TUNING OF AGGREGATE STORAGE DATABASES

ESSBASE 7.1.2

Hyperion®

PERFORMANCE VISIBILITY FOR EVERYONE IN THE ENTERPRISE

The main purpose of this paper is to discuss the sizing and tuning methods available when designing, deploying and maintaining an Aggregate Storage database throughout its lifecycle. We will explain how Aggregate Storage uses resources such as disk space and memory at the various stages of the database construction and maintenance, and provide directions on how to optimize the performance and the resource usage. The reader should have general knowledge of Essbase.

PRELIMINARY ANALYSIS

The first step in your analysis is to identify measures and dimensions for the model. Even with increased dimensional scalability of the ASO, it is preferable to include a dimension into the model only if the analysis along that dimension is required.

In addition, dependent dimensions should be modeled as attribute dimensions or alternative hierarchies if possible.

A dimension is dependent if you can always determine the member within that dimension when the member within some other (primary) dimension is specified. For example, consider a "Product" dimension with individual SKUs as the lowest level. The packaging type for a product can be determined by the SKU of that product. Thus, it is recommended that you model the "Packaging Type" dimension as an attribute or alternative hierarchy on the product dimension, and not as a stand-alone dimension.

Whether you choose to use attributes or alternative hierarchies depends on the following factors:

- If members of different levels need to be included into groupings for the dependent dimension, choose alternative hierarchy modeling. For example, if you want to classify markets by size, and Large Market group includes several large individual cities and several states, the "Market Size" dependent dimension cannot be modeled as an attribute dimension. Attributes can be associated only with members of the same level.
- If a requirement exists for querying the members of the primary dimension across members of the dependent dimension, the dependent dimension should be implemented as an attribute dimension. For example, if cars can be classified by their color and model, and the requirement is to be able to ask queries, such as "Give me the sales of all red Civics in San Francisco," the "Color" should be an attribute dimension.

OUTLINE SIZING

After the dimensions are identified, the following metrics should be gathered, if possible:

- For every dimension, determine the number of stored levels within every hierarchy on this dimension.[1]

  Levels that consist only of label-only members are not stored, and they do not affect the sizing calculations. For example, consider the dimension Time with hierarchies Year-Quarter-Month-Day and YearByWeek-Week-Day. This dimension contains a total of six levels, four in the

first hierarchy and two additional levels in the second hierarchy (Day level is shared between the hierarchies).

Notice that all levels of attribute dimensions count as additional levels of their primary dimensions.

- Determine the total number of members across all dimensions.

NUMBER OF POSSIBLE VIEWS

With these metrics, you can determine if the model fits the limit on the number of possible views. The total count of views, or level combinations, cannot exceed 2^{52} in Essbase 7.1.2.

To find the total count of views in your model, multiply the number of levels for every dimension. For example, a model with five dimensions with 3, 5, 2, 1, and 3 levels, respectively, has 3 x 5 x 2 x 1 x 3 for a total of 90 possible views. If the result is less than 2^{52} or 4,503,599,627,370,496, the model fits. If the result is more, you must reduce the number of views. These are options to reduce the number:

- Try to find dimensions that are currently modeled as independent but could fit the dependent dimension criteria.
- Remove some dimensions or attempt to reduce the number of levels in some dimensions.
- Change one or more dimensions to be dynamically calculated. Dimensions that contain only dynamically calculated hierarchies are treated as if they had only one level in the calculation above. Notice that the dimension marked as "Accounts" is always dynamically calculated.

Although we tested artificial databases with more than 2^{50} potential views, the most complex real-world model we have seen so far contained about 1,200,000,000 potential views.

OUTLINE MEMORY CONSUMPTION

The second thing to consider is whether the number of members in the outline fits the available memory. Notice that on 32-bit platforms, the amount of memory available to the application is limited to 2 to 4 GB (depending on the platform), even if more RAM is installed on the machine. Also, you may need to set up the system to make as much memory as possible available to the Essbase server process. The memory addressability limits are as follows:

On 64-bit platforms, the limit is determined only by the physical RAM installed.

