Page life expectancy

Last updated by | Vitor Tomaz | Feb 24, 2023 at 3:32 AM PST

Contents

Introduction

When querying data, the SQL engine will pull data pages (table and index) into the buffer pool (if they are not already there) before returning it to the client.

If a page is already on the buffer pool, data retrieval will be faster than if it has to be pulled from disk. Like so, the more pages that we have in memory, the faster can data pages be retrieved.

The **buffer pool used per database** can be checked with the query below:

```
SELECT
    databases.name AS database_name,
    COUNT(*) * 8 / 1024 AS mb_used
FROM sys.dm_os_buffer_descriptors
INNER JOIN sys.databases
ON databases.database_id = dm_os_buffer_descriptors.database_id
GROUP BY databases.name
ORDER BY COUNT(*) DESC;
```

From customer side you can see how much of a table is on the buffer pool. The query below will give the percentage of the table is on the buffer pool (**run on a database with large buffer pool usage**):

```
WITH CTE_BUFFER_CACHE AS (
        SELECT
                objects.name AS object_name,
                objects.type_desc AS object_type_description,
                objects.object_id,
                COUNT(*) AS buffer_cache_pages,
                COUNT(*) * 8 / 1024  AS buffer_cache_used_MB
        FROM sys.dm_os_buffer_descriptors
        INNER JOIN sys.allocation_units
        ON allocation_units.allocation_unit_id = dm_os_buffer_descriptors.allocation_unit_id
        INNER JOIN sys.partitions
        ON ((allocation_units.container_id = partitions.hobt_id AND type IN (1,3))
        OR (allocation_units.container_id = partitions.partition_id AND type IN (2)))
        INNER JOIN sys.objects
        ON partitions.object_id = objects.object_id
        WHERE allocation_units.type IN (1,2,3)
        AND objects.is_ms_shipped = 0
        AND dm_os_buffer_descriptors.database_id = DB_ID()
        GROUP BY objects.name,
                        objects.type_desc,
                        objects.object_id)
SELECT
        PARTITION_STATS.name,
        CTE_BUFFER_CACHE.object_type_description,
        CTE_BUFFER_CACHE.buffer_cache_pages,
        CTE_BUFFER_CACHE.buffer_cache_used_MB,
        PARTITION_STATS.total_number_of_used_pages,
        PARTITION_STATS.total_number_of_used_pages * 8 / 1024 AS total_mb_used_by_object,
        CAST((CAST(CTE_BUFFER_CACHE.buffer_cache_pages AS DECIMAL) / CAST(PARTITION_STATS.total_number_of_used
FROM CTE_BUFFER_CACHE
INNER JOIN (
        SELECT
                objects.name,
                objects.object_id,
                SUM(used_page_count) AS total_number_of_used_pages
        FROM sys.dm_db_partition_stats
        INNER JOIN sys.objects
        ON objects.object_id = dm_db_partition_stats.object_id
        WHERE objects.is_ms_shipped = 0
        GROUP BY objects.name, objects.object_id) PARTITION_STATS
ON PARTITION_STATS.object_id = CTE_BUFFER_CACHE.object_id
ORDER BY CAST(CTE_BUFFER_CACHE.buffer_cache_pages AS DECIMAL) / CAST(PARTITION_STATS.total_number_of_used_page
```

This will allow to have some knowledge on the usage of tables. Look especially for **large tables** where most of the data is on the buffer pool. Some preliminary questions can come from this, like:

- is there an inefficient query that is performing a large scan? For example, queries that need only just a few rows are doing a full scan (all pages are being pulled to the buffer pool, but only a few pages are being retrieved to the client).
- the total memory is sufficient for the amount of data that exists on the databases.

This is some of the questions that you can start asking yourself when checking the memory consumption.

Also note that, since memory is the fastest way to retrieve a data page, SQL will always try to use as much memory as it can. Like so, it is perfectly normal to see high memory usage on SQL Server. Unlike CPU and IO, high memory consumption is not an indication of a problem.

What is Page life expectancy

Page life expectancy is the number of seconds that a page lives in memory.

Remembering what was mentioned on the **Introduction** section, when data pages are pulled to the buffer pool, if there is no space on memory, older pages must be discarded from the buffer pool so that new pages can be pulled in.

If pages don't live for long on the buffer pool, it means that more IO requests must be done.

Page life expectancy value check

Given what have been exposed until now, the higher the value for Page Life Expectancy, the better.

Some old posts point 300 as the minimum value for Page Life Expectancy (PLE). Some more recent articles will point you to a more updated value, using an adaptive formula - DataCacheSizeInGB/4GB*300.

The query below will give the PLE value. You can compare this value with the minimum PLE got from DataCacheSizeInGB/4GB*300. If the recorded PLE value is lower, some action probably needs to be taken.

Note that Page life expectancy value isn't tracked historically. When checking the Page Life expectancy value, it will give you the instant value. So, it fluctuates.

```
SELECT @@SERVERNAME AS [Server Name], RTRIM([object_name]) AS [Object Name],
       instance_name, cntr_value AS [Page Life Expectancy]
FROM sys.dm_os_performance_counters WITH (NOLOCK)
WHERE [object_name] LIKE N'%Buffer Node%'
AND counter_name = N'Page life expectancy' OPTION (RECOMPILE);
```

When Page life expectancy is low

When PLE is low you might notice the following:

- more IO requests
- slower queries
- higher IO related waits, like PAGEIOLATCH

Mitigation points:

- increase managed instance memory by scaling up
- tune queries, especially queries that perform scans on larger tables

How good have you found this content?