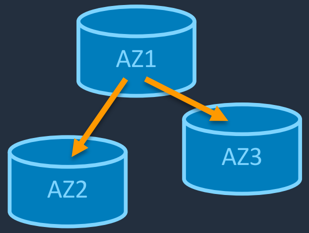
Transactions and Scalability in Amazon DynamoDB

Doug Terry

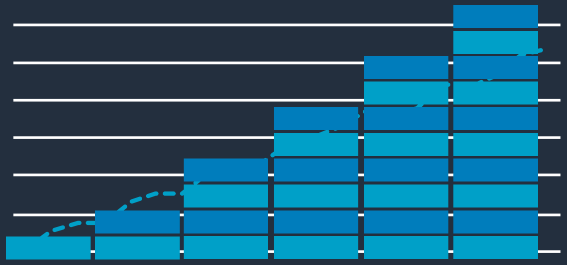
DynamoDB = NoSQL Database

Simple API

Flexible Schema



Highly Available



Unbounded Growth

Predictable Performance

DynamoDB Transaction Goals

Execute sets of operations
atomically and serializably
for any items in any tables
with predictable performance
and no impact on non-transactional workloads

Standard Approach

TxBegin

...

TxCommit

Explicit multi-step transactions

TxBegin

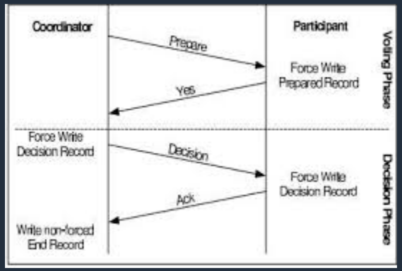
Put (...)

TxCommit

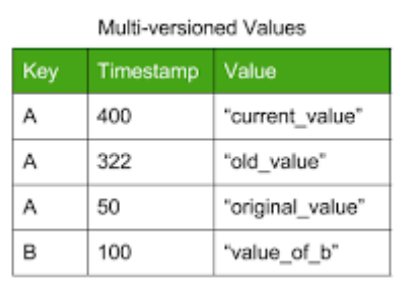
Implicit singleton transactions

Rejected

Two-phase locking


Coordinator
Participant
Prepare
Force Write Prepared Record
Yes
Voting Phase
Force Write Decision Record
Decision
Force Write Decision Record
Ack
Write non-forced End Record
Decision Phase

Two-phase commit



Multi-versioned Values

Key	Timestamp	Value
A	400	"current_value"
A	322	"old_value"
A	50	"original_value"
B	100	"value_of_b"

