SQL Standards

1. Version Compliance
2. Perform volume maintenance tasks
3. Lock Pages in Memory
4. Trace 1118 in startup parameters
5. Check\_MaxMemory

Check\_MinMemory

1. Check\_AWEEnabled
2. Check\_CPU\_Affinity
3. Check\_ErrorLogRetention
4. Check\_BuiltinAdmins

10) Check\_AnonymousLogin

11) Check\_NoUserLogins

12) Default Connections

13) Check\_RemoteAccess

14) Check\_BackupCompressionDefault

15) Check\_AdHocWorkloads

16) Check\_MDOP (Max Degree of Parallelism )

17) Check\_RecoveryMode

Check\_TempDBFileLocation

Check\_TempDbDataFileCount

Check\_TempDb\_FileSizeAllocation

Check\_TempDb\_LogSize

Check\_TempDbAutoGrowth

18) User DBs data file autogrow size is set properly

19) User DBs log file autogrow size is set properly

20) Check\_AutoClose

21) Check\_AutoCreateStats

22) Check\_AutoShrink

23) Check\_AutoUpdateStats

24) Check\_SnapshotIsolation

25) Check\_DriveExists\_H , O , T & E

26) Set the NTFS allocation unit size to 64 KB

1. **Version Compliance:**

**(Remediation Requires Downtime or a restart of services?: Yes)**

To know the current value, run below command from SSMS:

SELECT SERVERPROPERTY('ProductVersion') [Version]

Apply Latest CU on top of RTM.

1. **Perform volume maintenance tasks:**

**(Remediation Requires Downtime or a restart of services?: NO)**

True for service account

In order for SQL Server to be able to perform instant file initialization the SQL Server service account must be granted the Perform Volume Maintenance Task security permission. This can be done by using the Local Security Policy Editor via:

*Administrative Tools(or open secpol.msc) > then go to the tab Local Security Policy and then Local Policies > then go to User Rights Assignment > double click Perform volume maintenance tasks and check if the SQL Server service account exists. If doesn’t exists, add it.*

NOTE: Pay close attention to what other Groups on the server have the right ‘Perform Volume Maintenance Task. By default it is granted to the Local Administrators group and if the SQL Server service account is part of this group then it will have the permission too.

If we need to add this right to the SQL Server service account, we will need to restart the SQL Server service in order for the change to take effect.

Make sure below accounts are added:

**1. System (if the SQL Server service is running under Local System Account)**

**2. Service Account (if the SQL Server service is running under Domain Account)**

**3. Servername\SQLServerMSSQLUser$Servername$yourinstancename**

1. **Lock Pages in Memory:**

**(Remediation Requires Downtime or a restart of services?: NO)**

True for service account

Go to *Administrative Tools(or open secpol.msc) – Local Security Policy and then Local Policies – User Rights Assignment -doubleClick LockPages In memory and check if the SQL Service account is added or not*.

Make sure below accounts are added:

**1. System (if the SQL Server service is running under Local System Account)**

**2. Service Account (if the SQL Server service is running under Domain Account)**

**3. Servername\SQLServerMSSQLUser$Servername$yourinstancename**

For more details refer: <http://msdn.microsoft.com/en-us/library/ms190730.aspx>

1. **Trace 1118 in startup parameters:**

**(Remediation Requires Downtime or a restart of services?: YES)**

Run below query from SSMS to find trace 1118 is added in startup parameters:

DBCC TRACESTATUS(1118)

If the status is 1 that means it is already added.

If the status is 0 add it by running the below query from SSMS:

**Go to :**

*AllPrograms>MicrosoftSQLServer2008>ConfigurationTools>SQLServerConfigurationManager>SQLServerServices>right click on SQLServer(MSSQLServer)>Click Advanced tab>expand Startup parameters and check for the existence of -t1118 paramete. If the parameter doesn’t exists, add it in the parameters.*

***Exapmle:*** *-dH:\MSSQL10\_50.MSSQLSERVER\MSSQL\DATA\master.mdf;-eH:\MSSQL10\_50.MSSQLSERVER\MSSQL\Log\ERRORLOG;-lO:\MSSQL10\_50.MSSQLSERVER\MSSQL\DATA\mastlog.ldf;-T1118*

1. **Max Memory**

**(Remediation Requires Downtime or a restart of services?: NO)**

To know the current value, run below command from SSMS:

SELECT name,value FROM sys.configurations WHERE NAME = 'max server memory (MB)'

OR sp\_configure 'max server memory (MB)'

Use below image to know the configuration to set.

