# SQL - Long Running Query

|  |  |
| --- | --- |
| **Issue name** | **Long Running Query** |
| **Issue Description** | Queries are running longer than expected time, which is abnormal execution time. |
| **Issue Frequency** | Moderate |

|  |  |
| --- | --- |
| **Escalation** | **When and What** |
| Level 1 | When we receive Long Running Query issue email or teams’ text from users. |
| Level 3 | When active running query times are increasing and query's are taking long time to complete and SQL performance is decreasing: in extreme cases failover might be needed or restart, or clearing sessions before that. If any optimization is required at query level or server level, contact the DBA team. |

## Actions

|  |
| --- |
| **Instructions to resolve the issue** |
| Connect to the SQL server using SQL Server Management Studio. Open new query and execute below queries to verify top 10 High CPU and memory utilization query’s.  Execute the below command to capture the longest running queries on the SQL server.  execute Admin..SP\_Who3  The result of the query is shown below. The running\_time of the column will show the duration of the query. Also verify the column blk\_by if there are any blockings.  Image sp\_who3  If there is any blocking from SQL agent job ,Open SQL server agent and view job activity monitor and verify any jobs running. Verify the job duration and identify if it is a normal execution time or taking longer to complete.  If a call is opened with an app team, and If blocking are observed from the application connections and the number of sessions from application are very high, then request Application team to recycle the application or Web server.  Verify CPU and Memory usage at the server level on the primary replica. If the CPU usage above 80% Check the CPU utilization is by SQL service or other processes.  Windows<<Task manger (sort by CPU)  Image taskmgr  Note: Escalate to the concerned team if the CPU usage is above 80-85%.  Use below command to check if there are any long running queries with High CPU usage.  SP\_Who3 2 or ‘CPU’  The results looks as below.  Image with sp\_who3 2 |
| Capture Long running sessions, since how long it is running, etc. |
| From the result of the query “sp\_who3 2” the column “sql\_text” gives the queries. If those queries are running for a longer time than expected and it’s not completing, check the underlying objects by analyzing the query. |
| After finding the objects check when the statistics were updated last time using below query.  Option1:  SELECT OBJECT\_NAME(object\_id) AS [ObjectName] ,[name] AS [StatisticName] ,STATS\_DATE([object\_id], [stats\_id]) AS [StatisticUpdateDate] FROM sys.stats where OBJECT\_NAME(object\_id) in (**' Mention the object names here'**)  for sample refer below screenshots  A screenshot of a computer  Description automatically generated  Option 2:  DBCC SHOW\_STATISTICS ('table\View', index\statistics)  Refer screenshot below as a sample  A screenshot of a computer  Description automatically generated  Based on the query of the result if the statistics are not updated recently go ahead and update statics of that table using below query.  Below query to check the rows, modification counter and last statistics updated date for the required objects. -- Checks for statistics last update date in the current database. --  DECLARE @sqlcmd NVARCHAR(4000), @sqlmajorver int, @sqlminorver int, @sqlbuild int, @TableName varchar(250)  set @TableName = 'ALL' -- Add table name filter here. ALL for all tables.  SELECT @sqlmajorver = CONVERT(int, (@@microsoftversion / 0x1000000) & 0xff); SELECT @sqlminorver = CONVERT(int, (@@microsoftversion / 0x10000) & 0xff); SELECT @sqlbuild = CONVERT(int, @@microsoftversion & 0xffff);  IF (@sqlmajorver = 10 AND @sqlminorver = 50 AND @sqlbuild >= 4000) OR (@sqlmajorver = 11 AND @sqlbuild >= 3000) OR @sqlmajorver > 11 BEGIN SET @sqlcmd = 'USE ' + QUOTENAME(DB\_NAME()) + '; SELECT DISTINCT ''' + CONVERT(VARCHAR(12),DB\_ID()) + ''' AS [databaseID], mst.[object\_id] AS objectID, ss.[stats\_id], ''' + DB\_NAME() + ''' AS [DatabaseName], [t.name](http://t.name) AS schemaName, OBJECT\_NAME(mst.[object\_id]) AS tableName, [ss.name](http://ss.name) AS [stat\_name], ISNULL(sp.[rows],SUM(p.[rows])) AS [rows], sp.modification\_counter, STATS\_DATE(o.[object\_id], ss.[stats\_id]) AS [stats\_date] FROM sys.stats AS ss INNER JOIN sys.objects AS o ON o.[object\_id] = ss.[object\_id] INNER JOIN sys.tables AS mst ON mst.[object\_id] = o.[object\_id] INNER JOIN sys.schemas AS t ON t.[schema\_id] = mst.[schema\_id] INNER JOIN sys.partitions AS p ON p.[object\_id] = ss.[object\_id] CROSS APPLY sys.dm\_db\_stats\_properties(ss.[object\_id], ss.[stats\_id]) AS sp where ((OBJECT\_NAME(mst.[object\_id]) = ''' + @TableName + ''') OR ('''+@TableName +''' = ''ALL'')) GROUP BY o.[object\_id], mst.[object\_id], [t.name](http://t.name), ss.stats\_id, [ss.name](http://ss.name), sp.[rows], sp.modification\_counter ORDER BY [t.name](http://t.name), OBJECT\_NAME(mst.[object\_id]), [ss.name](http://ss.name)' END ELSE BEGIN SET @sqlcmd = 'USE ' + QUOTENAME(DB\_NAME()) + '; SELECT DISTINCT ''' + CONVERT(VARCHAR(12),DB\_ID()) + ''' AS [databaseID], mst.[object\_id] AS objectID, ss.[stats\_id], ''' + DB\_NAME() + ''' AS [DatabaseName], [t.name](http://t.name) AS schemaName, OBJECT\_NAME(mst.[object\_id]) AS tableName, [ss.name](http://ss.name) AS [stat\_name], SUM(p.[rows]) AS [rows], rowmodctr AS modification\_counter, STATS\_DATE(o.[object\_id], ss.[stats\_id]) AS [stats\_date] FROM sys.stats AS ss INNER JOIN sys.sysindexes AS si ON [si.id](http://si.id) = ss.[object\_id] INNER JOIN sys.objects AS o ON o.[object\_id] = [si.id](http://si.id) INNER JOIN sys.tables AS mst ON mst.[object\_id] = o.[object\_id] INNER JOIN sys.schemas AS t ON t.[schema\_id] = mst.[schema\_id] INNER JOIN sys.partitions AS p ON p.[object\_id] = ss.[object\_id] LEFT JOIN sys.indexes i ON [si.id](http://si.id) = i.[object\_id] AND si.indid = i.index\_id WHERE o.type <> ''S'' AND [i.name](http://i.name) IS NOT NULL AND ((OBJECT\_NAME(mst.[object\_id]) = ''' + @TableName + ''') OR ('''+@TableName +''' = ''ALL'')) GROUP BY o.[object\_id], mst.[object\_id], [t.name](http://t.name), rowmodctr, ss.stats\_id, [ss.name](http://ss.name) ORDER BY [t.name](http://t.name), OBJECT\_NAME(mst.[object\_id]), [ss.name](http://ss.name)' END EXECUTE sp\_executesql @sqlcmd  **Note: If the stats needs to be verified for any particular table input the table name as a value for the variable ‘@tablename’ in the above script. ‘ALL’ will provide the list of all tables details in that database.**  Image output table  **Note: If the table size is less than 10 GB use 5 %, if the table size is above 10 GB use 2 %.**  Command to get table size:  sp\_spaceused 'TableName'  A screenshot of a computer  Description automatically generated  Update Statistics Table\Index view with sample 5 Percent  **Sample :**  A screenshot of a computer  Description automatically generated |
| If update statistics also doesn't work please escalate the issues to the concerned team/body. |

