Final Project: Improving Hermes

Introduction to Databases
DataLab
CS, NTHU

Outline

- Project Goal
- Introduction to Hermes
- Target Workloads
- Where Can I Optimize?
- Stages
- Final Presentation
- Timeline

Outline

- Project Goal
- Introduction to Hermes
- Target Workloads
- Where Can I Optimize?
- Stages
- Final Presentation
- Timeline

Project Goal

- This final project tests whether you are capable to do a research on a complex distributed DBMS.

- In order to test this ability, you need to optimize **Hermes**, the latest elastic technique we propose, for better performance.

Outline

- Project Goal
- Introduction to Hermes
- Target Workloads
- Where Can I Optimize?
- Stages
- Final Presentation
- Timeline

Hermes

A Next Generation Data Re-partitioning Mechanism for Elastic NewSQL System

Our paper is going to show on **SIGMOD'21**!

Don't Look Back, Look into the Future: Prescient Data Partitioning and Migration for Deterministic Database Systems

Yu-Shan Lin, Ching Tsai, Tz-Yu Lin, Yun-Sheng Chang, Shan-Hung Wu
Nation Tsing Hua University
Taiwan, R.O.C.
{yslin,ctsai,tylin,yschang}@datalab.cs.nthu.edu.tw,shwu@cs.nthu.edu.tw

ABSTRACT

Deterministic database systems have been shown to significantly improve the availability and scalability of a distributed database system deployed on a shared-nothing architecture across WAN while ensuring strong consistency. However, their scalability and performance advantages highly depend on the quality of data partitioning due to the reduced flexibility in transaction processing. Although a deterministic database system can employ workload driven data (re-)partitioning and live data migration algorithms to partition data, we found that the effectiveness of these algorithms is limited in complex real-world environments due to the unpredictability of machine workloads. In this paper, we present Hermes, a deterministic database system prototype that, for the first time, does *not* rely on sophisticated data partitioning to achieve high scalability and performance. Hermes employs a novel transaction routing mechanism that jointly optimizes the balance of machine workloads, data (re-)partitioning, and live data migration by looking into the queued transactions to

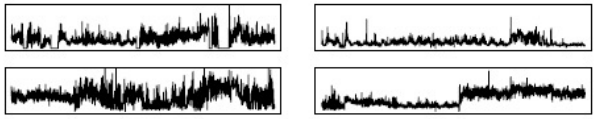


Figure 1: 30-day workloads of some nodes in a cluster owned by Google show unpredictable, episodic changes at small time scales and changes due to dynamic machine provisioning. Axes x and y represent the elapsed time and CPU loads, respectively.

between replicas or partitions when processing a transaction. The benefits drive the development of new commercial database systems such as VoltDB [4] and FaunaDB [2] that target high-performance applications at scale.

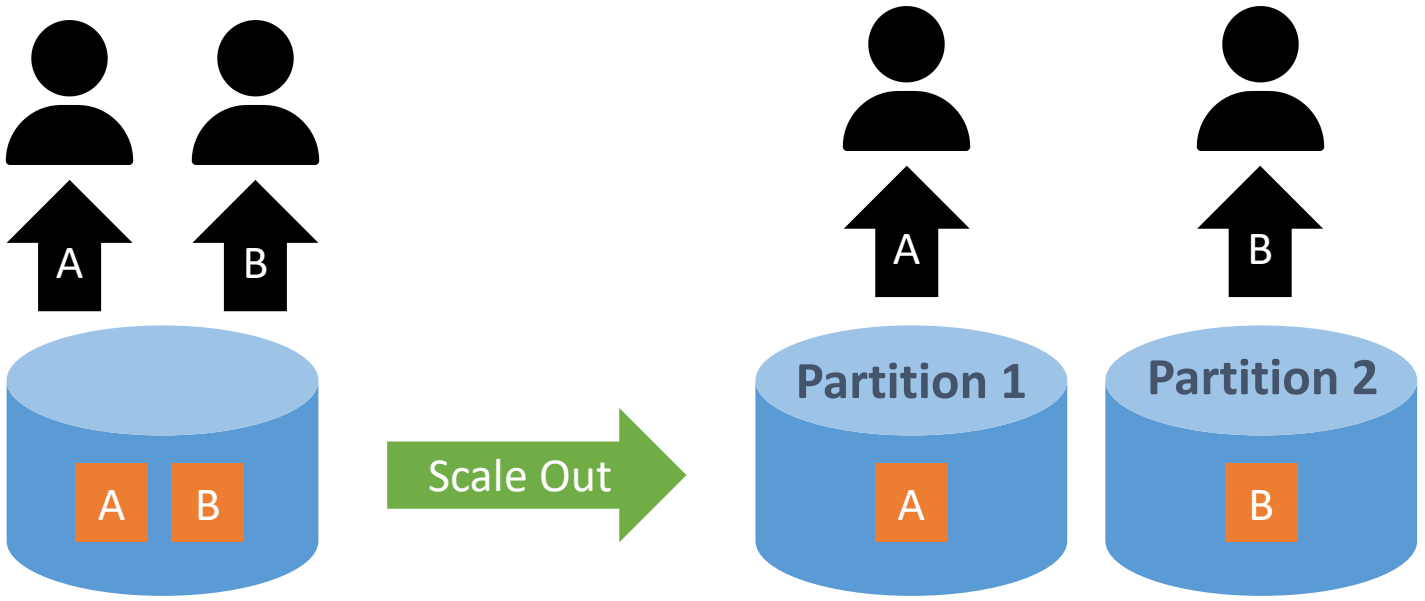
However, the availability and scalability advantages of a deterministic database system come at costs. One major drawback is the

Recap: Requirements for NewSQL

- Transactions
  - ACID
- Cloud Databases
  - **High scalability**
  - High availability
  - Elasticity

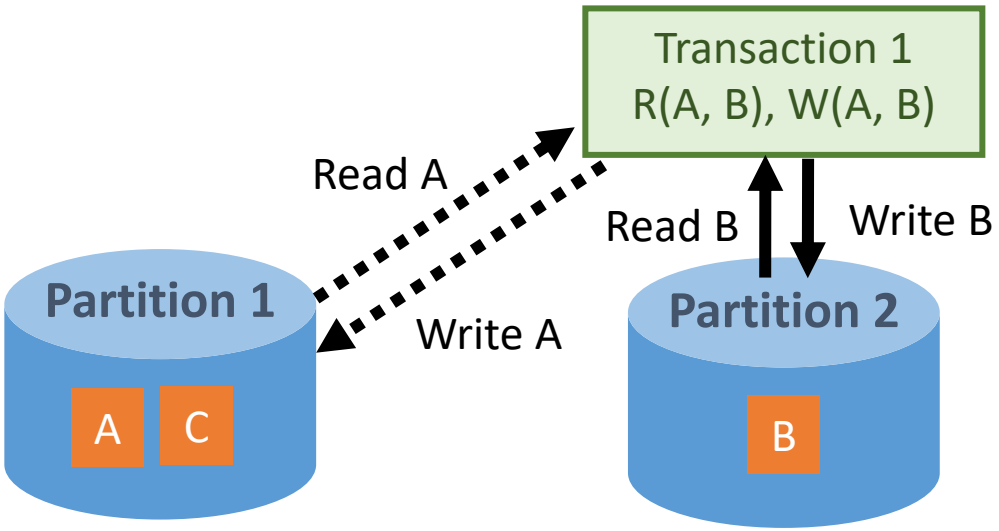
Scalability via Data Partitions

- A modern RDBMS usually **scales out** by partitioning its database.



Good Scalability Rely on High Quality of Data Partitions

- However, how to partition the database is also very important.
  - Poor data partitions may introduce **distributed transactions**.

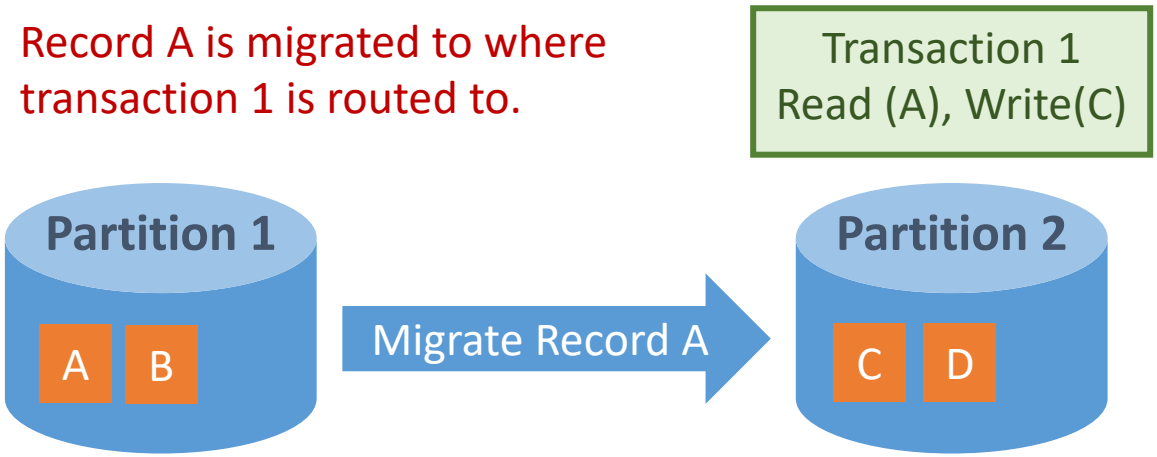


Things Gets Worse on Dynamic Workloads

- **Dynamic workloads** makes the best data partitions constantly change.
- In order to fit the latest workload, a DBMS may have to **keep re-partitioning** its database on-line.
- However, the regular way of re-partitioning a database is too costly.

Fast Data Re-partitioning by Transaction Routing

- Key Technique: transaction routing + data fusion
  - We can guide data flowing to where we want **by simply routing transactions** to proper locations.



How to Know Which Routing Plan is Best for **the Future**?

Insight: Transaction Batches = Future Workloads!

