

TOP 10 Reasons for the name "Bayou":

10. Why not?
9. It's better than "UbiData".
8. It's a lot better than "DocuData".
7. It's not an acronym.
6. It's not named after a soft drink (e.g. Tab, Sprite, Coda Cola, ...).
5. We're working on replication that's "fluid" like a bayou.
4. We're exploring a small part of the "UbiComp Swamp".
3. It's the name of a famous tapestry (spelled "Bayeux" however).
2. Our system will allow you to access data even when you're "bayou self".
1. It's pronounced "Bi-U", which makes it "Ubi" pronounced backwards.

Introduction

- Detailed presentation of Bayou's anti-entropy protocol for replica reconciliation
- It is based on:
 - pair wise communication
 - the propagation of write operations
 - a set of ordering and closure constraint on the write propagation

Introduction

- The goal is to support:
 - for arbitrary communication topologies
 - operation over low-bandwidth networks
 - incremental progress
 - eventual consistency
 - efficient storage management
 - propagation through transportable media
 - light-weight management of dynamic replica sets
 - arbitrary policy choices

Basic Anti-entropy

- To bring two replica up-to-date
- A replica: a ordered log of writes, a database
- A write: a set of updates, a dependency check, a merge procedure, accept-stamp
- Accept-stamps define a total order at a server and partial order over all writes in the system

Basic Anti-entropy

- Protocol:
 - between pairs of servers
 - the propagation of writes:
 - constrained by the accept-order
- **prefix property:** a server R that holds a write stamped W_i that was initially accepted by another server X will also hold all writes accepted by X prior to W_i .

Basic Anti-entropy

- Protocol:

```
anti-entropy(S,R) {  
    Get R.V from receiving server R  
    # now send all the writes unknown to R  
    w = first write in S.write-log  
    WHILE (w) DO  
        IF R.V(w.server-id) < w.accept-stamp  
        THEN  
            # w is new for R  
            SendWrite(R, w)  
            w = next write in S.write-log  
        END  
    }
```

- supports:

- variety communication topologies
- variety of policy choices for when, with whom
- low bandwidth networks
- incremental progress

Effective Write-log Management

- Storage is the concern
- Prune the prefix of write logs
- Primary-commit protocol to stabilize writes
- Primary replica commits write and assigns a monotonically increasing commit seq #
- Committed writes are totally ordered
- Propagation:
 - first send the committed writes
 - then send the tentative writes

Effective Write-log Management

- Write log truncation
- If the sender's omitted seq # is larger than receiver's commit seq #
- then there exist committed writes:
 - sender has truncated from its log
 - receiver has not seen
- a full database transfer is required
- challenge: to reduce write log size while keeping the chance of full database transfer low

Anti-entropy Protocol Extensions

- Through Transportable Media
 - off-line algorithm
- Session guarantees and eventual consistency
 - a causal order to provide session guarantees to applications
 - a total order to ensure eventual consistency of all replicas

Causal-accept order

- Any write A precedes another write B if and only if, at the time write B was accepted by some server from a client, write A was already known to that server.
- logical clock: advances both
 - when a new write is accepted from a client
 - when a write with higher accept stamp is received from another server

Light weight server creation and retirement

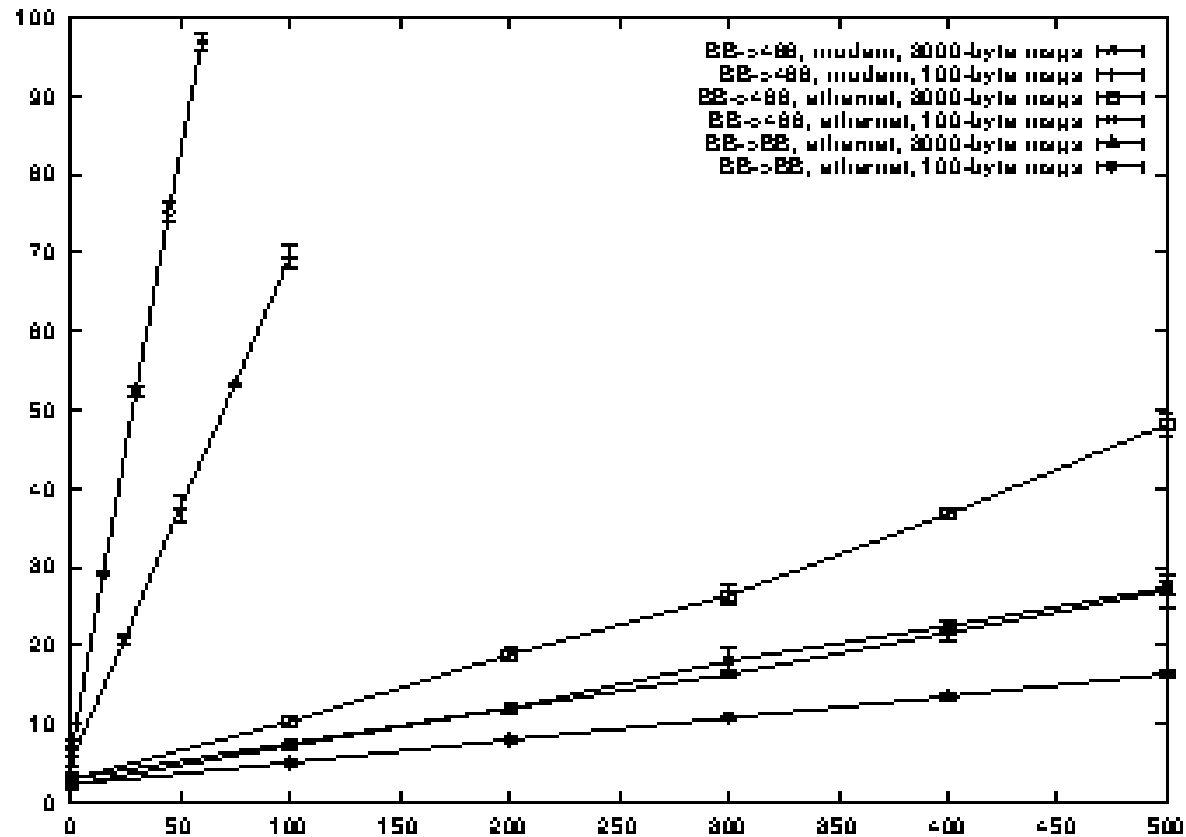
- A new server S_i can be created by communicating with any available server S_k
 - S_i sends a write to S_k
 - S_k insert the write into its log $\langle \text{infinity}, T_{k,i}, S_k \rangle$
 - $\langle T_{k,i}, S_k \rangle$ is S_i 's server id.
- A server S_i can retire
 - S_i issues a retirement write to itself
 - wait until performing anti-entropy with at least one server

