Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System

Authors: Terry et al.

Presented by Yitong Wang

Introduction

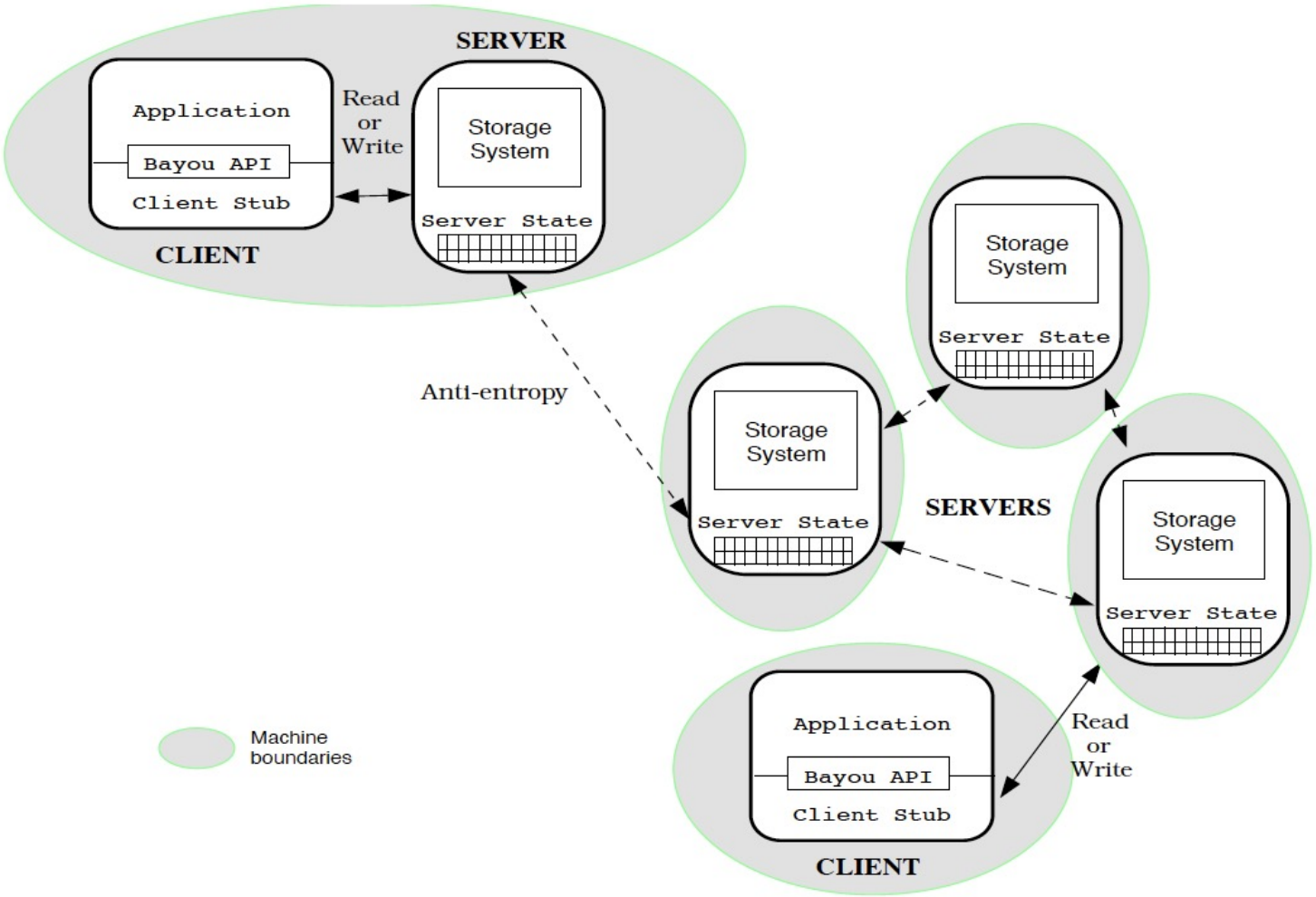
Bayou assumes

- A weak connectivity network model

Bayou supports

- Weakly consistent, replicated data
- Eventual data consistency
- Read-any/Write-any access for clients
- Application-specific conflict resolution

