Just-Right Consistency!

Valter Balegas, Nuno Preguiça, Marc Shapiro, Annette Bieniusa, Christopher S. Meiklejohn, Carla Ferreira & many others



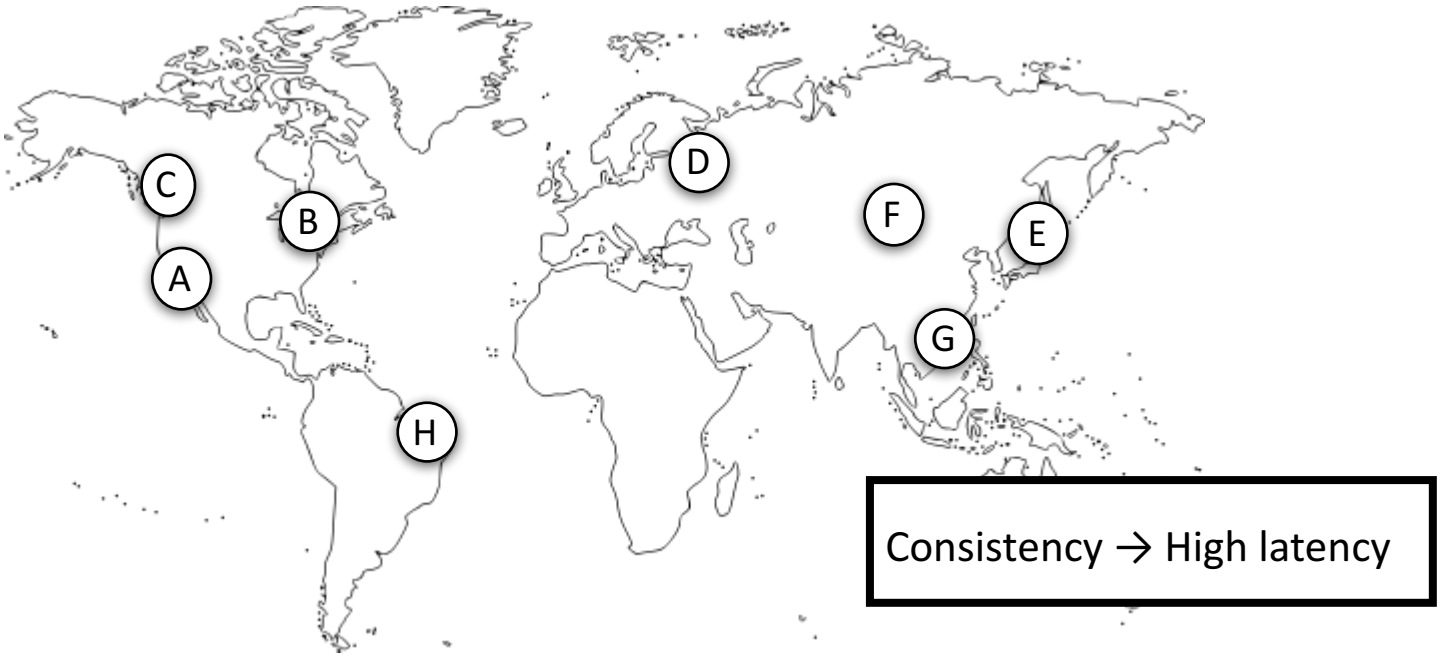

NOVALINCS
LABORATORY FOR COMPUTER
SCIENCE AND INFORMATICS

Agenda

- Geo-replication;
- Consistency/availability trade-off;
- Just-Right Consistency!
- … and how to use it.

Geo-replication

Geo-Replication



Consistency/availability trade-off

- Weak consistency is difficult to get right:
  - Not well-defined semantics;
  - Difficult to program;
  - Unpredictable errors in production due to uncoordinated executions.

- Strong consistency performs worse, but is safe:
  - Well-defined semantics;
  - Application's logic is protected from concurrency errors;
  - Scalable and low-latency in some deployments;
  - Cross-replica coordination affects performance in the wide-area.

Consistency/availability trade-off

- What we really care about is application correctness.
- Serializing operations is a (overly conservative) way of maintaining correctness.
- Example:
  - Operations on a bank account can be applied concurrently as long as money does not disappear, get duplicated, or used without permission.

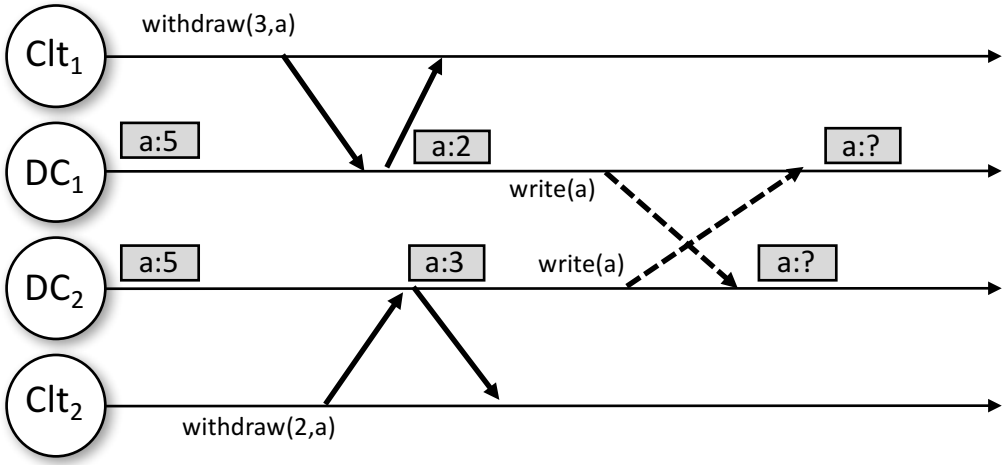
Just-Right Consistency!

- Ensure application correctness with minimal coordination.
- How to do it?

