How to Prevent Database Flooding

The high water mark of a segment can cause surprising effects on a database. What are these and how can you identify them.

By: Andrew Deighton, Labyrinth IT Services Ltd

This article discusses the use of the High Water mark by Oracle and the impact that it can have on database performance. It will possibly help to resolve some annoying performance problems that have been difficult to trace.

Introduction

If you go to the beach, you will see a line of seaweed, shells and such like washed up along the coastline. This is known as the "high water" mark, and indicates the level to which the water rises at high tide. In order to prevent your lunch from being washed away, you would set up your deck chair further up the beach than this line before falling asleep in it.

This has an analogy in Oracle, where the high water mark of a table represents the last block that has held rows of the table, or the maximum level to which data has risen in the table. (And similarly for clusters and indexes.) This mark is important for a number of reasons, and should be monitored to ensure it has a reasonable value.

Unrealistically high values for the high water mark are relatively easy to achieve, and can cause problems that are sometimes difficult to explain.

Performance Problems

I was once asked by a developer to investigate a performance problem with the code he was working on. After tracing the code and analyzing it, I found the offending statement – a full table scan. When I told him, his reply was "But that can't be the problem – the table only has one row!'

Further investigation showed that the table did, in fact, only have one row, but the full table scan was reading 20Mb for every execution. The reason for this being that the table had previously held many more rows, and all bar one had been deleted.

In order to do a full table scan, Oracle needs to read all the blocks that could potentially hold data for the table. This means reading all the blocks from the beginning of the table up to the high water mark, whether the block holds data or not. Conversely, blocks above the high water mark do not need to be read, as there cannot be any data in them.

When data is inserted into a table, causing a new block to be used, the high water mark is adjusted accordingly. If the data is subsequently deleted, causing the block to become empty, the high water mark is not adjusted, but remains pointing to the maximum data position for the table.

After a table has been analyzed, the "BLOCKS" field in USER_TABLES will contain a value. This value represents the number of blocks that have ever held data in the table – that is, the number of blocks between the start of the table and the high water mark. The field "EMPTY_BLOCKS" holds the number of blocks allocated to the table that are above the high water level. There is no easy method of determining the number of empty blocks below the high water mark.

In order to calculate the number of used blocks below the high water mark requires a full table scan:

```
select
  count(
      distinct dbms_rowid.ROWID_RELATIVE_FNO (rowid)||'_'||
          dbms_rowid.ROWID_BLOCK_NUMBER(rowid) )
from
  <<table_name>>;
```

The difference between this number, and the number given by "BLOCKS" in user_tables will show the number of unnecessary blocks being read each time a full table scan is performed on the table.

Storage Considerations

In release 7, Oracle introduced the ability to deallocate unused space from the end of a segment and place it back onto the free lists for use by other segments in the tablespace. This is a step forward from the previous situation, and removed the necessity to drop and re-create a segment that had been created with in-appropriate storage parameters. It is still somewhat restrictive, however, as only space above the high water mark can be deallocated in this way.

```
Alter table <<table_name>> deallocate unused;
```

SQL*Loader considerations

When data is loaded into a table using the SQL*Loader direct path option, the data structures in the SGA are bypassed in order to achieve performance gains. To be able to do this, SQL*Loader builds database blocks itself, and writes them directly to the database. These blocks must therefore be empty before SQL*Loader writes to them, or it would overwrite existing table data. To be able to ensure there is no data already in the blocks being written, direct path SQL*Loader only writes above the high water mark of a table.

Loading above the high water mark does not impact (much) on the performance of the load, but could have severe implications for the processing of the data after the load.

Consider a situation where data is loaded into a work table, processed in some way, and then moved to the final destination. The first time this process is carried out, the table is empty, so data is loaded into the beginning of the first extent and processed. The next time the process runs, data is loaded into the table starting at the first block above the position where the previous load finished. The full scan of the table to complete the processing therefore requires twice as many blocks to be read. In the same way, the next run takes longer and so on. The segment also continues to grow, as the space freed by the deletes is never reused by subsequent loads.

Resetting the High Water mark

The problem of ever-increasing table sizes under direct path SQL*Loader can be overcome by using the TRUNCATE option for the load. This causes the database server to truncate the table before the loading the new data.

Truncating a table removes all the data from the table and resets the relevant pointers, including the high water mark. When a table is truncated, the associated indexes are also re-initialized.

To reset the high water mark without removing the data requires a rebuild, as in:

```
Alter table <<table_name>> move …
```

Conclusion

The high water mark on a table can have a significant impact on the performance of full table scans, as well as affecting storage considerations. It therefore needs to be monitored to ensure that it is appropriate for the use of the table. Significant disk access savings can be made by keeping the high water mark as close to the end of the actual data as possible.

The same applies to indexes, and affects the performance of any index scans.

Andrew Deighton is a database consultant and technical director at Labyrinth IT Services. He has been working with Oracle for 10 years. He can be contacted at andrew@labyrinth-it.co.uk.