LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-CONF-491537

Oracle Database DBFS Hierarchical Storage Overview

A. Rivenes

July 29, 2011

IAEA 8th Technical Meeting
San Francisco, CA, United States
June 20, 2011 through June 24, 2011

Disclaimer

ORACLE DATABASE DBFS HIERARCHICAL STORAGE OVERVIEW

Andy Rivenes, Lawrence Livermore National Laboratory, Livermore, CA 94551, U.S.A.

Abstract

The National Ignition Facility (NIF) at the Lawrence Livermore National Laboratory creates large numbers of images during each shot cycle for the analysis of optics, target inspection and target diagnostics. These images must be readily accessible once they are created and available for the 30 year lifetime of the facility. The Livermore Computing Center (LC) runs a High Performance Storage System (HPSS) that is capable of storing NIF's estimated 1 petabyte of diagnostic images at a fraction of what it would cost NIF to operate its own automated tape library.

With Oracle 11g Release 2 database, it is now possible to create an application transparent, hierarchical storage system using the LC's HPSS. Using the Oracle DBMS_LOB and DBMS_DBFS_HS packages a SecureFile LOB can now be archived to storage outside of the database and accessed seamlessly through a DBFS "link". NIF has chosen to use this technology to implement a hierarchical store for its image based SecureFile LOBs. Using a modified external store and DBFS links, files are written to and read from a disk "staging area" using Oracle's backup utility. Database external procedure calls invoke OS based scripts to manage a staging area and the transfer of the backup files between the staging area and the Lab's HPSS.

INTRODUCTION

With Oracle Database 11g Release 2 Oracle has enabled seamless hierarchical storage for SecureFile LOBs. Using the DBMS_LOB and DBMS_DBFS_HS packages a SecureFile LOB can now be archived to storage outside of the database and accessed seamlessly through a DBFS "link". Of course this requires that the database “knows” where the SecureFile LOB is and how to retrieve it. Currently Oracle supports two types of external storage, or "stores", directly and provides some limited information on how to define custom stores as well. Oracle provides external storage support for tape using its backup utility Recovery Manager (RMAN) and also Amazon S3 (http://aws.amazon.com/s3/?utm_source=eclipse&utm_medium=lp&utm_campaign=galileo).

NIF has chosen to implement a DBFS hierarchical store for its SecureFile LOBs using a modified tape based external store and DBFS links. Tape files are created by RMAN but are written to disk rather than tape. External OS based scripts then manage the disk "staging area" and the transfer of the RMAN created files between the staging area and the Lab's HPSS. The Livermore Computing Center runs a High Performance Storage System (HPSS) that is capable of storing NIF's estimated 1 petabyte of diagnostic images at a fraction of what it would cost NIF to operate and maintain an automated tape library capable of storing a petabyte of data.

NIF DATA ARCHIVE

The first version of the NIF Data Archive was designed before the DBFS HS option was available and it required an "application specific" API in order to access the data archive along with supporting code to handle SecureFile LOB archiving and retrieval. In the 11gR2 DBFS HS implementation the concept of "links" has been created. A DBFS link provides the connection between the SecureFile LOB's original location and its archived location. The database handles the retrieval of the SecureFile LOB transparently for the invoking transaction. This means that only the initial archival of the target SecureFile LOBs and the movement of archive files between the database server and the HPSS need to be managed. See Figure 1 for a diagram of the new architecture.

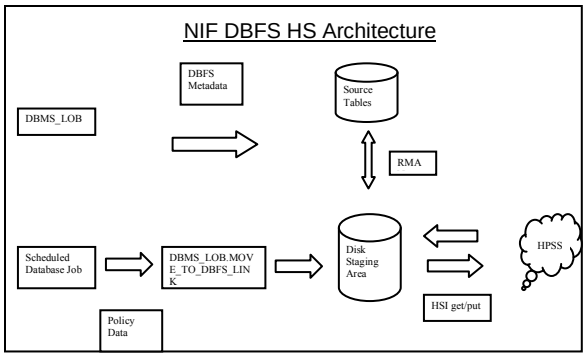

Figure 1. NIF Data Archive

STORE ARCHIVE

When a DBFS HS store is created, several activities occur in the background. As the store is defined, a cache table is created in the store owner's schema. The DBMS_DBFS_HS.CreateStore procedure accepts a store name, a cache table name, a tablespace name for the cache table, a cache table size and a threshold value to initiate a push to

external storage. Also as part of the store creation a metadata table is created in the store owner's schema. The metadata table is used by Oracle to track the properties of the archived files. The cache table is used both to buffer files before creating an archive file and as a container to make "linked" SecureFile LOBs available for queries. Note that as long as the link exists the file is not returned to its original location.

