My Experience with Freelists

Timothy S. Boles
Email: tboles@ma.rr.com
Version 1 – May 2007

The Problem:

About 5 years ago I was on a system that was experiencing unaccounted for growth on a couple of tables. This system was using an 8.1.7 Database and 8 KB block sizes. The tables were basically used as a temporary hold by the application to keep track of transactions that were being processed by the system. The insertions and deletions were basically one to one and therefore you would expect no growth on the table. However, this was not the case. I found that although the number of rows in the table hovered around 100 the tablespace usage continued to grow. In other words, the High Water Mark continued to move up although there was plenty of space below it to be reused. When I became aware of the problem the team was already mitigating it by doing an export/truncation/import process during every maintenance window (once a quarter) to reclaim the extra space the table was acquiring. I was tasked to investigate the problem and provide recommendations to alleviate the problem.

The Investigation:

I began my investigation by looking at the mechanism that Oracle uses to keep track of and reuse space. I knew the High Water Mark was used to keep track of the highest mark of tablespace usage. I also knew Oracle attempts to track space usage through a mechanism called 'freelist'. The basic idea is that a freelist is a linked list of datablocks having enough space in them to accept new rows. Once the block has enough rows so that PCTFREE is meet it is removed from the freelist. When enough space has been removed from the block and it meets the PCTUSED criteria it is placed back on the freelist.

So I felt that my research should concentrate on determining the proper settings for PCTFREE, PCTUSED and FREELISTS for the two tables. The first table ***ALLCHAIN*** contains only transactions that involve a chained row (insert >8 KB) and the ***FEWCHAIN*** table contained less than 6% chained rows. In general LONG data type columns have a capacity to hold 2 GB of characters but the application limit was 15000 characters (14.6 KB) of data for each insertion. Taking into account that a datablock reserves some space for management information rather than data and there are other columns the rows inserted into the ***FEWCHAIN*** table could be almost twice as large as an Oracle datablock (8 KB).

I worked with the developers to get a good understanding of the purpose of the table, the type of information they were submitting and see if they had any useful tools to use to investigate the problem. I found that the information in these tables was only inserted during the initial processing of the transaction and then deleted once all software processing was complete, it was never updated. The tables did have a LONG datatypes, however the application limited the amount of space inserted into that column to 15000 characters (14.6 KB) of data. They also had a tool to mimic the inserts and deletes done by the software for testing. After working with the developers I looked closer at the tables and found a couple of interesting points.

I found that they both were using only one freelist and that the PCTUSED was set at 80% and PCTFREE was set at 20%. In layman's terms, when a datablock had less than 6554 bytes used then it was placed on the freelist and when it had less than 1638 bytes free it was removed from the freelist.

I did some investigation in the Oracle documentation, MetaLink and Google. I found the following informative documents:

- "Questions and Answers Data Blocks and Freelists" at http://www.ixora.com.au/q+a/datablock.html
- "FREELISTS AND FREELIST GROUPS" http://metalink.oracle.com Doc ID 1029850.6
- "Freelist Management with Oracle 8i" http://metalink.oracle.com Doc Id 157250.1

These documents were very informative and talked about things like Master freelist, Transaction freelist, Process freelists, freelist chain, block head, block tail, block structures and other very informative things. I would suggest reading them.

A very simplistic view of this process is that for each transaction Oracle checks (by default) the first five datablocks on the freelist and determines if any of them contain enough space for the transaction. As Oracle progresses through the freelist any blocks that do not have enough space to accommodate the row are removed from the freelist. The only way these blocks can now be moved back to the freelist is if an update or delete occurs on them. If any datablock within the first 5 has enough space available then the row is added to that datablock. If there is not enough space on any of the first five datablocks then they are removed from the freelist and 5 new datablocks are added by bumping up the High Water mark (by 5 datablocks by default) and allocating the new space.

Early Speculation

I speculated that the freelist mechanism that Oracles uses for managing free space within a table does not work very well when the table has columns with "large" datatypes (LONG and LONG RAW) and the data is volatile. My initial supposition was that LONG type columns holding more the 8 KB of data will in effect cause the tablespace to continue to grow infinitely because the row will always contain more data than the free space in a datablock and force an increase of the HWM. Therefore rows that are larger than one block in size will always cause the high Water Mark to be increased and therefore more space to be added to the table. ***It turned out that this very basic view of the situation was not entirely correct.***

To give speculate is all well and good, but I was supposed to give recommendations. The recommendations should be based on reading as well as testing.

