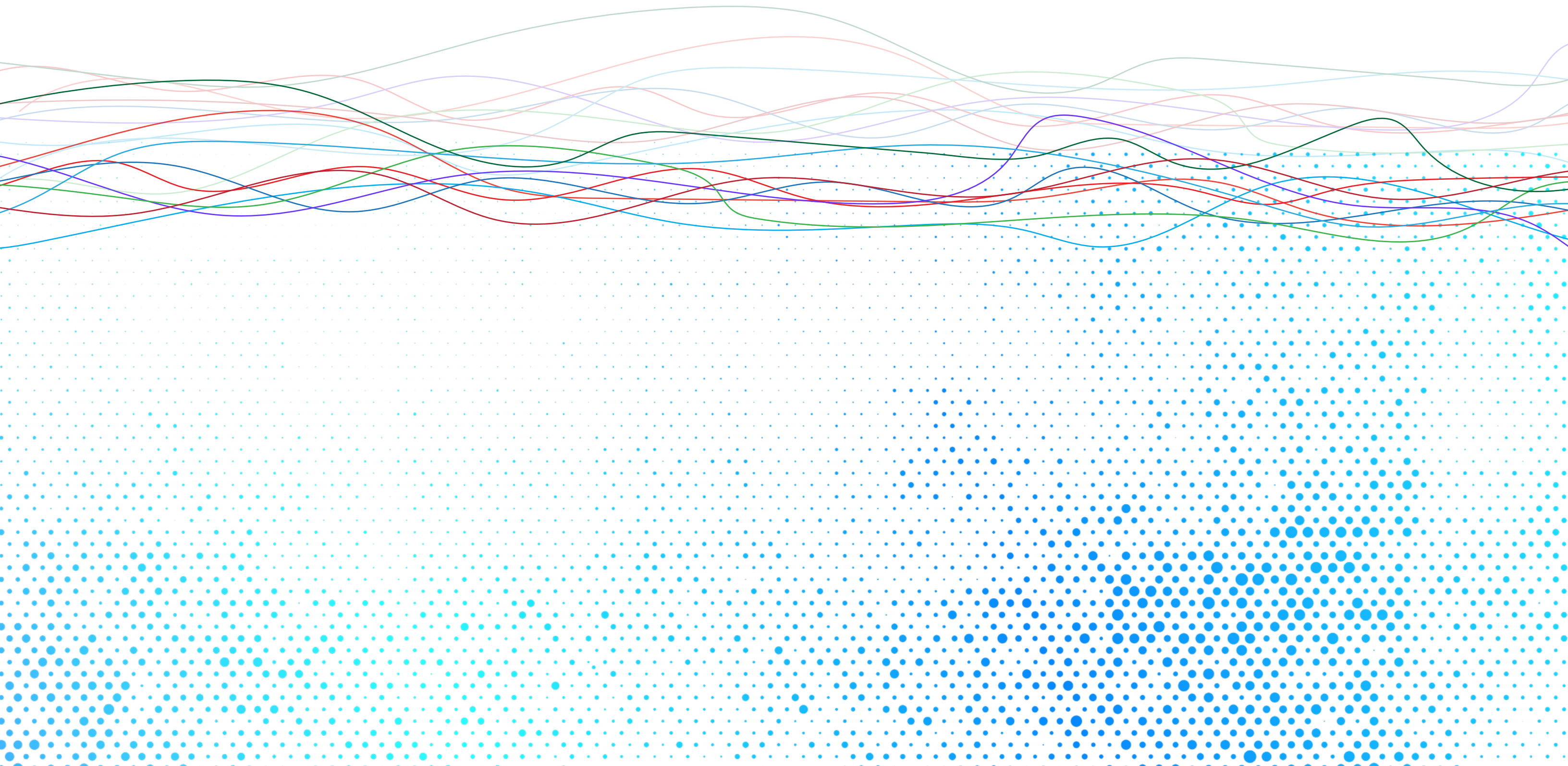
CockroachDB

# Database Benchmarking for the Real World

"Performance under Adversity" Benchmarking Guide



# “Performance under Adversity” Benchmarking Guide

Unlike traditional databases that highlight performance in ideal conditions, organizations are looking for databases that adapt to changing, real-world conditions to support their modern database infrastructure requirements. With this user guide, organizations can meet their performance and efficiency goals by shifting benchmarking from measuring ideal performance to simulating real-world conditions without sacrificing highly available and consistent data using CockroachDB.