Operating System	Default Limit	Maximum Limit
Windows 2000 Advanced Server, 2000 Datacenter, 2003	2 GB	3 GB
Linux	2 GB	2 GB
Solaris	3.5 GB	3.5 GB
HP-UX	1.75 GB	3.5 GB
AIX	2 GB	3 GB

Of course, if the system does not have that amount of physical memory or if there are multiple memory-hungry processes running on the system, the actual amount of memory that Essbase can use will be lower than the limit indicated in the previous table.

Out of the memory available, roughly 50 MB will be used by Essbase code and fixed-size data structures. Also, some memory will be required for the ASO cache, which will be covered later in this paper.

The memory requirements of the outline members depend on multiple factors. You can use 150 bytes per member, however, as a rough and somewhat pessimistic estimate.

For a more precise estimate, first you must calculate the average memory required for the member name and aliases. This memory requirement for one member can be calculated as total length of the name and all aliases of this member in bytes, plus (number-of-aliases + 1) x 4 bytes.

The per-member memory amount can then be calculated as (90+average-memory-for-names) bytes. In addition to that, every member that has a shared copy requires an additional 50 bytes. Also, attribute dimensions introduce additional memory requirements, which can be calculated as follows: assume that the primary dimension has M members and has N associated attribute dimensions. Then the memory requirement for the attribute mapping will be roughly M x N x 6 bytes.[2]

Memory used by the member names can be freed by enabling namespace paging (see the documentation for PRELOADALIASNAMESPACE and PRELOADMEMBERNAMESPACE configuration parameters); however, the performance of the data loads, and to a lesser degree queries, will be impacted. You may also consider using Hybrid Analysis to store the lower level of details of the largest dimension in the relational database.

ABOUT BUILDING THE ASO OUTLINE

The process of dimension building in Essbase consists of two stages. The first stage begins with the server opening the existing outline for editing, using the same outline API available to the client applications. At this point, two copies of the existing outline are in memory: one in read-only format and one in editing format. Then the input data stream is processed and new members are added to the new outline. Accordingly, the memory requirement during dimension build should be calculated based on the number of members equal to (2 x old-number-of-members)+number-of-the-additional-new-members.[3]

The first stage ends with outline verification, and the new outline is written down to a separate outline file with an OTN extension.

In the second stage, the new outline is loaded in read-only mode and the restructuring process takes place. During that process, both new and old outlines are resident in memory. Also, an additional set of data structures occupy approximately 25 bytes per member.

It is always best to try and combine as many outline changes as possible using incremental dimension build, so that there is only one restructuring phase for all changes. If one restructuring phase is not possible for any reason, it is recommended that you build the large dimension as the last step after the small one, so that at no point, two copies of the large dimension are in memory.

In most cases, it is much faster to modify an existing outline with new changes than to build an outline from scratch.[4] When a very large outline already exists in the server, however, it may be impossible to load a small portion of new members into it, because there will be no memory for the two copies of the outline during build and restructuring. It may still be possible to build the new outline completely, end-to-end from source data.

When modifying a large outline of an existing database, it may be helpful to restart the application before the build. Because the ASO cache is allocated on demand and not needed during dimension build, memory used by the ASO cache is freed for the build and restructuring processing.

To provide a rough idea of dimension build performance, on a test workstation[5] a dimension with 3,000,000 members was built from a text file in five minutes.[6] The largest production outline that we are aware of at this point contains about 10 million members.

When building dimensions with multiple hierarchies and shared members using parent-child data, you will have to keep in mind that ASO applications require that the primary member appear before its shared copies. This means that the parent-child pairs should be ordered so that the first hierarchy is built completely before the build of the secondary hierarchy begins.

