**故障诊断**

云和恩墨(北京)信息技术有限公司

技术顾问 燕鑫

http://www.enmotech.com

**文档控制：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序** | **版本号** | **更改人** | **日期** | **备注** |
| 1 | 1.0版 | 燕鑫 | 2018-09-05 | 初始版本 |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[1. buffer busy wait的相关事件 - 4 -](#_Toc522952378)

# Crash,Hang and Loop situations

一个实例或者进程crash掉：

1. internal errors（ora-600）
2. operating system violation：  
   比如不正确的访问数据结构（segment violation）  
   不正确的访问内存结构（access violation）

一个oracle进程hang住，等待一个可能并不会发生的event。与loop的区别是，进程是不活动的，可能会观测到cpu并不繁忙，甚至为0。

一个进程不断的loop，无限的重复着同一个操作。与hang的区别是，进程一定是活动的，甚至可能导致cpu利用率百分之百。

## Diagnostic Files

诊断文件主要有如下几种：

1. 告警日志
2. Trace文件
3. Application log files，比如sqlplus工具的日志文件
4. Core dump files，个别操作系统有
5. System log files

### Alert日志

Alter日志中会把报错的详细内容写到trace文件中。

### Trace file

Trace file的文件头有如下内容：

1. 操作系统和操作系统版本
2. Oracle版本和一些已安装的options
3. 实例名
4. 进程号

Trace file /u01/app/oracle/diag/rdbms/prod4/PROD4/trace/PROD4\_m000\_9113.trc

Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production

Build label: RDBMS\_12.2.0.1.0\_LINUX.X64\_170125

ORACLE\_HOME: /u01/app/oracle/product/12c/db\_1

System name: Linux

Node name: host01

Release: 4.1.12-94.3.9.el7uek.x86\_64

Version: #2 SMP Fri Jul 14 20:09:40 PDT 2017

Machine: x86\_64

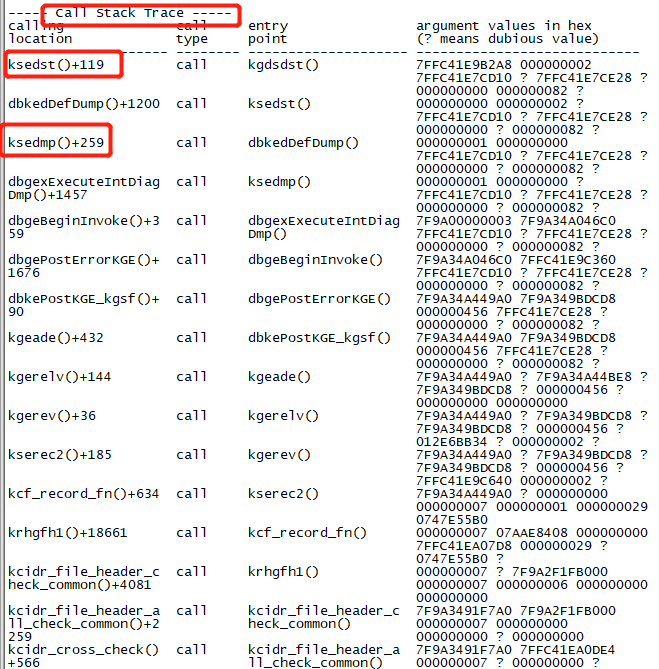
Instance name: PROD4

Redo thread mounted by this instance: 1

Oracle process number: 12

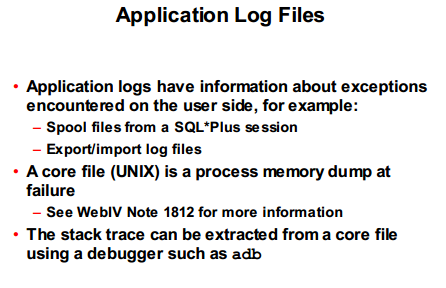
Unix process pid: 9113, image: oracle@host01 (M000)

