8

Monitoring and Tuning RMAN

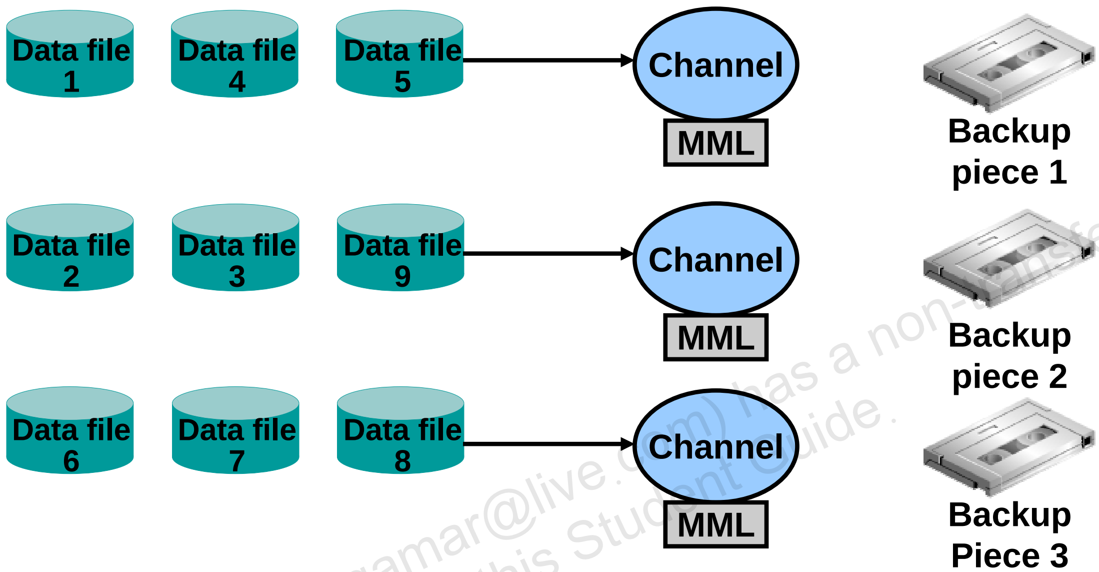
Objectives

After completing this lesson, you should be able to:

- Monitor the progress of RMAN jobs
- Configure RMAN appropriately for asynchronous I/O
- Configure RMAN multiplexing so as to keep tape drives streaming efficiently
- Evaluate the balance between speed of backup versus speed of recovery
- Explain the effect of the following parameters on RMAN performance: `MAXPIECESIZE`, `FILESPERSET`, `MAXOPENFILES`
- Explain how the RMAN `BACKUP DURATION` option can cause backups to either execute faster or take longer, (freeing up resources for other processing)

Parallelization of Backup Sets

For performance, allocate multiple channels and assign files to specific channels.



ORACLE

Parallelization of Backup Sets

You can configure parallel backups by setting the `PARALLELISM` option of the `CONFIGURE` command to greater than 1 or by manually allocating multiple channels. RMAN parallelizes its operation and writes multiple backup sets in parallel. The server sessions divide the work of backing up the specified files.

Example

```
RMAN> RUN {
2>     ALLOCATE CHANNEL c1 DEVICE TYPE sbt;
3>     ALLOCATE CHANNEL c2 DEVICE TYPE sbt;
4>     ALLOCATE CHANNEL c3 DEVICE TYPE sbt;
5>     BACKUP
6>   INCREMENTAL LEVEL = 0
7>   (DATAFILE 1,4,5 CHANNEL c1)
8>   (DATAFILE 2,3,9 CHANNEL c2)
9>   (DATAFILE 6,7,8 CHANNEL c3);
10>   SQL 'ALTER SYSTEM ARCHIVE LOG CURRENT';
11> }
```

Parallelization of Backup Sets (continued)

When backing up data files, you can specify the files to be backed up by either their path name or their file number. For example, the following two commands perform the same action:

```
BACKUP DEVICE TYPE sbt DATAFILE '/home/oracle/system01.dbf';
BACKUP DEVICE TYPE sbt DATAFILE 1;
```

When you create multiple backup sets and allocate multiple channels, RMAN automatically parallelizes its operation and writes multiple backup sets in parallel. The allocated server sessions share the work of backing up the specified data files, control files, and archived redo logs. You cannot stripe a single backup set across multiple channels.

Parallelization of backup sets is achieved by:

- Configuring `PARALLELISM` to greater than 1 or allocating multiple channels
- Specifying many files to back up

Example

- There are nine files that need to be backed up (data files 1 through 9).
- Assign the data files to a backup set so that each set has approximately the same number of data blocks to back up (for efficiency).
  - Data files 1, 4, and 5 are assigned to backup set 1.
  - Data files 2, 3, and 9 are assigned to backup set 2.
  - Data files 6, 7, and 8 are assigned to backup set 3.

Note: You can also use the `FILESPERSET` parameter to limit the number of data files that are included in a backup set.

