**[Linchi Shea](http://sqlblog.com/blogs/linchi_shea/default.aspx)**

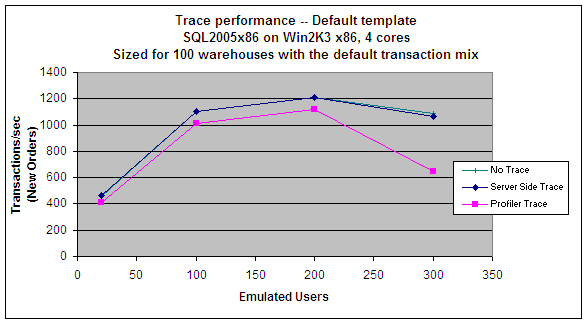
Checking out SQL Server via empirical data points

**Performance Impact: Profiler Tracing vs. Server Side SQL Tracing**

Adam Machanic asked me whether I had done any tests to compare the performance impact of using SQL Profiler on the client side and that of using SQL Trace on the server side. This is an interesting question because the frequently heard recommendation is to prefer SQL Trace on the server side, but the recommendation is almost always made without much support in terms of hard empirical data.

So I decided to run several tests to see the behavior difference, if any, in a controlled setting for myself. The intention of these tests was not to arrive at a thorough or comprehensive comparison between these two approaches to SQL Server tracing, but rather to obtain a few data points using a common workload. As long as we don't delude ourselves into thinking the results reported here paint a comprehensive picture of the issue, the tests should add value. And hopefully with more data points from the SQL Server community, a more comprehensive picture does emerge.

Enough warnings and disclaimers. What were the results? A picture is worth a thousand words. Let me first give you the following chart that summarizes the test results, and then I'll explain how the tests were conducted to obtain these results, and highlight some observations.



Before discussing the results, let's describe the test workload. The test workload was OLTP in nature, generated using a TPC-C test kit that I wrote. The transactions followed the TPC-C standard mix. In other words, for each simulated user, the chance of submitting a New Order, a Payment, an Order Status, a Delivery, or a Stock Level transaction was 45%, 43%, 4%, 4%, and 4%, respectively. The wait time between two consecutive transactions--from the time the first completed to the time the second started--was set to 10 milliseconds.

The test database was scaled to 100 warehouses, which translates into ~9GB of used database space.

The number of users simulated were 20, 100, 200, and 300. In addition, 5 users and 500 users were simulated, but the load was either too light (in the case of 5 users) or too heavy (in the case of 500 users). So their results are not included in the chart.

This was a relatively balanced workload with both processors and the disk I/Os sufficiently stressed, but not overloaded. Total processor usage was observed to be in the ~50% for 20 users and ~ 80% for 300 users. While the wait stats showed that SQL Server was mainly waiting on transaction log writes (> 60% of the total wait time), the transaction logging I/O response time was well under 5ms, a strong indicator of the I/O path not being overwhelmed.

The hardware and software configurations used in the tests are listed in Appendix B at the end of this post.

The transaction throughput reported in the chart was measured in terms of the total number of the New Order transactions processed per second.

Three test scenarios were included in the tests:

* **No Trace**,
* **Profiler Trace** using the standard default template supplied with SQL Server 2005 with SQL Profiler running on a client machine that was connected to the test server over a 100Mbps network. The trace data were not saved either to a file or to a table.
* **Server Side Trace** with the trace data sent to a file on a separate disk. The trace files were cycled at the 200MB threshold. The script used to create the trace was listed in the Appendix A at the end of this post, and was generated (with the folder name modified) for the Profiler trace using the standard default template.

The chart above represents the average transaction throughputs of multiple test runs. For each configured number of users (20, 100, 200, or 300), each test run lasted for 250 seconds with the transaction performance stats (i.e. transaction counts and their individual response times) recorded only for the final 100 seconds.

So what did we learn from these tests? Here are four salient points.

First, it's clear from the above chart that there wasn't a significant difference in the transaction throughput between the No Trace scenario and the Server Side Trace scenario.

