【等待事件】等待事件系列（5.1）--Enqueue(队列等待)

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 Enqueue 队列等待

② Enq 数据字典

③ enq: AE - lock

④ enq: MR 锁

⑤ enq: DX - contention

⑥ enq: SQ - contention 序列等待

1.2.2 相关参考文章链接

【推荐】 等待事件系列（1）--User I/O 类型（下）		http://blog.itpub.net/26736162/viewspace-2124435/
【推荐】 等待事件系列（1）--User I/O 类型（上）		http://blog.itpub.net/26736162/viewspace-2124417/
2016-09-23	【等待事件】日志类 等待事件（4.7）--LGWR wait for redo copy、switch logfile 等	http://mp.weixin.qq.com/s?__biz=MzIzOTA2NjEzNQ==&mid=2454771555&idx=1&sn=3c488fabff8d508fbb942c5bd206c532&chksm=fe8bba1bc9fc330d9a422b7795fc5ed41d72d4fb0c12aff97b5be37e8334de4a2f3663841630&scene=21#wechat_redirect
2016-09-22	【等待事件】日志类 等待事件（4.6）--log file single write	http://mp.weixin.qq.com/s?__biz=MzIzOTA2NjEzNQ==&mid=2454771552&idx=1&sn=bafa6049cf16da92f07b1d50da7f274e&chksm=fe8bba18c9fc330eb3e3b636a1021e57814ea6b4cadc4ac2e4e34ce88c5be6a636a6b724e8ec&scene=21#wechat_redirect
2016-09-21	【等待事件】日志类 等待事件（4.5）--log file sequential read	http://mp.weixin.qq.com/s?__biz=MzIzOTA2NjEzNQ==&mid=2454771549&idx=1&sn=08452a61dcebf9aca50b1483799510fe&chksm=fe8bba25c9fc3333bbd3298ac9a3ecf54e86e7c71e5429a3a034179585a412b77e32ae745c96&scene=21#wechat_redirect
2016-09-20	【等待事件】日志类 等待事件（4.4）--log buffer space（日志缓冲空间）	http://mp.weixin.qq.com/s?__biz=MzIzOTA2NjEzNQ==&mid=2454771546&idx=1&sn=a3397cc535a453ed4a5e77c04a6fa767&chksm=fe8bba22c9fc33346ca821489faee74732d876201a6d082e789d1653316461f27dfc81fa4275&scene=21#wechat_redirect
2016-09-19	【等待事件】日志类 等待事件（4.3）--log file parallel write	http://mp.weixin.qq.com/s?__biz=MzIzOTA2NjEzNQ==&mid=2454771527&idx=1&sn=a5564f7c056fa89f59257ca7e8aaa898&chksm=fe8bba3fc9fc33292b365784c327ec2d6a682fc92de5de08e6cc3c22690462c2d092aa4e9f4b&scene=21#wechat_redirect
2016-09-18	【等待事件】日志类 等待事件（4.2）--log file sync（日志文件同步）	http://mp.weixin.qq.com/s?__biz=MzIzOTA2NjEzNQ==&mid=2454771524&idx=1&sn=70969b743781b035eb50e9c993192d99&chksm=fe8bba3cc9fc332a7d3a0fda5d589b8803379e6cb90d2a86397f31f003a621414849c77e0cfc&scene=21#wechat_redirect
2016-09-17	【等待事件】日志类 等待事件（4.1）--log file switch（日志文件切换）	http://mp.weixin.qq.com/s?__biz=MzIzOTA2NjEzNQ==&mid=2454771523&idx=1&sn=99224ba8fc53353c5c6f03ac7babdb5e&chksm=fe8bba3bc9fc332d558890b7278c2b6f51e194b958bd11b7f6d33d443ec45245ec6800461a7e&scene=21#wechat_redirect
2016-09-07	【等待事件】System I/O 类 等待事件（3.4）--control file single write	http://mp.weixin.qq.com/s?__biz=MzIzOTA2NjEzNQ==&mid=2454771471&idx=1&sn=5922a52ac6294acf2802f44e2bb0d724&chksm=fe8bba77c9fc336151a61bdf876cb058df0d61d1404d8450cb7771330b6d44309d86dae4bb54&scene=21#wechat_redirect
2016-09-06	【等待事件】System I/O 类 等待事件（3.3）--control file sequential read	http://mp.weixin.qq.com/s?__biz=MzIzOTA2NjEzNQ==&mid=2454771468&idx=1&sn=fc7d83d1a9b12911f3c93d3b5b444e9a&chksm=fe8bba74c9fc3362b58717fca9e95c68d45e701fa2f733a643ba01db7969cca668858272fbfc&scene=21#wechat_redirect
2016-09-04	【等待事件】System I/O 类 等待事件（3.2）--control file parallel write	http://mp.weixin.qq.com/s?__biz=MzIzOTA2NjEzNQ==&mid=2454771458&idx=1&sn=e949dfa5bff65ce4a596005955c5be5a&scene=21#wechat_redirect
2016-09-03	【等待事件】System I/O 类 等待事件（3.1）--db file parallel write	http://mp.weixin.qq.com/s?__biz=MzIzOTA2NjEzNQ==&mid=2454771454&idx=1&sn=e90248954475dfd2c78bdec592405735&scene=21#wechat_redirect
2016-09-01	【等待事件】User I/O 类 等待事件（2.10）--所有 User I/O 类 等待事件总结	http://mp.weixin.qq.com/s?__biz=MzIzOTA2NjEzNQ==&mid=2454771447&idx=1&sn=22ae192f0d8a161f65514339ad763985&scene=21#wechat_redirect
2016-08-31	【等待事件】User I/O 类 等待事件（2.9）--local write wait	http://mp.weixin.qq.com/s?__biz=MzIzOTA2NjEzNQ==&mid=2454771443&idx=1&sn=02b4ad5ca03052013b69ae6bcb7e3487&scene=21#wechat_redirect
2016-08-30	【等待事件】User I/O 类 等待事件（2.8）--read by other session	http://mp.weixin.qq.com/s?__biz=MzIzOTA2NjEzNQ==&mid=2454771439&idx=1&sn=b3c01eed444cd6e597a63a3ed0687768&scene=21#wechat_redirect

