Transactional Memory in Java

Adam Lider

Transactions

Database transactions

- Atomic
- Consistent
- Isolated
- Durable

Isolation levels

- No locking
- Uncommitted reads
- Committed reads
- Repeatable reads
- Serializable

Transactions - internals

- Locking - 2 phase locking
  - shared and exclusive locks
- Multi version concurrency control (MVCC)
  - global version clock
  - row stamps
- Mix of both

MVCC

- Snapshot of the data at some point in time
  - global version clock
  - two values per row:
    - created at
    - expired at

Concurrency in Java?

- < 1.5: Plain threads
  - mutex, monitor
- >= 1.5: High-level APIs (java.util.concurrent)
  - tasks, synchronizers, new data structures

Application

```java
public class Account {

    private long balance;

    public void deposit(long amount) {
        this.balance += amount;
    }

    public void withdraw(long amount) {
        if (balance - amount > 0) {
            this.balance -= amount;
        } else {
            throw new IllegalStateException("Not enough money");
        }
    }

    public long getBalance() {
        return this.balance;
    }
}
```

Concurrent access

```
public class Account {

    private long balance;

    public synchronized void deposit(long amount) {
        this.balance += amount;
    }

    public synchronized void withdraw(long amount) {
        if (balance - amount > 0) {
            this.balance -= amount;
        } else {
            throw new IllegalStateException("Not enough money");
        }
    }

    public synchronized long getBalance() {
        return this.balance;
    }
}

public class Bank {

    public void transfer(Account from, Account to, long amount) {
        from.withdraw(amount);
        to.deposit(amount);
    }
}
```

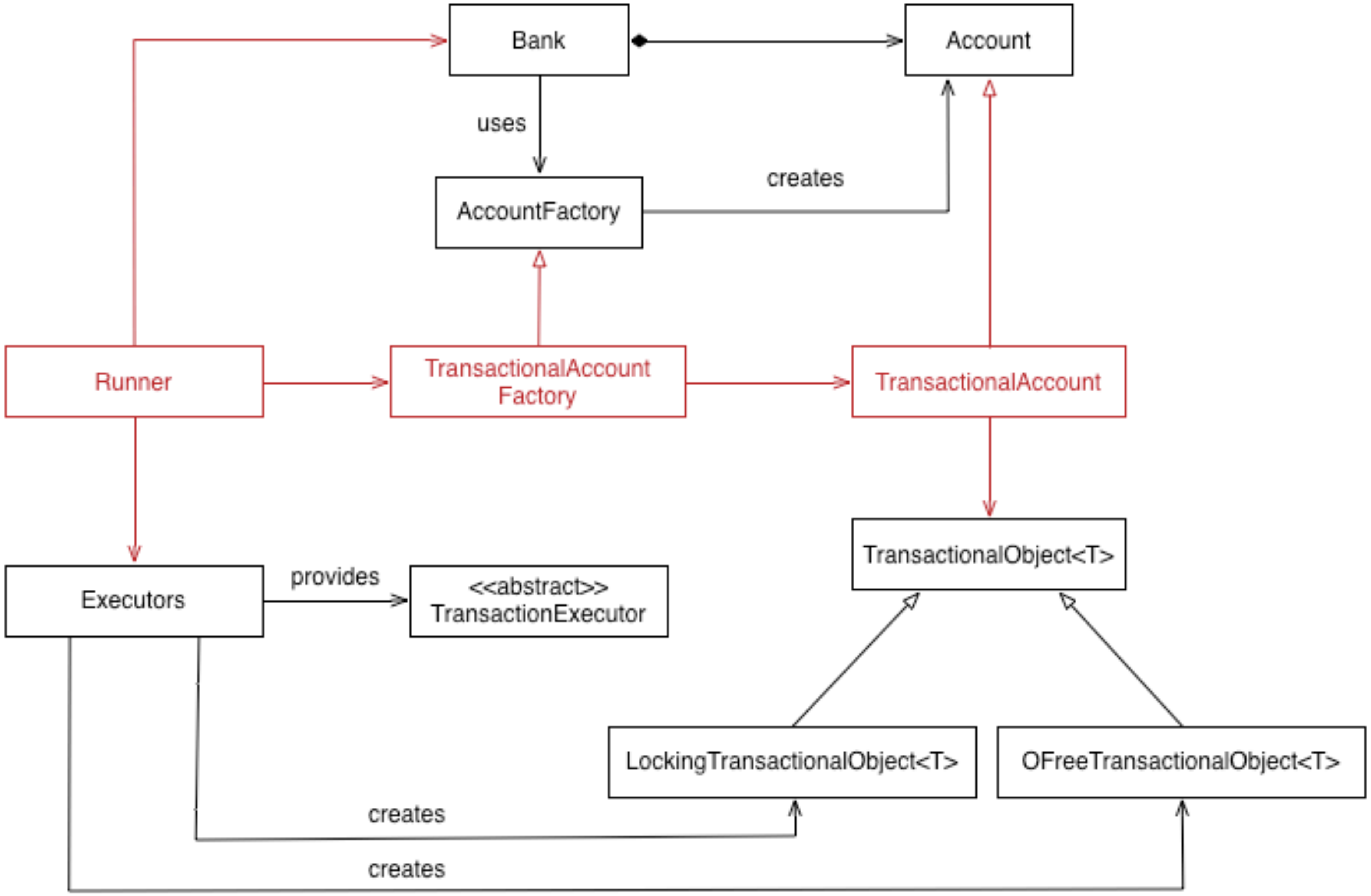
Transactional bank

```
public void transfer(Account from, Account to, long amount) {
    transactional {
        from.withdraw(amount);
        to.deposit(amount);
    }
}
```

