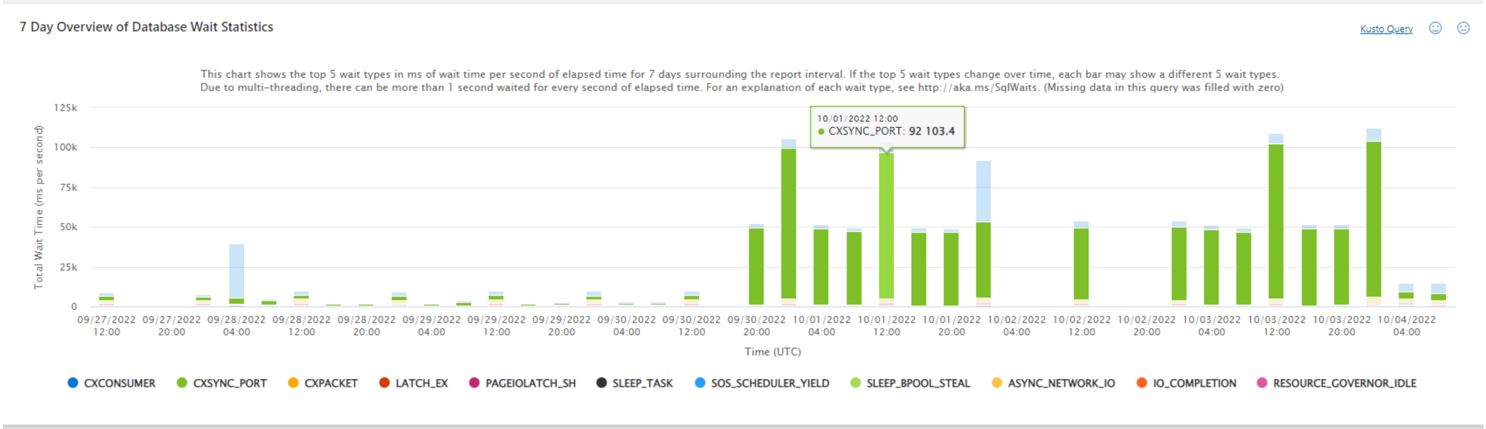
Performance issue due to CXSYNC_port waits

Last updated by | Vitor Tomaz | Feb 24, 2023 at 3:28 AM PST

Contents

Issue

The customer reported performance issues and timeouts for their queries. While checking the performance parameters on the ASC Troubleshooter, you notice a massive amount of `CXSYNC_PORT` waits on the customer database:



7 Day Overview of Database Wait Statistics

This chart shows the top 5 wait types in ms of wait time per second of elapsed time for 7 days surrounding the report interval. If the top 5 wait types change over time, each bar may show a different 5 wait types.
Due to multi-threading, there can be more than 1 second waited for every second of elapsed time. For an explanation of each wait type, see http://aka.ms/SqlWaits. (Missing data in this query was filled with zero)

Investigation / Analysis

`CXSYNC_PORT` waits usually occur due to issues with parallelism where different threads are waiting for each other. Or as stated in Wait types:

> Occurs with parallel query plans when waiting to open, close, and synchronize Exchange Iterator ports between producer and consumer threads. For example, if a query plan has a long sort operation, CXSYNC_PORT waits may be higher because the sort must complete before the Exchange Iterator port can be synchronized.

The longer-than-expected waits might occur because the MAXDOP setting is too high and the workload gets spread over too many NUMA nodes (if the database runs with more than 8 vCores). Another possible cause is that one or more of the parallel threads trigger an auto-update statistics and need to wait until the statistics data has been sampled.

Mitigation

To minimize the risk of running into `CXSYNC_PORT` waits, consider the following steps:

- Set MAXDOP to 8 or lower to limit the number of parallel threads. See MAXDOP Issues mitigation for further information and steps.
- Run auto-update statistics asynchronously to avoid that the threads have to wait on the statistics sampling. With `ALTER DATABASE AzureDatabaseName SET AUTO_UPDATE_STATISTICS_ASYNC ON`, the update statistics will be triggered for background execution but will allow the thread to continue its work.

More Information

From Improving concurrency of asynchronous statistics update:

> While up-to-date statistics often improve query plan quality, the extra time added to some query executions due to statistics update may be undesirable, particularly in transactional workloads with short queries, where updating statistics may take longer than query execution itself. For this reason, the SQL Server database engine also supports an option to update statistics asynchronously.

> When the database option AUTO_UPDATE_STATISTICS_ASYNC is set to ON, query optimizer proceeds to compile and execute the query even if statistics are considered stale; however, stale statistics are then updated on a background thread asynchronously, so that future query executions starting after this asynchronous process has completed can benefit from up-to-date statistics.

Public Doc Reference

- Blog: Improving concurrency of asynchronous statistics update
- Blog: Improving Performance with AUTO_UPDATE_STATISTICS_ASYNC

How good have you found this content?