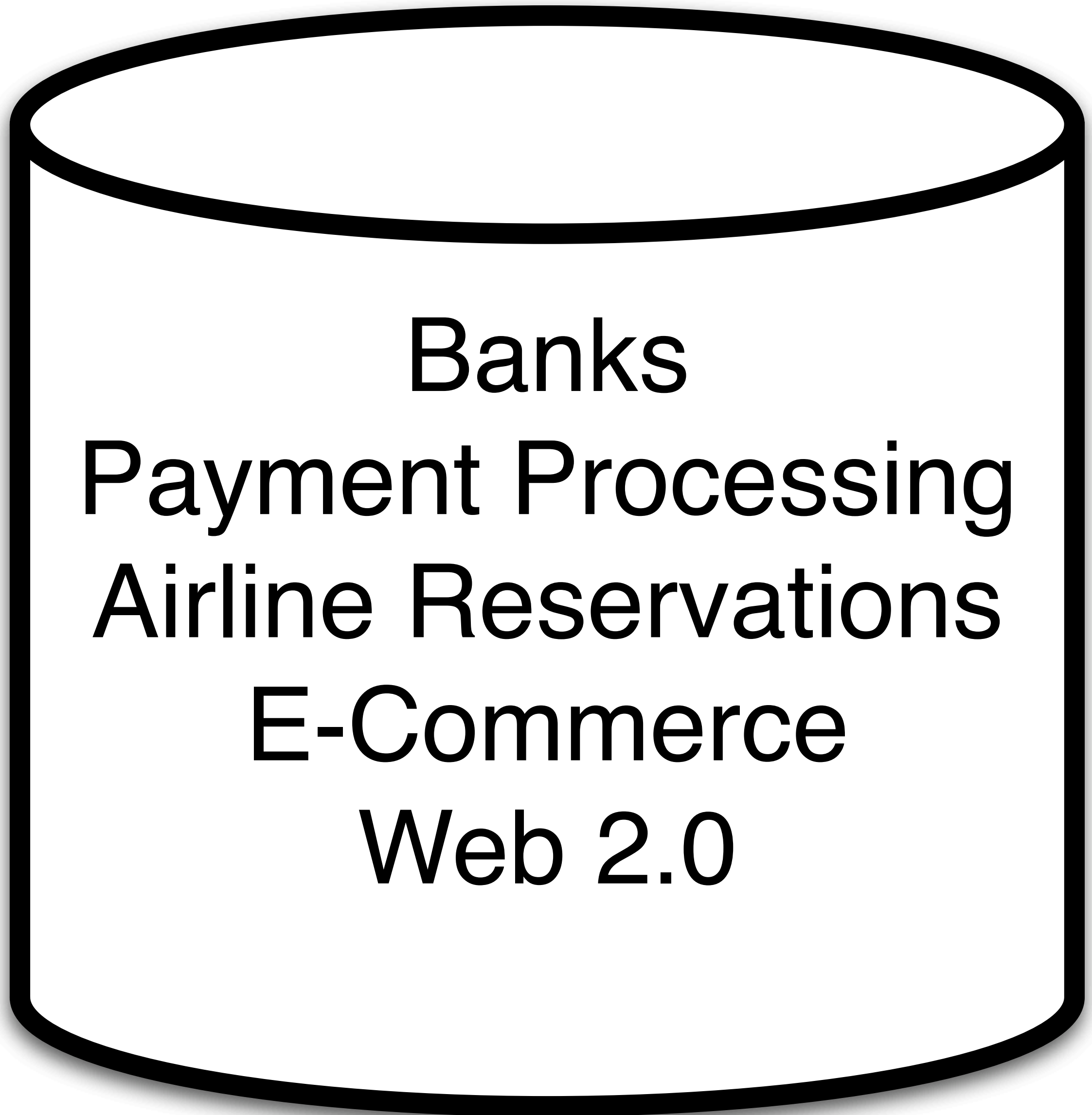
Low Overhead Concurrency Control for Partitioned Main Memory Databases

Evan P. C. Jones   Daniel J. Abadi   Samuel Madden


Banks
Payment Processing
Airline Reservations
E-Commerce
Web 2.0

Problem:

Millions of transactions per second

Problem:

Millions of transactions per second




Problem:

Millions of transactions per second

=

$$$$

Alternative: H-Store Project

Redesign specifically for OLTP

Prototype: ~10X throughput

Idea: Remove un-needed features

Source: Stonebraker et. al, “The End of an Architectural Era”, VLDB 2007.

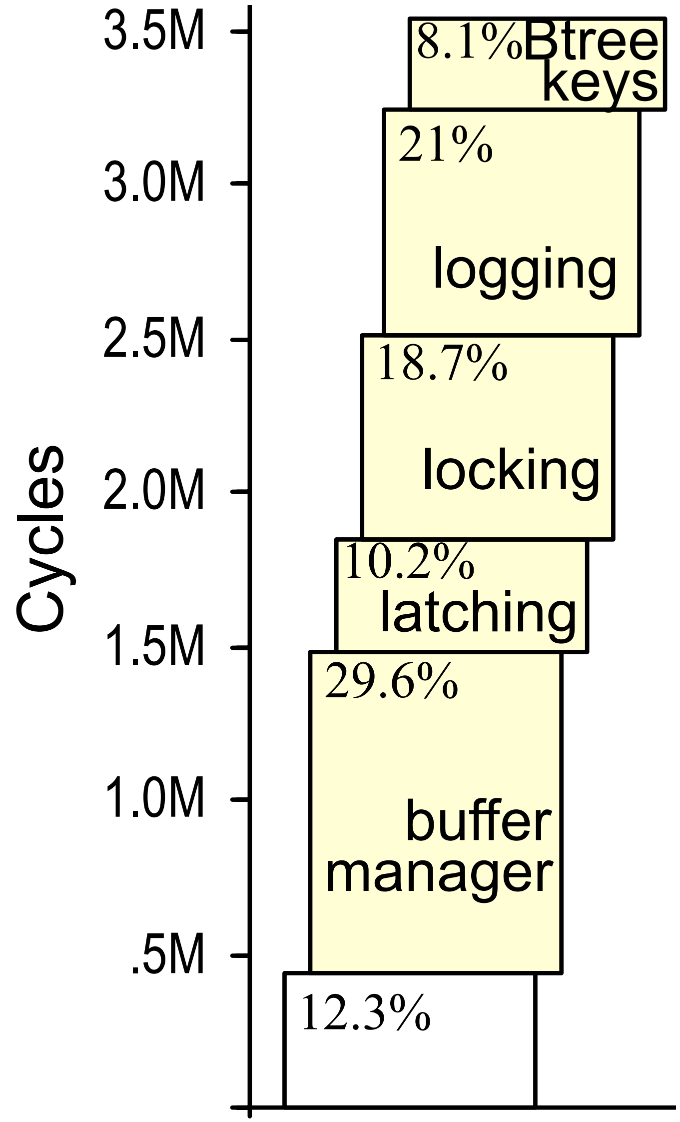
H-Store: High Throughput OLTP

Redesign DB specifically for OLTP

Prototype: ~10X throughput

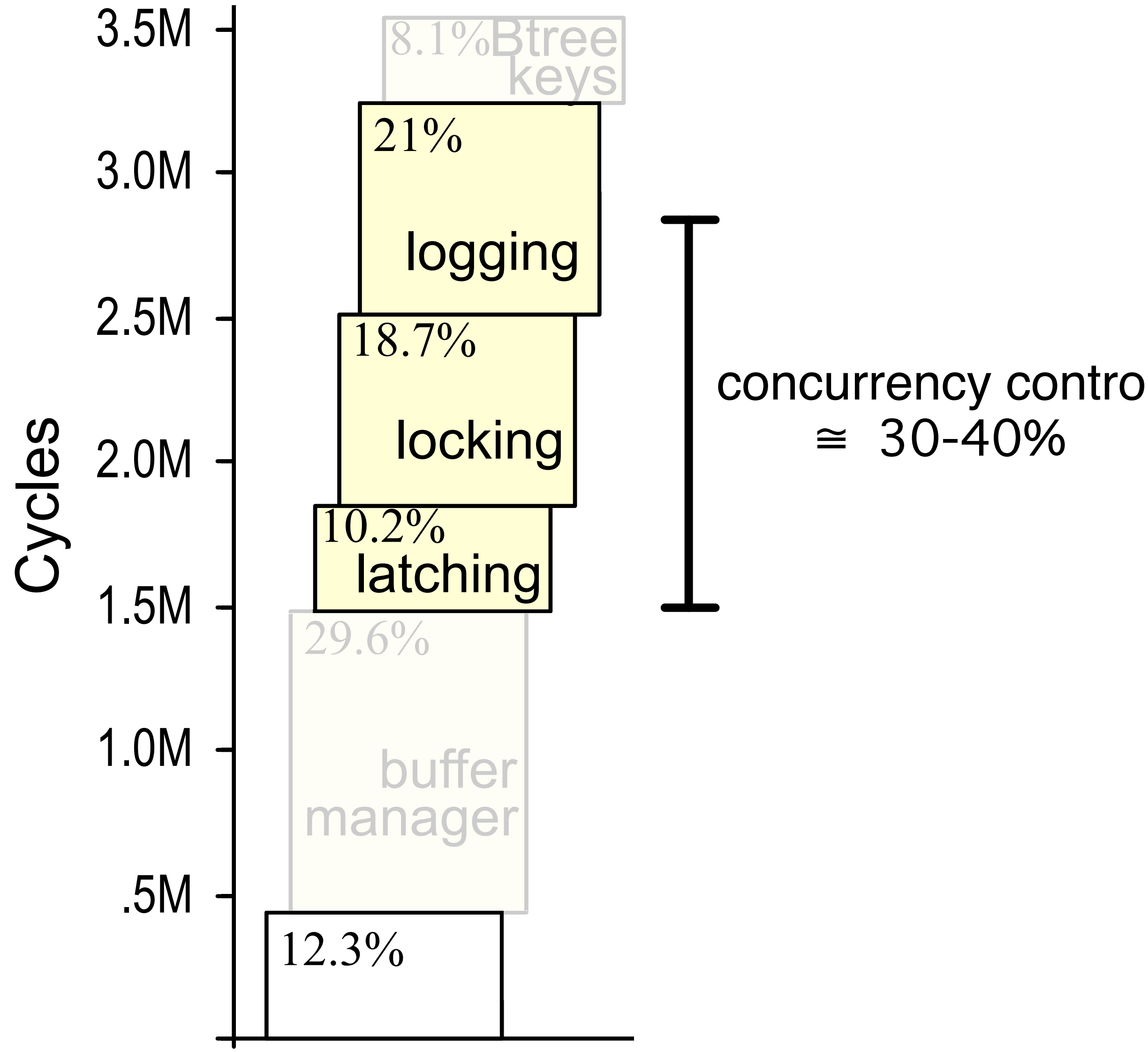
Main memory database

Concurrency control consumes ~30-40% of CPU time



CPU Cycle Breakdown for Shore on TPC-C New Order

Source: Harizopoulos, Abadi, Madden and Stonebraker, “OLTP Under the Looking Glass”, SIGMOD 2008



CPU Cycle Breakdown for Shore on TPC-C New Order

Source: Harizopoulos, Abadi, Madden and Stonebraker, "OLTP Under the Looking Glass", SIGMOD 2008

Speculative Concurrency Control

Eliminate fine-grained access tracking (locks or read/write sets)

Eliminate undo logs (where possible)

Up to 2X faster than locking for appropriate workloads

Why Support Concurrency?

Use idle resources:

disk stalls	main memory
user stalls	stored procedures

Physical resources:

multiple CPUs	partition per core
multiple disks	

Long running txns: don’t do them

H-Store: Single thread engine

Assumptions:

Database divided into partitions

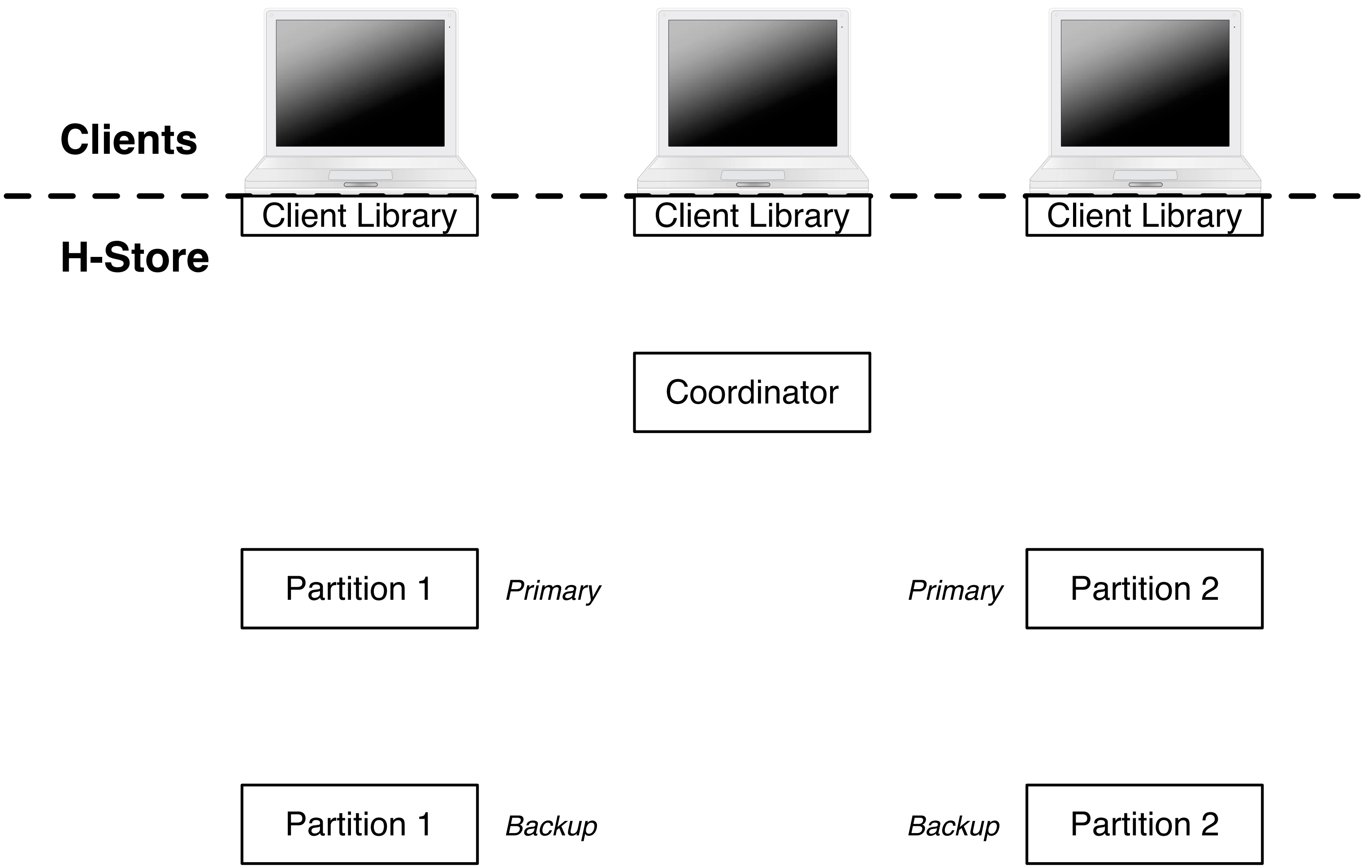
Transactions access one partition (mostly)