2016-08-29	【等待事件】User I/O类 等待事件(2.7)--direct path read/write temp	http://mp.weixin.qq.com/s?__biz=MzIzOTA2NjEzNQ==&mid=2454771429&idx=1&sn=50b5684e699165a34087db88e07edb34&scene=21#wechat_redirect
2016-08-27	【等待事件】User I/O类 等待事件(2.6)--direct path write(直接路径写、DRW)	http://mp.weixin.qq.com/s?__biz=MzIzOTA2NjEzNQ==&mid=2454771420&idx=1&sn=458eb18dc26da94debcea62643d15181&scene=21#wechat_redirect
2016-08-26	【等待事件】User I/O类 等待事件(2.5)--direct path read(直接路径读、DPR)	http://mp.weixin.qq.com/s?__biz=MzIzOTA2NjEzNQ==&mid=2454771416&idx=1&sn=b26c3135584c5b60ce14cc0749ac58a7&scene=21#wechat_redirect
2016-08-20	【等待事件】User I/O类 等待事件(2.4)--db file single write	http://mp.weixin.qq.com/s?__biz=MzIzOTA2NjEzNQ==&mid=2454771403&idx=1&sn=054dd852dac5ac8837fa251f0e84332e&scene=21#wechat_redirect
2016-08-16	【等待事件】User I/O类 等待事件(2.3)--db file parallel read	http://mp.weixin.qq.com/s?__biz=MzIzOTA2NjEzNQ==&mid=2454771387&idx=1&sn=0037fb89470d8e6dd5ff72714b18a3b7&scene=21#wechat_redirect
2016-08-15	【等待事件】User I/O类 等待事件(2.2)--db file scattered read(数据文件离散读)	http://mp.weixin.qq.com/s?__biz=MzIzOTA2NjEzNQ==&mid=2454771379&idx=1&sn=5887eee02885000c1d293adfd04ee044&scene=21#wechat_redirect
2016-08-14	【等待事件】User I/O类 等待事件(2.1)--db file sequential read(数据文件顺序读)	http://mp.weixin.qq.com/s?__biz=MzIzOTA2NjEzNQ==&mid=2454771376&idx=1&sn=42de046e73190f4e265f81bbb6e3ae00&scene=21#wechat_redirect
2016-08-13	【等待事件】等待事件概述(1)--等待事件的源起和分类	http://mp.weixin.qq.com/s?__biz=MzIzOTA2NjEzNQ==&mid=2454771373&idx=1&sn=1e55af795aae5f641b2c3cc610814ead&scene=21#wechat_redirect

1.3 Enqueue(队列等待)

1.3.1 简介

Enqueue 是一种保护共享资源的锁定机制，是协调访问数据库资源的内部锁。该锁定机制保护共享资源，以避免因并发操作而损坏数据，比如通过锁定保护一行记录，避免多个用户同时更新。Enqueue 采用排队机制，即 FIFO(先进先出)来控制资源的使用。

Enqueue 是一组锁定事件的集合，如果数据库中这个等待事件比较显著，还需要进一步追踪是哪一个类别的锁定引发了数据库等待。

Enqueue 这个词其实是 LOCK 的另一种描述语。当我们在 AWR 报告中发现长时间的 Enqueue 等待事件时，说明数据库中出现了阻塞和等待，可以关联 AWR 报告中的 Enqueue Activity 部分来确定是哪一种锁定出现了长时间等待。

所有以“enq：”打头的等待事件都表示这个会话正在等待另一个会话持有的内部锁释放，它的名称格式是 enq:enqueue_type - related_details。数据库动态性能视图 v$event_name 提供所有以“enq：”开头的

等待事件的列表。

```
SELECT * FROM V$EVENT_NAME WHERE NAME LIKE 'enq%';
```

	EVENT#	EVENT_ID	NAME	PARAMETER1	PARAMETER2	PARAMETER3
1	87	3141712284	enq: PW - flush prewarm buffers	name|mode	0	0
2	107	3500532018	enq: RO - contention	name|mode	2	0
3	108	143262751	enq: RO - fast object reuse	name|mode	2	0
4	109	4205197519	enq: KO - fast object checkpoint	name|mode	2	0
5	153	1567037747	enq: MV - datafile move	name|mode	type	file #
6	235	668627480	enq: TM - contention	name|mode	object #	table/partition
7	236	1649608974	enq: ST - contention	name|mode	0	0
8	241	310662678	enq: TX - row lock contention	name|mode	usn<<16 | slot	sequence
9	242	281768874	enq: TX - allocate ITL entry	name|mode	usn<<16 | slot	sequence
10	243	1035026728	enq: TX - index contention	name|mode	usn<<16 | slot	sequence
11	244	1435178951	enq: TW - contention	name|mode	0	operation
12	254	1645217925	enq: HW - contention	name|mode	table space #	block

4:1 512 rows selected in 0.212 seconds

```
SELECT D.PARAMETER1, COUNT(1)
  FROM V$EVENT_NAME D
 WHERE NAME LIKE 'enq%'
 GROUP BY D.PARAMETER1;
```

	PARAMETER1	COUNT(1)
1	type|mode	193
2	name|mode	319

可以看出 11.2.0.4 中大约有 512 种 Enqueue 等待事件。

这一类的等待事件 P1 参数一般有“name|mode”和“type|mode”2 种形式，其中：

Name： enqueue 的名称和类型。

Mode： enqueue 的模式。

可以使用如下 SQL 查看当前会话等待的 enqueue 名称和类型（当然，这里的视图不仅仅可以是 v$session_wait，只要包含 p1 的值即可，比如 v$session、DBA_HIST_ACTIVE_SESS_HISTORY 等视图）：

```
SELECT   CHR (TO_CHAR (BITAND (P1, -16777216)) / 16777215)
       || CHR (TO_CHAR (BITAND (P1, 16711680)) / 65535)
          "LOCK",
       TO_CHAR (BITAND (P1, 65535)) "MODE"
  FROM   V$SESSION_WAIT
 WHERE   EVENT = 'ENQUEUE'
```

Oracle 的 enqueue 包含以下模式：

模式代码	解释
1	Null mode
2	Sub-Share
3	Sub-Exclusive
4	Share
5	Share/Sub-Exclusive
6	Exclusive

Oracle 的 enqueue 有如下类型：

Enqueue 缩写	缩写解释

BL	Buffer Cache management
BR	Backup/Restore
CF	Controlfile transaction
CI	Cross-instance Call Invocation
CU	Bind Enqueue
DF	Datafile
DL	Direct Loader Index Creation
DM	Database Mount
DR	Distributed Recovery Process
DX	Dirstributed Transaction
FP	File Object
FS	File Set
HW	High-water Lock
IN	Instance Number
IR	Instance Recovery
IS	Instance State
IV	Library Cache Invalidation
JI	Enqueue used during AJV snapshot refresh
JQ	Job Queue
KK	Redo Log “Kick”
KO	Multiple Object Checkpoint
L[A-p]	Library Cache Lock
LS	Log start or switch
MM	Mount Definition
MR	Media recovery
N[A-Z]	Library Cache bin
PE	Alter system set parameter =value
PF	Password file
PI	Parallel slaves
PR	Process startup
PS	Parallel slave synchronization
Q[A-Z]	Row Cache
RO	Object Reuse
RT	Redo Thread
RW	Row Wait
SC	System Commit Number
SM	SMON
SN	Sequence Number
SQ	Sequence Number Enqueue

