

SYNERGY DEVPARTNER CONFERENCE 2026

Rebuilding SQL Connection from the Ground Up

Michael Oleshchuk, Jr. Software Engineer

What is SQL Connection?

SQL Connection provides an API and database drivers that enable Synergy applications to access and manipulate relational databases.

Connects Synergy to relational databases

DBL programs connect to databases such as SQL Server through one API.

Works with your DBL data

Bind DBL variables to SQL statements and fetch results straight into DBL data areas.

Covers the full database workflow

Queries, inserts and updates, cursors, transactions, and stored-procedure calls.

%SSC_CONNECT **%SSC_OPEN** **%SSC_MOVE** **%SSC_COMMIT** **%SSC_RELEASE**

Why bring SQL Connection in-house?

SYNERGY DEVPARTNER CONFERENCE 2026

1

Long-term reliability

- Eliminate reliance on third-party driver vendors
- Ensure long-term stability and support for customers

2

Faster development and issue resolution

- Full source access enables faster diagnosis and bug fixes
- Deliver new features and improvements independently

3

Greater control and flexibility

- Own the development roadmap and technical direction
- Expand support to additional database platforms

A Modern Foundation for SQL Connection

SYNERGY DEVPARTNER CONFERENCE 2026

1

Native SQL Server connectivity

- Connect directly to SQL Server without SQL OpenNet or ODBC
- Removes deployment dependencies

2

Built with Rust

- Memory-safe design helps prevent common memory errors
- Modern foundation for long-term maintenance

