## Abstract

The primary goal of this paper is to provide important counters in the PERFMON and SQL server Profiler while tackling resource bottlenecks for diagnosing and troubleshooting SQL Server performance problems in common customer scenarios by using publicly available tools. This paper can be used as a reference or guide for database administrators, database developers and for all MS SQL users who are facing performance issues.

#### **Author Information:**

Website: http://www.aliencoders.com Facebook: https://www.facebook.com/aliencoders Twitter: http://www.twitter.com/aliencoders YouTube: http://www.youtube.com/user/jassics/ Slideshare: http://www.slideshare.net/jassics Pinterest: http://pinterest.com/aliencoders/

#### Rakesh Pandey

Solution Architect, TCS

rakesh.techy@gmail.com

Contact: +91 8420731341



# Contents

Introduction	. 3
Scope	. 3
Important System monitor (Perfmon) counters and their use	. 3
Important SQL profiler counters and their use	. 4
Profiler	. 5
Evaluating Profiler data	. 7
Conclusion	. 8
References	. 9

### Introduction

Performance is one of the major factors for successful execution of any site or business. So performance tuning becomes one of the major tasks for the database administrators.

Many customers can experience an occasional slowdown of their SQL Server database. The reasons can range from a poorly designed database to a system that is improperly configured for the workload. As an administrator, you want to proactively prevent or minimize problems and, when they occur, diagnose the cause and, when possible, take corrective actions to fix the problem. For troubleshooting common performance problems by using publicly available tools such as SQL Server Profiler, System Monitor (Perfmon), and the new Dynamic Management Views in Microsoft SQL Server<sup>TM</sup> 2005.

The primary goal of this paper is to provide useful counters in the PERFMON and SQL server Profiler while handling resource bottlenecks for diagnosing and troubleshooting SQL Server performance problems in common customer scenarios by using publicly available tools.

While using tools such as PERFMON and SQL server profiles we use all of the counters which increase the file size (e.g. trace file) and also the unnecessary time needed for the analysis. This brings us to the goal of this paper, which to showcase the useful or necessary counters. Target audience of this article is the database administrators and developers throughout the world, who are facing database performance issues.

#### Scope

This article is not proposing a new software development methodology. It is not promoting any particular software or utility. Instead, the purpose of this article is to provide important counters in the PERFMON and SQL server Profiler while tackling resource bottlenecks for diagnosing and troubleshooting SQL Server performance problems in common customer scenarios by using publicly available tools.

## Important System monitor (Perfmon) counters and their use

The SQL Statistics object provides counters to monitor compilation and the type of requests that are sent to an instance of SQL Server. You must monitor the number of query compilations and recompilations in conjunction with the number of batches received to find out if the compiles are contributing to high CPU use. Ideally, the ratio of SQL Recompilations/sec to Batch Requests/sec should be very low unless users are submitting ad hoc queries. Batch Requests/sec, SQL Compilations/sec and SQL Recompilations/sec are the key data counters for SQL Server: SQL Statistics.

# Important counters in the Perfmon and SQL server Profiler while tackling resource bottlenecks.

In order to find Memory bottlenecks we can use Memory object (Perfmon) such as Cache Bytes counter for system working set, Pool Non paged Bytes counter for size of unpaged pool and Available Bytes (equivalent of the Available value in Task Manager)

I/O bottlenecks can be traced using Physical Disk Object. Avg. Disk Queue Length represents the average number of physical read and writes requests that were queued on the selected physical disk during the sampling period. If your I/O system is overloaded, more read/write operations will be waiting. If your disk queue length frequently exceeds a value of 2 during peak usage of SQL Server, then you might have an I/O bottleneck.

Avg. Disk Sec/Read is the average time, in seconds, of a read of data from the disk and Avg. Disk Sec/Write is the average time, in seconds, of a write of data to the disk. Please refer below table while analyzing the trace file.

Avg. Disk Sec/read (value) or Avg. Disk Sec/Write	Comment
Less than 10 ms	Very Good
Between 10 - 20 ms	okay
Between 20 - 50 ms	slow, needs attention
Greater than 50 ms	Serious I/O bottleneck

Physical Disk: %Disk Time is the percentage of elapsed time that the selected disk drive was busy servicing read or writes requests. A general guideline is that if this value is greater than 50 per cent, it represents an I/O bottleneck. Avg. Disk Reads/Sec is the rate of read operations on the disk. You need to make sure that this number is less than 85 per cent of the disk capacity. The disk access time increases exponentially beyond 85 per cent capacity and Avg. Disk Writes/Sec is the rate of write operations on the disk. Make sure that this number is less than 85 per cent of the disk capacity. The disk access time increases exponentially beyond 85 time increases exponentially beyond 85 per cent of the disk capacity. The disk access time increases exponentially beyond 85 per cent of the disk capacity.

