【故障处理】队列等待之 TX - allocate ITL entry 案例

1.1 BLOG 文档结构图

1.2 前言部分

1.2.1 导读和注意事项

各位技术爱好者，看完本文后，你可以掌握如下的技能，也可以学到一些其它你所不知道的知识，~O(∩_∩)O~：

① enq: TX - allocate ITL entry 等待事件的解决

② 一般等待事件的解决办法

③ 队列等待的基本知识

Tips：

① 本文在 ITpub（http://blog.itpub.net/26736162）、博客园（http://www.cnblogs.com/lhrbest）和微信公众号（xiaomaimiaolhr）有同步更新

② 文章中用到的所有代码，相关软件，相关资料请前往小麦苗的云盘下载（http://blog.itpub.net/26736162/viewspace-1624453/）

③ 若文章代码格式有错乱，推荐使用搜狗、360 或 QQ 浏览器，也可以下载 pdf 格式的文档来查看，pdf 文档

下载地址：http://blog.itpub.net/26736162/viewspace-1624453/，另外 itpub 格式显示有问题，可以去博客园地址阅读

④ 本篇 BLOG 中命令的输出部分需要特别关注的地方我都用灰色背景和粉红色字体来表示，比如下边的例子中，thread 1 的最大归档日志号为 33，thread 2 的最大归档日志号为 43 是需要特别关注的地方；而命令一般使用黄色背景和红色字体标注；对代码或代码输出部分的注释一般采用蓝色字体表示。

```
  List of Archived Logs in backup set 11
  Thrd Seq     Low SCN    Low Time            Next SCN   Next Time
  ---- ------- ---------- ------------------- ---------- ---------
  1    32      1621589    2015-05-29 11:09:52 1625242    2015-05-29 11:15:48
  1    33      1625242    2015-05-29 11:15:48 1625293    2015-05-29 11:15:58
  2    42      1613951    2015-05-29 10:41:18 1625245    2015-05-29 11:15:49
  2    43      1625245    2015-05-29 11:15:49 1625253    2015-05-29 11:15:53

[ZHLHRDB1:root]:/>lsvg -o
T_XDESK_APP1_vg
rootvg
[ZHLHRDB1:root]:/>
00:27:22 SQL> alter tablespace idxtbs read write;

====》2097152*512/1024/1024/1024=1G
```

本文如有错误或不完善的地方请大家多多指正，ITPUB 留言或 QQ 皆可，您的批评指正是我写作的最大动力。

1.3 故障分析及解决过程

1.3.1 故障环境介绍

项目	Source db
db 类型	RAC
db version	11.2.0.3.0
db 存储	ASM
OS 版本及 kernel 版本	AIX 64 位 7.1.0.0

1.3.2 故障发生现象及报错信息

最近事情比较多，不过还好，碰到的都是等待事件相关的，同事发了个 AWR 报告，说是系统响应很慢，我简单看了下，简单分析下吧：

WORKLOAD REPOSITORY report for

DB Name	DB Id	Instance	Inst num	Startup Time	Release	RAC
[redacted]	3860591551	[redacted]	2	29-Jul-16 15:07	11.2.0.4.0	YES

Host Name	Platform	CPUs	Cores	Sockets	Memory (GB)
[redacted]	AIX-Based Systems (64-bit)	32	8		48.00

	Snap Id	Snap Time	Sessions	Cursors/Session	Instances
Begin Snap:	1337	05-Sep-16 16:55:07	94	2.3	2
End Snap:	1340	05-Sep-16 17:14:43	1454	3.2	2
Elapsed:		19.60 (mins)			
DB Time:		11,461.33 (mins)			

