Distributed ACID Transactions in Apache Ignite

Akmal Chaudhri
GridGain

http://ignite.apache.org

#apacheignite

GridGain

My Background

- Pre-2000
  - Developer
  - Academic (City University)
  - Consultant
  - Technical Architect
- Post-2000
  - Senior Architect
  - Senior IT Specialist
  - Technical Instructor
  - Solutions Architect
  - Independent Consultant

My Background

- Broad industry experience
- Worked with various technologies
  - Programming languages
  - IDE
  - Database systems
- Client-facing roles
  - Developers
  - Senior executives
  - Journalists
- Publications and presentations


XML Data Management
Native XML and XML-Enabled Database Systems
Akmal B. Chaudhri
Awais Rashid
Roberto Zicari
INNOVATIVE TECHNOLOGY SERIES
INFORMATION SYSTEMS AND NETWORKS
Java & Databases
edited by
Akmal Chaudhri
WILEY
Succeeding with Object Databases
A Practical Look at Today's Implementations with Java and XML
Akmal B. Chaudhri
Roberto Zicari
OBJECT DATABASES in PRACTICE
AKMAL B. CHAUDHRI
MARY LOOMIS
Zohra Bellahsène
Akmal B. Chaudhri
Erhard Rahm
Michael Rys
Rainer Unland (Eds.)
LNCS 2824
Database and XML Technologies
First International XML Database Symposium, XSym 2003
Berlin, Germany, September 2003
Proceedings
Springer
Akmal B. Chaudhri
Mario Jeckle
Erhard Rahm
Rainer Unland (Eds.)
LNCS 2593
Web, Web-Services, and Database Systems
NODe 2002 Web- and Database-Related Workshops
Erfurt, Germany, October 2002
Revised Papers
Springer
Stéphane Bressan Akmal B. Chaudhri
Mong Li Lee Jeffrey Xu Yu
Zoé Lacroix (Eds.)
LNCS 2590
Efficiency and Effectiveness of XML Tools and Techniques and Data Integration over the Web
VLDB 2002 Workshop EEXTT
and CAiSE 2002 Workshop DIWeb
Revised Papers
Springer
Akmal B. Chaudhri
Rainer Unland
Chabane Djeraba
Wolfgang Lindner (Eds.)
LNCS 2490
XML-Based Data Management and Multimedia Engineering – EDBT 2002 Workshops
EDBT 2002 Workshops XMLDM, MDDE, and YRWS
Prague, Czech Republic, March 2002
Revised Papers
Springer
XML Data Management
Native XML and XML-Enabled Database Systems
XML数据管理
纯XML和支持XML的数据库系统
事例で学ぶ
オブジェクトデータベース活用技法

Agenda

- Apache Ignite Transactions Overview
- Concurrency Modes and Isolation Levels
- Two-Phase Commit (2PC) Protocol
- One-Phase Commit (1PC) Optimization
- RAM and Disk Level Consistency
- Deadlock Handling
- Demos
- Q & A

Apache Ignite Transactions Overview

- ACID-compliant
- Distributed
- Integrated with JTA
- Used by Financial Institutions
- Atomicity modes
  - Atomic
  - Transactional

GridGain

Atomicity Modes

- Atomic mode
  - Updates are applied right away
  - No locks => better performance
- Transactional mode
  - Operations meets transaction boundaries
  - Fully ACID-compliant

Concurrency Modes and Isolation Levels

- Concurrency modes for transactions
  - Pessimistic
  - Optimistic
- Isolation levels
  - READ_COMMITTED
  - REPEATABLE_READ
  - SERIALIZABLE
- All combinations can be used simultaneously

```
try (Transaction tx = transactions.txStart()) {
    Integer hello = cache.get("Hello");

    if (hello == 1)
        cache.put("Hello", 11);

    cache.put("World", 22);

    tx.commit();
}
```

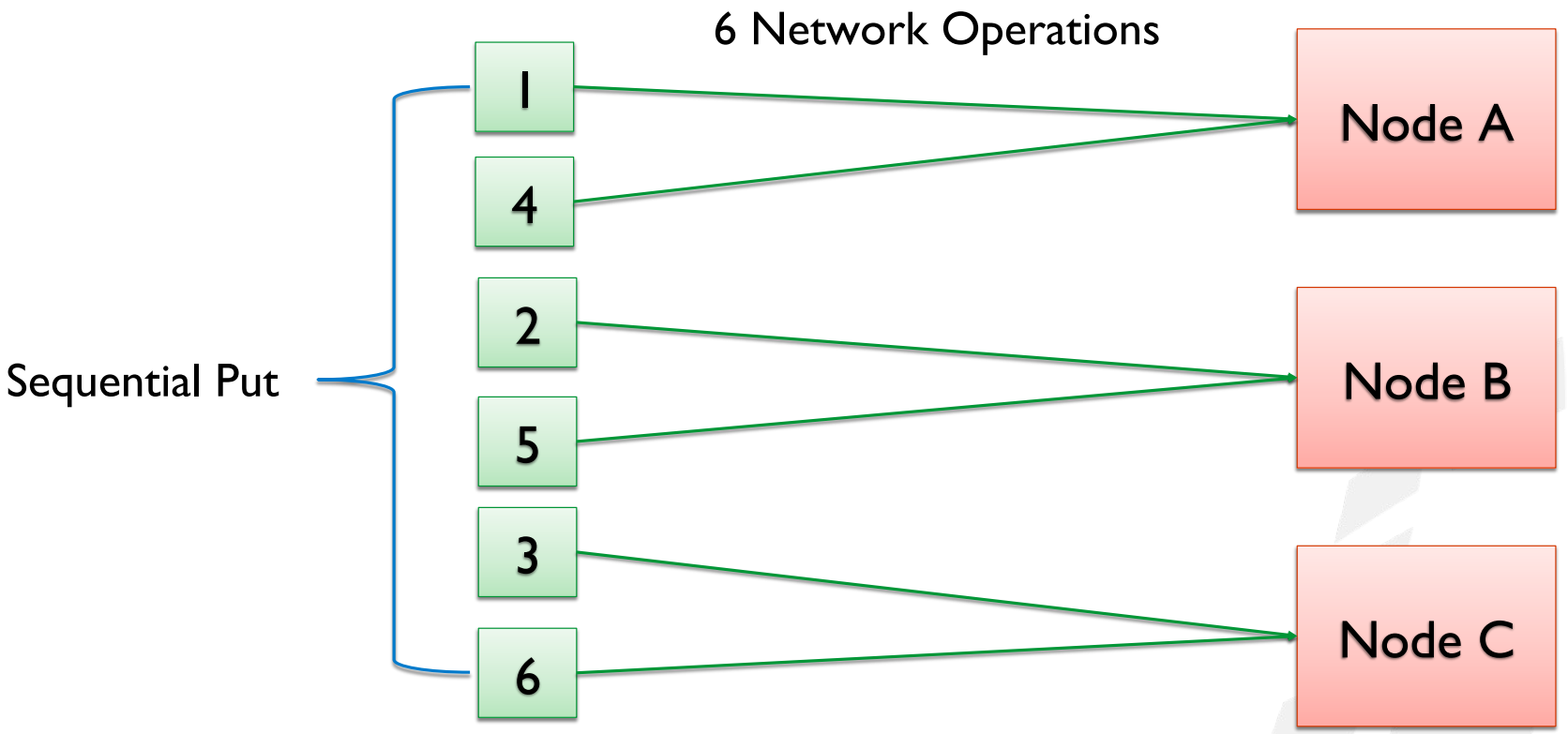
Pessimistic Transactions

```
try (Transaction tx = Ignition.ignite().transactions().txStart(PESSIMISTIC,
    REPEATABLE_READ)) {
    Account acct = cache.get(acctId);

    assert acct != null;

    // Deposit into account.
    acct.update(amount);

    // Store updated account in cache.
    cache.put(acctId, acct);

    tx.commit();
}
```

