Paper 135-2011

Best Solutions for Tuning Performance of ETL Jobs in SAS® Data Integration Studio

Qingfeng Liang, Community Care Behavioral Health Organization, Pittsburgh, PA

ABSTRACT

SAS® Data Integration Studio is a great tool for building and maintaining data warehouses and data marts. The performance of the extract, transform, and load (ETL) job is critical for building data warehouses and data marts. This paper discusses the time-consuming data transformations related to ETL processes in SAS Data Integration Studio. The performance for each data transformation is benchmarked and compared. The best solutions to speed up ETL job performance are discussed in detail. The paper also suggests the best practices for designing efficient ETL jobs.

INTRODUCTION

When data warehouses and data marts are built, significant numbers of ETL (Extract, Transform, Load) processes need to be implemented. The time to build and to update data warehouses and data marts depends heavily on the performance of the ETF processes. SAS Data Integration (DI) Studio is a special tool to help to simplify the ETL process. It offers some important features which were really beneficial to the ETL process. First, the ETL process design in DI Studio is an intuitive, point-and-click tool which makes it easy to build logical process workflows and to quickly identify input and output data flows. Second, this tool provides a powerful yet easy-to-use transformation language that supports collaboration and the reuse of processes and ETL jobs. Transformations can run on any platform with any data source. SAS programmers can also create their own custom transformations which incorporate specific business logic. Third, it includes an embedded data quality feature which allows data quality processing to be imbedded within ETL jobs.

As data volume and complexity increase tremendously in recent years, the performance of the ETL job is very critical to build data warehouses and data marts. This paper focuses on improving the efficiency of the ETL job on EXTRACT, TRANSFORM and LOAD processes on large volume of data. In terms of large volume of data, table should have at least more than 10 million records. The transformation and techniques available in DI Studio for each process will be addressed and the performance will be compared. The paper will also provide best practices for ETL processes.

EXTRACT

In EXTRACT process, data from source systems is extracted to build the target table. The job of this process creates either a view or physical work table for each data set that will be used as a source for the next job. It is also possible to consider data type transformations at this stage as well to ensure that the structure of the extract table mimics that of the final target table. Note that if subsequent jobs were to fail, the staging file generated in this step can be used to avoid having to re-run the job that created it.

Several different transformations can be used to extract data.

- EXTRACT TRANSFORMATION
- SORT TRANSFORMATION
- USER DEFINED SET TRANSFORMATION

EXTRACT TRANSFORMATION

The SAS code behind the EXTRACT transformation is the SELECT statement from PROC SQL. The target table can be sorted or unsorted depending on whether the ORDER BY statement is used.

SORT TRANSFORMATION

If the target table needs to be sorted by certain fields, the SORT transformation is a good choice to extract data. The SORT transformation utilizes PROC SORT procedure from Base SAS. There are several built in options which can be used to tune up SORT performance.

USER DEFINED SET TRANSFORMATION

This transformation is not one of the transformations built in SAS Data Integration Studio, but you can easily either put data step SET statement in the User Written Code transformation or define your owned SET transformation. You can also throw in some data options which are allowed you to control the process, such as WHERE, OBS=, etc. Figure 1 shows the user defined SET transformation.