20 分钟时间而 DB Time 为 11461 分钟，DB Time 太高了，负载很大，很可能有异常的等待事件，系统配置还是比较牛逼的。

Load Profile

	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	584.7	1.7	0.33	0.25
DB CPU(s):	0.6	0.0	0.00	0.00
Redo size (bytes):	479,794.6	1,371.4		
Logical read (blocks):	12,277.9	35.1		
Block changes:	2,319.6	6.6		
Physical read (blocks):	5.3	0.0		
Physical write (blocks):	63.2	0.2		
Read IO requests:	5.3	0.0		
Write IO requests:	27.5	0.1		
Read IO (MB):	0.0	0.0		
Write IO (MB):	0.5	0.0		
Global Cache blocks received:	475.7	1.4		
Global Cache blocks served:	465.0	1.3		
User calls:	2,348.6	6.7		
Parses (SQL):	363.6	1.0		
Hard parses (SQL):	0.1	0.0		
SQL Work Area (MB):	0.5	0.0		
Logons:	1.4	0.0		
Executes (SQL):	1,757.5	5.0		
Rollbacks:	0.0	0.0		
Transactions:	349.9			

事务量很大，其它个别参数有点问题，不一一解说了。**Instance Efficiency Percentages** 也有点问题：

Instance Efficiency Percentages (Target 100%)

Buffer Nowait %:	64.44	Redo NoWait %:	100.00
Buffer Hit %:	99.96	In-memory Sort %:	100.00
Library Hit %:	99.98	Soft Parse %:	99.97
Execute to Parse %:	79.31	Latch Hit %:	95.82
Parse CPU to Parse Elapsd %:	57.04	% Non-Parse CPU:	99.30

等待事件很明显了：

Top 10 Foreground Events by Total Wait Time

Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time	Wait Class
enq: TX - allocate ITL entry	2,437,456	444.8K	183	64.7	Configuration
enq: TX - row lock contention	8,043	85.8K	10666	12.5	Application
buffer busy waits	2,052,827	59.6K	29	8.7	Concurrency
gc buffer busy release	1,227,486	54.9K	45	8.0	Cluster
gc buffer busy acquire	652,899	18.4K	28	2.7	Cluster
latch: cache buffers chains	2,000,333	12K	6	1.7	Concurrency
latch: ges resource hash list	1,689,468	9642.8	6	1.4	Other
DB CPU		692.6		.1	
log file sync	408,435	441.9	1	.1	Commit
enq: TX - contention	89	428.9	4819	.1	Other

AWR 的其它部分就不分析了，首先这个等待事件：enq: TX - allocate ITL entry 比较少见，查了一下 MOS，有点收获：Troubleshooting waits for 'enq: TX - allocate ITL entry' (文档 ID 1472175.1)

Observe high waits for event enq: TX - allocate ITL entry

Top 5 Timed Foreground Events

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
enq: TX - allocate ITL entry	1,200	3,129	2607	85.22	Configuration
DB CPU			323	8.79	
gc buffer busy acquire	17,261	50	3	1.37	Cluster
gc cr block 2-way	143,108	48	0	1.32	Cluster
gc current block busy	10,631	46	4	1.24	Cluster

CAUSE

By default INITRANS value for table is 1 and for index is 2. This defines an internal block structure called the Interested Transaction List (ITL). In order to modify data in a block, a process needs to use an empty ITL slot to record that the transaction is interested in modifying some of the data in the block. If there are insufficient free ITL slots then new ones will be taken in the free space reserved in the block. If this runs out and too many concurrent DML transactions are competing for the same data block we observe contention against the following wait event - "enq: TX - allocate ITL entry".

You can see candidates for re-organisation due to ITL problems in the "Segments by

ITL Waits" section of an Automatic Workload Repository (AWR) report:

```
Segments by ITL Waits
  * % of Capture shows % of ITL waits for each top segment compared
  * with total ITL waits for all segments captured by the Snapshot
Owner Tablespace Name Object Name Subobject Name Obj. Type       ITL  Waits % of Capture
PIN   BRM_TABLES      SERVICE_T                  TABLE           188               84.30
PIN   BRM_TABLES      BILLINFO_T  P_R_06202012   TABLE PARTITION  35               15.70
```

SOLUTION

The main solution to this issue is to increase the ITL capability of the table or index by re-creating it and altering the INITRANS or PCTFREE parameter to be able to handle more concurrent transactions. This in turn will help to reduce "enq: TX - allocate ITL entry" wait events.