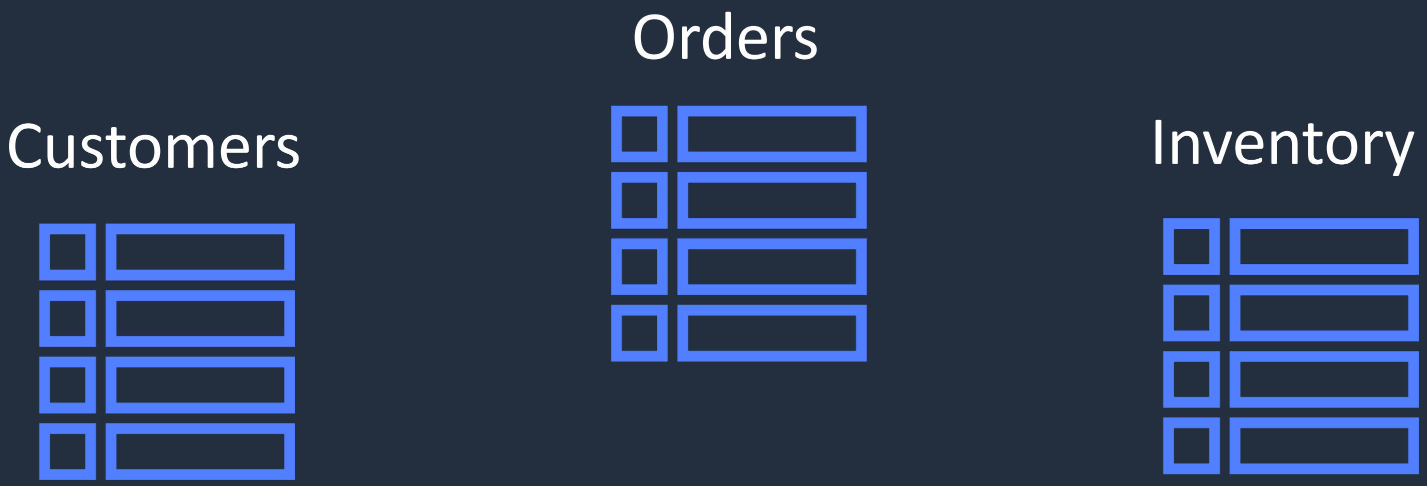
Multi-version Concurrency Control

DynamoDB Transactions

```
TransactGetItems (
    Get (table: “T1”, key: k1),
    Get (table: “T2”, key: k2),
    Get (table: “T3”, key: k3)
)
```

```
TransactWriteItems (
      Put (table: “T1”, key: k1,
          value: v1),
      Delete (table: “T2”, key: k2),
      Update (table: “T3”, key: k3,
          value: +1),
      Check (table: “T3”, key: k3,
          value: < 100)
)
```

