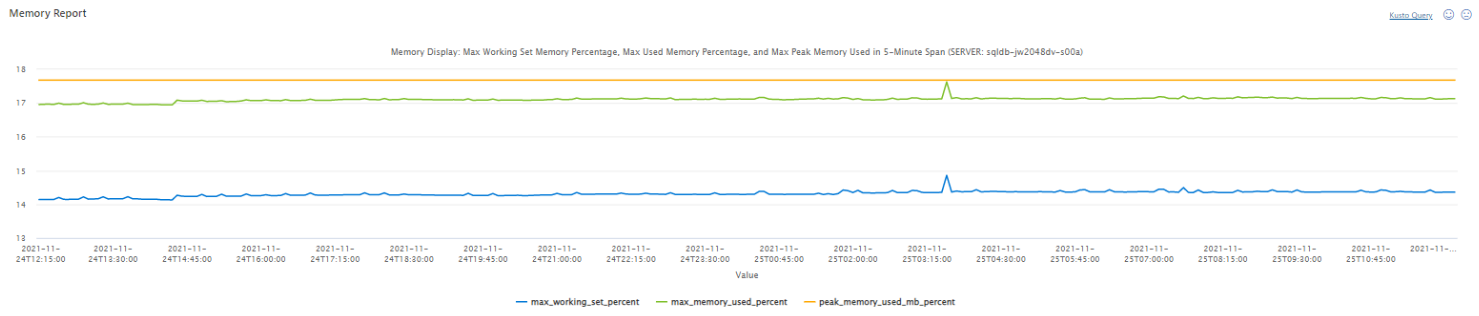
# High CPU utilization

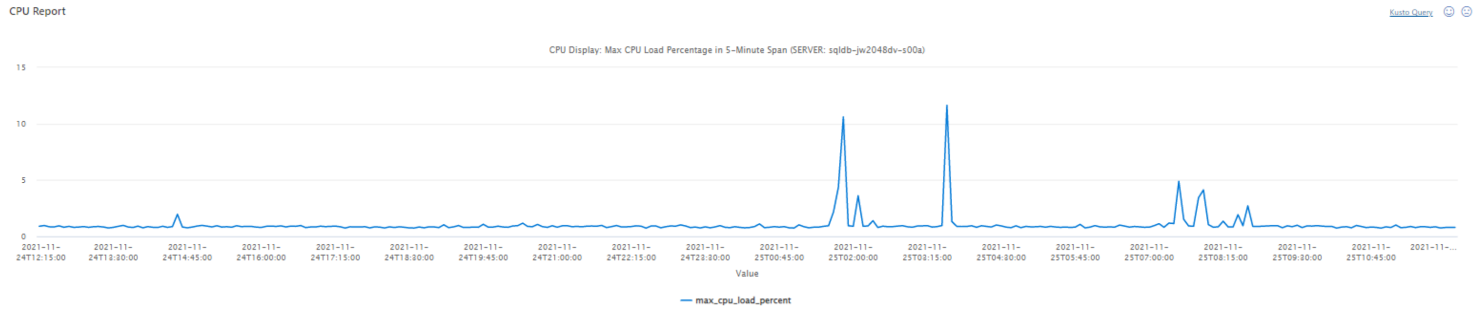
Last updated by | Hamza Aqel | Feb 22, 2022 at 8:22 AM PST

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

Customers might report that there is a high CPU utilization on their server more than usual , so you can use the below steps to help you investigating this issue :
1- Checking CPU usage from ASC :

Check customer resources (CPU), you can do that from our ASC (Perf tab) to see if there is any resource utilizations :



