Basic Guideline on performance troubleshooting call for McDonalds

* Expectation of customer: -
* Whenever we receive a call or reminder for call, attend the call on time. Mostly we received the call for the US PROD servers and the issue reported by GSM , application is timeout or not working.
* Sometime we receive the email or to join the call for high CPU or Memory utilization
* On call we have to provide the details form SQL side

I am taking here two scenarios

1. Ongoing issue and observe load on SQL server
2. The issue is resolve and now we have check is there was issue from SQL side.

**Scenarios -- I**

* Displaying all active processes

----------------------------------------------

USE master;

GO

EXEC sp\_who 'active';

GO

---------------------------------------------

* To Check the blocking on SQL server and the lead blocker

SELECT \* FROM master.dbo.sysprocesses where blocked <>0

select blocking.spid

from master.dbo.sysprocesses blocking

inner join master.dbo.sysprocesses blocked

on blocking.spid = blocked.blocked

where blocking.blocked = 0

---------------------------------------------

* The below is very helpful to check which current process is utilizing high resource and check the execution plan for that SPID. We get the key information from the query, highlighted columns

SELECT r.session\_id,

s.program\_name,

s.login\_name,

r.start\_time,

r.status,

r.command,

Object\_name(sqltxt.objectid, sqltxt.dbid) AS ObjectName,

Substring(sqltxt.text, ( r.statement\_start\_offset / 2 ) + 1, ( (

CASE r.statement\_end\_offset

WHEN -1 THEN

datalength(sqltxt.text)

ELSE r.statement\_end\_offset

END

- r.statement\_start\_offset ) / 2 ) + 1) AS active\_statement,

r.percent\_complete,

Db\_name(r.database\_id) AS DatabaseName,

r.blocking\_session\_id,

r.wait\_time,

r.wait\_type,

r.wait\_resource,

r.open\_transaction\_count,

r.cpu\_time,-- in milli sec

r.reads,

r.writes,

r.logical\_reads,

r.row\_count,

r.prev\_error,

r.granted\_query\_memory,

Cast(sqlplan.query\_plan AS XML) AS QueryPlan,

CASE r.transaction\_isolation\_level

WHEN 0 THEN 'Unspecified'

WHEN 1 THEN 'ReadUncomitted'

WHEN 2 THEN 'ReadCommitted'

WHEN 3 THEN 'Repeatable'

WHEN 4 THEN 'Serializable'

WHEN 5 THEN 'Snapshot'

END AS Issolation\_Level,

r.sql\_handle,

r.plan\_handle

FROM sys.dm\_exec\_requests r WITH (nolock)

INNER JOIN sys.dm\_exec\_sessions s WITH (nolock)

ON r.session\_id = s.session\_id

CROSS apply sys.Dm\_exec\_sql\_text(r.sql\_handle) sqltxt

CROSS apply

sys.Dm\_exec\_text\_query\_plan(r.plan\_handle, r.statement\_start\_offset, r.statement\_end\_offset) sqlplan

WHERE r.status <> 'background'

ORDER BY r.session\_id

go

* Signal waits > 25% of total waits. High signal waits indicate a CPU bottleneck.

SELECT '%signal (cpu) waits' = CAST(100.0 \* SUM(signal\_wait\_time\_ms) / SUM (wait\_time\_ms) AS NUMERIC(20,2)),

'%resource waits'= CAST(100.0 \* SUM(wait\_time\_ms - signal\_wait\_time\_ms) / SUM (wait\_time\_ms) AS NUMERIC(20,2)) FROM sys.dm\_os\_wait\_stats;

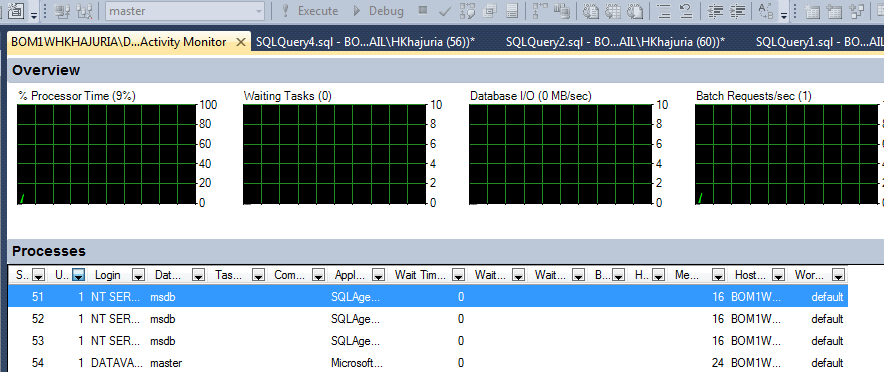
* A nonzero value indicates that tasks have to wait for their time slice to run high values for this counter are a symptom of a CPU bottleneck.