Shopping Example



Customers

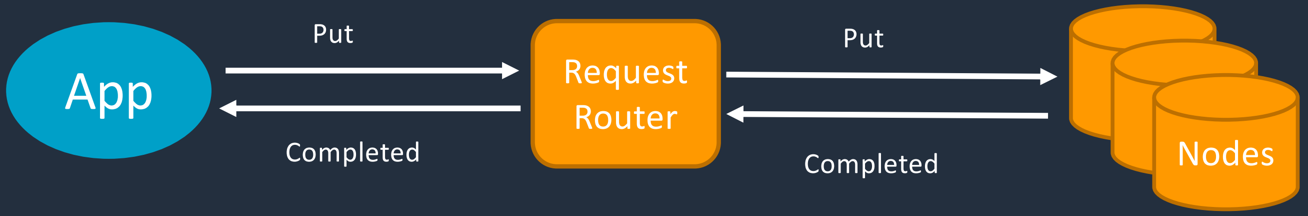
Orders

Inventory

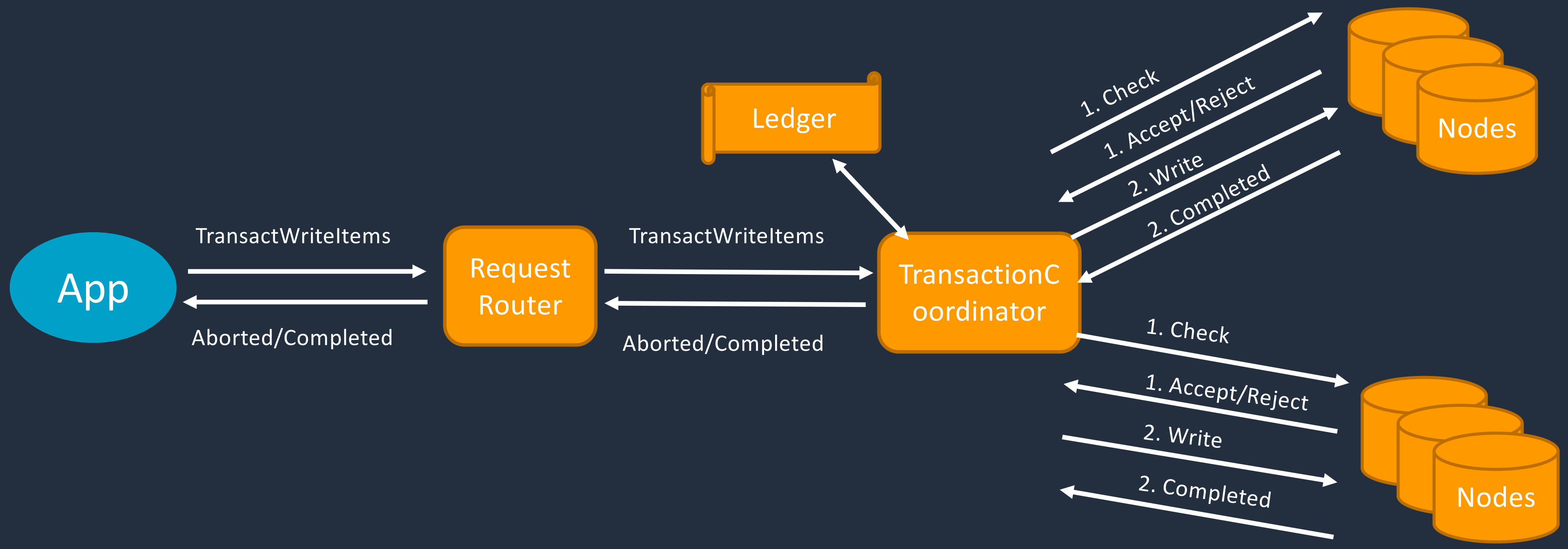
Shopping Example

```
TransactWriteItems (
        Check (table: “Customers”, key: “Susie” EXISTS),
        Check (table: “Inventory”, key: “book-99”, amount: >= 5),
        Put (table: “Orders”, key: newGUID(), customer: “Susie”,
           product: “book-99”, copies: 5, ...),
        Update (table: “Inventory”, key: “book-99”, amount: – 5)
)
```