Gathering My thoughts:

On the ***ALLCHAIN*** table, every transaction involves a row that is larger than 1 datablock. This means at every transaction will cause the high water mark to continue to grow and no reuse of space unless there is a method of chaining datablocks in the freelist on an insert. Or perhaps reorganizing the space lowering the high water mark after the rows on the blocks immediate below it are deleted. The only methods that I have found for reorganizing the table in this manner is by exporting the rows, truncating the table and importing the rows or by executing the

ALTER TABLE MOVE command.

I felt however that on the ***FEWCHAIN*** table most of the transactions are not larger than 1 datablock. Therefore it may be possible to curtail this type of growth but it will still occur.

An interesting point is that the ***FEWCHAIN*** table had a freelist set to 5 and it seemed to help better utilize free space. I found that this in my opinion conflicted with the information I had read so far. The Oracle Metalink articles 157250.1 FREELIST MANAGEMENT WITH ORACLE 8I and 1029850.6 FREELISTS AND FREELIST GROUPS make statements like "When using multiple freelists, the amount of unused space within datablocks can increase." and "Partitioning the Free Space in multiple freelists is used essentially to improve the performance of OLTP applications with a high degree of concurrent transactions…..It is important to note that this improves the inserts performance, but wastes disk space usage."

Testing the Hypothesis

One thing that I have tried to learn is not to reinvent the wheel (unless I want to know how it works). I worked with the developers on site and see what tools they have access to that might make my life easier. In this case I found a developer whom had created two offline application functions to "simulate" the applications method of inserting and deleting rows. In this manner I could be assured that I would still see the type of activity and problems that was occurring on the production system. The first function was ADD_FEWCHAIN_ROWS which would take two parameters *number_of_rows* and *size_of_string/100.* The second function is DELETE_FEWCHAIN_ROW which takes one parameter of FEWCHAIN_RPT_SEQ_NUMBER which is a primary key on the ***FEWCHAIN*** table.

The method for testing the hypotheses was:

1. Take baseline table statistics including PCTFREE, PCTUSED
2. Complete multiple inserts and deletes using the functions ADD_FEWCHAIN_ROWS and DELETE_FEWCHAIN_ROW. Finish with approximately 100 rows in table.
3. Record Freelists Statistics. Include the following information.
    a. STATEMENT/s (the statement executed before recording of statistics.)
    b. BLOCKS in the freelist (the datablock address)
    c. The datablock address with the Relative File Number –Block Number – File Number
    d. HEX size, Decimal Size, Number of Free Blocks, High Water Mark, Number of Rows, Average Size of Rows
4. Add a few LARGE size rows.
5. Record Statistics (see #3)
6. Loop through steps 2 to 5 until enough data has been gathered.
7. Alter Storage of Table to include more freelists and Loop through steps 2 to 6.

Results of Test

The actual process of dumping the data files and analyzing the information was so time consuming that only a small sample set was gathered. In the initial tests only information on ***available space*** within a datablock was gathered. During the initial analysis I decided that a

truer picture of what was occurring might be found by looking at the **total free space** within a data block.

Freelist storage parameter = 1

In first test the freelist storage parameter was set to 1. One hundred and thirteen inserts and 13 consecutive deletes were performed on this table. Afterwards, baseline statistics were gathered as outlined above. Thereafter for each insertion or deletion statistics where gathered and behavior noted. The same operation was never performed more than twice in a row. The size for the record ranged from 1200 bytes to over 15000 bytes. The behavior observed was for the most part what was expected as outlined in MetaLink notes 157250.1 and 1029850.6, however two inserts (see Table 1 : Normal Activity Freelist 1) showed some interesting points.

Table 1 : Normal Activity Freelist 1

Action	Anticipated Result	Actual Result
G) Insert row was approximately 2200 bytes, the first datablock has an available space of 2352 bytes.	New row should fit within available space of first datablock.	First datablock is removed from freelist and row inserted into second datablock
O) Insert row is larger than 8192 bytes. (Larger than a full datablock)	First 5 datablocks removed, HWM moved and row inserted into newly allocated space.	HWM is not moved. Datablocks are removed from the Master freelist and 5 datablocks are moved from the Transaction Freelist to the Master freelist with the row inserted into the first two blocks.

A second test was conducted concentrating on inserts of greater then 14000 bytes. Again most of the behavior was as expected but shows those transactions that seem to fall outside the rules described by the MetaLink notes.

Action	Anticipated Result	Actual Result
11A) Insert row is larger than 15000 bytes	First 5 datablocks removed, HWM moved and row inserted in newly allocated space.	In addition to the anticipated results the Transaction Freelist blocks are moved to the Master Freelist
12A) Insert row is larger than 15000 bytes.	First 5 datablocks removed, HWM moved and row inserted in newly allocated space.	First 7 datablocks are removed, the Transaction Freelist blocks are moved to the Master freelist, HWM is increased with the row being inserted into the first newly allocated block plus some space from evidently the 2 blocks on the Transaction Freelist that have been removed.
13A) Insert row is larger then 14000 bytes	First 5 datablocks removed, HWM moved and row inserted in newly allocated space.	The HWM is **NOT** moved. The freespace within the first 4 blocks (16021 bytes available) is evidently used plus some from the 5th block.

