Silk

White Paper

On Premises Oracle Exadata to Silk + Azure

Oracle Exadata is a proprietary combination of (former) Sun hardware, firmware, Exadata only I/O features, Oracle Clusterware and the Oracle RDBMS. Exadata is an x86_64 system running Linux. It is little Endian running Oracle Enterprise Linux, a derivative of Red Hat Linux. The point here is that though it is a dedicated database appliance running many custom features it is at the end of the day 'just another Linux Oracle database'. This means the same techniques used to migrate 'regular Oracle RAC and non-RAC databases' – with some consideration given to its proprietary features – can be used. This paper will discuss those considerations.

One of the primary features of Exadata that Oracle Corporation touts and F.U.D.'s is Exadata Hybrid Columnar Compression (EHCC). EHCC is an Exadata extended feature that utilizes a combination of both row and columnar methods for storing data. In EHCC, tables are stored in columnar order to allow for storage optimization and deduplication. This hybrid approach achieves the performance benefits of columnar storage but may also achieve compression ratios of 10x plus. These are coupled with storage cells – dedicated Linux based I/O CPU's – to service queries on hybrid columnar compressed data run in the Exadata storage cells with "Smart Scans". These storage cells are a highly performant query engine that utilize special columnar processing techniques. Note that only the data segments – not the system, temp, undo or sysaux segments can have EHCC. In smaller databases this can be 50% or more of the storage overhead.

EHCC is not available on non-Exadata platforms with the exception of Oracle Solaris ZFS storage appliances and the SAN vendor Pillar. However, conversion from EHCC to non-HCC is possible and straight forward (as will be discussed shortly). The first step is to identify which tables are compressed with HCC and what their level of compression is. If this isn't well documented, the following query identifies if a table is currently configured with HCC:

```
SELECT DECODE(DBMS_COMPRESSION.GET_COMPRESSION_TYPE(
            ownname    => 'HR',
            tabname    => 'EMPLOYEES',
            subobjname => '',
            row_id     => 'AAAVEIAAGAAAABTAAD'),
  1,  'No Compression',
  2,  'Advanced Row Compression',
  4,  'Hybrid Columnar Compression for Query High',
  8,  'Hybrid Columnar Compression for Query Low',
  16, 'Hybrid Columnar Compression for Archive High',
  32, 'Hybrid Columnar Compression for Archive Low',
  4096, 'Basic Table Compression',
  'Unknown Compression Type') compression_type
FROM DUAL;
```

The above query identifies the type of compression -- if any -- on the table EMPLOYEES in the schema HR.

Once it is determined whether a table is compressed or not determine the degree of compression with the dbms_compression package. Several procedures within this package can be used. The first is the function GET_COMPRESSION_TYPE:

```
DBMS_COMPRESSION.GET_COMPRESSION_TYPE (
   ownname      IN    VARCHAR2,
   tabname      IN    VARCHAR2,
   row_id       IN    ROWID,
   subobjname   IN    VARCHAR2 DEFAULT NULL))
  RETURN NUMBER;
```

Next, the procedure GET_COMPRESSION_RATIO Procedure

Can be invoked to determine the compression ratio of the table:

```
DBMS_COMPRESSION.GET_COMPRESSION_RATIO (
   scratchtbsname        IN     VARCHAR2,
   ownname               IN     VARCHAR2,
   objname               IN     VARCHAR2,
   subobjname            IN     VARCHAR2,
   comptype              IN     NUMBER,
   blkcnt_cmp            OUT    PLS_INTEGER,
   blkcnt_uncmp          OUT    PLS_INTEGER,
   row_cmp               OUT    PLS_INTEGER,
   row_uncmp             OUT    PLS_INTEGER,
   cmp_ratio             OUT    NUMBER,
   comptype_str          OUT    VARCHAR2,
   subset_numrows        IN     NUMBER  DEFAULT COMP_RATIO_MINROWS,
   objtype               IN     PLS_INTEGER DEFAULT OBJTYPE_TABLE);
```

The value of cmp_ratio will gives the amount of compression that is in use. This compression ratio is key to determining how much post HCC data inflation occurs. Armed with this information the target database environment can be storage sized correctly.

