Sonata: Multi-Database Transactions Made Fast and Serializable

Chuzhe Tang[1], Zhaoguo Wang[1], Jinyang Li[2], Haibo Chen[1]

[1] Institute of Parallel and Distributed Systems, Shanghai Jiao Tong University

[2] New York University

Transaction Abstraction

- **Transactions are an important programming abstraction.**
  - Ensures correctness in the presence of concurrent operation and failures.

Large-Scale Applications

ebay Booking.com amazon HSBC CHASE

Safeguarding Critical Business Logic

Transactional Databases

PolarDB Google Cloud Spanner Amazon Aurora PostgreSQL Microsoft SQL Server

Transaction Processing

Transaction Logic

Checking $50

Savings $50

T1: withdraw(Checking, $100)

```
Begin Transaction
total := s_bal + c_bal
if total >= 100:
  c_bal -= 100
Commit Transaction
```

T2: withdraw(Savings, $100)

```
Begin Transaction
total := c_bal + s_bal
if total >= 100:
  s_bal -= 100
Commit Transaction
```

Transaction Processing

Transaction Logic → **Submit** → **Transactional Database**

Checking: $50 Savings: $50

T1: withdraw(Checking, $100)

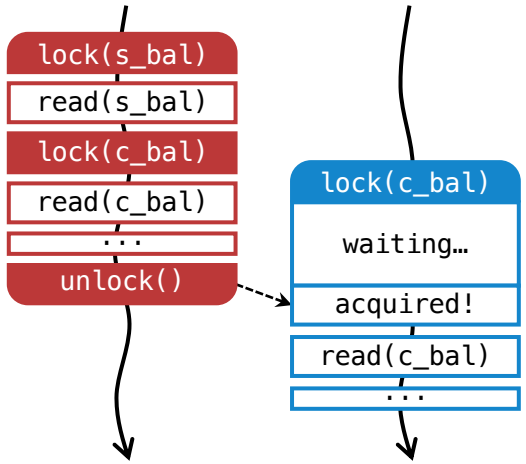
```
Begin Transaction
total := s_bal + c_bal
if total >= 100:
  c_bal -= 100
Commit Transaction
```

T2: withdraw(Savings, $100)

```
Begin Transaction
total := c_bal + s_bal
if total >= 100:
  s_bal -= 100
Commit Transaction
```

Concurrent Execution

(e.g., via two-phase locking)



Transaction Processing

Transaction Logic → Submit → **Transactional Database** → Return → **Correct Result**

Checking	Savings
$50	$50

T1: withdraw(Checking, $100)

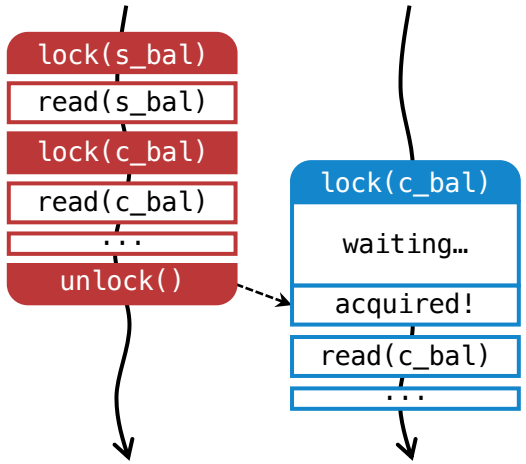
```
Begin Transaction
total := s_bal + c_bal
if total >= 100:
  c_bal -= 100
Commit Transaction
```

T2: withdraw(Savings, $100)

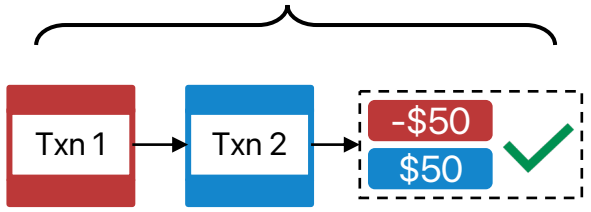
```
Begin Transaction
total := c_bal + s_bal
if total >= 100:
  s_bal -= 100
Commit Transaction
```

Concurrent Execution

(e.g., via two-phase locking)



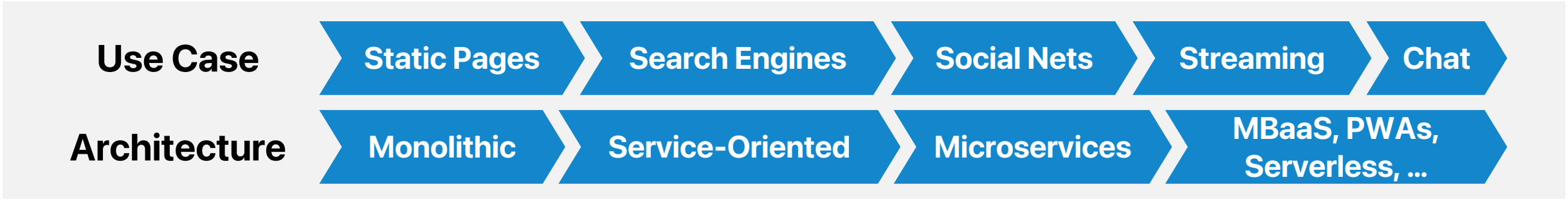
ACID Properties



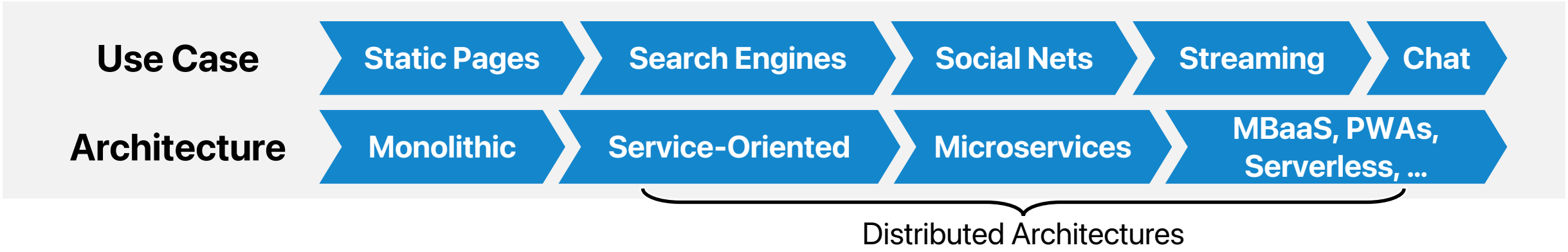
Isolation: Serializable Execution

Atomicity Consistency Durability

Today: Evolving Applications



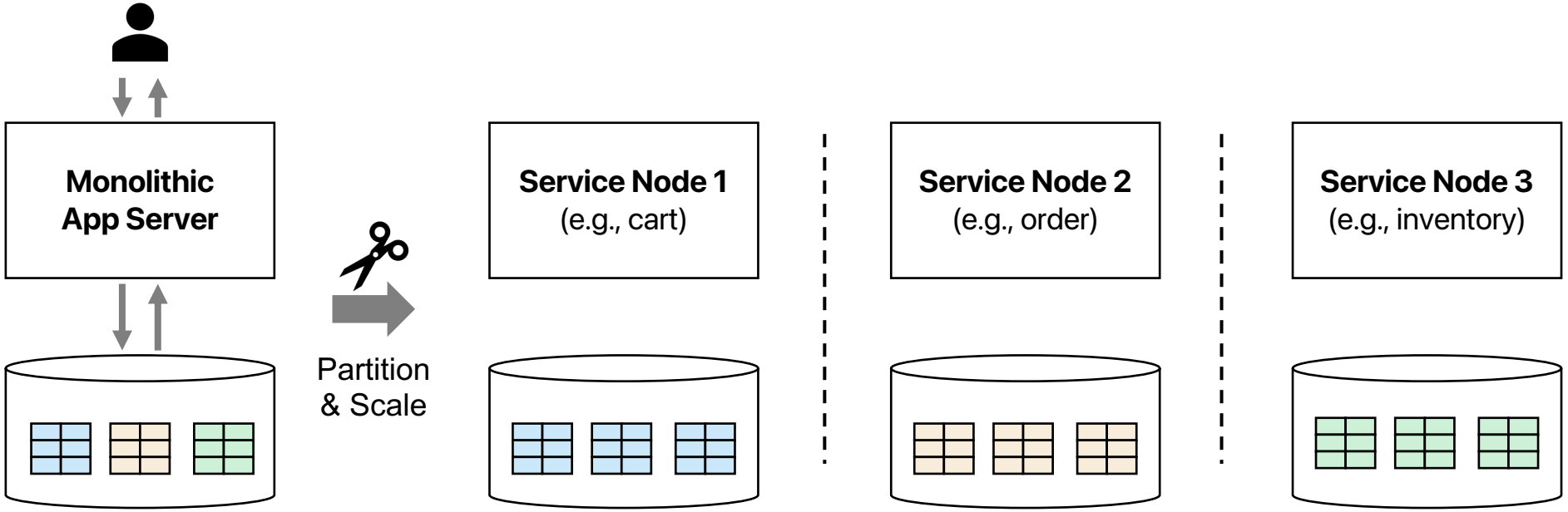
Today: Evolving Applications



Today: Evolving Applications

Use Case	Static Pages	Search Engines	Social Nets	Streaming	Chat
Architecture	Monolithic	Service-Oriented	Microservices	MBaaS, PWAs, Serverless, …	

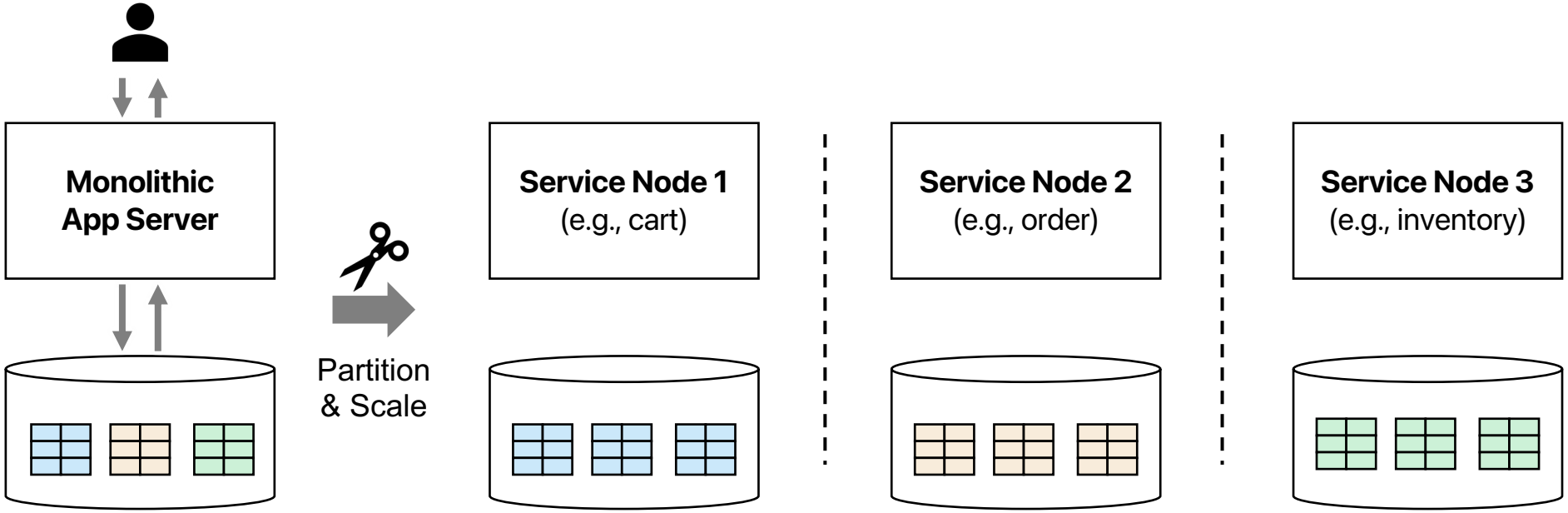
Distributed Architectures



Today: Evolving Applications

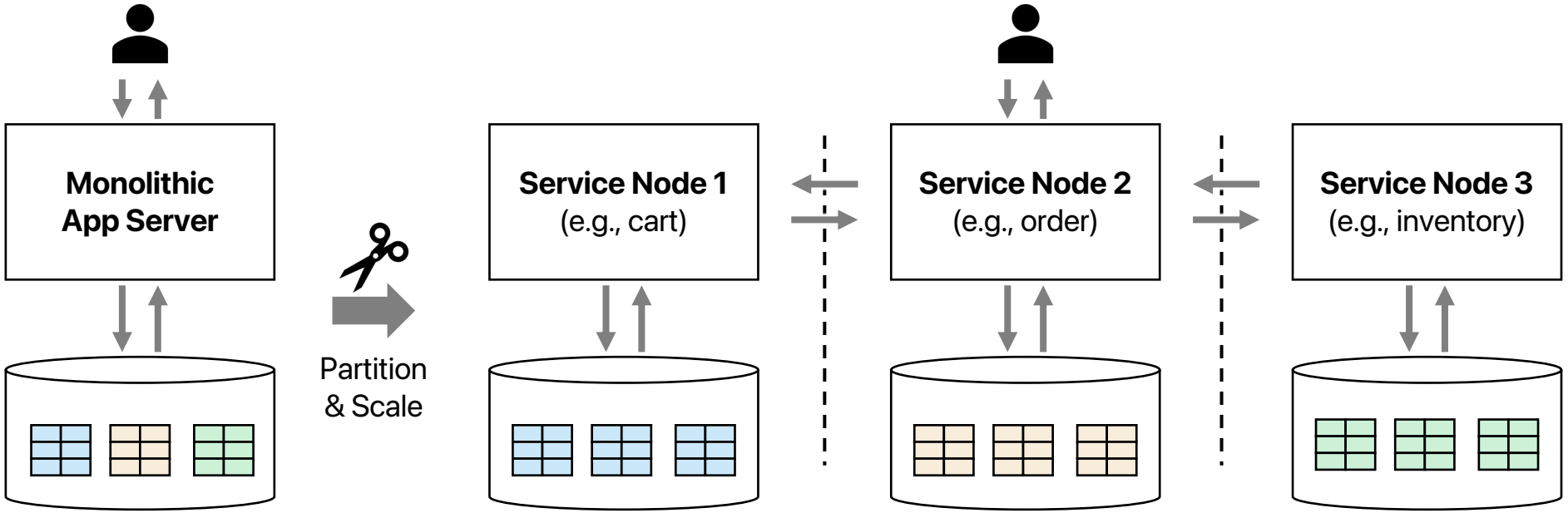
Use Case	Static Pages	Search Engines	Social Nets	Streaming	Chat
Architecture	Monolithic	Service-Oriented	Microservices	MBaaS, PWAs, Serverless, ...	

