Ίδρυμα Προώθησης Έρευνας

ΚΥΠΡΙΑΚΗ ΔΗΜΟΚΡΑΤΙΑ

ΔΙΑΡΘΡΩΤΙΚΑ ΤΑΜΕΙΑ της Ευρωπαϊκής Ένωσης στην Κύπρο
οι ιδέες μας, πράξη και ανάπτυξη

ΕΥΡΩΠΑΪΚΗ ΕΝΩΣΗ

Η ΔΕΣΜΗ 2009-10 ΣΥΓΧΡΗΜΑΤΟΔΟΤΕΙΤΑΙ ΑΠΟ ΤΗΝ ΚΥΠΡΙΑΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΚΑΙ ΤΟ ΕΥΡΩΠΑΪΚΟ ΤΑΜΕΙΟ ΠΕΡΙΦΕΡΕΙΑΚΗΣ ΑΝΑΠΤΥΞΗΣ ΤΗΣ ΕΕ

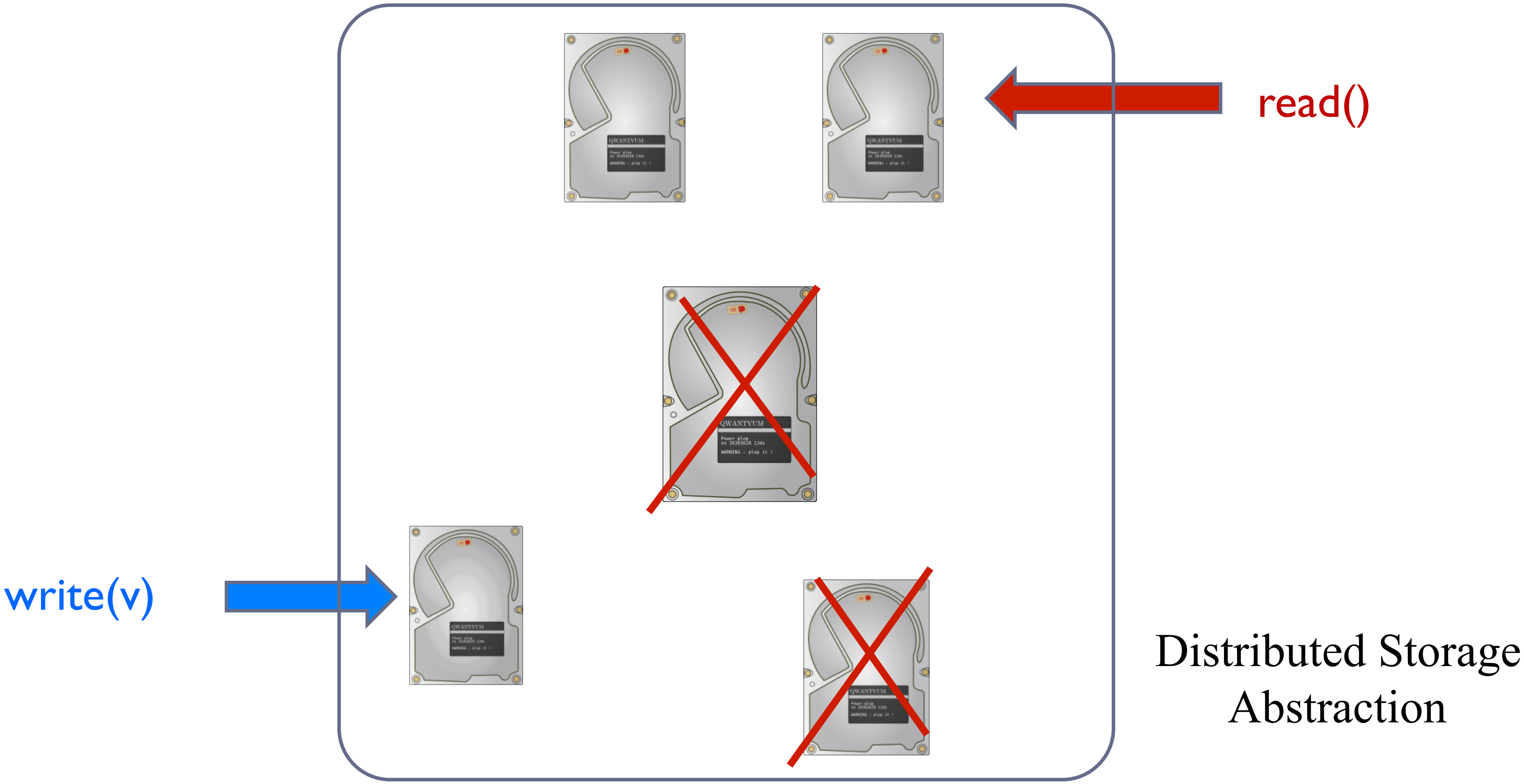
Towards Feasible Implementations of Low-Latency Multi-Writer Atomic Registers