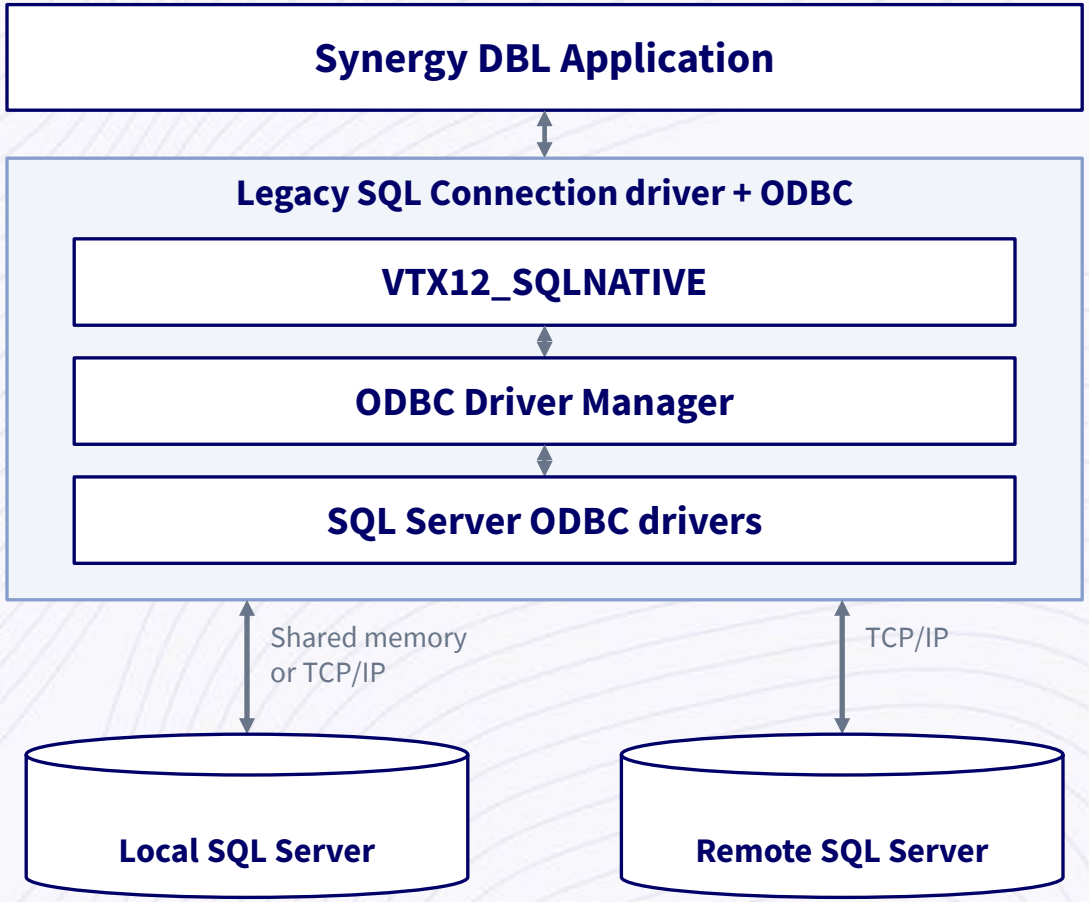
3

Backward compatibility

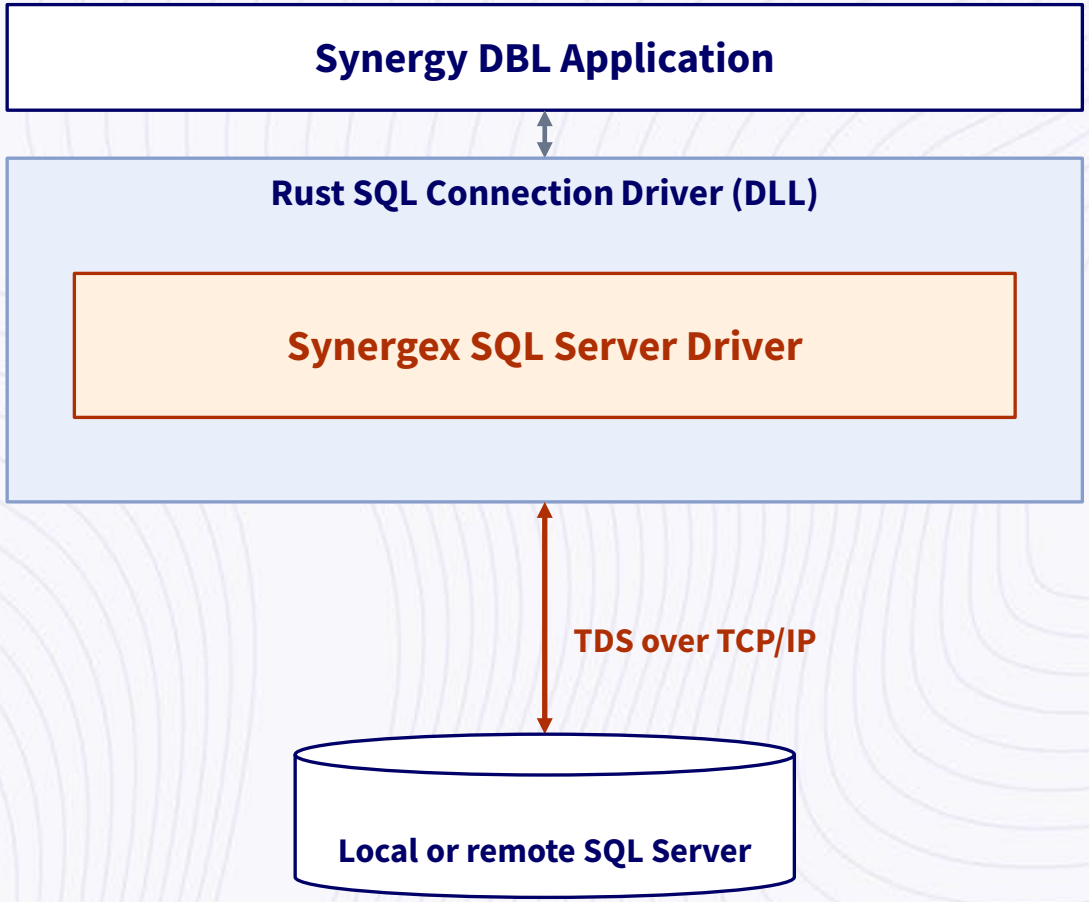
- Preserve the existing Synergy DBL API
- Minimize changes required in customer applications

