Paper 1794-2014

Big Data? Faster Cube Builds? PROC OLAP Can Do It

Yunbo (Jenny) Sun, Canada Post; Michael Brule, SAS Canada

ABSTRACT

In many organizations the amount of data we deal with increases far faster than the hardware and IT infrastructure to support them. As a result, we encounter significant bottlenecks and I/O bound processes. However, clever use of SAS® software can help us find a way around. In this paper we will look at the clever use of PROC OLAP to show you how to address I/O bound processing by spreading I/O traffic across different data mounts to increase cube building and querying efficiency.

This paper assumes experience with SAS® OLAP Cube Studio and/or PROC OLAP.

INTRODUCTION

SAS® code has been introduced throughout this paper to demonstrate how the functionality that PROC OLAP procedure provides can optimize cube creation and query performance on SAS 9.2 environment. It mainly covers optimizing cube creation and cube query performance by splitting I/O traffic to different data mounts. The advantage of incremental update to a cube is not covered. Also, hardware setup will not be covered beyond the prerequisite of having different data mounts created. Readers need to contact their system administrator or service provider to evaluate their environment.

In this paper, the terms data mounts, data partitions, multi thread write/read points or multi mount points are interchangeable.

BIG DATA? FASTER CUBE BUILDS? PROC OLAP CAN DO IT

Data carries power. Organizations use information contained within data to generate value, seek opportunities and implement competitive strategies.

In the past few years, data has been growing at exponential rate. As more data flows into businesses, not only does volume increase, but also different types of data increases. Hundreds of Gigabytes of data accumulated creates challenges in the cube building process as well as in end-user query performance.

IT professionals usually face infrastructure limitations, such as lack of memory capability, shortage of storage, and overloaded I/O bandwidth. Business users often complaint about query performance, data latency and not capturing enough data to support their decision-making needs.

IT professionals prefer to build cubes as quick and small as possible to fit IT infrastructure. On the other hand, business end-users may require daily refreshed cube with large amount of pre-calculated aggregations and indexes on big data in order to achieve timely market competitive advantage.

In addition, cost control on IT budgets and the profit generating business goal may sometimes cause cube building process and end-user query performance to be at odds with each other. A balanced approach is required to overcome those challenges. Ultimately, achieving an organization's long term business strategies and short term business tactics should drive decisions and designs throughout the cube building process. In other words, cube building is a business driven process and not an IT driven decision. Fortunately, PROC OLAP has options to optimize both.

Searching on the web, you will find lots of papers that talk about cube building efficiency. Here, focus will be given to the following SAS® 9.2 PROC OLAP Procedure Options.

1. DATAPATH=

First, **DATAPATH =** (*'pathname' ...'pathnameN'*) option. This PROC OLAP statement or AGGREGATION statement option specifies the location of one or more partitions in which to place aggregation table data. The first aggregation file location is randomly selected from path specified in the option during the build process. This means that the distribution of the files will be different for each build. When building aggregations or different cubes concurrently, the benefit will be two fold:

1. the files will be more evenly distributed across different mount points,

2. and the I/O load during the build and querying time will be more evenly distributed across available I/O channels.

Instead of barraging one mount point, this load sharing approach will split I/O traffic across many mount points at the same time, thus increasing your overall I/O throughput rate. This leads to improving cube creation speed during the build process as well as improving read performance during the query process.

In this paper, the following example uses three mount point locations showing on how to split aggregation table data.

Step 1: Verify cube creation required partition location on different mount points is available. If not, create one. In this paper, &path1./&cubename., &path2./&cubename. and &path3./&cubename. are created before cube creation.

```
%let path1 = /nfshome/sunyunb/SGF/data1;
%let path2 = /nfshome/sunyunb/SGF/data2;
%let path3 = /nfshome/sunyunb/SGF/data3;
%let cubename = SGF14;
%let fref = cube;

/* Code for Multi-Mount Point Location creation      */
/* Add check/create Location                         */
/* Add Location for DATAPATH and INDEXPATH options   */

%Macro DCreate (fref, dpath, dcube);
       %let rc=%sysfunc(filename(&fref., &dpath./&dcube.));
       %let dir = %sysfunc(dopen(&fref.));
       %if &dir. = 0 %then
             %let newdir = %sysfunc(dcreate(&dcube., &dpath.));
       %else %let newdir = %sysfunc(dclose(&dir.));
%Mend Dcreate;

%DCreate(&fref., &path1., &cubename.);
%DCreate(&fref., &path2., &cubename.);
%DCreate(&fref., &path3., &cubename.);
```

Step 2: Place 3 mount point locations in PROC OLAP Statement DATAPATH option. For example, if &path1. is randomly selected to be the first aggregation file location, the first partition of first aggregation table will be placed in to directory &path1.; the second partition of the table into directory &path2.; the third partition of the table into &path3.; the fourth partition of the table into &path1., and so on. By doing this, I/O traffic will be spit to different mount-points and cube building time will be decreased, as especially if parallel aggregation creation is enabled.

```
PROC OLAP

       DATA=CE.CE_FACT
       DRILLTHROUGH_TABLE=CE.CE_V2
       CUBE=SGF14
       PATH="&path1."
       DESCRIPTION="SGF 2014 Cube"

       DATAPATH=("&path1." "&path2." "&path3.")

       COMPRESS
       INDEX
       INDEXPATH=("&path1." "&path2." "&path3.")
       CONCURRENT=3
       PARTSIZE=256
       ;
```

… … … … …

… … … … …

PROC OLAP automatically creates NWAY aggregation for the new cube or fully refreshed cube (unless specifically disabled). The NWAT is the aggregation crossing of all dimension levels and usually is the largest in the cube.

