Unexpected increase in the storage consumption

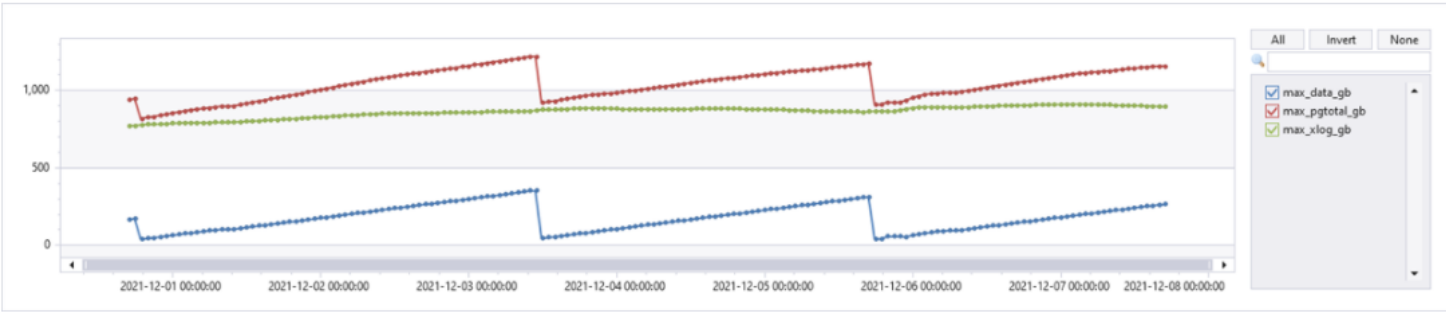
Last updated by | Hamza Aqel | Mar 2, 2022 at 2:57 AM PST

This TSG is part of GT for any change please contact haaqel@microsoft.com

Customer sometimes are complaining that the current storage utilization is not reflecting the actual database size, one of the main reasons that the WAL files are still on his server storage , to check what is exactly is going on , please check the below :

Check the storage utilization:

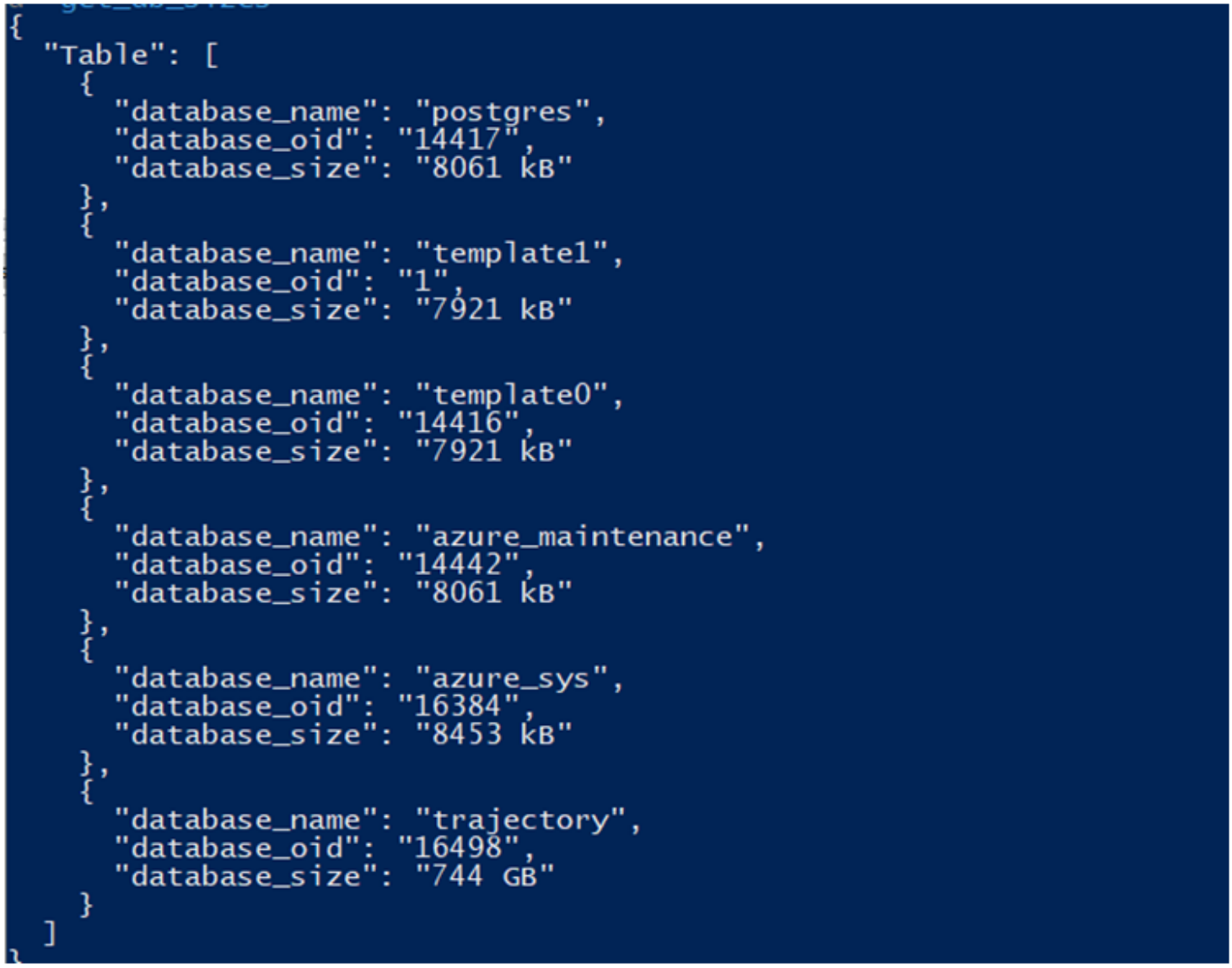
```
MonRdmsInstanceAgent
| where originalEventTimestamp > ago(7d)

| where LogicalServerName == "pgservername"
| where message_systemmetadata !contains 'Fileshare
directory' and message_systemmetadata contains '[PostgresStorageMetricsCollector].RefreshMetric:'
| project originalEventTimestamp
, ClusterName, AppName, AppTypeName, LogicalServerName, SourceNamespace, SourceMoniker
, xlog_gb = tolong(extract(' xlog:([ ])([-0-9]+)',
2, message_systemmetadata, typeof(real)))/(1024*1024*1024)
, data_gb = tolong(extract(' data:([ ])([-0-9]+)',
2, message_systemmetadata, typeof(real)))/(1024*1024*1024)
, pgtotal_gb = tolong(extract(' pgtotal:([ ])([-0-9]+)',
2, message_systemmetadata, typeof(real)))/(1024*1024*1024)
| summarize max(data_gb),max(xlog_gb),max(pgtotal_gb) by bin(originalEventTimestamp, 1h)
| render timechart
```



