Transactional Memory

A Very Brief Intro

Concurrent Algorithms Project — Fall 2018

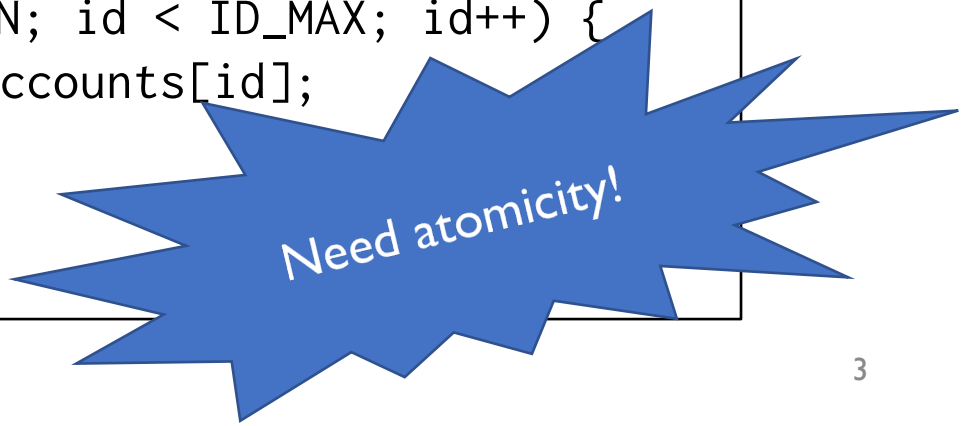
Igor Zablotchi

Example: a Simple Bank

```
transfer(accountId from, accountId to, int amt);

sumAccounts();
```

A Naïve Implementation

```
Shared: accounts[ID_MIN…ID_MAX]; // array of ints

transfer(accountId from, accountId to, int amt) {
        accounts[from] -= amt;
        accounts[to] += amt;
}

sumAccounts() {
        sum = 0;
        for (id = ID_MIN; id < ID_MAX; id++) {
                sum += accounts[id];
        }
        return sum;
}
```



Locks – One Possible Solution

```
transfer(accountId from, accountId to, int amt) {
        LOCK(…);
        accounts[from] -= amt;
        accounts[to] += amt;
        UNLOCK(…);
}
sumAccounts() {
        LOCK(…);
        sum = 0;
        for (id = ID_MIN; id < ID_MAX; id++) {
                sum += accounts[id];
        }
        UNLOCK(…);
        return sum;
}
```

Locks Have Problems

- **Priority inversion**: a lower-priority thread holding a lock is pre-empted by a higher-priority thread that needs the lock
- **Convoying**: a thread holding a lock is descheduled, causing other threads that need the lock to queue up.
- **Deadlock**: two or more threads try to acquire the same locks in different orders

An Ideal Solution

```
transfer(accountId from, accountId to, int amt) {
atomic {
        accounts[from] -= amt;
        accounts[to] += amt;
}
}
sumAccounts() {
atomic {
        sum = 0;
        for (id = ID_MIN; id < ID_MAX; id++) {
                sum += accounts[id];
        }
        return sum;
}
}
```

The Solution You Will Implement

```
transfer(accountId from, accountId to, int amt) {
        tx = tm_start(…);
        fromAmt = tx_read(tx, accounts[from], …);
        tx_write(tx, fromAmt - amt, …);
        toAmt = tx_read(tx, accounts[to], …);
        tx_write(tx, toAmt + amt, …);
        tm_end(…, tx);
}
sumAccounts() {
        tx = tm_start(…);
        sum = 0;
        for (id = ID_MIN; id < ID_MAX; id++) {
                sum += tx_read(tx, accounts[id], …);
        }
        tm_end(…, tx);
        return sum;
}
```

Further Reading

- R. Guerraoui and M. Kapałka. *The Theory of Transactional Memory.*
- M. Herlihy and N. Shavit. *The Art of Multiprocessor Programming, Revised Reprint.* ***Chapter 18***
- N. Shavit and D. Touitou. *Software transactional memory.*
- M. Herlihy, V. Luchangco, M. Moir, and W. N. Scherer III. *Software Transactional Memory for Dynamic-Sized Data Structures*
- P. Felber, C. Fetzer and T. Riegel. *Dynamic Performance Tuning of Word-Based Software Transactional Memory*
- Dave Dice, O. Shalev, and N. Shavit. *Transactional Locking II*
- L. Dalessandro, M. Spear, and M. Scott. *NOrec: Streamlining STM by Abolishing Ownership Records*