#### Important SQL profiler counters and their use

If the Perfmon counters indicate a high number of recompiles, the recompiles could be contributing to the high CPU consumed by SQL Server. We would then need to look at the profiler trace to find the stored procedures that were being recompiled. The SQL Server Profiler trace gives us that information along with the reason for the recompilation.

We can use the SP:Recompile and SQL:StmtRecompile events to get this information. The SP:Recompile and the SQL:StmtRecompile event classes indicate which stored procedures and statements have been recompiled. When you compile a stored procedure, one event is generated for the stored procedure and one for each statement that is compiled.

However, when a stored procedure recompiles, only the statement that caused the recompilation is recompiled (not the entire stored procedure as in SQL Server 2000). Some of the more important data columns for the SP:Recompile event class are listed below. The EventSubClass data column in particular is important for determining the reason for the recompile.

SP:Recompile is triggered once for the procedure or trigger that is recompiled and is not fired for an ad hoc batch that could likely be recompiled. In SQL Server 2005, it is more useful to monitor SQL:StmtRecompiles as this event class is fired when any type of batch, ad hoc, stored procedure, or trigger is recompiled.

The key data columns we look at in these events are EventClass, EventSubClass, ObjectID (represents stored procedure that contains this statement), SPID, Start Time, Sql Handle and Text Data.

#### Profiler

Profiler can run, similar to Performance Monitor, either in a GUI mode, or in an automated manner with outputs to files or databases. Sitting and watching the GUI window is usually referred to as SQL-TV. That may be a good way to spot check issues on a database server, or do some ad hoc troubleshooting, but for real performance monitoring you need to set up an automated process and capture the data for processing later. Profiler collects information on events within SQL Server.

Cursors, Database, Errors and Warnings, Locks, Objects, Performance, Scans, Security Audit, Server, Sessions, StoredProcedures, TSQL and Transactions are the broad categories of events used in profiler.Each of these categories has a large number of events within it. Rather than detail all the various options, the following is a minimum set of events for capturing basic TSQL performance.

Stored Procedures - RPC:Completed which records the end point of a remote procedure call (RPC). These are the more common events you'll see when an application is running against your database.

Stored Procedures - PC:Completed which would be calls against procedures from the system itself, meaning you've logged into SQL Server and you're calling procedures through query analyzer, or the server is calling itself from a SQL Agent process.

TSQL: SQL Batch:Completed and these events are registered by TSQL statements running locally against the server which is not the same as a stored procedure call, for example:

SELECT \* FROM table name. Each of these events can then collect a large number of columns of information, each one may or may not be collected from a given event, depending on the event and column in question and each one may collect different data

# Important counters in the Perfmon and SQL server Profiler while tackling resource bottlenecks.

from the event, again depending on the event and column in question. These columns include but are not limited to which can be analyzed with the help of below table.

Column	Usage
Name	
TextData	In the events listed textdata column represents the text of the stored
	procedure call, including the parameters used for the individual call, or the
	SQL batch statement executed.
Application	This may or may not be filled in, depending on the connections string used
Name	by the calling application. In order to facilitate trouble shooting and
	performance tuning, it's worth making it a standard within your
	organization to require this as part of the connection from any application.
LoginName	The NT domain and user name that executed the procedure or SQL batch
CPU	This is an actual measure of CPU time, expressed in milliseconds, used by
	the event in question.
Reads	These are the count of read operations against the logical disk made by the
	event being captured.
Writes	Unlike the Reads, this is the physical writes performed by the procedure or
	SQL batch.
Duration	This is the length of time that the event captured took to complete. In SQL
	Server 2000 this piece of data is in milliseconds. As of SQL Server 2005,
	this has been changed and is now recorded in microseconds. Keep this in
	mind if you're comparing the performance of a query between the two
	servers using Trace events.
SPID	The server process ID of the event captured. This can sometimes be used
	to trace a chain of events.
StartTime	This record the start time of the event.

In short, a great deal of information can be gleaned from Profiler. You may or may not be aware, but in previous versions of SQL Server, running Trace, as Profiler was called previously, against a system could bring the system to its knees before you gathered enough information to get a meaningful set of data. This is no longer true.