Nicolas Nicolaou
Joint with: C. Georgiou, A. Russell, A. Shvartsman

University of Cyprus &
University of Connecticut

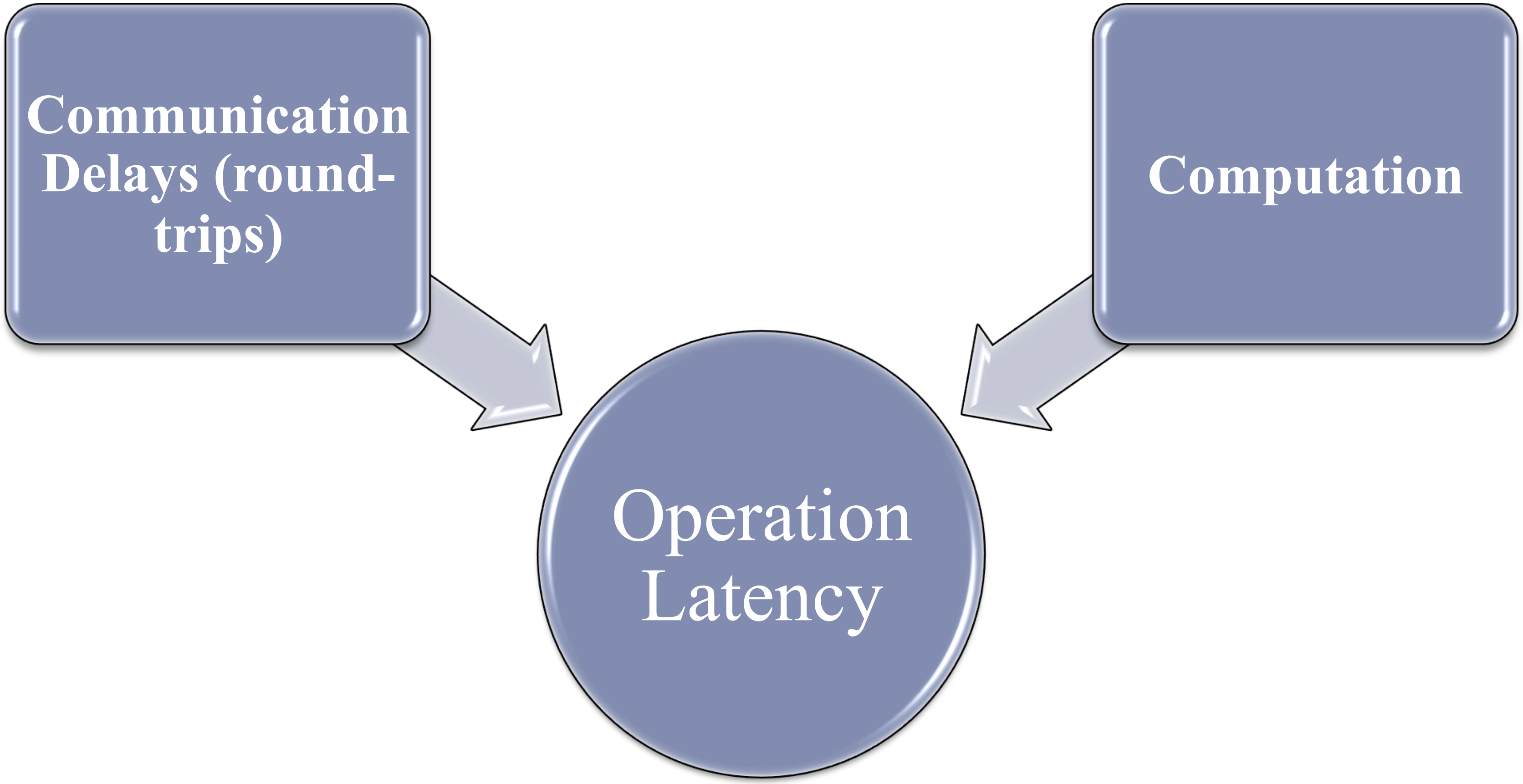
Funded by the Cyprus Research Promotion Foundation and co-funded by the Republic of Cyprus and the European Regional Development Fund

What is a Distributed Storage System?



- Data Replication – Servers/Disks
  - Survivability and Availability
- Read/Write operations
- Consistency Semantics
  - Atomicity

Complexity Measure



What was known…

Traditional SWMR

- [Attiya et al. 95]
- Single round writes
- Two round reads
  - Phase 1: Obtain latest value
  - Phase 2: Propagate latest value
    - Folklore belief: "Read must Write"

Traditional MWMR

- Two round writes
  - Phase 1: Discover latest value
  - Phase 2: Order new value after the latest and propagate
    - Belief: "Writes must Read"
- Two round reads

Minimal Computation

The Era of Fast Implementations…

Fast SWMR

- Single round (Fast) Writes
- All fast reads with bounded readers [Dutta et al. 04]
- Semi-fast: A single slow read per write and unbounded readers [Georgiou et al. 06]
- Not applicable in the MWMR model

Fast MWMR

- Algorithm SFW [Englert et al. 2009]
- Server Side Ordering (SSO)
  - Order writes at the servers
  - Avoid discovery of the latest value from the writer
- Enables Fast (Single Round) Writes and Reads -- first such algorithm

