## Checklist for Server Slowness

This is a checklist of potential issues or fixes when we experience issues/slowness with servers, please feel free to add as you see fit. NOTE: If you have not been trained on one or all of the processes below, please check with your supervisor before performing any of the below tasks.

* Network
  + Use [network map](https://app.diagrams.net/#G1xvwuvBkNLpbtyLOWzj0wnmBjHre_Ld6V)
    - Gather information including what servers and/or processes are slow, who is experiencing slowness (what networks) and how widespread this issue is
  + Check Grafana
* Antivirus
  + Check if updates are currently running - currently using Bitdefender on servers
* Servers - All
  + Check CPU usage: if it stays at 100% continuously, server may restart or at least will slow down
    - Look into what processes are causing this spike and note - close out this process and reopen if possible
  + Check memory: if remaining continuously high, server may need to be restarted or need to check what query/process is causing that
    - \*\*SQL Servers run at 95% normally
  + File System
    - Check if process is slow specific to saving or opening files from a certain folder
    - IF related to vault, check if vault is running slowly
* DB
  + **Initial Check**
    - Turn on SQL Performance Monitoring (add to NR)
      * Buffer Cache Hit Ratio
        + How often it finds what it needs
        + Should be 100%
      * Page life expectancy
        + How long a page exists in memory before needing to be refreshed
        + The higher the better
      * Free list stalls
        + Memory metric -> number of times it had to wait
        + Should be 0
      * Pending memory grants
        + Should be 0
      * Planned Cache Hit Ratio
        + Query Execution Plans
        + Should be above 96%
      * Lock waits per seconds
        + Comparable to db
        + \*\*Add after monitoring for our specific db
      * Lock timeouts per second
        + Should be 0/close to 0
    - Check Server health of SQLAOA-04, SQLAOA-03, RDS-02 (for Response) and APP-INT-01 (for other apps)
    - CPU-intensive queries currently running
      * Use script found here: <https://drive.google.com/file/d/1TdkHDPRtRuFQnqSdCefAx7OWaY2UzSdo/view?usp=sharing>)
    - Memory
    - Page file usage
      * Healthy size is below 12% (comparable to memory)
      * Figure out memory pressure
    - Wait times per session
      * I use this query. Leslie runs confirms/prints often, so I’ll check the DJS/Whs group chat if her session(s) is(are) causing long wait times.

|  |
| --- |
| **select** top 10 \*  **from** sys.dm\_exec\_session\_wait\_stats w  **join** sys.dm\_exec\_sessions s **on** s.session\_id = w.session\_id  **order** **by** w.signal\_wait\_time\_ms **desc** |