It is possible to turn on enough events and columns to impact the performance of a system but with a reasonable configuration Profiler will use much less than 1% of system resources. That does not mean that you should load up counters on the GUI and sit back to watch your server. This will add some load and can be easily avoided. Instead, take advantage of the extended stored procedures that are available for creating and managing SQL Traces. These will allow you to gather data and write it out to disk (either a local one or a remote one).

This means that you'll have the data in a transportable format that you can import into databases or spreadsheets to explore search and clean to your heart's content. You can also write the results directly to a database, but I've generally found this to be slower, therefore having more impact on the server, than writing to a file. This is supported by

recommendations in the BOL as well. Here is a basic script to create a trace for output to a file.

In order to further limit the data collected, you may want to add filters to restrict by application or login in order to eliminate noise:

EXEC sp\_trace\_setfilter @trace\_id, @columnid, @logicaloperator, @comparisonoperator, @value

So, for example to keep any trace events from intruding on the data collection above, we could add below fields.

EXEC sp\_trace\_setfiter

@trace\_id = @TraceId, @columnid = 10, --app name column @logicaloperator = 1, -- logical "or" @comparisonoperator = 0, -- equals @value = N'SQL Profiler'

The output can be loaded into the SQL Server Profiler GUI for browsing or you can run this procedure to import the trace data into a table.

```
SELECT * INTO temp_trc
```

FROM fn\_trace\_gettable('c:\temp\my\_trace.trc', DEFAULT);

#### **Evaluating Profiler data**

Now that you've collected all this data about your system, what do you do with it? There are many ways in which you can use Profiler data. The simplest approach is to use the data from the Duration column as a measure of how slowly a procedure is performing.

After collecting the information and moving it into a table, you can begin writing queries against it. Grouping by stored procedure name, stripping off the parameters from the string, will allow you to use aggregates, average, maximum, and minimum, to outline the poorest performing procedures on the system. From there you can look at their CPU or I/O and query plans in order to identify the tuning opportunities available to you.

With the release of SQL Server 2005, one additional piece of functionality was added to Profiler which will radically increase your ability to identify where you have performance issues. You now have the ability to load a performance monitor log file and a profiler trace file into the Profiler. This allows you to identify points of time when the server was under load and see what was occurring within the database at the same time.