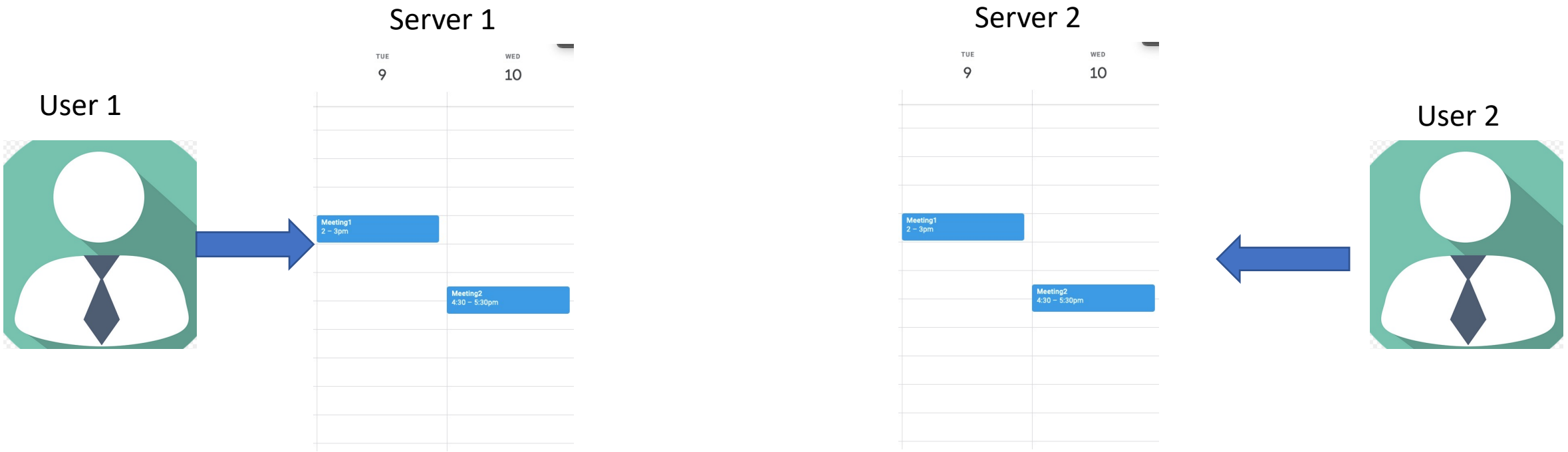
System Model



- Data Collection is replicated in full in servers.
- Clients access the service through Bayou API to Read/Write.
- Pair-wise anti-entropy session to synchronize operations.