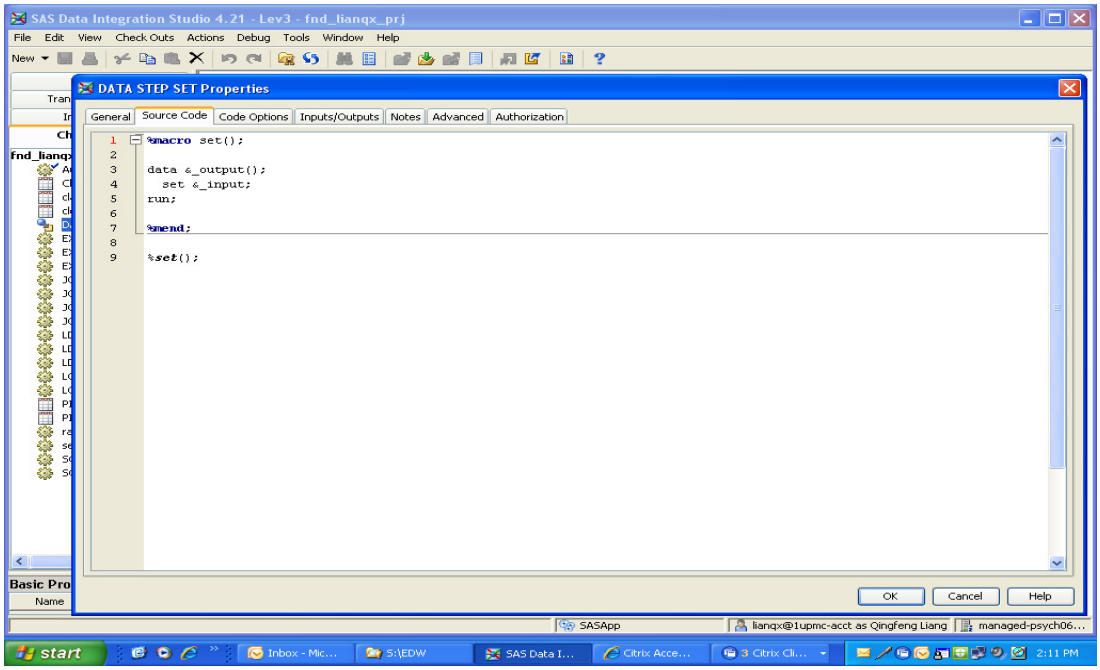

Figure 1. User Defined SET transformation

PERFORMANCE COMPARISON

CPU time, I/O operation and Memory usage were used to benchmark performance of three different extracting techniques. CPU time is the amount of time the Central Processing Unit uses to perform extracting task, I/O operation measures the read and the write operations performed as data from storage to memory or from memory to storage. It measures the difference between real time and CPU time. Memory usage is the size of the work area required to hold data, buffers, etc.

The performances of three transformations outlined above have been benchmarked on a desktop with Window XP, SAS Data Integration Studio 4.21 and SAS 9.2 installed. The source table “Claim_Detail” has 39,394,412 rows and is used as base source to construct new source tables with different sizes. Different percentages of records (25%, 50%, and 75%) were randomly extracted from and saved as new source tables. Each transformation will run 10 times for each different size of source tables and the performance was the average of 10 runs. All the runtime statistics were collected after each job ran successfully.