# Important counters in the Perfmon and SQL server Profiler while tackling resource bottlenecks.

EvenCles     TexDues     ApplicationName     NTUseName     CoprName     CPU     Reads       Audit Logout     SELECT N'Testing Connection'     SQLAgent - 3     fritcheyger     CORP\f     0     81       SQLiBatchStarting     SELECT N'Testing Connection'     SQLAgent - A     fritcheyger     CORP\f     0     81       SQLiBatchStarting     DECUTE msdb.dbo.sp.sqlagent.get.pt     SQLAgent - A     fritcheyger     CORP\f     0     0       SQLiBatchCompleted     DECUTE msdb.dbo.sp.sqlagent.get.pt     SQLAgent - A     fritcheyger     CORP\f     0     0       SQLiBatchCompleted     SELECT N'Testing Connection'     SQLAgent - A     fritcheyger     CORP\f     0     0       SQLiBatchCompleted     SELECT N'Testing Connection'     SQLAgent - A     fritcheyger     CORP\f     0     0       SQLiBatchCompleted     SELECT N'Testing Connection'     SQLAgent - A     fritcheyger     CORP\f     0     0       SQLiBatchCompleted     SELECT N'Testing Connection'     SQLAgent - A     fritcheyger     CORP\f     0     0       SQLiBatchCompleted     SELECT N'Testing Connection'     SQLAgent - A     fritcheyger     CORP\f     0     0       SQLiBatchCompleted     SELECT N'Testing Connection'<	EventCla Audit SQLIBA	ET Logout	н н   St Çt 🗐 [ В		Þ			1.00.1	- #
EventClass     TestDate     ApplicationName     NTUseName     LogoName     CPU     Reads       Audit Logout     SQLAgent - 3     fritcheygr     CORP\f     0     BI       SQLIBatchistarting     SELECT N'Testing Connection'     SQLAgent - A     fritcheygr     CORP\f     0     BI       SQLIBatchistarting     DECUTE modb.dbo.sp_sqlagent_get_pet     SQLAgent - A     fritcheygr     CORP\f     0     0       SQLEBatchistarting     DECUTE modb.dbo.sp_sqlagent_get_pet     SQLAgent - A     fritcheygr     CORP\f     0     0       SQLEBatchistarting     SELECT N'Testing Connection'     SQLAgent - A     fritcheygr     CORP\f     0     0       SQLEBatchistarting     SELECT N'Testing Connection'     SQLAgent - A     fritcheygr     CORP\f     0     0       SQLEBatchistarting     SELECT N'Testing Connection'     SQLAgent - A     fritcheygr     CORP\f     0     0       SQLEBatchistarting     SELECT N'Testing Connection'     SQLAgent - A     fritcheygr     CORP\f     0     0       SQLBatchistarting     UPART modb.obs.sylalerts SET last     SQLAgent - A     fritcheygr     CORP\f     0     0       SQLBatchistarting     UPART modb.dbsylalerts SET last     SQLAgent - A     fritc	EventCla Audit SQL184 SQL184	Logout			•			L and L	
Audit Logout       SQLAgent - 3       fritcheyge       CORP\f       0       SI         SQLIBatchStarting       SELECT N'Testing Connection'       SQLAgent - A       fritcheyge       CORP\f       0       SI         SQLIBatchStarting       DELOT N'Testing Connection'       SQLAgent - A       fritcheyge       CORP\f       0       0         SQLIBatchStarting       DEGCUTE msdb.dbo.sp.sqlagent_get_pt       SQLAgent - A       fritcheyge       CORP\f       0       0         SQLIBatchCompleted       DEGCUTE msdb.dbo.sp.sqlagent_get_pt       SQLAgent - A       fritcheyge       CORP\f       0       0         SQLIBatchCompleted       SELECT N'Testing Connection'       SQLAgent - A       fritcheyge       CORP\f       0       0         SQLIBatchCompleted       SELECT N'Testing Connection'       SQLAgent - A       fritcheyge       CORP\f       0       0         SQLIBatchCompleted       SELECT N'Testing Connection'       SQLAgent - A       fritcheyge       CORP\f       0       0         SQLIBatchCompleted       SELECT N'Testing Connection'       SQLAgent - A       fritcheyge       CORP\f       0       0         SQLIBatchCompleted       SELECT N'Testing Connection'       SQLAgent - A <td< th=""><th>Audit SQLIBA SQLIBA</th><th>Logout</th><th>TexDate</th><th></th><th></th><th></th><th></th><th>Land</th><th></th></td<>	Audit SQLIBA SQLIBA	Logout	TexDate					Land	