Monitoring RMAN Sessions

- Query `V$SESSION` and `V$PROCESS` to identify the relationship between server sessions and RMAN channels.
- If you are monitoring multiple sessions, use the `SET COMMAND ID` command to correlate a process with a channel during a backup.

```
SQL> COLUMN CLIENT_INFO FORMAT a30
SQL> COLUMN SID FORMAT 999
SQL> COLUMN SPID FORMAT 9999
SQL> SELECT s.sid, p.spid, s.client_info
  2  FROM v$process p, v$session s
  3  WHERE p.addr = s.paddr
  4  AND CLIENT_INFO LIKE 'rman%';
```


Monitoring RMAN Sessions

To identify which server sessions correspond to which RMAN channels, you can query `V$SESSION` and `V$PROCESS`. The `SPID` column of `V$PROCESS` identifies the operating system ID number for the process or the thread. On UNIX, the `SPID` column shows the process ID. On Windows, the `SPID` column shows the thread ID. There are two basic methods for obtaining this information, depending on whether you have multiple RMAN sessions active concurrently. When only one RMAN session is active, execute the following query on the target database while the RMAN job is running:

```
SQL> COLUMN CLIENT_INFO FORMAT a30
SQL> COLUMN SID FORMAT 999
SQL> COLUMN SPID FORMAT 9999
SQL> SELECT s.sid, p.spid, s.client_info
  2  FROM v$process p, v$session s
  3  WHERE p.addr = s.paddr
  4  AND CLIENT_INFO LIKE 'rman%';
 SID SPID         CLIENT_INFO
---- ------------ ------------------------------
  15 2714         rman channel=ORA_SBT_TAPE_1
  13 2715         rman channel=ORA_SBT_TAPE_2
```

Monitoring RMAN Sessions (continued)

When multiple RMAN sessions are running, it helps to correlate a process with a channel during a backup by using the `SET COMMAND ID` command as shown below:

1. In each session, set the command ID to a different value and then back up the desired object. For example, enter the following in session 1:

```
RUN
{
SET COMMAND ID TO 'sess1';
BACKUP DATABASE;
}
```

   Set the command ID to a string such as `sess2` in the job running in session 2:

```
RUN
{
SET COMMAND ID TO 'sess2';
BACKUP DATABASE;
}
```

2. Start a SQL*Plus session and then query the joined `V$SESSION` and `V$PROCESS` views while the RMAN job is being executed. For example, enter:

```
SELECT SID, SPID, CLIENT_INFO
FROM V$PROCESS p, V$SESSION s
WHERE p.ADDR = s.PADDR
AND CLIENT_INFO LIKE '%id=sess%';
```

   If you run the `SET COMMAND ID` command in the RMAN job, then the `CLIENT_INFO` column is displayed in the following format:

   `id=command_id,rman channel=channel_id`

   For example, the following shows a sample output:

```
SID  SPID         CLIENT_INFO
---- ------------ ------------------------------
11   8358         id=sess1
15   8638         id=sess2
14   8374         id=sess1,rman channel=c1
9    8642         id=sess2,rman channel=c1
```

Monitoring RMAN Job Progress

Monitor the progress of backup and restore operations by querying V$SESSION_LONGOPS.

```
SQL> SELECT OPNAME, CONTEXT, SOFAR, TOTALWORK,
  2  ROUND(SOFAR/TOTALWORK*100,2) "%_COMPLETE"
  3  FROM V$SESSION_LONGOPS
  4  WHERE OPNAME LIKE 'RMAN%'
  5  AND OPNAME NOT LIKE '%aggregate%'
  6  AND TOTALWORK != 0
  7  AND SOFAR <> TOTALWORK;

SID SERIAL# CONTEXT   SOFAR TOTALWORK %_COMPLETE
--- ------- ------- ------- --------- ----------
 13      75       1    9470     15360      61.65
 12      81       1   15871     28160      56.36
```


Monitoring RMAN Job Progress

Monitor the progress of backups, copies, and restores by querying the V$SESSION_LONGOPS view. RMAN uses detail and aggregate rows in V$SESSION_LONGOPS. *Detail rows* describe the files that are being processed by one job step. *Aggregate rows* describe the files that are processed by all job steps in an RMAN command. A job step is the creation or restoration of one backup set or data file copy. The detail rows are updated with every buffer that is read or written during the backup step, so their granularity of update is small. The aggregate rows are updated when each job step is completed, so their granularity of update is large.

Note: Set the STATISTICS_LEVEL parameter to TYPICAL (the default value) or ALL to populate the V$SESSION_LONGOPS view.

The relevant columns in V$SESSION_LONGOPS for RMAN include:

- **OPNAME:** A text description of the row. Detail rows include RMAN:datafile copy, RMAN:full datafile backup, and RMAN:full datafile restore.
- **CONTEXT:** For backup output rows, the value of this column is 2. For all the other rows except proxy copy (which does not update this column), the value is 1.

Monitoring RMAN Job Progress (continued)

- **`SOFAR`:** For image copies, the number of blocks that have been read; for backup input rows, the number of blocks that have been read from the files that are being backed up; for backup output rows, the number of blocks that have been written to the backup piece; for restores, the number of blocks that have been processed to the files that are being restored in this one job step; and for proxy copies, the number of files that have been copied
- **`TOTALWORK`:** For image copies, the total number of blocks in the file; for backup input rows, the total number of blocks to be read from all files that are processed in this job step; for backup output rows, the value is 0 because RMAN does not know how many blocks it will write into any backup piece; for restores, the total number of blocks in all files restored in this job step; and for proxy copies, the total number of files to be copied in this job step

Interpreting RMAN Message Output

RMAN troubleshooting information can be found in:

- RMAN command output
- RMAN trace file
- Alert log
- Oracle server trace file
- `sbtio.log` file

Interpreting RMAN Message Output

The RMAN command output contains actions that are relevant to the RMAN job as well as error messages that are generated by RMAN, the server, and the media vendor. RMAN error messages have an `RMAN-nnnn` prefix. The output is displayed to the terminal (standard output) but can be written to a file by defining the `LOG` option or by shell redirection.