One thing should be noted. In most cases HCC is not universally implemented. HCC requires additional steps to implement and is typically used for only the heaviest and largest of workload data sets. In order to use HCC the data must be loaded by one of these techniques:

• Insert statements with the APPEND hint

• Parallel DML

• Direct Path SQL*LDR

• Create Table as Select (CTAS)

Or the table may also be altered to be compressed in the following manners:

```
ALTER TABLE … COLUMN STORE COMPRESS FOR QUERY HIGH
                                 … COLUMN STORE COMPRESS FOR QUERY LOW
             … COLUMN STORE COMPRESS FOR ARCHIVE LOW
             … COLUMN STORE COMPRESS FOR ARCHIVE HIGH
```

This approach enables Hybrid Columnar Compression for all future DML -- however, the existing data in the table will remain uncompressed.

Online Redefinition (DBMS_REDEFINITION): This approach enables Hybrid Columnar Compression for future DML and compresses existing data. With DBMS_REDEFINITION the table remains online throughout the operation. It does however invalidate global indexes and an intermediate table requiring storage is also used.

ALTER TABLE … MOVE COLUMN STORE COMPRESS FOR QUERY HIGH puts an exclusive lock on the table so that it is read only during the operation. Also, all indexes will have to be rebuilt.

There is quite a bit of work to implementing HCC and typically one will find it implemented on a strategic rather than global basis. This is key to any exit Exadata migration strategy as steps need to be taken to de-implement it during the migration phase. This is not as hard as it sounds.

How can we exit Exadata?

First characterize the data and the workload. Real Application Clusters (RAC) is nearly universal on Exadata systems so identify cluster nodes and corresponding database instances. RAC allows customers to run a single Oracle Database across multiple servers to maximize availability and enable horizontal scalability. Note the load and load characteristics of the systems. Two key metrics are how much CPU is being consumed and what is the peak bandwidth and IOPS (I/O Operations per second). These two metrics will help determine the vCPU shape required in terms of vCPUs and memory. Take multiple performance snapshots of the databases with AWR, especially during peak load periods. This is the key to successfully profiling the system workload in preparation for migration.

Once a reasonable profile of the workload is collected, sizing the target system begins. This includes garnering information about the OS and RDBMS version, number of CPUs, RAM and Disk capacity and extends to profiling the workload characteristics especially regarding I/O in the AWR report's IO Stats subsection. Below is a key area of the IOSTAT section of an AWR regarding the IO broken out by database function:

IOStat by Function summary

- 'Data' columns suffixed with M,G,T,P are in multiples of 1024 other columns suffixed with K,M,G,T,P are in multiples of 1000
- ordered by (Data Read + Write) desc

Function Name	Reads: Data	Reqs per sec	Data per sec	Writes: Data	Reqs per sec	Data per sec	Waits: Count	Avg Time
Smart Scan	220.7T	104772792.78	4.8G	1.3T	7.04	28.057M	0	
Buffer Cache Reads	16.9T	4970683.57	377.827M	0M	0.00	0M	197M	687.40us
Direct Reads	13T	7055763.64	290.408M	1.3T	122.76	30.163M	0	
Others	4.2T	1578271.92	93.549M	5.6T	268.01	125.242M	12.4M	2.45ms
Direct Writes	58.7G	18396.02	1.285M	9.7T	460.61	217.14M	1468	645.78us
LGWR	88M	0.15	.002M	2.9T	199.31	65.954M	597K	128.01us
DBWR	41M	0.06	.001M	2T	883.28	45.523M	2576	206.13us
RMAN	33.6G	8738.83	.737M	284M	0.13	.006M	330.6K	471.50us
Streams AQ	6M	0.04	0M	0M	0.00	0M	167	7.95ms
Data Pump	0M	1977.15	0M	0M	0.00	0M	0	
TOTAL:	**254.7T**	**118406624.16**	**5.6G**	**22.8T**	**1941.14**	**512.085M**	**210.3M**	**789.57us**

In this section of the AWR, we see both total I/O and I/O by category. Note the total data per sec value is 5.8GB for reads and 512MB per sec for writes. These are two critical pieces of data for sizing. Also of note is the value of reads per sec for Smart Scan. This shows that Exadata storage cell technology is in use on this system. This number can be quite deceiving – and alarming. Smart Scan functions will have astronomical values for I/O that exceed even the published capabilities of the physical machine. This is because Oracle shows what it 'thinks' the I/O would look like without Smart Scan.

