Flexible Update Propagation for Weakly Consistent Replication

By Karin Peterson, Mike J. Spreitzer, Douglas B. Terry, Marvin M. Theimer, and Alan J. Demers

Written in 1997

Presented by Paul Baranowski

1

Introduction

What is the goal of replication?

What are some examples where replication is useful or necessary?

2

Background

Previously developed systems for data replication:

- Coda
- Ficus
- Oracle 7

3

Features of Anti-entropy

- Support for arbitrary communication topologies
- Operation over low-bandwidth networks
- Incremental progress
- Eventual consistency

4

Features of Anti-entropy continued

- Efficient storage management
- Propagation through transportable media
- Light-weight management of dynamic replica sets
- Arbitrary policy choices

5

Outline of Presentation

- Basic Anti-entropy
- Write-log Management
- Server creation and retirement
- Design Choices and their implications
- Disadvantages
- Anti-entropy policies
- Experimental results

6

Basic Anti-entropy

The goal of anti-entropy is for two replicas to bring each other up-to-date.

7

Prefix Property

Anti-entropy maintains the 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).

8

Prefix Property

R

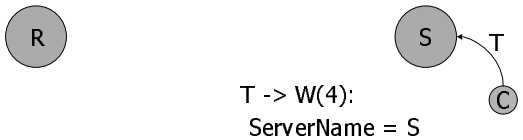
X

If R has received W(5) from X we also know that R has received W(1), W(2), W(3), and W(4)

X's write log: W(1), W(2), W(3), W(4), W(5), W(6), W(7), W(8)

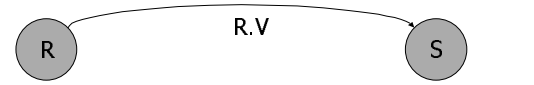
9

Anti-entropy in Action



10

Anti-entropy in Action



Write log for server R(R.V):
Server X: (W(10))
Server S: (W(2))
Server R: (W(24), W(25))

Write log for server S(S.V):
Server X: (W(10), W(11))
Server S: (W(2), W(3), W(4))
Server R: (W(24))

X:W(11), S:W(3), S:W(4)

11

Write-log Management

- Problem: How do we determine which writes have fully propagated to all replicas?
- Past solutions
- Bayou: each replica independently decides when and how aggressively to prune, subject to the constraint that only stable writes are discarded.

12

Write Stability

- Definition of *stable write*:

  A write whose position in the write-log will not change and hence never needs to be re-executed at that server.

13

Propagation of Committed Writes

- When updating, a server will notify the receiver of any committed writes.

14

Write-log Truncation

- Since a server can truncate its write-log, it is possible that it may not be able to incrementally reconcile with another replica.
- This can occur when a server has truncated its writes, but a receiver did not know about those writes.

15

Light-weight Server Creation

- A server creates itself by sending a creation write to another server S_k.
- S_k inserts this into its write log as <infinity, $T_{k,i}$, S_k>, where T_k is the accept-stamp assigned by S_k.
- <$T_{k,I}$, S_k> becomes the new server-id.

16

Server Retirement

- When a server wants to stop acting as a database, it issues a retirement write to itself.
- The server must perform anti-entropy with another server before it takes itself offline.

17

Design Choices

18

Disadvantages of Anti-entropy

- The drawbacks for anti-entropy are associated with the size of the version vectors and the write-log.

19

Disadvantages of Anti-entropy

- The size of the vectors grows in proportion to the number of replicas and the complexity of the replica creation pattern.
- A server must keep all writes it has received until it is notified of the writes' commitment.

20

Anti-Entropy Policies: When to Reconcile?

- Increasing the frequency of anti-entropy increases the rate of update propagation, the degree to which servers are up to date, and the rate at which writes stabilize.
- However, this is at the expense of greater bandwidth consumption.

21

Anti-entropy Policies: Reconcile with which server?

- Tradeoff between bandwidth usage and propagation rates.
- The key is to favor nearby servers and avoid overloading slow network links.

22

Anti-entropy Policies: Write-log Truncation

- There is a storage/bandwidth tradeoff based on how aggressively replicas prune their logs and how frequently replicas perform anti-entropy.

23

Anti-entropy Policies: Create a replica from which server?

- A new server creating itself should choose a server based on these factors:
  - server-id length
  - how up-to-date the logs are
  - how complete the logs are
  - connection bandwidth

24

Experimental Results

- The bulk of the anti-entropy algorithm execution time is spent on the network and applying the newly received writes to the write-log and database of the receiver.
- Version vector storage grows between linearly and quadratically with the number of replicas.

25

Experimental Results

- Checking whether a write is covered by a version vector takes time between linear and cubic to the number of servers, depending on how the servers are created.

26

Anti-entropy Execution as a Function of the Number of Writes Propagated

27

Anti-entropy Execution Time Breakdown for the Propagation of 100 Writes

28

Network Independent Anti-entropy Algorithm Components for the Propagation of 100 Writes

29

Anti-entropy Execution Time for 100 Writes as a Function of the Number of Replicas

30

Were the goals achieved?

- Support for arbitrary communication topologies
- Operation over low-bandwidth networks
- Incremental progress
- Eventual consistency

31

Were the goals achieved?

- (Sometimes) Efficient storage management
- Propagation through transportable media
- Light-weight management of dynamic replica sets
- Arbitrary (and complex) policy choices

32

Conclusion

- Anti-entropy provides the flexibility necessary to cope with a diversity of networking environments such as unreliable wireless networks, modems, and the Internet.

33