SQL Server Connection Architecture

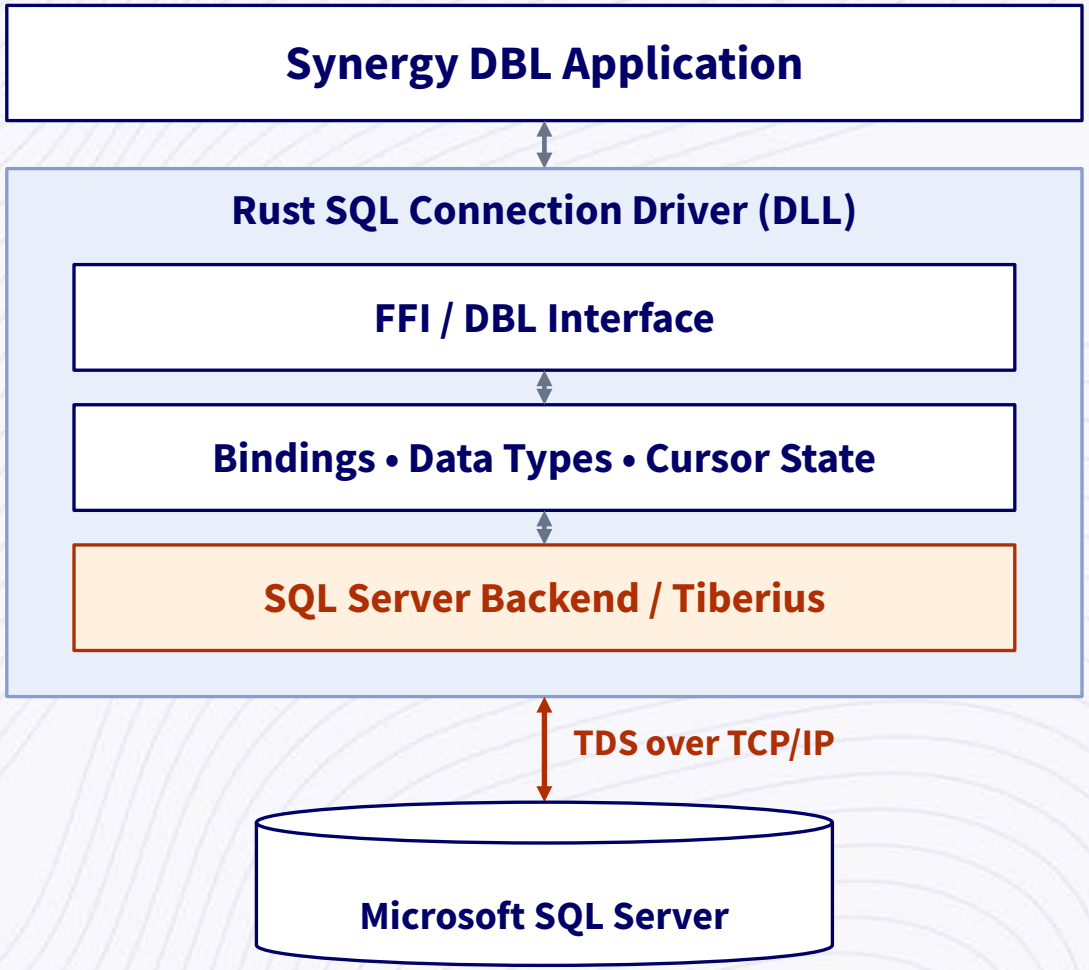
Legacy driver



New driver



Inside the New SQL Server Driver



- Calls the familiar SQL Connection API
- DBL runtime calls enter the DLL; buffers and status cross the boundary
- Preserves SQL Connection behavior: binds, conversions, metadata, cursors
- SQL Server-specific execution through our fork of Tiberius, a Rust TDS client
- Local or remote; no SQL OpenNet or ODBC layer

Requests travel down; results return up the same path.

