Areas to Explore for Performance Gain

# The Issue

Primarily, the issue can be stated as simply as saying the current maintenance window is under great duress. The need to keep the database in top shape with regards to tables, indexes, and backups is paramount. This mandate is being short circuited by the time required to complete these tasks and the amount of time available or allotted to conclude the same.

# Rebuilding and Reorganizing Indexes

As with many of the Donnelly database, this one has very large tables as well. The extreme tables have multiple indexes and the indexes are hit hard during inserts and deletes; and potentially updates if an indexed field is modified. An index, when updated, will lock at the row level and all indexes on Donnelly databases, to date, have been instantiated to lock at the row and the page level. This means that locking one integer item for insert at the row level, can possibly lock hundreds of other rows contained on that same page. This is not the issue, the real issue is the size of each and the number of individual indexes. We are experiencing times of 7 hours or more to rebuild/reorg a single index. When multiplied by hundreds, even thousands of indexes, the maintenance window is totally overwhelmed. The answer is not to have fewer indexes, but to have many smaller indexes and run the reorgs in parallel. (*See partitioning large tables and Dynamic Re-indexing plans*)

# One Storage Area per Incremental Backup

Presently, the backup plan includes a nightly incremental backup to blob storage. Multiple incremental backups are being sent to one storage area. It may be that separating out each incremental backup to its own storage area may gain some performance and potentially decrease the time required to complete the tasks in total.

# Partition Large, Hard Hit Tables

Partitioning can provide tremendous benefit to a wide variety of applications by improving performance, manageability, and availability. It is not unusual for partitioning to greatly improve the performance of certain queries or maintenance operations. Moreover, partitioning can greatly simplify common administration tasks.

Partitioning also enables database designers and administrators to solve some difficult problems posed by cutting-edge applications and required maintenance facilities reducing time required by a factor of 10. Partitioning is a key tool for building multi-terabyte systems or systems with extremely high availability requirements. To ignore such a benefit is similar to virtual suicide.

Why partition tables and indexes:

* Partitioning for Performance
* Partitioning for Manageability
* Partitioning for Availability
* Partitioning for maintenance
* We have developed a concept and practice of dynamic repartitioning based on gathered statistics from DFIN Analytics, a new tool being drought in and implemented, that provides an automated re-indexing and reorganization of tables and indexes that optimizes the layout of partitions based on individual table load as determined throughout periods of the day, week, and month.

# Dual Transfer from SSD to Standard Blob Storage

This particular server appears to be using a complete array of standard DASD. Now we need to get into physical tuning parameters based on placement of the drive index, the traversal speed of the disk read arm, and the partitioning of the disk and of course, the disk’s fragmentation. Yes, I know, this is another hard hit to the maintenance window.

A viable approach might be to use an SSD to receive the incremental backup(s) and then, when the backup is complete and the load subsides on the parent server, move the SSD backup to a standard disk blob storage using something as simple as RichCopy. When copied, delete and start the next one.

Another consideration might be considered to use a raid of Striped Zero disk. We will get the minimum specifications for the raid disk if considered.

# Critical Backup Measurements

The backup can possibly be improved if the 3 critical factors of the backup are fully understood and analyzed.

1. Read speed, buffer rates, buffer overload & unload speeds in microseconds, optimum transfer block size as matched explicitly to the receiving disk partitioning size (e.g. 8192 matched to an 8192 and a block transfer of 1000 packets –perfect match, one arm depression on the receiving drive, and the next buffer is waiting to unload.
2. Speed across the connective network and a secure measurement of TCPIP packet retransmit vs. UDP and a precise measurement of packet loss and any detrimental effects on either loss or retransmission.
3. Buffer load speeds, buffer write speeds, disk revolution speed, number of consecutive blocks written with one arm touch, and last disk traversal speed of the arm, and location of the disk index physically on the drive. If it is in the middle, arm traversal is reduced significantly and when partitioned correctly, approaches immeasurable speeds.

# Dynamically Generate Re-indexing Plans

Our plan is to install of full set of server diagnostics that build and feed data into a Data Mart of cubes. The cubes allow us to more accurately pinpoint and understand areas of heavy load, items affected by high index usage, the operations hitting the index causing the problem(s), the worst queries that have run throughout the day and what plan was in use at the time, low use times of the server, the database, and each individual table, the least hit and most hit partitions and what data segmentation was in effect… This allows us to have the system determine which indexes to reorganize immediately and which ones to put off until an upcoming lag in server demand and load.

Understanding table and file group growth patterns allows us to prepare for and partition before the need becomes critical to process and client success. Knowing which queries are the worst performers and which plans are in effect will give us remarkable insight on what to tune, how critical it is, and when it will most likely fail again and under what circumstances.

# Replicant Cluster

To open up the maintenance window to where it becomes unencumbered by processes that require non-interrupted, minimum server load run capabilities, we should also consider running a parallel, mirrored server. This will allow one server to run full capacity while the other is detached from the replicant roles. When the workload server completes its job(s), replication or sync framework can be reengaged to synchronize the data. We are not limited to these two options as we also have other methods and options. While the workload server is processing, the detached server can perform its required maintenance activities with full access to the machine and not interfere with the work machine. In addition to replication, we can look at the Microsoft Sync Framework. It is a wide-ranging synchronization platform that enables collaboration and offline access for applications, services, and devices with support for any data type, any data store, any transfer protocol, and any network topology. Additionally, it works with Azure in looking ahead.