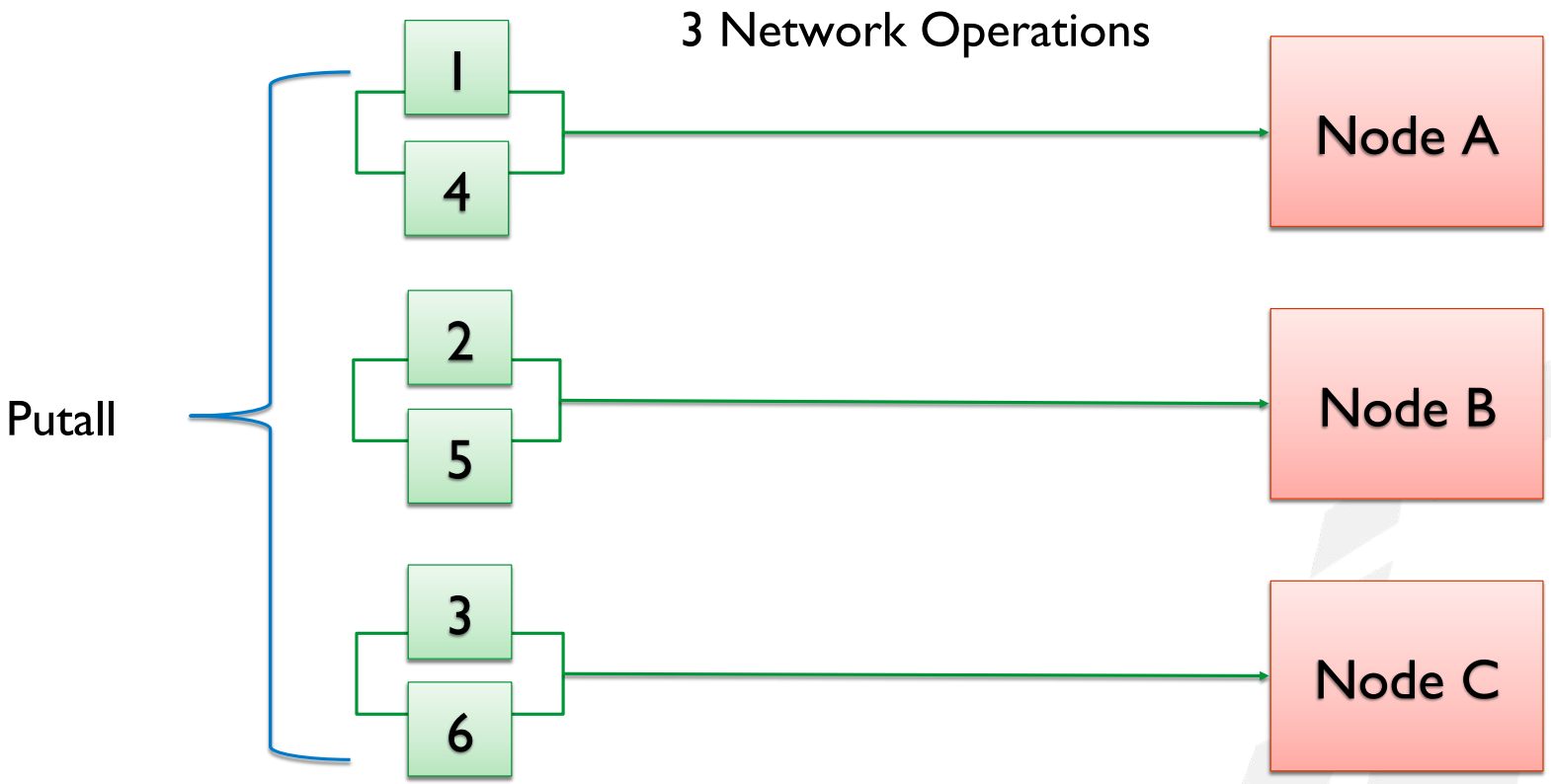
Optimistic Transactions

```
while (true) {
    try (Transaction tx =
        ignite.transactions().txStart(TransactionConcurrency.OPTIMISTIC,
            TransactionIsolation.SERIALIZABLE)) {
        Account acct = cache.get(acctId);

        assert acct != null;

        // Deposit into account.
        acct.update(amount);

        // Store updated account in cache.
        cache.put(acctId, acct);

        tx.commit();
        break;
    } catch (TransactionOptimisticException e) {
        // Transaction has failed. Retry.
    }
}
```

