Transactions and Abstractions over HBase

Andreas Neumann
@anew68

Continuuity

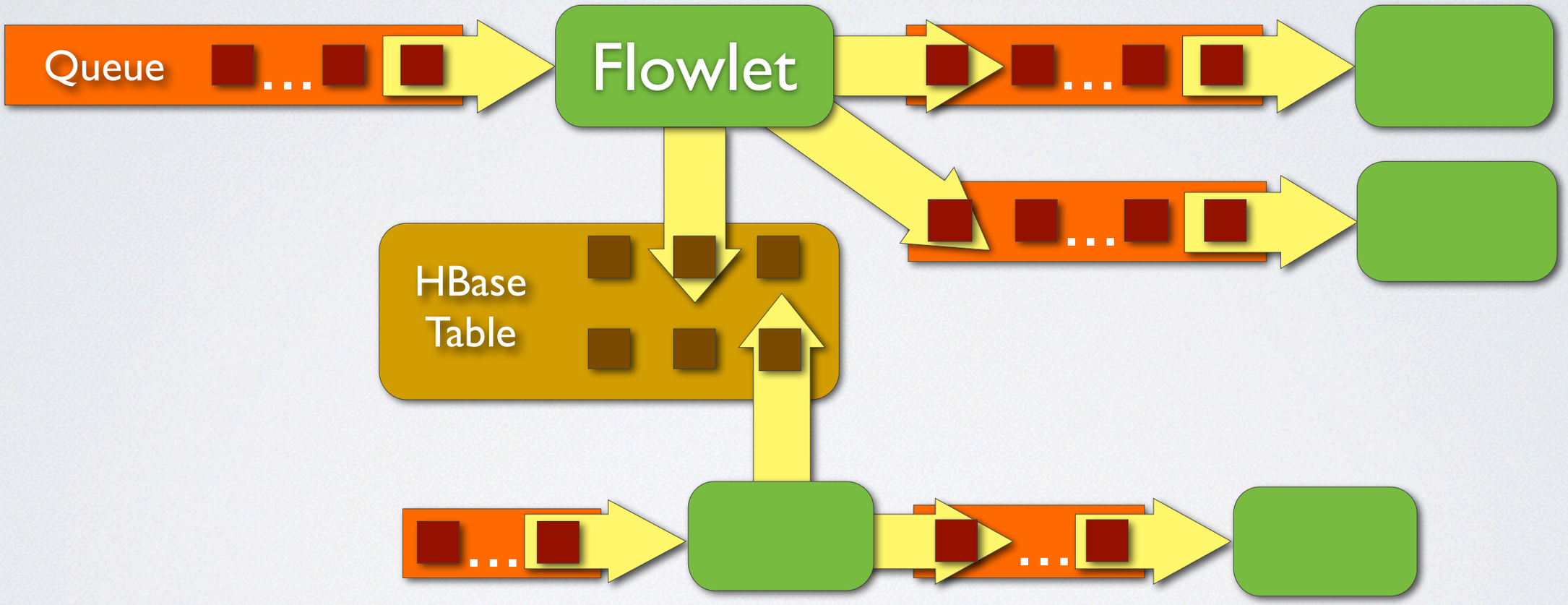
Agenda

- Transactions over HBase: **Why? What?**
- Implementation: **How?**
  - The approach
  - Transaction Manager
- Abstractions
- Future

