

Digging Deeper

Arup Nanda
Proligence, Inc.

Quick Poll

- DBAs
- Developers

This is primarily for DBAs for some specific areas are of interest to developers, too.

STATSPACK Report

Top 5 Timed Events

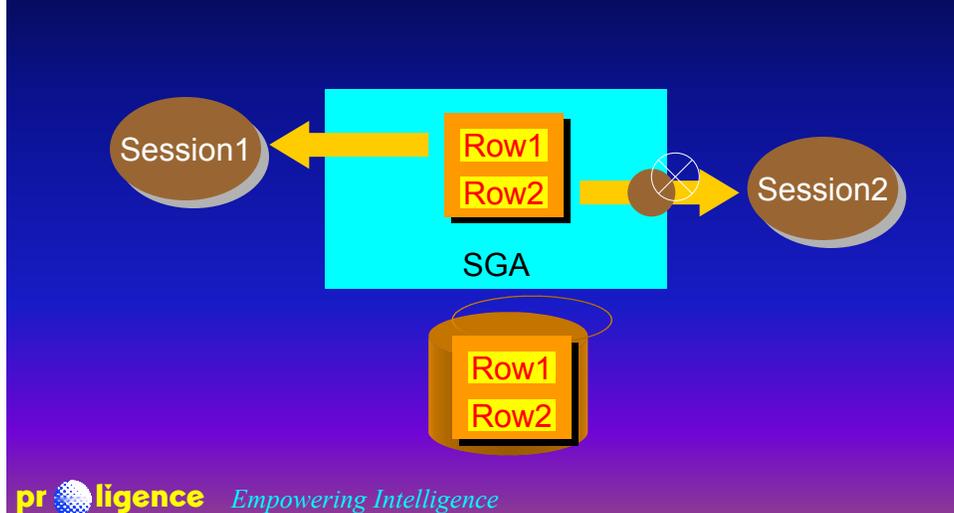
```
~~~~~
```

| Event | Waits | Time (s) | % Total Ela Time |
|-------------------------|-------|----------|---------------------|
| db file scattered read | 2,241 | 731 | 48.73 |
| CPU time | | 312 | 20.80 |
| db file parallel write | 752 | 43 | 2.86 |
| buffer busy waits | 968 | 34 | 2.26 |
| db file sequential read | 1,023 | 32 | 2.13 |

```
~~~~~
```

The STATSPACK report does provide a good snapshot of problems in the database, but at a very macro level, not at the segment level. For instance the excerpt above shows the wait events summarized for the whole instance. It does not show which segments suffered this.

Buffer Busy Waits



Let's examine one of the statistics, buffer busy waits. The common cause of the wait event is the simultaneous access of the same buffer by two sessions.

Session 1 requests the update of Row1. The entire block containing the row Row1 is loaded into the SGA. The block also contains the row Row2. At the same time as the Session1, another session Session2 is trying to update Row2. Since the block containing Row2 is already in the SGA, the server process of the session will try to update the block. But since the session Session1 is modifying the block, it has to wait, creating a buffer busy wait scenario.

The problem can be alleviated by packing the segment less so that the a buffer contains less number of rows, making the it less likely that the buffer will be a contention point for access. The resolutions is thus to reorganize the segment that contributed to the problem. But, what segment? STATSPACK report does not provide the details of the segments.

Event

- Event 10046 can be set to get all waits for a session
- Alter session set events '10046 trace name context forever, level 8'
- You can also use DBMS_SUPPORT

The segments can still be obtained through some other techniques, e.g. by setting an event 10046. This can be done via alter session and via dbms_support.

Tk Prof Output

Elapsed times include waiting on following events:

| Event waited on | Times | Max. Wait | Total Waited |
|----------------------------|--------|-----------|--------------|
| ----- | Waited | ----- | ----- |
| latch free | 11 | 0.12 | 0.49 |
| db file scattered read | 312 | 0.99 | 7.85 |
| db file sequential read | 114 | 0.23 | 3.78 |
| buffer busy waits | 293 | 0.86 | 3.03 |
| log file switch completion | 2 | 0.38 | 0.43 |

The trace file can be fed through tkprof to produce a report; but the report still does not report the individual segments.

Output

```
WAIT #2: nam='db file scattered read' ela= 12235 p1=8 p2=76411 p3=8
WAIT #2: nam='db file scattered read' ela= 15933 p1=8 p2=76419 p3=5
WAIT #2: nam='db file scattered read' ela= 1153 p1=8 p2=76425 p3=2
WAIT #2: nam='db file sequential read' ela= 264 p1=8 p2=76433 p3=1
WAIT #2: nam='db file sequential read' ela= 2643 p1=8 p2=76440 p3=1
WAIT #2: nam='db file scattered read' ela= 11340 p1=8 p2=76444 p3=8
WAIT #2: nam='buffer busy waits' ela= 1517 p1=8 p2=76452 p3=130
WAIT #2: nam='buffer busy waits' ela= 11 p1=8 p2=76457 p3=130
WAIT #2: nam='buffer busy waits' ela= 115505 p1=8 p2=76457 p3=130
```

This event produces a very long text file in the user_dump_destination directory. A small excerpt of the file is shown here. This shows the wait event and the corresponding p1 and p2 values that can be used to get the segment name.