Performance Considerations: Put vs PutAll

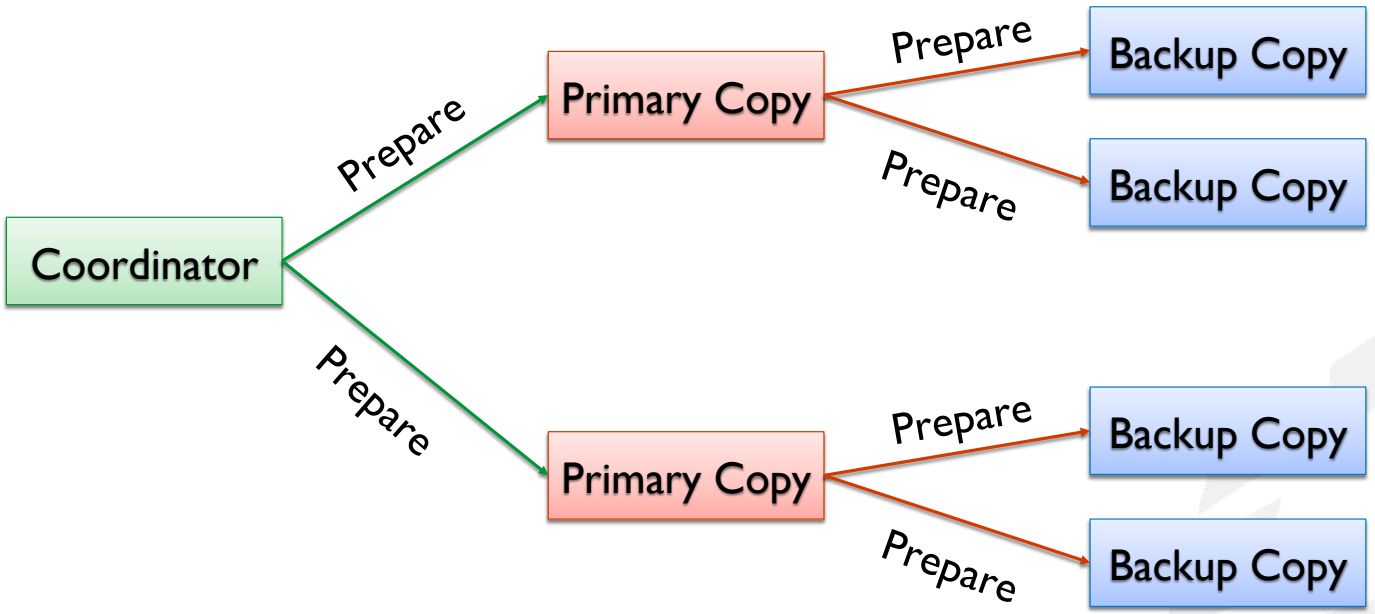


Performance Considerations: Put vs PutAll



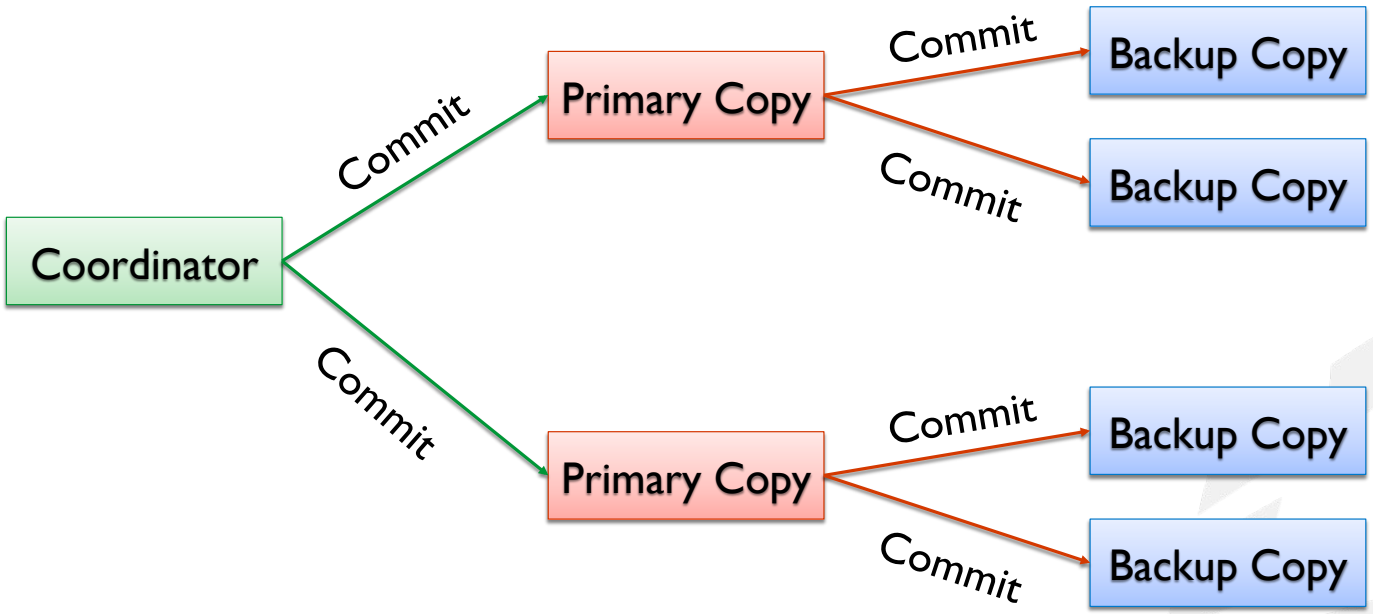
Two-Phase Commit protocol

Phase 1 - Prepare



Two-Phase Commit protocol

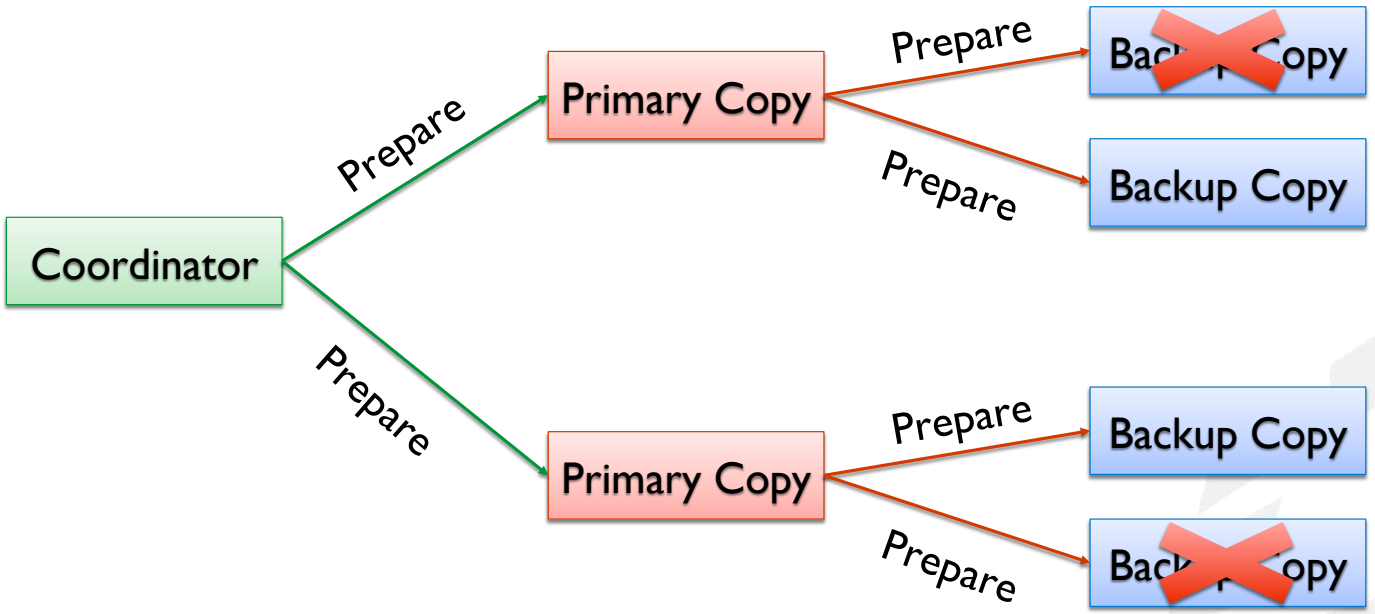
Phase 2 - Commit



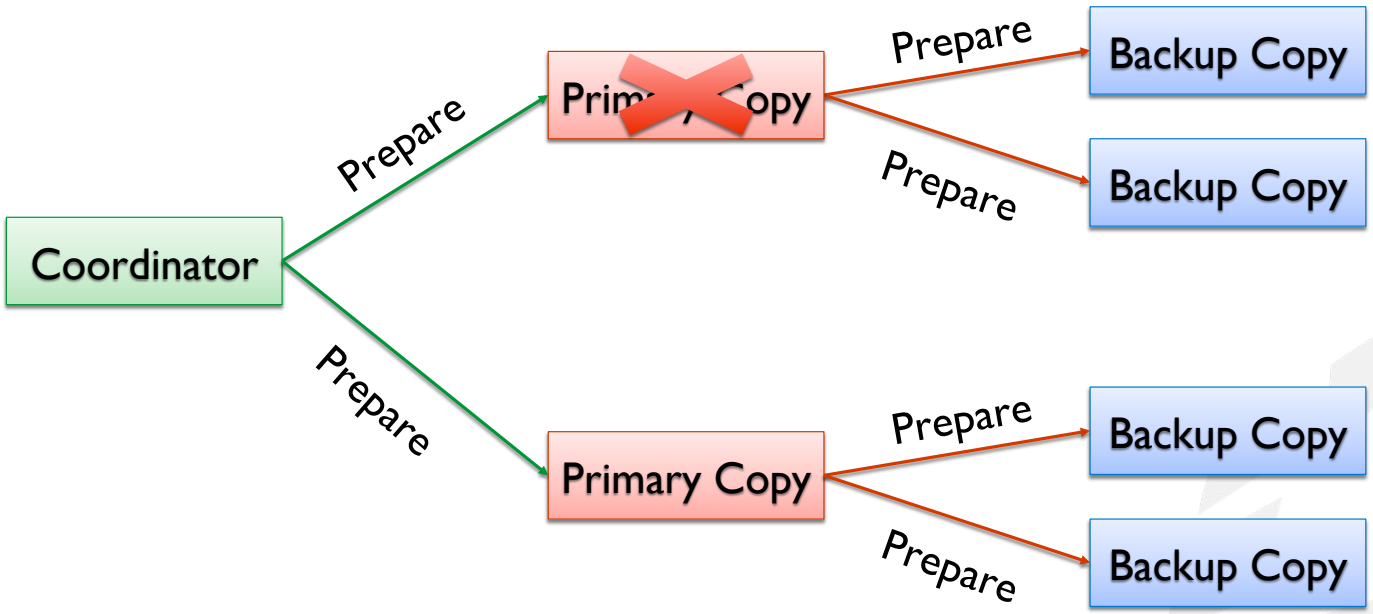
Two-Phase Commit: Benefits

- Advantages of in-memory
  - In-memory cache is volatile
  - The only reply for the "Prepare" is "Yes"
- Expands to the disk level
  - Ignite Persistent Store
  - 3rd party Persistence