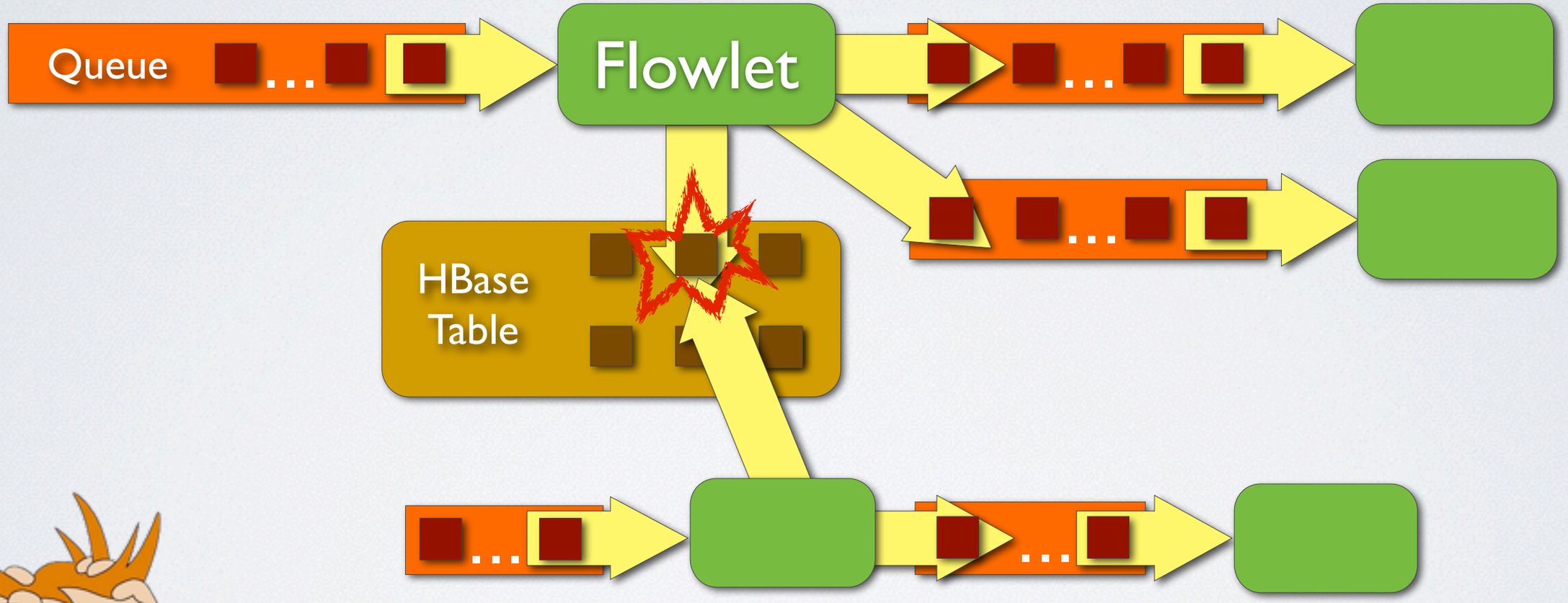
Who we are

- Simple Access to Powerful Technology
- **Continuuity Reactor**: the world's first scale-out application server for Hadoop
    - Fast, easy development, deployment and management of **Hadoop** and **HBase** apps
    - Collect, **Process**, **Store**, and Query data
- **Real-time** Stream Processing