Run below query to change the memory settings:

USE master;

GO

EXEC sp\_configure 'max server memory (MB)', 'value in MBs';

RECONFIGURE WITH OVERRIDE

**Min Memory**

**(Remediation Requires Downtime or a restart of services?: NO)**

To know the current value, run below command from SSMS:

SELECT name,value FROM sys.configurations WHERE NAME = 'min server memory (MB)'

OR sp\_configure 'min server memory (MB)'

Use below image to know the configuration to set.

Run below query to change the memory settings:

USE master;

GO

EXEC sp\_configure 'min server memory (MB)', 'value in MBs';

RECONFIGURE WITH OVERRIDE

Below is standard configuration for SQL/OS

Machine generated alternative text:
HW Memory OS Reserved SQL Memory 
10 
12 
13 
14 
15 
16 
17 
18 
19 
20 
21 
16 
32 
144 
32 
53 
74 
81 
17 _ 
102 
109 
116 

1. **AWE Enabled**

**(Remediation Requires Downtime or a restart of services?: NO)**

To know the current value, run below command from SSMS:

SELECT name,value FROM sys.configurations WHERE NAME = 'awe enabled'

OR sp\_configure 'awe enabled'

Value= 0 (if it is 64 bit server); 1 ((if it is 32 bit server))

Run below query to change the setting:

USE master;

GO

EXEC sp\_configure 'awe enabled', '0';

RECONFIGURE WITH OVERRIDE

1. **CPU Affinity**

**(Remediation Requires Downtime or a restart of services?: NO)**

To know the current value, run below command from SSMS:

SELECT COUNT(1) OutOfCompliance FROM sys.configurations WHERE name IN ('affinity64 I/O mask','affinity I/O mask','affinity64 mask','affinity mask') AND value != 0

Value=0

To remediate run the following:

DECLARE @ConfigName nvarchar(35);

DECLARE Config CURSOR FOR

SELECT name

FROM sys.configurations

WHERE name IN ('affinity64 I/O mask','affinity I/O mask','affinity64 mask','affinity mask')

AND value != 0;

OPEN Config;

FETCH NEXT FROM Config INTO @ConfigName;

WHILE @@FETCH\_STATUS = 0

BEGIN

EXECUTE ('EXECUTE sp\_configure '''+@ConfigName+''', 0;');

FETCH NEXT FROM Config INTO @ConfigName;

END

CLOSE Config;

DEALLOCATE Config;

RECONFIGURE WITH OVERRIDE;

GO

1. **Error Log Retention:**

**(Remediation Requires Downtime or a restart of services?: NO)**

Value=99

Script to Check: EXECUTE master.dbo.xp\_instance\_regread N'HKEY\_LOCAL\_MACHINE','Software\Microsoft\MSSQLServer\MSSQLServer','NumErrorLogs'

If the above script doesn’t return the value of ‘99’, run below script.

Script/Process to Remediate: EXECUTE xp\_instance\_regwrite N'HKEY\_LOCAL\_MACHINE', N'Software\Microsoft\MSSQLServer\MSSQLServer', N'NumErrorLogs', REG\_DWORD, 99

1. **BUILTIN\Administrators does not exist:**

**(Remediation Requires Downtime or a restart of services?: NO)**

SELECT name FROM sys.server\_principals WHERE name = 'BUILTIN\Administrators'

**Note: Before deleting this account from Logins, run beow command to add the account SQL Service account as sysadmin.**

USE [master]

GO

CREATE LOGIN [REDMOND\SCCM\_SQL\_Providers] FROM WINDOWS WITH DEFAULT\_DATABASE=[master]

GO

EXECUTE sp\_addsrvrolemember N'**SQL Service account’**, N'sysadmin'

GO

To delete **BUILTIN\Administrators** account, from SSMS got to>Security>Logins and delete the login BUILTIN\Administrators.

1. **NT AUTHORITY\ANONYMOUS LOGON does not exist:**

**(Remediation Requires Downtime or a restart of services?: NO)**

SELECT name FROM sys.server\_principals WHERE name = 'NT AUTHORITY\ANONYMOUS LOGON'

From SSMS got to>Security>Logins and delete the login 'NT AUTHORITY\ANONYMOUS LOGON.