Because NWAY is spread across all the mount points, any additional aggregation builds in the AGGREGATION statement will also benefit from the multi thread reading of the NWAY (NWAY is typically used to build the higher level aggregations).

2. INDEX AND INDEXPATH=

Second, INDEX and INDEXPATH=(*'pathname' ...'pathnameN'*) options. For fast cube build, NOINDEX option can be used. However, the lack of indexes will affect end-user query performance. INDEXPATH specifies the locations of the index component files that correspond to each aggregation table partition as specified by the DATAPATH= option. INDEXPATH locations could be different from DATAPATH locations. Similar to DATAPATH option, when building a cube, it takes advantage of multi thread write.

```
PROC OLAP

        DATA=CE.CE_FACT
        DRILLTHROUGH_TABLE=CE.CE_V2
        CUBE=SGF14
        PATH="&path1."
        DESCRIPTION="SGF 2014 Cube"
        DATAPATH=("&path1." "&path2." "&path3.")
        COMPRESS
        INDEX

        INDEXPATH=("&path1." "&path2." "&path3.")

        CONCURRENT=3
        PARTSIZE=256
        ;
```

… … … … …

… … … … …

3. CONCURRENT =

This option specifies the maximum number of aggregations to create in parallel. The default value is 2. Experiment should be based on individual IT resource. Here, the option is setting to 3. That means maximum 3 aggregations could be building at the same time.

```
PROC OLAP

        DATA=CE.CE_FACT
        DRILLTHROUGH_TABLE=CE.CE_V2
        CUBE=SGF14
        PATH="&path1."
        DESCRIPTION="SGF 2014 Cube"
        DATAPATH=("&path1." "&path2." "&path3.")
        COMPRESS
        INDEX
        INDEXPATH=("&path1." "&path2." "&path3.")

        CONCURRENT=3

        PARTSIZE=256
        ;
```

… … … … …

… … … … …

Basically, limiting aggregations that run concurrently gives each aggregation enough CPU resources to build and can be used to reduce I/O contention in I/O bound systems.

4. OTHER COMMON USED PERFORMANCE OPTIONS

- WORKPATH=, it gives one or more locations for temporary work files. It equivalents to DATAPATH= for aggregation. The WORKPATH is used when MEMORY is limited and it must write temporary data to disk. Note: this may be mitigated by increasing the MEMSIZE setting during the cube build process.
- ASYNCINDEXLIMIT=, it limits the number of indices that will be created in parallel during cube building. It is similar as CONCURRENT for aggregation.
- INDEXSORTSIZE=, it specifies the available memory when creating aggregation.
- PARTSIZE=, it specifies the size of aggregation table partitions and their index components.

CONCLUSION

In today's fast-paced marketplace, utilizing, leveraging and maximizing IT resources and capability to gain clear business insight and make quick decisions will enable organizations to outperform their competitors. Therefore, unlocking the power of data is the key to success. The above options discussed in the paper will help IT professionals to overcome their challenges and deliver business value.

REFERENCES

- SAS® 9.2 OLAP Server User Guide, SAS Document
- SAS KNOWLEDGE BASE: DATAPATH and INDEXPATH locations are randomly selected for aggregation files

  Available at http://support.sas.com/kb/38/755.html

RECOMMENDED READING

- *SAS® 9.2 OLAP Server User Guide*

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name: Yunbo (Jenny) Sun
Organization: Canada Post
Address: 2701 Riverside Drive, Suite N250
City, State ZIP: Ottawa, Ontario K1A 0B1
Work Phone: 613-734-8744
Fax: 613-734-8814
Email: yunbo.sun@canadapost.ca

Name: Michael Brule
Organization: SAS Canada
Address: 360 Albert Street, Suite 1600
City, State ZIP: Ottawa, Ontario K1R 7X7
Work Phone: 613-755-2305
Fax: 613-231-8526
Email: Michael.Brule@sas.com

APPENDIX A: LIST OF OPTIONS USED FOR PERFORMANCE

The following options could be used to optimize cube creation and query performance.

ASYNCINDEXLIMIT=

COMPACT_NWAY

COMPRESS | NOCOMPRESS

CONCURRENT =

DATAPATH=

INDEXPATH=

INDEXSORTSIZE=

MAXTHREADS=

INDEX | NOINEX

PARTSIZE=

SEGSIZE=