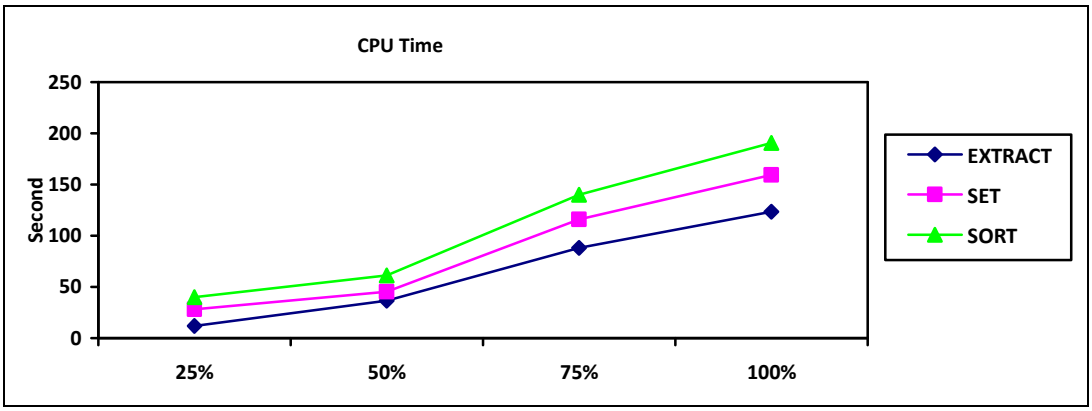

Figure 2: Performance of CPU time on EXTRACT

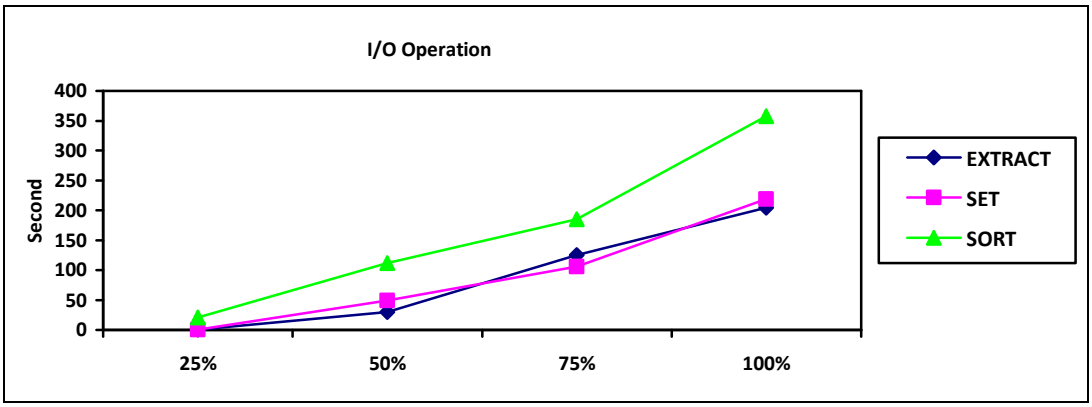

Figure 3: Performance of I/O operation time on EXTRACT

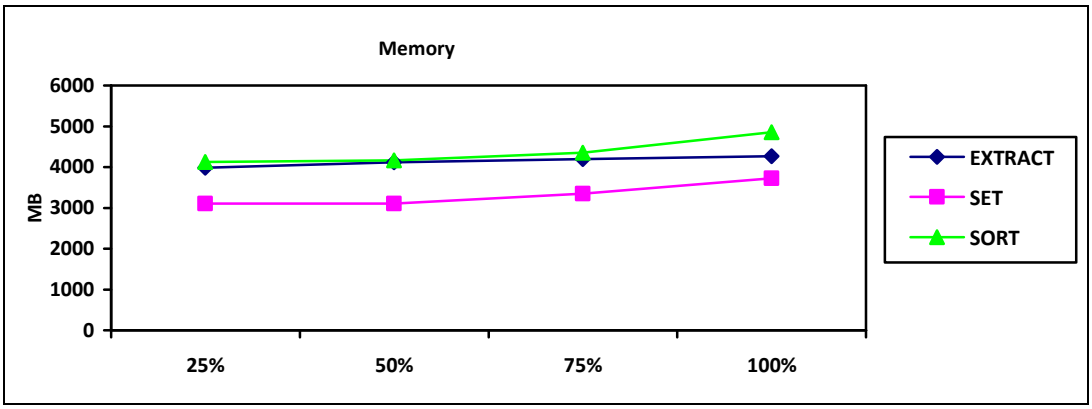

Figure 4: Performance of Memory on EXTRACT

EXTRACT transformation has the best performance on both CPU time and I/O operation, SET transform has the second best performance. SORT transformation also performs well when the size of the source table is relatively small, especially when the target table needs only few columns to be ordered from the source table. SET has the best performance on memory usage; one of disadvantages for SET is that data in target table can't be ordered by certain columns.

TRANSFORM

In TRANSFORM process, it could involve all kinds of data manipulation tasks. In this paper, we will focus on two specific areas which are common and very time consuming:

- Join
- Sort

JOIN

In SAS Data Integration Studio several transformations can perform tasks of joining tables. Each transformation equips with different underlying joining technique. Four transformations are commonly used:

- SQL Join Transformation
- User Defined Merge Transformation
- Lookup Transform
- Data Validation

Data Validation transformation utilizes huge volume of memory beside other limitations, and it is not practically useful for data with large volume. We will not discuss it in this paper.

SQL Join

SQL JOIN implements PROC SQL to join a source table and a lookup table on certain common columns. SQL JOIN performs the task with one of three algorithms: sort-merge, index join, and hash object.

Sort-merge is the algorithm most often selected by SQL optimizer. The source table and the lookup table are sorted before they can be joined. The problem with this algorithm is that sorting is a very resource intensive process which will occupy a lot of computing resource.

An index join looks up each row of the lookup table by querying an index of the source table. The SAS SQL optimizer considers an index join under following conditions: (1) the join is an equijoin – tables are related by equivalence conditions on key columns; (2) here are multiple conditions, and they are connected by the AND operator; (3) the source table has an index composed of all the join keys. If these conditions are satisfied and the index join usually performs better than the sort-merge join.

Hash join defines lookup table as a hash object and SQL sequentially checks each row in source table to create the matched result set. An internal memory-sizing formula determines whether or not a hash join is chosen.

There are several options that you can influence SAS SQL optimizer to use which algorithm, but you cannot force SAS SQL optimizer to use the one you think is going to have the best performance.

User Define MERGE Transformation

There is no MERGE transformation in SAS Data Integration Studio, but customized MERGE transformation can be built easily or you can add data step MERGE statement along with PROC SORT into User Written Transformation. Both the source table and lookup table need to be sorted prior to MERGE. The following screen shot shows the user define MERGE transformation.