1. **No individual user accounts exist (only SG and Machine logins):**

**(Remediation Requires Downtime or a restart of services?: NO)**

SELECT name NumNotCompliant FROM sys.server\_principals WHERE type = 'U' AND name NOT LIKE '%$' AND name NOT LIKE '%\[\_]%' AND name NOT IN ('REDMOND\smsinsta','NT AUTHORITY\SYSTEM','NT AUTHORITY\NETWORK SERVICE','redmond\smsard09')

Make sure that only DL accounts exist in the Logins and delete the individual accounts. Make a note of all the logins that are deleted.

**Note: Before deleting this account from Logins, run beow command to add the account SQL Service Account as sysadmin.**

USE [master]

GO

CREATE LOGIN [REDMOND\SCCM\_SQL\_Providers] FROM WINDOWS WITH DEFAULT\_DATABASE=[master]

GO

EXECUTE sp\_addsrvrolemember N' **SQL Service Account ‘**, N'sysadmin'

GO

To delete individual accounts, From SSMS got to>Security>Logins and delete the individual user accounts if any exist.

Make a note of the logins that you deleted and document it in the history of the WI.

1. **Default Connections**

**(Remediation Requires Downtime or a restart of services?: NO)**

To know the current value, run below command from SSMS:

SELECT name,value FROM sys.configurations WHERE NAME = 'user options'

OR sp\_configure 'user options'

Value should be equal to 0

Run below query to change the setting:

USE master;

GO

EXEC sp\_configure 'user options', '0';

RECONFIGURE WITH OVERRIDE

1. **Allow remote connections**

**(Remediation Requires Downtime or a restart of services?: NO)**

To know the current value, run below command from SSMS:

SELECT name,value FROM sys.configurations WHERE NAME = 'remote access'

OR sp\_configure 'remote access'

Value should be equal to 1

Run below query to change the setting:

USE master;

GO

EXEC sp\_configure 'remote access', '1';

RECONFIGURE WITH OVERRIDE

1. **Backup compression**

**(Remediation Requires Downtime or a restart of services?: NO)**

To know the current value, run below command from SSMS:

SELECT name,value FROM sys.configurations WHERE NAME = 'backup compression default'

OR sp\_configure 'backup compression default'

Value should be equal to 1

Run below query to change the setting:

USE master;

GO

EXEC sp\_configure 'backup compression default', 1;

RECONFIGURE WITH OVERRIDE

1. **Optimize for ad-hoc workloads**

**(Remediation Requires Downtime or a restart of services?: NO)**

To know the current value, run below command from SSMS:

SELECT name,value FROM sys.configurations WHERE NAME = 'optimize for ad hoc workloads'

OR sp\_configure 'optimize for ad hoc workloads'

Value should be equal to 1

Run below query to change the setting:

USE master;

GO

EXEC sp\_configure 'optimize for ad hoc workloads', 1;

RECONFIGURE WITH OVERRIDE

1. **MDOP(max degree of parallelism)**

**(Remediation Requires Downtime or a restart of services?: NO)**

To know the current value, run below command from SSMS:

SELECT name,value FROM sys.configurations WHERE NAME = 'max degree of parallelism'

-OR- sp\_configure 'max degree of parallelism'

Value should be equal to 0

Run below query to change the setting:

USE master;

GO

EXEC sp\_configure 'max degree of parallelism', 0;

RECONFIGURE WITH OVERRIDE

1. **Recovery Mode**

**(Remediation Requires Downtime or a restart of services?: NO)**

To know the current value, run below command from SSMS:

SELECT name AS [Database Name], recovery\_model\_desc AS [Recovery Model] FROM sys.databases

Use below script to find and change the Recovery Model for all databases to SIMPLE.

USE master

GO

-- Declare a variable to store the value [database name] returned by FETCH.

DECLARE @dbname sysname, @cmd varchar(1000)

-- Declare a cursor to iterate through the list of databases

DECLARE db\_recovery\_cursor CURSOR FOR

select name FROM sys.databases WHERE recovery\_model != 3

-- Open the cursor

OPEN db\_recovery\_cursor

-- Perform the first fetch and store the value in a variable.