The RMAN trace file contains the `DEBUG` output and is used only when the `TRACE` command option is used.

The alert log contains a chronological log of errors, nondefault initialization parameter settings, and administration operations. Because it records values for overwritten control file records, it can be useful for RMAN maintenance when operating without a recovery catalog.

The Oracle trace file contains detailed output that is generated by Oracle server processes. This file is created when an `ORA-600` or `ORA-3113` (following an `ORA-7445`) error message occurs, whenever RMAN cannot allocate a channel, and when the Media Management Library fails to load. It can be found in `USER_DUMP_DEST`.

The `sbtio.log` file contains vendor-specific information that is written by the media management software and can be found in `USER_DUMP_DEST`. Note that this log does not contain Oracle server or RMAN errors.

Using the DEBUG Option

- The `DEBUG` option is used to:
  - View the PL/SQL that is generated
  - Determine precisely where an RMAN command is hanging or faulting
- The `DEBUG` option is specified at the RMAN prompt or within a run block.
- The `DEBUG` option creates an enormous amount of output, so redirect the output to a trace file:

```
$ rman target / catalog rman/rman debug trace trace.log
```


Using the DEBUG Option

The `DEBUG` option displays all SQL statements that are executed during RMAN compilations and the results of these executions. Any information that is generated by the recovery catalog PL/SQL packages is also displayed. In the following example, the `DEBUG` output is written during the backup of data file 3, but not data file 4:

```
RMAN> run {
      debug on;
      allocate channel c1 type disk;
      backup datafile 3;
      debug off;
      backup datafile 4; }
```

Remember that the `DEBUG` output can be voluminous, so make sure that you have adequate disk space for the trace file. This simple backup session that does not generate any errors creates a trace file that is almost half a megabyte in size:

```
$ rman target / catalog rman/rman debug trace sample.log
RMAN> backup database;
RMAN> host "ls –l sample.log";
-rw-r--r--   1 user02   dba       576270 Apr  6 10:38 sample.log
host command complete
```

Interpreting RMAN Error Stacks

- Read the stack from bottom to top.
- Look for `Additional information`.
- `RMAN-03009` identifies the failed command.

```
RMAN-00571: ===========================================
RMAN-00569: ======= ERROR MESSAGE STACK FOLLOWS =======
RMAN-00571: ===========================================
RMAN-03009: failure of backup command on c1 channel at
               09/04/2001 13:18:19
ORA-19506: failed to create sequential file,
              name="07d36ecp_1_1", parms=""
ORA-27007: failed to open file
SVR4 Error: 2: No such file or directory
Additional information: 7005
Additional information: 1
ORA-19511: Error from media manager layer,error text:
```


Interpreting RMAN Error Stacks

Because of the amount of data that RMAN logs, you may find it difficult to identify the useful messages in the RMAN error stack. Note the following tips and suggestions:

- Because many of the messages in the error stack are not meaningful for troubleshooting, try to identify the one or two errors that are most important.
- Check for a line that says `Additional information` followed by an integer. This line indicates a media management error. The integer that follows refers to code that is explained in the text of the error message.
- Read the messages from bottom to top because this is the order in which RMAN issues the messages. The last one or two errors that are displayed in the stack are often informative.
- Look for the `RMAN-03002` or `RMAN-03009` message immediately following the banner. The `RMAN-03009` is the same as `RMAN-03002` but includes the channel ID. If the failure is related to an RMAN command, then these messages indicate which command failed. The syntax errors generate an `RMAN-00558` error.

Tuning RMAN

- RMAN BACKUP and RESTORE operations perform the following tasks:
  - Read or write data.
  - Process data by copying and validating blocks.
- The slowest of these tasks is referred to as a bottleneck, for any particular process.
- Tuning RMAN requires that the bottlenecks be identified and addressed.
- Performance of backup versus recovery operations can be balanced to suit your needs.

Tuning RMAN

RMAN backup and restore operations perform the following distinct tasks:

- Reading or writing input data
- Processing data by validating and copying blocks from the input to the output buffers

The slowest of these tasks is called a *bottleneck*. RMAN tuning involves identifying the bottleneck (or bottlenecks) and attempting to make it more efficient by using RMAN commands, initialization parameter settings, or adjustments to the physical media. The key to tuning RMAN is in understanding the input/output (I/O). The backup and restore jobs of RMAN use two types of I/O buffers: disk and tertiary storage (usually tape). When performing a backup, RMAN reads input files by using disk buffers and writes the output backup file by using either the disk or the tape buffer. When performing restores, RMAN reverses these roles. I/O can be synchronous and asynchronous. Synchronous devices perform only one I/O task at a time. Therefore, you can easily determine how much time the backup jobs require. In contrast to synchronous I/O (SIO), asynchronous I/O (AIO) can perform more than one task at a time. To tune RMAN effectively, you must thoroughly understand the concepts of synchronous and asynchronous I/O, disk and tape buffers, and channel architecture. With an understanding of these concepts, you can use fixed views to monitor bottlenecks.

Tuning RMAN (continued)

You may be able to take advantage of some backup and recovery features that allow you to balance the performance of backup operations versus recovery operations. For example, if you require shorter recovery time, then you may want to perform image copy recovery on a regular basis. That takes more resources to prepare for recovery, but would lessen the amount of time needed to perform the recovery.

RMAN Multiplexing

- For reads:

Multiplexing Level	Allocation Rule
Level <= 4	1 MB buffers are allocated so that the total buffer size for all input files is 16 MB.
4 < Level <= 8	512 KB are allocated so that the total buffer size for all files is less than 16 MB.
Level > 8	RMAN allocates four 128 KB disk buffers per channel for each file, so that the total size is 512 KB per channel for each file.

