Tools to Analize Query Performance

Performance Monitoring and Tuning Tools

Microsoft SQL Server provides a set of tools for monitoring events in SQL Server and for tuning the physical database design. The choice of tool depends on the type of monitoring or tuning to be done and the particular events to be monitored.

Tool	Description
sp_trace_setfilter (Transact-SQL)	SQL Server Profiler tracks engine process events (i.e. start of a batch or a transaction, enable to monitor server and database activity (for example, deadlocks, fatal errors, or login activity)). One can capture SQL Server Profiler data to a SQL Server table or a file for later analysis, and can replay the events captured on SQL Server step by step, to see exactly what happened.
SQL Server Distributed Replay	Microsoft SQL Server Distributed Replay can use multiple computers to replay trace data, simulating a mission-critical workload.
Monitor Resource Usage (System Monitor)	System Monitor primarily tracks resource usage (i.e. the number of buffer manager page requests in use) enabling to monitor server performance and activity using predefined objects and counters or user-defined counters to monitor events. System Monitor (Performance Monitor in Microsoft Windows NT 4.0) collects counts and rates rather than data about the events (i.e. memory usage, number of active transactions, number of blocked locks, or CPU activity). One can set thresholds on specific counters to generate alerts that notify operators. System Monitor works on Microsoft Windows Server and Windows operating systems. SQL Server Profiler monitors Database Engine events and System Monitor monitors resource usage associated with server processes.
Open Activity Monitor (SQL Server Management Studio)	The Activity Monitor in SQL Server Management Studio is useful for ad hoc views of current activity and graphically displays information about: Processes running on an instance of SQL Server, Blocked processes, Locks, User activity.
Live Query Statistics	Displays real-time statistics about query execution steps. This data is available while the query is executing, so these execution statistics are extremely useful for debugging query performance issues.
SQL Trace	Transact-SQL stored procedures that create, filter, and define tracing: sp_trace_create, sp_trace_generateevent, sp_trace_setevent, sp_trace_setfilter, sp_trace_setstatus.
Error Logs	The Windows application event log provides an overall picture of events occurring on the Windows Server and Windows operating systems, events in SQL Server, SQL Server Agent, and full-text search. It contains information about events in SQL Server that is not available elsewhere. One can use the information in the error log to troubleshoot SQL Server-related problems.
System Stored Procedures (Transact-SQL)	The SQL Server system stored procedures provide an alternative for many monitoring tasks: sp_who - Reports snapshot information about current SQL Server users and processes, the currently executing statement and whether the statement is blocked. sp_lock - Reports snapshot information about locks, including the object ID, index ID, type of lock, and type or resource to which the lock applies. sp_spaceused - Displays an estimate of the current amount of disk space used by a table (or a whole database). sp_monitor - Displays statistics, including CPU usage, I/O usage, and the amount of time

	idle since **sp_monitor** was last executed.
DBCC (Transact-SQL)	DBCC (Database Console Command) statements enable you to check performance statistics and the logical and physical consistency of a database.
Built-in Functions (Transact-SQL)	Built-in functions display snapshot statistics about SQL Server activity since the server was started; these statistics are stored in predefined SQL Server counters. I.e. **@@CPU_BUSY** contains the amount of time the CPU has been executing SQL Server code; **@@CONNECTIONS** contains the number of SQL Server connections or attempted connections; and **@@PACKET_ERRORS** contains the number of network packets occurring on SQL Server connections.
Trace Flags (Transact-SQL)	Trace flags display information about a specific activity within the server and are used to diagnose problems or performance issues (for example, deadlock chains).
Database Engine Tuning Advisor	Database Engine Tuning Advisor analyzes the performance effects of Transact-SQL statements executed against databases you want to tune. Database Engine Tuning Advisor provides recommendations to add, remove, or modify indexes, indexed views, and partitioning.

The choice of a monitoring tool depends on the event or activity to be monitored.

Event or activity	SQL Server Profiler	Distributed Replay	System Monitor	Activity Monitor	Transact-SQL	Error logs
Trend analysis	Yes		Yes			
Replaying captured events	Yes (From a single computer)	Yes (From multiple computers)				
Ad hoc monitoring	Yes			Yes	Yes	Yes
Generating alerts			Yes			
Graphical interface	Yes		Yes	Yes		Yes
Using within custom application	Yes, by using SQL Server Profiler system stored procedures.				Yes	

How to use Statistics IO to Improve Query Performance

SQL Server's STATISTICS IO reporting is a great tool to help performance tune queries. The goal of performance tuning is to make the query run faster. One way to get a faster query is to reduce the amount of data that a query is processing. The STATISTICS IO output helps with performance tuning because the data it shows acts as a measuring stick for the performance tuning changes and it provides a good way of isolating the query changes.

