

ACDC

Scalable Distributed Transactions



Ivan Klianov
Transactum Pty Ltd

HPTS 2011

ACDC Transaction

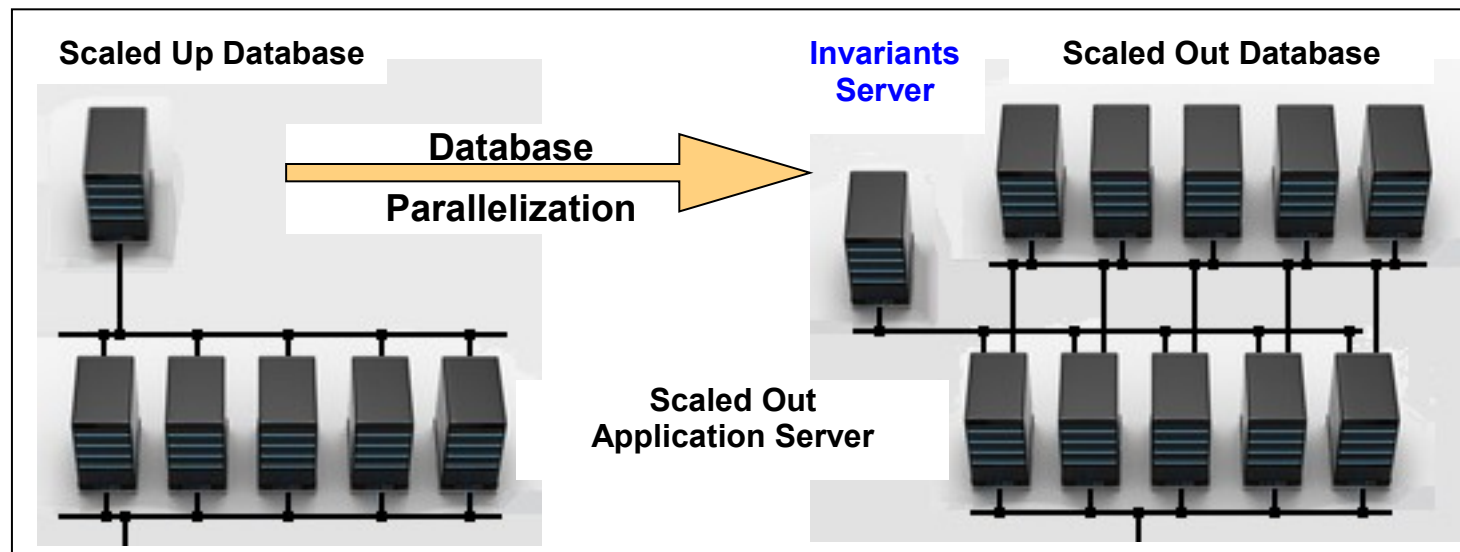
- Atomic
- Consistent
- Distributed
- Concurrent

Transaction coordination with:

- Serializing schedules and
- Prevented violation of integrity
- Enforced as **invariants**.

