Paper 176-27

V9 OLAP: An Architectural Overview

Duane Ressler, SAS® Institute Inc.

ABSTRACT

What's new in V9 SAS® OLAP Server™ software? Just about everything. Completely rewritten for Version 9, SAS OLAP Server software uses the SAS scalable architecture to support high performance scalable multidimensional analysis.

Our experience with HOLAP and Access Control has been integrated into a completely new package designed as a multi-user, scalable server component that natively supports industry standards such as OLE DB for OLAP and the MDX query language. That means that open clients can access OLAP information quickly and easily.

A basic knowledge of the internals of this new implementation can be helpful in getting the best performance out of SAS OLAP Server software by designing the cubes and sizing the hardware to support growing data sources. We will discuss how the different parts of an OLAP cube (metadata, data values, and user sessions/queries) interact and what features are available to help OLAP administrators and users get the best possible performance out of their applications. Specific items include cache management (both memory and disk-based) as well as thread management.

INTRODUCTION

The goals for SAS OLAP Server software for Version 9 were set to meet the needs of our OLAP customers. These goals include:

- Supporting the ability to scale to handle even more users by exploiting available hardware capabilities
- Answering individual OLAP queries faster by exploiting available hardware capabilities
- Following industry standards and emphasizing open architecture
- Providing a client API that supports consistent functionality across clients
- Including cube administration tools
- Supporting more complex queries and treating “time” intelligently
- Providing better support for local languages
- Integrating metadata with other SAS software products
- Allowing customers to log server queries to support better performance analysis and anticipate hardware needs for a growing user base.

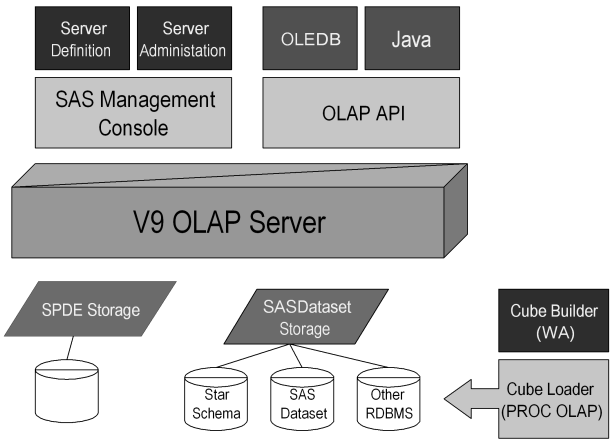
These goals were accomplished by focusing on four main areas of development:

1. Data and server administration
2. Adopting a standard data access API
3. Developing a threaded, scalable server
4. Enabling performance logging and analysis in the server

The following sections discuss these focus areas in more detail.

V9 OLAP – THE BIG PICTURE

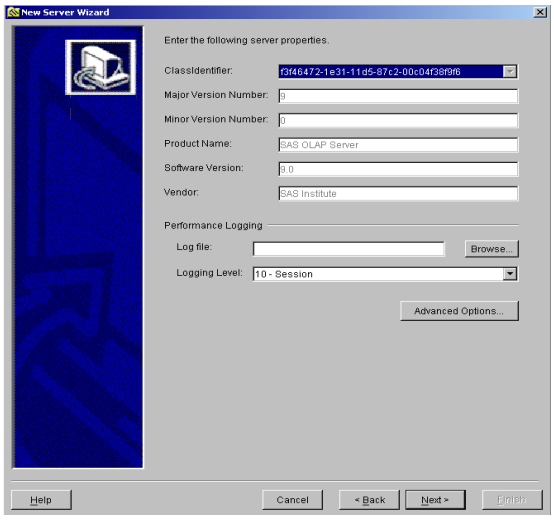
The following figure represents a high level view of SAS OLAP Server software in Version 9. The boxes on the upper left represent server administration interfaces which are used to define and administer the running OLAP server. The boxes on the lower right represent cube administration interfaces, which are used for defining, loading and updating cubes on the server. The boxes in the upper right represent client interfaces for both COM-based and Java-based clients. The middle box represents the OLAP Data Server and the boxes in the lower left and center represent the cube storage options in the server.