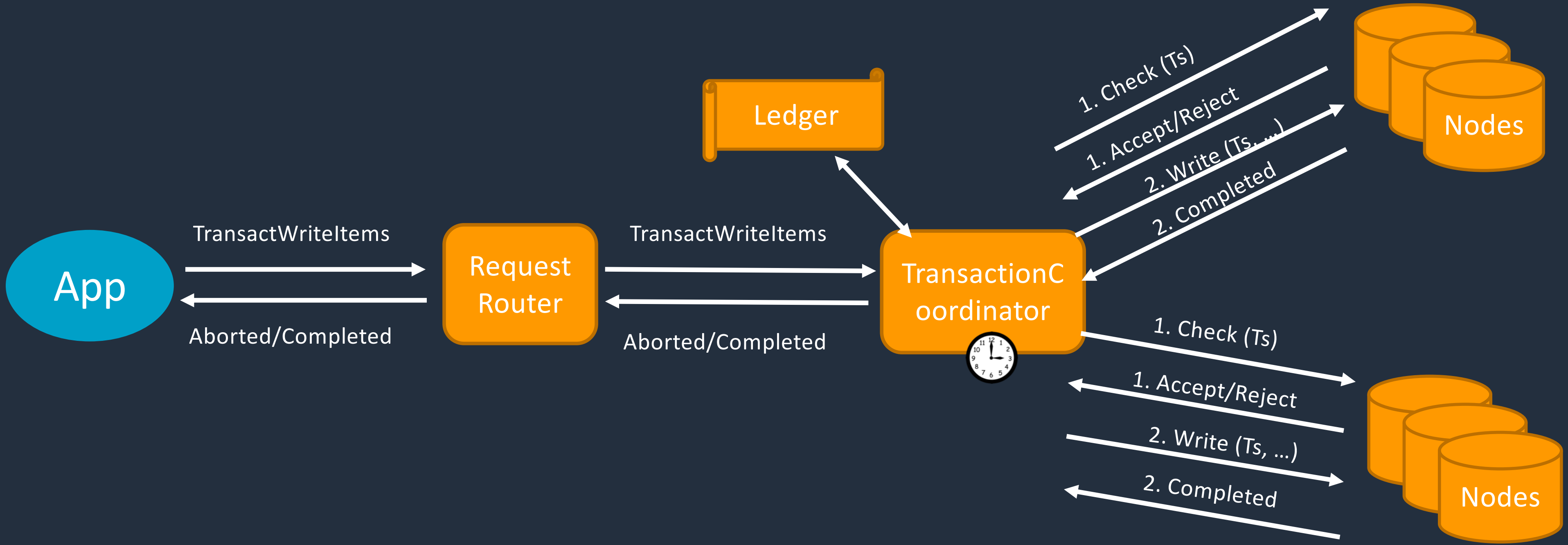
DynamoDB Operation Routing



DynamoDB Transactions Architecture



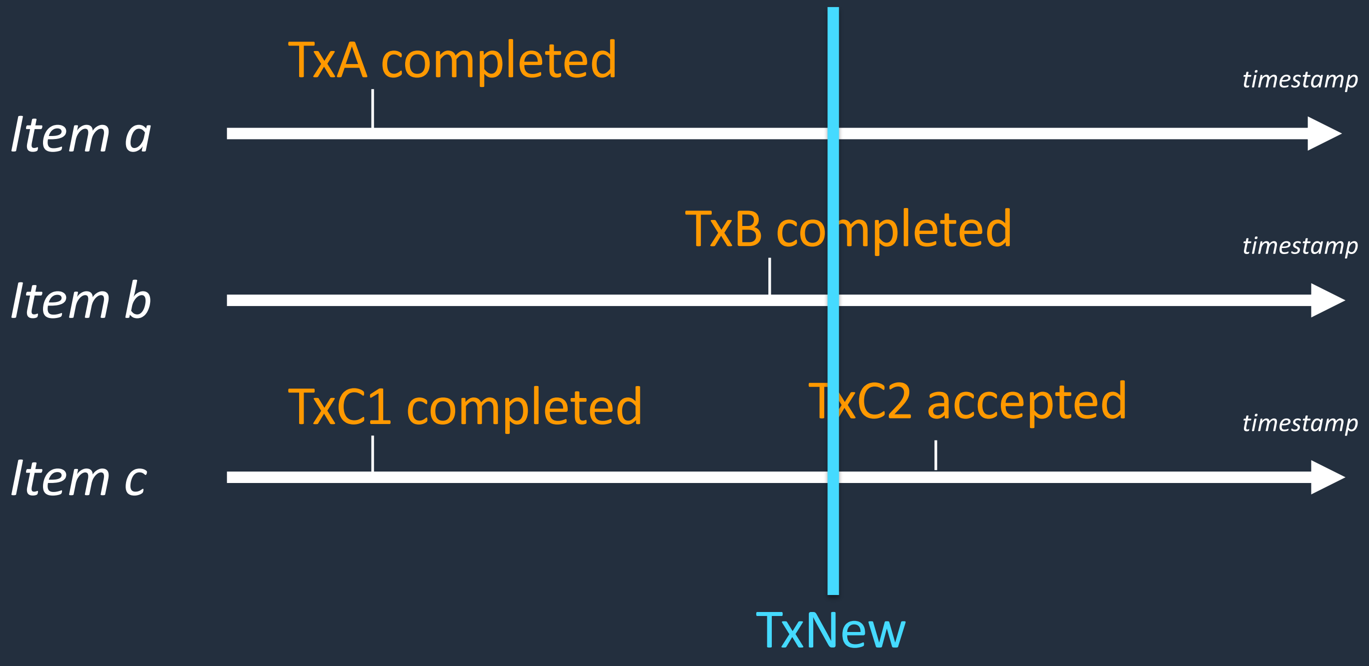
DynamoDB Transactions Architecture with Timestamp Ordering



