How Fast Is VoltDB in Reality?

The TPS Pitfall

Why “how many transactions per second?” is the wrong first question — and what real-world PoC data tells us about VoltDB performance across telco, financial services, and autonomous network workloads.

> **Adam Majcher**
Presales Engineer, Volt Active Data
adam.majcher@voltactivedata.com

The Two Questions Everyone Asks

As a presales engineer at Volt Active Data, I hear the same two questions in almost every customer conversation:

> “Do you have a benchmark we can look at?”

> “How many TPS can VoltDB do?”

Both questions are completely reasonable. Both are also surprisingly tricky to answer well.

The Benchmark Problem

The most widely recognised database benchmarks, such as TPC-C, were designed in an era of spinning disks, row-level locking, and single-node architectures. They measure how well a database handles contention on a small number of hot rows with traditional lock-based concurrency control. That is exactly what VoltDB was designed to eliminate.

VoltDB uses a shared-nothing, partitioned architecture with serial execution per partition. There are no locks, no latches, no write-ahead logs in the traditional sense. Running TPC-C on VoltDB is a bit like testing a Formula 1 car on a cobblestone track — it will still be fast, but you are not measuring what makes it special.

The TPS Problem

“How many TPS?” sounds like a simple metric. It is not. In VoltDB, and in most SQL databases, one transaction can be fundamentally different from another:

- A simple SELECT of a single row by primary key? Extremely fast.
- An UPDATE of one row? Also fast.
- A stored procedure that deduplicates incoming events, runs aggregations, updates multiple materialized views, and operates on tables with 5M+ rows? That is a very different transaction.

Well-known benchmarks like TPC-C were designed for traditional disk-based databases with lock-based concurrency. They do not reflect the workload patterns where NewSQL databases like VoltDB deliver their strongest performance: high-throughput, low-latency streaming decisions on partitioned data.

On top of transaction complexity, infrastructure and configuration factors directly affect throughput and latency:

Factor	Impact	Why It Matters
K-factor (HA)	Adds Replicas Per Partition	K-factor=2 means 2 nodes can fail and the system keeps running, but every write is replicated to 2 additional copies, adding latency
Command Logging	Durability Overhead	Ensures recoverability after a full cluster failure; adds disk I/O to the write path
VoltDB Topics	Avoids extra Kafka round-trips	Using internal topics to deliver results to consumers avoids the latency of an external Kafka hop
Transaction Complexity	CPU Time Per Operation	More SQL statements, larger tables, more joins = more time per transaction
Table Sizes	Scan and Index Cost	Operating on tables with millions of rows is slower than tables with thousands
TPS vs. Latency Priority	Batching Trade-offs	Tuning for max TPS often increases p99 latency; tuning for low latency may reduce peak TPS

So when someone asks “how many TPS?”, the honest answer is: it depends on what the transaction does, how resilient the deployment needs to be, and whether you optimize for throughput or latency.

To give a practical answer, the following scenarios are drawn from recent PoCs and demos — examining real-world TPS per vCPU across multiple, realistic workload profiles.

The Scenarios

Four real-world scenarios drawn from recent proof-of-concept engagements. Each represents a different workload profile, infrastructure configuration, and performance priority.

SCENARIO 1

Mediation

Telco BSS/OSS | Event Processing | Deduplication & Enrichment

K-factor=1 | Command Log | VoltDB Topics | Dedup + Enrichment + Aggregation | 5M+ Subscriber Table

Mediation sits between network elements and downstream BSS/OSS systems in a telecom operator's stack. Raw usage events — CDRs, data session records, SMS logs — are generated at massive scale. Before they can be used for billing, fraud detection, or analytics, they must pass through the mediation layer.

The pipeline typically involves:

> **Collection** — ingesting raw events from multiple network sources (switches, gateways, probes)
> **Validation and Deduplication** — filtering malformed records and eliminating duplicates
> **Normalisation** — converting records from vendor-specific formats into a unified schema
> **Enrichment** — looking up subscriber details, rate plans, and service profiles
> **Aggregation** — combining partial records into complete usage events
> **Routing** — directing processed records to billing, fraud, analytics, or archival systems

VoltDB enables real-time mediation by processing each event as it arrives, with ACID guarantees on deduplication and enrichment lookups.

~5K	< 5ms	48	~240K
TPS per vCPU *incl. HA (k=1)*	**p99 Latency**	**Total vCPUs** *24 active + 24 replica*	**Total TPS**

This is a Complex Transaction

Each event triggers dedupe checks against a rolling window, enrichment lookups against a 5M+ row subscriber table, partial record aggregation, and materialized view updates. The ~5K TPS/vCPU already includes the replica cores in the count, reflecting the true hardware cost of a fully redundant deployment.

SCENARIO 2

Online Charging (5G OCS)

Telco BSS | ACID Transactions | Quota & Balance Management

K-factor=0 and 1 | No Command Log | Multi-table ACID Transactions | 500K Users Per Core

The online charging function is an ideal candidate for understanding the performance demands of 5G. Solutions must handle not only the load from billions of new devices, but also the complexity of enforcing different policies for an increasing diversity of devices.

This benchmark simulates a simplified online charging system where each transaction:

- Allocates quota to a subscriber, verifying sufficient balance across 4 tables with full ACID guarantees
- Adds new users and adjusts balances, joining data from multiple tables
- Runs conditional logic within Java stored procedures to achieve maximum work per round-trip

Tested on Google Cloud across cluster sizes from 3 to 27 nodes, on both 4-core and 16-core machines.