Invariants-based Transactions With Parallel Databases

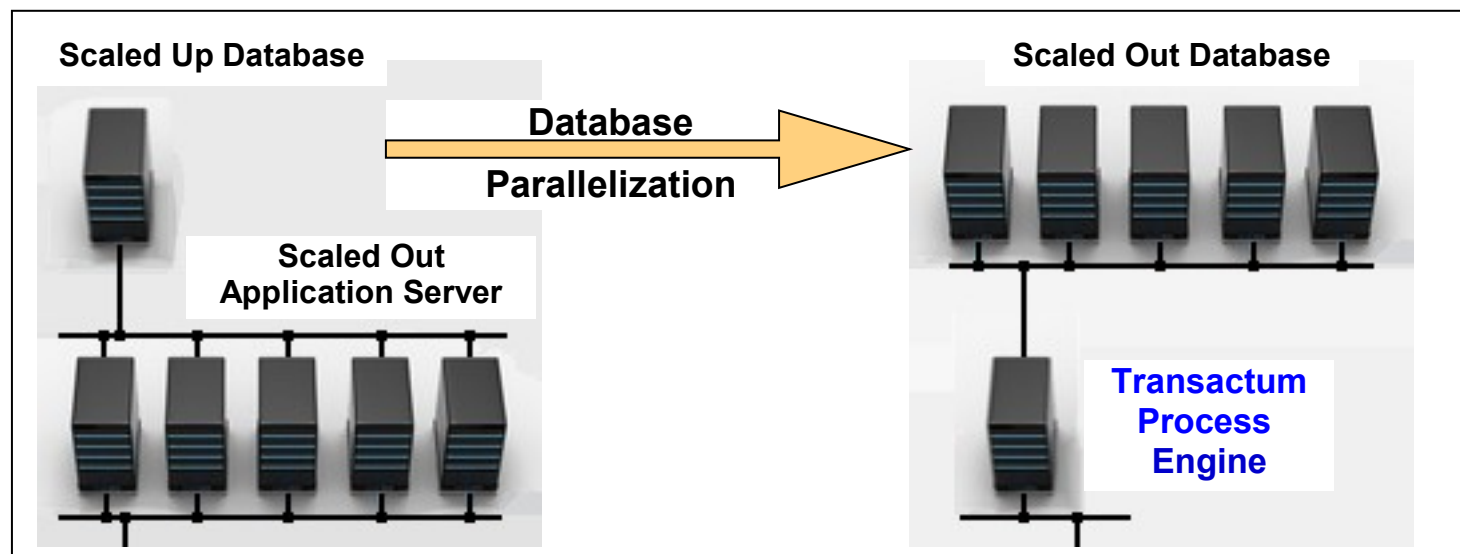
Implementation with **Mainstream** Platforms



- Requires a dedicated **Invariants Server**.
- Injects latency with networking loops.
- Makes the transactions inefficient.

Invariants-based Transactions With Parallel Databases

Implementation with Transactum Process Engine



Transactum Process Engine gives the **advantage** of:

- Scaling the application instances on **one** machine
- Running the invariants on **the same** machine.

ACDC Transaction Concept

Definition

ACDC is:

- Flat transaction,
- Chaining multiple sub-transactions,
- Which can execute on individual shards.

It guarantees:

- **Atomic** and
- **Consistent** execution with a
- **Distributed** database over multiple shards and
- **Concurrent** performance of multiple conflicting instances.

ACDC Transaction Concept

ACDC Atomicity

- Permits **multiple consistent** completion states.
- **Prevents rollback** as response to technical and semantic failures.
- Transactions with a **technical** failure:
 - In a **sub-transaction** – commit it on Replicated Database.
 - In **coordination** – continue on backup Process Engine.
- Transactions with **semantic** failure:
 - **Continue** with another chain of sub-transactions.
 - **Commit** with a state representing the semantic result.

ACDC Transaction Concept

ACDC Consistency

- **Serialization** of local sub-transactions:
 - With **syntactic** invariants
 - As **pre-execution** condition
- Prevented **violation of integrity**:
 - With **semantic** invariants
 - As **post-execution** condition

ACDC Transaction Implementation

Concurrency Control

- Based on global **serializability schedules**,
- Built from programmer-specified **transaction syntax**.

Integrity Control

- Checks for **violation** of integrity constraints,
- **Before** execution of a local sub-transaction, with
- Programmer-provided **binary code with semantics**.

ACDC Transaction Implementation

Recovery

- Coordination clustering:
 - **Persists** transactions' intermediate state.
 - Takes care for **availability** of transaction coordination.
 - **Continues** the active transactions from their last committed step.
- Replication clustering:
 - **Synchronously** replicates local sub-transactions.
 - Takes care for **partition tolerance** of the database.

ACDC Transaction Implementation

Partition Tolerance

Achieved with **real-time response** to failures of:

- **Network** by:
 - Duplication of network segments
 - Detection of failure and switching to backup segment
- **Node** by:
 - **Synchronous** replication of state of DB nodes.
 - **Asynchronous** replication of state of the Process Engine –
Lost messages are dealt with Idempotent steps.
 - Detection of failure and switching to backup node.