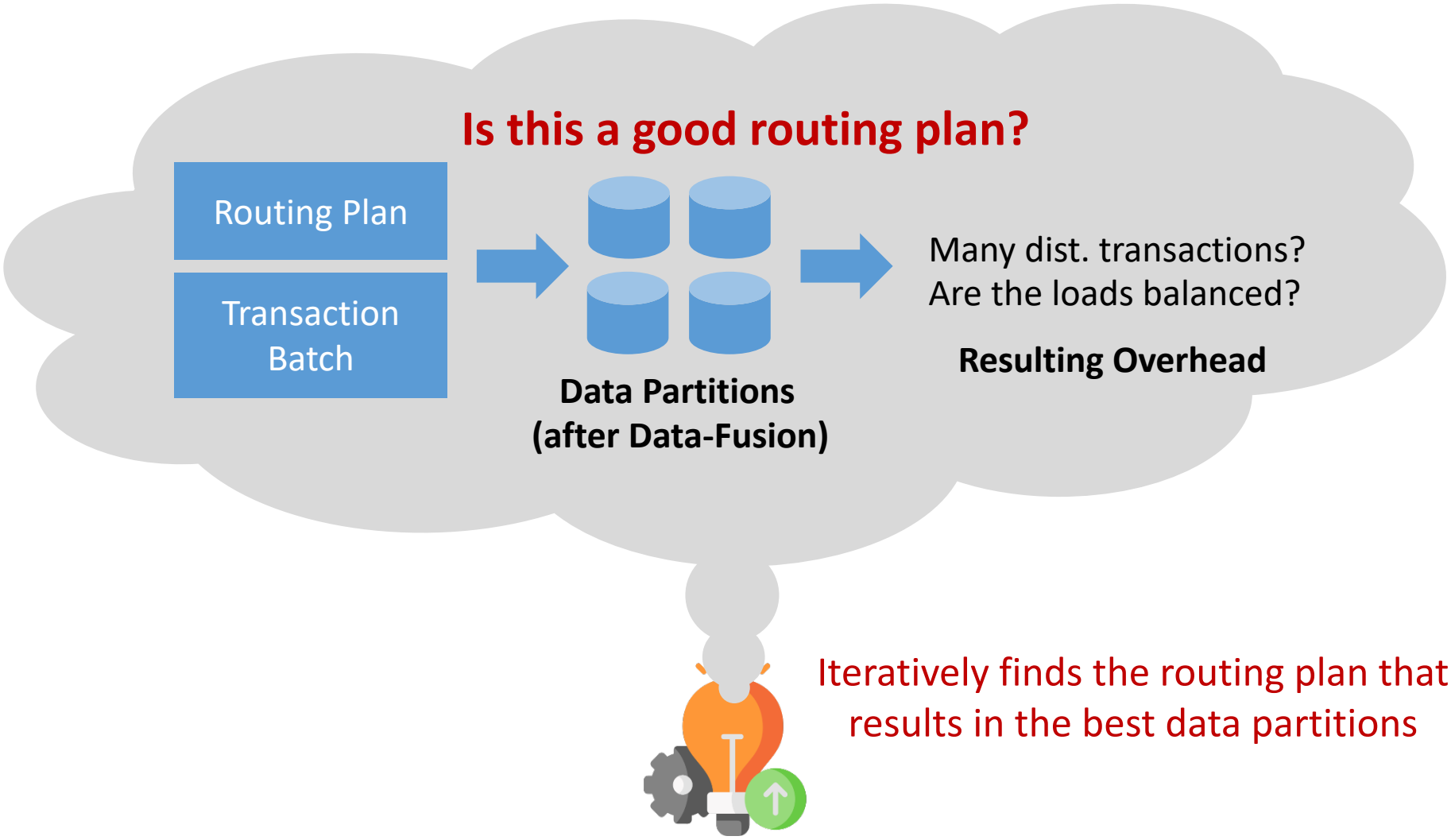
- A deterministic database system usually **batches transactions** when ordering transactions to reduce communication cost.

- We can leverage on these batches to analyze the future workload in advance!

A Transaction Batch

T1: Read {C}, Write {C}
T2: Read {C}, Write {C}
T3: Read {D}, Write {D}
T4: Read {D}, Write {D}
T5: Read {A, B, E}, Write {A, E}
T6: Read {A, B, E}, Write {A, E}

Prescient Transaction Routing



For More Details?
Check the Paper!

Outline

- Project Goal
- Introduction to Hermes
- Target Workloads
- Where Can I Optimize?
- Stages
- Final Presentation
- Timeline

Target Workloads

- We prepare the following workloads for testing:
  - The Hotspot Workload
  - The Google Workload
  - The Hot Counter Workload

- All these workloads uses the database and transactions defined in YCSB.

Yahoo! Cloud Serving Benchmark

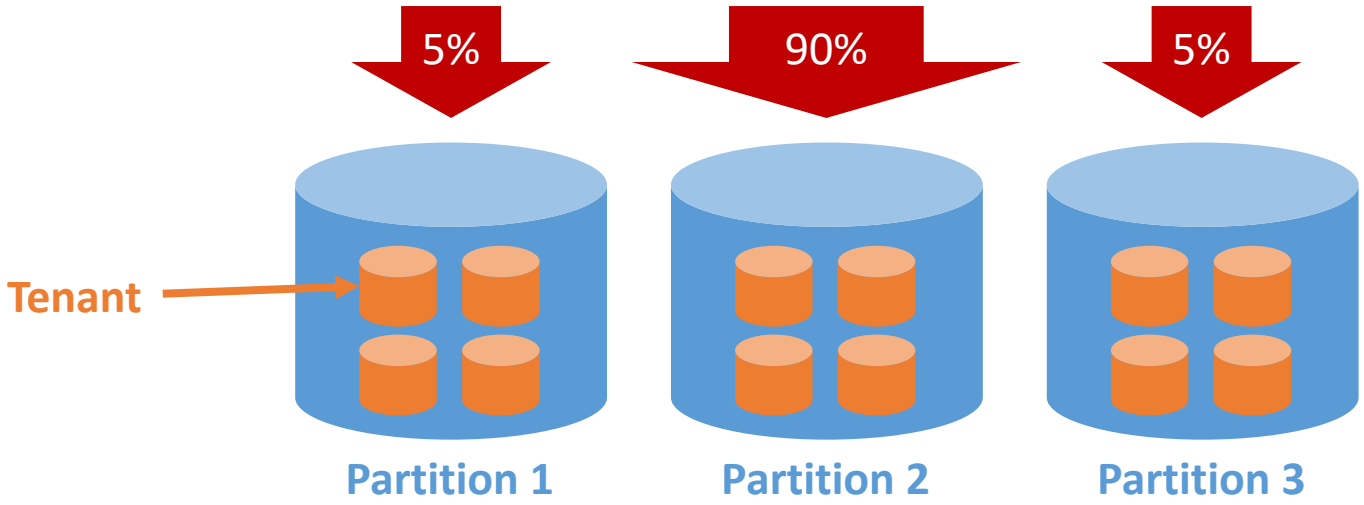
- Yahoo! Cloud Serving Benchmark (YCSB) is a very flexible industry benchmark proposed by yahoo!.

- The YCSB Database:
  - One table.
  - Each primary key is a string.
  - Each record has 10 fields (including the primary key).
  - Each field is approximately 100 bytes.
  - # of records are UNDEFINED.

- A YCSB transaction typically accesses 2 records.
  - How to select these records are UNDEFINED.

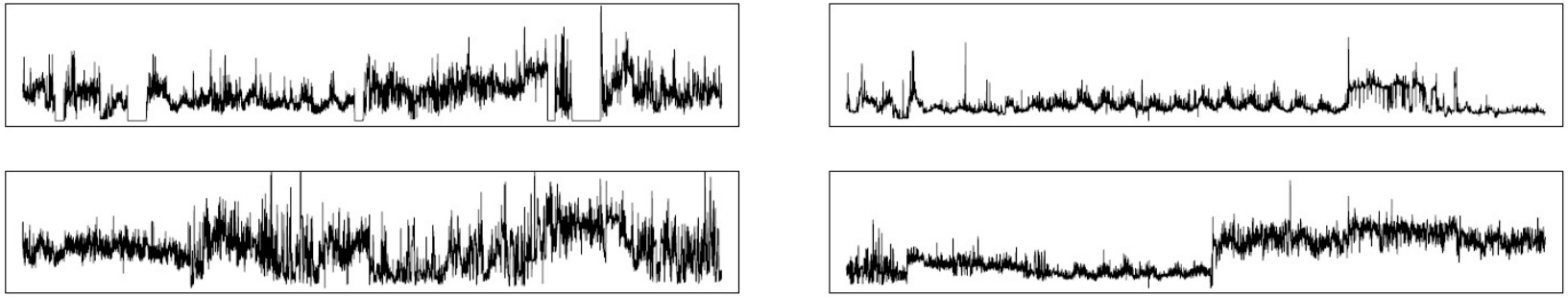
Workload #1: The Hotspot Workload

- Multi-tenant Workloads
  - Each partition is further divided to 4 tenants.
  - Each transaction only accesses the records in the same tenant. (no cross-tenant transactions)
  - One of partitions receive 90% of workloads.



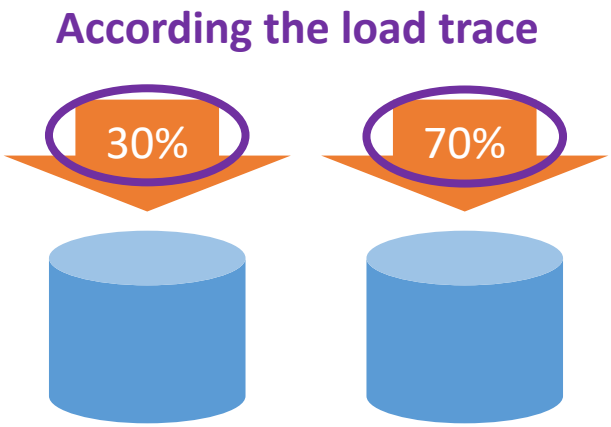
Workload #2: The Google Workload

- The Google workload is composed by replaying the load trace collected by Google.
- This workload has many unpredictable pattern, so it is not easy to summarize the best data partitions for it.