FETCH NEXT FROM db\_recovery\_cursor INTO @dbname

-- loop through cursor until no more records fetched

WHILE @@FETCH\_STATUS = 0

BEGIN

IF (SELECT DATABASEPROPERTYEX(@dbname,'RECOVERY')) <> 'SIMPLE' and @dbName <> 'tempdb' BEGIN

-- create the alter database command for each database

SET @cmd = 'ALTER DATABASE "' + @dbname + '" SET RECOVERY SIMPLE'

-- alter each dataabase setting the recovery model to SIMPLE

EXEC(@cmd)

PRINT @dbname

end

FETCH NEXT FROM db\_recovery\_cursor INTO @dbname

END

-- close the cursor and deallocate memory used by cursor

CLOSE db\_recovery\_cursor

DEALLOCATE db\_recovery\_cursor

1. **All tempdb files on T drive:**

**(Remediation Requires Downtime or a restart of services?: YES)**

1. **Tempdb Configuration and should be dedicated with separate drive:** **data files for tempdb are pre-allocated to the appropriate size:**

**(Remediation Requires Downtime or a restart of services?: NO)**

1. **User DBs data file autogrow size is set properly:**

**(Remediation Requires Downtime or a restart of services?: NO)**

1. Run below query to determine the db\_size and autogrowth of each database data/log file:

SELECT DB\_NAME(database\_id) DBName, name FileName,(size\*8.0/1024) FileSize, (growth\*8.0/1024) DbAutoGrowth FROM sys.master\_files WHERE database\_id >4

1. Determine the expected autogrowth value for each data/log file of the DB using below rules:
2. Small DBs (1GB or less): 100MB;
3. Medium DBs (1GB – 10GB): 200MB;
4. Large DBs (10GB or greater): 1GB
5. **User DBs log file autogrow size is set properly:**

**(Remediation Requires Downtime or a restart of services?: NO)**

You can change the autogrowth for data and log files at the same time.

Here change the value of FILEGROWTH as appropriate.

1. **Auto\_close = off**

**(Remediation Requires Downtime or a restart of services?: NO)**

To know the current value, run below command from SSMS:

SELECT name AS [Database Name], is\_auto\_close\_on AS [AutoClose Value] FROM sys.databases

OR see qc tool ruleset

The returned value should be equal to 0 for all databases.

Use below script to find and change the Auto\_close option for all databases on the server as per Standard.

USE master

GO

-- Declare a variable to store the value [database name] returned by FETCH.

DECLARE @dbname sysname, @cmd varchar(1000)

-- Declare a cursor to iterate through the list of databases

DECLARE db\_autoclose\_cursor CURSOR FOR

SELECT name FROM sys.databases where is\_auto\_close\_on=1

-- Open the cursor

OPEN db\_autoclose\_cursor

-- Perform the first fetch and store the value in a variable.

FETCH NEXT FROM db\_autoclose\_cursor INTO @dbname

-- loop through cursor until no more records fetched

WHILE @@FETCH\_STATUS = 0

BEGIN

IF (SELECT DATABASEPROPERTYEX(@dbname,'IsAutoClose')) <> '0'

BEGIN

-- create the alter database command for each database

SET @cmd = 'ALTER DATABASE "' + @dbname + '" SET AUTO\_CLOSE OFF'

EXEC(@cmd)

PRINT @dbname

end

FETCH NEXT FROM db\_autoclose\_cursor INTO @dbname

END

-- close the cursor and deallocate memory used by cursor

CLOSE db\_autoclose\_cursor

DEALLOCATE db\_autoclose\_cursor

1. **auto\_create\_statistics = on**

**(Remediation Requires Downtime or a restart of services?: NO)**

To know the current value, run below command from SSMS:

SELECT name AS [Database Name], is\_auto\_create\_stats\_on AS [AutoCreateStats Value] FROM sys.databases

OR see qc tool ruleset

The returned value should be equal to 1 for all databases.

Use below script to find and change the auto\_create\_statistics option for all databases on the server as per Standard.

USE master

GO

-- Declare a variable to store the value [database name] returned by FETCH.

DECLARE @dbname sysname, @cmd varchar(1000)

-- Declare a cursor to iterate through the list of databases

DECLARE db\_autocreatestatistics\_cursor CURSOR FOR

SELECT name FROM sys.databases where is\_auto\_create\_stats\_on=0

-- Open the cursor

OPEN db\_autocreatestatistics\_cursor

-- Perform the first fetch and store the value in a variable.