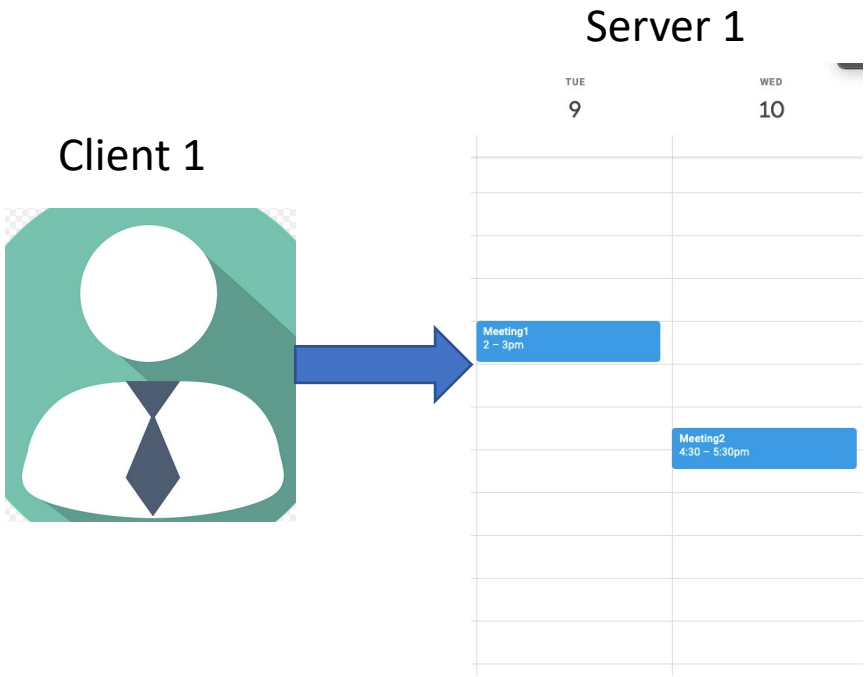
Example Application

Meeting Room Scheduler



Example Application

Meeting Room Scheduler



- Users' view may be outdated
- Reservation should be at first tentative, which may be accepted or rejected later.
- Records will eventually be synchronized

Conflict Detection and Resolution

Conflicts cannot be concluded by simply observing read/write operations from application

Conflict Detection and Resolution

Conflicts cannot be concluded by simply observing read/write operations from application.

Solution: Application-specific *dependency check* and *merge procedures*

Conflict Detection and Resolution

```
Bayou_Write (update, dependency_check, mergeproc) {
    IF (DB_Eval (dependency_check.query) <> dependency_check.expected_result)
        resolved_update = Interpret (mergeproc);
    ELSE
        resolved_update = update;
    DB_Apply (resolved_update);
}
```

Dependency check:

- Compares results of queries with expected results
- Works as a precondition for update
- Detects not only write-write but also read-write conflicts

Concrete Example

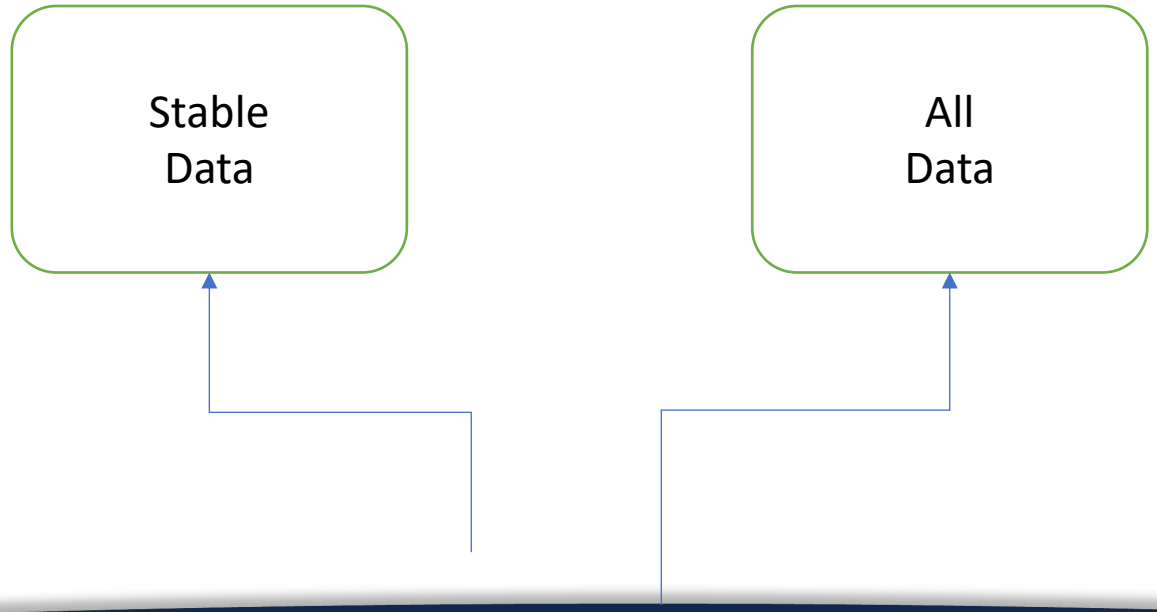
```
Bayou_Write(
    update = {insert, Meetings, 12/18/95, 1:30pm, 60min, "Budget Meeting"},
    dependency_check = {
        query = "SELECT key FROM Meetings WHERE day = 12/18/95
            AND start < 2:30pm AND end > 1:30pm",
        expected_result = EMPTY},
    mergeproc = {
        alternates = {{12/18/95, 3:00pm}, {12/19/95, 9:30am}};
        newupdate = {};
        FOREACH a IN alternates {
            # check if there would be a conflict
            IF (NOT EMPTY (
                SELECT key FROM Meetings WHERE day = a.date
                 AND start < a.time + 60min AND end > a.time))
                    CONTINUE;
            # no conflict, can schedule meeting at that time
            newupdate = {insert, Meetings, a.date, a.time, 60min, "Budget Meeting"};
            BREAK;
        }
        IF (newupdate = {})   # no alternate is acceptable
            newupdate = {insert, ErrorLog, 12/18/95, 1:30pm, 60min, "Budget Meeting"};
        RETURN newupdate;}
)
```