DEMO

Sample application. New driver.

Challenge #1: Parameter Compatibility

Your application's SQL

SELECT name FROM customers WHERE id = **:1**

What the driver sends to SQL Server

SELECT name FROM customers WHERE id = **@P1**

Why this matters

The translation happens inside the driver. Your SQL text and bindings stay the same.
String literals, quoted identifiers, and comments are left untouched.

Challenge #2: Cursors Are More Than Queries

OPEN → DESCRIBE → FETCH → CLOSE

Applications depend on how every step behaves, not just on getting rows back.

What the driver must match

- Column names, types, precision, and nullability
- Row position and scrolling
- Cleanup of server and driver resources

Why it was hard

- Much of this behavior is not documented
- We wrote probe programs to observe the legacy driver
- Example: the cursor type changes which result sets a query returns

Challenge #3: Extending Tiberius

Tiberius → Synergex fork → SQL Connection driver

An open-source Rust client for SQL Server. Our fork adds what SQL Connection needs.

What we added

- Server cursors with scrolling
- Stored-procedure calls with output parameters and return status
- Commit and rollback sent the way legacy does
- Batched requests and a faster read path

Why it mattered

- Existing applications depend on cursor and procedure behavior
- The upstream library had no public API for either
- Closing a cursor now takes 1 round trip instead of 2
- Less CPU per row on large reads

How We Validate Compatibility

Same tests, both drivers, compare the results.

The legacy driver is the reference for expected behavior.