FETCH NEXT FROM db\_autocreatestatistics\_cursor INTO @dbname

-- loop through cursor until no more records fetched

WHILE @@FETCH\_STATUS = 0

BEGIN

IF (SELECT DATABASEPROPERTYEX(@dbname,'IsAutoCreateStatistics')) <> 1

BEGIN

-- create the alter database command for each database

SET @cmd = 'ALTER DATABASE "' + @dbname + '" SET AUTO\_CREATE\_STATISTICS ON'

EXEC(@cmd)

PRINT @dbname

end

FETCH NEXT FROM db\_autocreatestatistics\_cursor INTO @dbname

END

-- close the cursor and deallocate memory used by cursor

CLOSE db\_autocreatestatistics\_cursor

DEALLOCATE db\_autocreatestatistics\_cursor

1. **auto\_shrink = off**

**(Remediation Requires Downtime or a restart of services?: NO)**

To know the current value, run below command from SSMS:

SELECT name AS [Database Name], is\_auto\_shrink\_on AS [AutoShrink Value] FROM sys.databases

OR see qc tool ruleset

The returned value should be equal to 0 for all databases.

Use below script to find and change the auto\_shrink option for all databases on the server as per Standard.

USE master

GO

-- Declare a variable to store the value [database name] returned by FETCH.

DECLARE @dbname sysname, @cmd varchar(1000)

-- Declare a cursor to iterate through the list of databases

DECLARE db\_autoshrink\_cursor CURSOR FOR

SELECT name FROM sys.databases where is\_auto\_shrink\_on=1

-- Open the cursor

OPEN db\_autoshrink\_cursor

-- Perform the first fetch and store the value in a variable.

FETCH NEXT FROM db\_autoshrink\_cursor INTO @dbname

-- loop through cursor until no more records fetched

WHILE @@FETCH\_STATUS = 0

BEGIN

IF (SELECT DATABASEPROPERTYEX(@dbname,'IsAutoShrink')) <> '0'

BEGIN

-- create the alter database command for each database

SET @cmd = 'ALTER DATABASE "' + @dbname + '" SET AUTO\_SHRINK OFF'

EXEC(@cmd)

PRINT @dbname

end

FETCH NEXT FROM db\_autoshrink\_cursor INTO @dbname

END

-- close the cursor and deallocate memory used by cursor

CLOSE db\_autoshrink\_cursor

DEALLOCATE db\_autoshrink\_cursor

1. **auto\_update\_statistics = on-- check and take action if required**

**(Remediation Requires Downtime or a restart of services?: NO)**

To know the current value, run below command from SSMS:

SELECT name AS [Database Name], is\_auto\_update\_stats\_on AS [auto\_update\_statistics Value] FROM sys.databases

OR see qc tool ruleset

The returned value should be equal to 1 for all databases.

Use below script to find and change the auto\_update\_statistics option for all databases on the server as per Standard.

USE master

GO

-- Declare a variable to store the value [database name] returned by FETCH.

DECLARE @dbname sysname, @cmd varchar(1000)

-- Declare a cursor to iterate through the list of databases

DECLARE auto\_update\_statistics CURSOR FOR

SELECT name FROM sys.databases where is\_auto\_update\_stats\_on=0

-- Open the cursor

OPEN auto\_update\_statistics

-- Perform the first fetch and store the value in a variable.

FETCH NEXT FROM auto\_update\_statistics INTO @dbname

-- loop through cursor until no more records fetched

WHILE @@FETCH\_STATUS = 0

BEGIN

IF (SELECT DATABASEPROPERTYEX(@dbname,'IsAutoUpdateStatistics')) <> '1'

BEGIN

-- create the alter database command for each database

SET @cmd = 'ALTER DATABASE "' + @dbname + '" SET AUTO\_UPDATE\_STATISTICS ON'

EXEC(@cmd)

PRINT @dbname

end

FETCH NEXT FROM auto\_update\_statistics INTO @dbname

END

-- close the cursor and deallocate memory used by cursor

CLOSE auto\_update\_statistics

DEALLOCATE auto\_update\_statistics

1. **allow\_snapshot\_isolation = on**

**(Remediation Requires Downtime or a restart of services?: NO)**

To know the current value, run below command from SSMS:

SELECT name AS [Database Name], snapshot\_isolation\_state AS [AllowSnapshotIsolations Value] FROM sys.databases

OR see qc tool ruleset

The returned value should be equal to 1 for all databases.