The above shows that total size (red) consists of PG data (blue) and PG WAL files (green). Majority of space is consumed by WAL files.

PG data seems to go up and down, executing get_db_sizes (CAS command) gives:

Invoke-ElasticServerDbCommand -SubscriptionId xxxxxxxx-xxxx-xxxxxxxx-xxxxxxxxxxxx -ResourceGroup RGNAME -ServerName pgservername -ServerType PostgreSQL.Server.PAL -command "get_db_sizes"



Now lets check the WAL files status , in PG single server we have archiver process to copy these WAL files , to get more details , use the below :

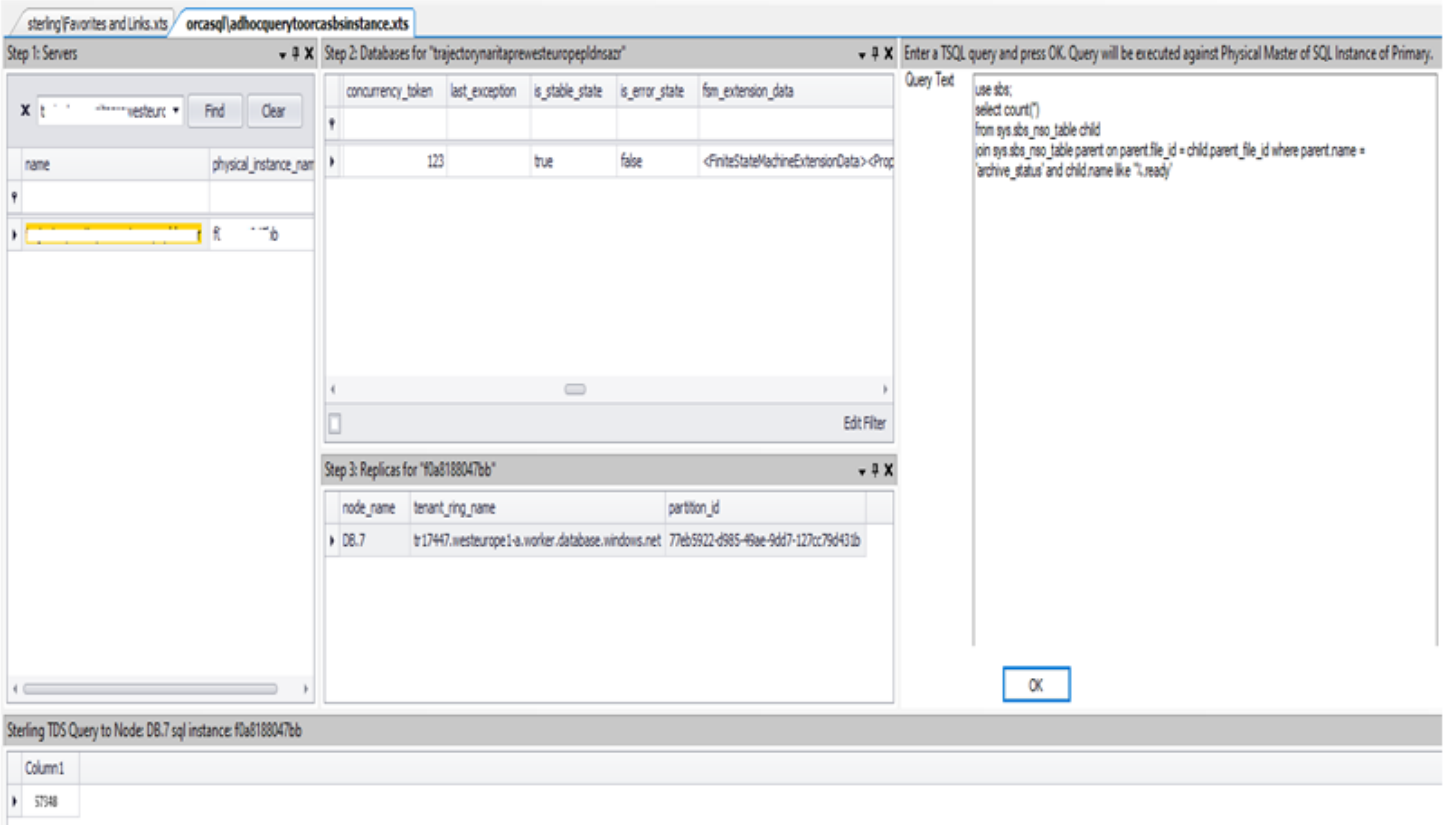
Use CMS query (adhocquerytoorcasbsinstance.xts) and run the below query :

