



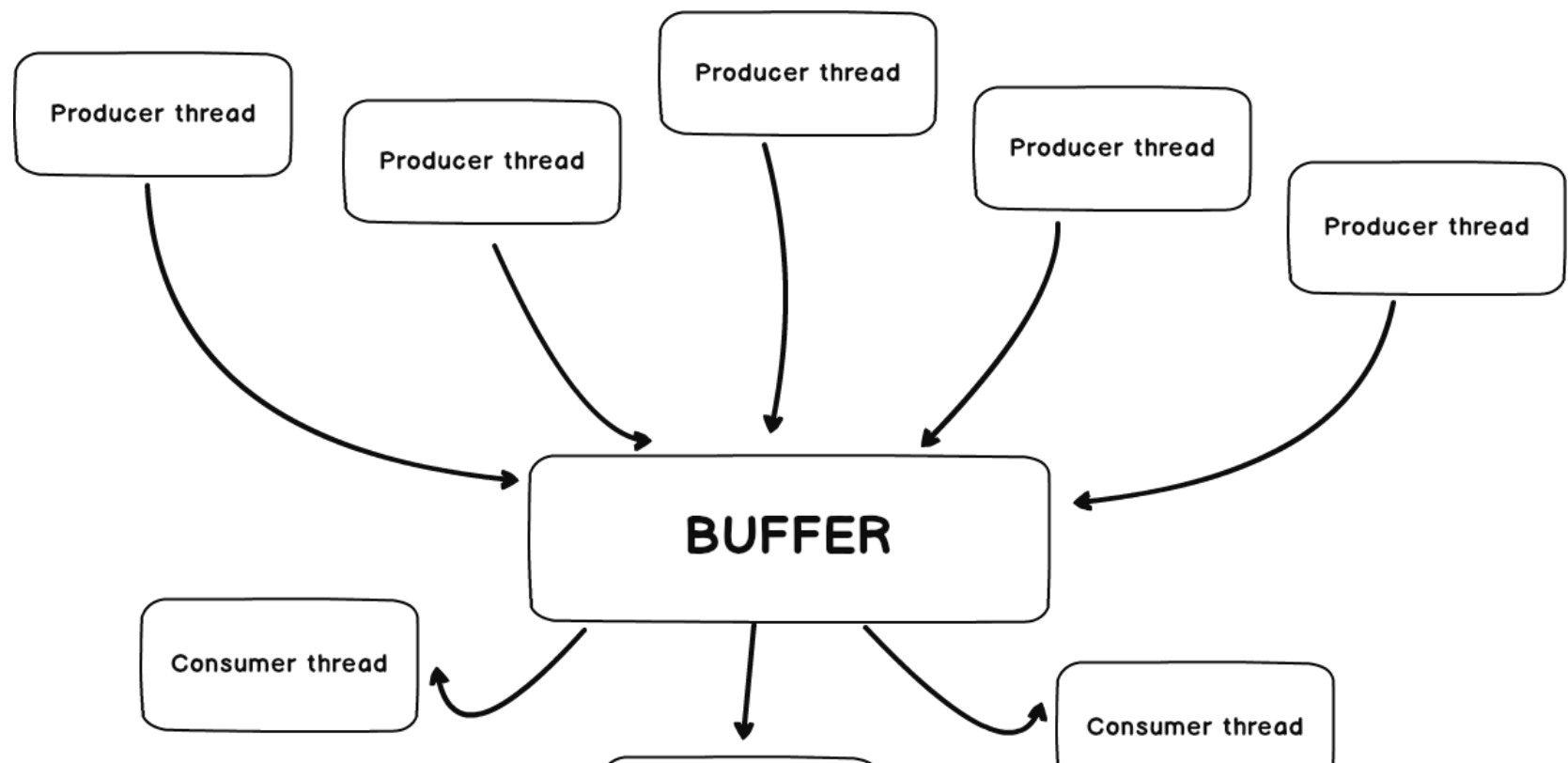
Solucionando problemas del tipo de espera CXPACKET en SQL Server

April 20, 2018 by [Nikola Dimitrijevic](#)

El tipo de espera de SQL Server CXPACKET es una de las estadísticas de espera más malinterpretadas. El término CXPACKET viene de Class Exchange Packet y, en esencia, esto puede ser descrito como filas de datos intercambiadas entre dos hilos paralelos que son parte de un solo proceso. Un hilo es el "hilo productor" y el otro hilo es el "hilo consumidor". Este tipo de espera está directamente relacionada al paralelismo y ocurre en SQL Server cuando este ejecuta una consulta usando un plan paralelo.