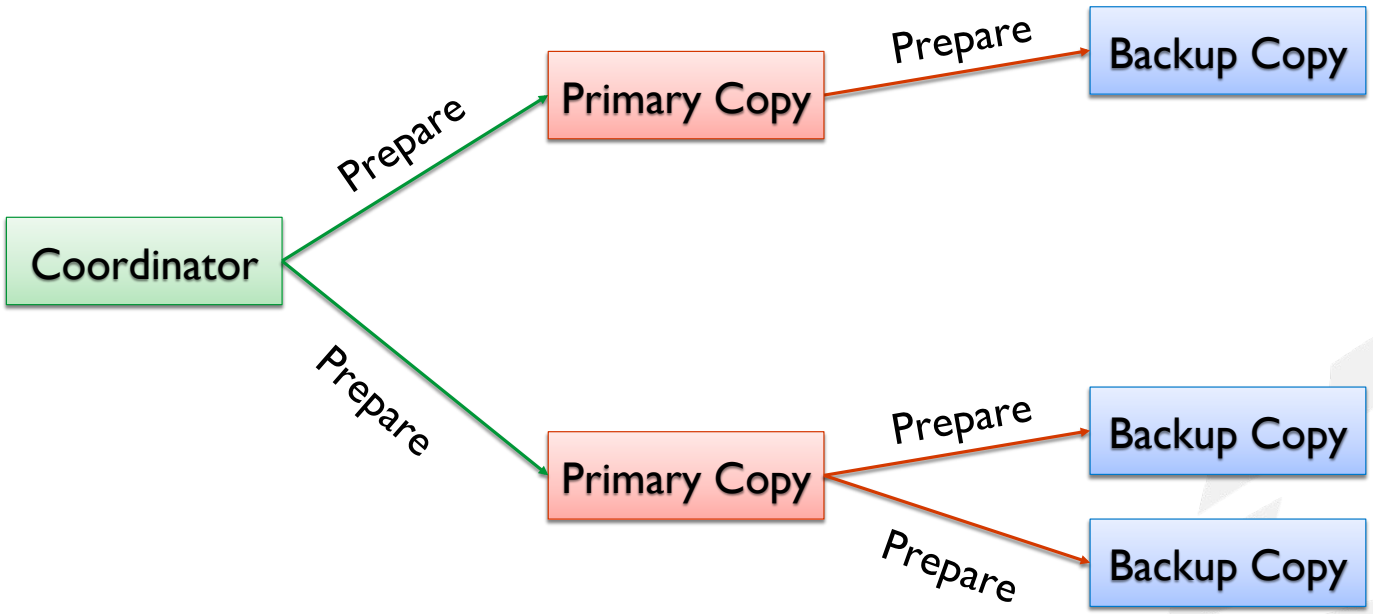
Two-Phase Commit: Backup Node Failures



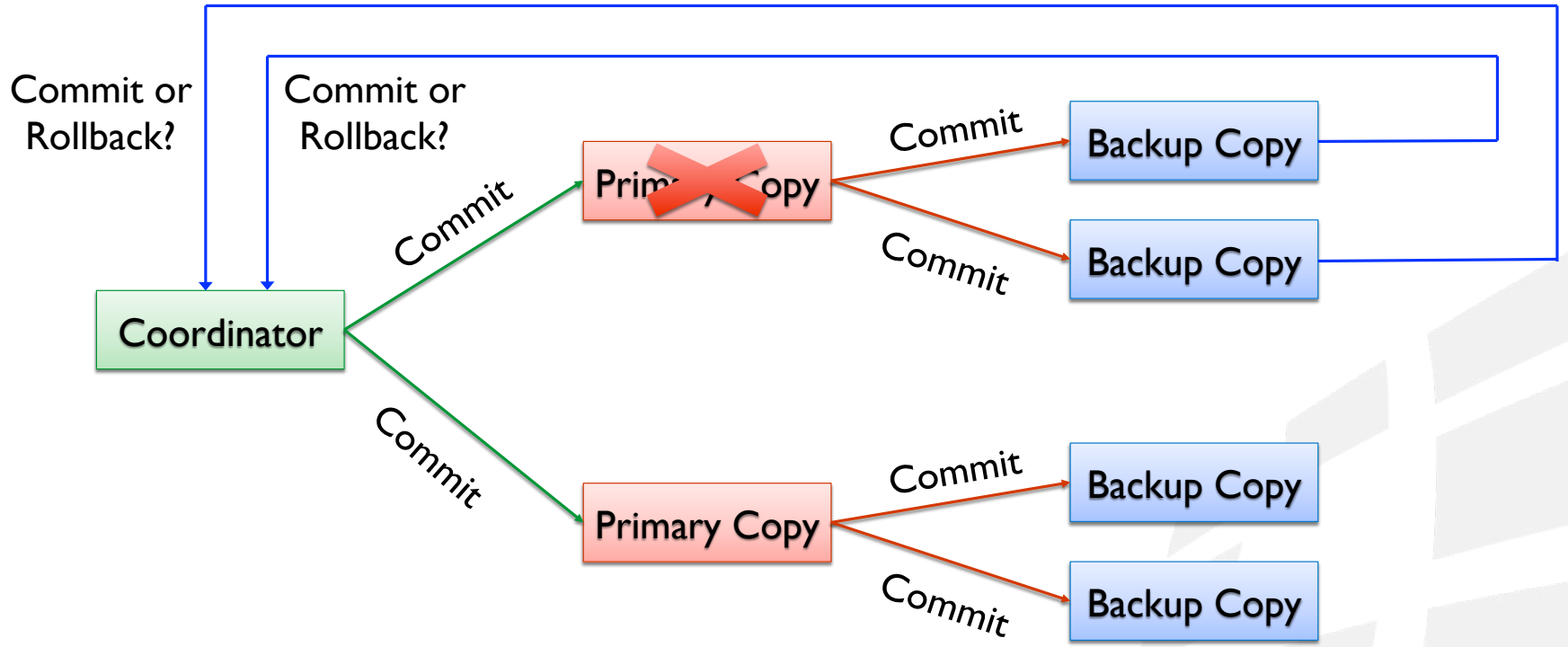
Two-Phase Commit: Primary Node Failures



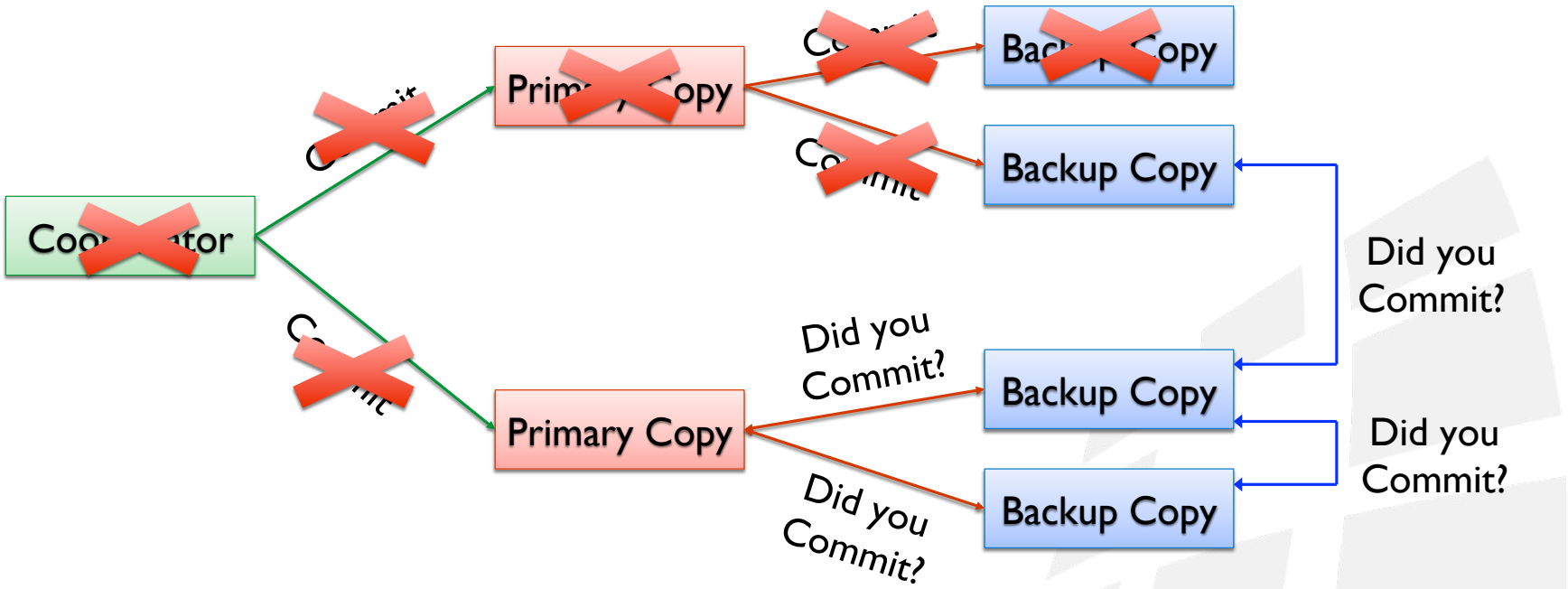
Two-Phase Commit: Primary Node Failures



Two-Phase Commit: Primary Node Failures



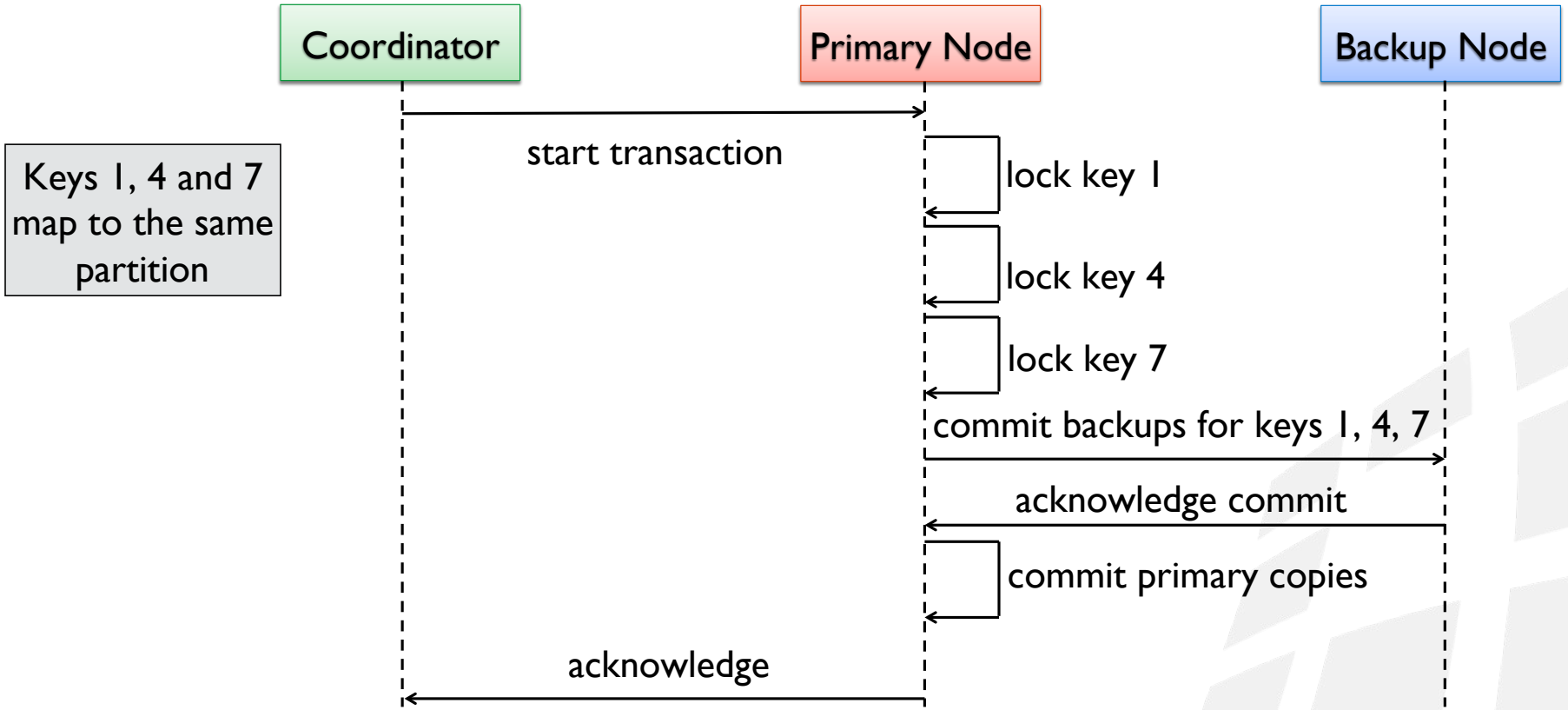
Two-Phase Commit: Coordinator Failure Recovery Protocol



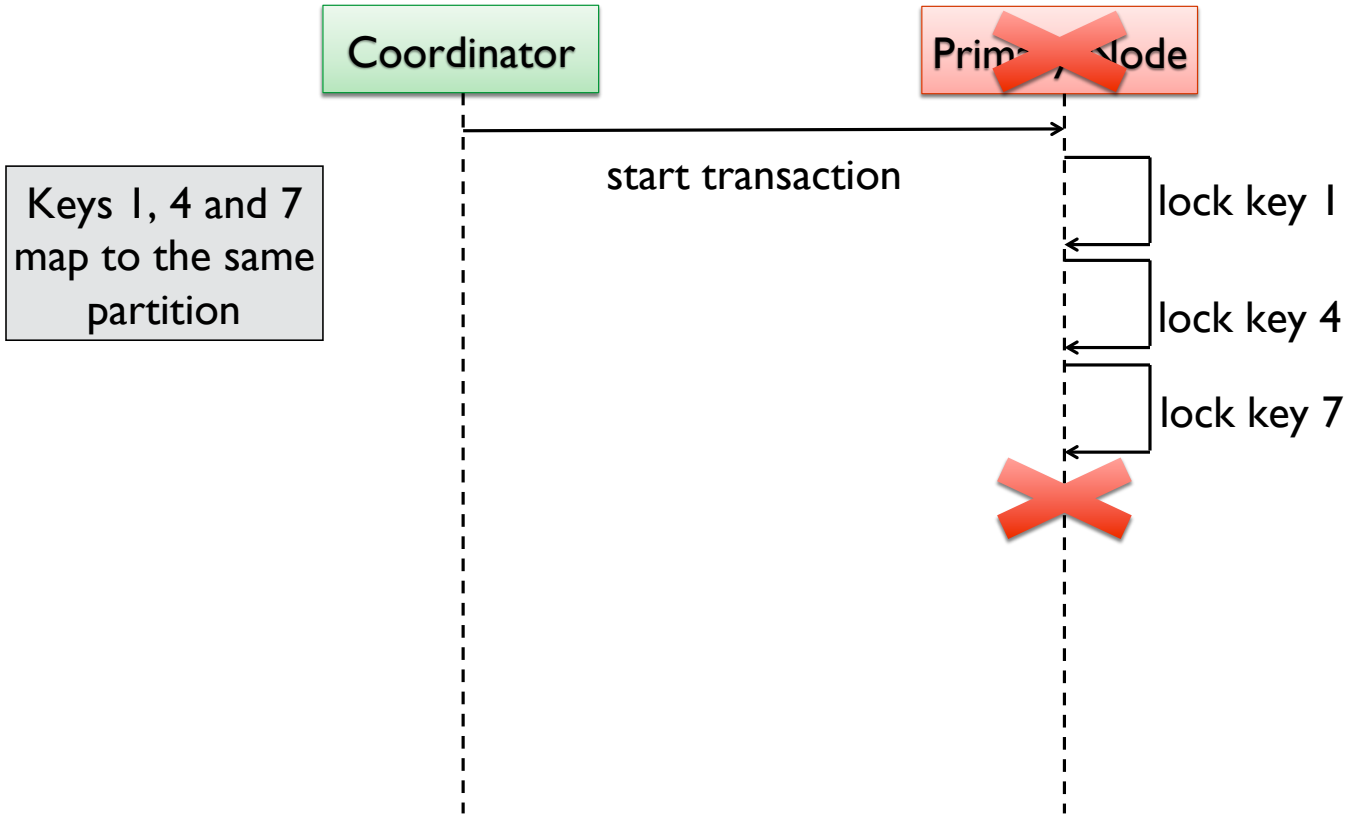
