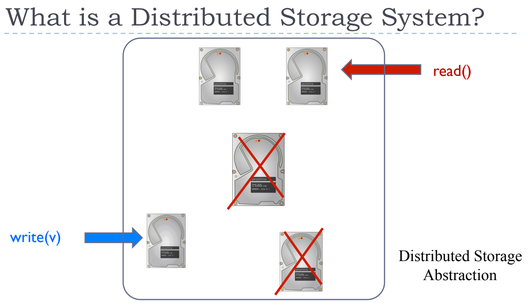
Seeking Fastness in Multiple-Writer Multiple-Reader Atomic Register Implementations

ΠΕΝΕΚ/0609/31

Chryssis Georgiou, Nicolas Nicolaou, Alexander Russell, Alexander Shvartsman

Problem - Background

What is a Distributed Storage System?



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

Traditional Implementations

Traditional SWMR
- Single round writes
- Two round reads
  - Phase 1: Obtain latest value
  - Phase 2: Propagate latest value
    - Folklore belief: "Reads 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 context

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

Computationally Demanding

Distributed Setting

Model

Environment:
- Asynchronous
- Message-Passing
- MWMR: $|W|\geq 1$, $|R|\geq 1$

Process sets:
- writers W,
- readers R,
- servers S (replica hosts)

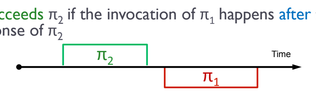
Failures: Crash Failures

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

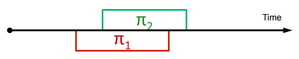
Definition: Operation Relations

- 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

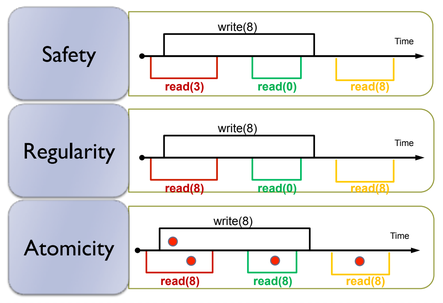
  - π_1 succeeds π_2 if the invocation of π_1 happens after the response of π_2



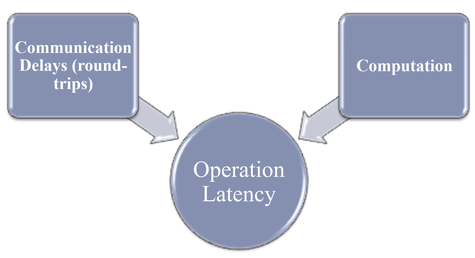
  - π_1 is concurrent with π_2 if π_1 neither precedes nor succeeds π_2



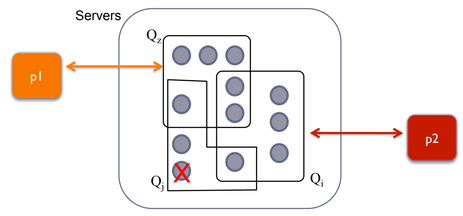
Consistency Semantics [Lamport86]



Complexity Measure



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

Motivation

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 (every n quorums intersect)
  - 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

Algorithm SFW

Key Idea

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

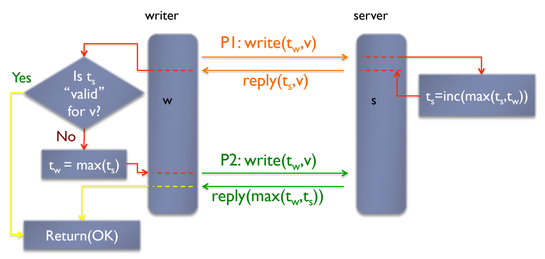
Algorithm: SFW

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

SSO Writer-Server Interaction



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

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

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

Algorithm SIMPLE

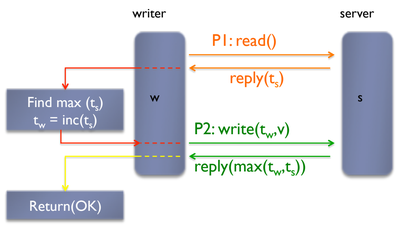
Algorithm: Simple

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>

SIMPLE Writer-Server Interaction



Algorithm APRX-SFW

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).

Algorithm Rationale

Let $T_m = \{R_{m,i} | Rm, i = (U \setminus M) \setminus (Q_i \setminus M)$ and $m \in M, Q_i\}$

If $\exists R_{m,1} \cup \ldots \cup R_{m,k} = U \setminus M$ then $\overline{R_{m,1}} \cap \ldots \cap \overline{R_{m,k}} = \emptyset$

Since $\overline{R_{m,i}} = (Q_i \setminus M)$ and $m \in Q_i$ then (De Morgan's)

