Exadata for Oracle DBAs

By Arup Nanda

If you are an Oracle DBA familiar with Oracle 11gR2 RAC, clusterware and ASM and about to become a Database Machine Administrator (DMA) – the folks who manage an Exadata system, the very first question you are bound to ask yourself – how much extra stuff you have to learn, isn't it? If you are wondering about how much of your own prior knowledge you can apply, what is different in Exadata, what makes it special, fast and efficient, and a whole lot of other questions, this article will explain all that. Here you will learn about the magic behind the premise of the Exadata database machine.

What is Exadata

Exadata is a consolidated appliance containing all those components that make up a database system – Storage, Flash Disks, Database Servers, Infiniband Switches, Ethernet Switches and KVM (some models). Yet, it is not an appliance. Why? Because of two very critical and important reasons: it has additional software to make it a better database machine and the components, which are engineered to work very well together, can be managed independently and not as a sealed blackbox. That's why Oracle calls it a Database Machine (DBM); not an appliance or a server. Although the components can be managed separately it is possible (and advisable) to manage the entire system as a whole. The administrators are not called DBAs or system admins; but by a special term – Database Database Machine Administrator (DMA).

Oracle Database System

Before we go down to the details, let's first start with a basic description of how an Oracle database works. The actual data is stored on the datafiles on the disk. However, when the user selects from the database, the data does not come from the disks directly. Instead, the data is sent to a memory area called database buffer cache. The smallest amount of data addressable by an Oracle database is a database block, which is generally 8 KB in size but could be as small as 2 KB or as large as 32 KB based on how the DBA has configured it. A single datablock may hold several rows of a table. When the user selects a table, the entire block, not a specific set of rows, are selected from the datafiles and moved to the buffer cache.

Referring to Fig 1, suppose the user wants to select the rows of all the customers who are angry. The following query would make it happen:

```
select name from customers where status = 'ANGRY';
```

However, the data stored in the datafiles do not have any knowledge of the status column inside the files. Instead database server process picks up a block from the datafile and places it in the buffer cache. From the buffer, it would then extract the rows that satisfy the condition and return them to the user.

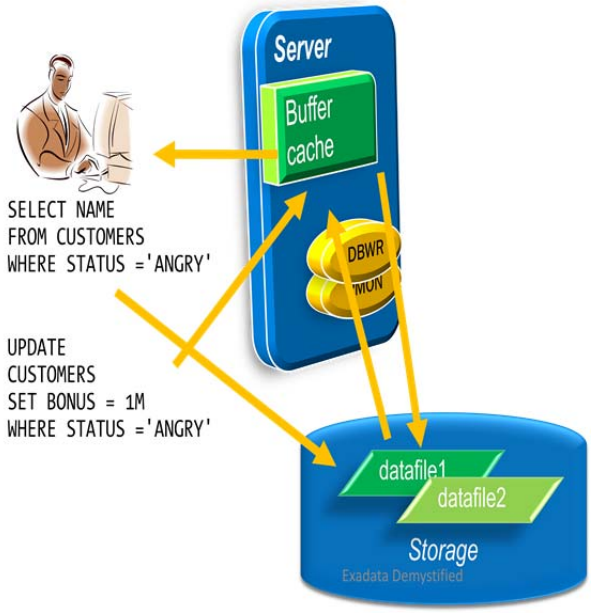

Figure 1 Oracle Instance

Suppose at this stage, the user wants to placate the angry customers by giving them 1 million bonus points. This can be done by issuing the following query:

```
update customers set bonus = 1M where status = 'ANGRY';
```

This update will not make changes in the datafile; but in the buffer cache. Later a different process, called Database Buffer Writer (DBWR) copies the buffers from the cache to the datafiles to make them consistent. This process is just of the many required to manage a database instance. Other processes such as System Monitor (SMON), process monitor (PMON), etc. make sure the Oracle database system works as expected. The combination of these processes and memory areas such as buffer cache is known as an *Oracle Instance*.