Mapping procedures to partitions is given

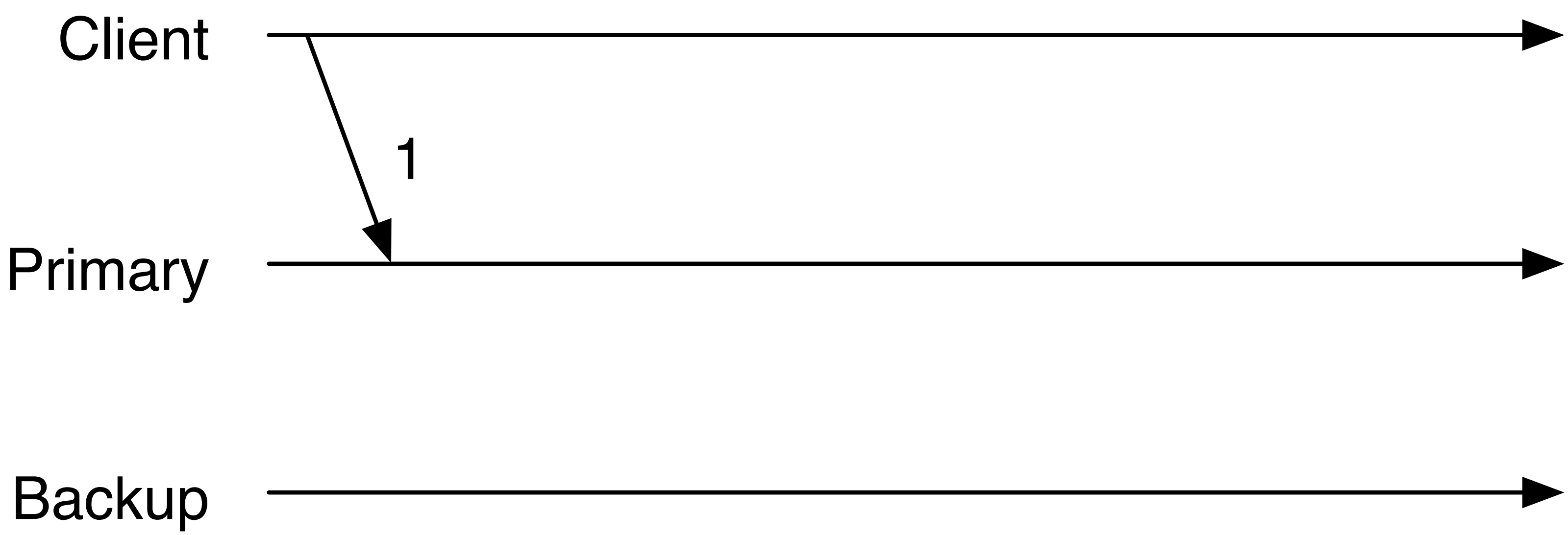
Total data fits in memory of N machines

Partitions are replicated on 2 machines

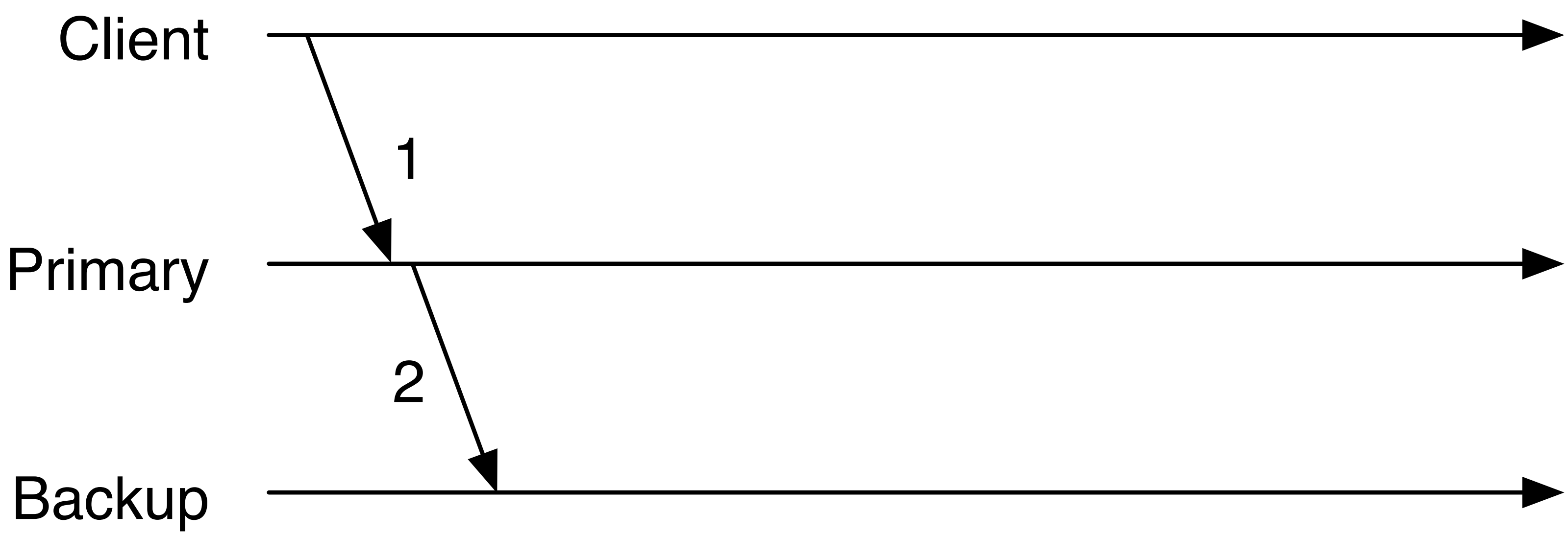
System Overview



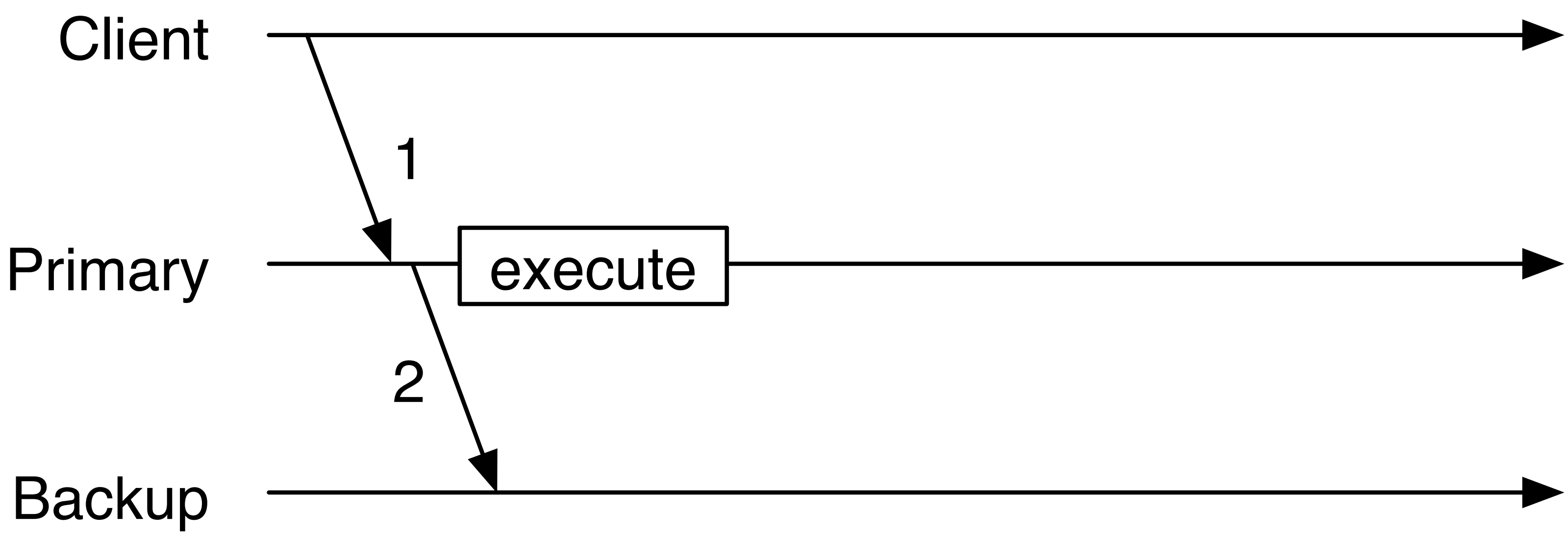
Single Partition Transaction



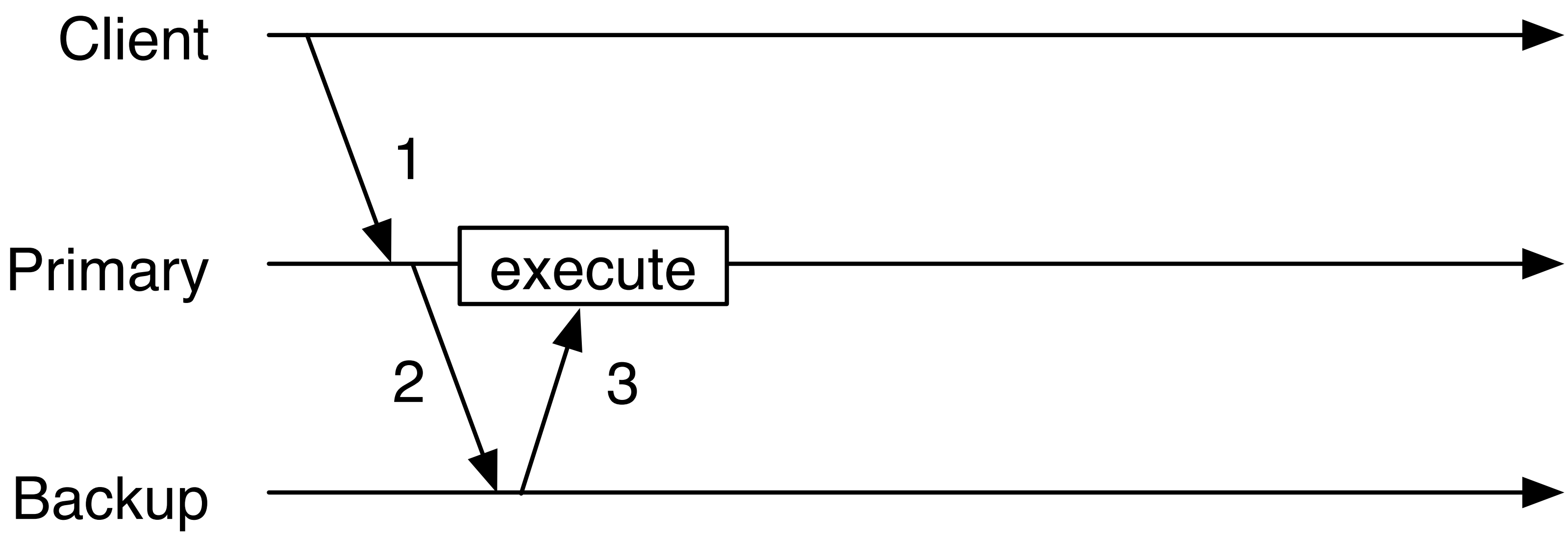
Single Partition Transaction



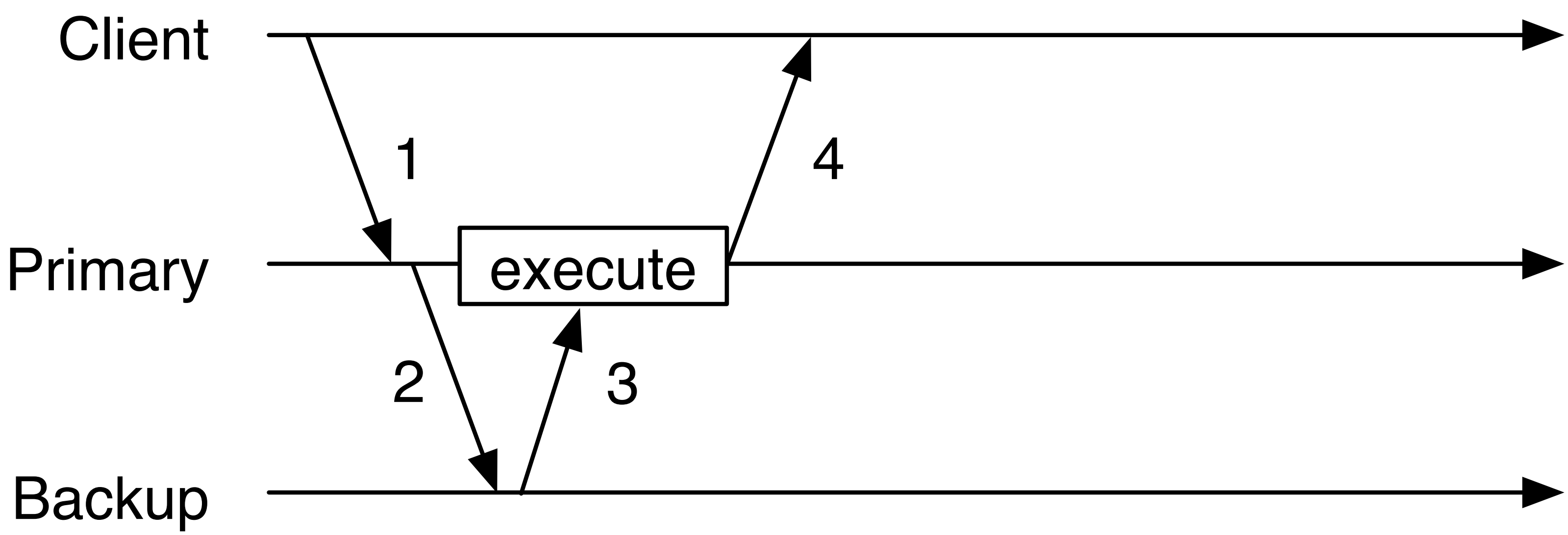
Single Partition Transaction



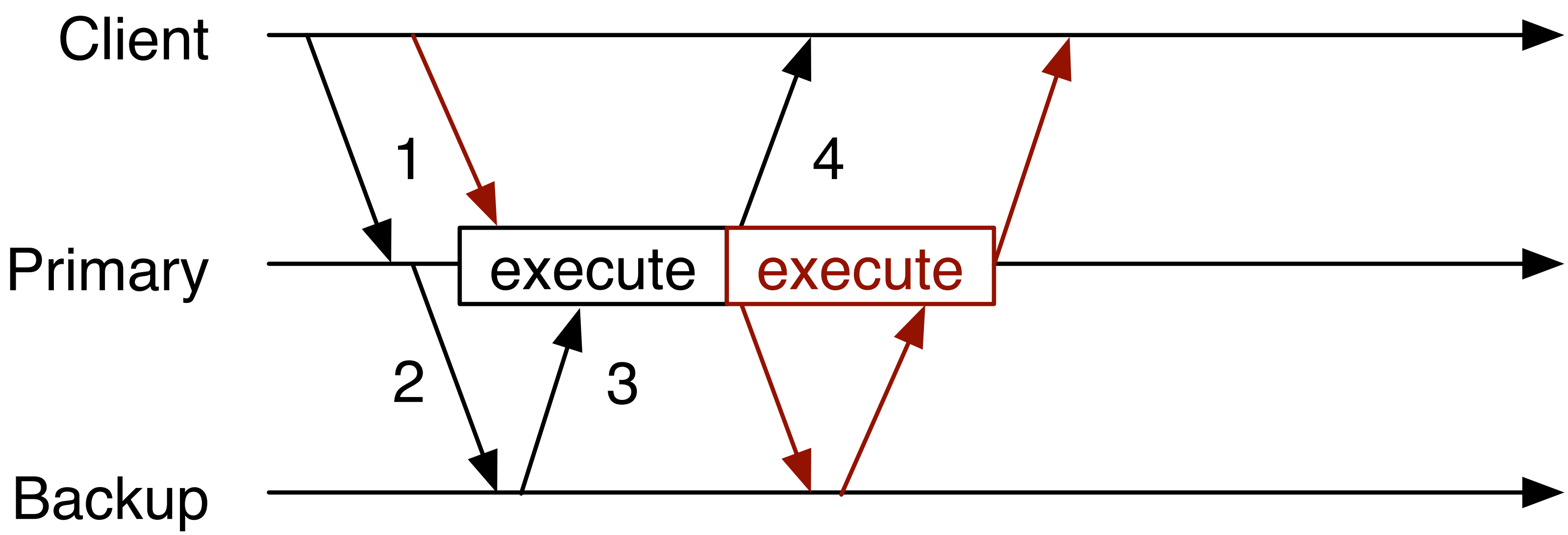
Single Partition Transaction



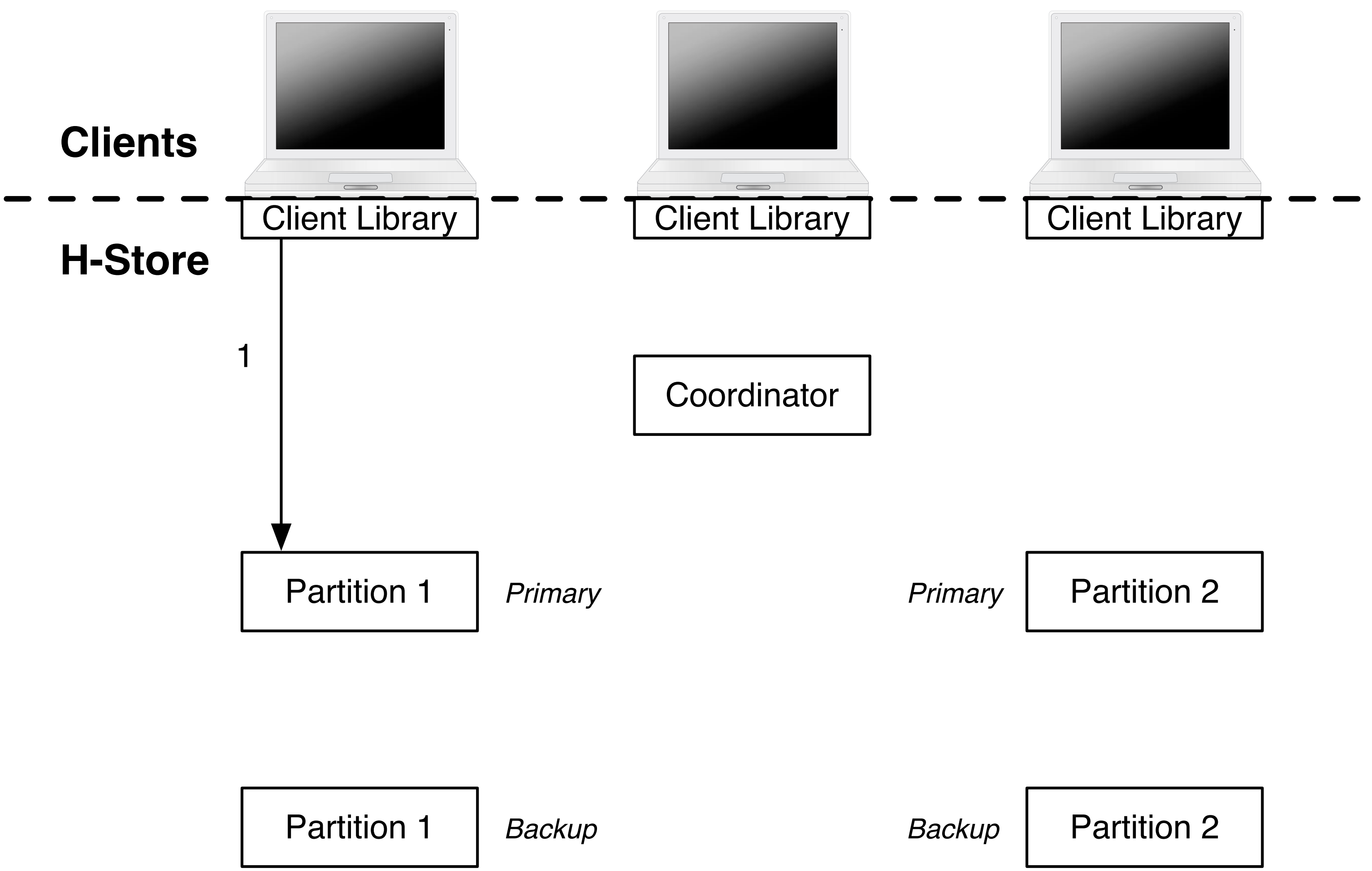
Single Partition Transaction



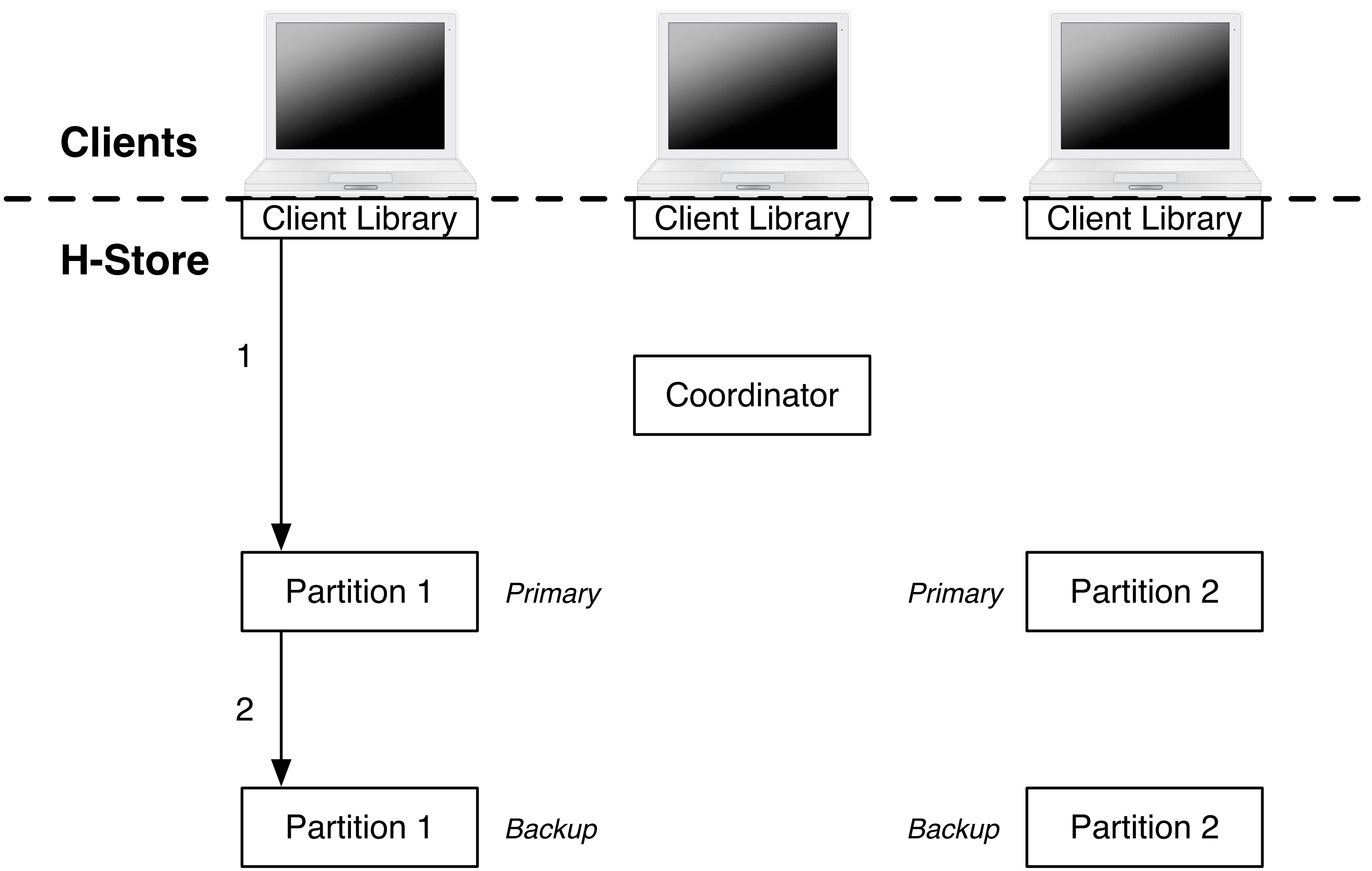
Single Partition Transaction



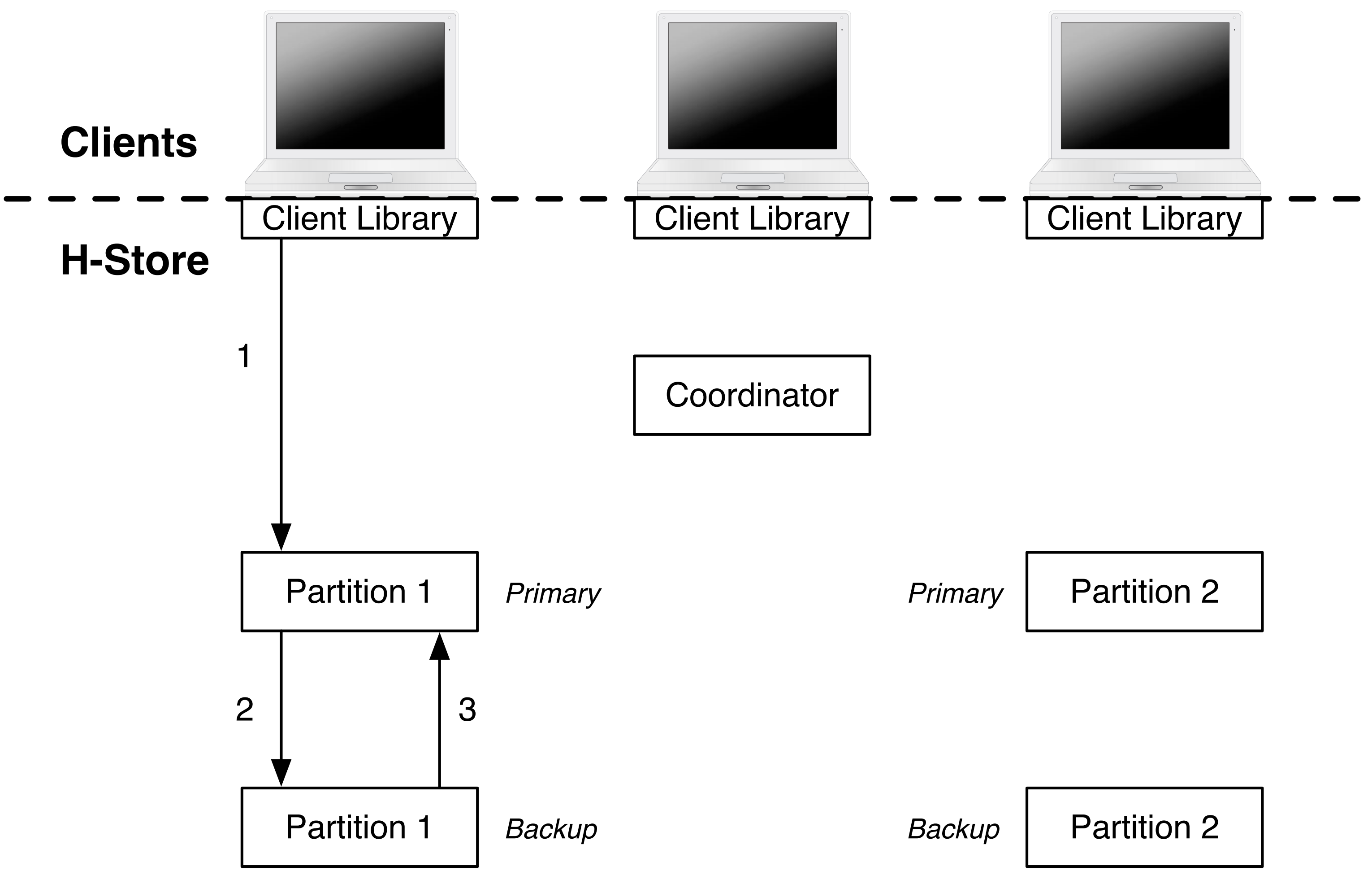
Single Partition Transaction



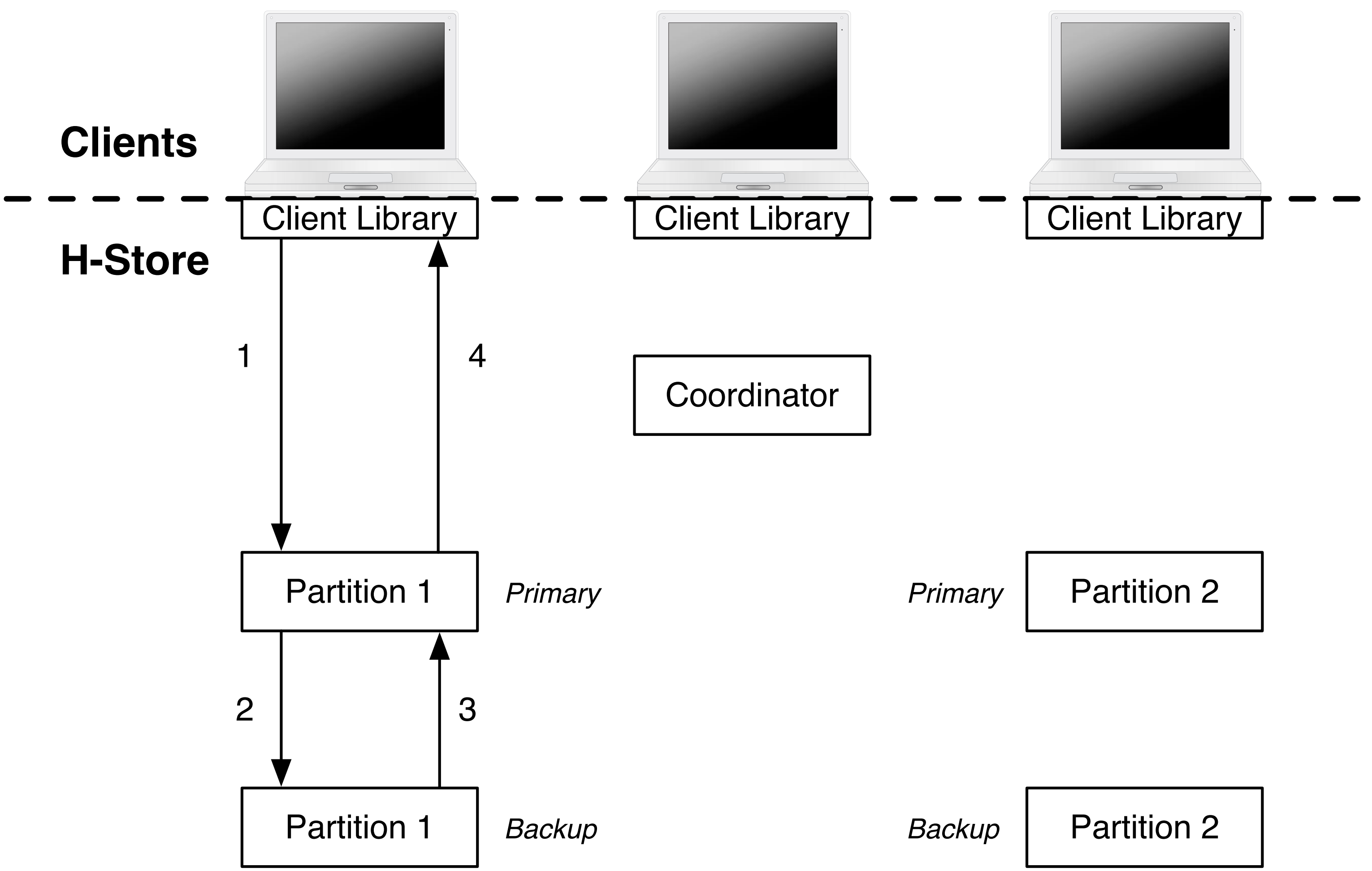
Single Partition Transaction



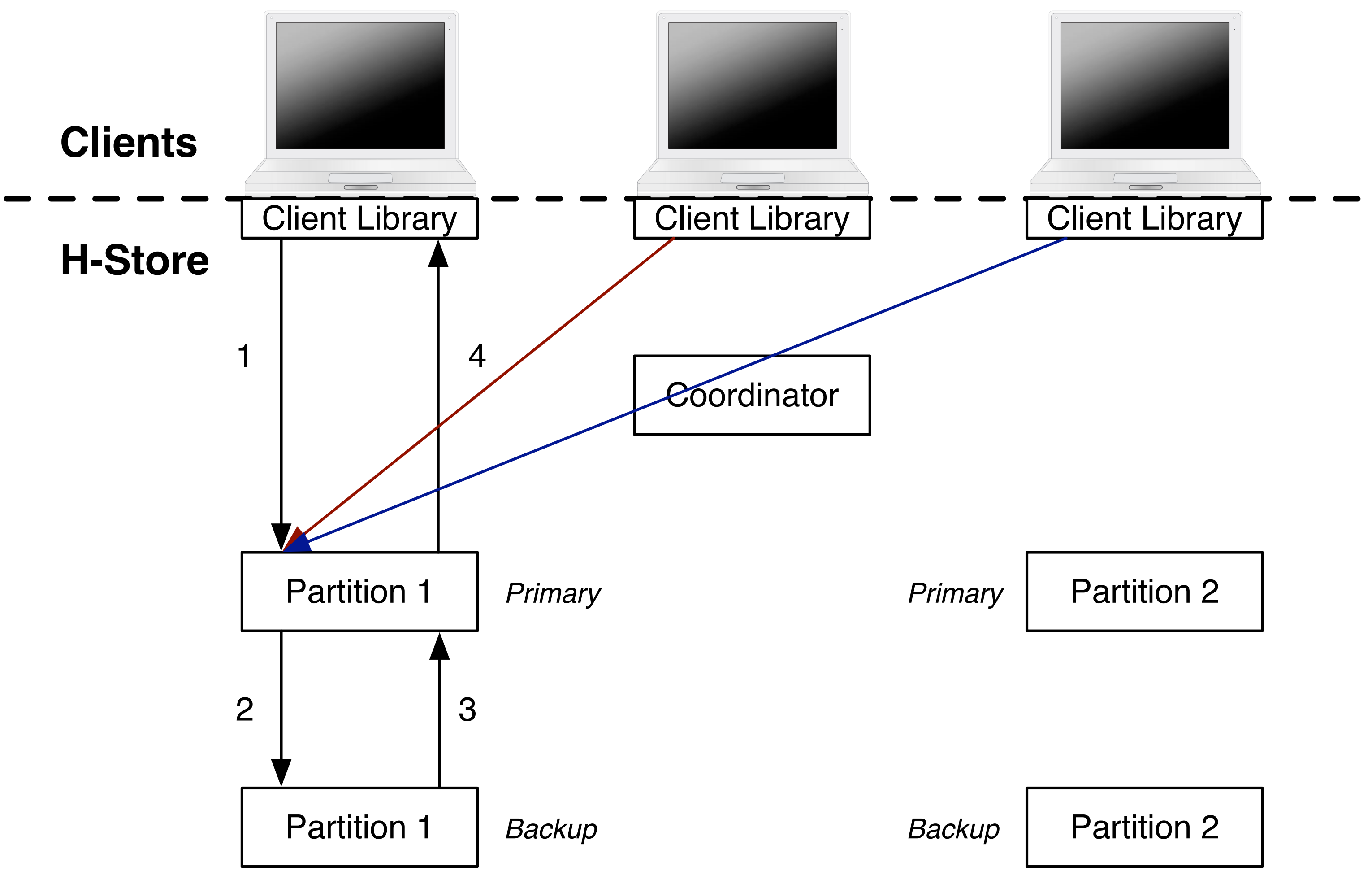
Single Partition Transaction