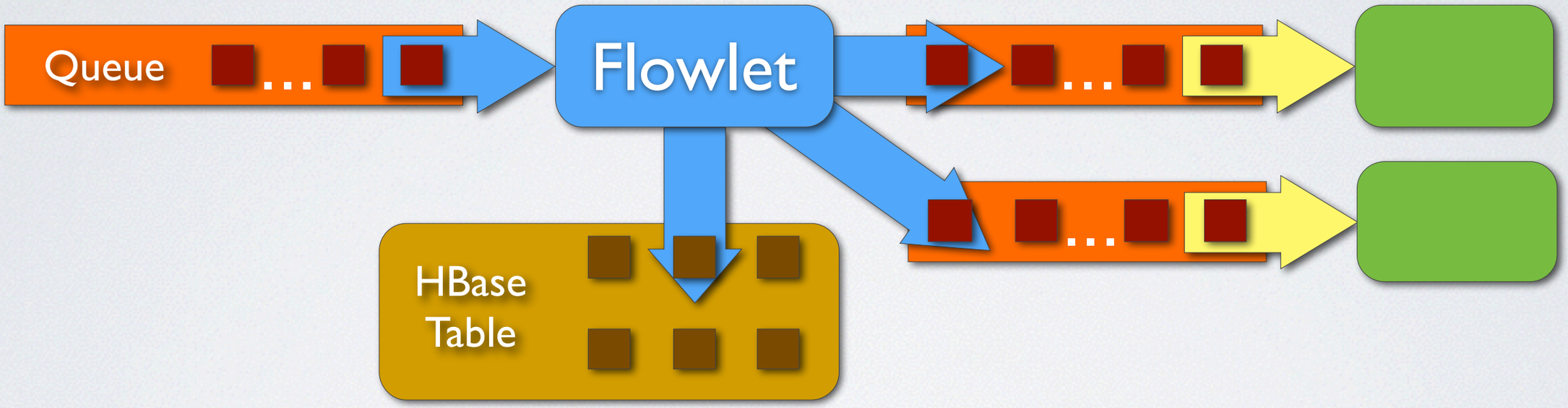
Real-Time Stream Processing



Real-Time Stream Processing



WRAP IN TRANSACTION!



Transactions: what?

- **Atomic** - Entire transaction is committed as one
- **Consistent** - No partial state change due to failure
- **Isolated** - No dirty reads, transaction is only visible after commit
- **Durable** - Once committed, data is persisted reliably

HBase: Quick "what?"

- Open-source non-relational **distributed** column-oriented **database** modeled after Google's BigTable
- Named Tables
  - Row key x Column key x *timestamp* -> Value
- Massive Scale
  - Key space partitioned into Regions.

What about HBase?

- **Atomic operations on cell value**:
  checkAndPut, checkAndDelete, increment, append
- **Atomic batch of operations** on rows **within region**
- **No** cross region atomic operations support
- **No** cross table atomic operations support
- **No** multi-RPC atomic operations support

