INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING

ICPP/2021/CHICAGO/USA

acm In-Cooperation

sighpc

AUGUST 9-12, 2021

A Universal Construction to implement Concurrent Data Structure for NUMA-multicore

Zhengming Yi

National University of Defense Technology

Changsha, China

yizhengming17@nudt.edu.cn

Concurrent Data Structures (CDS)

Used everywhere: kernel, libraries, applications

Issues:

- High copying overheads
- NUMA-oblivious design
- Read-side overhead
- Complex

Goals

- Design a new Universal Construction (called CR),which transforms a sequential implementation of a data structure into a concurrent implementation

➢ Provide efficient read-side performance

➢ Provide scalable write-side performance on NUMA-multicore

Our method: CR

- CR relies n+1 replicas of the data structure where n is the number of NUMA nodes, reader-writer lock, delegation and a shared log.

Our method: CR

- CR relies n+1 replicas of the data structure where n is the number of NUMA nodes, reader-writer lock, delegation and a shared log.
  - Efficient Read: Keep one up-to-date replica for read-only access at all times

Our method: CR

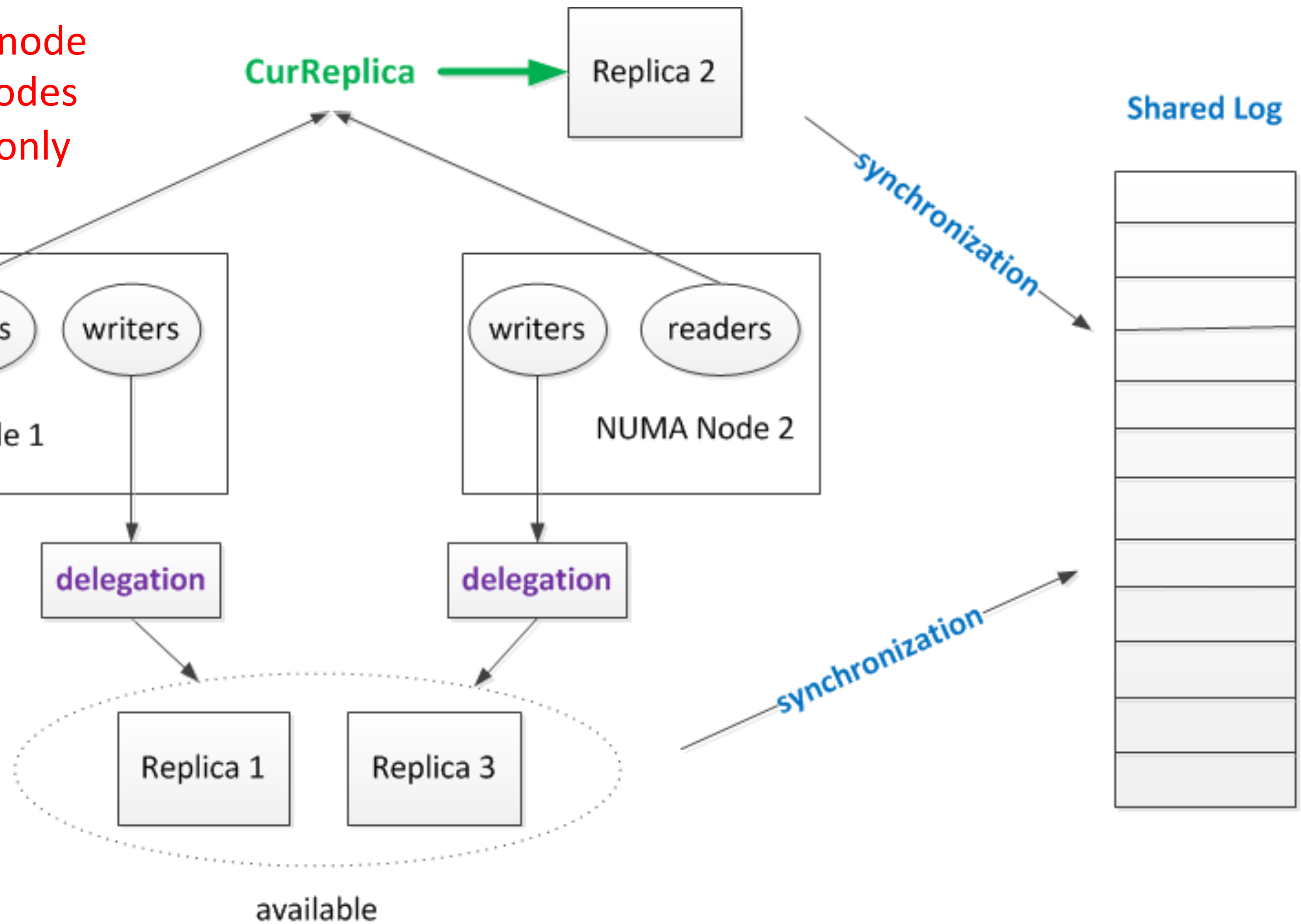
- CR relies n+1 replicas of the data structure where n is the number of NUMA nodes, reader-writer lock, delegation and a shared log.
  - Efficient Read: Keep one up-to-date replica for read-only access at all times
  - NUMA-aware write: Use a shared log to synchronize cross-nodes threads and use delegation to synchronize local threads

