Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 6, June 2015, pg.381 – 386

RESEARCH ARTICLE

PERFORMANCE TUNING IN MICROSOFT SQL SERVER DBMS

Sapna Dahiya, Pooja Ahlawat
M.Tech student, Department of CSE, R.N College of Engineering & Management
Assistant Professor, Department of CSE, R.N College of Engineering & Management
Sapna30july@gmail.com

ABSTRACT: ***Now a day's database size in so rapidly growing up in the big organisation that performance tuning as coming an important subject for discussion. I am writing down this paper to discuss the importance of performance tuning in large-scale organizations, which host massive applications.***
Many of you have encountered problems of website going slow while surfing Internet. To enhance the performance of applications hosted in front end, it is very important to tune the databases in the back end. As the number of Users in the site started growing problem started occurring.
As I have performed a vast research on this topic, therefore I want to help you do this by sharing my data access optimization experience during research work and share my findings with you in this article. I just hope this might enable you to optimize your data access routines in existing systems, or to develop data access routines in an optimized way in your future projects.

1. INTRODUCTION

Performance tuning is the process of improving system's performance so that system's ability improves to accept higher loads. In this article I would mostly be focusing on performance tuning in MS SQL Server.

I have highlighted different aspects, which should be considered while tuning your databases and common bottlenecks, which degrade the performance of your system.

Please focus that primary goal of this article is "data access performance optimization in transactional (OLTP) SQL Server databases". But, most of the optimization techniques are roughly the same for other database platforms.

I document focuses of the importance of appropriate Indexes for querying date in the tables. It focuses on best practices, which should be followed while designing the querying. Best techniques of query optimization, sql server performance tools such as sql server profiler and tuning advisor. It focuses on monitoring performance counters through perfmon and sql dmv's. Dealing with CPU bottlenecks and memory contention situations. SQL Database Engine is a highly IO intensive system so I have emphasized on gathering IO stats as well.
Dealing with blocking and deadlock kind of situations also plays a major role in tuning the performance of out database engine. And managing system tables such as tempdb. I have also

discussed in brief several wait types, which should be monitored closely to tune performance, and little focus is also paid on instance level setting which enhance the performance of sql server

2. Various Bottlenecks which adversely impact the performance of System

Bottlenecks occur when a resource reaches its capacity, causing the performance of the entire system to slow down. Insufficient or misconfigured resources, malfunctioning components, and incorrect requests for resources typically cause bottlenecks by a program.
There are five major resource areas that can cause bottlenecks and affect server performance: physical disk, memory, process, CPU, and network.

2.1 Hard Disk Bottleneck

Logical Disk\% Free Space This measures the percentage of free space on the selected logical disk drive.

Physical Disk\% Idle Time This measures the percentage of time the disk was idle during the sample interval
Physical Disk\Avg. Disk Sec/Read This measures the average time, in seconds, to read data from the disk
Memory\Cache Bytes This indicates the amount of memory being used for the file system cache. There may be a disk bottleneck if this value is greater than 300MB.

2.2 Memory Bottleneck

Memory\Available Mbytes This measures the amount of physical memory, in megabytes, available for running processes. If this value is less than 5 percent of the total physical RAM, that means there is insufficient memory, and that can increase paging activity. To resolve this problem, you should simply add more memory.

2.2.1 Checkpoint pages/sec It is triggered automatically by sqls server database engine to flush dirty pages to disk

2.2.2 High number of Lazy writes/sec: Lazy Writer come into picture whenever there is memory pressure on sql server.

2.2.3 High number of Page reads/sec : It’s alarming when number of disk reads in more than cache reads.

2.2.4. Low Buffer cache hit ratio: A consistent value below 90% indicates that more physical memory is needed on the server.

2.2.5. Low Page Life Expectancy: Ideally this value should not be less than 300 seconds.

2.3 CPU Bottlenecks*:*

High CPU Utilization is an important bottleneck, which adversely impact the performance of system. If SQL Server is using high CPU we need to terminate few batched responsible for it.

2.3.1. % Processor Time Processor time gives CPU Utilization in percentage

2.3.2 %Privilege Time gives time that processor take to executes system calls

2.4 Deadlocks and Blocking

On SQL Server, blocking occurs when one SPID holds a lock on a specific resource and a second SPID attempts to acquire a conflicting lock type on the same resource

We can check blocking through:

SP_WHO2 ACTIVE;

And

Select * from sys.dm.exec_request;

Blocking highly degrade the performance of system.

