Ίδρυμα Προώθησης Έρευνας
ΚΥΠΡΙΑΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΕΥΡΩΠΑΪΚΗ ΕΝΩΣΗ
Η ΔΕΣΜΗ 2009-10 ΣΥΓΧΡΗΜΑΤΟΔΟΤΕΙΤΑΙ ΑΠΟ ΤΗΝ ΚΥΠΡΙΑΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΚΑΙ ΤΟ ΕΥΡΩΠΑΪΚΟ ΤΑΜΕΙΟ ΠΕΡΙΦΕΡΕΙΑΚΗΣ ΑΝΑΠΤΥΞΗΣ ΤΗΣ ΕΕ

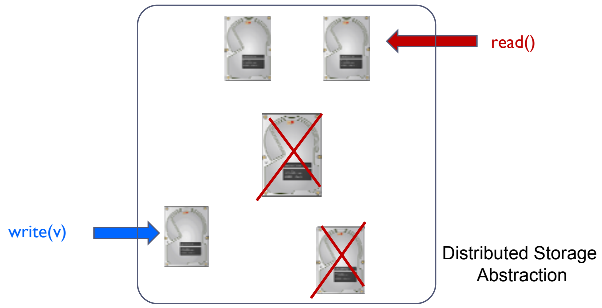
How fast read and write operations in MWMR atomic register implementations can be?

Nicolas Nicolaou

University of Cyprus &
University of Connecticut

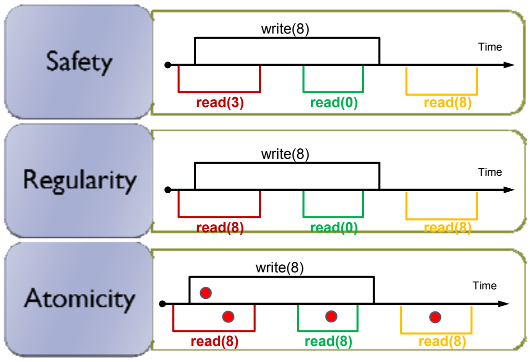
Funded by the Cyprus Research Promotion Foundation and co-funded by the Republic of Cyprus and the European Regional Development Fund

What is a Distributed Storage System?



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

Consistency Semantics [Lamport86]



Complexity Measure-Operation Latency

- Consistent Register Implementations
  - **Message-Passing, Asynchronous** model
  - Access **multiple replicas** per operation
  - Perform **multiple accesses** per operation

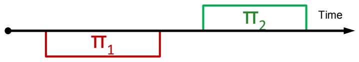
Operation Latency is measured in **Communication Rounds (round-trips)**

Model: Definitions

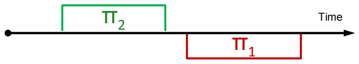
- Asynchronous, Message-Passing model
  - Process sets: writers W, readers R, servers S (replica hosts)
  - Reliable Communication Channels
  - Well Formedness
- Environments:
  - SWMR: |W|=1, |R|≥1
  - MWMR: |W|≥1, |R|≥1
- Failures:
  - Crash Failures
- Correctness: Atomicity (safety), Termination (liveness)

Definition: Operation Relations

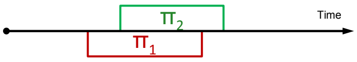
- Precedence Relations for two operations π_1 ,π_2:
  - π_1 precedes π_2 if the response of π_1 happens before the invocation of π_2



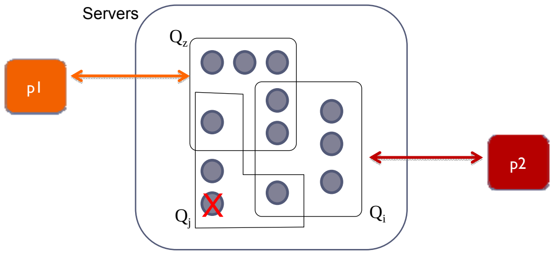
  - π_1 succeeds π_2 if the invocation of π_1 happens after the response of π_2



  - π_1 is concurrent with π_2 if π_1 neither precedes nor succeeds π_2

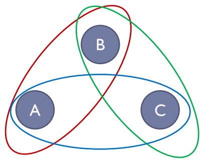


Definition: Quorum systems

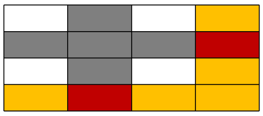


- Q_i, Q_j, Q_z are **quorums**
- **Quorum System** is the set $\{Q_i, Q_j, Q_z\}$
  - Property: every pair of quorums intersects
- Every R/W operation communicates with a single quorum
- Faulty Quorum: Contains a faulty process