Banking Application:

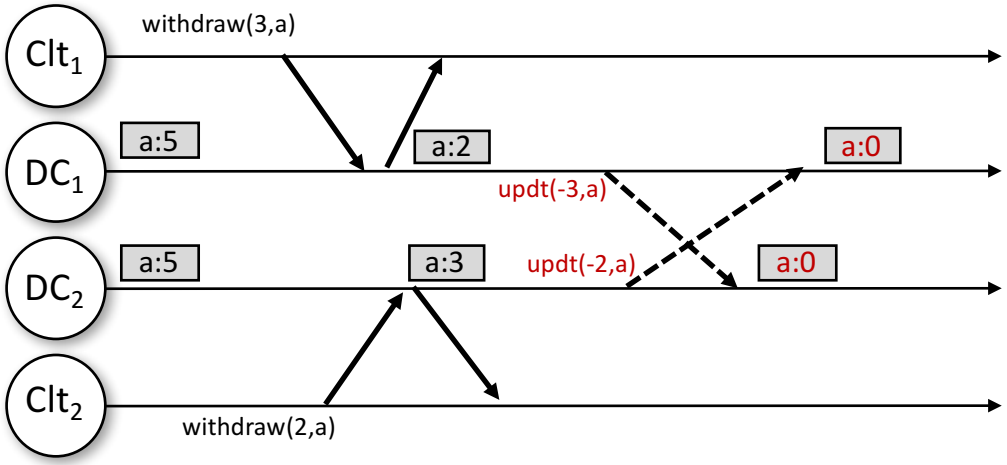
- All replicas must converge to the same state.
- Account balance is equal to $init_{balance} + \sum deposit() - \sum withdrawals()$;
- The money cannot temporarily disappear in a transfer.
- Account balance must be non-negative.

Problem 1: State divergence



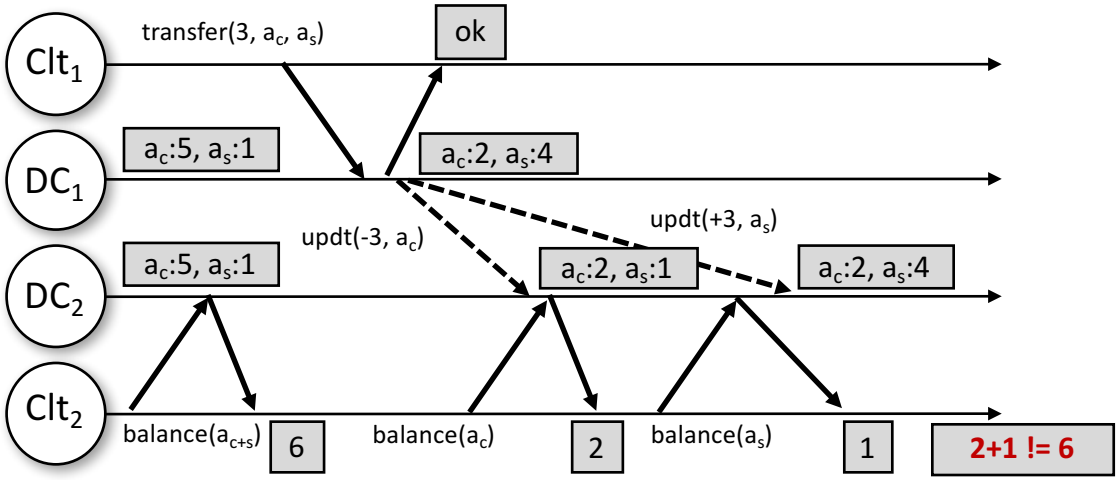
- Concurrent updates lead to state divergence.

Replicated data types



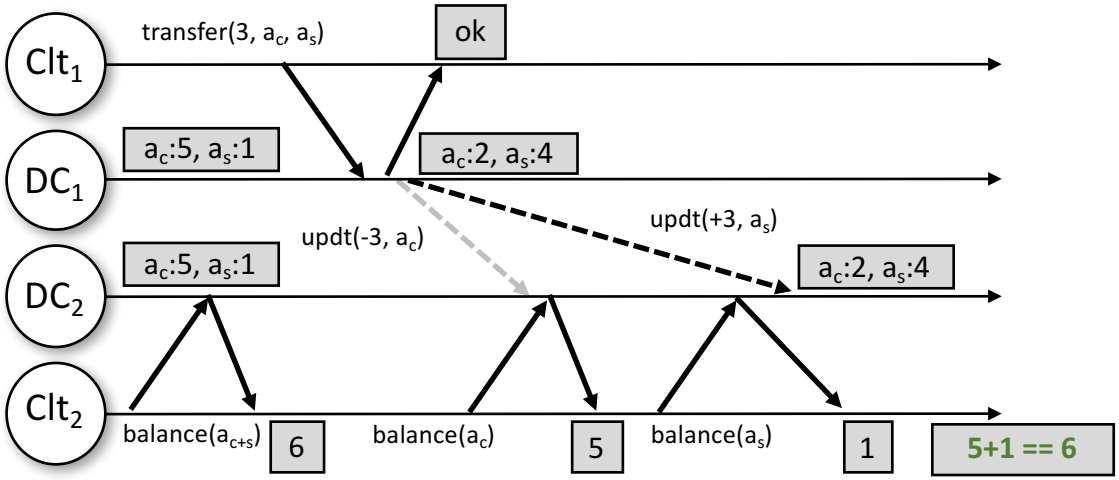
- Ensure state convergence without losing updates.

Problem 2: Atomic operations



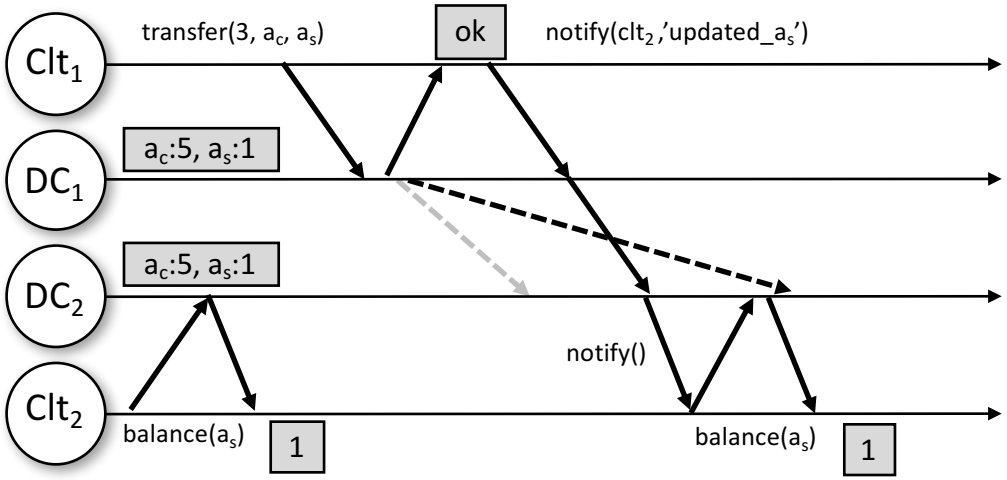
- ACID transactions are not highly available.