其中最重要的部分就是stack trace，这里按顺序记录了挂掉的进程生前的调用的函数，结合源码可以诊断出问题的根源。



开头的2到3个function可以忽略，就是开头是kse的（kernel service error），因为这些是用来抛出异常的函数。

### Application Log Files



注意的就是，像数据泵的log、rman的log这些都属于应用的logfile，包括在sqlplus中spool files这些，都属于该范畴。

### System Log Files

Linux下的系统日志在/var/log下：

/var/log/messages：系统服务及日志，包括服务的信息，报错等等

/var/log/secure：系统认证信息日志

/var/log/maillog：系统邮件服务信息

/var/log/cron：系统定时任务信息

/var/log/boot.log：系统启动信息

## Hang Situation

一个oracle进程hang住，就是一个进程在等待一个并不会发生的event。与loop的区别是，进程是不活动的，可能会观测到cpu并不繁忙，甚至为0。

诊断这类情况的工具有：

1. 各种state dumps：process state dump、system state dump、error stacks
2. 一些动态性能视图：v$session\_wait，v$lock，v$latch，v$latch holder
3. HANGANALYZE event

Hanganalyze可以自动的建立wait chain，比手动去查询要方便的多：

Alter session set events 'immediate trace name hanganalyze level x';

Level是用来确定dump哪些process的errostack：

10：dump所有的process，这会是一个非常大量的输出（voluminous data output）。

5：dump所有包含再wait chain中的process，这个输出量也比较大

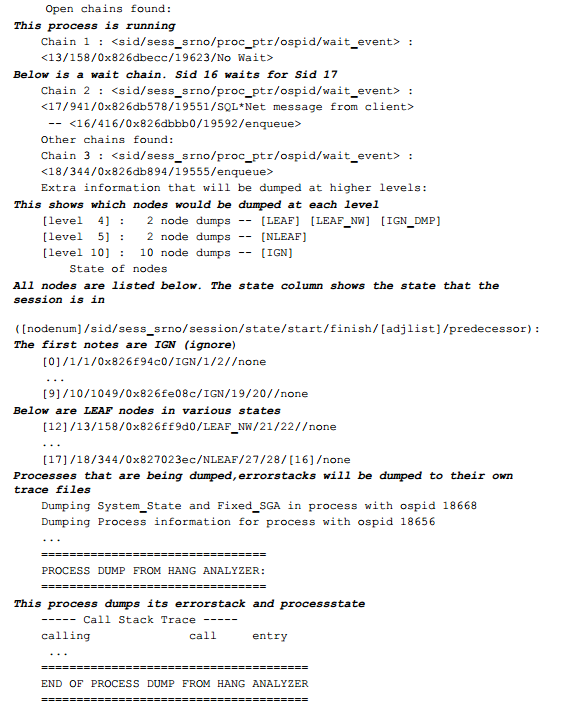
4：dump wait chain中的leaf nodes

3：只dump那些被认为hang住了的process

2：minimal output

1：very minimal output

Dump出来的trace文件大致如下：



其中对nodes state的描述中，有个别缩写含义如下：

IGN：ignore

LEAF：a waiting leaf node（wait chain中的最后一个元素）

LEAF\_NW：a running（use CPU cycle）leaf node

N\_LEAF：是wait chain中的一员，但不在最后（not leaf node）

## Looping

Loop也是一种等待，比如一个进程不停的loop就是为了等待另一个state object释放资源。

诊断loop可以通过dump和视图查询的方式：

1. process state dump，system state dump，errorstacks
2. 一些动态性能视图：v$session\_wait，v$lock，v$latch，v$latch holder