Secondly, the Profiler Trace scenario produced lower transaction throughput than the other two scenarios. The difference was around 10% (~12% for 20 users, ~9% for 100 users, and ~8% for 200 users). There was a precipitous drop in the transaction throughput with 300 users when Profiler Trace was used. This was consistently observed during the tests.

To ensure that the results weren't random, I repeated each test scenario multiple times a few weeks later. For 20, 100, and 200 users, although the transaction throughput numbers were not exactly the same, the general pattern of the Profiler trace scenario having lower transaction throughput than the other two scenarios held consistently. However, for 300 users, the precipitous drop in the transaction throughput was no longer consistently observed.

At this point, I don't have a solid explanation for the steep drop with 300 users. My educated guess is that it had something to do with the load level, and that brings us to our third observation on the issue of the network bandwidth.

A Profiler trace can consume a significant amount of network bandwidth. During the tests, at least 35% of the 100Mbps network was consumed by the Profiler traffic in all test scenarios. The Profiler traffic could consume 70% or even higher of the 100Mbps network with heavier load.

The fourth observation is that SQL Profiler buffers the trace data in temporary files on the client machine, and if the drive on which these temporary files reside doesn't have ample free space, a Profiler trace (e.g. with the standard trace template) could deplete the drive of the free space rather quickly and end up disabling the trace. During the tests, the trace data were buffered on the C drive which had 2.4GB free space to begin with, and it didn't take long for the C drive to become completely filled up. When that happened, SQL Profiler threw the following error message in a pop-up form, and stopped the trace:

*System is low on disk space on drive 'C'. All SQL Server profiler functions are temporarily disabled.*

A little bit digging with Process Explorer revealed that during my tests, the temporary files to buffer the profiler trace data were in directory *C:\Documents and Settings\lshea\local settings\Temp\1*, and the names of the temporary files had this pattern: Prf\*.tmp. Upon exiting SQL Profiler, these temporary files would be removed automatically. During my tests, I had to stop and start SQL Profiler in between the test runs to keep the tests going.

To summarize,

1. The SQL Profiler traces did have a measurable degradation in performance compared with the server side traces or no trace at all, and
2. The SQL Profiler traces were not as robust as the server side traces.

**Appendix A**. SQL script for the server-side trace

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Created by: SQL Server Profiler 2005 \*/

/\* Date: 06/02/2007 11:27:37 AM \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-- Create a Queue

declare @rc int

declare @TraceID int

declare @maxfilesize bigint

set @maxfilesize = 200

-- Please replace the text InsertFileNameHere, with an appropriate

-- filename prefixed by a path, e.g., c:\MyFolder\MyTrace. The .trc extension

-- will be appended to the filename automatically. If you are writing from

-- remote server to local drive, please use UNC path and make sure server has

-- write access to your network share

exec @rc = sp\_trace\_create @TraceID output, 2, N'd:\ServerSide\_Trace', @maxfilesize, NULL