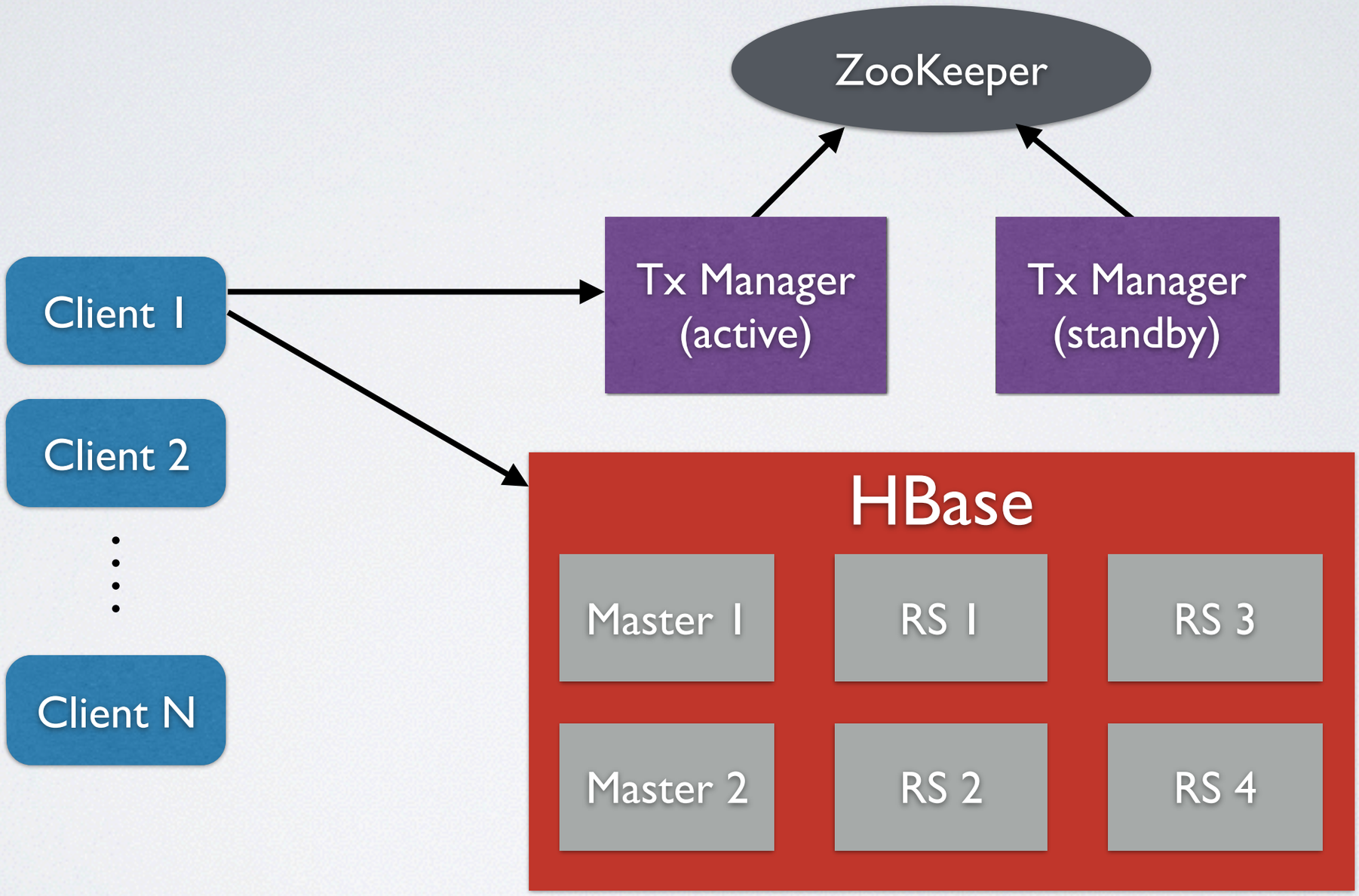
Implementation

- "OMID" style **Snapshot Isolation**
- **Multi-Version Concurrency Control**
  - **Cell version (timestamp) = transaction ID**
  - Reads exclude uncommitted transactions (for isolation)
- **Optimistic Concurrency Control**
  - Conflict detection at commit of transaction
  - Rollback in case of conflict (whichever commits later)