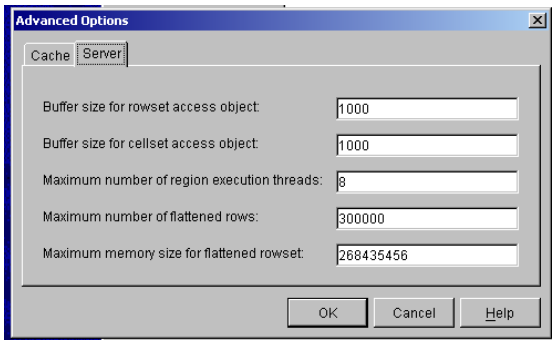
ADMINISTERING THE SERVER

In Version 9 of SAS OLAP Server software, new server administration tools are available to help you build and maintain your OLAP cubes and OLAP data server. This section discusses the new Java-based GUIs and APIs that allow you to administer all your OLAP server components, including the data server itself as well as cube definition and maintenance.

OLAP data server definition and maintenance is performed via plug-ins to the SAS Management Console (SMC). SMC is used to define and administer all server tasks in Version 9 of the SAS System. The OLAP New Server Wizard allows you to define many server options, including the performance logging level and the location of the performance log, both of which will be discussed later in this paper. A sample window from the New Server Wizard is shown below.

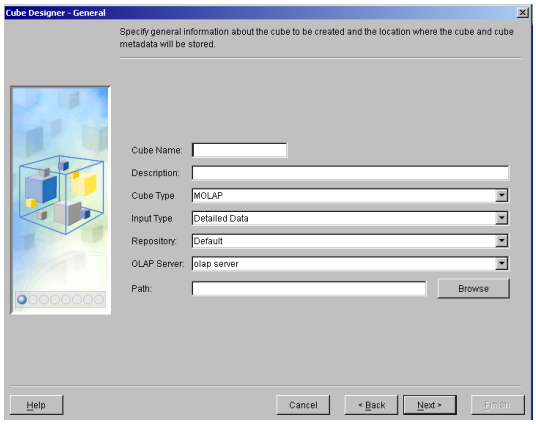


The Advanced Options window of the New Server Wizard (shown below) allows you to set additional server options that affect memory use and performance of the server.

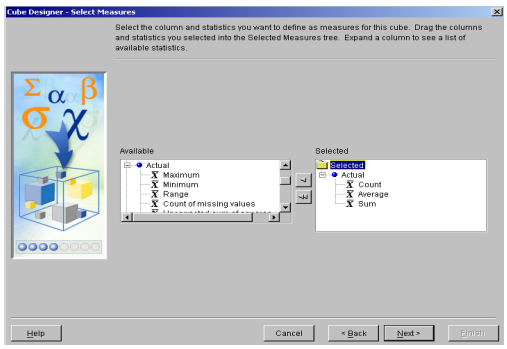


Once the OLAP data server is defined via SMC, you can begin defining OLAP cubes using the OLAP Cube Designer. The cube designer helps you to define and build both multidimensional OLAP (MOLAP) cubes as well as hybrid OLAP (HOLAP or classic OLAP) cubes. The Cube Designer is designed and implemented in such a way as to allow the designer to be used either as a standalone administration tool or as an integrated component of SAS/Warehouse Administrator® software. As an integrated component of SAS/Warehouse Administrator software, the Cube Designer lets you build cubes from data sources defined in the Open Metadata Repository (OMR) and stores the cube definitions in OMR for later use and update.

The figure below shows the Cube Designer window used to define general attributes of a cube, such as the name and type of cube and the OLAP Data Server that the cube is stored on.



The Cube Designer also assists you in defining other aspects of the cube, including its dimensions and dimension levels, the aggregations to be stored with the cube, and, as shown below, the measures to be stored with the cube.