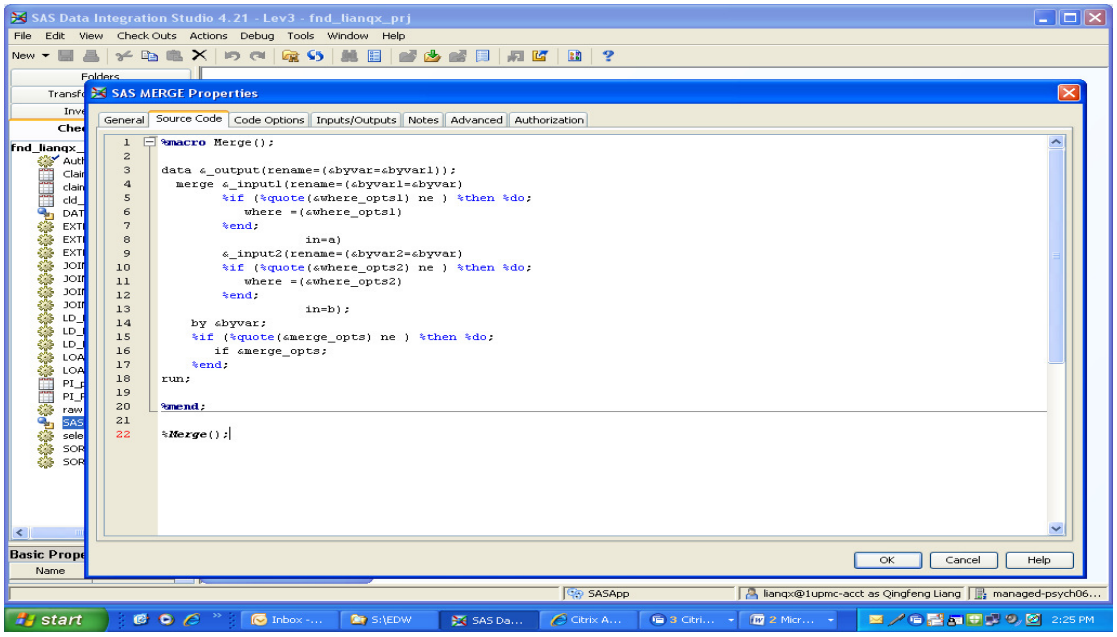

Figure 5. Customized MERGE transformation

LOOKUP TRANSFORM

This transform is based on SAS hash object technique and builds a hash table from lookup table. Each record in the source table will be checked against the hash table to determine if the current key is found in the hash table during the lookup process. If the key is found, write the row to a temporary table for next step processing.

PERFORMANCE COMPARISON

The source table “Claim_Detail” has 39,394,412 rows and the lookup table “Claim” has 20,208,536 rows. “Claim_Detail” is also used as base source to construct new source tables with different sizes (25%, 50%, and 75%). Each transformation run 10 times for each different size of source table against lookup table, and the performance statistics was the average of 10 runs.