use sbs;

select count(*)

from sys.sbs_nso_table child

join sys.sbs_nso_table parent on parent.file_id = child.parent_file_id where parent.name = 'archive_status' and child.name like '%.ready'



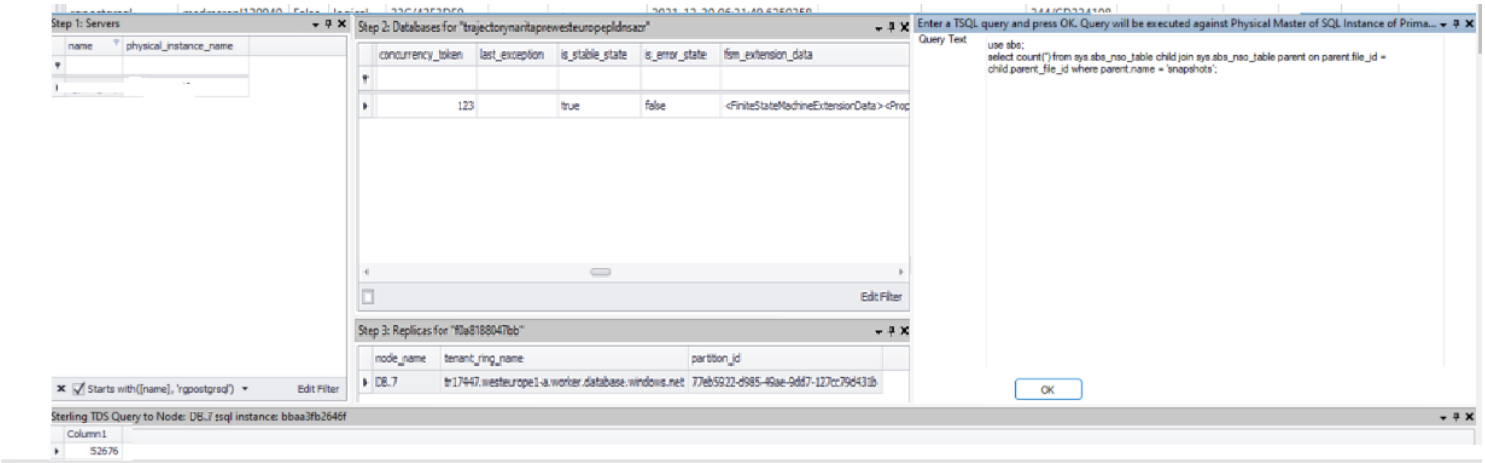
and to check the number of snapshots generated which will impact the checkpoint time or the recovery process , you can use the same CMS view:

use sbs;

select count(*)

from sys.sbs_nso_table child

join sys.sbs_nso_table parent on parent.file_id = child.parent_file_id where parent.name= 'snapshots'