Generalmente hablando, el tipo de espera CXPACKET es normal para SQL Server y es un indicador de que SQL Server está usando un plan paralelo al ejecutar una consulta, lo cual es generalmente más rápido comparándolo a una consulta ejecutada en un proceso serializado. Cuando el plan paralelo es usado, la consulta es ejecutada en múltiples hilos y la consulta puede continuar sólo cuando todos los hilos paralelos están completados. Esto significa que la consulta será tan rápida como el hilo más lento.

El diagrama siguiente será usado para un mejor entendimiento del tipo de espera de SQL Server CXPACKET, y ayudará en su interpretación.





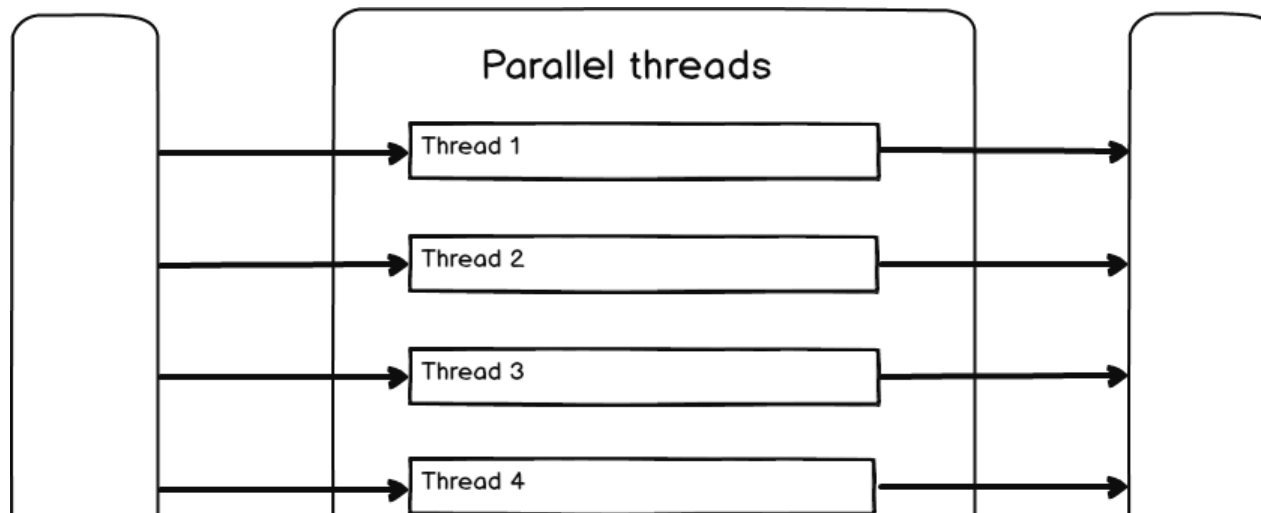
Consumer thread

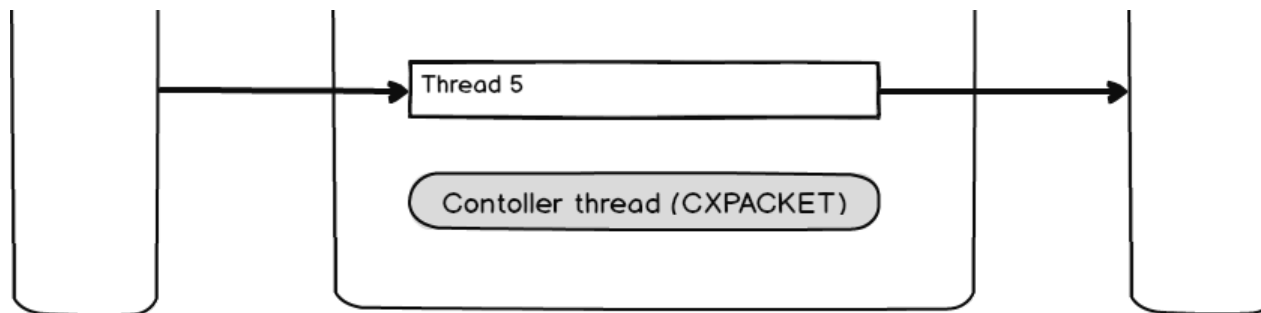
En este diagrama podemos ver que cuando sea que una ejecución paralela de consulta pueda proveer el beneficio a SQL Server, creará múltiples hilos para esa sentencia permitiendo que cada proceso paralelo produzca su propio subconjunto de datos. Cada hilo puede ser procesado por la CPU separada física o lógica. La comunicación entre los hilos productor y consumidor es realizada vía la cola productor-consumidor, la cual es realmente un búfer. El operador de consultas a cargo de implementar esta cola es llamado el operador Exchange.