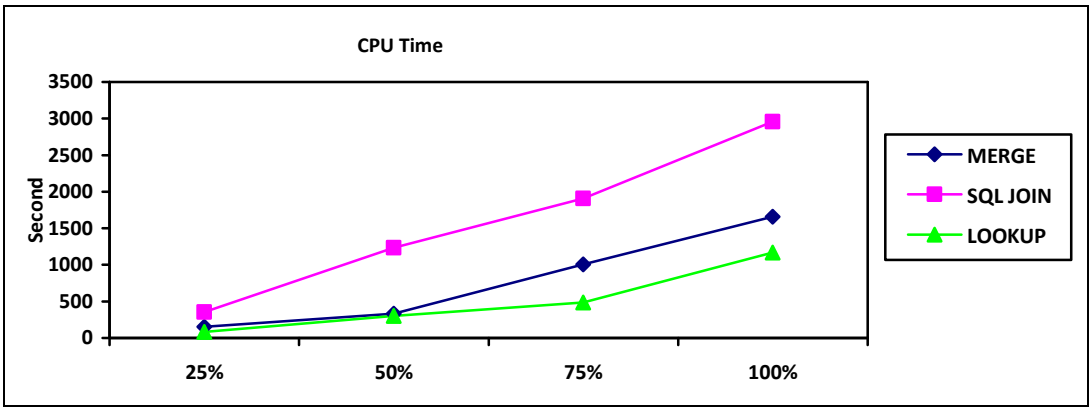

Figure 6: Performance of CPU time on JOIN

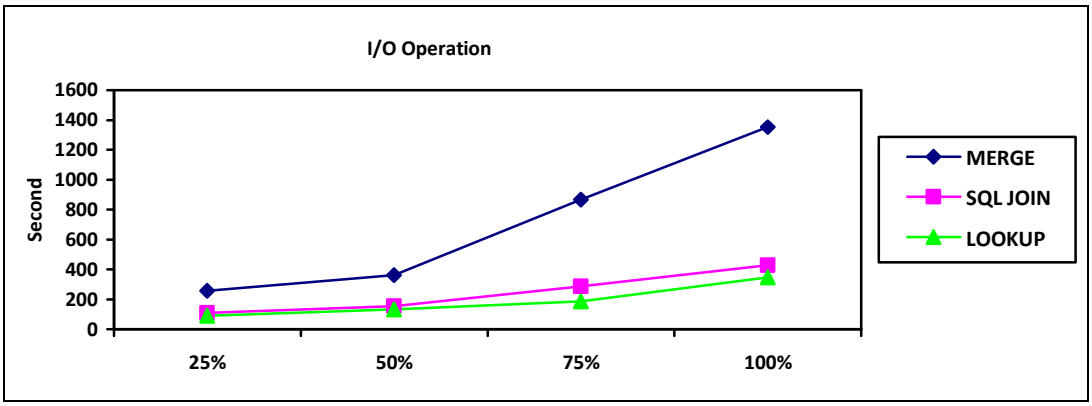

Figure 7: Performance of I/O operation time on JOIN

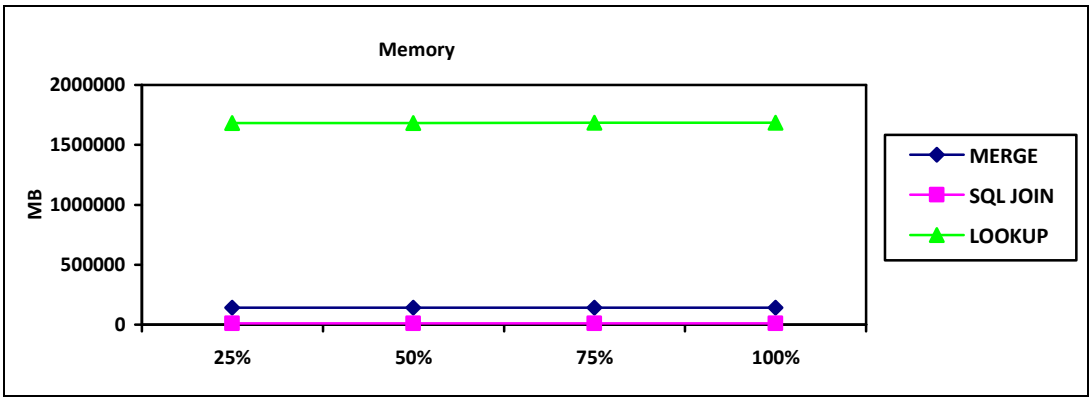

Figure 8: Performance of Memory on JOIN

The results show that LOOKUP performs better than SQL JOIN and MERGE on both CPU time and I/O operation, but it also required huge volume of system memory. Keep in mind that LOOKUP has memory limitation if you decide to use it. SQL JOIN uses shorter CPU time and less amount of memory comparing to MERGE, but MERGE has shorter I/O operation time.

BEST PRACTICES FOR JOIN

Besides choosing the best transformation to join tables, the following best practices can also help you optimize join performance. Most of them can be applied to any of the algorithms discussed above.

- Reduce unnecessary columns
- Optimize order of columns
- Presort data if possible
- Data aggregation at earliest stage
- Use SQL JOIN over other techniques if data volume is huge

SORT

Both SORT transformation and SQL JOIN with Order By option can be used to sort data. SORT transformation implements PROC SORT procedure from base SAS. Tuning sort performance is both a science and an art. The tactics apply could either increase performance or decrease performance. The options directed to PROC SORT like utility files, sort size, memory size, etc can affect performance significantly. The combinations of different options can

be experimented to tune the performance. The SORT transformation also includes THREADS option. With this option, it enables the SORT to do multi-threaded processing which means that multiple units of work are available to be scheduled for concurrent execution by operating system. In this paper, we will investigate SORT with default options, SORT with THREADS and SQL JOIN with Order By option.