SQL:BatchStarting SELECT N'TESTINg Connection' SQLAgent - A fritcheygr CORP\f 0 SQLAGENT - A SQLAGENT - A fritcheygr CORP\f 0 SQLAGENT - A SQLAGENT - A Fritcheygr CORP\f 0 SQLAGENT - A	SQLIBA SQLIBA				ApplicationName	NTUserName	LogeName	OPU	Reads
SQLIBatchCompleted SELECT N'Testing Connection' SQLAgent - A fritCheygr CDRP\f 0 ( SQLiBatchStarting DOGUTE msdb.dbo.sp_sqlagent_get_pe SQLAgent - A fritCheygr CDRP\f 14 ( SQLiBatchCompleted DOGUTE msdb.dbo.sp_sqlagent_get_pe SQLAgent - A fritCheygr CDRP\f 14 ( SQLiBatchCompleted SELECT N'Testing Connection' SQLAgent - A fritCheygr CDRP\f 0 ( SQLiBatchCompleted SELECT N'Testing Connection' SQLAgent - A fritCheygr CDRP\f 0 ( SQLiBatchCompleted SELECT N'Testing Connection' SQLAgent - A fritCheygr CDRP\f 0 ( SQLiBatchCompleted SELECT N'Testing Connection' SQLAgent - A fritCheygr CDRP\f 0 ( SQLiBatchCompleted SELECT N'Testing Connection' SQLAgent - A fritCheygr CDRP\f 0 ( SQLiBatchCompleted SELECT N'Testing Connection' SQLAgent - A fritCheygr CDRP\f 0 ( SQLiBatchCompleted SELECT N'Testing Connection' SQLAgent - A fritCheygr CORP\f 0 ( SQLiBatchCompleted SELECT N'Testing Connection' SQLAgent - A fritCheygr CORP\f 0 ( SQLiBatchCompleted UPDATE msdb.dbo.sysalerts SET last SQLAgent - A fritCheygr CORP\f 15 11 SQLIBatchCompleted UPDATE msdb.dbo.sysalerts SET last SQLAgent - A fritCheygr CORP\f 15 11 SQLIBatchCompleted UPDATE msdb.dbo.sysalerts SET last SQLAgent - A fritCheygr CORP\f 15 11 SQLIBatchCompleted UPDATE msdb.dbo.sysalerts SET last SQLAgent - A fritCheygr CORP\f 15 11 SQLIBATCHCOMPLETE UPDATE msdb.dbo.sysalerts SET last SQLAgent - A fritCheygr CORP\f 15 11 SQLIBATCHCOMPLETE UPDATE msdb.dbo.sysalerts SET last SQLAgent - A fritCheygr CORP\f 15 11 SQLIBATCHCOMPLETE UPDATE msdb.dbo.sysalerts SET last SQLAgent - A fritCheygr CORP\f 15 11 SQLIBATCHCOMPLETE UPDATE msdb.dbo.sysalerts SET last SQLAgent - A fritCheygr CORP\f 15 11 SQLIBATCHCOMPLETE UPDATE msdb.dbo.sysalerts SET last SQLAgent - A fritCheygr CORP\f 15 11 SQLIBATCHCOMPLETE UPDATE MSD.STIL AMBISTICE AMBISTICE AMBISTELE AMBISTICE AMBISTICE AMBISTICE AMBISTELE AMBISTELE AMB	SQL184	stchStarting			SQLAgent - J	fritcheygr	CORP\f	0	83
SQLIBAtchStarting DECUTE modb.dbo.sp_sqlagent_get_pt SQLAgent - A fritCheygr CDRP\f 14 1 SQLiBatchCompleted DECUTE modb.dbo.sp_sqlagent_get_pt SQLAgent - A fritCheygr CDRP\f 14 1 SQLiBatchCompleted SELECT N'Testing Connection' SQLAgent - A fritCheygr CDRP\f 0 0 SQLiBatchStarting SELECT N'Testing Connection' SQLAgent - A fritCheygr CDRP\f 0 0 SQLiBatchCompleted SELECT N'Testing Connection' SQLAgent - A fritCheygr CDRP\f 0 0 SQLiBatchCompleted SELECT N'Testing Connection' SQLAgent - A fritCheygr CDRP\f 0 0 SQLiBatchCompleted SELECT N'Testing Connection' SQLAgent - A fritCheygr CDRP\f 0 0 SQLiBatchCompleted SELECT N'Testing Connection' SQLAgent - A fritCheygr CDRP\f 0 0 SQLiBatchCompleted UPATE modb.dbo.sysalerts SET last SQLAgent - A fritCheygr CORP\f 15 11 10 - 0 0 - 0	Contraction of the		SELECT N'Testir	g Connection'	SQLAgent - A	fritcheygr	CORP\f		