Distributed Architectures

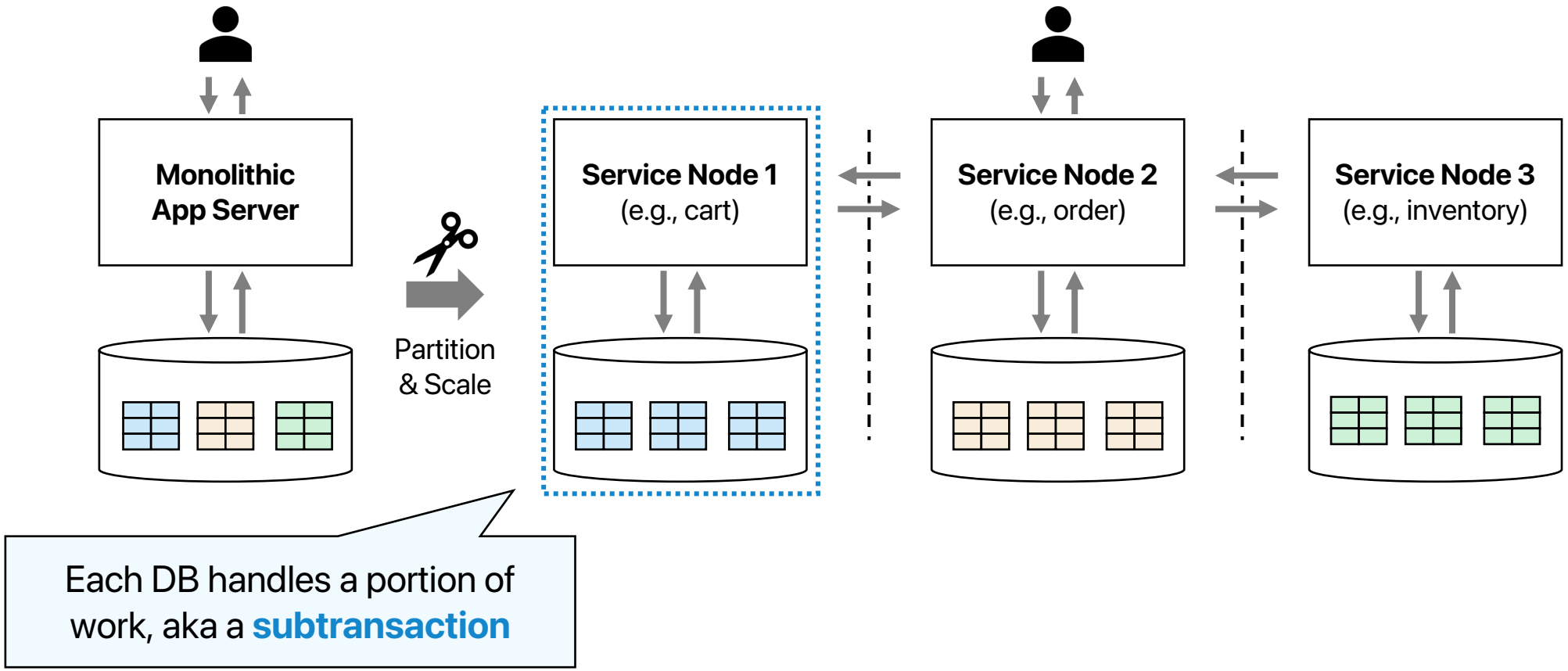


Better scalability in applications' development, deployment, and maintenance

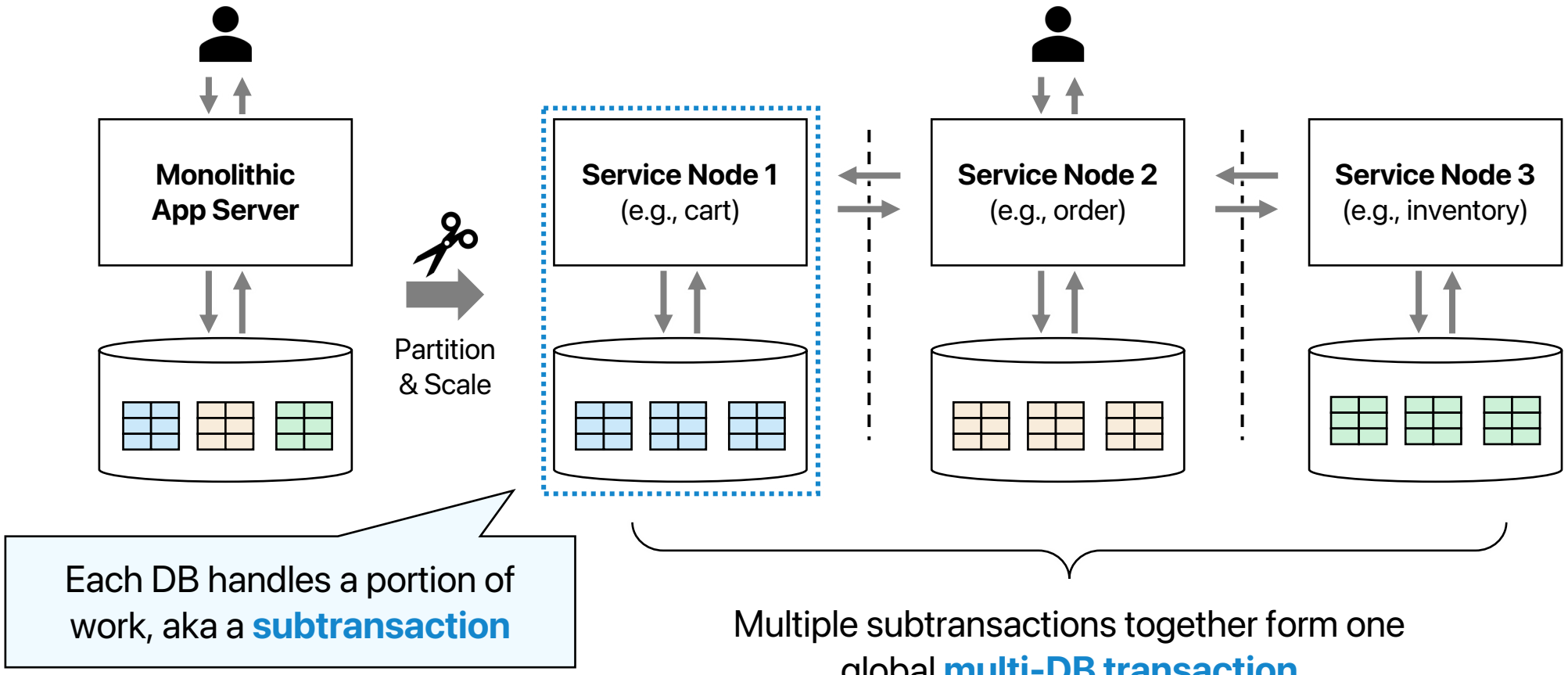
Today: Multi-Database Transactions



Today: Multi-Database Transactions



Today: Multi-Database Transactions



Goal: Global Serializability

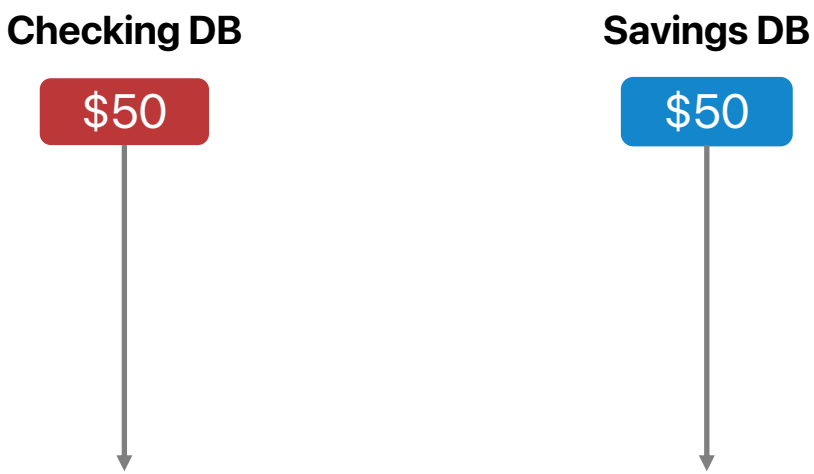
- **Global serializability means serializable multi-DB transactions.**
  - We use "global" to distinguish from the serializability of individual local DBs.

Goal: Global Serializability

- **Global serializability means serializable multi-DB transactions.**
  - We use "global" to distinguish from the serializability of individual local DBs.
- **While helpful for correctness, it is non-trivial to realize.**

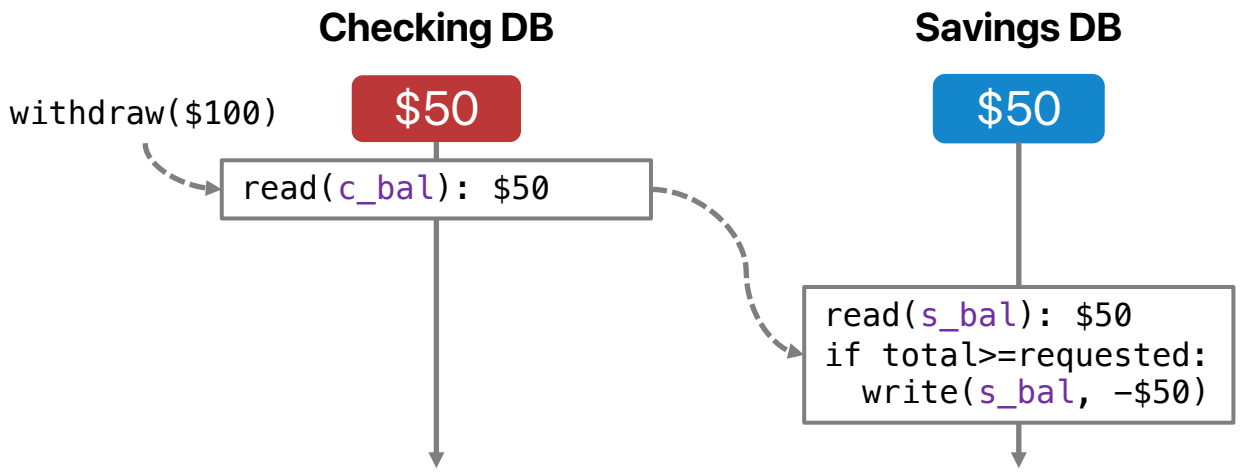
Goal: Global Serializability

- **Global serializability means serializable multi-DB transactions.**
  - We use "global" to distinguish from the serializability of individual local DBs.
- **While helpful for correctness, it is non-trivial to realize.**
  - Even if all DBs are locally serializable, global serializability can still be violated.