STATISTICS IO provides detailed information about the impact that the query has on SQL Server, telling the number of logical reads (including LOB), physical reads (including read-ahead and LOB), and how many times a table was scanned. This information helps you to establish whether or not the choices made by the optimizer are as efficient as possible at the time.

STATISTICS IO can be set as an option when execute a query. A message is sent via the connection that made a query, telling the cost of the query in terms of the actual number of physical reads from the disk and logical reads from memory, by the query.

To show IO statistics on your query, you first need to execute:

```
use AdventureWorks2012; go
-- To show IO statistics on your query -- This applies to your current session only
SET STATISTICS IO ON;
GO
```

```
select * from Person.ContactType
```

Results | Messages

	ContactTypeID	Name	ModifiedDate
1	1	Accounting Manager	2002-06-01 00:00:00.000
2	2	Assistant Sales Agent	2002-06-01 00:00:00.000
3	3	Assistant Sales Representative	2002-06-01 00:00:00.000
4	4	Coordinator Foreign Markets	2002-06-01 00:00:00.000
5	5	Export Administrator	2002-06-01 00:00:00.000
6	6	International Marketing Manager	2002-06-01 00:00:00.000
7	7	Marketing Assistant	2002-06-01 00:00:00.000
8	8	Marketing Manager	2002-06-01 00:00:00.000
9	9	Marketing Representative	2002-06-01 00:00:00.000
10	10	Order Administrator	2002-06-01 00:00:00.000
11	11	Owner	2002-06-01 00:00:00.000

Query executed successfully. DESKTOP-ATJN5FL\SQLEX

Messages:

```
(20 row(s) affected)
Table 'ContactType'. Scan count 1, logical reads 2, physical reads 1, read-ahead reads 0,
lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
```

Initial examination of the STATISTICS IO output contains:

- **Logical reads:** The number of 8kB pages SQL Server had to read from the buffer cache (memory) in order to process and return the results of the query. The more pages that need to be read, the slower will be the query.
- **Worktables/Workfiles:** These are temporary objects that SQL Server creates in tempdb in order to process query results.
- **Lob Logical Reads:** The number of large objects (e.g. varchar(max)) SQL is having to read.

LOB Logical Reads (0) – no read in any Large Objects (text, ntext, image, varchar(max), nvarchar(max) and varbinary(max))
LOB Physical Reads (0) - the number of physical reads the server performed to fetch the necessary pages to satisfy the query.
LOB Read-Ahead Reads (0) - the number of physical reads satisfied by the Read-Ahead mechanism.

The **STATISTICS IO Output -** part of a real query with over 700 lines long.
Scan count (208,450) - the optimizer has chosen a plan that caused this object to be read repeatedly. This number is used as a gauge later on in the process and one will see what object it is being scanned when go over the execution plan. This number does not change unless the query is modified.
Logical Reads (716751) - the actual number of pages read from the data cache. This is the number to focus on because it does not change unless one change the actual query structure or index structures.
Physical Reads (1421) - the number of pages actually read from the disk. These are the pages that weren't already in cache. If there is a requested page that is not in cache, it will read it from disk and place it in cache, then use that page. If you were to run your query multiple times in a row, you would see your physical reads decrease and ultimately become 0 (so long as there is enough room in memory to store all of the pages required).
Read-Ahead Reads (996) - how many of the physical reads were satisfied by SQL Servers 'Read-ahead' mechanism. This is directly tied to physical reads, so if there are no physical

reads, you will have 0 for Read-Ahead reads. This number will fluctuate as pages are swapped in/out of memory.

Analyze Queries with SHOWPLAN Results in SQL Server Profiler

One can add Showplan event classes to a trace definition that cause SQL Server Profiler to gather and display query plan information in the trace. One can also extract Showplan events in a separate XML file.

Extracting Showplan events from the trace: At trace configuration time, using the Events Extraction Settings tab (appear when one select a one of the Showplan events on the Events Selection tab); Using the Extract SQL Server Events option on the File menu; By extracting and saving individual events by right-clicking a specific event and choosing Extract Event Data**.**

Showplan Events

Event name	Description
Performance statistics	Indicates the first time a compiled Showplan is cached, when it is recompiled, and when it is dropped from the plan cache. The TextData column contains the Showplan in XML format.
Showplan All	Displays the query plan with full compilation details of the executed Transact-SQL statement.
Showplan All For Query Compile	Occurs when a query is compiled or recompiled on SQL Server. This is the compile time counterpart of the Showplan All event. Showplan All occurs when a query is executed. Showplan All For Query Compile occurs when a query is compiled.
Showplan Statistics Profile	Displays the query plan with full run-time details of the Transact-SQL statement being executed, including the actual number of rows passing through each operation.
Showplan Text	Displays as binary data the query plan tree of the Transact-SQL statement being executed.
Showplan Text (Unencoded)	Displays as text the query plan tree of the Transact-SQL statement being executed. This event class displays the same information as Showplan Text, except that this event class displays text instead of binary data.
Showplan XML	Displays the query plan with full data collected during query optimization. This event is generated only when a query plan is optimized.
Showplan XML For Query Compile	Displays the query plan when the query is compiled.
Showplan XML Statistics Profile	Displays the query plan with full run-time details in XML format.