- For writes, each channel allocates four output buffers of 1 MB each.

RMAN Multiplexing

RMAN uses two different types of buffers for I/O: disk and tape. RMAN multiplexing determines how RMAN allocates disk buffers. *RMAN multiplexing* is the number of files in a backup read simultaneously and then written to the same backup piece. The degree of multiplexing depends on the `FILESPERSET` parameter of the `BACKUP` command as well as the `MAXOPENFILES` parameter of the `CONFIGURE CHANNEL` command or `ALLOCATE CHANNEL` command. Note: RMAN multiplexing is set at the channel level.

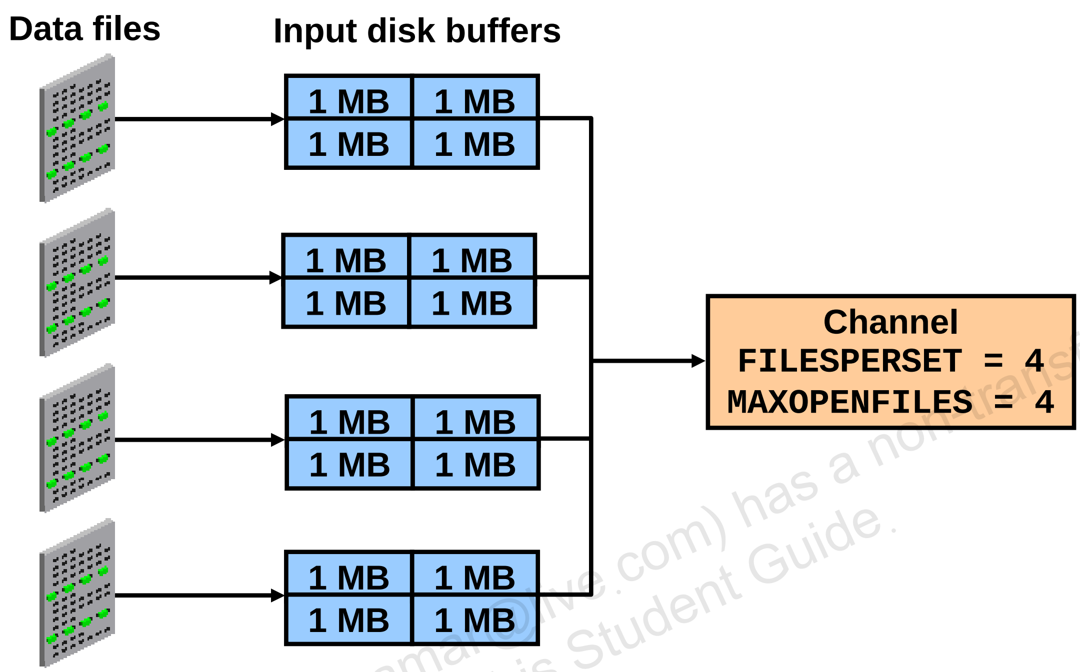
For example, assume that you back up two data files with one channel. You set `FILESPERSET` to `3` and `MAXOPENFILES` to `8`. In this case, the number of files in each backup set is 2 (the lesser of `FILESPERSET` and the files read by each channel) and the level of multiplexing is 2 (the lesser of `MAXOPENFILES` and the number of files in each backup set). When RMAN backs up from disk, it uses the algorithm that is described in the table shown in the slide.

For writing, each channel allocates four output buffers of size 1 MB each.

These buffers are allocated from the PGA unless `DBWR_IO_SLAVES` is set to a nonzero value.

Note: For best recovery performance, do not set `FILESPERSET` to a value greater than 8.

Allocating Disk Buffers: Example



Allocating Disk Buffers: Example

In the example shown in the slide, one channel is backing up four data files. `MAXOPENFILES` is set to 4 and `FILESPERSET` is set to 4. The level of multiplexing is 4 in this example. The total size of the buffers for each data file is 4 MB. To calculate the total size of the buffers that are allocated in a backup set, multiply the total bytes for each data file by the number of data files that are being concurrently accessed by the channel, and then multiply this number by the number of channels.

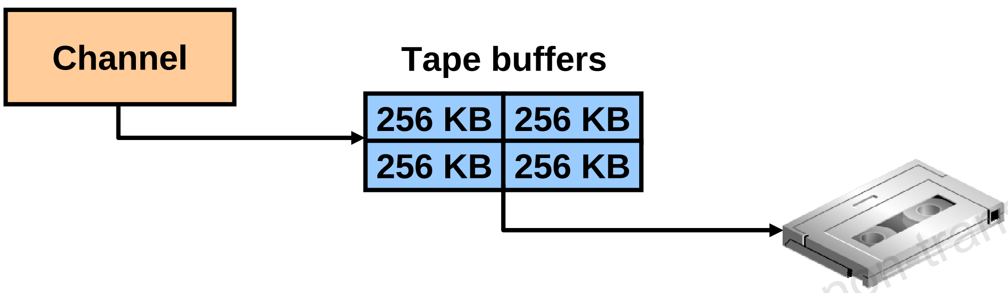
Assume that you use one channel to back up four data files, and use the settings that are shown in the slide. In this case, multiply as follows to obtain the total size of the buffers that are allocated for the backup:

4 MB per data file × 1 channel × 4 data files per channel = 16 MB

Set the `MAXOPENFILES` parameter so that the number of files that are read simultaneously is just enough to use the output device fully. This consideration is important when the output device is a tape.