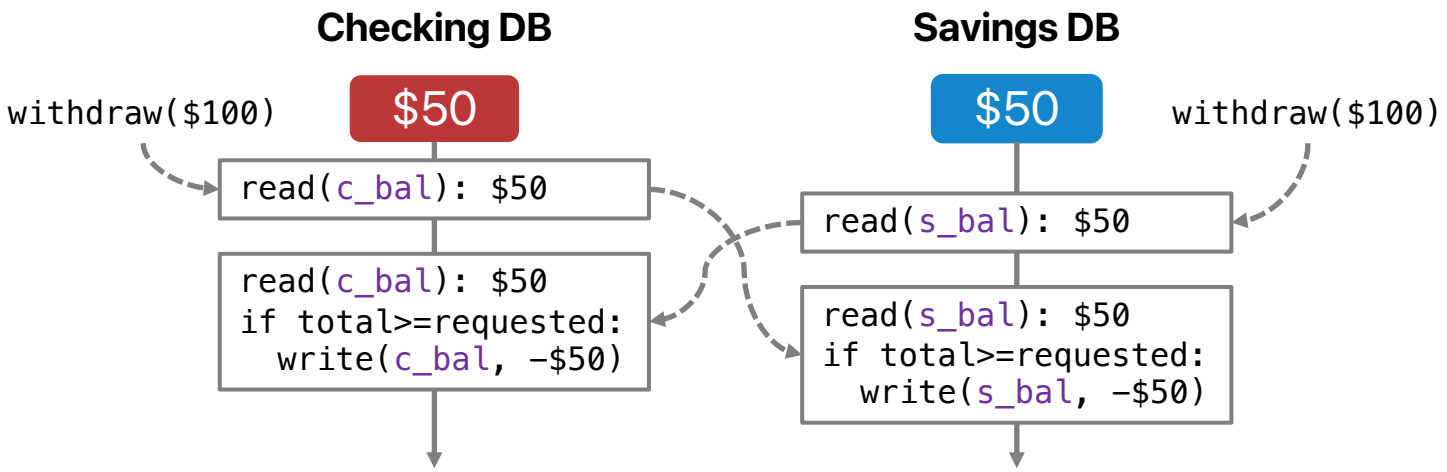
Goal: Global Serializability

- **Global serializability means serializable multi-DB transactions.**
  - We use "global" to distinguish from the serializability of individual local DBs.
- **While helpful for correctness, it is non-trivial to realize.**
  - Even if all DBs are locally serializable, global serializability can still be violated.



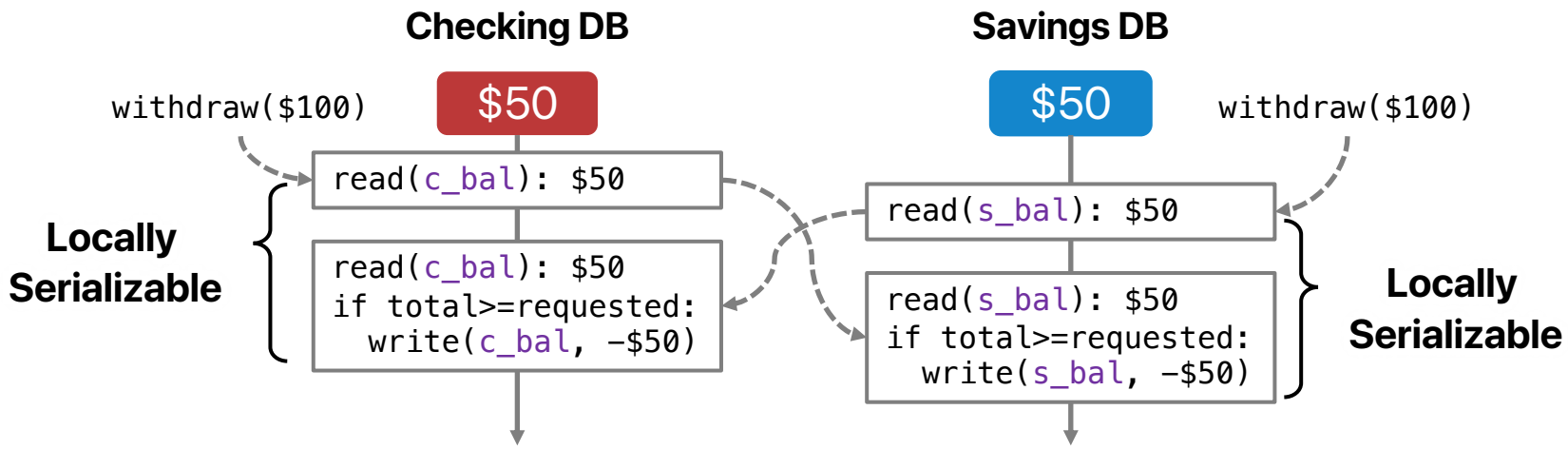
Goal: Global Serializability

- **Global serializability means serializable multi-DB transactions.**
  - We use "global" to distinguish from the serializability of individual local DBs.
- **While helpful for correctness, it is non-trivial to realize.**
  - Even if all DBs are locally serializable, global serializability can still be violated.



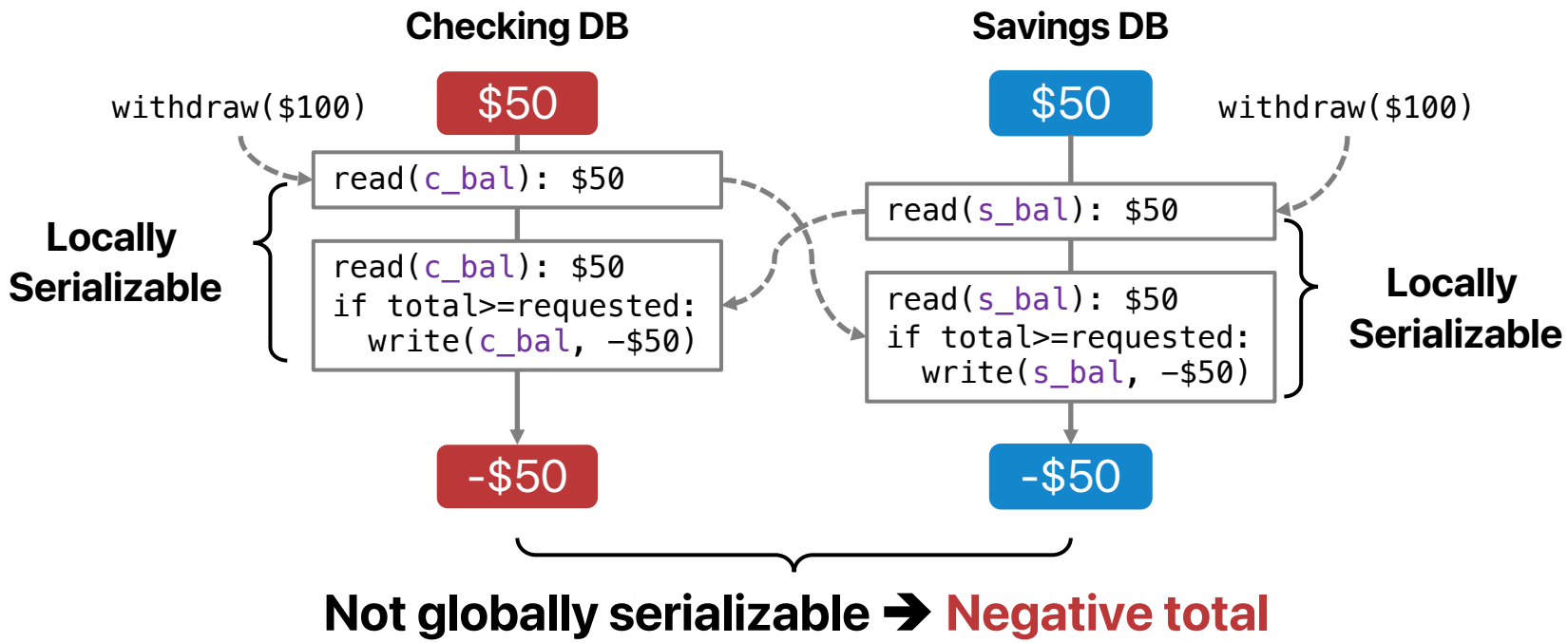
Goal: Global Serializability

- **Global serializability means serializable multi-DB transactions.**
  - We use "global" to distinguish from the serializability of individual local DBs.
- **While helpful for correctness, it is non-trivial to realize.**
  - Even if all DBs are locally serializable, global serializability can still be violated.



Goal: Global Serializability

- **Global serializability means serializable multi-DB transactions.**
  - We use "global" to distinguish from the serializability of individual local DBs.
- **While helpful for correctness, it is non-trivial to realize.**
  - Even if all DBs are locally serializable, global serializability can still be violated.

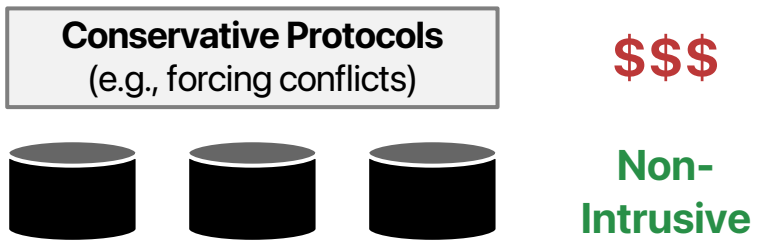


State-of-the-Art Approaches

Earlier Work [ICDE '91]*

Local DBs as serializable black boxes

- Force conflicts between subtransactions.
- **Large overhead** due to limited parallelism (>20-fold degradation)



* Georgakopoulos et al. 1991. On Serializability of Multidatabase Transactions Through Forced Local Conflicts. ICDE '91.

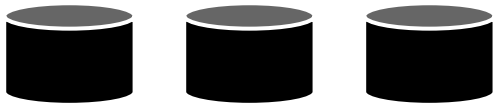
State-of-the-Art Approaches

Earlier Work [ICDE '91]*

Local DBs as serializable black boxes

- Force conflicts between subtransactions.
- **Large overhead** due to limited parallelism (>20-fold degradation)

Conservative Protocols
(e.g., forcing conflicts)



$$$

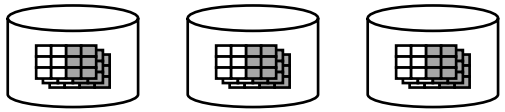
Non-Intrusive

Recent Work [VLDB '23]**

Local DBs as non-transactional stores

- Full concurrency control in middleware.
- **Still large overhead** as DB transaction mechanisms are exercised still.
- **Intrusive changes** to for additional CC metadata.

Fully App-Level Protocols
(e.g., app-level OCC)



$ $

Intrusive Metadata

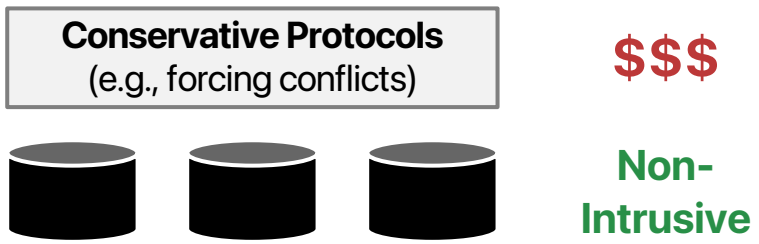
* Georgakopoulos et al. 1991. On Serializability of Multidatabase Transactions Through Forced Local Conflicts. ICDE '91.
** Yamada et al. 2023. ScalarDB: Universal Transaction Manager for Polystores. VLDB '23.

State-of-the-Art Approaches

Earlier Work [ICDE '91]*

Local DBs as serializable black boxes

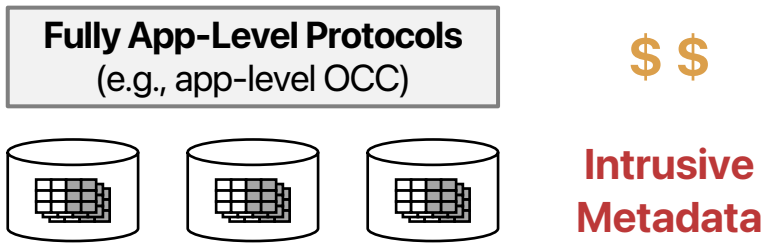
- Force conflicts between subtransactions.
- **Large overhead** due to limited parallelism (>20-fold degradation)



Recent Work [VLDB '23]**

Local DBs as non-transactional stores

- Full concurrency control in middleware.
- **Still large overhead** as DB transaction mechanisms are exercised still.
- **Intrusive changes** to for additional CC metadata.

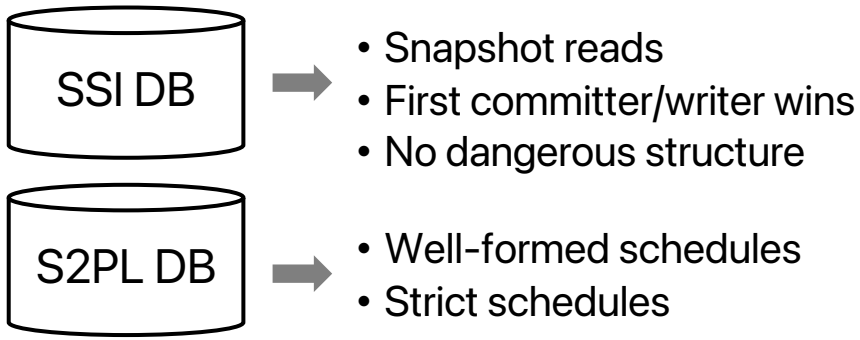


Can we achieve high performance without intrusive changes?

* Georgakopoulos et al. 1991. On Serializability of Multidatabase Transactions Through Forced Local Conflicts. ICDE '91.
** Yamada et al. 2023. ScalarDB: Universal Transaction Manager for Polystores. VLDB '23.