Uno o más hilos productores producirán paquetes y los enviarán al búfer. Los datos entonces serán leídos desde el búfer por los hilos consumidores. Durante este proceso, tres diferentes escenarios que pueden causar las esperas CXPACKET excesivas pueden ser encontrados:

- El Consumidor no puede leer los paquetes porque el búfer (cola) está vacío, lo que significa que los hilos Productores no proveen o proveen datos lentamente al búfer. Esto significa que algunos hilos Productores están trabajando lentamente debido a que están esperando un recurso como la CPU, los permisos de memoria, I/O, etc, o algunos hilos Productores son simplemente bloqueados.
- Los hilos productores no puede almacenar paquetes en el búfer si este está lleno. Esto significa que los hilos Consumidores no pueden procesar los datos lo suficientemente rápido, causando una situación donde los hilos productores deben esperar a almacenar los datos en el búfer, una vez que el búfer se llena.
- El paralelismo excesivo para consultas pequeñas, donde crear el plan paralelo y la ejecución paralela podría ser más costoso y más lento que un plan serializado.
- Balance desigual de paquetes a través de los hilos paralelos, podría causar que algunos hilos completen el trabajo más rápidamente que otros, y luego ellos están esperando a que otros paquetes completen sus trabajos.

Así que introduzcámonos el tipo de espera de SQL Server CXPACKET para entender este proceso a mayor detalle. Consideremos el escenario ideal para ejecutar una consulta cuando un plan paralelo ha sido usado.