SQL:BatchStarting SELECT N Testing Connection' SQLAgent - A fritcheygr CORP\f 14 1 SQL:BatchStarting SELECT N Testing Connection' SQLAgent - A fritcheygr CORP\f 0 0 SQL:BatchStarting SELECT N Testing Connection' SQLAgent - A fritcheygr CORP\f 0 0 SQL:BatchStarting SELECT N Testing Connection' SQLAgent - A fritcheygr CORP\f 0 0 SQL:BatchStarting UPDATE mdb.dbo.syslerts SET 1ast SQLAgent - A fritcheygr CORP\f 0 0 SQL:BatchStarting UPDATE mdb.dbo.syslerts SET 1ast SQLAgent - A fritcheygr CORP\f 0 0 SQL:BatchStarting UPDATE mdb.dbo.syslerts SET 1ast SQLAgent - A fritcheygr CORP\f 0 0 SQL:BatchStarting UPDATE mdb.dbo.syslerts SET 1ast SQLAgent - A fritcheygr CORP\f 15 1 SQL:BatchStarting UPDATE mdb.dbo.syslerts SET 1ast SQLAgent - A fritcheydr CORP\f 15 1 SQL:BatchStarting UPDATE mdb.dbo.syslerts SET 1ast SQLAgent - A fritcheydr CORP\f 15 1 SQL:BatchStarting UPDATE msdb.dbo.syslerts SET 1ast SQLAgent - A fritcheydr CORP\f 15 1 SQL:BatchStarting UPDATE msdb.dbo.syslerts SET 1ast SQLAgent - A fritcheydr CORP\f 15 1 SQL:BatchStarting UPDATE msdb.dbo.syslerts SET 1ast SQLAgent - A fritcheydr CORP\f 15 1 SQL:BatchStarting UPDATE msdb.dbo.syslerts SET 1ast SQLAgent - A fritcheydr CORP\f 15 1 SQL:BatchStarting UPDATE msdb.dbo.syslerts SET 1ast SQLAgent - A fritcheydr CORP\f 15 1 SQL:BatchStarting UPDATE msdb.dbo.syslerts SET 1ast SQLAgent - A fritcheydr CORP\f 15 1 SQL:BatchStarting UPDATE msdb.dbo.syslerts SET 1ast SQLAgent - A fritcheydr CORP\f 15 1 SQL:BatchStarting Context MesiJorita Mesi	SOLTEA	stchCompleted	SELECT N'TESTIF	g Connection'	SQLAgent - A	fritcheygr	CDRP\f		0
SQL:BatchStarting SELECT N'TESTING Connection' SQLAgent - A fritCheygr CORP\F 0 C SQL:BatchCompleted SELECT N'TESTING Connection' SQLAgent - A fritCheygr CORP\F 0 C SQL:BatchCompleted SELECT N'TESTING Connection' SQLAgent - A fritCheygr CORP\F 0 C SQL:BatchCompleted SELECT N'TESTING Connection' SQLAgent - A fritCheygr CORP\F 0 C SQL:BatchCompleted SELECT N'TESTING Connection' SQLAgent - A fritCheygr CORP\F 0 C SQL:BatchCompleted SELECT N'TESTING Connection' SQLAgent - A fritCheygr CORP\F 0 C SQL:BatchCompleted SELECT N'TESTING Connection' SQLAgent - A fritCheygr CORP\F 0 C SQL:BatchCompleted UPDATE msdb.dbo.sysalerts SET last SQLAgent - A fritCheygr CORP\F 15 11 SQL:BatchCompleted UPDATE msdb.dbo.sysalerts SET last SQLAgent - A fritCheygr CORP\F 15 11 SQL:BatchCompleted UPDATE msdb.dbo.sysalerts SET last SQLAgent - A fritCheygr CORP\F 15 11 SQL:BatchCompleted UPDATE msdb.dbo.sysalerts SET last SQLAgent - A fritCheygr CORP\F 15 11 SQL:BatchCompleted UPDATE msdb.dbo.sysalerts SET last SQLAgent - A fritCheygr CORP\F 15 11 SQL:BatchCompleted UPDATE msdb.dbo.sysalerts SET last SQLAgent - A fritCheygr CORP\F 15 11 SQL:BatchCompleted UPDATE msdb.dbo.sysalerts SET last SQLAgent - A fritCheygr CORP\F 15 11 SQL:BatchCompleted UPDATE msdb.dbo.sysalerts SET last SQLAgent - A fritCheygr CORP\F 15 11 SQL:BatchCompleted UPDATE msdb.dbo.sysalerts SET last SQLAgent - A fritCheygr CORP\F 15 11 SQL:BatchChCompleted UPDATE msdb.dbo.sysalerts SET last SQLAgent - A fritCheygr CORP\F 15 11 SQL:BatchChCompleted UPDATE msdb.dbo.sysalerts SET last SQLAgent - A fritCheygr CORP\F 15 11 SQL:BatchChCompleted UPDATE MSd:ST/SG AMB:ST/SG		tchStarting	EXECUTE #5 db. dt	o.sp_sqlagent_get_pe.	., SQLAgent - A	fritcheyer	COLUMN TO A STATE		