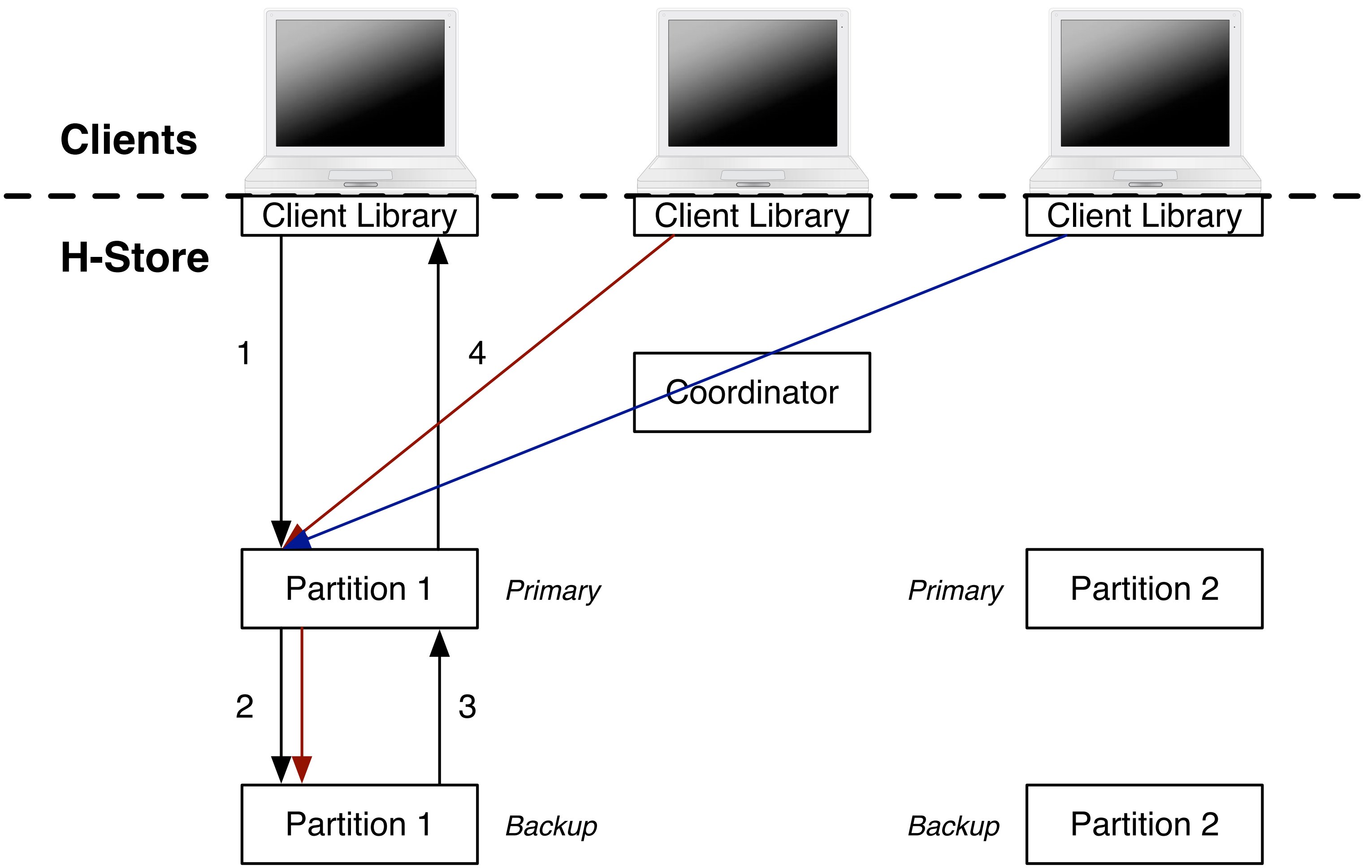
Single Partition Transaction



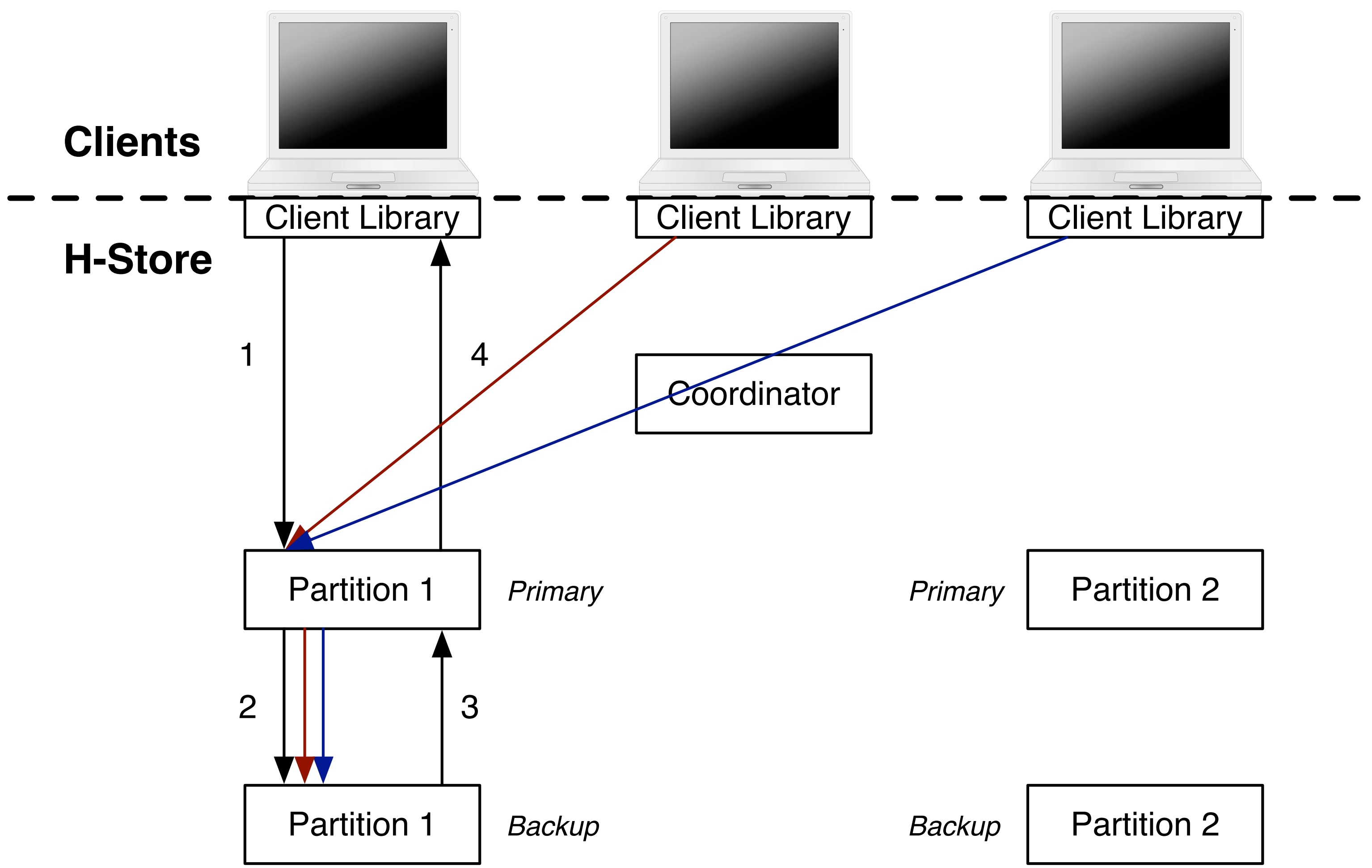
Single Partition Transaction



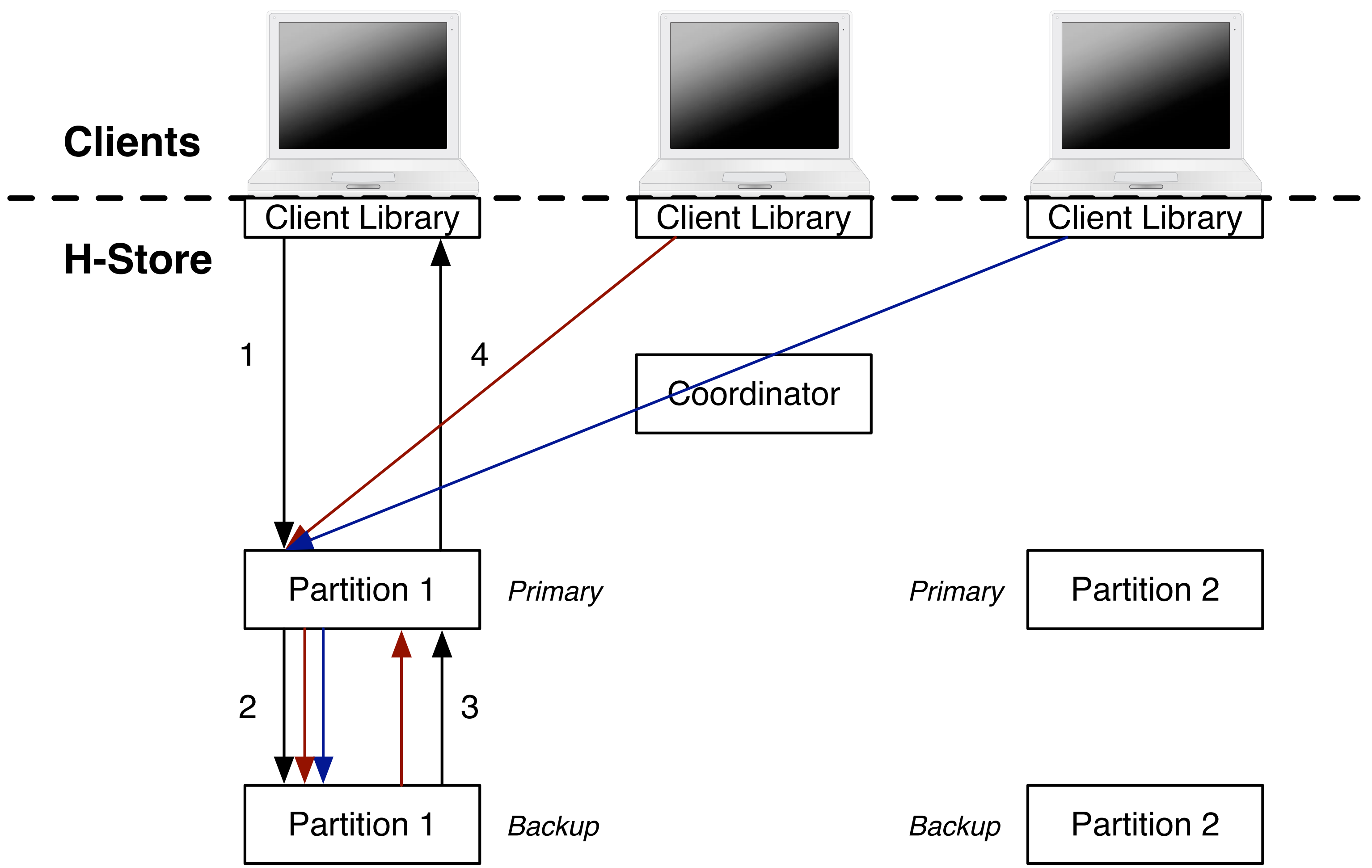
Single Partition Transaction



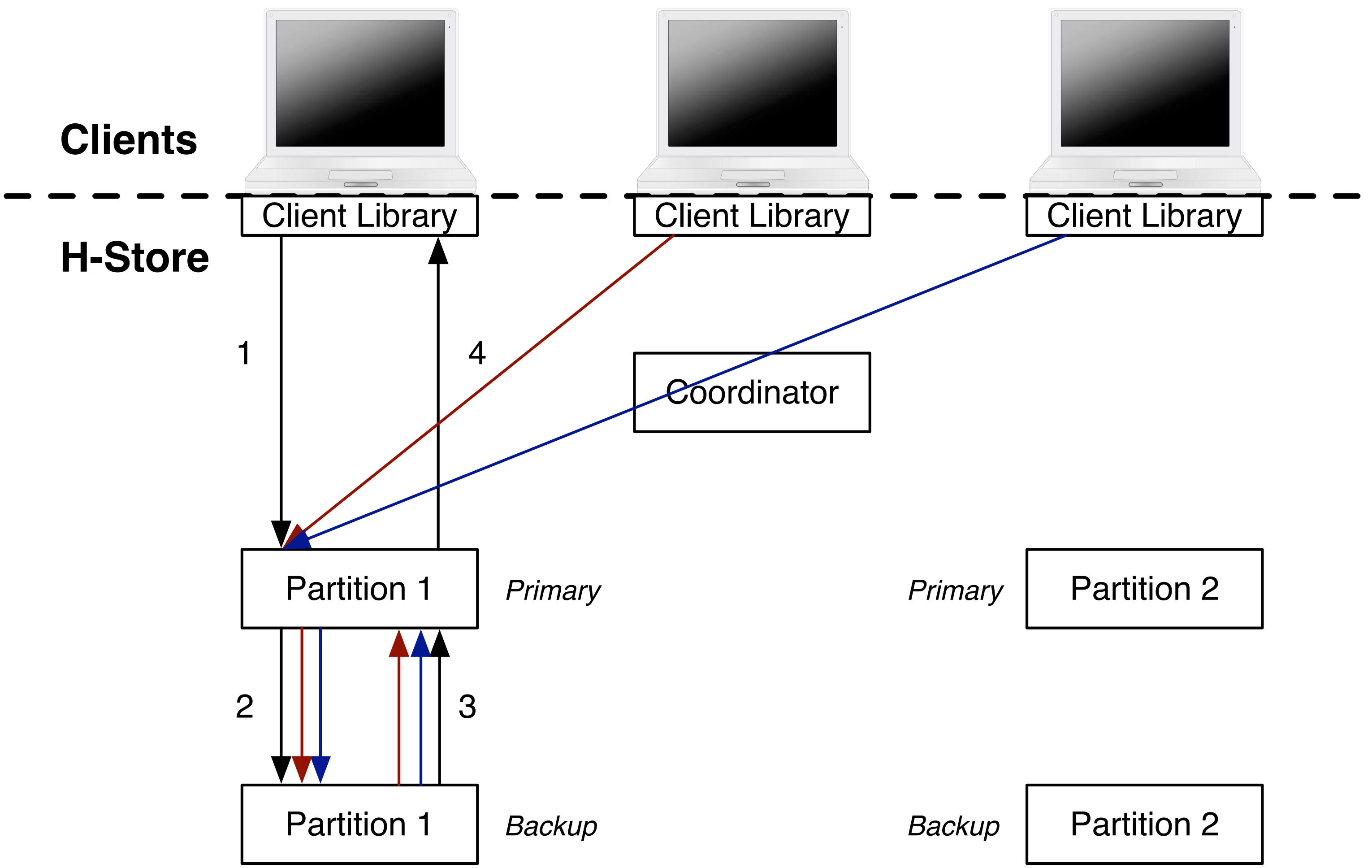
Single Partition Transaction



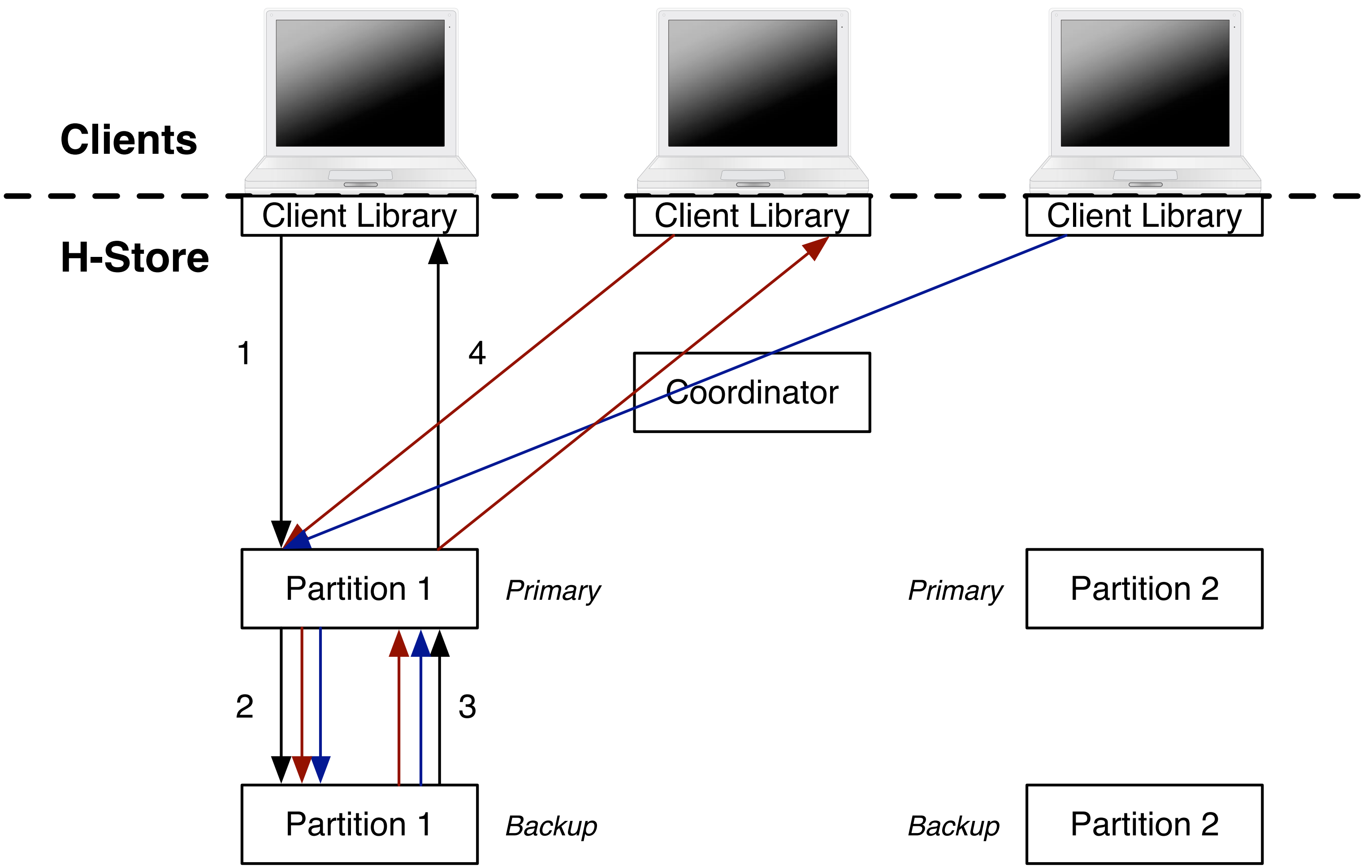
Single Partition Transaction



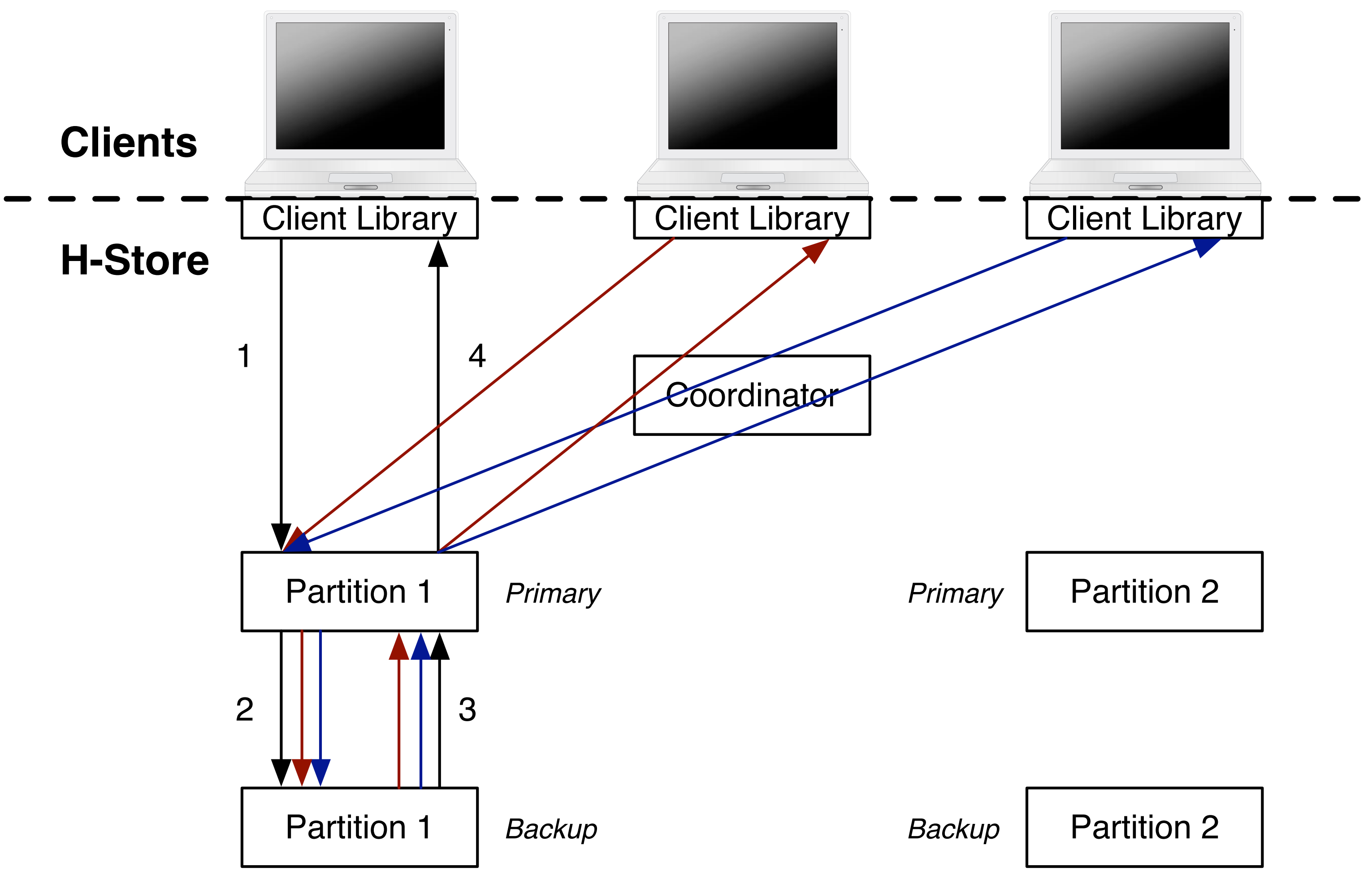
Single Partition Transaction



Single Partition Transaction



Single Partition Transaction



Not Perfectly Partionable?

Example: users and groups

Many applications are *mostly* partitionable

e.g. TPC-C: 11% multi-partition transactions

Distributed Transactions

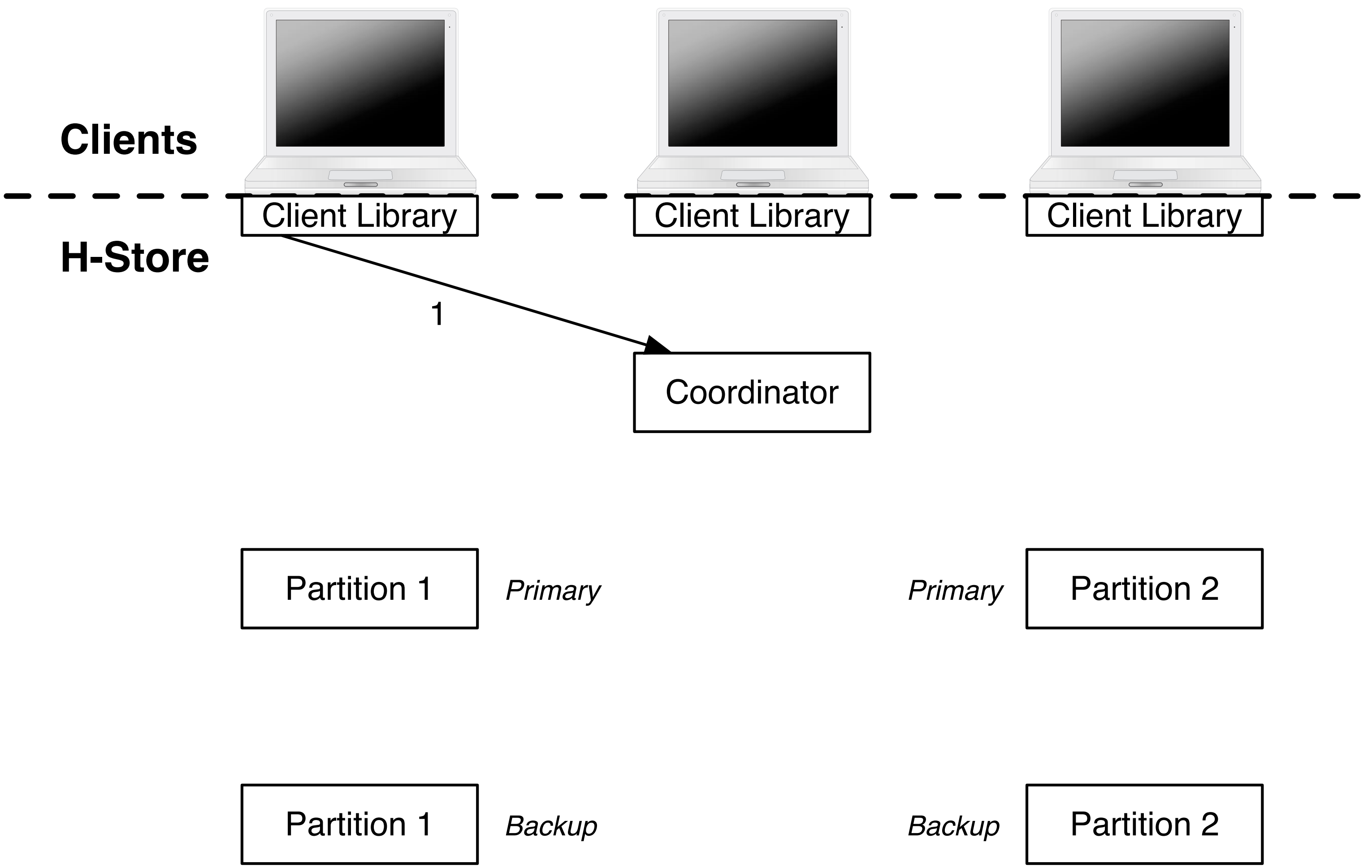
Need two-phase commit (consensus)

Simple solution:

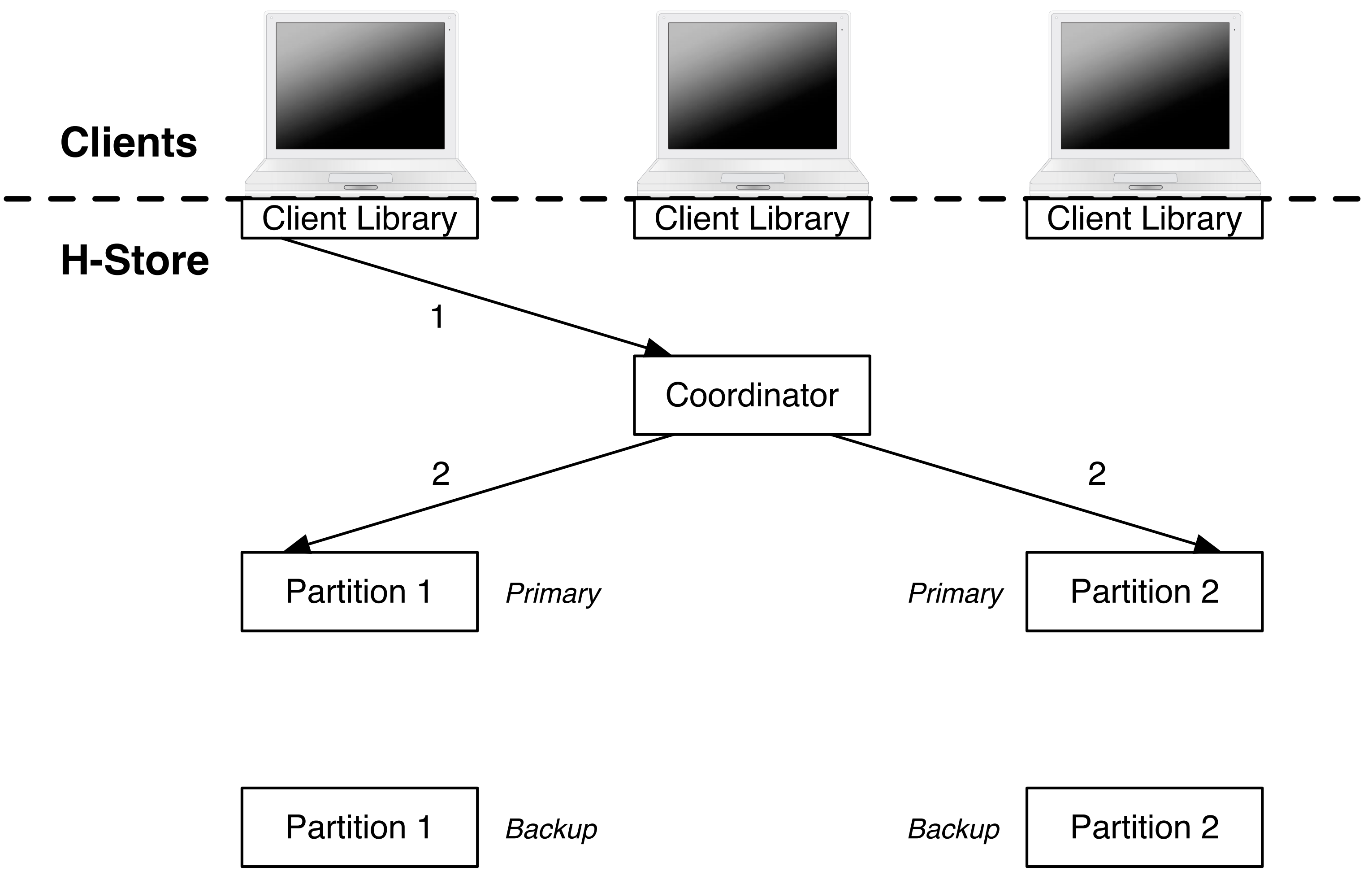
block until the transaction finishes

Introduces network stall (**bad**)

