Customer complains about used space is growing or too high

Last updated by | Ricardo Marques | Mar 6, 2023 at 8:40 AM PST

Contents

Issue

A customer might open a case referecing that he doesn`t know how a database is getting so much space used. The customer might reference some points like:

- "we deleted data but the used space is not coming down"
- "used space is growing exponencialy"
- "table X only contains Y number of rows, but is occupying a lot of space"

In some rare occasions the customer might have some kind of statistics with table growth details (that can be useful to pinpoint a specific object on where we want to concentrate)

Investigation/Analysis

Also check the documentation on Azure SQL side, especially the reference to Ghost records and PVS - https://supportability.visualstudio.com/AzureSQLDB/_wiki/wikis/AzureSQLDB.wiki/500577/Azure-SQL-DB-or-SQL-MI-used-data-space-is-larger-than-expected

First take a look at the used and allocated space of the database. Just make sure that the customer is not aware of concepts like Used and Allocated space. Also take a look at the Used Space growth trend. You might find a specific date where the Used space grew exponencialy. To check the database used and allocated space use the query below. Note that one database might have more than one datafile. Like so, make sure that you check the file id 3, 4, etc besides file 1 (file id 2 is the transaction log)

```
MonDmIoVirtualFileStats
| where LogicalServerName =~ "azuresqlmi2" //server name
| where db_name =~ "9a17224c-26aa-49da-80d8-30bee7ea686a" //on MI the database name is equal to the physical_d
| where type_desc == "ROWS" //looking only for data only
| where is_primary_replica == 1
| summarize sum(spaceused_mb), sum(size_on_disk_mb =(size_on_disk_bytes/1024/1024)) by TIMESTAMP
| render timechart
```

Checking by file id:

```
MonDmIoVirtualFileStats
| where LogicalServerName =~ "azuresqlmi2" //server name
| where db_name =~ "ac0bce2d-0ab1-4316-aa3e-ceb18a8f973f" //on MI the database name is equal to the physical d
| where type_desc == "ROWS" //looking only for data
| where is_primary_replica == 1
| project TIMESTAMP, file_id, type_desc,spaceused_mb, max_size_mb, size_on_disk_mb =(size_on_disk_bytes/1024/1
```

Take a look at table size. From the result of the query below, you can find table sizes. From here you can explore some intersting facts like:

- what is the biggest table?
- how is the trend of gowth for the biggest tables?
- what are the table types? Are they Heaps or Clustered tables?
- Do you see anything out of the ordinary? For example, a table with very few rows (in some cases 0 rows), but allocated space? If yes, is it a HEAP table?

```
let myAppName="d13a7e3bbd5a";
let dbName ="9a17224c-26aa-49da-80d8-30bee7ea686a"; //corresponds to physical_database_name
let PartitionStats=materialize(MonWiDmDbPartitionStats
| where AppName contains myAppName and logical_database_name !='master' and logical_database_name =~ dbName an
| summarize used_page_count=max(used_page_count), row_count=max(row_count) by database_id, logical_database_na
let FilteredResults=materialize(MonDatabaseMetadata
| where AppName contains myAppName and logical_db_name != 'master'
| where (table_name=='sysclsobjs' and class==50) or (table_name=='sysschobjs' and ['type']=='U ') or (table_na
| project TIMESTAMP, table_name, class, ['type'], id, name, nsid, indid);
let schemas=FilteredResults
| where (table_name=='sysclsobjs' and class==50)
| summarize by schema_id=id, schema_name=tolower(name);
let tables=FilteredResults
| where (table_name=='sysschobjs' and ['type']=='U ')
| summarize by schema_id=nsid, object_id=id, table_name=name;
let indexes=FilteredResults
| where (table_name=='sysidxstats' and indid in (0,1))
| extend index_type_desc=iff(['type']==0, 'HEAP', iff(['type']==1, 'CLUSTERED', iff(['type']==5, 'CCI', tostri
| summarize by object_id=id,index_id=indid,index_type=type,index_type_desc;
tables
| join kind=inner (schemas) on schema_id
| join kind=inner (indexes) on object_id
| join kind=inner (PartitionStats) on object_id
| project database_id, logical_database_name, schema_id, schema_name, object_id, table_name, table_type=index_
| sort by size_kb desc , schema_name asc, table_name asc, object_id asc, data_date asc
```

With this elements in mind, now jumping to the customer side:

- check index fragmentation and page density problems (take a look at the TSG - https://supportability.visualstudio.com/AzureSQLDB/_wiki/wikis/AzureSQLDB.wiki/724458/Clustered-Non-Clustered-indexes-and-Heaps-General-guidelines)
- take a look at possible problems with Heap table space used (check TSG - https://supportability.visualstudio.com/AzureSQLDB/_wiki/wikis/AzureSQLDB.wiki/725519/HEAP-table-size-doesn-t-change-after-row-deletion)

Mitigation

Depending on the analysis above, there will be multiple paths to follow:

- if a few tables are growing exponencialy, with no other reason than data increase, the issue must be checked on customer side (application).
- if you cannot pinpoint exacly a set of objects that are contributing the size grow, suggest for the customer to run and save the results of the query below on a table. With this elements the customer can check with this application the reason why more data is being ingested. This can be done using a SQL Agent Job.

```
SELECT
        (select getdate()) as Date
    t.NAME AS TableName,
    s.Name AS SchemaName,
    p.rows,
    SUM(a.total_pages) * 8 AS TotalSpaceKB,
    CAST(ROUND(((SUM(a.total_pages) * 8) / 1024.00), 2) AS NUMERIC(36, 2)) AS TotalSpaceMB,
    SUM(a.used_pages) * 8 AS UsedSpaceKB,
    CAST(ROUND(((SUM(a.used_pages) * 8) / 1024.00), 2) AS NUMERIC(36, 2)) AS UsedSpaceMB,
    (SUM(a.total_pages) - SUM(a.used_pages)) * 8 AS UnusedSpaceKB,
    CAST(ROUND(((SUM(a.total_pages) - SUM(a.used_pages)) * 8) / 1024.00, 2) AS NUMERIC(36, 2)) AS UnusedSpaceM
FROM
    sys.tables t
INNER JOIN
    sys.indexes i ON t.OBJECT_ID = i.object_id
INNER JOIN
    sys.partitions p ON i.object_id = p.OBJECT_ID AND i.index_id = p.index_id
INNER JOIN
    sys.allocation_units a ON p.partition_id = a.container_id
LEFT OUTER JOIN
    sys.schemas s ON t.schema_id = s.schema_id
WHERE
    t.NAME NOT LIKE 'dt%'
    AND t.is_ms_shipped = 0
    AND i.OBJECT_ID > 255
GROUP BY
    t.Name, s.Name, p.Rows
```

- if page density is low, rebuild indexes. Also rebuild Heaps when you see a problem with those.
- Check for other opportunities on the long run - (Check https://supportability.visualstudio.com/AzureSQLDB/_wiki/wikis/AzureSQLDB.wiki/629740/Managing-space-for-databases-and-pools?anchor=long-term-mitigations)

Root Cause Classification

How good have you found this content?