In a Real Application Cluster (RAC) database system, there are two (or more) machines working on the same physical database system. Each machine runs an Oracle Instance. Since each instance can potentially write buffers to the disk, there is a risk of DBWR process of one machine overwriting the

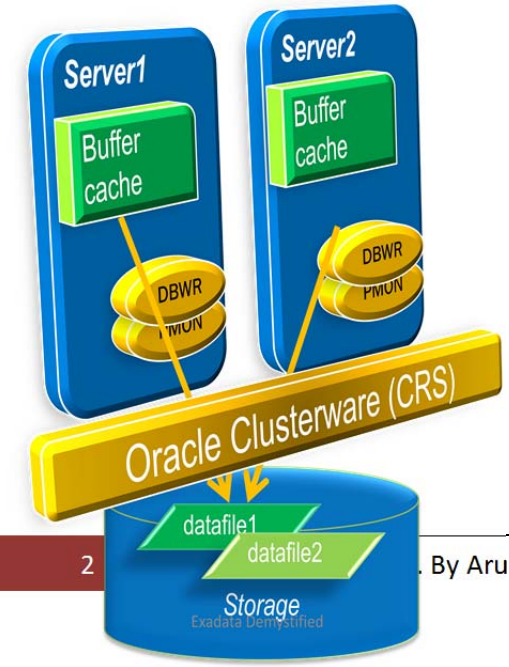

Figure 2 Oracle RAC

changes in the datafile made by the other leading to data corruption. To avoid this situation, a special software known as Oracle Clusterware coordinates the changes made by different machines in the same cluster. This is also known as Cluster Ready Services (CRS), as shown in Fig 2.

Limitations in the Current Model

Now that you understand how the traditional Oracle model works, let's see why there is an inherent bottleneck in the current design. The following components make up the processing parts of an Oracle database system

- CPU – the processing unit where the Oracle database software is installed and where the processes of the Oracle Instance run. The processes send instructions to the storage to get the datablocks.
- Memory – in the same system where the software runs and in which the database instance's memory areas (buffer cache, etc.) are located. When the datablocks are retrieved from the disk, they are stored here.
- Network – enables the communication between the CPU and the storage; could be Ethernet (for NAS) or fiber (for SANs). The datablocks come over this pathway.
- I/O Controller – the subsystems that enable flow of storage related information; examples include I/O Cards, Front End Adapter (FEA) cards, etc.
- Storage – the actual disks and the software system that manages them (e.g. SAN and NAS, etc.). This layer sends the datablocks as requested by the process running on the CPU.

The performance of the database system depends on the effectiveness of these components. The faster the component is, the better the performance. Over the years, we have seen the performance of these components going up by leaps and bounds, e.g. CPU powers have risen from KHz to GHz ranges; memory has become orders of magnitude faster and more expansive; I/O controllers appeared making the access faster and networks changed from bits per second to today's ubiquitous 10 Gbits/sec. However the story is different for storage – the disks haven't seen that kind of dramatic evolution. The law of physics comes in the way – the disks can rotate only so fast and not more.

While most of the components have become exponentially faster, they are still part of a chain and storage is the weakest link impacting the performance of the database system. Unless the storage becomes comparably faster, the overall performance will not improve with faster processors or more memory. System designers generally employ three techniques:

- Putting cache on the SAN or NAS systems avoiding spinning disks
- Moving to a flash based storage from harddisks
- Increasing the buffer cache by adding more memory so that the database instance can find everything there without going to the disk

While they work, they are not really solutions in case of a database system. The success of SAN caches is built upon predictive analytics, where the SAN software determines if a specific part of the disk is accessed more and moves it to the cache. This works well, if a small percentage of the disks is accessed most often. The emphasis is on disk; not data. The SAN does not know anything about data inside the

disks; it predicts the next hot areas of the disk using some heuristics, which may now work well for most database systems. Most database systems are way bigger than SAN caches; so a little portion of the disk being on the cache doesn't help. Besides, the SAN needs to get the data from the disk to the cache and that takes time leading to the I/O at the disk level being still high. SAN caches are excellent for filesystems or very small databases; not for a regular sized database.