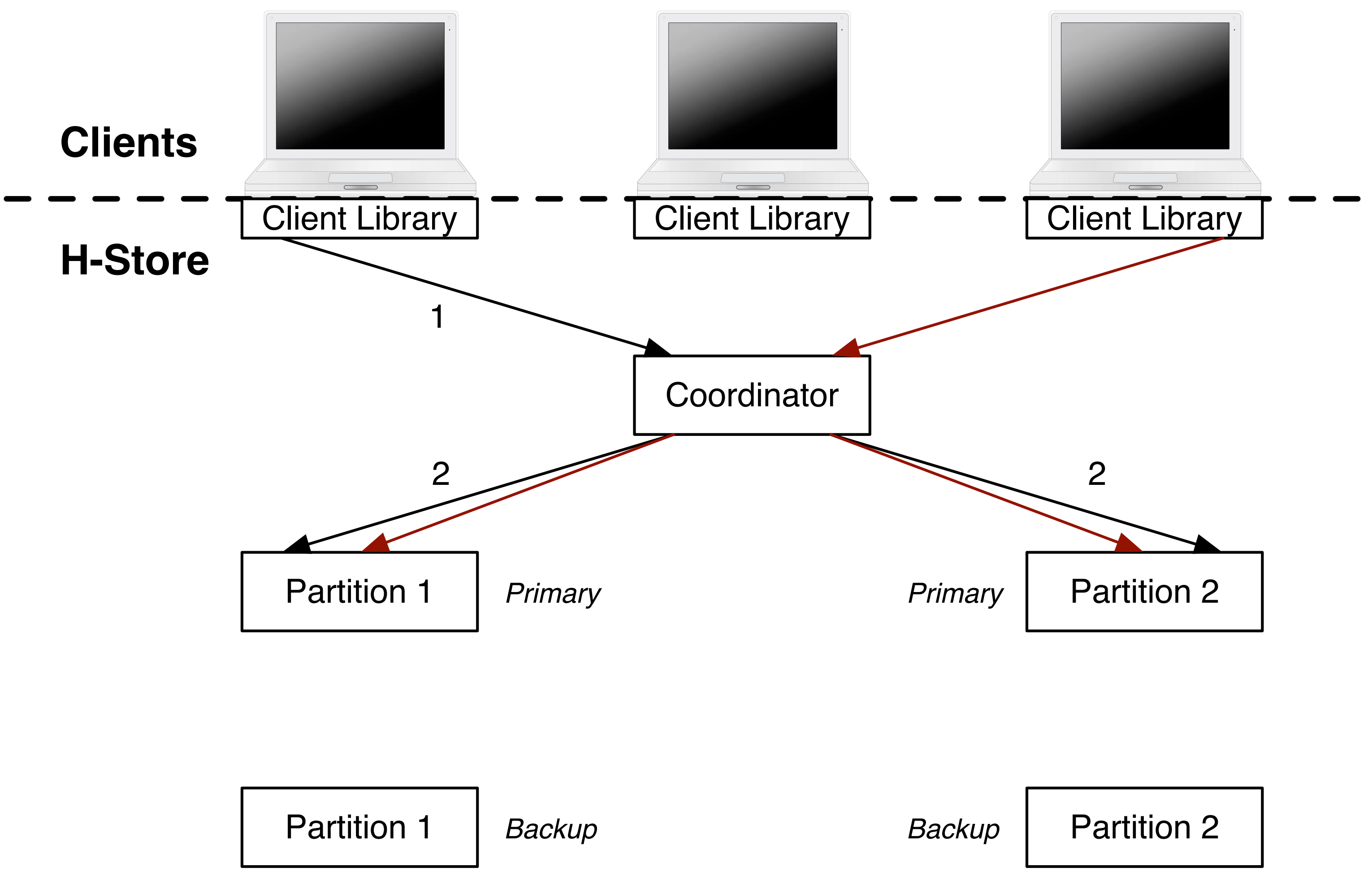
Blocking Multi-Partition



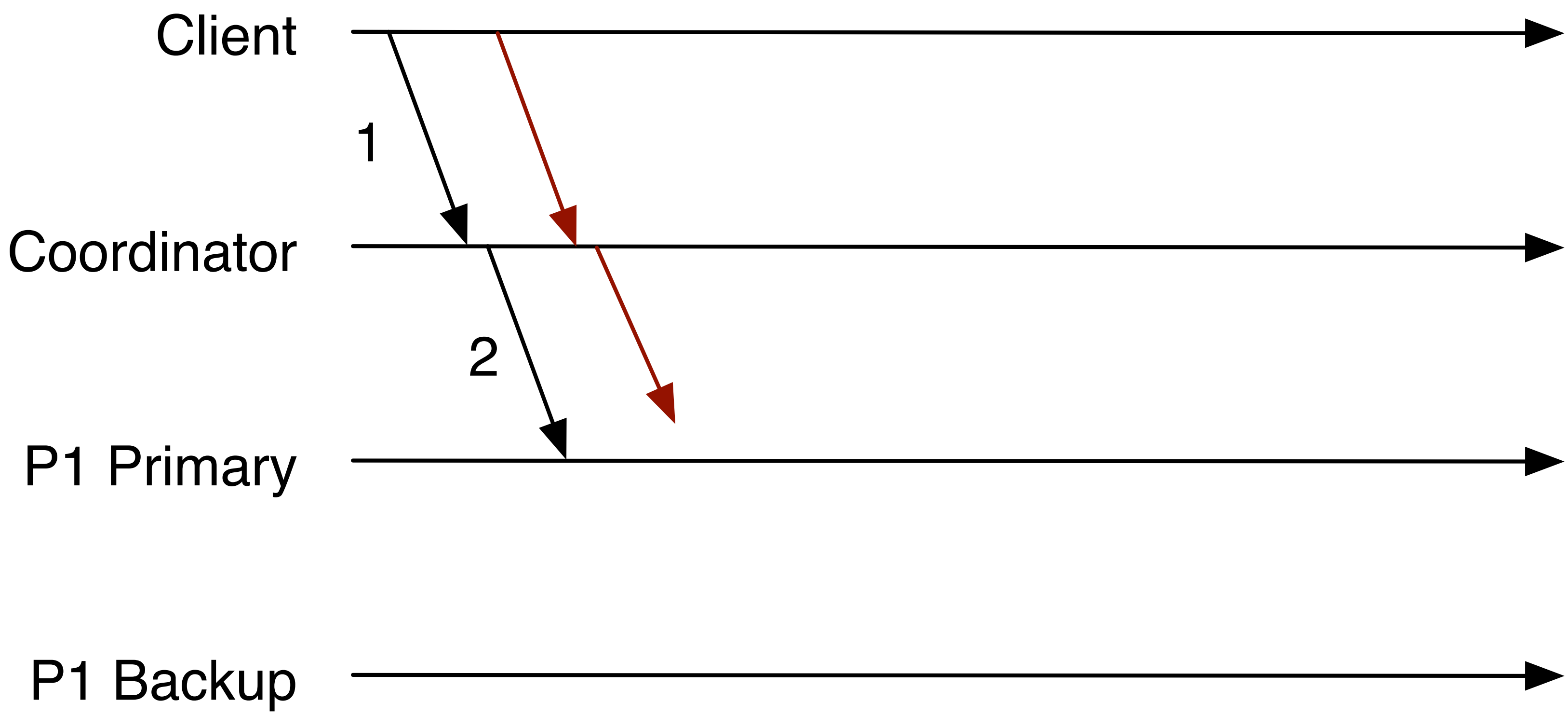
Blocking Multi-Partition



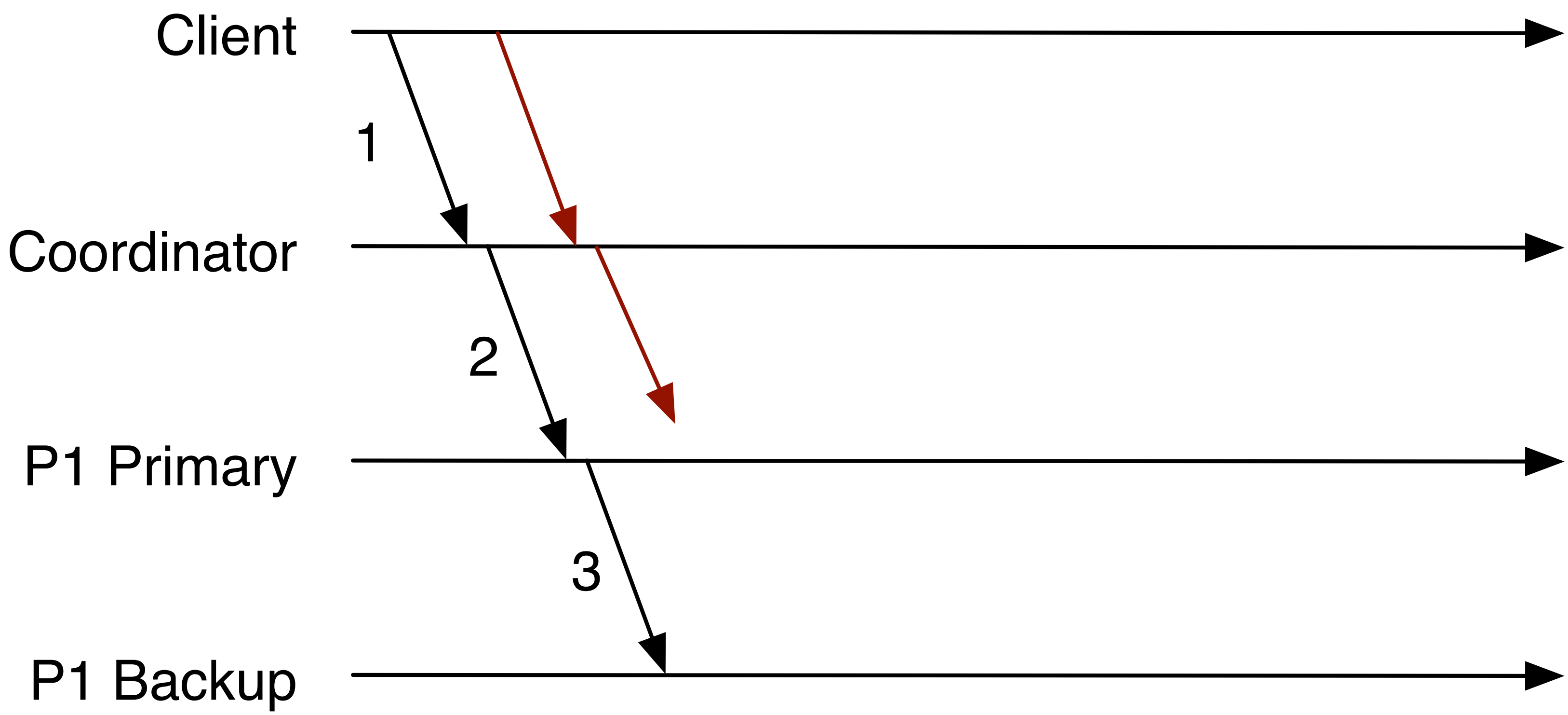
Blocking Multi-Partition



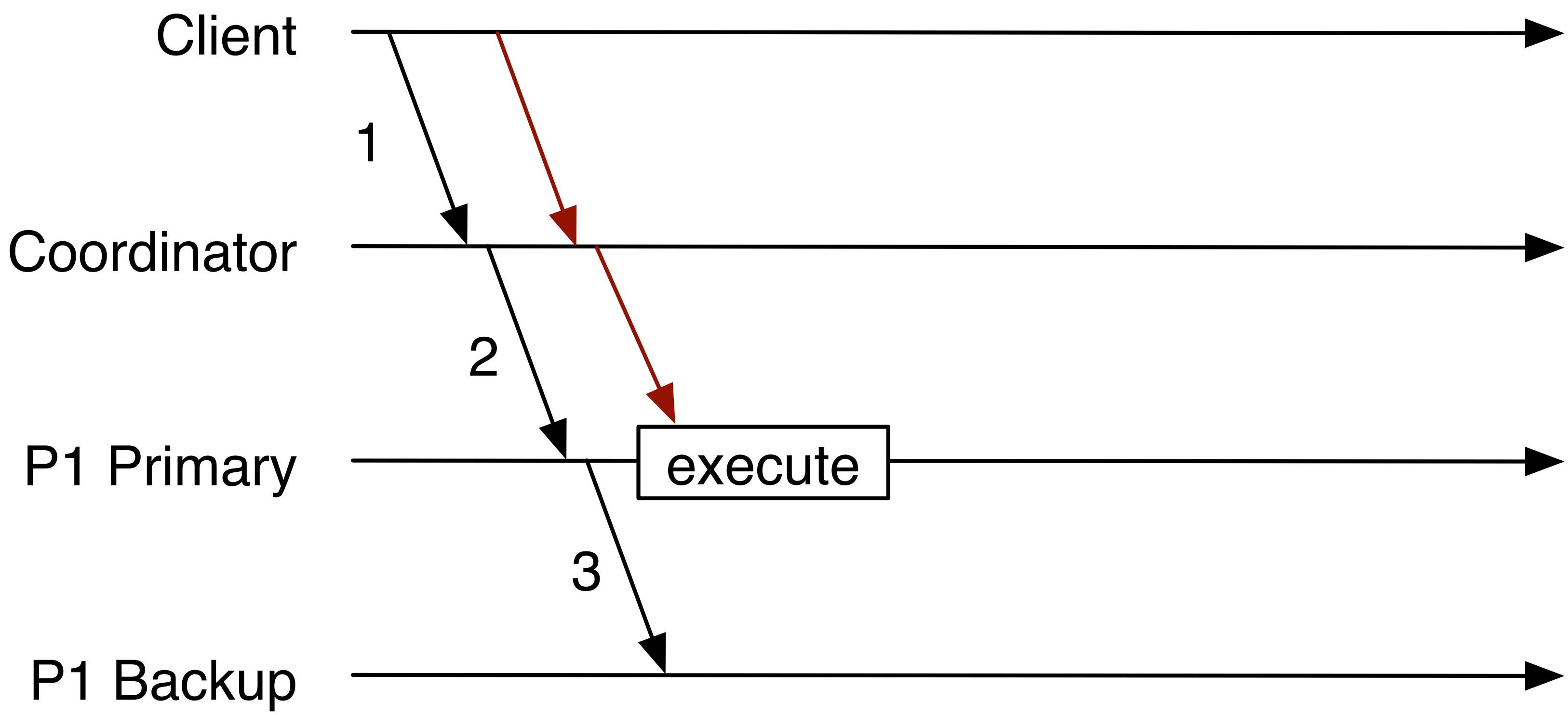
Blocking Multi-Partition



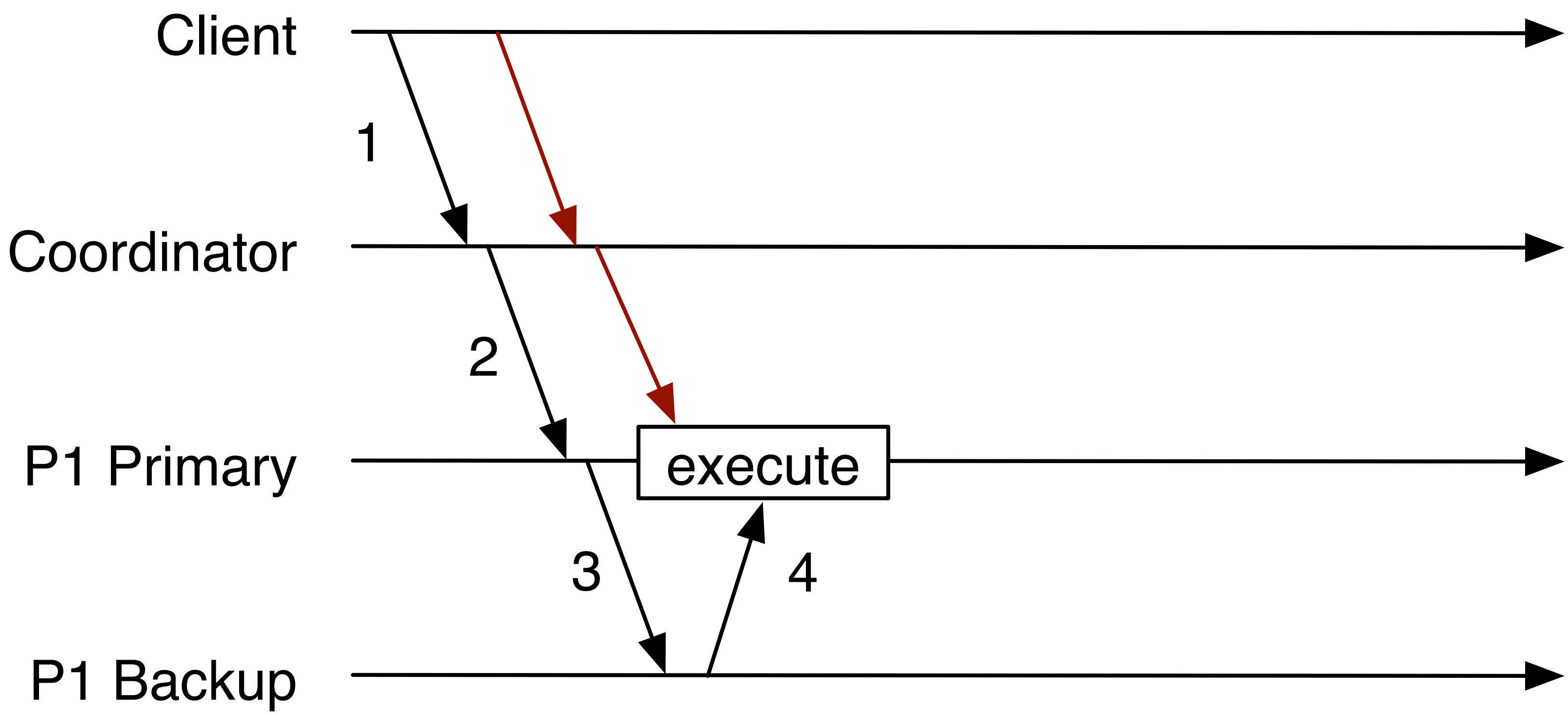
Blocking Multi-Partition



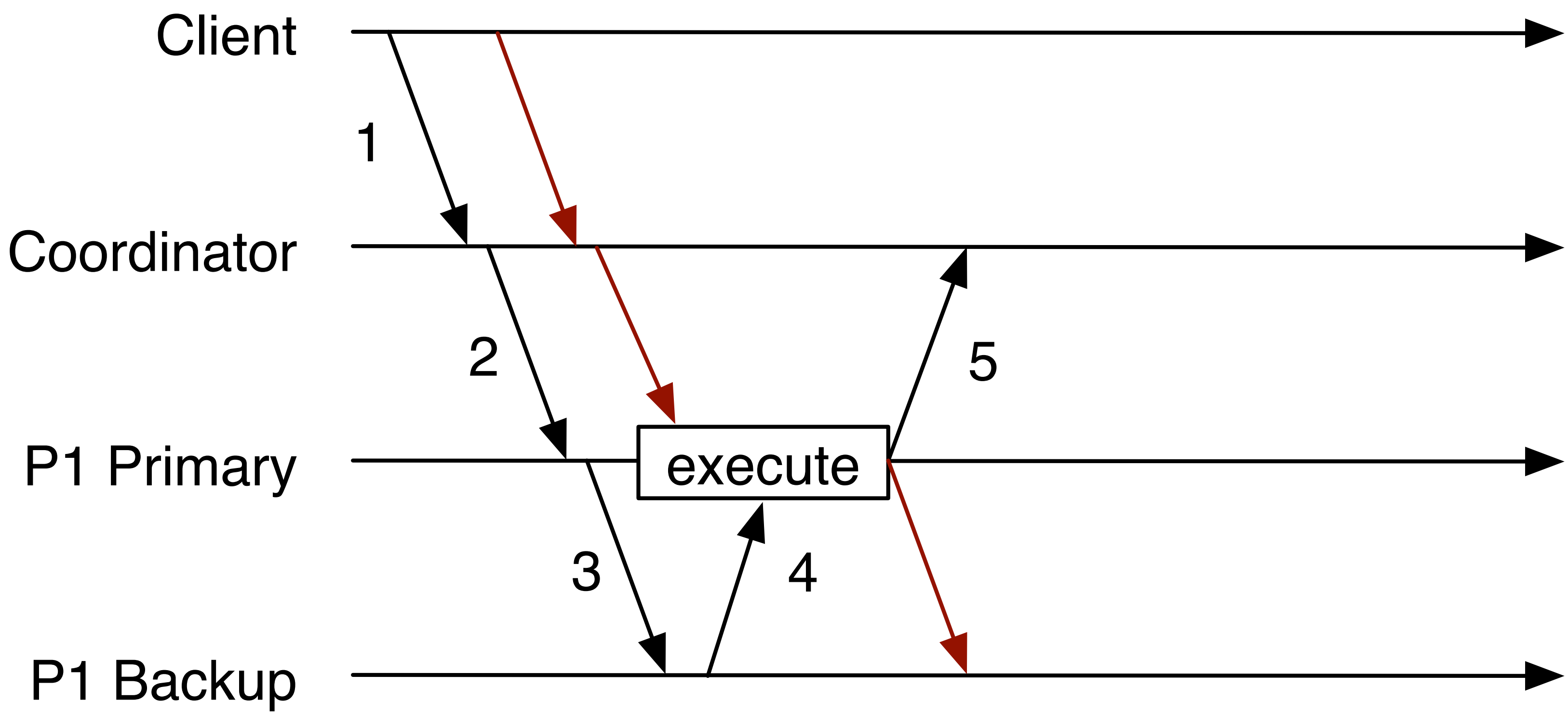
Blocking Multi-Partition



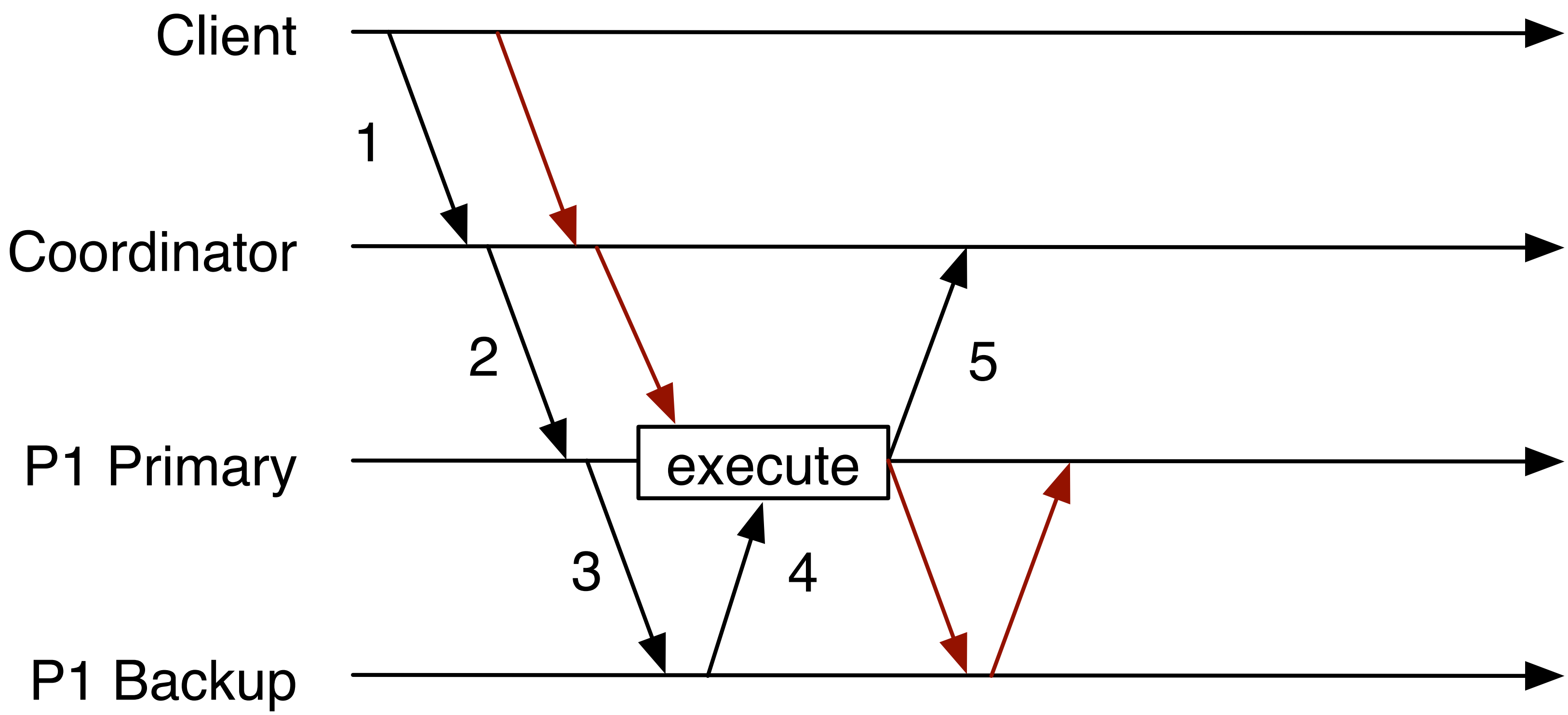
Blocking Multi-Partition



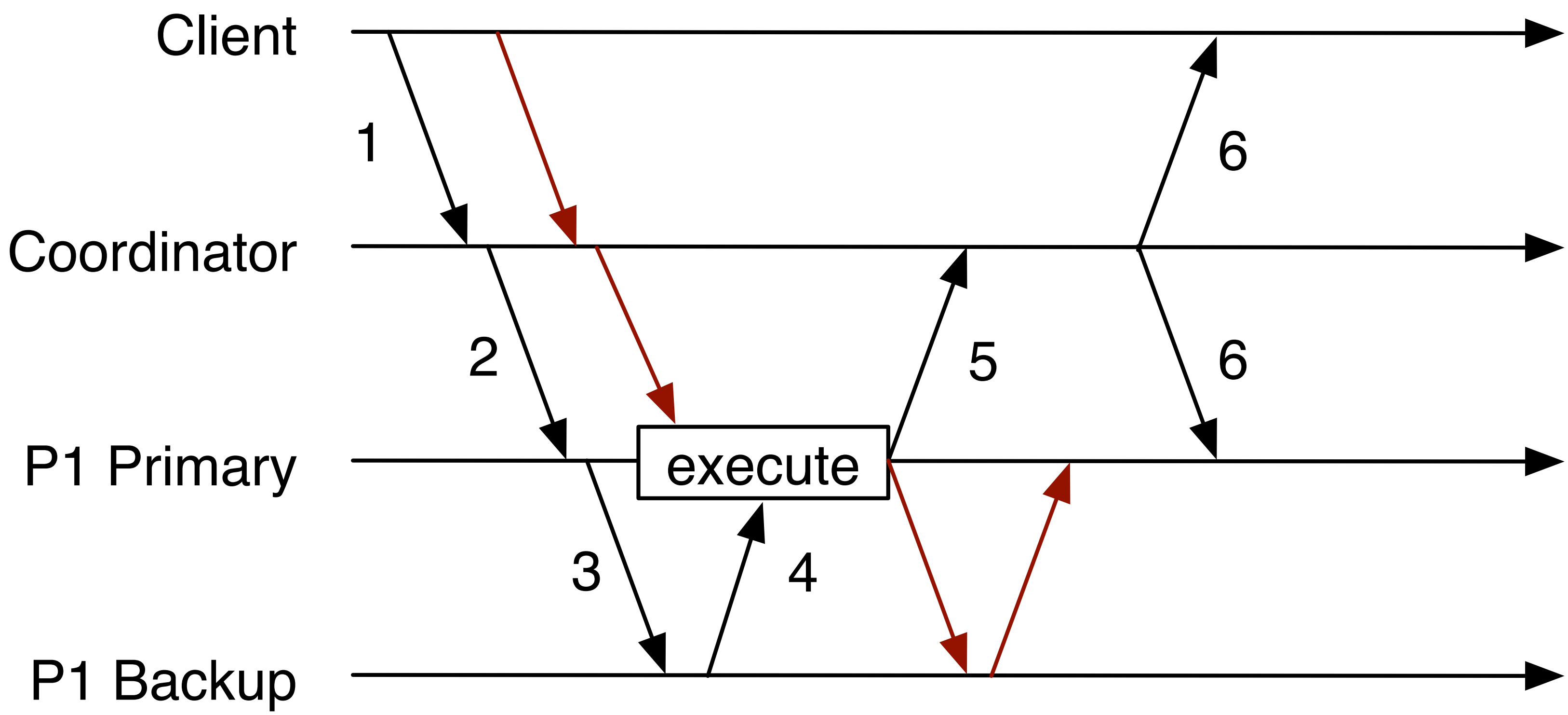
Blocking Multi-Partition



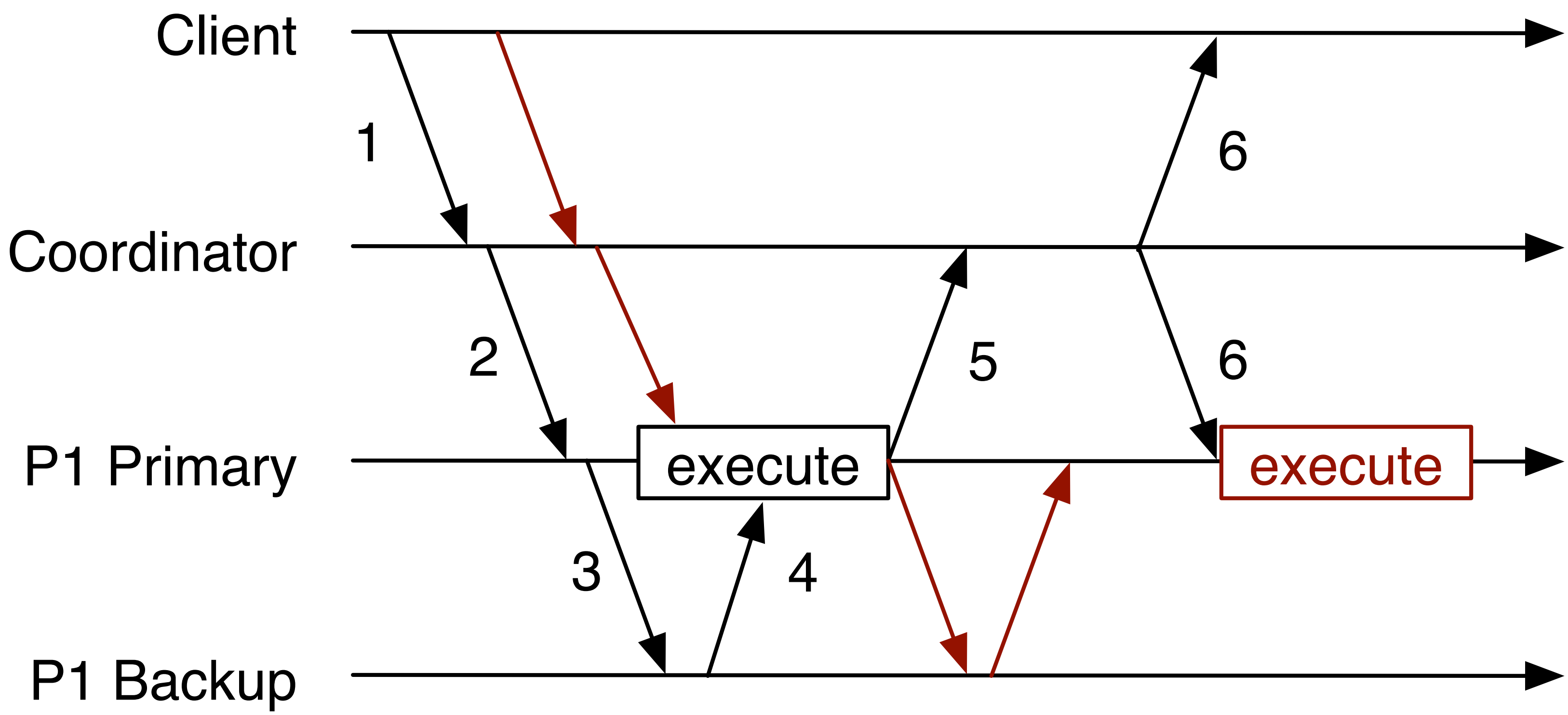
Blocking Multi-Partition



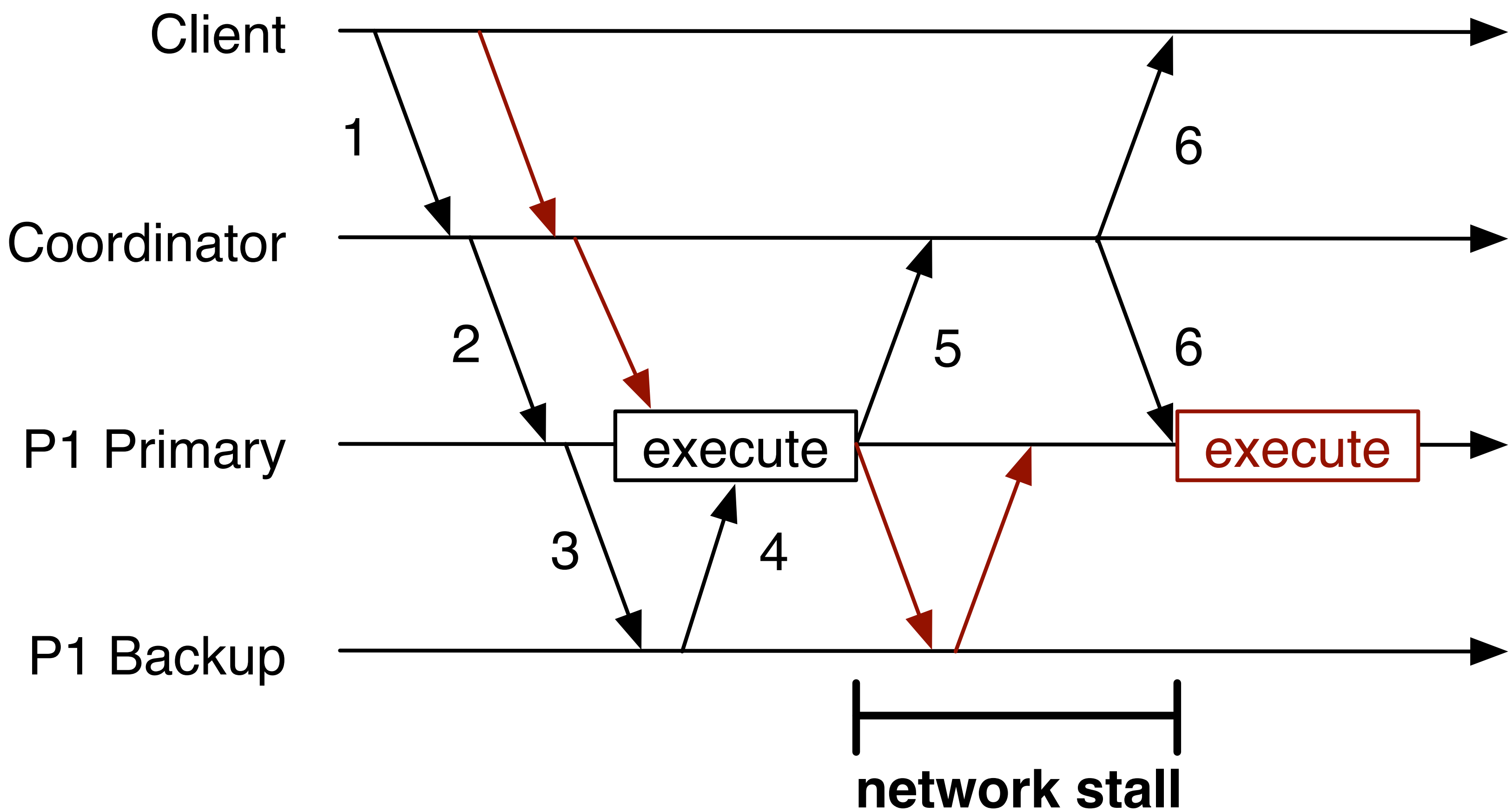
Blocking Multi-Partition



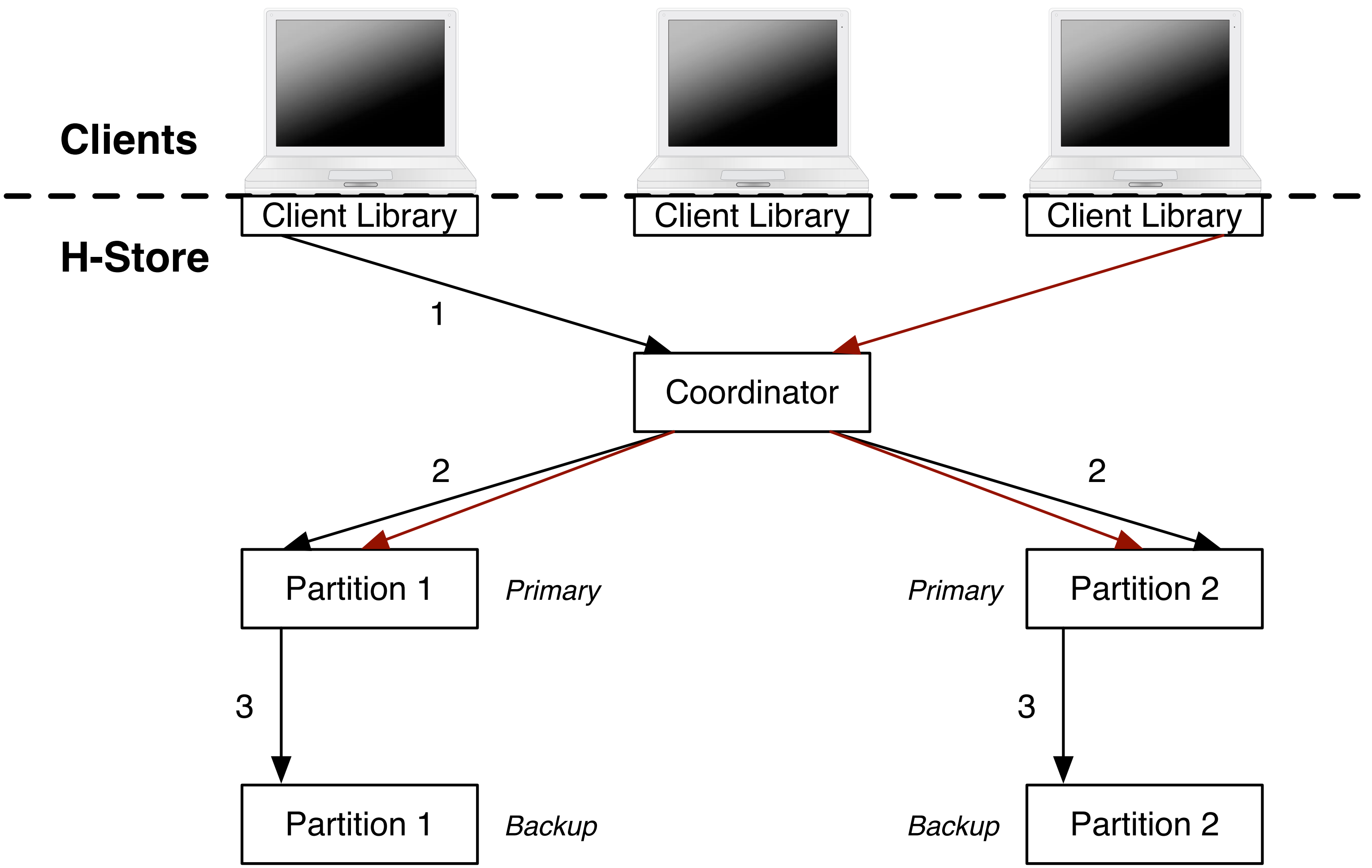
Blocking Multi-Partition