if (@rc != 0) goto error

-- Client side File and Table cannot be scripted

-- Set the events

declare @on bit

set @on = 1

exec sp\_trace\_setevent @TraceID, 14, 1, @on

exec sp\_trace\_setevent @TraceID, 14, 9, @on

exec sp\_trace\_setevent @TraceID, 14, 6, @on

exec sp\_trace\_setevent @TraceID, 14, 10, @on

exec sp\_trace\_setevent @TraceID, 14, 14, @on

exec sp\_trace\_setevent @TraceID, 14, 11, @on

exec sp\_trace\_setevent @TraceID, 14, 12, @on

exec sp\_trace\_setevent @TraceID, 15, 15, @on

exec sp\_trace\_setevent @TraceID, 15, 16, @on

exec sp\_trace\_setevent @TraceID, 15, 9, @on

exec sp\_trace\_setevent @TraceID, 15, 13, @on

exec sp\_trace\_setevent @TraceID, 15, 17, @on

exec sp\_trace\_setevent @TraceID, 15, 6, @on

exec sp\_trace\_setevent @TraceID, 15, 10, @on

exec sp\_trace\_setevent @TraceID, 15, 14, @on

exec sp\_trace\_setevent @TraceID, 15, 18, @on

exec sp\_trace\_setevent @TraceID, 15, 11, @on

exec sp\_trace\_setevent @TraceID, 15, 12, @on

exec sp\_trace\_setevent @TraceID, 17, 12, @on

exec sp\_trace\_setevent @TraceID, 17, 1, @on

exec sp\_trace\_setevent @TraceID, 17, 9, @on

exec sp\_trace\_setevent @TraceID, 17, 6, @on

exec sp\_trace\_setevent @TraceID, 17, 10, @on

exec sp\_trace\_setevent @TraceID, 17, 14, @on

exec sp\_trace\_setevent @TraceID, 17, 11, @on

exec sp\_trace\_setevent @TraceID, 10, 15, @on

exec sp\_trace\_setevent @TraceID, 10, 16, @on

exec sp\_trace\_setevent @TraceID, 10, 9, @on

exec sp\_trace\_setevent @TraceID, 10, 17, @on

exec sp\_trace\_setevent @TraceID, 10, 2, @on

exec sp\_trace\_setevent @TraceID, 10, 10, @on

exec sp\_trace\_setevent @TraceID, 10, 18, @on

exec sp\_trace\_setevent @TraceID, 10, 11, @on

exec sp\_trace\_setevent @TraceID, 10, 12, @on

exec sp\_trace\_setevent @TraceID, 10, 13, @on

exec sp\_trace\_setevent @TraceID, 10, 6, @on

exec sp\_trace\_setevent @TraceID, 10, 14, @on

exec sp\_trace\_setevent @TraceID, 12, 15, @on

exec sp\_trace\_setevent @TraceID, 12, 16, @on

exec sp\_trace\_setevent @TraceID, 12, 1, @on

exec sp\_trace\_setevent @TraceID, 12, 9, @on

exec sp\_trace\_setevent @TraceID, 12, 17, @on

exec sp\_trace\_setevent @TraceID, 12, 6, @on

exec sp\_trace\_setevent @TraceID, 12, 10, @on

exec sp\_trace\_setevent @TraceID, 12, 14, @on

exec sp\_trace\_setevent @TraceID, 12, 18, @on

exec sp\_trace\_setevent @TraceID, 12, 11, @on

exec sp\_trace\_setevent @TraceID, 12, 12, @on

exec sp\_trace\_setevent @TraceID, 12, 13, @on

exec sp\_trace\_setevent @TraceID, 13, 12, @on

exec sp\_trace\_setevent @TraceID, 13, 1, @on

exec sp\_trace\_setevent @TraceID, 13, 9, @on

exec sp\_trace\_setevent @TraceID, 13, 6, @on

exec sp\_trace\_setevent @TraceID, 13, 10, @on

exec sp\_trace\_setevent @TraceID, 13, 14, @on

exec sp\_trace\_setevent @TraceID, 13, 11, @on

-- Set the Filters

declare @intfilter int

declare @bigintfilter bigint

exec sp\_trace\_setfilter @TraceID, 10, 0, 7, N'SQL Server Profiler - 1e3470c1-ba7c-4138-af3d-c96f83654334'

-- Set the trace status to start

exec sp\_trace\_setstatus @TraceID, 1

-- display trace id for future references

select TraceID=@TraceID

goto finish

error:

select ErrorCode=@rc

finish:

go

**Appendix B**. Hardware and software configurations of the test server

**Hardware**  
DL585 G1 with four single core 2.6GHz AMD Opteron processors and 16GB of physical memory. Three drives were used: D was an internal RAID 1 set, E was RAID10 with 260GB in size, and F was another RAID 10 set with 260GB in size.

**Software**  
SQL Server 2005 SP2 (9.00.3042) Enterprise x86 Edition on Windows Server 2003 x86 Enterprise Edition SP1. The test database was 15GB is total size with 10GB for data and 5GB for log. The 10GB data file was on the E drive and the 5GB log file was on the F drive. The test database was populated with ~9GB of data per the TPC-C specifications for 100 warehouses. The automatic checkpoints were disabled, and a manual checkpoint was issued once every 60 seconds.

