Simulation of quantum circuits with array-like DBMSs

An Empirical Case Study of SciDB versus Relational DBMSs

Bruno Faliszewski
Supervisors: Dr. Rihan Hai, Tim Littau

TU Delft

1. Motivation

Simulating quantum circuits on classical hardware is essential for developing and verifying quantum algorithms, while real devices stay noisy and scarce. The obstacle is scale: a state vector grows as 2^n, so memory quickly becomes the bottleneck. Recent work repurposes database systems as simulation backends, inheriting decades of query optimisation and out-of-core processing.

Because a quantum state is naturally a tensor, an array-oriented DBMS like SciDB looks like an even better match than a relational one. This work tests that intuition. Can SciDB outperform relational engines at quantum circuit simulation, and can automatic tuning make it faster?

2. MLOS

- MLOS: an open-source framework for automatic DBMS configuration tuning
- Treats tuning as black-box optimisation — no model of the database internals needed
- Iterative loop (see diagram): suggest config → apply → run workload → measure → learn
- Powered by the SMAC3 Bayesian optimiser to search the parameter space efficiently
- Here: used to tune SciDB (see 5. Database Autotuning)

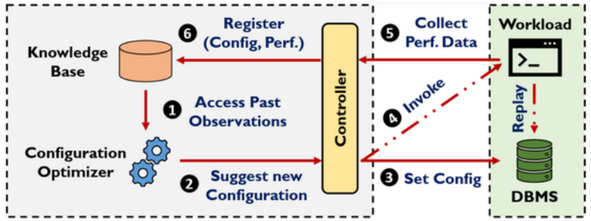


Figure 1: MLOS autotuning workflow. Reproduced from [1]

3. SciDB

- SciDB: a DBMS built on multidimensional arrays rather than tables
- Designed for scientific, analytical workloads over large, dense numeric data
- Quantum state vectors and gates map directly onto arrays - no relational encoding
- Reached via the SHIM HTTP gateway, which forwards queries to the engine (see diagram)

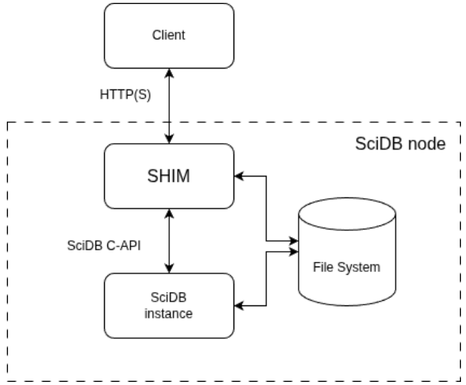


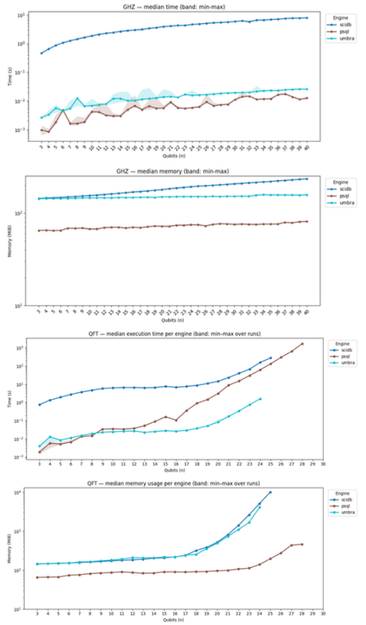
Figure 2: SciDB Architecture.
Adapted from the SHIM documentation [2]

4. SciDB vs. RDBMSs

Q: Does SciDB outperform relational engines (PostgreSQL, Umbra)?

- Benchmarked on two circuits: GHZ (a maximally entangled state) and QFT (Quantum Fourier Transform)
- SciDB is slower than both engines, on both circuits, at every qubit count
- GHZ: SciDB's time stays nearly flat — a fixed per-step overhead (~0.5 s); relational engines run in milliseconds
- QFT: the gap narrows as n grows, as that fixed overhead amortises over heavier computation
- Cause: per-query orchestration and array materialisation

A: No: SciDB does not outperform relational engines for in-core simulation.

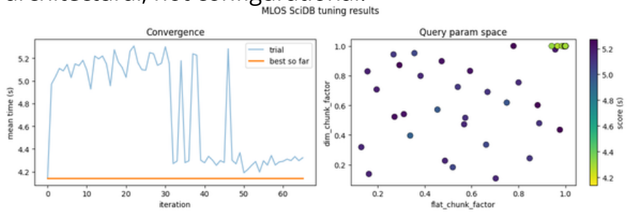


5. Database Autotuning

Q: Can MLOS tuning make SciDB competitive?

- Tuned SciDB's chunking parameters with MLOS + SMAC3, minimising execution time
- 60 trials exploring the parameter space (convergence shown left)
- No tuned configuration beat SciDB's defaults

A: No, tuning gives no speedup — the bottleneck is architectural, not configurational.



6. Conclusion

For in-core quantum circuit simulation, SciDB does not beat relational engines: a fixed per-query overhead dominates at the small sizes, and even though with bigger circuits it converges to RDBMSs, it never beats them.

Autotuning could not close the gap, the bottleneck is not caused by the tunable parameters. A big takeaway is that using chunk sizes that are smaller than the array itself significantly decreases performance.

7. References

[1] Microsoft. MLOS: A Project to Enable Autotuning for Systems. https://github.com/microsoft/MLOS.
[2] Paradigm4, Inc. shim: A Simple HTTP Service for SciDB. https://github.com/Paradigm4/shim. Documentation: http://paradigm4.github.io/shim/help.html.