Transactional consistency



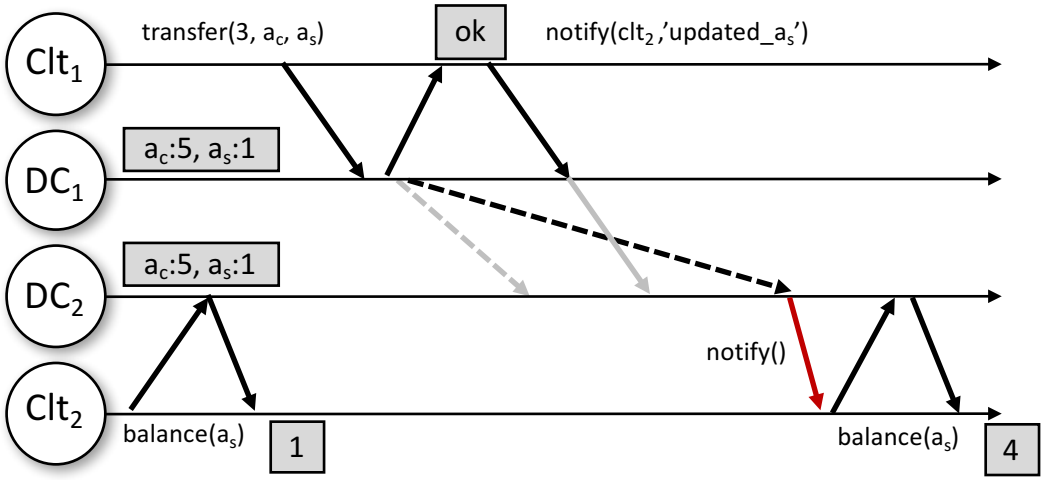
- Ensure atomicity with weaker isolation.
  - Snapshot reads; atomic updates.

Problem 3: Asynchronous replication



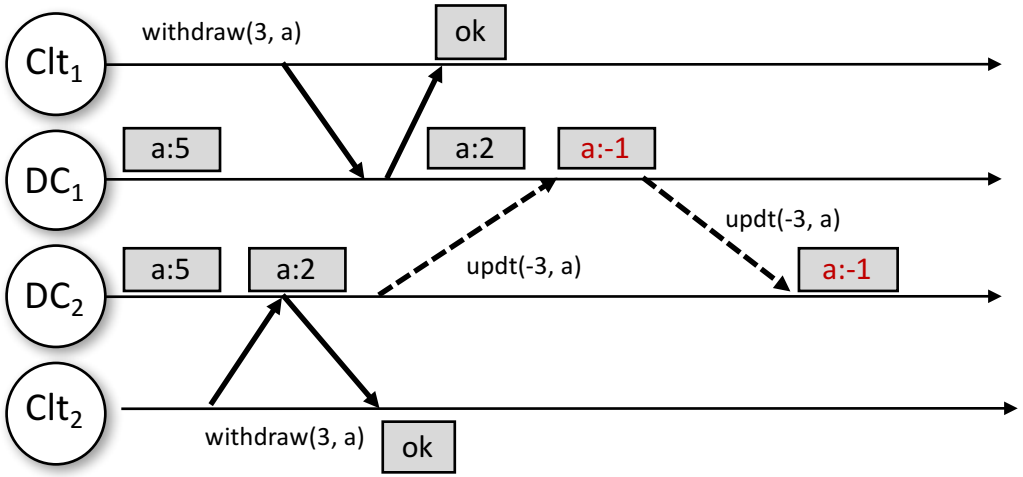
- Does not preserve the execution order of operations at the origin.

Causal consistency



- Execution order of operations at different replicas respect the happens-before relationship.

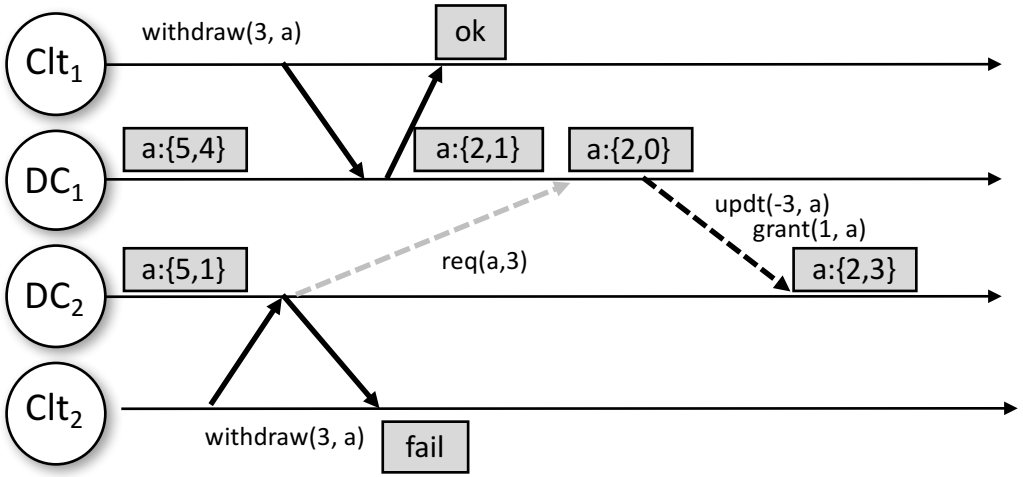
Problem 4: Invariant violation



- Concurrent executions might break the correctness of applications.

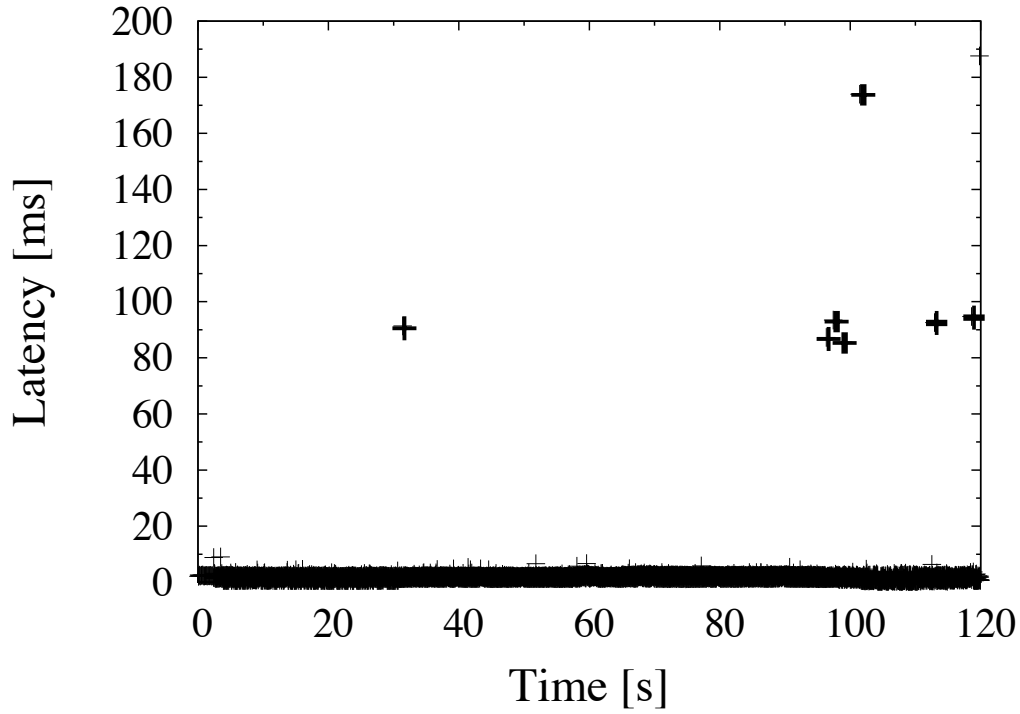
  → Use coordination?