Use below script to find and change the allow\_snapshot\_isolation option for all databases on the server as per Standard.

USE master

GO

-- Declare a variable to store the value [database name] returned by FETCH.

DECLARE @dbname sysname, @cmd varchar(1000)

-- Declare a cursor to iterate through the list of databases

DECLARE allowsnapshotisolation\_cursor CURSOR FOR

SELECT name FROM sys.databases where snapshot\_isolation\_state=0

-- Open the cursor

OPEN allowsnapshotisolation\_cursor

-- Perform the first fetch and store the value in a variable.

FETCH NEXT FROM allowsnapshotisolation\_cursor INTO @dbname

-- loop through cursor until no more records fetched

WHILE @@FETCH\_STATUS = 0

BEGIN

IF EXISTS(SELECT name FROM sys.databases where snapshot\_isolation\_state=0) BEGIN

-- create the alter database command for each database

SET @cmd = 'ALTER DATABASE "' + @dbname + '" SET ALLOW\_SNAPSHOT\_ISOLATION ON'

EXEC(@cmd)

PRINT @dbname

end

FETCH NEXT FROM allowsnapshotisolation\_cursor INTO @dbname

END

-- close the cursor and deallocate memory used by cursor

CLOSE allowsnapshotisolation\_cursor

DEALLOCATE allowsnapshotisolation\_cursor

1. **Server has the proper drives( HOTE):**

**(Remediation Requires Downtime or a restart of services?: YES)**

Server should have H, O, T, E drives. If it’s not having these drives, notify the infra team.

1. **Server has the proper drives( HOTE):**

The atomic unit of storage in SQL Server is a page, which is 8 KB in size. Eight physically contiguous pages make up an extent (which is 64 KB in size). SQL Server uses extents to store data. Therefore, on a SQL Server machine, the NTFS allocation unit size for hosting SQL database files (including tempdb) should be 64 KB.

The atomic unit of storage in SQL Server is a page, which is 8 KB in size. Eight physically contiguous pages make up an extent (which is 64 KB in size). SQL Server uses extents to store data. Therefore, on a SQL Server machine, the NTFS allocation unit size for hosting SQL database files (including tempdb) should be 64 KB.

1. **Disable TCP offloading and RSS settings**

If you observe random connectivity issues such as transport-level errors or packet transmission errors when TCP offloading (TCP Chimney Offload feature) offloads the processing of TCP/IP packets from the processor to the network adapter, to free the CPU for other tasks. • Receive-side scaling (RSS) helps distributes the processing of incoming network traffic on multiprocessor systems. It load balances the network processing efficiently among the CPUs. To check your current settings, at the command prompt, run the netsh command: $ netsh int tcp show global Here is sample output from the command. In this example, Receive-Side Scaling State and Chimney Offload State are both disabled. To get task offload information about a specific connection, at the command prompt, run: netstat –t and check the value of the offload state column. To disable TCP offloading and RSS, run these commands at the command prompt: netsh int ip set global taskoffload=disabled netsh int tcp set global chimney=disabled netsh int tcp set global rss=disabled netsh int tcp set global netdma=disabled To learn more about these settings, see: • TCP Chimney Offload, Receive Side Scaling, and Network Direct Memory Access features and Introduction to Receive Side Scaling on the Microsoft website

<https://support.microsoft.com/en-in/help/951037/information-about-the-tcp-chimney-offload-receive-side-scaling-and-net>

1. **Exclude SQL Server files from antivirus software**

When you configure your antivirus software settings, make sure that you exclude your SQL Server files and directories from virus scanning. For details and a list of files and directories to exclude, see How to choose antivirus software to run on computers that are running SQL Server on the Microsoft website

1. **Change the cost threshold of parallelism**

The cost threshold of parallelism determines which queries are candidates for parallel execution. This property’s default value is 5, which means that the optimizer switches to a parallel plan if the cost of a serial plan is more than 5 (estimated elapsed time, in seconds). We recommend that you set this property to a higher number. The default value was appropriate back when processors had high price tags, processing power was low, and query processing was slower than it is now. Processors today are much faster. As a result, comparatively smaller queries (for example, given a cost threshold of 32) won’t benefit a lot from parallel execution, especially given the overhead associated with coordinating parallel execution. In most cases, a cost threshold of parallelism setting of 50 is a good starting point. You can tune it further during plan cache review, as follows: $ sp\_configure 'cost\_threshold\_for\_parallelism'