Event

- Long reports – difficult to assess
- All wait events
- Performance hit
- After effect

So, the event does provide the segment names but does not provide in a manner that is useful in diagnosis. It's very long and a quick assessment is difficult.

Also, this report contains information on all the wait events the session has experienced, not just the ones we are interested in. This makes the report somewhat indeterministic.

The report can be run after a problem has occurred. When the problem did occur the report was non-existent. The use of the event assumes that the problem could be reproduced at will; not necessarily a valid one.

Oracle Trace

- Different from sql_trace
- Integration with OEM
- Database reporting

The other option is Oracle Trace. This is different from SQL Trace. This can be set at the database level to capture wait events that can be formatted to the database tables or reports. The stats can be analyzed via OEM. A very useful tool, but this is obscured by lack of documentation and general ignorance.

Oracle Trace - ??

- After effect
- Poor documentation

Still, this assumes that the effect can be reproduced, may not be true.

Release 2

```
select owner, object_name
from v$segment_statistics
where statistic_name = 'buffer busy waits'
and value > 0
```

In Oracle 9i Release 2, the solution is simple. In order to find out the segments that experienced the buffer busy waits, just issue this query.

Objective

- Setting up the collection
- Collection Types
- Extensions
- Case Study

pr  **ligence** *Empowering Intelligence*

In this presentation, we will set up the collection and study the results to identify the problem objects.

The author has extended the scope of the view that Oracle provides. We will study that. Finally, we will cement our learning via a case study.

Setting Up

- STATISTICS_LEVEL in init.ora
- Can be set up via alter system
- Three levels
 - BASIC
 - TYPICAL
 - ALL

An init.ora parameter called STATISTICS_LEVEL needs to be set for collection of the stats to occur. There are three values for the parameter – BASIC, TYPICAL and ALL. This parameter can also be set via ALTER SYSTEM.

At the setting of BASIC, nothing is collected at all.

TYPICAL

- Buffer Cache
- Mean Time to Recover
- Shared Pool Sizing
- PGA Target
- Timed Statistics
- Segment Level Statistics

In the TYPICAL setting the above stats are collected.

ALL

- All of TYPICAL plus
 - Row Execution Stats
 - Timed OS Stats

In ALL setting, all of TYPICAL and the above two are collected.

Current Level

```
SELECT ACTIVATION_LEVEL,  
       STATISTICS_NAME,  
       SYSTEM_STATUS, SESSION_STATUS  
FROM V$STATISTICS_LEVEL  
ORDER BY ACTIVATION_LEVEL,  
       STATISTICS_NAME;
```

How do you find out the current setting of the parameter? Just issue the following query.

Current Stat Levels...

| ACTIVAT | STATISTICS_NAME | SYSTEM_S | SESSION_ |
|---------|---------------------------|----------|----------|
| ALL | Plan Execution Statistics | DISABLED | DISABLED |
| ALL | Timed OS Statistics | DISABLED | DISABLED |
| TYPICAL | Buffer Cache Advice | ENABLED | ENABLED |
| TYPICAL | MTTR Advice | ENABLED | ENABLED |
| TYPICAL | PGA Advice | ENABLED | ENABLED |
| TYPICAL | Segment Level Statistics | ENABLED | ENABLED |
| TYPICAL | Shared Pool Advice | ENABLED | ENABLED |
| TYPICAL | Timed Statistics | ENABLED | ENABLED |

Here are the results of the query. Note that the Segment Level Stats are collected. This is where we would be interested in. Since the individual stats can be stopped in a session whereas the group setting may be on, there are two columns for the system level and the session level status.

V\$SEGSTAT

TS#
OBJ#
DATAOBJ#
STATISTIC_NAME
STATISTIC#
VALUE

The view that contains the stats collected is called v\$segstat owned by sys. These are the columns of the view. Most of these are self explanatory.

V\$SEGMENT_STATISTICS

OWNER
OBJECT_NAME
SUBOBJECT_NAME
TABLESPACE_NAME
OBJECT_TYPE

However, due to the cryptic nature of the view, Oracle has provided another view called v\$segment_statistics with more meaningful fields names, in addition to the ones in v\$segstat.

Examining the Stats

```
SELECT STATISTIC_NAME, VALUE  
FROM V$SEGMENT_STATISTICS  
WHERE OWNER = 'SCOTT'  
AND OBJECT_NAME = 'SALES';
```

If you want to find out the stats for an object, this is the query.