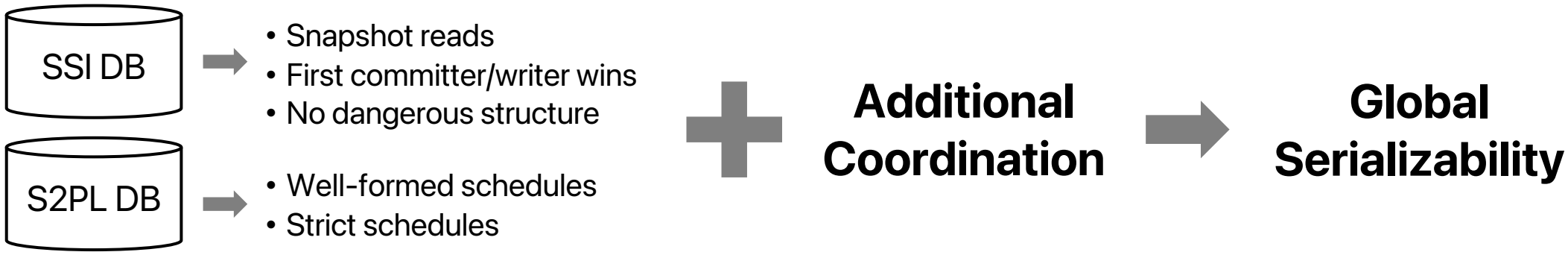
Our Gray-Box Approach

- **Observation: Dominant CC families in mainstream DBs, SSI & S2PL, offer useful common properties.**



Our Gray-Box Approach

- **Observation: Dominant CC families in mainstream DBs, SSI & S2PL, offer useful common properties.**
- **Basic Idea: Reuse such properties & add necessary coordination.**



Our Gray-Box Approach

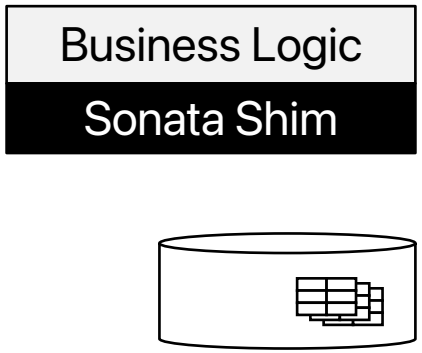
- **Observation: Dominant CC families in mainstream DBs, SSI & S2PL, offer useful common properties.**
- **Basic Idea: Reuse such properties & add necessary coordination.**

The theory of commitment ordering (CO) [VLDB '92] enables a locally enforceable condition, allowing lightweight coordination.

*Yoav Raz. 1992. The Principle of Commitment Ordering, or Guaranteeing Serializability in a Heterogeneous Environment of Multiple Autonomous Resource Mangers Using Atomic Commitment. VLDB '92.

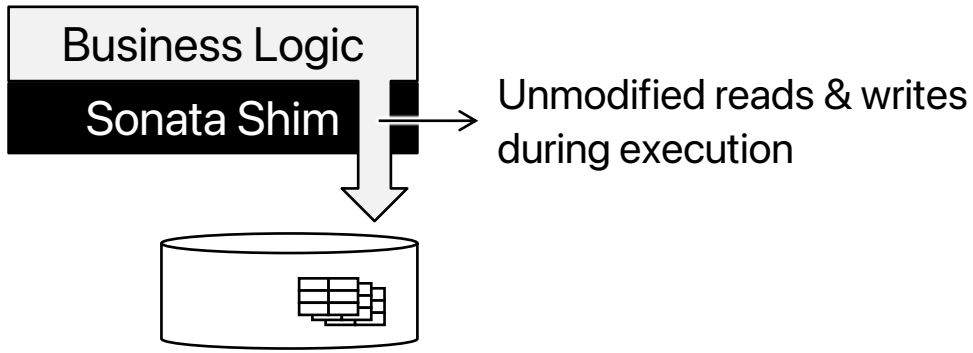
Sonata Overview

- **Sonata works as application-level shims.**



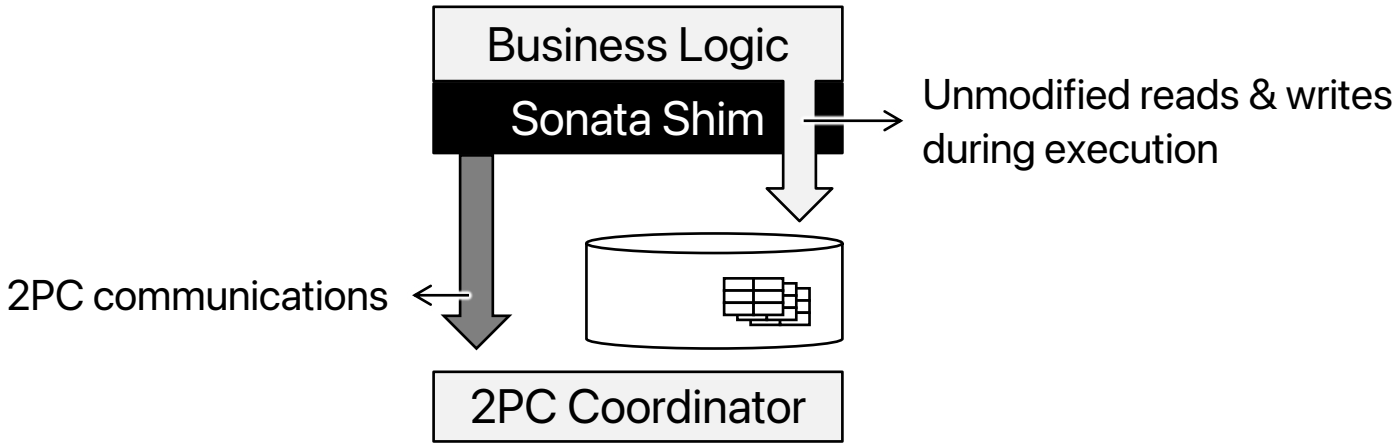
Sonata Overview

- **Sonata works as application-level shims.**
  - **No change** to apps' schemas, query statements, the DB drivers, or the DB systems.



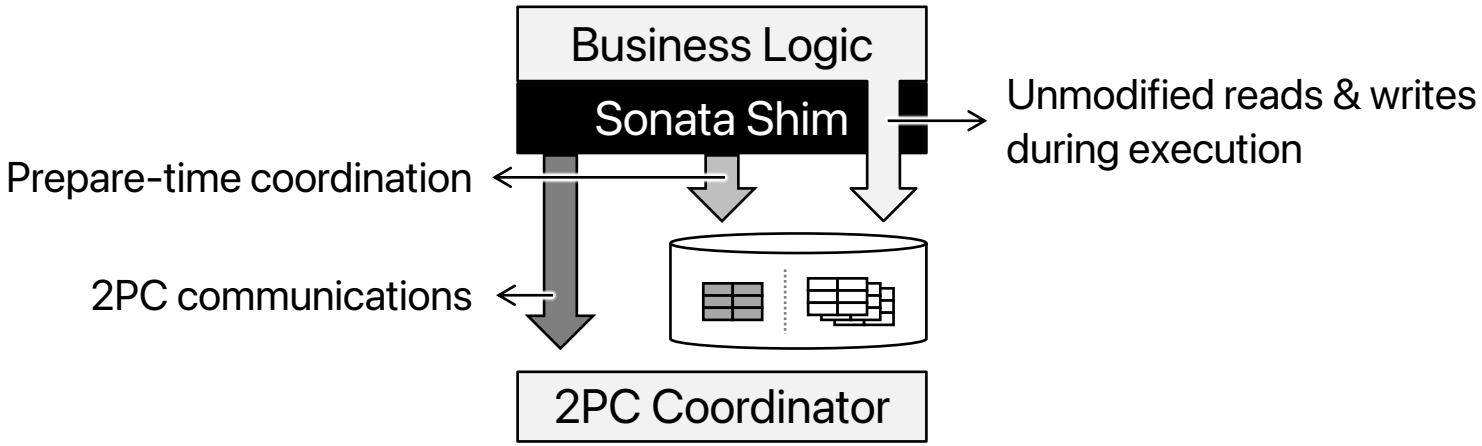
Sonata Overview

- **Sonata works as application-level shims.**
  - **No change** to apps' schemas, query statements, the DB drivers, or the DB systems.
  - Use 2PC for atomicity and durability.



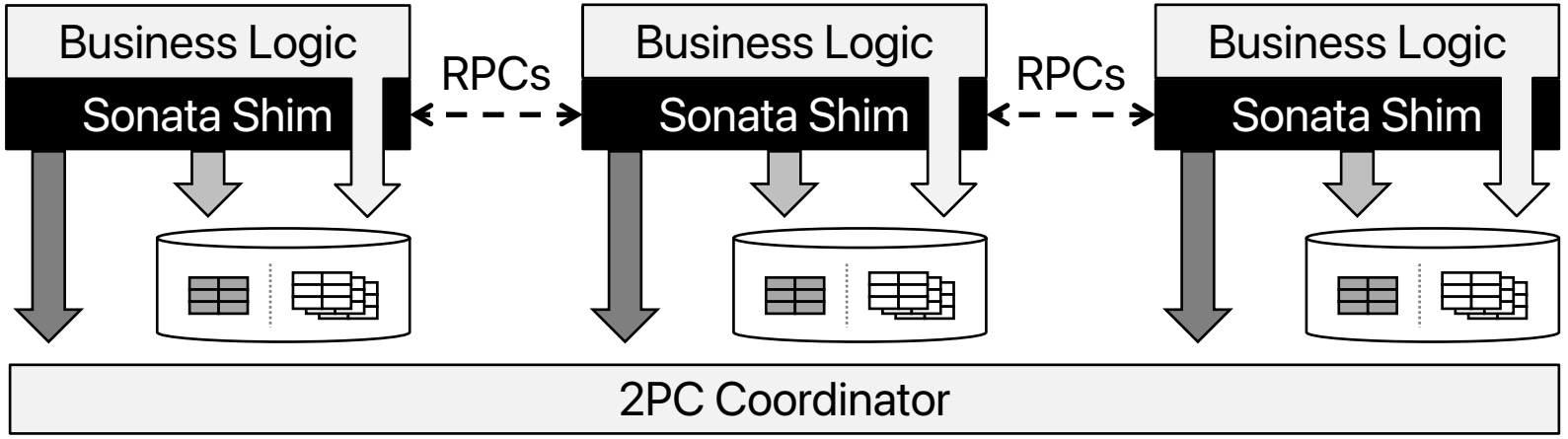
Sonata Overview

- **Sonata works as application-level shims.**
  - **No change** to apps' schemas, query statements, the DB drivers, or the DB systems.
  - Use 2PC for atomicity and durability.
  - Take actions **only at 2PC prepare time** independently at individual DBs.



Sonata Overview

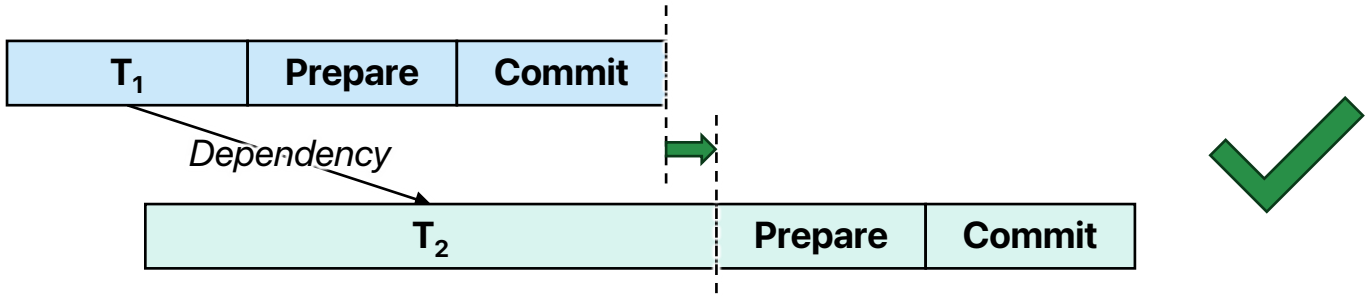
- **Sonata works as application-level shims.**
  - **No change** to apps' schemas, query statements, the DB drivers, or the DB systems.
  - Use 2PC for atomicity and durability.
  - Take actions **only at 2PC prepare time** independently at individual DBs.