Typical database OLTP benchmarks such as TPC-C usually require a minimum runtime of 1-2 hours to ensure the system’s stability and peak performance under heavy load. These benchmarks might typically simulate a company’s order processing system which may include order entry, payment processing, and status checks. The primary metric often used is throughput measured in transactions per minute (tpmC) and price/performance. Latency might be recorded for compliance but not heavily emphasized when used in a comparable configuration. While these benchmarks include durability tests, they often do not explicitly measure performance during failures or while under maintenance. Once the steady-state run is disrupted (for example by a power pull) the test is considered over.

Often organizations rely on application engineering teams and developers to supplement database benchmarking tools with chaos engineering tools to ensure applications are resilient in the face of changing, real-world conditions. Chaos tools such as Chaos Monkey can be used to test how a system responds to failure conditions and identify and fix failures before they become outages. Teams can schedule terminations during a given time period each day, based on a configurable mean number of days. However, most chaos engineering tools only target a single instance and do not simulate more complex failures such as loss of an entire availability zone (AZ) or region. They focus on a single point of failure and do not address the broader impact of cascading failures. Knowing how to deal with the broader real-life conditions on a given infrastructure is critical to assess the impact of resilience on sustained performance:

- Complexities of global applications and geo-distribution of data
- Planned administrative operations, such as rolling upgrades, online schema changes, or backup processes
- Recovery time after failures
- Inconsistent storage access
- Complex networking issues

The goal of this document is twofold: first, to enable readers using this guide to learn the details of a new, modern database benchmark methodology called “Performance under Adversity” and how it can help simulate real-world conditions and the impact to sustained performance. Second, to learn the results of applying this framework to CockroachDB, distributed across several geographically separate regions using a TPC-C workload over the course of ten hours under various real-world conditions. Additional details on how to set up and reproduce this benchmark using CockroachDB are outlined in a separate document.

# Methodology

The “Performance under Adversity” methodology extends traditional TPC-C benchmarking to include resilience metrics and failure scenarios within the performance measurement period. The failure scenarios include both hard and soft failures.

This benchmark uses a single continuous run composed of up to seven progressively harsher failure conditions. Failures are deliberately injected during the workload, and the performance during and after the fault is measured. Metrics comprehensively capture performance and resilience.

1. **Baseline performance:** Measure steady-state throughput under normal conditions.
2. **Internal operational stress:** Simulate database-intensive operations such as CDC, full backups, online schema changes, and rolling upgrades.
3. **Intermittent disk failures:** Randomly inject I/O freezes to simulate disk stalls and disk failures.
4. **Network failures:** Simulate partial and full networking failures preventing one partition from communicating with nodes in another partition.
5. **Node restarts:** Unpredictably reboot database nodes (1 at a time) to test recovery time.
6. **Zone outages:** For 5 minutes.
7. **Regional outages:** For 5 minutes.

Through all these phases, we measure throughput in transactions per minute (tpmC), latency (90th and 95 percentile), and recovery time to baseline. For each phase, the following is measured:

- **Transactions per minute (tpmC):** Calculated over the entire testing period, including adverse phases.  The goal is to achieve consistent high throughput.
- **Latency Metrics:** 90th, and 95th percentile commit times in milliseconds, captured as a detailed time series. The goal is to achieve consistent low latency.