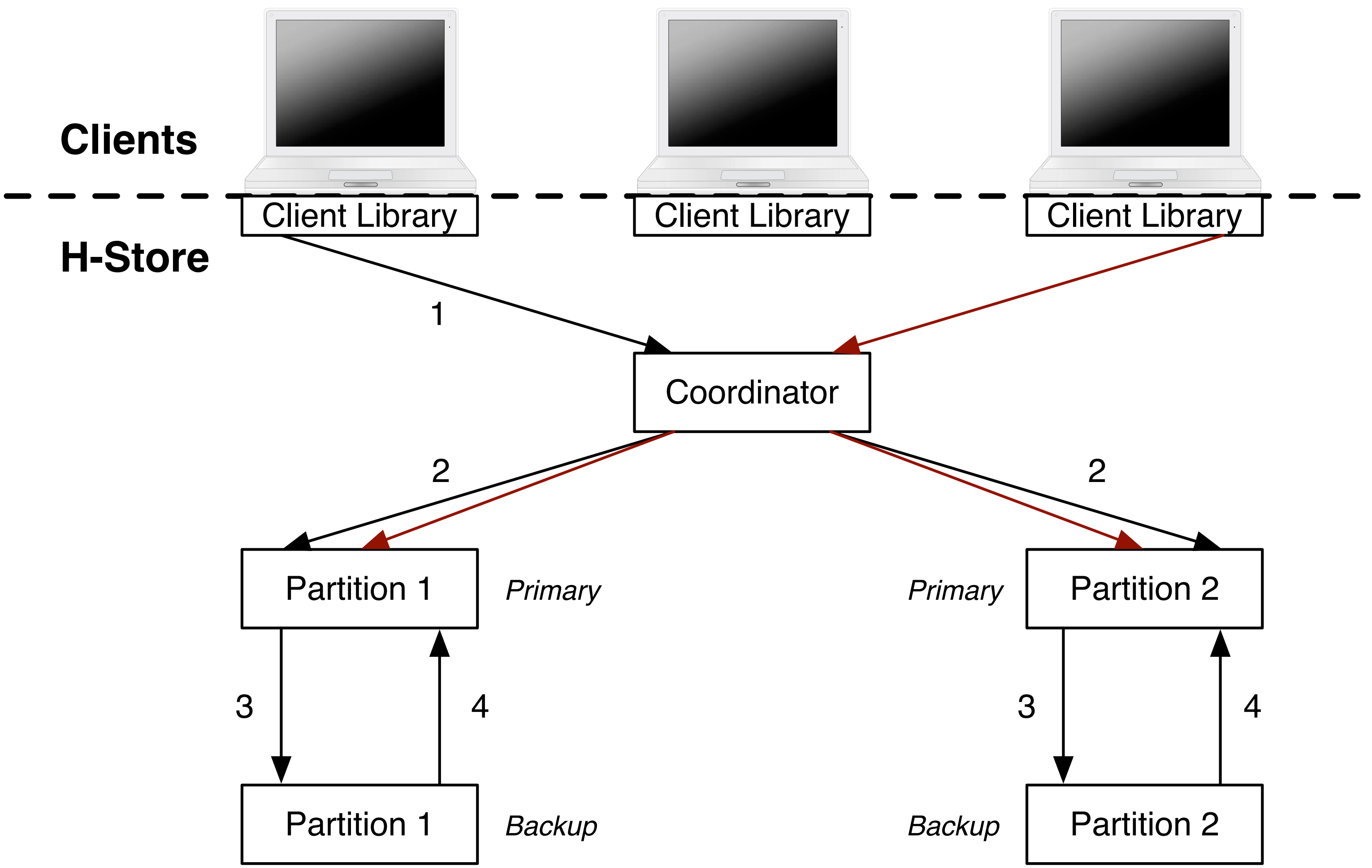
Blocking Multi-Partition



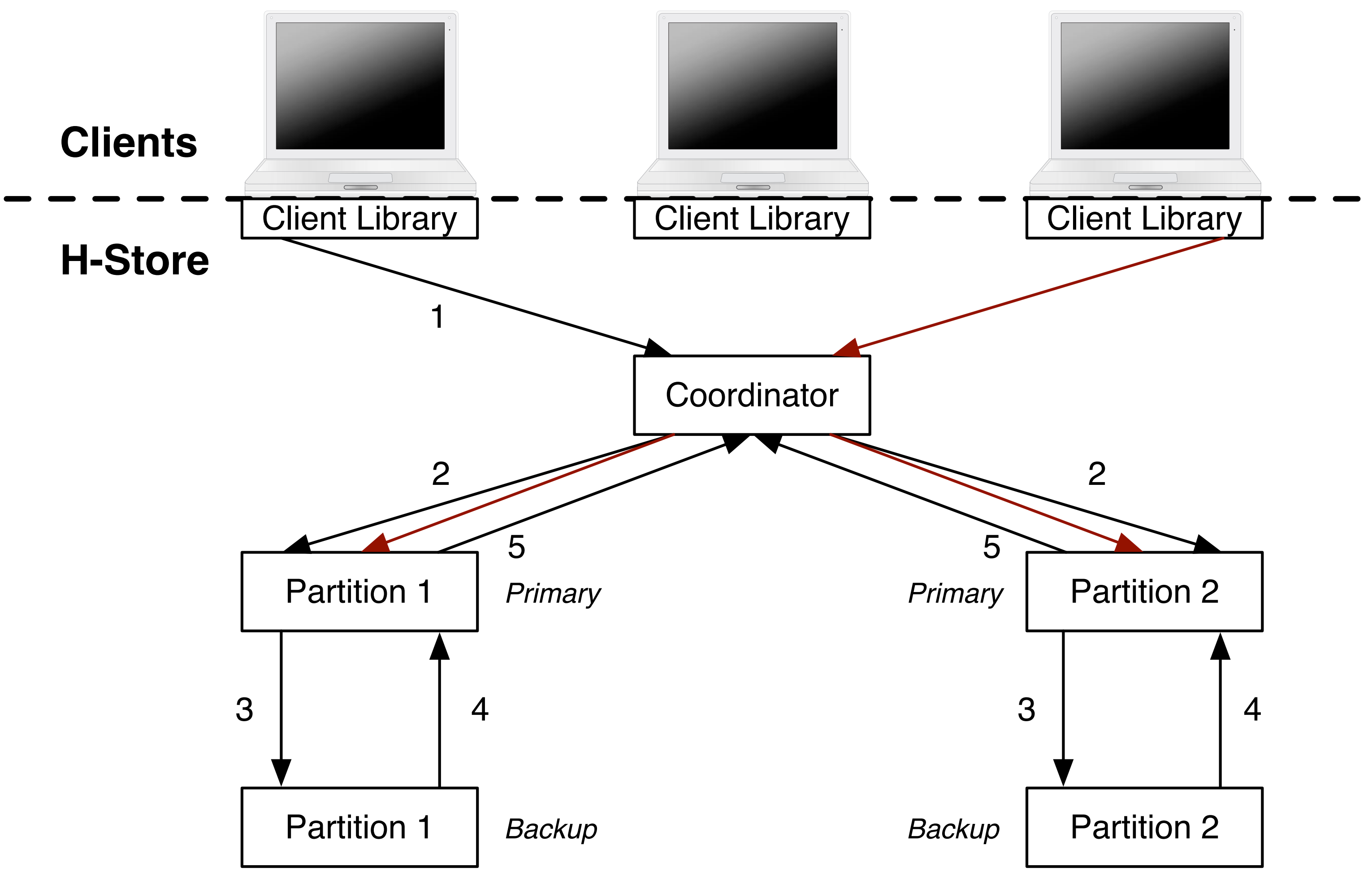
Blocking Multi-Partition



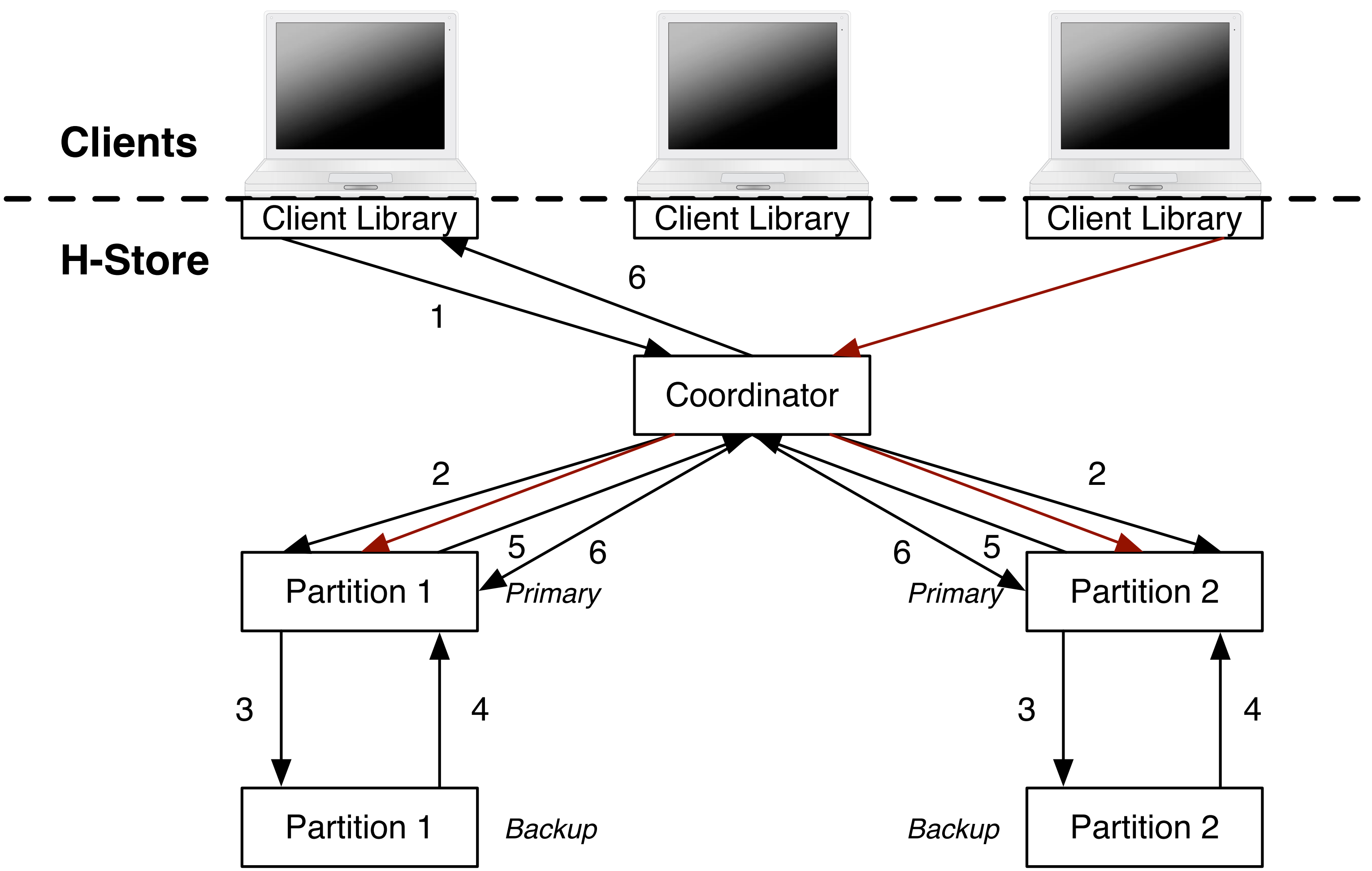
Blocking Multi-Partition



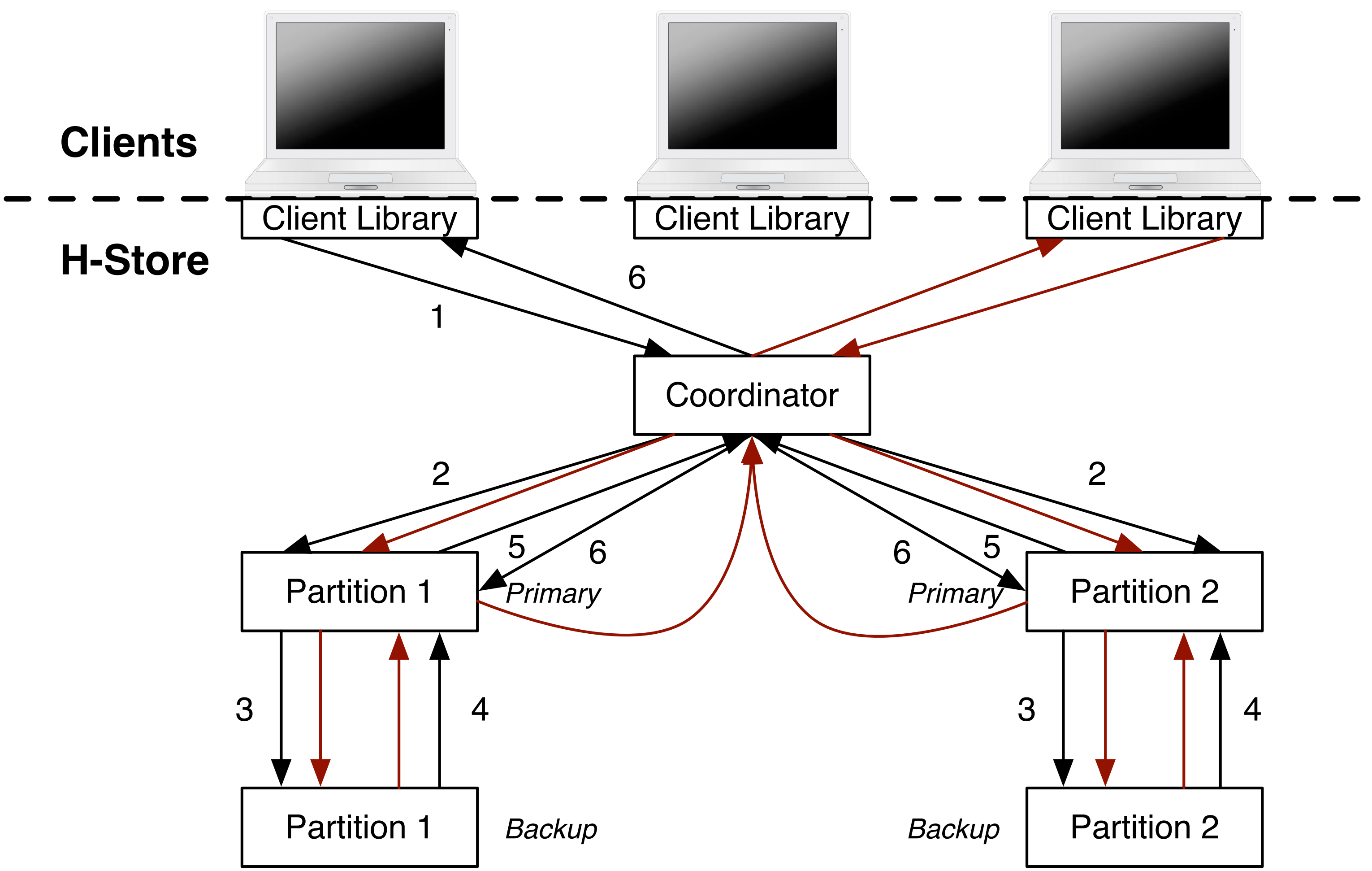
Blocking Multi-Partition



Blocking Multi-Partition



Blocking Multi-Partition



Two-Phase Locking

+ Execute non-conflicting txns during stall
+ No need to order in advance
– Locking overhead
– Deadlocks

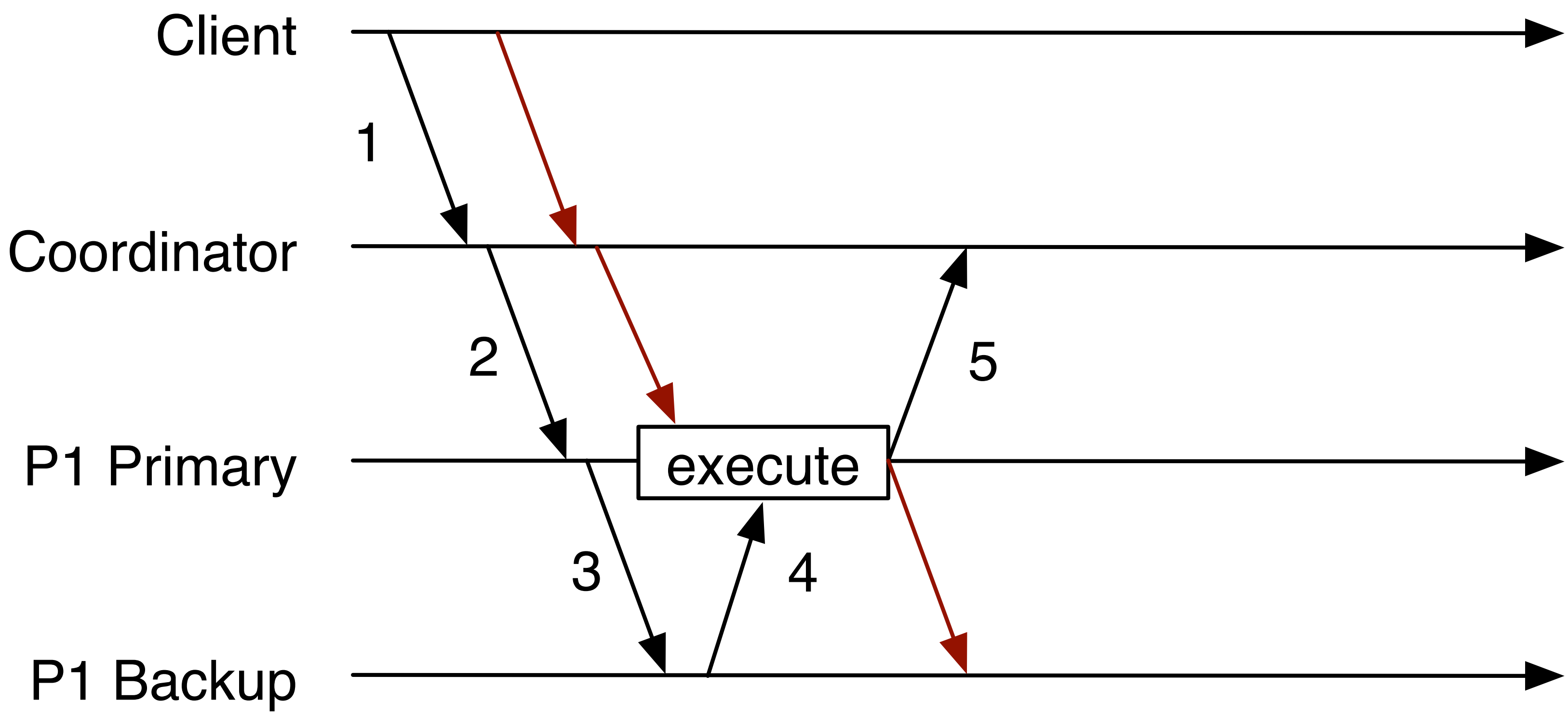
Optimization: turn off locks and undo logging when no multi-partition transactions

Speculative CC

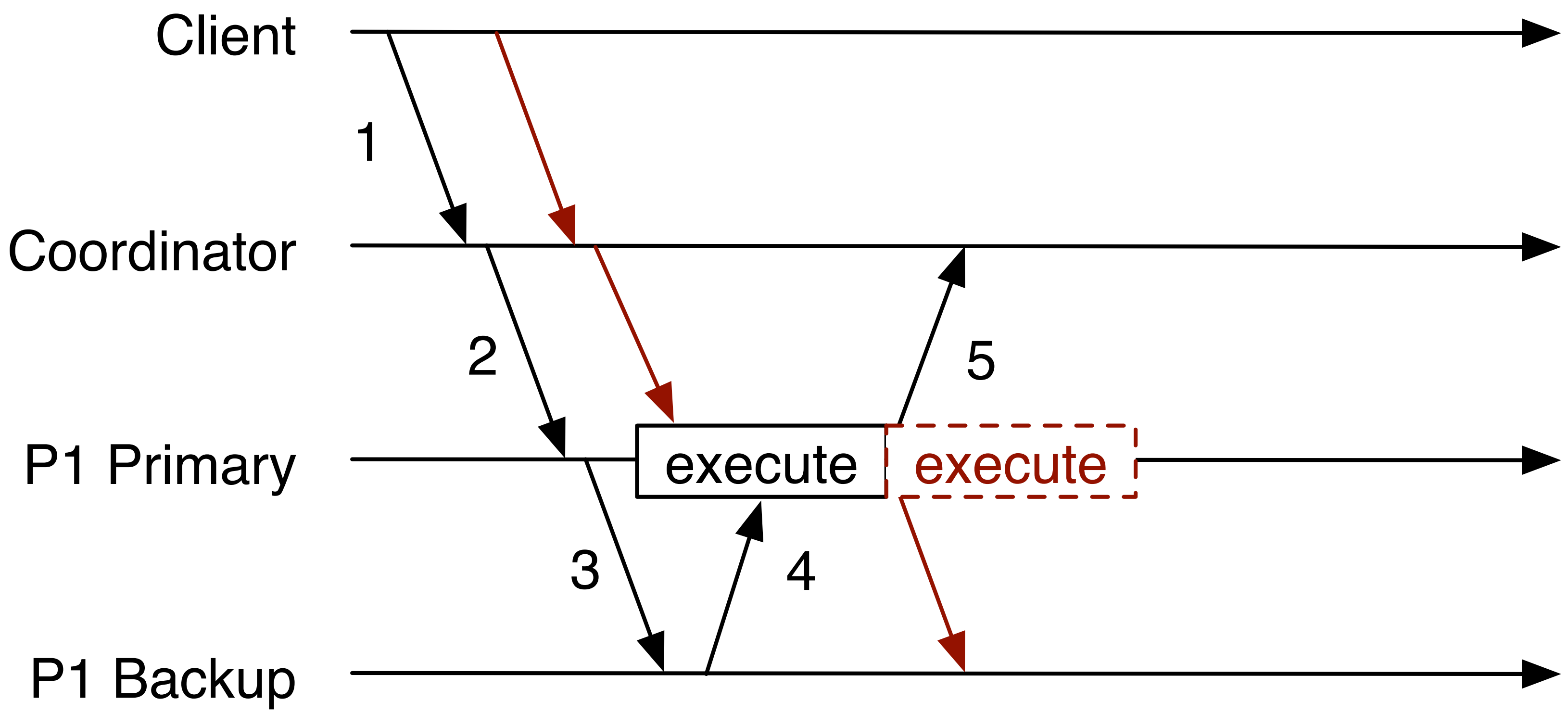
While waiting for commit/abort, speculatively execute other transactions

+ No locks; no read/write sets
– Need global transaction order
– Cascading aborts