Computationally Demanding

Model

- Asynchronous, Message-Passing model
  - Process sets: writers W, readers R, servers S (replica hosts)
  - Reliable Communication Channels
  - Well Formedness

- Environments:
  - SWMR: $|W|=1$, $|R|\geq 1$
  - MWMR: $|W|\geq 1$, $|R|\geq 1$

- Failures:
  - Crash Failures

- Correctness: Atomicity (safety), Termination (liveness)

Quorum Systems

- Quorum System $\mathbf{Q}$:

$$\mathbf{Q} = \{Q : Q \subseteq S\}\ s.t.\ \forall Q_i, Q_j \in \mathbf{Q} : Q_i \cap Q_j \neq \varnothing$$

- n-wise Quorum System $\mathbf{Q}$:

$$\mathbf{Q} = \{Q : Q \subseteq S\}\ where\ \forall A \subseteq \mathbf{Q} : |A| = n \text{ and } \bigcap_{Q \in A} Q \neq \varnothing$$

  - $2 \leq n \leq |\mathbf{Q}|$: intersection degree

- Faulty Quorum: Contains a faulty process
  - At least a single quorum contains non-faulty replicas

Algorithm: SFW (in a glance)

Write Protocol: one or two rounds

- P1: Collect candidate tags from a quorum
  - Exists tag t propagated in a bigger than (n/2-1)-wise intersection (PREDICATE PW)
    - YES – assign t to the written value and return => **FAST**
    - NO - propagate the unique largest tag to a quorum => **SLOW**

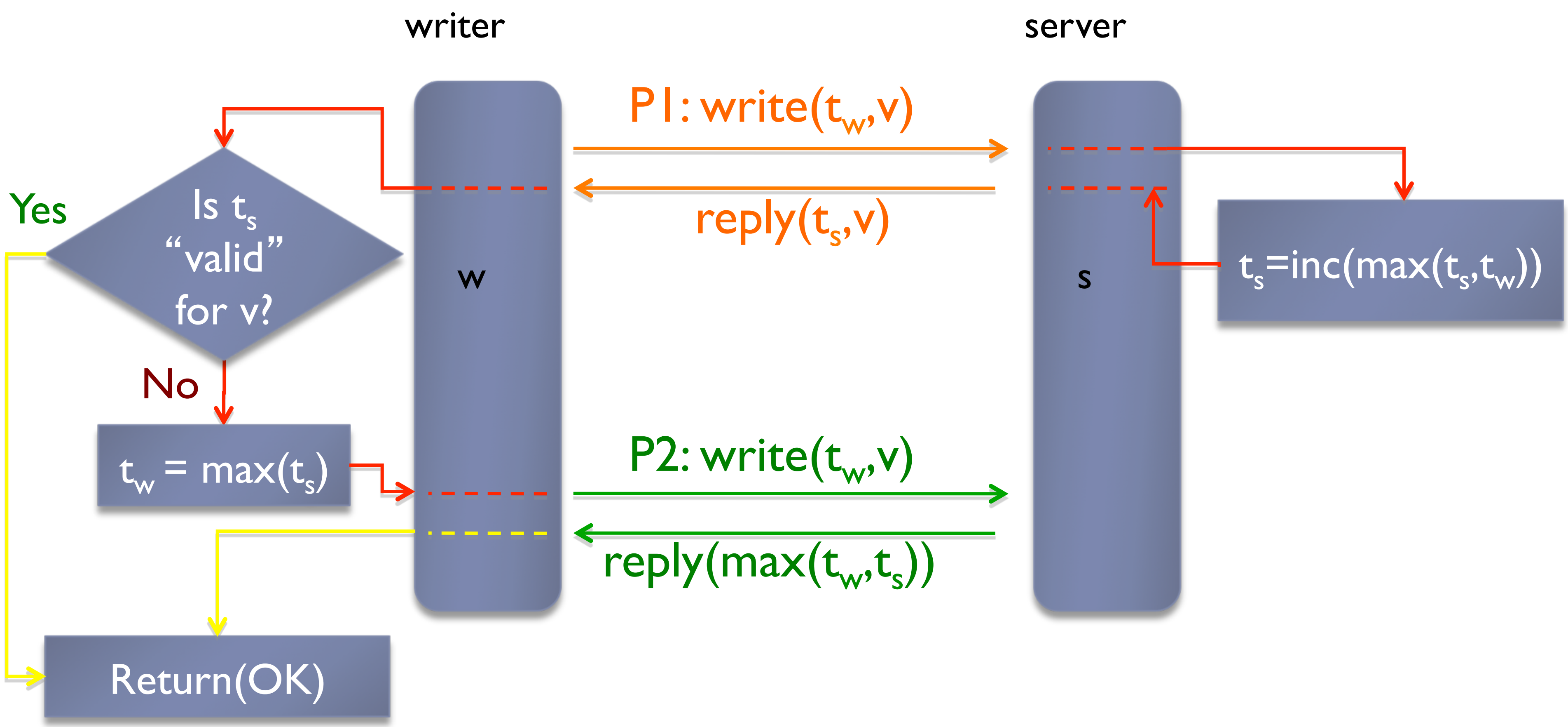
Read Protocol: one or two rounds

- P1: collect list of writes and their tags from a quorum
  - Exists max write tag t in a bigger than (n/2-2)-wise intersection (PREDICATE PR)
    - YES – return the value written by that write => **FAST**
    - NO – is there a confirmed tag propagated to (n-1)-wise intersection => **FAST**
    - NO - propagate the largest confirmed tag to a quorum => **SLOW**