Matched probe programs

Same DBL and C programs on both drivers

SQL Server tracing

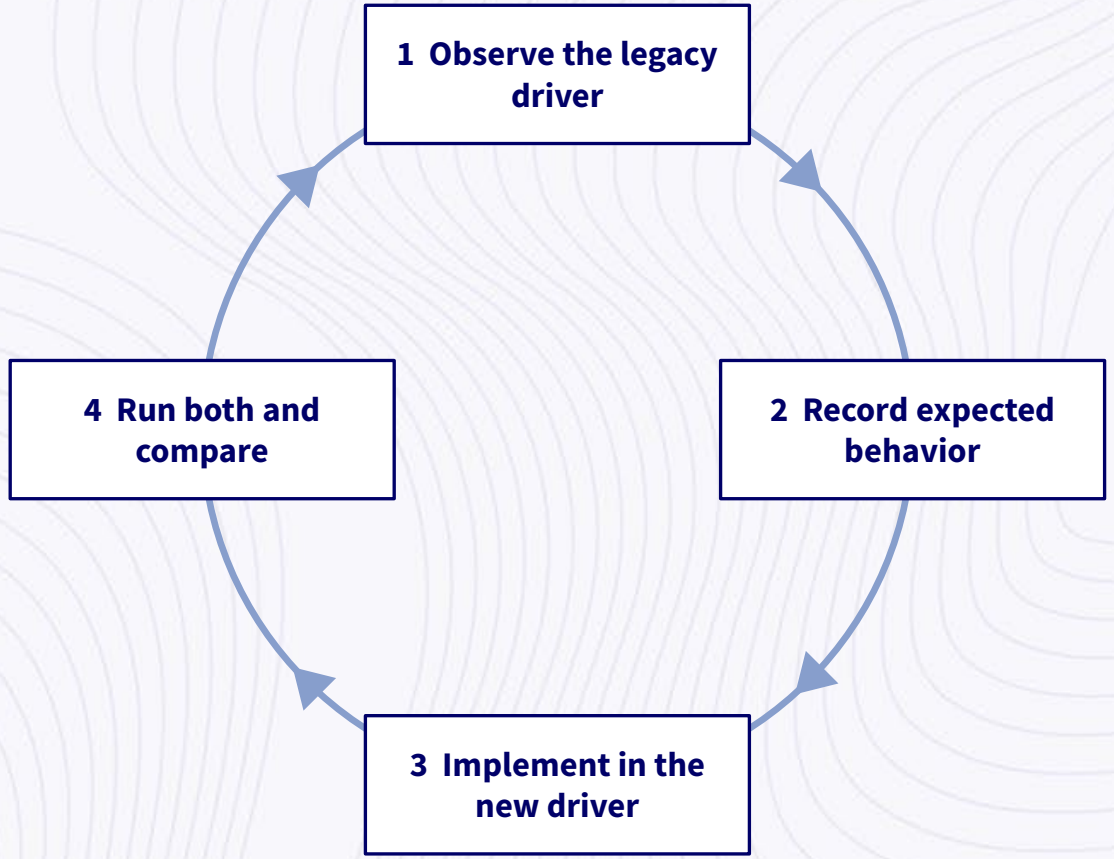
Compare what each driver sends

Existing test suite

Legacy harness tests run on the new driver

Rust unit tests

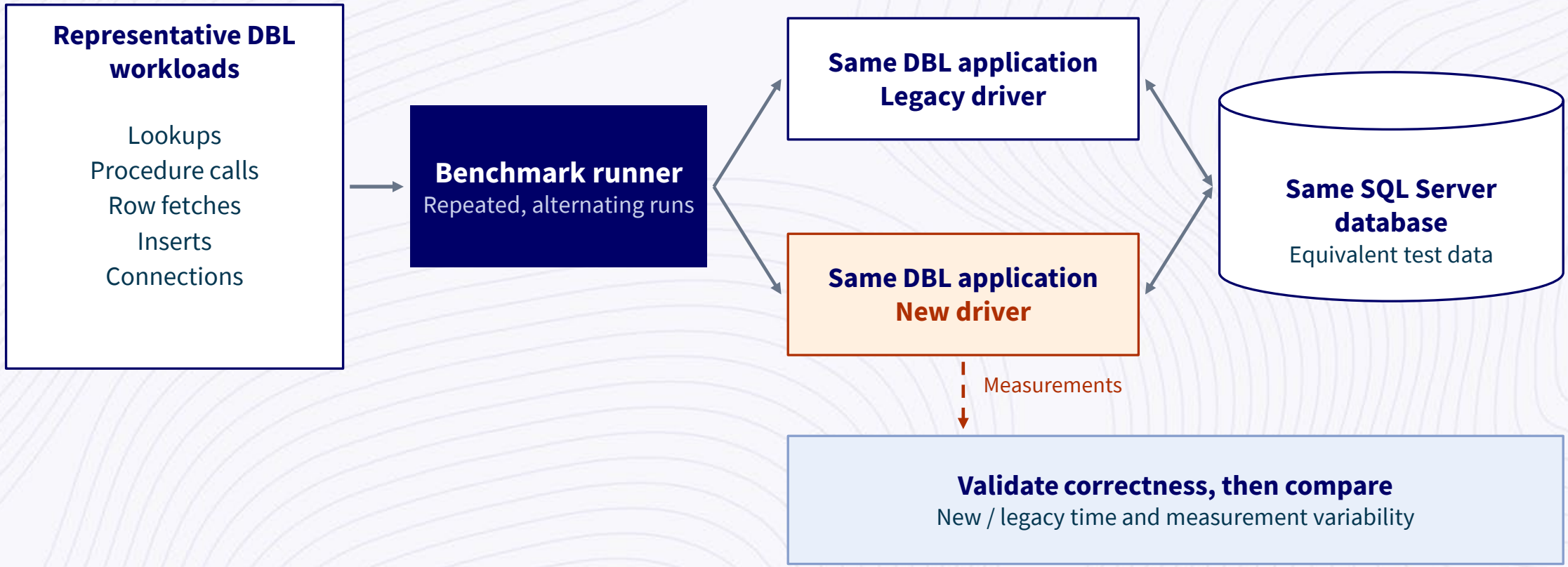
Driver logic checked without a database





So, performance: hit or miss?

How We Compared the Drivers



What Did We Benchmark?