Allocating Tape Buffers

- From SGA (large pool) with `BACKUP_TAPE_IO_SLAVES` is `TRUE`.
- From PGA with `BACKUP_TAPE_IO_SLAVES` is `FALSE`.



Allocating Tape Buffers

If you make a backup to a tape device, the Oracle server allocates four buffers per channel for the tape writers (or readers if doing a restore). The Oracle server allocates these buffers only if the channel is a System Backup to Tape (SBT) channel. Typically, each tape buffer is 256 KB. To calculate the total size of buffers that are used during a backup or restore, multiply the buffer size by four, and then multiply this product by the number of channels.

As illustrated in the example shown in the slide, assume that you use one tape channel and each buffer is 256 KB. In this case, the total size of buffers that are used during a backup is as follows:

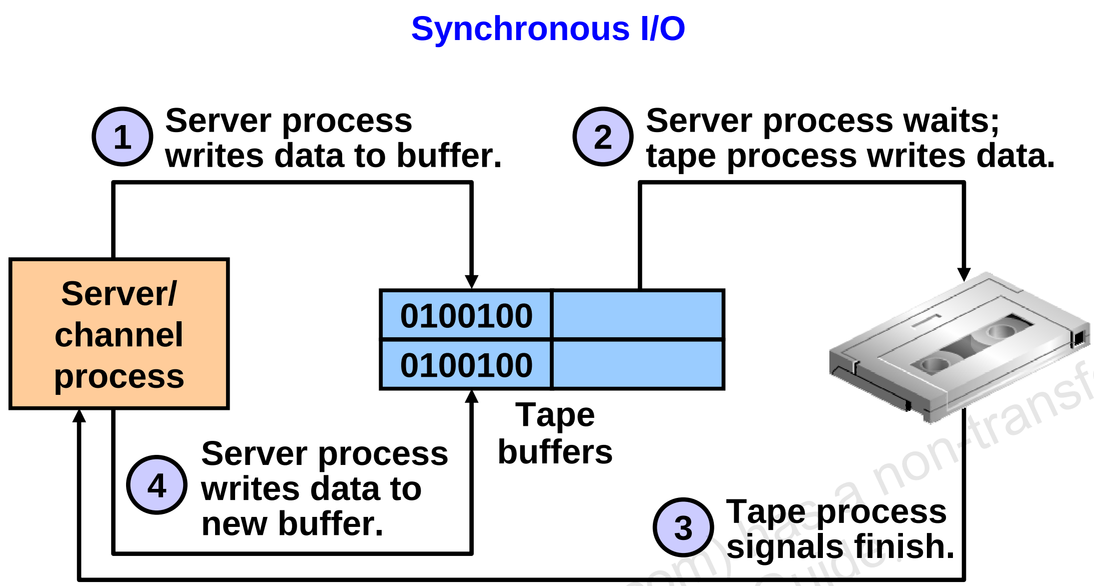
256 KB per buffer × 4 buffers per channel × 1 channel = 1,024 KB

RMAN allocates the tape buffers in the System Global Area (SGA) or the Program Global Area (PGA), depending on whether I/O slaves are used. If the `BACKUP_TAPE_IO_SLAVES` initialization parameter is set to `TRUE`, RMAN allocates tape buffers from the shared pool or the large pool if the `LARGE_POOL_SIZE` initialization parameter is set. If you set the parameter to `FALSE`, RMAN allocates the buffers from the PGA. If you use I/O slaves, set the `LARGE_POOL_SIZE` initialization parameter to set aside SGA memory that is dedicated to holding these large memory allocations. By doing this, the RMAN I/O buffers do not compete with the library cache for shared pool memory.

Allocating Tape Buffers (continued)

Oracle recommends that you set the BACKUP_TAPE_IO_SLAVES initialization parameter to TRUE. In most circumstances, this will provide the best performance of backups to tape. Also, this setting is required in order to perform duplexed backups. Duplexed backups are covered in the lesson titled “Using RMAN to Create Backups.”

Comparing Synchronous and Asynchronous I/O



Comparing Synchronous and Asynchronous I/O

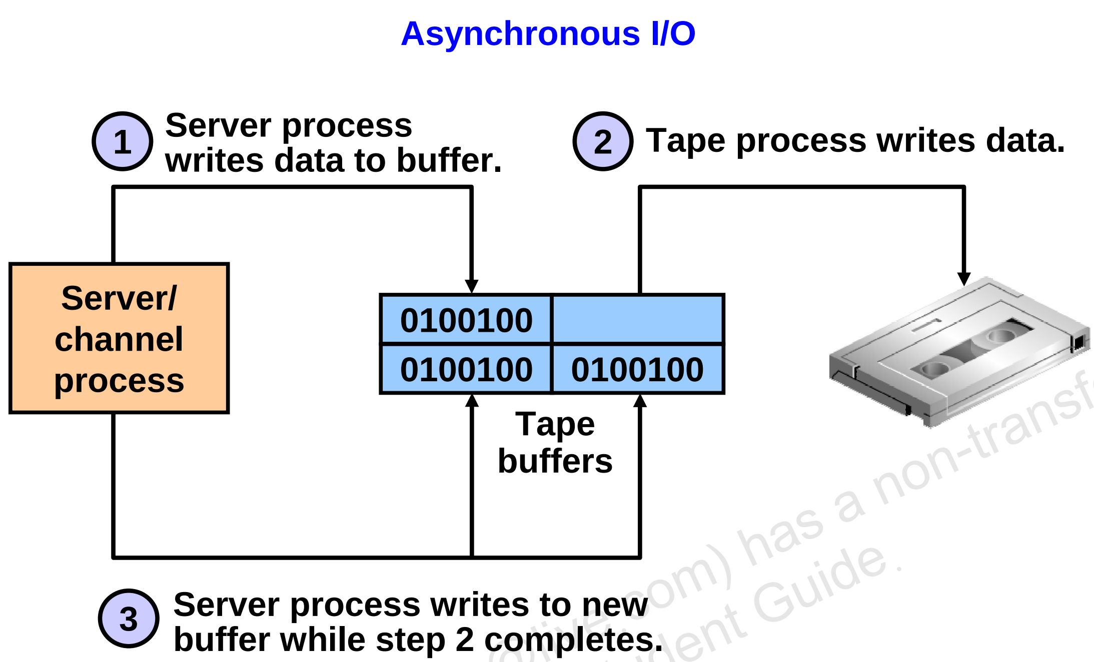
When RMAN reads or writes data, the I/O is either synchronous or asynchronous. When the I/O is synchronous, a server process can perform only one task at a time. When it is asynchronous, a server process can begin an I/O and then perform other tasks while waiting for the I/O to complete. It can also begin multiple I/O operations before waiting for the first to complete.