ACDC Experimental Tests Environment

Hardware

\$500-Computers with:

- Intel i5 Quad-Core CPU
- 4MB RAM
- 2 x Spinning HD
- 2 x 1Gb LAN

Software

- Windows 2003 Enterprise Server x64
- MS SQL Server 2008 x64
- Transactum Process Engine
- Test Application

ACDC Experimental Tests

Test Application

Performs 5 steps in two alternative configurations:

- With 5 Chained 2PL transactions.
- With a single ACDC transaction.

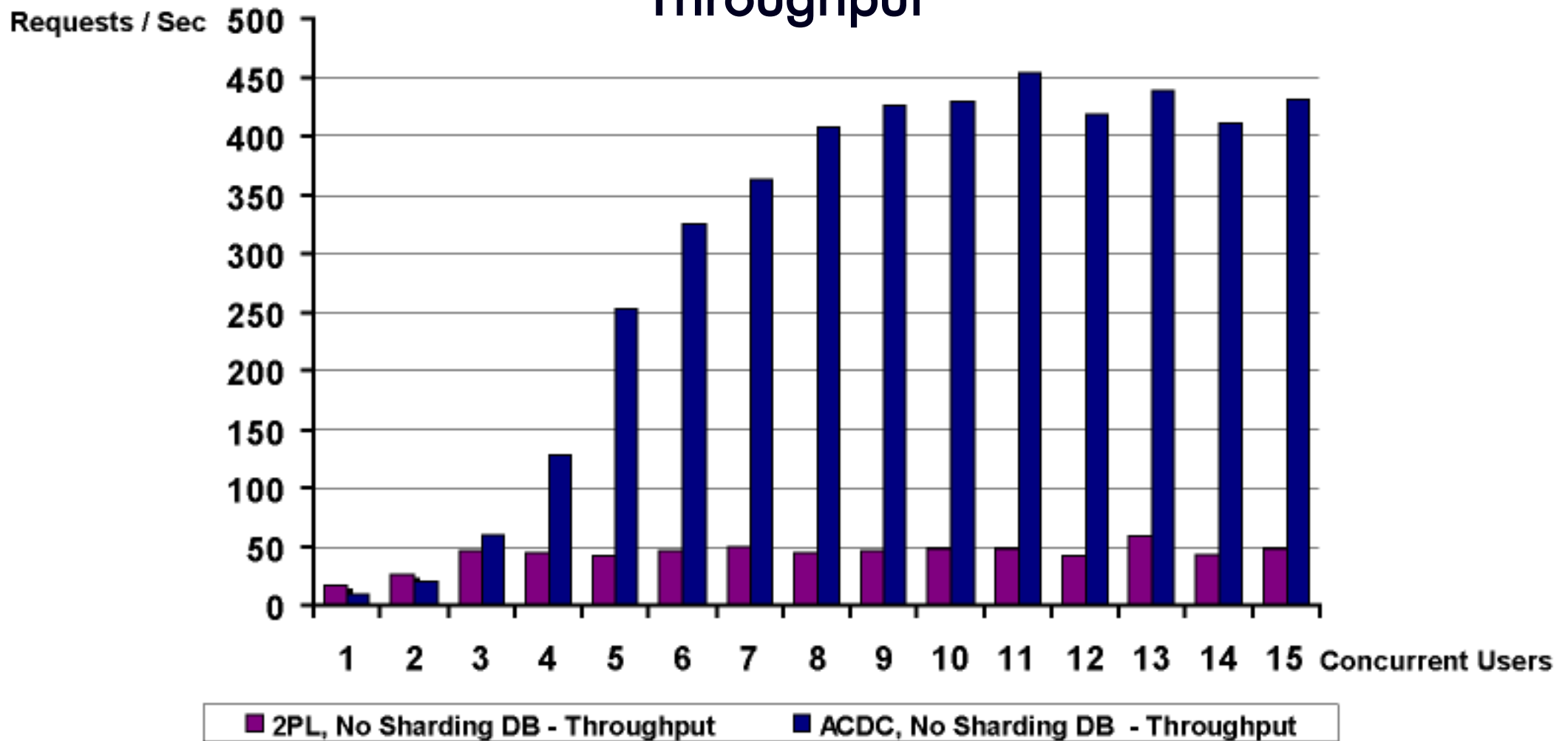
The steps:

- **Insert** a short DB record (all steps).
- **Update** a DB aggregation (all steps).
- **Insert** a DB record with 6 strings (step 5 only).