Anti entropy Policies

- Policies for when to reconcile
 - periodic, manually triggered, system triggered recon.
- Policies for choosing with which replicas
 - availability, connection speed, up-to-dateness
- Policies for deciding how aggressively to truncate the write log
 - trade off storage and network resources during recon.
- Policies for selecting a server from which to create a new replica
 - server id length effects the performance

Performance

- Linear function of the number of writes and the available network bandwidth



Flexible Update Propagation for Weakly Consistent Replication

Peterson, Spreitzer, Terry, Theimer, Demers

Presented by Alper MIZRAK

Protocol overhead

- Network transfer cost is dominant
- Anti entropy algorithm
 - compares the version numbers
 - transmits the write log

Effect of Server Creation Patterns

- Version vector storage requirements grow between linearly and quadratically with the number of replicas depending on the creation pattern

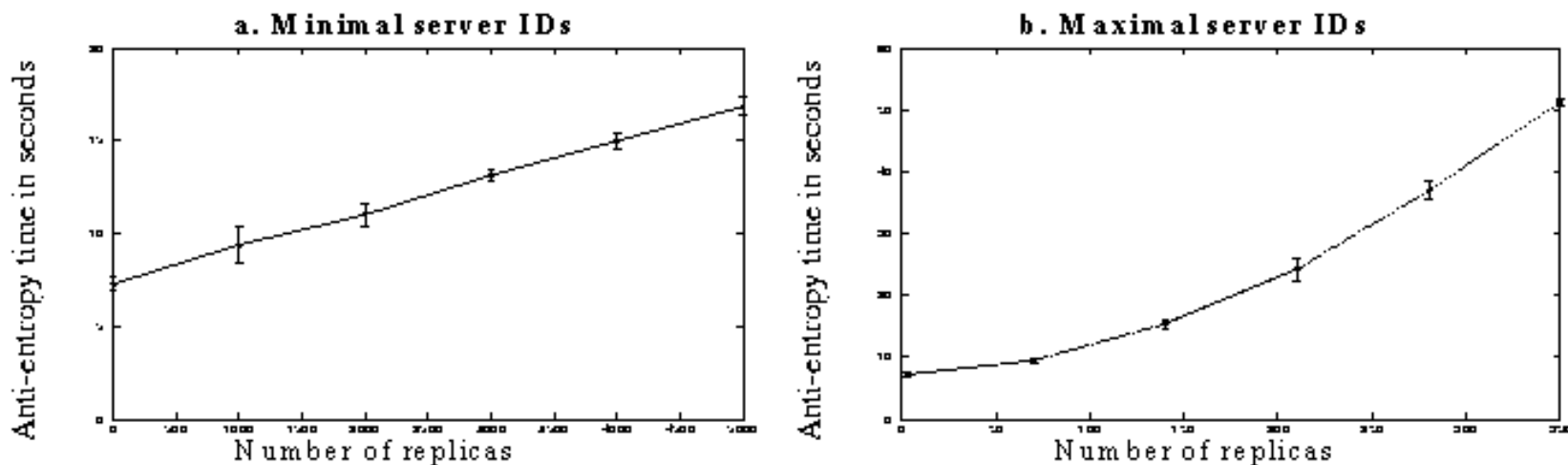


Figure 8. Anti-entropy execution time for 100 writes as a function of the number of replicas

Questions

- How reasonable is the primary commit protocol for the weakly consistent replication?
- What are the differences between Coda and Bayou?
 - design goals
 - implementation details