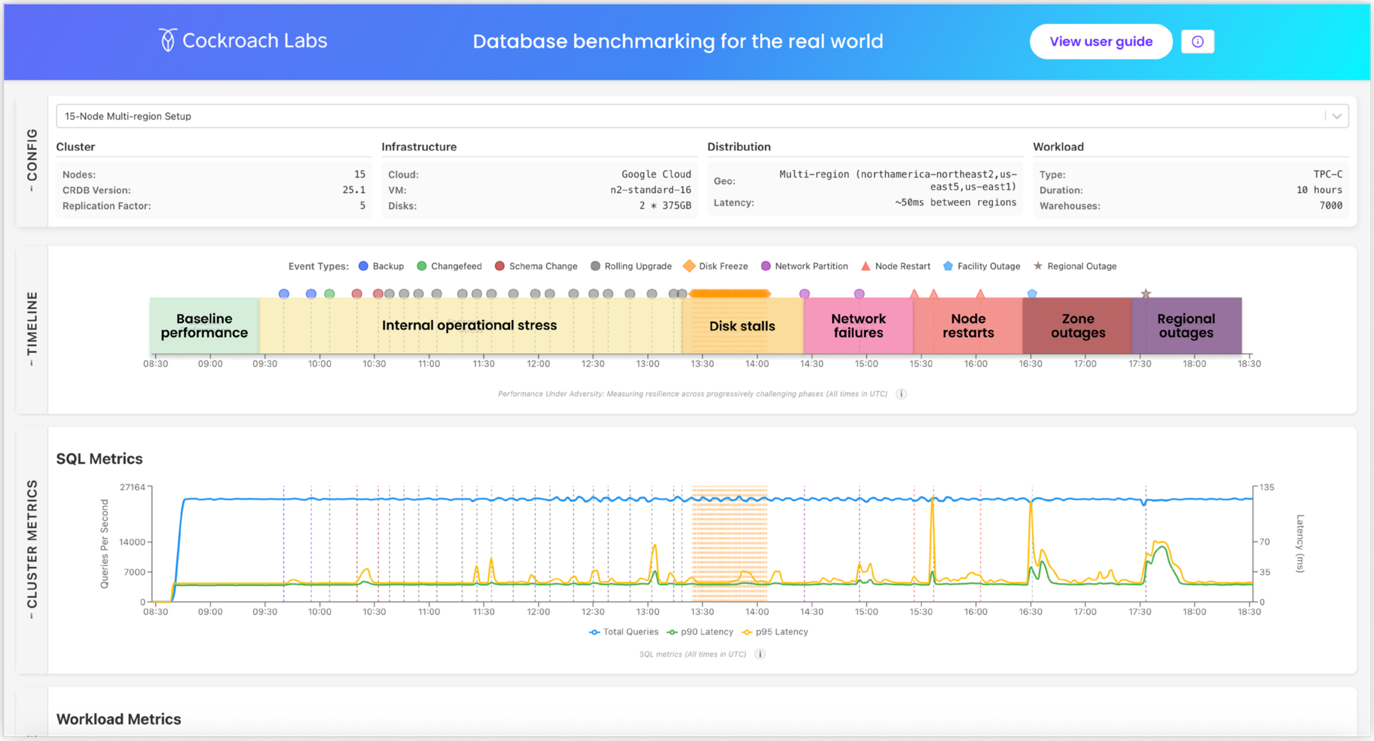
# Configuration

The test included two CockroachDB cluster setups:

- A 15-node, multi-region cluster (each node: 16 vCPUs, 64 GiB RAM, dual 375 GiB data storage disks), distributed across three regions running TPC-C 7K warehouses workload using 14000 connections. External proxy NLB routes traffic to the closest node via a single global IP.  Warehouse data is divided across three partitions. Tables are regional by row, with each region owning one partition.
- A 9-node, single region cluster (same node specs), distributed across three zones, running TPC-C 5K warehouses workload using 1800 connections. External proxy NLB routes traffic to the closest node via single global IP. Warehouse data is divided across three partitions.

# Interactive dashboard

To provide insight into cluster behavior during the test run, we've built an interactive dashboard that brings these benchmark results to life. This dashboard allows you to explore performance trends for each test phase and interact with the data directly, offering deeper insights into how CockroachDB behaves under stress.



Cockroach Labs — Database benchmarking for the real world

View user guide

CONFIG

15-Node Multi-region Setup

| Cluster | | Infrastructure | | Distribution | | Workload | |
|---|---|---|---|---|---|---|---|
| Nodes: | 15 | Cloud: | Google Cloud | Geo: | Multi-region (northamerica-northeast2,us-east5,us-east1) | Type: | TPC-C |
| CRDB Version: | 25.1 | VM: | n2-standard-16 | Latency: | ~50ms between regions | Duration: | 10 hours |
| Replication Factor: | 5 | Disks: | 2 * 375GB | | | Warehouses: | 7000 |

# Key features

- **Performance Graphs & Trends:** Dynamic charts for throughput and latency, captured during test runs. Users can zoom into specific time windows and hover to see exact metric values captured as the benchmark progressed.

- **Failure Simulations:** The ability to review each failure scenario (node restart, disk stall, zone outage, etc.) and analyze how the system responded at the time of testing.

- **Resilience Metrics:** Visual metrics highlighting recovery time, throughput variations, and availability during different failure events. These metrics quantify how quickly CockroachDB self-healed and the extent of any performance impact, providing a clear picture of overall resilience.

**Want to explore the data yourself?** Visit the interactive dashboard on our website to explore detailed graphs and failure scenarios. Our tool allows you to analyze how CockroachDB performed under stress and gain valuable insights from real-world benchmarking.