SR	Synchronized replication
SS	Sort segment
ST	Space management transaction
SV	Sequence number Value
TA	Transaction recovery
TC	Thread Checkpoint
TE	Extend Table
TM	DML enqueue
TO	Temporary Table Object Enqueue
TS	Temporary Segement(also TableSpace)
TT	Temporary Table
TX	Transaction
UL	User-defined Locks
UN	User name
US	Undo segment, Serialization
WL	Being Written Redo Log
XA	Instance Attribute Log
XI	Instance Registration Lock

所有队列等待锁：

Enqueue Type	Description
enq: AD - allocate AU	Synchronizes accesses to a specific OSM disk AU
enq: AD - deallocate AU	Synchronizes accesses to a specific OSM disk AU
enq: AF - task serialization	This enqueue is used to serialize access to an advisor task
enq: AG - contention	Synchronizes generation use of a particular workspace
enq: AO - contention	Synchronizes access to objects and scalar variables
enq: AS - contention	Synchronizes new service activation
enq: AT - contention	Serializes 'alter tablespace' operations
enq: AW - AW$ table lock	Global access synchronization to the AW$ table
enq: AW - AW generation lock	In-use generation state for a particular workspace
enq: AW - user access for AW	Synchronizes user accesses to a particular workspace
enq: AW - AW state lock	Row lock synchronization for the AW$ table
enq: BR - file shrink	Lock held to prevent file from decreasing in physical size during RMAN backup
enq: BR - proxy-copy	Lock held to allow cleanup from backup mode during an RMAN proxy-copy backup
enq: CF - contention	Synchronizes accesses to the controlfile
enq: CI - contention	Coordinates cross-instance function invocations
enq: CL - drop label	Synchronizes accesses to label cache when dropping a label
enq: CL - compare labels	Synchronizes accesses to label cache for label comparison
enq: CM - gate	Serialize access to instance enqueue
enq: CM - instance	Indicate OSM disk group is mounted
enq: CT - global space	Lock held during change tracking space management operations that

management	affect the entire change tracking file
enq: CT - state	Lock held while enabling or disabling change tracking, to ensure that it is only enabled or disabled by one user at a time
enq: CT - state change gate 2	Lock held while enabling or disabling change tracking in RAC
enq: CT - reading	Lock held to ensure that change tracking data remains in existence until a reader is done with it
enq: CT - CTWR process start/stop	Lock held to ensure that only one CTWR process is started in a single instance
enq: CT - state change gate 1	Lock held while enabling or disabling change tracking in RAC
enq: CT - change stream ownership	Lock held by one instance while change tracking is enabled, to guarantee access to thread-specific resources
enq: CT - local space management	Lock held during change tracking space management operations that affect just the data for one thread
enq: CU - contention	Recovers cursors in case of death while compiling
enq: DB - contention	Synchronizes modification of database wide supplemental logging attributes
enq: DD - contention	Synchronizes local accesses to ASM disk groups
enq: DF - contention	Enqueue held by foreground or DBWR when a datafile is brought online in RAC
enq: DG - contention	Synchronizes accesses to ASM disk groups
enq: DL - contention	Lock to prevent index DDL during direct load
enq: DM - contention	Enqueue held by foreground or DBWR to synchronize database mount/open with other operations
enq: DN - contention	Serializes group number generations
enq: DP - contention	Synchronizes access to LDAP parameters
enq: DR - contention	Serializes the active distributed recovery operation
enq: DS - contention	Prevents a database suspend during LMON reconfiguration
enq: DT - contention	Serializes changing the default temporary table space and user creation
enq: DV - contention	Synchronizes access to lower-version Diana (PL/SQL intermediate representation)
enq: DX - contention	Serializes tightly coupled distributed transaction branches
enq: FA - access file	Synchronizes accesses to open ASM files
enq: FB - contention	Ensures that only one process can format data blocks in auto segment space managed tablespaces
enq: FC - open an ACD thread	LGWR opens an ACD thread
enq: FC - recover an ACD thread	SMON recovers an ACD thread
enq: FD - Marker generation	Synchronization
enq: FD - Flashback coordinator	Synchronization
enq: FD - Tablespace flashback on/off	Synchronization
enq: FD - Flashback on/off	Synchronization
enq: FG - serialize ACD relocate	Only 1 process in the cluster may do ACD relocation in a disk group
enq: FG - LGWR redo generation enq race	Resolve race condition to acquire Disk Group Redo Generation Enqueue
enq: FG - FG redo generation enq race	Resolve race condition to acquire Disk Group Redo Generation Enqueue
enq: FL - Flashback database log	Synchronization

enq: FL - Flashback db command	Enqueue used to synchronize Flashback Database and deletion of flashback logs.
enq: FM - contention	Synchronizes access to global file mapping state
enq: FR - contention	Begin recovery of disk group
enq: FS - contention	Enqueue used to synchronize recovery and file operations or synchronize dictionary check
enq: FT - allow LGWR writes	Allow LGWR to generate redo in this thread
enq: FT - disable LGWR writes	Prevent LGWR from generating redo in this thread
enq: FU - contention	This enqueue is used to serialize the capture of the DB Feature, Usage and High Water Mark Statistics
enq: HD - contention	Serializes accesses to ASM SGA data structures
enq: HP - contention	Synchronizes accesses to queue pages
enq: HQ - contention	Synchronizes the creation of new queue IDs
enq: HV - contention	Lock used to broker the high water mark during parallel inserts
enq: HW - contention	Lock used to broker the high water mark during parallel inserts
enq: IA - contention	
enq: ID - contention	Lock held to prevent other processes from performing controlfile transaction while NID is running
enq: IL - contention	Synchronizes accesses to internal label data structures
enq: IM - contention for blr	Serializes block recovery for IMU txn
enq: IR - contention	Synchronizes instance recovery
enq: IR - contention2	Synchronizes parallel instance recovery and shutdown immediate
enq: IS - contention	Enqueue used to synchronize instance state changes
enq: IT - contention	Synchronizes accesses to a temp object's metadata
enq: JD - contention	Synchronizes dates between job queue coordinator and slave processes
enq: JI - contention	Lock held during materialized view operations (like refresh, alter) to prevent concurrent operations on the same materialized view
enq: JQ - contention	Lock to prevent multiple instances from running a single job
enq: JS - contention	Synchronizes accesses to the job cache
enq: JS - coord post lock	Lock for coordinator posting
enq: JS - global wdw lock	Lock acquired when doing wdw ddl
enq: JS - job chain evaluate lock	Lock when job chain evaluated for steps to create
enq: JS - q mem clnup lck	Lock obtained when cleaning up q memory
enq: JS - slave enq get lock2	Get run info locks before slv objget
enq: JS - slave enq get lock1	Slave locks exec pre to sess strt
enq: JS - running job cnt lock3	Lock to set running job count epost
enq: JS - running job cnt lock2	Lock to set running job count epre
enq: JS - running job cnt lock	Lock to get running job count
enq: JS - coord rcv lock	Lock when coord receives msg
enq: JS - queue lock	Lock on internal scheduler queue
enq: JS - job run lock - synchronize	Lock to prevent job from running elsewhere
enq: JS - job recov lock	Lock to recover jobs running on crashed RAC inst
enq: KK - context	Lock held by open redo thread, used by other instances to force a log switch
enq: KM - contention	Synchronizes various Resource Manager operations
enq: KP - contention	Synchronizes kupp process startup

