

## Re: Frustrating Execution Plan Analysis

**Source:**

<http://www.tech-archive.net/Archive/SQL-Server/microsoft.public.sqlserver.programming/2004-10/3278.html>

---

**From:** Hugo Kornelis (*hugo\_at\_pe\_NO\_rFact.in\_SPAM\_fo*)

**Date:** 10/14/04

Date: Thu, 14 Oct 2004 11:16:01 +0200

On Wed, 13 Oct 2004 18:01:40 -0400, Jerad Rose wrote:

>Ok, I have two queries: 1 and 2. They are both identical, except query 1  
>has the following WHERE clause:  
>  
>where Trade.TradeDateCreated > @Date  
>and TradeStatus.Value is not null  
>  
>When I run both queries back-to-back w/ Execution Plan turned on, here are  
>the results:  
>  
>Query 1: Query cost (relative to the batch): 35.11%  
>Query 2: Query cost (relative to the batch): 64.89%  
>  
>However, I also included a print getdate() at the beginning, middle, and end  
>of the batch, and here are the results:  
>  
>Oct 13 2004 5:49:24:090PM  
>Oct 13 2004 5:49:27:577PM  
>Oct 13 2004 5:49:28:030PM  
>  
>So, as you can see, Query 1 takes almost 3.5 seconds, but Query 2 takes less  
>than 0.5 seconds (these results are consistent, +/- 0.5 seconds).  
>  
>I've always been under the impression that Query cost is directly related to  
>execution time -- is this not the case? If not, then what is the point of  
>Query cost? The only thing that I'm concerned with is speeding up my query,  
>and I need the filters (WHERE clause) in place, so that's when I started  
>examining the execution plan. But if the execution plan isn't making sense,  
>then it's not going to help me tweak my query/indexes.

Hi Jerad,

In addition to the answers you already received, I'd like to point out three simple possible pitfalls for this type of comparison.

1. The longer execution time of the first query might well be caused by physical disk access (which is slow), whereas the second execution might have takes the data from the cache. To eliminate these effects from your comparisons, make sure to run `DBCC FREEPROCCACHE` and `DBCC DROPCLEANBUFFERS` before each execution.

2. Make sure your statistics are up to date. An execution plan based on outdated statistics is almost guaranteed to be suboptimal if the distribution of data in your tables has changed significantly.

3a. If you execute the query you posted as a batch, SQL Server will compile the whole batch before starting execution. At compile time, the `SET` statements that give variables their values have not yet been executed. SQL Server will make a plan based on assumptions about what will be in the variables. Example: SQL Server will assume 30% of all rows match a `>` or `<` comparison (like the date comparison in your query). If the actual data has a significant lower (e.g. 2%) or higher (e.g. 95%) hit rate, the query plan might be suboptimal.

3b. If you have the query enclosed in a stored procedure, the execution plan will be generated on the first call and SQL Server will use the parameter values used in that first call. Other variables that are set as part of the stored procedure are still unknown at compile time. In your example, the `@Date` variable is calculated in the procedure, so this would still be unknown. The `@OnlyOpen` and `@UserID` would be parameters and would be known – the actual values will be compared against the statistics when the execution plan is generated. Unfortunately, if the first call uses atypical parameters, the plan generated might well be suboptimal for all future calls!

Best, Hugo

--

(Remove `_NO_` and `_SPAM_` to get my e-mail address)