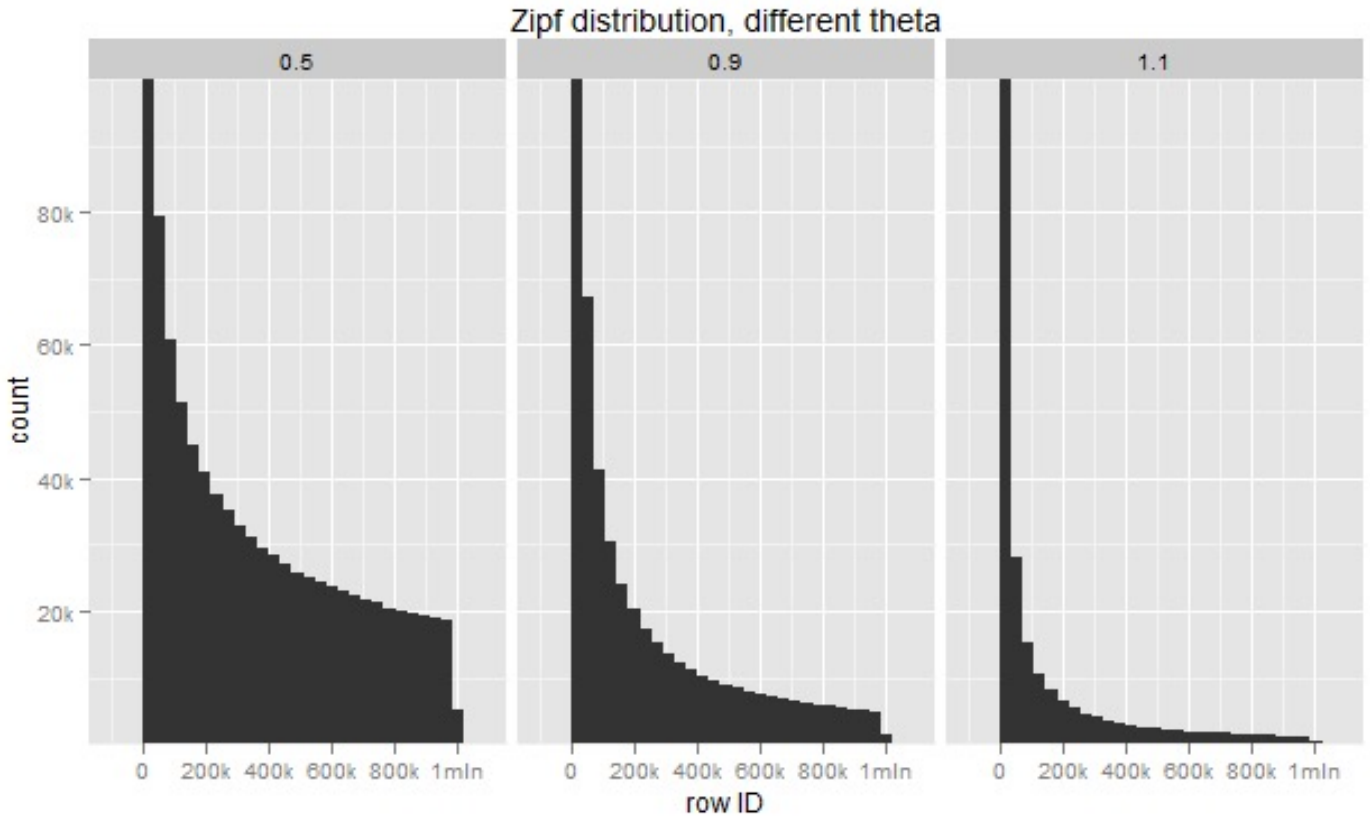


Workload #2: The Google Workload

- Transaction Type 1: Partition-Local Transactions
  1. Choose a partition following a distribution **proportional to the loads** in the load trace at the time.
  2. Choose two records in the partition following a Zipfian distribution ($\theta = 0.99$).



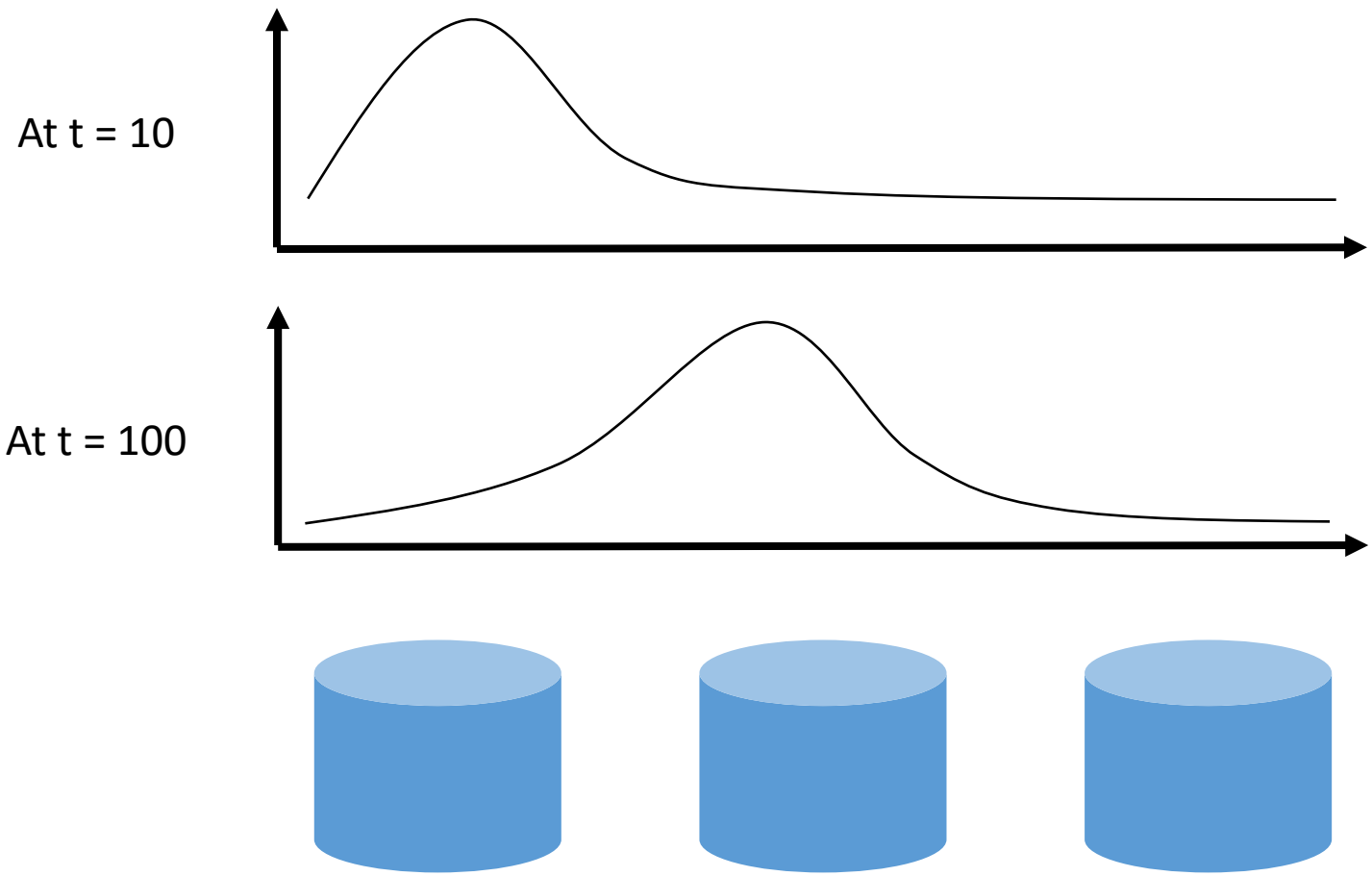
Zipfian Distribution



Workload #2: The Google Workload

- Type 2: Cross-Partition Transactions
  1. Choose a partition following a distribution proportional to the loads in the load trace at the time.
  2. Choose **one** record in the partition following a Zipfian distribution ($\theta = 0.99$).
  3. Choose another record following a global **two-sided Zipfian distribution** across all partitions.

Global Two-Sided Zipfian Distribution

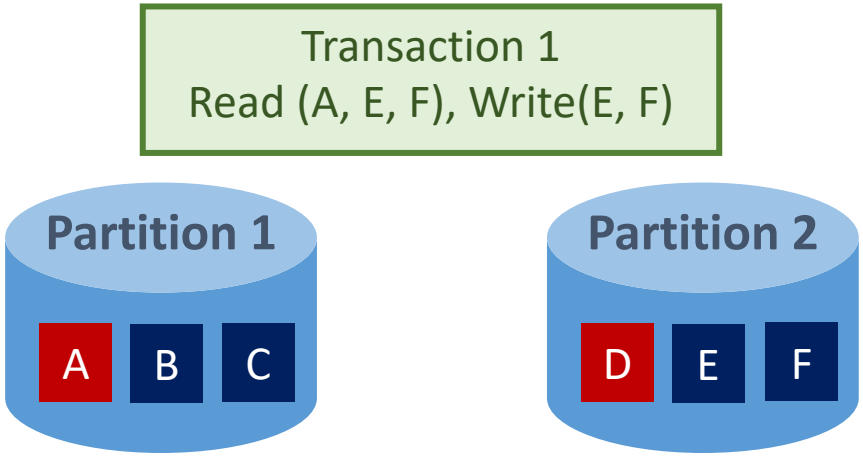


References

- Workload #1 and #2 also appear in Hermes' paper. You can find more details in:
  - Section 5.2.2 (The Google Workload)
  - Section 5.3.2 (The Hotspot Workload)
  - Section B.1 in the Supplementary (The Details of Google Workload)

Workload #3: The Hot Counter Workload

- Each partition has some hot records.
- Each transaction may choose a hot record in a partition and a few cold records in a partition.
  - It may choose all records from the same partition.
- The hot records are **rarely updated**.