1. **Use trace flags to improve performance**

Consider using SQL Server trace flags that are applicable to your environment to enhance performance. For example: 16 AWS Prescriptive Guidance Best practices for deploying SQL Server on Amazon EC2 Install the latest patches • 4199: Enables query optimizer (QO) changes that are released in SQL Server Cumulative Updates (CUs) and Service Packs (SPs). • 8048: Converts NUMA-partitioned memory objects to CPU-partitioned memory objects. • 9024: Converts a global log pool memory object to a NUMA-partitioned memory object. For a complete list of trace flags, see the Microsoft SQL Server documentation

<https://docs.microsoft.com/en-us/sql/t-sql/database-console-commands/dbcc-traceon-trace-flags-transact-sql?view=sql-server-ver15>

1. **Control VLFs**

<https://www.sqlskills.com/blogs/paul/important-change-vlf-creation-algorithm-sql-server-2014/>

1. **Autogrowth**

Check database autogrowth settings Any transaction that needs the data or the log file to grow includes the time taken by the file growth operation. The file grows by the increment size defined by the FILEGROWTH option. You can look for file growth events in SQL Server profiler traces. If file growth takes a long time, you might see wait types like ASYNC\_IO\_COMPLETION, which occurs when data processing is very slow. Such wait types not only affect performance but might also result in transaction timeouts. If that transaction holds locks on resources sought by other transactions, the timeout would lead to severe server blocking issues. For this reason, we recommend that you configure autogrowth settings very carefully. Also keep in mind that: • File growth is one of the costliest operations in SQL Server. • Frequent autogrowth in small chunks can lead to disk fragmentation. • Frequent autogrowth in log files results in a large number of virtual log files (VLFs) and affects performance, as discussed in the previous section (p. 18). All these reasons could lead to slow database startup and increased backup and recovery time. Ideally, you should pre-grow files proactively, based on regular monitoring. Choose carefully between setting autogrowth as a percentage or as a static value (in MB). Typically, setting autogrowth to one eighth of the file size is a good starting point, but this might not be the right choice. (For example, this percentage would be too high if your data file is several TBs in size.) In most cases, an autogrowth value of 1024 MB works well for data files in most large databases. For log files, 512 MB is a good starting point. For contingency measures, we strongly recommend that you set the autogrowth value, but grow the files manually for a few months based on past trends. Note Setting autogrowth should be a contingency measure, so you should set it after you pre-allocate storage to a file. You can change autogrowth settings by using SQL Server Management Studio (SSMS) or Transact-SQL. The following screenshot shows autogrowth settings in SSMS.

1. **Performance counters**

#### General Performance by Symptom (perfmon and trace)

##### **Memory**

***Memory: Pages/sec:*** The rate at which pages are read from or written to disk to resolve hard page faults. This counter is a primary indicator of the kinds of faults that cause system-wide delays. It is the sum of Memory\Pages Input/sec and Memory\Pages Output/sec. It is counted in numbers of pages, so it can be compared to other counts of pages, such as Memory\Page Faults/sec, without conversion. It includes pages retrieved to satisfy faults in the file system cache (usually requested by applications) non-cached mapped memory files. (Hard Page fault). A high reading may be symptom of system memory under stress.

***Memory: Available MBytes:*** A reading of less than 96MB is a symptom of memory pressure and a performance hit to the server.

***Page Life Expectancy:*** Capture and this needs to be baselined.

This is probably the most popular performance counter when it comes to memory in SQL Server. It represents the number of seconds a page will stay in the buffer pool without the references. If I’d have to choose one counter, this metric is the one that all SQL Server monitoring tools should be able to track. So, essentially, we want the page to stay in the buffer for as long as possible because it will lead to high performance. This is simply because if a request comes in for that page, it can be read from the cache rather than searching on the disk and ultimately reduce I/O. As for the expected value, this is a tricky one. You’ll probably read online that it should be more than 300. This is an old rule, when SQL Server memory could maximum be 4 GB. **These days, we can have much more RAM than that and this metric is basically memory dependent. Therefore, one of the most known memory formulas that can be found to calculate your PLE value online is the amount of allocated memory to SQL Server divide by 4 and multiply that number with** 300. For example, if 16 GB is allocated to SQL Server it should look like this 300 \* (16/4) = 1.200. So, this number is your base and if our PLE value is below this number that would be a good indication that SQL Server does not have enough memory to keep pages in the buffer long enough. By the way, this will directly affect buffer cache hit ratio. Therefore, those two are the most important SQL Server monitoring tools and commonly used counters to discover memory pressures.