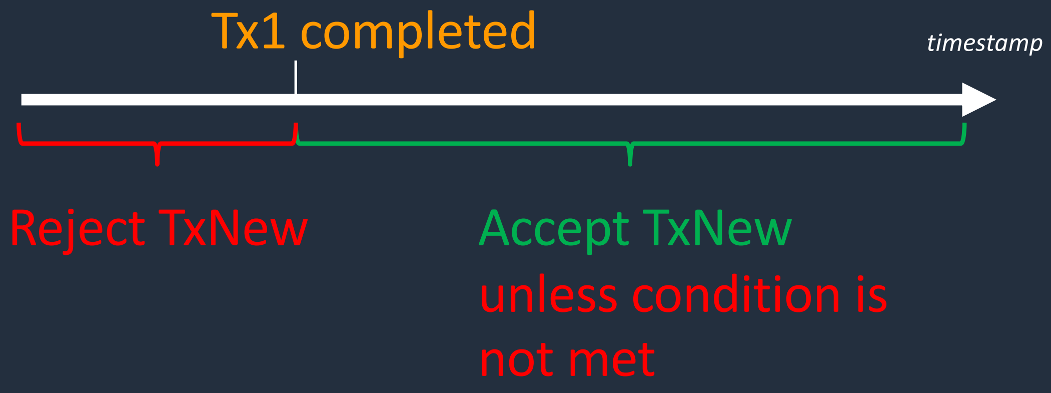
Phil A. Bernstein, David W. Shipman, and James B. Rothnie, Concurrency Control in a System for Distributed Databases (SDD-1), *ACM TODS*, 1980.

David P. Reed, Implementing Atomic Actions on Decentralized Data, *ACM TOCS*, 1983.

Rules for Accepting Transactions



Accepting New Write Transactions



Tx1 completed

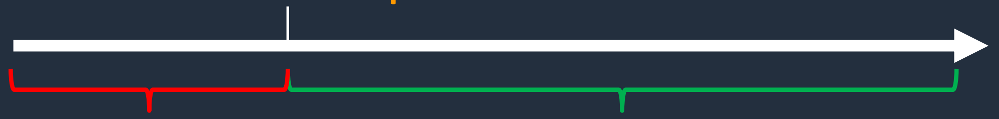
timestamp

Reject TxNew

Accept TxNew

unless condition is not met

Accepting New Write Transactions



Tx1 completed

timestamp

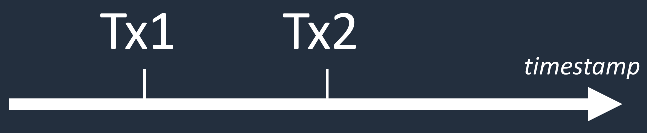
Reject TxNew unless Tx1 is unconditional Put or Delete

Accept TxNew unless condition is not met

Ordering Transaction Execution

Example:

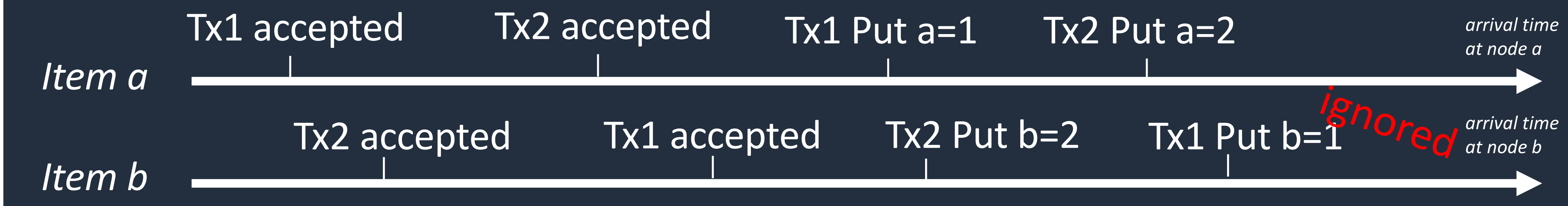
Tx1 = (Put a=1, Put b=1)
Tx2 = (Put a=2, Put b=2)