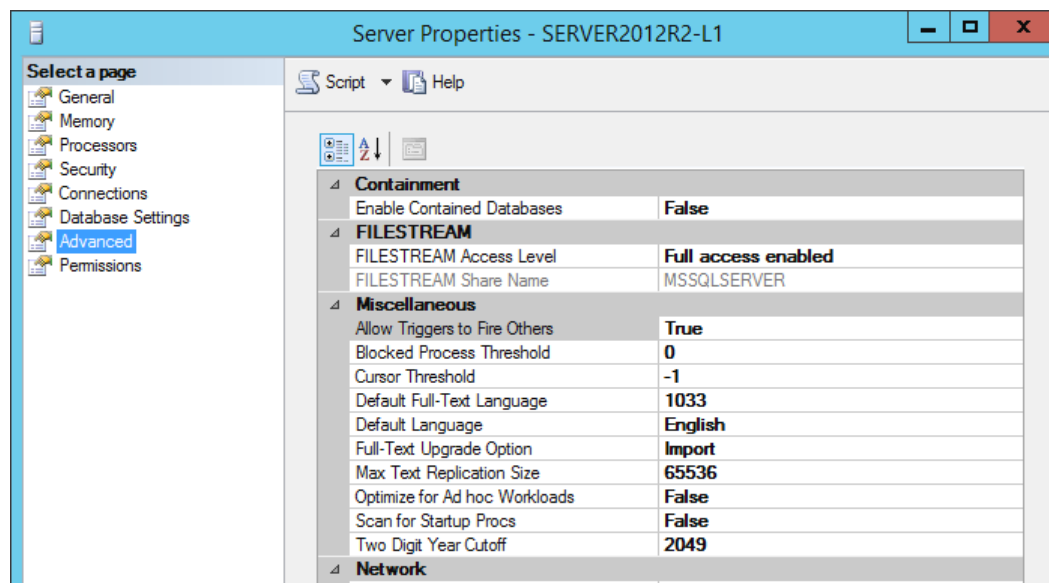


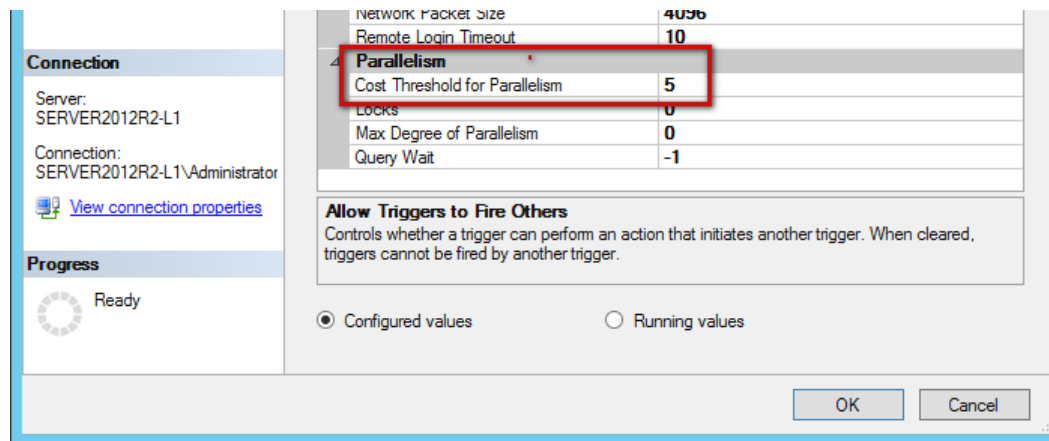


Lo que tenemos en la imagen de arriba es el ejemplo de un balance de carga apropiadamente balanceado en cada hilo paralelo, lo cual es una situación ideal, ya que serán ejecutados en paralelo sin esperar el uno al otro. Pero incluso en el escenario ideal, el plan paralelo siempre tiene un "hilo de control" y está a cargo de registrar las esperas CXPACKET. En el caso del hilo de control, la espera CXPACKET representará el tiempo necesario para un plan paralelo a ser ejecutado. Es ahora claro que el tipo de espera de SQL Server CXPACKET está siempre presente en ejecuciones paralelas incluso en un escenario ideal y que esta es más una indicación de paralelismo en ejecuciones de consultas que una indicación de que algo salió mal. Siempre que CXPACKET es menos que 50% de las esperas totales, no debería considerarse como un problema, sino como un indicador.

Cuando valores altos de CXPACKET son encontrados, un posible problema, incluso en el caso cuando el paralelismo está igualmente distribuido, es cuando el costo de crear el plan paralelo es más alto que el costo del hilo serializado. Esto es a menudo algo que se pasa por alto y por la regla de alterar el Máximo Grado de Paralelismo (MAXDOP), estableciéndolo en 1 (cada consulta será procesada por un solo núcleo de la CPU). Configurar MAXDOP a 1 debería ser el último recurso usado en la resolución de problemas de tiempos de espera CXPACKET excesivos.

Es importante saber que el optimizador de consultas de SQL Server está usando el Umbral de Costo para Paralelismo (CostThresholdforParallelism, CTFP) para determinar cuándo la consulta debería ser paralelizada o, en otras palabras, cuando el plan de consultas serializadas excede el umbral de costo para paralelismo, creará un plan paralelo de consultas. El CTFP está establecido a 5 por defecto, lo cual significa que incluso un plan no tan caro podría iniciar la creación del plan paralelo.