NIF has defined the DBFS HS external storage store type as tape. This implies that RMAN will be used to read and write files to tape. In order to make use of HPSS, rather than maintaining a local tape library system, some modifications to the DBFS system have been made. The first change was to use the "oracle.disksbt" option to the SBT_LIBRARY parameter to create "tape files" on disk. With this change when a store push occurs Oracle will invoke RMAN to write tape files to a defined disk directory. It is then a relatively trivial process to copy files between this disk directory and the HPSS. The second change made, after consulting with some of the Oracle development team, was to create a "shadow" store using an intermediary package to intercept all DBFS calls. This allows NIF custom code to manage the disk "staging" area and put or get files from the HPSS based on space and access requirements. Two examples of needing to be able to intercept DBFS calls are: 1) when creating a new archive file there is a need to ensure that there is enough space in the staging directory and 2) when retrieving a SecureFile LOB from a file that only exists in the HPSS there is a need to ensure that there is space in the staging directory before making an HSI get call. In both of these examples we also need to lock the archive file and the staging directory, so that our file or reserved space isn't changed by another process.

ARCHIVAL

A SecureFile LOB is "archived" with the DBMS_LOB procedure MOVE_TO_DBFS_LINK. Running this procedure requires that a LOB locator and storage path be defined. The LOB locator can be defined by simply selecting the SecureFile LOB into a BLOB variable. However, in order to move the SecureFile LOB to an archive store the SecureFile LOB must also be locked. This can be accomplished with a SELECT … FOR UPDATE statement. The storage path has already been defined as part of the store creation (specifically when the store is mounted with the DBMS_DBFS_CONTENT.MOUNTSTORE procedure) and so is already known. NIF is using a "URN" to uniquely identify individual SecureFile LOBs as defined by the Oracle CM SDK application framework.

MOVE_TO_DBFS_LINK

```
PROCEDURE MOVE_TO_DBFS_LINK
Argument Name  Type           In/Out Default?
-------------- -------------   ------ --------
LOB_LOC        BLOB            IN/OUT
STORAGE_PATH   VARCHAR2        IN
FLAGS          BINARY_INTEGER IN     DEFAULT

LOB_LOC         LOB to be archived
STORAGE_PATH    The store path created with
                the "MOUNTSTORE" procedure
                and a unique identifier.
FLAGS           Either DBFS_LINK_CACHE in
                which the LOB is not removed,
                or DBFS_LINK_NOCACHE in which
                the LOB is removed once it
                has been written to the
                archive (not available to
                DBMS_LOB though).
```

When the link is created the SecureFile LOB is moved to the "cache" table defined for the store. A token is created as a placeholder in the original SecureFile LOB's location. The cache table is used to accumulate SecureFile LOBs up to a specified number of bytes, and then a store "push" is invoked to add the SecureFile LOBs to a "tarball" and write an output file using the RMAN interface. The cache table is also used when SecureFile LOBs are retrieved from their respective output file.

Archive Example

The following shows a simple PL/SQL procedure that will archive a single SecureFile LOB:

```
SQL> SET SERVEROUTPUT ON;
DECLARE
  p_whereval    NUMBER := 1776056;
  --
  v_mountpoint  VARCHAR2(32):='/nif_store/';
  v_store       VARCHAR2(32) := 'NIF_STORE';
  v_LOB         BLOB;
  v_LOBsize     NUMBER;
  l_urn         cms_urn_map.urn%TYPE;
BEGIN
  SELECT urn
  INTO l_urn
  FROM cms_urn_map
  WHERE tablename = 'ODMM_NONINDEXEDSTORE'
  AND columnname = 'NONINDEXEDBLOB2'
  AND content = p_whereval;
  --
  SELECT nonindexedblob2
  INTO v_LOB
  FROM odmm_nonindexedstore
  WHERE id = p_whereval FOR UPDATE;
  --
  v_LOBsize :=
    SYS.DBMS_LOB.GETLENGTH(v_LOB);
  DBMS_OUTPUT.PUT_LINE('URN: ' || l_urn);
  DBMS_OUTPUT.PUT_LINE('LOB size: ' ||
    TO_CHAR(v_LOBsize));
  --
```

```
  DBMS_LOB.MOVE_TO_DBFS_LINK (
    lob_loc => v_LOB,
    storage_path => v_mountpoint || l_urn,
    flags => DBMS_LOB.DBFS_LINK_CACHE
  );
  --
  COMMIT;
END;
/
SQL>   2    3    4    5    6    7    8    9
10   11   12   13   14   15   16   17   18
19   20   21   22   23   24   25   26   27
28   29   30   31   32   33   34  URN:
urn:llnl.gov:nif:archive:67339dfa-206c-4ca9-
a503-7d9b7b2caa93
LOB size: 13041713

PL/SQL procedure successfully completed.

SQL>
```

RETRIEVAL

Using DBFS links for archiving allows retrieval to be completely transparent to the application when using the DBMS_LOB package.