PERFORMANCE COMPARISON

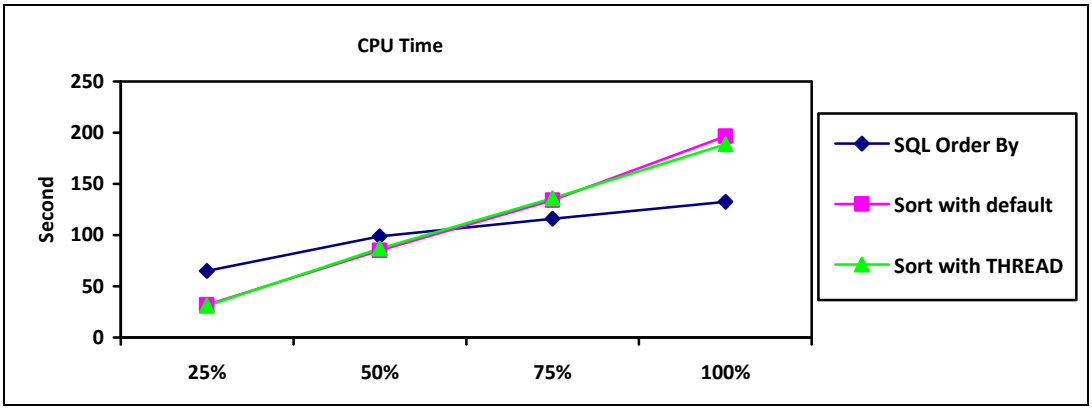

Figure 9: Performance of CPU time on SORT

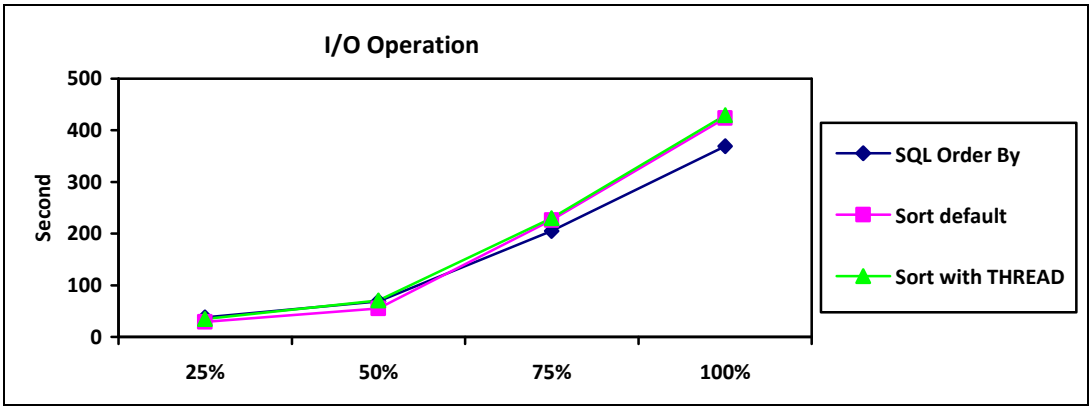

Figure 10: Performance of I/O operation time on SORT

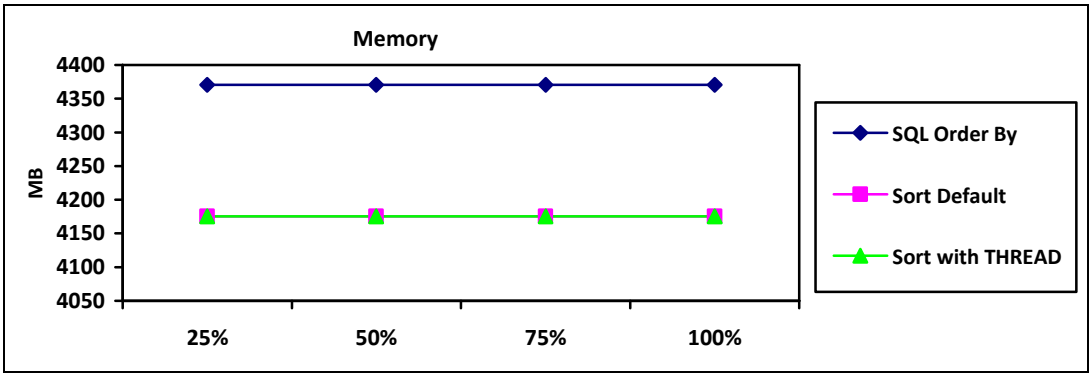

Figure 11: Performance of Memory on SORT

Surprisingly, SQL JOIN with Order By option outperforms SORT transformation not only on CPU time but also on I/O operation, especially when the data volume is increasing. SORT uses less memory than SQL JOIN. The performance between SORT with default options and SORT with THREADS is also identical. The reason behind is that SORT automatically picks multi-threaded processing with default when system resources are available.

