CXPACKET waits in your SQL Servers

CXPACKET waits for OLTP servers

OLTP servers have fast running transactions that should be never parallelized and therefore never cause CXPACKET waits. If you tune your queries properly, verify that the index usage is correct, and perform index and statistics maintenance when necessary, CXPACKET waits should not be of concern. If you experience high CXPACKET waits, try the following:

- Increase the Cost threshold for parallelism server configuration option until the waits are under control.
- Use the 'OPTION (MAXDOP 1)' query hint to prevent parallelism when you identify specific queries that cause the CXPACKET waits.
- Set the Max degree of parallelism server configuration option to half the number of physical processors on the server up to a max of eight. This configuration prevents a query from monopolizing all of the processors at the detriment of other activity on the server. As a last resort, set the Max degree of parallelism server configuration option to 1 to prevent any query parallelism.

CXPACKET waits for non-OLTP servers

CXPACKET waits for non-OLTP servers (Reporting, OLAP, Hybrid-OLTP, etc.) are indicators that parallelism takes place.

CXPACKET waits indicate that:

- Queries may need to be tuned
- Indexes may need to be added and/or correctly tuned for the workload
- Index reorganization/rebuilding may be needed
- Statistics may be out of date

Configure your server options for parallelism

Max degree of parallelism

Leave the Max degree of parallelism server configuration option at zero. Note that this configuration may be acceptable for non-OTPL Servers. You can also consider setting the Max degree of parallelism server configuration option at half the number of physical processors on the server up to a max of eight.

Cost threshold for parallelism

Set the Cost threshold for parallelism server configuration option at the default setting of 5.

Identify what is causing CXPACKET waits
To identify the responsible for CXPACKET waits in your SQL Server, run the following query: `select`

```
      t.wait_type,
      t.wait_duration_ms,
      t.session_id,
      t.resource_description,
      s.program_name,
      b.sql_handle,
      b.statement_start_offset,
      b.statement_end_offset,
      [statement_text]=SUBSTRING(st.text, (b.statement_start_offset/2)
+1,
                                 ((CASE b.statement_end_offset
                                   WHEN-1 THEN DATALENGTH(st.text)
                                   ELSE b.statement_end_offset
                                 END - b.statement_start_offset)/2) +
1),
      st.text

      from sys.dm_os_waiting_tasks t
      left join sys.dm_exec_requests b
      left join sys.dm_exec_sessions s on b.session_id = s.session_id
      on t.waiting_task_address = b.task_address
      outer apply sys.dm_exec_sql_text (b.sql_handle) st
      where b.sql_handle is not null and t.wait_type = 'CXPACKET';
```

To identify the responsible for CXPACKET waits and also display the execution plan, run the following modified script:

```
select
      t.wait_type,
      t.wait_duration_ms,
      t.session_id,
      t.resource_description,
      s.program_name,
      qp.query_plan,
      b.statement_start_offset,
      b.statement_end_offset,
```

```
      [statement_text]=SUBSTRING(st.text, (b.statement_start_offset/2)
+1,
                                ((CASE b.statement_end_offset
                                    WHEN -1 THEN DATALENGTH(st.text)
                                    ELSE b.statement_end_offset
                                END - b.statement_start_offset)/2) +
1),
      st.text

      from sys.dm_os_waiting_tasks t
      left join sys.dm_exec_requests b
      left join sys.dm_exec_sessions s on b.session_id = s.session_id
      on t.waiting_task_address = b.task_address
      outer apply sys.dm_exec_sql_text (b.sql_handle) st
      cross apply sys.dm_exec_query_plan(b.plan_handle) qp
      where b.sql_handle is not null and t.wait_type = 'CXPACKET'
```

After running the query, click the **ShowPlan** link in the query_plan column to display the execution plan. Review the plan to see if the query in question needs additional tuning.