Minimize coordination



- Use coordination only when not safe to execute operations asynchronously.

Performance



Latency of operations for a single site, in a 3DC deployment.

Other application Invariants

- Bidirectional relationship → use transactions;
- Referential integrity → automatic repair;
- Overdraft → compensations;
- Sequential identifiers → fall back to coordination.

Just-Right Consistency!

- Ensure application correctness with minimal coordination.

- How to do it?
  - Convergent data-types;
  - Transactions;
  - Causality;
  - Maintain application-level invariants.

JRC! Tools

- Help programmers verify application correctness using a sound approach:
  - Static program analysis.
  - Detect concurrency conflicts.
  - Modify applications and test again.
- Publicly-available tools on the way.

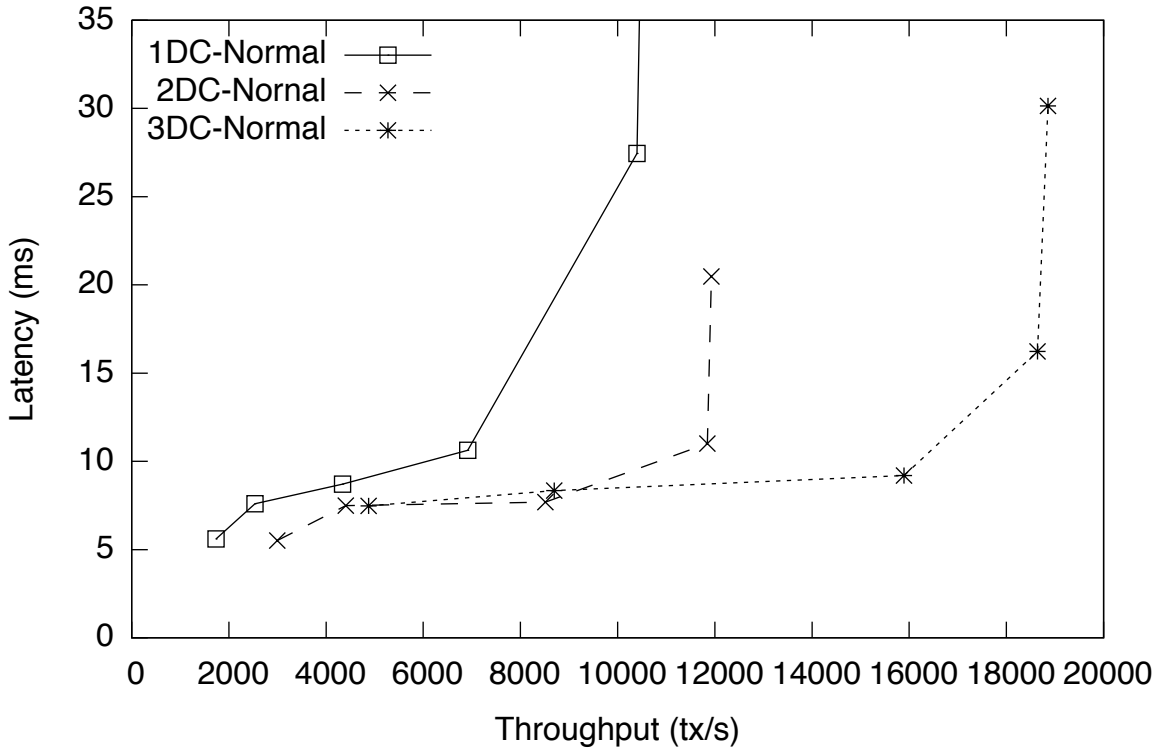
JRC Tools video:
http://tiny.cc/JRC-TOOLS-VIDEO

AntidoteDB

http://syncfree.github.io/antidote/

- Just-Right Consistency database.
- Geo-/Partial-replication.
- SQL-like interface on the way:
  - Well-known and widely adopted;
  - Out-of-the-box support for maintaining many common classes of invariants.

AntidoteDB performance in FMKe Benchmark



Latency of operations for multiple sites with 4 nodes per site.