LOAD

The purpose of this process is to load the temporary table or intermediate table into the final target table. The LOAD process is built into the transformation called "Table Loader". Two different load techniques are implemented in "Table Loader", REPLACE and UPDATE.

REPLACE

In this approach, a table is completely rebuilt from input data sources. Typically, the data is deleted and new data is populated using a bulk load facility. Two advantages for using table replacement: (1) the bulk load facility is extremely fast; (2) 100% of rows will be refreshed. But there are also two disadvantages: (1) the whole table needs to be replaced, it could be common in data warehouse that some of those tables have over tens of millions of rows in each. It could become time and resource consuming process. (2) If there is any error during the LAOD process, historical data may be lost because the historical data is supposed to be deleted before the new data is populated. Two options are available to perform REPLACE: APPEND and INSERT.

UPDATE

In this approach, a table is modified instead of being totally replaced. Input data sources are scanned to locate new and changed rows. The old values are overwritten with the updated values. The advantages of this method are: (1) Only rows with updated values or new rows are posted to the target table, so the output volume could be significantly lower; (2) Most of historical data may be preserved if any errors occurs during the UPDATE process. The disadvantages are: (1) Input sources may not have clear indicator of new and changed records. (2) Process to identify new and changed rows is costly, especially for a table has a lot of columns and rows. Basically, a column by column comparison of the data in input rows against the data in the target rows needs to be conducted, it becomes very costly when the numbers of rows exceed millions or ten millions.

Which approach is better in terms of data refresh? Theoretically, UPDATE should be a better approach. But in reality, a lot of fact tables are not designed in a way which a single field can be used to identify any new or changed rows. Also it could be very costly to construct such field. Therefore, UPDATE approach becomes less valuable when a fact table has over millions rows and over 50 columns. We suggest REPLACE approach if the data volume is huge in practice. The performances of two options (Append and Insert) under REPLACE are benchmarked for comparison.