(Also works with single-server, multi-DB cases)

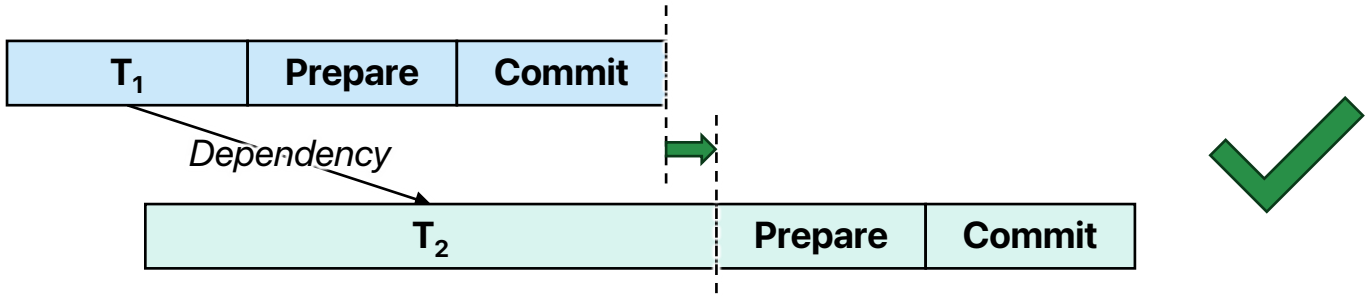
Global Serializability via Local Enforcement

- **The local condition** (derived from the CO theory):
  In a DB, if $T_1 \rightarrow T_2$, then T_1 commits before T_2 prepares.



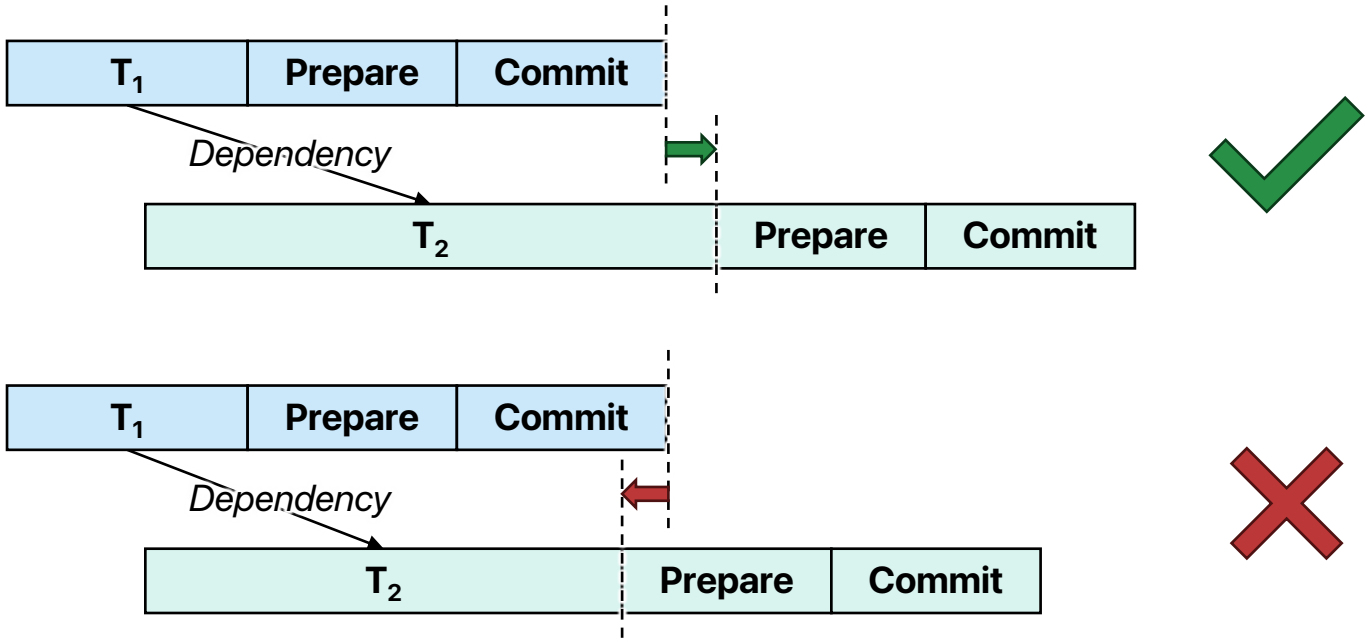
Global Serializability via Local Enforcement

- **The local condition** (derived from the CO theory):
  In a DB, if $T_1 \rightarrow T_2$, then T_1 commits before T_2 prepares.
- **If all DBs satisfies this condition, global serializability holds.**



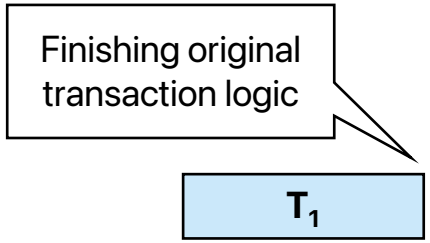
Global Serializability via Local Enforcement

- **The local condition** (derived from the CO theory):
  In a DB, if $T_1 \rightarrow T_2$, then T_1 commits before T_2 prepares.
- **If all DBs satisfies this condition, global serializability holds.**