enq: KT - contention	Synchronizes accesses to the current Resource Manager plan
enq: MD - contention	Lock held during materialized view log DDL statements
enq: MH - contention	Lock used for recovery when setting Mail Host for AQ e-mail notifications
enq: ML - contention	Lock used for recovery when setting Mail Port for AQ e-mail notifications
enq: MN - contention	Synchronizes updates to the LogMiner dictionary and prevents multiple instances from preparing the same LogMiner session
enq: MR - contention	Lock used to coordinate media recovery with other uses of datafiles
enq: MS - contention	Lock held during materialized view refresh to setup MV log
enq: MW - contention	This enqueue is used to serialize the calibration of the manageability schedules with the Maintenance Window
enq: OC - contention	Synchronizes write accesses to the outline cache
enq: OL - contention	Synchronizes accesses to a particular outline name
enq: OQ - xsoqhiAlloc	Synchronizes access to olapi history allocation
enq: OQ - xsoqhiClose	Synchronizes access to olapi history closing
enq: OQ - xsoqhistrecb	Synchronizes access to olapi history globals
enq: OQ - xsoqhiFlush	Synchronizes access to olapi history flushing
enq: OQ - xsoq*histrecb	Synchronizes access to olapi history parameter CB
enq: PD - contention	Prevents others from updating the same property
enq: PE - contention	Synchronizes system parameter updates
enq: PF - contention	Synchronizes accesses to the password file
enq: PG - contention	Synchronizes global system parameter updates
enq: PH - contention	Lock used for recovery when setting Proxy for AQ HTTP notifications
enq: PI - contention	Communicates remote Parallel Execution Server Process creation status
enq: PL - contention	Coordinates plug-in operation of transportable tablespaces
enq: PR - contention	Synchronizes process startup
enq: PS - contention	Parallel Execution Server Process reservation and synchronization
enq: PT - contention	Synchronizes access to ASM PST metadata
enq: PV - syncstart	Synchronizes slave start shutdown
enq: PV - syncshut	Synchronizes instance shutdown_slvstart
enq: PW - perwarm status in dbw0	DBWR 0 holds enqueue indicating prewarmed buffers present in cache
enq: PW - flush prewarm buffers	Direct Load needs to flush pre-warmed buffers if DBWR 0 holds enqueue
enq: RB - contention	Serializes OSM rollback recovery operations
enq: RF - synch: per-SGA Broker metadata	Ensures r/w atomicity of DG configuration metadata per unique SGA
enq: RF - synchronization: critical ai	Synchronizes critical apply instance among primary instances
enq: RF - new AI	Synchronizes selection of the new apply instance
enq: RF - synchronization: chief	Anoints 1 instance's DMON as chief to other instances' DMONs
enq: RF - synchronization: HC master	Anoints 1 instance's DMON as health check master
enq: RF - synchronization: aifo master	Synchronizes apply instance failure detection and fail over operation
enq: RF - atomicity	Ensures atomicity of log transport setup
enq: RN - contention	Coordinates nab computations of online logs during recovery

enq: RO - contention	Coordinates flushing of multiple objects
enq: RO - fast object reuse	Coordinates fast object reuse
enq: RP - contention	Enqueue held when resilvering is needed or when data block is repaired from mirror
enq: RS - file delete	Lock held to prevent file from accessing during space reclamation
enq: RS - persist alert level	Lock held to make alert level persistent
enq: RS - write alert level	Lock held to write alert level
enq: RS - read alert level	Lock held to read alert level
enq: RS - prevent aging list update	Lock held to prevent aging list update
enq: RS - record reuse	Lock held to prevent file from accessing while reusing circular record
enq: RS - prevent file delete	Lock held to prevent deleting file to reclaim space
enq: RT - contention	Thread locks held by LGWR, DBW0, and RVWR to indicate mounted or open status
enq: SB - contention	Synchronizes Logical Standby metadata operations
enq: SF - contention	Lock used for recovery when setting Sender for AQ e-mail notifications
enq: SH - contention	Should seldom see this contention as this Enqueue is always acquired in no-wait mode
enq: SI - contention	Prevents multiple streams table instantiations
enq: SK - contention	Serialize shrink of a segment
enq: SQ - contention	Lock to ensure that only one process can replenish the sequence cache
enq: SR - contention	Coordinates replication / streams operations
enq: SS - contention	Ensures that sort segments created during parallel DML operations aren't prematurely cleaned up
enq: ST - contention	Synchronizes space management activities in dictionary-managed tablespaces
enq: SU - contention	Serializes access to SaveUndo Segment
enq: SW - contention	Coordinates the 'alter system suspend' operation
enq: TA - contention	Serializes operations on undo segments and undo tablespaces
enq: TB - SQL Tuning Base Cache Update	Synchronizes writes to the SQL Tuning Base Existence Cache
enq: TB - SQL Tuning Base Cache Load	Synchronizes writes to the SQL Tuning Base Existence Cache
enq: TC - contention	Lock held to guarantee uniqueness of a tablespace checkpoint
enq: TC - contention2	Lock of setup of a unique tablespace checkpoint in null mode
enq: TD - KTF dump entries	KTF dumping time/scn mappings in SMON_SCN_TIME table
enq: TE - KTF broadcast	KTF broadcasting
enq: TF - contention	Serializes dropping of a temporary file
enq: TL - contention	Serializes threshold log table read and update
enq: TM - contention	Synchronizes accesses to an object
enq: TO - contention	Synchronizes DDL and DML operations on a temp object
enq: TQ - TM contention	TM access to the queue table
enq: TQ - DDL contention	TM access to the queue table
enq: TQ - INI contention	TM access to the queue table
enq: TS - contention	Serializes accesses to temp segments
enq: TT - contention	Serializes DDL operations on tablespaces
enq: TW - contention	Lock held by one instance to wait for transactions on all instances to finish