Write Stability and Commitment

Definition:

A write is *stable* or *committed* if it's executed for the last time.

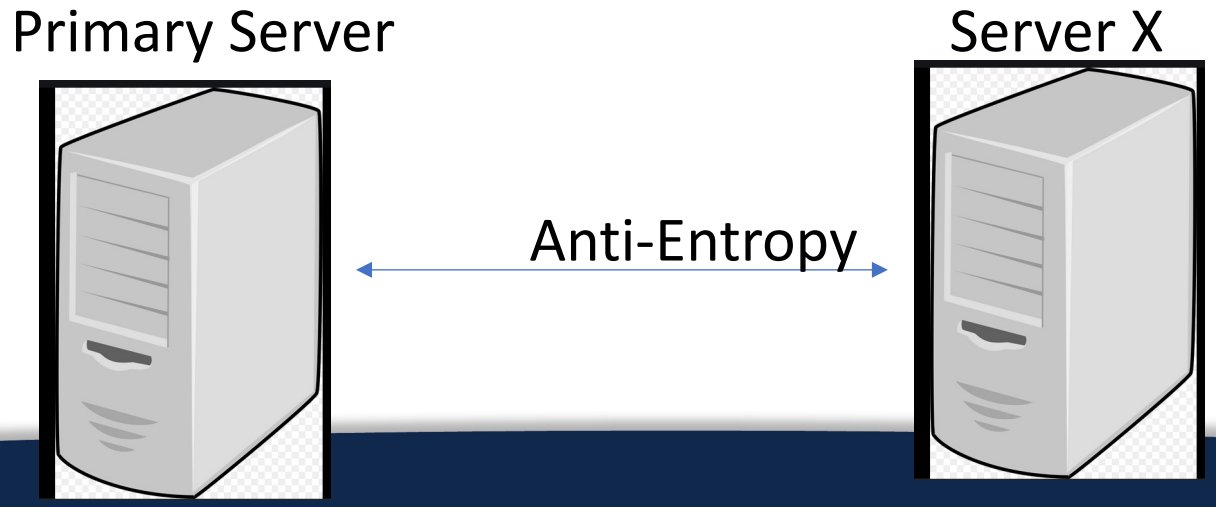
Bayou allows accessing both stable and complete data
(use <timestamp, server ID> to identify)



Write Stability and Commitment

Bayou uses primary commit scheme:

- A primary server determines commit and propagates relevant knowledge.
- Bayou inherently accommodate temporary unavailability of primary
- Writes may not be committed in the order of when they are received