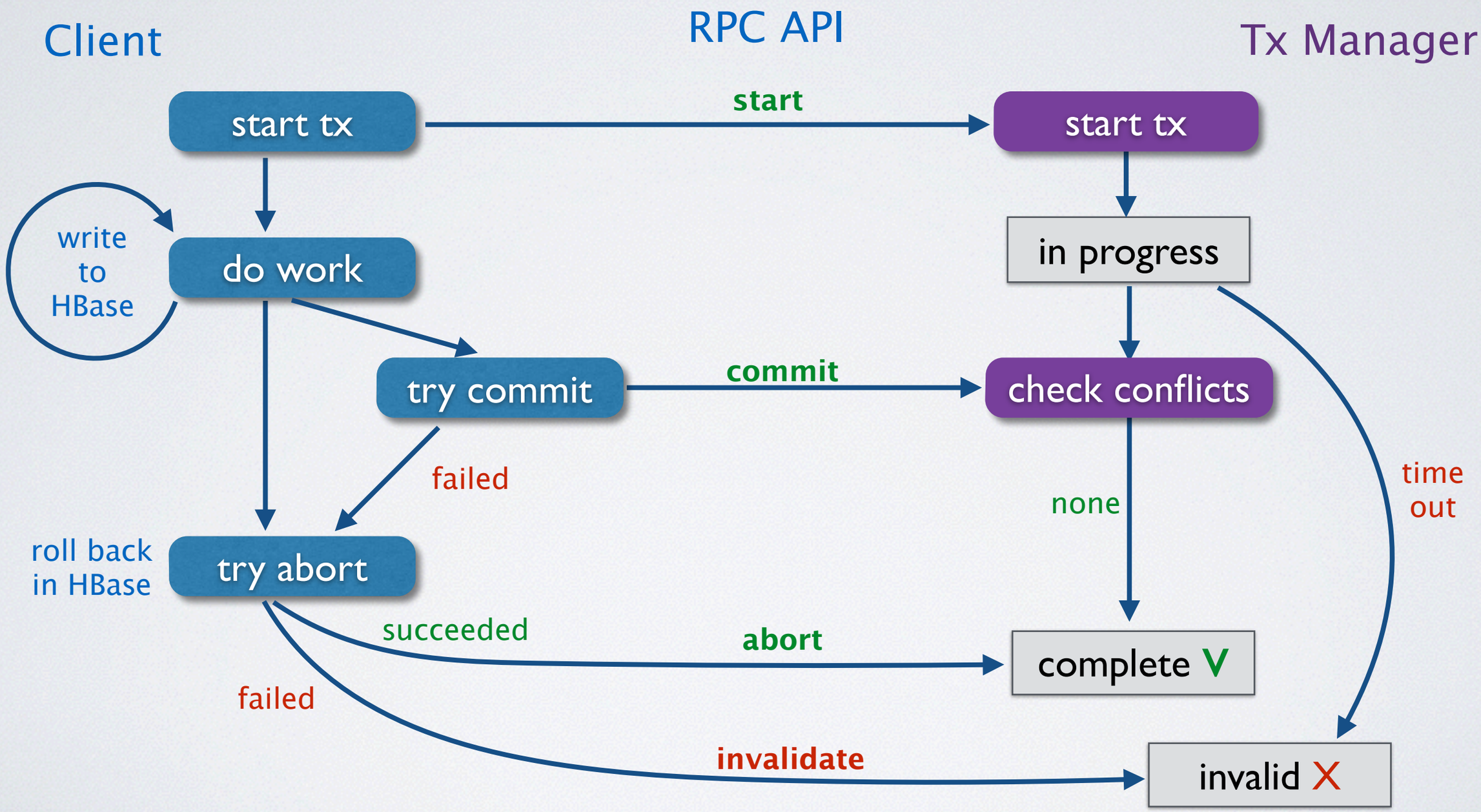
Optimistic Concurrency Control

- **Avoids cost of locking** rows and tables
- **No deadlocks** or lock escalations
- **Cost of conflict** detection and possible rollback is higher
- **Good if conflicts are rare**: short transaction, disjoint partitioning of work

Transactions in Context



Transaction Life Cycle



Transaction Manager

- Create new transactions
  - Provides **monotonically increasing** write pointers
- Maintains all in-progress, committed, and invalid transactions
- Detect conflicts
- Transaction =

  Write Pointer: Timestamp for HBase writes

  Read pointer: Upper bound timestamp for reads

  Excludes: List of timestamps to exclude from reads

Transaction Manager

- Simple & Fast
  - All required state is in-memory
- Single point of failure?
  - Periodically persist **snapshot** of all state
  - **Write-ahead log** for all changes since last snapshot
  - Secondary Tx Manager watches for failure of Primary
  - Failover can happen quickly

Transaction Cleanup

- Some transactions time out or fail to roll back
  - Invalid transactions must be excluded from reads
    - Exclude list can get large over time
- Old versions may not be visible to any transaction
- TTL (time-to-live) expires old versions

Data Janitor

- RegionObserver coprocessor
- Maintains in-memory snapshot of recent invalid & in-progress sets
- Periodically updates from transaction snapshot in HDFS
- Purges data from invalid transactions and older versions on flush & compaction

Abstraction

- Dataset implements TransactionAware interface

```
void startTx(Transaction tx);

Collection<byte[]> getTxChanges();

boolean commitTx() throws Exception;

boolean rollbackTx() throws Exception;
```

- Other Data Stores than HBase
- HBase, LevelDB, HyperSQL, In-Memory, …

Transaction Aware

- Modulation of ACID by dataset implementation
  - Granularity of keys - row, column, …
  - In-memory caching of writes
  - Skip rollback
  - …

What's next?

- Continue Scaling Tx Manager
  - Transaction Groups?
- Integration across other transactional stores
- Open Source
  - http://continuuity.com/blog

Looking for the chance to work with a team that is defining a new category within Big Data?

We are hiring!

http://continuuity.com/careers

careers[at]continuuity.com

Andreas Neumann @anew68 andreas[at]continuuity.com