**Auto Vacuum Common Issues and Resolutions:**

**How Autovacuum cost is calculated:**

Autovacuum reads pages looking for dead tuples, and if none are found, autovacuum discards the page.

When autovacuum finds dead tuples, it removes them. The cost is based on:

* vacuum\_cost\_page\_hit: Cost of reading a page that is already in shared buffers and doesn't need a disk read. The default value is set to 1.
* vacuum\_cost\_page\_miss: Cost of fetching a page that isn't in shared buffers. The default value is set to 2.
* vacuum\_cost\_page\_dirty: Cost of writing to a page when dead tuples are found in it. The default value is set to 20.

**Common Issues and Solutions:**

Issue: Autovacuum running slow:

1. Tables are getting vacuum slow and Vacuum process constantly appear in pg\_stat\_activity.

SELECT query FROM pg\_stat\_activity WHERE backend\_type = 'autovacuum worker';

Resolution :

1. maintenance\_work\_mem \Autovacuum\_work\_mem: Increase to allow each autovacuum worker process to store more dead tuples while scanning a table.
2. autovacuum\_vacuum\_cost\_delay:Decrease to reduce cost limiting sleep time and make vacuuming faster.
3. autovacuum\_vacuum\_cost\_limit: Increase the cost to be accumulated before vacuum will sleep, thereby reducing sleep frequency and making vacuum go faster.

(Good for Large number of Databases in Cluster).

1. autovacuum\_max\_workers Increase to allow more parallel workers to be triggered by autovacuum.

Issue: Autovacuum not happening enough.

1. SELECT relname, last\_vacuum, last\_autovacuum FROM pg\_stat\_user\_tables;

Resolution:

1. autovacuum\_vacuum\_scale\_factor Lower the value to trigger vacuuming more frequently, useful for larger tables with more updates / deletes.
2. autovacuum\_vacuum\_insert\_scale\_factor Lower the values to trigger vacuuming more frequently for large, insert-heavy tables.

Issue: Autovacuum is consuming too much system resource.

1. Spike in system resources memory/ Disk i-o
2. Slow other query performance.

Resolution:

1. Increase autovacuum\_vacuum\_cost\_delay and reduce autovacuum\_vacuum\_cost\_limit if set higher than the default of 200.
2. Reduce the number of autovacuum\_max\_workers if it's set higher than the default of

Issue: Vacuum does not clean up dead rows efficiently.

1. Tables are not getting vacuum properly and dead rows still show up.

Resolution:

1. Check for long running transaction which block vacuum process.
2. Termination long running transaction helps in freeing up dead tuples for deletion.
3. Query to check long running transaction.

SELECT pid, age(backend\_xid) AS age\_in\_xids,

now () - xact\_start AS xact\_age,

now () - query\_start AS query\_age,

state,

query

FROM pg\_stat\_activity

WHERE state != 'idle'

ORDER BY 2 DESC

LIMIT 10;

**Monitor Autovacuum:**

1)Find Auto vacuum is turned on or not:

SELECT name, setting FROM pg\_settings WHERE name='autovacuum';

2) Find how many dead rows in a table:

SELECT relname, n\_dead\_tup FROM pg\_stat\_user\_tables;

3) Find Track-Count is turned on or not (Enables collection of statistics on database activity)

SELECT name, setting FROM pg\_settings WHERE name='track\_counts';

4) Check Autovacuum is enabled on table level:

SELECT reloptions FROM pg\_class WHERE relname='Tablename';

5) Check parameter settings related with autovacuum.:

SELECT \* from pg\_settings where category like 'Autovacuum';

6) Find when was a table last vacuum/Auto vacuumed:

SELECT relname, last\_vacuum, last\_autovacuum FROM pg\_stat\_user\_tables;

7) To check progress of a running vacuum:

select \* from pg\_stat\_progress\_vacuum;

8) Dead Tuples percentage /Last Autovacuum.

select schemaname,relname,n\_dead\_tup,n\_live\_tup,round(n\_dead\_tup::float/n\_live\_tup::float\*100) dead\_pct,autovacuum\_count,last\_vacuum,last\_autovacuum,last\_autoanalyze,last\_analyze from pg\_stat\_all\_tables where n\_live\_tup >0;

9) Tables currently qualify for vacuum:

SELECT \*

,n\_dead\_tup > av\_threshold AS av\_needed

,CASE

WHEN reltuples > 0

THEN round(100.0 \* n\_dead\_tup / (reltuples))

ELSE 0

END AS pct\_dead

FROM (

SELECT N.nspname

,C.relname

,pg\_stat\_get\_tuples\_inserted(C.oid) AS n\_tup\_ins

,pg\_stat\_get\_tuples\_updated(C.oid) AS n\_tup\_upd

,pg\_stat\_get\_tuples\_deleted(C.oid) AS n\_tup\_del

,pg\_stat\_get\_live\_tuples(C.oid) AS n\_live\_tup

,pg\_stat\_get\_dead\_tuples(C.oid) AS n\_dead\_tup

,C.reltuples AS reltuples

,round(current\_setting('autovacuum\_vacuum\_threshold')::INTEGER + current\_setting('autovacuum\_vacuum\_scale\_factor')::NUMERIC \* C.reltuples) AS av\_threshold

,date\_trunc('minute', greatest(pg\_stat\_get\_last\_vacuum\_time(C.oid), pg\_stat\_get\_last\_autovacuum\_time(C.oid))) AS last\_vacuum

,date\_trunc('minute', greatest(pg\_stat\_get\_last\_analyze\_time(C.oid), pg\_stat\_get\_last\_autoanalyze\_time(C.oid))) AS last\_analyze

FROM pg\_class C

LEFT JOIN pg\_namespace N ON (N.oid = C.relnamespace)

WHERE C.relkind IN (

'r'

,'t'

)

AND N.nspname NOT IN (

'pg\_catalog'

,'information\_schema'

)

AND N.nspname !~ '^pg\_toast'

) AS av

ORDER BY av\_needed DESC ,n\_dead\_tup DESC;