I continued this process for 2 more tests (inserts around 4000 bytes and inserts arounc 10000 bytes) with the freelist set to 1 and noted the following observations in general.

Observations for Freelist of 1

The behavior of the single freelist slightly differs from the information that I had read within metalink, articles and papers. The following is what from my limited testing and should be in no way construed to be a definitive guide to the subject.

Observation List 1

1. If there are multiple datablocks on the freelist within the search depth that each of free space of ~8000 bytes, Oracle will recognize this an allow them to be combined with other datablocks of ~8000 bytes. [This may be a smaller amount but my limited data showed to be more than 4000 bytes.]
2. When searching the freelist of a datablock with a large amount of free space found (6-7KB+) then the search depth increases.
3. Available space and NOT total space seems to be used for comparing if a row can fit into a datablock at various times.
4. If there is not enough space on a datablock for the row to fit it is immediately removed from the free list (except in cases noted in #1).
5. If the Master free list contains no blocks with enough free space within the search depth of the free list, then the searched blocks are removed (usually the first 5) and the entirety of the transaction freelist is moved to the Master freelist. The search will continue until the appropriate search depth is reached. [This can be combined with #2 to allow greater search depth.]

These observations I combine into a behavior that I call the TOWER EFFECT.

Tower Effect

The behavior of the single free list that moves the entire Transaction Freelist to the Master freelist could cause a "tower" effect. The "tower" effect is a gradual build up of datablocks on the free list which prevent the checking/use of blocks on the lower levels of the "tower". This can be caused by intermittent "large" (+8KB) transactions that add more blocks to the Master free list than are removed. This can happen after the default 5 block search on the free list produces no results and all blocks on the Transaction Freelist are moved to the master Freelist, as well as adding "fresh" blocks with the move of the High Water mark. A good example was observed when there were initially 13 datablocks on the Master Freelist. A row of 14000+ bytes was inserted into the table and caused the following: (See Table 2)

1. The scan depth is increased from 5 to 7 because of encountering a freelist block with more than 6 KB.
2. Block 2 (D15) held in reserve.
3. The first 7 blocks removed from the Master Freelist.
4. The Transaction Freelist which contained 10 blocks was moved to the Master Freelist.
5. High Water Marked moved 5 datablocks and those are added to the Master Freelist.
6. Transaction take up approximately 2 datablocks (using D15 and one of the new ones added by moving the HWM.

So 13 blocks on the Master Freelist is now 20 (13– 7+ 10 + 5 – 1 = 20). If this situation occurs frequently enough then this "tower" could continue to grow and the datablocks on the base would never be checked.

Table 2 shows a free list at three different points in time. Freelist A is before the row insertion takes place, Freelist X is after blocks are moved/removed from the freelist and the HWM is moved, Freelist B is after the row insertion takes place.

Table 2

	Freelist A Before Row Insertion		Freelist X After HWM move		Freelist B After Insertion Complete	
	Address	Free Space	Address	Free Space	Address	Free Space
HWM	**HWM**	**HWM**	**HWM**	**HWM**	**HWM**	**HWM**
1	D0b	1553	D17	8106	d18	8049
2	D15	8096	D18	8106	d19	8106
3	Cf5	3551	D19	8106	d1a	8106
4	Cf0	4402	D1a	8106	d1b	8106
5	D0e	5051	D1b	8106	Cc7	6047
6	Cfd	3551	Cc7	6047	Ccb	6549
7	Cf8	6549	Ccb	6549	Ccf	5497
8	D04	5849	Ccf	5497	Cd2	8092
9	Cf4	8098	Cd2	8092	Cd9	6547
10	Cff	2902	Cd9	6547	Cda	5048
11	D09	4198	Cda	5048	Cd0	5048
12	Cf9	8096	Cd0	5048	D02	8100
13	Cf1	6545	D02	8100	Ce7	6047
14	D03	4551	Ce7	6047	D03	4551
15	Ce7	6047	D03	4551	D04	5849
16	D02	8100	D04	5849	Cf4	8098
17	Cd0	5048	Cf4	8098	Cff	2902
18	Cda	5048	Cff	2902	D09	4198
19	Cd9	6547	D09	4198	Cf9	8096
20	Cd2	8092	Cf9	8096	Cf1	6545
21	Ccb	6549	Cf1	6545		
22	Ccf	5497				
23	Cc7	6047				

What I concluded is that this type of growth could be curtailed somewhat by adjusting the PCTUSED and PCTFREE to better accommodate the table usage. Since the tables where used only for inserts and deletes, no room was needed for growth. According to some articles on metalink the appropriate settings would be PCTFREE=5% to maximize the number of rows inserted per datablock. The PCTUSED is found by the formula (1-(.05+PCTUSED))*blocksize>AverageRowSize. So for this system it would be (1-(.05+PCTUSED))*8192>2048 => PCTUSED<70%. That means that datablocks would only be put back on the freelist when the space available for an insert was more than 2457 bytes (409 bytes larger than an average row).