Solid state disks (also called flash disks) are definitely faster than hard drives; but they are still too expensive to be practical. The third argument, adding more memory to the buffer cache seems to have some merit, at least on the surface. However, memory is still very expensive. Besides, how much memory is enough to accommodate a major portion of the database? If you have a 1 GB database and 1 GB buffer cache, you might assume that the whole database will fit in the memory. Unfortunately, it's not true. Oracle database fills up to 7 times the DB size in the buffer cache, as I have explained in my blogpost: http://arup.blogspot.com/2011/04/can-i-fit-80mb-database-completely-in.html. Trying to get as much memory to fit in a typical database may not be practical at all. So this option is non-viable.

So, where does that leave us? We need to think outside the box. Let's revisit the problem. A typical query may:

- Select 10% of the entire storage
- Use only 1% of the data it gets

To gain performance, adding CPU and memory is not the answer, unless you have pockets as deep as Mariana Trench. The feasible solution is the database needing to get less from the storage, which will make the storage more efficient elevating it from the weakest link position. To enable this, we can't have the storage get all the blocks of the table from the spinning disks and send them to the CPU. We need to have some filtering at the storage level, i.e. the storage must be cognizant of the data it stores and it must understand what is being requested. This is what Exadata enables to bring that performance. Let's see how that works.

Magic # 1 Intellegent Communication

Since we identified that the storage must be smart enough to filter at the disk level and return only what is relevant, it has to get that instruction clearly from the CPU. In other words, the CPU and IO must speak the same language. In Exadata, a special type of protocol, called iDB (Intelligent Protocol) allows that communication, as shown in Fig 3. In line with the example shown earlier, the iDB protocol allows the CPU to send what the user wants (the "NAME" column) and the filter (the column "STATUS" = 'ANGRY') to the storage rather than asking to get all the data blocks of the CUSTOMERS table. iDB is built on the top of the Infiniband network protocol and media.

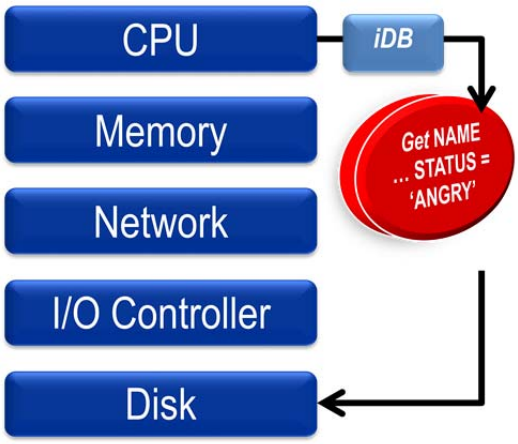

Figure 3 Intelligent iDB Communication

Magic # 2 Storage Cell Server

As you saw earlier, iDB is the medium that lets the storage know what the CPU wants; but how can the storage act on that information? Just a bunch of disks will not make it happen. This calls for some type of processing to be present at the storage layer. Fig 4 shows the conceptual layout of the storage. The physical disks are attached to a server known as Storage Cells. The Storage Cells, which are Sun blade servers, run a special software called Exadata Storage Server (ESS) that can process the iDB calls and send the data back as needed.