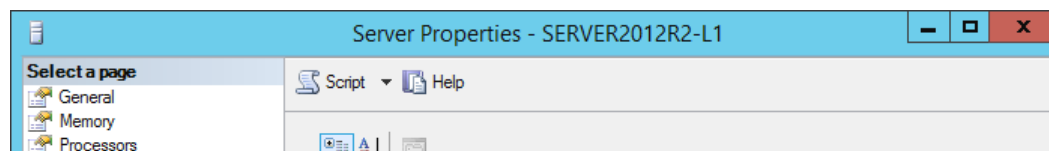
El valor del Umbral de Costo para Paralelismo es en segundos y significa que por cada consulta por la que SQL Server estima que el tiempo de corrida será más largo que 5 segundos, un plan paralelo será creado. Este valor por defecto ha sido establecido en los noventa, cuando computadoras de un solo núcleo, discos duros lentos y poca memoria eran usados, para computadoras modernas eso definitivamente no es óptimo. Lo que era ejecutado en esa era en 5 segundos, en máquinas modernas será ejecutado en un fragmento de segundo. En general, estimar la ejecución de consultas en segundos no es un buen enfoque, ya que el costo de consulta realmente depende de la CPU, la memoria, I/O, etc., y SQL Server no sabe la velocidad de la CPU y cuántos núcleos/CPU's están disponibles o la velocidad del HDD/SDD usado.

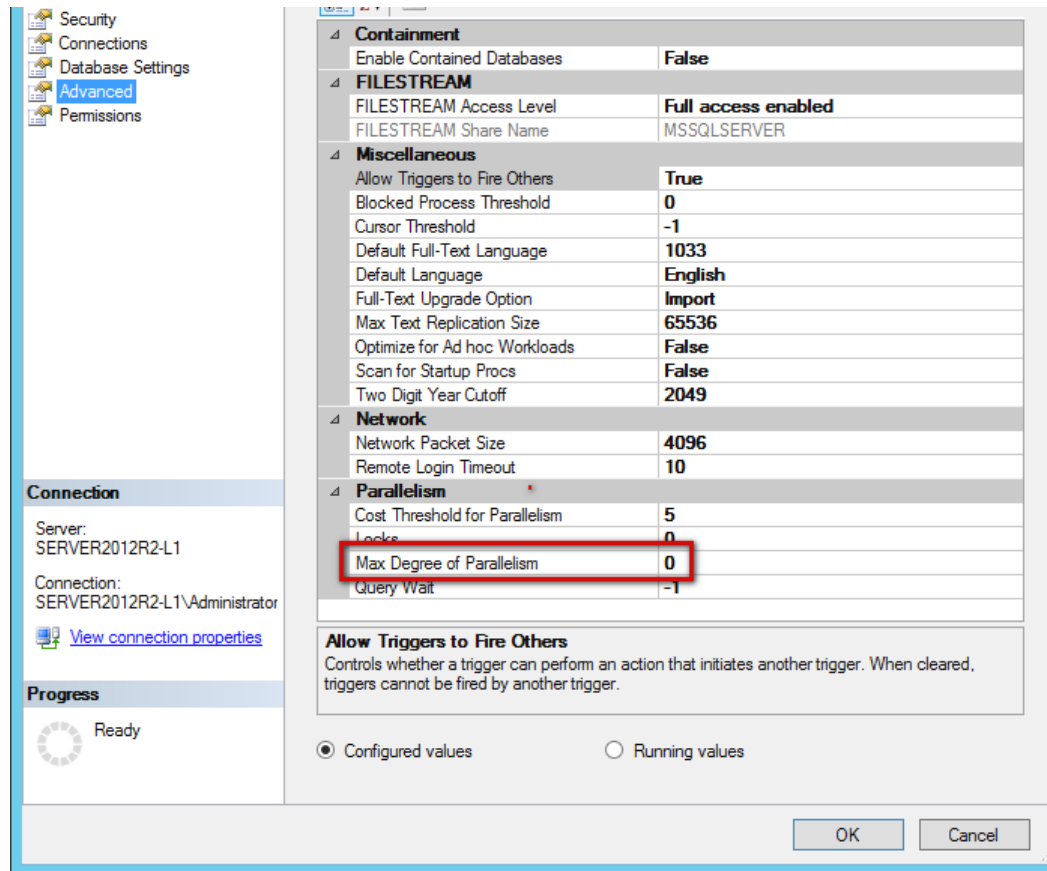
Para prevenir el paralelismo no deseado, el número de CTFP podría ser incrementado y, por la regla antes mencionada, un mínimo valor de 25. Análisis recientes indican que 50 debería ser el número mínimo óptimo para computadoras modernas. Aunque, encontrar el número CTFP apropiado y ajustarlo para máximo desempeño, que es algo que tiene que ser hecho analizando los planes de consultas y recursos disponibles, es la manera de determinar qué configuración de CTFP funcionaría mejor para el sistema especificado. Un gran lugar para aprender cómo determinar apropiadamente el valor de CTFP es el artículo [Ajustando el 'umbral de costo para paralelismo' desde el Plan de la Caché](#).