For the server side trace, the trace data files were on drive D.

Published Wednesday, August 01, 2007 6:11 PM by [Linchi Shea](http://sqlblog.com/user/Profile.aspx?UserID=2236)

Filed under: [Performance](http://sqlblog.com/blogs/linchi_shea/archive/tags/Performance/default.aspx), [Best Practices](http://sqlblog.com/blogs/linchi_shea/archive/tags/Best+Practices/default.aspx)

**http://sqlblog.com/Themes/default/images/paperclip-18-red.gifAttachment(s):** [**Profil1.gif**](http://sqlblog.com/blogs/linchi_shea/attachment/2057.ashx)

**Comment Notification**

If you would like to receive an email when updates are made to this post, please register [here](http://sqlblog.com/user/CreateUser.aspx?ReturnUrl=/blogs/linchi_shea/archive/2007/08/01/trace-profiler-test.aspx)

Subscribe to this post's comments using [RSS](http://sqlblog.com/blogs/linchi_shea/commentrss.aspx?PostID=2057)

**Comments**

|  |  |
| --- | --- |
| http://sqlblog.com/utility/anonymous.gif | [**Greg Linwood**](http://sqlblog.com/utility/Redirect.aspx?U=http%3a%2f%2fblogs.sqlserver.org.au%2fblogs%2fgreg_linwood%2fdefault.aspx) **said:**  Good article Linchi  As you're aware already, I'm in the remote db admin business. Understanding the impact of SQLTrace is very important to my business because we need to keep traces running all the time if we are going to have any hope of answering customer questions about what was wrong with their servers in the past hour or so whenever they might call up complaining about performance. Obviously there's no point starting traces after the fact so we need them running all the time but we also don't want them impacting performance in the first place..  So we have undertaken many similar tests over the past 4 years and have consistently come up with similar results. Server-side traces have almost no impact on overall server performance but interactive Profiler sessions can have a noticably bigger impact. However, I've never seen \*such\* a significant difference as with your results, but then again I never run unfiltered Profiler sessions on live systems so there might be a major difference there. Your testing only has the default "no profiler activity" filter but is otherwise collecting all events. Perhaps a more useful test would be to put a simple filter such as Reads > 200 on just to fish out the lionshare of relatively meaningless information which most people wouldn't collect?  One very obvious problem with running SQL Profiler traces on the server which I don't think you mentioned is that it causes heavy CPU context switching if a lot of events are being captured (as the application's busy & taking CPU time away from SQL Server). Another interesting test would be to see how much difference occurred if you had SQL Server affinitised to 3 CPUs..  I'd also like to point out that the systems we do our trace impact evaluation on are real systems - we always collect various Perfmon counters but sometimes turn off SQLTracing to compare results. All of the servers are different (different customers in different industries running different systems etc) but the SQL Trace impact differential is always fairly consistent even though the servers are all very different..  One lesson we've learned is that there's a significant difference in overall server performance impact based on how many filters are configured for any given trace. Again, your trace only had the default filter which excludes Profiler events, but our tracing (& most routine tracing) always includes some sort of filter. We've found that too MANY filters (eg 7 or 8 on a single trace) can have a huge impact on server-side traces, even if only a few otherwise healthy events such as those in this trace are being collected.  It would be interesting to see how adding more filters affected your results.  Cheers,  Greg Linwood  August 2, 2007 7:24 AM |