One-Phase Commit Protocol

- 1PC is 2PC optimization
- For best performance
  - Minimize cluster nodes in a transaction
  - All transaction entries belong to same partition
  - Backup copies will also be grouped together
  - Custom affinity mapping
- Example
  - All employees working for the same company will be mapped to the same partition using companyId

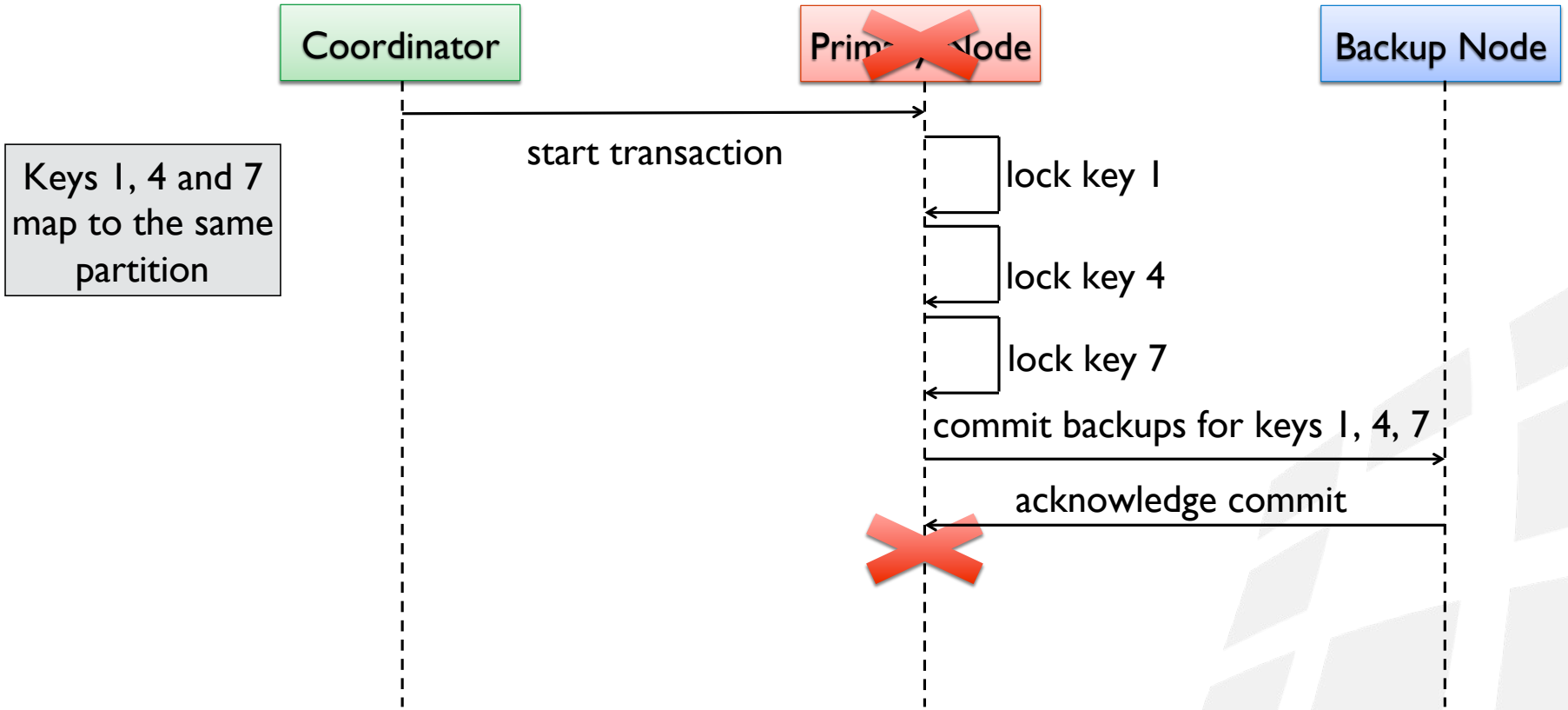
One-Phase Commit (1PC)