Prior Work: Quorum Systems Examples



Majorities [Thomas79,Gifford79]



Matrix Quorums [VA92]

Definition: Fastness

- A process p performs a communication round during an operation π if:
  - p sends a message m to a set of servers for π
  - Any server that receives m replies to p
  - Once p receives responses from a single quorum completes π or proceeds to a next communication round
- Fast Operation
  - Completes at the end of its first round
- Fast Implementation
  - All operations are fast
- Communication scheme
  - Message delivery: Servers to Clients
  - No server to server or client to client communicaiton

Definition: Tag - <Timestamp, W> pair

- How can we order the written values?
  - without clock synchronization
  - without assuming access to centralize clock (e.g. , GPS)
- TAG : <timestamp, wid> pair
  - tag1 > tag2 if either:
    - tag1.timestamp > tag2.timestamp, or
    - tag1.timestamp = tag2.timestamp AND tag1.wid > tag2.wid
- Writers assign tags to written values
  - Writes <tags, value> pairs

Algorithm: Simple

Write Protocol: two rounds

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

Read Protocol: two rounds

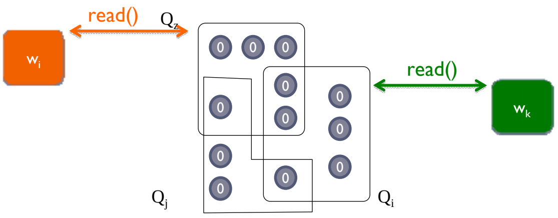
- **P1: Query a single quorum for the latest tag**
- **P2: Propagate <maxtag,v> to a single quorum**

Server Protocol: passive role

- **Receive requests, update local timestamp (if msg.tag>server.tag) and reply with <server.tag,v>**

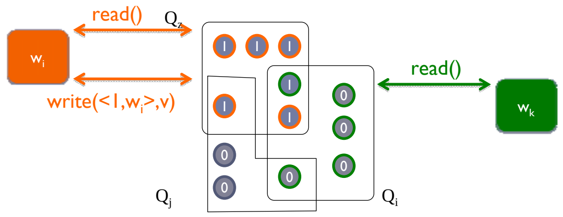
Example: Simple (write operations)

- Assume $w_i > w_k$



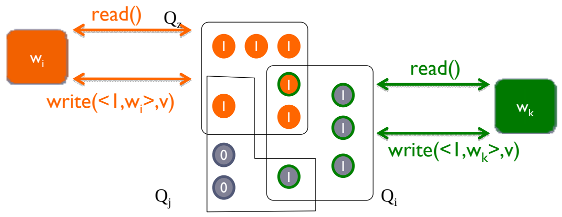
Example: Simple (write operations)

- Assume $w_i > w_k$



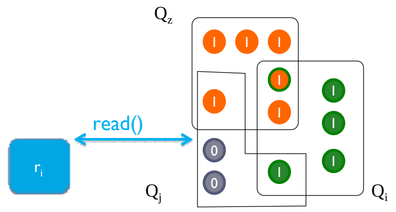
Example: Simple (write operations)

- Assume $w_i > w_k$



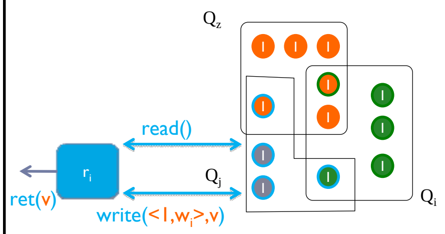
Example: Simple (read operation)

- Assume $w_i > w_k$



Example: Simple (read operation)