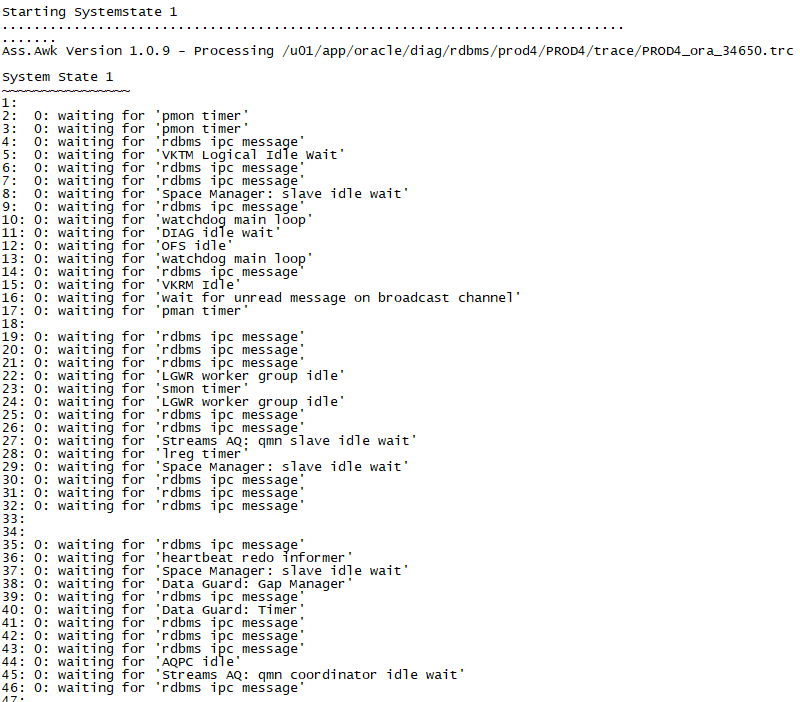
通常会使用system state dump：

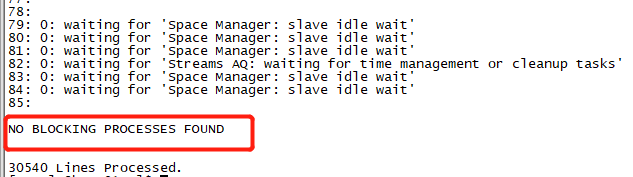
Alter session set events 'immediate trace name systemstate level 10';

### Ass.awk

是用该脚本，可以把system state dump的等待事件抽取出来。如下：

[oracle@host01 ~]$ awk -f ass109.awk /u01/app/oracle/diag/rdbms/prod4/PROD4/trace/PROD4\_ora\_34650.trc





## Slow Performance

收集awr报告。

# State Objects

State objects是SGA中与各种各样的database entries相关的结构体（structure）。

这些database entries包括：

1. 前台后台进程
2. 会话（sessions）
3. Latches和enqueues
4. Buffer handles
5. Transactions

每一个process、session、latch、lock、enqueue等等都有自己单独的state object。

State objects占有的资源由PMON进行清理。

State object可以在process state dump和system state dump中查看。

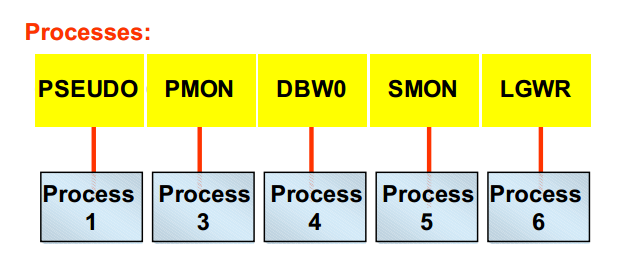
## Proccess State Dump

### Process state

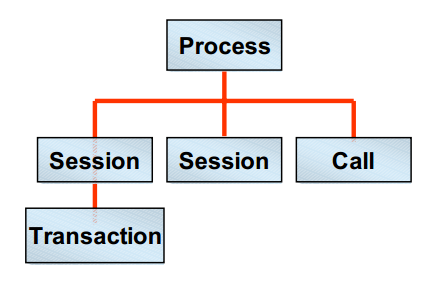
每一个后台进程和前台进程都有自己的state objects。

在oracle shared server环境下的pseudo process会有一个额外的state object。Pseudo process就是用来在共享连接的情况下，调节session migration的。