Our method: CR

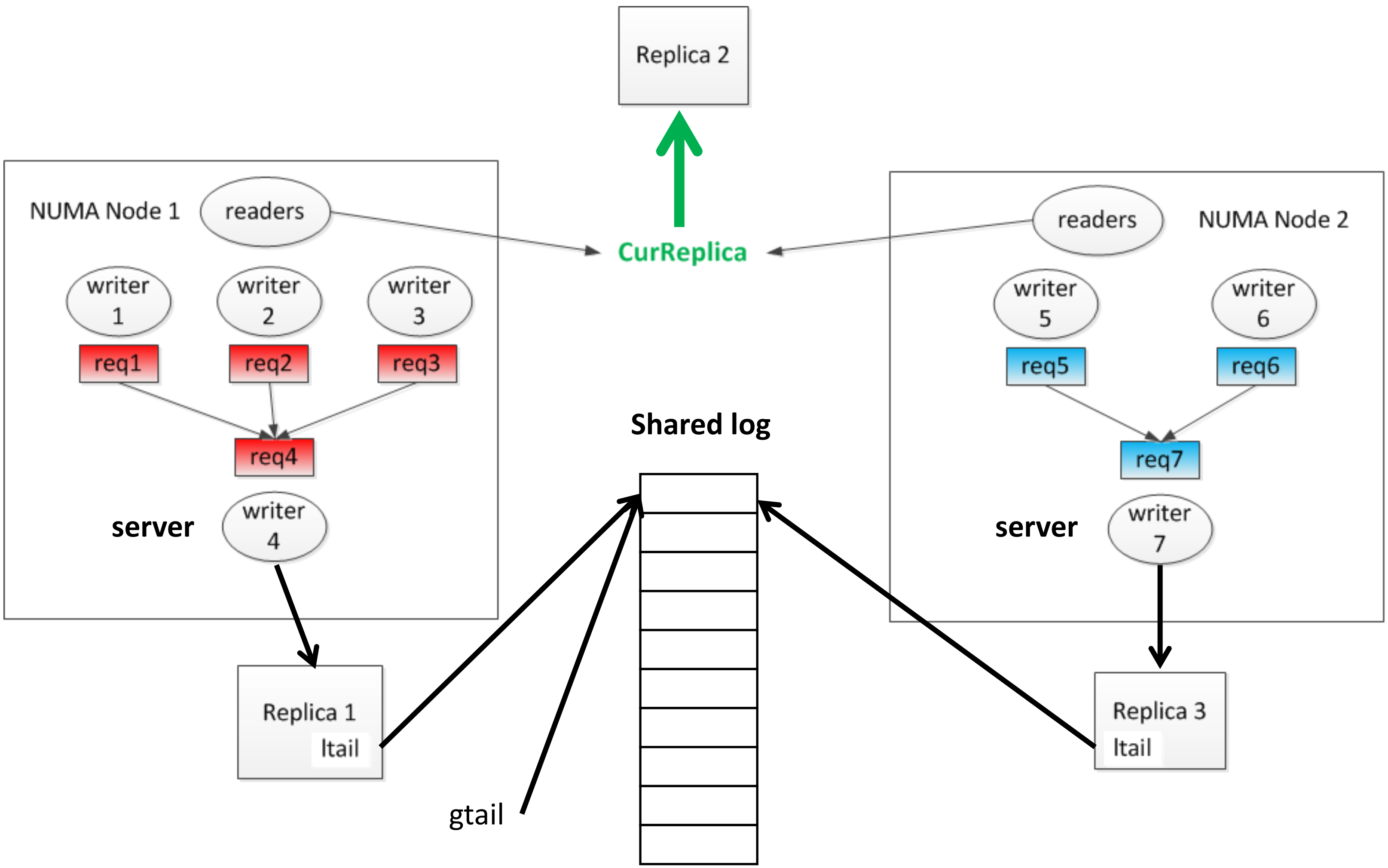
- CR relies n+1 replicas of the data structure where n is the number of NUMA nodes, reader-writer lock, delegation and a shared log.
  - Efficient Read: Keep one up-to-date replica for read-only access at all times
  - NUMA-aware write: Use a shared log to synchronize cross-nodes threads and use delegation to synchronize local threads
  - Without requiring inner knowledge of the data structure

Structure Chart for CR

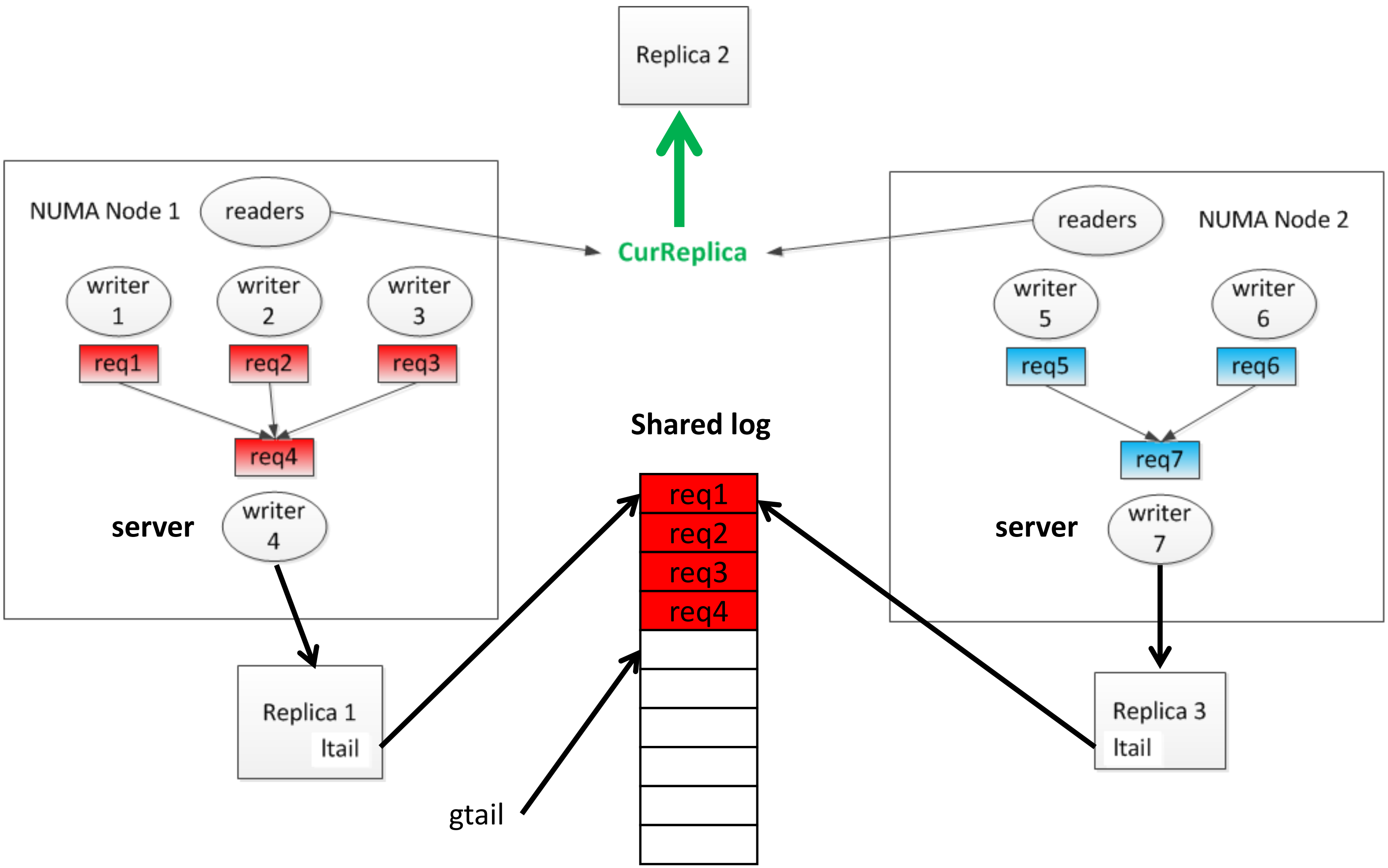
1 Delegation within a node
2 Shared log across nodes
3 CurReplica for read-only