Retrieval Example

The following query shows that the SecureFile LOB for this URN has been archived and is not in the store's cache. The store cache has been named "nif_store" and this table identifies the "content" name, size, internal id and the store name to which it belongs.

```
SQL> select urn, cache_size, contentid,
storename from nif_store_v
  2  where urn =
'urn:llnl.gov:nif:archive:19c95706-cf46-
4c8d-b9b2-600988f375cd';

URN
----------------------------------------
----------------------------------
CACHE_SIZE
----------
CONTENTID
----------------------------------------
----------------------------------
STORENAME
----------------------------------------
----------------------------------
urn:llnl.gov:nif:archive:19c95706-cf46-4c8d-
b9b2-600988f375cd
         0
NIF_STORE_ILMDS_96C2FB6E83821E09E0407380058E
5C84
NIF_STORE


SQL>
```

The cache size is zero signifying that it is not in the database. Now we'll attempt to read the SecureFile LOB using the DBMS_LOB package.

```
SQL> SET SERVEROUTPUT ON;
DECLARE
  l_LOB                BLOB;
  l_buffer             RAW(4000);
  l_ctr                NUMBER := 1;
  l_amount             NUMBER := 100;
  l_id                 NUMBER := 2857207;
BEGIN
  SELECT nonindexedblob2
  INTO l_LOB
  FROM odmm_nonindexedstore
  WHERE id = l_id;
  --
  dbms_lob.read(l_LOB, l_amount, l_ctr,
    l_buffer);
  dbms_output.put_line( RAWTOHEX(l_buffer)
);
END;
/
SQL>   2    3    4    5    6    7    8    9
10   11   12   13   14   15   16
FFD8FFE000104A4649460001020000010001000FFDB
0043000806060706050807070709090080A0C
140D0C0B0B0C1912130F141D1A1F1E1D1A1C1C20242E
2720222C231C1C2837292C30313434341F27
393D38323C2E333432FFDB0043010909090C0B0C

PL/SQL procedure successfully completed.

SQL>
```

This code reads the first 100 bytes of the SecureFile LOB and the output shows that were able to read the LOB without having to do anything else to retrieve it.

NIF DBFS MODIFICATIONS

Shadow Store

The shadow store is implemented as a PL/SQL wrapper package that is defined at store creation. In the wrapper package are generic calls to each of the DBMS_DBFS_HS procedures and functions. For the functions that involve accessing the archive file, custom code is inserted to insure that there is either space in the staging directory or that the archive file has already been recalled from the HPSS. We've called this custom package NIF_DBFS_HS. A second package has been created to define the code that manipulates the staging directory by calling external procedures and the HSI interface to the HPSS along with several utility packages for locking, logging and instrumentation.

HPSS Code

The HPSS support has required the creation of several different components. A staging directory has been created to allow the database to write and read archive files which are made up of one or more SecureFile LOBs. Oracle calls these files "tarballs", which in this context are simply RMAN tape formatted files. OS scripts have been created to list files and space usage in the staging directory, and to

provide HPSS get, put and ls commands using the HPSS interface HSI. For the database to be able to initiate actions at the OS level external procedure calls have been defined. A generic Java external procedure call has been created which allows the nif_dbfs_utilities package to invoke the necessary OS scripts and to get information about the results of those scripts back into the database.

SUMMARY

Using Oracle's database hierarchical storage feature NIF has been able to leverage the HPSS system to archive diagnostic images and still make them transparently available to NIF applications. These diagnostic images are created during each NIF' shot cycle and loaded into an Oracle database. It is expected that over a petabyte of diagnostic images will be created during the 30 year life of the facility and the hierarchical storage implementation will significantly reduce the online storage footprint for images that are infrequently accessed. The HPSS provides a long term storage solution that insures that the images will be available for the 30 year life of the facility.

REFERENCES

[1] Oracle® Database SecureFiles and Large Objects Developer's Guide
11*g* Release 2 (11.2),
Part Number E10645-05

[2] Oracle® Database PL/SQL Packages and Types Reference
11*g* Release 2 (11.2),
Part Number E10577-05

[3] High Performance Storage System User Guide,
https://computing.llnl.gov/LCdocs/hpss

[4] Hierachical Storage Interface (HSI),
https://computing.llnl.gov/LCdocs/hsi/index.jsp?show=s4.1

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.