CACHED ORACLE DISK MANAGER: USAGE GUIDELINES AND BEST PRACTICES

Scott Myers, Senior Principal Technical Product Manager
Storage and Availability Management Group
December 2011

1. INTRODUCTION

Cached ODM (CODM), introduced with the Storage Foundation (SF) 5.1 product release, is an enhancement to the ODM (Oracle Disk Manager) interface, which has been available since 2000 and was first introduced with Oracle 9i. Currently, ODM/CODM supports Oracle 10g and 11g across all the Unix and Linux platforms that SF 5.1 and later support. The specifics of this paper are based on the **SF 5.1SP1** release. Reference the "Veritas Storage Foundation: Storage and Availability Management for Oracle Databases Solaris 5.1 Service Pack 1" available at:

- Sun Solaris: **https://sort.symantec.com/public/documents/sfha/5.1sp1/solaris/productguides/pdf/sf_adv_ora_51sp1_sol.pdf**)
- IBM AIX **https://sort.symantec.com/public/documents/sfha/5.1sp1/aix/productguides/pdf/sf_adv_ora_51sp1_aix.pdf**
- HP HP-UX **https://sort.symantec.com/public/documents/sfha/5.1sp1/hpux/productguides/pdf/sf_adv_ora_51sp1_hpux.pdf**
- Linux **https://sort.symantec.com/public/documents/sfha/5.1sp1/linux/productguides/pdf/sf_adv_ora_51sp1_lin.pdf**

Note: If you plan to implement CODM with an SF release earlier than the SF5.1SP1 (e.g. SF5.1), please refer closely to that earlier release's appropriate documentation ("Veritas Storage Foundation: Storage and Availability Management for Oracle Databases") for CODM specifics. For example, the 'dbed_codm_adm' command referenced later in this document was introduced with SF5.1SP1 – it does not exist for SF5.1 or SF5.1RP1.

ODM I/O bypasses the file system cache and directly reads from and writes to disk. Cached ODM enables selected I/O to use caching (file system buffering) and read ahead, which can improve overall Oracle DB I/O performance. Cached ODM performs a conditional form of caching that is based on per-I/O hints from Oracle. The hints indicate what Oracle will do with the data. ODM uses these hints to perform caching and read ahead for some reads, but ODM avoids caching other reads, possibly even for the same file.

1.1. Considerations for using Cached ODM

Cached ODM will not improve the performance for every Oracle DB workload. Cached ODM is most useful for read-intensive workloads. For write-intensive workloads or low physical server memory environments, Cached ODM is not advised.

Cached ODM provides value add when complementing ODM:

- ODM does direct I/O
    - Oracle will be configured to use larger System Global Area (SGA) to compensate because ODM does not support read-aheads
- Some read-intensive Oracle workloads can perform poorly when ODM is used. CODM can improve the read-intensive performance.
- Oracle SGA is not good enough for some cases where a host may have more than one database instance.
    - Host may have more than one database instance. (Page cache used by CODM can serve as a shared global buffer for multiple database instances)
    - Parallel Query processes many times do not use SGA and thus can benefit from CODM file system buffering.

1.2. Supported configurations

Cached ODM is supported for:
- Storage Foundation (HA)
- Storage Foundation Cluster File System (HA)
- Local and cluster mounts
- Oracle 10g and 11g

Storage Foundation for Oracle RAC is not currently recommended because no performance benefit has yet been observed because of the simultaneous access of data from the multiple nodes of RAC, which causes cache invalidations. Cached ODM does not affect the performance of files and file systems for which you have not enabled caching. The intent of this white paper is provide the reader with guidance on how to use CODM effectively to further enhance performance for those Oracle Databases that can benefit from using a combination of ODM and CODM.

2. ADMINISTRATION

2.1. Verify that DBED package is installed.

The Cached ODM Manager command **dbed_codm_adm,** that will be discuss later in this paper, is an integral part of the Cached ODM capability with SF 5.1SP1, and resides in the DBED package of SF. Thus, it should be verified that this package (VRTSd bed) is installed. To verify the DBED package is installed, run the following command as root:

```
# pkginfo -l VRTSdbed          (for Sun Solaris)

# lslpp -L VRTSdbed            (for IBM AIX)

# rpm -qa | grep VRTSdbed      (for Linux)

# swlist VRTSdbed              (for HP-UX)
```

2.2. Verify that ODM is enabled

CODM is an extension to the core ODM feature, and thus ODM is a pre-requisite that needs to be installed and functioning.

To verify that ODM is enabled, perform the following steps: (information shown here is specific for Solaris Sparc and documented in "Veritas Storage Foundation: Storage and Availability Management for Oracle Databases Solaris 5.1 Service Pack 1".

1. Verify that Oracle Disk Manager is running
    - Start the Oracle database.
    - Check that the instance is using the Oracle Disk Manager function:
      # **cat /dev/odm/stats**
      # **echo $?**
      0
    - Verify that the Oracle Disk Manager is loaded:
      # **modinfo | grep ODM | grep VRTS**
      162 7b76c000 184a0 25 1 odm VRTS ODM 5.1,REV=5.1_SP1
    - In the Oracle alert log, verify the Oracle instance is running. The log should contain output similar to the following:
      For 5.1SP1, the alert log shows the Oracle instance running with ODM: Veritas

1.1 ODM Library, Version 2.0
1.2
Note: Cached ODM functionality requires ODM Library, Version 2.0 or later.

If step 1 above is **not** successful, then execute the following steps 2-5.

2. Check that the VRTSodm package is installed:
   # **pkginfo VRTSodm**
   system VRTSodm Veritas ODM

3. Make sure that the Veritas odm library exists and is linked to $ORACLE_HOME/lib

   /opt/VRTSodm/lib/libodm.so must exist.

   If you are using Oracle 10g on SPARC,
   $ORACLE_HOME/lib/libodm10.so is linked to
   /opt/VRTSodm/lib/sparcv9/libodm.so.

   If you are using Oracle 11g on SPARC,
   $ORACLE_HOME/lib/libodm11.so is linked to
   /opt/VRTSodm/lib/sparcv9/libodm.so.

4. Verify that the ODM feature is included in the license:
   # **/opt/VRTS/bin/vxlicrep | grep ODM**
   QLOGODM = Enabled
   ODM = Enabled
   The output verifies that ODM is enabled.

5. Verify that /dev/odm exists:
   #df | grep odm
   /dev/odm (/dev/odm): 0 blocks 0 files

2.3. Configuring Cached ODM

Configuring Cached ODM requires first enabling Cached ODM for a file system. After enabling Cached ODM, it can be configured in two ways:

- The primary configuration method: turn caching on or off for all I/O on a per-file basis.
- The secondary configuration method: adjust theODM cachemap. The cachemap maps file type and I/O type combinations into caching advisories.