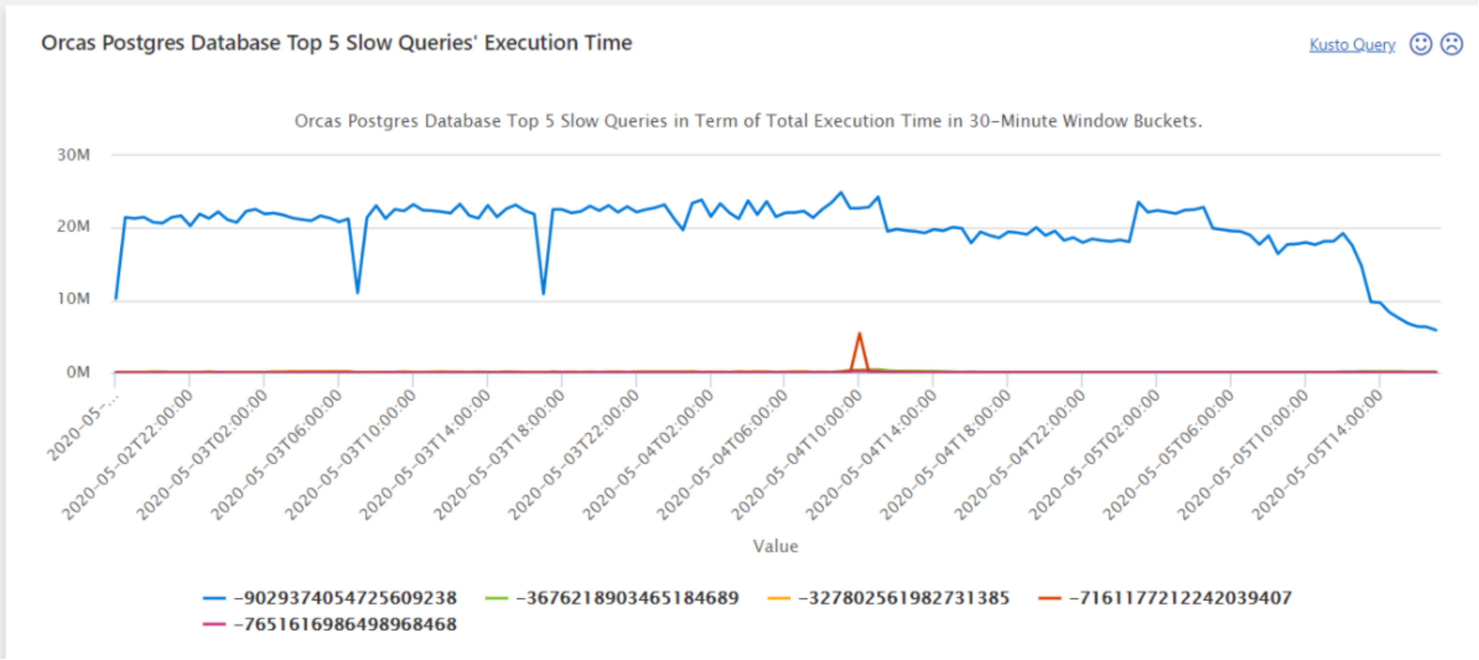


if you notice a high CPU usage please check the below :

**a- check the customer workload** :

check if the customer workload has been increased compared with previous period , you can refer to this TSG to get this information, the customer can leverage the Query Store to determine which queries are taking the longest. If after optimizing, the long running query's CPU usage is still high, consider scaling up to the next vCore tier. For example, if the CPU usage is hovering around 100 percent consistently for General Purpose 4 vCore, scale up to a General Purpose 8 vCore.

you can check the customer query store if it is already enabled from the same ASC Perf tab:



Orcas Postgres Database Top 5 Slow Queries' Execution Time

Orcas Postgres Database Top 5 Slow Queries in Term of Total Execution Time in 30-Minute Window Buckets.

and correlate this result with the High CPU utilization and you can share these queries with the customer.

if the customer did not enable the query store , you can use the below queries from pg_stat_statements from the customer side :