You can set initialization parameters that determine the type of I/O. If you set `BACKUP_TAPE_IO_SLAVES` to `TRUE`, the tape I/O is asynchronous. Otherwise, the I/O is synchronous.

The example in the slide shows synchronous I/O in a backup to tape. The following steps occur in a synchronous transfer:

1. A server process writes blocks to a tape buffer.
2. The tape process writes data to tape. The server process is idle while the media manager copies data from the Oracle buffers to the media manager’s internal buffers.
3. The tape process relays to the server process that it has completed writing.
4. The server process can initiate a new task.

Comparing Synchronous and Asynchronous I/O



Comparing Synchronous and Asynchronous I/O (continued)

Many operating systems support native asynchronous I/O and Oracle can take advantage of this feature whenever it is available. It is recommended that you always set `BACKUP_TAPE_IO_SLAVES` to `TRUE` when the platform supports it. On operating systems that do not support native asynchronous I/O, Oracle can simulate it by using special I/O slave processes that are dedicated to performing I/O on behalf of another process. You can control disk I/O slaves by setting the `DBWR_IO_SLAVES` parameter to a nonzero value. Oracle allocates four backup disk I/O slaves for any nonzero value of `DBWR_IO_SLAVES`.

The example in the slide illustrates asynchronous I/O in a backup to tape. The steps that occur in an asynchronous exchange are detailed below:

1. A server process writes blocks to a tape buffer.
2. The tape process writes data to the tape. While the tape process is writing, other server processes are free to process more input blocks and fill more output buffers.
3. The spawned server process writes to the tape buffers while the initial tape process writes to the tape.

Monitoring RMAN Job Performance

- The following views can be used to monitor backup and restore performance:
  - `V$BACKUP_SYNC_IO`
  - `V$BACKUP_ASYNC_IO`
- The following rows exist for a backup or restore:
  - One row for each data file
  - One aggregate data file row
  - One row for each backup piece
- Whether or not I/O is synchronous depends on how the controlling process views it.

Monitoring RMAN Job Performance

The maximum backup speed is limited by the available hardware. It is not possible to back up any faster than the aggregate tape bandwidth. One exception to this is if there are many empty blocks in the data files that need not be backed up.

One of the components of the backup system will be a bottleneck—which one depends on the relative speeds of the disk, tape drive, and any other transport components such as the network. As an example, if the bottleneck is the tape drive, and the tape is streaming, then the backup cannot possibly proceed any faster.

Note: If you have synchronous I/O and have set the `BACKUP_DISK_IO_SLAVES` initialization parameter to `TRUE`, I/O is displayed in `V$BACKUP_ASYNC_IO`.

Asynchronous I/O Bottlenecks

- Use V$BACKUP_ASYNC_IO to monitor asynchronous I/O.
- The file that has the largest ratio of LONG_WAITS to IO_COUNT is probably the bottleneck.
  - IO_COUNT: Number of I/Os performed on the file
  - LONG_WAITS: Number of times the backup/restore process told the OS to wait until I/O was complete
- Wait times should be zero to avoid bottlenecks.
  - SHORT_WAIT_TIME_TOTAL
  - LONG_WAIT_TIME_TOTAL

Asynchronous I/O Bottlenecks

You can use V$BACKUP_ASYNC_IO to monitor asynchronous I/O. The LONG_WAITS column shows the number of times the backup or restore process directed the operating system to wait until an I/O was complete. The SHORT_WAITS column shows the number of times the backup/restore process made an operating system call to poll for I/O completion in a nonblocking mode. On some platforms, the asynchronous I/O implementation may cause the calling process to wait for the I/O to complete while performing a nonblocking poll for I/O.

The simplest way to identify the bottleneck is to query V$BACKUP_ASYNC_IO for the data file that has the largest ratio for LONG_WAITS divided by IO_COUNT.

Synchronous I/O Bottlenecks

- Synchronous I/O is considered to be a bottleneck.
- Query the `DISCRETE_BYTES_PER_SECOND` column from `V$BACKUP_SYNC_IO` to view the I/O rate.
  - Compare this rate with the device's maximum rate.
  - If the rate is lower than what the device specifies, this is a tuning opportunity.

Synchronous I/O Bottlenecks

When using synchronous I/O, it can easily be determined how much time the backup jobs require because devices perform only one I/O task at a time. Oracle I/O uses a polling mechanism rather than an interrupt mechanism to determine when each I/O request completes. Because the backup or restore process is not immediately notified of I/O completion by the operating system, you cannot determine the duration of each I/O.