## (blue star)Resolve Issue (Option to fix the issue)

|  |  |  |
| --- | --- | --- |
|  | **Option to fix instructions** | **Documentation** |
| 1 | Run update statistics on specific objects |  |
| 2 | Please check the high utilization query's and give it to app or dba team for the tuning of the queries. |  |
| 3 |  |  |

# SQL - High CPU

|  |  |
| --- | --- |
| **Issue name** | High CPU |
| **On this page** | High CPU consumption Alerts |
| **Issue frequency** | Moderate |

|  |
| --- |
| **Instructions to resolve the issue** |
| Below are sample High CPU alerts . In the below sample the CPU is high on two different servers are high.  Image sample alert on high cpu (email + spotlight) |
| RDP to the server  Verification:   1. When an alert is received on high CPU connect to the server and verify the server resources. 2. Login to the server open windows and select task manager   A screenshot of a computer  Description automatically generated  3.  Open the task manager and verify the CPU and Memory  A screenshot of a computer  Description automatically generated  Verify the Details of the CPU from Task manager.  A computer screen shot of a computer  Description automatically generated  Open resource monitor and verify the detailed description of the resources causing high CPU.  A screenshot of a computer  Description automatically generated  A screenshot of a computer  Description automatically generated |
| Verify the details of high CPU from SQL server.   1. Connect to the SQL server from SQL server Management studio. 2. Open new query and execute sp\_who3 2 or sp\_who3 'CPU'   exec dbautil.dbo.sp\_who3 2  A screenshot of a computer  Description automatically generated  Verify the columns Duration, Total CPU time to identify the queries with High CPU.  Identify the objects from the column objectName that been used in the query and verify if the fragmentation and statistics of the object.  Click on the query\_plan and it will display the plan of the query and from the plan we can identify the cost of the query and check for any index recommendations.  Execute the query SP\_who3.  A screenshot of a computer  Description automatically generated  Identify the long running queries. The column Running\_time. This will help to find the long running queries.  Verify the blocking on the server. Use below queries to identify the blockings.  Use DBUtil  Exec SP\_WHO3 5  This pulls the blocking tree which displays the session ids of blocking and blocked session.  Identify the blocks and root cause for the blocks. Follow the blocking document to resolve the blockings.  [SQL - Blocking or High number of database connections](/wiki/spaces/SSE/pages/1585284084/SQL+-+Blocking+or+High+number+of+database+connections)  **Identify the CPU utilization from the query Store.** (Note that this is not available for all the databases. This been available to only on few databases. )  Connect to SQL server management studio.  Browse to SQL server Querystore.  A screenshot of a computer  Description automatically generated  Expand the Query store and find the 'Top Resource Consuming Queries'.  A screenshot of a computer  Description automatically generated  Right click on the Top Resource Consuming Queries and select 'ViewTop Resource Consuming Queries'.  A screenshot of a computer  Description automatically generated  The report is shown as below.  A screenshot of a computer  Description automatically generated  The query plan will provide all details on the high utilization.  A screenshot of a computer  Description automatically generated  Additionally, check on CPU history from Spotlight  go to Spotlight put instance name and In the details click on instance icon (which is in color green) and Click on CPU tab. Adjust Start time-end time as required  image cpu load check |