DATA SIZING AND THE ASO CACHE

The following information is needed to estimate the amount of disk storage required for the ASO database:[7]

- Number of rows in the input table or text files
- Number of measure columns in a single row
- If possible, average number of non-null, nonzero measures per row

This number can be approximated by using a sample of rows. Unless there is a requirement to distinguish between zeros and missing values in the database, replacing zeros with missing values reduces the size of the database.

The size of an ASO database depends on the number of non-empty cells, which equals the number of rows multiplied by the average number of non-empty measures per row. Each cell occupies roughly 10 to 30 bytes of disk space, depending on the degree of compression.

COMPRESSION DIMENSION

Essbase uses the dimension marked as "Accounts" to compress the cells by storing multiple cell values with a single key. Essbase cannot compress data if none of the dimensions are marked as "Accounts," so it is recommended that you always designate the compression dimension.

Usually the dimension that corresponds to different values in the input rows makes a good choice of compression dimension. For example, if the values for 12 months come in a single database record and the outline has a Time dimension with these 12 months, chances are that Time is a good choice for the compression.

Often, multiple measures come in a single record instead, and an "Accounts" tag should be placed on the Measures dimension. Generally speaking, you should pick the densest dimension. The dimension is a good choice if for every combination of the members of other dimensions, either none or all members of the compression dimension have data. Also, Essbase can store as many as 16 cells with a single key, so a dense dimension with 10 level zero members will provide better opportunity for compression than a dimension with only 2 members.

For a better estimate of the database size, you need to know the key length for the database as well as the average number of cells that can be stored with a single key. The key length in bits is displayed in the ASO database properties. Round the number of bits up to the next multiple of 64 and divide by 8 to get the number of bytes used by the key. The most common key sizes are 8, 16, and 24 bytes.

Assuming that the "Accounts" dimension in the model is completely dense and corresponds to measures in the fact table, the number of cells stored with a single key is simply the number of measures. The number of bytes per cell may then be calculated as (key-length-in-bytes + 8 x number-of-measures) / number-of-measures.

For example, a database with a typical key length of 16 and 6 measures per row of the fact table will use (16 + 8 x 6)/6 or about 11 bytes per cell. If for half the rows the last 4 measures are not present, the average number of cells per key will be (2 + 6)/2 = 4, and the number of bytes per cell will be (16 + 8 x 4)/4 or 12 bytes.

If none of the dimensions is marked as "Accounts," you should use 1 as the number-of-measures parameter. Thus, in the example, if the dimension of measures is not marked as "Accounts," the space requirement will be (16 + 8 x 1)/1 or 24 bytes per cell.

You can sometimes achieve better compression by rearranging the members of the "Accounts" dimension in the order of increasing rarity, so that the level zero members that always have data occur first in the outline, and the members that hardly ever have data occur last.

After the aggregation, the size of the database increases. You have control over the amount of space used by aggregate views; in most cases, 50 percent or smaller increase in the database size after aggregation provides good query performance.

ALLOCATION OF DISK SPACE FOR ASO DATA FILES

The ASO kernel stores data in tablespaces. Two tablespaces are of interest to the server administrator: "default" and "temp." The "default" tablespace is used to store cube cells, both level-zero and aggregated cells. The "temp" tablespace is used for intermediate storage of cells during data load, aggregation, and large queries.

Each tablespace contains one or more file locations; the notion of a file location more or less corresponds to the notion of a volume in the Block Storage kernel. You can change the file locations of the two tablespaces using application properties in Essbase Administration Services (unlike Block Storage, ASO tablespaces are logically considered to be a part of the application, not a database).

For every location, you can specify the physical path where the data files will be created, the maximum size of each individual file, and the total space that all the files at that location are allowed to occupy. Essbase starts creating files in the second file location of a tablespace only after it exhausts the disk space or hits the total space limit in the first location.

It is recommended that you allocate space for "default" and "temp" tablespaces on different physical disk devices, if possible.

DATA LOAD PERFORMANCE

Data loads from multiple data sources or files should always be combined into a single ASO data load using an ASO load buffer. Multiple data loads are accumulated in the buffer and then moved to the final storage location in a single operation.