Deadlock

A deadlock occurs when two or more tasks permanently block each other by each task having a lock on a resource, which the other tasks are trying to lock

To view deadlock information, the Database Engine provides monitoring tools in the form of two trace flags, and the deadlock graph event in SQL Server Profiler.
Trace Flag 1204 and Trace Flag 1222
When deadlocks occur, trace flag 1204 and trace flag 1222 return information that is captured in the SQL Server error log. Trace flag 1204 reports deadlock information formatted by each node involved in the deadlock. Trace flag 1222 formats deadlock information, first by processes and then by resources. It is possible to enable both trace flags to obtain two representations of the same deadlock event.

SQL Server itself handle deadlock situation by terminating the current batch involved in the deadlock and throwing error 1205.

3. How to tackle these bottlenecks

3.1 Applying proper indexing in the table columns in the Databases

Well, some could argue whether implementing proper indexing should be the first step in the performance optimization process for a database. But I would prefer applying indexing properly in the database in the first place, because of the following two reasons:

This will allow you to achieve the best possible performance in the quickest amount of time in a production system.

Applying/creating indexes in the database will not require you to do any application modification, and thus will not require any build and deployment.
Of course, this quick performance improvement can be achieved if you find that indexing is not properly done in the current database. However, if indexing is already done, it is still very important to fine-tune them regularly which involves following steps:

Make sure that every table in your database has a primary key**.**

Create appropriate indexes either clustered or non-clustered depending upon he search criteria.

CREATE INDEX IndexName ON tabl*e*name (LastName, FirstName)

In case indexes are already created, look for index fragmentation. And depending upon the percentage of fragmentation either rebuild or reorganize your indexes.
Index fragmentation can be identified through SQL DMVs and DBCC commands:

SELECT * sys.dm_db_index_physical_stats;

DBCC ShowContig

If Index fragmentation is greater than 50% then rebuild indexes;

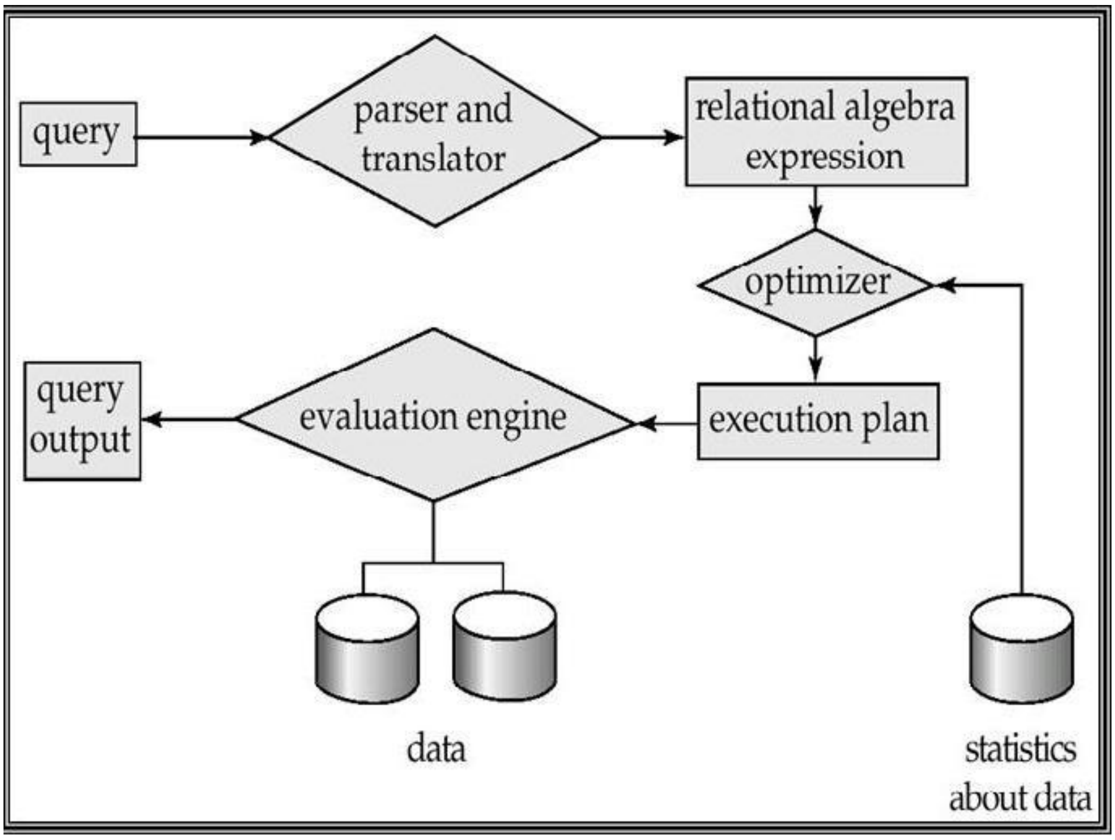
ALTER INDEX ALL ON TableName REBUILD WITH (FILLFACTOR=90,ONLINE=ON) ;

