Distributed Transactions in MySQL

Percona Live MySQL C&E 2013

Randy Wigginton (Square)

Ryan Lowe (Percona)

Marcos Albe (Percona)

Fernando Ipar (Percona)

What are transactions?

ACID

Atomicity
Each transaction is “all or nothing”.

Consistency
The database will always be in a valid state.

Isolation
Appropriate interaction between transactions.

Durability
Once committed, a transaction is permanent.

Sample (bad) Transaction in SQL

Ryan (id 1234) wants to transfer $1,000,000 to Randy (id 2345)

```
SELECT balance FROM accounts WHERE id=1234; # Verify sufficient funds
UPDATE accounts SET balance = (balance - 1000000) WHERE id=1234;
UPDATE accounts SET balance = (balance + 1000000) WHERE id=2345;
```

Thanks, Ryan!

Sample Transaction in SQL

Ryan (id 1234) wants to transfer $1,000,000 to Randy (id 2345)

```
SELECT balance FROM accounts WHERE id=1234; # Verify sufficient funds
# Right here, Ryan may transfer all of his balance to somebody else
# so Randy's account may be credited with imaginary money
UPDATE accounts SET balance = (balance - 1000000) WHERE id=1234;
UPDATE accounts SET balance = (balance + 1000000) WHERE id=2345;
```

Thanks, Ryan!

Sample Transaction in SQL

Ryan (id 1234) wants to transfer $1,000,000 to Randy (id 2345)

```
SELECT balance FROM accounts WHERE id=1234; # Verify sufficient funds
UPDATE accounts SET balance = (balance - 1000000) WHERE id=1234;
# If the system crashes here, Ryan loses $1,000,000 and
# Randy never gets paid
UPDATE accounts SET balance = (balance + 1000000) WHERE id=2345;
```

Thanks, Ryan!

Sample Transaction in SQL

Ryan (id 1234) wants to transfer $1,000,000 to Randy (id 2345)

```
BEGIN
SELECT balance FROM accounts WHERE id=1234; # Verify sufficient funds
UPDATE accounts SET balance = (balance - 1000000) WHERE id=1234;
UPDATE accounts SET balance = (balance + 1000000) WHERE id=2345;
COMMIT
```

Thanks, Ryan!

Sample Transaction in SQL

Ryan (id 1234) wants to transfer $1,000,000 to Randy (id 2345)

```
BEGIN
SELECT balance FROM accounts WHERE id=1234 FOR UPDATE; # Verify sufficient funds
UPDATE accounts SET balance = (balance - 1000000) WHERE id=1234;
UPDATE accounts SET balance = (balance + 1000000) WHERE id=2345;
COMMIT
```

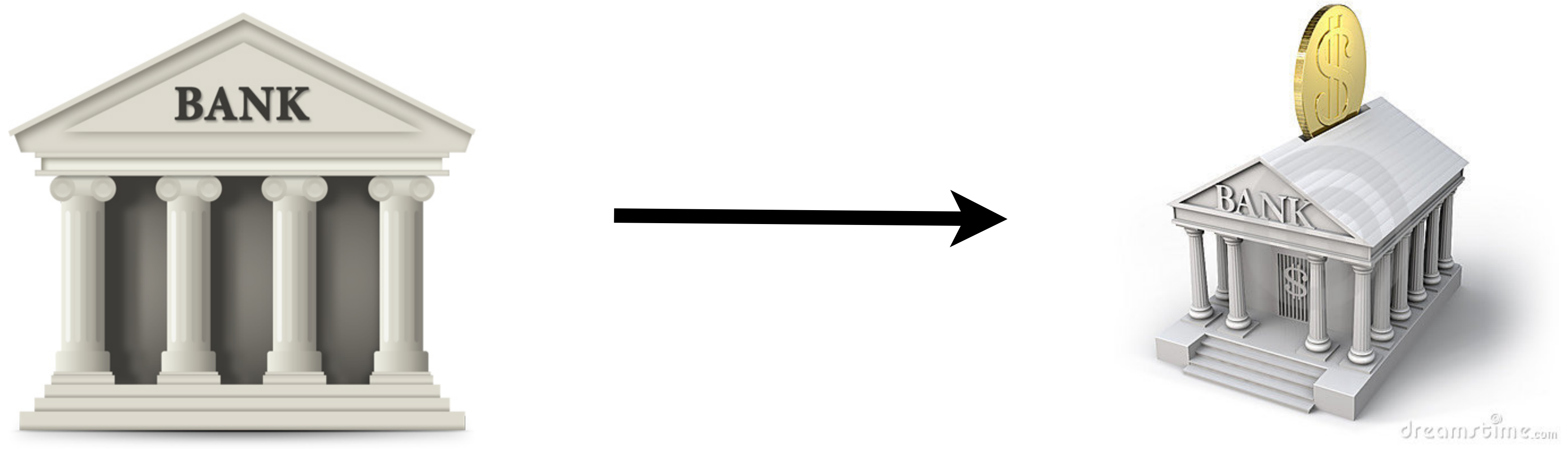
Thanks, Ryan!

