Multi-Threaded Reads in SAS/Access for Relational Databases
Sarah Whittier, ISO New England, Holyoke, MA

ABSTRACT
Multi-threading was implemented in SAS 9®. This new feature affects the performance of certain SAS procedures and SAS/ACCESS for Relational Databases. However, it can be challenging to figure out how best to use multi-threading in order to improve the performance of relational databases queries. Contributing to this confusion are the several parameters that affect how threading works. This paper has three objectives. First, we define and discuss multi-threading in general. Second, we show how to use the DBSLICEPARM option and present significant performance improvements at our site that resulted from simply using the default DBSLICEPARM option. Third, we present results of our testing of the impact of multi-threaded reads on our Oracle® databases.

INTRODUCTION
At our site, we have several applications written in SAS that read large amounts of data from Oracle databases. We found that one simple option, DBSLICEPARM, greatly improved the speed of reading data from Oracle, without causing problems for our Oracle database servers. But we almost missed it: when we were preparing to implement SAS 9 at our site, we weren’t sure how to make threaded reads work, and we thought multi-threading would probably overload our Oracle systems. After we figured out which options to use for threaded reads to Oracle, SAS users worked with our Oracle database administrators (DBAs) to test the impact on the Oracle databases. The goal of this paper is to explain how to use multi-threading and share our test results with other SAS users who may hesitate to use this option because they assume that it is complicated to implement or will cause problems.

OVERVIEW OF MULTI-THREADING AND THE TEST ENVIRONMENT
Without multi-threading, a task is processed sequentially. In general, when working with large amounts of data, multi-threaded processing is faster than sequential processing because a task is split into partitions that are processed in parallel. Multi-threading was implemented in SAS 9 in some Base SAS Procedures and in SAS/ACCESS for Relational Databases.

MULTI-THREADING IN BASE SAS PROCEDURES: A BRIEF OVERVIEW
Along with SAS/ACCESS, some procedures in Base SAS, including PROC SQL, PROC SORT, PROC SUMMARY, PROC MEANS, and PROC TABULATE are multi-threaded. This means that on a machine with multiple processors, SAS can use parallel processing and the procedure will run more quickly. According to the SAS documentation, the “THREADS|NOTHREADS option enables or prevents the activation of multi-threaded processing” for the PROCs listed above. The default for this option is THREADS. This paper focuses on threaded reads in SAS/Access, and does not cover multi-threaded procedures in Base SAS.

MULTI-THREADING IN SAS/ACCESS FOR RELATIONAL DATABASES: A BRIEF OVERVIEW
Unlike threading in Base SAS procedures, threaded reads in SAS/ACCESS are not dependant on the number of processors on a machine and are not controlled by the THREADS option. The options to control threaded reads in SAS/ACCESS are DBSLICE and DBSLICEPARM. The default for the option DBSLICEPARM is DBSLICEPARM=(THREADED_APPS, 2). You can see this default by running PROC OPTIONS. However, you also need to put the DBSLICEPARM option on your LIBNAME statement or data step statement. The DBSLICE option allows the programmer to specify how to break up data for threads. These two options work only when reading data via the LIBNAME access engine: they do not work with the SQL Pass-Through Facility.

HOW TO USE THE DBSLICEPARM OPTION
The simplest way to enable multi-threaded reads from an Oracle database is to put the DBSLICEPARM=ALL option on the LIBNAME statement.

In this example, any data steps following the libname and reading data from the “dat” library would use the default number of threads (unless you have changed your system options, the default will be two):

```sas
libname dat oracle schema=dat_owner user=testuser pw=XXXXXXXX path=datdbp
dbsliceparm=ALL;
```
In the next example, all data steps reading data from the “dat” library would use four threads:

```
libname dat oracle schema=dat_owner user=testuser pw=XXXXXXXX path=datdbp
dbsliceparm=(ALL,4);
```

You can also use the DBSLICEPARM option on only selected data steps, rather than on the LIBNAME statement, as this example shows:

```
libname dat oracle schema=dat_owner user=testuser pw=XXXXXXXX path=datdbp;

data lll;
  set dat.disc_lll_v
       (dbsliceparm=ALL
        keep=day
        hour
        price
        most_recent
       where=(most_recent='Yes' and
             day between '01JAN2006:00:00:00'dt and
             30MAY2006:00:00:00'dt ));
run;
```

During testing, it is useful to include OPTIONS SASTRACE=",,T" SASTRACELOC=SASLOG NOSTSUFFIX in your program. These options will output information about threaded reads to the SAS log. Once testing is complete, it is not necessary to leave these options in your program.

**ABOUT OUR ENVIRONMENT AND SAS/ACCESS FOR ORACLE EXPERIENCE**

At our site, SAS/ACCESS for Oracle is installed on Linux servers (development and production environments) which SAS users access via SAS/CONNECT from their PCs. The company has several Oracle databases that are used for analysis. Tables in these databases are not partitioned. Users throughout the company have read access to these databases, and in addition to SAS, tools including Oracle Discoverer and PL/SQL are used to query the databases.