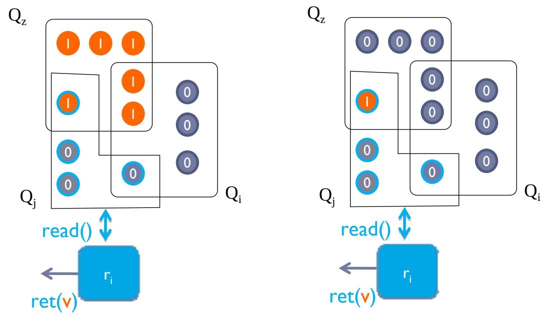
- Assume $w_i > w_k$



Operation Ordering: w_k -> w_i -> r_i

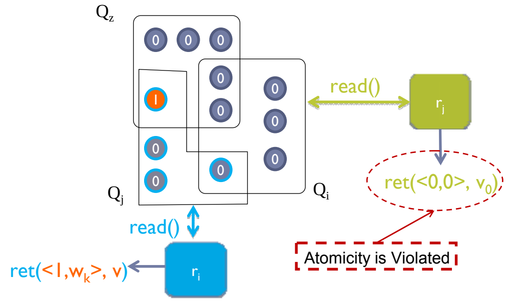
Why a read performs 2 rounds?

Consider the following executions with single round reads:



Why a read performs 2 rounds? (Cont.)

Extend the second execution with a read from rj:



Folklore Belief: **Reads must Write in MR environments**

Question

Can we allow some reads to be fast (single round reads) and still guarantee atomicity?

Answer: **YES!!**

Tool: Quorum Views

Idea:

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

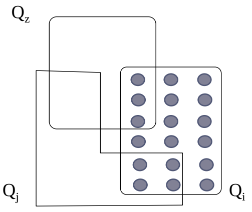
- Write State in the First Round of Read Operation

  Determinable => Read is Fast

  Undeterminable => Read is Slow

Determinable Write - Qview(1)

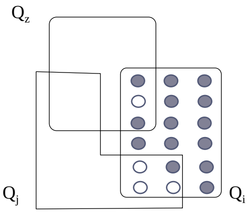
- All members of a quorum contain the maxTag



(Potentially) Write Completed

Determinable Write - Qview(2)

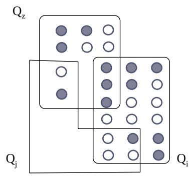
- Every intersection contains a member with tag<maxTag



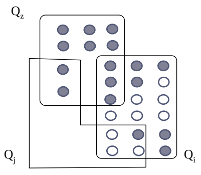
(Definitely) Write <maxTag,v> Incomplete

Undeterminable Write - Qview(3)

- There is intersection with all its members with tag=maxTag



qV(3) and Incomplete Write



qV(3) and Complete Write

Undeterminable => second Com. Round

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

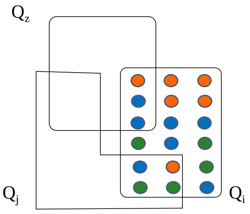
- Iterate to discover smallest completed write
- P1: receive replies from a quorum Q
  - $QView_Q(1)$ – Fast: return maxTag of current iteration
  - $QView_Q(2)$ – remove servers with maxTag and re-evaluate
  - $QView_Q(3)$ – Slow: propagate and return $maxTag_0$

Server Protocol: passive role

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

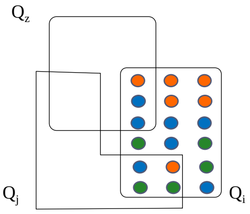
Quorum Views and Multiple Writers

- MWMR environment
  - Concurrent writes
  - Multiple concurrent values
- For tags <1,*> , <2, *>, <3,*>



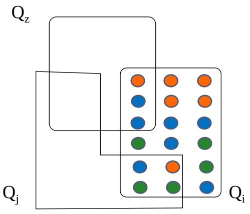
Idea: Predict the Past

- Discover the latest potentially completed write
- For tags <1,*> , <2, *>, <3,*> :
  - <3,*> not completed (servers possibly contained <2, *>)
  - <2, *> not completed (servers possibly contained <1,*>)
  - <1,*> potentially completed

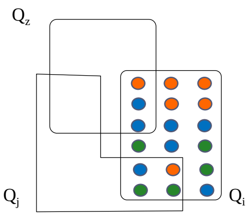


Read Iteration: Discard Incomplete Tags

- For tags <1,*> , <2, *>, <3,*> :
  - <3,*> not completed: remove servers that contain <3,*>
  - <2, *> not completed: remove servers that contain <2, *>
  - <1,*> potentially completed in Q_i
    - Qview(1) : all remaining servers contain <1,*>



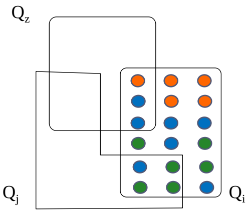
Server Removal



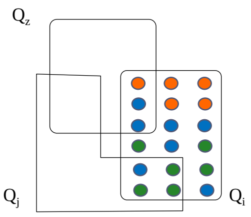
Past Prediction

Read Iteration: Discard Incomplete Tags

- For tags <1,*> , <2, *>, <3,*> :
  - <3,*> not completed: remove servers that contain <3,*>
  - <2, *> potentially completed in Q_j
    - Qview(3) : an intersection of the remaining servers contains <2, *>
    - P2: propagate <3,*> to a complete quorum (help <3,*> to complete)



Server Removal



Past Prediction

What about writes?

