SQL SERVER Performance Tuning Notes

http://msbiskills.com/

Execution Plan Operator – Properties, Types – Blocking / Non-Blocking?

Notes only

Ref - http://blogs.msdn.com/b/craigfr/archive/2006/06/19/637048.aspx

In this post, I'll discuss properties of iterators and non-blocking vs. blocking. Note that these things can affect performance of the query.

SQL Server has two kinds of execution plan operators. They are

- Blocking
- Non-blocking (Stop-and-go)

Non-blocking iterator get rows in and sends rows out at the same time (in the GetRow method).

For instance, the Nested Loops iterator gets rows from its outer input and sends the matching rows on to the next iterator without having to wait for all of the input rows.

Compute scalar iterator is also an example of a non-blocking iterator. It read an input row, computes a new output value using the input values from the current row, immediately outputs the new value, and continues to the next input row.

Blocking iterator - Consume all of the input rows (in the open method) before it can send rows out to the next iterator.

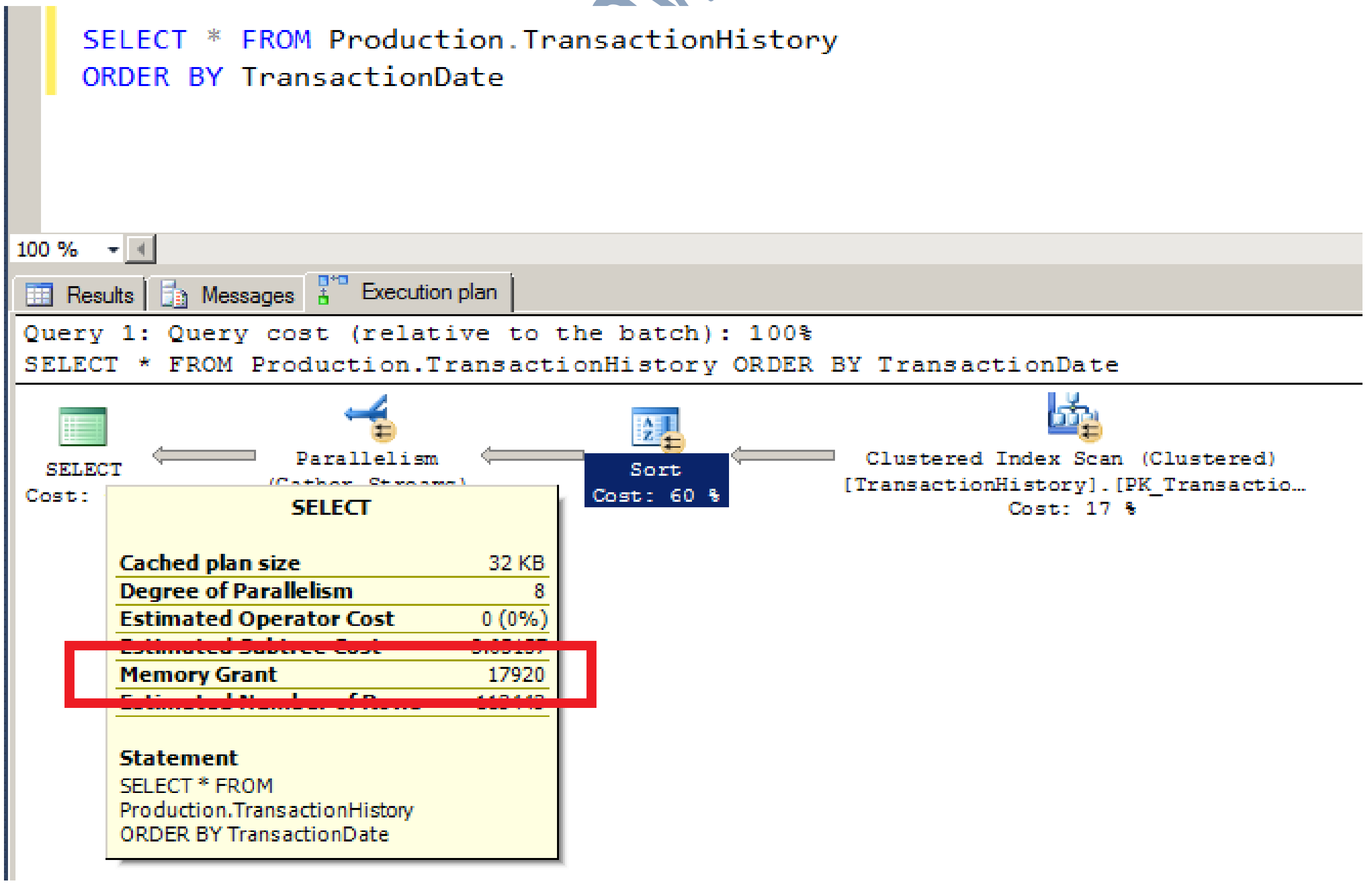
For instance sort is a blocking operator. It has to consume all of the input rows and sort them before it can send them on to the next iterator.

Blocking operators may or may not consume memory. It is also possible that iterators have phases and one phase is blocking and another one is non-blocking. Example of one such iterator is Hash Join. In this iterator build phase is blocking and probe is pipelined.

A blocking operator may lead to concurrency issues and reduce performance. Non-blocking iterators are good for OLTP systems.

NOTE - Microsoft has not released a definitive list about blocking and non-blocking operators. You can mostly check which of the operators within a plan are blocking or not by reading their descriptions.

Also note that all iterators require some amount of memory to perform calculations and state management. Every query will have to acquire some memory to execute the query. This is called Memory Grant. You can view this memory grant in the first operator that Select, Update or Insert. Now if the server is executing other queries and does not have enough memory to grant, our query will have to wait until the grant is available. I will explain Memory grant in detail in coming up articles are this is a very big topic.



```
SELECT * FROM Production.TransactionHistory
ORDER BY TransactionDate
```

```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM Production.TransactionHistory ORDER BY TransactionDate
```

SELECT
Cost:
Parallelism
(Gather Streams)
Sort
Cost: 60 %
Clustered Index Scan (Clustered)
[TransactionHistory].[PK_Transactio…
Cost: 17 %

SELECT

Cached plan size	32 KB
Degree of Parallelism	8
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	3.05197
Memory Grant	17920
Estimated Number of Rows	113443

Statement
SELECT * FROM
Production.TransactionHistory
ORDER BY TransactionDate

Now if the estimates are bad or we have not conveyed correct information to the optimizer and memory consuming operator requests a small amount of memory it will lead to spill data to disk during execution. Spills are not good for query performance due to extra I/O overhead.

Major memory consuming iterators are Sort, hash join and hash aggregate. Sort spills are very famous. We can track this using warning in the execution plan or using Sort Warnings in SQL profiler.

Summary

So all in all we have try to remove the blocking and memory consuming operators for OLTP systems to get good query performances.

That's all folks; I hope you've enjoyed learning about operators and their types, and I'll see you soon with more "Performance Tuning" articles.

Thanks!

Pawan Kumar Khowal

MSBISKills.com