Figure 4 Storage Cell

Magic # 3 Storage Indexes

Now that you know the ESS software can understand what the user is asking for and then filter at the storage layer, the next logical question is how it can do so. This is where the third magic of Exadata comes in. Consider a table where a column named RATING is queried as follows:

```
select name from customers where rating = 1;
```

The ESS server needs to know which parts of the disks to search for these values. It divides the disk into 1 MB parts and stores the minimum and maximum values of the column for that part. Refer to the Fig 5. Suppose the occupied part of the disk is 4 MB meaning there will be 4 parts of 1 MB each. The figure shows how the minimum and maximum values of each part are recorded. Using this data, the storage server can easily determine that inside area 1, the minimum value of RATING is 3; so a row with RATING = 1 will not be found there. Therefore the ESS software will skip searching that part of this disk for that row. Similarly it will skip part 2 and part 4 of the disk. Only part 3 will have some rows. Using these min and max values, the ESS software can eliminate searching for 75% of the disk, reducing I/O by 75%! This powerful feature is known as Storage Indexes.

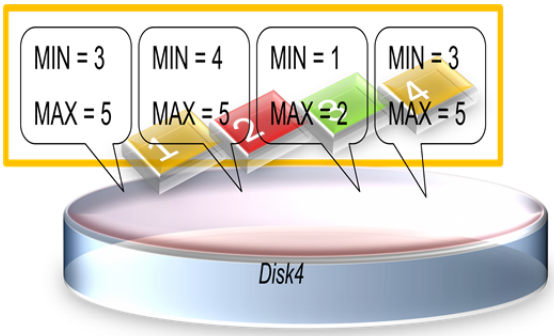

Figure 5 Storage Indexes

Storage Indexes are different from normal Oracle database indexes. Normal indexes reside in the database; storage indexes are in the memory on the storage cells. Normal indexes tell the database which specific blocks where a row can be found; storage indexes tell the storage server where the rows will not be found.

But the storage indexes do not work every time. The mechanism that allows the use of power of the Storage Cells is called Smart Scan. If you want to find out whether Smart Scan and Storage Indexes are being used, you can use the following query:

```
select decode(name,
 'cell physical IO bytes saved by storage index',
    'SI Savings',
 'cell physical IO interconnect bytes returned by smart scan',
    'Smart Scan'
  ) as stat_name, value/1024/1024 as stat_value
from v$mystat s, v$statname n
where s.statistic# = n.statistic#
and n.name in (
  'cell physical IO bytes saved by storage index',
  'cell physical IO interconnect bytes returned by smart scan');
```

Here is a sample output:

```
STAT_NAME  STAT_VALUE
---------- ----------
SI Savings   0.000
Smart Scan   0.000
```

In this Smart Scan and Storage Indexes did not yield any savings. Why not? There are several reasons. First, a simple explanation: Smart Scans have been disabled by database level parameters

```
cell_offload_processing = true;
_kcfis_storageidx_disabled = true;
```

Otherwise, the pre-requisites for Smart Scan haven't been satisfied. Here are the pre-reqs:

- Direct Path Operations
  - Full Table or Full Index Scan
  - 0 Predicates
- Simple Comparison Operators

Other reasons are:

- Cell is not offload capable
- The diskgroup attribute cell.smart_scan_capable set to FALSE
- Smart Scan is not possible on clustered tables, IOTs, etc.

Magic # 4 Smart Flash

These are flash cards presented as disks; not as memory to the Storage Cells. The data, after being retrieved from the disk is stored in the Smart Flash. When the same data is requested, the ESS server satisfies the request from the Smart Flash instead of the disk. So, they are very similar to SAN cache; but Oracle, not the SAN, controls what goes in there and how long data stays in them. For instance, Oracle can determine the tables or indexes (not areas of the disk) that are accessed frequently and moves them to Smart Flash. These could come from various parts of the disks. As a DBA, you can also put specific objects in Smart Flash if you think they will benefit.

While Smart Flash makes reading faster, writing is still done to the spinning disks. So the database writes do not benefit from Smart Flash.

Magic # 5 Operation Offloading

The ESS software does much more than just filtering at the storage level. It can perform some of the operations directly at the storage level and return the results to the database processes, making the overall performance quite high. This is known as cell offloading. Some of the processes offloaded are:

- Bloom Filters
- Functions Offloading. You can know which functions can be offloaded by querying the view V$SQLFN_METADATA
- Decompression with Hybrid Columnar Compression. The compression operations handled by database nodes
- Expansion of Virtual Columns

There are some additional elements for the supercharged performance as well; but these are the major ones.