In addition to defining and loading a cube using the Cube designer, you can also define and load a cube using the new OLAP procedure.

```
proc olap data=permdata.cars cube=mddbcars;

METASVR HOST=z555.domain.com PORT=9999
PROTOCOL=bridge
   USERID=duane PW="123" REPOSITORY=default
OLAP_SERVER=polap;

dimension date hierarchies=(dte)
sort_order=ASCENDING;
hierarchy dte levels=(dte);

dimension cars hierarchies=(cars)
sort_order=ASCENDING;
hierarchy cars levels=(car color);

dimension dealers hierarchies=(dealer)
sort_order=ASCENDING;
hierarchy dealer levels=(dealer dest);

measure sales_sum column=SALES stat=sum;
measure sales_n column=SALES stat=n;

run;
```

When a cube is created using the OLAP procedure, the cube definition is stored in OMR. In this way, the cube definition can later be viewed or edited using the Cube Designer. It is also possible to create a cube in batch mode using the OLAP procedure by referencing an existing cube definition stored in OMR.

The data for a cube can be loaded from a number of different data formats, including a base table (as with the MDDB procedure) or a star schema and from a SAS dataset or an RDBMS table.

EXPLOITING YOUR DATA – THE OLAP CLIENT INTERFACES

As mentioned earlier, the OLAP Data Server provides interface APIs for both COM and Java based clients, allowing both client types to fully exploit the capabilities of the OLAP Data Server. The COM interface implements the OLE DB for OLAP (ODBO) specification while the Java interface is based on a SAS software specific data model. Both interfaces follow the interface concepts of ODBO and use the Multidimensional Expression Language (MDX) to query data from OLAP cubes.

OLE DB for OLAP is an extension of OLE DB and is part of Microsoft's data access APIs. COM-based clients have the option of using the ODBO interfaces directly (for example, using Visual C++) or by using ADO MD (for example, in Visual Basic) which is a simplified scripting version of the interface.

Whether you are an application developer writing your own tools to connect to the OLAP Data Server or a data analyst using a point-and-click client to explore your OLAP data, you will be going through four basic steps:

1. Connecting to the server
2. Reading metadata from the server to find the cube, dimensions, levels, members, and measures of interest to you
3. Sending the query to the server
4. Reading the data returned from the query.

When you connect to the OLAP Data Server, you establish a query session on the server. As part of the connection information, you supply access credentials and information about your locale (national language and text encoding) to the server.

The server uses the access credentials to determine the cubes that are visible to you in the metadata and the specific data within a given cube that you can view. Your locale is used to determine the national language used for server messages and in some cases, the language used in the data being returned to you. Once your query session has been established, you can make one or more queries within that single session.

After establishing a query session, the next step is to request the list of OLAP cubes that are available to you on the server. Then, once you have selected an individual cube, you can get additional information about the dimensions, dimension levels, dimension members, and measures in the cube to help you formulate a query.

Once you have determined the contents and shape of the data you want to view, the next step is to submit your query to the OLAP data server. For the application developer, this means sending the MDX language string to the server. For the data analyst, this means processing the data view you have set up using the point-and-click interface.

As mentioned above, OLAP queries are performed using the MDX query language. MDX is an SQL-like language specifically designed for querying multidimensional data and returning the data in either multidimensional or tabular (two dimensional) form. In addition, MDX was designed with client exploitation in mind so that the language constructs can be readily represented in a graphical user interface (GUI). Typically a data analyst using a client connected to SAS OLAP Server software will never see nor write a query in MDX. Instead, they will use a GUI that builds the MDX query behind the scenes.

By using an OLAP query language like MDX, it becomes possible for the server to analyze the entire query and optimize the processing of the entire query. This approach gives us much more flexibility in making use of parallel processing.

