Amazon RDS for PostgreSQL 17.4

Kyle Kingsbury
2025-04-29

Amazon RDS for PostgreSQL is an Amazon Web Services (AWS) service which provides managed instances of the PostgreSQL database. We show that Amazon RDS for PostgreSQL multi-AZ clusters violate Snapshot Isolation, the strongest consistency model supported across all endpoints. Healthy clusters occasionally allow Long Fork and other G-nonadjacent cycles. These phenomena occurred in every version tested, from 13.15 to 17.4. Amazon RDS for PostgreSQL may instead provide Parallel Snapshot Isolation. This work was performed independently by Jepsen, without compensation, and conducted in accordance with the Jepsen ethics policy.

1 Background

PostgreSQL is a popular open source general-purpose SQL database. It uses multiversion concurrency control (MVCC) to provide three levels of transaction isolation. PostgreSQL's "Read Uncommitted" and "Read Committed" are both Read Committed. The "Repeatable Read" level actually provides Snapshot Isolation, not Repeatable Read. "Serializable" provides Serializability.

Amazon RDS for PostgreSQL is an AWS service which provides managed PostgreSQL clusters. RDS automates provisioning, storage management, replication, backups, upgrades, and more. Multi-AZ deployments distribute database nodes across multiple availability zones, reducing the probability of correlated failure. RDS uses synchronous replication to ensure that transactions are durable both on primary and (at least one) secondary instances before acknowledging.

From a user perspective, Amazon RDS for PostgreSQL provides a pair of URLs which speak the PostgreSQL wire protocol: a primary endpoint for read-write transactions, and a reader endpoint for read-only transactions. The primary endpoint supports all PostgreSQL isolation levels, while secondaries do not support Serializable. The strongest level supported across all nodes is therefore Snapshot Isolation (which PostgreSQL terms "Repeatable Read").

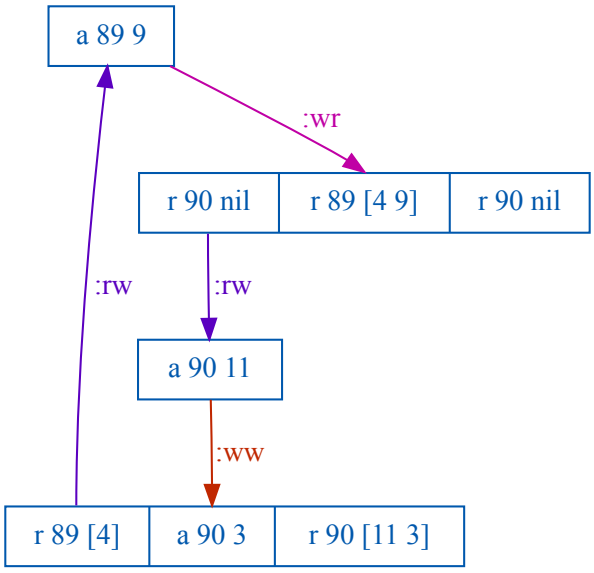
2 Test Design

We adapted Jepsen's test library for PostgreSQL for use with Amazon RDS for PostgreSQL with a small wrapper program. For each round of tests, we provisioned an RDS cluster using AWS's `CreateDBCluster` API, using `gp3` storage and `db.m6id.large` instances. We then launched a single EC2 node to run our tests, and provided it with the main and read-only endpoints of the RDS cluster. We performed no fault injection, and triggered no failovers.

As in our previous work on PostgreSQL, our primary workload consisted of transactions over lists of unique integers. We stored each list in a single row, encoded as a `TEXT` field of comma-separated values. Transactions could either read a list by primary key, or append a unique integer to a list using `CONCAT`. This workload allowed our Elle checker to verify a variety of isolation levels, mainly by inferring dataflow dependencies between transactions and finding cycles in the resulting graph.

3 Results

Under healthy conditions, with moderate concurrency, Amazon RDS for PostgreSQL 17.4 exhibited G-nonadjacent cycles every few minutes. Consider this two-minute test run, which performed approximately 150 write transactions per second, along with 1600 read-only transactions per second. It contains the following cycle of four transactions:



From top to bottom, call these transactions T_1, T_2, T_3, and T_4. T_1 appended 9 to row 89, resulting in the list `[4 9]`, which T_2 observed. T_3 appended 11 to row 90, resulting in the list `[11]`. That version was overwritten by T_4, which appended `3` to row `90`, and read the resulting list `[11, 3]`. While T_2 observed T_1's append to row 89, it failed to observe T_3's append to row 90. Symmetrically, T_4 observed T_3's append to row 90, but failed to observe T_1's append to 89.

Since this cycle includes read-write dependencies which are not adjacent to each other, this cycle is G-nonadjacent, a violation of Snapshot Isolation. This behavior should not occur in standard PostgreSQL at "Repeatable Read" and we have not observed it there.

To understand why this cycle is illegal, recall that in Snapshot Isolation, every transaction (apparently) operates on a snapshot of the database taken at some start timestamp s. That transaction's effects are made visible to others at some later commit timestamp c. In order for T_2 to read T_1's append, its start timestamp must have followed T_1's commit timestamp: $c_1 < s_2$. Since T_2 did not observe T_3's append, $s_2 < c_3$. Since T_4 overwrote (and observed) T_3, $c_3 < s_4$. But T_4 did not observe T_1's append, so $s_4 < c_1$. We have a contradiction! There is no way these timestamps can each precede each other.

This cycle is also an example of Long Fork. The first and second transactions compose one logical fork of the state. The third and fourth comprise a second. Each fork updates a different row, but neither fork observes the other's effects. Curiously, we did *not* observe Short Fork, also known as Write Skew. This suggests that Amazon RDS for PostgreSQL might provide Parallel Snapshot Isolation, a slightly weaker consistency model.

We observed a variety of G-nonadjacent anomalies, including those linked only by write-read edges, as well as several with more than four transactions. They occurred in every PostgreSQL version we tested, from 13.15 (the oldest version which AWS supported) to 17.4 (the newest).

4 Discussion

From the presence of Long Fork and other G-nonadjacent cycles, we conclude that Amazon RDS for PostgreSQL multi-AZ clusters do not ensure Snapshot Isolation. Instead, they may provide Parallel Snapshot Isolation, a slightly weaker model. In this respect Amazon RDS for PostgreSQL multi-AZ clusters offer weaker safety semantics than a single-node PostgreSQL system, which, in our previous testing, appeared to provide Strong Snapshot Isolation.

Users of Amazon RDS for PostgreSQL may wish to examine their transaction structures with an eye towards Long Fork, or design experiments to verify whether their intended invariants are preserved. A read transactions may disagree with other transactions as to the order in which transactions were executed. Since these anomalies appear to involve queries against read-only secondaries, it may be possible to recover Snapshot Isolation by only using the writer endpoint, or ensuring that every safety-critical transaction includes at least one write.

This report is the product of a cursory exploration—we have not investigated Amazon RDS for PostgreSQL behavior in detail. As always, Jepsen takes an experimental approach to safety verification: we can prove the presence of bugs, but not their absence. While we make extensive efforts to find problems, we cannot prove correctness.

Our thanks to Irene Kannyo for her editorial support. This work was performed independently by Jepsen, without compensation, and conducted in accordance with the Jepsen ethics policy.