Enabling Cached ODM for file systems

Cached ODM is initially disabled on a file system. You can enable Cached ODM for a file system by setting the odm_cache_enable option of the vxtunefs command after the file system is mounted.

See the vxtunefs(1M) manual page.

Note: The vxtunefs command enables conditional caching for all of the ODM files on the file system.

To enable Cached ODM for a file system

1. Enable Cached ODM on the VxFS file system /database01:
   # **vxtunefs -o odm_cache_enable=1 /database01**
2. Optionally, you can make this setting persistent across mounts by adding a file system entry in the file /etc/vx/tunefstab:
   # **/dev/vx/dsk/datadg/database01 odm_cache_enable=1**
   See the tunefstab(4) manual page.
3. For Cluster File System, you must modify /etc/vx/tunefstab on all nodes.

Tuning Cached ODM settings for individual files

You can use the odmadm setcachefile command to override the cachemap for a specific file so thatODM caches either all or none of the I/O to the file. The caching state can be ON, OFF, or DEF (default). The DEF caching state is conditional caching, meaning that for each I/O, ODM consults the cachemap and determines whether the specified file type and I/O type combination should be cached. The ON caching state causes the specified file always to be cached, while the OFF caching state causes the specified file never to be cached.

See the odmadm(1M) manual page.

Note: The cache advisories operate only if Cached ODM is enabled for the file system. If the odm_cache_enable flag is zero, Cached ODM is OFF for all of the files in that file system, even if the individual file cache advisory for a file is ON.

To enable unconditional caching on a file
Enable unconditional caching on the file /mnt1/file1:
/opt/VRTS/bin/odmadm setcachefile /mnt1/file1=on
With this command, ODM caches all reads from file1.

To disable caching on a file
Disable caching on the file /mnt1/file2:
/opt/VRTS/bin/odmadm setcachefile /mnt1/file2=off
With this command, ODM does not cache reads from file2.

To check on the current cache advisory settings for a file
Check the current cache advisory settings of the files /mnt1/file1 and /mnt2/file2:
/opt/VRTS/bin/odmadm getcachefile /mnt1/file1 /mnt2/file2
/mnt1/file1,ON
/mnt2/file2,OFF

To reset all files to the default cache advisory
Reset all files to the default cache advisory:
 # /opt/VRTS/bin/odmadm resetcachefiles mountpoint

Tuning Cached ODM settings via the cachemap

You can use the odmadm setcachemap command to configure the cachemap. The cachemap maps file type and I/O type combinations to caching advisories. ODM uses the cachemap for all files that have the default conditional cache setting. Such files are those for which caching has not been turned on or off by the odmadm setcachefile command. Note that 'setcachefile' overrides the cachemap, either turning-on or turning-off caching unconditionally.

See the odmadm(1M) manual page.

By default, the cachemap is empty, but you can add caching advisories by using the odmadm setcachemap command.

To add caching advisories to the cachemap

1. Add a caching advisory to the cachemap for a subset of files:
   # **/opt/VRTS/bin/odmadm setcachemap data/data_read_seq=cache,readahead**
   With this example command, ODM uses caching and readahead for I/O to data files (contain tables and/or indexes) that have the data_read_seq I/O type. You can view the valid file type and I/O type values from the output of the odmadm getcachemap command.

   See the odmadm(1M) manual page.

2. Add a caching advisory to the cachemap for all files:
   # **/opt/VRTS/bin/odmadm setcachemap all/all=all**
   This example command would make the “default” behavior to cache everything for all files. One could still turn off caching on individual files using the **setcachefile** command as described above. Please note that the scope of the **setcachemap all/all=all** command, as with all “setcachemap” commands, is for all file systems that have CODM enabled. Turning on caching for all files will probably never be advisable, as typically, only a small subset of database files will benefit from the caching feature. Section 3 of this document discusses how to identify the files that will benefit from caching.

3. The cachemap is local on each node. To make the same caching decisions on each node in a cluster, keep the cachemap settings consistent by running the same /opt/VRTS/bin/odmadm setcachemap commands on each node in a cluster.

Making the caching settings persistent across mounts

By default, the Cached ODM settings are not persistent across mounts. You can make the settings persistent by addng them to odmadm configuration files.

1. Cachemap settings can be added to /etc/vx/odmadm:
   setcachemap data/read_data_header=cache
   setcachemap all/datapump=cache,readahead

2. The cachemap is local on each node. To keep the cachemap consistent in a cluster, the contents of /etc/vx/odmadm must be the same on each node.

3. Per-file settings can be manually added to the lost+found/odmadm file in the file system.

   For example, to disable caching for oradata/file1 each time the filesystem is mounted, enter the following in the lost+found/odmadm file in the file system.

   setcachefile oradata/file1=off

   The per-file settings in lost+found/odmadm file may also be managed using Cached ODM Manager (see below). The file should not be edited manually while using the Cached ODM Manager.

2.4. Administering Cached ODM settings with Cached ODM Manager

The Cached ODM Manager simplifies the task of managing the cached ODM settings for database administrators:

- Cached ODM Manager enables you to manage and configure cached ODM on database files without requiring root privileges.
- The settings applied with the Cached ODM Manager are automatically persistent and common across all the cluster nodes.
- While the Cached ODM Manager does not provide an interface to the ODM cachemap, it enables setting the cached ODM setting to ON/ OFF (and not toDEF).

The Cached ODM Manager command dbed_codm_adm should be run by a DBA.

Note: If you are using Cached ODM Manager for clone databases, a clone database will not have the cached ODM settings enabled when it is enabled for the primary database. You must manually enable cached ODM by setting odm_cache_enable = 1 for the clone database.

The CachedODMManager command dbed_codm_adm syntax is illustrated below and its options are listed in Table 1.

dbed_codm_adm -S ORACLE_SID -H ORACLE_HOME -o display [-n num -c col -t tbs]

dbed_codm_adm -S ORACLE_SID -H ORACLE_HOME -o [on | off | odmstats] datafile | -f list_file

dbed_codm_adm -S ORACLE_SID -H ORACLE_HOME -o filestate [datafile | -f list_file] -o iostats

Table 1 Cached ODM Manager command options

Option	Use
-S ORACLE_SID	Specify the ORACLE_SID
-H ORACLE_HOME	Specify the ORACLE_HOME
-o display	Display the top 10 files (10is the default) sorted on a certain column of the V$FILESTAT view of the database (the default sort column is PHYRDS).
-o filestate	Show the file state whether the file has Cached ODM turned on or not.
-o on	Enable Cached ODM.
-o off	Disable Cached ODM.
-o odmstats	Displays Cached ODM I/O statistics per file.
-o iostats	Displays cumulative I/O statistics for file-type and I/O type combinations.
-c col	The File I/O statistics will be sorted on specified column index. This is an optional field. By default, I/O statistics will be sorted on number of physical reads (PHYRDS).
-n num	Used to change the default number of files displayed. Use this option together with *–o display* to show the top number

	of files which are candidates for enabling Cached ODM.
-f list_file	*list_file* is a file that contains a list of the datafiles of the Oracle database that will be processed by Cached ODM Manager commands. If you do not provide a list, all of the datafiles will be processed, except in the case of *–o on* or *–o off* options.
-t tbs	Specifies the tablespace for *the –o display* option to display candidate files for that tablespace only.
datafile	Specify one datafile to be processed.