| http://sqlblog.com/utility/anonymous.gif | [**Jason Haley**](http://sqlblog.com/utility/Redirect.aspx?U=http%3a%2f%2fjasonhaley.com%2fblog%2farchive%2f2007%2f08%2f02%2f140130.aspx) **said:**  August 2, 2007 9:20 AM |
| http://sqlblog.com/users/avatar.aspx?userid=2103 | [**Adam Machanic**](http://sqlblog.com/utility/Redirect.aspx?U=http%3a%2f%2fsqlblog.com%2fblogs%2fadam_machanic) **said:**  Greg,  Just to clarify, when you say that Profiler causes context switching, you mean only if Profiler itself is run on the same server running SQL Server, i.e. from the console or a RD connection, right?  August 2, 2007 9:57 AM |
| http://sqlblog.com/utility/anonymous.gif | [**Greg Linwood**](http://sqlblog.com/utility/Redirect.aspx?U=http%3a%2f%2fblogs.sqlserver.org.au%2fblogs%2fgreg_linwood%2fdefault.aspx) **said:**  Adam, Yes - the overhead can be substantial because it's not just about screen refreshes. There's a lot of overhead associated with re-structing event rows from their constituent column fragments. The more events, the more CPU workload the Profiler generates.  Cheers,  Greg Linwood  August 2, 2007 10:02 AM |
| http://sqlblog.com/utility/anonymous.gif | [**Profiler Research « Home of the Scary DBA**](http://sqlblog.com/utility/Redirect.aspx?U=http%3a%2f%2fscarydba.wordpress.com%2f2008%2f12%2f18%2fprofiler-research%2f) **said:**  PingBack from <http://scarydba.wordpress.com/2008/12/18/profiler-research/>  December 18, 2008 8:22 AM |
| http://sqlblog.com/utility/anonymous.gif | [**Jose Mariano Alvarez**](http://sqlblog.com/utility/Redirect.aspx?U=http%3a%2f%2fblog.josemarianoalvarez.com%2f) **said:**  Remember you can use server side traces with profiler if run locally in server.  April 15, 2009 10:54 PM |
| http://sqlblog.com/utility/anonymous.gif | [**Scripting Server Side Traces « WebbTech Solutions**](http://sqlblog.com/utility/Redirect.aspx?U=http%3a%2f%2fwebbtechsolutions.com%2f2009%2f08%2f31%2fscripting-server-side-traces%2f) **said:**  PingBack from <http://webbtechsolutions.com/2009/08/31/scripting-server-side-traces/>  August 31, 2009 6:48 AM |
| http://sqlblog.com/utility/anonymous.gif | [**Midnight DBA**](http://sqlblog.com/utility/Redirect.aspx?U=http%3a%2f%2fmidnightdba.itbookworm.com%2fmidnightdba%2fblog%2fpost%2fVIDEO-Server-Side-Trace.aspx) **said:**  VIDEO: Server Side Trace  September 3, 2009 11:43 AM |
| http://sqlblog.com/utility/anonymous.gif | [**davetiye**](http://sqlblog.com/utility/Redirect.aspx?U=http%3a%2f%2fwww.davetiyesozleri.com) **said:**  güzel davetiye sözleri ve davetiye modeloleri  November 3, 2009 7:59 AM |
| http://sqlblog.com/utility/anonymous.gif | [**davetiye**](http://sqlblog.com/utility/Redirect.aspx?U=http%3a%2f%2fwww.davetiyesozleri.com) **said:**  güzel davetiye sözleri ve davetiye modelleri  November 3, 2009 7:59 AM |
| http://sqlblog.com/utility/anonymous.gif | [**davetiye**](http://sqlblog.com/utility/Redirect.aspx?U=http%3a%2f%2fwww.davetiyesozleri.com) **said:**  güzel davetiye sözleri ve davetiye metinleri  November 4, 2009 2:48 AM |
| http://sqlblog.com/utility/anonymous.gif | [**How to run a server-side trace | Trigger Happy**](http://sqlblog.com/utility/Redirect.aspx?U=http%3a%2f%2fmikedefehr.com%2f2010%2f05%2f01%2fhow-to-run-a-server-side-trace%2f) **said:**  PingBack from <http://mikedefehr.com/2010/05/01/how-to-run-a-server-side-trace/>  May 1, 2010 4:11 PM |