# Benchmark summary

| | 15-node multi-region cluster<br>7K Warehouses | 9-node single-region cluster<br>6K Warehouses |
|---|---|---|
| **Baseline performance** | ~60% CPU; ~87K tpmC;<br>Workload p95 latency: 180-200 ms;<br>SQL p95 latency: ~20 ms | ~50% CPU; ~62K tpmC;<br>Workload p95 latency: ~25-40 ms;<br>SQL p95 latency: ~3 ms |
| **Internal operational stress:** Rolling upgrades, backups, index creation, and continuous CDC | tpmC remained unaffected during internal operational stress without destabilizing the cluster. Brief latency spikes during some of the events.<br>Rolling upgrade over 165 minutes. Brief spike in latency: Workload p95 latency to 1.6 seconds and SQL p95 latency to 50 ms.<br>Backup over 20 minutes. For 3-6 minutes, spike in workload p95 latency to 220-240 ms and SQL p95 latency to 25 ms.<br>Changefeed has no impact.<br>During index creation, a brief spike in workload p95 latency to 730 ms, and SQL p95 latency to 40 ms. | Workload p95 latency remained unaffected during internal operational stress without destabilizing the cluster. Brief tpmC spikes during some of the events.<br>Rolling upgrade over 102 minutes with no impact on latency. Brief spike in tpmC and SQL QPS for 10 seconds.<br>Backup over 8 minutes. For 5 minutes, spike in workload p95 latency between 30 to 80 ms and SQL p95 latency to 7 ms.<br>Changefeed has no impact.<br>No impact on latency during index creation. |
| **Disk stalls:** 50 random disk stalls, 10 seconds each | Minimal latency impact and stable throughput without noticeable drops. | No impact on tpmC, workload p95 latency and SQL p95 latency during this phase. |
| **Network failures:** Partial and full network partitions | No latency or throughput impact during partial network partition. A minor latency spike to 1 second for 30 seconds and minimal throughput impacts during full network partition. | No latency or throughput impact during partial network partition. A minor latency spike to 5 seconds for 30 seconds and minimal throughput impacts. |
| **Node restarts:** 3 node-restarts in an hour at random intervals | A minor latency spike up to 6 seconds for 2 minutes and negligible throughput impacts | Minor latency spike to 2 seconds for 30 seconds and negligible throughput impacts. |
| **Zone outages:** Nodes in an entire availability zone (AZ) taken offline for 5 minutes | Minor latency spike to 4 seconds for 3 minutes and a temporary negligible drop in throughput during outage of a 2-node zone. The cluster maintained overall availability and stability throughout this event. | Minor latency spike to 3 seconds and temporarily a drop in throughput tpmC from 62K to 42K for 5 minutes; during outage of a 3 nodes zone. The cluster maintained overall availability and stability throughout this event |
| **Region outages:** Nodes in an entire geographic region taken offline for 5 minutes | Latency increased to approximately 2 seconds, without client errors or throughput drops, and recovery to baseline took around 26 minutes. The cluster remained stable and available throughout this event. | N/A |

# Conclusion

This user guide introduces and outlines a new benchmark methodology – “Performance under Adversity” – one that measures database resilience along with performance to reflect the realities of the cloud world where something is always failing.

“Performance under Adversity” uses a single continuous run composed of up to seven progressively harsher failure conditions and reports performance and latency under these failure conditions. Failures are deliberately injected during the workload, and performance during and after the fault is measured. Metrics comprehensively capture measures of resilience and impact to performance, and are displayed on an interactive dashboard that people can use to explore each test phase and interact with the data directly.

The test environment includes two CockroachDB cluster setups, one 9-node cluster and one 15-node cluster.

In each of the failure scenarios, CockroachDB performed exceedingly well, with negligible impact on throughput and with only brief and minor latency spikes. Both clusters maintained overall availability during the entire 12-hour run.

The outcome of this benchmark demonstrates CockroachDB’s state-of-the-art resilience while delivering highly consistent performance for a standard transactional workload.

Disclaimer: This guide and the results provided herein are solely for informational purposes to illustrate the performance of CockroachDB under specified stress test conditions, without any warranties of any kind. They do not constitute official product documentation or guarantee of future performance. Actual performance may vary based on system configurations, network conditions, workload, and other factors. Users are encouraged to perform their own independent evaluations of CockroachDB to ensure accuracy and suitability for their specific environment and requirements.