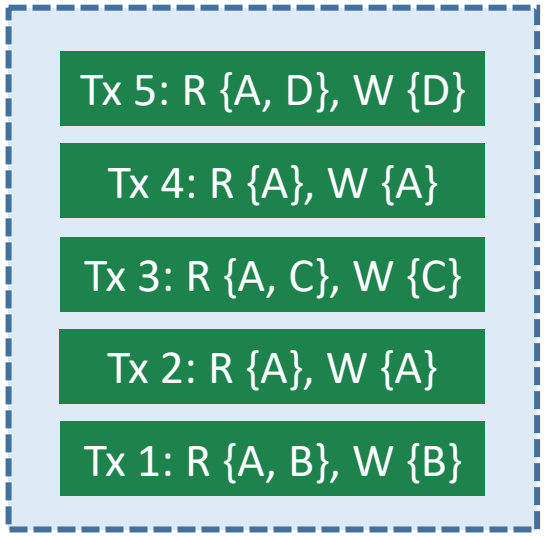
Outline

- Project Goal
- Introduction to Hermes
- Target Workloads
- Where Can I Optimize?
- Stages
- Final Presentation
- Timeline

Chances to Optimize

- Hermes works very well in Workload #1, so there may be little chance to optimize it.
  - However, it is good for verifying your algorithm.
- Hermes works also well in Workload #2, but we believe there should be chance to do better.
- There is also high chance to optimize Hermes in Workload #3.

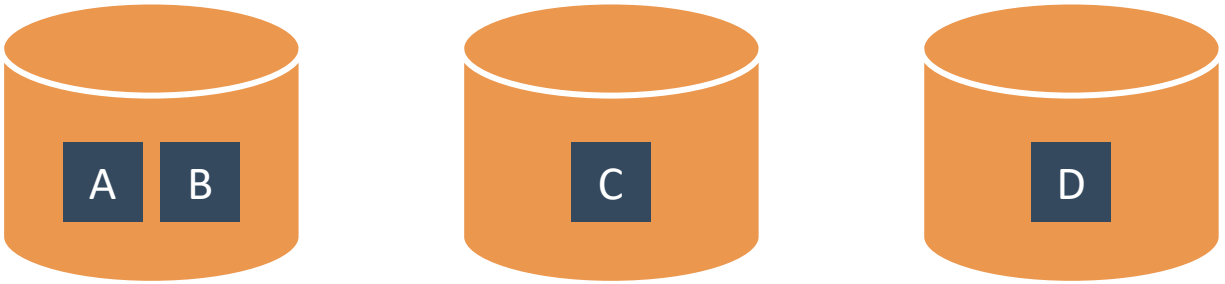
An Example Workload of Workload #3



Transaction Batch

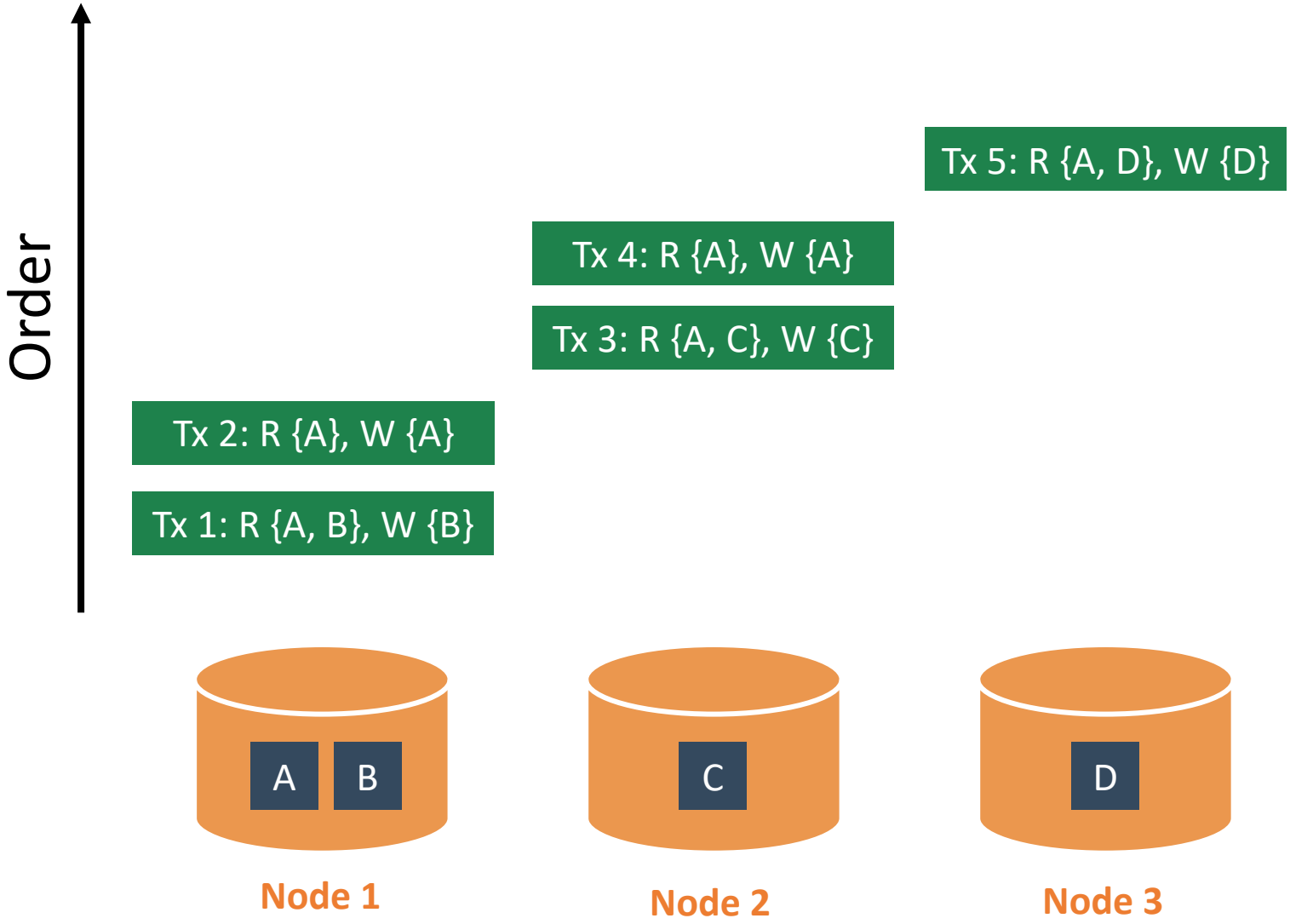
Key Characteristics:

- **A shared hot record**
- **The hot record may be updated occasionally**

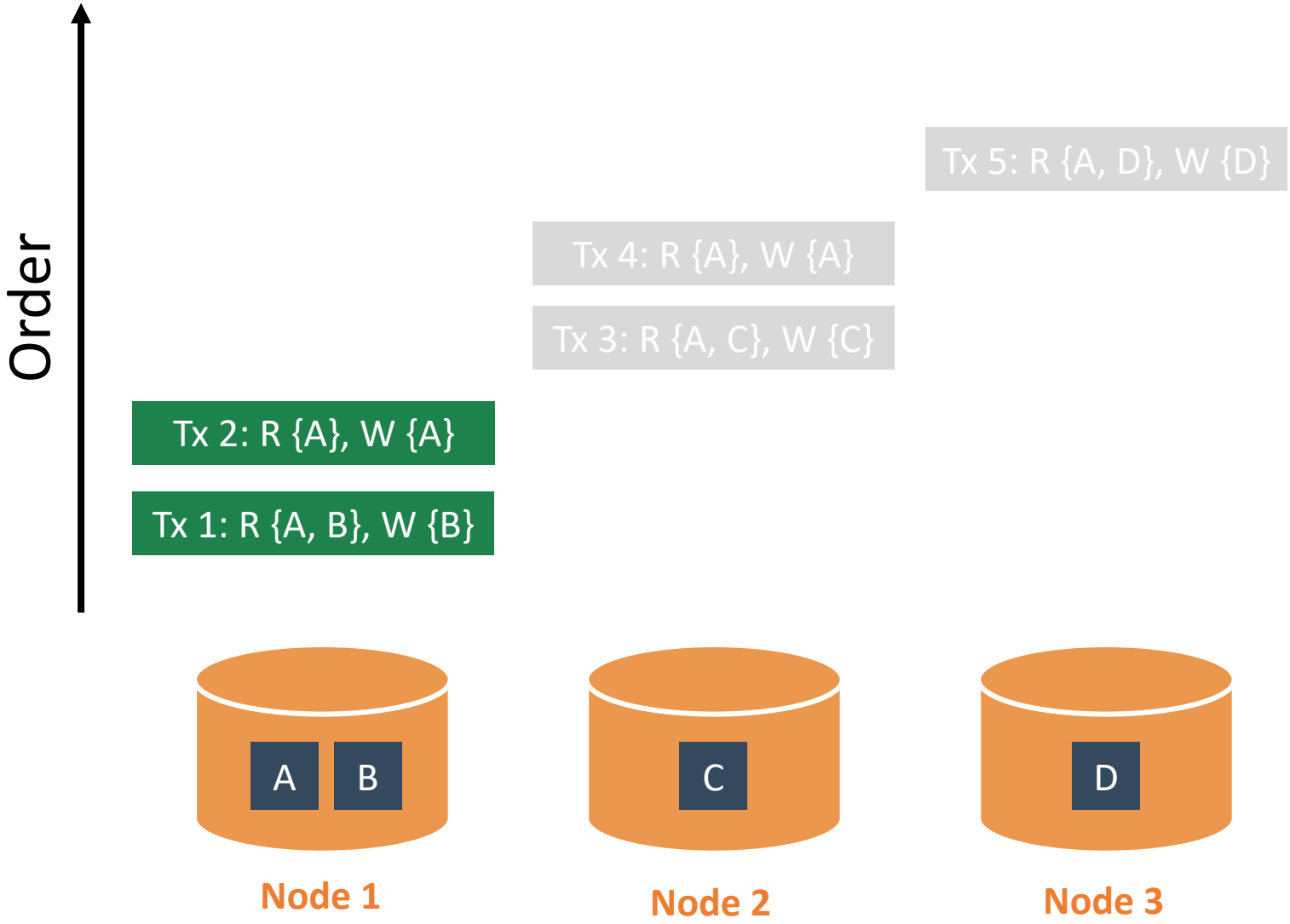


Node 1 Node 2 Node 3

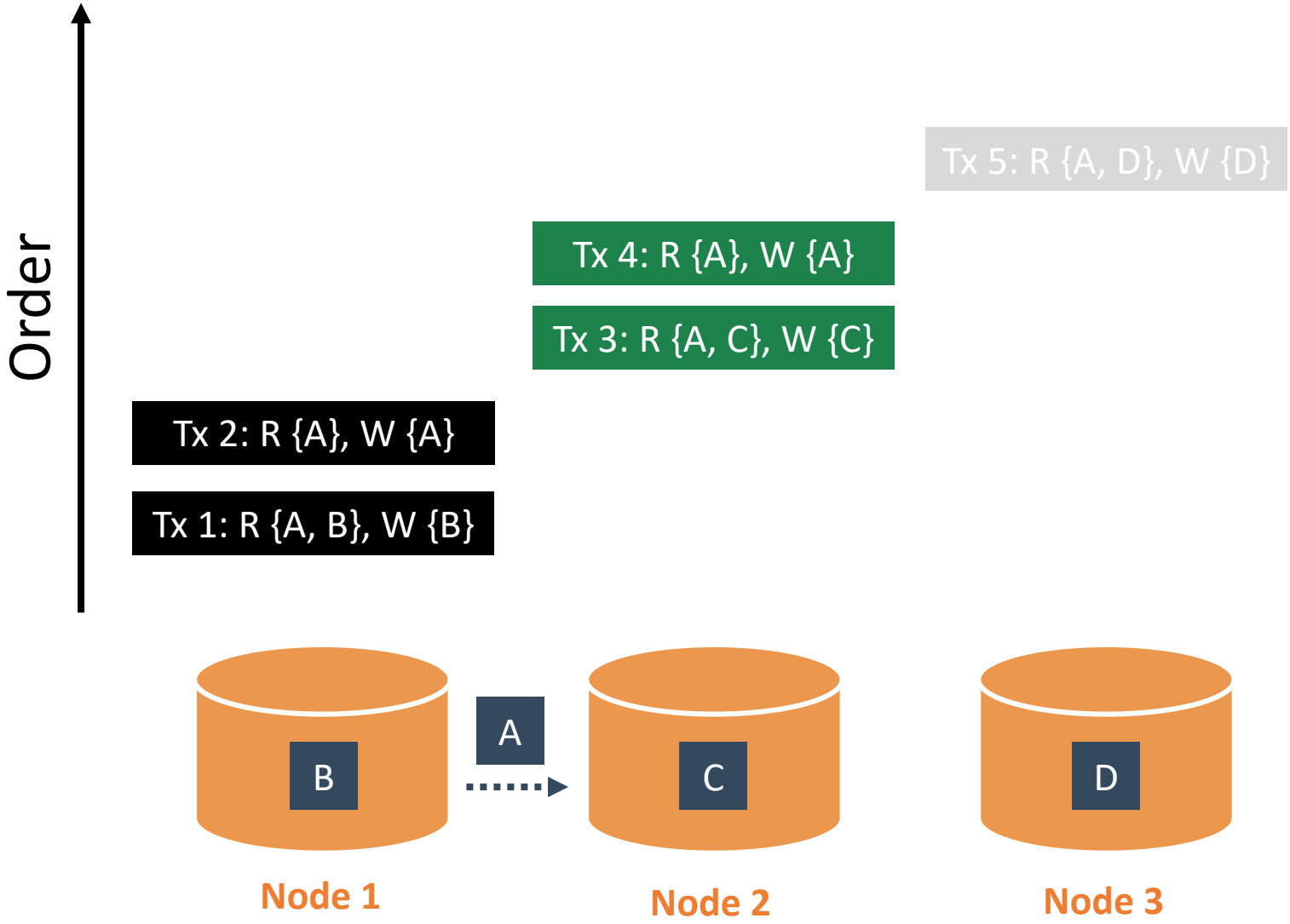
Hermes' Routing



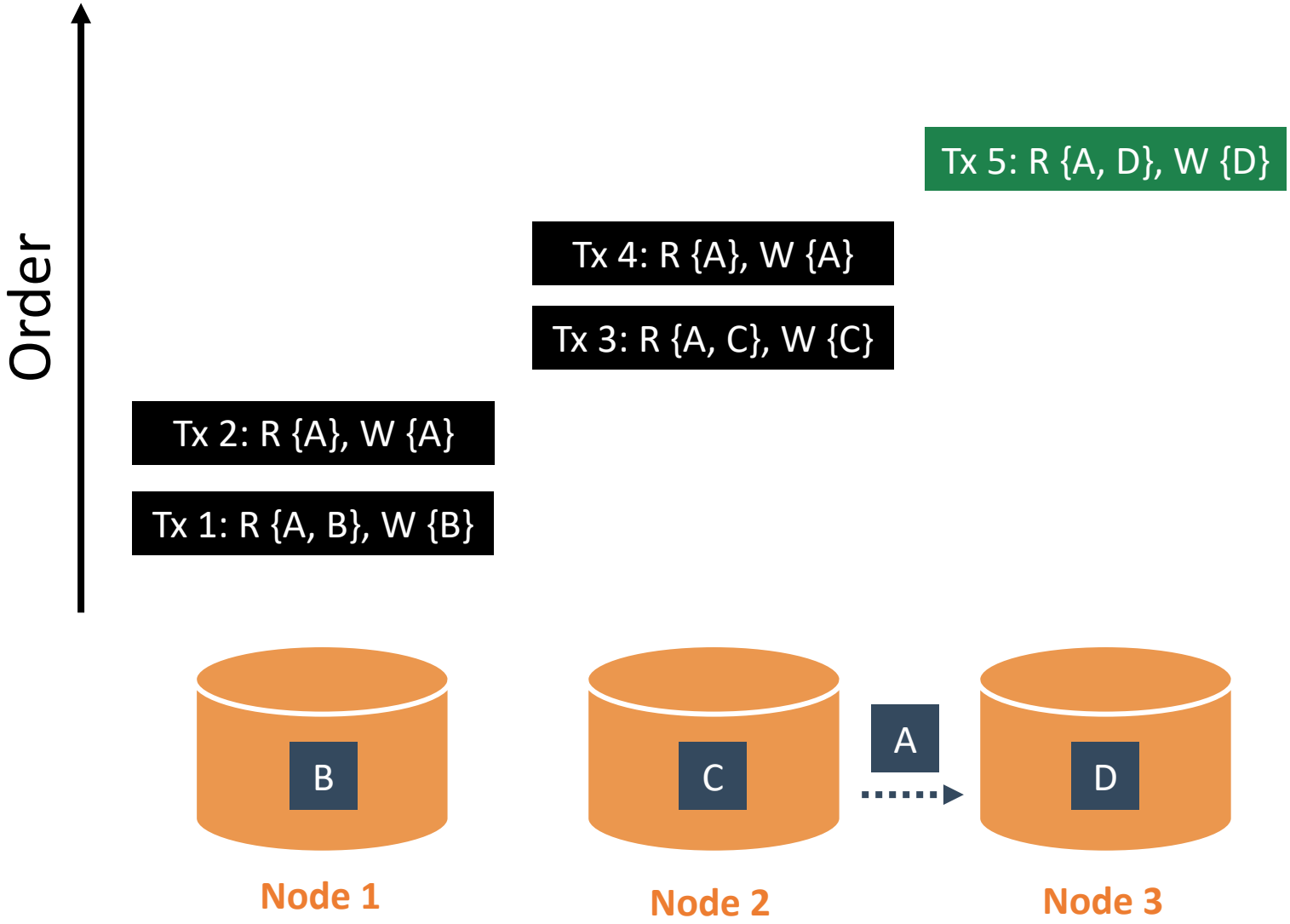
Hermes' Routing



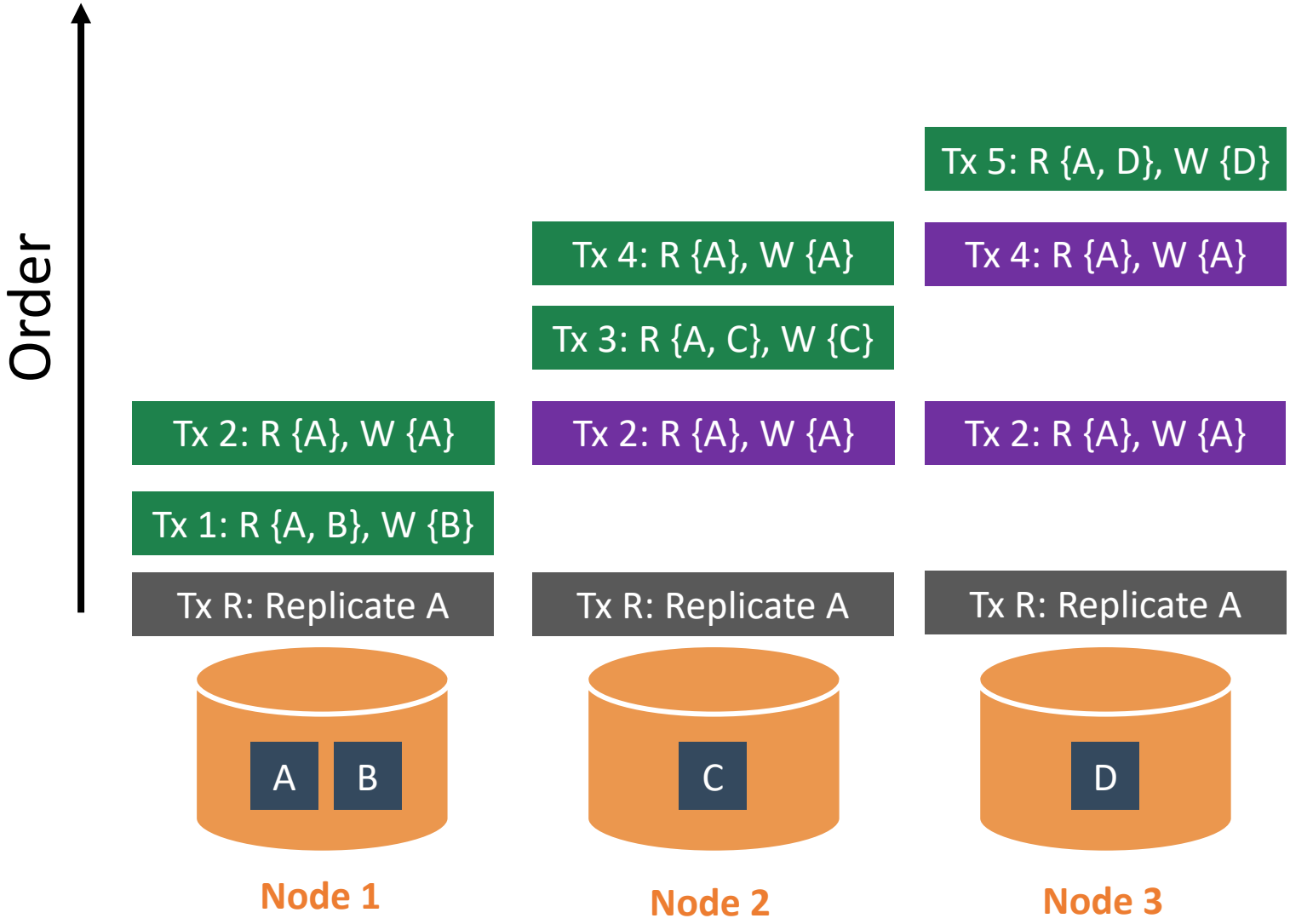