* + - * <https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-exec-sessions-transact-sql?view=sql-server-ver16>
    - Blocked processes
      * Login to sqlaoa2 and perform the following steps:
        + Open the database locks check SQL file (located right at the root of the server’s C: drive)
        + After logging in with your AD credentials upon launching the file, click the execute button on the toolbar and then scroll through the list that populates in a table below
        + If there are any numbers that show up under the BlkBy column (these numbers reference the SPID column, of which are pointers to the currently logged in users for Response/rds01), keep following them until you find the ‘source’
        + If you do see any locks, take note of the username(s) under the login column, and reach out to that person directly to sign out. If you don’t hear from them and you know they are not running a confirm/release, go to the user tab in the Task Manager on RDS-02 and sign out of their sessions
    - Verify primary/secondary is correctly set
  + **Specific Checks**
    - Lock error - often times people will get a specific error in the Response application that says this
      * You’ll need to clear the lock. Instructions are in the Blocked Processes section above
    - Person or people report a specific process is slow —->
      * Run a trace and send to colinear
        + NOTE: This should ONLY be done by one user at a time
        + 1. When you have identified an operation you want to "trace", exit the Response application, then navigate to d:\colinear\r4w\bin5 on the RDS01 remote desktop server and double-click the "ResponseWithTrace.bat" file.
        + 2. Login to Response and run whatever process you expected to run slowly, then exit the App
        + 3. Navigate to the newly created file called D:\colinear\r4w\bin5\trace.log
        + 4. Rename and send to [scott@colinear.com](mailto:scott@colinear.com) along with a description of what was running, how long it took, and how long it should take
  + **Deeper fixes (likely done after hours)**
    - Reset DB Statistics
      * EXEC sp\_updatestats (From Dustin - The SQL statistics for R4W\_Primary are in varying states of accuracy, so what can happen is that query compilation is building inefficient execution plans when certain tables are out of date. It’ll be a lot easier to identify causes of locks and slowness when these stats are updated so we know the plans are being built correctly. This one is easy to fix, and we probably want to integrate into the regular index maintenance cycle, but all it takes is running `EXEC sp\_updatestats`. This can be done without taking the database offline but it does impact performance while its running so it’s better to run after hours.)
    - Rebuild Indexes with High Fragmentation
      * Check for highly fragmented indexes on R4W (use script found here: <https://drive.google.com/file/d/1TdkHDPRtRuFQnqSdCefAx7OWaY2UzSdo/view?usp=sharing>)
        + \*\*For R4W Primary, don’t rebuild if their name = null
    - Check for lock escalations - disable LOCK\_ESCALATION
      * Management->Extended Events-> Sessions->lock\_esc
    - Remove old data from database
    - SELECT
    - *QUOTENAME*(*SCHEMA\_NAME*(sOBJ.schema\_id)) + '.' + *QUOTENAME*(sOBJ.name) AS [TableName]
    - , *SUM*(sPTN.Rows) AS [RowCount]
    - FROM
    - sys.objects AS sOBJ
    - INNER JOIN sys.partitions AS sPTN
    - ON sOBJ.object\_id = sPTN.object\_id
    - WHERE
    - sOBJ.type = 'U'
    - AND sOBJ.is\_ms\_shipped = 0x0
    - AND index\_id < 2 -- 0:Heap, 1:Clustered
    - GROUP BY
    - sOBJ.schema\_id
    - , sOBJ.name
    - ORDER BY [RowCount]
    - GO
* AWS RDS
  + 1. Enable Enhanced Monitoring to capture further OS-metrics for CPU utilization and process list visibility.
  + <https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_Monitoring.OS.html>
  + 2. Enable Performance Insights to view wait events, top SQL queries and session activity for further troubleshooting.
  + <https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_PerfInsights.html>
  + 3. Use the below diagnostic MySQL queries to isolate cause for high CPU utilization during the time of the workload increase:
    - => View current activity of each session currently running.
    - SHOW FULL PROCESSLIST;
    - (Note: must be run by RDS master user or have PROCESS server privilege assigned to user in order to see all processes on the system)
    - => Check for requested locks an InnoDB transaction has requested but hasn't received yet.
    - SELECT \* FROM performance\_schema.data\_locks; ()
    - => Check for blocked InnoDB transactions.
    - SELECT \* FROM performance\_schema.data\_lock\_waits;
    - => Checking for blocking and locking:
    - SELECT
    - r.trx\_id waiting\_trx\_id,
    - r.trx\_mysql\_thread\_id waiting\_thread,
    - r.trx\_query waiting\_query,
    - b.trx\_id blocking\_trx\_id,
    - b.trx\_mysql\_thread\_id blocking\_thread,
    - b.trx\_query blocking\_query
    - FROM performance\_schema.data\_lock\_waits w
    - INNER JOIN information\_schema.innodb\_trx b
    - ON b.trx\_id = w.blocking\_engine\_transaction\_id
    - INNER JOIN information\_schema.innodb\_trx r
    - ON r.trx\_id = w.requesting\_engine\_transaction\_id;
    - => Check the current state of the InnoDB storage engine.
    - SHOW ENGINE INNODB STATUS\G
  + 4. Using additional logging to such as General Log[1] and Slow Query Log[2] to find problem queries.
  + 5. Once you have identified high CPU causing queries:
  + a. Optimize the query or queries using EXPLAIN =><https://dev.mysql.com/doc/refman/8.0/en/using-explain.html>
  + b. Use the PROFILING command to see resource usage for statements that are running during the current session.<https://dev.mysql.com/doc/refman/8.0/en/show-profile.html>
  + c. Consider updating table statistics using ANALYZE TABLE command can help the optimizer choose an appropriate plan to run the query =><https://dev.mysql.com/doc/refman/8.0/en/analyze-table.html>
  + For further information you may view our Knowledge Center Article "How do I troubleshoot and resolve high CPU utilization on my Amazon RDS for MySQL or Amazon Aurora MySQL instance?" =><https://aws.amazon.com/premiumsupport/knowledge-center/rds-instance-high-cpu/>
  + [1]<https://dev.mysql.com/doc/refman/8.0/en/query-log.html>
  + [2]<https://dev.mysql.com/doc/refman/8.0/en/slow-query-log.html>