Paper 103-2011

Five Ways to Speed Up Your Data Loading Using SAS/ACCESS® for Relational Databases

Douglas B. Liming, SAS Institute Inc., Cary, NC, USA

ABSTRACT

When moving large amounts of data into a database, the mantra is: Quicker is better. SAS/ACCESS has always leveraged the native database load utilities using the BULKLOAD=YES option. Most users are familiar with BULKLOAD, but here are five additional ways to turbocharge BULKLOAD to make loading data even faster. Even advanced users might not know about some of these undocumented options.

INTRODUCTION

When moving large amounts of data into a database, the mantra is: Quicker is better. SAS/ACCESS has always leveraged the native database loader utilities using the BULKLOAD=YES option. Most users are familiar with BULKLOAD, but can anything be done to make it even faster? The answer is yes and it depends. It depends on the database's load utilities, on how you have your database's client configured, on what version of the client you are using, and on how SAS® leverages them. We are going to look at five examples of some leading database utilities and at what BULKLOAD options you can use to further leverage the relational database load utilities.

THE TEST DATA

For this paper, in all the examples below, I am using a very narrow table of S&P 500 stock data with a SAS data set of a little over ten million rows (10,116,050). I ran SAS® 9.3 on a 64-bit Linux (x86_64) Red Hat 5 box, with 64-bit database clients.

A describe of the table looks like the output in Output 1.

```
Name                                        Type
 ------------------------------------------ ----------------------------
 STOCK                                       VARCHAR2(5)
 CLOSING_DATE                                DATE
 OPEN                                        NUMBER(10,3)
 HIGH                                        NUMBER(10,3)
 LOW                                         NUMBER(10,3)
 CLOSE                                       NUMBER(10,3)
 VOLUME                                      NUMBER(21,3)
 ADJ_CLOSE                                   NUMBER(10,3)
```

Output 1. Describe of the Data Used during the Test Scenarios

Now, on to five ways to load data faster!

NUMBER 1: DB2 AND CLILOAD (BL_OPTION=CLILOAD)

To start off with, let us look at DB2. By default, when using BULKLOAD=YES and the SAS/ACCESS Interface to DB2, you will use the DB2 load utility named IMPORT. However, you have three choices when bulk loading with SAS/ACCESS to DB2: LOAD, IMPORT, and CLI LOAD. All of these are installed with the DB2 Administrator Client. If you install only the Runtime Client, you will not get any of the load utilities and therefore cannot use BULKLOAD. For loading tasks, there is a performance lift using CLI LOAD. When CLI LOAD is used, SAS simplifies the invocation of LOAD, so you do not have to know all the details of the options and configuration required to invoke LOAD. You can use CLI LOAD only if your DB2 client is version 7 fix pack 4 and above. Now that DB2 is currently at version 9.7, that should not really be a limitation, yet it needs to be noted.

First, Output 2 is from a standard DATA / SET assignment. The SAS/ACCESS Interface to DB2 will assist by auto calculating a READBUFF and an INSERTBUFF, as discussed in Number 2, calculating a more optimal insert buffer

based on the 32K internal DB2 I/O buffer and the row size that is being loaded, allowing for efficient DATA / SET loads, though not as effective as the LOAD utility.

```
5     data mydb2.sasSP;
6     set sasdata.sasSP;
7     run;

NOTE: SAS variable labels, formats, and lengths are not written to DBMS tables.
NOTE: There were 10116050 observations read from the data set SASDATA.SASSP.
NOTE: The data set MYDB2.SASSP has 10116050 observations and 8 variables.
NOTE: DATA statement used (Total process time):
      real time           10:38.68
      cpu time            17.10 seconds
```

Output 2. Output from SAS/ACCESS Interface to DB2 Using a Standard DATA / SET Assignment

Second, Output 3 represents using a SAS/ACCESS Interface to DB2 load with BULKLOAD=YES. In this case, only the IMPORT load utility is called, and it is basically doing a simple DB2 insert under the covers. These inserts with IMPORT are not optimized as with DATA / SET or CLI LOAD. Thus, they will more than likely incur performance degradation as with our example.

```
20   data mydb2.sasSP_BULKLOAD (BULKLOAD=yes);
21   set sasdata.sasSP;
22   run;

NOTE: SAS variable labels, formats, and lengths are not written to DBMS tables.
NOTE: There were 10116050 observations read from the data set SASDATA.SASSP.
NOTE: The data set MYDB2.SASSP_BULKLOAD has 10116050 observations and 8 variables.
NOTE:
 Using data file: BL_SASSP_BULKLOAD_0.ixf
 Using log file:  BL_SASSP_BULKLOAD_0.log
 Results for bulk load operation:
 10116050 rows loaded.
 0 rows skipped.
 0 rows rejected.
 0 rows deleted as duplicates.
 10116050 rows committed.

NOTE: DATA statement used (Total process time):
      real time           1:12:47.10
      cpu time            52.88 seconds
```