Server Protocol

- Increment tag when receive write request and send to read/write the latest writes

SFW Writer-Server Interaction



The Weak Side of SFW

- Predicates are Computationally Hard
  - NP-Complete (Shown in this paper)

- Restriction on the Quorum System
  - Deploys n-wise Quorum Systems
  - Guarantees fastness iff $n>3$

The Good News…

- Approximation Algorithm (APRX-SFW)
  - Polynomial
  - Log-approximation
    - log|S| times the optimal number of fast operations
- Algorithm CWFR
  - Based on Quorum Views
    - SWMR prediction tools
  - Fast operations in General Quorum Systems
  - Trades Speed of Write operations
    - Two Round Writes

NP-Completeness

K-SET-INTERSECTION: (captures both PR and PW)

Given a set of elements U, a subset of those elements $M \subseteq U$, a set of subsets $\mathbb{Q} = \{Q_1, \ldots, Q_n\}$ s.t. $Q_i \subseteq U$, and an integer $k \leq |\mathbb{Q}|$, a set $I \subseteq \mathbb{Q}$ is a k-intersecting set if: $|I| = k$, $\bigcap_{Q \in I} Q \subseteq M$, and $\bigcap_{Q \in I} Q \neq \emptyset$.

Theorem: K-SET-INTERSECTION is NP-complete (reduction from 3-SAT).

k-Set-Intersection Approximation

- Greedy algorithm
  - Uses Set Cover greedy approximation algorithm at its core
- Given $(U, M, \mathbb{Q}, k)$ do:

Step 1:

$$\forall m \in M, T_m = \{(U \setminus M) \setminus (Q_i \setminus M) : m \in Q_i\}$$

Step 2: Run k-SET-COVER greedy algorithm on $(U \setminus M, T_m, k)$

- 2a: Pick $R \in T_m$ with the maximum uncovered elements
- 2b: Take the union of every set picked in 2a
- 2c: If the union is $U \setminus M$ go to step 3, else if we picked less than k sets go to 2a, else repeat for another $m \in M$

Step 3:

- For every set $(U \setminus M) \setminus (Q_i \setminus M)$ in the set cover, add Q_i in the intersecting set

Algorithm Rationale

- Let for $m \in M, Q_i$

$$R_{m,i} \in T_m : R_{m,i} = (U \setminus M) \setminus (Q_i \setminus M)$$

- If we can find k sets such that:

$$R_{m,1} \cup \ldots \cup R_{m,k} = U \setminus M$$

- By de Morgan's: $\overline{R_{m,1}} \cap \ldots \cap \overline{R_{m,k}} = \emptyset$

- Since $\overline{R_{m,i}} = (Q_i \setminus M)$ and $m \in Q_i$ for $i \in [1, \ldots, k]$

$$m \in Q_1 \cap \ldots \cap Q_k \text{ and } Q_1 \cap \ldots \cap Q_k \subseteq M$$

Approximation Algorithm: APRX-SFW

- Adopt k-Set-Intersection Approximation:
  - $U = S$ the set of servers
  - $\mathbb{Q} = \{Q_1, \ldots, Q_q\}, Q_i \subset S$ is the quorum system
  - $M \subseteq S$ the servers that replied with the latest value
  - k the number of quorums required by the predicates

- Log-Approximation
  - Invalidates RP and WP a factor of log|S| times

- What does it mean for SFW?
  - Extra Communication Rounds (esp. for writes)
  - Slower acceptance of a new value
  - Does not affect correctness

Unrestricting Quorums

- APRX-SFW
  - Improves Computation Time
  - Still relies on n-wise Quorum Systems
    - n>3 to allow fast operations

Can we allow fast operations in the MWMR when deploying General Quorum Systems?

Tool: Quorum Views

Used in the SWMR [Georgiou et al. 08]

Idea:

- Try to determine the state of the write operation based on the distribution of the latest value in the replied quorum.