Así que sólo en las situaciones cuando los recursos mencionados arriba están exhaustos y el tiempo de espera CXPACKET es aún grande se debería considerar jugar con el Máximo Grado de Paralelismo. El número MAXDOP representa el número de núcleos de CPU que SQL Server usará para la ejecución paralela. El ajuste por defecto para MAXDOP es 0, y significa que todos los núcleos de CPU deberían ser usados para el procesamiento. Con máquinas modernas que tienen 8, 12, 32, 64 o incluso más núcleos, no se recomienda permitir que una sola consulta tome todos los núcleos.

Cuando un valor alto de CXPACKET está acompañado con un LATCH_XX y con PAGEIOLATCH_XX o SOS_SCHEDULER_YIELD, es un indicador de que el paralelismo lento/ineficiente en sí mismo es la causa real de los problemas de desempeño. Y, en tal escenario, si las esperas LATCH_XX son de las clases ACCESS_METHODS_DATASET_PARENT o ACCESS_METHODS_SCAN_RANGE_GENERATOR, entonces es altamente probable que el nivel de paralelismo es el cuello de botella y la causa real del problema de desempeño de la consulta. Este es un ejemplo típico donde MAXDOP debería ser reducido.

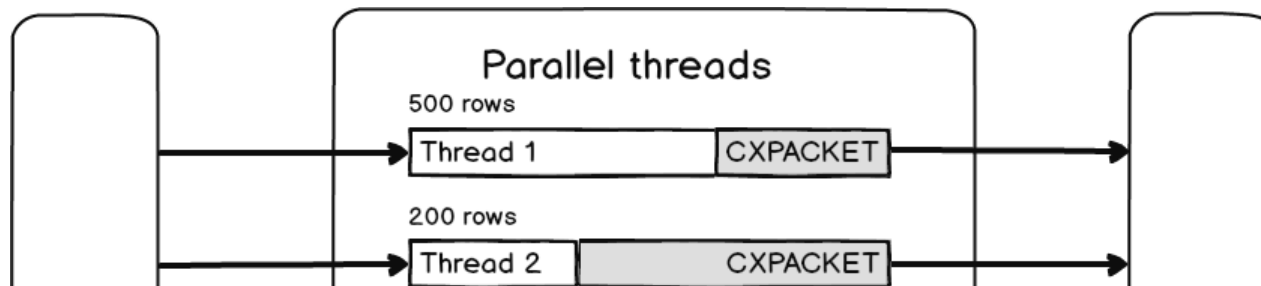
Para aquellos interesados en más detalles acerca de cómo configurar MAXDOP apropiadamente para Intel, AMD y/o máquinas virtuales, aquí hay un buen artículo [Recomendaciones y guías para la configuración de la opción "máximo grado de paralelismo" en SQL Server](#).