What are distributed transactions?

Sample Transaction in SQL

Ryan (id 1234) wants to transfer $1,000,000 to Randy (id 2345)

```
BEGIN
(Chase Bank) SELECT balance FROM accounts WHERE id=1234 FOR UPDATE;
(Chase Bank) UPDATE accounts SET balance = (balance - 1000000) WHERE
id=1234;
(B of A) UPDATE accounts SET balance = (balance + 1000000) WHERE id=2345;
COMMIT
```



Thanks, Ryan!

Distributed Transaction Walkthrough

Actors

Transaction Manager

javax.transaction.xa.XAResource, or Java::BitronixTm
 (aka “Transaction Coordinator”)

Resource Managers

MySQL, PostgreSQL, Oracle, Redis, MQueue, MongoDB, etc.
(aka “cohorts”)

Begin transaction

- Initiated by Coordinator
- Participants
    - Open connection to cohorts
    - Execute normal SQL statements

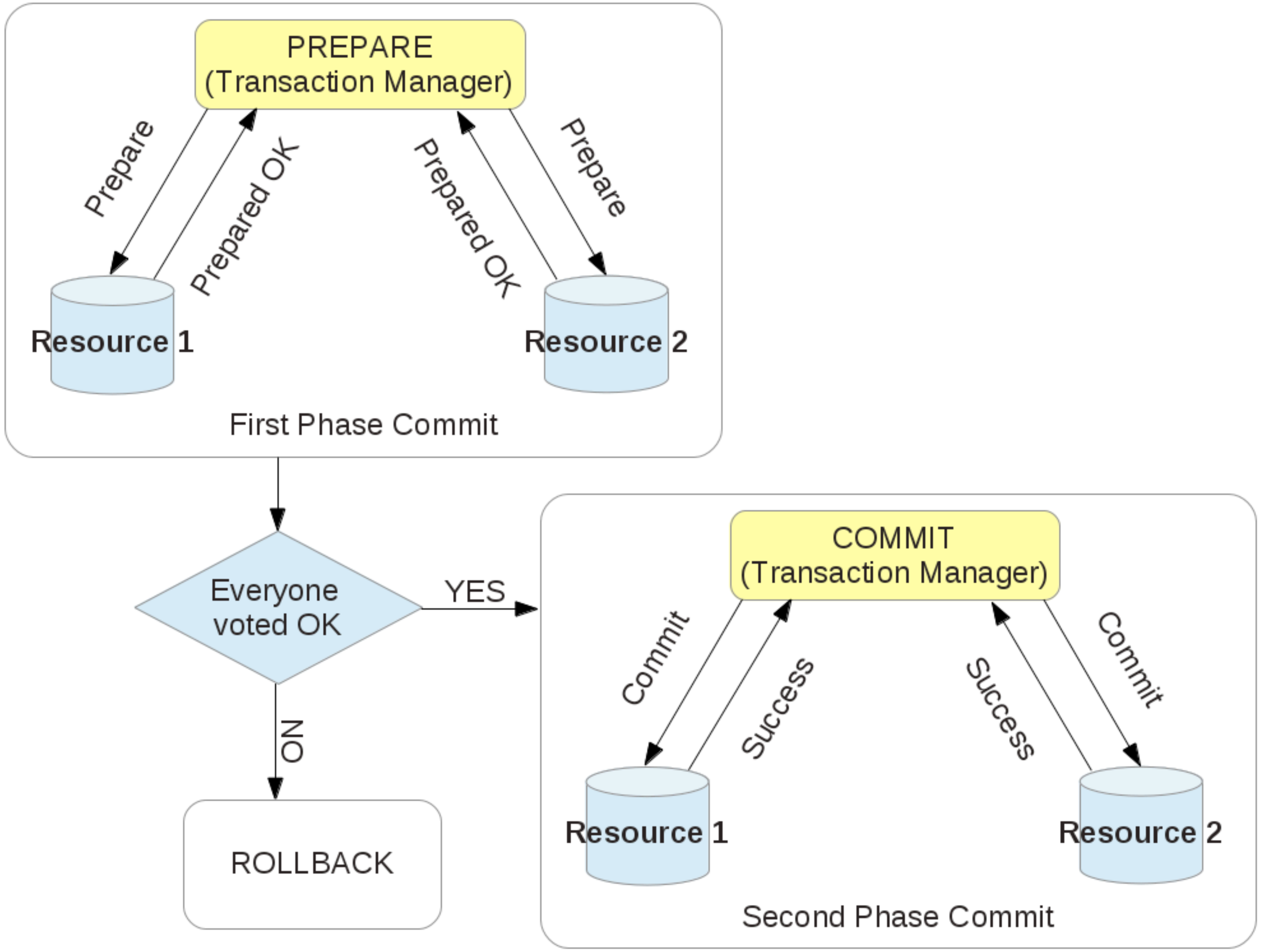
Prepare Phase

- Initiated by Coordinator
- Participants/cohorts
    - Attempt Local Commit
    - Announce whether able to commit
- Vote

COMMIT

<=> Everybody Votes Yes!

(Otherwise Rollback)


PREPARE
(Transaction Manager)
Prepare
Prepared OK
Prepared OK
Prepare
Resource 1
Resource 2
First Phase Commit
Everyone voted OK
YES
NO
ROLLBACK
COMMIT
(Transaction Manager)
Commit
Success
Success
Commit
Resource 1
Resource 2
Second Phase Commit

XA Transactions

- Open Group (X/Open)
- XA == eXtended Architecture
- Uses Two-Phase Commit (2PC)

XA In MySQL

- Only a Resource Manager (or cohort)
- InnoDB only
- innodb_support_xa