In the past, users running large data queries using SAS or other tools have caused serious performance issues with the Oracle databases. Some Oracle DBAs and other IT administrators were resistant to SAS use, believing SAS enabled users to run large queries that would cause chronic performance problems with the Oracle databases. While users obviously can cause problems with queries that pull very large amounts of data from Oracle, they can do this with PL/SQL just as easily as with SAS. We have worked with users to improve SAS jobs that read data from Oracle by using appropriate WHERE and KEEP data set options.

Given our prior problems with SAS and Oracle, we were cautious about implementing multi-threaded capability. We undertook testing to determine the performance gain that using threaded reads would provide and the impact on the Oracle side. One goal of testing was to create situations DBAs would object to (e.g., using threaded reads when a database already had a heavy load) and find out what happened to database performance.

**DESCRIPTION OF TESTING AND RESULTS**

We tested several scenarios: a base case using the LIBNAME access engine and no multi-threading, cases using two or more threads, and a case with two threads run simultaneously with other jobs that produced a heavy load on the server. We also benchmarked queries using the SQL Pass-Through Facility against threaded reads.
We ran our tests in a development environment, using SAS/ACCESS for Oracle on our development server and a test version of an Oracle database. We selected an Oracle view with about 85 million observations and 15 variables to use in many of our tests; we used a WHERE clause that was met by about 40% of observations (about 34 million). This view was selected because it was large enough for performance differences to be significant and because it is one we use frequently in our work. We also tested against other Oracle tables and views; however, those results are not included in this paper.

We set OPTIONS SASTRACE=",,T" SASTRACELOC=SASLOG NOSTSUFFIX to help us track SAS performance. An Oracle DBA monitored the impact on the database of all these cases except for those using the SQL Pass-Through Facility. When testing was complete, we evaluated SAS logs and server impacts. Representative results, including partial SAS logs and comments from the DBA, are included below.

**TEST CASE: NO THREADING (BASE CASE)**

In this case, we submitted our job without using the DBSLICEPARM option on the Oracle LIBNAME statement. The job took almost 24 minutes (real time) to read the data from Oracle.

The DBA reported, “this was one of the top Oracle processes but didn’t cause the system load to go up. In fact the CPU was idle about 80% of the time and the load averages were less than 2 consistently, which is light load. I didn’t notice anything unusual during this test.”

**SELECTED RESULTS FROM THE SAS LOG:**

```sas
1 libname dev oracle schema=disc_owner user=testuser pw=XXXXXXXX path=devmisi;
2 dbsliceparm=NONE;
3 options fullstimer;
4 options sastrace=",,t" sastraceloc=saslog nostsuffix;
5
6 *----- Scenario A: one user, no multithreading to Oracle, pre-Oracle 9i drivers -----*/
7 data lll;
8 set dev.disc_lll_v(where=(most_recent='Yes'));
9 run;

NOTE: There were 33319540 observations read from the data set SMD.DISCLLL_V.
WHERE most_recent='Yes';
NOTE: The data set WORK.LLL has 33319540 observations and 15 variables.
NOTE: DATA statement used (Total process time):
   real time 23:45.92
   user cpu time 6:20.64
   system cpu time 34.27 seconds
   Memory 279k
   Page Faults 627
   Page Reclaims 47716
```

**TEST CASE: WITH THREADING (DEFAULT NUMBER OF TWO THREADS)**

In this case, we submitted our job using the option DBSLICEPARM=ALL on the Oracle LIBNAME statement. The job took about eight minutes (real time) to read the data from Oracle.

---

1 During testing, the database was refreshed, resulting in slightly different numbers of observations among test cases. There were a few other processes running on the database during testing, and the number and load of these processes varied. However, the other processes never put a high load on the database.
The DBA reported, “this time there were three Oracle database connections and two were active. The load average was between 2 and 3 and it was already like that even before the testing started. No significant difference in the load on the system. Average CPU idle time was around 60%.”

**SELECTED RESULTS FROM THE SAS LOG:**

```plaintext
1   libname dev oracle schema=disc_owner user=testuser pw=XXXXXXX path=devmisi
2   dbsliceparm=ALL;
NOTE: Libref DEV was successfully assigned as follows:
   Engine: ORACLE
   Physical Name: devmisi
3   options fullstimer;
4   options sastrace=",,t" sastraceloc=saslog nostsuffix;
5
6   /*----- Scenario A: one user, multithreading to Oracle, pre-Oracle 9i drivers ------*/
7   data lll;
8   set dev.disc_lll_v(where=(most_recent="Yes"));
9   run;
ORACLE: No application input on number of threads.
ORACLE: Thread 1 contains 17169602 obs.
ORACLE: Thread 2 contains 17176850 obs.
ORACLE: Threaded read enabled. Number of threads created: 2
NOTE: There were 34346452 observations read from the data set SMD.DISC_LLL_V.
WHERE most_recent='Yes';
NOTE: The data set WORK.LLL has 34346452 observations and 15 variables.
NOTE: DATA statement used (Total process time):
   real time 8:25.24
   user cpu time 27.51 seconds
   system cpu time 30.12 seconds
Memory 1051k
Page Faults 410
Page Reclaims 286
```

**TEST CASE: MULTIPLE JOBS SUBMITTED SIMULTANEOUSLY**

The goal of this test was to create a heavy load on Oracle by submitting multiple large jobs, and then submit our test job using threaded reads to determine if this would cause a problem. Under these heavier load conditions, our test job with two threads took almost 13 minutes (real time) to complete.