3. HOW TO DETERMINE WHAT FILES TO ENABLE FOR CODM

The key to CODM being effective, is enabling its use for only those files that can benefit from this selective file system buffering capability. We will examine two methods for determining which files will benefit:

- Using Cached ODM Manager
- Using Oracle Automated Workload Repository (AWR)

3.1. Using Cached ODM Manager

The Cached ODM Manager enables you to use the dbed_codm_adm command to display a list of candidate files.

dbed_codm_adm -S ORACLE_SID -H ORACLE_HOME -o display [-c col_index] \
[-n num] [-t tablespace]
The *-o display* option checks for unused system memory (free memory). If the free memory is less than 25% of the total system memory, it will flag a warning not to turn on cached ODM for any files. When the system is low on memory there is no performance gain with Cached ODM. This is just a warning, and 25% is a 'rule of thumb' advisory regarding low memory.

The *display* shows the files read count (PHYRDS) using Oracle V$FILESTAT view. The flag *–c* is used to sort the output on specified column.

After you enable Cached ODM on the first set of datafiles using the –o on option, you can continue to call the dbed_codm_adm -o display option to display the next set of candidate datafiles.

The following example procedures indicate the usage for this command.

To show the top 10 files **without** Cached ODM enabled

- To find the top 10 files (default is 10) with the highest Read I/O and for which Cached ODM is not enabled, run the command:

  $ **dbed_codm_adm -S prod -H /orahome -o display**

File I/O statistics from Oracle V$FILESTAT view sorted on PHYRDS in descending order :

```
FILENAME                    PHYRDS    PHYWRTS  PHYBLKR  PHYBLKWRT  READTIM   WRITETIM
------------------------------------------------------------------------------------------------------------------------
/data/CODM/system01.dbf       5303       3410       6715       4507       1440          1
/data/CODM/sysaux01.dbf       1187      10433       3115      15160        549         21
/data/CODM/undotbs01.dbf        37       4983         37       8643         26        286
```

To show the top 15 files without Cached ODM enabled for tablespace *tbs1*

- To find the top 15 files for tablespace *products* with the highest Read I/O and for which Cached ODM is not enabled, run the command:

$ dbed_codm_adm -S $ORACLE_SID -H $ORACLE_HOME -o display -n 15 -t products

File I/O statistics from Oracle V$FILESTAT view sorted on PHYRDS in descending order:

```
FILENAME                                PHYRDS    PHYWRTS  PHYBLKRD  PHYBLKWRT  READTIM   WRITETIM
------------------------------------------------------------------------------------------------------------------------
/dst_demo/oradata/dstdemo/products1.f        5          1         5          1         0          0
/dst_demo/oradata/dstdemo/products3.f        5          1         5          1         0          0
/dst_demo/oradata/dstdemo/products2.f        5          1         5          1         0          0
```

3.2. Using Oracle Automated Workload Repository (AWR)

Prior to enabling CODM for any files, capture an AWR report. The AWR reports that we captured are from a TPC-C workload. From the 'Main Report' menu of the AWR report, select the 'IO Stats' entry as shown here:

Main Report

- **Wait Events Statistics**
- **Instance Activity Statistics**
- **IO Stats**
- **Buffer Pool Statistics**
- **Advisory Statistics**
- **Wait Statistics**
- **Undo Statistics**
- **Latch Statistics**
- **Segment Statistics**
- **Dictionary Cache Statistics**
- **Library Cache Statistics**
- **Memory Statistics**
- **Streams Statistics**
- **Resource Limit Statistics**
- **Shared Server Statistics**
- **init.ora Parameters**

This will take you to:

IO Stats

- **IOStat by Filetype summary**
- **Tablespace IO Stats**
- **File IO Stats**

Select 'File IO Stats, which will take you to:

File IO Stats (CODM not enabled)

- **ordered by Tablespace, File**

Tablespace	Filename	Reads	Av Reads/s	Av Rd(ms)	Av Blks/Rd	Writes	Av Writes/s	Buffer Waits	Av Buf Wt(ms)
CUST_0	/data_disks/cust_0_0	129,917	42	6.57	1.00	56,442	18	2	0.00
CUST_0	/data_disks/cust_0_1	90,470	29	6.53	1.01	44,895	15	2	10.00
CUST_0	/data_disks/cust_0_10	49,451	16	6.85	1.03	31,761	10	3	3.33
CUST_0	/data_disks/cust_0_11	51,603	17	6.74	1.01	34,754	11	7	4.29
……									
STOK_0	/data_disks/stok_0_19	1,355,552	439	2.3	1	427,930	139	56	5.71
STOK_0	/data_disks/stok_0_20	1,291,314	418	2.26	1	404,456	131	51	4.9
……									

Notice that the table is sorted by tablespace, file name. We want to sort on 'Reads', so we cut and paste this table into a spreadsheet and sort on the 'Reads' column:

Tablespace	Filename	Reads	Av Reads/s	Av Rd(ms)	Av Blks/Rd	Writes	Av Writes/s	Buffer Waits	Av Buf Wt(ms)
STOK_0	/data_disks/stok_0_19	1,355,552	439	2.3	1	427,930	139	56	5.71
STOK_0	/data_disks/stok_0_20	1,291,314	418	2.26	1	404,456	131	51	4.9
STOK_0	/data_disks/stok_0_17	1,206,762	391	2.4	1	380,259	123	44	6.82

STOK_0	/data_disks/stok_0_22	1,093,721	354	2.32	1	337,366	109	39	10.26
STOK_0	/data_disks/stok_0_21	1,007,080	326	2.36	1	307,721	100	31	8.06
STOK_0	/data_disks/stok_0_18	970,237	314	2.54	1	303,491	98	25	7.6
STOK_0	/data_disks/stok_0_9	948,634	307	2.79	1	295,543	96	21	10
STOK_0	/data_disks/stok_0_24	948,307	307	2.38	1	287,957	93	37	6.22
STOK_0	/data_disks/stok_0_16	932,762	302	2.5	1	290,726	94	19	5.79
STOK_0	/data_disks/stok_0_15	926,587	300	2.52	1	289,153	94	19	5.26
STOK_0	/data_disks/stok_0_39	893,589	289	2.81	1	277,501	90	26	5.77
STOK_0	/data_disks/stok_0_10	891,996	289	2.72	1	276,645	90	26	5.38
STOK_0	/data_disks/stok_0_14	885,309	287	2.56	1	274,512	89	33	4.24
STOK_0	/data_disks/stok_0_23	828,505	268	2.43	1	248,135	80	32	7.81
STOK_0	/data_disks/stok_0_13	825,819	267	2.67	1	255,934	83	20	9
STOK_0	/data_disks/stok_0_4	808,331	262	2.9	1	248,552	80	29	5.52