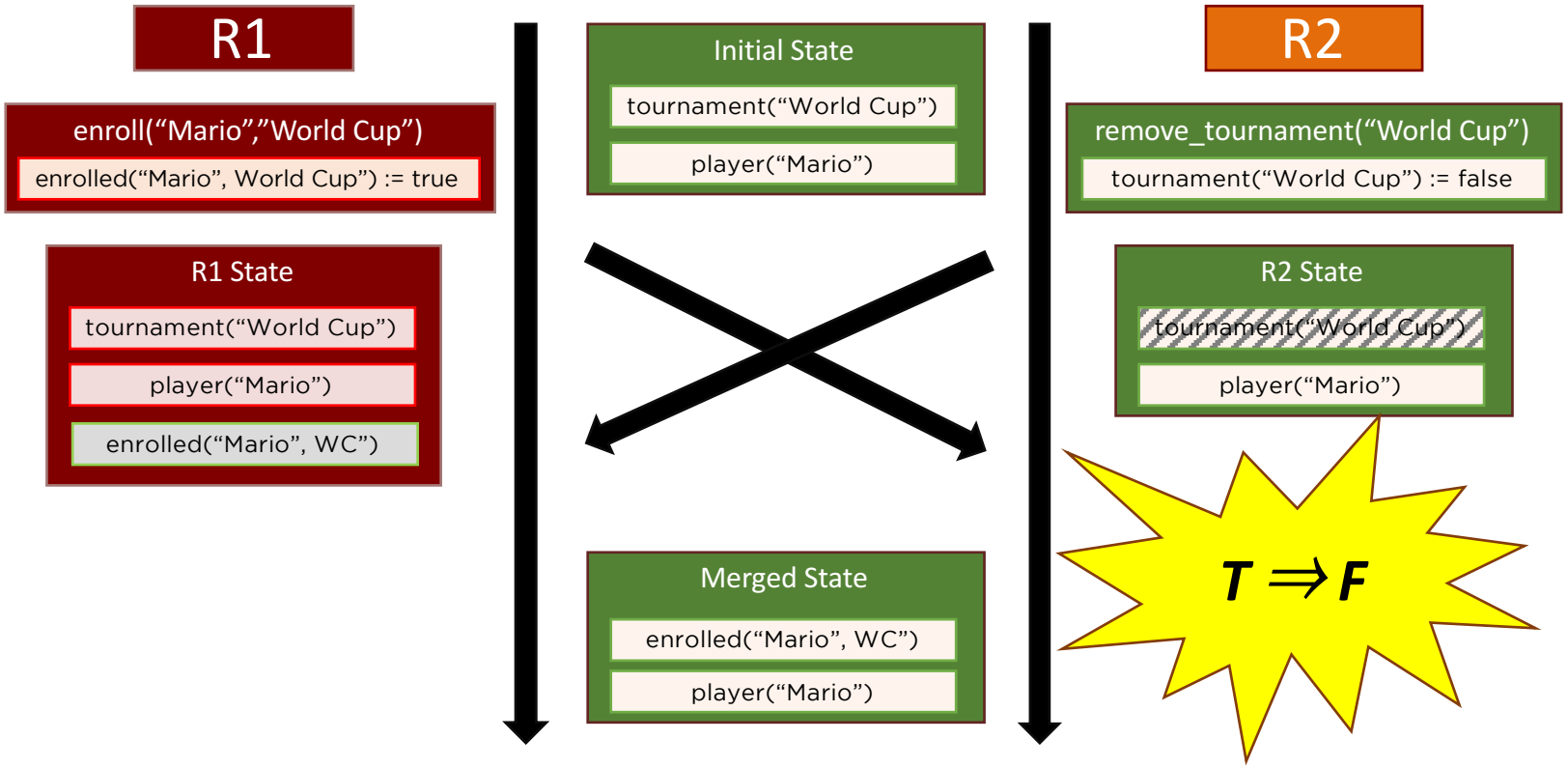
Questions?

- JRC Tools video:
  http://tiny.cc/JRC-TOOLS-VIDEO
- AntidoteDB:
  http://syncfree.github.io/antidote/
- FMKe Benchmark
  - http://tiny.cc/fmke

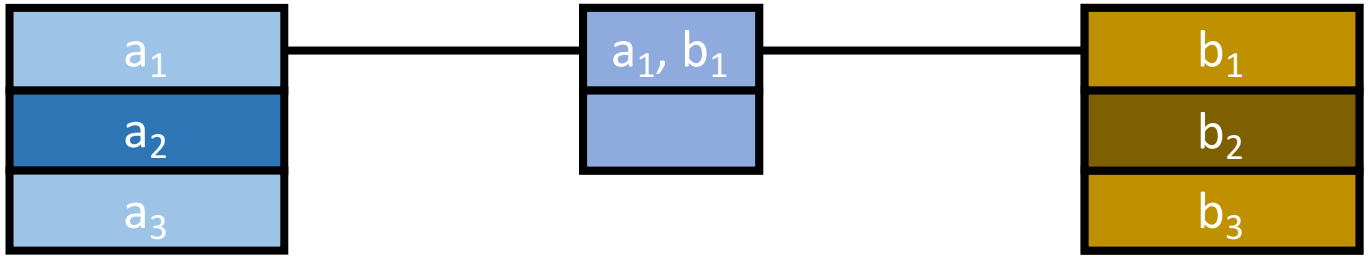
Backup slides

Conflict detection algorithm

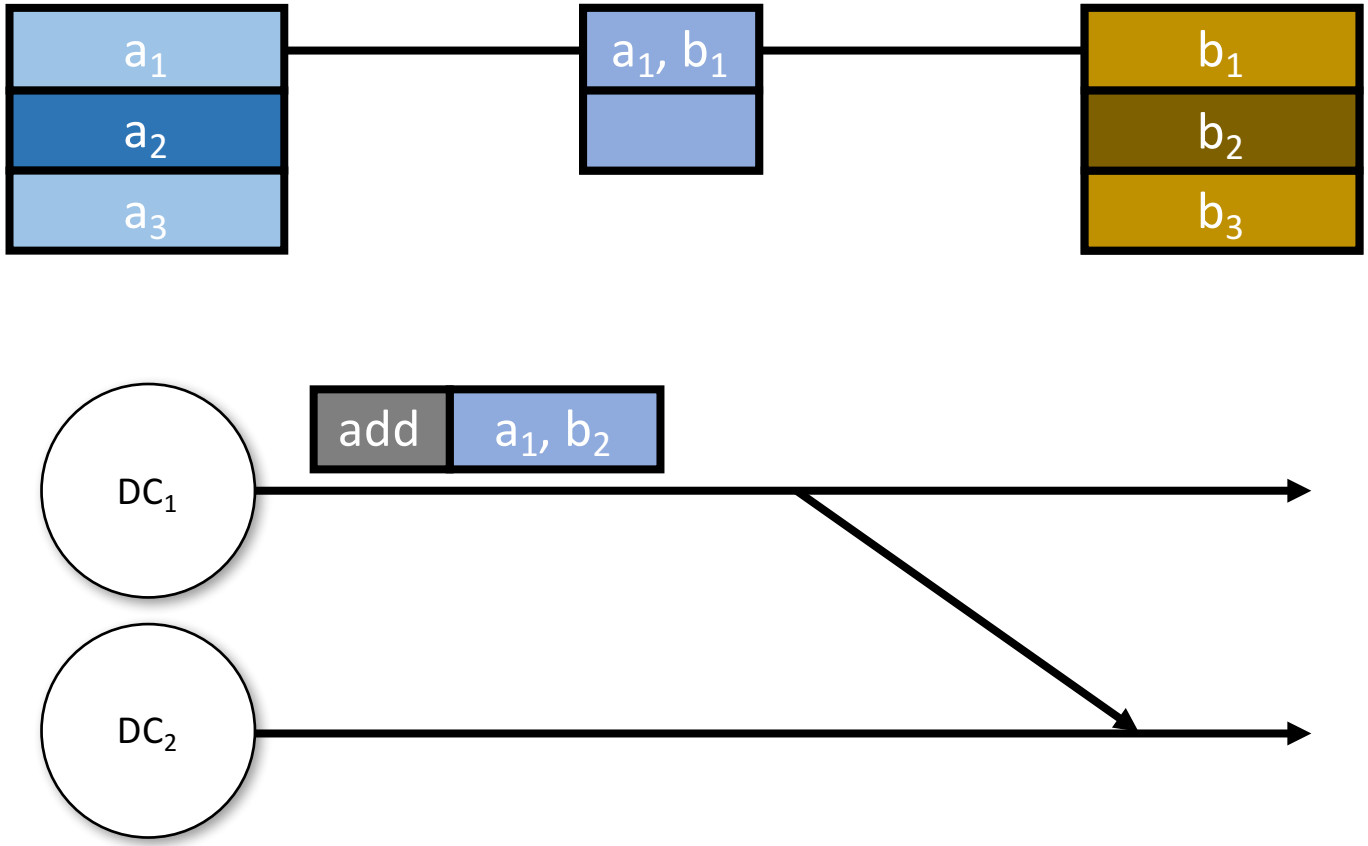
$$INV = enrolled(p, t) \Rightarrow player(p) \wedge tournament(t)$$