Output 3. Output from SAS/ACCESS Interface to DB2 Using Only BULKLOAD=YES with DB2

Finally, Output 4 shows the CLI LOAD method as a **66% performance gain** over DATA / SET. Using CLI LOAD is always the recommended path. The only reason for using IMPORT is that you only need the insert privilege to use it as a loader, and some could argue that that is less restrictive. On the contrary, the LOAD privilege is not pervasive, and I would argue that it is a relatively safe privilege to grant to users.

```
24   data mydb2.sasSP (BULKLOAD=yes BL_METHOD=CLILOAD);
25   set sasdata.sasSP;
26   run;

NOTE: SAS variable labels, formats, and lengths are not written to DBMS tables.
NOTE: There were 10116050 observations read from the data set SASDATA.SASSP.
NOTE: The data set MYDB2.SASSP has 10116050 observations and 8 variables.
NOTE:
```

```
 Results for CLI LOAD operation:
 10116050 rows loaded.
 0 rows skipped.
 0 rows rejected.
 0 rows deleted as duplicates.
 10116050 rows committed.

NOTE: DATA statement used (Total process time):
      real time           3:37.76
      cpu time            8.42 seconds
```

Output 4. Output from SAS/ACCESS Interface to DB2 Using BULKLOAD with CLILOAD

NUMBER 2: READBUFF AND INSERTBUFF

What if you do not have access to these load utilities? Or what if (oops!) you installed only the Runtime Client! Diverting slightly from BULKLOAD, but still well within the SAS/ACCESS world, you can use READBUFF and INSERTBUFF to give standard DATA / SET operations a little more zip. The default values are limiting, with the default for READBUFF being 250 and the default for INSERTBUFF being 1. By changing those values, you are effectively changing the size of the "data packets" that you are moving around. Note that READBUFF is not used when reading from SAS data sets; I just included it in case you switch to a database source for reading in data. Let us look at an example using an Oracle libref, starting with a standard DATA / SET in Output 5.

```
2    libname mysas '/users/sasdbl/';
NOTE: Libref MYSAS was successfully assigned as follows:
      Engine:        V9
      Physical Name: /users/sasdbl
3     libname myora oracle user=stock password=XXXXXX path=tktsora;
NOTE: Libref MYORA was successfully assigned as follows:
      Engine:        ORACLE
      Physical Name: tktsora
4    data myora.sasSP; set sasdata.sasSp;
5    run;
      real time           17:23.27
      cpu time            11.00 seconds
```

Output 5. Output from a Standard DATA/SET with an Oracle Libref

For comparison sake, let us also capture a timing for SAS/ACCESS Interface to Oracle using BULKLOAD=YES in Output 6.

```
9    data myora.sasSP_BULKLOAD (BULKLOAD=yes);
10   set sasdata.sasSP;
11   run;
      real time           3:12.01
      cpu time            16.80 seconds
```

Output 6. Output from a BULKLOAD=YES with an Oracle Libref

Now, how does it look if you increase READBUFF and INSERTBUFF significantly from the default? Output 7 shows an **80% gain** over the standard DATA / SET and is just **2%** shy of the BULKLOAD command.

```
17   libname myora oracle user=stock password=XXXXXX
18   path=tktsora
19   INSERTBUFF=100000 ReadbufF=100000;
20   data myora.sassp_ir;
21   set mysas.sasSP500;
22   run;
```

```
NOTE: SAS variable labels, formats, and lengths are not written to DBMS tables.
NOTE: There were 10116050 observations read from the data set MYSAS.SASSP500.
NOTE: The data set MYORA.SASSP_IR has 10116050 observations and 8 variables.
NOTE: DATA statement used (Total process time):
      real time           3:33.19
      cpu time            8.03 seconds
```

Output 7. Output from READBUFF and INSERTBUFF Showing an Increase from the Default with an Oracle Libref

NUMBER 3: BL_RECOVERABLE

Continuing to build from the SAS/ACCESS Interface to Oracle example from above, it is also important to understand the default options of the load utility you are using. To illustrate, the load utility for Oracle is SQLLDR, which defaults to a mode that turns on database-level logging so that a failed load can be recovered and rolled back. Depending on how the database is set up, this can have a varying degree of impact. Turning off logging will speed up the load, as it did for the test in Output 8, increasing the BULKLOAD percentage to **85%**, and with large loads that 3% difference will continue to grow.

```
26   data myora.sasSP_BL (bulkload=yes BL_RECOVERABLE=NO);
27   set sasdata.sasSP500;
28   run;
NOTE: DATA statement used (Total process time):
      real time           2:51.34
      cpu time            13.54 seconds
```

Output 8. Output from SAS/ACCESS Interface to Oracle with BULKLOAD=YES and BL_RECOVERABLE=NO

NUMBER 4: TERADATA AND TERADATA PARALLEL TRANSPORT (TPT)

With SAS/ACCESS Interface to Teradata version 9.2 and up, the default BULKLOAD in SAS changed to use TPT first. TPT is the Teradata Parallel Transport utility, which is highly customizable and handles large loads well. It is worth mentioning, because if you do not have TPT installed, BULKLOAD will fall back to the older (read as slower) MULTILOAD, and this is not completely apparent unless you are looking at a sastrace. Thus, make sure you have it installed! Also, using TPT as your bulk loader within SAS enables you to restart a load from a checkpoint. SAS has also included several TPT options using BL_OPTIONS, some of which are listed below, to further customize loading data with TPT and to make it easier.