(Eventual) Replica Consistency

To support this, Bayou ensures

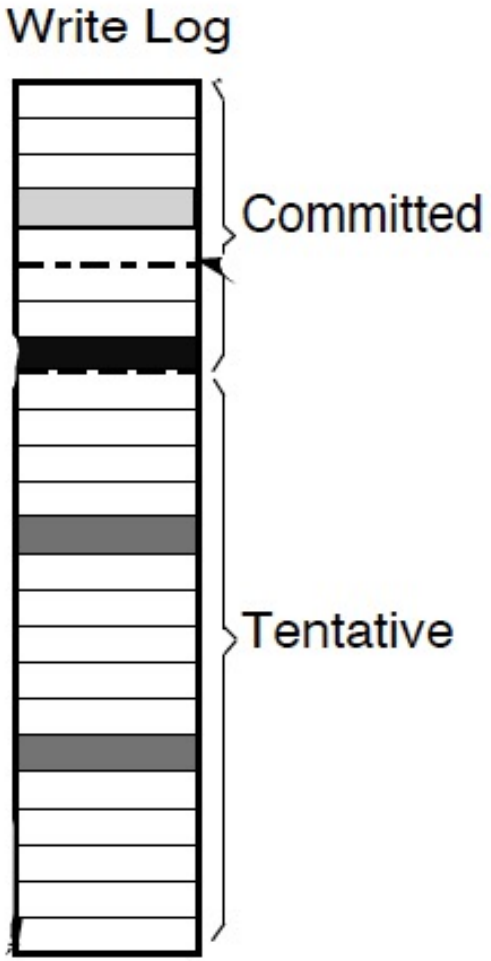
1. Writes are performed in a globally well-define order

2. Conflict detection and merge procedures are deterministic

(Eventual) Replica Consistency

To support this, Bayou ensures
1. Writes are performed in a well-define order

- Tentative writes ordered by timestamp
- Committed writes ordered by time and before tentative ones
- Need ability to undo write