For more information about monitoring two types of counters above, see the [SQL Server memory performance metrics – Part 4 – Buffer Cache Hit Ratio and Page Life Expectancy](https://www.sqlshack.com/sql-server-memory-performance-metrics-part-4-buffer-cache-hit-ratio-page-life-expectancy/) article.

***SQL Server: Buffer Manager: Buffer Cache Hit Ratio:*** Total amount of dynamic memory the server is using for the dynamic SQL cache Anything above 90% is acceptable, however this ideally should be as near to 99% as possible depending on the system, namely if OLTP. Anything less than 90% may be indication of memory pressure.

##### **Disk I/O**

To evaluate whether you have a disk array resource bottleneck, you will utilize both of these counters to efficiently diagnose. Alerts should also be in place to notify an engineer of low disk space available, which can be a contributor to this issue as well.

You should utilize both of these counters in unison. If you have a sustained % Disk Idle Time of < 45% and an average disk queue length of greater than 2 per physical disk (spindle), then you can be confident you are experiencing an I/O bottleneck.

***Physical Disk: Disk Reads/sec:*** The rate at which bytes are transferred from the disk during read operations. Growth of your baseline over time is symptomatic of memory pressure; this should be no higher than 85%.

***Physical Disk: Disk Writes/sec:*** The rate at which bytes are transferred to the disk during write operations. Growth of your baseline over time is symptomatic of memory pressure; this should be no higher than 85%.

***Physical Disk: % Disk Time:*** The percentage of elapsed time that the selected disk drive was busy servicing read or writes requests. Spikes are common. Results of >55% over a period of time warrants investigation.

***Physical Disk: Avg. Disk Queue Length:*** The average number of both read and writes requests that were queued for the selected disk during the sample interval. Results of > than 2 x number of physical disks in volume is not optimal.

***Physical Disk: % Free Space:*** The amount of free space available on physical disk. This should be no less than 5%

***Logical Disk: % Free Space:*** The amount of free space available on logical disk. This should be no less than 5%.

##### **CPU Utilization**

***Processor: Processor Time % (\_Total):*** The percentage of elapsed time that the processor spends to execute a non-Idle thread. This counter is the primary indicator of processor activity, and displays the average percentage of busy time observed during the sample interval. >80% % over a period of time may be an indicator of a CPU bottleneck. Spikes are common.

***System: Processor Queue Length*:** The number of threads in the processor queue. >2 per CPU is indication of a possible CPU bottleneck.

##### **Network Structure and Bandwidth**

***Network Interface: Bytes Received/sec:*** The rate at which bytes are transferred from the disk during read operations. a lot higher compared to a baseline. A substantial sudden baseline increase may be indication of an external attack; investigation is required.

***Network Interface: Bytes Sent/sec:*** The rate at which bytes are sent over the network adapter. A sudden increase over baseline may be indication of a large volume of data being accessed. If you cannot explain the sudden increase, investigation is required.

***Network Interface: Bytes/sec:*** This is the level rate as to traffic that is passed over the network. A substantial sudden baseline increase may be indication of an external attack; investigation is required. (ex. DOS – denial of service – attack)

***Network Interface: Output Queue Length:*** Number of packets sent over the network adapter that had to wait for transmission. A non-zero value would indicate a faulty NIC or excessive use for current NIC capabilities.

##### **User Connections**

***SQL Server: General: User Connections:*** This will show the number of user connections, not the number of users concurrently connected to SQL Server. A number of 255 is extraordinary and should be looked at for a corrective action.

 select \* from sys.dm\_os\_performance\_counters

Select avg(current\_tasks\_count),   
  avg(runnable\_tasks\_count),  
  avg(pending\_disk\_io\_count),  
  getdate() as reportdate  
  from sys.dm\_os\_schedulers where scheduler\_id < 8 and is\_online = 1  
  
Select \* from sys.dm\_os\_memory\_brokers  
Select \* from sys.dm\_os\_memory\_allocations