For more information on using ASO load buffers from MAXL scripts, see the documentation of "alter database create load_buffer" and "import database … to/from load_buffer" MAXL commands. When using the Administration Services GUI to load data into the ASO database, multiple data sources or files correspond to different lines in the data load dialog box; Administration Services uses the load buffer automatically to combine them.

Data load temporarily uses space in the "temp" tablespace; the amount of space required is about the same as the estimated database size.

ASO cache size has an effect on data load performance. For small databases with 2 million input cells or fewer, the default ASO cache size of 32 MB is sufficient. For a larger database with 20 million cells, 64 or 128 MB cache is more appropriate. For a database with 1 billion cells or more, the cache size may be set as high as 512 MB or 1 GB if the available memory

permits it. On machines with less than 4 GB of physical memory, however, it is usually a good idea to keep ASO cache size lower than 25 percent of the available physical memory to leave more space for the OS file cache.

Performance of the data load is very much hardware-dependent. The first stage when the data is being loaded into the buffer is usually CPU-bound, especially when using plain text or SQL load with load rules. Text substitutions, field splits/joins and white space truncation rules are fairly expensive and should be avoided if possible. The buffer commit stage is often IO-bound and may benefit from allocating the storage for "temp" and "default" tablespaces on different physical drives.

To provide an example of data load performance, the data load of 679 million cells on the test workstation took 63 minutes.[8] Database size after the load was 9.36 GB. Many production models cross the 1-billion-input-cells mark.

AGGREGATION PERFORMANCE

The time required to aggregate an ASO database depends mostly on the database size and the specified amount of storage allocated for the aggregate views. The same database with 679 million cells was aggregated to the default stopping size in 40 minutes,[9] to the total database size of 11.0 GB.

By default, ASO uses two threads to perform view aggregation in parallel whenever possible. You can increase the number of parallel threads up to eight by using the CALCPARALLEL setting in the Essbase configuration file. It is recommended that you increase the parallelism setting to the number of CPUs available for the calculation. In some tests, CALCPARALLEL set to eight provided benefits even on four CPU machines, so setting it higher than the number of CPUs could be worth a try.

Cache size configured for the data load should also provide good aggregation performance; however, when increasing the number of calculation threads, you may also need to increase the cache size, because cache memory will need to be divided between more threads.

Calculation may use "temp" tablespace for intermediate aggregation results, so allocating "temp" and "default" tablespaces on different physical devices may help the performance, although it is of less importance than for the data load.

QUERY PERFORMANCE

On a very small database (500 K cells or fewer), queries may be reasonably fast even with no aggregate views. On a larger data set, it is necessary to create aggregate views to improve query response time. The amount of disk space allocated for aggregate views can be controlled by the administrator. If no amount is specified, Essbase attempts to determine a good value for this parameter; however, notice that this default setting is only provided as a first estimate for tuning. It is always a good idea to test query performance for several aggregate sizes and choose the one that provides the optimal balance between the aggregation time and the query response time for your particular database.

When selecting aggregate views, Essbase assumes that all possible level combinations in the database are equally important to users and are queried equally often. In most cases, the selection of aggregate views can be improved by providing Essbase with information about the actual queries that users run by switching to query-based view selection.

An even more important reason to use query-based view selection is that the standard selection never considers any views aggregated along alternative hierarchies. All aggregations are performed along the first stored hierarchy within a dimension. In contrast, query-based view selection will consider and select views aggregated along alternative hierarchies modeled using shared members, and alternative hierarchies modeled using attribute dimensions,[10] as long as these hierarchies are queried during the test run.

It is recommended that you start from selecting and materializing a minimal set of aggregate views using the standard selection. These views will improve performance of test queries and will enable the completion of the test query run in a reasonable time. Also, these views will help queries that were not included in the test run.

After this minimal aggregation is materialized, you can turn on query tracking and run enough queries to represent the common query load for your model. Query tracking has a very low overhead, so it is also possible to deploy the database in production with query tracking turned on.