The DBA reported, "I saw 15 processes running at one time. The load average jumped to a high of 5.3. This is not alarming but if other CPU intensive processes have to run at the same time then there may be some delay. On an average CPU was still idle for about 35% of the time."

**TEST CASE: SQL PASS-THROUGH FACILITY VS. TWO-THREADED READ**

We completed the testing of the impact on Oracle described in this paper in 2005. Prior to using threaded reads, we had found SQL pass-through to be faster than the LIBNAME access engine in most cases. However, we typically used the LIBNAME access engine anyway, because we found it simpler to code. We wanted to determine whether SQL pass-through was faster than threaded reads.

Benchmarking, which was performed against a different Oracle view than the test cases included above, showed the LIBNAME access engine with two threads was faster than the SQL pass-through. In the logs shown below, SQL pass-through took about five minutes and 50 seconds compared to about three minutes for the two-threaded
read. SQL pass-through was faster than the LIBNAME engine with no threads (log not shown), which took just over seven minutes.\(^2\)

SELECTED RESULTS FROM THE SAS LOG FROM THREADED READ:

```sas
libname dat oracle schema=dat_owner user=testuser pw=XXXXXXXX path=datdbp
libname dat oracle schema=dat_owner user=testuser pw=XXXXXXXX path=datdbp
NOTE: Libref DAT was successfully assigned as follows:
  Engine: ORACLE
  Physical Name: datdbp

data lmp;
set dat.dat_lmp_v
(keep=day hour price congestion_component most_recent
where=(most_recent='Yes' and
day between '01JAN2005:00:00:00'dt AND '30MAY2006:00:00:00'dt ));
run;
```

ORACLE: No application input on number of threads.
ORACLE: Thread 1 contains 11318501 obs.
ORACLE: Thread 2 contains 11318491 obs.

Thread 1 fetched 11318501 rows
DBMS Threaded Read Total Time: 179262 mS
DBMS Threaded Read User CPU: 0 mS
DBMS Threaded Read System CPU: 0 mS

Thread 2 fetched 11318491 rows
DBMS Threaded Read Total Time: 179266 mS
DBMS Threaded Read User CPU: 0 mS
DBMS Threaded Read System CPU: 0 mS

ORACLE: Threaded read enabled. Number of threads created: 2
NOTE: There were 22636992 observations read from the data set SMD.DISC_LMP_V.
WHERE (most_recent='Yes') and (day>=' 01JAN2005:00:00:00'DT and day<='30MAY2006:00:00:00'DT);

Summary Statistics for ORACLE are:
Total SQL prepare seconds were: 0.002040
Total seconds used by the ORACLE ACCESS engine were 180.362837

NOTE: The data set WORK.LMP has 22636992 observations and 5 variables.
NOTE: DATA statement used (Total process time):
real time 3:00.44
cpu time 21.11 seconds
```

SELECTED RESULTS FROM THE SAS LOG FROM SQL PASS-THROUGH:

```sas
proc sql;
connect to oracle
  (user=&username
  pw=&password
  path='datdbp'
  connection=global);
```

\(^2\) In the time between our initial testing of multi-threading and the comparison of SQL pass-through and two-threaded read, our Linux server was upgraded.
CONCLUSIONS FROM TESTING AND REAL WORLD EXPERIENCES

Our testing demonstrated to our satisfaction that using the DBSLICEPARM option on the LIBNAME statement provided performance gains with an acceptable impact on the Oracle databases. We left DBSLICEPARM=(THREADED_APPS, 2) as the default option, and did not place any restrictions on analysts’ use of threaded reads in their programs.

Since completing testing, we have added the DBSLICEPARM=ALL option to Oracle LIBNAME statements in many of our applications, and also use it in ad-hoc projects. We have not encountered any problems or adverse impacts on the Oracle database servers. As we develop new applications and rework existing ones that read large data sets, we will continue to benchmark the relative performance of different methods to determine which is best for each application.

We encourage readers of this paper to familiarize themselves with how threading works and to do their own testing. However, we also encourage you not to be overwhelmed by the information. Go ahead and try DBSLICEPARM=ALL on your Oracle LIBNAME statements.

REFERENCES


ACKNOWLEDGMENTS

I would like to thank Panayiotis Manolakos and Suresh Simhadri for their work on the project on which this paper is based, and David Gionet and Craig Kazin for their editorial review.

SAS is a registered trademark of the SAS Institute, Inc. of Cary, North Carolina.
Oracle is a registered trademark of the Oracle Corporation.

CONTACT INFORMATION
Contact the author at:

Sarah Whittier
ISO New England
One Sullivan Rd
Holyoke, MA 01040
(413) 540-4282
swhittier@iso-ne.com