enq: TX - contention	Lock held by a transaction to allow other transactions to wait for it
enq: TX - row lock contention	Lock held on a particular row by a transaction to prevent other transactions from modifying it
enq: TX - allocate ITL entry	Allocating an ITL entry in order to begin a transaction
enq: TX - index contention	Lock held on an index during a split to prevent other operations on it
enq: UL - contention	Lock used by user applications
enq: US - contention	Lock held to perform DDL on the undo segment
enq: WA - contention	Lock used for recovery when setting Watermark for memory usage in AQ notifications
enq: WF - contention	This enqueue is used to serialize the flushing of snapshots
enq: WL - contention	Coordinates access to redo log files and archive logs
enq: WP - contention	This enqueue handles concurrency between purging and baselines
enq: XH - contention	Lock used for recovery when setting No Proxy Domains for AQ HTTP notifications
enq: XR - quiesce database	Lock held during database quiesce
enq: XR - database force logging	Lock held during database force logging mode
enq: XY - contention	Lock used for internal testing

1.3.1.1 Enq 数据字典

受到排队锁影响的数据库资源，我们称之为"排队资源"。Oracle 使用内部数组结构来处理排队资源，可以通过 x$ksqrs(内核服务排队资源)或 v$resource 视图来查看。

```
SELECT S.ADDR, S.TYPE, S.ID1, S.ID2 FROM V$RESOURCE S;
SELECT * FROM x$ksqrs;
```

v$resource_limit 视图可查看排队锁资源的总体使用情况。查询系统资源的使用情况：

```
SELECT S.RESOURCE_NAME,
       S.CURRENT_UTILIZATION AS "当前使用数",
       S.MAX_UTILIZATION     AS "系统最大使用数",
       S.INITIAL_ALLOCATION  AS "系统初始化参数分配数",
       S.LIMIT_VALUE
  FROM V$RESOURCE_LIMIT S
 WHERE S.RESOURCE_NAME IN ('enqueue_resources',
                           'enqueue_locks',
                           'dml_locks',
                           'processes',
                           'processes');
```

	RESOURCE_NAME	当前使用数	系统最大使用数	系统初始化参数分配数	LIMIT_VALUE
1	processes	32	50	150	150
2	enqueue_locks	23	47	3160	3160
3	enqueue_resources	19	44	1308	UNLIMITED
4	dml_locks	0	0	1088	UNLIMITED

排队锁使用单独的数组而不是排队资源数组来管理排队锁，通过查询 x$ksqeq(内核服务排队对象)或 v$enqueue_lock 视图来看到这种结构。

v$equeue_lock 视图(除 TX 和 TM 锁)

```
SELECT S.ADDR,
       S.KADDR,
       S.SID,
       S.TYPE,
       S.ID1,
       S.ID2,
       S.LMODE,
       S.REQUEST,
       S.CTIME,
       S.BLOCK
  FROM V$ENQUEUE_LOCK S;
```

从 equeue 等待事件中，解码排队类型及模式：

```
SELECT s.sid,
s.event,
s.p1,
s.p1raw,
chr(bitand(s.p1, -16777216) / 16777215) ||
chr(bitand(s.p1, 16711680) / 65535) AS "TYPE",
MOD(s.p1, 16) AS "MODE"
FROM v$session_wait s
WHERE s.event = 'enqueue';
```

V$ENQUEUE_STATISTICS 用于显示队列锁的统计数据：

V$ENQUEUE_STATISTICS displays statistics on the number of enqueue (lock) requests for each type of lock. V$ENQUEUE_STATISTICS encompasses V$ENQUEUE_STATand gives more detailed information (several rows for same enqueues with different reasons).

Column	Datatype	Description
EQ_NAME	VARCHAR2(64)	Name of the enqueue request
EQ_TYPE	VARCHAR2(2)	Type of enqueue requested
REQ_REASON	VARCHAR2(64)	Reason for the enqueue request
TOTAL_REQ#	NUMBER	Total number of enqueue requests or enqueue conversions for this type of enqueue
TOTAL_WAIT#	NUMBER	Total number of times an enqueue request or conversion resulted in a wait
SUCC_REQ#	NUMBER	Number of times an enqueue request or conversion was granted
FAILED_REQ#	NUMBER	Number of times an enqueue request or conversion failed
CUM_WAIT_TIME	NUMBER	Total amount of time (in milliseconds) spent waiting for the enqueue or enqueue conversion
REQ_DESCRIPTION	VARCHAR2(4000)	Description of the enqueue request

Column	Datatype	Description
EVENT#	NUMBER	Event number

其视图结构定义如下：

```
SELECT st.inst_id, eqt.NAME, st.ksqsttyp, st.ksqstrsn, st.ksqstreq,
       st.ksqstwat, st.ksqstsgt, st.ksqstfgt, st.ksqstwtm, st.ksqstexpl,
       st.ksqstevidx
  FROM x$ksqst st, x$ksqeqtyp eqt
 WHERE (st.inst_id = eqt.inst_id)
   AND (st.ksqsttyp = eqt.resname)
   AND (st.indx > 0);
```

这里包含了非常重要的一个信息，就是锁定及其描述：

```
SELECT d.EQ_NAME, d.EQ_TYPE,d.REQ_REASON, d.REQ_DESCRIPTION FROM
V$ENQUEUE_STATISTICS d;
```