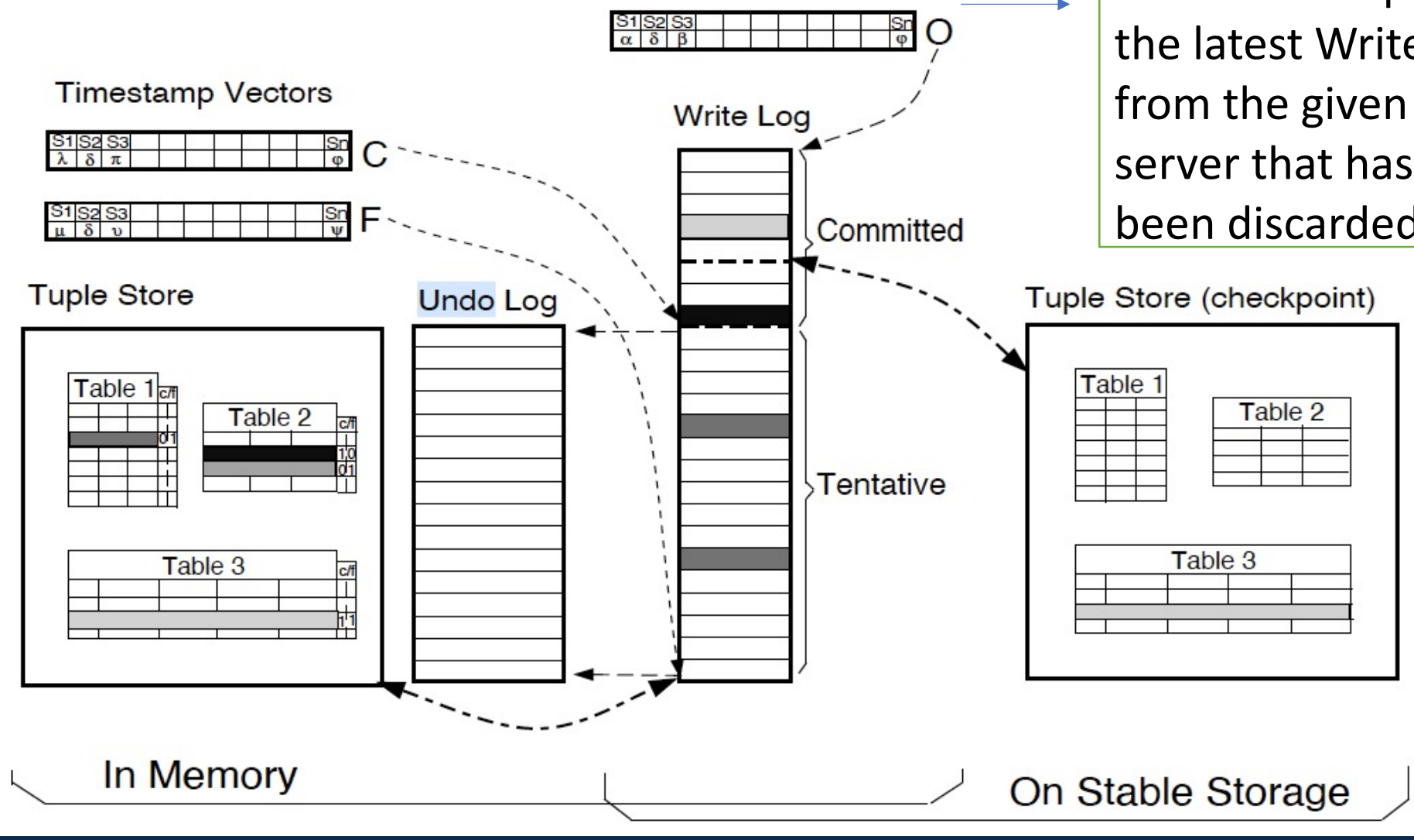
(Eventual) Replica Consistency

To support this, Bayou ensures
2. Conflict detection and merge procedures are deterministic

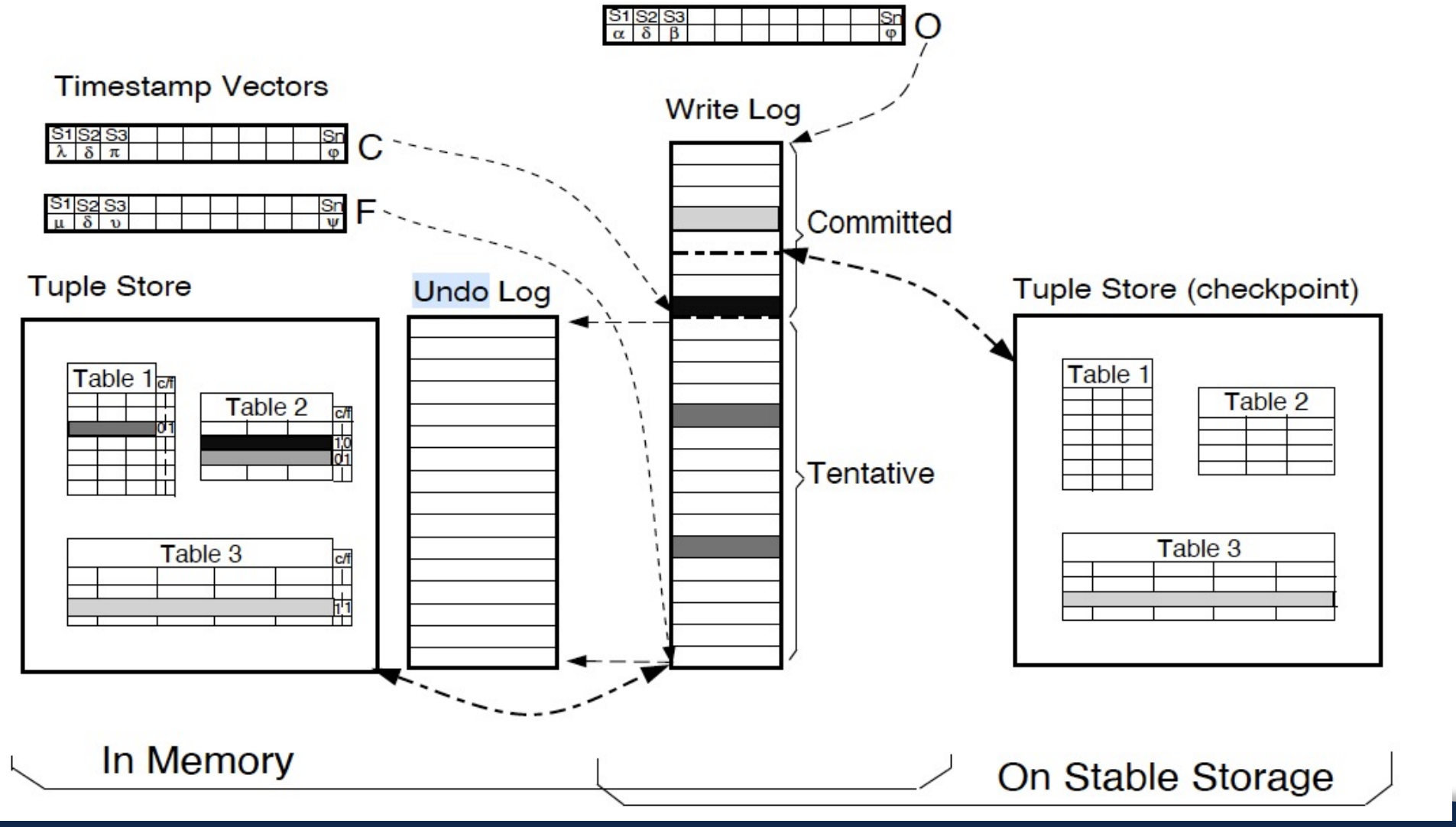
- Procedures cannot access time-dependent or machine-specific info
- Computation resources such as CPU and memory are bounded identically