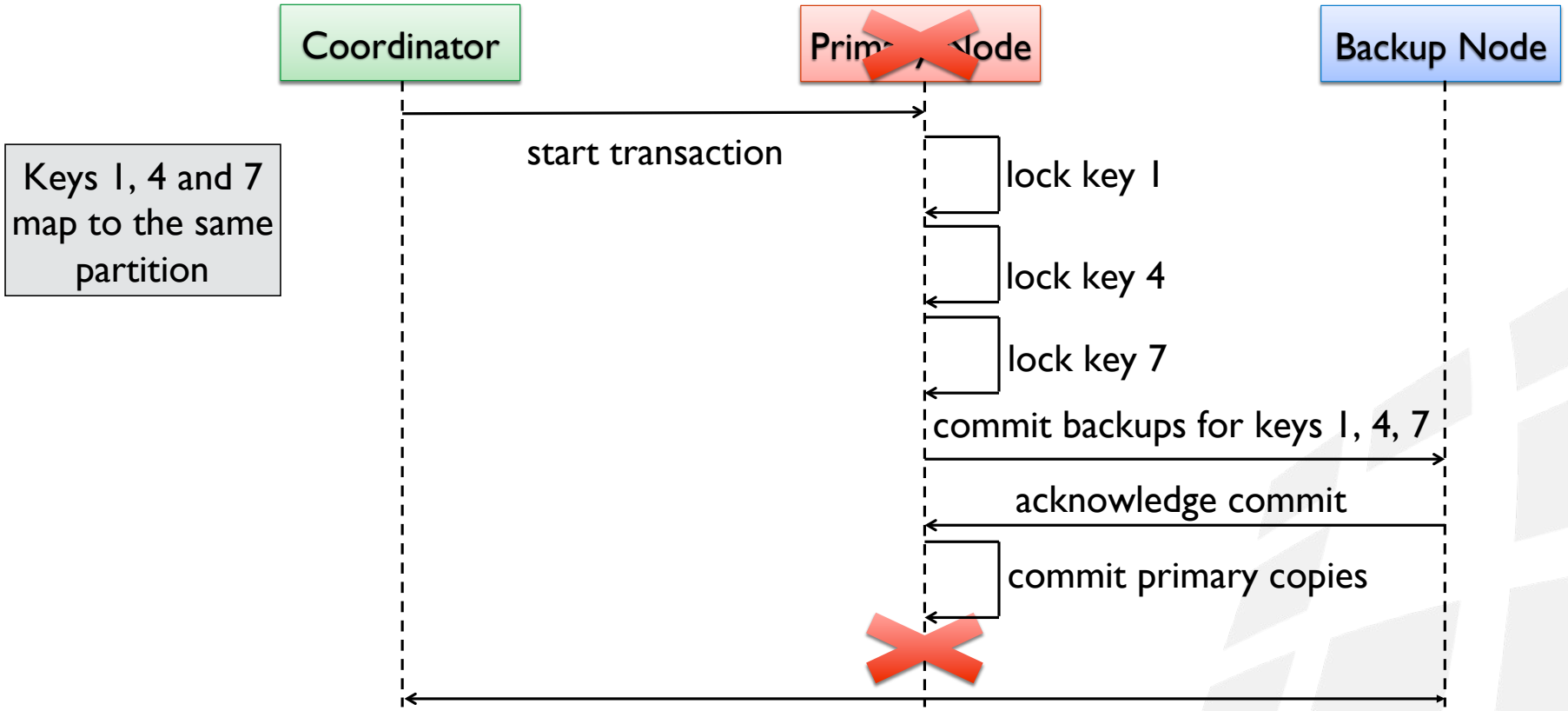
One-Phase Commit: Primary Node Failure



One-Phase Commit: Primary Node Failure

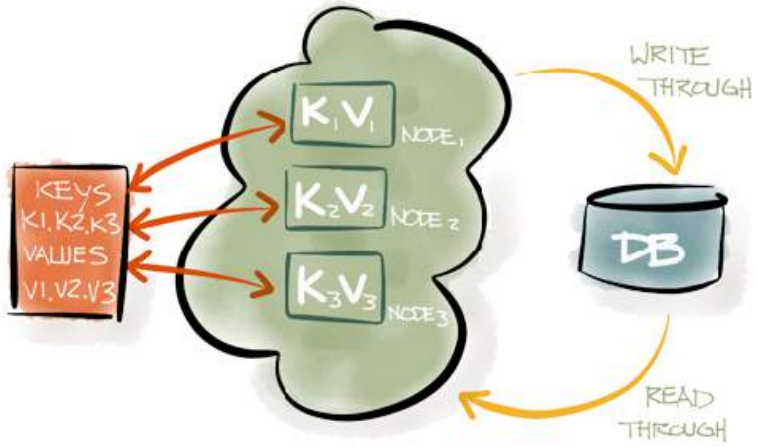


One-Phase Commit: Primary Node Failure Recovery Protocol



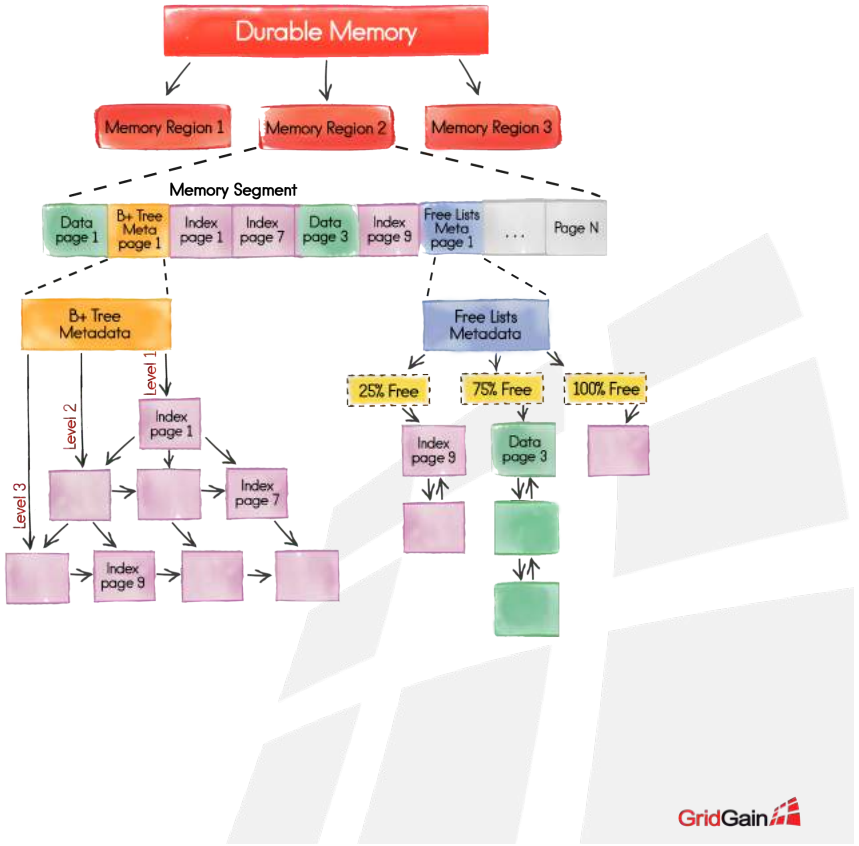
RAM and Disk Level Consistency

- Data are stored both in RAM and on disk
- ACID-compliant on all the levels
- 2PC Protocol is expanded to disk
  - Ignite Persistent Store
  - 3rd Party Stores



