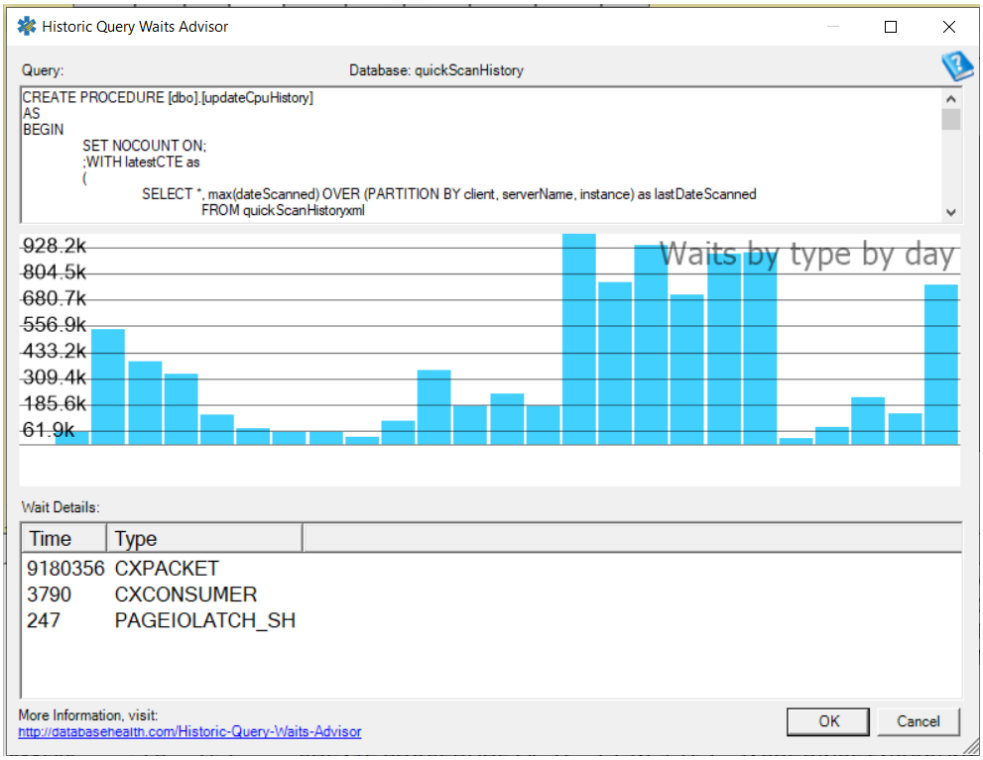
CXPACKET and CXCONSUMER Wait

CXCONSUMER and CXPACKET are associated with parallelism. When a query that has a cost beyond the cost threshold for parallelism then that query will be split out to be work on by multiple cores. This is a good thing. It usually means that your queries are taking advantage of multiple cores.



Reducing your max degree of parallelism setting is not necessarily a good thing to do unless it is excessive. You might find documentation that says you can get rid of CXPACKET by setting the MAXDOP setting to 1. This does indeed eliminate all CXPACKET waits, however setting MAXDOP to 1 turns off all parallel processing on your SQL Server, which will likely slow down the execution of many queries. DO NOT DO THIS.

The CXCONSUMER wait type was added in 2016 SP2 and 2017 RTM CU3.

Suggestion filter out or ignore CXCONSUMER then focus on the CXPACKET waits where the real issues are. CXCONSUMER can be safely ignored, where if CXPACKET is excessive you may want to look into the queries causing the CXPACKET waits.

Some common ways to reduce CXPACKET and effectively CXCONSUER waits are:

- Adding Missing indexes
- Relieving CPU pressure, but adding more or faster cores, or fixing inefficient queries.
- Memory pressure. Adding memory, or reducing memory needed for inefficient queries.
- Out of data statistics causing SQL Server to incorrectly divide the query into equal sized sets. This is a pretty common cause.
- Fragmented indexes causing slower IO speeds that impact one thread over the others. Used to be more of an issue on slower storage, but with faster storage not as much of an issue.
- Missing search predicates. Adding more search predicates on your where clause or a join will help reduce the amount of data fed into a query.
- Queries that are forcing a row by row processing of results rather than using sets.
- Client applications not efficiently processing result sets.
- Nested views can also be a problem. For instance on view calling another and then another can throw off the parameters and lead to an inefficient plan.

Focus on the CXPACKET waits on not on the CXCONSUMER waits. Reduce CXPACKET waits by improving query performance not just by reducing the number of cores.