Reads		Writes and connections	
1,000	lookups: number key, text key, cursor reused	**1,000**	inserts, commit after each
1,000	stored-procedure calls	**1,000**	inserts, one reused statement
100,000	rows, one per move	**1,000**	rows inserted in batches of 100
100,000	rows, 100 per move	**100**	connections opened and closed

Reports and lists: 2.4× faster



2.4× faster

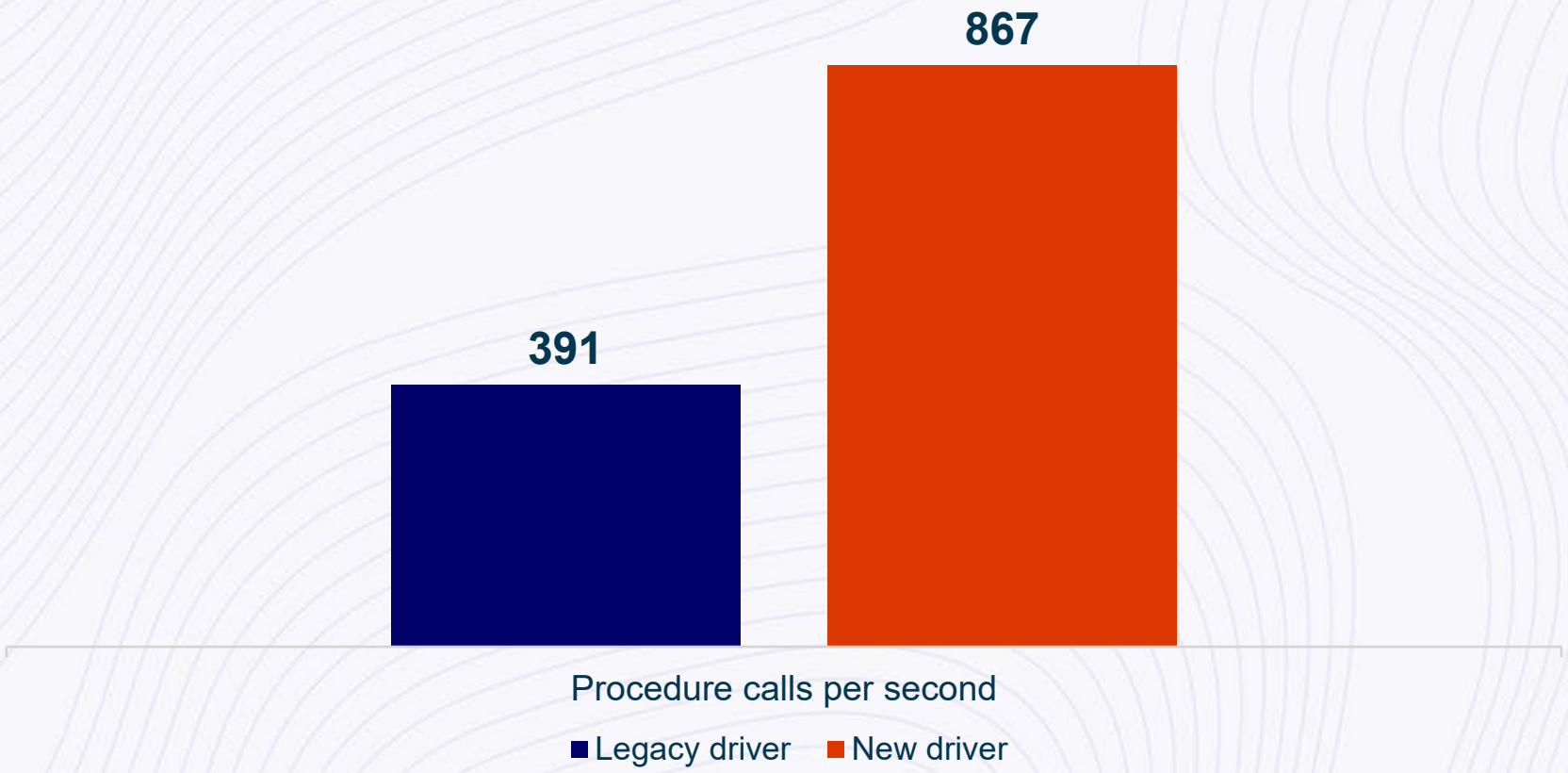
Nearly every application

0.8 s

instead of 1.9 s to read 100,000 rows

Elapsed seconds for the whole workload; lower is better.