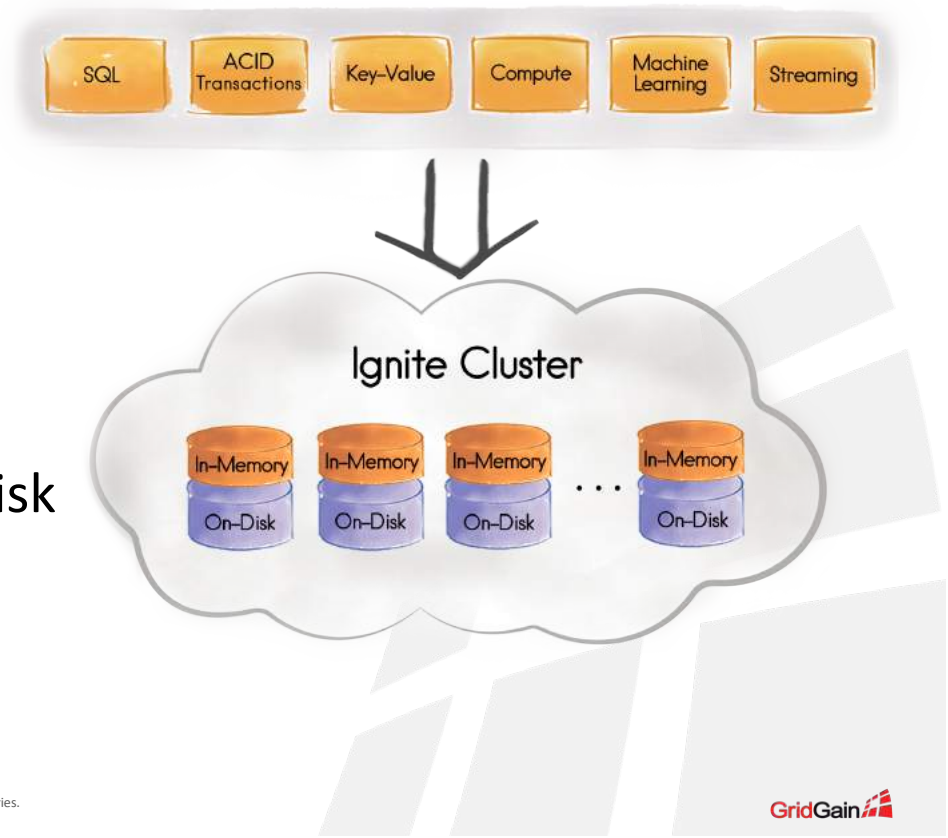
Durable Memory Architecture

- Memory split into *regions*
- Regions split into *segments*
- Segments split into *pages*
- Free-Lists track free space
- Pages can store:
  - Data
  - Indexes
  - Metadata



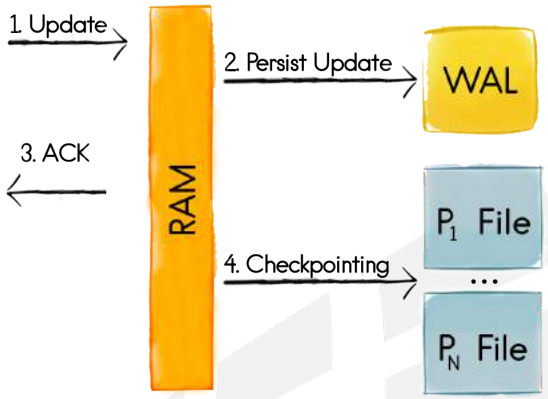
Ignite Persistent Store

- Expands Virtual Memory to Disk
  - Flash, SSD, Intel 3D Xpoint
- ANSI-99 SQL-Compliant
  - Join RAM and disk datasets
- ACID-Compliant
  - Write-ahead logs
- Stores superset of data
  - If a page is in RAM, it is always on disk
  - Same format
- Instantaneous Restarts



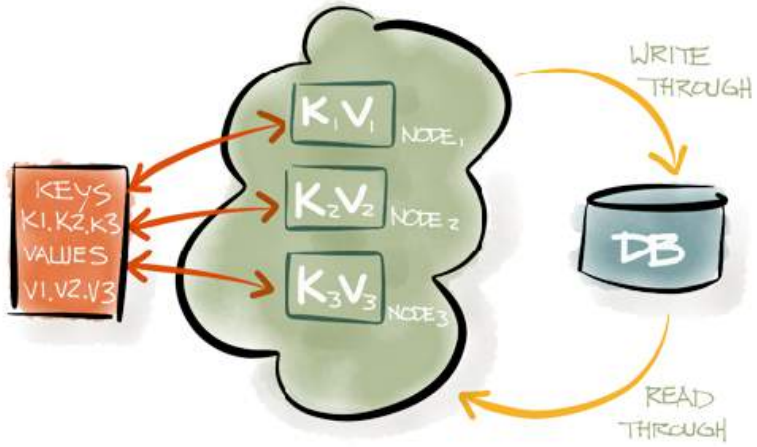
Durability and Consistency

- Write-Ahead Log (WAL)
  - Updates appended to node's WAL
  - WAL propagates updates to disk
  - Provides recovery mechanism
- Checkpointing
  - Triggered periodically based upon config
  - Copy dirty pages from RAM to disk
  - Reduces WAL size



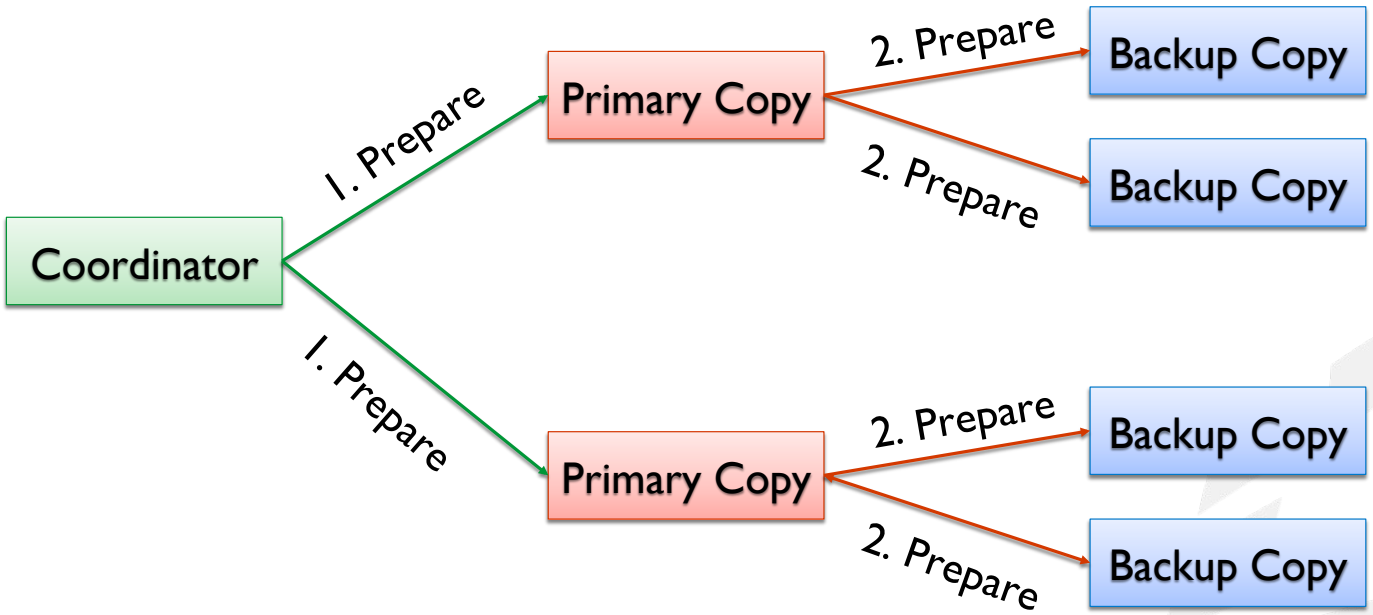
3rd Party Stores: Consistency

- Coordinator writes to the database first
- Commits in the cluster afterwards
- The database must be transactional
  - RDBMS