Components

Now that you know the magic behind the efficiency of Exadata, let's examine the components that make up the Exadata frame. Fig 6 shows the various components and how they are represented in the Exadata frame. The CPU and memory are located in Sun blade servers, where the database and clusterware softwares run. These are called Compute Nodes. The Operating System is either Oracle Enterprise Linux or Solaris. On the bottom of the stack, the I/O controllers and disks are located on Sun blades as well, known as Storage Cells or Cell servers. It runs Oracle Enterprise Linux software. The Exadata Storage Server (ESS) runs here. The network component is implemented on Infiniband and Ethernet switches and infrastructure.

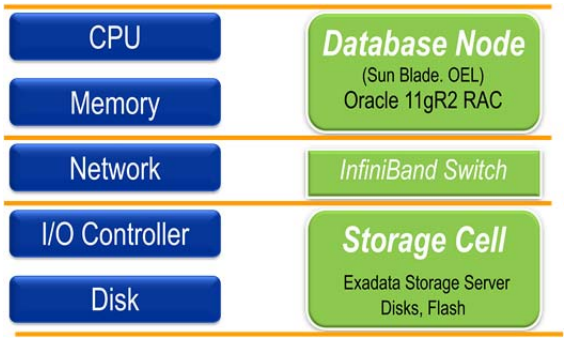

Figure 6 Components of Exadata

Three configuration types are available in Exadata: full rack, half rack, or quarter rack. The architecture is identical across all three types but the number of components differs. On a full rack, there are 14 Storage Cells and 8 Compute Nodes . The other configurations have proportionally number of components.

Network

Infiniband circuitry – the cells and nodes are connected through infiniband for speed and low latency. There are 3 infiniband switches for redundancy and throughput. Note: there are no fiber switches since there is no fiber component.

Ethernet switch – the outside world can communicate via infiniband, or by Ethernet. There is a set of Ethernet switches with ports open to the outside. The clients may connect to the nodes using Ethernet. DMAs and others connect to the nodes and cells using Ethernet as well. Backups are preferably via

Storage Cells

Each cell has 12 disks. Depending on the configuration, these disks are either 600GB high performance or 2TB high capacity (GB here means 1 billion bytes, not 1024MB). You have a choice in the disk type while making the purchase. Each cell also has 384GB of flash disks. These disks can be presented to the compute nodes as storage (to be used by the database) or used a secondary cache for the database cluster (called smart cache).

Since the cells have the disks, how do the database compute nodes access them - or more specifically, how do the ASM instances running on the compute nodes access the disks? Well, the disks are presented to cells only, not to the compute nodes. The compute nodes see the disks through the cells. For the lack of a better analogy, this is akin to network-attached storage. (Please note, the cell disks are not presented as NAS; this is just an analogy.)

Two of the 12 disks are also used for the home directory and other Linux operating system files. These two disks are divided into different partitions as shown in Figure 7 below. he physical disks are divided into multiple partitions. Each partition is then presented as a LUN to the cell. Some LUNs are used to create a filesystem for the OS. The others are presented as storage to the cell. These are called cell disks. The cell disks are further divided as grid disks, ostensibly referencing the grid infrastructure the disks are

Figure 7 Disk Layout

used inside. These grid disks are used to build ASM Diskgroups, so they are used as ASM disks. An ASM diskgroup is made up of several ASM disks from multiple storage cells.

If the diskgroup is built with normal or high redundancy (which is the usual case), the failure groups are placed in different cells, as shown in Fig 8. As a result, if one cell fails, the data is still available on other cells. Finally the database is built on these diskgroups.

Figure 8 ASM Redundancy in Exadata

Administration

Now that you understand the building blocks of Exadata, hopefully you have got an idea of what type of commands are used where. Here is a summary of the commands necessary on various building blocks.