Hermes' Routing



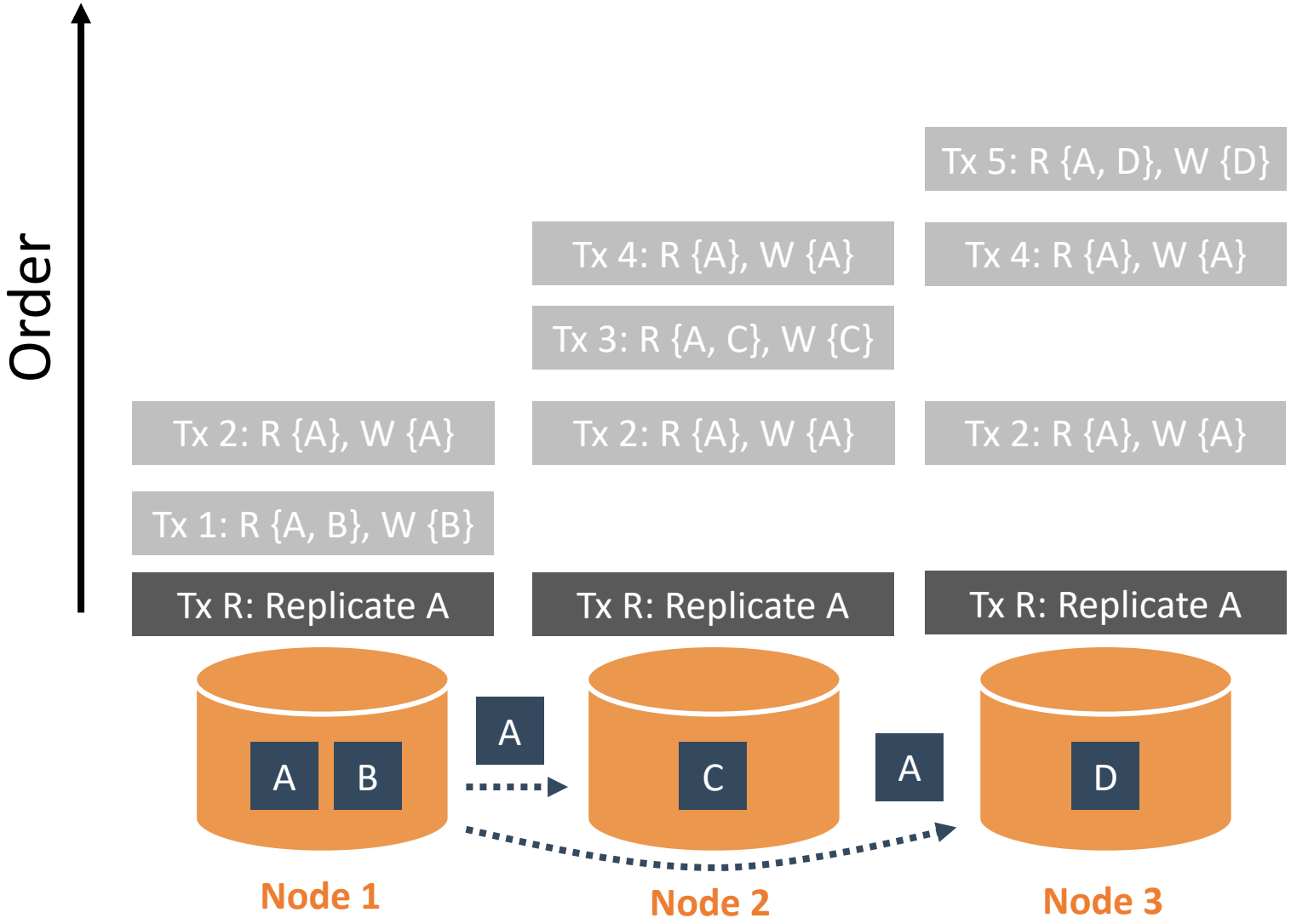
Hermes' Routing



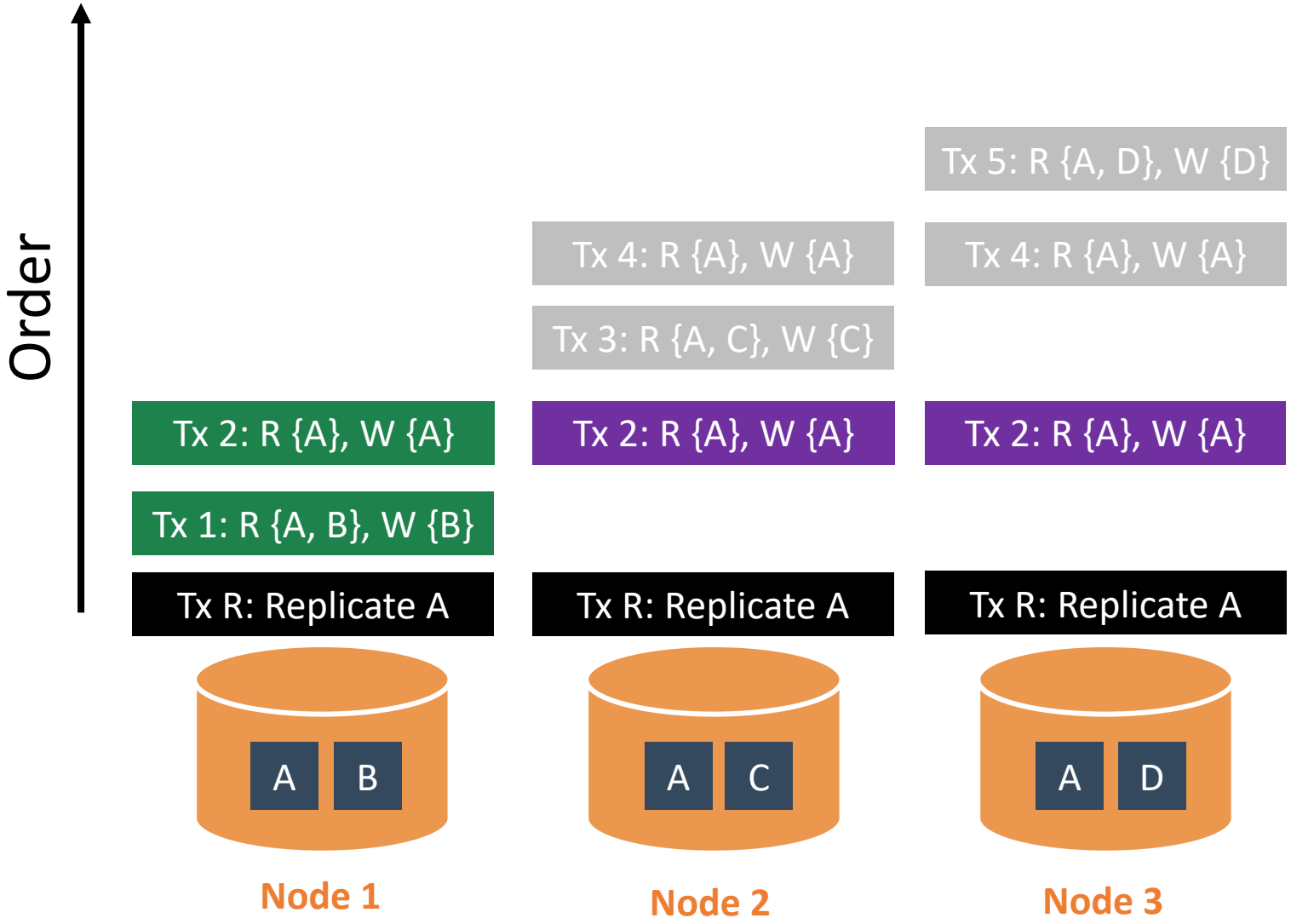
With Replication



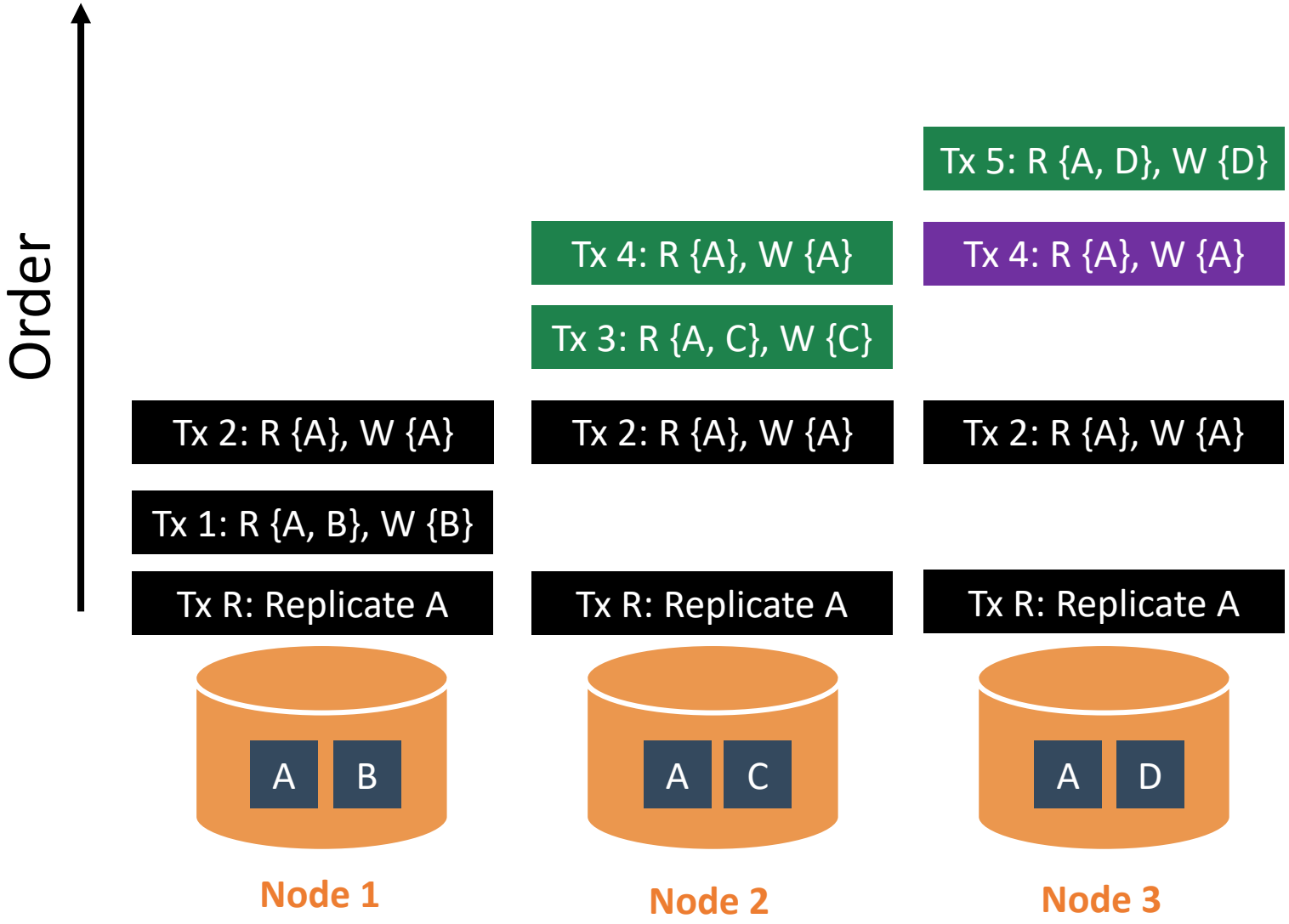
With Replication



With Replication



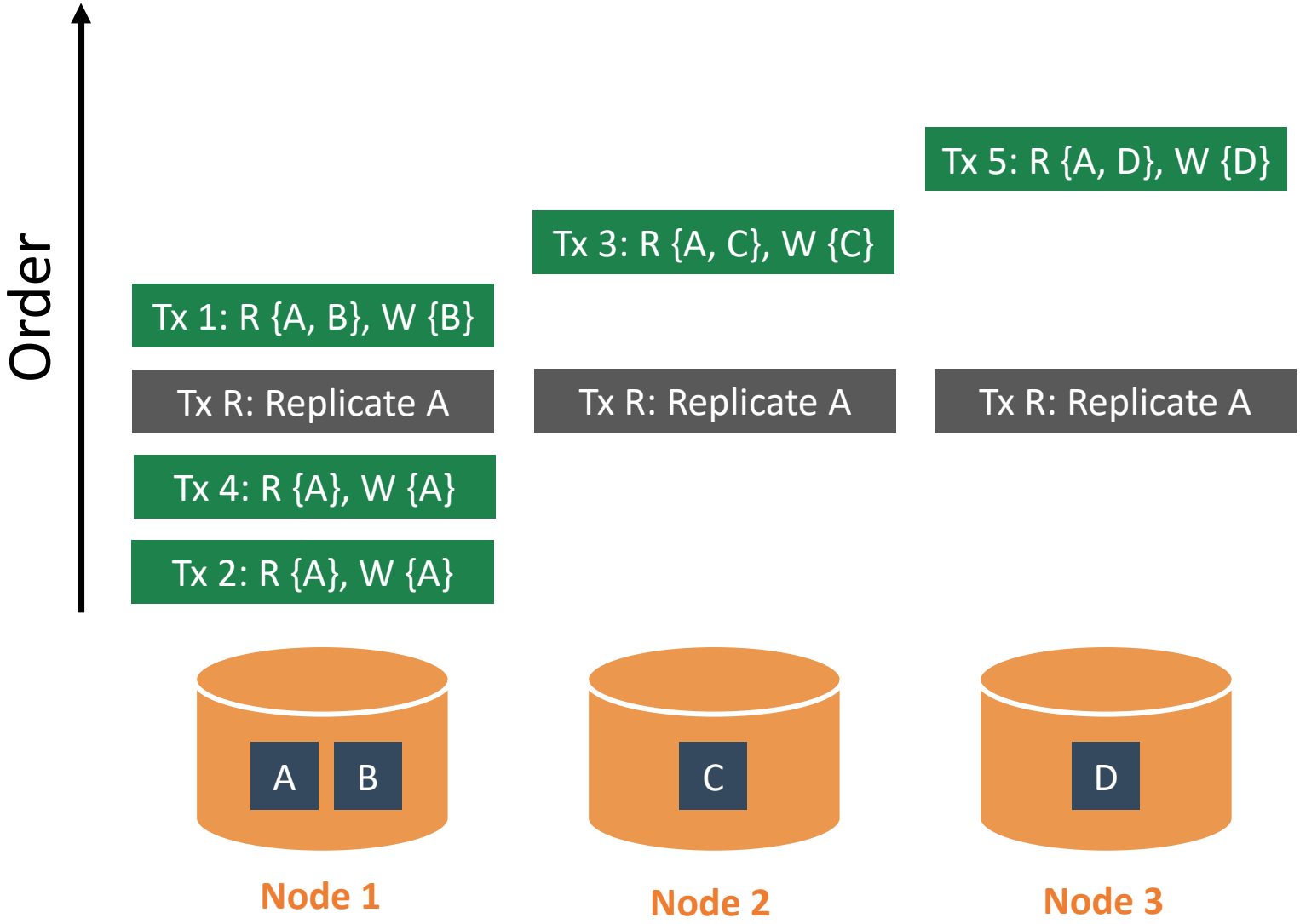
With Replication



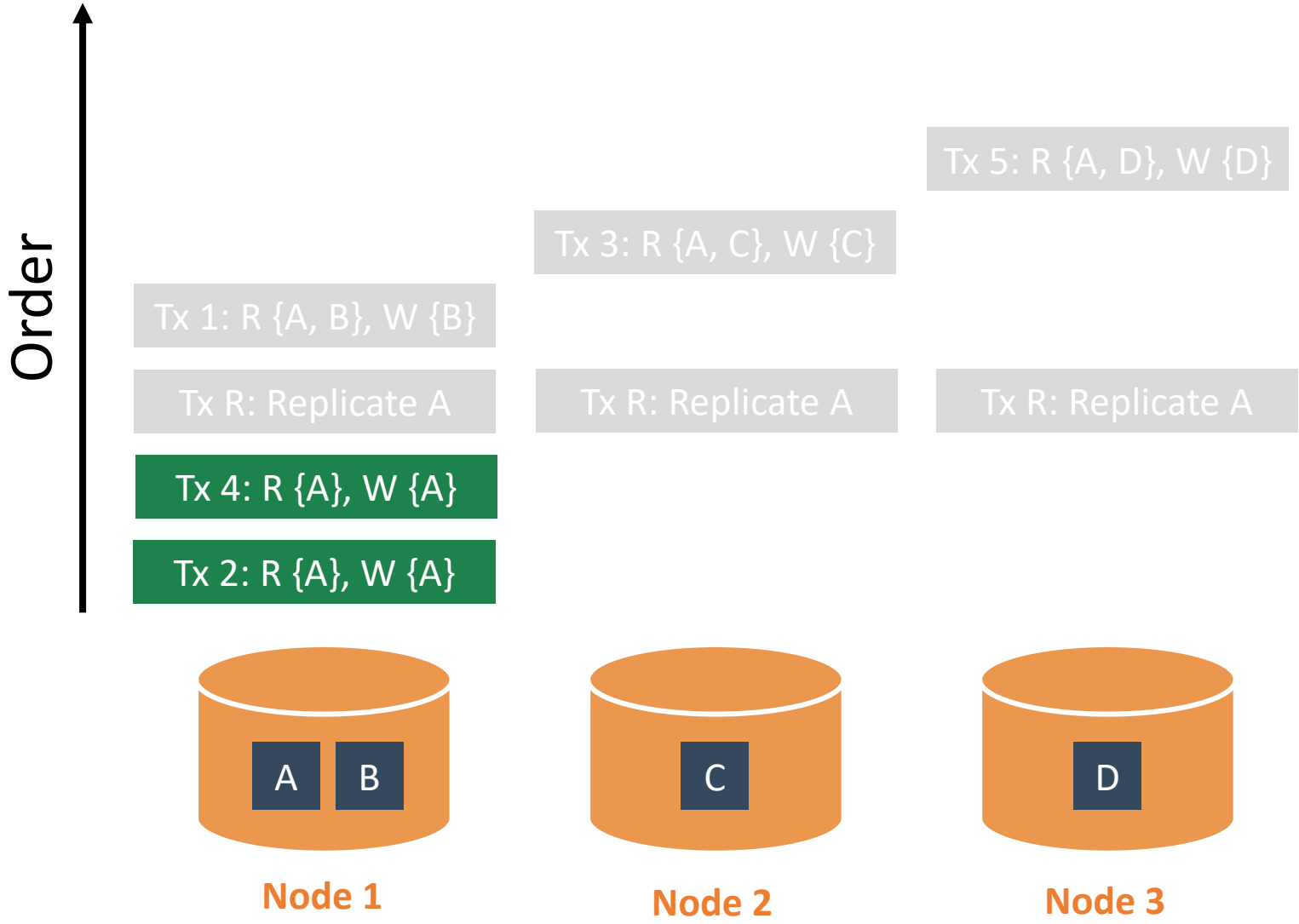
Comparison with the Original Design

- Record A only needs to be migrated **once**.
  - It was migrated twice in the original design.
- Slow machines **will not block** other machines.
  - Because all the transactions become local transactions.
- Con: needs additional computing power.