| http://sqlblog.com/utility/anonymous.gif | [**Twitter Trackbacks for Linchi Shea : Performance Impact: Profiler Tracing vs. Server Side SQL Tracing [sqlblog.com] on Topsy.com**](http://sqlblog.com/utility/Redirect.aspx?U=http%3a%2f%2ftopsy.com%2fsqlblog.com%2fblogs%2flinchi_shea%2farchive%2f2007%2f08%2f01%2ftrace-profiler-test.aspx%3futm_source%3dpingback%26utm_campaign%3dL2) **said:**  PingBack from <http://topsy.com/sqlblog.com/blogs/linchi_shea/archive/2007/08/01/trace-profiler-test.aspx?utm_source=pingback&utm_campaign=L2>  June 22, 2010 5:38 AM |
| http://sqlblog.com/utility/anonymous.gif | [**rich**](http://sqlblog.com/utility/Redirect.aspx?U=http%3a%2f%2fsqlsolace.blogspot.com%2f2010%2f06%2fbookmark-performance-impact-of-profiler.html) **said:**  Pingback from <http://sqlsolace.blogspot.com/2010/06/bookmark-performance-impact-of-profiler.html>  June 22, 2010 6:00 AM |
| http://sqlblog.com/utility/anonymous.gif | [**Mark Broadbent**](http://sqlblog.com/utility/Redirect.aspx?U=http%3a%2f%2ftenbulls.co.uk) **said:**  Good post Linchi.  "the temporary files to buffer the profiler trace data were in directory C:\Documents and Settings\lshea\local settings\Temp\1, and the names of the temporary files had this pattern: Prf\*.tmp."  ..yes in fact the danger is (especially if on server) that you will fill out your C drive. You can minimize the risk somewhat when the GUI is absolutely necessary by creating a symbolic link to another less important drive for this temp directory. I've tried it in the past and it works fine.  Mark Broadbent aka retracement  June 22, 2010 7:04 AM |
| http://sqlblog.com/utility/anonymous.gif | [**Russian SQL Server Club**](http://sqlblog.com/utility/Redirect.aspx?U=http%3a%2f%2fwww.itcommunity.ru%2fblogs%2frsug%2farchive%2f2010%2f07%2f23%2f115308.aspx) **said:**  Для всех, кто работает с SQL Profiler для отслеживания событий на сервере, хорошо известно, как создавать...  July 23, 2010 2:00 PM |
| http://sqlblog.com/utility/anonymous.gif | [**Niraj Dube**](http://sqlblog.com/utility/Redirect.aspx?U=http%3a%2f%2fwww.exact-solutions.com) **said:**  Very interesting thread, guys.  To Greg’s comment that there is “no point starting traces after the fact”: There are tools in the market that allow you to monitor SQL queries on an “always on” basis -- without imposing any performance overhead on the server.  Sort of "have your cake and eat it too".  Full disclosure: I work for one of the vendors who provide such tools.  Vendors use various mechanisms to ensure that their performance monitoring is non-intrusive. Ours does not open a connection to the db nor poll it in any way; instead, we listen to network traffic to passively capture queries and calculate associated performance metrics (such as server reaction time, query response time, round-trip time, etc); and we have a loopback module to capture local traffic. We specialize in high-transaction / high-volume production database monitoring – where, to quote a blogger I saw elsewhere, “you cannot have the odometer slow down the car”.  More about us at: <http://www.exact-solutions.com/products/iwatch>  November 18, 2010 6:40 PM |
| http://sqlblog.com/utility/anonymous.gif | [**SQL SERVER 2008 – Server Side Tracing | Alisson Cardoso**](http://sqlblog.com/utility/Redirect.aspx?U=http%3a%2f%2fwww.alissoncardoso.com.br%2f2011%2f02%2fsql-server-2008-server-side-tracing%2f) **said:**  PingBack from <http://www.alissoncardoso.com.br/2011/02/sql-server-2008-server-side-tracing/>  February 16, 2011 2:38 PM |
| http://sqlblog.com/utility/anonymous.gif | [**Adam Machanic**](http://sqlblog.com/utility/Redirect.aspx?U=http%3a%2f%2fsqlblog.com%2fblogs%2fadam_machanic%2farchive%2f2011%2f04%2f01%2fa-month-of-monitoring-part-1-of-30-a-brief-history-of-monitoring.aspx) **said:**  This post is part 1 of a 30-part series about the Who is Active stored procedure. A new post will run  April 20, 2011 10:43 AM |