TPT BL_OPTIONS

- SLEEP=
- TENACITY=
- TPT=
- TPT_CHECKPOINT_DATA=
- TPT_DATA_ENCRYPTION=
- TPT_LOG_TABLE=
- TPT_MAX_SESSIONS=
- TPT_MIN_SESSIONS=
- TPT_RESTART=
- TPT_TRACE_LEVEL=
- TPT_TRACE_LEVEL_INF=

- TPT_TRACE_OUTPUT=

Since the Teradata loaders do not allow duplicate rows, I did tests with only a million rows. Output 9 shows a standard DATA / SET. However, Output 10 shows a vast performance improvement using TPT of **99%**.

```
42   data mytera.littlesasSP;
43   set sasdata.teradataSP;
44   run;
      real time           18:01.48
      cpu time            9.83 seconds
```

Output 9. Output from SAS/ACCESS Interface to Teradata Using DATA / SET

```
45   data mytera.littlesasSP_BL (bulkload=yes);
46   set sasdata.teradataSP;
47   run;
      real time           7.04 seconds
      cpu time            2.65 seconds
```

Output 10. Output from SAS/ACCESS Interface to Teradata Using BULKLOAD=YES and TPT

NUMBER 5: PIPES

Using PIPES enables you to save the step and time of having to first land a file, and then load it, which is the basic procedure of most of the SAS/ACCESS products when it comes to BULKLOAD=YES. Thus, PIPES can save you a lot of disk space that gets used only momentarily, and it can save you a lot of time. Currently, the documented databases that do this are Neoview, Netezza, Oracle, and Sybase IQ. All of these use the option BL_USE_PIPES=YES. In SAS 9.3, SAS/ACCESS for Aster nCluster will also use PIPES. This is handy when dealing with the large (as in huge) data volumes that most of the appliance databases like nCluster handle. Specifically with nCluster, the database replicates data based on how the nCluster server is set up. When loading data with DATA / SET, the database has to do all the replication on the node workers similar to logging with other databases, and it is basically putting multiple copies of the same data for recoverability. SAS will wait for a response from the replication during a standard DATA / SET operation. Thus, the load times using the DATA / SET method can become intolerable. For that reason, I skipped the DATA / SET test and went straight to BULKLOAD=YES.

Loading via PIPES (see Output 12) is **36%** faster than BULKLOAD (see Output 11). That percentage of time will continue to grow over larger volumes, and you do not have to worry about the hardware resources as much.

```
14   data myaster.sassp500 (bulkload=yes);
15   set sasdata.sassp500;
16   run;
      real time           1:22.82
      cpu time            31.94 seconds
```

Output 11. Output from a Load Using SAS/ACCESS Interface to Aster nCluster and BULKLOAD=YES

```
17   data myaster.sassp555 (bulkload=yes BL_USE_PIPE=YES);
18   set sasdata.sassp500;
19   run;
      real time           53.72 seconds
      cpu time            39.49 seconds
```

Output 12. Output from a Load Using SAS/ACCESS Interface to Aster nCluster and BL_USE_PIPE=YES

CONCLUSION

I would like to close with a summary table, Table 1 below, of all five methods discussed in this paper. Echoing the subtext from above, knowledge of the settings and defaults of the database's load utilities can only help in eking out a few more valuable seconds, which turn into minutes, which turn into ….

Method	DATA / SET	BULKLOAD=YES	Test Scenario
#1 CLI LOAD	10:38.68	1:12:47.10 (n/a)	3:37.76 (66%)
#2 READBUFF/INSERTBUFF	17:32.27	3:12.01 (82%)	3:33.19 (80%)
#3 BL_RECOVERABLE	17:32.27	3:12.01 (82%)	2:49.34 (85%)
#4 TPT (only 1 million rows)	18:01.48	7.08 (99%)	n/a
#5 PIPES	n/a	1:22.83	53.72 (36%)

Table 1. Summary of Timings with Percent Improvement from DATA / SET Load

ACKNOWLEDGMENTS

- I would like to thank my wife, Brennan, for her continual support.
- David Shamlin.
- SAS/ACCESS development team members: Keith Handlon, Pravin Boniface, and Chris Dehart.

RECOMMENDED READING

- *SAS/ACCESS 9.2 for Relational Databases: Reference*, Fourth Edition.
- The vendor documentation.

CONTACT INFORMATION

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

Douglas B. Liming
SAS Institute Inc.
100 SAS Campus Dr.
Cary, NC 27513
Douglas.Liming@sas.com
http://www.sas.com