SELECT scheduler\_id,current\_tasks\_count,runnable\_tasks\_count

FROM sys.dm\_os\_schedulers

WHERE scheduler\_id < 255

* We can also use the activity monitor to view the status



* SQL Server is to use the built-in performance reports.  The reports provide useful information that can assist you in determining where your performance bottlenecks may be. The data from these reports is pulled from DMVs as well as the default trace that is always running.
* To access the reports, open SSMS, right click on the SQL Server instance name and select **Reports** > **Standard Reports**
* We can also use the perfmon counters to analyse.

|  |  |  |
| --- | --- | --- |
| **Counters** | **Resource Area** | **Indicators** |
| SQLServer:Access Methods\Full Scans/sec | IO related | A healthy ratio is 800-1000 times more index searches than scans Missing indexes are a common cause. |
| SQLServer:Access Methods\Index Searches/sec |
|  |  |  |
| Physical Disk\Avg. Disk Reads/sec Physical Disk\Avg. Disk Writes/sec | IO related | Latency associated with SQL Server read and write operations will depend on the type and speed of your disks, but commonly-cited guidance values are: Less than 10 ms = good performance Between 10ms and 20 ms = slow performance Between 20ms and 50ms = poor performance Greater than 50ms = signicant performance problem |
| SQLServer:Buffer Manager\Lazy Writes/sec | Memory related | A set of counters that together may indicate memory pressure. E.g. fluctuating Page Life Expectancy, nonzero Free List Stalls/sec and high Lazy Writes/sec indicates the buffer pool cannot handle the amount of amount of data the workload needs. Solution could be more memory or query tuning or missing indexes. |
| SQLServer:Buffer Manager\Page life expectancy |
| SQLServer:Buffer Manager\Free list stalls/sec |
| SQL Server:Buffer Manager\ Free Pages |
| SQLServer:Memory Manager\Memory | Amount of memory that SQL Server wants to commit (target) versus total memory allocated. If target>total it could indicate memory pressure – but the SQL Server counters above are better indicators |
| SQL Server:Memory Manager\ Target Server Memory(KB) |
| SQL Server:Memory Manager\Total Server Memory (KB) |
| Processor/ %Privileged Time | CPU related | What is eating the CPU? Is it SQLServer (%user time)or some otheroperating system process (%privileged time)ideally it should be < 80% |
| Processor/ %User Time |
| Process (sqlservr.exe)/ %Processor Time |
|  |  |  |
| SQLServer:SQL Statistics/Batch Requests/sec | CPU related | For example, many ad-hoc queries and a general lack of parameterization can cause excessive plan compilation and many auto parameterization attempts. This is a common cause of many CPU issues. |
| SQLServer:SQL Statistics/SQL ReCompilations/sec |
| SQLServer:SQL Statistics/AutoParam Attempts/sec |
| SQLServer:SQL Statistics/Failed Auto-params/sec |
| SQLServer:Plan Cache/Cache hit Ratio |
| Processor Queue Length | < 4 per CPU | Number of threads waiting for CPU cycles, where < 12 per CPU is good/fair, < 8 is better, < 4 is best. |
|
|
|
|
| Context Switches/sec | < 4 per CPU | Number of execution contexts switched in the last second, where >6000 is poor, |
|
|
|
|

**Scenarios – II**

Here if the customer asks for the past details form SQL side for the issue occur.

We will start with the basic investigation like check following things

* Any SQL job was running on that specific time
* Check the SQL error log, windows event log
* Check the default trace with below query
* To identify active traces running on server.

SELECT \* FROM ::fn\_trace\_getinfo (default)

* To query trace file.

SELECT \* FROM ::fn\_trace\_gettable('Trace file path', default)

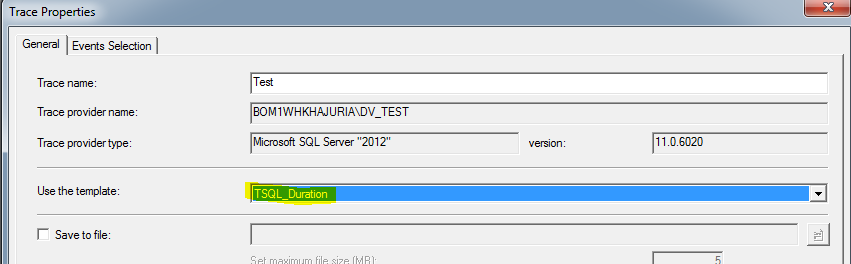
SELECT StartTime,\* FROM fn\_trace\_gettable ('Trace file path',default)

WHERE StartTime between '2014-03-06 23:00:00.000' and '2014-03-07 02:00:00.000'