一般来说，对进程的编号如下，这个也会应系统而异：



Process state的等级（hierarchy）如下：



一般来说是一个process一个会话，共享连接或者分布式事务（XA）例外。

### Process state dump

Dump process state有三种方法：

1. alter session set events 'immediate trace name processstate level 10'
2. oradebug dump processstate 10
3. 指定dump一个报错，需要在init.ora中添加<error#> "trace name processstate level 10"

Dump process state会把该process下的所有的state objects全部dump出来。

当明确了问题所在的process的时候，可以只dump该process state。当然，如果不能确定问题所在的process，则需要dump system state，来找到问题process（offending process）。

SQL> Alter session set events 'immediate trace name processstate level 10';

Session altered.

当然，也可以使用oradebug指定dump出某个process：

SQL> select spid from v$process p,v$session s where p.addr=s.paddr and s.sid=(select sid from v$mystat where rownum=1);

SPID

------------------------

47183

发起一个事务：

SQL> insert into yx.t1 values(1);

1 row created.

在另一个会话中dump该process 47183的process state：

SQL> oradebug setospid 47183

Oracle pid: 79, Unix process pid: 47183, image: oracle@host01 (TNS V1-V3)

SQL> oradebug dump processstate 10

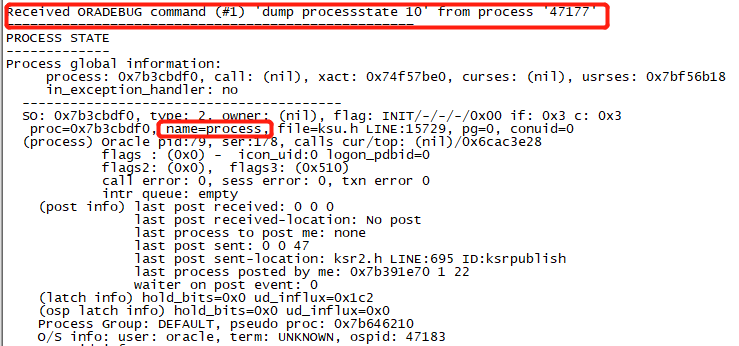
Statement processed.

然后找到该trace文件，找\*\_ora\_47183.trc：

[oracle@host01 ~]$ ls $ORACLE\_BASE/diag/rdbms/prod4/PROD4/trace/\*\_ora\_47183.trc

/u01/app/oracle/diag/rdbms/prod4/PROD4/trace/PROD4\_ora\_47183.trc

我们看其中dump出来的trace文件：



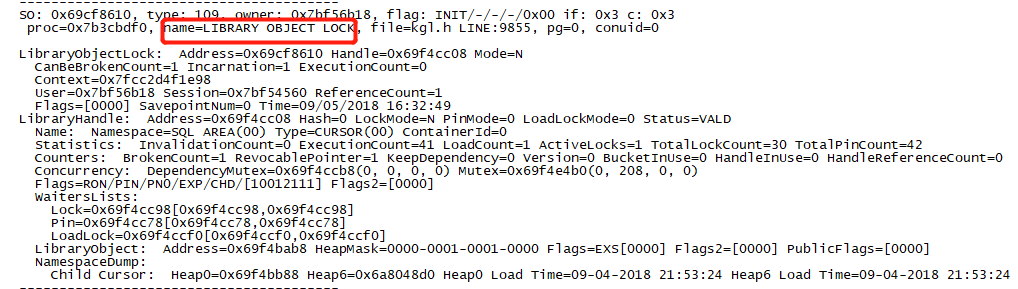
从第一行，我们看出这个dump是从47177的进程用oradebug dump出来的。