We see that the tablespace 'STOK_O' is the 'busiest' tablespace. To get a sense of how much of the read activity is logical (read satisfied with data in Oracle SGA buffer cache), vs physical (read is satisfied by getting it from disk) we go to the 'Segments by Logical Read' and 'Segments by Physical Read' section of the AWR report:

Segments by Logical Reads

Total Logical Reads: 158,235,834

Captured Segments account for 92.3% of Total

Owner	Tablespace Name	Object Name	Subobject Name	Obj. Type	Logical Reads	%Total
TPCC	STOK_0	STOK		TABLE	81,460,432	51.48
TPCC	ITEM_0	ITEM		TABLE	16,414,304	10.37
TPCC	ORDR_0	ORDL		TABLE	15,257,296	9.64
TPCC	IORDR2_0	IORDR2		INDEX	7,288,320	4.61
TPCC	NORD_0	NORD		TABLE	6,335,040	4.00

Segments by Physical Reads

Total Physical Reads: 37,162,918

Captured Segments account for 99.9% of Total

Owner	Tablespace Name	Object Name	Subobject Name	Obj. Type	Physical Reads	%Total
TPCC	STOK_0	STOK		TABLE	30,365,818	81.71
TPCC	CUST_0	CUST		TABLE	3,639,774	9.79
TPCC	IORDR2_0	IORDR2		INDEX	994,905	2.68
TPCC	ORDR_0	ORDL		TABLE	885,602	2.38
TPCC	ICUST2_0	ICUST2		INDEX	881,774	2.37

Here we can readily see that this same tablespace, STOK_0, has the most logical and physical activity. Notice that the % of physical reads/total reads is quite high (30 million/(81 million + 30 million)= 28%. And its physical read activity is 82% of the total database physical activity. So the files upon which the STOK_0 tablespace reside are surely good candidates for using Cached ODM, in order to drive down the physical i/o activity. We also see that the database object that resides in this tablespace is the STOK table.

We should also look at how much free memory will be available for CODM to use, and the total physical size of the STOK table, in order to get a sense of how effective caching will be. For this particular environment, the server has 32GB of physical memory and the size of the Oracle SGA is 4GB. Thus, there is quite a bit of 'free memory'. The size of the STOK table is approximately 26GB. Thus it is reasonable to expect that a sizable portion of the table will be able to be 'cached' in file system buffers that CODM would be using. We also know that the nature of read data access to the STOK table is that of many simultaneous users going after the same rows and blocks over time. Thus, there would be real benefit if this repeatedly accessed data could be cached in memory instead of always having to be obtained from physical disk.

We have identified the files that should be enabled to use CODM. So we now enable caching for the file system and these files:

Enable Cached ODM for file system:

```
vxtunefs –o odm_cache_enable=1 /data_disks
```

Enable Cached ODM for the files:

```
##The STOK tablespace spans 40 files, so the following commands are entered.
# /opt/VRTS/bin/odmadm setcachefile /data_disks/stok_0_0=on
# /opt/VRTS/bin/odmadm setcachefile /data_disks/stok_0_1=on
...
# /opt/VRTS/bin/odmadm setcachefile /data_disks/stok_0_38=on
# /opt/VRTS/bin/odmadm setcachefile /data_disks/stok_0_39=on
```

We now run the identical TPC-C workload on the Oracle Database with CODM enabled and then examine the AWR report. So we go to the IO Stats/File IO Stats section of the report which shows:

File IO Stats (CODM enabled)

- **ordered by Tablespace, File**

Tablespace	Filename	Reads	Av Reads/s	Av Rd(ms)	Av Blks/Rd	Writes	Av Writes/s	Buffer Waits	Av Buf Wt(ms)
CUST_0	/data_disks/cust_0_0	270,273	88	4.84	1.00	118,834	38	6	3.33
CUST_0	/data_disks/cust_0_1	196,297	64	4.78	1.02	101,435	33	8	5.00
CUST_0	/data_disks/cust_0_10	98,337	32	5.73	1.02	62,557	20	3	6.67
CUST_0	/data_disks/cust_0_11	109,129	35	5.45	1.02	74,324	24	8	3.75
CUST_0	/data_disks/cust_0_12	176,559	57	4.61	1.00	86,782	28	7	2.86
CUST_0	/data_disks/cust_0_13	131,132	42	5.39	1.00	67,899	22	6	5.00
CUST_0	/data_disks/cust_0_14	155,626	50	4.81	1.00	73,346	24	5	4.00
…									

As before, we want to sort on 'Reads', so we cut and paste this table into a spreadsheet and sort on the 'Reads' column:

Tablespace	Filename	Reads	Av Reads/s	Av Rd(ms)	Av Blks/Rd	Writes	Av Writes/s	Buffer Waits	Av Buf Wt(ms)
STOK_0	/data_disks/stok_0_19	2,943,410	953	0.13	1	962,135	312	172	4.36
STOK_0	/data_disks/stok_0_20	2,795,791	905	0.13	1	911,263	295	163	4.36
STOK_0	/data_disks/stok_0_17	2,578,059	835	0.14	1	846,627	274	119	5.55

STOK_0	/data_disks/stok_0_22	2,326,679	753	0.13	1	759,482	246	135	4.37
STOK_0	/data_disks/stok_0_21	2,116,190	685	0.13	1	691,310	224	104	4.62
STOK_0	/data_disks/stok_0_18	2,038,084	660	0.15	1	671,740	218	64	5.63
STOK_0	/data_disks/stok_0_24	1,984,424	643	0.13	1	647,128	210	127	4.88
STOK_0	/data_disks/stok_0_9	1,977,999	641	0.17	1	654,145	212	94	4.79
STOK_0	/data_disks/stok_0_16	1,960,099	635	0.14	1	645,128	209	71	7.46
STOK_0	/data_disks/stok_0_15	1,942,040	629	0.14	1	638,850	207	72	3.89
STOK_0	/data_disks/stok_0_10	1,850,842	599	0.18	1	613,795	199	77	4.81
STOK_0	/data_disks/stok_0_14	1,845,692	598	0.15	1	608,278	197	79	6.08
STOK_0	/data_disks/stok_0_39	1,831,096	593	0.17	1	608,163	197	66	4.24
STOK_0	/data_disks/stok_0_13	1,731,157	561	0.16	1	572,717	185	52	5.58
STOK_0	/data_disks/stok_0_23	1,709,240	553	0.13	1	558,735	181	107	4.58
STOK_0	/data_disks/stok_0_4	1,651,580	535	0.18	1	548,369	178	52	2.88

Let us now compare the ODM and CODM results side by side.