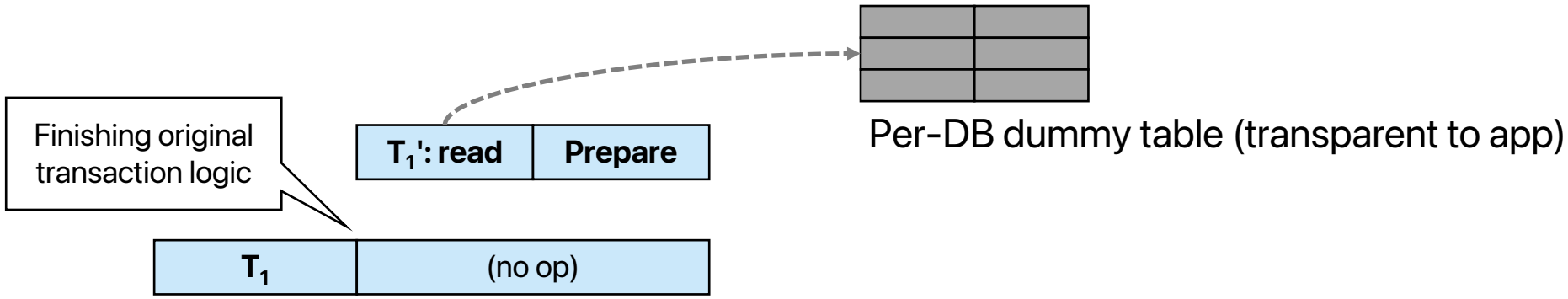
Enforcement in SSI DBs

- **For SSI DBs, we introduce helper transactions.**



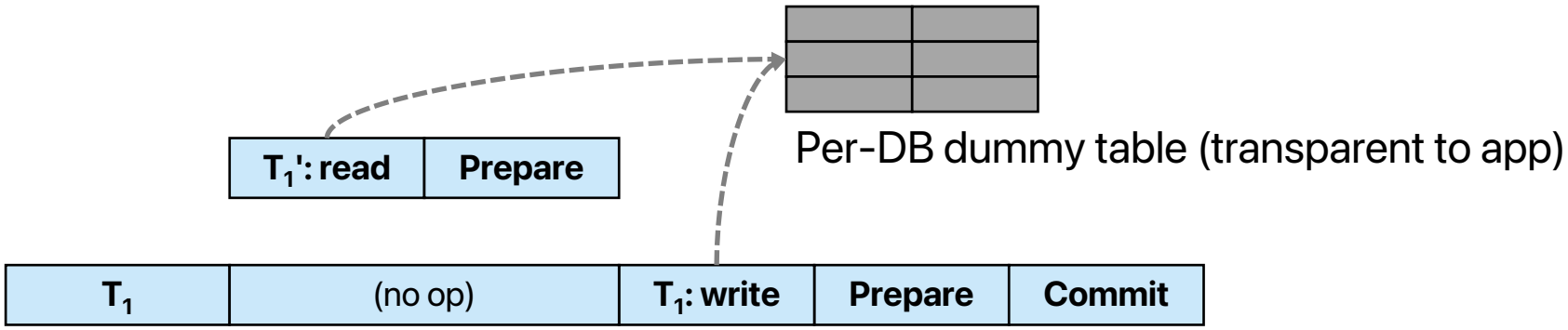
Enforcement in SSI DBs

- **For SSI DBs, we introduce helper transactions.**



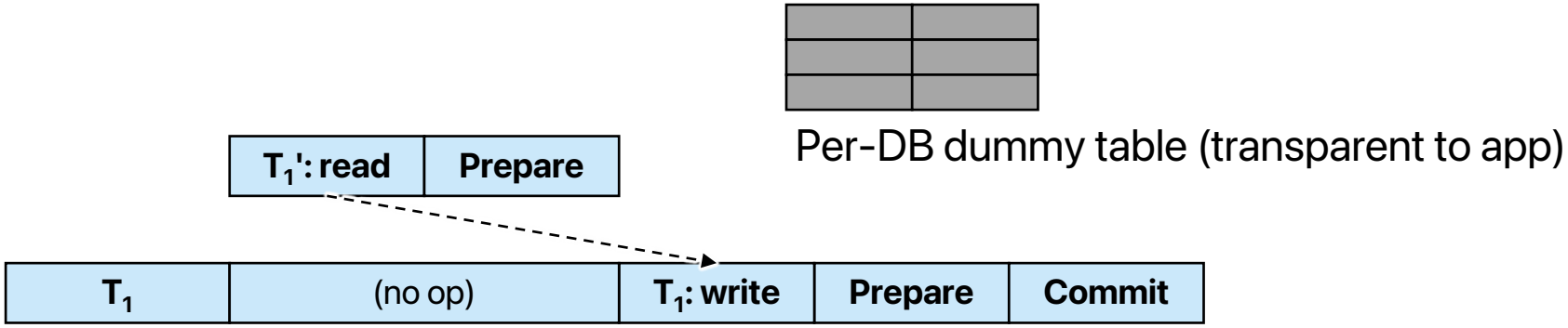
Enforcement in SSI DBs

- **For SSI DBs, we introduce helper transactions.**



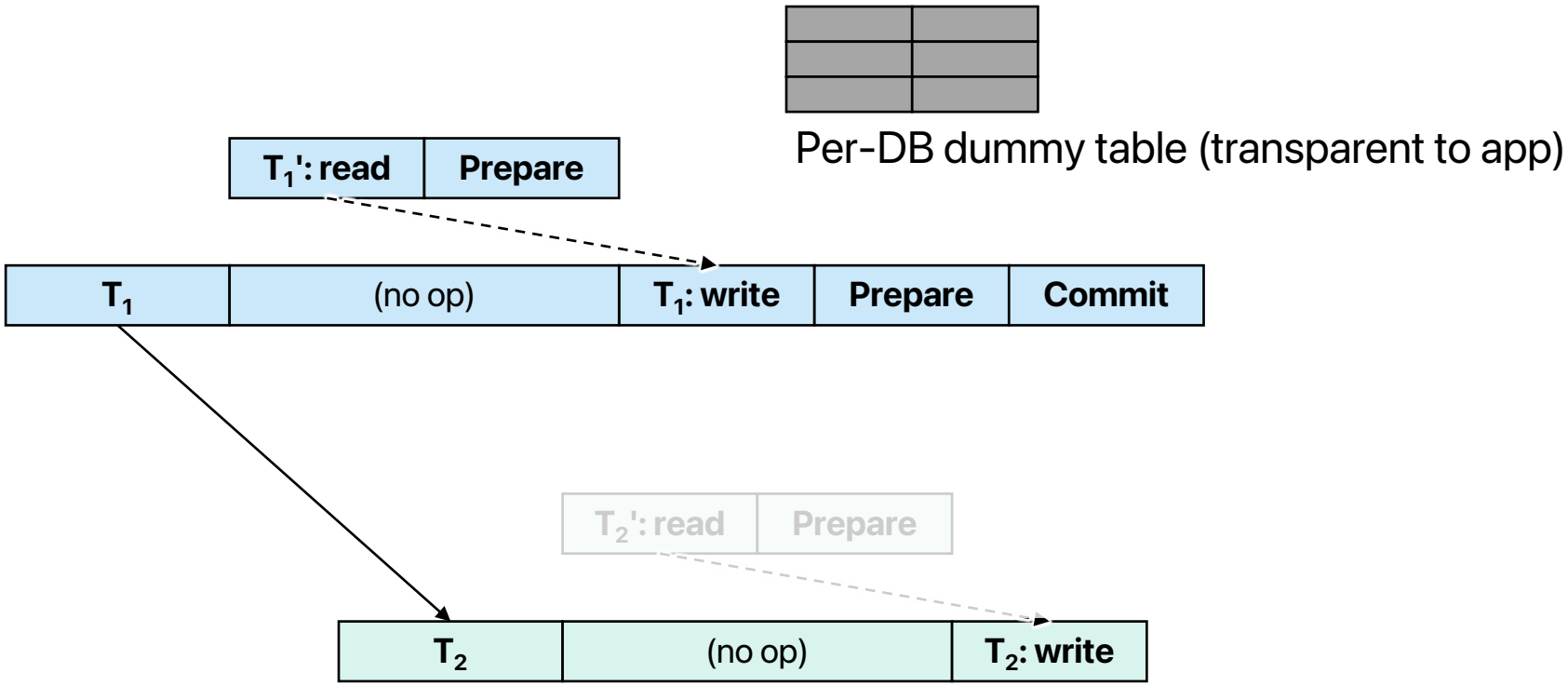
Enforcement in SSI DBs

- **For SSI DBs, we introduce helper transactions.**



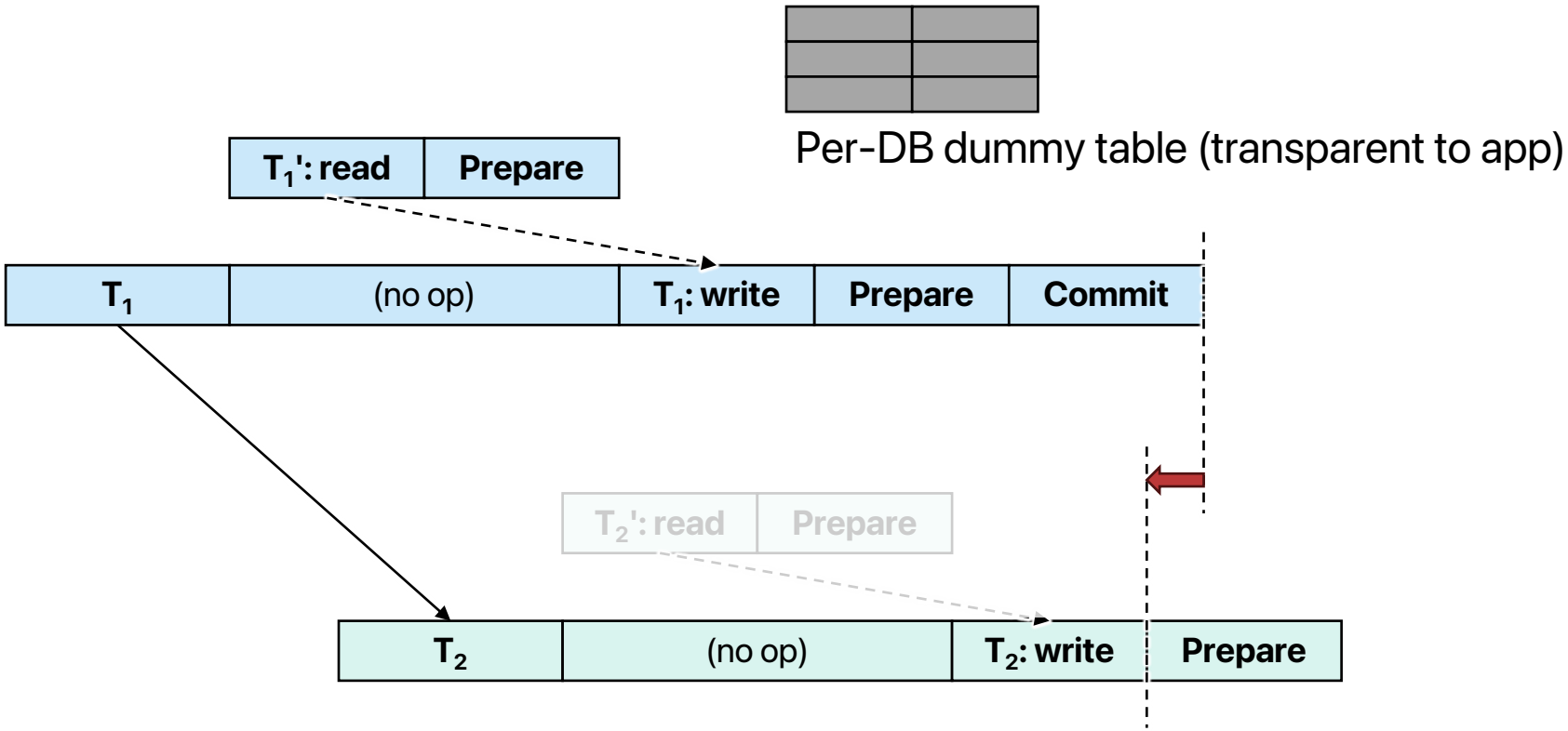
Enforcement in SSI DBs

- **For SSI DBs, we introduce helper transactions.**



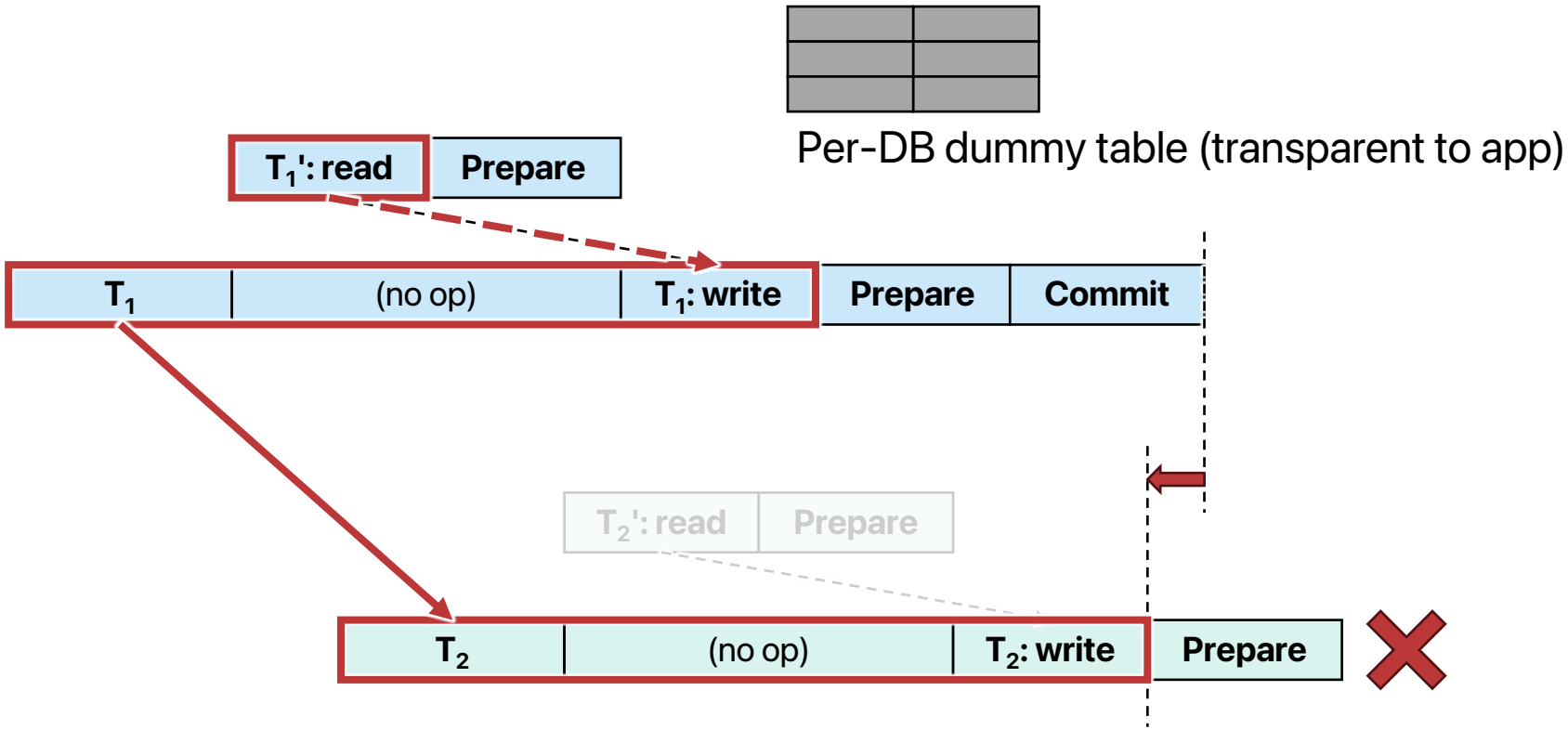
Enforcement in SSI DBs

- **For SSI DBs, we introduce helper transactions.**



Enforcement in SSI DBs

- **For SSI DBs, we introduce helper transactions.**

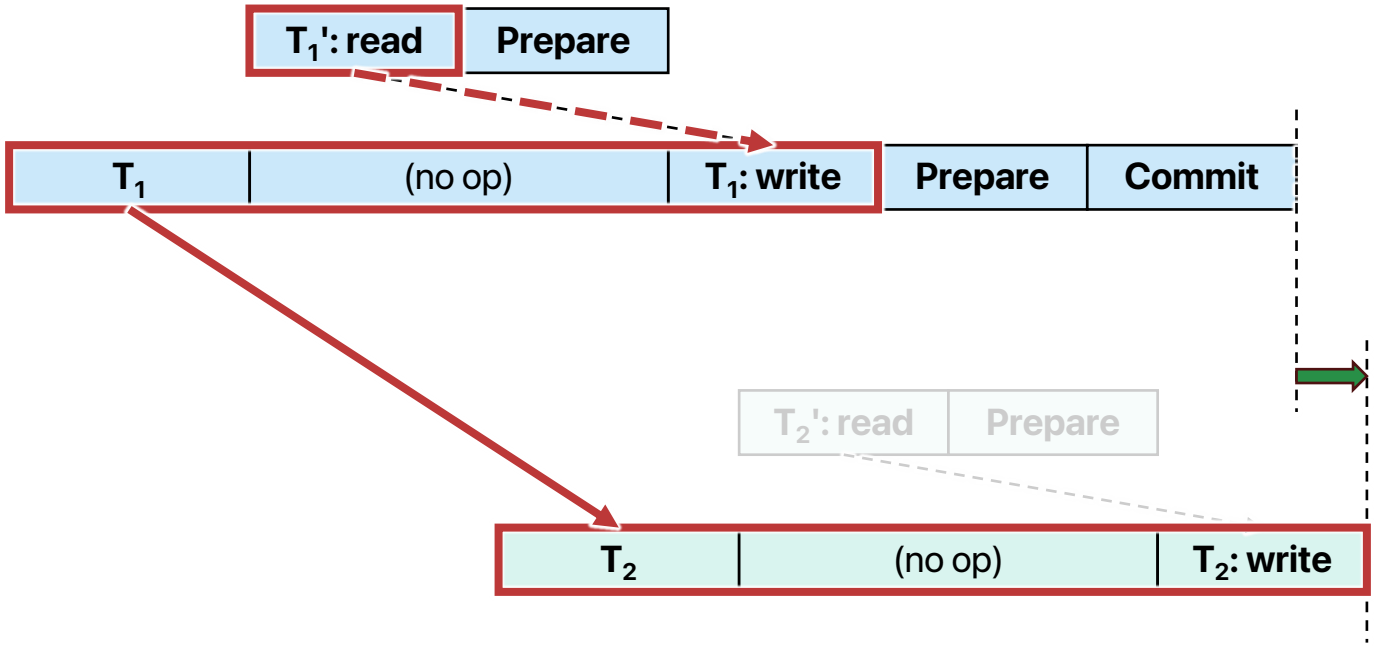


SSI dangerous structure!

T2 will be aborted → No violation

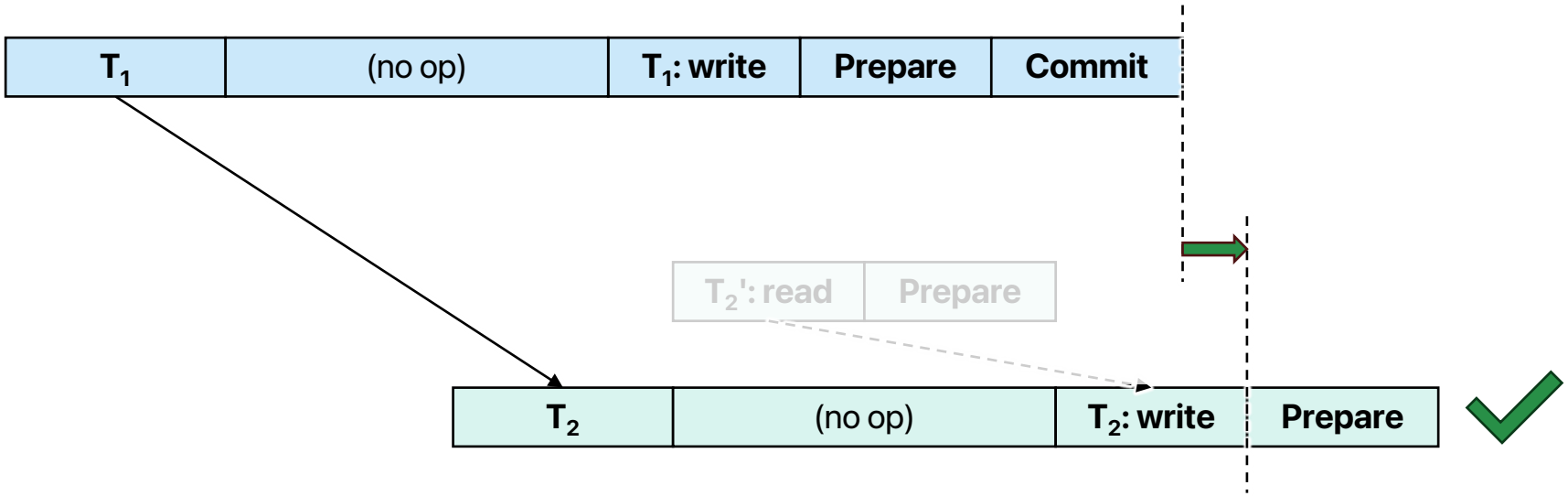
Enforcement in SSI DBs

- **For SSI DBs, we introduce helper transactions.**



Enforcement in SSI DBs

- **For SSI DBs, we introduce helper transactions.**

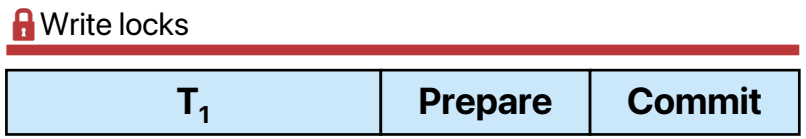


No SSI dangerous structure!