To reduce enq: TX - allocate ITL entry" wait events, We need to follow the steps below:

1) Set INITRANS to 50 and pct_free to 40

```
alter table <table_name> PCTFREE 40  INITRANS 50;
```

2) Re-organize the table using move (alter table <table_name> move;)

3) Then rebuild all the indexes of the table as below

```
alter index <index_name>  rebuild PCTFREE 40 INITRANS 50;
```

总结一下：

原因：表和索引的默认 INITRANS 值不合适，引起的事务槽分配等待。当一个事务需要修改一个数据块时，需要在数据块头部获取一个可用的 ITL 槽，用于记录事务的 id，使用 undo 数据块地址，scn 等信息。如果事务申请不到新的可用 ITL 槽时，就会产生 enq: TX - allocate ITL entry 等待。 发生这个等待时，要么是块上的已分配 ITL 个数(通过 ini_trans 参数控制)达到了上限 255(10g 以后没有了 max_trans 限制参数，无法指定小于 255 的值)，要么是这个块中没有更多的空闲空间来容纳一个 ITL 了(每个 ITL 占用 24bytes)。 默认情况下创建的表 ITL 槽数最小为 1+1，pctfree 为 10，那么如果是这样一种情况，如果表中经常执行 update 语句，然后块中剩余的 10%空间所剩无几，而且业务的并发量还很大，此时就很容易遇到 enq: TX - allocate ITL entry 等待。

解决：解决方式就是调整表和索引的 INITRANS，有必要还需要调整 pcfree 值。

1) Set INITRANS to 50 and pct_free to 40

alter table <table_name> PCTFREE 40 INITRANS 50;