**CPU (long running queries):**

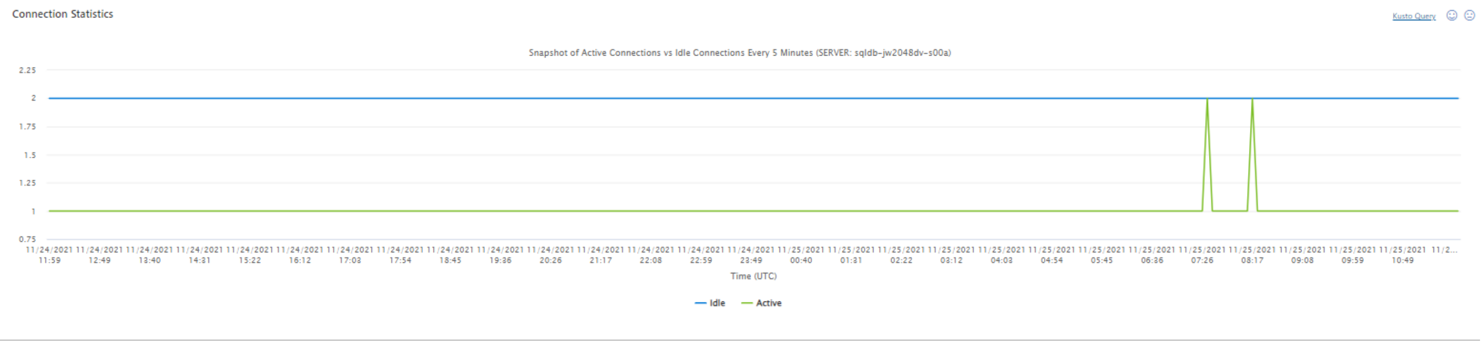
SELECT substring(query, 1, 50) AS short_query, round(total_time::numeric, 2) AS total_time, calls, round(mean_time::numeric, 2) AS mean, round((100 * total_time / sum(total_time::numeric) OVER ())::numeric, 2) AS percentage_cpu FROM pg_stat_statements ORDER BY total_time DESC LIMIT 20;

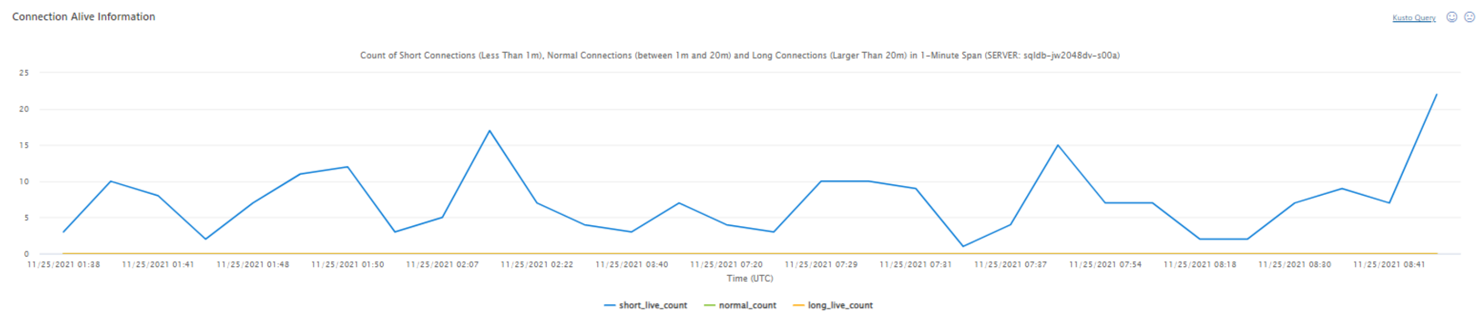
and if the issue still occurring , it is worth to run pg_stat_activity from customer side too to check what are the current active sessions running to get more details :

**SELECT pid, user, client_addr, pg_stat_activity.query_start, now() - pg_stat_activity.query_start AS query_time, query, state, wait_event_type, wait_event FROM pg_stat_activity WHERE state='active'**

**b- check connectivity patterns :**

you can check the connectivity from our ASC (connectivty tab) and correlate with high CPU usage:





and make sure that the customer is not creating too many short live connectoions , if that the case , please share with the customer that We highly recommend you use connection pooling, especially if you run into the following scenario:

1- If there are many concurrent, short duration connections. For example, an application with a high level of concurrency creates a connection for each interaction with the database, runs a simple query, and then closes the connection.

2- If there is sudden influx of new connections. For example, when the database server is restarted by users, when the scheduled maintenance is completed, or at the start of peak business hours.

Azure Database for PostgreSQL-Flexible Server offers PgBouncer as a built-in connection pooling solution. See PgBouncer in Azure Database for PostgreSQL - Flexible Server