Use `V$BACKUP_SYNC_IO` to determine the source of backup or restore bottlenecks and to determine the progress of backup jobs. `V$BACKUP_SYNC_IO` contains rows when the I/O is synchronous to the process (or thread, on some platforms) that is performing the backup.

Channel Tuning

Use the `CONFIGURE CHANNEL` and `ALLOCATE CHANNEL` commands to:

- Limit the size of backup pieces
- Prevent RMAN from consuming too much disk bandwidth
- Determine the level of multiplexing for each channel
- Configure multiple disks, thus spreading the I/O activity across multiple devices.
- Configure multiple channels on the SBT device, allowing you to assign different data files to each one.

Channel Tuning

You can set various channel limit parameters that apply to operations that are performed by the allocated server session in the `CONFIGURE CHANNEL` and `ALLOCATE CHANNEL` commands.

The `MAXPIECESIZE` parameter specifies the maximum size of a backup piece. Use this parameter to make RMAN create multiple backup pieces in a backup set. RMAN creates each backup piece with a size that is no larger than the value that has been specified in the parameter.

The `RATE` parameter specifies the bytes per second that RMAN reads on the channel. This parameter is useful in preventing RMAN from consuming excessive disk bandwidth and degrading online transaction processing (OLTP) performance. For example, if each disk drive delivers 3 MB per second and you set `RATE=1500K`, some disk bandwidth will still be available to the online system.

The `MAXOPENFILES` parameter determines the maximum number of input files that a backup or copy can have open at a given time. If not set manually, the value defaults to `8`. The level of RMAN multiplexing is partially determined by `MAXOPENFILES`. The level of multiplexing in turn determines how RMAN allocates disk buffers. Multiplexing is the number of input files that are simultaneously read from and then written into the same backup piece.

Channel Tuning (continued)

If you configure multiple channels for an SBT device, then you can specifically spread data files across those channels. Here is an example:

```
RUN
{
   ALLOCATE CHANNEL c1 DEVICE TYPE sbt;
   ALLOCATE CHANNEL c2 DEVICE TYPE sbt;
   ALLOCATE CHANNEL c3 DEVICE TYPE sbt;
   BACKUP (DATAFILE 1,2,5 CHANNEL c1)
                 (DATAFILE 4,6 CHANNEL c2)
                 (DATAFILE 3,7,8 CHANNEL c3);
   BACKUP DATABASE NOT BACKED UP;
}
```

Tuning the BACKUP Command

- MAXPIECESIZE limits the size of each backup piece.
- FILESPERSET prevents RMAN from reading from too many disks at once.
- MAXOPENFILES may inhibit streaming to tape if not set high enough.
- BACKUP DURATION decreases the amount of load on the system that the backup operation causes.

Tuning the BACKUP Command

The MAXPIECESIZE parameter specifies the maximum size of each backup piece created on the channel.

The FILESPERSET parameter specifies the maximum number of files to place in a backup set. If you allocate only one channel, then you can use this parameter to make RMAN create multiple backup sets. For example, if you have 50 input data files and two channels, you can set FILESPERSET=5 to create 10 backup sets. This strategy can prevent you from splitting a backup set among multiple tapes.

The MAXOPENFILES parameter setting depends on your disk subsystem characteristics. If you use ASM, then set it to 1 or 2. Otherwise, if your data is not striped, then you may want to set this higher. To gain performance, increase either the number of files per backup set, or this parameter. If you are not using ASM or striping of any kind, then try increasing MAXOPENFILES.

Tuning the `BACKUP` Command (continued)

The `BACKUP DURATION` option of the `BACKUP` command can be used in different ways. If you specify a shorter duration than needed for the backup to complete, then you can use this to keep the backup activity inside a specific time window. In specific cases, the partial backup that does complete is not lost.

Also, this option has two modifiers:

- **`MINIMIZE TIME:`** The backup runs as fast as possible.
- **`MINIMIZE LOAD:`** The backup attempts to use the full amount of time available in the window. This reduces load on the system.

Tuning RMAN Backup Performance

To tune RMAN backup performance, follow these steps :

1. Remove RATE settings from configured and allocated channels.
2. Set DBWR_IO_SLAVES if you use synchronous disk I/O.
3. Set LARGE_POOL_SIZE.
4. Tune RMAN tape streaming performance bottlenecks.
5. Query V$ views to identify bottlenecks.

Tuning RMAN Backup Performance

Follow this set of steps to obtain the best backup performance:

1. Remove RATE settings from configured and allocated channels. The RATE parameter is used to set the maximum number of bytes (default), kilobytes (K), megabytes (M), or gigabytes (G) that RMAN reads each second on the channel. It sets an upper limit for bytes read so that RMAN does not consume too much disk bandwidth and degrade performance. If your backup is not streaming to tape, ensure that the RATE parameter is not set on the ALLOCATE CHANNEL or CONFIGURE CHANNEL command.
2. Set DBWR_IO_SLAVES if you use synchronous disk I/O. If your disk does not support asynchronous I/O, then try setting the DBWR_IO_SLAVES initialization parameter to a nonzero value. Any nonzero value for DBWR_IO_SLAVES causes a fixed number (four) of disk I/O slaves to be used for backup and restore, simulating asynchronous I/O. If I/O slaves are used, I/O buffers are obtained from the SGA. The large pool is used if configured. Otherwise, the shared pool is used.
   Note: By setting DBWR_IO_SLAVES, the database writer processes will use slaves as well. You may need to increase the value of the PROCESSES initialization parameter.
3. Set LARGE_POOL_SIZE as described on the next page.
4. Tune RMAN tape streaming performance bottlenecks as described later in the lesson.
5. Use V$ views as described earlier in the lesson.