SQL:Batch:completed SELECT N'Testing Connection' SQLAgent - A fritcheygr CORP\f 0 ( SQL:Batch:Starting SELECT N'Testing Connection' SQLAgent - A fritcheygr CORP\f 0 ( SQL:Batch:Starting UPDATE m50b.dbo.sysalerts SET 1ast SQLAgent - A fritcheygr CORP\f 0 ( SQL:Batch:Completed UPDATE m50b.dbo.sysalerts SET 1ast SQLAgent - A fritcheygr CORP\f 15 11 SQL:Batch:Completed UPDATE m50b.dbo.sysalerts SET 1ast SQLAgent - A fritcheygr CORP\f 15 11 SQL:Batch:Starting UPDATE m50b.dbo.sysalerts SET 1ast SQLAgent - A fritcheygr CORP\f 15 11 SQL:Batch:Starting UPDATE m50b.dbo.sysalerts SET 1ast SQLAgent - A fritcheygr CORP\f 15 11 SQL:Batch:Starting UPDATE m50b.dbo.sysalerts SET 1ast SQLAgent - A fritcheygr CORP\f 15 11 SQL:Batch:Starting UPDATE m50b.dbo.sysalerts SET 1ast SQLAgent - A fritcheygr CORP\f 15 11 SQL:Batch:Starting UPDATE m50b.dbo.sysalerts SET 1ast SQLAgent - A fritcheygr CORP\f 15 11 SQL:Batch:Starting UPDATE m50b.dbo.sysalerts SET 1ast SQLAgent - A fritcheygr CORP\f 15 11 SQL:Batch:Starting UPDATE m50b.dbo.sysalerts SET 1ast SQLAgent - A fritcheygr CORP\f 15 11 SQL:Batch:Starting UPDATE m50b.dbo.sysalerts SET 1ast SQLAgent - A fritcheygr CORP\f 15 11 SQL:Batch:Starting UPDATE m50b.dbo.sysalerts SET 1ast SQLAgent - A fritcheygr CORP\f 15 11 SQL:Batch:Starting UPDATE m50b.dbo.sysalerts SET 1ast SQLAgent - A fritcheygr CORP\f 15 11 SQL:Batch:Starting UPDATE m50b.dbo.sysalerts SET 1ast SQLAgent - A fritcheygr CORP\f 15 11 SQL:Batch:Starting UPDATE m50b.dbo.sysalerts SET 1ast SQLAgent - A fritcheygr CORP\f 15 11 SQL:Batch:Starting UPDATE M50b.dbo.sysalerts SET 1ast SQL:Batch:Starting UPDATE SET 1ast	1.0000			Contraction of the second s	sqLAgent - A				3
SQL:BatchStarting SELECT N'TESTING Connection' SQLAGent - A fritCheygr CORP\f 0 Contest Completed SELECT N'TESTING Connection' SQLAGent - A fritCheygr CORP\f 0 Contest Completed UPDATE mscb.dbo.sysalerts SET last SQLAGent - A fritCheygr CORP\f 15 Li SQL:BatchCompleted UPDATE mscb.dbo.sysalerts SET last SQLAGent - A fritCheygr CORP\f 15 Li Contest choose and the state of the state	1.0		SELECT N'TESTIN	o connection'	SQLAgent - A				
SQL:BatchCompleted SQL:Ba	20000			The second share the second second second second	and the second se	the set of			0
SQL:BatchCompleted UPDATE msdb.dbo.sysalerts SET last SQLAQent - A fritCheyge CORP.f 15 11 Sol:BatchCompleted UPDATE msdb.dbo.sysalerts SET last SOLAGENT - A fritCheyge CORP.f 15 11 100- 00- 00- 00- 00- 00- 00- 00- 00-				Sector Contraction Contractor					
SOLIBATCH COMPLETED UPDATE HISCH. dbb.svsalerts SET last SOLADERT - A Tritchever CORPIT 15 11 100- 0- 0- 0- 0- 0- 0- 0- 0- 0-		111221005000270		and a standard in the schedule hard and a second second				0	0
100 - 60 -				tions in the second s				2	110
e0 - e0 -	SOLIES	tchcompleted	UPDATE MS db. dbc	sysalerts set last .	SOLADENT - A	Tritchever	CORPAT	15	12
	80 - 60 - 40 -	$\Delta$		IJ	A	P	VIV		
2.83 N BROTAFFOR THE BROTAFFOR TOTAL LIPETY/PUT/DUT	80 - 60 - 40 - 20 - 7	. M 8136156 AN 81371	1 MILETY/LE MILETY/L	1 AME: 27/16 AME: 2011		AME: SOLAR AN	X//X	813	9112 4
	80 - 40 - 20 - 0 - 1:36:41 \$/4/20	107							9:12 #
7.18 Batch Requests/Sec MSSqLSGF20051Sq \\FRITCHEVG1XP	80 - 40 - 20 - 0 - 1:36:41 \$/4/20	107				Computer	Mir		9:12 4
49.53 AVG. DISK Queue FhysicalDiskTotal \\FRITCHEYGLXP	80 - 40 - 20 - 0 - 1:36:41 5/4/20	507 Scale 2.82 7.18	Counter N Processor Time	Object Processor	nstance	Computer \\PRITCI \\PRITCI	Min (EYG1># (EYG1>#		