- Also Used Internally

http://tinyurl.com/xa-perf-boost

XA In MySQL
(Restrictions)

- InnoDB Only
- Full Specification Not Supported
- dev.mysql.com/doc/refman/5.6/en/xa-restrictions.html

XA In MySQL
(Failure Scenarios)

Replication
In certain (crash) scenarios, a successful COMMIT may not make it into the binary log.

Edge Cases
While one site is in the “prepared to commit” state, the other may be in the “commit” or “abort” state.

XA In MySQL

Command syntax

XA Start to begin a transaction
XA End puts transaction into idle (abort on failure)
XA Prepare to perform local commit, return success or failure
XA Commit if all cohorts vote yes (all succeed prepare)
XA Rollback if any cohort votes no

...and also XA Recover

XA In MySQL

Java code

```
MysqlXADataSource srcDB = setup.getXADataSource();
XAConnection sourceXAConnection = srcDB.getXAConnection();
Connection conn1 = sourceXAConnection.getConnection();
XAResource xar1 = sourceXAConnection.getXAResource();
Xid xid1 = createXid("globally_unique_id");
xar1.start(xid1, XAResource.TMNOFLAGS);

 (similar code for destination)
```

XA In MySQL

Java code

```
xar1.end(xid1, XAResource.TMSUCCESS);
xar2.end(xid2, XAResource.TMSUCCESS);
int prp1 = xar1.prepare(xid1);
int prp2 = xar2.prepare(xid2);
if (prp1 != XAResource.XA_OK || prp2 != XAResource.XA_OK)
{
  xar1.rollback(xid1);
  xar2.rollback(xid2);
} else {
  xar1.commit(xid1, false);
  xar2.commit(xid2, false);
}
```