ACDC Tests Results

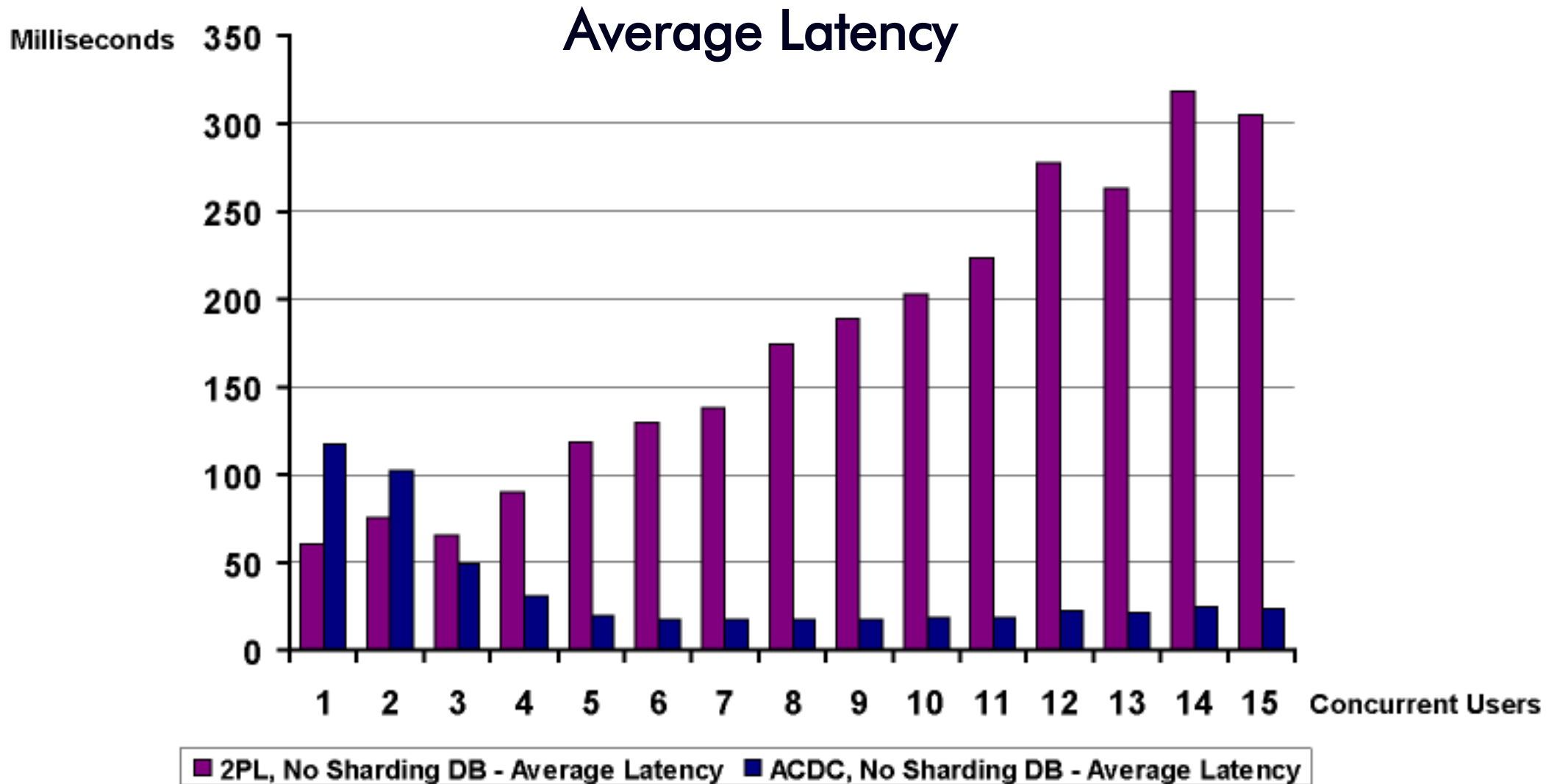
2PL Chained vs. ACDC

Throughput



ACDC Tests Results

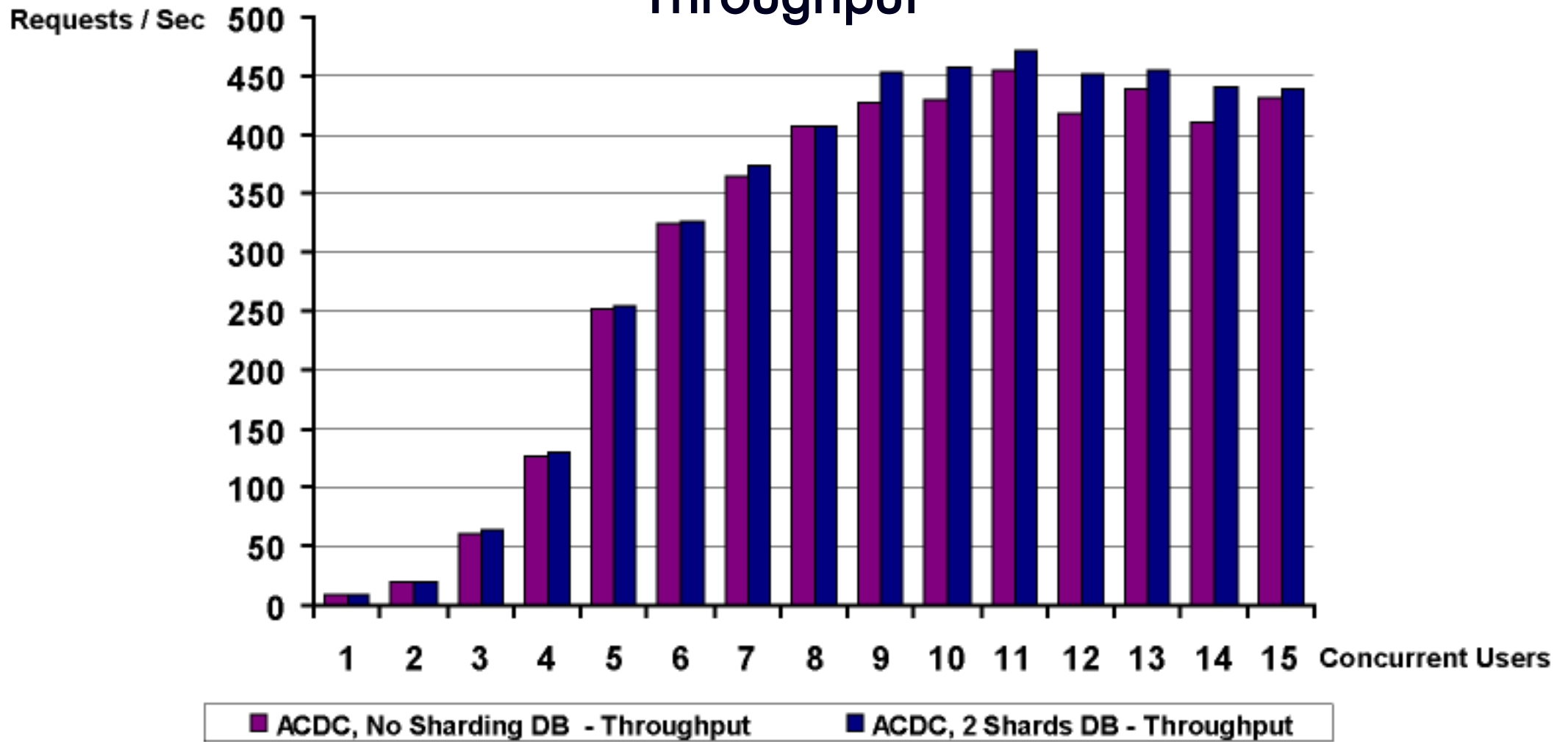