Delegation : Collect requests within a NUMA node 1



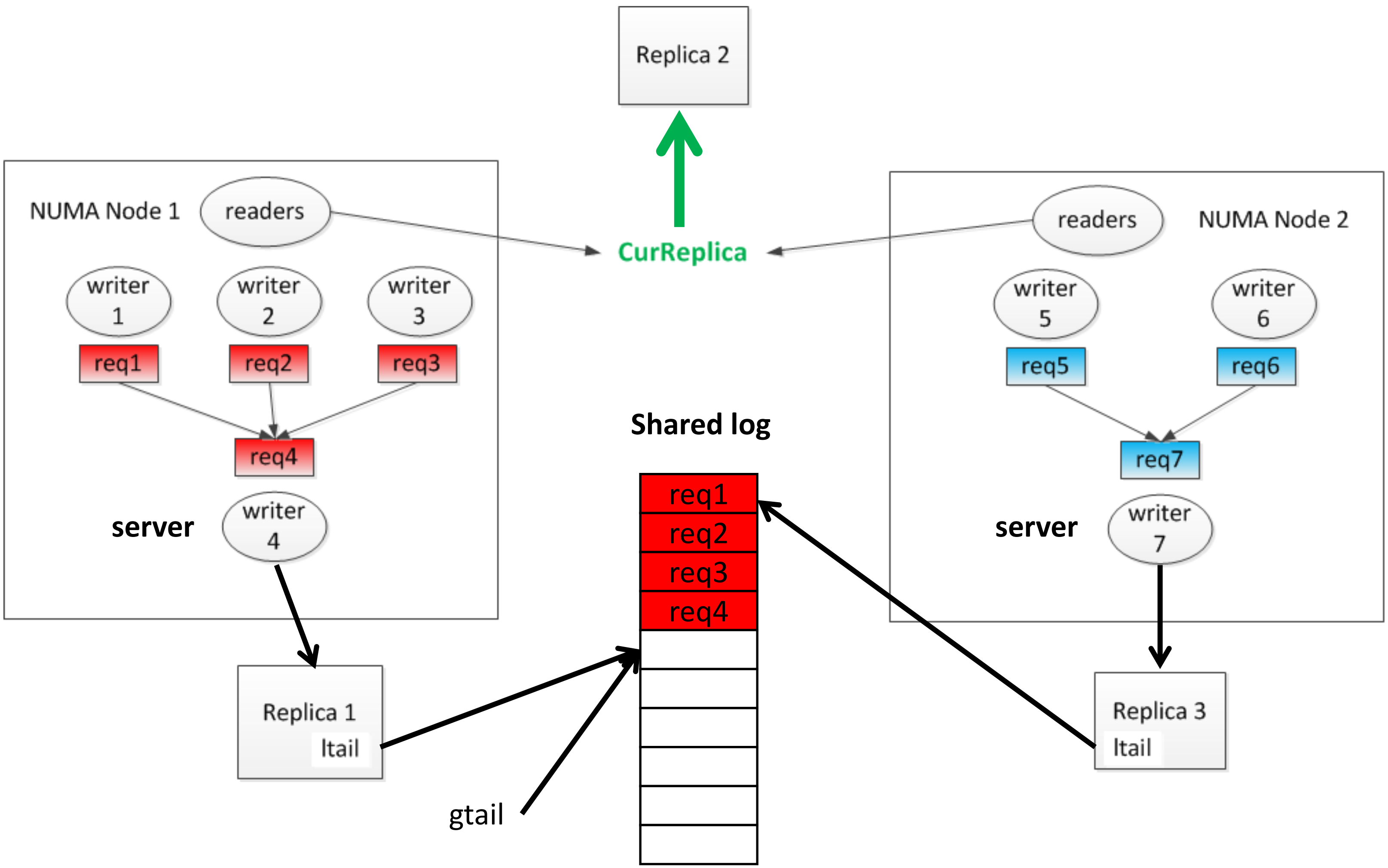
Delegation : Write requests within a NUMA node 1



INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING

acm In-Cooperation sighpc

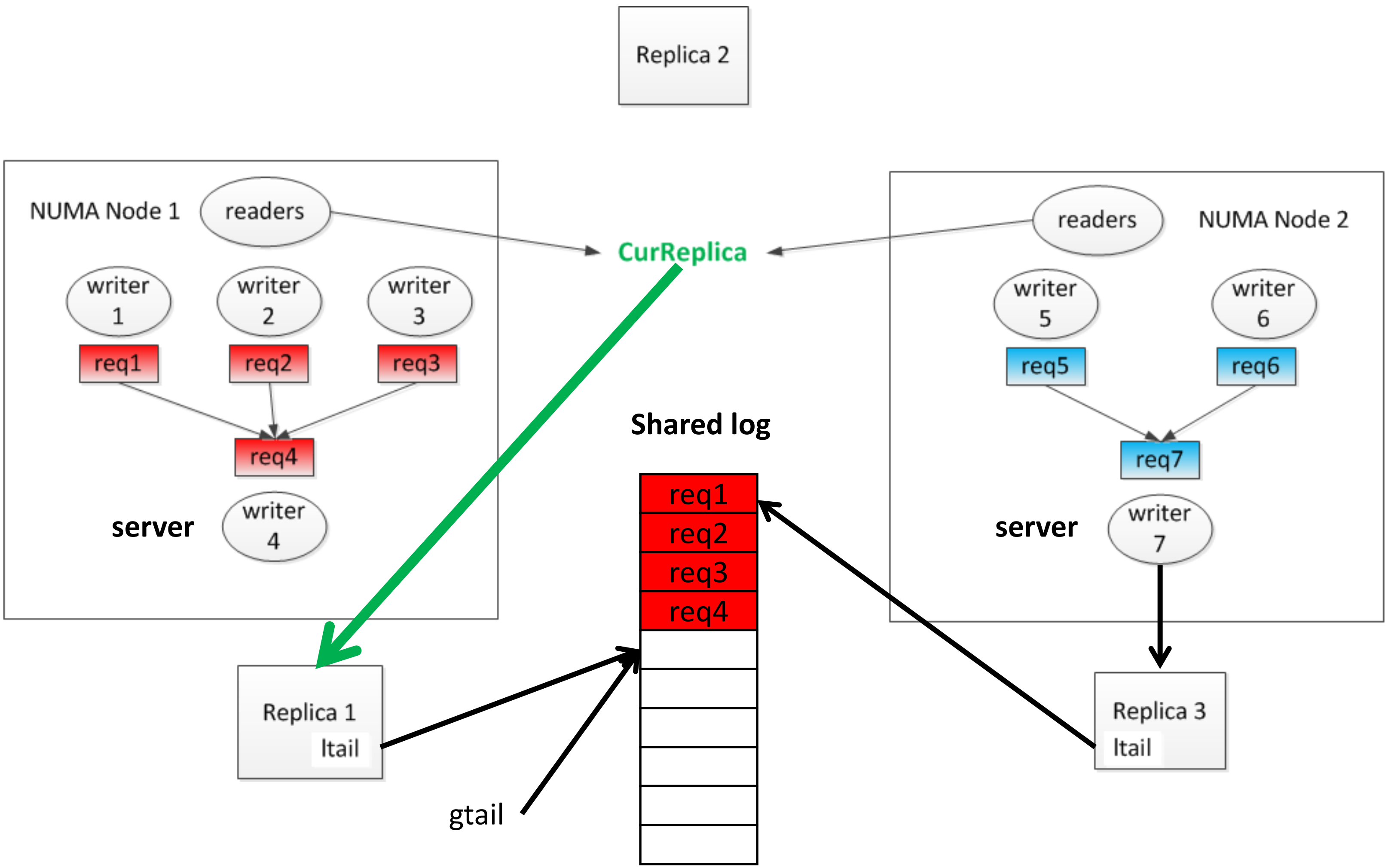
Shared log : Execution for requests from a NUMA node 1



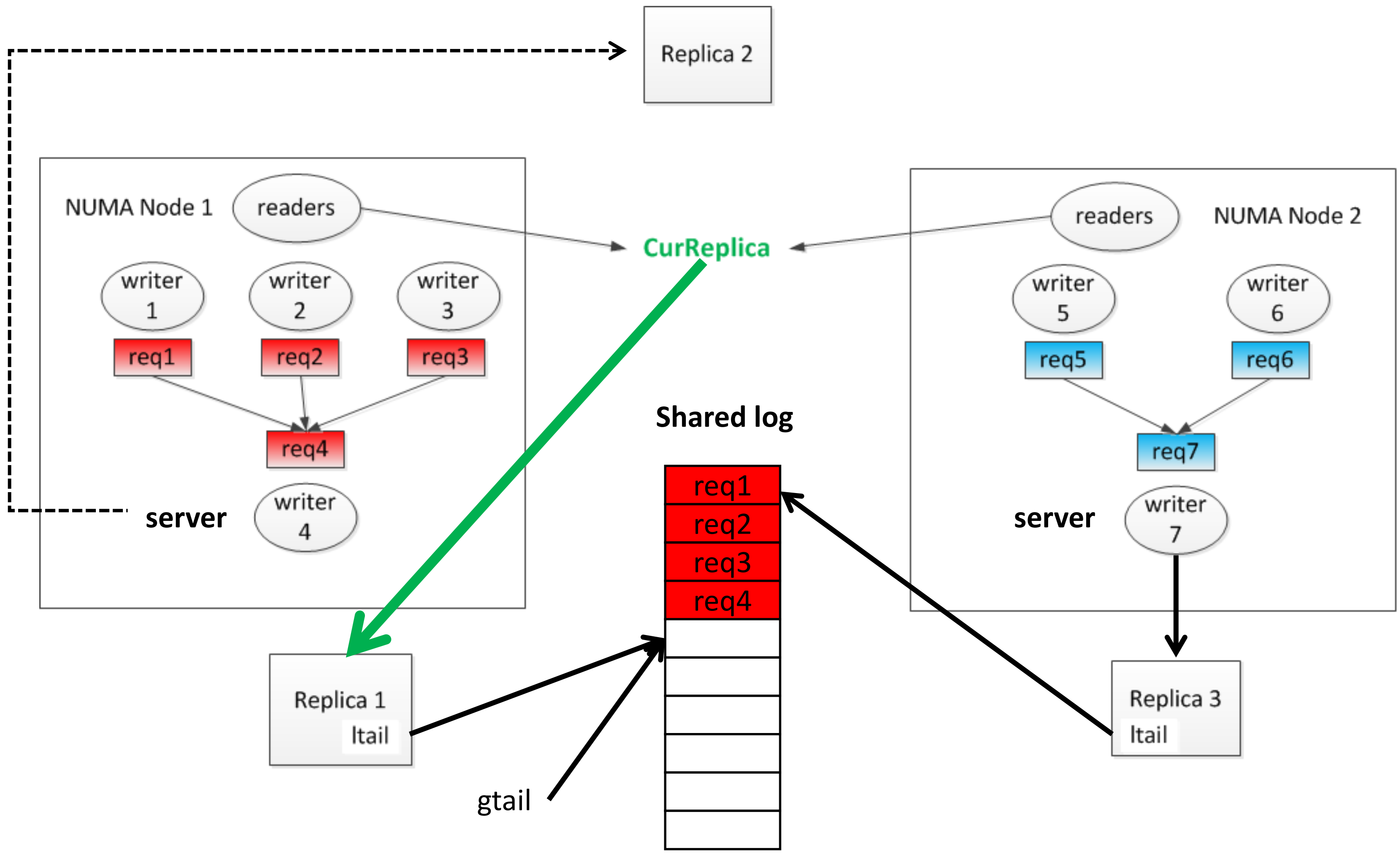
INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING

acm In-Cooperation sighpc

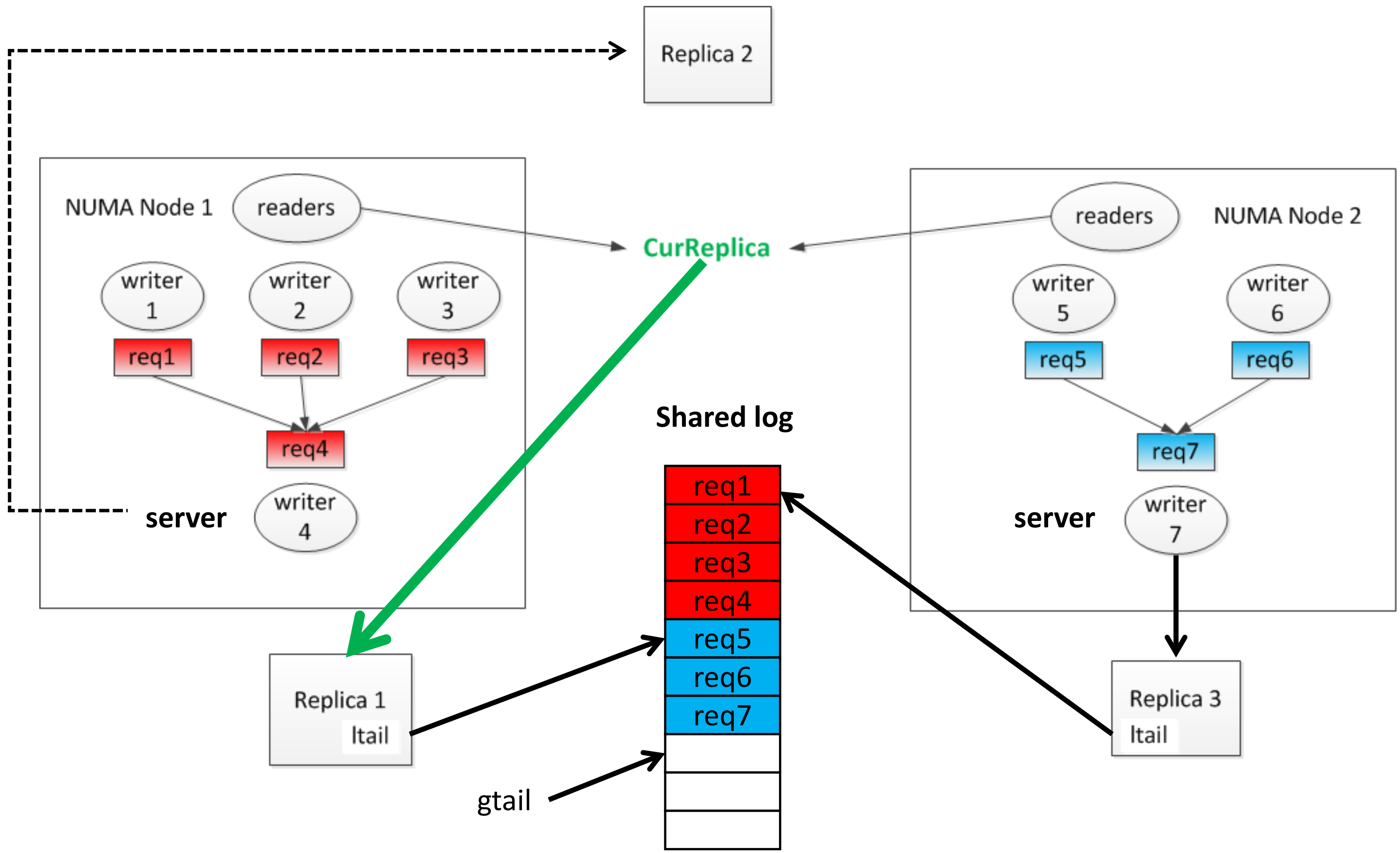
Transition CurReplica to Replica 1



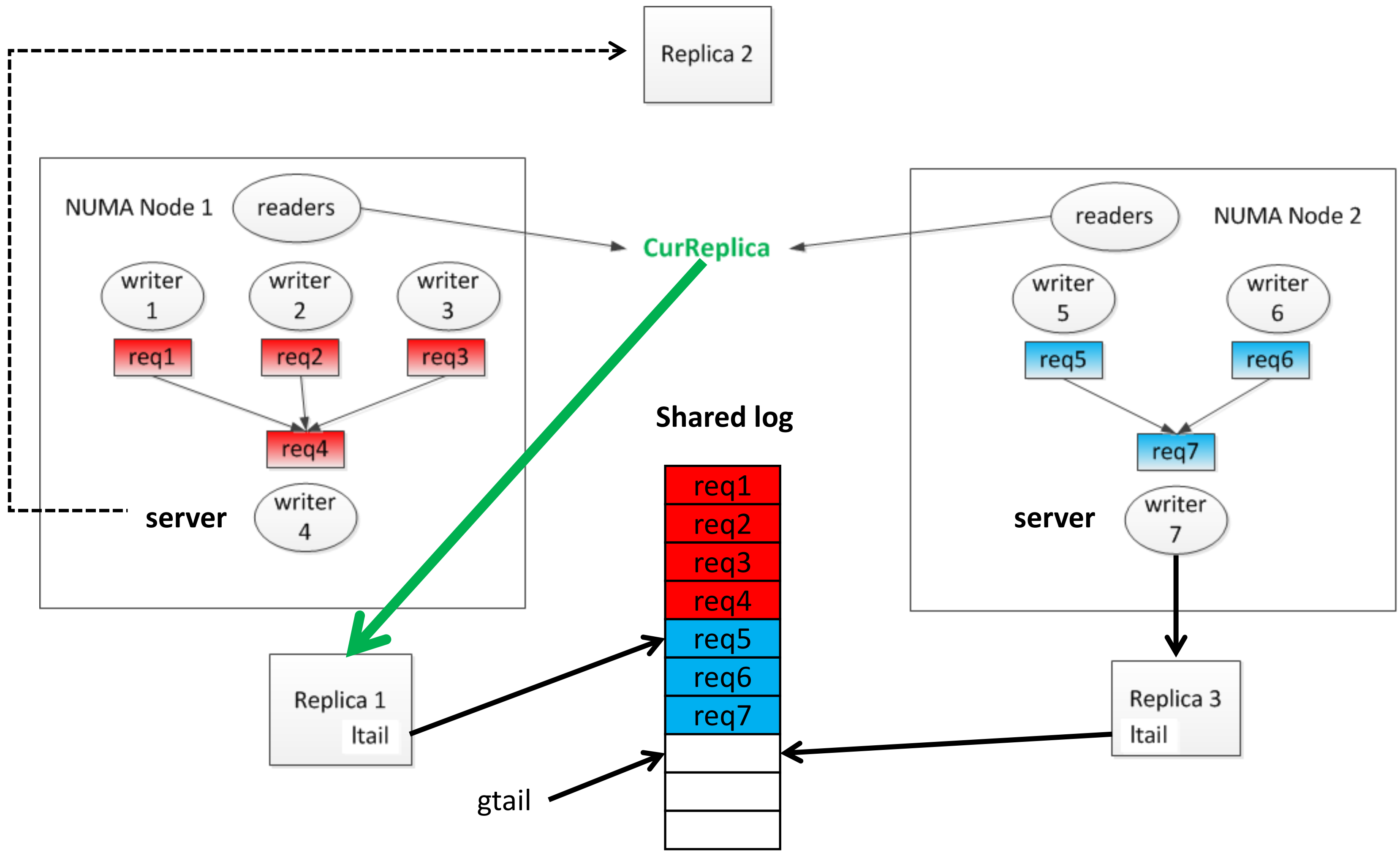