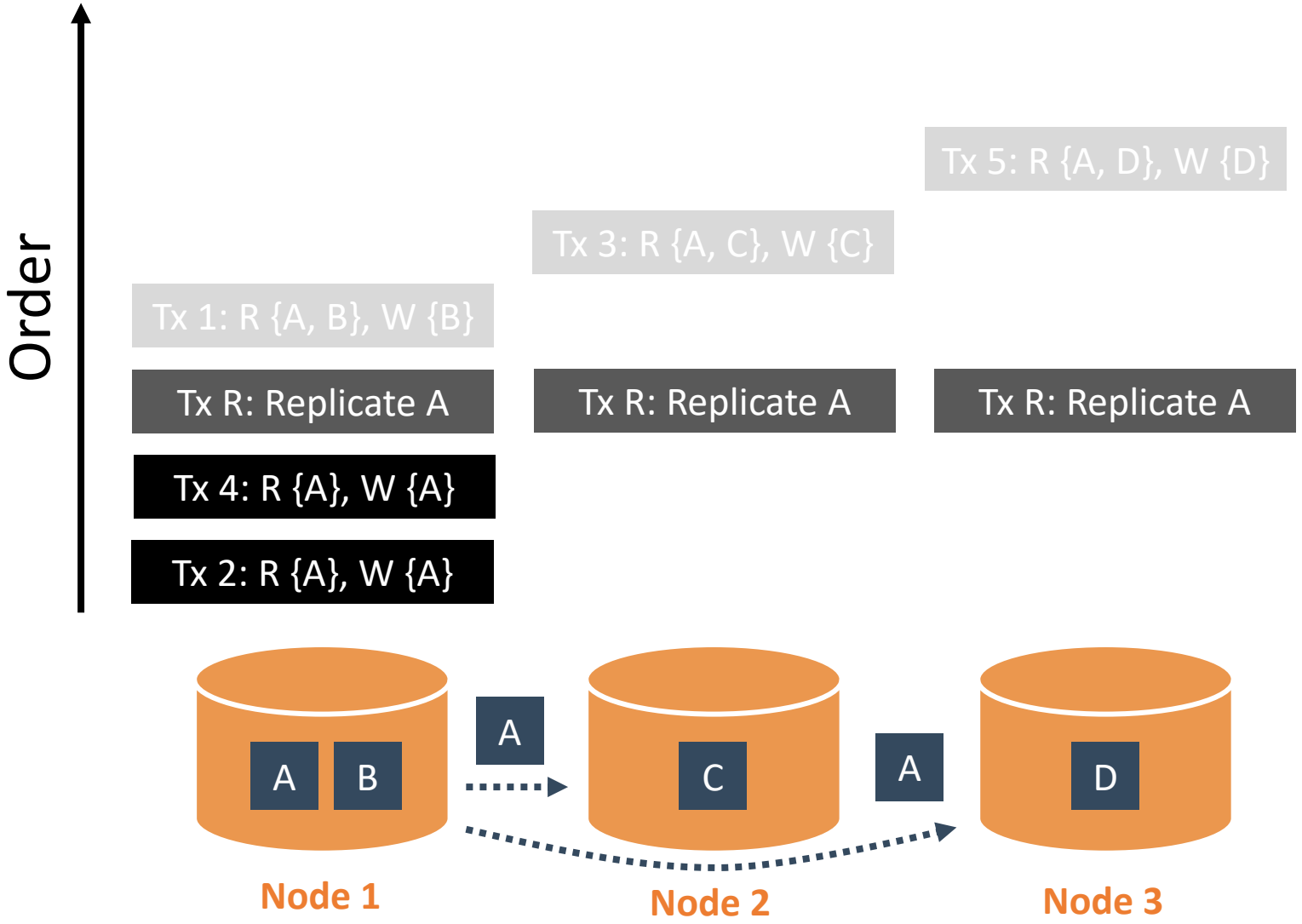
With Replication & Reordering



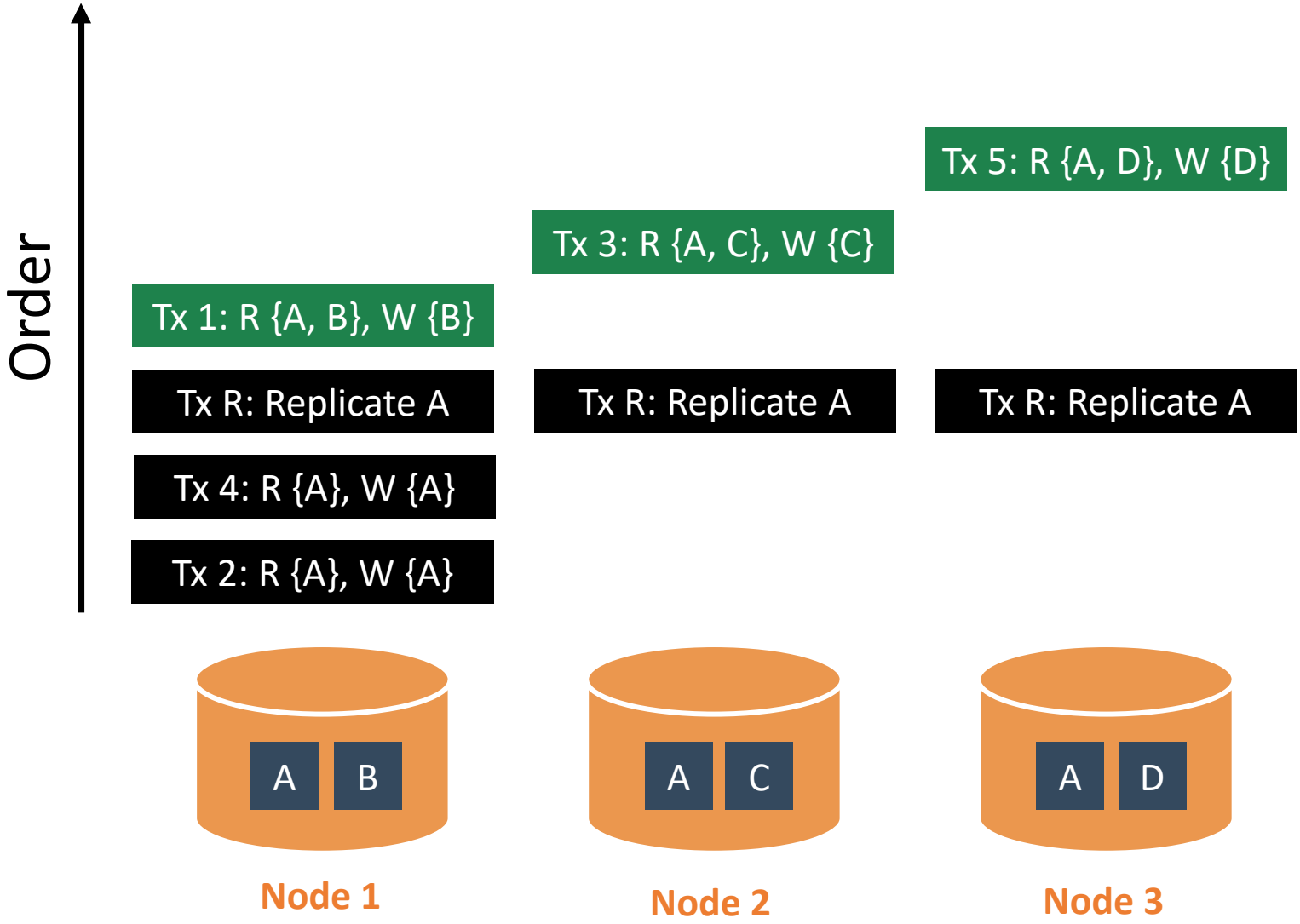
With Replication & Reordering



With Replication & Reordering



With Replication & Reordering



Benefits

- Record A only needs to be migrated once.
  - It was migrated twice in the original design.
- Some slow machines will not block other machines.
  - Such as node 2, but not node 1.
- **No need for additional computing power!**

Outline

- Project Goal
- Introduction to Hermes
- Target Workloads
- Where Can I Optimize?
- Stages
- Final Presentation
- Timeline

Stages

- Stage 1: Read the papers
  - To learn the essential knowledge for accomplishing this project.
- Stage 2: Trace the Code of Hermes
  - To understand the logic and architecture in order to modify the code
- Stage 3: Improve Hermes and Test
  - To improve the algorithm of Hermes in order to get better performance.

Assigned Readings for Stage 1

- VLDB'10 - The Case for Determinism in Database Systems
  - To get the main idea of deterministic database systems.
- SIGMOD'12 - Calvin: Fast Distributed Transactions for Partitioned Database Systems
  - To know how to implement a deterministic database system.
- SIGMOD'21 - Don't Look Back, Look into the Future: Prescient Data Partitioning and Migration for Deterministic Database Systems
  - The target system we need to improve in this project.

Outline

- Project Goal
- Introduction to Hermes
- Target Workloads
- Where Can I Optimize?
- Stages
- Final Presentation
- Timeline

Final Presentation

- To present how you optimize Hermes and results
  - Insight & Idea
  - Experiments
  - Analysis
- Date: 2021/6/21、2021/6/24
  - If you need to get your score by then, please contact TAs.
- More details will be announced later.

Timeline

5/20 Announce the assigned reading

5/27 Announce the details of the Final Project

5/31 Introduction to ElaSQL and ElaSQLBench

6/7 Walkthrough the Codebase of ElaSQL & Hermes

6/21、6/24 Final Project Presentation


Good Luck !