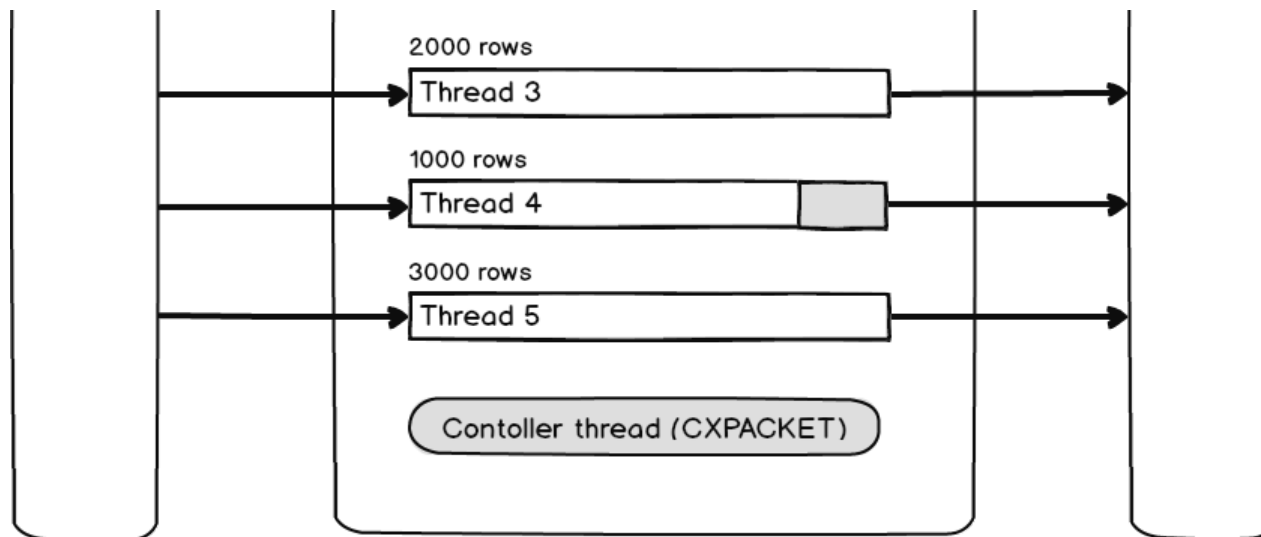




Todo lo anteriormente descrito tiene un objetivo, y es permitir a grandes consultas ser ejecutadas en paralelo, ya que se pueden beneficiar de eso significativamente, y asegurarse que las consultas pequeñas sean corridas de forma serializada, ya que es un enfoque más eficiente para consultas pequeñas.

Otro escenario donde altos valores para el tipo de espera de SQL Server CXPACKET pueden ocurrir es debido a una distribución desigual de datos a través de los hilos. Este es un escenario típico donde CXPACKET no es el problema, sino que el valor de CXPACKET es un indicador de que un problema existe. En tales casos, la resolución de problemas debería estar enfocada en otros problemas potenciales para entender mejor cómo este escenario puede el valor alto de CXPACKET. El siguiente gráfico será usado para ilustrar el escenario.





En este escenario particular, se puede ver que Thread 1 y Thread 2 fueron ejecutados y completaron su procesamiento, así que ahora están esperando a que otros hilos terminen su ejecución. Como se muestra en este caso particular, los hilos 3 y 5 aún están corriendo. Este tipo de espera de hilo es llamado espera CXPACKET. Debido a la distribución desigual de datos que cada hilo tiene que procesar, el tipo de espera CXPACKET puede tener significativamente valores más altos algunas veces. La mayor parte de la carga puede estar en uno o dos hilos en lugar de todos los cinco, como tenemos en nuestro ejemplo, así que el tiempo necesario para completar será más alto. En tales casos, la espera CXPACKET es de nuevo un indicador de que hay algo mal, aunque no con el paralelismo en sí mismo, sino con los recursos externos que están causando la distribución desigual de datos por hilo. La fuente del problema debería ser investigado enfocándose en la indexación no apropiada, por ejemplo, o estadísticas obsoletas, entre otras razones.

También es posible que ese hilo tenga que esperar por algunos recursos externos, los más comunes de los cuales son:

- Cuando el hilo tiene que compartir recursos I/O con otra base de datos o aplicación, lo cual causa un procesamiento más lento y requiere más tiempo para completar el trabajo.
- Una consulta grande paralizada que se está ejecutando por un largo tiempo, donde diferentes hilos tienen que acceder a diferentes bases de datos que están en un almacenamiento físico o lógico diferente y de otra velocidad.
- Los recursos necesarios por algunos hilos paralizados están bloqueados por consultas ad-hoc ejecutadas al mismo tiempo.

Esto también es un ejemplo donde el tipo de espera CXPACKET es sólo un indicador de que algo está mal. En tales situaciones, se recomienda revisar los tipos de espera asociados LCK_M_XX o PAGEIOLATCH_XX, así como las esperas IO_COMPLETION y ASYNC_IO_COMPLETION, que a menudo están acompañando los dos anteriormente mencionados. Diagnosticar y solucionar problemas de esos tipos de espera, más allá de CXPACKET, es algo que resolverá la causa raíz de los problemas de paralelismo que fueron advertidos vía el alto valor del tipo de espera CXPACKET.