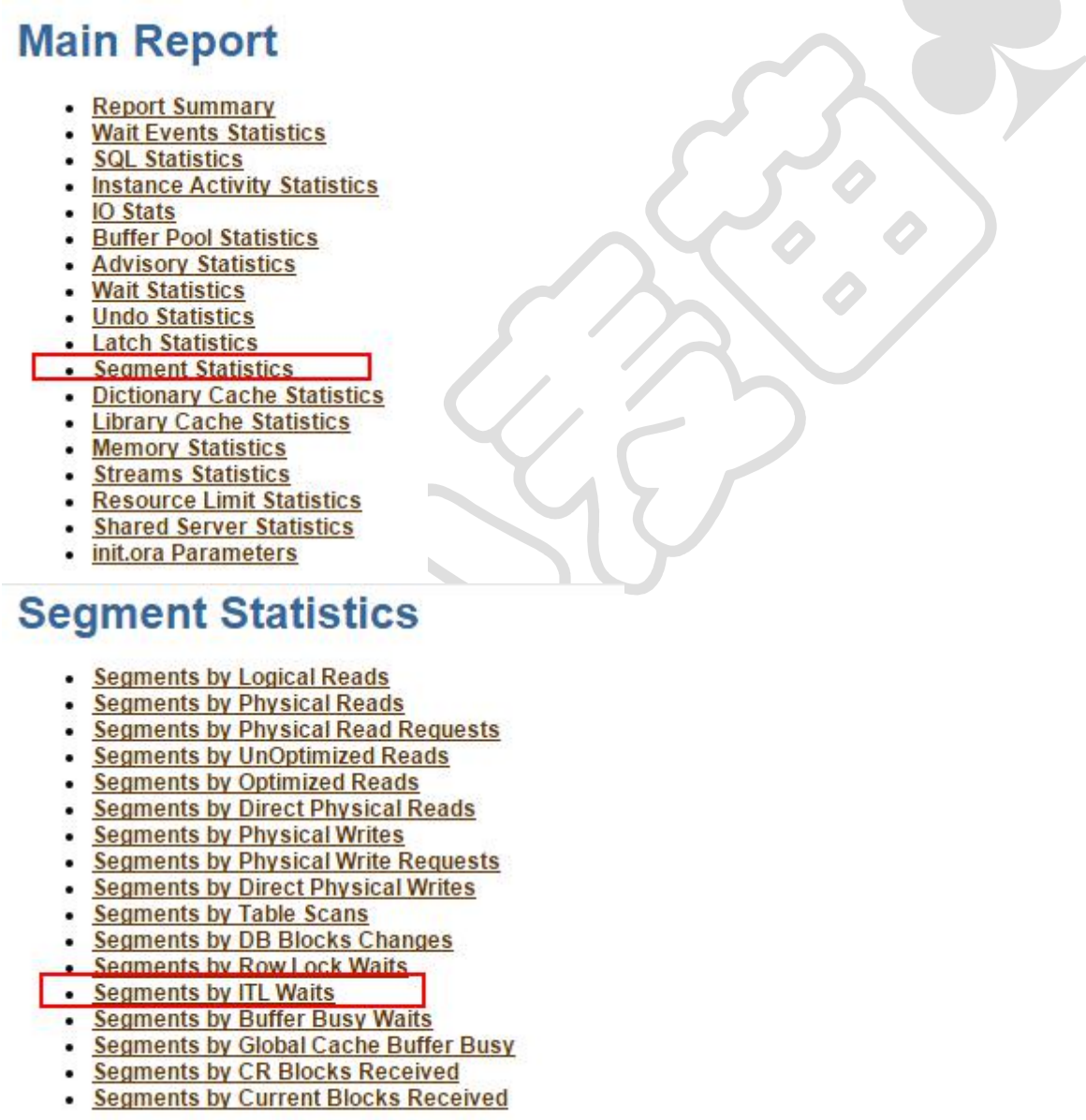
2) Re-organize the table using move (alter table <table_name> move;)

```
3) Then rebuild all the indexes of the table as below
alter index <index_name>  rebuild PCTFREE 40 INITRANS 50;
```

1.3.3 故障分析及解决

有了以上的知识，我们知道，目前首先需要找到产生等待事件的表，然后修改 INITRANS 和 PCTFREE 来重构表就可以了。

我们查看 AWR 中的 Segments by ITL Waits 部分：



Segments by ITL Waits

- % of Capture shows % of ITL waits for each top segment compared
- with total ITL waits for all segments captured by the Snapshot

Owner	Tablespace Name	Object Name	Subobject Name	Obj. Type	ITL Waits	% of Capture
[redacted]	[redacted]	ORGANIZATION		TABLE	2,292,832	100.00
[redacted]	[redacted]	[redacted]		INDEX	3	0.00

```
SELECT D.SQL_ID,
       CHR(BITAND(P1, -16777216) / 16777215) ||
       CHR(BITAND(P1, 16711680) / 65535) "Lock",
       BITAND(P1, 65535) "Mode",
       D.CURRENT_OBJ#,
       COUNT(1),
       COUNT(DISTINCT D.SESSION_ID)
  FROM DBA_HIST_ACTIVE_SESS_HISTORY D
 WHERE D.SAMPLE_TIME BETWEEN
       TO_DATE('2016-09-05 16:55:00', 'YYYY-MM-DD HH24:MI:SS') AND
       TO_DATE('2016-09-05 17:15:00', 'YYYY-MM-DD HH24:MI:SS')
   AND D.EVENT = 'enq: TX - allocate ITL entry'
 GROUP BY D.SQL_ID,
          (CHR(BITAND(P1, -16777216) / 16777215) ||
          CHR(BITAND(P1, 16711680) / 65535)),
          (BITAND(P1, 65535)),
          D.CURRENT_OBJ#;
```