Stored procedures: 2.2× faster



2.2× faster

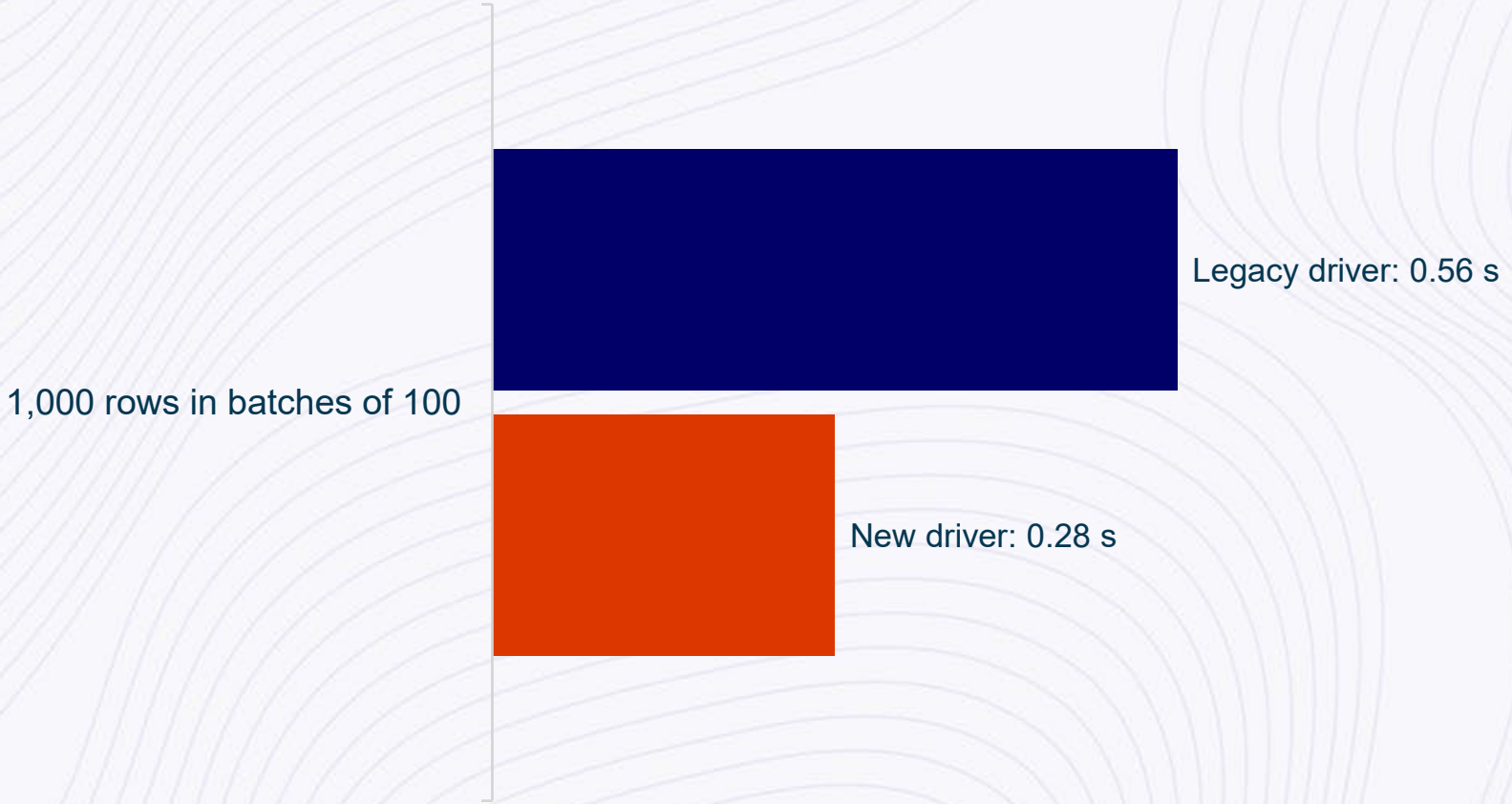
Procedure-based applications

867 calls/s

vs 391 calls/s today

Calls per second = 1,000 calls ÷ elapsed seconds; higher is better.

Batch inserts: 2× faster



2× faster

Data loads and syncs

0.28 s

instead of 0.56 s for 1,000 rows

Elapsed seconds for 1,000 rows; lower is better.

Everyday operations: as fast as today

Single lookups

Same speed as today, by number key or text key

Single inserts

As fast as today or slightly faster, with a commit after each

Prepared inserts

Same speed as today, reusing one prepared statement

Your workloads are the deciding test

Validate representative queries, latency, and transaction patterns

Migration: change the connection string

- Change your connection string to:

  sqlserver:user/pswd/hostIP/port

- The **sqlserver:** prefix selects the new driver
- Supply your own user, password, host, and port
- Most SQL Connection calls stay the same, but some applications may need code changes
- Follow the migration guide before you switch
- Save your previous connection string before switching

Compatibility during migration

Release and connection scope

SQL Server in 12.6

Additional databases are future plans

Driver selection

Your connection string selects the route

Direct connection setup

Confirm database login and TLS settings

Application regression checks

Values and bindings

NULLs, numeric and date conversions

Execution and results

Arrays, row counts, and SQLCODE

Cursor and transaction behavior

Scroll, updates, LOBs, commit and rollback

Confirmed differences will be published with their impact and a workaround.

What to Expect After Switching

Options and connections that now fail

- **SSQL_PERFORMANCE_SQL**
- **SSQL_CMD_SSL**
- **SSQL_CMD_NOUNICHAR** set to **no**
- **SSQL_CURSOR_TYPE** values other than 0–3