Can we allow any writes to be fast (single round reads) and still guarantee atomicity?

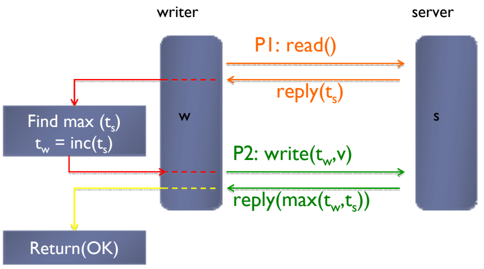
Answer: **YES!!**

New Technique - SSO

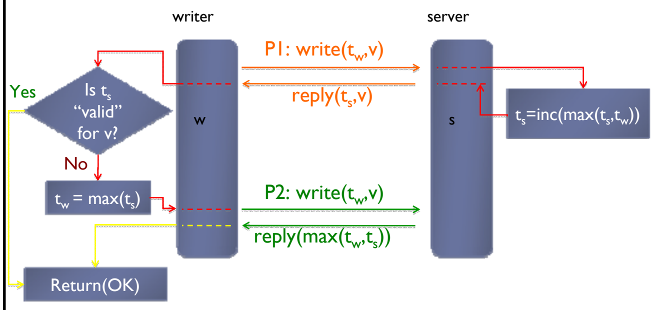
[Englert et. al 09]

- SSO: Server Side Ordering
  - Tag is incremented by the servers and not by the writer.
    - Generated tags may be different across servers
    - Clients decide operation ordering based on server responses
- SSO Algorithm
  - Enables Fast Writes and Reads -- first such algorithm
  - Allows Unbounded Participation

Traditional Writer-Server Interaction



SSO Writer-Server Interaction



Definition: n-wise Quorum Systems

Definition: A quorum system Q is an n-wise quorum system, if:

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

Predicates: Read and Write

Writer predicate for a write ω (PW): $\exists\ \tau, \mathbb{Q}^i, MS$ where: $\tau \in \{\langle ., \omega\rangle : \langle ., \omega\rangle \in m(\omega)_{s,w}.inprogress \ \wedge\ s \in Q\}, MS = \{s : s \in Q \ \wedge\ \tau \in m(\omega)_{s,w}.inprogress\}$, and $\mathbb{Q}^i \subseteq \mathbb{Q}, 0 \leq i \leq \lfloor \frac{n}{2} - 1 \rfloor$, s.t. $(\bigcap_{\mathcal{Q} \in \mathbb{Q}^i \cup \{Q\}} \mathcal{Q}) \subseteq MS$.

Reader predicate for a read ρ (PR): $\exists\ \tau, \mathbb{Q}^j, MS$, where: $\max(\tau) \in \bigcup_{s \in Q} m(\rho)_{s,r}.inprogress$, $MS = \{s : s \in Q \ \wedge\ \tau \in m(\rho)_{s,r}.inprogress\}$, and $\mathbb{Q}^j \subseteq \mathbb{Q}, 0 \leq j \leq \lfloor \frac{n}{2} - 2 \rfloor$, s.t. $(\bigcap_{\mathcal{Q} \in \mathbb{Q}^j \cup \{Q\}} \mathcal{Q}) \subseteq MS$.

Lower bounds

Theorem: No execution of safe register implementation that use an N-wise quorum system, contains more than $N-1$ consecutive, quorum shifting, fast writes.

Theorem: It is impossible to get MWMR safe register implementations that exploit an N-wise quorum system, if

$$|W \cup R| > N - 1$$

Remarks

Remark: SSO algorithm is near optimal since it allows up to $\left(\frac{N}{2} - 1\right)$ consecutive, quorum shifting, fast writes.

Problem: Communication vs Computation

<u>Theorem:</u> The read and write predicates are NP-complete. (provided a reduction from 3-SAT)