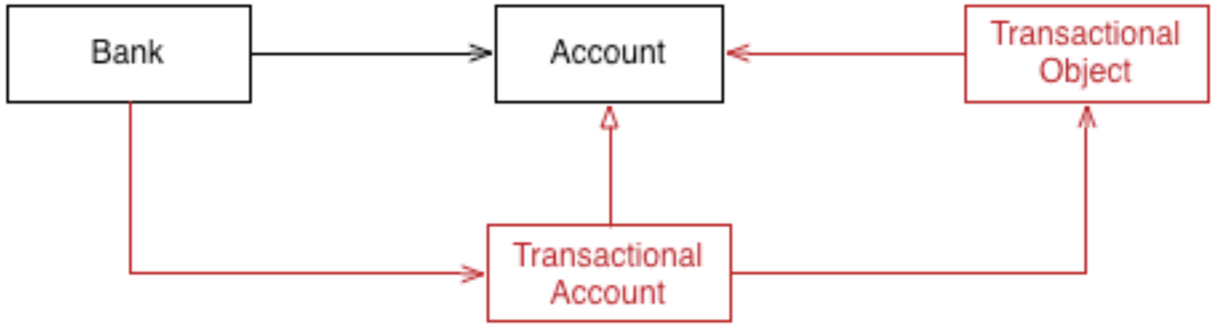
```
@Transactional
public void transfer(Account from, Account to, long amount) {
    from.withdraw(amount);
    to.deposit(amount);
}
```

```
public void transfer(final Account from, final Account to, final long amount) {
    TransactionExecutor.execute(new Runnable() {
        public void run() {
            from.withdraw(amount);
            to.deposit(amount);
        }
    });
}
```