Setting LARGE_POOL_SIZE

- If LARGE_POOL_SIZE is not set, the Oracle server tries to get memory from the shared pool.
- If LARGE_POOL_SIZE is not big enough, the server does not allocate buffers from the shared pool.
- If the server cannot get enough memory, it allocates buffers from the local process memory.
- The Oracle server writes a message to the alert log indicating that synchronous I/O is used for this backup.

```
ksfqxcre: failure to allocate shared memory means sync
I/O will be used whenever async I/O to file not
supported natively
```


Setting LARGE_POOL_SIZE

The requests for contiguous memory allocations from the shared pool are small, usually under 5 KB in size. It is possible that a request for a large contiguous memory allocation can fail or require significant memory housekeeping to release the required amount of contiguous memory. The large pool may be able to satisfy the memory request. The large pool does not have a least recently used list, so Oracle does not attempt to age memory out of the large pool.

Use the LARGE_POOL_SIZE initialization parameter to configure the large pool. Query V$SGASTAT.POOL to see in which pool (shared pool or large pool) the memory for an object resides. The suggested value for LARGE_POOL_SIZE is calculated as:

```
#_of_allocated_channels * (16 MB + (4*size_of_tape_buffer ))
```

For backups to disk, the tape buffer is obviously 0, so set LARGE_POOL_SIZE to 16 MB. For tape backups, the size of a single tape buffer is defined by the RMAN channel parameter BLKSIZE, which defaults to 256 KB. Assume a case in which you are backing up to two tape drives. If the tape buffer size is 256 KB, then set LARGE_POOL_SIZE to 18 MB. If you increase BLKSIZE to 512 KB, then increase LARGE_POOL_SIZE to 20 MB.

Note: The large pool is used only for disk buffers when DBWR_IO_SLAVES > 0 and for tape buffers when BACKUP_TAPE_IO_SLAVES = TRUE. If you are using Automatic Shared Memory Management, the large pool is sized automatically in response to system workload.

Tuning RMAN Tape Streaming Performance Bottlenecks

- Use `BACKUP... VALIDATE` to determine whether tape streaming or disk I/O is the bottleneck.
- Use multiplexing to improve tape streaming with disk bottlenecks.
- Use incremental backups to improve backup performance with tape bottlenecks.

Tuning RMAN Tape Streaming Performance Bottlenecks

To identify and remedy bottlenecks that affect RMAN's performance on tape backups, perform the following actions :

- Use `BACKUP... VALIDATE` to determine whether tape streaming or disk I/O is the bottleneck in a given backup job. Compare the time required to run backup tasks with the time required to run `BACKUP VALIDATE` of the same tasks. `BACKUP VALIDATE` of a backup to tape performs the same disk reads as a real backup but performs no tape I/O. If the time required for the `BACKUP VALIDATE` to tape is significantly less than the time required for a real backup to tape, then writing to tape is the likely bottleneck.
- Use multiplexing to improve tape streaming with disk bottlenecks. In some situations when RMAN is performing a backup to tape, it may not be able to send data blocks to the tape drive fast enough to support streaming. For example, during an incremental backup, RMAN backs up only blocks changed since a previous data file backup as part of the same strategy. If you do not enable change tracking, RMAN must scan entire data files for changed blocks, and fill output buffers as it finds such blocks. If there are not many changed blocks, RMAN may not fill output buffers fast enough to keep the tape drive streaming. You can improve performance by increasing the degree of multiplexing used for backups. This increases the rate at which RMAN fills tape buffers, which makes it more likely that buffers are sent to the media manager fast enough to maintain streaming.

Tuning RMAN Tape Streaming Performance Bottlenecks (continued)

- Use incremental backups to improve backup performance with tape bottlenecks. If writing to tape is the source of a bottleneck for your backups, consider using incremental backups as part of your backup strategy. Incremental level 1 backups write only the changed blocks from data files to tape, so that any bottleneck on writing to tape has less impact on your overall backup strategy. In particular, if tape drives are not locally attached to the node running the database being backed up, then incremental backups can be faster.

Quiz

Select which statements are true about RMAN tuning:

1. You can configure parallel backups by setting the `PARALLELISM` option of the `CONFIGURE` command to greater than 1 or by manually allocating multiple channels.
2. You can stripe a single backup set across multiple channels to improve performance.
3. Whenever you improve the speed of the backup operation, you also automatically improve the speed of the restore and recover operations.

Answer: 1

Quiz

You can never have RMAN bottlenecks because the Tuning Advisor fixes them automatically.

1. True
2. False

Answer: 2

Summary

In this lesson, you should have learned how to:

- Monitor the progress of RMAN jobs
- Configure RMAN appropriately for asynchronous I/O
- Configure RMAN multiplexing so as to keep tape drives streaming efficiently
- Evaluate the balance between speed of backup versus speed of recovery
- Explain the effect of the following parameters on RMAN performance: `MAXPIECESIZE`, `FILESPERSET`, `MAXOPENFILES`
- Explain how the RMAN `BACKUP DURATION` option can cause backups to either execute faster or take longer (freeing up resources for other processing)

Practice 8 Overview: Monitoring and Tuning RMAN

This practice covers the following topics:

- Monitoring RMAN jobs
- Using EM to monitor RMAN