Name=process表示这个SO是process的SO。

我们接着看看该process的session的SO：



下面是library object lock的SO：

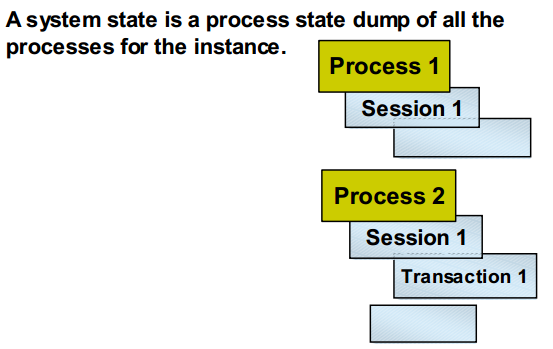


简单说一下，state object的各个属性：

1. SO：state object adress
2. Type：就是该SO的类型，是一个编号，对应name
3. Flag：  
   INIT---- object is initialized  
   FLST----- on freelist  
   CLN------ object freed by PMON
4. DID：resource id

## System State Dump

System state dump就是所有的process的process state dump的组合。如下图：



Dump system state有三种方法：

1. alter session set events 'immediate trace name systemstate level 10'
2. oradebug dump systemstate 10
3. 指定dump一个报错，需要在init.ora中添加<error#> "trace name systemstate level 10"

我们来dump一次system state看看，这次我们用oradebug来dump。

首先先说一下dump出来的文件有3个区域：

1. normal trace file header
2. system global information
3. process information

我们实验的时候，同时开了3个会话，然后现在我们来在其中一个会话中dump：

SQL> oradebug setmypid

Statement processed.

SQL> oradebug dump systemstate 10;

Statement processed.

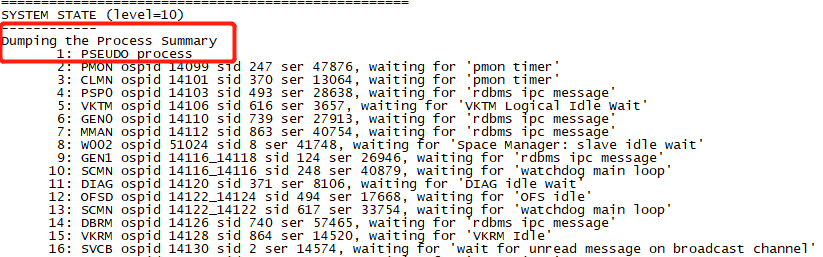
SQL> oradebug tracefile\_name

/u01/app/oracle/diag/rdbms/prod4/PROD4/trace/PROD4\_ora\_51347.trc

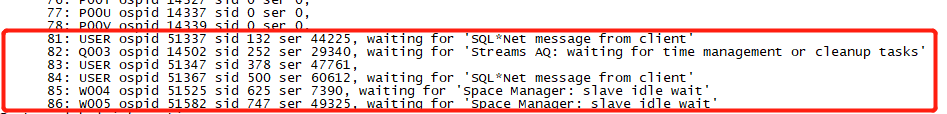
我们来先看看trace file header：



那么紧接着“SYSTEM STATE (level=10)”的字样下面是进程汇总：



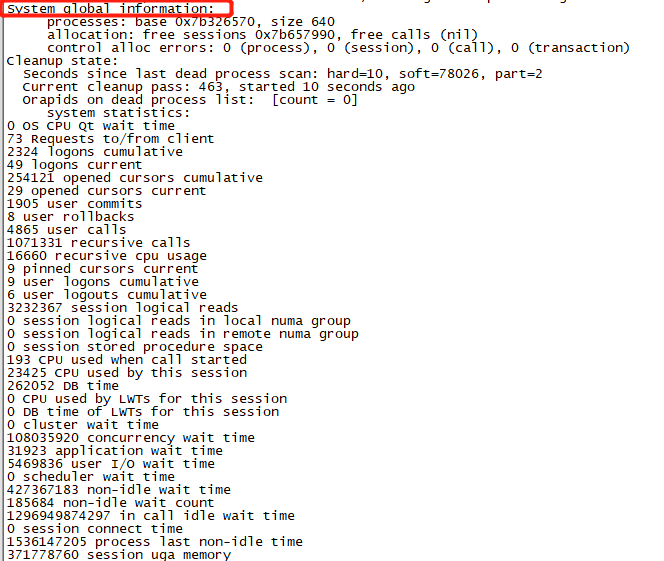
我们看到，先列出的是后台进程，然后才是用户进程，再process information部分，也是这样的顺序：