Some examples of OLAP clients that connect to V9 OLAP Server software are SAS Enterprise Guide® software, the SAS Business Intelligence Platform, and Microsoft Excel XP. In addition, there are other third party clients that conform to the OLE DB for OLAP specification that can be used for viewing the data in the OLAP Data Server.

After the MDX query has been submitted and processed, the application reads the axis information and data cell values returned by the server. The data can be returned to the client either as a multidimensional ODBO dataset or as a tabular OBDO rowset, depending on the format requested in the query.

The following code fragment illustrates how a Java application might connect to and query from an OLAP Data Server. Note that this example does not query the metadata to formulate a query but instead uses an MDX query string to produce a “canned” report.

```
// Establish connection to OLAP server
OLAPConnection connection = new
OLAPConnection(host, port, username,
password);

// Create instance of data model
OLAPDataSet olapDataSet = new OLAPDataSet();

// Set connection on data model
olapDataSet.setConnection(connection);

// Execute query on data model
olapDataSet.executeQuery(mdxString);
```

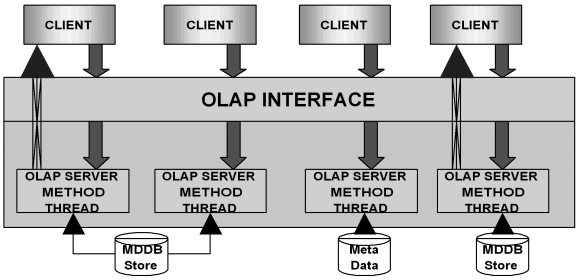
The next code fragment illustrates how a Visual Basic program could connect to the OLAP data server, submit a query, and read the axis information returned by the query. This example uses ADO MD to script the connection to the OLAP Data Server and also uses hard-coded MDX query string.

```
Sub Main()
    Dim obCon As New ADODB.Connection
    Dim obCelSet As New ADOMD.Cellset
    Dim obRec As New ADODB.Recordset

    '=== to connect to a COM server running
    '    on the local machine
    obCon.Provider = "SAS.IOMProvider"
    obCon.Properties("SAS Server Type") = _
      DBPROPVAL_DST_MDP

    '=== submit the MDX query
    Set obCelSet.ActiveConnection = obCon
    obCelSet.Source = mdxStr
    obCelSet.Open

    '=== Print the cellset axis structure and
    '    count the number of cells
    Dim cCells As Long, i As Long, j As Long
    Dim obAxis As ADOMD.Axis
    Dim obPos As ADOMD.Position
    Dim obMem As ADOMD.Member
    Dim obCel As ADOMD.Cell
    Dim strPos As String
    cCells = 1
    For Each obAxis In obCelSet.Axes
        PrintString obAxis.Name _
                    & ", cells on axis: " _
                    & obAxis.Positions.Count
        cCells = cCells * _
                 obAxis.Positions.Count
        For Each obPos In obAxis.Positions
            For Each obMem In obPos.Members
                PrintString vbTab _
                            & obMem.Name _
                            & ": " _
                            & obMem.Caption
            Next
        Next
    Next
```

The OLAP data server allows many different users or sessions to connect to the server simultaneously. This is accomplished using SAS Integration Technologies™ to connect to asynchronous query threads. This gives each user session its own query context as well as enabling the server to take advantage of multiprocessor hardware.

SERVING UP YOUR DATA

In addition to using separate threads for each user session, the OLAP Data Server also uses individual threads for each user query. This allows the server to process many users requests at the same time. The following figure illustrates how each session’s query, whether it’s a metadata query or an MDX query, is processed by a separate thread.



The OLAP Data Server's multi-threaded architecture is discussed in the section *THE OLAP SERVER – SCALING TO LARGE PROBLEMS* later in this paper.

ARMED FOR PERFORMANCE

The OLAP Data Server makes use of the Application Response Measurement (ARM) functionality in Version 9 of the SAS System. ARM is an industry standard for general purpose application logging. The OLAP Data Server uses ARM to supply OLAP data administrators with information to monitor:

- Cube optimization and query performance
- The server load
- Cube and user usage patterns.

