Identifying latency in a SQL Server-based application stack involves understanding where time is spent across **different layers** (app, network, database engine, storage, etc.). To isolate **"pure SQL Server time"**, you need to distinguish it from time consumed in other layers.

**✅ To identify pure SQL Server execution time (engine-level latency):**

These are the most effective methods:

**1. SQL Server Profiler / Extended Events**

Use these to capture the following metrics:

* CPU time: Time the query spent actively using CPU.
* Duration: Total time from start to end (includes waits).
* Reads/Writes: Logical and physical IO.
* **Event types**:
  + SQL:BatchCompleted
  + RPC:Completed
  + Showplan XML Statistics Profile (for query plan with run-time stats)

🔍 *Difference between Duration and CPU time helps identify waits inside SQL Server.*

**2. SET STATISTICS Commands**

Run this before executing a query:

sql

CopyEdit

SET STATISTICS TIME ON;

SET STATISTICS IO ON;

Output (in Messages tab):

* **SQL Server parse/compile time**
* **SQL Server execution time**
* Logical reads per table
* CPU time vs elapsed time

✅ Only shows SQL Server time, not network or application delays.

**3. Query Store**

If Query Store is enabled, use it to see:

* **avg\_duration**
* **avg\_cpu\_time**
* **execution\_count**
* Wait stats per query (if you use sys.query\_store\_runtime\_stats + wait stats views)

sql

CopyEdit

SELECT

qsrs.avg\_duration,

qsrs.avg\_cpu\_time,

qsqt.query\_sql\_text

FROM sys.query\_store\_runtime\_stats qsrs

JOIN sys.query\_store\_query qsq ON qsrs.query\_id = qsq.query\_id

JOIN sys.query\_store\_query\_text qsqt ON qsq.query\_text\_id = qsqt.query\_text\_id

ORDER BY qsrs.avg\_duration DESC;

**🔍 To compare with end-to-end latency (app/network)**

If you’re seeing high latency from the application side and want to isolate SQL Server’s responsibility:

* Use app logs or APM tools (e.g., AppDynamics, New Relic, Application Insights) to measure:
  + Time before/after DB call
  + Connection pool wait time
  + Network transmission time
* Compare with the SQL Server duration or execution time

**🧩 Breakdown by Layers**

| **Layer** | **Tool/Method** | **Metric** |
| --- | --- | --- |
| **Application** | App logs, APM | Request latency, connection open time |
| **Network** | TCP trace, latency monitor | Packet round-trip time |
| **SQL Server** | Profiler, STATISTICS TIME | CPU time, execution time |
| **Waits (inside SQL Server)** | sys.dm\_exec\_requests, sys.dm\_os\_wait\_stats | Wait types: I/O, CPU, locks |
| **Storage (IO latency)** | sys.dm\_io\_virtual\_file\_stats | Avg read/write latency per DB file |

**💡 Example: Checking Wait Stats**

To find SQL Server internal bottlenecks:

sql

CopyEdit

SELECT

wait\_type,

wait\_time\_ms,

waiting\_tasks\_count,

wait\_time\_ms / waiting\_tasks\_count AS avg\_wait

FROM sys.dm\_os\_wait\_stats

WHERE wait\_type NOT IN ('CLR\_SEMAPHORE','LAZYWRITER\_SLEEP','RESOURCE\_QUEUE','SLEEP\_TASK','SLEEP\_SYSTEMTASK')

ORDER BY wait\_time\_ms DESC;

**🔚 Summary**

To identify and isolate **pure SQL Server latency**:

* Use SET STATISTICS TIME ON for per-query insight
* Use **Profiler / Extended Events** or **Query Store** for systematic tracking
* Compare **CPU time vs Duration** to detect internal waits
* Use system DMVs (dm\_exec\_requests, dm\_os\_wait\_stats) to profile server internals