A better more reassuring statistic can be found in the Exadata Configuration and Statistics >Exadata Statistics > Performance Summary section of an Exadata AWR.

Performance Summary

I/O Summary

- average requests/second and MB/s per disk

	Requests		**Throughput**	
Disk Type	Reads/s	Writes/s	Read MB/s	Write MB/s
F/2.9T	2,387.69	354.44	208.09	8.29
H/7.2T	316.73	39.62	8.05	1.41

Here we see the actual physical I/O of the sub system broken down into the two tiers of physical storage Exadata HC models have: flash and spinning disk. Here the reads to both flash and disk tally to some 3.7 GB/second while the writes are far less at about 216 MB/second. This number This number neatly fits within Silk's capabilities but would strain or be unsustainable with Azure premium or ultra disk.

Exadata CPU core count is supplemented by processors in the I/O subsystem (Storage Cells) and Exadata Oracle init.ora parameters typically have a smaller SGA than non-Exadata systems. Often on Exadata, indexes are dropped or made invisible – all to force the I/O to the storage cells. This means that additional vCPUs may need to be added and that some indexing work will need to be performed to retain performance. What we do in effect is "fatten the SGA", re-index and increase the core CPU count to replace the storage cell’s I/O processing capabilities.

On the plus side a monolithic, non-RAC instance is more efficient than a RAC instance. RAC is an Oracle proprietary technology for horizontal scaling and some limited high availability but that comes at a performance cost and the scaling isn’t linear. An interconnect layer of software/hardware clusters several Oracle instances together so that they can service the same database on shared disk. This requires quite a bit of CPU and network bandwidth to coordinate the caches and transactions. The clustering overhead, global cache overhead (locking and synchronizing) is all eliminated when running a single monolithic instance. This can be a significant performance boost of up to a 10% to 20%. All factors being equal in Oracle, vertical scaling is preferred over horizontal scaling because of this cluster overhead. After this initial sizing is accomplished migration can begin.

How then do we go about migrating an Exadata workload to Azure + Silk? The answer is DataGuard. DataGuard is available in two flavors. Oracle DataGuard is an Oracle Enterprise edition entitlement while Oracle active DataGuard is an additional Oracle license pack. The version of DataGuard bundled with Enterprise edition licensing allows redo log information to be shipped from one Oracle instance to another copy of it called a ‘standby instance’ where the redo log is then ‘played out’ on the standby. The standby is a copy of the source in a mounted but unopened state in a state of constant transaction recovery. Should the source DB fail, Oracle will sense the failure and failover to the standby database in a clear transaction consistent manner. When the Active DataGuard license is employed further, high availability features are available but most key is that the database can now be opened in read only mode while still playing out transactions from the primary– and the standby database can be used as a Read Only copy for reporting purposes.

For our purpose of migration, Enterprise Edition of DataGuard – that is the non-active standby version will suffice. So if your organization has Oracle Enterprise edition processor licenses, no further license cost is needed and it is bundled into the RDBMS natively. Key to this strategy is the fact that Oracle ‘heterogenous’ Exadata to non-Exadata DataGuard configurations are possible. This is taken from an Exadata MAA Oracle whitepaper:

Exadata Database Machine: It is transparent to Data Guard whether primary and/or standby databases reside on an Exadata Database Machine or on other hardware, as long as the platform IDs of primary and standby systems within the same Data Guard configuration conform to the support requirements….