Else Reorganize Index:

ALTER INDEX ALL ON TableName REORGANIZES;

3.2 QUERY OPTIMIZATION

In SQL Server Query Processing, Query Optimization is one of the most important tasks which database engine performs.



SQL Server optimizer uses up to date stats to optimize query and choose the best available execution plan. Statistics contain information about relations.

Therefore statistics should be accurate and up to date.

This can be done through the stored procedure SP_UPDATESTATS

3.3 Using SQL Server Performance Tools

3.3.1 SQL Server Profiler and Tuning Advisor

Microsoft SQL Server Profiler is a graphical user interface to SQL Trace for monitoring an instance of the Database Engine or Analysis Services. You can capture and save data about each event to a file or table to analyze later. For example, you can monitor a production environment to see which Stored Procedures are affecting performance by executing too slowly.

Tuning advisor helps to get the performance report that is generated by SQL Profiler and provide the appropriate indexing. It takes one or more SQL statements as input and invokes the Automatic Tuning Optimizer to perform SQL tuning on the statements.

3.3.2 SQL Query Analyzer:

Microsoft® SQL Server™ 2000 SQL Query Analyzer is a graphical tool that allows
You to:

Create queries and other SQL scripts and execute them against SQL Server databases. (Query window)

- Quickly create commonly used database objects from predefined scripts. (Templates)
- Quickly copy existing database objects. (Object Browser scripting feature)
- Execute stored procedures without knowing the parameters. (Object Browser procedure execution feature)
- Debug stored procedures. (T-SQL Debugger)
- Debug query performance problems. (Show Execution Plan, Show Server Trace, Show Client Statistics, Index Tuning Wizard)
- Locate objects within databases (object search feature), or view and work with objects. (Object Browser)
- Quickly insert, update, or delete rows in a table. (Open Table window)
- Create keyboard shortcuts for frequently used queries. (Custom query shortcuts feature)
- Add frequently used commands to the **Tools** menu. (Customized **Tools** menu feature)

3.3.3 SQL DMV's

SQL DMV's means SQL dynamic management views. By querying a single DMV, sys.dm_os_performance_counters to be precise, you can collect counter information that you would receive from PerfMon for the various SQL Server counters.
SQL DMV's in it self a vast subject to cover up. It can be used to capture any performance counter whether related to OS performance, index fragmentation, locking, deadlocking, wait types, system infi etc.

4. Conclusion

The principle objective of performance tuning is to improve the performance of SQL Server so that he can process queries faster and make better use of system resources. Performance tuning now a days is very hot research topic for researchers and data analysts. In this paper I presented common performance bottlenecks and their troubleshooting. And we concluded that for enhance the performance of sql database engine it is very important to have proper indexes created and fine tune them periodically. Secondly it is very important to measure the performance of system proactively on regular basis using different performance measurement tools and troubleshoot them to enhance the throughput of system and ensure proper utilization of resources.

References:

1. SQL Server 2012 Query Performance Tuning (Kindle Edition) by Grant Fritchey
2. SQL Server Query Performance Tuning Distilled By Sajal Dam (Author) 2nd Edition
3. SQL Server 2008 Query Performance Tuning Distilled By Sajal Dam (Author), Grant Fritchey (Author)
4. Microsoft SQL Server 2014 Query Tuning & Optimization by Nevarez (Author)