我们可以看到，确实有3个用户进程，而且进程号也是一致的：



接下来我们看system global information：

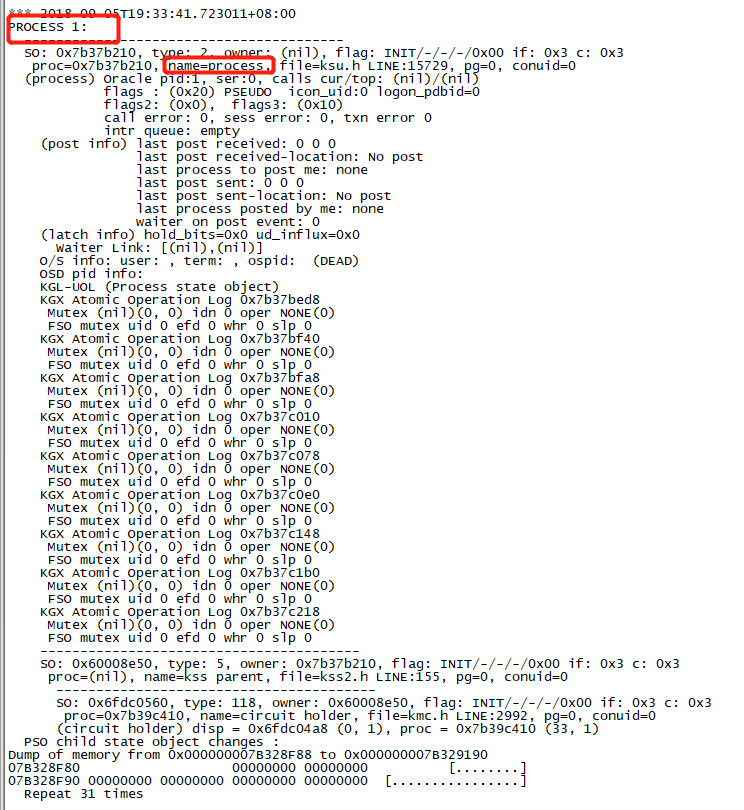


然后我们来看3部分中最重要的一部分：process information。当我们诊断一个wait chain的root cause的时候，靠的就是这部分。

下面分别展示process state object、session state object、call state object：

### Process SO

首先是process state object：



Process information区域从process 1开始，也就是从PSEUDO process开始，其实顺序就是跟dumping the process summary列出来的顺序是一样的。

(process)前面部分是proccess的state，这个上一章节说过了，(process)后面部分中的oracle pid来源于v$process中的pid，还有address of current call（cur）from the state object和address of top level call（top），nil表示null。

Flag代表如下几种进程：process dead and will be clean up，A system process，the process is PMON，SMON process，PSEUDO process，oracle shared server，dispatcher process。其实这个不用记，因为值后有说明，说明这是个种类的process。

接下来是error code部分：

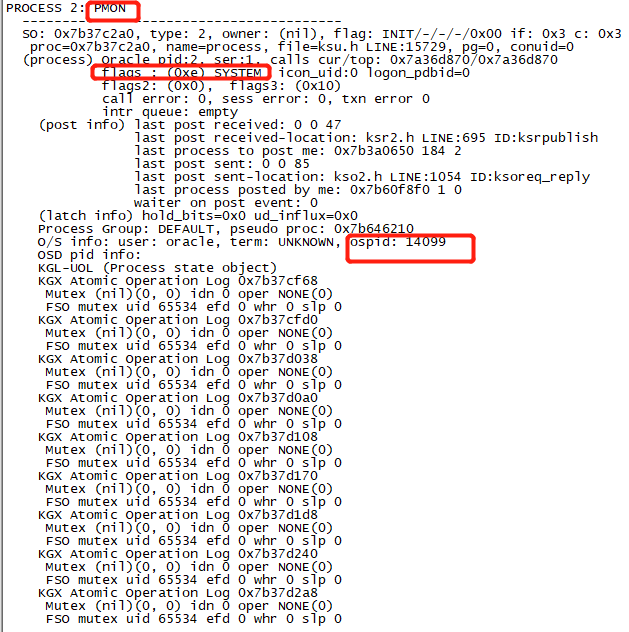
1. call error：user call返回的error code
2. sess error：session creation返回的error code
3. txn error