	EQ_NAME	EQ_TYPE	REQ_REASON	REQ_DESCRIPTION
1	WLM Plan Operations	WM	WLM Plan activation	Synchronizes new WLM Plan activation
2	Cross-Instance Call Invocation	CI	contention	Coordinates cross-instance function invocations
3	Process Startup	PR	contention	Synchronizes process startup
4	Parameter	PE	contention	Synchronizes system parameter updates
5	Global Parameter	PG	contention	Synchronizes global system parameter updates
6	File Object	FP	global fob contention	Synchronizes various File Object(FOB) operations
7	Block Repair/Resilvering	RE	block repair contention	Synchronize block repair/resilvering operations
8	clonedb bitmap file access	BM	clonedb bitmap file write	synchronizes clonedb bitmap file operations
9	Scheduler	KM	contention	Synchronizes various Resource Manager operations
10	Scheduler Plan	KT	contention	Synchronizes accesses to the current Resource Manager plai
11	Calibration	CA	contention	Synchronizes various IO calibration runs
12	Scheduler Master DBRM	KD	determine DBRM master	Determine DBRM master
13	KSV slave startup	PV	syncshut	Synchronizes instance shutdown_slvstart
14	KSV slave startup	PV	syncstart	Synchronizes slave start_shutdown
15	Spare Enqueue	SP	contention 4	(4) due to one-off patch
16	Spare Enqueue	SP	contention 3	(3) due to one-off patch
17	Spare Enqueue	SP	contention 2	(2) due to one-off patch
18	Spare Enqueue	SP	contention 1	(1) due to one-off patch
19	File Mapping	FM	contention	Synchronizes access to global file mapping state
20	Internal Test	XY	contention	Lock used for internal testing
21	Service Operations	AS	service activation	Synchronizes new service activation
22	Property Lock	PD	contention	Prevents others from updating the same property
23	Rolling Migration	RU	waiting	Results of rolling migration GIC

```
SELECT * FROM V$ENQUEUE_STATISTICS;
SELECT * FROM V$ENQUEUE_LOCK;
SELECT * FROM V$ENQUEUE_STAT;
```

在RAC环境中，Oracle建议使用NOORDER和CACHE（缓存）选项创建序列，以减少索引争用。从V$SYSTEM_EVENT视图获得的行锁（rowlock）缓存统计信息将显示，单调递增的序列是否导致数据库中的索引争用。可以检查GV$ENQUEUE_STAT视图的EQ_TYPE列，以确定序列是否造成任何索引排队等待。如果EQ_TYPE列的值是SQ Enqueue，这通常是序列争用的迹象。如果索引键的值是从序列中派生的，那么可以增加序列缓存的大小，以减少索引争用。你可以轻松地用alter sequence语句提高序列缓存的大小。

1.3.2 enq: AE - lock

```
SELECT * FROM V$EVENT_NAME WHERE NAME  LIKE 'enq: AE%';
```

EVENT#	EVENT_ID	NAME	PARAMETER1	PARAMETER2	PARAMETER3	WAIT_CLASS_ID	WAIT_CLASS#	WAIT_CLASS
817	3963940642	enq: AE - lock	name|mode	edition obj#	0	1893977003	0	Other

从 Oracle Database 11g 开始，除了每个文件要获得 MR 锁之外，每个登录数据库的会话现在都会缺省获得一个 AE 锁：

```
SQL> set line 9999
SQL> select * from v$lock where type='AE' and rownum <5;

ADDR             KADDR                 SID TY        ID1        ID2      LMODE    REQUEST      CTIME      BLOCK
---------------- ---------------- ---------- -- ---------- ---------- ---------- ---------- ---------- ----------
00000000774D8978 00000000774D89D0        132 AE         100          0          4          0    1308736          0
00000000774D9C08 00000000774D9C60        141 AE         100          0          4          0        179          0
00000000774DA308 00000000774DA360        152 AE         100          0          4          0         11          0
00000000774DA3E8 00000000774DA440        153 AE         100          0          4          0        150          0
```

1.3.3 enq: MR 锁

```
SELECT * FROM V$EVENT_NAME WHERE NAME  LIKE 'enq: MR%';
```

	EVENT#	EVENT_ID	NAME	PARAMETER1	PARAMETER2	PARAMETER3	WAIT_CLASS_ID	WAIT_CLASS#	WAIT_CLASS
1	662	3113891348	enq: MR - contention	name|mode	0 or file #	type	1893977003	0	Other
2	663	3480024039	enq: MR - standby role transition	name|mode	0 or file #	type	1893977003	0	Other

可能很多朋友都注意过，在 V$LOCK 视图中，最常见的其实是 MR 锁，也就是介质恢复锁（Media Recovery）：

```
SQL> col name format a100
SQL> select file#,name from v$datafile;
    FILE# NAME
---------- ----------------------------------------------------------------------------------------
        1 /u02/app/oracle/oradata/oratest/system01.dbf
        2 /u02/app/oracle/oradata/oratest/sysaux01.dbf
        3 /u02/app/oracle/oradata/oratest/undotbs01.dbf
        4 /u02/app/oracle/oradata/oratest/users01.dbf
        5 /u02/app/oracle/oradata/oratest/example01.dbf
        6 /u02/app/oracle/oradata/oratest/users02.dbf
        7 /u02/app/oracle/oradata/oratest/ts_ogg01.dbf
        8 /u02/app/oracle/oradata/oratest/users03.dbf
8 rows selected.

SQL> select FILE#,NAME from v$tempfile;
    FILE# NAME
---------- ----------------------------------------------------------------------------------------
        1 /u02/app/oracle/oradata/oratest/temp01.dbf

SQL> select * from v$lock where type='MR' order by id1;

ADDR             KADDR                 SID TY        ID1        ID2      LMODE    REQUEST      CTIME      BLOCK
---------------- ---------------- ---------- -- ---------- ---------- ---------- ---------- ---------- ----------
00000000774D8C18 00000000774D8C70          5 MR         1          0          4          0    1309241          0
00000000774D86D8 00000000774D8730          5 MR         2          0          4          0    1309241          0
00000000774D8278 00000000774D82D0          5 MR         3          0          4          0    1309241          0
00000000774D8FB0 00000000774D9008          5 MR         4          0          4          0    1309241          0
00000000774D8B38 00000000774D8B90          5 MR         5          0          4          0    1309241          0
00000000774D87B8 00000000774D8810          5 MR         6          0          4          0    1309241          0
00000000774D8CF8 00000000774D8D50          5 MR         7          0          4          0    1309241          0
00000000774D8DF0 00000000774D8E48          5 MR         8          0          4          0    1309241          0
00000000774D8ED0 00000000774D8F28          5 MR       201          0          4          0    1309241          0
9 rows selected.
```

MR 锁用于保护数据库文件，使得文件在数据库打开、表空间 Online 时不能执行恢复。当进程对数据文件执行恢复时，需要排他的获得 MR 锁。当数据库打开时，每个文件上都分配一个 MR 锁。注意在以上输出中 ID1 代表文件号，其中也包含了 201 号临时文件。

1.3.4 enq: DX - contention

```
SELECT * FROM V$EVENT_NAME WHERE NAME = 'enq: DX - contention';
```