Storage System Implementation

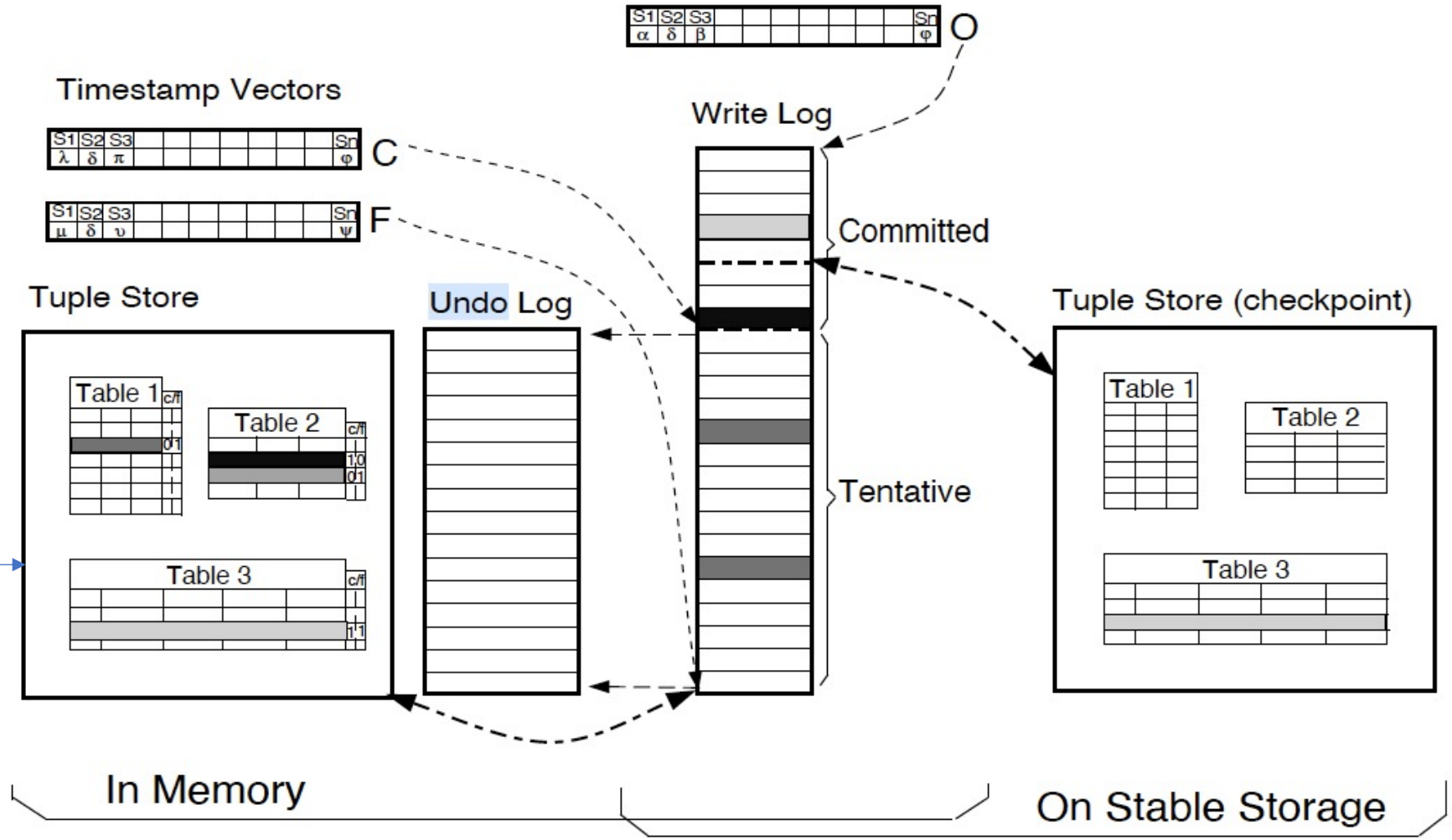
the timestamp of the latest Write from the given server that has been discarded