XA In MySQL

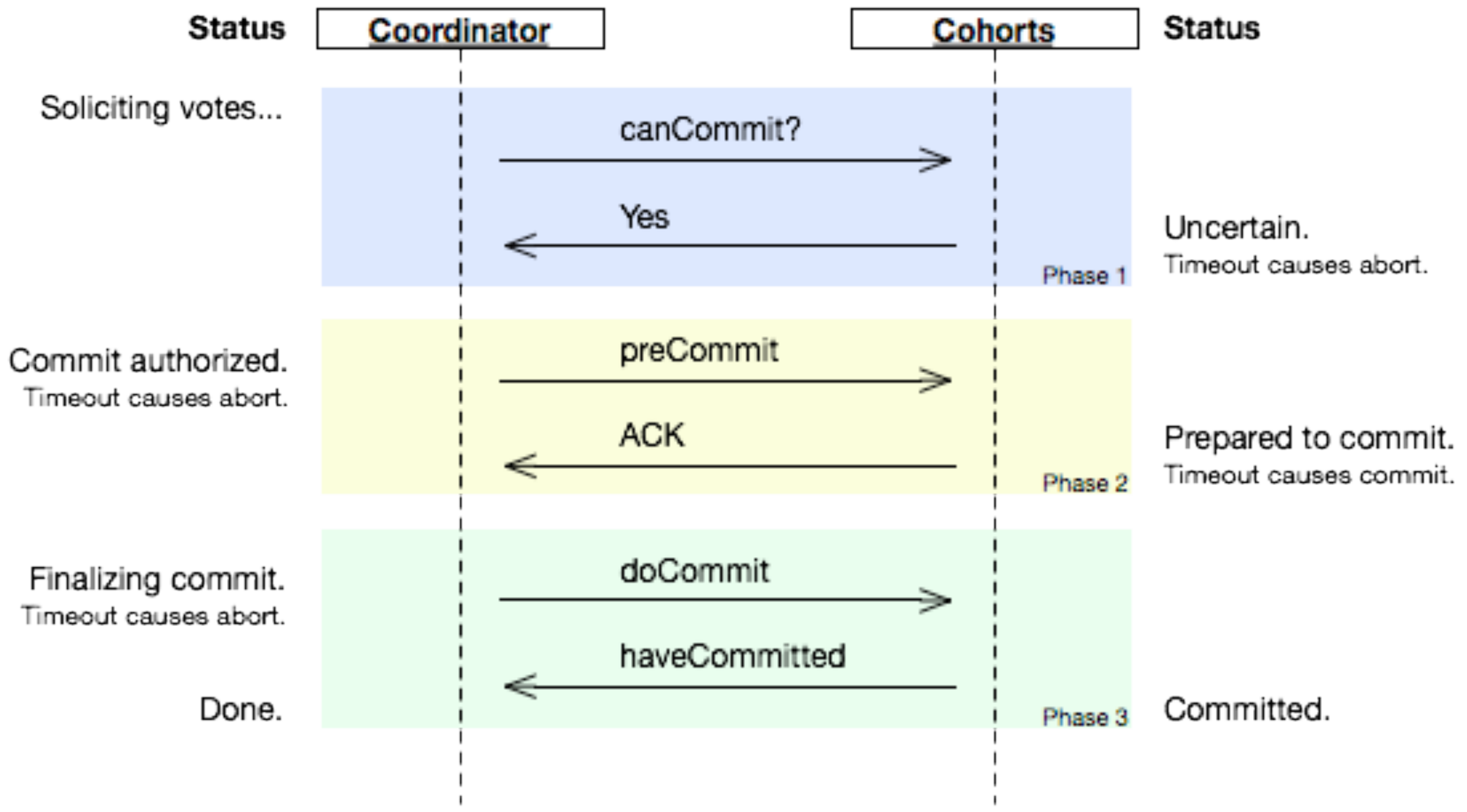
Ruby code

```
c1 = ds1.get_connection
c2 = ds2.get_connection
btm = TxnSvc.get_transaction_manager
btm.begin
begin
  <usual SQL statements>
  btm.commit
  puts "Successfully committed"
rescue
  puts "Something bad happened: " + $!
  btm.rollback
end
```