Transition CurReplica to Replica 1



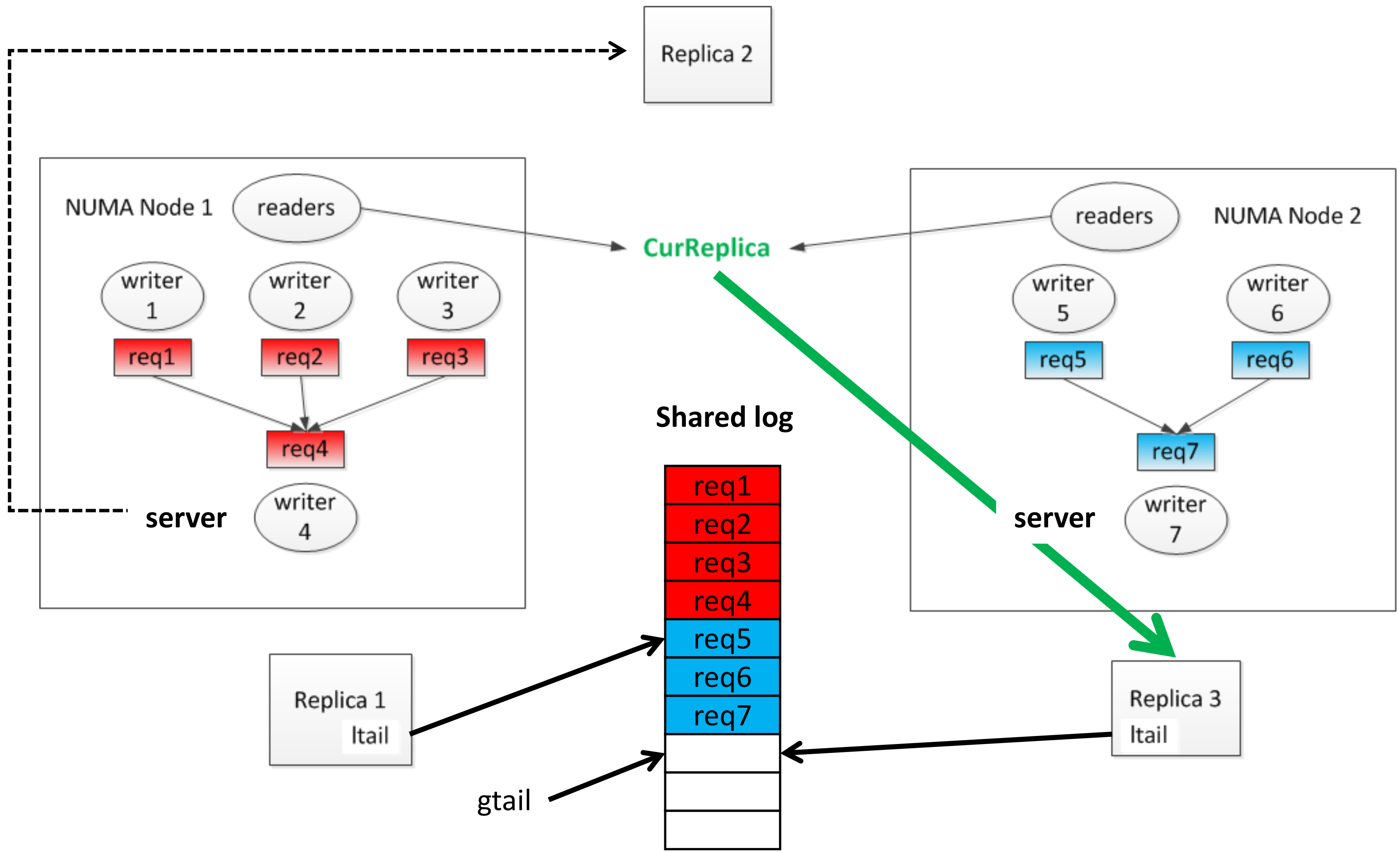
Delegation : Collect and write requests within a NUMA node 2