Using the output from ARM monitoring, a data administrator should be able to answer questions like:

- Which queries are taking the most time?
- What are the average, minimum, and maximum query response times?
- What time of day is the OLAP Data Server the busiest?
- What is the average, minimum, and maximum result set size?
- Which users are the most active?
- What are the most common queries?

The data administrator can then use the answers to these questions to adjust cube structures and server options to achieve maximum performance.

You can use SAS System options or the SMC New Server Wizard to set the logging level and log file location for a server. The log information for a server session can be processed into SAS datasets and reported on using a variety of SAS reporting tools.

THE OLAP SERVER – SCALING TO LARGE PROBLEMS

In addition to using multiple threads to better scale the OLAP Data Server to multiple users, the server also uses parallel processing to improve query response time. The multi-threaded server improves query response time by processing multiple portions of a query simultaneously.

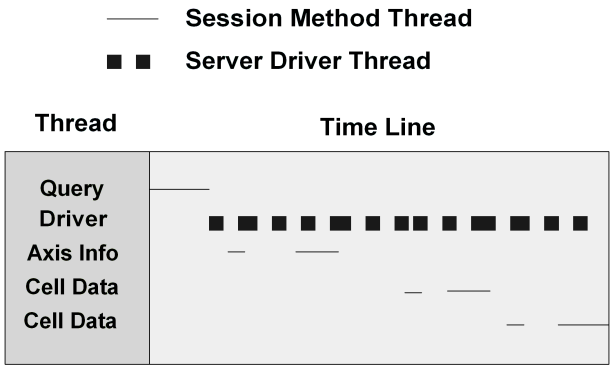
Some of the parallel processing enhancements in the server include planning a query into multiple regions to allow concurrent resolution of parts of the query by separate threads. In addition, MOLAP data is stored in SPDE tables, which take advantage of hybrid indexes and parallel where clause evaluation for fast row selection. The OLAP Data Server also takes advantage of parallel "group by" processing to group on selected rows if the selected aggregate contains hierarchy levels that are not part of the MDX query itself.

Finally, the OLAP Data Server minimizes the size of the MOLAP cube data footprint by using a single numeric column to represent all members of a given dimension or hierarchy. An index is created in the SPDE table for each dimension or hierarchy column. Using this approach, the OLAP Data Server can easily map member names used in an MDX query to their hierarchy ordinal, supporting rapid data retrieval.

A MULTI-THREADED SERVER

The following diagram illustrates how threads are used to simultaneously resolve parts of a simple MDX query. The query thread receives the text of the query for execution on the server. Minimal work is performed in this thread before returning to the client application. The driver thread coordinates the MDX statement parsing, the query execution planning, and the actual query execution.

While the driver thread is running, the client application makes a request to read the axis information that results from the query. This call initiates the Axis Info thread to wait on events raised by the driver thread to return axis information to the client. Likewise, a request from the client for the actual cell data from the query starts the Cell Data thread which also waits on events from the driver thread to return cell information. The client can choose to retrieve the cell data in several chunks for use in the client application and is represented by the two Cell Data threads that execute sequentially.