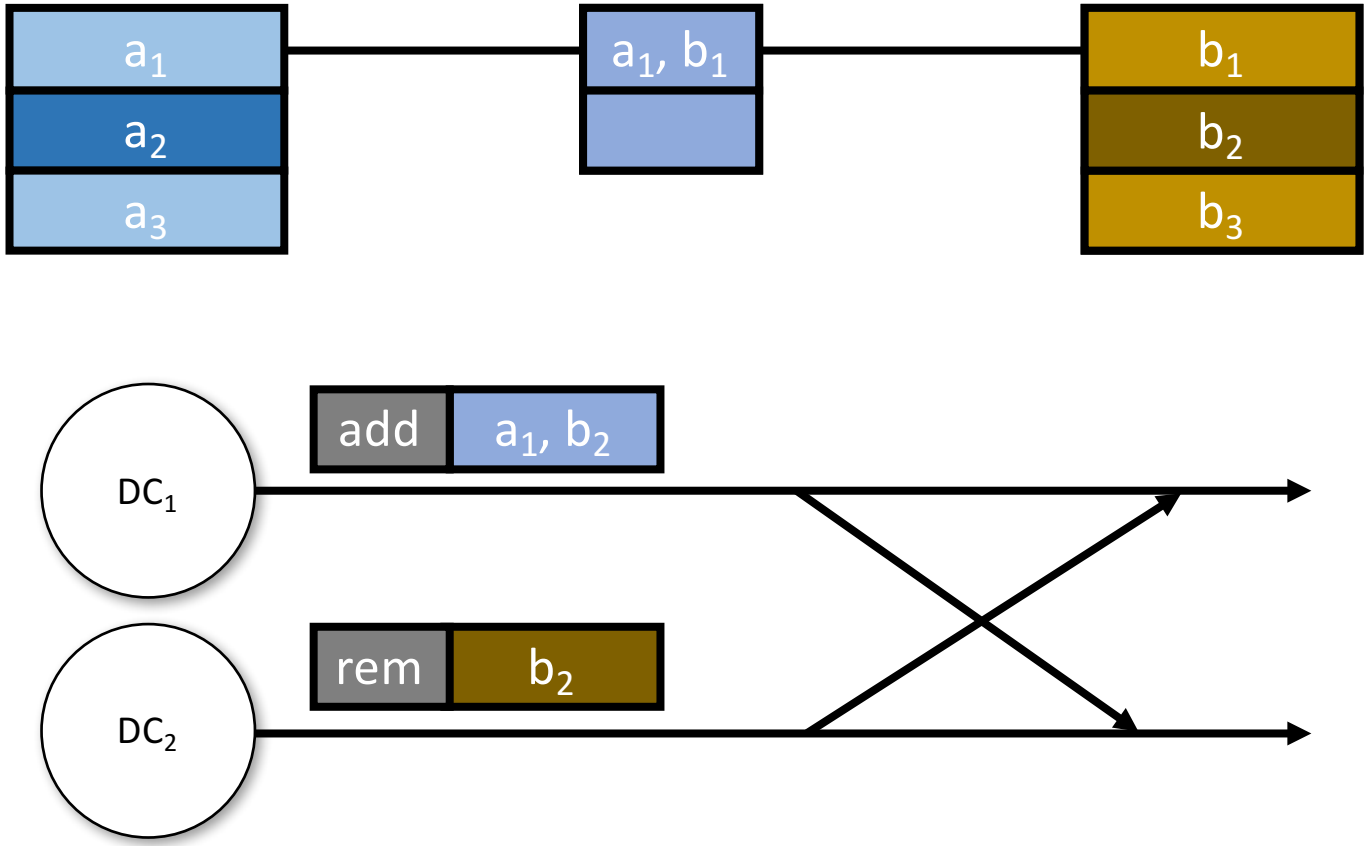
Referential integrity example



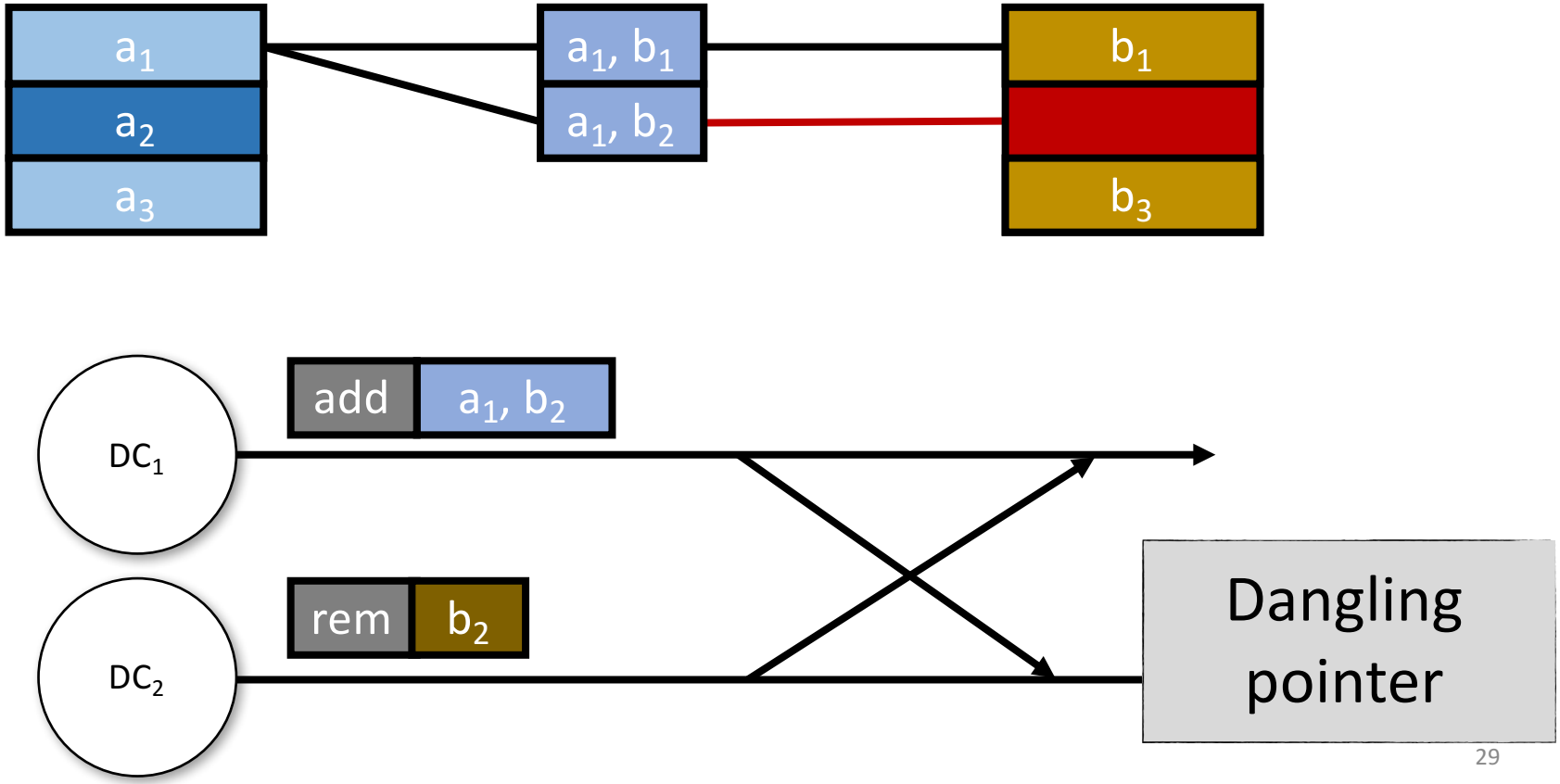
Referential integrity example



