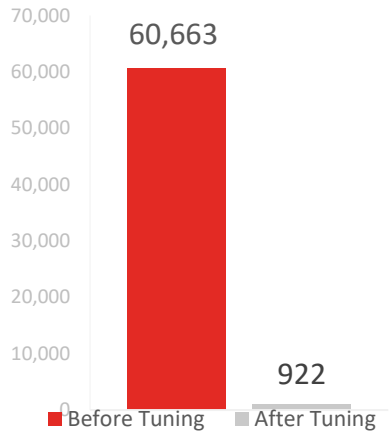
Performance Tuning Report



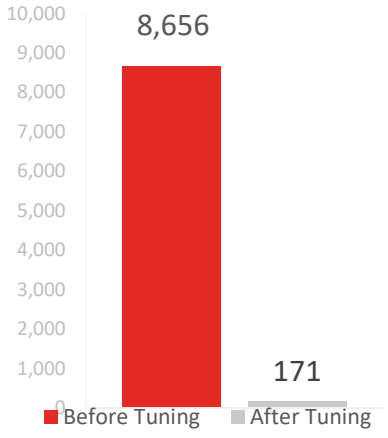
CPU

CPU is **66X** times faster

OR

6,580%

CPU improvement



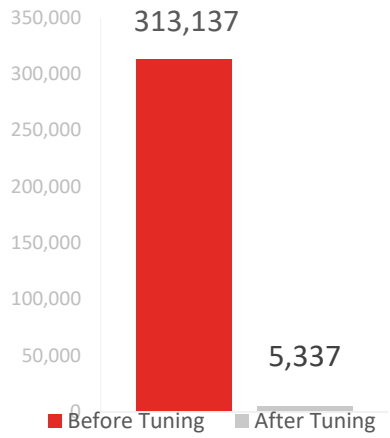
Speed

Speed is **51X** times faster

OR

5,062%

Speed improvement



Disk

Disk is **59X** times faster

OR

5,867%

Disk improvement

Description:

Problem: Stored procedure takes 38 seconds to execute. It times out on the app end.

Why: Stored procedure s performing multiple scans and reading too much data in.

Change: Stored procedure re-written. Also added a second select with a UNION ALL, this was necessary to divide the WHERE clause selectivity

Other notes: Result set for both version of the same stored procedure is same. However, row ordering is different. Therefore, before putting this change into production, please make sure that application does not care how data is received (we suspect it does not matter), as the app sorts the result set anyway.

Performance gains:

Execution time

Old execution = 38s

New = (under) 1s

Improvement: 37 times faster

Reads

old: 687,475

new: 314

improvement: 2,188 times less IO

More details:

Statistics IO Output	Object	Scans	Logical Reads	Physical Reads	Logical Reads as a Percentage of All Reads
Table [redacted] can count 9, logical rea [redacted]		9	28924	0	4.205%
Table [redacted] n count 9, logical reads [redacted]		9	363931	0	52.912%
Table [redacted] count 0, logical reads 0	Worktable	0	0	0	0.000%
Table [redacted] an count 9, logical reads [redacted]		9	294620	0	42.835%
Table [redacted] ypes'. Scan count 0, lo [redacted] y		0	8	0	0.001%
Table [redacted] count 0, logical reads 0	Worktable	0	0	0	0.000%
New version					
Table [redacted] lTypes'. Scan count 0, log [redacted]		0	8	0	0.001%
Table [redacted] n count 0, logical reads 0	Worktable	0	0	0	0.000%
Table [redacted] Scan count 28, logical re	tblRcptheader	28	123	0	0.018%
Table [redacted] an count 2, logical reads	tblRcptitem	2	133	0	0.019%
Table [redacted] can count 9, logical reads	tblLineItems	9	58	0	0.008%

Actual

Table [redacted]r'. Scan count 9, logical reads **28924**, physical reads 0,
Table [redacted] Scan count 9, logical reads **363931**, physical reads 0,
Table [redacted] Scan count 9, logical reads **294620**, physical reads 0,

New version

Table [redacted]r'. Scan count 28, logical reads **123**, physical reads 0,
Table [redacted] Scan count 2, logical reads **133**, physical reads 0,
Table [redacted] Scan count 9, logical reads **58**, physical reads 0,

Technical Background:

Most SQL Servers bottleneck on Disk access (or disk “reads”).

It’s not CPU or RAM – which most customers often suspect first.

And that makes a lot of sense. Here is why.

Inefficient queries scan (or read) lot of data. Data read in is stored in RAM. As more data is read in, “older” data is pushed out from RAM. If there isn’t enough RAM to keep ALL data in memory (which is often not possible), SQL Server has to read from disk – and that is the slowest operation SQL Server can do.

When query can be tuned to read 10 rows vs 10M – less CPU and RAM automatically are necessary. Therefore, tuning for less disk “reads” is often the primary goal.

To the end user nothing is more important than Speed (or Duration of the query) though.

Tuning to reduce CPU/RAM resources are helpful too.

When queries are tuned to need less CPU & RAM, it means that same server now has more capacity. Which means that same server can process double or triple the load. Which means it extends lifespan of the same server. Which means hardware upgrades can be pushed out further into the future.