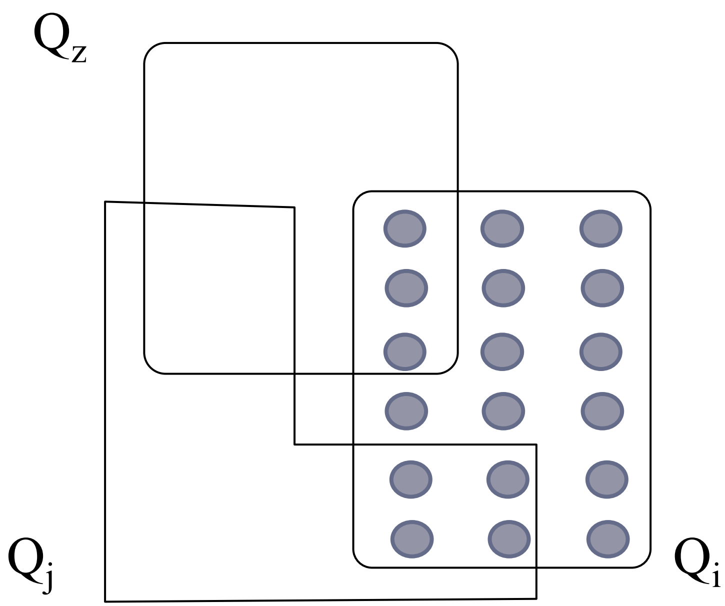
- Write State in the First Round of Read Operation

  Determinable => Read is Fast

  Undeterminable => Read is Slow

Determinable Write - Qview(1)

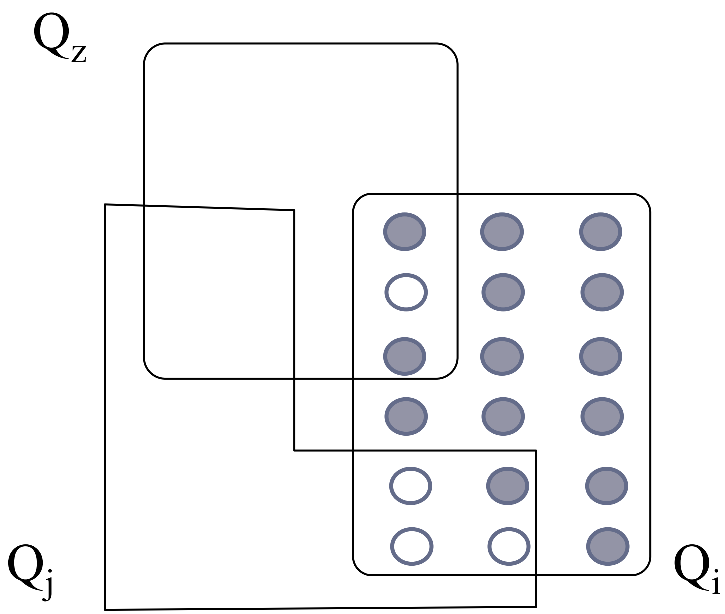
- All members of a quorum contain maxTs



(Potentially) Write Completed

Determinable Write - Qview(2)

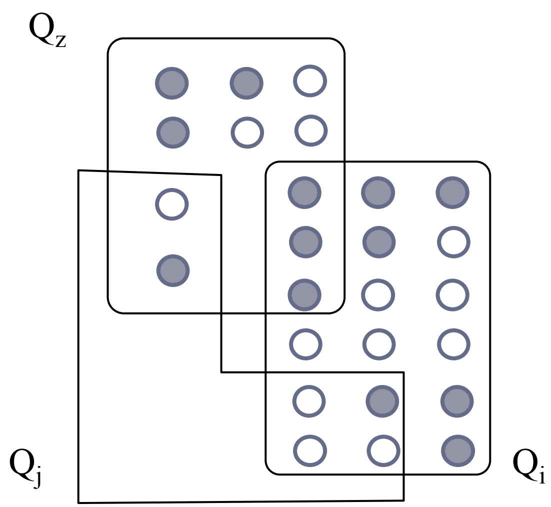
- Every intersection contains a member with ts<maxTs



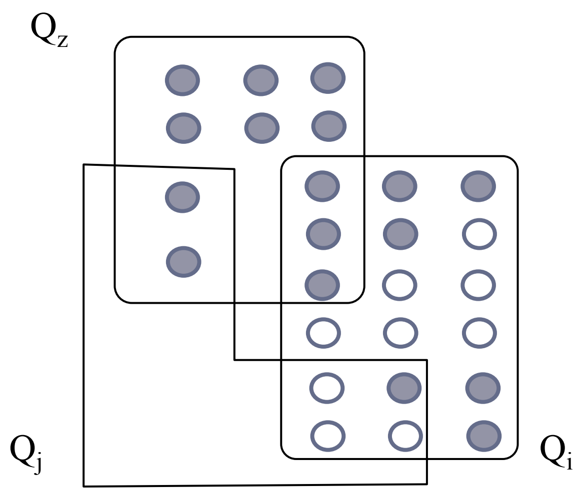
(Definitely) Write <maxTag,v> Incomplete

Undeterminable Write - Qview(3)