Showplan is a feature in SQL Server to display and read query plans. Two types of Showplans: one at query compilation time (when an optimized query plan is produced) (by using query SET option) and the second at query runtime (when the optimized query plan is executed) (by using Profiler Trace events). The various SET options available are:

Type	***Compile Time***	***Runtime***
Legacy Showplan	SET SHOWPLAN_ALL ON SET SHOWPLAN_TEXT ON	SET STATISTICS PROFILE ON
Showplan XML (Preferred)	SET SHOWPLAN_XML ON	SET STATISTICS XML ON

SSMS uses Showplan to display a graphical representation of the query plan. To view the graphical plan, click on the Execution Plan tab in the Results Pane. The graphical plan is read from top-to-bottom, right-to-left. When you select any operator in the query tree or simply hover the mouse on it, you will see a tooltip that describes the operator. It displays the Query Optimizer

cost estimates (operator and subtree costs, number of rows, row size, etc.) and additional information like output columns, predicates. The detailed operator information is shown in the Properties window (View --> Properties Window), which is usually displayed on the extreme right frame in SSMS. To save the graphical showplan to a file you can right click on the Execution Plan and select 'Save Execution Plan As'. The query plan is saved with extension '.sqlplan' and can be reloaded into SSMS anytime.

<u>SET STATISTICS XML (Transact-SQL)</u>

SET STATISTICS xml {on|off} - is set at execute / run time, not at parse time

When SET STATISTICS XML is ON, SQL Server returns execution information for each statement after executing it and until the option is set to OFF. It returns output as nvarchar(max) for applications, as a set of XML documents. Each statement after the SET STATISTICS XML ON statement is reflected in the output by a single document (with the details of the execution steps). The output shows run-time information (i.e. the costs, accessed indexes, and types of operations performed, join order, the number of times a physical operation is performed, the number of rows each physical operator produced). It can be found in \Microsoft SQL Server\100\Tools\Binn\schemas\sqlserver\2004\07\showplan\showplanxml.xsd.

```sql
USE AdventureWorks2012;
GO
SET STATISTICS XML ON;
GO
-- First query.
SELECT BusinessEntityID
FROM HumanResources.Employee
WHERE NationalIDNumber = '509647174';
GO
-- Second query.
SELECT BusinessEntityID, JobTitle
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Production%';
GO
SET STATISTICS XML OFF;
GO
```

Results | Messages

	BusinessEntityID
1	3

	Microsoft SQL Server 2005 XML Showplan
1	<ShowPlanXML xmlns="http://schemas.microsoft.com...

	BusinessEntityID	JobTitle
1	26	Production Control Manager
2	27	Production Supervisor - WC60
3	28	Production Technician - WC60
4	29	Production Technician - WC60
5	30	Production Technician - WC60
6	31	Production Technician - WC60
7	32	Production Technician - WC60
8	33	Production Technician - WC60

	Microsoft SQL Server 2005 XML Showplan
1	<ShowPlanXML xmlns="http://schemas.microsoft.com...

References:

https://www.slideshare.net/MarekMasko/sql-server-using-tools-to-analyze-query-performance

https://dataginger.com/2014/02/11/using-tools-to-analyze-sql-server-query-performance/

https://docs.microsoft.com/en-us/sql/relational-databases/performance/performance-monitoring-and-tuning-tools?view=sql-server-2017

https://bertwagner.com/2017/12/05/how-to-use-statistics-io-to-improve-your-query-performance/

https://www.red-gate.com/simple-talk/sql/performance/simple-query-tuning-with-statistics-io-and-execution-plans/

https://docs.microsoft.com/en-us/sql/tools/sql-server-profiler/analyze-queries-with-showplan-results-in-sql-server-profiler?view=sql-server-2017

https://blogs.msdn.microsoft.com/sqlqueryprocessing/2006/09/29/introduction-to-showplan/

https://docs.microsoft.com/en-us/sql/t-sql/statements/set-statistics-xml-transact-sql?view=sql-server-2017