Transactions
2PC Review

Chris Rossbach

cs378h

Transactions

Core issue: multiple updates

Canonical examples:

```
move(file, old-dir, new-dir) {
   delete(file, old-dir)
   add(file, new-dir)
}
```

```
create(file, dir) {
   alloc-disk(file, header, data)
   write(header)
   add (file, dir)
}
```

Transactions

Core issue: multiple updates

Canonical examples:

```
move(file, old-dir, new-dir) {
   delete(file, old-dir)
   add(file, new-dir)
}
```

```
create(file, dir) {
   alloc-disk(file, header, data)
   write(header)
   add (file, dir)
}
```

- Modified data in memory/caches
- Even if in-memory data is durable, multiple disk updates

Transactions

Core issue: multiple updates

Canonical examples:

```
move(file, old-dir, new-dir) {
   delete(file, old-dir)
   add(file, new-dir)
}
```

```
create(file, dir) {
   alloc-disk(file, header, data)
   write(header)
   add (file, dir)
}
```

Problem: crash in the middle

- Modified data in memory/caches
- Even if in-memory data is durable, multiple disk updates

Transactions: Implementation

Transactions: Implementation

- Key idea: turn multiple updates into a single one

Transactions: Implementation

- Key idea: turn multiple updates into a single one
- Many implementation Techniques
  - Two-phase locking
  - Timestamp ordering
  - Optimistic Concurrency Control
  - Journaling
  - 2,3-phase commit
  - Speculation-rollback
  - Single global lock
  - Compensating transactions

Transactions: Implementation

- Key idea: turn multiple updates into a single one
- Many implementation Techniques
  - Two-phase locking
  - Timestamp ordering
  - Optimistic Concurrency Control
  - Journaling
  - 2,3-phase commit
  - Speculation-rollback
  - Single global lock
  - Compensating transactions

Key problems:

- output commit
- synchronization

Transactions: Implementation

- Key idea: turn multiple updates into a single one
- Many implementation Techniques
  - Two-phase locking
  - Timestamp ordering
  - Optimistic Concurrency Control
  - Journaling
  - 2,3-phase commit
  - Speculation-rollback
  - Single global lock
  - Compensating transactions

Key problems:

- output commit
- synchronization

Two-phase commit

- N participants agree or don’t (atomicity)
- Phase 1: everyone “prepares”
- Phase 2: Master decides and tells everyone to actually commit
- What if the master crashes in the middle?

2PC: Phase 1

1. Coordinator sends REQUEST to all participants
2. Participants receive request and
3. Execute locally
4. Write VOTE_COMMIT or VOTE_ABORT to local log
5. Send VOTE_COMMIT or VOTE_ABORT to coordinator

Example—move: C→S1: delete foo from /, C→S2: add foo to /

Failure case:
S1 writes rm /foo, VOTE_COMMIT to log
S1 sends VOTE_COMMIT
S2 decides permission problem
S2 writes/sends VOTE_ABORT

Success case:
S1 writes rm /foo, VOTE_COMMIT to log
S1 sends VOTE_COMMIT
S2 writes add foo to /
S2 writes/sends VOTE_COMMIT

2PC: Phase 2

- Case 1: receive VOTE_ABORT or timeout
  - Write GLOBAL_ABORT to log
  - send GLOBAL_ABORT to participants
- Case 2: receive VOTE_COMMIT from all
  - Write GLOBAL_COMMIT to log
  - send GLOBAL_COMMIT to participants
- Participants receive decision, write GLOBAL_* to log

2PC corner cases

Phase 1

1. Coordinator sends REQUEST to all participants
2. X Participants receive request and
3. Execute locally
4. Write VOTE_COMMIT or VOTE_ABORT to local log
5. Send VOTE_COMMIT or VOTE_ABORT to coordinator

Phase 2

- Y Case 1: receive VOTE_ABORT or timeout
  - Write GLOBAL_ABORT to log
  - send GLOBAL_ABORT to participants
- Case 2: receive VOTE_COMMIT from all
  - W Write GLOBAL_COMMIT to log
  - send GLOBAL_COMMIT to participants
- Z Participants recv decision, write GLOBAL_* to log

- What if participant crashes at X?
- Coordinator crashes at Y?
- Participant crashes at Z?
- Coordinator crashes at W?

2PC limitation(s)

2PC limitation(s)

- Coordinator crashes at W, never wakes up

2PC limitation(s)

- Coordinator crashes at W, never wakes up
- All nodes block forever!

2PC limitation(s)

- Coordinator crashes at W, never wakes up
- All nodes block forever!
- Can participants ask each other what happened?

2PC limitation(s)

- Coordinator crashes at W, never wakes up
- All nodes block forever!
- Can participants ask each other what happened?
- 2PC: always has risk of indefinite blocking

2PC limitation(s)

- Coordinator crashes at W, never wakes up
- All nodes block forever!
- Can participants ask each other what happened?
- 2PC: always has risk of indefinite blocking
- Solution: (yes) 3 phase commit!
  - Reliable replacement of crashed “leader”
  - 2PC often good enough in practice

Questions?