- There is intersection with all its members with ts=maxTs



qV(3) and Incomplete Write

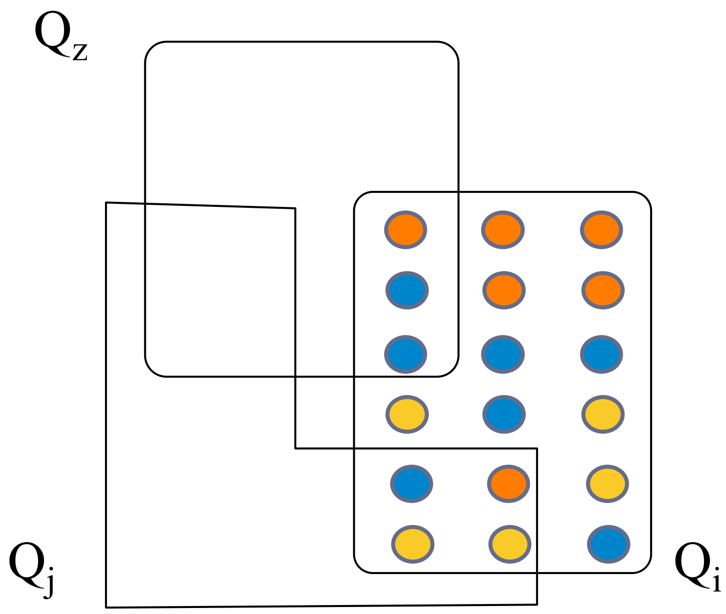


qV(3) and Complete Write

Undeterminable => second Com. Round

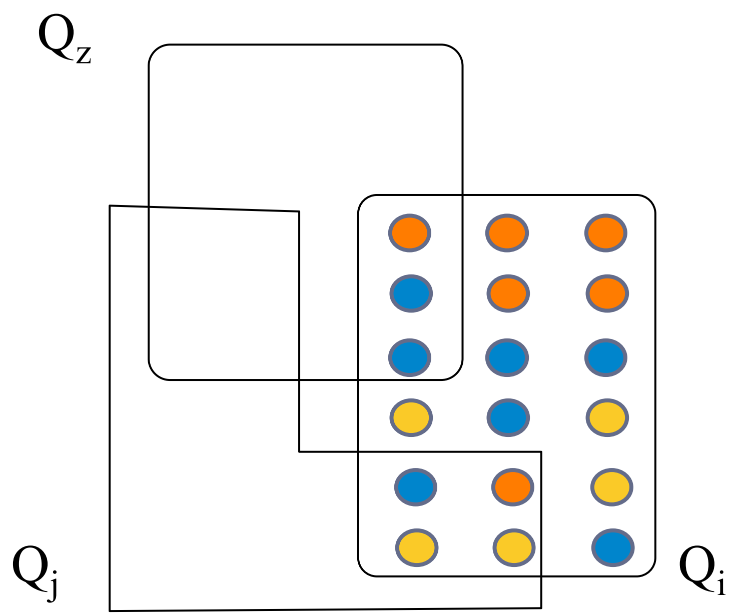
What happens in MWMR?

- MWMR environment
  - Concurrent writes
  - Multiple concurrent values
- For values <tag1,v1> , <tag2, v2>, <tag3,v3>
  - Let tag1 < tag2 < tag3



Idea: Uncover the Past

- Discover the latest potentially completed write
- For values <tag1,v1> , <tag2, v2>, <tag3,v3>:
  - <tag3,v3> not completed (servers possibly contained <tag2, v2>)
  - <tag2, v2> not completed (servers possibly contained <tag1,v1>)
  - <tag1,v1> potentially completed



Algorithm: CWFR

Traditional Write Protocol: two rounds

- P1: Query a single quorum for the latest tag
- P2: Increment the max tag, send <newtag, v> quorum

Read Protocol: one or two rounds

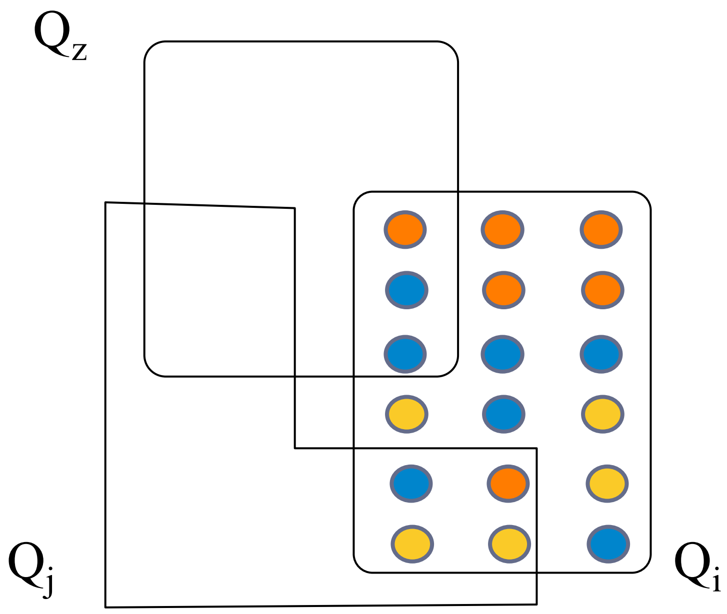
- Iterate to discover smallest completed write
- P1: receive replies from a quorum Q
  - $QView_Q(1)$ – **Fast**: return maxTag of current iteration
  - $QView_Q(2)$ – remove servers with maxTag and re-evaluate
  - $QView_Q(3)$ – **Slow**: propagate and return $maxTag_0$