Tablespace	Filename	ODM Reads	CODM Reads	ODM Av Reads/s	CODM Av Reads/s	ODM Av Rd(ms)	CODMA v Rd(ms)
STOK_0	/data_disks/stok_0_19	1,355,552	2,943,410	**439**	**953**	2.3	0.13
STOK_0	/data_disks/stok_0_20	1,291,314	2,795,791	418	905	**2.26**	**0.13**
STOK_0	/data_disks/stok_0_17	1,206,762	2,578,059	391	835	2.4	0.14
STOK_0	/data_disks/stok_0_22	1,093,721	2,326,679	354	753	2.32	0.13
STOK_0	/data_disks/stok_0_21	1,007,080	2,116,190	326	685	2.36	0.13
STOK_0	/data_disks/stok_0_18	970,237	2,038,084	314	660	2.54	0.15
STOK_0	/data_disks/stok_0_9	948,634	1,984,424	307	643	2.79	0.13
STOK_0	/data_disks/stok_0_24	948,307	1,977,999	307	641	2.38	0.17
STOK_0	/data_disks/stok_0_16	932,762	1,960,099	302	635	2.5	0.14
STOK_0	/data_disks/stok_0_15	926,587	1,942,040	300	629	2.52	0.14
STOK_0	/data_disks/stok_0_39	893,589	1,850,842	289	599	2.81	0.18
STOK_0	/data_disks/stok_0_10	891,996	1,845,692	289	598	2.72	0.15
STOK_0	/data_disks/stok_0_14	885,309	1,831,096	287	593	2.56	0.17
STOK_0	/data_disks/stok_0_23	828,505	1,731,157	268	561	2.43	0.16
STOK_0	/data_disks/stok_0_13	825,819	1,709,240	267	553	2.67	0.13
STOK_0	/data_disks/stok_0_4	808,331	1,651,580	262	535	2.9	0.18

We can readily see dramatic differences between ODM and CODM performance. CODM is providing more than twice as many reads/sec (953 vs 439) as compared to ODM. And the average read response time is almost twenty times faster for CODM vs ODM (0.13 vs 2.26 ms)! Not surprisingly, the Transactions Per Minute (TPM) for the CODM configuration were much higher than for the ODM configuration (69,785 vs 34,841).

4. SINGLE ORACLE INSTANCE ON SERVER

Let us now discuss the considerations for using CODM for Oracle DBMS environments where just a single instance of Oracle is running. One can think of CODM as 'secondary' Oracle SGA – access to the data cached by CODM will be slower than access to data that is already cached in Oracle SGA, but much faster than having to go to physical disk. So one might logically think, that because SGA is faster than CODM, and if only one instance of Oracle is running on the server, then one should just configure SGA to use as much physical memory as is available, and not even consider using CODM. In a perfect, simplistic world, that makes sense. But there are circumstances and considerations in the real world where CODM can add value for single instance per server environments:

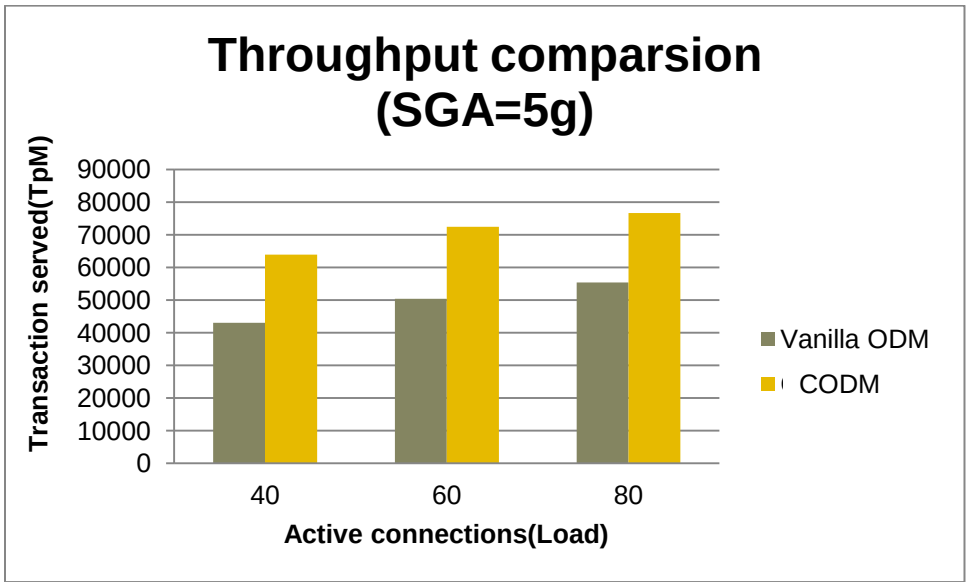
- Oracle SGA cannot be dynamically reconfigured. If one wants to increase or decrease the size of SGA, it requires shutting down the Oracle instance and then restarting it. This means an outage for the application. Furthermore, tuning SGA is typically an iterative exercise requiring multiple changes before finally getting it right, which means multiple or longer production outages. CODM can be dynamically activated/deactivated, at the granularity level of individual file, without requiring any interruption to the Oracle instance.
- Even though just a single instance of Oracle is running on the server, there might be other applications running on the same server that require variable amounts of memory during the day, week, month, etc. So sufficient free memory would always have to be available to support the peak workload requirements for this non-Oracle activity. Thus, SGA could not be permanently assigned this required high-water mark of memory. On the other hand, with CODM, whatever memory is available during the varying workload periods could be dynamically allocated to CODM without impacting the memory requirements for the other applications.