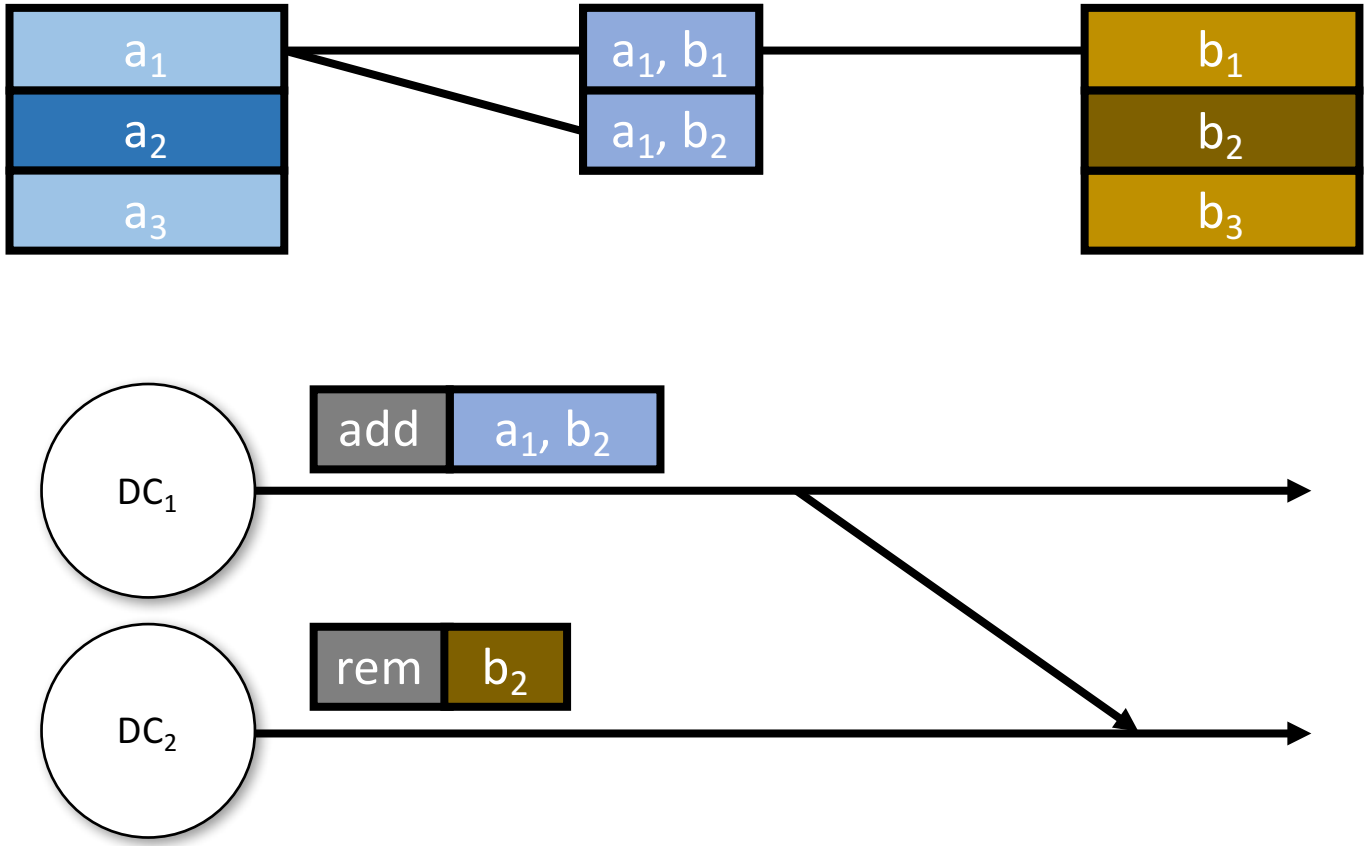
Referential integrity example



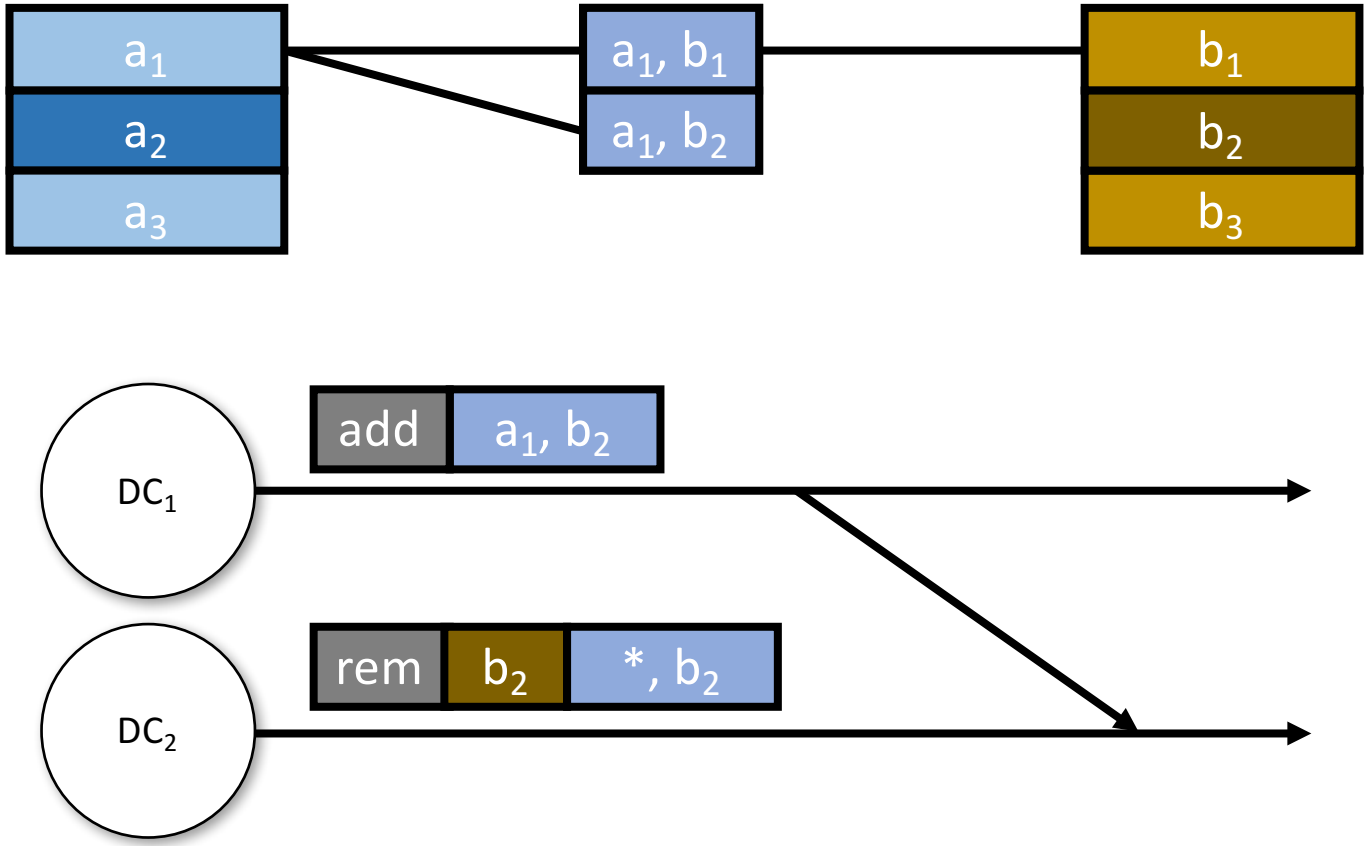
Referential integrity example