Server Protocol: passive role

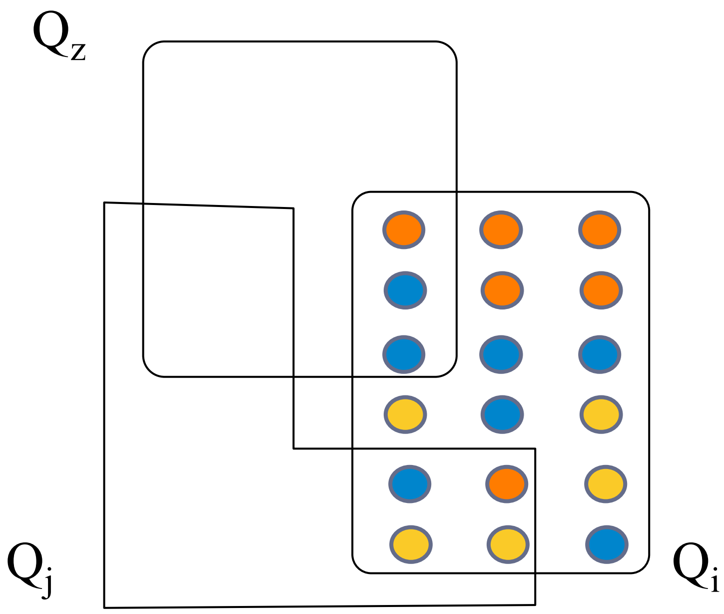
- Receive requests, update local timestamp and return <tag,v>

Read Iteration: Discard Incomplete Tags

- For values <tag1,v1> , <tag2, v2>, <tag3,v3>:
  - <tag3,v3> not completed: remove servers that contain <tag3,v3>
  - <tag2, v2> not completed: remove servers that contain <tag2, v2>
  - <tag1,v1> potentially completed in Q_i
    - Qview(1) : all remaining servers contain <tag1,v1>