There are 57348 WAL files in 'ready' state, i.e. WAL files that the archiver has yet to process. Each WAL file is 16MB, which accounts for almost 900GB. High number of 'ready' WAL files indicates that archiver is not able to keep up with the load or we have an issue in the WAL archiver, to check the wal archiver status use the below Kusto queries :

```
MonRdmsPgSqlSandbox

| where TIMESTAMP > ago(5d)
| where LogicalServerName == "trajectorynaritaprewesteuropepldnsazr"
| where text contains "[info] xlogcopy.uploadXlogFile: success"
| summarize count()
```

count_
52147

```
MonRdmsPgSqlSandbox

| where TIMESTAMP > ago(5d)
| where LogicalServerName == "pgservername"
| where text contains "[info] xlogcopy.uploadXlogFile: start to upload"
| summarize count()
```

count_
52198

From the above results we can see that the archiver is working as expected and don't forget

to correlate these WAL files with the backup retention the customer has. if you

notice any high discrepancy in the above, please file an ICM to take actions.

From the customer side he can use the below queries to check:

Status of WAL archiver:

```
SELECT * FROM pg_stat_archiver;
```

Current WAL file:

```
SELECT pg_walfile_name(pg_current_wal_lsn());
```

This should show the difference between current WAL file and last archived WAL file, and could be used to determine how far WAL archival process has fallen behind:

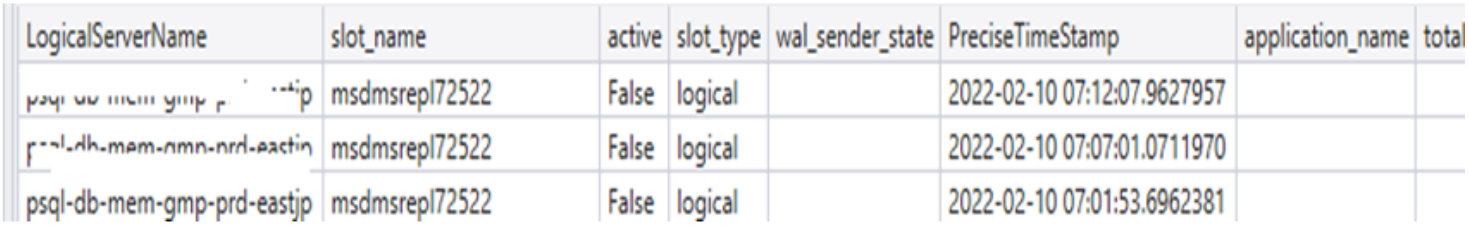
```
select pg_walfile_name(pg_current_wal_lsn()),last_archived_wal,
  ('x'||substring(pg_walfile_name(pg_current_wal_lsn()),9,8))::bit(32)::int*256 +
  ('x'||substring(pg_walfile_name(pg_current_wal_lsn()),17))::bit(32)::int  -
  ('x'||substring(last_archived_wal,9,8))::bit(32)::int*256 -
  ('x'||substring(last_archived_wal,17))::bit(32)::int
  as diff from pg_stat_archiver;
```

As each WAL file has a size of 16MB the "diff" column also allows to determine the storage size impact.

Check if there is any inactive replication slot:

Another main reason for storage increase is the replication slot, inactive replication slot can reserve the WAL files and impacting the storage growth, to check use the below Kusto queries:

```
MonDmPgSqlReplicationStatsPrimary

| where LogicalServerName == "psql-db-mem-gmp-prd-eastjp"

| project  LogicalServerName, slot_name, active, slot_type, wal_sender_state, PreciseTimeStamp, application_name, total_lag_in_bytes, receiving_lag_in_bytes, replaying_lag_in_bytes, sent_lsn, current_wal_lsn,restart_lsn, sending_lag_in_bytes, write_lsn, write_lag,  flush_lsn, flush_lag, replay_lsn, replay_lag

| order by PreciseTimeStamp desc
```



LogicalServerName	slot_name	active	slot_type	wal_sender_state	PreciseTimeStamp	application_name	total
psql-db-mem-gmp-p...-eastjp	msdmsrepl72522	False	logical		2022-02-10 07:12:07.9627957		
psql-db-mem-gmp-prd-eastjp	msdmsrepl72522	False	logical		2022-02-10 07:07:01.0711970		
psql-db-mem-gmp-prd-eastjp	msdmsrepl72522	False	logical		2022-02-10 07:01:53.6962381		

And from the customer side he can check using this query :

```
select * from pg_replication_slots where slot_type='logical' and active=false;
```

once he has the output , he can run below query to drop the replication slot.

```
SELECT pg_drop_replication_slot('slot name from the above query');
```

And you can monitor the storage consumption after that.