- Compute Nodes
  - Linux Management – vmstat, mpstat, top, fdisk, etc.
  - ASM Commands – SQL*Plus, ASMCMD, ASMCA
  - Database Commands – startup, alter database, etc.
  - Clusterware Commands – CRSCTL, SRVCTL, etc.
- Storage Cells
  - Linux Management – vmstat, mpstat, top, fdisk, etc.
  - CellCLI – command line tool to manage the Cell

How difficult are these commands to master? Actually, they are not that difficult for an Oracle DBA. Here is a breakdown of the skills expected of a Database Machine Administrator (DMA).

Skill	Needed
System Administrator	15%
Storage Administrator	0%
Network Administrator	5%
Database Administrator	60%
Cell Administration	20%

As you can see, if you have been an Oracle 11.2 RAC Database Administrator, you already know 60% of the beast anyway. For the others, it's quite easy to master. For the linux components, visit my 5-part article series on Oracle Technology Network, " Linux Commands" http://bit.ly/k4mKQS. The Cell Administration is new for anyone, including seasoned Storage Administrators. For that, you can refer to my 4-part Exadata Command Reference article series http://bit.ly/lljFl0, also on OTN. All the articles are free. Using these articles you can master the rest 40% very easily to be successful Database Machine Administrator.

Frequently Asked Questions

Let's go through some quite frequently asked questions on Exadata.

Q: Do clients have to connect using Infiniband?

A: No; Ethernet is also available. However, Infiniband allows more throughput and lower latency.

Q: How do you back up Exadata?

A: Through the normal RMAN Backup, just like an ordinary Oracle RAC Database. No special tricks needed for Exadata.

Q: How do you enable Disaster Recovery?

A: Since the storage does not support any hardware based replication, the only solution is Data Guard. The standby could be a non-Exadata box as long as it runs Oracle Enterprise Linux and Oracle 11.2 software. However, you can't use any Exadata specific features such as Smart Scan.

Q: Can I install any other software?

A: Nothing is allowed on Cells. On Compute nodes software can be installed but the space is limited. Generally you may want to install client software like Golden Gate, etc.

Q: How do I monitor it?

A: There are several ways to monitor it. Oracle Enterprise Manager Grid Control has a plugin specific ally for Exadata. Besides the CellCLI commands (available in Exadata Comamnd Reference article series shown above) is used to monitor Cells. Ordinary SQL commands are used to monitor the database.

Conclusion

The purpose of this article was to explain various components of Exadata and how they play together to bring the fast performance. In summary:

- Exadata has an Oracle Database running 11.2 RAC
- The storage cells have added intelligence about data placement
- The compute nodes run Oracle database and Grid Infrastructure software
- Nodes communicate with Cells using iDB which can send more information on the query
- Smart Scan, when possible, reduces I/O at cells for Direct Path operations
- Cell is managed by CellCLI commands
- DMA skills = 60% RAC DBA + 15% Linux + 20% CellCLI + 5% miscellaneous

Hopefully you will now appreciate the engineering behind the Exadata Database Machine making it faster than a regular Oracle database running on same or similar hardware. In this article I attempted to make sure you understand the workings under the hood so that it is no longer a “magic” but simply technology at work. Understanding this technology helps you shed the fear and uncertainty and enable you to assume the role of a Database Machine Administrator with aplomb. Good luck and happy DMA’ing.

About the Author

Arup Nanda (arup@proligence.com) has been an Oracle DBA for last 16 years and now a DMA, has seen all aspects of Oracle Database management from performance tuning to backup recovery. He works as principal global database architect at a major New York area corporation. He has written 4 books, 500+ article, presented 300+ sessions, blogs regularly at arup.blogspot.com, tweets at @arupnanda and spends time with his family in Connecticut when not doing any of these. He was the recipient of Oracle’s DBA of the Year award in 2003.