Para resumir, estos son los pasos recomendados para diagnosticar la causa de los valores altos de las estadísticas de espera CXPACKET (antes de tomar una reacción impulsiva y cambiar algo en SQL Server):

- No establezca MAXDOP en 1, ya que esto nunca es la solución.

- Investigue el historial de CXPACKET y las consultas para entender y determinar si es algo que ocurrió sólo una o dos veces, ya que podría ser sólo la excepción en el sistema que normalmente está trabajando correctamente.
- Revise los índices y estadística en tablas usadas por la consulta y asegúrese de que están actualizados.
- Revise el Umbral de Costo para Paralelismo (CTFP) y asegúrese de que el valor usado es apropiado para su sistema.
- Revise si CXPACKET está acompañado con un LATCH_XX (posiblemente con PAGEIOLATCH_XX or SOS_SCHEDULER_YIELD también). Si este es el caso, entonces el valor de MAXDOP debería ser reducido para coincidir con su hardware.
- Revise si CXPACKET está acompañado con un LCK_M_XX (usualmente acompañado de IO_COMPLETION and ASYNC_IO_COMPLETION). Si este es el caso, entonces el paralelismo no es el cuello de botella. Revise esas estadísticas de espera para encontrar la causa raíz del problema y su solución.

Vea más

Para obtener 3 licencias gratis de una [herramienta de monitoreo de SQL Server](#), descargue [ApexSQL Monitor](#) y llene este [simple formulario](#)

Enlaces útiles

- [Introducción a Usar Estadísticas de Espera Para Identificar Y Remediar Cuellos de Botella de Paralelismo de Consultas](#)
- [Suprimiendo el paralelismo de consultas, esperas CXPACKET eliminadas y 30% del CPU liberado](#)
- [Procesamiento de consultas paralelas](#)
- [Artículo de mejores prácticas de SQL Server](#)
- [Máximo número de procesadores soportados por las ediciones de SQL Server](#)

The future of **SQL Server** monitoring



Nikola Dimitrijevic

Nikola es un fanático de las computadoras desde 1981 y un entusiasta de SQL con intención de convertirse en fanático. Especializado en la auditoría, el cumplimiento de requerimientos y el monitoreo del desempeño de SQL Server.

Devoto de la aviación militar y modelador de naves aéreas a escala. Fan de los deportes extremos; instructor de paracaidismo y salto bungee. Una vez serio, ahora es un fotógrafo en su tiempo libre.

[Vea todas las entradas de Nikola Dimitrijevic](#)

Related Posts:

1. [Una guía para Administradores de Bases de Datos para resolución de problemas de SQL Server – Parte 1 – Métricas de problemas y desempeño](#)
2. [Problemas de desempeño de cursores en SQL Server](#)
3. [Las vistas/tablas/funciones de sistema de SQL Server. Preguntas frecuentes y soluciones a problemas de la vida real](#)
4. [Troubleshooting the CXPACKET wait type in SQL Server](#)
5. [Monitor de Actividad de SQL Server](#)

Optimización de rendimiento SQL

1,124 Views

0 Comments **SQL Shack****Login** ▾ **Recommend** 1  **Share****Sort by Best** ▾

Start the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS 

Be the first to comment.

ALSO ON SQL SHACK

SQL Server Blogs

2 comments • 5 months ago

**Kevin Hill** — www.DallasDBAs.com :)
Avatar**CE 2016 – Simple Join**



2 comments • 5 months ago

**LeMaciek** — i moved my question and the possible answer to:
Avatar <https://dba.stackexchange.c...>**Retention successes and failures; combatting the Peter Principle for managers**

3 comments • 3 months ago

**Brian Lockwood** — My guess is you won't need to worry about the Peter Principle - you are constantly learning and challenging yourself. I would call this the "Prashanth Principle" - hmmm ... potential idea for another blog post ;)
Avatar**Rubber Balls vs Glass balls – a metaphor for task prioritization**

2 comments • 3 months ago

**Prashanth Jayaram** — Hi Brian, Thanks for the great write-up!. I feel, its a perfect fitment for all the life time events:). Thanks again for the great post!. Best Regards, Prashanth
Avatar **Subscribe**  **Add Disqus to your site** Add Disqus Add  **Disqus' Privacy Policy** Privacy Policy Privacy Policy