When enough queries are recorded for you to consider it a representative set, run the selection based on query data and materialize the selected views. Query tracking can be turned on or off by right-clicking the database name in Administration Services or by using "alter database enable/disable query_tracking" MAXL commands. The aggregation wizard contains a check box that switches Essbase to query-based view selection. In MAXL scripts, you can use the "based on query_data" clause of the view selection and aggregation commands.

Notice that the query tracking data is lost when you load additional cells, materialize any views, or shut down the application.

In most cases, the ASO cache configured as described previously in the data load section should work fine for queries. In most cases, increasing the size of the ASO data cache does not improve query performance more than 10 to 15 percent, because the operating system performs its own caching of files.

One exception is when the ASO kernel needs to store intermediate results of a query on a disk temporarily, because

there is not enough space for the results in the ASO cache. For the default 32 MB ASO cache, this happens when the query retrieves more than about 90,000 non-missing cells.[11] The threshold number of cells increases proportionally to the configured cache size, so for a 64 MB cache, that number will be approximately 180,000 cells, and so on. If possible, consider increasing the ASO cache size so that Essbase can store all retrieved cells in memory.

The retrieval buffer size setting has very little effect on the performance of reports and spreadsheet queries in ASO mode. MDX query performance, however, may still be affected by the size of the retrieval buffer.

Complexity of formulas on the queried members is a major factor in query performance. Formulas that need to be executed in cell mode are usually expensive to compute; common examples of such formulas are the ones that contain IIF expressions or cross-dimensional tuple references. Essbase lists all members with cell-mode formulas in the application log.

Hyperion Visual Explorer uses MDX to retrieve data from Essbase. To identify queries that take a very long time, you can refer to the HVE log, usually stored on the client machine, in the directory "%HOMEDRIVE%%HOMEPATH%\My Documents\My HVE Repository."[12]

MAINTENANCE OF ASO APPLICATIONS

The following topics describe the operations that usually follow the initial creation in the lifecycle of an ASO application.

MODIFICATION OF ASO OUTLINES

Any change in the outline causes a restructuring process when the outline is saved on the server. In Release 7.1.2, data is preserved in the following cases:

- When a member is renamed or an alias of a member is changed
- When changes are made on a dynamically calculated member

Changing attribute associations in any way clears all aggregate views, but the input data is preserved.

When a member is deleted from an ASO outline, the record for that member is marked for deletion, but not physically removed from the file. To shrink the outline file after deleting large sets of members, you can use the "Compact Outline" operation in the Essbase Administration Services console.

INCREMENTAL DATA LOADS

When you load data into an ASO database that already has cells loaded into it, Essbase switches to incremental data load mode. An incremental data load is executed as a transaction; that is, either all data is loaded to the database, or the database remains in the state that it was in before the beginning of the data load. Essbase merges the incoming data with the existing data and writes the combined data to a new location within the "default" tablespace. Also, every existing aggregate view is updated by the incoming data and also written to a new location.

At the end of the data load, the space previously occupied by the old data is marked as free. It is not always possible, however, to return this space to the operating system, so the total size of the Essbase data files can be as large as two times the old size, plus the size of the incremental data set. Limiting the file size for the file locations in the "default" tablespace increases the chances that the files can be truncated returning the space to the operating system.[13]

Notice that at some point in the incremental data load process, disk space requirements can be as large as two times the original size of the ASO database for the "default" tablespace plus three times the size of the incremental data stream (2x in the "temp" tablespace and 1x in the "default"). It is very important to ensure that there is sufficient space on the disk, because peak disk usage occurs very close to the end of the data load process. Performing the data load just to find out if the disk has enough space can be time-consuming.

If you load multiple files without using a load buffer, every step will be a separate incremental data load, resulting in worse than possible performance.

RESELECTION OF AGGREGATE VIEWS

Aggregate view selection can be stored in aggregation scripts to be used later. As outline changes build up and new data is loaded into the database, however, the selection may gradually become obsolete and provide less optimal query performance. At some point, it is a good idea to rerun the selection and replace the old aggregation scripts with new scripts. You may want to consider doing this when the increase in the data volume since the last view selection reaches 20 to 50 percent.

