TECHNICAL NOTE

GFC_DEFRAG: FREE SPACE DEFRAGMENTATION UTILITY PACKAGE

Prepared By David Kurtz, Go-Faster Consultancy Ltd.
Technical Note
Version 1.01
Tuesday 5 February 2013
(E-mail: david.kurtz@go-faster.co.uk, telephone +44-7771-760660)
File: gfc_defrag.docx, 5 February 2013

Contents

Introduction

This document discusses some ideas for reducing the physical size of a database by defragmenting free space such that it effectively moves to the end of a data file from which it can be removed by resizing the data file.

This will be necessary after purge, archive and compression activities that occur during archiving. These are separate to the archiving processes because they do not logically affect any data in the database. Rather archiving affords the DBA team the opportunity to conduct these additional actions.

- Although table compression will reduce the size of table segments, it will leave large amounts of free space in the middle of a data file[1]. In order to reduce the overall database size, we need to be able move that free space to end of the data file and trim it off by resizing the data file.

This document also explains the use of the GFC_DEFRAG package has been provided to assist with these operations.

- The HIGH_BLOCK_SEGMENTS program can be used to shuffle free space to the ends of data files where it can then be trimmed with the TRIM_FREE_SPACE program.

[1] A favourite analogy for this is that of a swiss cheese with holes. The holes take up space in the cheese. We could reduce the overall size of the cheese, by filling the holes up with cheese taken from the outside. This is neither possible nor desirable with a real physical piece of Emmental, but it is with a data file!

GFC_DEFRAG package

This package contains programs that dynamically generate and execute DDL to move segments appropriate tables and set store options as defined in the meta-data for the GFC_PSPART package. The packge is owned by SYS to deliberately override any permission issues and to ensure that only a user with SYSDBA permission can execute it.

The package contains the following sub-programs:

- HIGH_BLOCK_SEGMENTS (see page 4)

Instrumentation

Each program in the package uses DBMS_APPLICATION_INFO to set the module and action. These values can be used to track the progress of the package in Oracle Enterprise Manager etc.

- Module is set to the name of the package followed by the name of the program. For example, while the HIGH_BLOCK_SEGMENTS package is running it would be set to GFC_DEFRAG.HIGH_BLOCK_SEGMENTS.
- Action is initially set to the list of parameters passed to the program. As the programs move on to rebuilding segments the action is updated to the name of the segment.

HIGH_BLOCK_SEGMENTS

This procedure rebuilds tables, lob segments and index segments in an order such that objects at the end of a data file are moved nearer the bottom of a tablespace thus resulting in all the free space in a data file moving to the end of the data file where it can be trimmed off with the TRIM_FREE_SPACE program in this package.

Syntax

```
PROCEDURE high_block_segments
      (p_tablespace  VARCHAR2 DEFAULT '%'
      ,p_file_id     NUMBER   DEFAULT NULL
      ,p_relfileno   NUMBER   DEFAULT NULL
      ,p_updateind   BOOLEAN  DEFAULT TRUE
      ,p_statsjob    BOOLEAN  DEFAULT TRUE
      ,p_resize      BOOLEAN  DEFAULT TRUE
      ,p_max_parts   INTEGER  DEFAULT NULL
      ,p_testmode    BOOLEAN  DEFAULT FALSE);
```

Parameters

Parameter	Description
p_tablespace	Name of tablespace to process.
p_file_id	File ID to process. If null all files will be considerd.
p_relfileno	Relative file ID to process. If null assume p_relfileno is the same value as p_file_id. If both p_relfileno and p_file_id are null then files will be considered.
p_updateind	If this parameter is true the ALTER TABLE … MOVE PARTITION command includes the UPDATE INDEXES command. Otherwise indexes or index partitions will be invalidated. The default value of the parameter is TRUE.
p_statsjob	If this parameter is true a job will be submitted to the database job scheduler to collect statistics on the table partition and its indexes. The default value is TRUE.
p_resize	If this parameter is true the package resizes the data file to the previous maximum block occupied by the segment that it has just rebuilt. The default value is TRUE.
p_max_parts	If specified, the number of partitions processed by the program will be limited to the value of this parameter. The default value is null, in which case there is no limit.
p_testmode	If this parameter is TRUE then the package will not submit the DDL that it generates to the database, it will only be printed through *dbms_output*.

The driving query, initially based on the query in the DBA_EXTENTS view, determines the maximum block ID in each segment. Rows are retrieved in descending order of maximum block ID within each tablespace and data file. Rows are not retrieved by the query if the volume of free space below the maximum block ID of the segment in the same data file, or in any other data file is smaller than the segment.