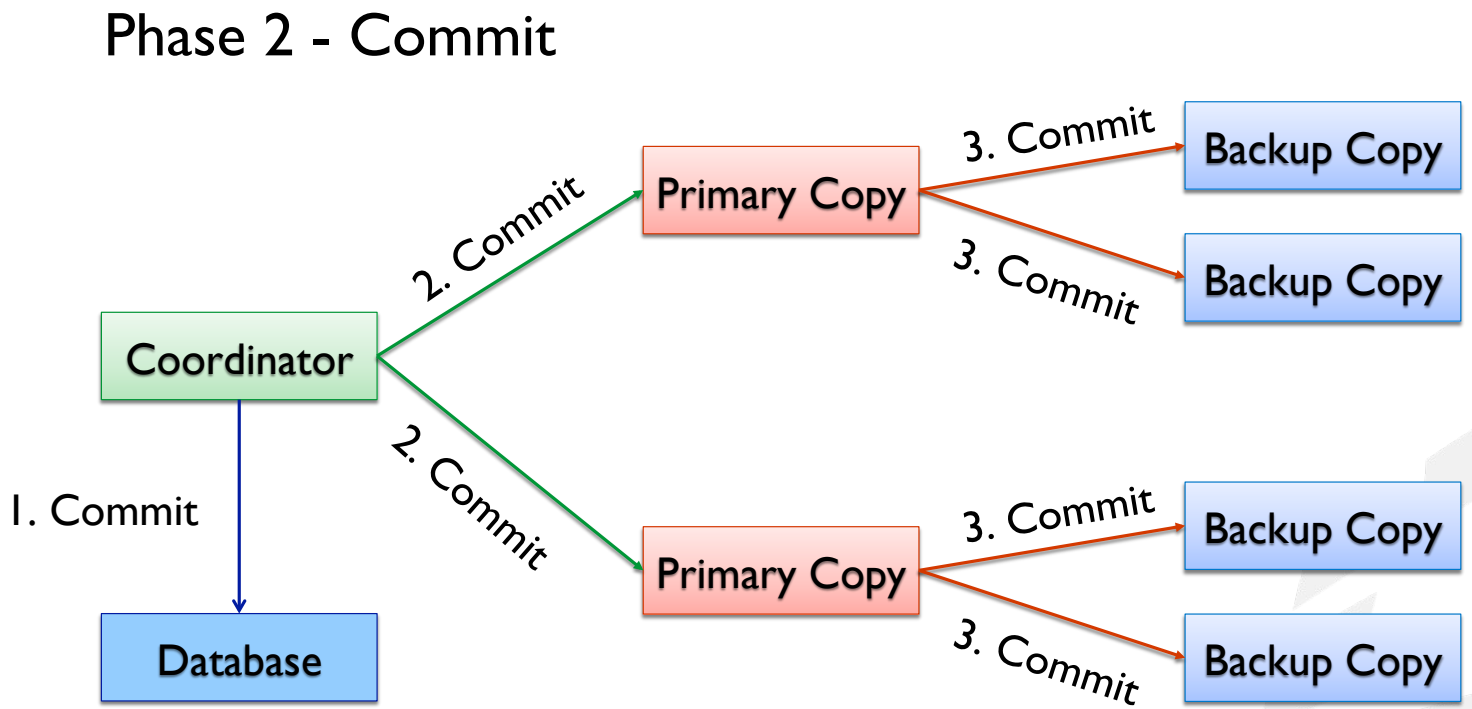


3rd Party Stores: 2PC Protocol

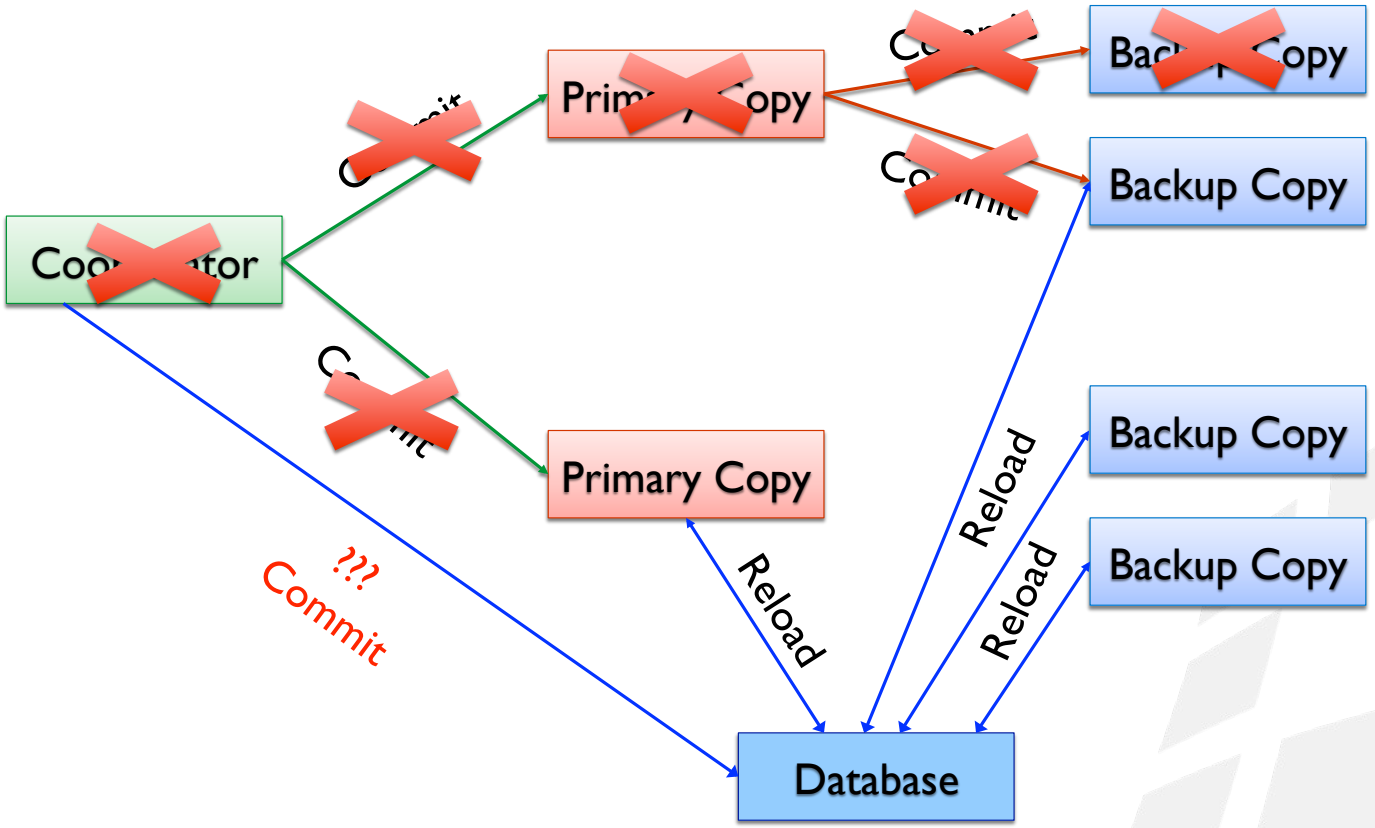
Phase 1 - Prepare



3rd Party Stores: 2PC Protocol



3rd Party Stores: Failures



Deadlock Handling

- Deadlock Detection
  - Helps prevent distributed deadlocks
  - Facilitates deadlock debugging and avoidance
- Deadlock-free Transactions
  - Optimistic and Serializable mode
  - Avoids possibility of deadlock
  - One of the transactions always fails
  - Failed transaction can be retried

Demos

GridGain

ANY QUESTIONS?

Thank you for joining us. Follow the conversation.

http://ignite.apache.org

 #apacheignite

GridGain

Resources

- Two-Phase-Commit for Distributed In-Memory Caches
  - http://gridgain.blogspot.co.uk/2014/09/two-phase-commit-for-distributed-in.html
- Two-Phase-Commit for In-Memory Caches - Part II
  - http://gridgain.blogspot.co.uk/2014/09/two-phase-commit-for-in-memory-caches.html
- One-Phase-Commit - Fast Transactions For In-Memory Caches
  - http://gridgain.blogspot.co.uk/2014/09/one-phase-commit-fast-transactions-for.html