Any outline change that modifies the number of levels in any dimension in the outline invalidates all existing view selection scripts; an attempt to use an old script on the new outline results in an error.

Backup

Because text export is not supported for ASO applications, the only way to back up an ASO database is to make a copy of all application and database files when the server is not running. ASO does not support backups of a running application. If files in the application are located outside of the application directory, the files in those locations must be backed up as well.

An archive copy of an ASO application can always be restored to a different machine with the same operating system if the application uses only predefined file locations within the application directory. You will need to create an ASO application and database manually with the same names

as on the source machine, and then shut down the server and override all files in the application directory with the archive copy.

If the application uses any custom file locations, the directory structure on the target machine must be the same as on the source machine. In other words, if on the source machine a custom file location was created on drive E, and on the target machine there is no drive E, then it is not possible to restore the application to that machine.

Unlike ASO data files, ASO outline files are portable across different operating systems and machine architectures.

FOOTNOTE

[1] If the model includes a hierarchy encoded in parent-child tables, determining the number of levels in this hierarchy is somewhat tricky. If the number of levels is not known, the easiest way to determine it may be to build the hierarchy in Essbase.

[2] This formula actually overestimates the memory requirements for all cases except when there is an attribute dimension with more than 2^{16} members.

[3] Strictly speaking, the per-member memory requirements for the editable and read-only formats differ; however, they are close enough to be considered equal for the estimation purposes.

[4] Performance of restructuring may be affected when the number of children per single parent in the outline is very high. This effect becomes noticeable starting from approximately 100,000 members per parent. If the number of children is even higher, it may be faster to rebuild the outline from scratch instead of updating the existing one.

[5] IBM Intellistation ZPro, 2x 2.7 GHz Intel CPUs, 1 GB RAM, single 10,000 RPM SCSI hard drive, Windows 2000 Advanced Server, Essbase 7.1.2, ASO cache set to 256 MB.

[6] This result included a 234-second dimension build of a single large dimension and 26 seconds of restructuring from a small starting outline.

[7] Notice that the size of input text files is almost useless for the estimation of the ASO database size. That is because the number of data values stored in a megabyte of text file can vary dramatically depending on the text file format and other factors.

[8] Tested using export files from a Block Storage database. 1810s loading three files into the load buffer, 1965s buffer commit.

[9] Not including the time used by the view selection process; aggregation time only. Using the default two-thread aggregation.

[10] Even when using query-based view selection, Essbase does not materialize views for queries that restrict the values based on multiple attributes of the same dimension. For example, if the dimension "Product" has attributes "Caffeinated" and "Package Type," queries such as "select total sales for all caffeinated products" may be resolved from a view that contains pre-aggregated values for caffeinated-true and caffeinated-false. In contrast, a query similar to "select total sales for all non-caffeinated products sold in bottles" will be resolved dynamically from the lowest level of the "Product" dimension. If the performance of such queries is still unacceptable after query-based view selection, the only choice is to fall back on modeling alternative hierarchies as full-fledged dimensions.

[11] This is just an estimate, and the exact threshold may vary significantly based on the query. To determine whether a given query uses temporary on-disk storage, restart the application and run the query. If a file is created in the "temp" tablespace, the query needed to store cells on the disk during execution.

[12] HOMEDRIVE and HOMEPATH are environment variables set for every user by the operating system.

[13] On Windows by default, there is no practical limit on the file size, so you may want to set the limit in file location settings to be 1 or 2 GB. On UNIX platforms, Essbase always limits the file size to 2 GB, but you can lower the limit further.

HYPERION SOLUTIONS CORPORATION WORLDWIDE HEADQUARTERS
5450 GREAT AMERICA PARKWAY SANTA CLARA, CA 95054
TEL 408.744.9500 FAX 408.588.8500