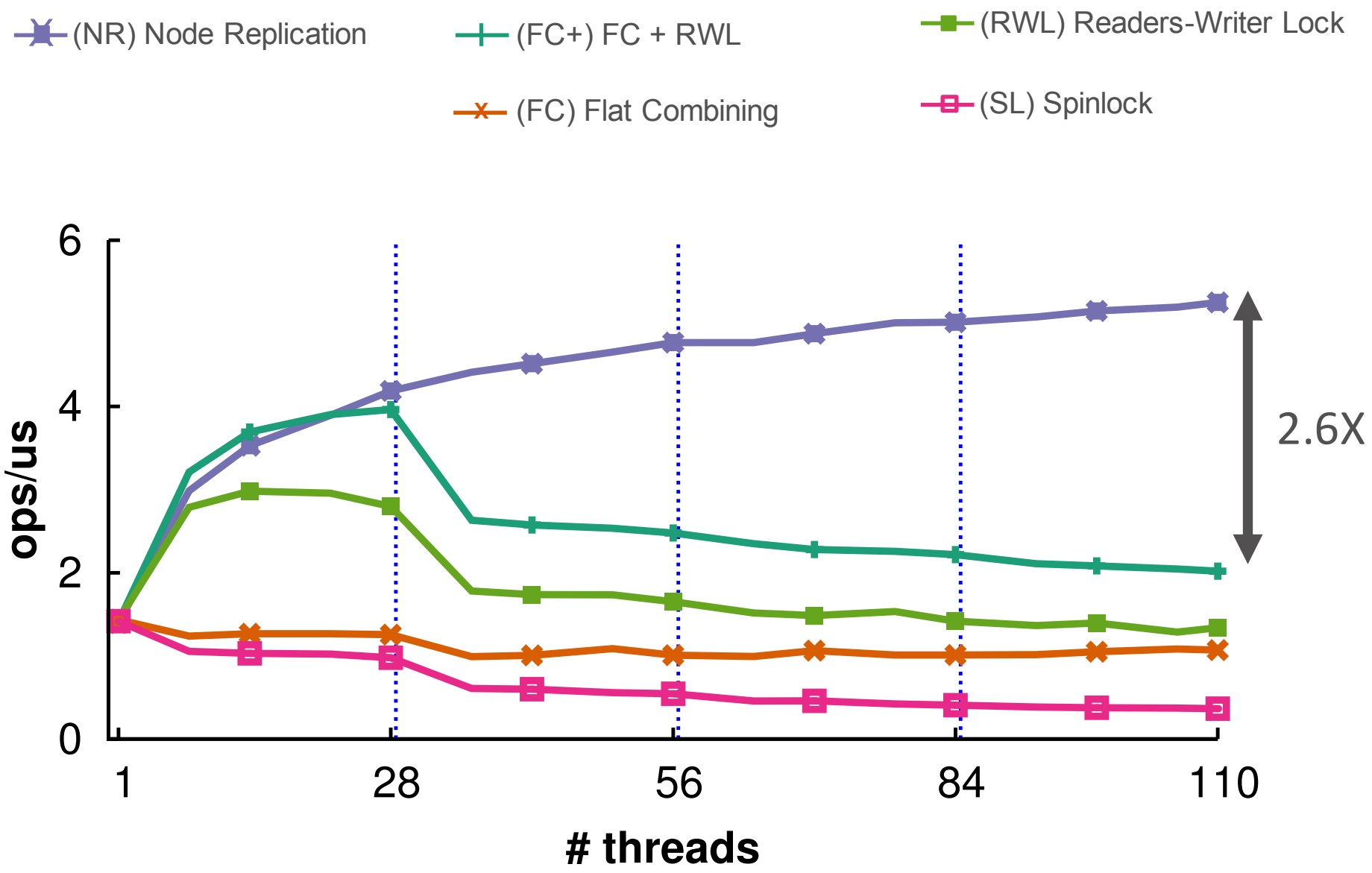
14 cores/node + hyperthreading

(total 112 hardware threads)

Skiplist Priority Queue – 10% Updates



Using Replication in REDIS: 10% Updates



Rationale

- Trade memory + computation for less communication
  - Compact representation of operations
  - Limited cross-node synchronization and contention
- Enable parallelism
  - Combiners across nodes
  - Readers within a node
  - Readers and the combiner on the same node
- Leverage batching and reordering

Conclusion: NodeReplication Works Well

- Black-box: works for any data structure
- Good for small and medium size CDS
- Beneficial for contended CDS

Thank you!

icalciu@vmware.com

https://research.vmware.com/