...Examining the Stats

| STATISTIC_NAME | VALUE |
|------------------------------------|---------|
| logical reads | 1363168 |
| buffer busy waits | 1649 |
| db block changes | 1430448 |
| physical reads | 238620 |
| physical writes | 15572 |
| physical reads direct | 300 |
| physical writes direct | 0 |
| global cache cr blocks served | 0 |
| global cache current blocks served | 0 |
| ITL waits | 0 |
| row lock waits | 0 |

pr  **ligence** *Empowering Intelligence*

Here is the output of the query. Note the buffer busy waits have a value of 1649. This number is cumulative, not a snapshot.

Extensions

X\$KSOLSFT

FTS_STMP

INST_ID

Adding

Instance Number

Timestamp

 *Empowering Intelligence*

Examining the view definition of v\$segment_statistics we find that the main table is X\$KSOLSFT, an internal table that contains two fields important to us – the Instance Number and Timestamp. The field FTS_STMP is that value. We can use this to our advantage.

New View: segstat_with_time

```
create or replace view segstat_with_time as
select s.inst_id Instance_id,
       u.name Owner,
       o.name Object_name,
       o.subname Sub_object_name,
       ts.name Tablespace_name,
       decode(o.type#,
              0, 'NEXT OBJECT',
              57, 'SECURITY PROFILE',
              'UNDEFINED') Object_type,
       s.fts_statnam Statistic_name,
       s.fts_staval Value,
       to_char(fts_stmp, 'mm/dd/yyyy hh24:mi:ss')
       time_stamp
from obj$ o, user$ u, x$ksolsfts s, ts$ ts
where o.owner# = u.user#
and s.fts_inte = 0
and s.fts_obj# = o.obj#
and s.fts_tsn = ts.tsn#
and s.fts_obj# = o.dataobj#
and o.linkname is null
and
(o.type# not in (1, 10) or
 and 1 =
   (select 1 from ind$ I where i.obj# = o.obj#
    and i.type# in (1, 2, 3, 4, 6, 7, 9)
   )
 )
and o.name != 'NEXT OBJECT'
and o.name != '_default_auditing_options'
```

to_char (fts_stmp,
'mm/dd/yyyy
hh24:mi:ss')
time_stamp

Note how we have created a new view called segstat_with_time that has two new fields for instance number and timestamp. The instance number is important in RAC environments.

Case Study

- OLTP System
- Table SALES

Finally, the case study. We have an OLTP system with a table called SALES.

Table: SALES

```
SQL> desc sales
```

| Name | Null? | Type |
|----------------|----------|----------------|
| SALES_TRANS_ID | NOT NULL | NUMBER |
| CUSTOMER_ID | | NUMBER (2) |
| PRODUCT_ID | | CHAR (10) |
| PRICE | | NUMBER (10, 2) |
| QUANTITY | | NUMBER (5) |
| COMMENTS | | VARCHAR2 (20) |

 *Empowering Intelligence*

The sales table has a column named sales_trans_id as PK.

Stress.sql

```
declare
  v_cust_id      number(6) := 0;
begin
  for v_cust_id in 1..60 loop
    update sales
      set comments = 'CHANGED by &&1'
      where customer_id = v_cust_id
      and mod(sales_trans_id,2) = &&1;
    commit;
  end loop;
end;
```

pr  **ligence** Empowering Intelligence

The stress.sql is a script that updates the rows. However, based on the input (1 or 2) the updates happen to either odd or even numbered sales_trans_id.

If stress.sql is run from one session with a parameter 1 and from another with parameter 2 at the same time, they will not contend with each other for the same row (since they will operate on different rows) but they will operate on the same block, creating a buffer busy waits event.

Stressing

- Running from SQL*Plus Concurrently
SQL> @stress 0
SQL> @stress 1

Putting it all together

- STATSPACK Report
- Segment Statistics

We saw the segment stats for the SALES table from v\$segment_statistics and also from the statspack reports and we have identified that the buffer busy waits occur on the SALES table.

Resolution

- Packing Factor Reduction
- Increase PCFREE, INITRANS, MAXTRANS, FREELISTS, FREELIST GROUPS
- Reloading the Data

As a solution, we will now pack the sales table less and increase the other parameters that will contend less.

Table Design

- Increase PCTFREE, INITRANS, MAXTRANS, FREELISTS and FREELIST GROUPS

```
pctfree 20  
pctused 70  
storage (freelists 4  
         freelist groups 2 )  
initrans 4 maxtrans 30
```

pr  **ligence** *Empowering Intelligence*

We recreated the table with new parameters as above and run the same stress.sql again in the same manner. We saw that the buffer busy conditions are now severely reduced.

In Conclusion

- Only a few stats are collected. Hope for new stats.
- Provides insight into segment level stats hitherto impossible to access.

The new view collects only a few stats. It will be useful to have some more stats there.

More Information

- Oracle Documentation
http://otn.oracle.com/docs/products/oracle9i/doc_library/release2/server.920/a96533/instance.htm#34509
- Proligence Website
<http://www.proligence.com>
- Contact Me
arup@proligence.com

Thank you!

pr  **ligence**

www.proligence.com

pr  **ligence** *Empowering Intelligence*