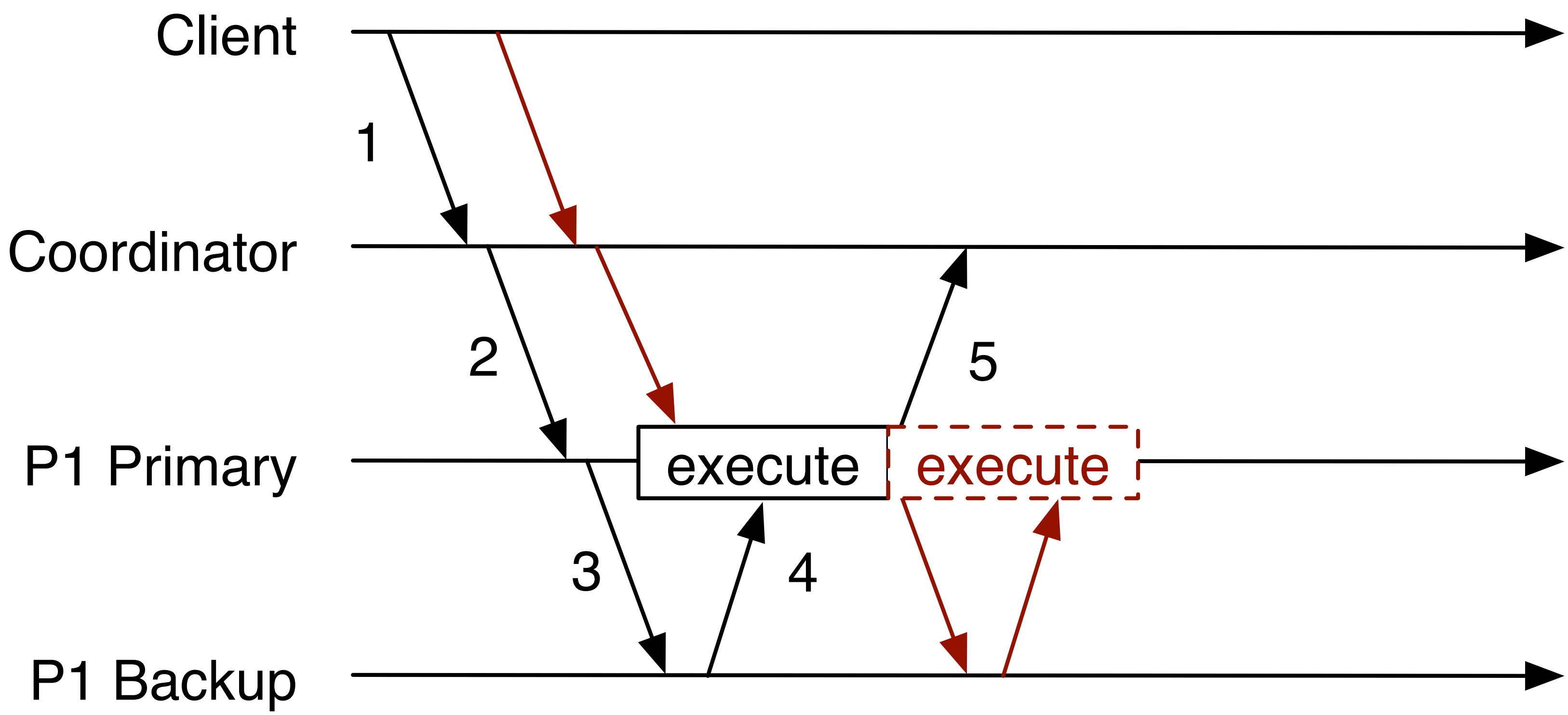
Speculative Multi-Partition



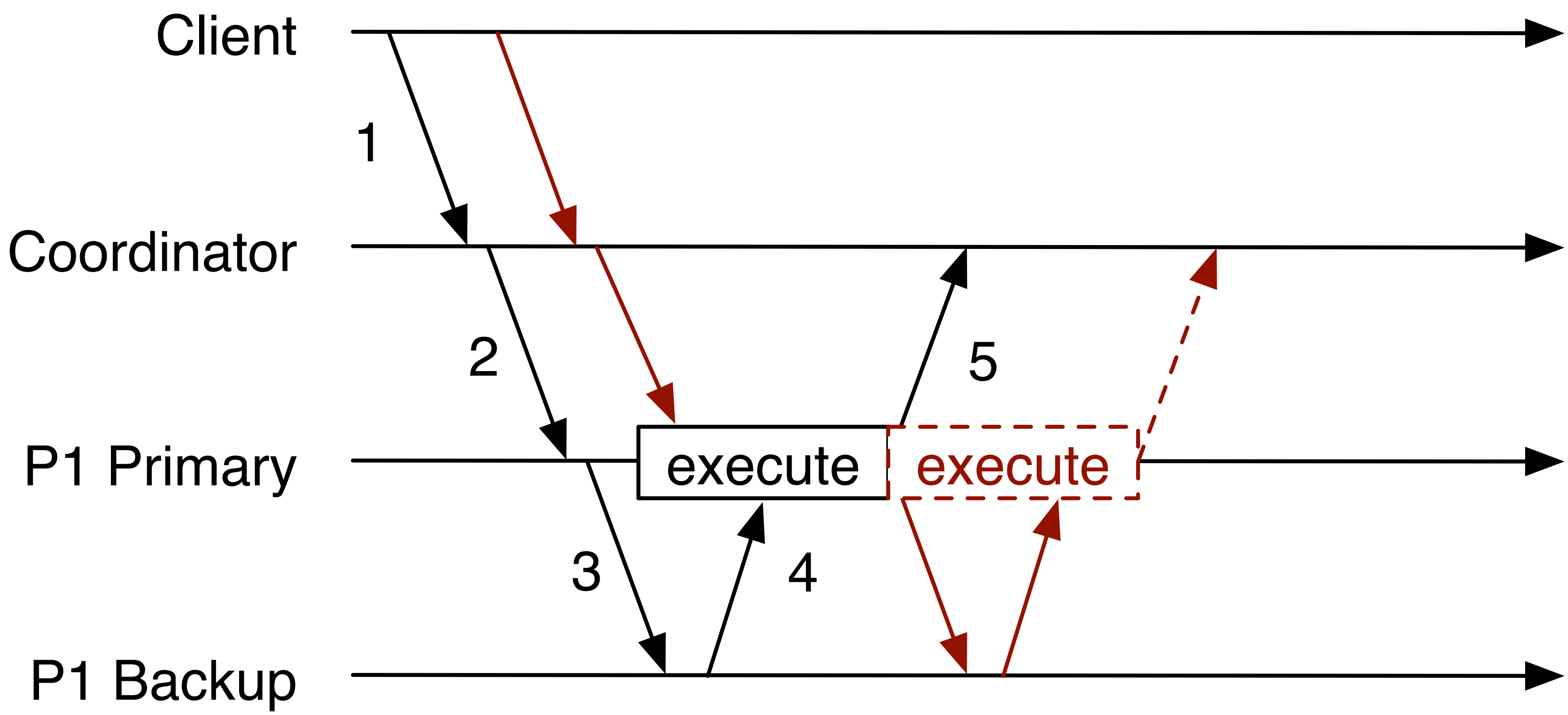
Speculative Multi-Partition



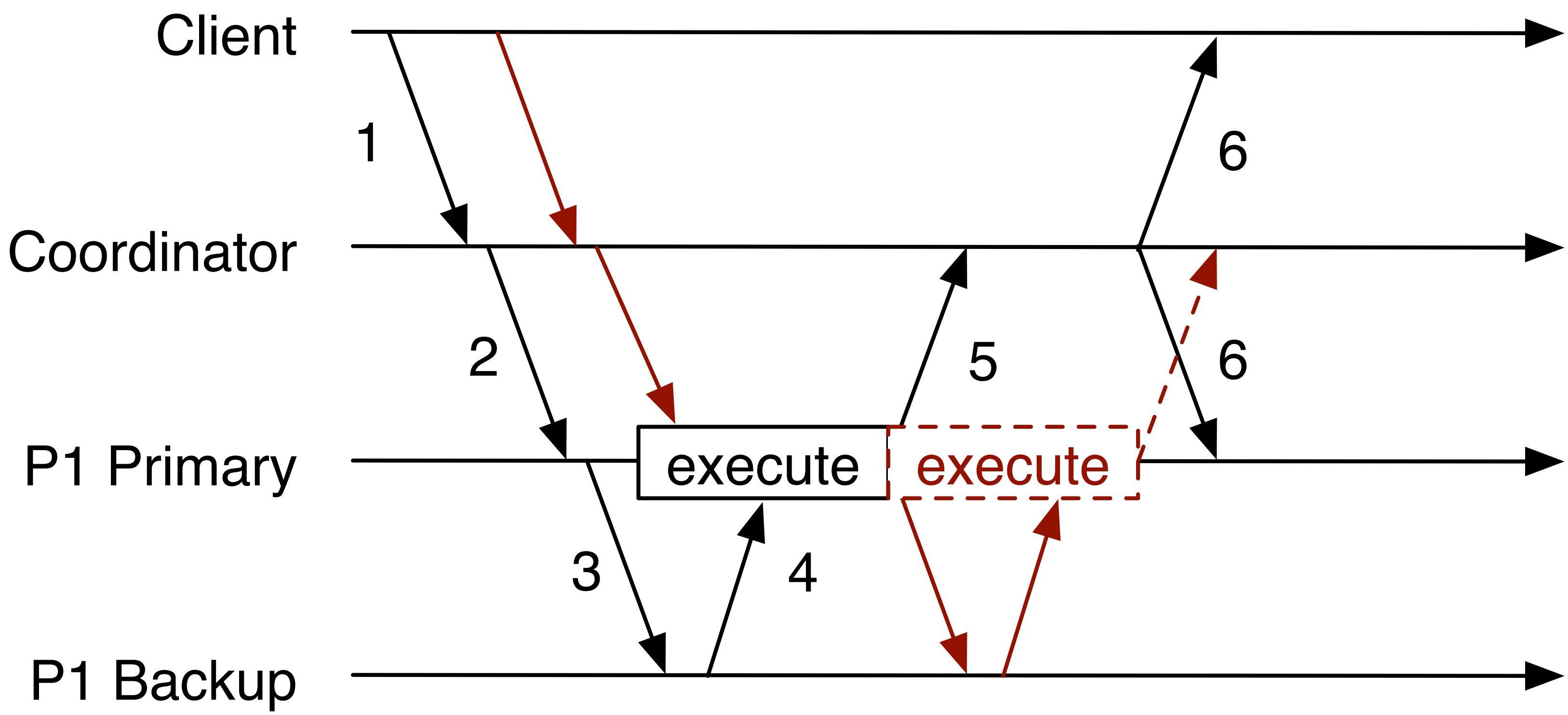
Speculative Multi-Partition



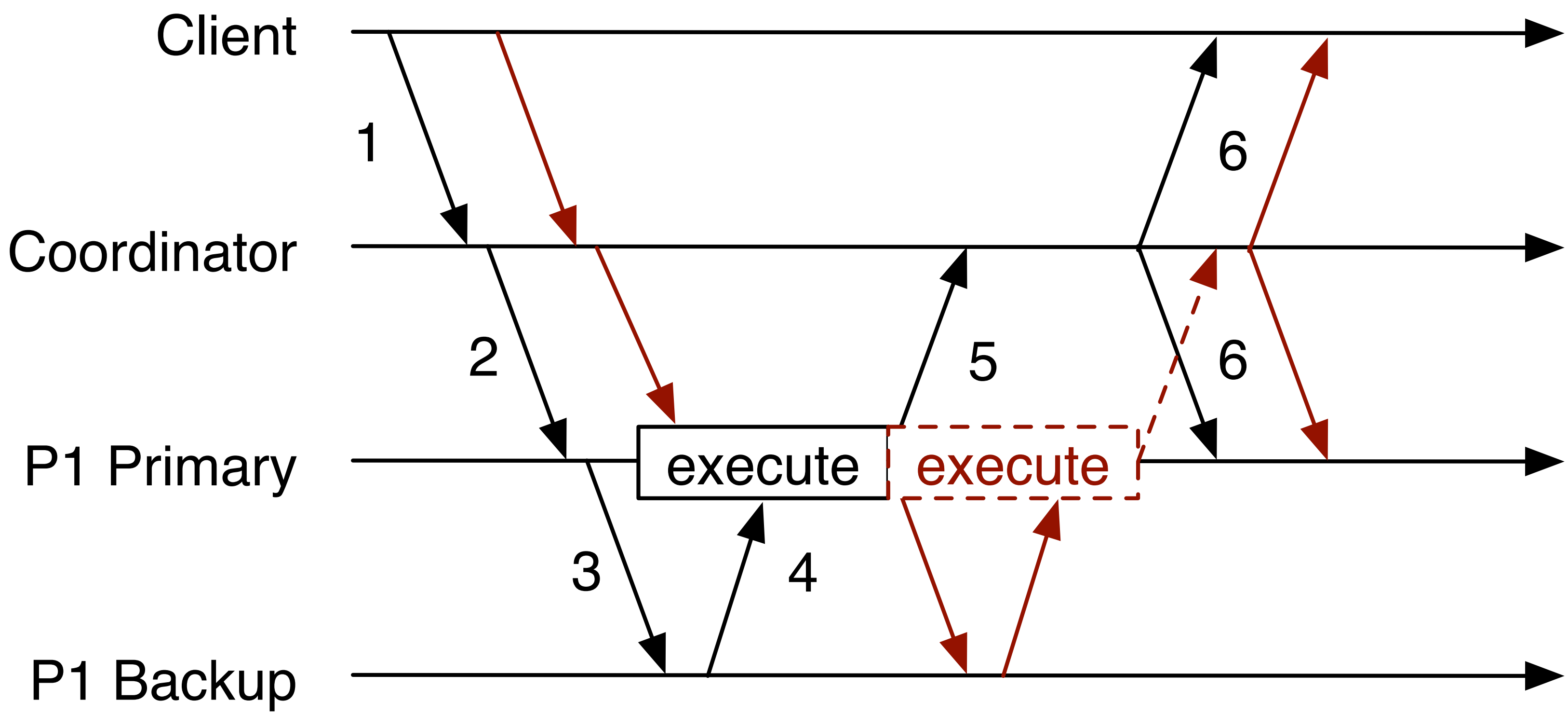
Speculative Multi-Partition



Speculative Multi-Partition



Speculative Multi-Partition



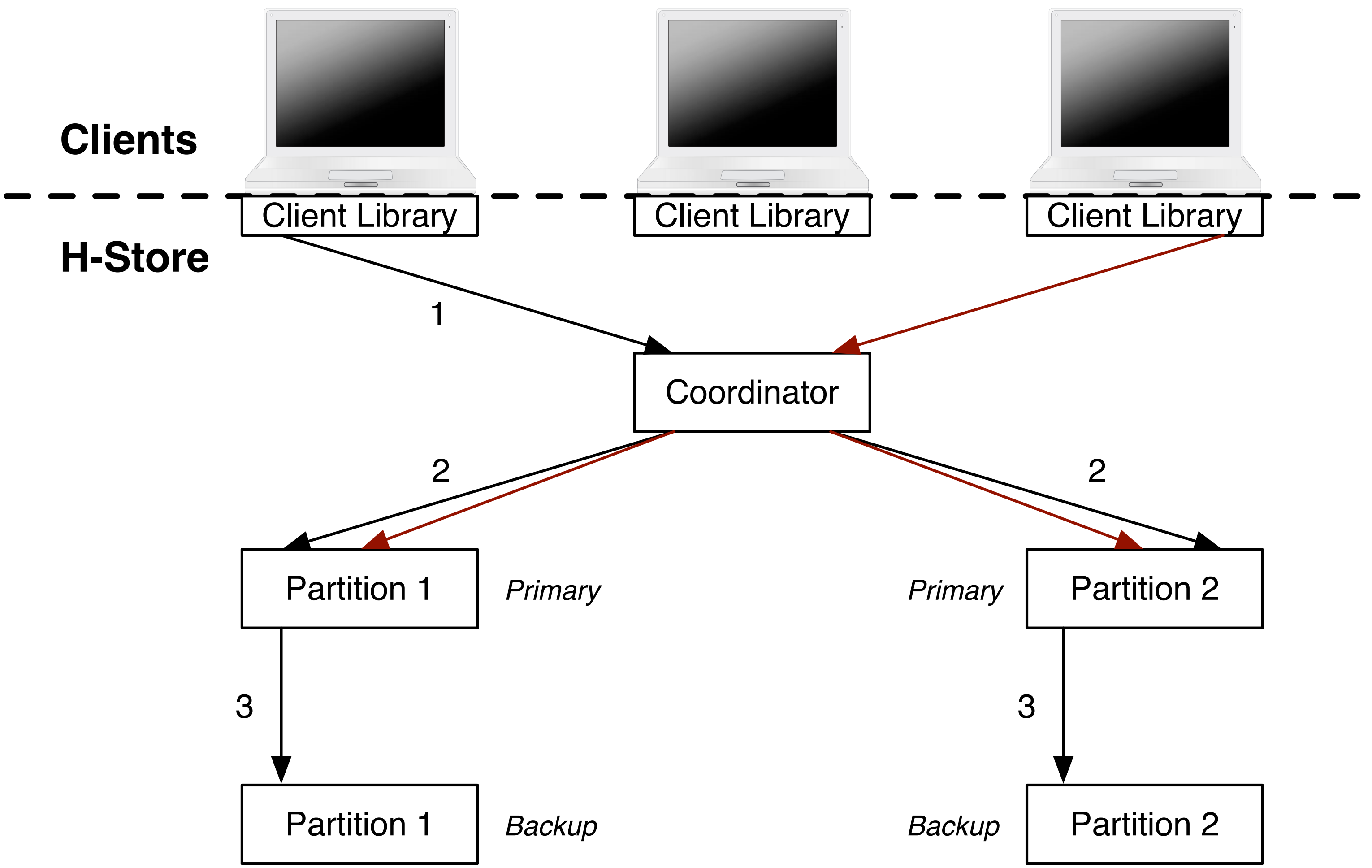
Speculation Limitation

Transactions with multiple “rounds” of work: need network stall

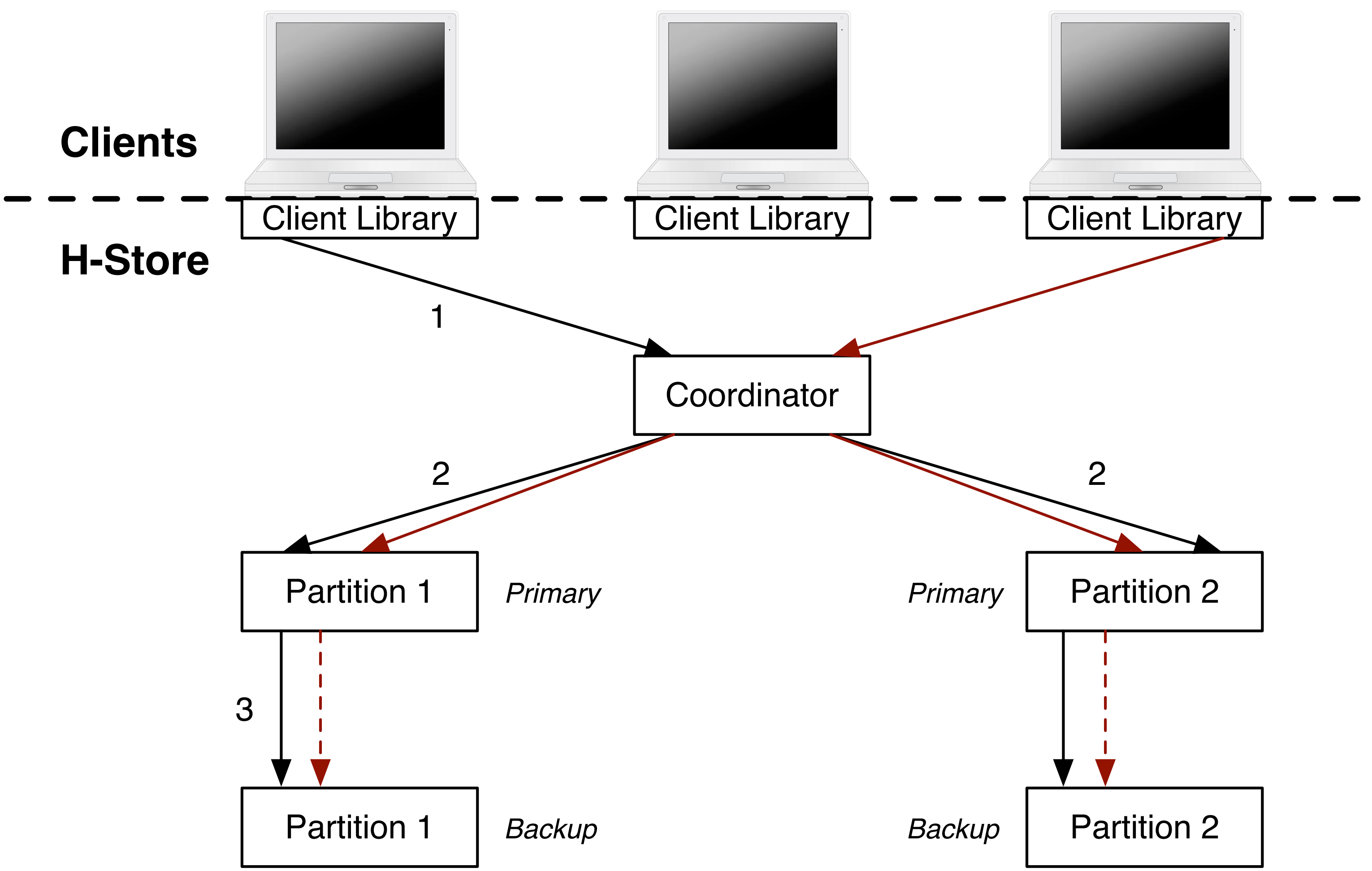
Example:

1. Read x on partition 1, y on partition 2
2. Update x = f(x, y); y = f(x, y)

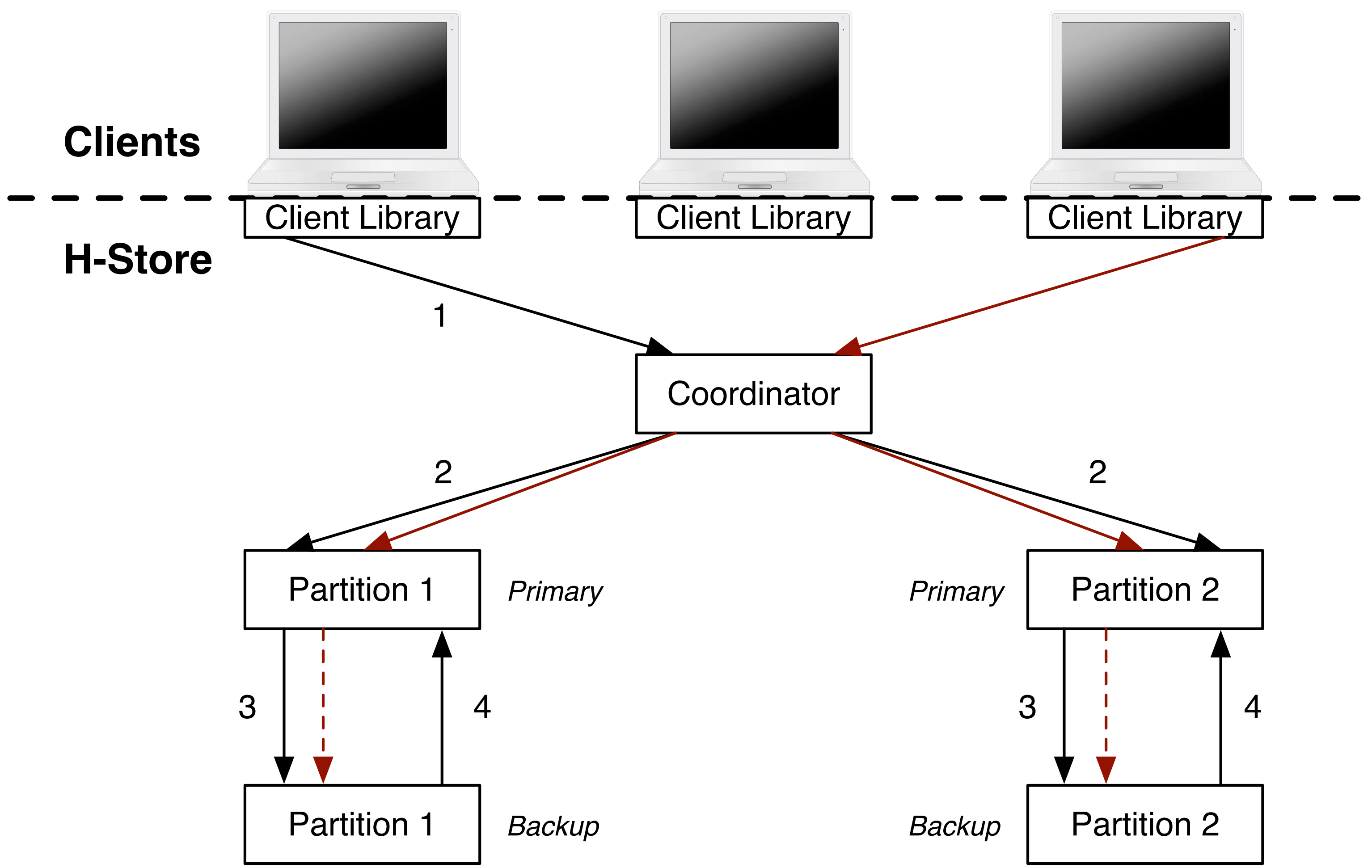
Speculative Multi-Partition



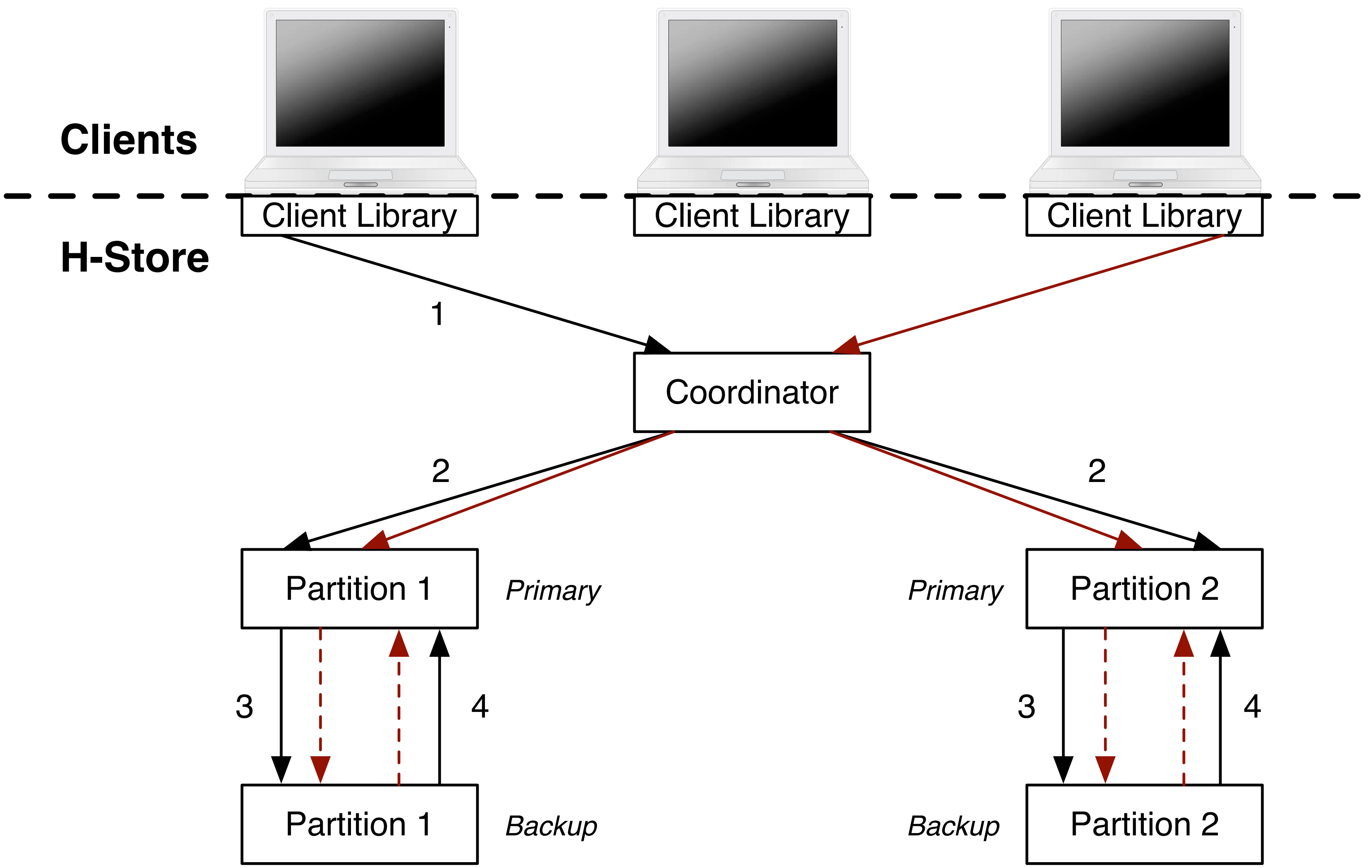
Speculative Multi-Partition



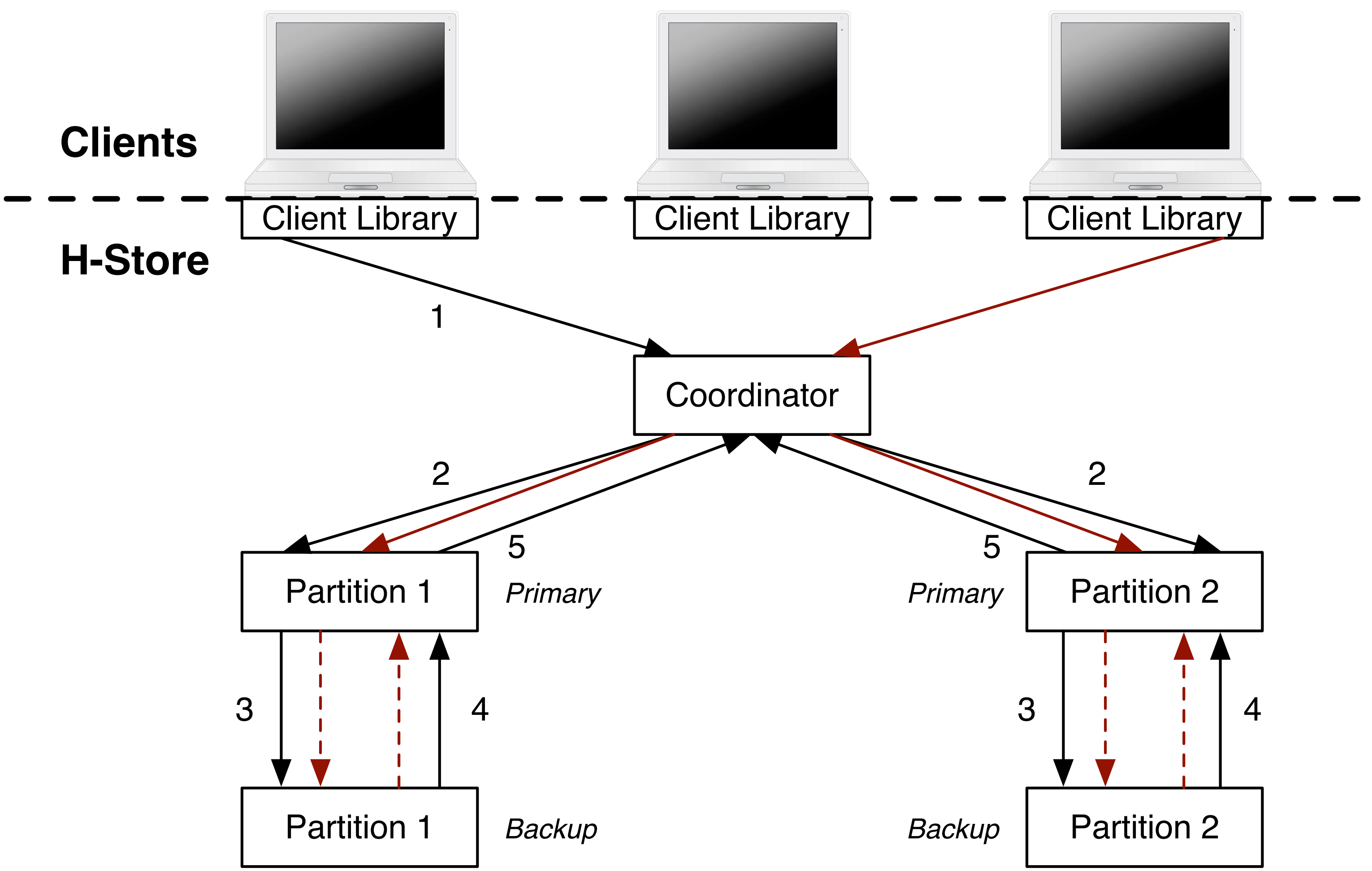
Speculative Multi-Partition



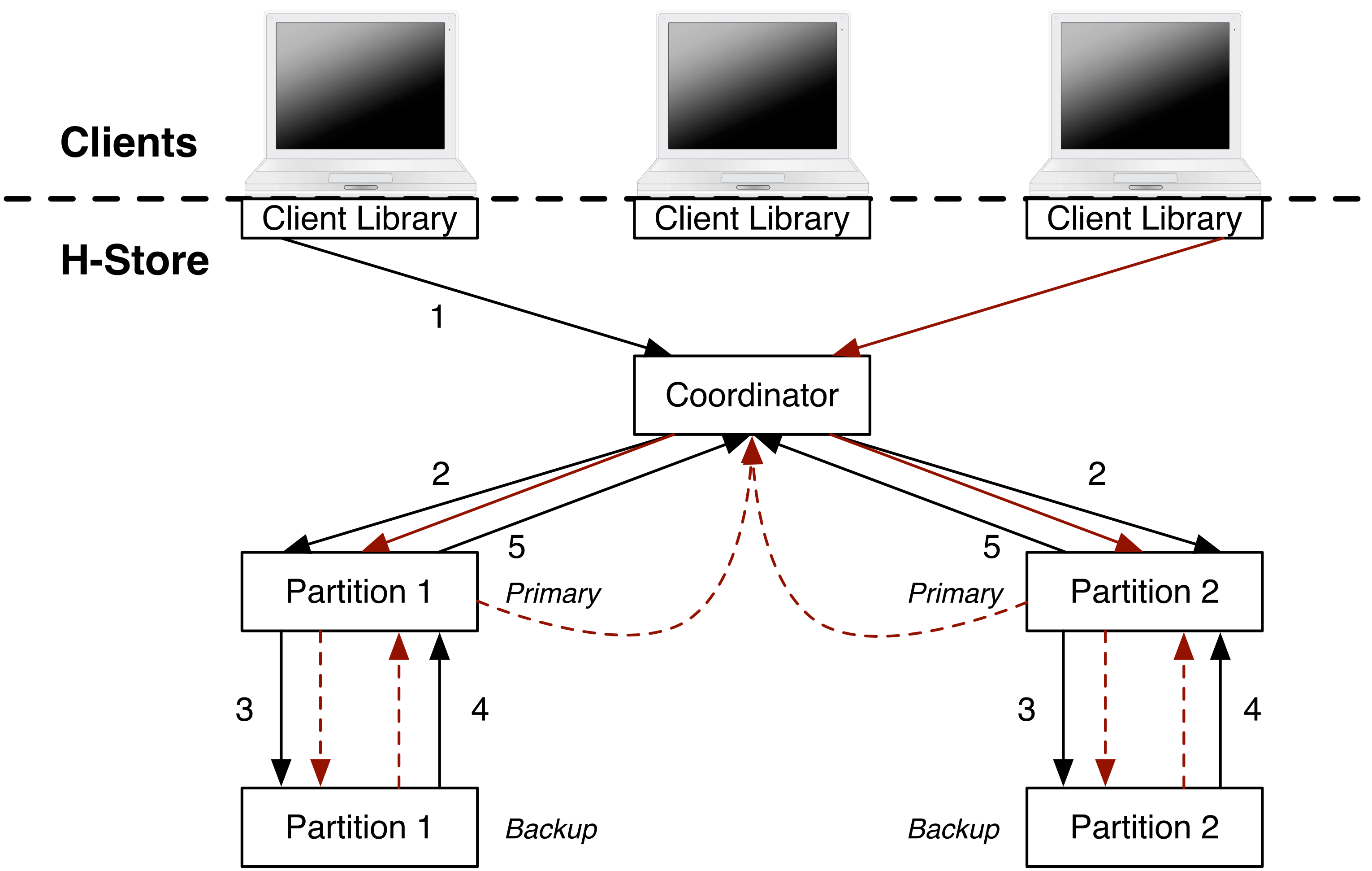
Speculative Multi-Partition



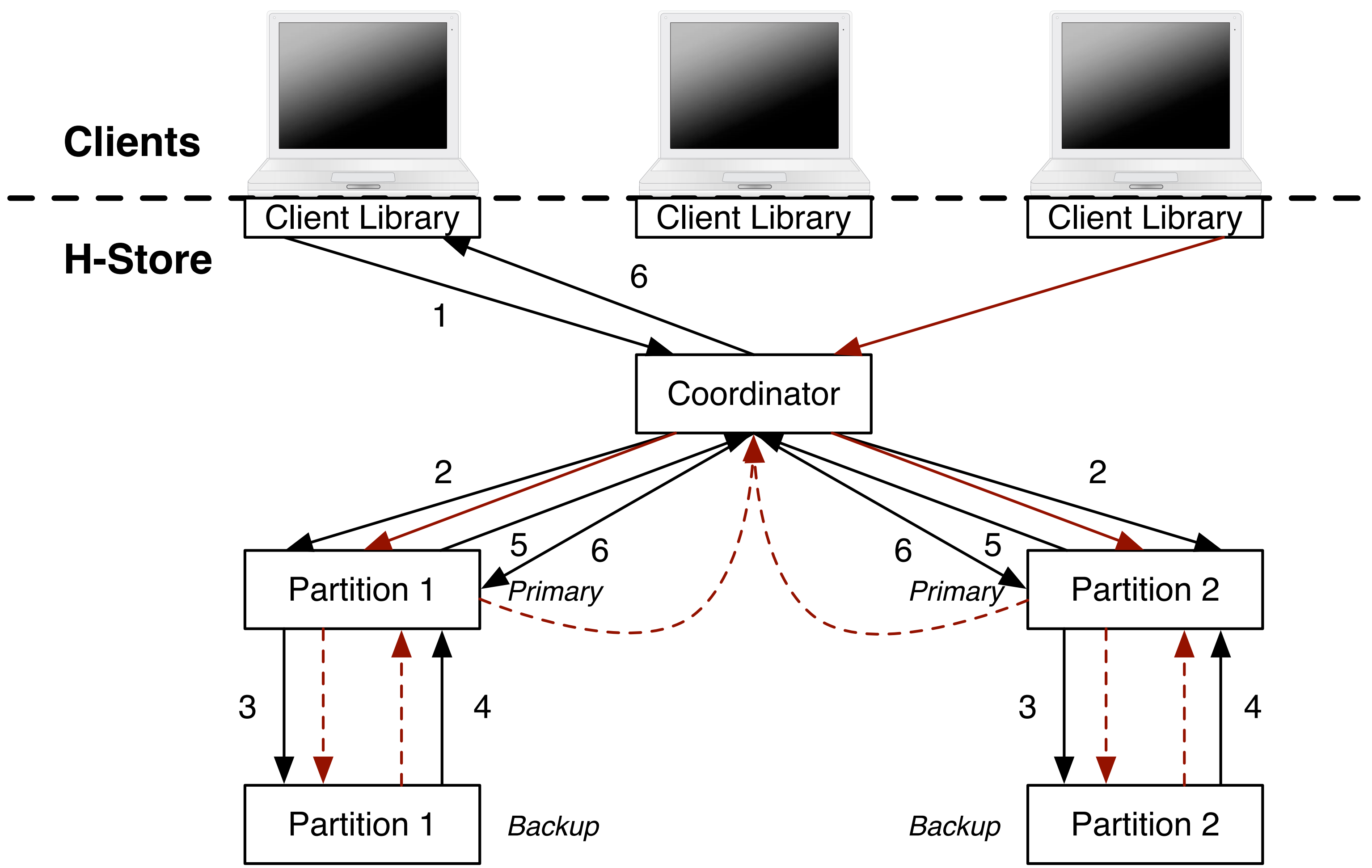
Speculative Multi-Partition



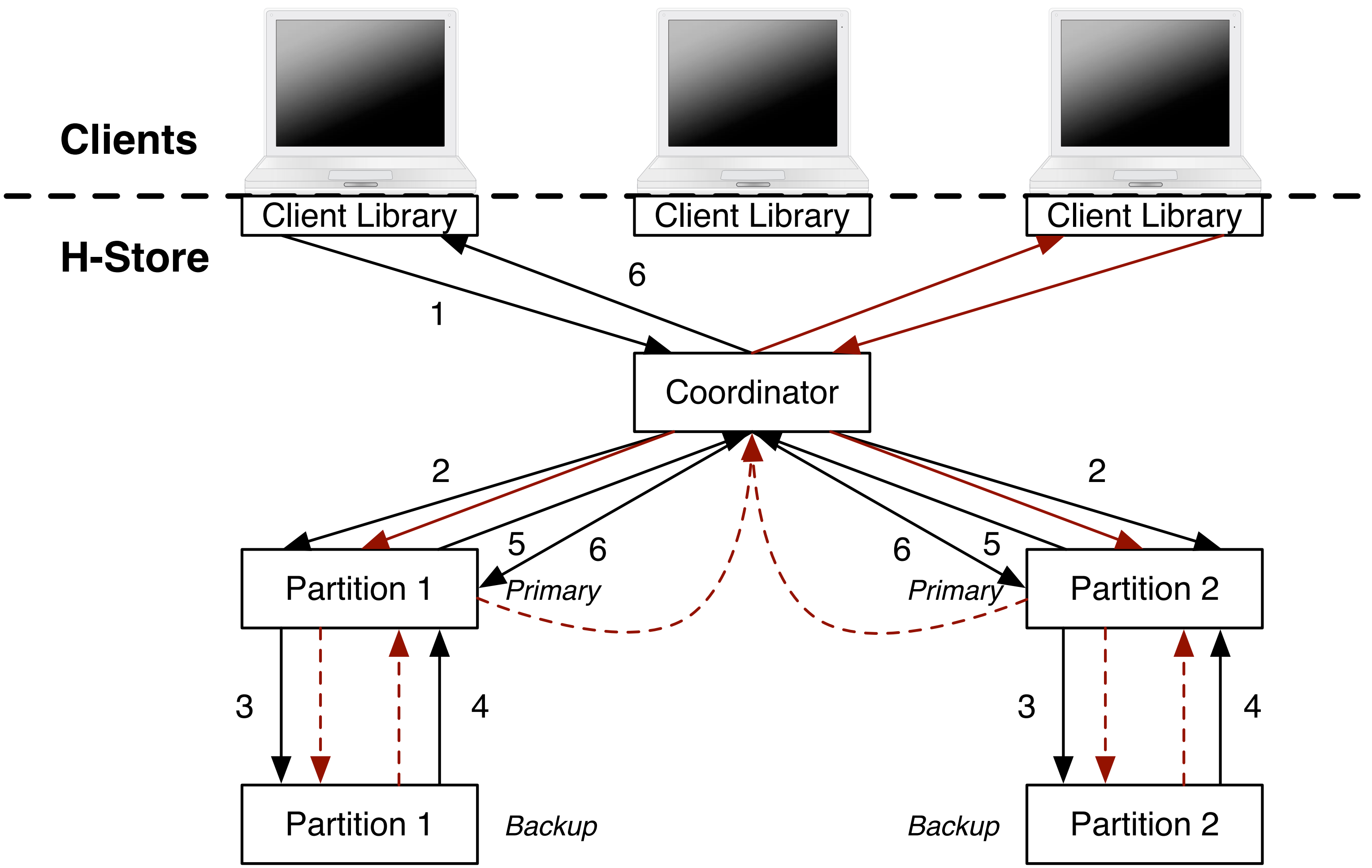
Speculative Multi-Partition



Speculative Multi-Partition



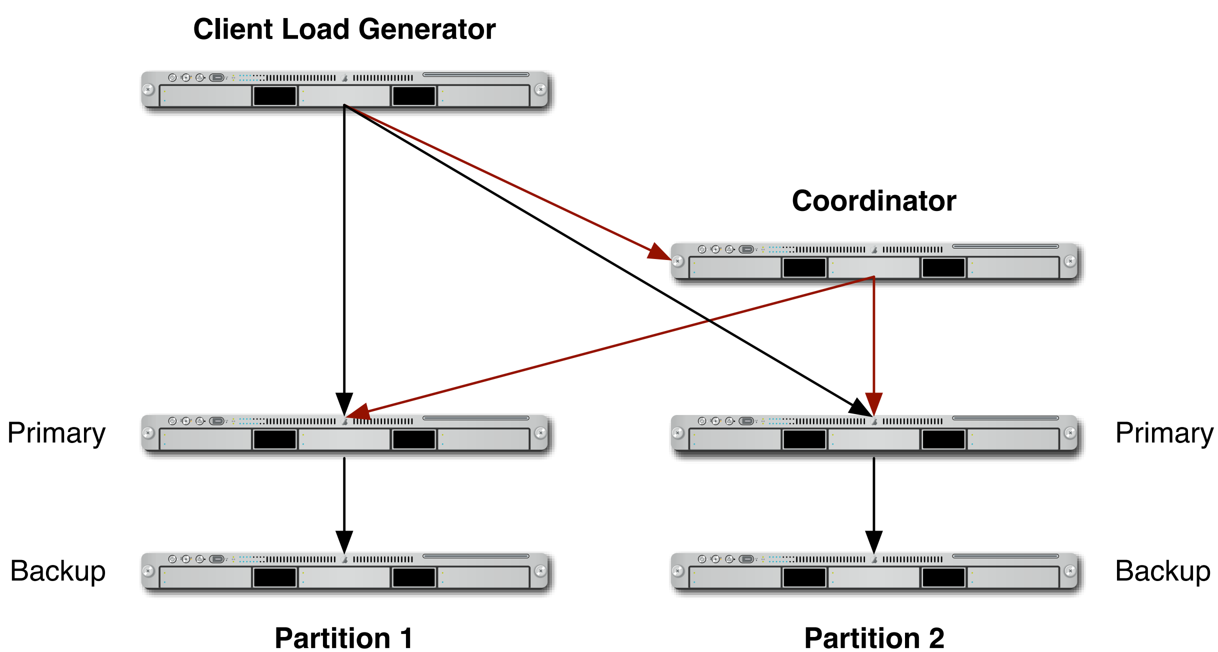
Speculative Multi-Partition



Microbenchmark

Two partitions of a single table

`(id INTEGER PRIMARY KEY, value INTEGER)`



Microbenchmark

Single partition transaction:

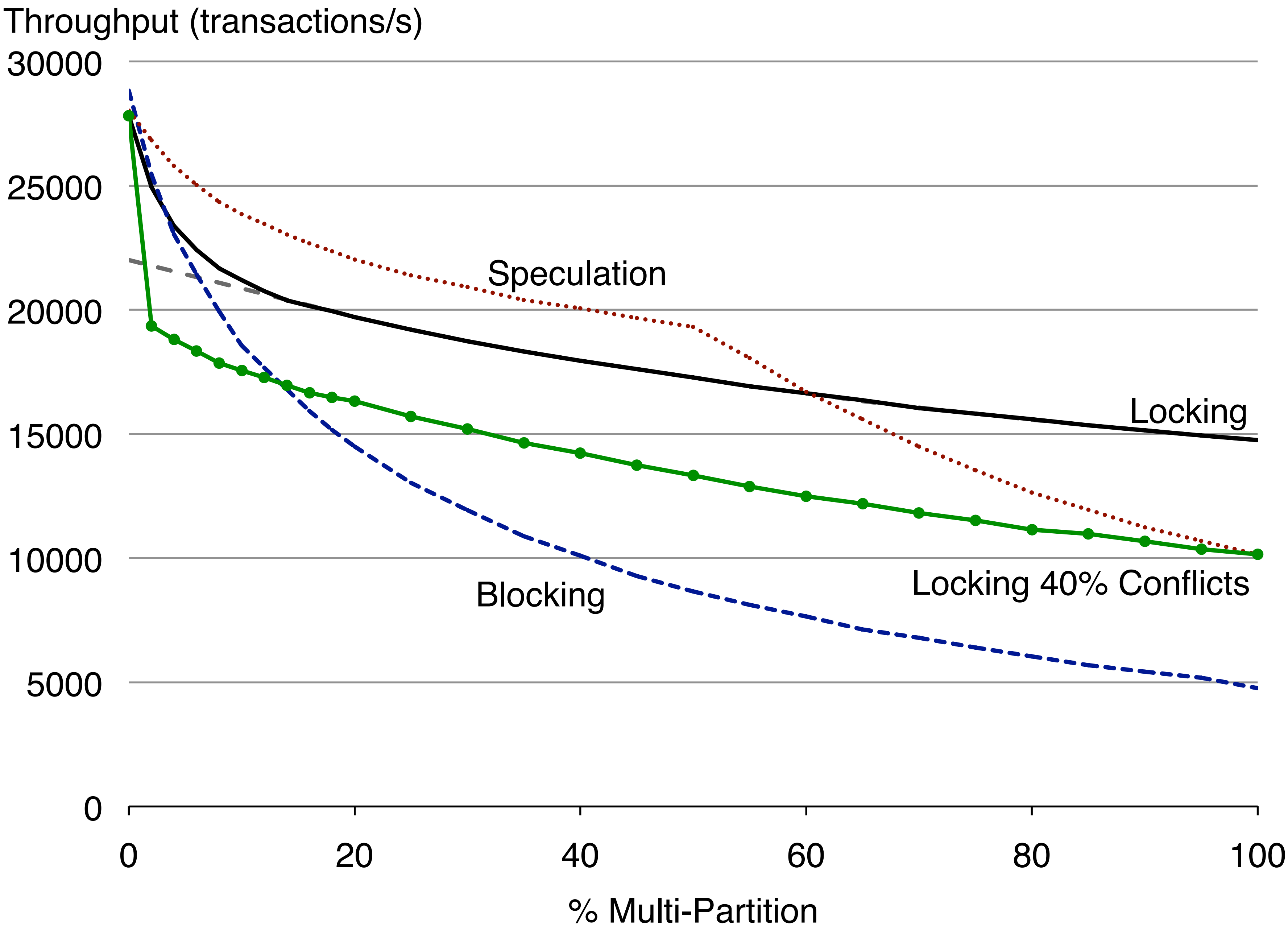
read/write keys on one partition

Multi-partition transaction:

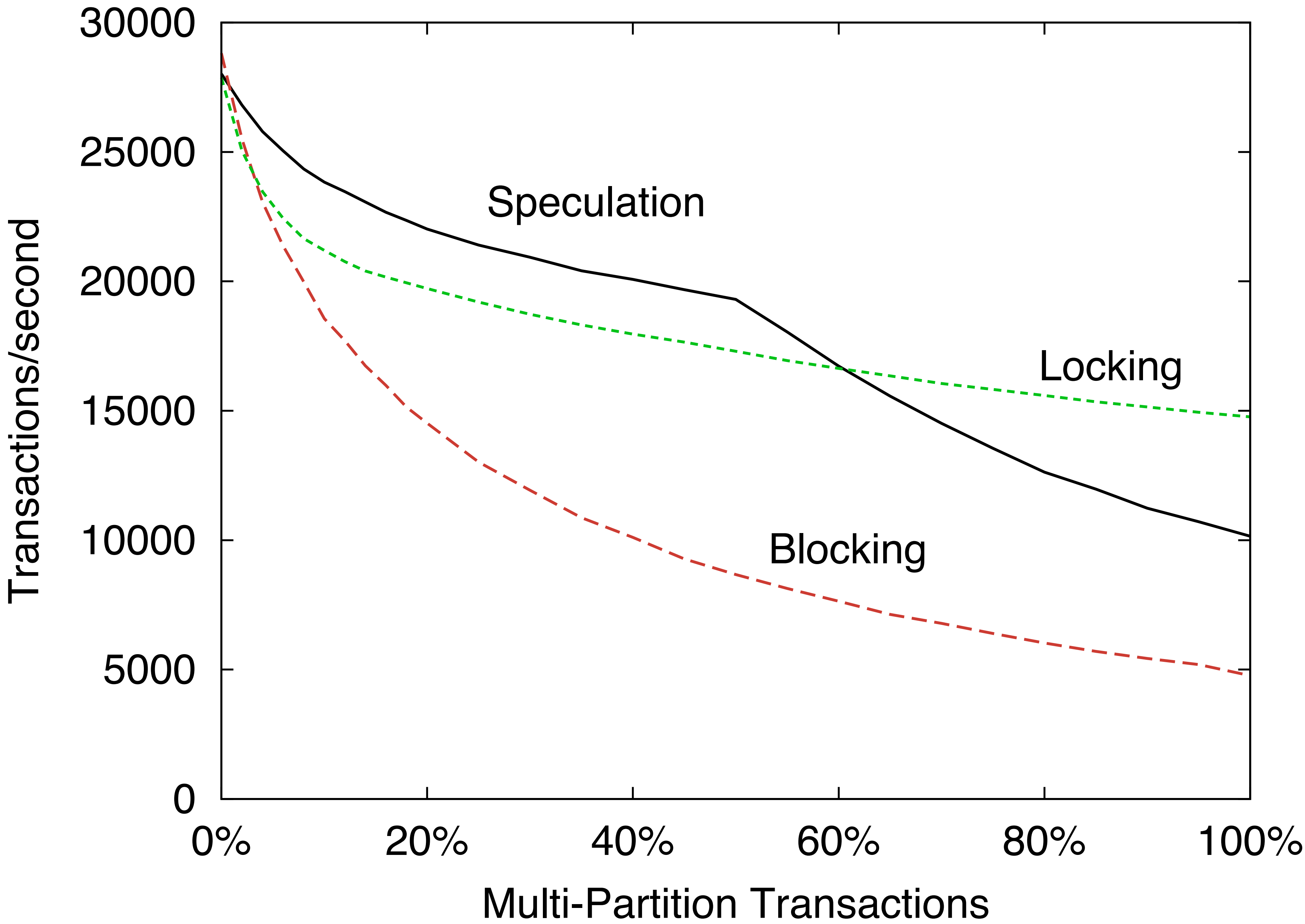
access half keys from each partition

single partition work = multi-partition work

No deadlocks, no aborts, no conflicts



Throughput (transactions/s)
30000
25000
20000
15000
10000
5000
0
0
20
40
60
80
100
% Multi-Partition
Speculation
Locking
Locking 40% Conflicts
Blocking



30000
25000
20000
15000
10000
5000
0
Transactions/second
0%
20%
40%
60%
80%
100%
Multi-Partition Transactions
Speculation
Locking
Blocking

TPC-C Based

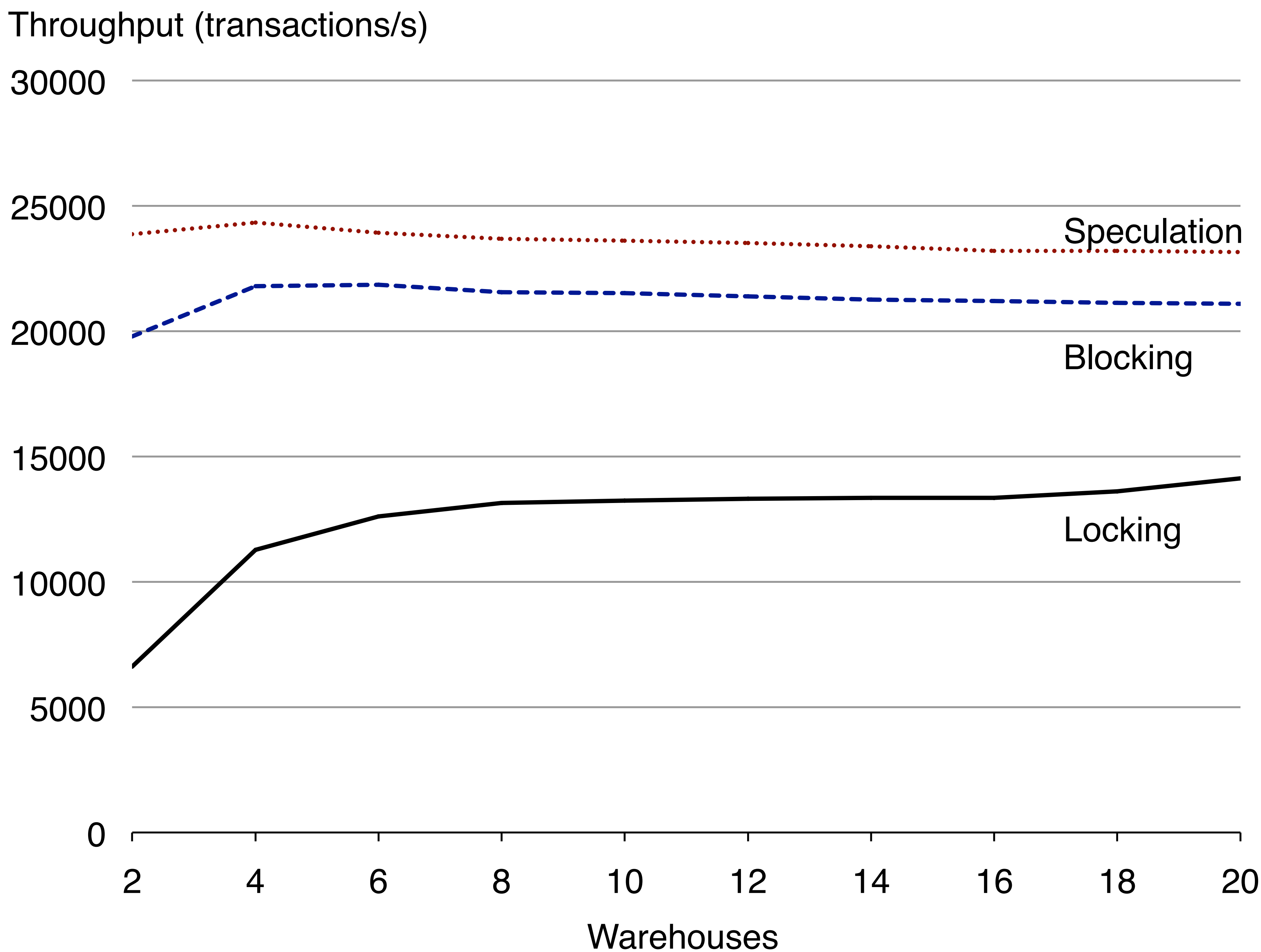
~11% multi-partition transactions

More complex locking

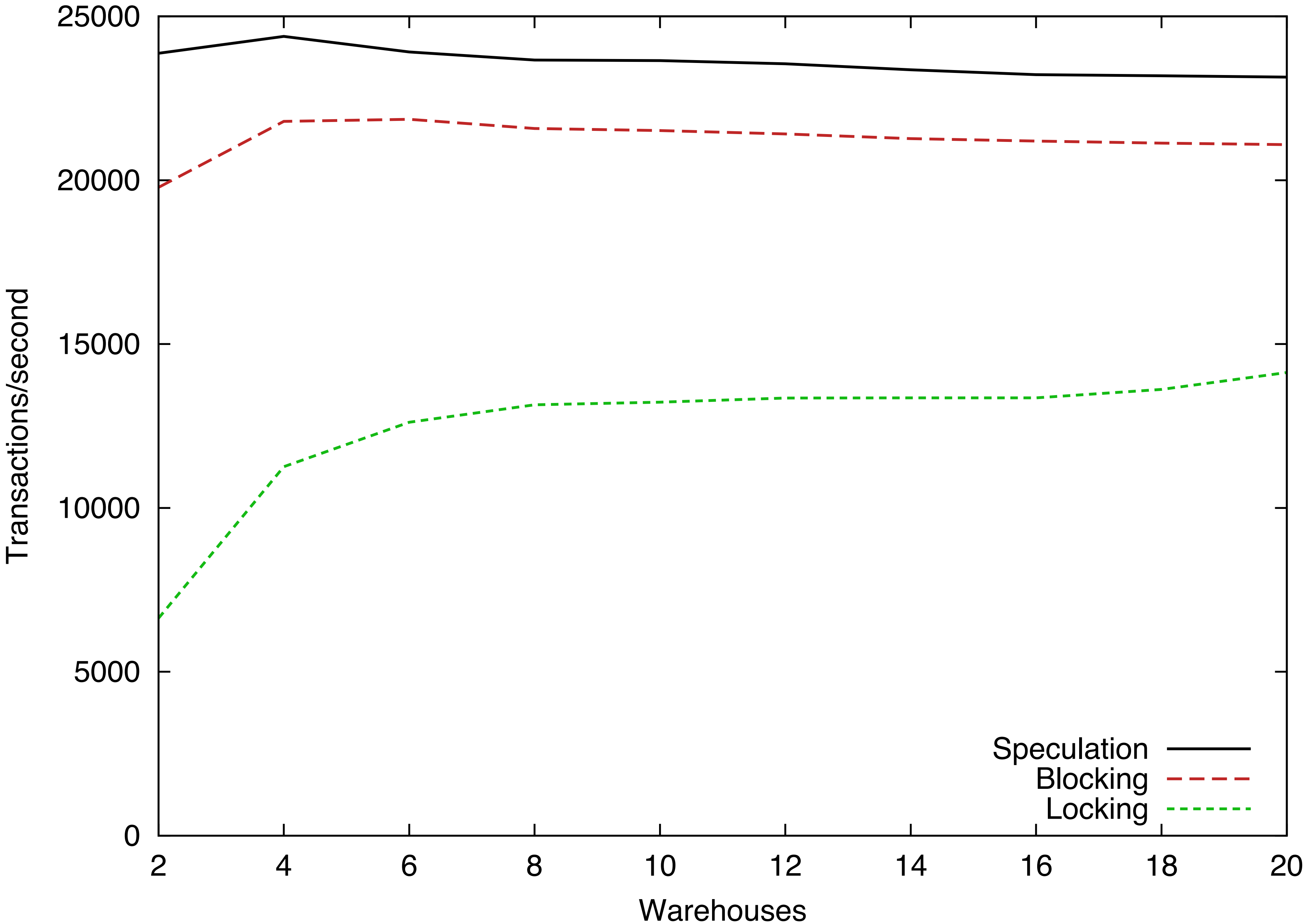
Many conflicts

Some deadlocks

Some aborts



Throughput (transactions/s)
30000
25000
20000
15000
10000
5000
0
2
4
6
8
10
12
14
16
18
20
Warehouses
Speculation
Blocking
Locking