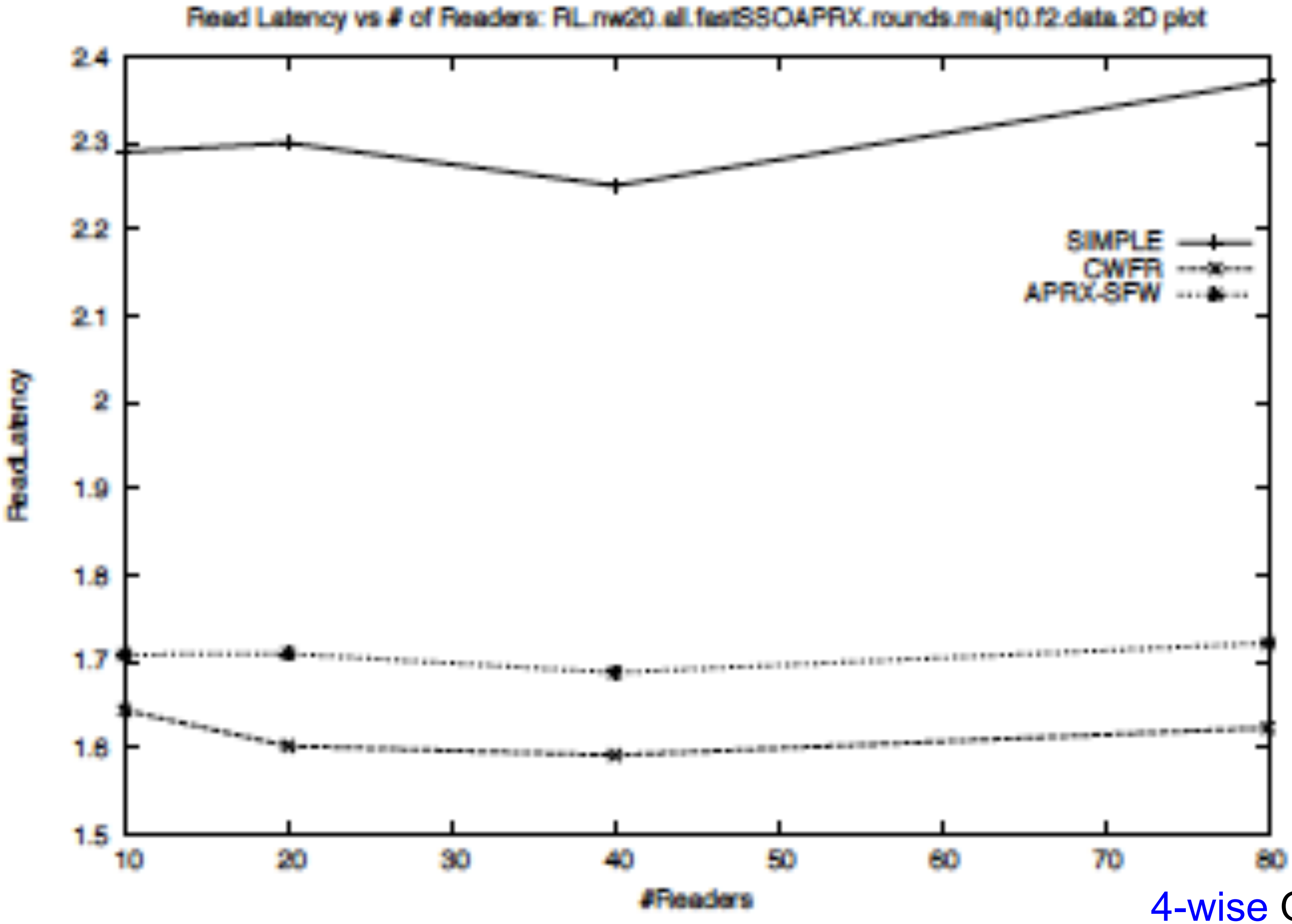


Server Removal



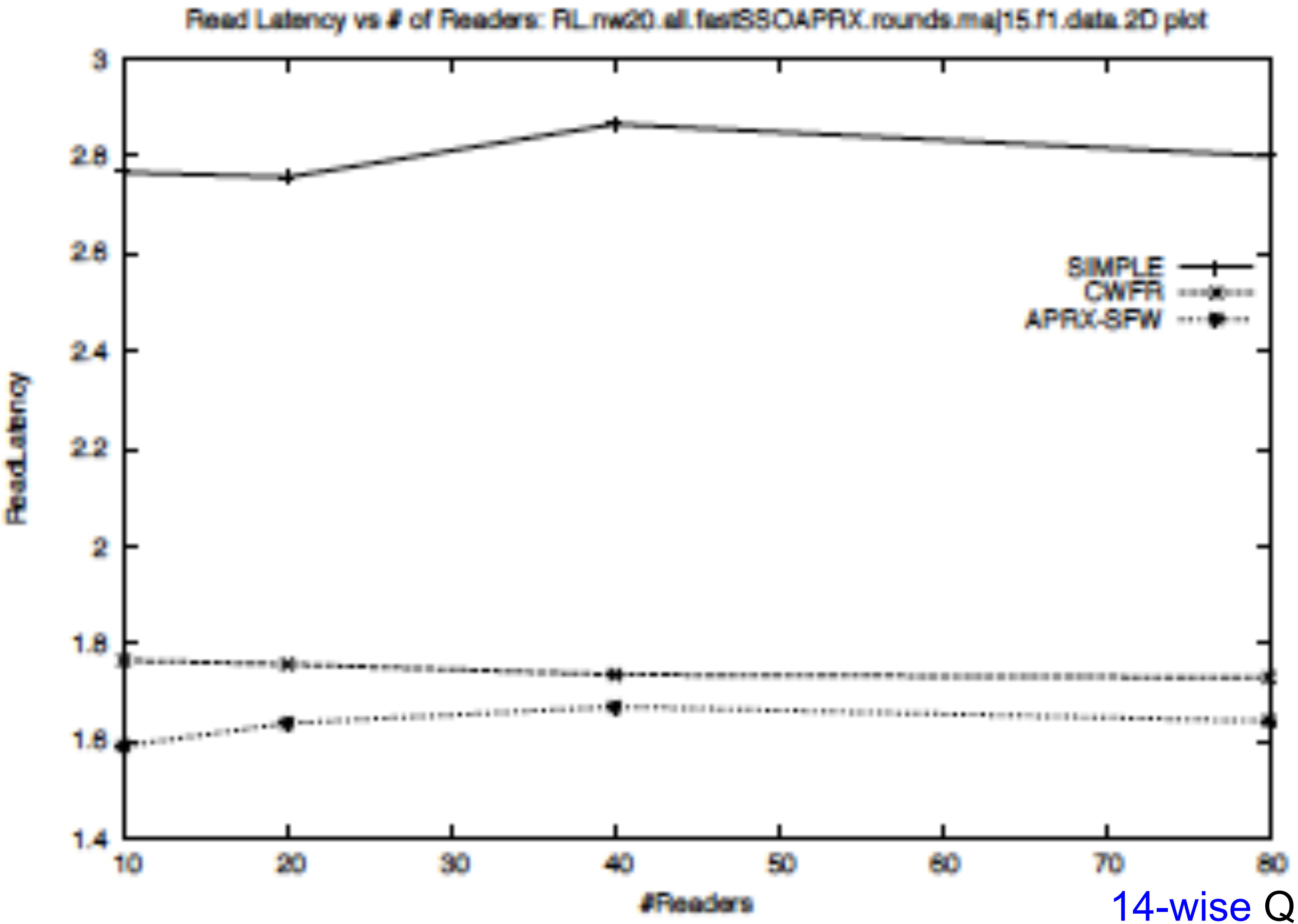
Past Prediction

APRX-SFW – CWFR Empirical Results



4-wise Quorum System

APRX-SFW – CWFR Empirical Results



14-wise Quorum System

Conclusions

- Presented two Atomic Register MWMR implementations
  - Computation and Communication factor
- Algorithm: APRX-SFW
  - Polynomial-Approximation of SFW predicates
  - log|S|-approximation
  - Requires n-wise Quorum Systems for n>3
- Algorithm: CWFR
  - General Quorum systems
  - Trades the Speed of write operations
- Preliminary Experiments
  - Both algorithms overperform classic approach
  - Bigger Intersections favor the APRX-SFW