Tx1

Tx2

timestamp



Item a

Tx1 accepted

Tx2 accepted

Tx1 Put a=1

Tx2 Put a=2

arrival time at node a

Item b

Tx2 accepted

Tx1 accepted

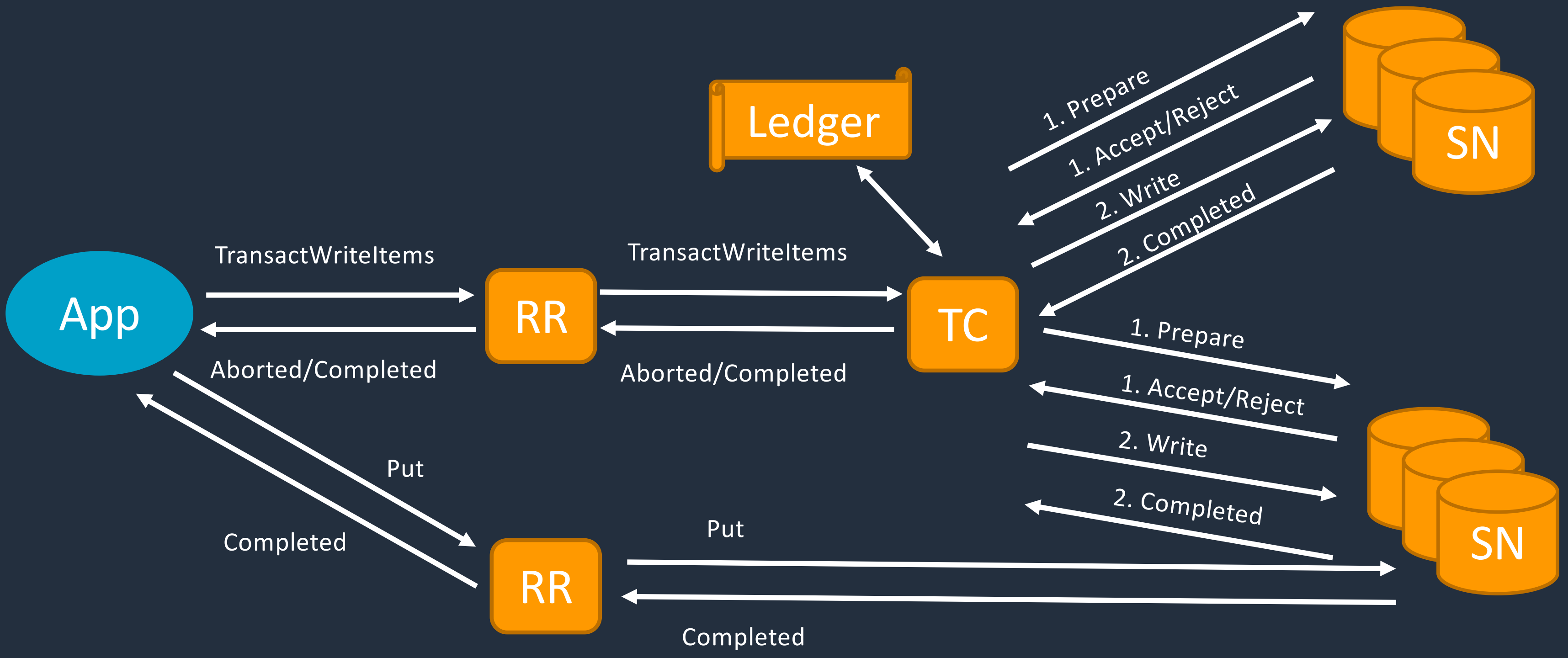
Tx2 Put b=2

Tx1 Put b=1

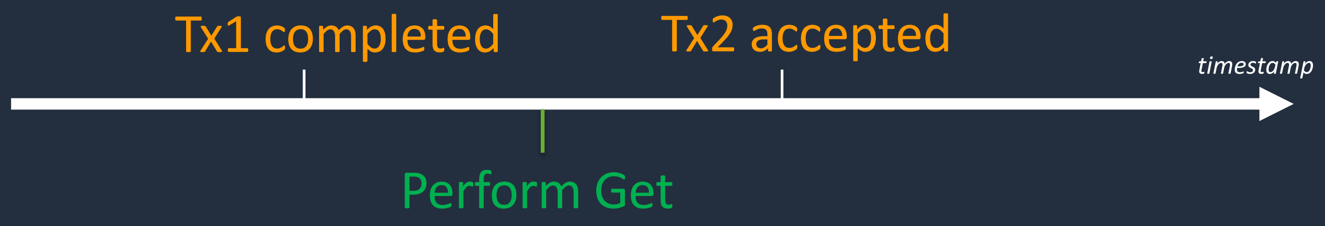