PERFORMANCE COMPARISON

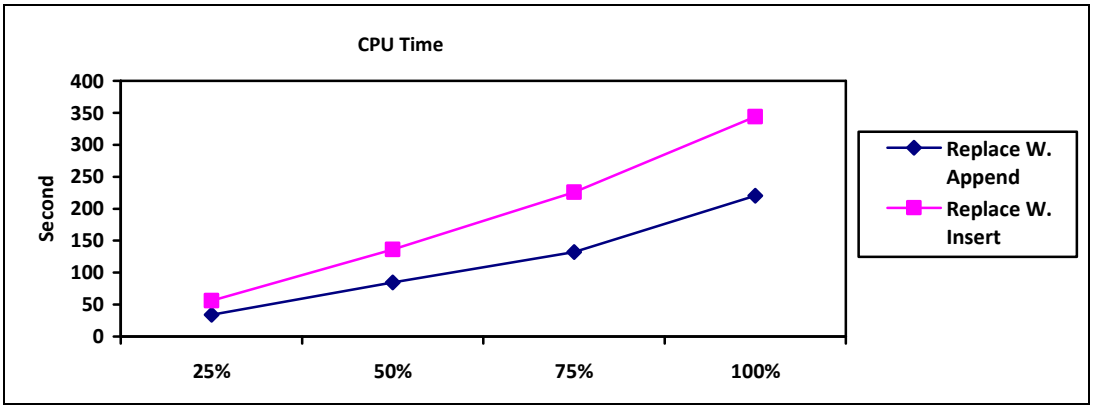

Figure 12: Performance of CPU time on LOAD

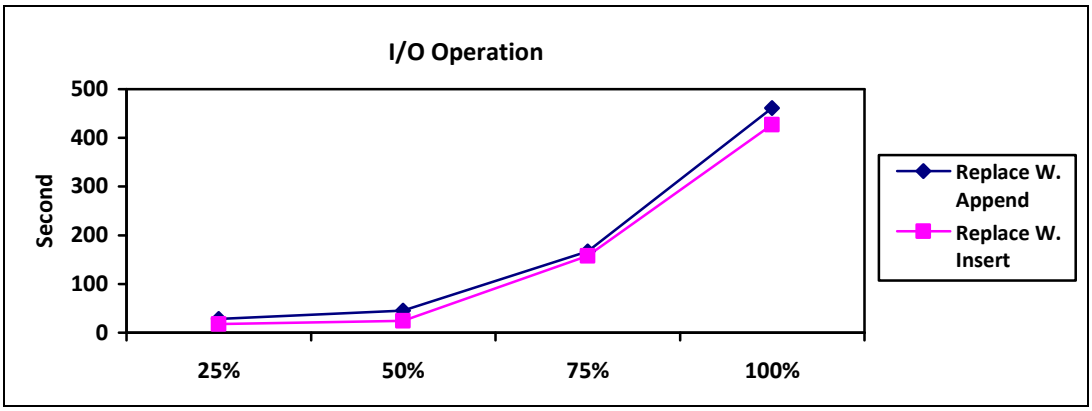

Figure 13: Performance of I/O operation time on LOAD

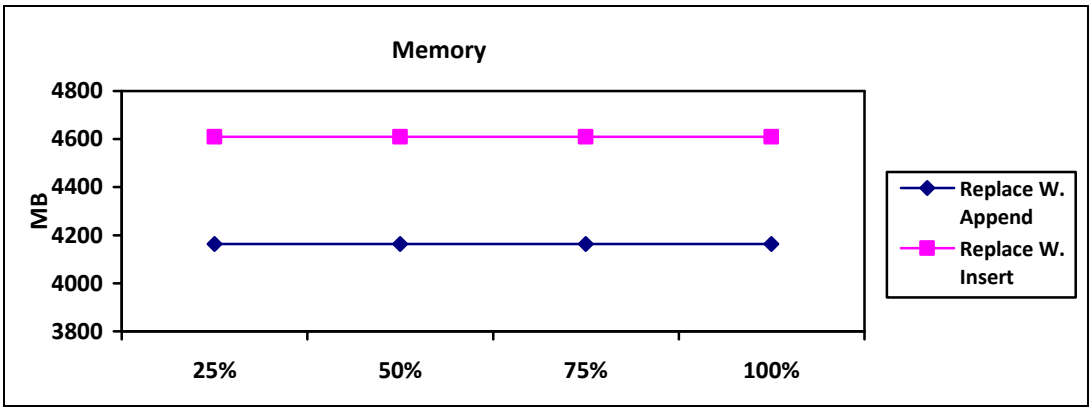

Figure 14: Performance of Memory on LOAD

The results show that REPLACE with Append performs better that REPLACE with Insert on all three indicators.

BEST PRACTICES FOR LOAD

- Prefer replace whole table over update table when load table with large volume
- Prefer option Append over Insert in REPLACE
- Bulk loading into relational database
- Leave constraints off before loading
- Remove indexes before loading

CONCLUSION

According to our experience and the results of benchmarking, optimal ETL performance can be achieved by utilizing built in transformations like EXTRACT, SQL JOIN, LOOKUP, and SORT in SAS Data Integration Studio. Those transformations are well built and usually generate optimal performance. Some base SAS procedures and statement, like MERGE, SET are also consider as good alternatives under different circumstances. All the solutions discussed in this paper are based on SAS Data Integration Studio 4.2, if you are using different version of SAS Data Integration Studio, these solutions may not be applicable and the results may vary.

REFERENCES

Doyle M, Liang Q, Ciocco G, Mesiano M, Wang S. (2009) Using SAS to Build the Managed Care Data Warehouse at Community Care Behavioral Health Organization, 169-2009, Proceeding of SAS Global Forum 2009, Washington, DC.

Liang Q, (2009) Choosing the Right Technique to Merge Large Data Sets Efficiently, 071-2009, Proceeding of SAS Global Forum 2009.

SAS Institute Inc. (2009) SAS Data Integration Studio 4.2: User's Guide. Cary, NC 27513: SAS Institute Inc.

Rausch, N. A., Wills, N. J. (2007) Super Size It!!! Maximize the Performance of Your ETL Processes, 108-2007, Proceeding of SAS Global Forum 2007.

SAS Institute Inc. (2007) Using SAS Data Integration Studio to Create Efficient ETL Processes – Course Notes. Cary, NC 27513: SAS Institute Inc.

ACKNOWLEDGEMENTS

CONTACT INFORMATION

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

Qingfeng Liang, Lead Programmer
Community Care Behavioral Health Organization
Suite 700, Chatham One Center
Pittsburgh, PA 15241
Phone: 412-402-8748
Fax: 412-454-2177
E-mail: liangQ@ccbh.com
Web: www.ccbh.com