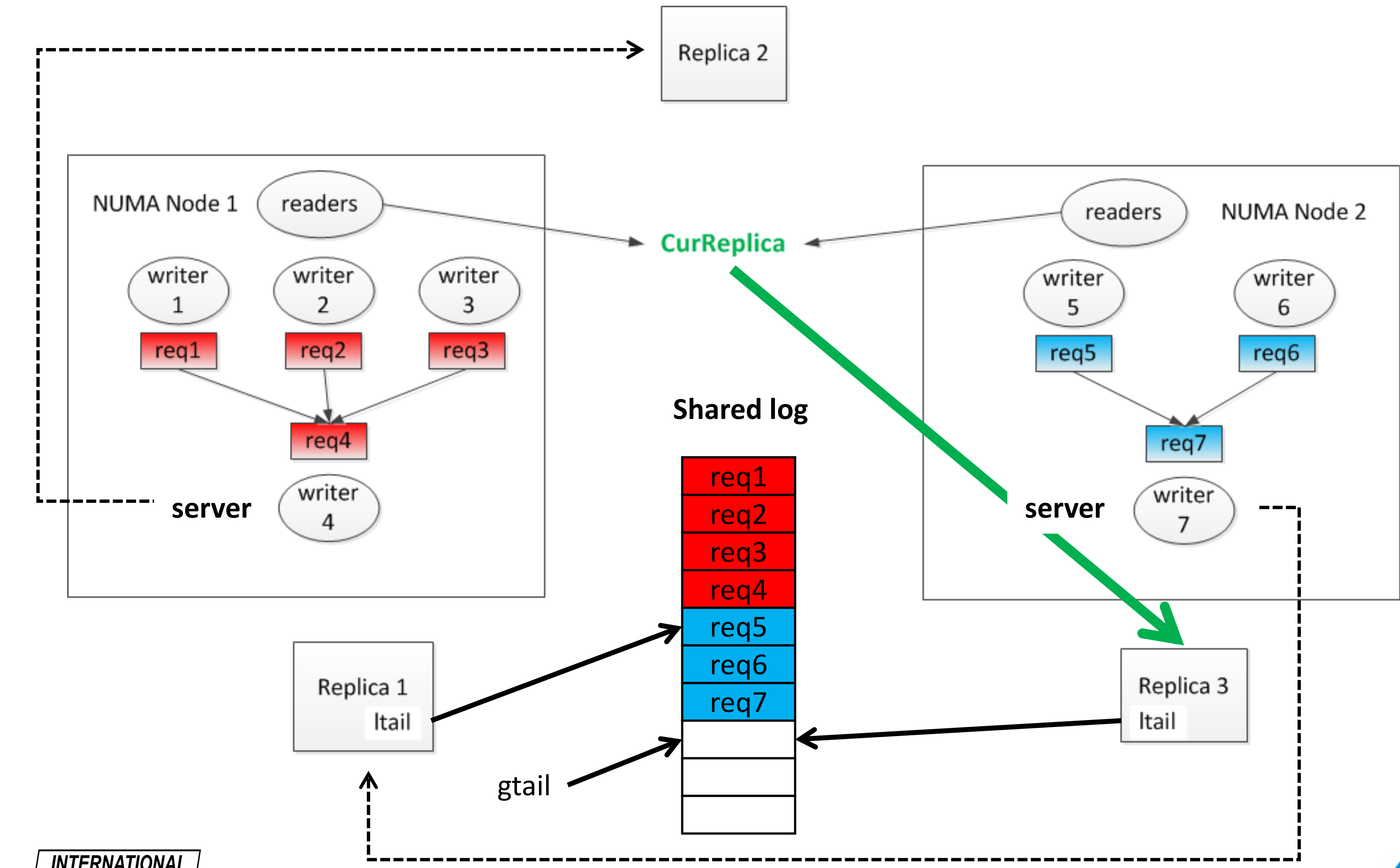
Shared log: Synchronization and execution for requests



Transition CurReplica to Replica 3



Transition CurReplica to Replica 3



INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING

acm In-Cooperation sighpc

Results

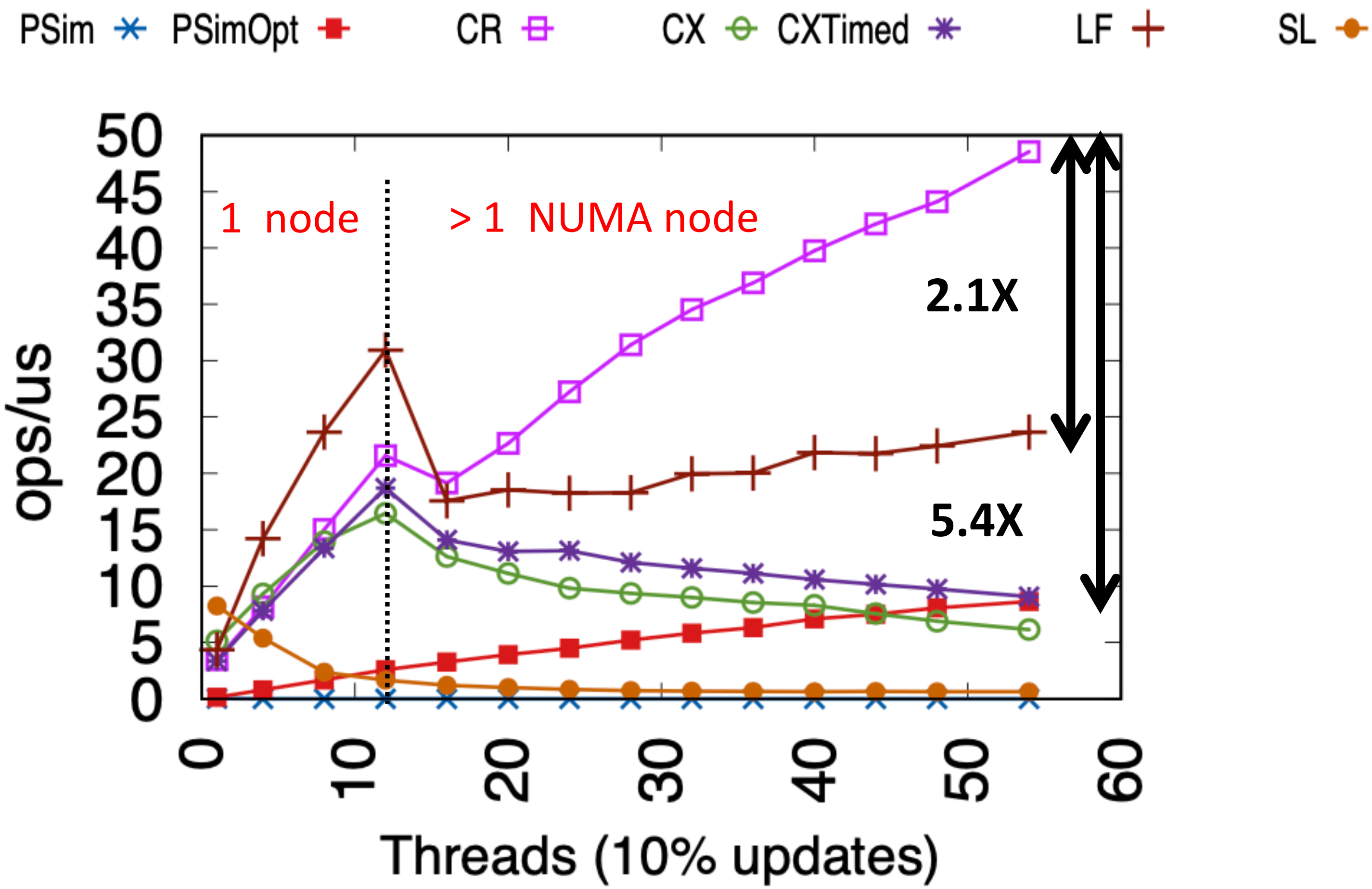
Sever:

2 NUMA nodes

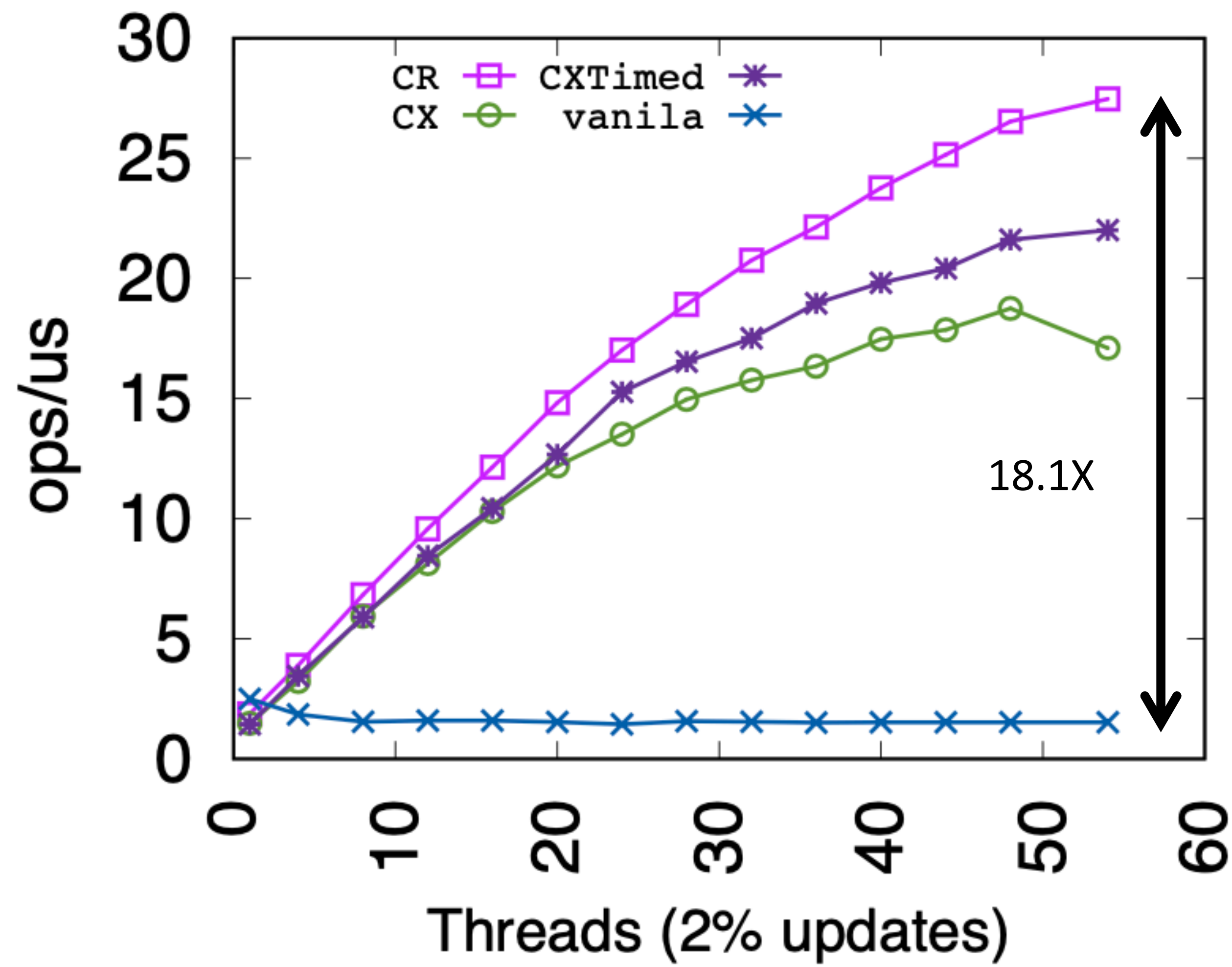
14 cores/node + hyperthreading

(total 56 hardware threads)

Skiplist Priority Queue – 10% Updates



Using CR in KyotoCabinet – 2% Updates



INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING

acm In-Cooperation sighpc

Conclusion: CR Works Well

- Keep one update-to-date replica for read-only access at all times
- Use a shared log to synchronize cross-nodes threads
- use delegation to synchronize local threads

Thank you!