	EVENT#	EVENT_ID	NAME	PARAMETER1	PARAMETER2	PARAMETER3	WAIT_CLASS_ID	WAIT_CLASS#	WAIT_CLASS
1	830	2881503609	enq: DX - contention	name|mode	transaction entry #	0	1893977003	0	Other

DX：Distributed transaction entry

enq: DX - contention 是一个分布式事务锁。

enq: DX - contention 和 inactive transaction branch 这两个事件是相伴的。这两个等待事件是和 DBLINK 相关的，metalink 上有相关的文章：High CPU by Sessions Holding DX Enqueue; Others Waiting 'enq: DX - contention' [ID 1275884.1]

大意就是执行 dblink 语句时候，由于人为取消终止或网络等问题导致语句触发上面的等待事件

High CPU by Sessions Holding DX Enqueue Others Waiting 'enq DX - contention' (文档 ID 1275884.1).mhtml

在 Oracle 中 enq: DX 队列锁一般用意保护分布式事务(used to protect distributed transactions)，对应的就存在 enq: DX – contention 等待事件。

Id1 / Id2 含义

id2 总是 0。id1 代表其希望锁定的记录，所以总是 distributed transaction elements 队列中的一个条记录数（一个整数），由实例参数”distributed_transactions”决定。

1.3.4.1 案例

http://www.ssc.stn.sh.cn/html/zsk/ITyw/2012-04/5793.html
http://blog.itpub.net/4227/viewspace-709121/
http://www.codeweblog.com/oracle-enq-dx-contention-of-the-resolution-process/

1.3.5 enq: SQ - contention 序列等待

enq: SQ - contention/row cache lock/DFS lock handle 这三个等待事件都与 Oracle 的 Sequence 有关。

```
SELECT *
  FROM V$EVENT_NAME
 WHERE NAME IN
       ('row cache lock', 'enq: SQ - contention', 'DFS lock handle');
```

EVENT#	EVENT_ID	NAME	PARAMETER1	PARAMETER2	PARAMETER3	WAIT_CLASS_ID	WAIT_CLASS#	WAIT_CLASS
200	2322460838	enq: SQ - contention	name|mode	object #	0	3290255840	2	Configuration
210	1714089451	row cache lock	cache id	mode	request	3875070507	4	Concurrency
360	3595075359	DFS lock handle	type|mode	id1	id2	1893977003	0	Other

使用如下的 SQL 我们可以查询到锁的名称和请求的 MODE，表的 mode 值参考表格：

```
select chr(bitand(p1,-16777216)/16777215)||
chr(bitand(p1, 16711680)/65535) "Lock",
bitand(p1, 65535) "Mode"
from v$session_wait
where event = 'DFS enqueue lock acquisition';
```

Table C-1 Lock Mode Values

Mode Value	Description
1	Null mode
2	Sub-Share
3	Sub-Exclusive
4	Share
5	Share/Sub-Exclusive
6	Exclusive

```
SELECT * FROM V$LOCK_TYPE D WHERE D.TYPE IN ('SV','SQ');
```

TYPE	NAME	ID1_TAG	ID2_TAG	IS_USER	DESCRIPTION
SQ	Sequence Cache	object #	0	NO	Lock to ensure that only one process can replenish the sequence cache
SV	Sequence Ordering	object #	0	NO	Lock to ensure ordered sequence allocation in RAC mode

Oracle 为了管理 Sequence 使用了以下三种锁。

① **row cache lock**：在调用 SEQUNECE.NEXTVAL 过程中，将数据字典信息进行物理修改时获取。赋予了 **NOCACHE** 属性的 SEQUENCE 上发生，等待事件为 **row cache lock**。

② **SQ 锁**：在内存上缓存（CACHE）的范围内，调用 SEQUENCE.NEXTVAL 期间拥有此锁。赋予了 CACHE 属性的 SEQUENCE 上发生。赋予了 CACHE 属性的 SEQUENCE 调用 NEXTVAL 期间，应该以 SSX 模式获得 SQ 锁。许多会话同时为了获取 SQ 锁而发生争用过程中，若发生争用，则等待 enq: SQ - contention 事件。enq: SQ - contention 事件的 P2 值是 Sequence 的 OBJECT ID。因此，若利用 P2 值与 DBA_OBJECTS 的结合，就可以

知道对哪个 SEQUENCE 发生了等待现象。

③ **SV 锁**:RAC 上节点之间顺序得到保障的情况下,调用 SEQUENCE.NEXTVAL 期间拥有。赋予 **CACHE + ORDER** 属性的 SEQUENCE 上发生,等待事件为 **DFS lock handle,解决办法为:尽量设置为 NOORDER 并增大其 CACHE 值。**

根据创建 Sequence 时赋予的属性,整理等待事件的结果如下:

- **NOCACHE: row cache lock**
- **CACHE + NOORDER: enq: SQ - contention**
- **CACHE + ORDER(RAC): DFS lock handle**

创建 SEQUENCE 赋予的 CACHE 值较小时,有 enq: SQ - contention 等待增加的趋势。CACHE 值较小时,内存上事先 CACHE 的值很快被耗尽,这时需要将数据字典信息物理修改后,再次执行 CACHE 的工作。在此期间,因为一直拥有 SQ 锁,相应的 enq: SQ - contention 事件的等待时间也会延长。很不幸的是,在创建 SEQUENCE 时,将 CACHE 值的缺省值设定为较小的 20。因此创建使用量多的 SEQUENCE 时,CACHE 值应该取 1000 以上的较大值。

另外,偶尔一次性同时创建许多会话时,有时会发生 enq: SQ - contention 等待事件。其理由是 V$SESSION.AUDSID(auditing session id)列值是利用 Sequence 创建的。Oracle 在创建新的会话后,利用名为 SYS.AUDSES$的 Sequence 的 nextval,创建 AUDSID 值。SYS.AUDSES$ Sequence 的 CACHE 大小的缺省值设定为 20。许多会话同时连接时,可以将 SYS.AUDSES$ Sequence 的 CACHE 大小扩大至 1000,以此可以解决 enq: SQ - contention 等待问题。 10g 下默认 20,11g 下默认为 10000,通过如下的 SQL 可以查询:

```
SELECT * FROM dba_sequences d WHERE d.sequence_name ='AUDSES$';
```

	SEQUENCE_OWNER	SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY	CYCLE_FLAG	ORDER_FLAG	CACHE_SIZE	LAST_NUMBER
1	SYS	AUDSES$	1	2000000000	1	Y	N	10000	892977