- **VTXIPC**
- **net:** (SQL OpenNet) connection strings are rejected
- Options the driver doesn't implement fail instead of being silently ignored

Errors and results

- SQL Server errors return their own code (**-208**, **-2627**) instead of **-1**
- A missing stored procedure returns the server's error (**-2812**) instead of **-1** "not found"
- Messages drop the ODBC driver prefix; driver failures name the routine
- No "Cursor type changed" warning on open

A Safe Migration and Rollback Process

SYNERGY 2026 DEVPARTNER CONFERENCE

1 **Save**

Save the existing connection string.

2 **Switch in test**

Switch to the sqlserver: driver in a test environment.

3 **Validate**

Validate representative application workflows.

4 **Compare**

Compare correctness and performance.

5 **Roll out or restore**

Roll out gradually, or restore the previous string and reconnect.

Support with implementation context

Trace failures through the source

For example, trace a connection error through the driver implementation.

Investigate and develop fixes directly

Our engineers can reproduce an issue, find its cause, and address it.

Deliver feature and security updates sooner

Source ownership lets us act directly on findings and customer needs.

SQL Server is Just the Beginning

FIRST RELEASE

SQL Server

Synergy/DE 12.6

64-bit Windows and Linux

FUTURE PLANS

Oracle
MySQL
Gantry / PostgreSQL

Release dates TBD

What This Means for Your Applications

1. The SQL Connection API you use today stays the same
2. Faster where it counts: reports 2.4×, procedures 2.2×, batch loads 2×
3. Direct SQL Server connectivity starts with Synergy/DE 12.6
4. Validate against the workloads that matter to your application

Questions?

Your applications and migration plans

Deployment requirements

Compatibility and diagnostics

Performance on your workloads

---

support@synergex.com