3 Phase Commit

Three Phase Commit

Source: wikipedia



Extended 3 Phase Commit

Extended Three Phase Commit

Source: wikipedia

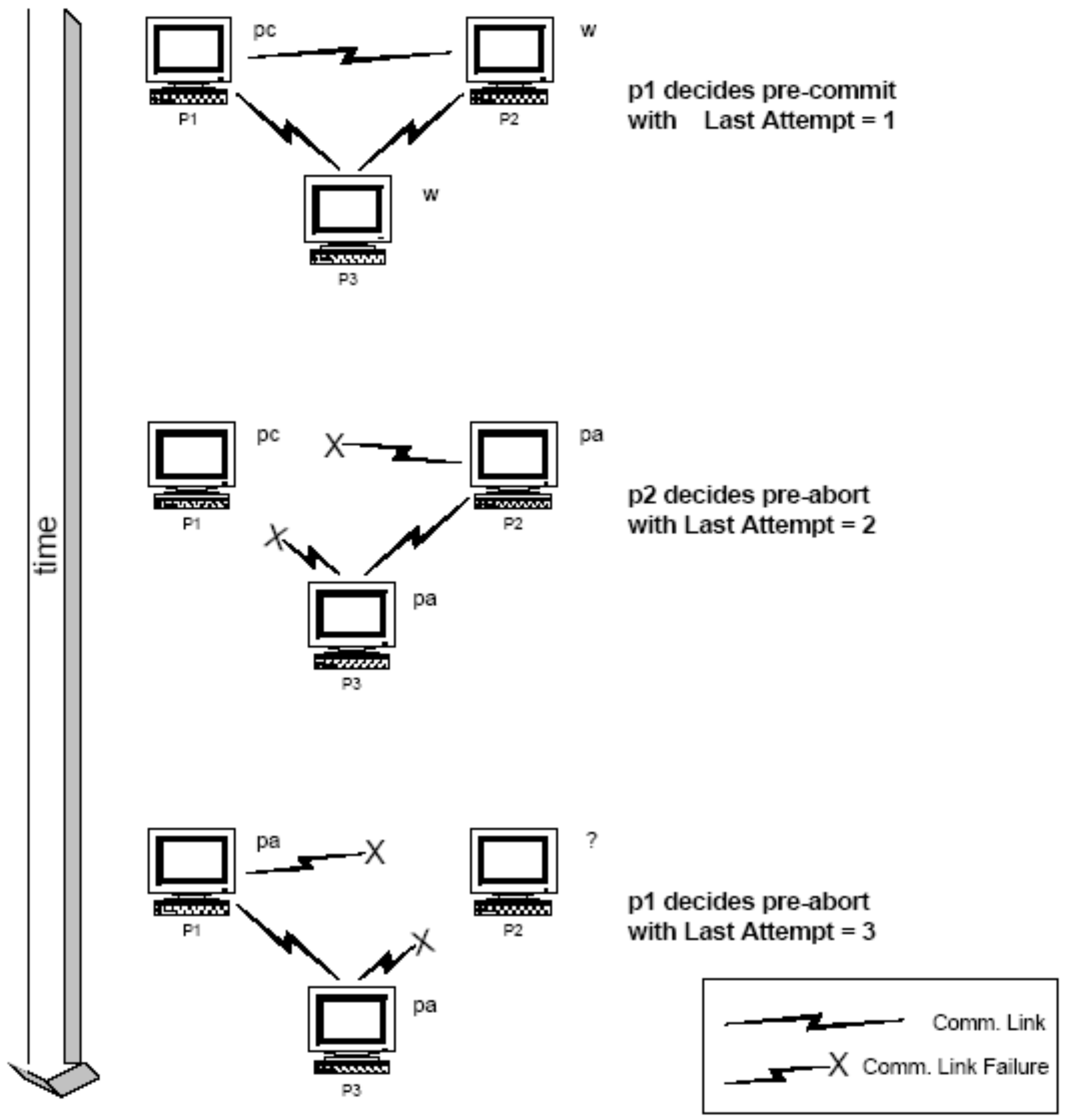


Figure 9: E3PC does not Block a Quorum

4 Phase Commit?

What if I have no other choice?

Does it hurt?

Performance with Two Databases

Threads	Single DB	Multi DB	Penalty
1	3760	43922	1068%
2	2083	22962	1002%
4	1387	16127	1063%
8	854	10849	1170%
16	891	11317	1170%

All times in milliseconds, smaller is better

Performance with Five Databases

Threads	Single DB	Multi DB	Penalty
1	4180	46298	1008%
2	2148	22993	970%
4	1300	16592	1176%
8	863	10927	1166%
16	992	11514	1061%

Links

- docs.codehaus.org/display/BTM/Home
- github.com/fipar/plmce13_xa_examples (ruby)
- github.com/squarenerd/distributed_txn (java)

Distributed Transactions are evil

Don’t do it!

Questions?

rlowe@pablowe.net
rwigginton@squareup.com
marcos.albe@percona.com