RAC 上创建 SEQUENCE 时,在赋予了 CACHE 属性的状态下,若没有赋予 ORDER 属性,则各节点将会把不同范围的 SEQUENCE 值 CACHE 到内存上。比如,拥有两个节点的 RAC 环境下,创建 CACHE 值为 100 的 SEQUENCE 时,1 号节点使用 1~100,2 号节点使用 101~200。若两个节点之间都通过递增方式使用 SEQUENCE,必须赋予如下 ORDER 属性。

```
SQL> CREATE SEQUENCE ORDERED_SEQUENCE CACHE 100 ORDER;
```

如果是已赋予了 CACHE+ORDER 属性的 SEQUENCE,Oracle 使用 SV 锁进行行同步。即,对赋予了 ORDER 属

性的 Sequence 调用 nextval 时，应该以 SSX 模式拥有 SV 锁。在获取 SV 锁过程中，如果发生争用时，不是等待 row cache lock 事件或 enq: SQ - contention 事件，而是等待名为 DFS lock handle 事件。正因如此，V$EVENT_NAME 视图上不存在类似"enq:SV-contention"的事件。DFS lock handle 事件是在 OPS 或 RAC 环境下，除了高速缓冲区同步之外，还有行高速缓冲区或库高速缓冲区的为了同步获取锁的过程中等待的事件。若要保障多个节点之间 Sequence 顺序，应该在全局范围内获得锁，在此过程中会发生 DFS lock handle 等待。在获取 SV 锁的过程中发生的 DFS lock handle 等待事件的 P1、P2 值与 enq: SQ - contention 等待事件相同（P1=mode+namespace、P2=object#）。因此从 P1 值能确认是否是 SV 锁，通过 P2 值可以确认对哪些 Sequence 发生过等待。SV 锁争用问题发生时的解决方法与 SQ 锁的情况相同，就是将 CACHE 值进行适当调整，这也是唯一的方法。

在 RAC 等多节点环境下，Sequence 的 CACHE 值给性能带来的影响比单节点环境更严重。因此，尽量赋予 CACHE+NOORDER 属性，并要给予足够大的 CACHE 值。如果需要保障顺序，必须赋予 CACHE+ORDER 属性。但这时为了保障顺序，实例之间不断发生数据的交换。因此，与赋予了 NOORODER 属性的时候相比性能稍差。

有一点必须要注意，没有赋予 CACHE 属性时，不管 ORDER 属性使用与否或 RAC 环境与否，一直等待 row cache lock 事件。row cache lock 是可以在全局范围内使用的锁，单实例环境或多实例环境同样可以发生。

没有赋予 CACHE 属性时，不管 ORDER 属性是否或 RAC 环境是否，一直等待 ROW CACHE 事件，ROW CACHE LOCK 是否可以在全局范围内使用的锁，单实例环境或多实例环境同时可以发生。

Oracle Sequence 默认是 NOORDER，如果设置为 ORDER；在单实例环境没有影响，在 RAC 环境此时，多实例实际缓存相同的序列，此时在多个实例并发取该序列的时候，会有短暂的资源竞争来在多实例之间进行同步。因次性能相比 noorder 要差，所以 RAC 环境非必须的情况下不要使用 ORDER，尤其要避免 NOCACHE ORDER 组合。

但是如果使用了 Cache，如果此时 DB 崩溃了，那么 sequence 会从 cache 之后重新开始，在 cache 中没有使用的 sequence 会被跳过。即 sequence 不连续。所以只有在多节点高峰并发量很大的情况且对连续性要求不高的情况下，才使用：noorder + cache。

在下面的链接中讲到了 RAC 之间序列同步：

Sequences in Oracle 10g RAC

http://www.pythian.com/news/383/sequences-in-oracle-10g-rac/

How does RAC synchronize sequences?

In Oracle 10g RAC, if you specify the "ordered" clause for a sequence, then a global lock is allocated by the node when you access the sequence.

This lock acquisition happens only at the first sequence access for the node (A), and subsequent uses of the sequence do not wait on this lock. If another node (B) selects from that sequence, it requests the same global lock and once acquired it returns the sequence's next value.

The wait event associated with this activity is recorded as "events in waitclass Other" when looked in gv$system_event. So much for event groups, it couldn't be more obscure. That view shows overall statistics for the session.

However if you look in the gv$session_wait_history it shows as "DFS lock handle" with the "p1" parameter been the object_id of the sequence. This second view has a sample of the last 10 wait events for a session.

In a SQL_TRACE with waitevents (10046 trace) it will be a "DFS lock handle" but in AWR or statspack reports it will be "events in wait class Other". So much for consistency.

1.3.5.1 DFS lock handle

The session waits for the lock handle of a global lock request. The lock handle identifies a global lock. With this lock handle, other operations can be performed on this global lock (to identify the global lock in future operations such as conversions or release). The global lock is maintained by the DLM.

Wait Time: The session waits in a loop until it has obtained the lock handle from the DLM. Inside the loop there is a wait of 0.5 seconds.

Parameter	Description
name	See "name and type"
mode	See "mode"
id1	See "id1"
id2	See "id2"

The session needs to get the lock handle.

该等待事件的发生，若不是 SV 锁的话，多半为 bug 引起。

DFS lock handle"这一 event 是在 RAC 环境中，会话等待获取一个全局锁的句柄时产生的。在 RAC 中，全局锁的句柄是由 DLM(Distributed Lock Manager 分布式锁管理器)所管理和分配的。大量发生这一 event 说

明全局锁句柄资源不够分配了。决定 DLM 锁数量的参数是_lm_locks ,9i 以后 ,它是一个隐含参数 默认值是 12000。没有特殊情况，这一值对于一个 OLTP 系统来说是足够的。我们不能盲目地直接增加资源，而是需要找到导致资源紧张的根本原因。锁资源紧张，说明存在大量事务获取了锁，但是事务没有提交、回滚。那么，又是什么导致了这些事务不结束呢?应用程序代码不完善，没有提交事务?或者那些事务还在等待别的资源?

1.3.5.2 我碰到的案例

【故障处理】序列 cache 值过小导致 CPU 利用率过高 ：http://blog.itpub.net/26736162/viewspace-2123996/

2016-08-24	【故障处理】序列 cache 值过小导致 CPU 利用率过高	http://mp.weixin.qq.com/s?__biz=MzIzOTA2NjEzNQ==&mid=2454771414&idx=1&sn=0cbc454bc7d5a513bbca6083958e2f34&scene=21#wechat_redirect

手机长按下图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,免费学习最实用的数据库技术。