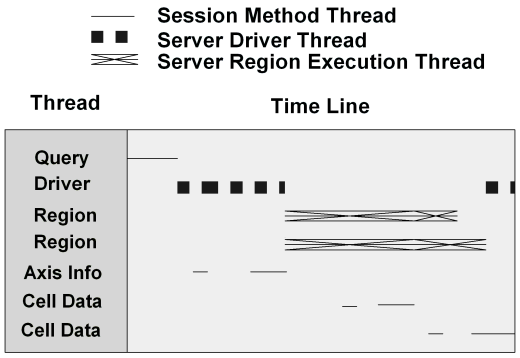


REGION PLANNING AND EXECUTION

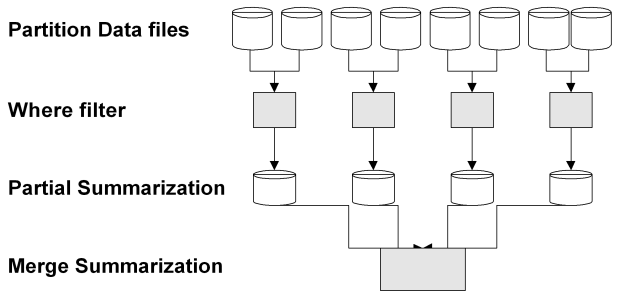
While the previous figure illustrated how a simple MDX query can be processed in parallel, more complex queries are handled in such a way as to further improve the speed of query execution. This is done through region planning to allow data for different portions of a complex query to be retrieved and processed in multiple threads.

An MDX query will always have at least one region planned for query execution. Regions define a portion of the resultant cell set and are usually determined based on the dimension and hierarchy level from which the data is being retrieved or derived. A separate thread is used to execute each region and query execution is complete when all region threads are finished.

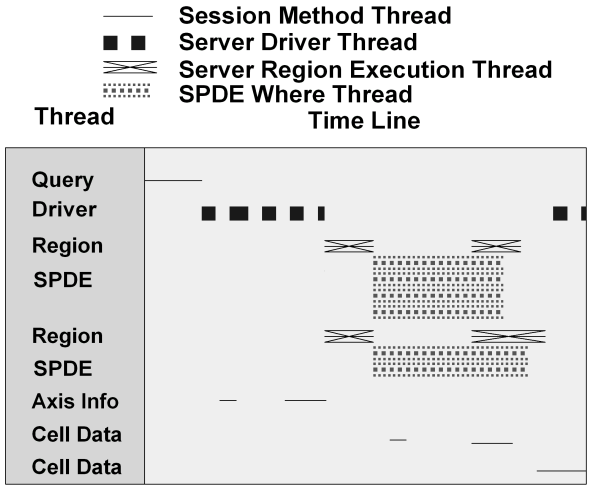
The diagram below illustrates how two Region threads in a single query are executed simultaneously. Note also that the Cell Data threads wait on an event from the Region threads to return the requested data.



Finally, the inclusion of SPDE parallel where processing provides a third mechanism for improving the OLAP Data Server's data retrieval. This diagram shows how the MOLAP data, which may have been divided into data set partitions during cube creation, may have multiple where clause processors working on the various data partitions. These processes in turn produce a subset representing only the data needed to answer the query.



The following diagram again shows how the SPDE parallel where filters allow more of the work of answering the query to occur in parallel. This is a result of the SPDE where filters executing at the same time on separate threads.



DATA CACHE

Another way in which the OLAP Data Server addresses the issue of scaling to large problems is through the use of an in memory data cache. This data cache is designed to keep the most frequently accessed OLAP data in memory so that the server can more quickly respond to the most common data queries. The cache is shared across user sessions, so if one user makes a request for an aggregate of a cube and that aggregate is loaded into the cache, all of the users on the server will benefit from the data being loaded into the cache. In addition to loading data into the cache in response to user queries, the cache also supports the ability to pre-load data into the cache at server start-up.

CONCLUSION

The SAS OLAP Server software in Version 9 of the SAS System provides improved administration support, improved support for multiple users, as well as additional support for the server to scale up with hardware. This is accomplished through the use of Java-based administration tools, SAS Integration Technologies, and a multi-threaded architecture to support parallel processing. The server also provides improved functional capabilities for OLAP client applications through the use of a industry standard API.

TRADEMARKS

ACKNOWLEDGEMENTS

The author would likely to express his sincere appreciation for the assistance provided by the OLAP Server development team in Base SAS Research and Development.

The author may be contacted at:

Duane Ressler
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
Work Phone: 919.677.8000 Ext. 6950
Fax: 919.677.4444
Email: Duane.Ressler@sas.com