2PL Chained vs. ACDC



ACDC Tests Results

No Sharding ACDC vs. 2 Shards ACDC

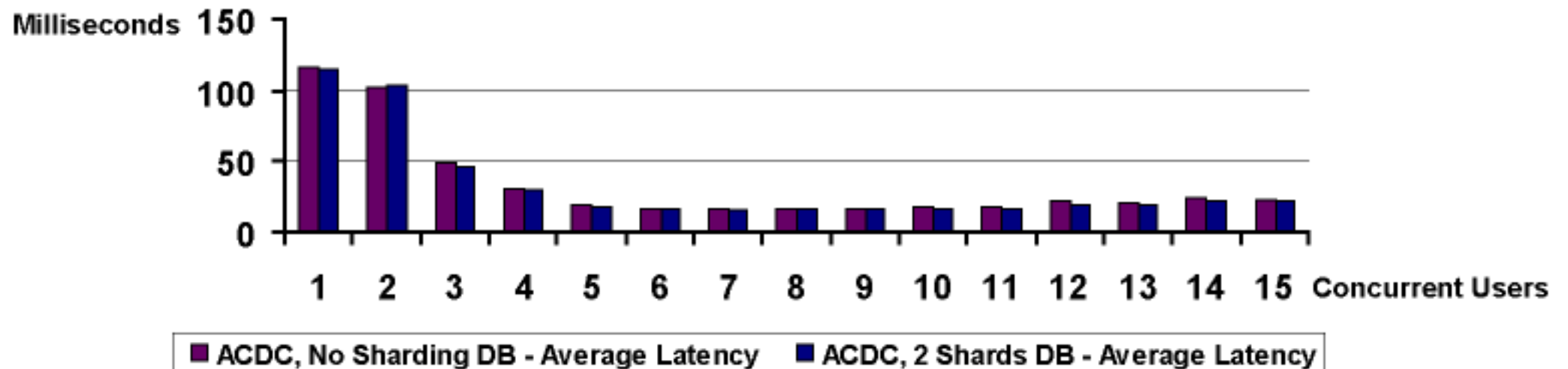
Throughput