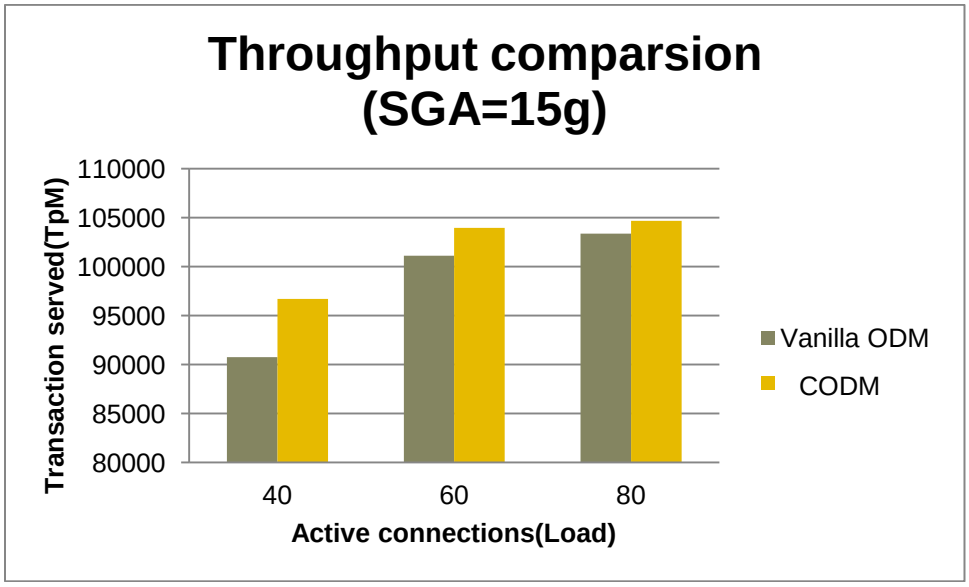
Let's now look at some performance results for the TPC-C benchmark (industry standard benchmark for OLTP workloads) that compare single instance performance with just ODM and with ODM/CODM.

Configuration Tested:

- Hardware
  - Server : x4600(32 Gb Memory, 4 * Dual Core AMD opteron 2.8Ghz)
  - Storage: Sunstoredge 6320(140 *35Gb 15k rpm disks, 10GB cache)
  - Storage Connectivity: 4*2 Gbps FC-HBA from each node.
- Software:
  - OS: RHel 5u3
  - Oracle: 11.2.0.1
  - SF 5.1SP1
- BenchMark
  - TPC-C (fully scaled to 1000 warehouse, ~200 GB size and ~500 files).
- SGA scaling :Runs with multiple SGA sizes
  - 5GB and 15GB
- CPU Scaling :Runs at different Load (active connections)
  - 40, 60 and 80 Connections

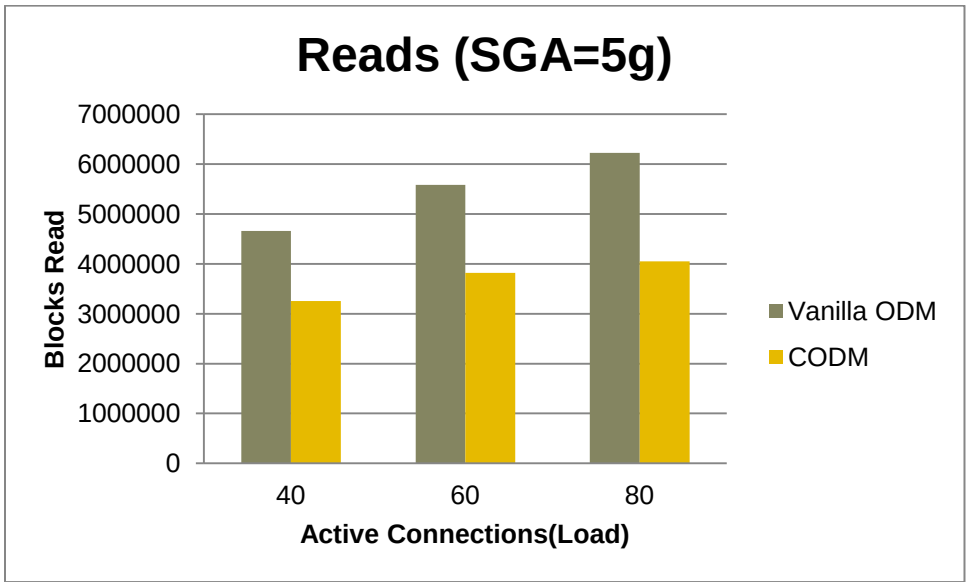
Throughput Results:

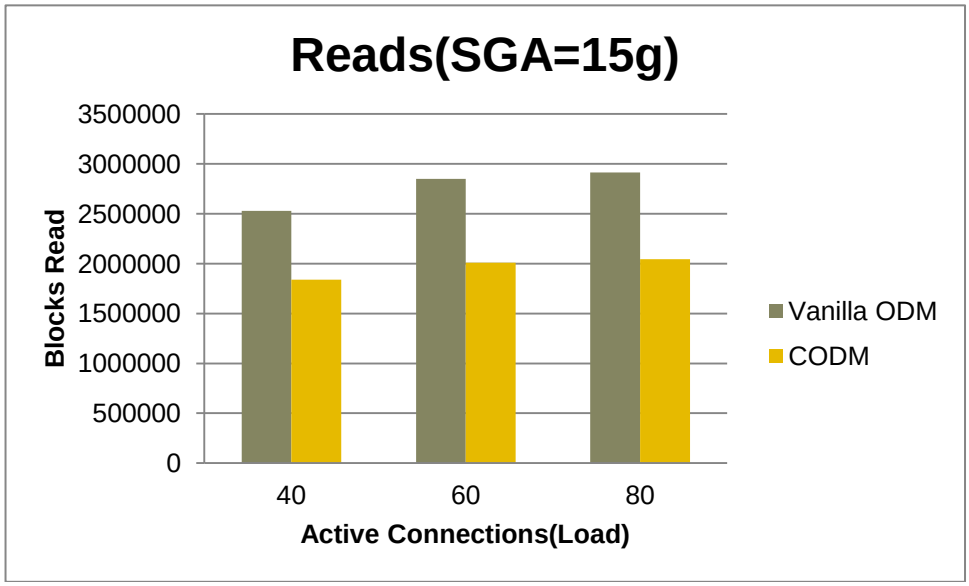




We can readily see that CODM provides significant high throughput (Transactions per Minute [TpM] as compared to ODM, especially for the smaller SGA configuration – about 50% higher for the SGA=5g configuration.