SQL_ID	Lock	Mode	CURRENT_OBJ#	COUNT(1)	COUNT(DISTINCTD.SESSION_ID)
1cmnjddakrqbv	TX	4	87620	76441	1428

```
SELECT * FROM v$sql a WHERE a.SQL_ID='1cmnjddakrqbv';
```

SQL_TEXT	SQL_FULLTEXT	SQL_ID
update organization o set o.quota_unused = o.quota_unus ...	<CLOB> ...	1cmnjddakrqbv

```
SELECT * FROM Dba_Objects d WHERE d.object_id=87620;
```

	OWNER	OBJECT_NAME	SUBOBJECT_NAME	OBJECT_ID	DATA_OBJECT_ID	OBJECT_TYPE	CREATED
1	[redacted] ...	ORGANIZATION ...	...	87620	90657	TABLE	2016/8/4 11:34:45

好吧，知道了表名，我们查看一下表的属性：

```
SELECT * FROM Dba_Tables d WHERE d.table_name='ORGANIZATION';
```

TABLE_NAME	TABLESPACE_NAME	CLUSTER_NAME	IOT_NAME	STATUS	PCT_FREE	PCT_USED	INI_TRANS
ORGANIZATION ...	CNSL_DATA ...	...	...	VALID	10		

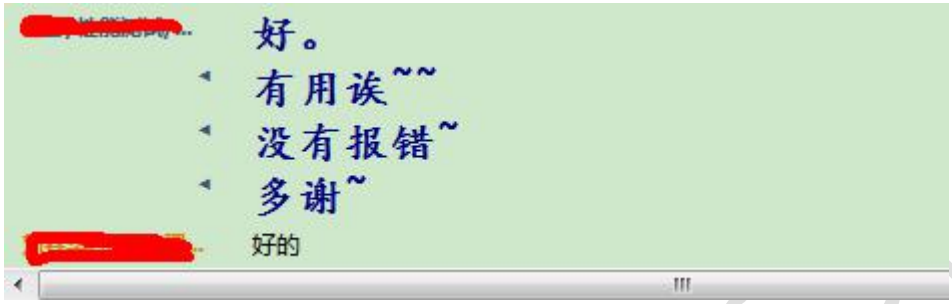
pct_free 为 10，ini_trans 为 1，我们根据 MOS 应该修改这 2 个值，SQL 如下：

```
ALTER TABLE ORGANIZATION PCTFREE 20  INITRANS 16;
ALTER TABLE ORGANIZATION MOVE;
```

```
ALTER INDEX PK_ORGANIZATION  REBUILD PCTFREE 20 INITRANS 16 NOLOGGING;
```

这里需要注意：该表大约 2000 条记录，很小，所以 MOVE 的时候可以不用并行，也不用 NOLOGGING，若是表很大的时候可以考虑并行+NOLOGGING 属性，另外，还需要 REBUILD 索引才可以。

修改完成后，开发人员经过测试后终于可以了，给我简单回复了一下。



About Me

- 本文作者：小麦苗，只专注于数据库的技术，更注重技术的运用
- 本文在 itpub（http://blog.itpub.net/26736162）、博客园(http://www.cnblogs.com/lhrbest)和个人微信公众号（xiaomaimiaolhr）上有同步更新，推荐 pdf 文件阅读
- QQ 群：230161599 微信群：私聊
- 本文 itpub 地址：http://blog.itpub.net/26736162/viewspace-2124735/ 博客园地址：http://www.cnblogs.com/lhrbest/articles/5854407.html
- 本文 pdf 版：http://yunpan.cn/cdEQedhCs2kFz （提取码：ed9b）
- 小麦苗分享的其它资料：http://blog.itpub.net/26736162/viewspace-1624453/
- 联系我请加 QQ 好友(642808185)，注明添加缘由
- 于 2016-09-06 09:00~2016-09-06 19:00 在中行完成

长按识别二维码或微信客户端扫描下边的二维码来关注小麦苗的微信公众号：xiaomaimiaolhr,学习最实用的数据库技术。