Application structure



Objects composition



TransactionalObject

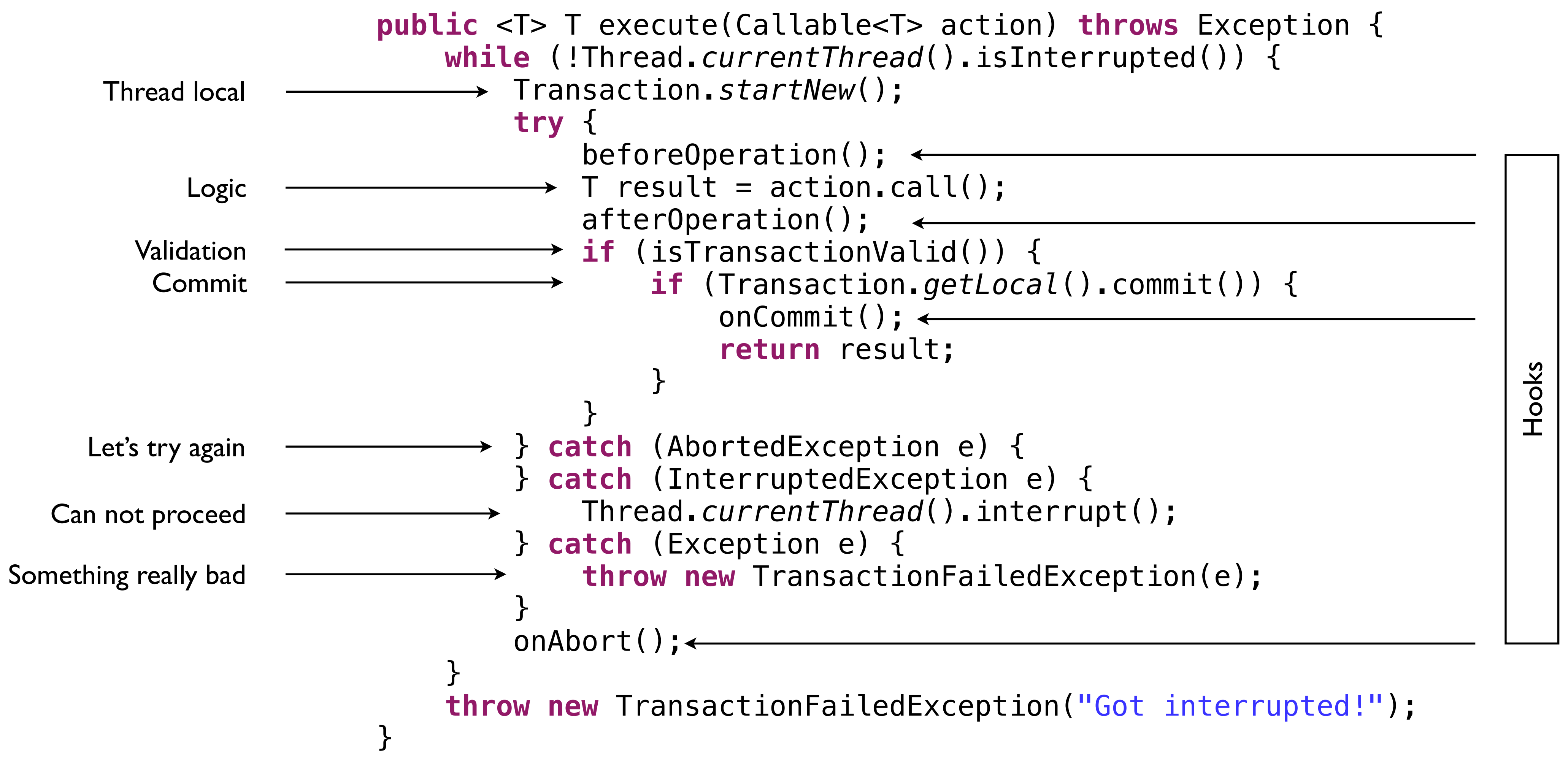
```
public interface TransactionalObject<T> {

    T openForRead();

    T openForWrite();

    boolean isValid();
}
```

TransactionExecutor



```
public <T> T execute(Callable<T> action) throws Exception {
    while (!Thread.currentThread().isInterrupted()) {
        Transaction.startNew();
        try {
            beforeOperation();
            T result = action.call();
            afterOperation();
            if (isTransactionValid()) {
                if (Transaction.getLocal().commit()) {
                    onCommit();
                    return result;
                }
            }
        } catch (AbortedException e) {
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        } catch (Exception e) {
            throw new TransactionFailedException(e);
        }
        onAbort();
    }
    throw new TransactionFailedException("Got interrupted!");
}
```

Thread local

Logic

Validation

Commit

Let’s try again

Can not proceed

Something really bad

Hooks

Algorithms

- Dynamic Software Transactional Memory
  - Maurice Herlihy, Victor Luchangco
- Transactional Locking 2
  - Dave Dice, Ori Shalev, Nir Shavit