T2 won't be aborted → No false positive

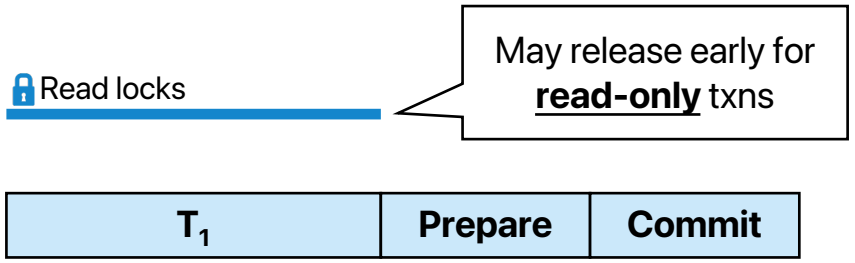
Enforcement in S2PL DBs

- **For S2PL DBs, we simply introduce dummy writes.**



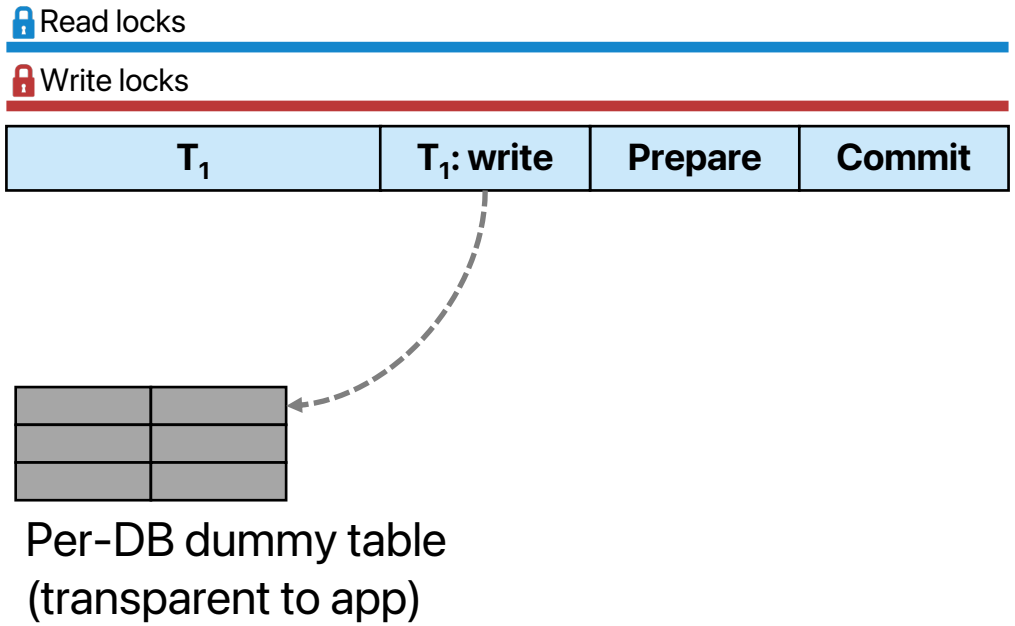
Enforcement in S2PL DBs

- **For S2PL DBs, we simply introduce dummy writes.**



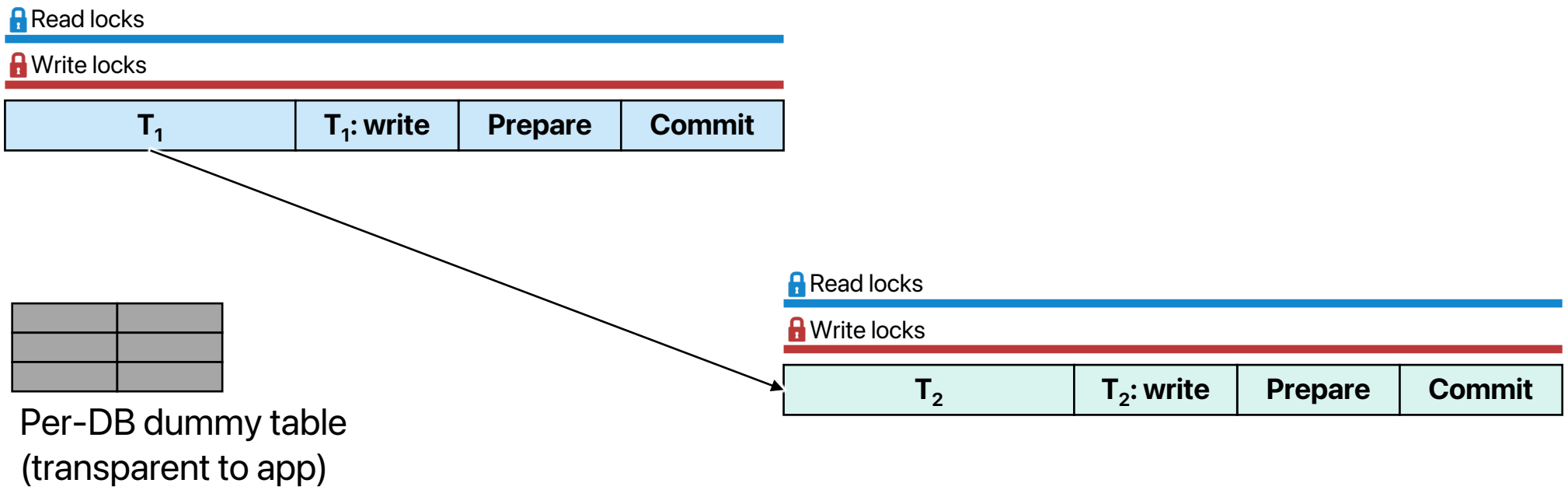
Enforcement in S2PL DBs

- **For S2PL DBs, we simply introduce dummy writes.**



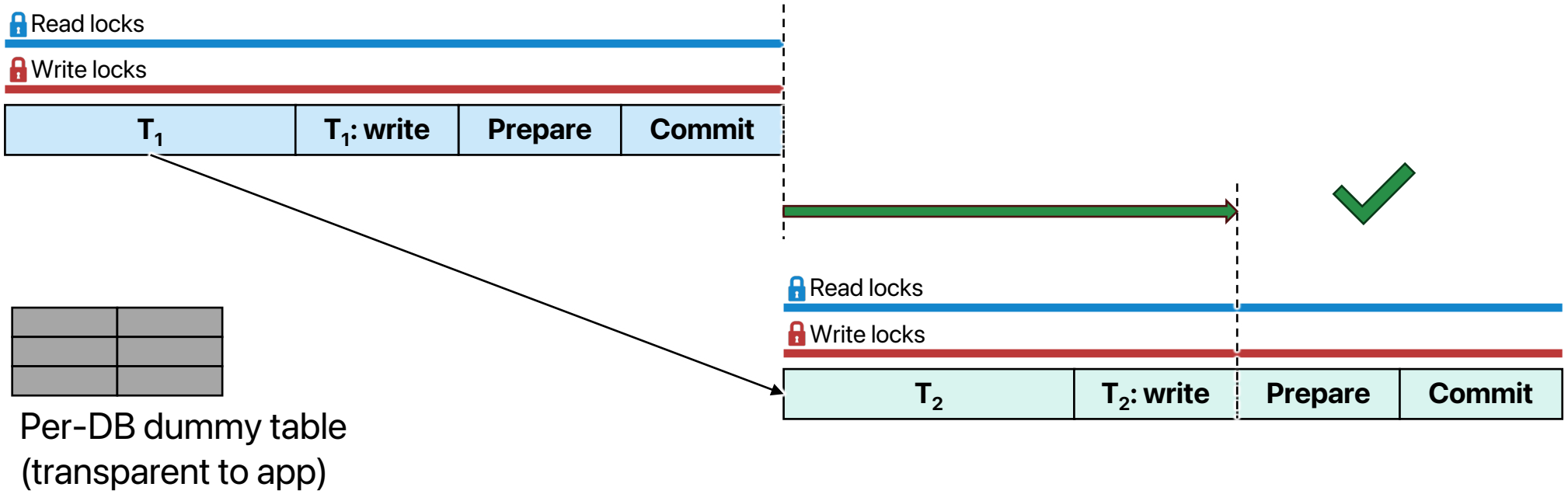
Enforcement in S2PL DBs

- **For S2PL DBs, we simply introduce dummy writes.**



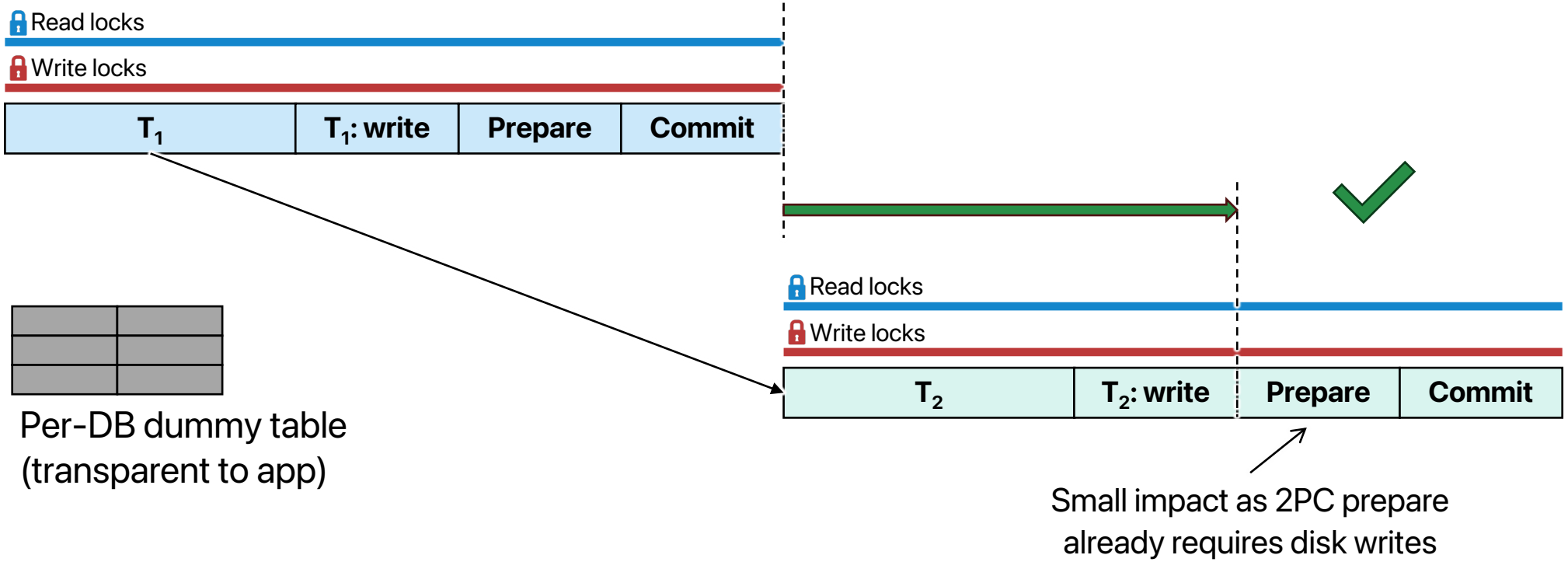
Enforcement in S2PL DBs

- **For S2PL DBs, we simply introduce dummy writes.**



Enforcement in S2PL DBs

- **For S2PL DBs, we simply introduce dummy writes.**



Correctness and False Positives

- **Sonata provably guarantees the local condition, thus global serializability.** (Sec. 4.1)
  - The proofs leverage lemmas from [TODS 05]*.

- **With PostgreSQL and MySQL, there's no false positive if the DB's conflict detection is accurate.** (Sec. 4.2)
  - The analysis depends on DB's implementation details.

* Fekete et al. 2005. Making snapshot isolation serializable. ACM TODS.