再接下来是（post info）：insert-process posting information

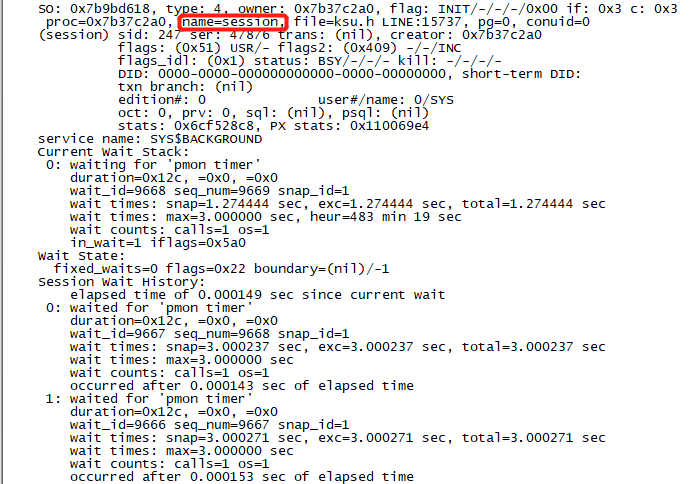
再接下来时（latch info）：包括wait event和latch的bit array。

再接下来就是OS data：user，ospid等等。

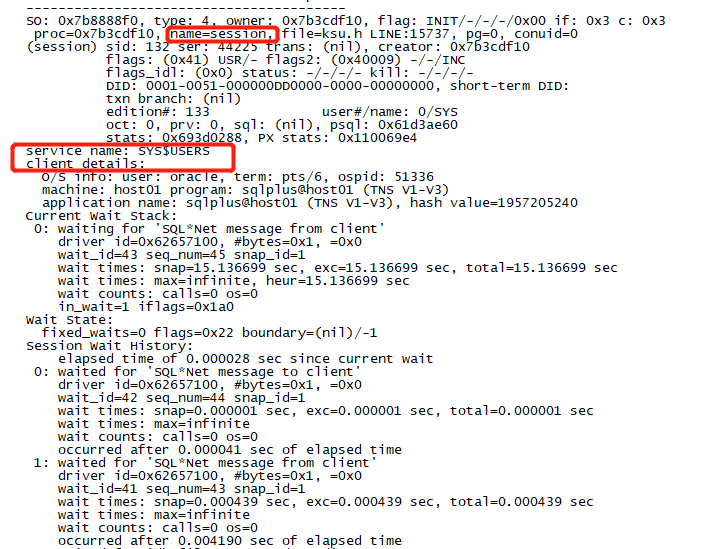
我们再来看看pmon的SO：



### Session SO



这是PMON的session。我们再看个user的session：



我们主要介绍（session）中的flag的几个标识的含义：

1. USR：user session
2. BSY：session is busy，it is in a call
3. DED：session marked deab by user process
4. DEL：session being deleted（在执行了alter system kill session后）
5. KIL：session marked for kill（在执行了alter system kill session后）

### Call SO



我们看到name=call，这就是call SO。这里我们注意depth就是递归深度（recursive level）。

### 总结

全面仔细的读system state dump是很累的，最好是用ass.awk脚本把等待事件抽取出来，确定问题process，然后dump process state来诊断问题。

# Data Dictionary Views

列出几个对于诊断hang或者loop很有用的视图：

1. v$session\_wait，v$session\_event
2. v$latch，v$latchholder，v$latchname
3. v$sysstat，v$lock
4. v$process，v$session，v$transaction
5. x$kcbfwait（buffer waits），x$ksqst（enqueues）

## v$session\_wait,v$session\_event