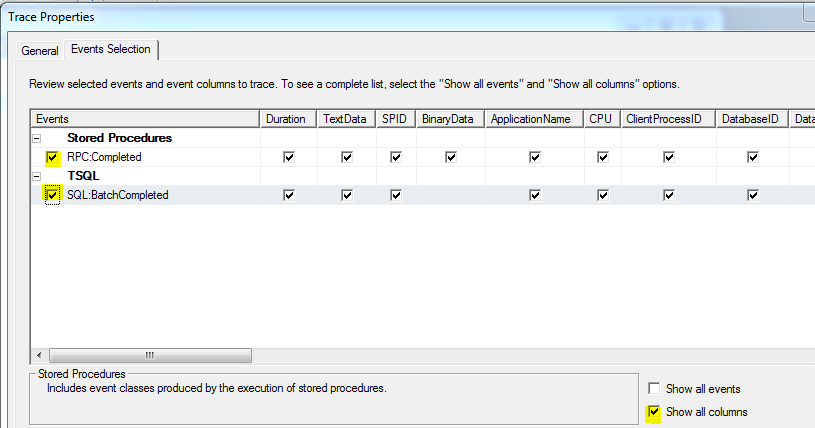
and TextData is not null

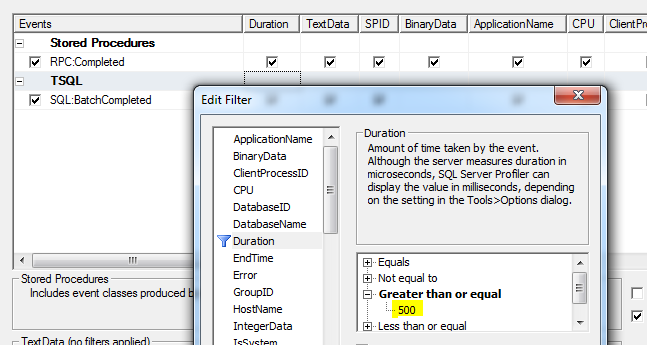
Suppose if we want to run the SQL trace to capture the query which are running above 500ms then we can set the trace as following.

IN SSMS click on tools 🡪 SQL profiler and select Use the template “TSQL\_Duration”



Click on Event Selection, check the require box and filter with duration (500) and row count (5 or 10)





We can also use the sp\_BlitzFirst to check the SQL server check.



**In SQL server, new feature is introduced “Query Store”**

* Automatically captures a history of queries, plans, and runtime

statistics

* Performance data is retained during a restart
* Information is aggregated over different time intervals
* Works for natively compiled procedures and in-memory OLTP queries
* Supported on all SQL Server editions
* Enabled at database level
* Query Store
* Statement level
* Query text given
* Retain over time and restarts
* Aggregate by time windows
* Works for in-memory OLTP and disk-based workloads
* Explore the queries execution
* Resource consuming queries
* Figure out query plan
* Identify possible performance degradation queries
* Figure out why regressions happen
* Force the query processor to use a particular plan
* Query Store is accessible through Transact-SQL.
* Identify and improve ad-hoc workloads
* Pinpoint and fix queries
* Custom reporting and/or alerting through Dynamic Management Views (DMVs)

Since this feature is available at database level, we have follow the below steps

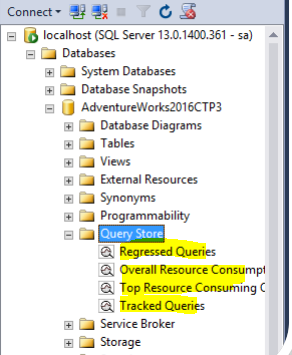
Object Explorer >Select Database >Query Store >Regressed Queries

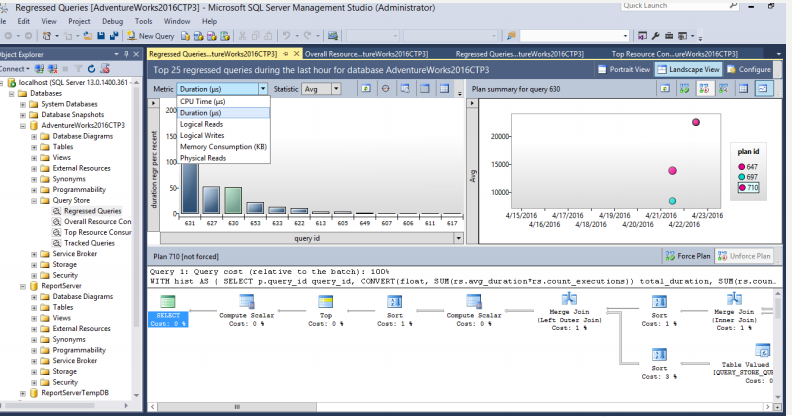
**Regressed Queries**: Pinpoint queries for which execution metrics have recently regressed (i.e. changed to worse).

**Top Resource Consuming Queries**: Queries which have the biggest impact to database resource consumption.

**Tracked Queries**: when you have queries with forced plans and you want to make sure that query performance is stable.

**Overall Resource Consumption**: Analyse the total resource consumption for the database. Use this view to identify resource patterns (daily vs. nightly workloads) and optimize overall consumption for your database.





We can also view data as per the timeline click on configure tab and change the parameter and see that graphs.