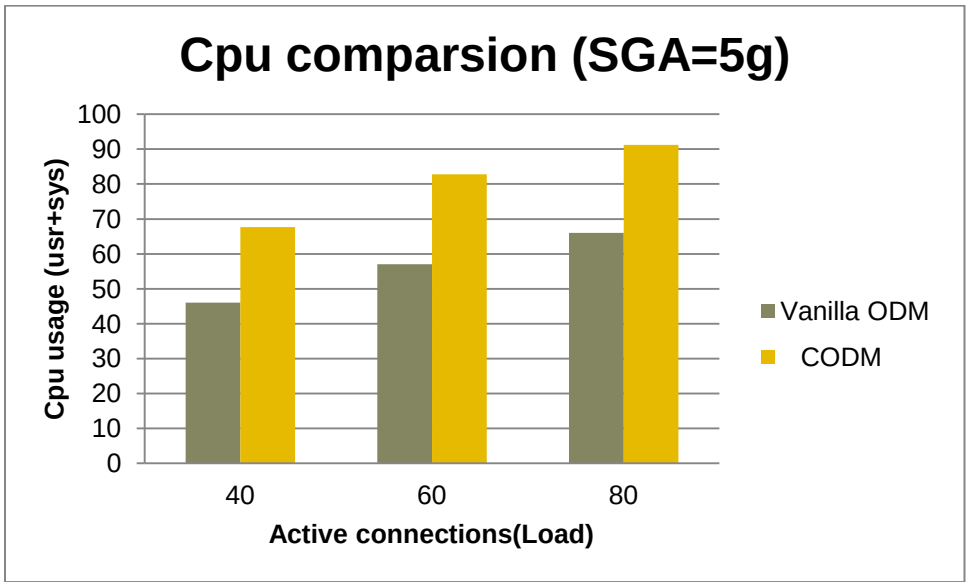
Disk read results:

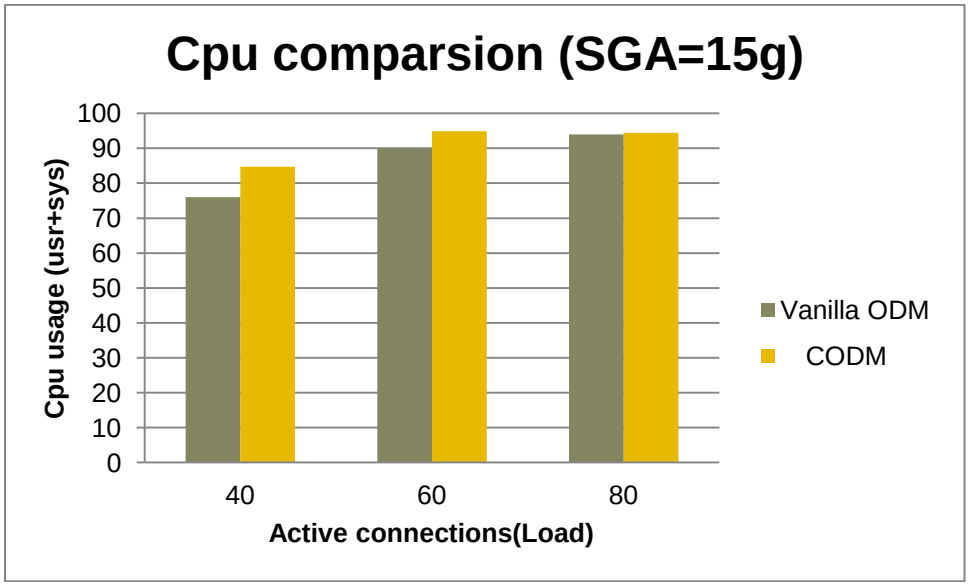




We can readily see that CODM requires less physical disk reads than ODM. This is not surprising since CODM is benefitting from use of memory for file system buffer space.

CPU Comparison:





We see here that CODM drives up CPU usage. This is expected behavior since substantially more throughput is being generated. The results shown here clearly show that CODM can improve performance for single instance Oracle environments.

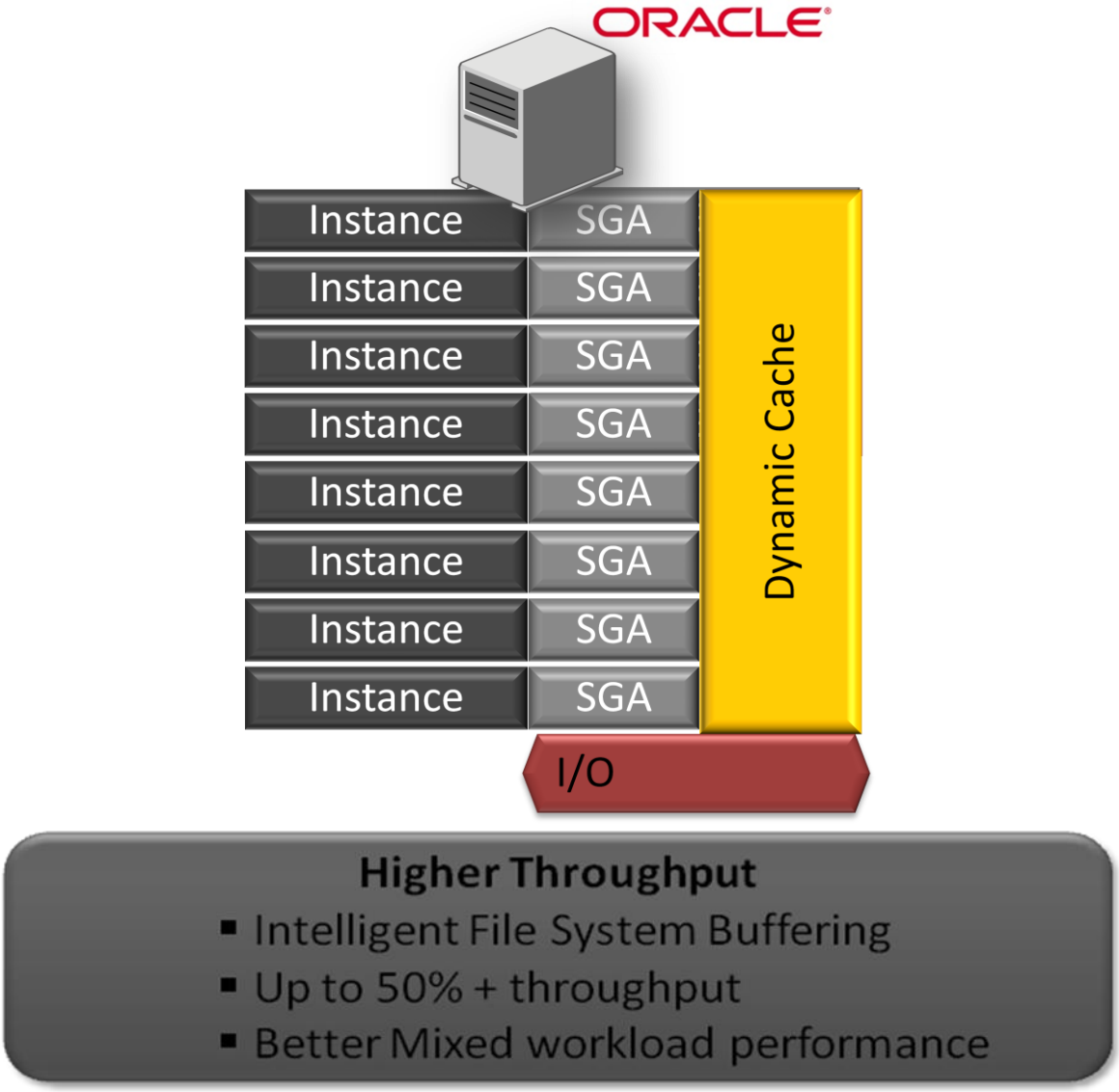
5. MULTIPLE ORACLE INSTANCES ON SERVER

Databases are a major component of the operational IT budget both in terms of manpower (e.g., database administrators or system administrators) and hardware (e.g., database servers). Therefore, the consolidation of databases and database servers is a priority for IT architects. Many IT architects design systems for peak or worst-case scenarios. There are two main reasons for this:

1. No tried-and-tested way currently exists to provide elasticity in both compute and storage capacity to meet the ever changing demands of evolving business processes
2. Insufficient attention is placed on capacity planning.