Programming with Sonata

- **Our prototype is in Java and is based on Apache Seata** (25k+ GH ★)**.**

Programming with Sonata

- **Our prototype is in Java and is based on Apache Seata** (25k+ GH ★).
- **For existing applications, codebase changes can be as small as adding a few annotations.**

```java
@GlobalTransactional
public void workflow() {
    // Access the local database
    jdbcClient.sql("Select ...").query();

    // Request other services
    restClient.post().uri( uri: "/api").retrieve().toBodilessEntity();
}
```

Programming with Sonata

- **Our prototype is in Java and is based on Apache Seata** (25k+ GH ★).
- **For existing applications, codebase changes can be as small as adding a few annotations.**

```java
@GlobalTransactional
public void workflow() {
    // Access the local database
    jdbcClient.sql("Select ...").query();

    // Request other services
    restClient.post().uri( uri: "/api").retrieve().toBodilessEntity();
}
```

→ **JDBC or any JDBC-based library**
(We intercept begin/commit calls)

Programming with Sonata

- **Our prototype is in Java and is based on Apache Seata** (25k+ GH ★).
- **For existing applications, codebase changes can be as small as adding a few annotations.**

```
@GlobalTransactional
public void workflow() {
    // Access the local database
    jdbcClient.sql("Select ...").query();

    // Request other services
    restClient.post().uri( uri: "/api").retrieve().toBodilessEntity();
}
```

JDBC or any JDBC-based library
(We intercept begin/commit calls)

Any RPC framework able to piggyback transaction ID strings
(We currently intercept HTTP requests)

Sonata Performance

- **We build a multi-DB TPC-C by partitioning by warehouses.**
  - Either PostgreSQL or MySQL as local DBs.
  - Global serializability: ScalarDB [VLDB '23] & Ticket [ICDE '91].

* Kraft et al. 2023. Epoxy: ACID Transactions across Diverse Data Stores. VLDB '23.

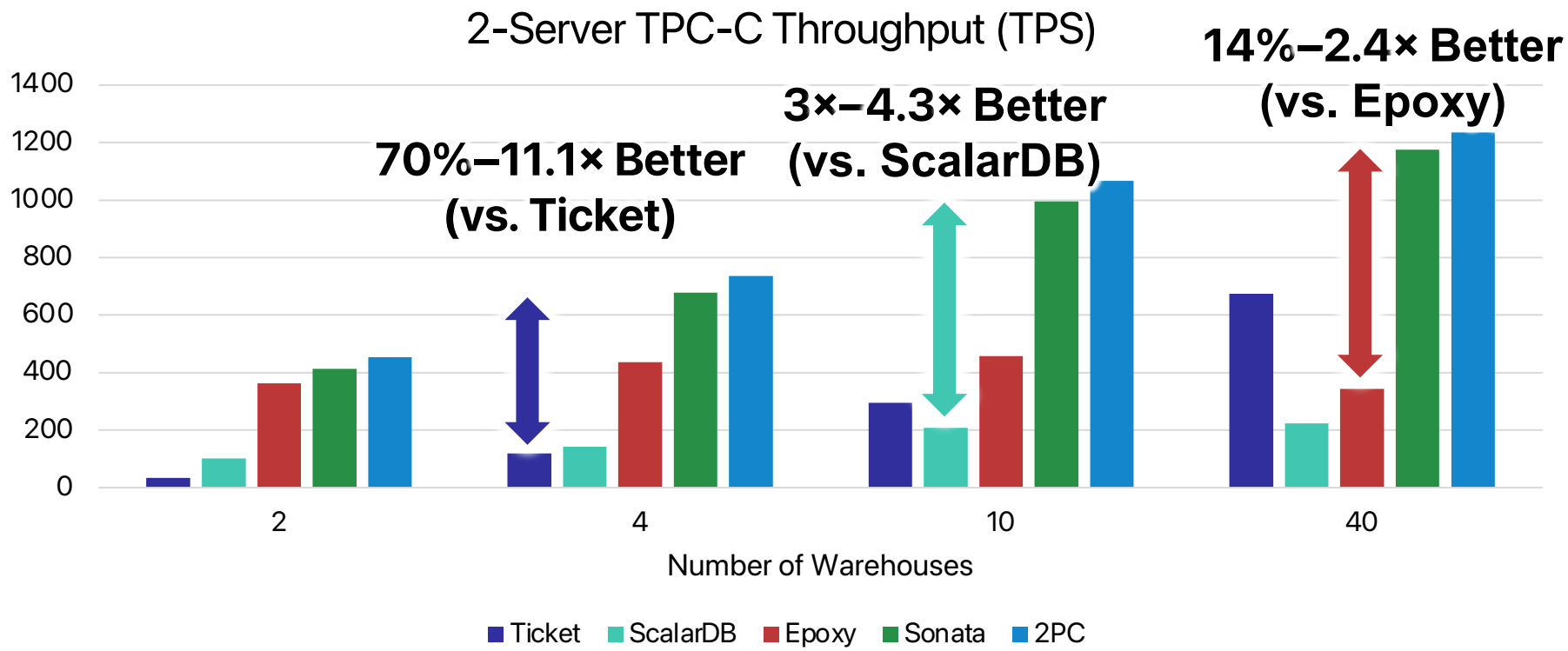
Sonata Performance

- **We build a multi-DB TPC-C by partitioning by warehouses.**
  - Either PostgreSQL or MySQL as local DBs.
  - Global serializability: ScalarDB [VLDB '23] & Ticket [ICDE '91].
  - Snapshot isolation: Epoxy* [VLDB '23].

* Kraft et al. 2023. Epoxy: ACID Transactions across Diverse Data Stores. VLDB '23.

Sonata Performance

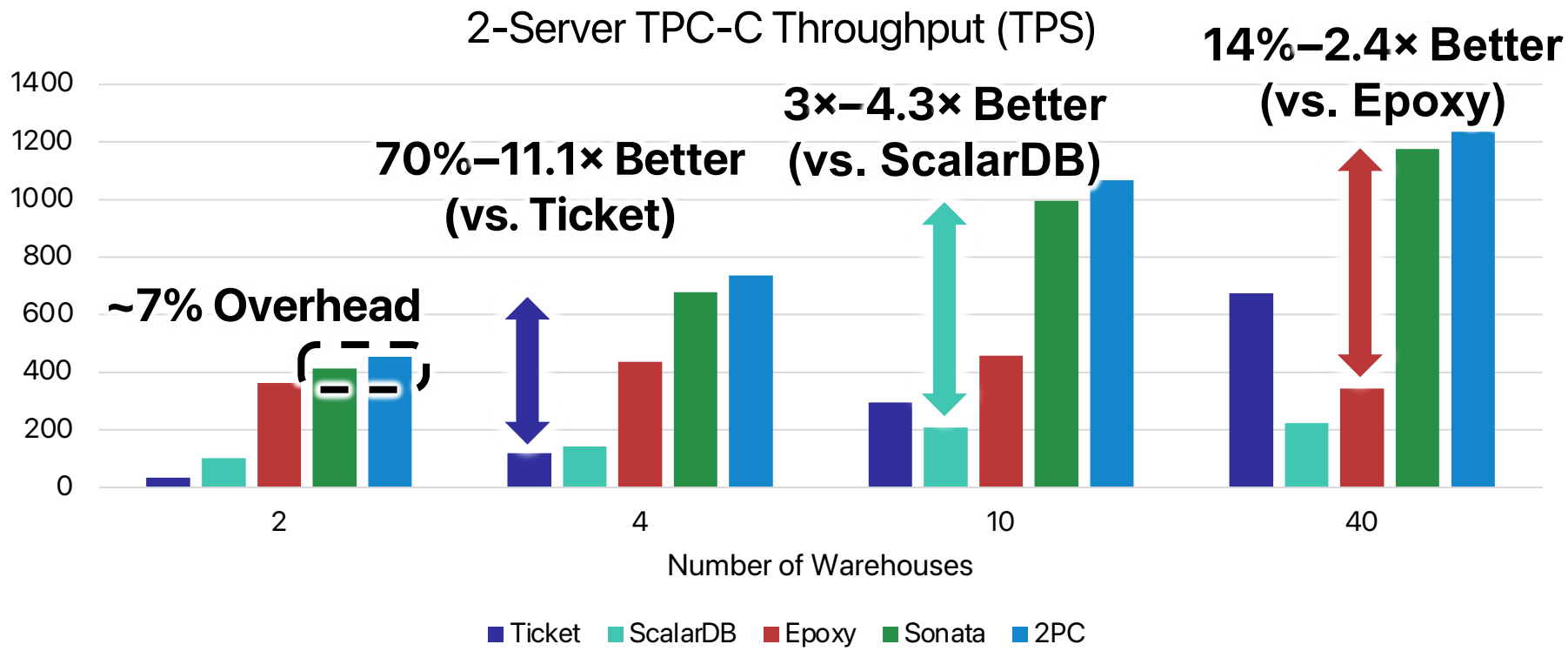
- **We build a multi-DB TPC-C by partitioning by warehouses.**
  - Either PostgreSQL or MySQL as local DBs.
  - Global serializability: ScalarDB [VLDB '23] & Ticket [ICDE '91].
  - Snapshot isolation: Epoxy* [VLDB '23].



* Kraft et al. 2023. Epoxy: ACID Transactions across Diverse Data Stores. VLDB '23.

Sonata Performance

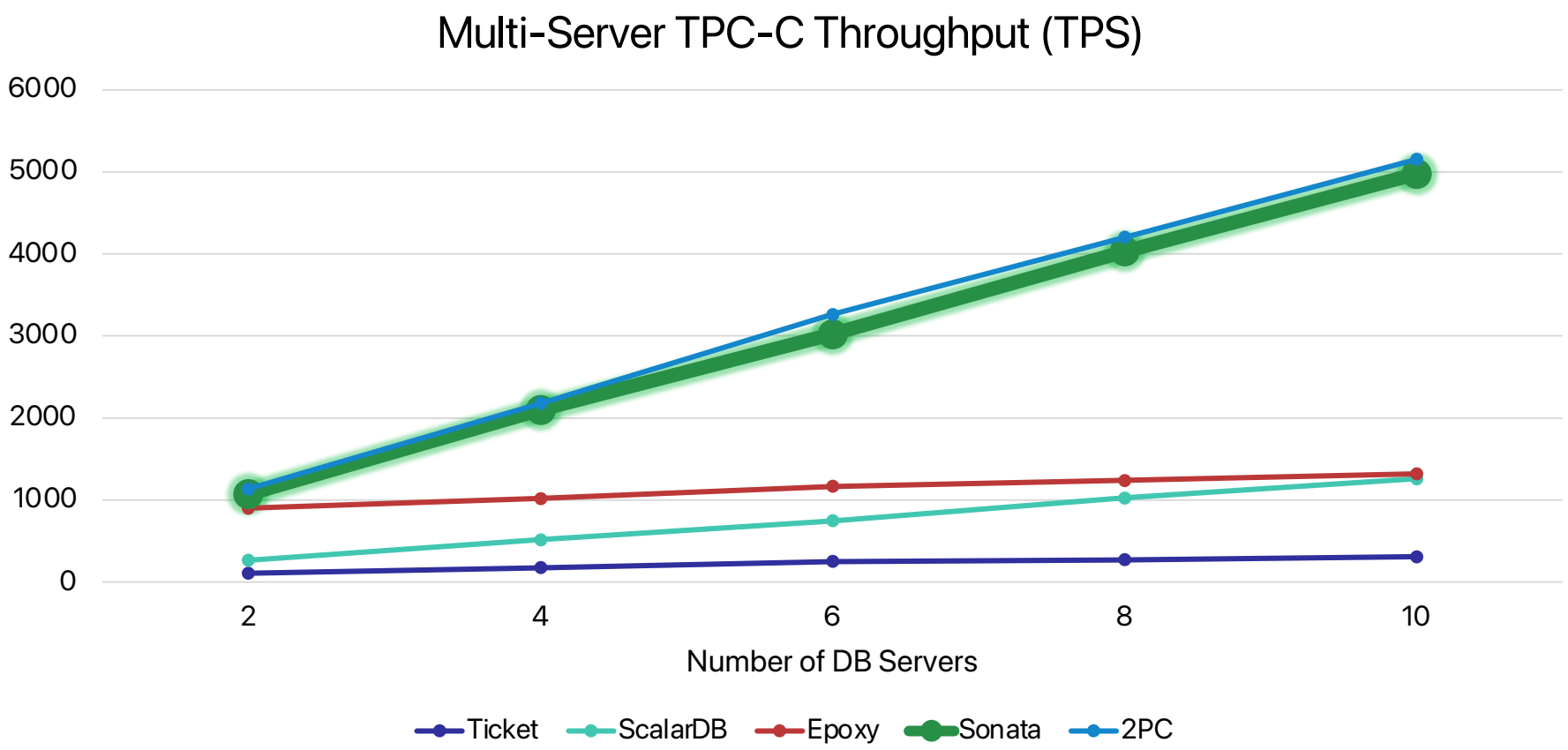
- **We build a multi-DB TPC-C by partitioning by warehouses.**
  - Either PostgreSQL or MySQL as local DBs.
  - Global serializability: ScalarDB [VLDB '23] & Ticket [ICDE '91].
  - Snapshot isolation: Epoxy* [VLDB '23].



* Kraft et al. 2023. Epoxy: ACID Transactions across Diverse Data Stores. VLDB '23.

Sonata Performance

- **For scalability, at 10 nodes, Sonata (green) achieves 4.65× its 2-node throughput.**



Summary and Q/A

- **Sonata offers global serializability in an efficient, non-intrusive way.**
- **It reuses DB concurrency control to add lightweight coordination.**
- **It achieves up to 11.1× higher throughput than prior solutions.**

- **Check out our paper and code!**
  - Design details, optimizations, proofs, & more evaluation results.

Paper

Code