Having rebuilt the object it trims the data file down to the now vacated maximum block ID. Despsite this, the procedure will usually leave some free space that has to be separately trimmed.

Parallelism of rebuild is limited to the number extents in the segment being rebuilt by explicitly specifying the degree of parallelism on the DDL. This prevents additional extents being created thus expanding the object. If the number of extents is greater or equal to the default parallelism (the product of the Oracle initialisation parameters *cpu_count* and *parallel_threads_per_cpu*) the degree of parallelism is not specified. If there is only one extent parallelism is explicitly disabled.

The algorithm works well for single data files tablespaces. However, in some cases can arise.

- If the tablespace consists of multiple data files it is difficult to predict the space allocation. An object might have extents in other data files, and Oracle may put all the extents in the rebuilt objects into the current data file. This can result in an error when the package tries to trim the free space to the maximum block ID in the current object. In this case the error is not trapped and package will terminate. However, this is a reasonable behaviour because the data file has grown rather than shrunk and there is no point continuing.

```
ORA-03297: file contains used data beyond requested RESIZE value
ORA-06512: at "SYS.GFC_DEFRAG", line …
```

    - When this error occurs on a tablespace with multiple data files, try processing all the data files before returning to the one that failed. If you persistently get this error suppress the resize by setting the p_resize parameter to FALSE and see if a subsequent TRIM_FREE_SPACE reclaims the space.
- If a table contains a BLOB or CLOB, then rebuilding it might also rebuild the lob segment and lob index. This may consume more free space than expected causing the same error when it comes to trim off the free space.

The program is driven by a single read-consistent query. The actual space allocation of the database as a number of objects are rebuilt will progressively diverge from the estimated allocation assumed by the package. Therefore, it will be necessary to work iteratively. It is my understanding that in a multi-datafile tablespace, Oracle will a new extent in the lowest possible block ID available in any of the datafiles.

Recommendation:

- Work datafile by datafile, perhaps starting with the data files that have the most free space.
- If the process runs to success on a datafile it is worth running it again to check that there is no more space to be reclaimed.
- If the process does generate ORA-3297 on a multi-datafile tablespace then move on to a different datafile and come back to this one.
- To permit parallel rebuild of indexes during table rebuilds, the PSFT_DDL_LOCK trigger should be temporarily disabled.

The following query reports the data files with the most free space and they highest free block ID.

```
SET LINES 70
TTITLE 'Top 20 Data Files by Free Space'
COLUMN  ranking1        HEADING '#'                FORMAT 999
COLUMN  Mb              HEADING 'Total|Free|MB'    FORMAT 99999.9
COLUMN  max_free_Mb     HEADING 'Max|Free|MB'      FORMAT 99999.9
COLUMN  max_free_block  HEADING 'Max|Free|Block'   FORMAT 9999999
COLUMN  largest_free_MB HEADING 'Largest|Free|MB'  FORMAT 99999.9
COLUMN  file_ID         HEADING 'File|ID'          FORMAT 9999
COLUMN  num_free        HEADING 'Num|Free|Spaces'  FORMAT 9999
COLUMN  tablespace_name HEADING 'Tablespace|Name'  FORMAT a14
WITH x as (SELECT tablespace_name, file_id
, COUNT(*) num_free
, SUM(bytes)/1024/1024 Mb
, MAX(bytes)/1024/1024 largest_free_MB
, MAX(block_id)*8/1024 max_free_mb
, MAX(block_id) max_free_block
FROM dba_free_space
GROUP BY tablespace_name, file_id
), y as (
SELECT
  ROW_NUMBER() OVER (ORDER BY mb desc) as ranking1
, x.* from x)
SELECT * FROM y
WHERE ranking1 <= 20
ORDER BY ranking1
/
TTITLE OFF
```