ACDC Tests Results

No Sharding ACDC vs. 2 Shards ACDC

Average Latency



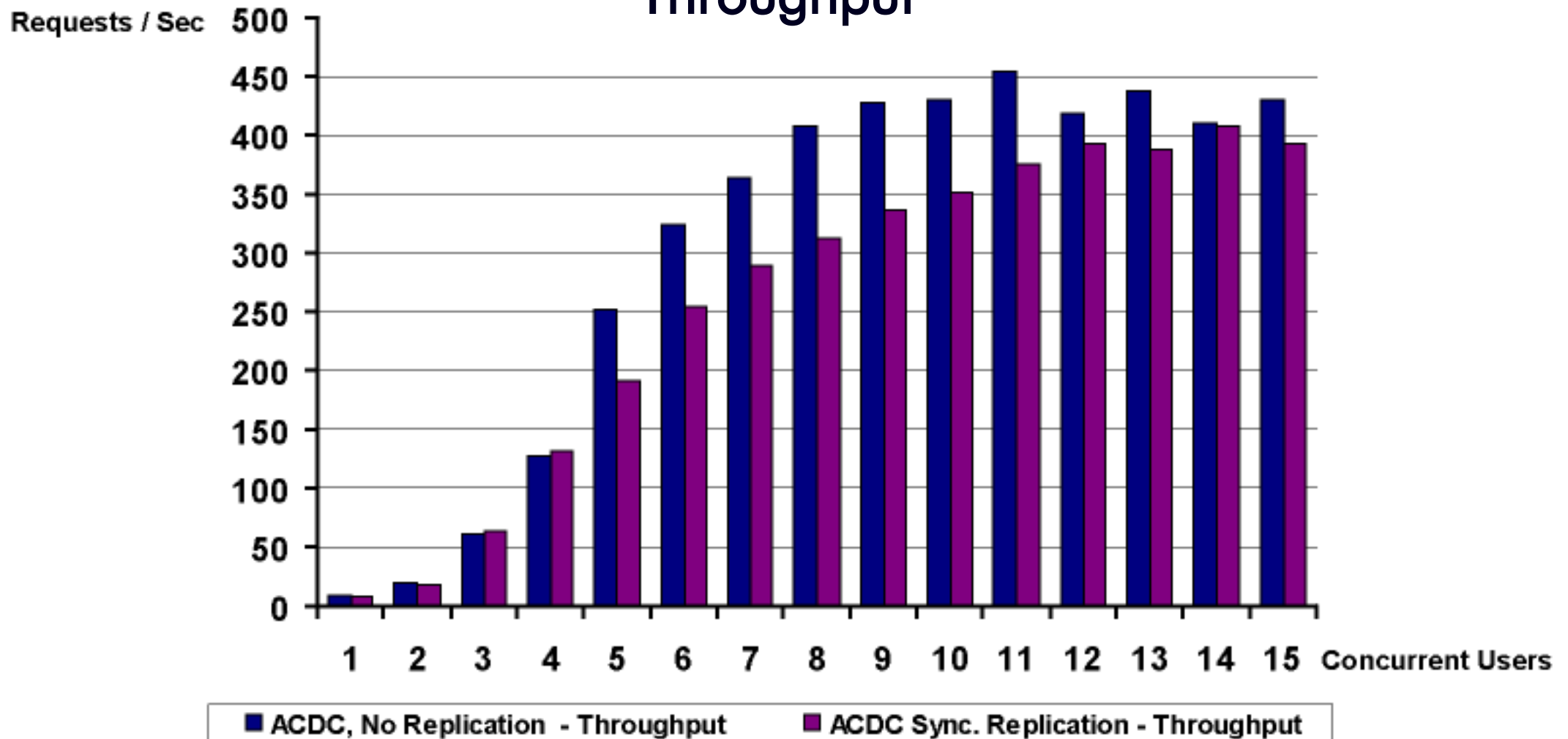
The results with 2-Shards ACDC:

- Show that DB is already not the system bottleneck.

ACDC Tests Results

No Replication vs. Synchronous Replication ACDC

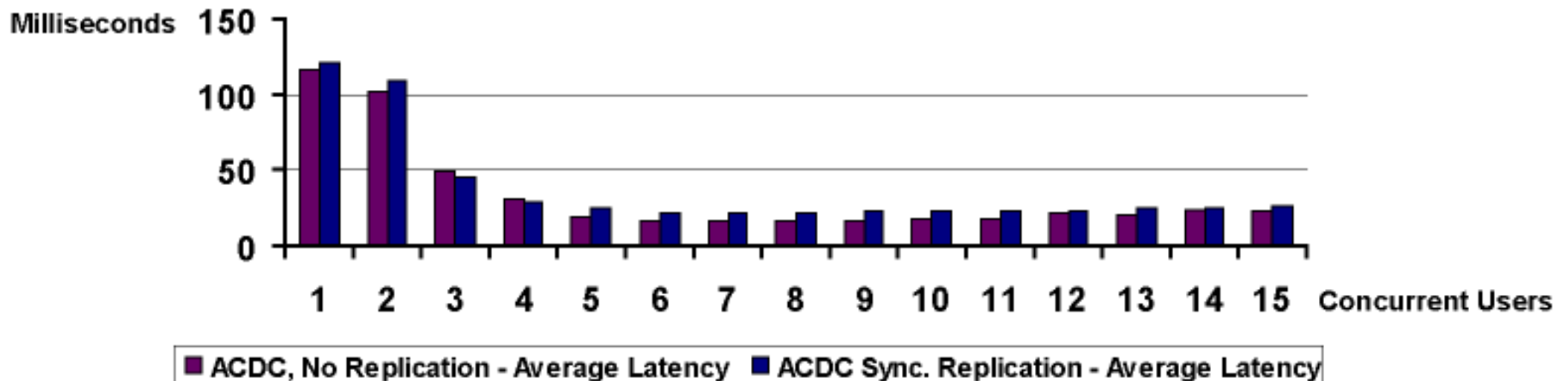
Throughput



ACDC Tests Results

No Replication vs. Synchronous Replication ACDC

Average Latency



Results with the synchronous replication show:

- The price for **partition tolerance**:
 - 14% average decrease of throughput
 - 17% average increase of latency

Conclusion

ACDC transactions run on Transactum Process Engine:

- Guarantee the **consistency**,
- Enhance the **partition tolerance**,
- Boost the **performance** with
One **order of magnitude**.

Interest for test-driving is welcome!

Thank You

Ivan Klianey

TRANSACTUM PTY LTD

SYDNEY, AUSTRALIA

WWW.TRANSACTUM.COM