ignored

arrival time at node b

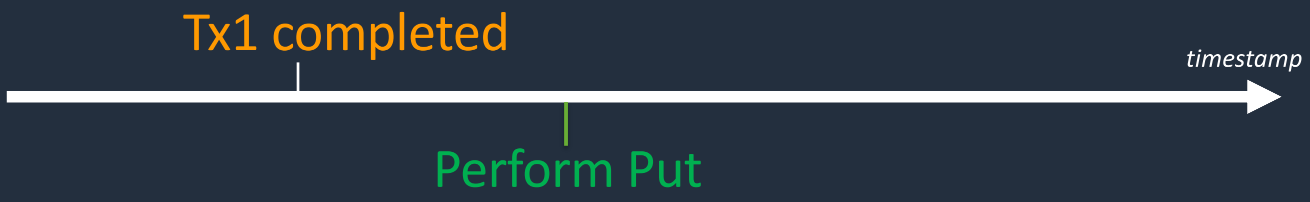
Non-transactional Operations



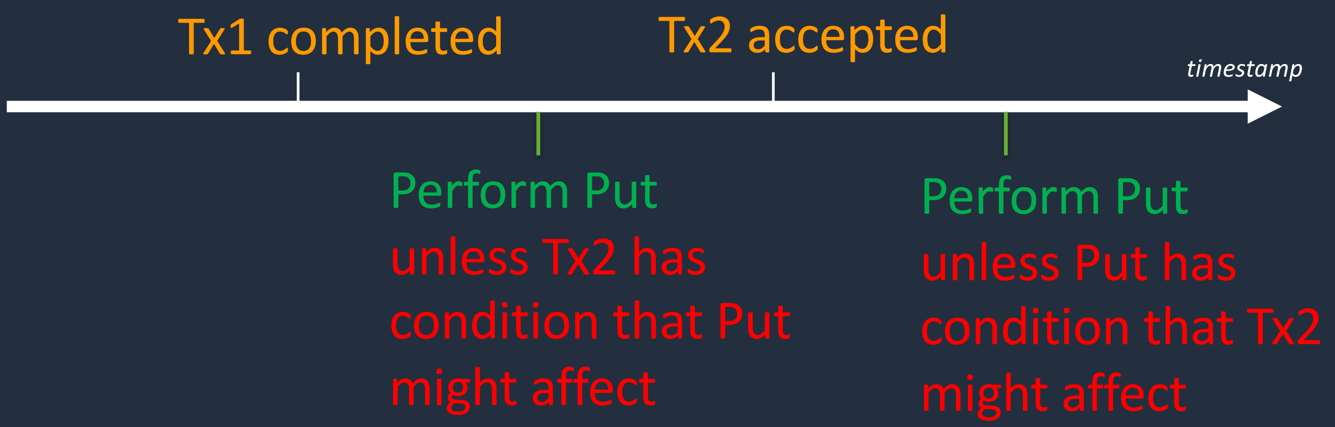
Non-transactional Reads



Non-transactional Writes



Non-transactional Writes



Tx1 completed