$m \in Q_1 \cap \ldots \cap Q_k$ and $Q_1 \cap \ldots \cap Q_k \subseteq M$

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

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

Algorithm CWFR

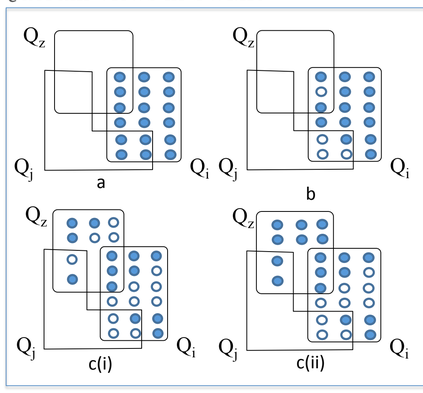
Algorithm: CWFR

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>

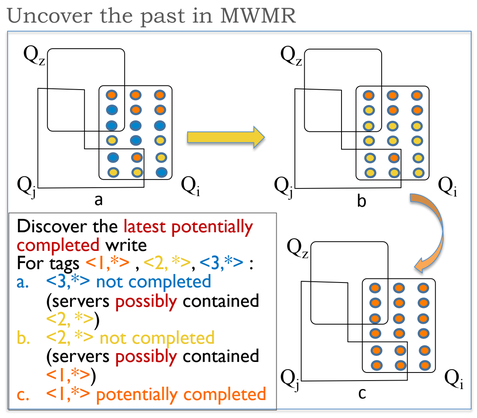
Quorum Views in SWMR



If a read operation receives replies from quorum Q_i will observe one of the following:
a. Full quorum contains max(tag) => Write potentially completed
b. Every quorum contains an old tag => Write of max(tag) not completed
c. Every server of an intersection replied with max(tag) => either write not completed c(i), or write completed in Q_z c(ii)

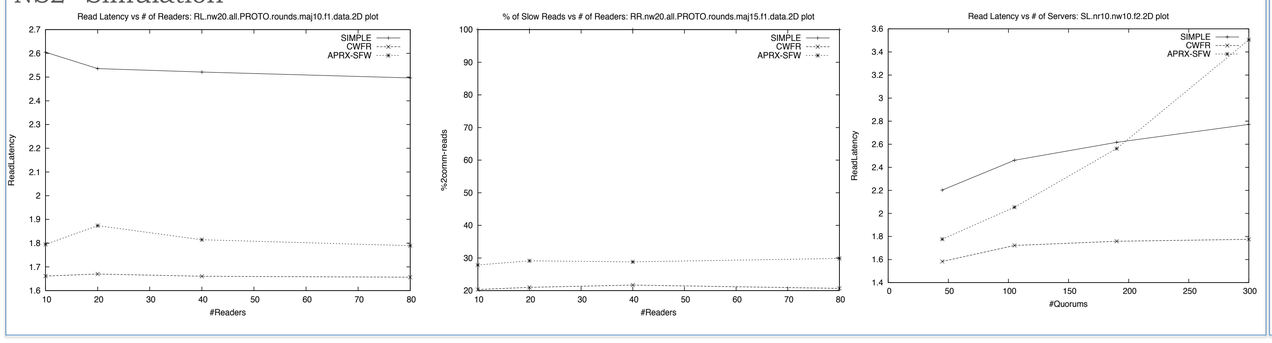
a,b = write status Determinable => FAST
c = write status Undeterminable => SLOW

Uncover the past in MWMR

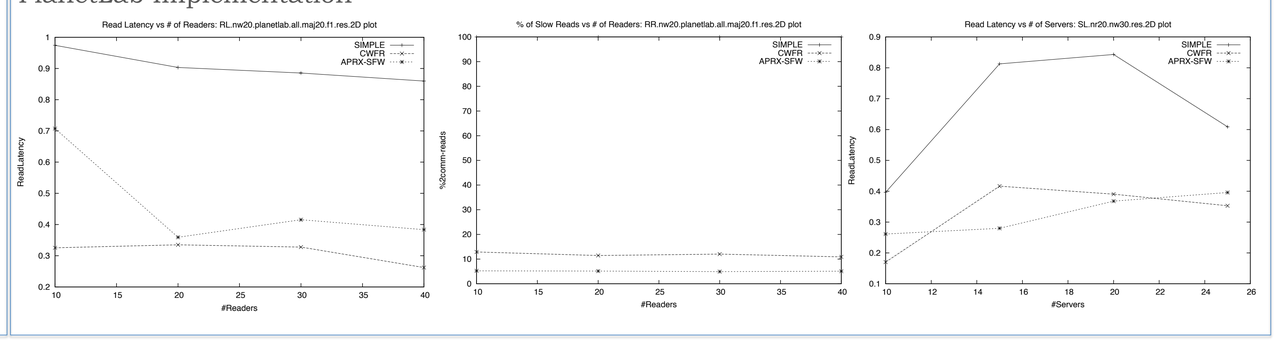


Discover the latest potentially completed write
For tags <1,*> , <2, *>, <3,*> :
a. <3,*> not completed (servers possibly contained <2, *>)
b. <2, *> not completed (servers possibly contained <1,*>)
c. <1,*> potentially completed

NS2- Simulation



PlanetLab Implementation



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

UCONN