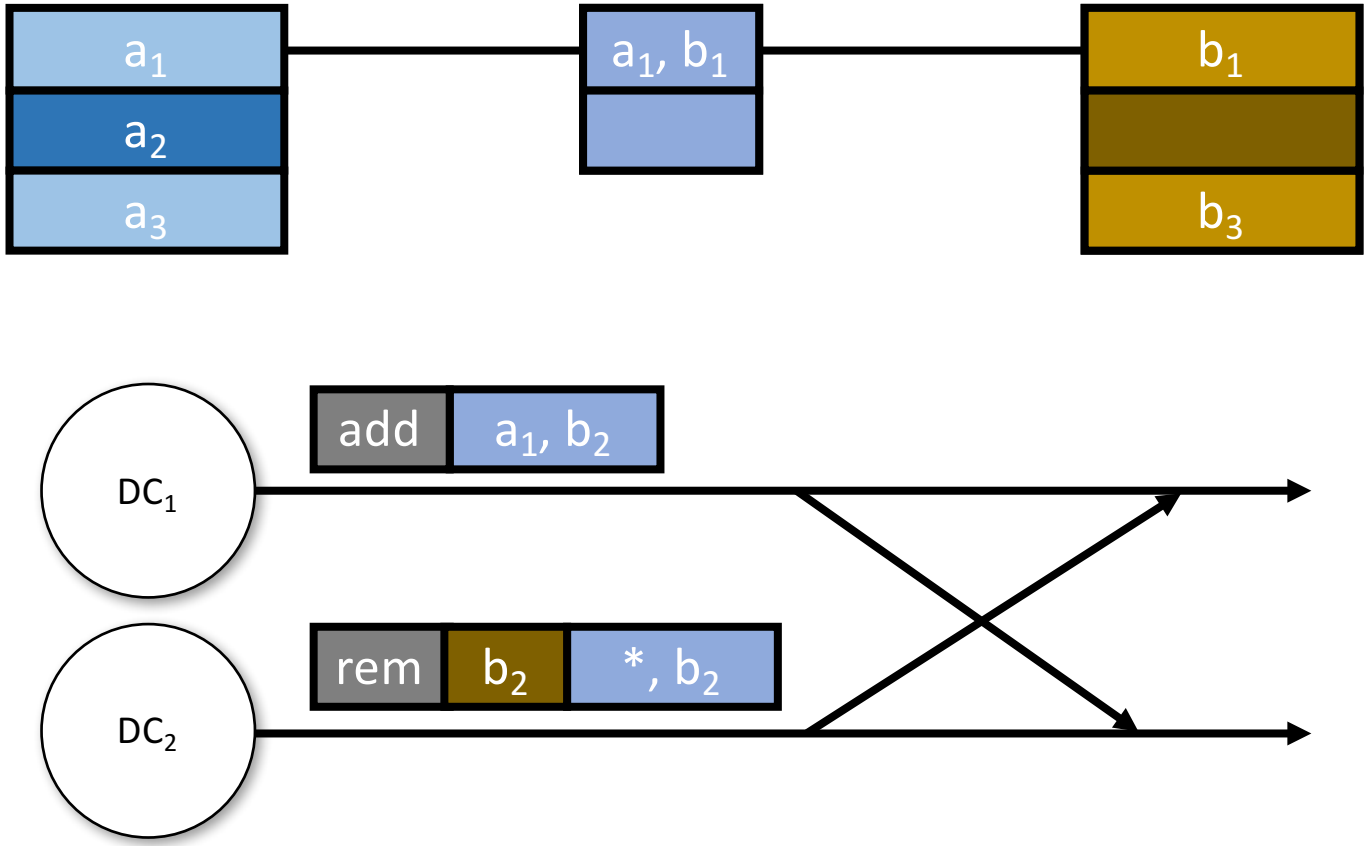
Preventive repair example



Preventive repair example



Preventive repair example



Scalability