These are the data files with the most free space, and where this program will have the greatest saving.

```
Sun Jul 15                                page
         Top 20 Data Files by Free Space

                           Num    Total      Max
     Tablespace    File   Free     Free     Free
   # Name            ID Spaces       MB       MB
---- ------------ ----- ------ -------- --------
   1 PSIMAGE        461     15  22985.2  23115.0
   2 PSINDEX        459    199  18374.8  28542.9
   3 PSIMAGE        505     12  17943.4  18131.7
   4 TL2011M01TAB   410    269  17057.0  19035.1
   5 PTAMSG          92      9  16963.2  16967.0
   6 TL2010M08TAB   400    299  16577.0  18628.1
   7 TL2011M07TAB   422    225  16345.0  18262.1
   8 TL2010M12TAB   408    204  15878.0  17584.1
   9 TL2011M04TAB   416    199  15865.0  17502.1
  10 TL2010M09TAB   402    179  15543.0  17193.1
  11 TL2010M07TAB   398      8  14587.0  15907.1
  12 TL2011M05TAB   418    280  12821.0  14700.1
  13 TL2011M10TAB   428    270  12524.0  14460.1
  14 TL2011M02TAB   412    143  12075.0  13656.1
  15 TL2010M11TAB   406    204  11979.0  13667.1
  16 TL2011M11TAB   430    151  11759.0  13133.1
  17 TL2011M06TAB   420    151  11748.0  13132.1
  18 USERS            4    178  11471.1  32170.9
  19 PSIMAGE        495     17  11096.9  11227.8
  20 TLAPP          117    170  10041.8  11718.1
```

Therefore it would make sense to work down this list of datafile.

```
execute GFC_DEFRAG.high_block_segments(p_tablespace=>'PSIMAGE',p_file_id=>461);
execute GFC_DEFRAG.high_block_segments(p_tablespace=>'PSINDEX',p_file_id=>459);
execute GFC_DEFRAG.high_block_segments(p_tablespace=>'TL2011M01TAB');
…
```

TRIM_FREE_SPACE

This procedure trims free space at the ends of data files to reduce the size of the database. It is quite slow to scan all the used blocks in a data blocks. This is also why the DBA_EXTENTS view performs so slowly in 10g. However, it is relatively easy to scan the free space map using the DBA_FREE_SPACE view. This procedure works through the entries in DBA_FREE_SPACE for selected data files in descending block ID order checking whether the end of the free extent matches the end of the file or the start of the previous free extent and resizes the data file as it goes.

Note:

- DBA_FREE_SPACE reports free space in 4Gb chunks (in an 8K tablespace). This makes it difficult to trim free space with a purely SQL approach. However, it is easy to test with PL/SQL.
- Simply dropping objects does not reduce the size of the RMAN backup because the blocks are still formatted and contain an SCN.

Syntax

```
PROCEDURE trim_free_space
      (p_tablespace  VARCHAR2 DEFAULT '%'
      ,p_file_id     NUMBER   DEFAULT NULL
      ,p_testmode    BOOLEAN  DEFAULT FALSE);
```

Parameters

Parameter	Description
p_tablespace	Name of tablespace to process.
p_file_id	File ID to process. If null all files will be considerd.
p_testmode	If this parameter is TRUE then the package will not submit the DDL that it generates to the database, it will only be printed through *dbms_output*.

Sample Output

This shows the procedure processing file 120 which is part the WAAPP tablespace. It is 128 blocks, or 1024Mb in size. There is a free space extent of 120 blocks from block 9 to block 128. So we could trim tablespace down to 8 blocks. Except that the minimum extent size is 64Kb so the minimum size of the tablespace is twice that.

```
…
00:47:32 18.07.2012:Tablespace: WAAPP, file: 120, 128 blocks, 120 user blocks, 1024KB
00:47:32 18.07.2012:Free space 1: 120 @ block id: 9-128, next free block id:
00:47:32 18.07.2012:ALTER DATABASE DATAFILE '+DATA/xxxxxx1a/datafile/waapp.1203.785636623' RESIZE 128K
…
```

In this example there are two contiguous free space extents. The first from 507913 to 525000 is adjacent to the end of data file 450 and then a second from 47369 to 507912. So we can resize the data file to 378952K. The next free space is much lower down the data file at 26889 to 39176 and is therefore not contingous so processing of this data stops at this point.

```
…
07:16:08 18.07.2012:Tablespace: PSIMAGE, file: 450, 525001 blocks, 524992 user blocks, 4200008KB
07:16:08 18.07.2012:Free space 1: 17088 @ block id: 507913-525000, next free block id:
07:16:08 18.07.2012:ALTER DATABASE DATAFILE '+DATA/xxxxxx1a/datafile/psimage.861.785589779' RESIZE
4063304K
07:16:11 18.07.2012:Free space 2: 460544 @ block id: 47369-507912, next free block id: 507913
07:16:11 18.07.2012:ALTER DATABASE DATAFILE '+DATA/xxxxxx1a/datafile/psimage.861.785589779' RESIZE
378952K
07:16:13 18.07.2012:Free space 3: 12288 @ block id: 26889-39176, next free block id: 47369
…
```