	80 - 40 - 20 - 0 - 1:36:41 5/4/20	5cale 2.82 7.16 49.53	Counter N Processor Time Batch Requests/sec Avg. D15k Queue	Object 1 Processor MssqL\$07200515Q Phys1ca1D15k	notance	Computer \\PRITCI \\PRITCI	Min (EYG1>9 (EYG1>9 (EYG1>9		2
	80 - 40 - 20 - 0 - 1:36:41 5/4/20	5cale 2.82 7.16 49,53 2.36	Counter N Processor Time Batch Requests/sec Avg. Disk Queue N Processor Time	Object ] Processor MSSQL\$GF20051SQ PhysIcalDisk Processor	notance	Computer \\FRITCI \\FRITCI \\FRITCI \\FRITCI	Min (EVG1)07 (EVG1)07 (EVG1)07 (EVG1)07		
7.18 Batch Requests/sec MSSQL\$GF2005:5Q \\FRITCHEYG1XP	00 - 40 - 40 - 20 - 0 - 0 - 5,44/20	107							9132
	80 - 40 - 20 - 0 - 1 - 1 - 5/4/20	107							9132 A
	80 - 40 - 20 - 0 - 1:36:41 \$/4/20	io7 Scale	Counter	Object	nitance	Computer	Mir		
7.18 Batch Requests/sec MSSQL50F200515Q \\FRITCHEVG1XP	80 - 40 - 20 - 0 - 1:36:41 \$/4/20	io7 Scale	Counter	Object	nitance	Computer	Mir		
7.18 Batch Requests/sec MSSQL50F200515Q \\FRITCHEVG1XP	80 - 40 - 20 - 0 - 1:36:41 \$/4/20	io7 Scale	Counter	Object	nitance	Computer	Mir		
	80 - 40 - 20 - 0 - 1:36:41 \$/4/20	Scale	Counter N Processor Time	Object Processor	nitance	Computer \\IN(ITC)	Min (EYG1>#		
49.53 AVG. DISK Queue FitysicalDisk Total \\FRITCHEVG1XP	80 - 40 - 20 - 0 - 1:36:41 \$/4/20	Scale	Counter N Processor Time	Object Processor	nitance	Computer \\IN(ITC)	Min (EYG1>#		
	80 - 40 - 20 - 0 - 1:36:41 \$/4/20	507 Scale 2.82 7.18	Counter N Processor Time Batch Requests/sec	Object 1 Processor MssqL\$GF200519Q	nstance	Computer \\PRITCI \\PRITCI	Min (EYG1># (EYG1>#		3
	80 - 40 - 20 - 0 - 1:36:41 5/4/20	5cale 2.82 7.16 49.53	Counter N Processor Time Batch Requests/sec Avg. D15k Queue	Object 1 Processor MssqL\$07200515Q Phys1ca1D15k	notance	Computer \\PRITCI \\PRITCI	Min (EYG1>9 (EYG1>9 (EYG1>9		3
1.36 N Processor Time Processor 1 \\PRITCHEVG1XP	80 - 40 - 20 - 0 - 1:36:41 5/4/20	5cale 2.82 7.16 49,53 2.36	Counter N Processor Time Batch Requests/sec Avg. Disk Queue N Processor Time	Object ] Processor MSSQL\$GF20051SQ PhysIcalDisk Processor	notance	Computer \\FRITCI \\FRITCI \\FRITCI \\FRITCI	Min (EVG1)07 (EVG1)07 (EVG1)07 (EVG1)07		3

For Memory bottlenecks we can use SQL Server: Buffer Manager Object and Low Buffer cache hit ratio, Low Page life expectancy, High number of Checkpoint pages/sec and High number Lazy writes/sec counters in Profiler.

## Conclusion

We can say that database administrators, database developers and all MS SQL users who are facing performance issues will be able to save their precious time while using Perfmon and SQL server Profiler. As we know that capturing and analyzing a trace file is tedious and time taking task which can be eased with the help of this paper.

Most of the time as soon as we receive a database performance issue we use all the counters irrespective of their usage and importance and so, this paper will provide important counters in the PERFMON and SQL server Profiler while tackling resource bottlenecks. And as we know that performance is one of the major factors for successful execution of any site or business. So performance tuning becomes one of the major tasks for database administrators and this paper will help in reducing their burden.

### References

```
http://msdn.microsoft.com/en-us/library/ms191246.aspx
http://msdn.microsoft.com/en-us/library/ms187929.aspx
http://www.sql-server-performance.com/2005/sql-server-performance-monitor-
coutners/
```