~7K	~3.5K	< 5ms	3.2M
TPS per vCPU (k=0) *active cores only*	**TPS per vCPU (k=1)** *incl. HA*	**p99 Latency**	**Peak TPS** *27x16-core cluster*

Key Finding

VoltDB demonstrates near-linear horizontal scaling. Throughput per vCPU remains remarkably consistent whether the cluster has 3 nodes or 27. The 27-node cluster with 16 cores per node achieved over 3.2 million operations per second with p99 latency under 5ms.

SCENARIO 3

Autonomous Network (ML Inference)

Telco Core Network | ML Decisioning | Self-Healing Infrastructure

K-factor=0 and 1 | VoltDB Topics | VoltSP (ML Inference) | 5M Endpoints | 300-350B Per Event

This scenario addresses autonomous network operations: ingesting real-time monitoring data from core network equipment, running ML model inference on each event, and deciding corrective actions for the network to self-heal — all within a single streaming pipeline.

Architecture uses the full VoltDB streaming stack:

- Kafka as the ingest source for network telemetry events
- VoltSP (Volt Stream Processor) for event validation and normalisation
- VoltDB stored procedures for stateful ML inference and action decisioning
- VoltDB Topics to deliver decisions back to consumers without an extra Kafka hop

Processes events from 5 million network endpoints, with each event carrying 300-350 bytes of telemetry payload. The ML model runs inside VoltDB using VoltSP's inline inference capabilities.

~9.3K	~4.5K	~73ms	280K
TPS per vCPU (k=0) *active cores only*	**TPS per vCPU (k=1)** *incl. HA*	**Avg E2E Latency** *k=0, full pipeline*	**Total TPS (k=0)** *30 VoltDB cores, 3 pods*

E2E Latency Includes the Full Pipeline

Kafka ingestion, VoltSP processing, ML inference, stored procedure execution, and result delivery via VoltDB Topics. With k=1, total hardware for the database doubles but if any VoltDB node fails, the cluster continues processing with zero data loss.

SCENARIO 4

VWAP — Volume Weighted Average Price

Capital Markets | Financial Calculations | ACID Precision

K-factor=1 | Command Log Disabled | ACID Financial Calculations | Materialized Views | Streaming Market Data

VWAP is one of the most commonly used benchmarks in algorithmic trading. It calculates the average price of a security weighted by trading volume over a specific time period. Every trade execution needs to update running VWAP calculations in real-time with absolute consistency — even a small rounding error or missed update can result in incorrect pricing and real financial loss.

Each transaction triggers:

- > Price and volume validation
- > Running VWAP recalculation with full decimal precision
- > Materialized view updates for multiple aggregation windows
- > ACID-guaranteed state updates across partitioned tables

Processes events from 5 million network endpoints, with each event carrying 300-350 bytes of telemetry payload. The ML model runs inside VoltDB using VoltSP's inline inference capabilities.

~6.2K	< 10ms	48	300K
TPS per vCPU *incl. HA (k=1)*	**p99 Latency**	**Total vCPUs** *3 VMs × 16 SPH*	**Total TPS**

The sub-10ms latency reflects VoltDB's strength in financial workloads where consistent low latency matters more than raw throughput. The ~6.25K TPS/vCPU reflects full ACID replication cost combined with the computational weight of maintaining running VWAP calculations across multiple aggregation windows on every trade event.

Putting It Together

Consolidated performance view across all four scenarios:

Scenario	TPS/vCPU (k=0)	TPS/vCPU (k=1, incl. HA)	p99 Latency	Complexity	Key Characteristics
ML Inference	~9.3K	~4.5K	~73ms E2E	Medium	Full pipeline: Kafka, VoltSP, ML model, Topics
Charging (5G OCS)	~7K	~3.5K	< 5ms	High	Multi-table ACID, balance checks, quota allocation
Mediation		~5K	< 5ms	High	Dedup, enrichment, aggregation, 5M+ row tables
VWAP (Financial)		~6.25K	< 10ms	High	Financial precision, materialized views, 48 vCPUs

Without HA: 7,000 – 9,300 TPS per active vCPU
With k=1 (full redundancy): 3,500 – 6,250 TPS per total vCPU

The ~50% drop from k=0 to k=1 is the price of guaranteeing that no node failure causes data loss or service interruption. Variation across scenarios is driven by transaction complexity, table sizes, and pipeline depth.

So How Do You Actually Compare?

Comparing database solutions from a buyer's perspective is inherently difficult. Every vendor can produce impressive benchmark numbers under conditions optimised for their architecture. Published benchmarks are useful for directional understanding, but they rarely reflect your specific workload, data model, infrastructure constraints, and SLA requirements.

The factors that matter in your evaluation may be very different from the factors that matter in someone else's:

- Do you need k-factor=2 for regulatory compliance, or is k-factor=0 acceptable?
- Is your priority maximum TPS or minimum p99 latency?
- How complex are your transactions? Simple key-value lookups or multi-step business logic?
- How large are your state tables? Thousands of rows or millions?
- Do you need end-to-end streaming with Kafka integration, or direct API calls?

The Best Approach for a Meaningful Comparison

1. Define your exact workload — schema, transaction logic, data volumes, concurrency patterns
2. Specify your infrastructure constraints — cloud provider, instance types, HA requirements
3. Set your SLAs — target latency percentiles, minimum throughput, availability requirements
4. Send these conditions to each vendor and ask them to demonstrate under your specific scenario
5. Compare apples to apples — same workload, same hardware, same SLAs

The Bottom Line

Do not compare marketing TPS numbers. Compare results from your own PoC scenario, run under your own conditions. That is the only benchmark that matters for your decision.

We'll run that with you.

Want to Benchmark VoltDB on Your Workload?

Let's define your scenario together and run a PoC that reflects your actual production requirements. Contact:

Adam Majcher
Presales Engineer, Volt Active Data
amajcher@voltactivedata.com