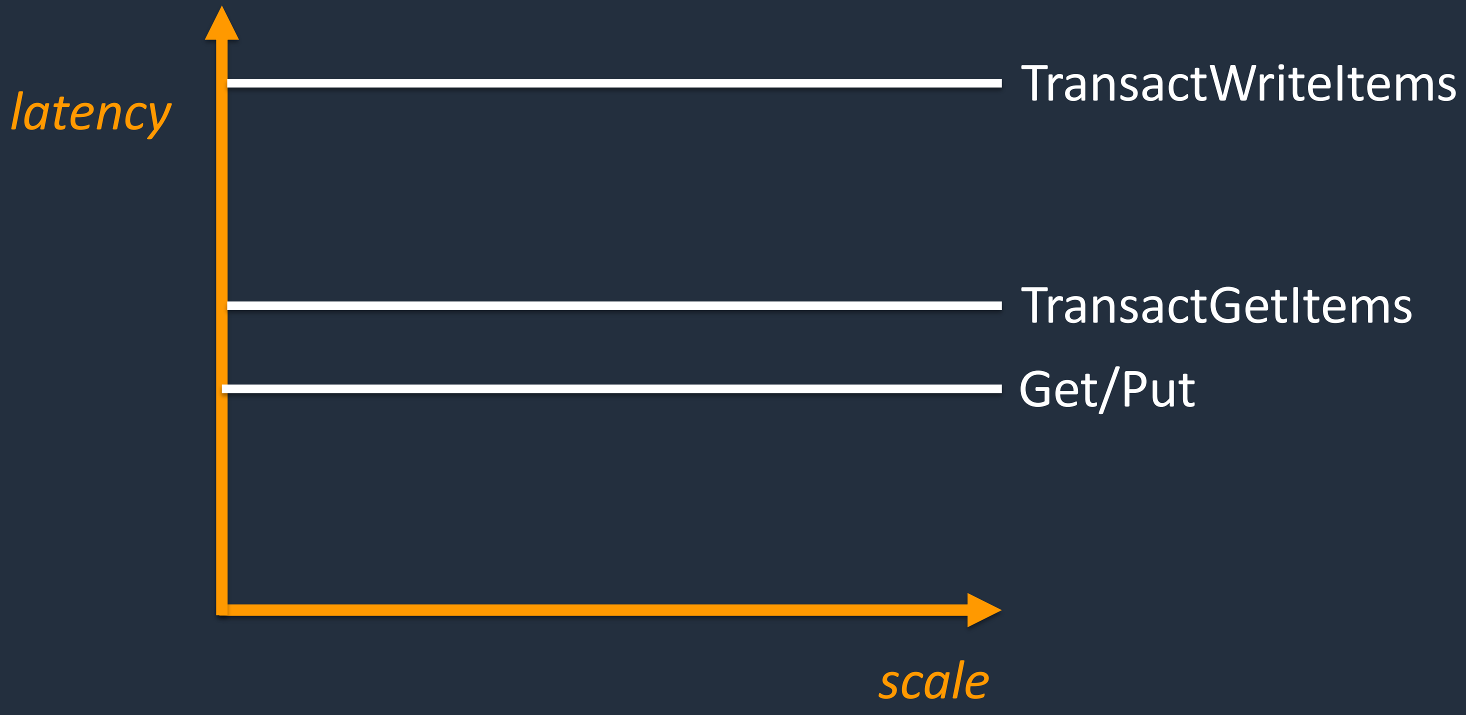
Tx2 accepted

timestamp

Perform Put unless Tx2 has condition that Put might affect

Perform Put unless Put has condition that Tx2 might affect

Scalability Concerns Revisited



Q&A

Doug Terry douglasbterry@hotmail.com