A typical Oracle DataGuard configuration is based on an RMAN duplicate command to instantiate the database and would look something like this:

```
DUPLICATE TARGET DATABASE
FOR STANDBY
FROM ACTIVE DATABASE
SPFILE
  PARAMETER_VALUE_CONVERT ‘’, ‘’
  SET DB_FILE_NAME_CONVERT ‘’, ‘’
  SET LOG_FILE_NAME_CONVERT ‘’, ‘’
  SET SGA_MAX_SIZE 200M
  SET SGA_TARGET 125M;
```

There are of course multiple precursor steps, but that command will create the standby remotely.

One Pre-requisite is that Grid Infrastructure is the same version, and the Oracle RDBMS must be the same version and release at both primary/standby sites. Also, the GRID/RDBMS Software must already be installed at both Primary/Standby environments. Verify this at both ends with ‘opatch lsinventory’ in all

Oracle and Grid Homes.

Implementing an RMAN based duplicate command over the wire to instantiate the standby database as a prerequisite can be problematic if proper bandwidth on the circuit connecting the on-premises site to Azure isn't available. This cannot be done with a VPN over the publicly switched internet. The DataGuard duplication command requires continuous, uninterrupted connectivity for the duration of the operation. If the network hiccups the operation will time out, clean-up will need to be done and the duplicate command will need to be started from the beginning. The good news is that that data will go over in whatever compression format it is in. – (It is uncompressed once it reaches the standby server). This will help save time pushing the data through the wire compressed rather than uncompressed. One of the key prerequisites to a successful migration is having a large bandwidth, reliable interconnect between on-premises and Azure. This is typically accomplished with Azure ExpressRoute.

With DataGuard you have the options of a live migration or a quiesced migration. In a quiesced migration the database will be closed off to all but admins with no transactions occurring. The RMAN duplicate command is invoked, the duplicate completed and a manual switch over is initiated promoting the standby to the new primary. In a live database migration, the cutover is not done immediately. Rather log shipping is enabled, and the original primary can be reopened for user activity which will be recorded on the standby ongoing. The following command initiates managed recovery on the standby:

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM SESSION;
```

At this point the database is fully instantiated and Change Data Capture (CDC) is being accomplished via DataGuard. Typically, the target database will be within sub second synchronization with the source. This is not a DR configuration but a migration tool so there are several limitations to this technique.

It will require more storage for decompressing the HCC tables

Active Data Guard cannot be used to read EHCC tables on non-Exadata Stdby

Post failover role reversal – Primary Exa will not be able to leverage EHCC

At this point, when the database is opened for read/write activity all the data will be there – but anything in HCC will not be available for selects and transactions. Anything in an HCC table will get this error when being queried on a non-Exadata system.

```
ORA-64307: hybrid columnar compression is not supported for tablespaces on this storage type
```

In order to be able to access the HCC data un-compress commands have to be performed on the HCC tables:

```
ALTER TABLE example MOVE NOCOMPRESS;
```

Part of the elegance of an Oracle DataGuard migration is all this activity except for the last step of un-compressing after the DB cutover require no production down time. The database can be instantiated, and log shipping started with no impact on the performance of the Source Exadata machine (Some alter actions such as adding standby redo logs will be needed on the production system but nothing that requires a reboot or downtime). Finally, when it comes time to cutover, some downtime is incurred as the tables must be un-compressed. Obviously, this will need to be done as expeditiously using parallelism as much as possible.

```
ALTER TABLE table_name MOVE NOCOMPRESS PARALLEL;
```

The final steps to production cutover include verifying that all objects were successfully migrated, and that all data is accessible in non-compressed format. Collect stats and verify any newly designed/implemented indexes are valid and visible. The deduplication and compression technologies of Silk will be implemented transparently.

This paper attempted to discuss what Oracle Exadata is, how HCC and Storage cell technology presents some migration challenges and how to use Oracle DataGuard to successfully migrate to Azure + Silk. Nothing more can be said than planning is key to it all. Identify your HCC and indexing challenges early on. Make sure your network bandwidth is adequate and that a security model is in place. But remember this is just another Oracle RDBMS ultimately. With proper planning and testing the migration can be accomplished with minimal downtime and service interruption.

Ready to get started? Visit www.silk.us to learn more.