25000
20000
15000
10000
5000
0
Transactions/second
2
4
6
8
10
12
14
16
18
20
Warehouses
Speculation
Blocking
Locking

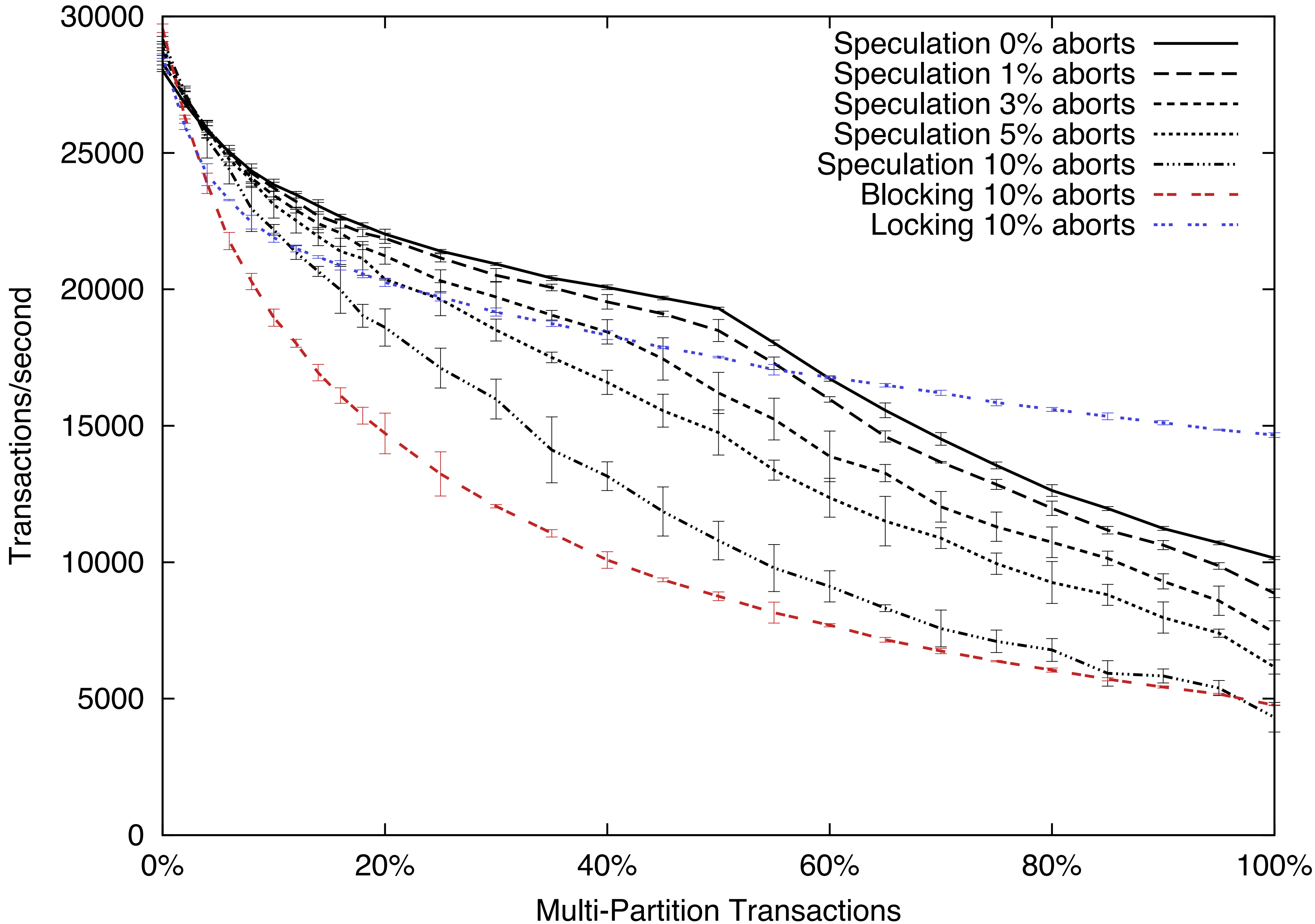
Speculative CC

better for “mostly partitionable” apps on main memory DBs

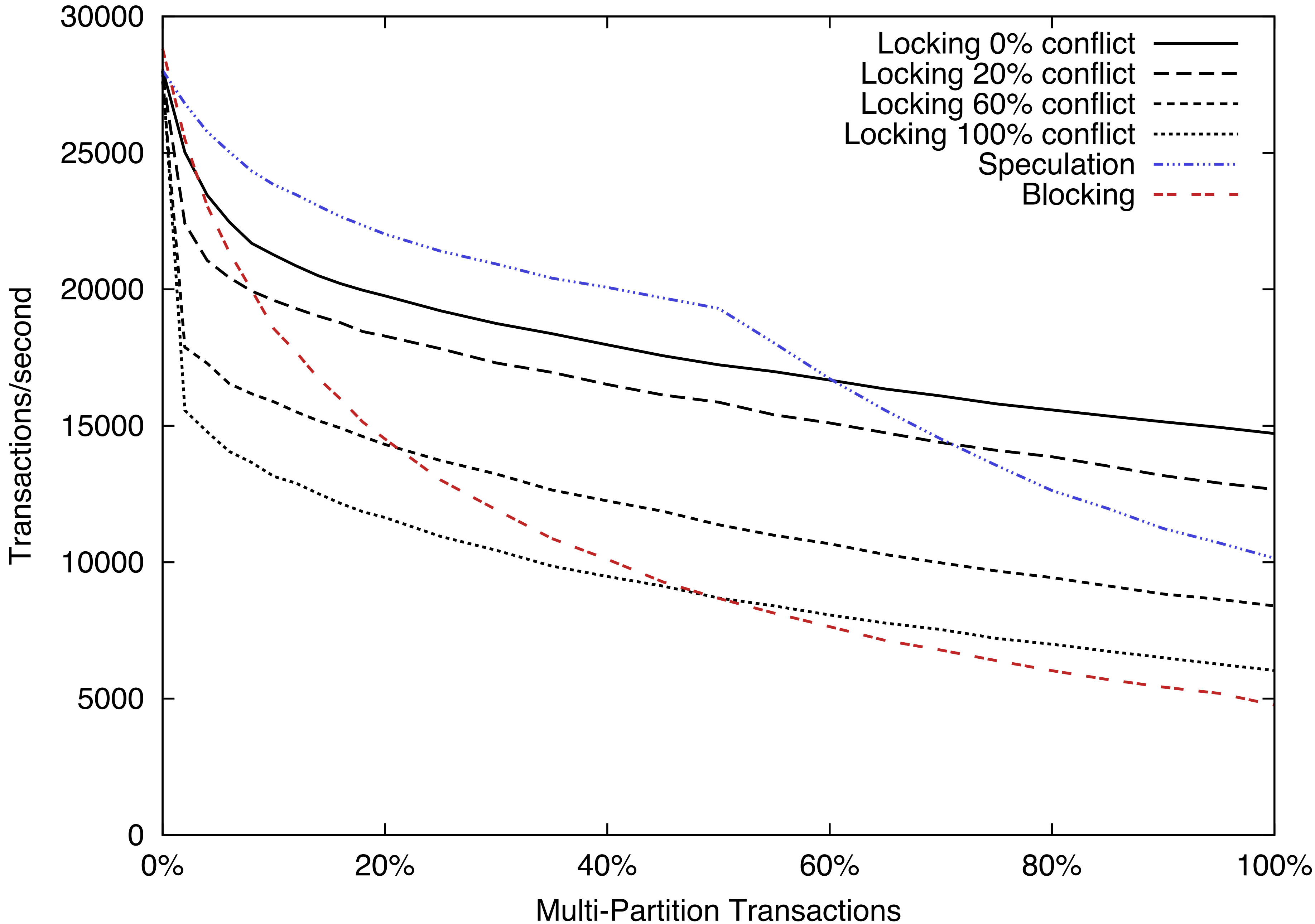
Up to 2X throughput

No locking overhead

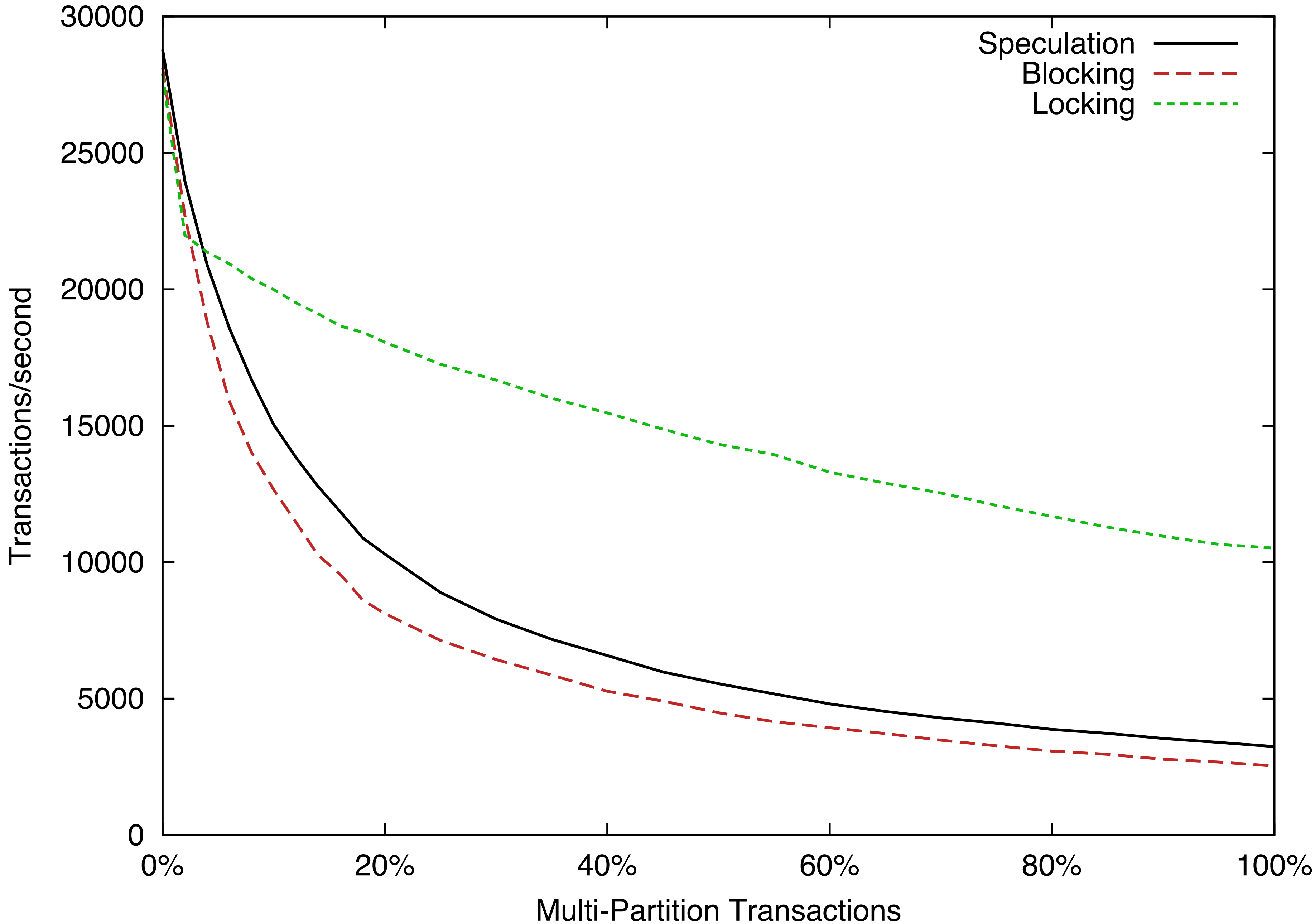
No deadlocks



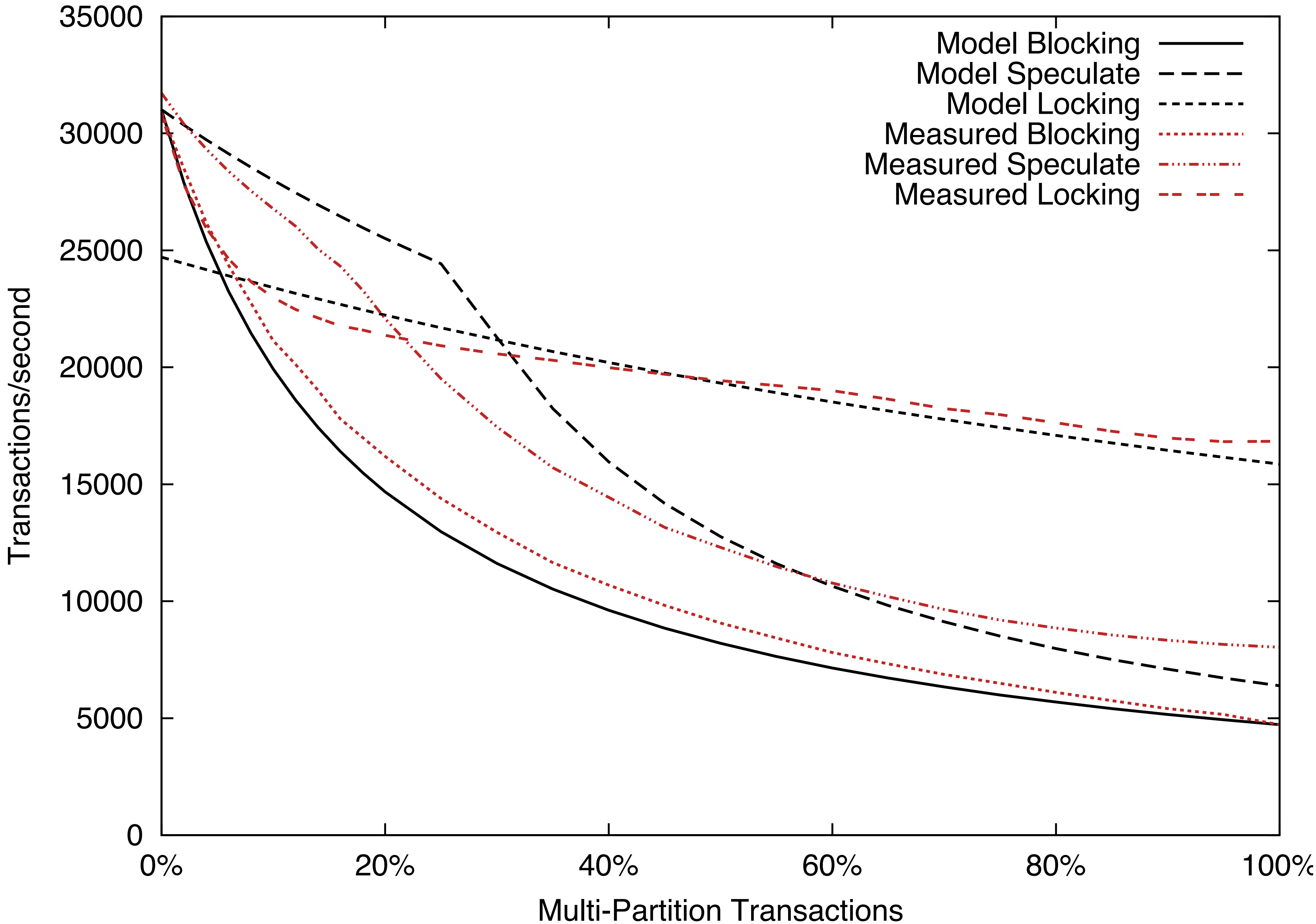
Speculation 0% aborts
Speculation 1% aborts
Speculation 3% aborts
Speculation 5% aborts
Speculation 10% aborts
Blocking 10% aborts
Locking 10% aborts
Transactions/second
Multi-Partition Transactions
30000
25000
20000
15000
10000
5000
0
0%
20%
40%
60%
80%
100%

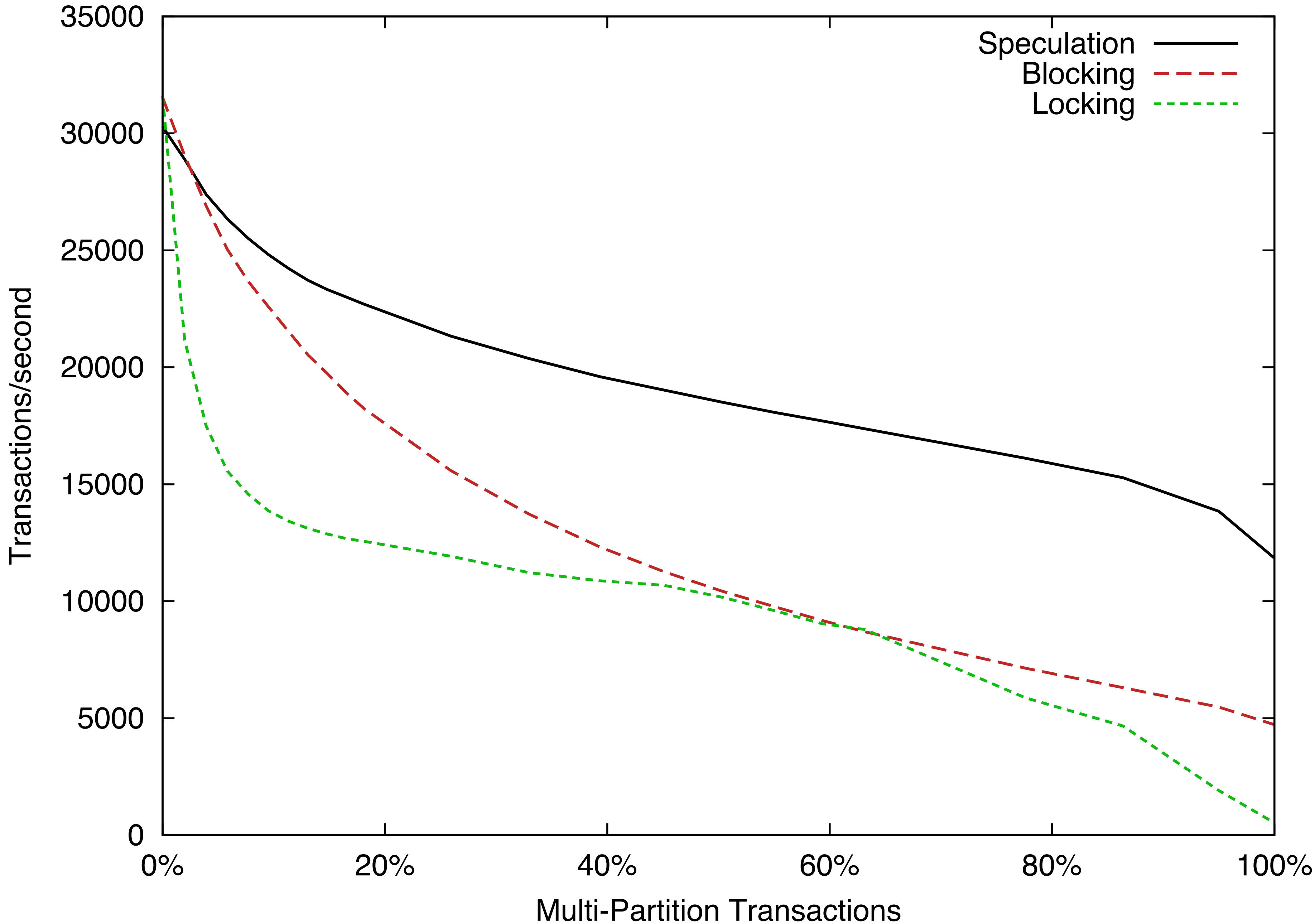


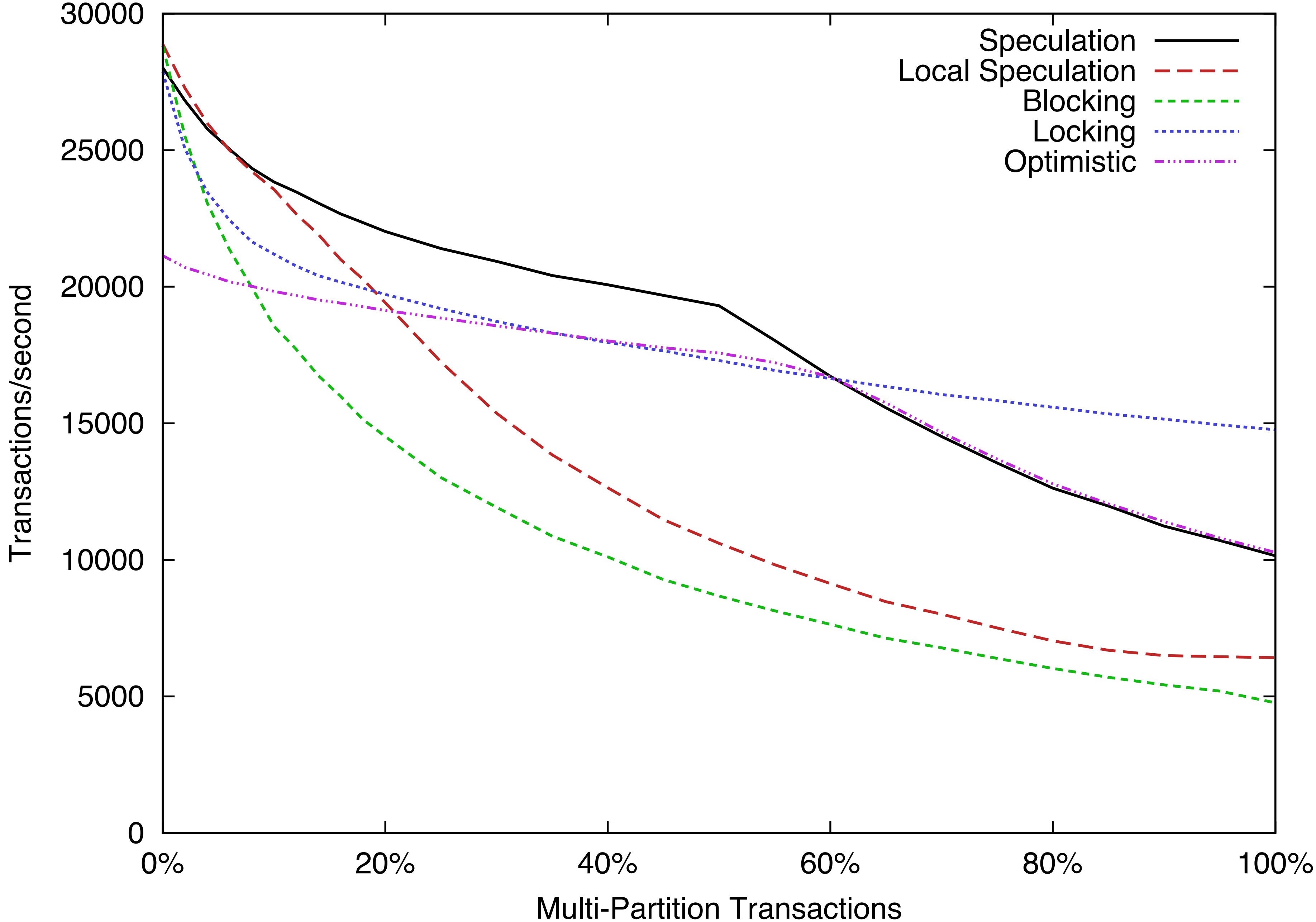
Locking 0% conflict
Locking 20% conflict
Locking 60% conflict
Locking 100% conflict
Speculation
Blocking
30000
25000
20000
15000
10000
5000
0
0%
20%
40%
60%
80%
100%
Transactions/second
Multi-Partition Transactions



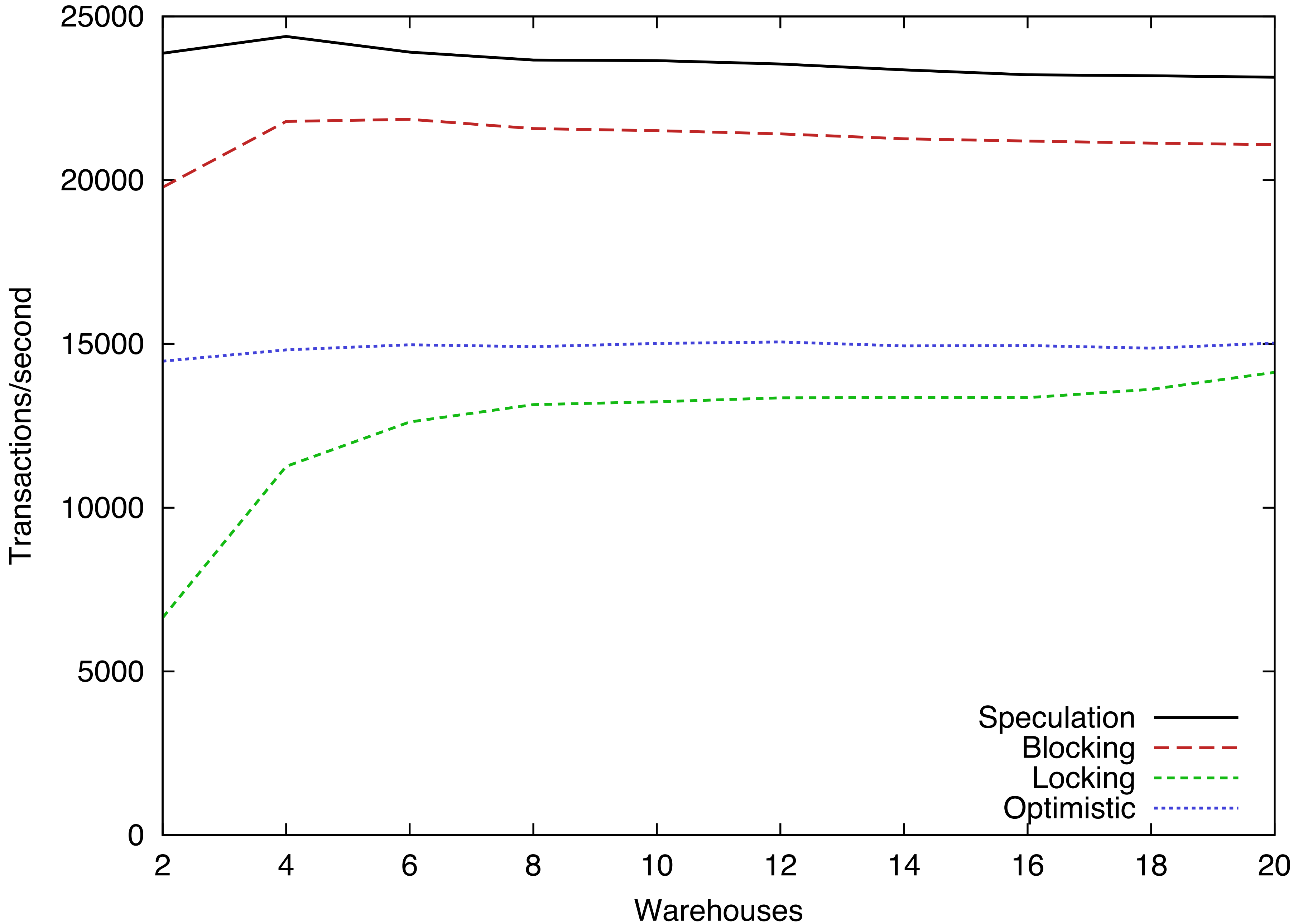
Speculation
Blocking
Locking
Transactions/second
Multi-Partition Transactions
0
5000
10000
15000
20000
25000
30000
0%
20%
40%
60%
80%
100%


Model Blocking
Model Speculate
Model Locking
Measured Blocking
Measured Speculate
Measured Locking
35000
30000
25000
20000
15000
10000
5000
0
Transactions/second
0%
20%
40%
60%
80%
100%
Multi-Partition Transactions


Speculation
Blocking
Locking
35000
30000
25000
20000
15000
10000
5000
0
0%
20%
40%
60%
80%
100%
Transactions/second
Multi-Partition Transactions



Speculation
Local Speculation
Blocking
Locking
Optimistic
Transactions/second
30000
25000
20000
15000
10000
5000
0
0%
20%
40%
60%
80%
100%
Multi-Partition Transactions



25000
20000
15000
10000
5000
0
2
4
6
8
10
12
14
16
18
20
Transactions/second
Warehouses
Speculation
Blocking
Locking
Optimistic