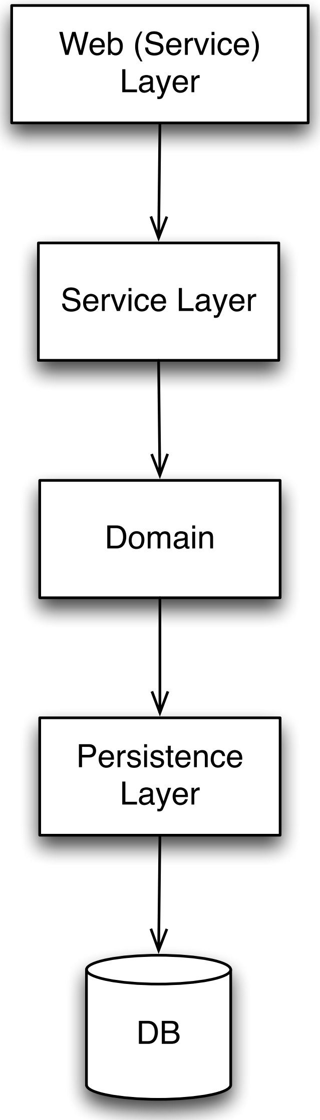
Progress conditions

- Non-blocking
    - wait-free
    - lock-free
    - obstruction-free
- Blocking

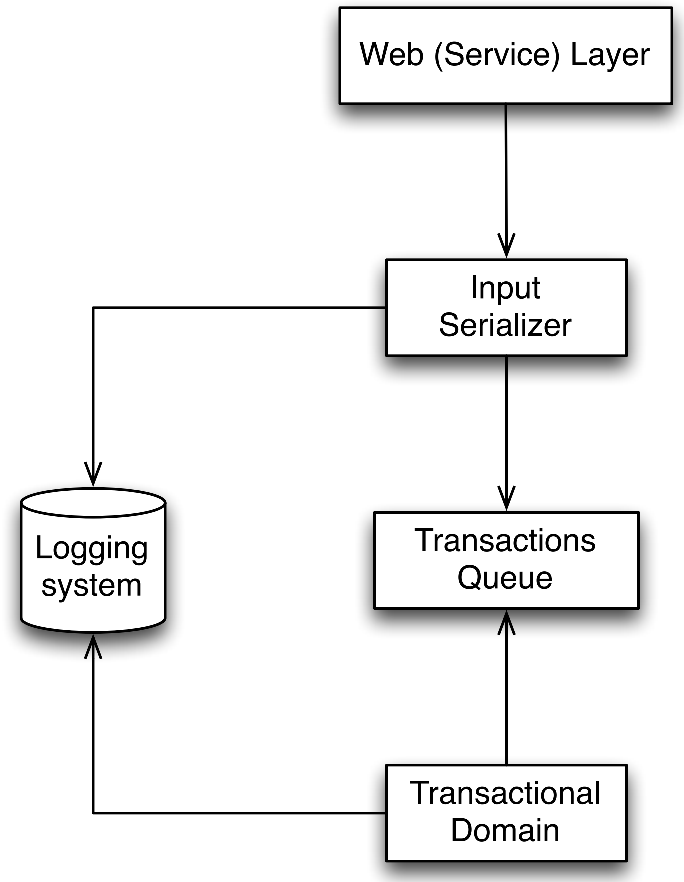
CAS

- CMPXCHG
- Java compareAndSet
    - java.util.concurrent.atomic
- ABA problem

Typical architecture



STM based architecture



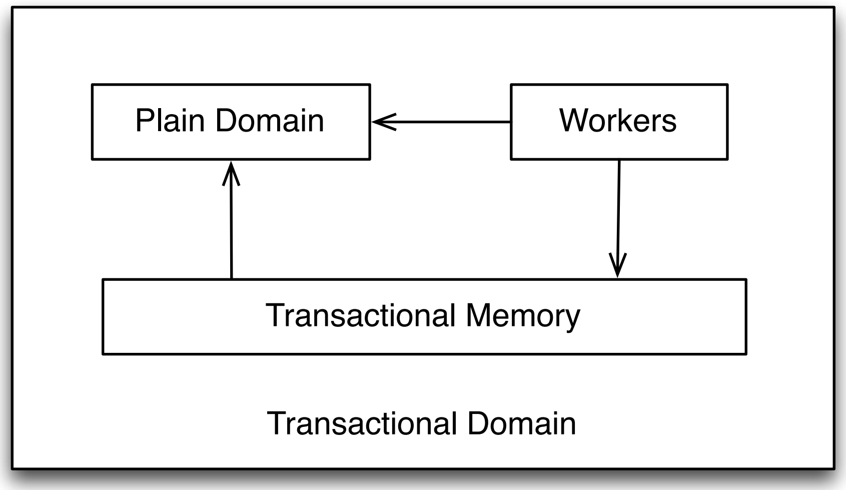
Web (Service) Layer

Input Serializer

Logging system

Transactions Queue

Transactional Domain



Plain Domain

Workers

Transactional Memory

Transactional Domain

Resources

- Transactional Locking II - Dave Dice, Ori Shalev, and Nir Shavit
- Software Transactional Memory for Dynamic-Sized Data Structures - Maurice Herlihy, Victor Luchangco, Mark Moir, William N. Scherer III
- A Flexible Framework for Implementing Software Transactional Memory - Maurice Herlihy, Victor Luchangco, Mark Moir
- Software Transactional Memory Should Not Be Obstruction-Free - Robert Ennals