|  |
| --- |
| **Option to fix** |
| If High CPU is identified due to any SQL  jobs for Rebuild Index, update stats, don't stop the job. Inform DBA or concerned team. |
| If High CPU is caused due to any application connections Open call Involve all DBAs and application team for further troubleshooting.  If necessary, send email to the concerned team |

# SQL - SQLWS\_LockPro

|  |  |
| --- | --- |
| **Issue name** | **Using LockPro for troubleshooting or support during MI** |
| **Issue Description** | Looking for Blocking from Lock Wait Types |
| **Issue Frequency** | Often |
| **On this page** | [LockPro Explained](#SQL-SQLWS_LockPro-LockProExplained) | [🎢 LockPro Uses](#SQL-SQLWS_LockPro-LockProUses) | [🎽 Background Information](#SQL-SQLWS_LockPro-BackgroundInformation) |

## LockPro Explained

LockPro is part of the SQLWS Pro Toolset. LockPro was created to identify Blocking in processes with a ‘LCK\_%’ Wait Type. Identifying and killing processes with ‘LCK\_%’ wait types during periods of high database contention will help bring relief to the database server. LockPro has multiple uses depending on the situation, and they are detailed below.

## (blue star)LockPro Uses

LockPro is very flexible and has many different parameters that apply to different situations

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Statement** | **Usage** | **Documentation** |
| 1 | dbautil..sqlws\_LockPro - This will show lead blockers, blocking tree, transaction log usage by spid and will help determine processes to kill. | Analysis | Analyze Current blocking from Lock Wait Types  A screenshot of a computer  Description automatically generated |
| 2 | exec sqlws\_LockPro NULL, '2021-01-19 14:23:40.447' - In this example you can see a lead blocker that is sleeping. This is candidate to be killed to free up the blocked process | Analysis | Analyze time stamp to see what was blocking during an issue.  A screenshot of a computer  Description automatically generated |
| 3 | exec sqlws\_LockPro NULL, '2021-01-19 12:00:00.000', '2021-01-19 12:05:00.000' | Analysis | Analyze timeframe for incident. Giving DateRange will give you starting point for timeframe analysis. Look for a high session count to point to potential times to investigate.  A screenshot of a computer  Description automatically generated |
| 4 | exec sqlws\_LockPro 'KillHeadBlockers' | NOC/SI/MI Call | This is used to kill all the Spids at the top of blocking trees. This should be done on NOC Call or with approval from leads |
| 5 | exec sqlws\_LockPro 'KillAllBlockers' | NOC/SI/MI Call | This is used to kill both Spids at top of blocking tree AND every Spid currently blocking another process. High Impact and should be done on NOC MI/SI with approval from leads. This can be step to do before failover. |

## (blue star)Background Information

|  |
| --- |
| **LockPro Columns Explained** |
| session\_id |
| blocking\_session\_id |
| is\_blocked |
| is\_blocker |
| is\_head\_blocker |
| is\_dead\_locked |
| blocker\_session\_ids |
| blocked\_session\_ids |
| head\_blocker\_session\_ids |
| command |
| status |
| wait\_type |
| wait\_time |
| wait\_resource |
| **transaction\_log\_usage** - trans start time, number of records and log bytes size, in tempdb first and all other dbs next |
| open\_transaction\_count |
| is\_stuck\_in\_application |
| idle\_time |
| cpu\_time |
| total\_elapsed\_time |
| reads |
| logical\_reads |
| writes |
| nest\_level |
| database\_name |
| object\_name |
| sql\_statement |
| program\_name |
| host\_name |
| login\_name |
| last\_write |
| last\_request\_end\_time |
| is\_user\_process |
| statement\_sql\_handle |
| ctime |