Now I needed to understand, why freelist settings of greater than 1 was doing more good than harm.

Freelist storage parameter = 4

Similar tests were conducted with the freelist storage parameter = 4, but only two since so much information had to be shifted through. The following observations listed as 1-4 when the freelist storage parameter =1 were the same however #5 was different.

#5) The Master Freelist is used as a reservoir for all freelist groups. If the assigned freelist group contains no blocks with enough free space within the search depth of the freelist group, then the searched blocks are removed (usually the first 5) and 5 blocks are moved from the Master Freelist to the freelist group. If the Master Freelist does not have adequate number of datablocks then the entirety of the Transaction Freelist is moved to the Master freelist and then datablocks are moved to the appropriate freelist group. The search will continue until the appropriate search depth is reached.

Well then this had to be the answer in some manner since everything else acted the same. I call this behavior the "MINI - TOWERS".

Mini - Towers Effect

Even with the multiple freelist parameter set the entire Transaction Freelist is still moved to the Master Freelist, but the master Freelist is used as a reservoir to the multiple freelist groups and only moves 5 (maybe a little more depending on the search depth has outline in Observations for Freelist of 1 #2) per request to any freelist. The "tower" effect could still be seen but would happen on a much smaller scale since a limited amount of blocks (usually 5) are moved form the Master Freelist to the freelist group (unlike when the parameter was set to 1 and the entire Transaction Freelist was moved to the Master Freelist). This effect would also be spread out across the number of free lists available, causing "mini" towers to be created. These towers should theoretically be smaller than the tower created by a freelist parameter of one, and therefore they should have a greater chance to reach the bottom of towers and reuse more space.

What's the Scoop?

Although the data gathered is limited in size and scope, it does point to some interesting factors that could be the cause for the general increase in the HWM and available free blocks at the same time.

1. The sizing of PCTUSED and PCTFREE. The sizing of PCTUSED needs to be set so that the average (or perhaps median) size row is able to fit into the space available. Although Oracles does combine free space on blocks to meet the row insertion requirements it only happens with the total free space on a block is above a certain threshold, 7-8 KB from our observations.
2. The search depth for available blocks. The search depth is how far the freelist is searched before moving the HWM and adding "fresh" blocks to the top of the the list. Increasing the search depth will help shrink the "tower" effect of the freelist. However, it could also have an adverse effect. For each one found that did not have enough room to accommodate the insert they are removed from the freelist until an update (which does not happen) or a delete occurs. This may increase the number of blocks in a tablespace with enough room for an "average" row addition by not being scanned because they have been removed from the freelist.
3. The movement of all blocks on the Transaction Freelist to the Master Freelist. If you have the freelist parameter set to one, with 10000 blocks on the Transaction Freelist and a search of the Master Freelist has not provided a block with enough space all 10000 blocks could be moved to the Master Freelist. This in affect buries the other blocks on

the Master Freelist and contributes to a “tower” effect.

The Recommendations

The best way to control the use of the freelist is by adjusting the PCTFREE and PCTUSED storage parameters on the tables. According to the information on Oracle MetaLink Note 1029850.6 the formula for setting PCTFREE and PCTUSED is

(100%-PCTFREE+PCTUSED)))*datablock_size>maximum_row_size.

1. Set PCTFREE=5
   The table involves no updates it would be best to leave the PCTFREE at 5% (Oracle recommends not setting it any lower).
2. Set PCTUSED<55
   The maximum row size is at least 15KB for these tables the above formula can not be used. An adjustment was made based on the following The one table actually only has 6% of rows greater than 8KB so the average row size (3210 bytes) was used to determine the PCTUSED from the formula above.
3. Set FREELISTS = 5
   OLTP environments are recommend by Oracle to have the number of freelist groups be equal to the maximum number of concurrent transactions on the table. I asked the developers how many concurrent users they usually saw updating the two tables in questions and the answer was about 5.

What Happened

The recommendations I proposed were implemented during the next maintenance window. We saw a drop in the table growth and the export/truncate/import routine only had to be done once every couple of years instead of once a quarter. If anyone ever wants to see it I still have the 30+ pages of excel spreadsheets that show the statistics that I gathered.