Storage System Implementation



Storage System Implementation



2 bits to indicate committed or full

Evaluation

Table 1: Size of Bayou Storage System for the Bibliographic Database with 1550 Entries
(sizes in Kilobytes)

Number of Tentative Writes	0 (none)	50	100	500	1550 (all)
Write Log	9	129	259	1302	4028
Tuple Store Ckpt	396	384	371	269	1
Total	**405**	**513**	**630**	**1571**	**4029**
Factor to 368K bibtex source	1.1	1.39	1.71	4.27	10.95

Evaluation

Table 2: Performance of the Bayou Storage System for Operations on Tentative Writes in the Write Log
(times in milliseconds with standard deviations in parentheses)

Tentative Writes	0	50		100		500		1550	
	Server running on a Sun SPARC/20 with Sunos								
Undo all	0	31	(6)	70	(20)	330	(155)	866	(195)
(avg. per Write)		.62		.7		.66		.56	
Redo all	0	237	(85)	611	(302)	2796	(830)	7838	(1094)
(avg. per Write)		4.74		6.11		5.59		5.05	
	Server running on a Gateway Liberty Laptop with Linux								
Undo all	0	47	(3)	104	(7)	482	(15)	1288	(62)
(avg. per Write)		.94		1.04		.96		.83	
Redo all	0	302	(91)	705	(134)	3504	(264)	9920	(294)
(avg. per Write)		6.04		7.05		7.01		6.4	

Evaluation

Table 3: Performance of the Bayou Client Operations
(times in milliseconds with standard deviations in parentheses)

Server Client	Sun SPARC/20 same as server		Gateway Liberty same as server		Sun SPARC/20 Gateway Liberty	
Read: 1 tuple	27	(19)	38	(5)	23	(4)
100 tuples	206	(20)	358	(28)	244	(10)
Write: no conflict	159	(32)	212	(29)	177	(22)
with conflict	207	(37)	372	(17)	223	(40)

Conclusion

In a weakly connected network, Bayou

- Can achieve eventual consistency
- Uses tentative and stable writes
- Supports application-specific confliction detection