However, in many database applications, peaks and troughs exist in processing (e.g., online transaction processing [OLTP] systems may be heavily loaded using daylights hours; and data warehouses may be heavily loaded during overnight load and batch reporting windows). CODM is applicable in such mixed workload environments and help architects consolidate database such that the overall license cost of the database environment is reduced.

With such consolidation efforts that are happening today to reduce Oracle licensing costs, it is now becoming very common to run multiple Oracle instances on a single server. So, typically the aggregate Oracle SGA gets sized to consume most of the free memory. As mentioned earlier in this document, these SGA sizes cannot be dynamically changed. With CODM, SGA size can be reduced and the memory freed up can be used as file system buffer cache for all instances. Think of it as secondary SGA. Thus, it is possible to stack more instances per server because the file system buffer cache can be dynamically allocated as the aggregate workload varies over time. The resulting benefit of ODM/CODM is higher throughput because of the ability to improve mixed workload performance and utilizing intelligent file system buffering – enabling CODM for just the files that will benefit from it.



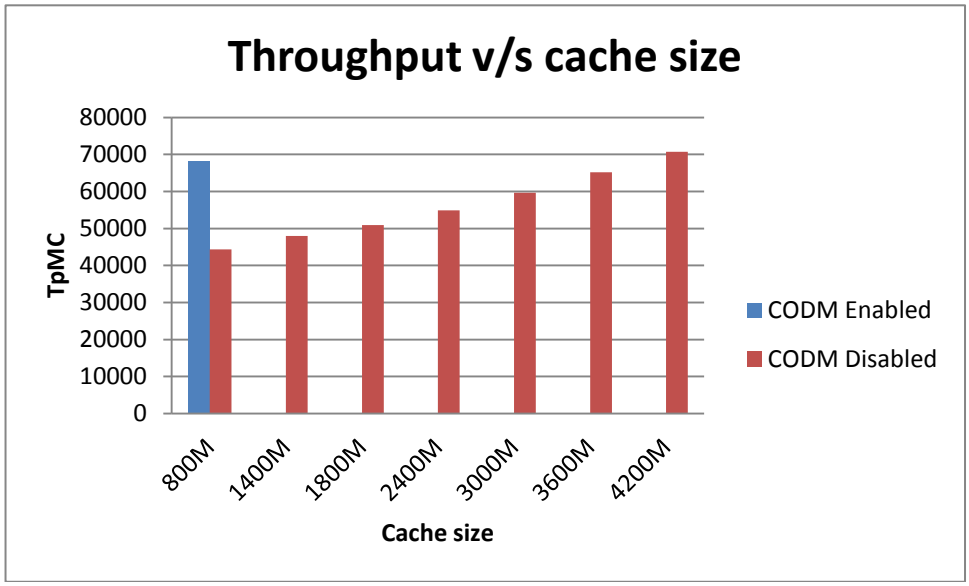
Let us now look at some performance results for the TPC-C benchmark (industry standard benchmark for OLTP workloads) that compare multiple instances performance with just ODM and with ODM/CODM.

Configuration Tested:

- Hardware

  - Server : Sun M4000(32GB,4*Quadcore sparcv9(2.15Ghz))
  - Storage: CLARiiON Cx4(105 spindles, 9GB write cache), 31 raid0 LUNs, 26 data luns, 5 log luns
  - Storage Connectivity: 4*4Gbps FC-HBA.
- Software:
  - OS: sol10u9
  - Oracle: 11.2.0.1
  - SF 5.1SP1
- Benchmark
  - TPC-C (fully scaled to 1000 warehouse, ~200 GB size and ~500 files).
- Tested 6, 7, and 8 parallel instances with just one instance active – others in nomount mode
- Concurrent Users = 50
- segmap_percent – 6

Throughput Results:



Looking at the data, CODM gave a throughput of ~67,000 TPM (Transactions per Minute) for an Oracle buffer cache size of 800M (SGA ~2GB as the rest of the SGA consumes ~1.2GB) , with 8 such parallel instances running, and there is still about 8GB of free memory. Without CODM enabled, to get the equivalent performance required a cache size of ~3800M(SGA ~5GB). So, we can definitely stack multile CODM instances ad get comparable performance.

Another important finding is that the single instance ODM configuration has higher data disk service times than the multiple instances CODM configuration, because of much larger disk reads that ODM needs to do as compared to CODM. For the case of single instance ODM (SGA ~5GB) the disk service time was 2.7 milliseconds as compared to 1.9 milliseconds for 8 instances of CODM (SGA ~2GB).

Granted, this particular performance test is an extreme example of dynamic workload variation among multiple Oracle DBMS instances running on a single server – one with very active workload and all of the others idle, but the results clearly indicate that there is significant benefit to being able to size SGA relatively modestly so that there is free memory to provide for dynamic allocation of buffer cache for CODM to use as needed across the multiple active instances.

6. CONCLUSION

We have shown that Cached ODM (CODM) is a valuable enhancement to the ODM capability of Storage Foundation for providing performance enhancements for Oracle DBMS. Administration of CODM is straightforward and can be done dynamically and non-intrusively as it does not require stopping of the Oracle DBMS instance(s).

To use CODM effectively, one must expend some investigative effort to determine what files can benefit from enabling of CODM. This can be done via the Cached ODM Manager and/or by using the Oracle Automated Workload Repository (AWR).

CODM has the potential to provide performance benefits for both 'single instance per server Oracle environments' and 'multiple instances per server Oracle environments'.

This important CODM capability (to selectively enable file system buffering for just those files that can reap performance benefit) should benefit a very high percentage of Oracle DBMS environments.

About Symantec

Symantec is a global leader in providing security, storage and systems management solutions to help businesses and consumers secure and manage their information. Headquartered in Mountain View, Calif., Symantec has operations in 40 countries. More information is available at www.symantec.com.

For specific country offices and contact numbers, please visit our Web site. For product information in the U.S., call toll-free 1 (800) 745 6054.

Symantec Corporation
World Headquarters
350 Ellis Street
Mountain View, CA 94043 USA
+1 (408) 517 8000
1 (800) 721 3934
www.symantec.com

12/12